منشورات المركز الليبي للمنظومات الإلكترونية والبرمجيات وبحوث الطيران

المجمل في المفاهيم الأساسية لنظم تشغيل الحاسوب

د. عمر المبروك أبوزيد

الطبعة الثانية

المجمل في المفاهيم الأساسية لنظم تشغيل الحاسوب — تأليف د. عمر المبروك أبوزيد — الطبعة الثانية 2023م

ما يُميز هذا الكتاب

حرص المؤلف على إعداد هذا الكتاب باللغة العربية وبأسلوب سلس اعتمد على ترابط مكوناته لتسهيل قراءته وفهمه. كما اجتهد المؤلف في تقليل استخدام المصطلحات الإنجليزية داخل النصوص قدر المستطاع واكتفى باستخدامها فقط في الحالات الضرورية الذي يتطلبه سياق الشرح. وحتى يكون الفهم واضحًا ومكتملًا ضمّن المؤلف في نهاية هذا الكتاب معجم ترجمة باللغة العربية لأغلب المصطلحات العلمية الواردة فيه، بوبها أبجديًا لتسهيل عملية البحث عنها داخل هذا المعجم. ومن باب حرص المؤلف في هذا الخصوص اعتنى باختيار الترجمة الدقيقة لأي مصطلح أستُخدم في الكتاب استنادًا إلى الترجمات الشائعة لهذه المصطلحات في مجال علوم الحاسوب، أو في بعض الأحيان إلى الاجتهادات الفردية له، والتي يأمل من الله أن يكون قد وُفّق فيها.

من المزايا الأخرى لهذا الكتاب هو تقديم العديد من الأمثلة التوضيحية كلما استلزم الأمر ذلك. واختتام كل فصل منه بأسئلة مرجعية تُساهم في جعل الطالب يفهم مادة الكتاب ويتفاعل معها بشكل إيجابي، لما لذلك من دلالات على تبسيط المعاني وتقريبها وتذكرها بشكل أفضل.

أخيرًا، أُعدَّ الكتاب كحصيلة لعدة سنوات تدريسية لمادة نظم التشغيل استعان فيها المؤلف بعدة مقالات، وصفحات متخصصة في مجالات نظم التشغيل، وكتب مرجعية كان من أهمها مؤلفات تانن باوم، وزلبرشاتس، وجاريدو والتي ساهمت بشكل كبير في وضع أسس هذا الكتاب.

ختامًا أدعو الله العلي القدير أن يُثري هذا الكتاب المكتبة العربية ويسد فجوةً فيها، وأن يستفيد منه كل من اطّلع عليه.

د. عمر المبروك أبوزيد
غريان-ليبيا: يونيو 2023م

المُجْمَل في المفاهيم الأساسية لنُظم تشغيل الحاسوب

عمر المبروك أبوزيد

ماجستير هندسة حاسوب، جامعة كلاوست هال التقنية، ألمانيا، 2002م.

دكتوراة هندسة حاسوب، جامعة نيوكاسل، المملكة المتحدة، 2013م.

أستاذ مشارك – الأكاديمية الليبية للدراسات العليا

غريان، ليبيا

للتواصل والمقترحات: drombouzid@gmail.com

الطبعة الثانية

2023م

الوكالة الليبية للترقيم الدولي الموحد للكتاب
دار الكتب الوطنية
بنغازي – ليبيا
هاتف: 9097074 – 9096379 – 9090509
بريد مصور: 9097073
البريد الإلكتروني: nat_lib_libya@hotmail.com

رقم الإيداع بدار الكتب الوطنية ببنغازي
2020/459
الترقيم الدولي: ISNB: 978-9959-9648-7-8

﴿وَقُلِ ٱعمَلُواْ فَسَيَرَى ٱللَّهُ عَمَلَكُم وَرَسُولُهُۥ وَٱلمُؤمِنُونَ وَسَتُرَدُّونَ إِلَىٰ عَٰلِمِ ٱلغَيبِ وَٱلشَّهَٰدَةِ فَيُنَبِّئُكُم بِمَا كُنتُم تَعمَلُونَ﴾

سورة التوبة، الآية 105.

إلى روح رمز الحنان ... أمي،

إلى رمز الآمان ... أبي،

إلى من ساندني في إنجاز هذا الكتاب ... زوجتي، أبنائي، وبناتي،

إلى أختي، وإخوتي،

إلى كل من يطمح إلى خدمة بلاده.

شُكر وتَقدير

يقول النبي صَلى الله عليه وسلم: (لا يَشْكُرُ اللَّهَ مَنْ لَا يَشْكُرُ النَّاسَ)، لذلك، أتقدم بأسمى آيات الشكر والعرفان إلى كل من ساهم في إظهار هذا الكتاب إلى حيز الوجود سواءً بالمراجعة العلمية واللغوية أو بإخراجه وطباعته على الوجه المطلوب، وأخص بالذكر:

د. عياد كشلاف، عضو هيئة تدريس بكلية الهندسة جامعة صبراتة،

د. نبيل دراويل، عضو هيئة تدريس بكلية الهندسة جامعة طرابلس،

د. الصديق الغراري، عضو هيئة تدريس بكلية الهندسة جامعة غريان،

د. أحمد الرجيبي، عضو هيئة تدريس بكلية الهندسة جامعة صبراتة،

د. صلاح الثلوثي، عضو هيئة تدريس بكلية الهندسة جامعة غريان.

تقديم الكتاب

بسم الله الرحمن الرحيم والصلاة والسلام على أشرف الأنبياء والمرسلين، يسرني ويسعدني في هذه الفقرات تقديم كتاب "المُجْمَل في المفاهيم الأساسية لنُظم تشغيل الحاسوب" لمؤلفه الدكتور/ عمر المبروك أبوزيد.

لقد دخلت الأجهزة المحوسبة، أيًا كان نوعها أو حجمها، في مختلف نواحي الحياة حتى بات من نافلة القول أن الأمية لم تعد أمية القراءة والكتابة، بل صار الجهل باستخدام هذا النوع من الأجهزة هو الأمية الحقيقية في هذا العصر. ولفهم آلية عمل هذه الأجهزة والتمكن منها، لابد للدارس من معرفة نظم التشغيل الخاصة بها معرفة متعمقة تُوسع إدراكه وتُمكنه من التعامل معها بالصورة المثلى وعلى الوجه المطلوب.

وقد جاء هذا الكتاب ليحقق هذا الغرض بصفة خاصة، وليقوم أيضاً بسد النقص الحاصل في المراجع العربية في هذا المجال، فهو يشرح نظم التشغيل وطرق عملها بأسلوب مبسط يغطي المفاهيم الأساسية بصورة وافية. ومما يتميز به هذا الكتاب، احتوائه على العديد من الأمثلة التوضيحية، واختتامه لكل فصل منه بملخص وبأسئلة مرجعية تُساهم في تمكين الدارس من فهم المحتوى العلمي والتفاعل معه بشكل إيجابي بكل سهولة ويسر.

في الختام، إنني أُثَمِّن الجهد الذي بذله المؤلف في جمع وتأليف هذه المادة العلمية، ليُقدم من خلالها المفاهيم الأساسية لنظم تشغيل الحاسوب بين دفَتي كتاب واحد بلغة عربية سلسة مُدعمة برسوم توضيحية وبأسلوب مبسط اعتمد على ربط المكونات وتسهيل عملية القراءة والفهم. ومن وجهة نظري المتواضعة، أعتقد بأن هذا الكتاب، سيقدم إضافة علمية جيدة من شأنها أن تُثري المكتبة العربية، وليكون مرجعًا ومنارًا للدارسين المهتمين بعلوم الحاسوب والراغبين في معرفة أغواره وطرق عمله.

أسأل الله أن يُجازي المؤلف خيراً على ما بذل من جهد وعطاء وأشكر له دعوته لي بكتابة هذا التقديم، وآخر دعوانا أن الحمد لله رب العالمين.

أ. د. محمد سمير البوني
عضو هيئة تدريس بقسم هندسة الحاسوب
كلية الهندسة – جامعة طرابلس

تمهيد الكتاب

بسم الله الرحمن الرحيم، والصلاة والسلام على أشرف الخلق والمرسلين، والحمد لله رب العالمين الذي وفقنا لهذا. يُسعدني أن أُقدم هذا الكتاب إلى القارئ الكريم والذي يتناول شرحًا وافيًا للمفاهيم الأساسية لأنظمة تشغيل الحاسوب.

يَشهد الحاسوب تطورًا سريعًا جعله يُستخدم في أغلب مجالات الحياة، لذلك تُعتبر دراسة علوم الحاسوب وتحديدًا نظم التشغيل من أهم ركائز هذا التطور، لأنها جزء أساسي من أي نظام حاسوبي مستخدم، ومع التنوع الكبير في تطبيقات الحاسوب تبقى المفاهيم الأساسية لهذه النظم موحدة إلى حد ما.

تشمل هذه المفاهيم إدارة العمليات، وإدارة أجهزة الإدخال، والإخراج، ومعضلة حالة الجمود، وإدارة الذاكرة، وكذلك إدارة نظم الملفات، وبعض المفاهيم الأخرى كالشبكات ونظم الحماية والأمن، والتي تناولتها عدة مصادر، ومراجع باللغة الإنجليزية، إلَّا إنَّ الطالب في هذا المجال سيجد شُحًا في الحصول على كتاب مرجعي باللغة العربية يُغَطي هذه المفاهيم بنوع من التفصيل أُسوةً بالمراجع الإنجليزية. لذلك جاء كتاب "المجْمَل في المفاهيم الأساسية لنُظم تشغيل الحاسوب" باللغة العربية لسَد هذه الفجوة العلمية في مجال المراجع المتخصصة في أنظمة تشغيل الحاسوب.

يهدف هذا الكتاب في طبعته الثانية إلى تقديم المبادئ والمفاهيم الأساسية لنظم التشغيل في إطار غير مرتبط بنظام تشغيل محدد، إلَّا إنَّه يُقدم بعض من الأمثلة المتعلقة بأنظمة التشغيل الأكثر شيوعًا في هذا المجال، ويهدف أيضًا إلى توفير المكون النظري لهذه المادة سواءً على مستوى طلبة البكالوريوس أو على مستوى طلبة الدراسات العليا المبتدئين، ويحدو المؤلف الأمل في أن يجده الطالب مفيدًا ويُلبي أيضًا طموحاته التعليمية في التعرف على المفاهيم التي تكمن وراء أنظمة التشغيل في إطار تخصصات علوم الحاسوب، ونظم المعلومات، وهندسة الحاسوب، والهندسة الكهربائية، وهندسة البرمجيات، وتقنية المعلومات. كمتطلبات أساسية لهذا الكتاب، يُفترض أن يكون الطالب على دراية بهياكل البيانات الأساسية، وبتنظيم الحاسوب، وبالأساليب

البرمجية الخاصة به، ومع ذلك لا يزال بإمكانه استيعاب مواضيعه دون معرفة دقيقة بهذه المتطلبات.

بعد اكتمال الفصل الأول من هذا الكتاب والذي تضمن لمحة عن نظم التشغيل، وأنواعها والبنية الهيكلية الخاصة بها، مرورًا ببعض من نظم التشغيل الحديثة والمتاحة، تتطرق بقية فصول الكتاب إلى المفاهيم الأساسية الخاصة بنظم التشغيل من حيث: إدارة كل من العمليات، وأجهزة الإدخال، والإخراج، ومعضلة حالة الجمود، والذاكرة، وكذلك نظم الملفات، والتي تضمنت خمسة فصول تمحورت في التالي:

يتناول الفصل الثاني إدارة العمليات والخيوط، ويُناقش خصائصهم، وكيفية تواصلهم مع بعضهم البعض، كما يُقدم عددًا من الأمثلة التفصيلية عن عملية الاتصالات الداخلية للعمليات والخيوط، وعن المشاكل المصاحبة لمثل هذه الاتصالات والحلول المقترحة لها، وعن كيفية تجنب حدوث مثل هذه المشاكل. أيضًا يُعرِّج هذا الفصل على كل من الجدولة في بيئات التشغيل المختلفة كأنظمة الدفعة، والأنظمة التفاعلية، وأنظمة الزمن الحقيقي. يتضمن التعريج كذلك خوارزميات الجدولة الخاصة بكل بيئة مصحوبة بأمثلة توضيحية، ويُختم في نهايته بموجز وأسئلة للمراجعة.

يلي ذلك الفصل الثالث الذى يُناقش أغلب القضايا المتعلقة بإدارة وحدات الإدخال، والإخراج وأساسيتها المادية، والمعنوية، كما يتطرق أيضًا إلى متحكمات أجهزة الإدخال، والإخراج، والسبل المتاحة لها للتواصل مع وحدة المعالجة المركزية مستعينًا في ذلك بمراجعة المقاطعات وأهداف ومفاهيم برمجيات الإدخال، والإخراج الرئيسية والتي على رأسها مفهوم استقلالية الجهاز. يتحدث هذا الفصل كذلك عن الناقلات الذكية ويتعرض بالتفصيل إلى القرص كأحد أمثلة الإدخال، والإخراج، حيث عُرِّج على كل من كيانه المادي، وعملية تهيئته، والخوارزميات الخاصة بجدولة ذراعه، وكذلك كيفية التعامل مع الأخطا المحتملة له. أخيرًا، يذكر الفصل بعض من الطرق لتحسين أداء الإدخال، والإخراج، وتُذيل نهايته بموجز وأسئلة للمراجعة.

أمَّا حالة الجمود فخُصص لها الفصل الرابع للتعريف بها ومناقشتها، وللتعريف بمصادر الحاسوب وآلية التعامل معها. يُناقش الفصل أيضًا نمذجة حالة جمود المصادر والاستراتجيات التي يُمكن من خلالها التعامل معها من حيث الإهمال، والاكتشاف والتعافي، والمنع، والتجنب.

أخيرًا، يتناول الفصل بعض من المواضيع الأخرى ذات العلاقة بحالة الجمود كجمود الاتصالات والمجاعة، ويُختم في نهايته بموجز وأسئلة للمراجعة.

بينما خُصص الفصل الخامس للمفاهيم الأساسية في إدارة الذاكرة الرئيسية المتعلقة بالتسلل الهرمي لها، وإدارة كل من الذاكرة المفردة، والمجزأة، ومعظم المفاهيم ذات العلاقة بعملية المبادلة. تشمل هذه المفاهيم طرق إدارة فراغ الذاكرة وخوارزميات تخصيص أجزائها، كما يُناقش الفصل أيضًا بنوع من التفصيل مفهوم الذاكرة الظاهرية، جنبًا إلى جنب مع المفاهيم وثيقة الصلة بعملية التصفح وتسريعه، وجداول الصفحة للذواكر الكبيرة. بعدها يُعرِّج الفصل على تسع خوارزميات خاصة بعملية استبدال الصفحات مستعينًا بأمثلة توضيحية وملخص يقارن بينها. ومن ثم يتناول هذا الفصل موضوع التقطيع، وجداوله، وآلية تنفيذه، ومزاياه إضافة إلى ذلك مقارنة بينه وبين التصفح. أخيرًا، يشرح هذا الفصل مفهوم دمج التقطيع والتصفح من خلال التطرق إلى نظام MULTICS، ويُختم في نهايته بموجز وأسئلة للمراجعة.

أخيرًا، يُناقش الفصل السادس إدارة نظم الملفات والكيفية التي يرى بها كل من المستخدم، والمصمم هذه الملفات، وكذلك المهام الأساسية المنبثقة من هذه الإدارة، والتي تمحورت في مجملها حول إدارة كافة العمليات المتعلقة بالملفات والمجلدات، وتتبع كل من المساحات الحرة، والمستخدمة داخل وسائط التخزين الثانوية، وإجراءات تخصيصها، واسترجاعها على التوالي، كما يتعرض للسمات المتعلقة بالملفات، والمجلدات، والعمليات المطبقة عليهما، بما في ذلك بنية كل من الملفات، والمجلدات، وطرق تنفيذها في إطار نظم التشغيل. يُعرِّج الفصل أيضًا على مفهوم مشاركة الملفات بين المستخدمين لما له من دور مهم في تحقيق الأهداف المشتركة بينهم. إضافة إلى ذلك، يُقدم الفصل توضيحًا شاملًا لأهم دلالات توافق هذه المشاركة، ونقاشًا وافيًا لنظام الملف الظاهري والذي يجمع بين أنظمة ملفات مختلفة داخل نظام التشغيل الواحد، بعدها يتحدث الفصل عن اعتمادية نظام الملفات معرجًا على النسخ الإحتياطي وأنواعه، وتوافقية نظم الملف وطرق تعزيزها. أخيرًا، يطرح هذا الفصل بعض من الأمثلة لنظم الملفات الواقيعة، ويُختم في نهايته بموجز وأسئلة للمراجعة.

ما يُميز هذا الكتاب

حرص المؤلف على إعداد هذا الكتاب باللغة العربية وبأسلوب سلس اعتمد على ترابط مكوناته لتسهيل قراءته وفهمه. كما اجتهد المؤلف في تقليل استخدام المصطلحات الإنجليزية داخل النصوص قدر المستطاع واكتفى باستخدامها فقط في الحالات الضرورية الذي يتطلبه سياق الشرح. وحتى يكون الفهم واضحًا ومكتملًا ضمَّن المؤلف في نهاية هذا الكتاب معجم ترجمة باللغة العربية لأغلب المصطلاحات العلمية الواردة فيه، بوبها أبجديًا لتسهيل عملية البحث عنها داخل هذا المعجم. ومن باب حرص المؤلف في هذا الخصوص اعتنى باختيار الترجمة الدقيقة لأي مصطلح أُستُخدم في الكتاب استنادًا إلى الترجمات الشائعة لهذه المصطلحات في مجال علوم الحاسوب، أو في بعض الأحيان إلى الاجتهادات الفردية له، والتي يأمل من الله أن يكون قد وُفِّق فيها.

من المزايا الأخرى لهذا الكتاب هو تقديم العديد من الأمثلة التوضيحية كلما استلزم الأمر ذلك، واختتام كل فصل منه بأسئلة مرجعية تُساهم في جعل الطالب يفهم مادة الكتاب ويتفاعل معها بشكل إيجابي، كما أنه اعتمد على تضمين الألوان داخل النصوص، والجداول، والأشكال التوضيحية لما لذلك من دلالات على تبسيط المعاني وتقريبها وتذكرها بشكل أفضل.

أخيرًا، أُعدَّ الكتاب كحصيلة لعدة سنوات تدريسية لمادة نظم التشغيل استعان فيها المؤلف بعدة مقالات، وصفحات متخصصة في مجالات نظم التشغيل، وكتب مرجعية كان من أهمها مؤلفات تانن باوم، وزلبرشاتس، وجاريدو والتي ساهمت بشكل كبير في وضع أسس هذا الكتاب.

ختامًا أدعو الله العلي القدير أن يُثري هذا الكتاب المكتبة العربية ويسد فجوةً فيها، وأن يستفيد منه كل من اطَّلع عليه.

د. عمر المبروك أبوزيد

غريان–ليبيا : أغسطس 2023م

فهرس محتويات الكتاب

الفصل الأول المقدمة

الفصل الثاني العمليات والخيوط

الفصل الثالث وحدات الإدخال، والإخراج

الفصل الرابع حالة الجمود

الفصل الخامس إدارة الذاكرة

الفصل السادس إدارة نظم الملف

الفصل الأول المقدمة

1.1 مدخل إلى نُظم التشغيل

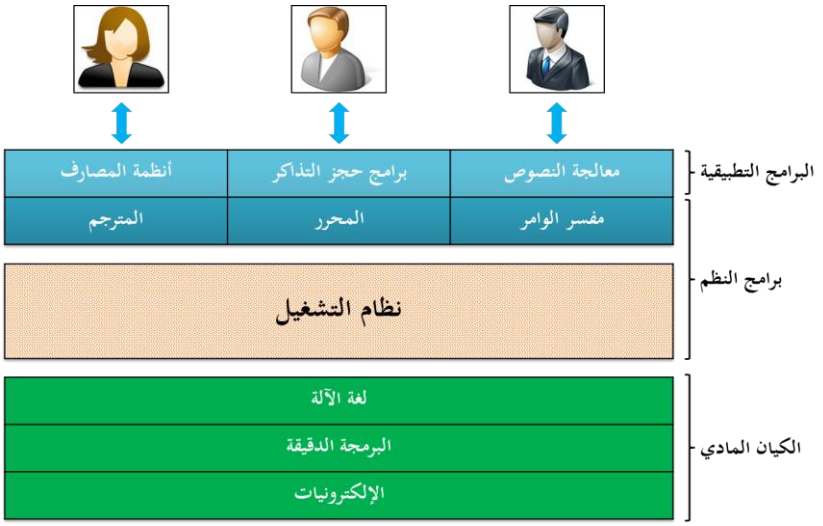
يتكون أي حاسوب من شقين هما: الكيان المادي والكيان المعنوي، هذان الشقان لا يستغنى أحدهما عن الآخر، فهما يمثلان تكاملًا فيما بينهما، فعندما لا يحتوي الحاسوب على الشق المعنوي يكون مجرد صندوق به مكونات كهربائية وإلكترونية، وميكانيكية، عديمة الفائدة. فالشق المعنوي يُمكِّن من معالجة، وتخزين، وتتبع المعلومات، وكذلك القيام بالأنشطة المتعارف عليها في مجال الحاسوب.

ينقسم الكيان المعنوي إلى قسمين: برامج تطبيقية تهدف إلى حل مشاكل المستخدمين، وبرامج النظم التي تدير بنفسها عمليات الحاسوب، تُعتبر نظم التشغيل في حقيقة الأمر أحد أهم برامج النظم، التي تتحكم في كل أجزاء الحاسوب، كما تُوفر البيئة أو الأساس الذي يُساعد في كتابة البرامج التطبيقية.

أمَّا الكيان المادي لأغلب أجهزة الحاسوب الحديثة فيحتوى على معالج أو أكثر، وذاكرة، ومولد نبضات، وأقراص، وكذلك بعض من أجهزة الإدخال، والإخراج، مثل: الشاشة، ولوحة المفاتيح، هذه الأجزاء وغيرها يتم التعامل معها واستخدامها عن طريق البرامج الخاصة بتشغيلها والتحكم فيها. في الواقع هذه المهمة هي ليست بالمهمة السهلة فهي تُكلف الكثير من الجهد والوقت، فكتابة البرامج التي تتبع سير كل هذه المكونات وتستخدمها بالطريقة الصحيحة تُعتبر من أصعب المهام التي يقوم بها المبرمج، لذلك كان من الضرورة بمكان وجود طريقة– ما– تُريح المبرمجين من تعقيدات الكيان المادي وتُخَفِفْ عنهم عبء التعامل معه. وللقيام بذلك وُضعت طبقة من البرمجيات فوق الكيان المادي مباشرة، من أجل إدارة كل أجزاء النظام وتقديم واجهة بينية للمستخدم، أو تقديم ما يُسمى بالآلة الظاهرية بحيث تكون سهلة الفهم والبرمجة، هذه الطبقة من البرمجيات تُدعى نظام التشغيل وهي موضوع هذا الكتاب.

يوضح الشكل 1. 1 البناء التفصيلي لطبقات البرمجيات الموضوعة فوق الكيان المادي، أسفلها طبقة الإلكترونيات والمتمثلة في الدوائر المتكاملة، والأسلاك، ومغديات القدرة، ولوحة المفاتيح وغيرها. يلي هذه الطبقة طبقة البرمجيات الدقيقة، وهي التي تتحكم مباشرة في الطبقة السابقة الذكر، وتُوفر واجهة بينية للطبقة التي تليها، هذه الطبقة عادة ما تكون موجودة في ذاكرة

القراءة فقط، ومن مهامها التأويل، والبحث عن أوامر لغة الآلة[1] مثل، **Add** و **Move** ومن ثم تنفيذها على هيئة سلسلة من الخطوات. مجموعة الأوامر التي تقوم طبقة البرمجيات الدقيقة بتأويلها، تُعرِّف الطبقة التالية في الشكل 1. 1 ألا وهي طبقة لغة الآلة. تتكون هذه اللغة من حوالي 50 إلى 300 أمر، تُستخدم غالبيتها في نقل البيانات داخل الجهاز، وفي إجراء العمليات الحسابية، وكذلك مقارنة القيم والنتائج.



الشكل 1. 1: محتويات نظام الحاسوب: الكيان المادي، وبرامج النظم، والبرامج التطبيقية.

يُتَحكم في أجهزة الإدخال، والإخراج داخل طبقة لغة الآلة عن طريق تحميل قيم محددة في سجل خاص بالجهاز. على سبيل المثال، يمكن إصدار أمر خاص بالقرص للقراءة منه أو الكتابة فيه عن طريق تحميل قيم عنوان القرص، وعنوان الذاكرة الرئيسية، وحجم البيانات، وكذلك الاتجاه (قراءة أو كتابة) في سجل القرص الخاص به.

[1] لغة الآلة هي أحد لغات البرمجة المتدنية المستوى واللغة البرمجية الوحيدة التي يفهمها ويُنفذها الحاسوب مباشرة.

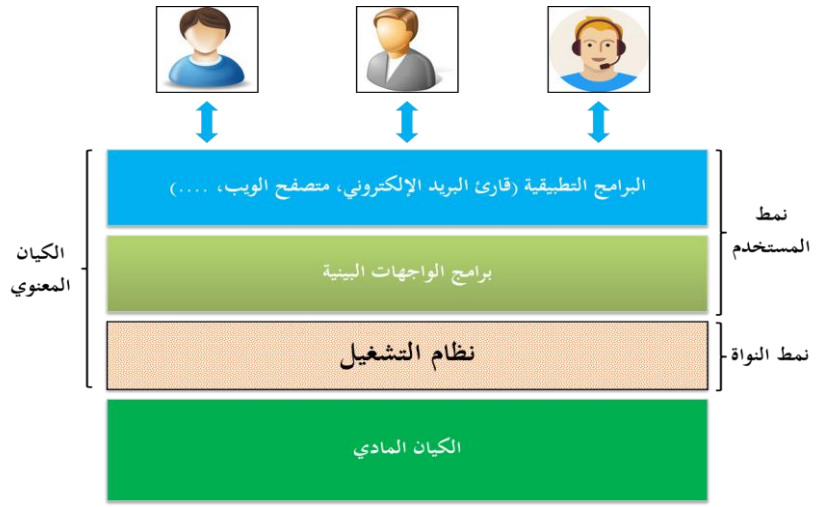
من الناحية التطبيقية هناك عدة معاملات يجب معرفتها لإصدار الأوامر إلى القرص، كما أن الحالات الراجعة بعد إجراء هذه العمليات تكون غاية في التعقيد. إضافة إلى ذلك في حالة وجود عدة أجهزة إدخال، وإخراج متصلة بالحاسوب، فإن عامل الوقت يلعب دورًا مهمًا في عملية البرمجة. لذلك فإن من أهم وظائف نظم التشغيل هو إخفاء كل هذه التعقيدات، وتذليلها أمام المبرمج، وتوفير عدة أوامر سهلة الاستعمال، لكي يتعامل معها المبرمج بكل بساطة. على سبيل المثال لا الحصر، الأمر **'Read block from file'** أي اقرأ مقطعًا من ملف، تجده أبسط بكثير من الاهتمام بعملية تحريك رأس القراءة والانتظار حتى يستقر، وهلم جرا.

توجد طبقة برامج النظم أعلى طبقة الكيان المادي والتي من أهم أجزائها برامج نظام التشغيل. يُنفذ هذا النظام أي أمر متعلق بالوصول إلى الكيان المادي، والتعامل مع وحدات الإدخال، والإخراج. يلي طبقة نظام التشغيل توجد باقي برمجيات النظام والمتمثلة في مفسر الأوامر، والمترجمات، والمحررات، وكذلك أنواع أخرى من البرمجيات المستقلة والمتشابهة، التي تُستخدم من قبل مبرمجي الحاسوب لغرض تحرير كتابة البرامج، وترجمتها، وإنتاج النسخ التنفيذية. أخيرًا، توجد في أعلى هذه الطبقات طبقة البرامج التطبيقية التي هي عبارة عن برامج تُكتب من قبل المبرمجين وتُستخدم من قبلهم لغرض حل مشاكل محددة مثل معالجة البيانات التجارية، والحسابات الهندسية ... إلخ.

من جهة أخرى يُوضح الشكل 1. 2 البناء التجريدي للكيان المعنوي والمتكون في أغلب الحواسيب من نمطي عمليات هما: نمط النواة، ونمط المستخدم. يعمل نظام التشغيل وهو الجزء الأساسي من برمجيات الحاسوب في النمط الأول الذي يُدعى أيضًا نمط المشرف. في هذا النمط، لدى نظام التشغيل الحق في الوصول الكامل إلى كافة أجزاء الكيان المادي وتنفيذ أي تعليمة قابلة للتنفيذ من قبل الحاسوب، بينما تعمل بقية البرمجيات في النمط الأخير الذي تتوفر فيه فقط مجموعة فرعية من تعليمات الآلة. على وجه الخصوص، يُحظر على التعليمات التي تؤثر على السيطرة في الآلة أو القيام بعمليات الإدخال، والإخراج الاشتغال في نمط المستخدم. سنعود إلى توضيح الفرق بين هذين النمطين مرارًا وتكرارًا طي هذا الكتاب.

من خلال هذا الشكل نلاحظ أيضًا أنَّ نظام التشغيل يقع مباشرة بعد الكيان المادي، ليُوفر أساسًا ترتكز عليه بقية البرمجيات الأخرى المذكورة سابقًا. ما يُميز برامج نظام التشغيل عن

البرامج العادية هو أنها إذا لم تكن لدى المستخدم رغبة في استخدام قارئ بريد إلكتروني معين فهو حر في الحصول على قارئ آخر، أو حتى تصميم قارئ جديد خاص به إذا اختار ذلك، لكن ليس لديه الحرية في كتابة برامج معالجة المقاطعة النبضية، التي هي جزء من نظام التشغيل ومحمية من قبل الكيان المادي ضد محاولات تعديله من قبل المستخدمين.



الشكل 1. 2: البناء التجريدي للكيان المعنوي.

في كثير من النظم هناك برامج تشتغل في نمط المستخدم ولكنها تُساعد نظام التشغيل على أداء وظائف مميزة، مثلًا، غالبًا ما يكون هناك برنامج يسمح للمستخدمين بتغيير كلمات المرور الخاصة بهم، هذا البرنامج ليس جزءًا من نظام التشغيل ولا يعمل في نمط النواة، لكنه يحمل بوضوح وظيفة حساسة ويجب حمايته بطريقة خاصة. في بعض من النظم قد يكون الأمر متطرفًا إلى الحد الذي يكون فيه– ما يُعتبر تقليديًا– جزء من نظام التشغيل مثل، نظام الملفات، يشتغل في نمط المستخدم. في مثل هذه الأنظمة، فإنه من الصعب رسم حدود واضحة تفصل بين النمطين، كل شيء يعمل في نمط النواة بشكل واضح هو جزء من نظام التشغيل، ولكن بعض من البرامج التي قد تشتغل خارج هذا النمط يمكن القول أنها جزء منه، أو على الأقل على صلة وطيدة به.

تختلف أنظمة التشغيل أيضًا عن برامج المستخدم أي البرامج التطبيقية في كونها، على وجه

الخصوص ضخمة ومعقدة، وتعيش لفترات زمنية طويلة، فشفرة المصدر لنظام تشغيل 'ويندوز' أو 'يونيكس' قد تحتوي على أكثر من خمسة ملايين سطر من التعليمات البرمجية، ولتصور ما يعنيه هذا يمكن التفكير في طباعة خمسة ملايين سطر في هيئة كتاب، مع 50 سطر في كل صفحة، وألف صفحة في كل مجلد، سيستغرق بالتالي الأمر 100 مجلد لإدراج نظام تشغيل من هذا الحجم وهو ما يستهلك مكتبة كتب بأكملها، ولمعرفة مدى الصعوبة في تتبع هذا النظام يمكنك أن تتخيل بأنك تحصلت على وظيفة صيانة نظام التشغيل، وفي اليوم الأول يجلب لك رئيسك في العمل هذه المكتبة ويقول: "اذهب لتتعلم هذا". هذا فقط خاص بالجزء الذي يعمل في النواة ناهيك عن برامج المستخدم مثل واجهة المستخدم الرسومية، والمكتبات، وبرامج التطبيقات الأساسية، أشياء مثل مستكشف 'ويندوز' الذي قد يتضاعف حجمه بسهولة إلى 10 أو 20 ضعفًا.

والآن يجب أن يكون واضحًا لماذا يجب على أنظمة التشغيل أن تعيش لفترة زمنية طويلة؟ والإجابة ببساطة، لصعوبة كتابتها من جديد، بدلًا من ذلك فإنها تتطور على مدى فترات زمنية طويلة، على سبيل المثال، 'ويندوز 98/95/إم إي' هي في الأساس نظام تشغيل واحد بينما تختلف 'ويندوز إكس بي/2000/إن تي/فيستا'، التي تبدو متشابهة بالنسبة للمستخدمين، وذلك بسبب أن شركة ميكروسوفت حرصت على أن تكون واجهة المستخدم لنظام التشغيل 'ويندوز إكس بي/2000' مماثلة تمامًا للأنظمة المستبدلة.

أخيرًا، سنتطرق في هذا الفصل إلى عدد من الجوانب الرئيسية لأنظمة التشغيل، تشمل ماهيتها، تاريخها، علاقتها بالكيان المادي، ماهية أنواع نظم التشغيل المتاحة، وبعض من مفاهيمها، وهياكلها الأساسية، بالإضافة إلى بعض من أنظمة التشغيل المتاحة. كما أننا سنعود لكثير من هذه المواضيع المهمة في الفصول اللاحقة بمزيد من التفصيل.

2.1 ما هو نظام التشغيل؟

لقد ذكرنا سابقًا في بداية هذا الفصل أن نظم التشغيل هي تلك البرمجيات التي تُزيل العبء عن المبرمج. فما هو تعريف نظام التشغيل؟ وما هي الوظائف التي يقوم بها؟

بالرغم من وجود بعض من الخبرات البرمجية لدى معظم مستخدمي الحاسوب حول نظم

التشغيل، إلَّا إنَّه من الصعب وضع تعريف محدد لنظام التشغيل، وذلك لكونه يقوم في الأساس بوظيفتين غير مرتبطتين بعلاقة بينهما. لذا فإن وضع تعريف له سيكون مشتقًا من الوظيفة التي يقوم بها هذا النظام.

1.2.1 تعريف نظام التشغيل على أساس آلة موسعة

أشرنا سابقًا إلى أن نظام التشغيل هو البرنامج الذي يُريح المبرمج من مشاكل التعامل المباشر مع الكيان المادي. عليه فإنه يمكن تعريف نظام التشغيل على أنه ذلك البرنامج الذي يُخفي حقيقة الكيان المادي عن المبرمج، ويُقدم له طُرقًا بسيطة وسهلة للتعامل مع أجزاء الحاسوب على أساس ملفات يُمكن قراءتها والكتابة عليها.

من وجهة النظر هذه فإن وظيفة نظام التشغيل هي تقديم الآلة الموسعة للمستخدم بحيث تكون أسهل في البرمجة من الكيان المادي المباشر، بالإضافة إلى تنظيم عملية الوصول إلى مصادر الحاسوب (من ذاكرة ومعالج ...)، وتسهيل إجراء عمليات الإدخال، والإخراج.

2.2.1 تعريف نظام التشغيل على أساس مدير المصادر

قبل وضع هذا التعريف سيتم أولًا توضيح ماذا سيحدث عندما تحاول ثلاثة برامج– تشتغل داخل نفس الحاسوب– سحب بعض من النتائج في وقت واحد على نفس الطابعة. هذا الجهاز سيقوم بسحب الأسطر الأولى من البرنامج الأول، ثم الأسطر الأولى من البرنامج الثاني، تليها الأسطر الأولى من البرنامج الثالث وهكذا، الأمر الذي سيجعل نتيجة السحب عشوائية وغير مرتبة.

باستطاعة نظام التشغيل حل هذه المعضلة عن طريق التخزين اللحظي لكل المخرجات المتجهة إلى الطابعة على القرص. عندما ينتهي أحد البرامج من عمله المكلف به، ينسخ نظام التشغيل مخرجاته من القرص إلى الطابعة وفي أثناء ذلك وفي نفس الوقت بإمكان برنامج آخر الاستمرار في توليد مخرجات جديدة.

في أنظمة الحواسيب متعددة المستخدمين تكون الحاجة إلى إدارة وحماية الذاكرة وأجهزة الإدخال، والإخراج، وأي مصادر أخرى خاصة بالحاسوب من المهام الضرورية لاستخدام هذا

النوع من الحواسيب. هذا الإحتياج يرجع إلى التقاسم أو التشارك المستمر من قبل المستخدمين لبعض من المصادر المكلفة. أيضًا من وجهة النظر الاقتصادية فإنه من الضروري تقاسم وتشارك المعلومة من قبل المستخدمين الذين يتعاونون مع بعض من أجل تحقيق وظائف مشتركة.

يُبين كل هذا لنا أن المهمة الرئيسية لنظام التشغيل هي حفظ وتتبع وإدارة استخدام المصادر، وذلك لتلبية كل الطلبات، وتنظيم تخصيص المصادر بالنسبة للمستخدمين، ومنع حدوث تضارب أو تداخل بين الطلبيات من قبل البرامج والمستخدمين. عليه فإن تعريف نظام التشغيل من وجهة النظر هذه هو: ذلك البرنامج الذي يُدير ويتحكم في كل مصادر، وأجزاء الحاسوب، وذلك لغرض تحقيق أفضل أداء. بناءً على هذا التعريف يمكن تلخيص وظائف نظام التشغيل في الآتي:

- مشاركة العتاد بين المستخدمين.
- السماح بمشاركة المعلومات بين المستخدمين أنفسهم.
- منع التداخل بين الوظائف المختلفة التي يؤديها المستخدمون.
- استرداد حالة النظام في حالة حدوث الأخطاء.
- تنظيم البيانات بصورة آمنة وتسهيل سرعة الوصول إليها.

هذه الوظائف وغيرها جعلت تطور نظام التشغيل يستوجب الشروط التالية:

- الملاءمة: بحيث يُوظف نظام التشغيل الكيان المادي ليكون أكثر ملاءمة للاستخدام.
- الكفاءة: بحيث يسمح نظام التشغيل باستخدام مصادر النظام بطريقة أكثر كفاءة.
- قابلية التطور: يجب أن يكون نظام التشغيل مصمما بطريقة تسمح بالتطوير والاختبار المستمر له، مع وجود إمكانية إضافة وظائف جديدة للنظام، دون وجود تعارض مع الوظائف المستحدثة، بالإضافة إلى توفير بيئة مناسبة وسهلة للتطوير بالنسبة لمبرمجي الحاسوب.

3.1 تاريخ نظم التشغيل

بمرور السنوات تطورت نظم التشغيل بشكل كبير وملحوظ، وذلك لارتباطها بتطور الكيان المادي للحاسوب، فكلاهما مرتبط بالآخر ومتأثر به. فكما هو ملحوظ التطور السريع للكيان المادي جعل نظم التشغيل تحتاج إلى تطور مماثل وسريع يُلبي ويُواكب كافة متطلبات تطور الكيان

المادي واحتياجات العصر. عليه سنحاول في هذا القسم إعطاء لمحة مختصرة عن الأجيال أو المراحل التي مرت بها نظم التشغيل، والمرتبطة بشكل وثيق مع مراحل تطور الحاسوب، علمًا أن هذه المراحل قد تتداخل فيما بينها، أي بمعنى قد تبدأ المرحلة التالية قبل إنتهاء سابقتها.

1.3.1 الجيل الأول (1945–1955) الأنابيب المفرغة ولوحات التوصيل

في هذه الفترة الزمنية لم يكن الحاسوب على الشكل الذي نعرفه اليوم، بل كان عبارة عن آلة ضخمة تتكون من عشرات الألوف من الأنابيب المفرغة التي تُوضع داخل حجرة كبيرة. سرعة هذه الآلة كانت أبطأ بكثير من أبطأ حاسوب موجود اليوم، في هذه الفترة لم تكن نظم التشغيل معروفة بعد، حيث كانت المسؤولية تقع على مجموعة من المشغلين المتخصيصين والذين تتمثل مهمتهم في تصميم، وبناء، وبرمجة، وتشغيل، وصيانة كل الآلات. كما كانت تُكتب كافة البرمجيات بلغة الآلة وغالبًا ما يُتَحكم في الوظائف الأساسية للآلة عن طريق لوحات التوصيل.

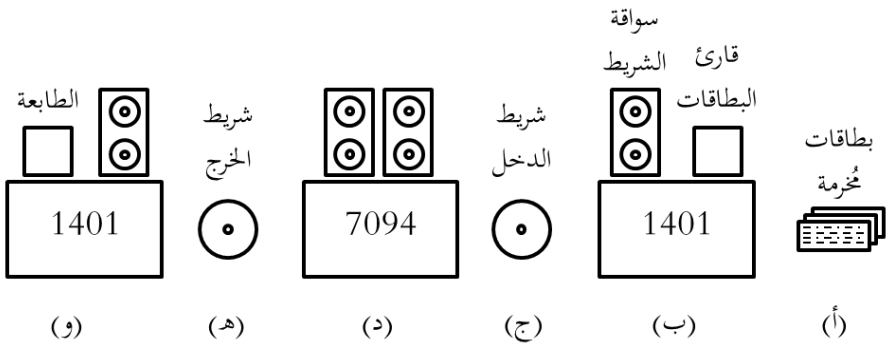
2.3.1 الجيل الثاني (1955–1965) الترانزستور والأنظمة المقطعة

باكتشاف الترانزستور ودخوله عالم الحواسيب في منتصف الخمسينيات تطور الحاسوب بشكل جوهري، بحيث أصبح يقوم بعدة مهام وبطريقة أفضل، الأمر الذي أدى إلى زيادة ثقة المستخدم به، ولأول مرة أصبح هناك فصل واضح بين المصممين، والصناع، والمشغلين، والمبرمجين، وكذلك أفراد الصيانة. وبدأت الآلات تُعرف باسم الحواسيب المركزية التي أصبحت تُوضع في حجر مكيفة ذات مواصفات خاصة وكانت تُشغل من قبل طاقم متخصص مستعينة في ذلك بنسخ مبدئية لنظم التشغيل، كما أن تكلفة هذا النوع من الحواسيب كانت تصل إلى ملايين الدولارات الأمر الذي جعل عملية اقتنائه مقتصرة فقط على المؤسسات الحكومية والكبيرة كالجامعات.

لتنفيذ أي وظيفة– في هذا الجيل من الحواسيب– يكتب المبرمج البرامج بلغة الفورتران أو التجميع على الورق أولًا ومن ثم تخريمها على بطاقات التخريم، ثم تُسلم هذه البطاقات إلى طاقم التشغيل والذي يُدخلها بدوره إلى الحاسوب. بعد إنجاز الحاسوب لهذه الوظيفة، يُسلِّم المشغل المخرجات إلى المبرمج. من خلال تتبع عمل هذه الطريقة، نلاحظ بأنها لا تستغل الحاسوب في حقيقة الأمر بشكل أمثل، وذلك لضياع وقته في أثناء انشغال المشغل بتسلم البطاقات وتسليم

النتائج إلى المبرمج. لذا فإنه ليس من المدهش البحث عن طرق أو حلول بديلة لاستغلال الوقت الضائع من زمن الحاسوب بطريقة مثلى.

ولحل هذه المشكلة أُستخدمت طريقة أنظمة الدفعة. الفكرة الأساسية لهذا الحل تتجلى في الشكل 1. 3، فهي تكمن في تجميع مجموعة من الوظائف المتشابهة وتسجيلها على شريط ممغنط باستخدام حاسوب صغير وغير مكلف مثل **IBM1401**. يتميز هذا الحاسوب بإمكانياته الجيدة في قراءة البطاقات، ونسخ الأشرطة، وكذلك طباعة النتائج. بعد ذلك يُدخَلُ الشريط الممغنط إلى حاسوب آخر مثل **IBM7094** وهو جهاز مرتفع الكلفة، ولكنه مناسب جدًا لإجراء العمليات الحسابية. بعد تحميل الشريط الممغنط على سواقة الشريط يُحمِّل المشغل برنامج نظام التشغيل الذي يقرأ بدوره أول وظيفة ومن ثم يُنفذها.

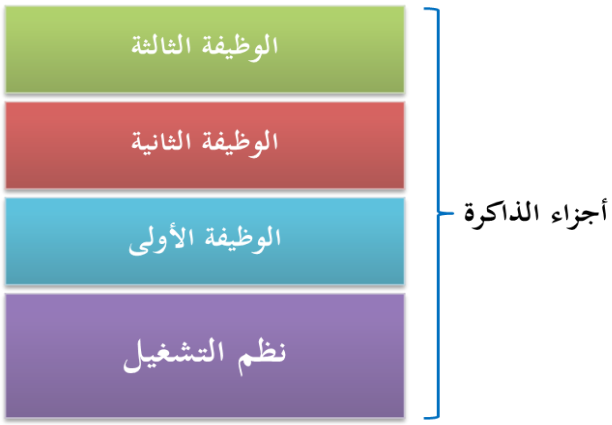


الشكل 1. 3: أنظمة الدفعة القديمة– أ) جلب البطاقات إلى 1401– ب) يُسجِّل الجهاز 1401 الوظائف المقطعة على الشريط– ج) يُحمَّل شريط الدخل إلى 7094– د) يُجهز الجهاز 7094 الحسابات– هـ) يُحمَّل شريط الخرج إلى 1401– و) يطبع الجهاز 1401 الخرج.

تُكتب بعد ذلك مخرجات هذه العملية على الشريط الممغنط بدلًا من طباعتها، وبعد إنجاز أي وظيفة يقرأ نظام التشغيل آليًا الوظيفة التالية، لكي يُنفذها. عندما تُنجز كل الوظائف يستبدل المشغل شريط الدخل بالشريط التالي، ويضع شريط الخرج في الحاسوب من نوع **IBM1401** وذلك من أجل طباعة محتوياته.

3.3.1 الجيل الثالث (1965–1980) الدائرة المتكاملة والبرمجة المتعددة

تطور الحاسوب في بداية هذه الفترة بشكل كبير بحيث أصبح يحتوي على عدة طرفيات للإدخال والإخراج. ساعد هذا الأمر في سرعة الوصول إلى المعلومات، إلَّا إنَّ هذه السرعة مازالت أبطأ بكثير من سرعة المعالج المركزي التي أدت بدورها إلى عدم الاستفادة من سرعته بشكل أمثل. ولحل هذه الإشكالية ظهر ما يُعرف بالبرمجة المتعددة، تتمثل فكرة هذا الحل في تقسيم الذاكرة الرئيسية إلى عدة أجزاء بحيث يحتوي كل جزء على وظيفة أو برنامج مختلف كما هو موضح في الشكل 1. 4. في حالة انتظار إحدى الوظائف لعملية إدخال أو إخراج لكي تُنجز عملها، يكون بإمكان وظيفة أخرى الاستفادة من المعالج المركزي. هذه الطريقة تجعل بالإمكان الاستفادة من المعالج المركزي بنسبة تصل إلى 100٪ تقريبًا، وذلك عندما يكون هناك عدد كاف من الوظائف داخل الذاكرة الرئيسية.



الشكل 1. 4: البرمجة المتعددة لثلاث وظائف موجودة في الذاكرة.

إن ظهور هذا الجيل لم يكن خاليًا من المخاطر والأخطاء، فعلى سبيل المثال، قد يأخذ الوقت بين بدء تنفيذ برنامج معين ووقت الحصول على نتائجه عدة ساعات. فلو تخيلنا وجود خطأ في البرنامج مثل عدم وضع الفاصلة المنقوطة في مكانها الصحيح، سيؤدي ذلك إلى عدم نجاح الترجمة وبالتالي ضياع وقت المبرمج لعدة ساعات. ولحل مثل هذه المشاكل ظهر مفهوم المشاركة الزمنية الذي يتطلب تشغيلًا متزامنًا للبرامج في نماذج منسقة لوقت المعالج المركزي،

بحيث تُجهَّز كافة برامج المستخدمين للتشغيل وبحيث يُحدَّد وقت معين لكل مستخدم للاستفادة من المعالج المركزي بشكل أمثل.

إلى حد الآن لم تختف كل المشاكل بعد، فرغم دخول المشاركة الزمنية التي وفرت استجابة ملائمة للمستخدمين، إلَّا إنَّها لازالت تُعاني من بعض المشاكل، والمتمثلة في كونها لا تتماشى مع التطبيقات المسماة بالتطبيقات ذات الوقت الحقيقي التي تتطلب استجابة في حدود زمنية معينة أو بمعنى آخر في حدود زمنية لحظية، أي فورية.

4.3.1 الجيل الرابع (1980–1990) الحواسيب الشخصية

إنَّ التطور الهائل في عالم الإلكترونيات وظهور الدوائر التكاملية الكبيرة ساهما بشكل كبير في تطور الحواسيب، وظهور الحاسوب الشخصي، وبظهور هذا الجيل ظهر معه نوعان من نظم التشغيل هما: 'إم إس دوس' و 'يونيكس'.

في أواسط الثمانينيات بدأ ظهور نظام شبكات الحاسوب الشخصي، والذي أدى إلى ظهور نظام تشغيل الشبكات وكذلك نظام تشغيل الأنظمة الموزعة. في النظام الأول يمكن للمستخدم الاطلاع على الأجهزة الأخرى الموجودة داخل الشبكة وبإمكان كل مستخدم أن يدخل على الشبكة عن طريق أي جهاز متصل بها ويتحصل على أي معلومة من أي جهاز آخر، وفي نفس الوقت يتمتع كل جهاز بنظام تشغيل خاص به وله مستخدم خاص به (أو مستخدمون). في واقع الأمر لا يختلف نظام تشغيل الشبكات جوهريًا عن نظام تشغيل المعالج الواحد، إلَّا إنَّه يحتاج إلى مراقب واجهة شبكات وبعض من البرمجيات ذات المستوى الدوني، وكذلك بعض من البرامج الخاصة للقيام بالتحكم في عملية الولوج إلى الشبكة والخروج منها.

من ناحية أخرى، يظهر نظام تشغيل الأنظمة الموزعة لمستخدميه على أساس أنه نظام معالج وحيد تقليدي، إلَّا إنَّه في الواقع مصمم لعدد من المعالجات، فاستخدام مثل هذا النظام لا يتطلب من المستخدم أن يكون على دراية بالمكان الذي يبدأ فيه تنفيذ البرنامج ولا على أماكن تواجد ملفاته، لأن ذلك كله من إختصاص نظام التشغيل الذي يقوم بكل هذا آليًا وبكفاءة عالية. على سبيل المثال، يسمح هذا النوع من نظم التشغيل بتنفيذ البرامج على أكثر من معالج في نفس الوقت، إلَّا إنَّ هذا التنفيذ يتطلب وجود خوارزميات جدولة خاصة بالمعالج.

5.3.1 الجيل الخامس (1990 – الحاضر) الحواسيب المتنقلة

تطورت تقنية الاتصالات اللاسلكية في الأربعينيات بشكل ملحوظ بحيث سمحت بظهور نوع جديد من الهواتف عُرف بالهاتف المحمول أو المتنقل. هذا الجهاز بلغ وزنه في عام 1946 حوالي 40 كيلوجرام مما أدى إلى صعوبة كبيرة في نقله من مكان إلى آخر. إلَّا إنَّه مع التطور الهائل في تقنية الإلكترونيات ظهر في السبعينيات أول هاتف يدوي محمول بوزن يقرب من كيلوجرام واحد. اختلف الأمر كثيرا هذه الأيام بحيث أصبحت تقنية الهواتف منتشرة بشكل كبير إلى الحد الذي يمكننا القول فيه بأن حوالي 90٪ من سكان العالم يمتلكونها. علاوة على ذلك، فإن جزئية استعمال الهاتف للتخابر السمعي لم تعد بذلك الأمر المثير للاهتمام، بل تعدَّ الأمر إلى إمكانية استخدامه في تلقي البريد الإلكتروني، وتصفح المواقع على الشبكة المعلوماتية، وإرسال الرسائل النصية والمرئية، واستخدامه كوسيلة ترفيهية.

بالرغم من أن فكرة الجمع بين الهاتف والحاسوب في جهاز يشبه الهاتف كانت موجودة أيضًا منذ السبعينيات، إلَّا إنَّ أول هاتف ذكي حقيقي لم يظهر إلَّا في منتصف التسعينيات مع إصدار نوكيا لجهازها N9000، والذي جمع بين جهازين هما في الغالب منفصلين: الهاتف، والمساعد الرقمي الشخصي. وفي عام 1997، أصدرت شركة إريكسون هاتفها الذكي المسمَّى (Penelope GS88).

في الوقت المعاصر أصبحت الهواتف الذكية موجودة في كل مكان، مما تولدت معها الحاجة إلى وجود أنظمة تشغيل تتلائم مع هذا النوع من الأجهزة، كما برزت مظاهر المنافسة بين هذه الأنظمة. من أنظمة التشغيل المعروفة في هذا المجال نظام 'جوجل أندرويد'[2] يليه نظام التشغيل 'آبل أي أو إس'، إلَّا إنَّ هذا ليس هو الحال دائمًا، فقد يتغير ترتيب هذه الأنظمة في غضون بضع سنوات. فيما يلي سرد لبعض من هذه الأنظمة.

[2] نظام التشغيل 'أندرويد' هو نظام مبني على نظام 'لينوكس' أنتجته شركة 'جوجل' في 2008.

في بداية العقد الأول للهواتف الذكية كانت معظمها تعمل بنظام التشغيل 'سيمبيان' والذي كان مفضلًا لدى عدة شركات تجارية معروفة مثل، سامسونج، وسوني اريكسون، وموتورولا، وخصوصا نوكيا، ومع ذلك، فإن أنظمة تشغيل أخرى أصبحت تُنافس نظام 'سيمبيان' في السوق مثل نظام تشغيل 'بلاك بيري رِم' ونظام 'آبل أي أو إس'. بعدها توقع الكثيرون أن الأول سوف يُهيمن على سوق العمل، في حين أن النظام الأخير من شأنه أن يكون ملك الأجهزة الاستهلاكية. في عام 2011، فشلت 'نوكيا سيمبيان' وأعلنت أنها ستُركز على 'ويندوز' الهواتف كمنصة أساسية لها، لفترة من الزمن ليست بالطويلة، اشتهر كل من نظام 'آبل' و 'رِم' كأنظمة تشغيل للهواتف الذكية (إلَّا إنَّهما لم يهيمنا على السوق مثلما حدث مع نظام 'سيمبيان')، ولكن لم يمضى وقتًا طويلًا حتى تجاوز نظام 'أندرويد' كل منافسيه.

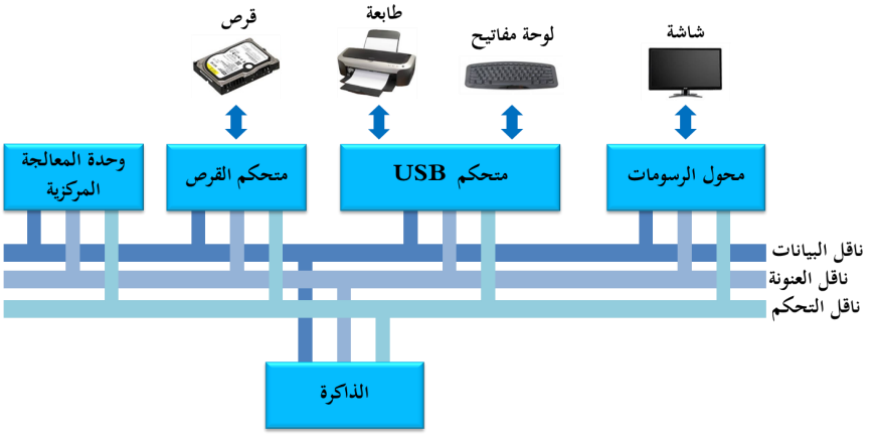
بالنسبة لمصنعي الهواتف، كان نظام 'أندرويد' يتمتع بميزة أنه نظام مفتوح المصدر ومتاح للجميع وذلك بموجب ترخيص متساهل، ونتيجة لذلك، كان بالإمكان تطويره وتكييفه بكل سهولة مع أي جهاز خاص، كما تميَّز هذا النظام بكثرة مستخدميه من مطوري التطبيقات الذين يستخدم معظمهم لغة البرمجة المألوفة حاليًا جافا. ومع ذلك، أظهرت السنوات الماضية أن هيمنته قد لا تدوم، وذلك لحرص منافسيه على استعادة البعض من حصتهم في السوق.

4.1 مراجعة الكيان المادي لنظام الحاسوب

يرتبط نظام التشغيل بشكل وطيد مع الكيان المادي للحاسوب الذي يقوم بتشغيله، فهو الذي يتحكم في مصادره ويعمل على توسيع مجموعة تعليماته، لذلك تَحتَّم على نظام التشغيل معرفة كل ما يتعلق بهذا الكيان، أو على الأقل معرفة الكيفية التي يظهر بها الكيان بالنسبة للمبرمج. لهذا السبب سيعرض هذا القسم مراجعة مختصرة للكيان المادي للحاسوب، والذي نجده في أغلب الحواسيب الشخصية الحديثة، بعدها سيتم التعريج بشكل مفصَّل على الكيفية التي تعمل بها أنظمة التشغيل والمهام التي تقوم بها.

بالنظر إلى الحاسوب الشخصي البسيط يُمكن تجريد بنيته إلى الهيكلية الموضحة في الشكل 1. 5. تتمثل هذه البنية في وحدة المعاجة المركزية، والذاكرة، وأجهزة الإدخال، والإخراج والمرتبطة جميعها بواسطة ناقل النظام، والتي تتصل عن طريقه ببعضها البعض. بالمقابل تَكون بنية

الحواسيب الشخصية الحديثة أكثر تعقيدًا، فهي تتضمن عدة ناقلات سيتم النظر إليها لاحقًا. في الأقسام التالية ستُراجع هذه المكونات بشكل موجز، والتعرض لبعض من الخصائص المادية والمتعلقة بمصممي أنظمة التشغيل.



الشكل 1. 5: المكونات المادية للحاسوب الشخصي الحديث.

1.4.1 المعالجات

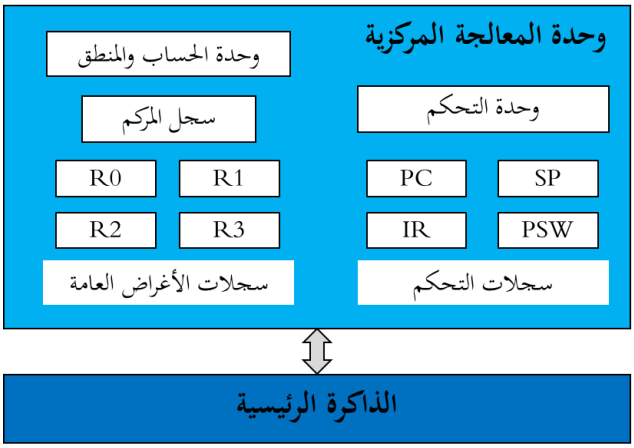
تُعتبر وحدة المعالجة المركزية عقل نظام الحاسوب والمحرك الرئيسي لعمله، فهي التي تجلب التعليمات (البرنامج) من الذاكرة ومن ثم تُنفِذها في دورة أساسية متكاملة. تتمثل هذه الدورة في جلب أول تعليمة من الذاكرة، وفك شفرتها، وتحليلها لتحديد نوعها، ومعاملاتها، ومن ثم تنفيذها. بعد ذلك تتكرر نفس الخطوات مع التعليمات التالية حتى نهاية البرنامج، الأمر الذي يجعل تنفيذ كافة البرامج يجري بنمط واحد.

لكل وحدة معالجة مركزية طقم خاص من التعليمات لا يُمكن تنفيذها إلَّا من قبلها، أي لا يمكن لمعالج إنتل x86– على سبيل المثال– تنفيذ برامج خاصة بمعالج من نوع ARM والعكس صحيح. ولأن التواصل بين وحدة المعالجة المركزية والذاكرة لغرض الحصول على البيانات أو التعليمات يأخذ زمنًا أكبر من زمن تنفيذ التعليمات نفسها، أُضيفت سجلات لوحدات المعالجة المركزية لحفظ المتغيرات الرئيسية والنتائج المؤقتة، كما هو موضح في الشكل 1. 6. هذا الأمر جعل طقم التعليمات يتضمن في العموم تعليمات مثل:

- تحميل كلمة من الذاكرة إلى سجل.
- تخزين كلمة من السجل إلى الذاكرة.

هناك كذلك تعليمات أخرى تُنفذ عمليات على معاملين في سجلين، أو في موقعي ذاكرة، أو في كلاهما مع تخزين الناتج. مثال ذلك عملية جمع عددين في سجلين أو في الذاكرة مع تخزين الناتج في سجل أو موقع في الذاكرة.

بالإضافة إلى السجلات العامة المستخدمة لحفظ المتغيرات والنتائج المؤقتة لدى معظم المعالجات عدة سجلات خاصة منها عدَّاد البرنامج والذي يحتوي على عنوان التعليمة التالية في الجلب من الذاكرة، والذي يُحدَّث بعنوان العملية التالية في التنفيذ بعد كل عملية جلب. السجل الآخر هو مؤشر المكدس والذي يُشير إلى قمة المكدس الحالي في الذاكرة. من هذه السجلات أيضًا سجل كلمة حالات البرنامج والذي يحتوي على خانات إعلام الحالات المحددة من قبل مجموعة تعليمات المقارنة، كما يحتوي كذلك على أولويات وحدة المعالجة المركزية، ونمط التشغيل (مستخدم أو نواة)، وخانات تحكم مختلفة الأغراض. لبرنامج المستخدم الحق في قراءة هذا السجل بالكامل، ولكن ليس له الحق إلَّا في كتابة البعض من حقوله، علمًا بأن سجل كلمة حالات البرنامج يلعب دورًا هامًا في استدعاءات النظام والإدخال، والإخراج.



الشكل 1. 6: بعض من سجلات وحدة المعالجة المركزية.

أخيرًا، يجب على نظام التشغيل أن يكون على دراية تامة بجميع السجلات المستخدمة،

لأنه عند الاشتراك الزمني لوحدة المعالجة المركزية، غالبًا ما يقوم نظام التشغيل بتوقيف برنامج حاليًا قيد التنفيذ لبدء أو لإعادة تشغيل برنامج آخر، وفي كل مرة يتوقف فيها برنامج قيد التنفيذ يجب على نظام التشغيل أن يحتفظ بجميع السجلات بحيث يُمكن استعادتها عند تشغيل نفس البرنامج في وقت لاحق.

بناءً على ما سبق ذكره يمكن تنظيم نظام الحاسوب بعدة طرق والتي بالإمكان تصنيفها تبعًا لعدد المعالجات العامة المستخدمة في النظام والموضحة في الأقسام التالية.

1.1.4.1 الأنظمة أحادية المعالج

إلى زمن ليس بالبعيد كانت أغلب أنظمة الحواسيب تستخدم معالج وحيد يُمثل وحدة معالجة مركزية رئيسية قادرة على تنفيذ مجموعة تعليمات ذات عدة أغراض، بما فيها التعليمات الخاصة بعمليات المستخدم. أيضًا تتضمن أغلب هذه الأنظمة معالجات خاصة بأجهزة محددة تُعرف بالمتحكمات، مثل متحكمات القرص، ولوحة المفاتيح، والرسومات. كما توجد في الحواسيب المركزية معالجات ذات أغراض عامة، مثل معالج الإدخال، والإخراج المسؤول عن النقل السريع للبيانات ما بين مكونات نظام الحاسوب.

تُنفذ كل هذه المتحكمات الخاصة تعليمات محددة لا تتضمن تعليمات المستخدم، كما أنها تُدار في بعض الأحيان من قبل نظام التشغيل بحيث يُرسل لها معلومات عن عملها القادم، ومن ثم يُراقب حالتها. وكمثال على ذلك يستقبل متحكم القرص سلسلة طلبات من وحدة المعالجة المركزية لكي يُنجز لوحده طلب الحصول على البيانات بالإضافة إلى خوارزمية جدولة ملفات البيانات الخاصة به. هذا النمط من التنظيم يُعفي وحدة المعالجة المركزية في النظام من عبء جدولة القرص. كمثال آخر تحتوي لوحة المفاتيح الخاصة بالحاسوب الشخصي على معالج مُصغر تستخدمه لتحويل أحداث الضغط على المفاتيح إلى رموز ثنائية تُرسلها إلى وحدة المعالجة المركزية. في بعض من الأنظمة الأخرى قد تتواجد معالجات ذات أغراض خاصة على المستوى الدوني للكيان المادي وغير مسموح لنظام التشغيل بالاتصال بها، إلَّا إنَّها تُنجز عملها بشكل مستقل، إنَّ استخدام هذا النوع من المعالجات هو أمر شائع ولا يجعل النظام الأحادي يُناظر نظام متعدد المعالجات.

2.1.4.1 الأنظمة متعددة المعالجات

في السنوات القليلة المنصرمة بدأت الأنظمة متعددة المعالجات والتي تُعرف كذلك بالأنظمة المتوازية بالسيطرة على مسرح الحوسبة. تحتوى مثل هذه الأنظمة على معالجين أو أكثر تتصل فيما بينها بشكل مغلق، وتتشارك في ناقل النظام وفي بعض الأحيان قد تتشارك أيضًا في ساعة النظام، والذاكرة، والأجهزة الملحقة. ظهر هذا النوع من الأنظمة في بادئ الأمر في الخوادم ثم انتقل إلى أنظمة الحواسيب الشخصية والمحمولة، مؤخرًا وَجَدت الأنظمة متعددة المعالجات طريقها إلى أجهزة الهواتف النقالة مثل الهواتف الذكية، والحواسيب اللوحية، وذلك للمزايا الثلاث الأساسية التي تُوفرها هذه الأنظمة، والمتمثلة في الآتي:

1. زيادة الإنتاجية

بزيادة عدد المعالجات في النظام من المتوقع إنجاز عمل أكبر في زمن أقل، علمًا بأن معدل التسريع لعدد 'n' من المعالجات لن يكون 'n' بل أقل من ذلك، لأنه عندما تتعاون عدة معالجات في عمل– ما– فإن مقدار من التكلفة العارضة سوف يُخصص للحفاظ على جعل كل أجزاء النظام تعمل بشكل صحيح، هذه التكلفة إضافة إلى التزاحم على الموارد المشتركة سوف يُقلل من الكسب المتوقع نتيجة زيادة عدد المعالجات على النظام.

2. التوفير الاقتصادي

من الممكن أن تكون تكلفة استخدام الأنظمة متعددة المعالجات أقل من استخدام عدة أنظمة أحادية المعالج، وذلك نتيجة لأن الأنظمة متعددة المعالجات يمكنها الاشتراك في الأجهزة الملحقة، ووحدات التخزين، والتغذية الكهربائية، بحيث إذا كانت هناك عدة برامج تعمل على نفس مجموعة البيانات ستكون أرخص تكلفة إذا ما خُزنت هذه البيانات على قرص واحد مشترك بين جميع المعالجات مقارنة باستخدام عدة أنظمة حاسوبية لكل منها ذاكرة محلية وبها عدة نسخ من البيانات.

3. زيادة الاعتمادية

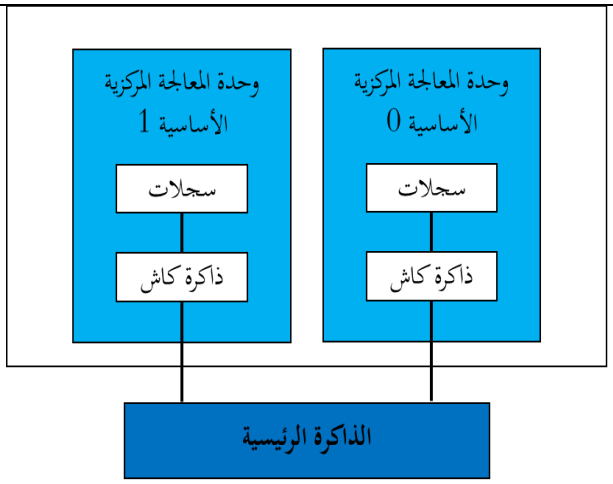
إذا ما وُزِّع العمل بصورة حسنة على مجموعة المعالجات داخل الأنظمة متعددة المعالجات

فإن فشل إحداها لن يُوقف عمل النظام ولكنه سيبطئه فقط، مثلًا، إذا احتوى حاسوب عشرة معالجات وتعطل إحداها، فإنه بإمكان التسع الباقية أن تتقاسم الشغل المناط بالمعالج المعطوب. في هذا الحالة سيتأتر النظام بالكامل سلبًا بمقدار 10٪ بدلًا من أن يتوقف النظام بالكامل. أخيرًا، إن زيادة الاعتمادية في أنظمة الحاسوب مهمة، إلَّا إنَّها حرجة في بعض من التطبيقات.

3.1.4.1 الأنظمة متعددة النوى

يتمثل الاتجاه الحديث في تصميم وحدة المعالجة المركزية في وضع نوى حوسبة متعددة على شريحة واحدة يُطلق عليها اسم الأنظمة متعددة النوى. يُمكن لهذه الأنظمة أن تكون أكثر فاعلية من عدة شرائح لكل منها نواة واحدة، لأن الاتصالات على الشريحة الواحدة أسرع من تلك التي ما بين الشرائح المتعددة. إضافة إلى ذلك، فإن الطاقة المستهلكة من شريحة واحدة متعددة النوى أقل بكثير من الطاقة المستهلكة من عدة شرائح ذوات النواة الواحدة. كما أنه من المهم هنا ملاحظة أن الأنظمة متعددة النوى هي أنظمة متعددة المعالجات، ولكن ليس كل الأنظمة متعددة المعالجات هي متعددة النوى.

يُبين الشكل 1. 7 تصميم ثنائي النواة، أي بنواتين على الشريحة الواحدة. كما هو ملاحظ في هذا التصميم، كل نواة لها سجلات، وذاكرة كاش محلية خاصة بها، إلَّا إنَّه قد تتواجد في تصاميم أخرى ذاكرة كاش مشتركة، أو مزيج من ذاكرة كاش محلية، وأخرى مشتركة. وبصرف النظر عن الاعتبارات المعمارية لكل من ذاكرة كاش، والذاكرة الرئيسية، وتزاحم الناقلات، فإن وحدات المعالجة المركزية متعددة النوى تظهر لنظام التشغيل على أساس أن هناك عدد 'n' من المعالجات، الأمر الذي سيُرهق كل من مصممي نظام التشغيل ومبرمجي التطبيقات للاستفادة من الأنظمة متعددة النوى.



الشكل 1. 7: تصميم ثنائي النواة، أي بنواتين على الشريحة الواحدة.

4.1.4.1 الأنظمة العنقودية

من الأنواع الأخرى للأنظمة متعددة المعالجات هي الأنظمة العنقودية والتي هي عبارة عن تجمع لعدة وحدات معالجة مركزية. تختلف هذه الأنظمة عن الأنظمة متعددة المعالجات– التى جرى وصفها سابقًا– في كونها تتألف من اثنين أو أكثر من الأنظمة الأحادية (أو العنقودية) مقترنة ببعضها البعض، بحيث يُمكن أن تُمثل كل عقدة نظام أحادي المعالج أو نظام متعدد النوى. هنا ينبغي التنويه إلى تفاوت التعريفات الخاصة بالأنظمة العنقودية، إلَّا إنَّ التعريف المقبول عمومًا في كونها تتمثل في تلك الأنظمة التي تشترك في نظام تخزين البيانات وتترابط بشدة عن طريق شبكة محلية أو أي رابط سريع.

من مزايا الأسلوب العنقودى أنه يُستخدم لتوفير خدمة ذات وفرة عالية، بمعنى استمرار الخدمة حتى لو فشل نظام أو أكثر داخل العنقود نفسه، في العموم تأتي هذه الوفرة عن طريق إضافة مستوى من التكرار في الأنظمة العنقودية عن طريق جعل طبقة من البرامج العنقودية تعمل على عقد النظام بحيث تُمكِّن كل عقدة من مراقبة عقدة أو أكثر من العقد الآخرى (عبر الشبكة المحلية). هذه الوضعية تجعل بالإمكان تعويض العقدة المعطوبة عن العمل بالعقدة الراصدة لها مع امتلاك قدراتها التخزينية والقيام بإعادة تشغيل التطبيقات التي كانت تُنفذ من قبل العقدة

المتضررة، لذلك لن يلاحظ المستخدمين وعملاء التطبيقات سوى انقطاع قصير في الخدمة.

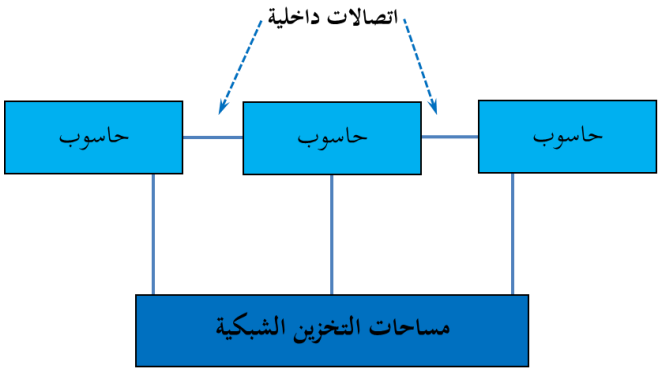
للقيام بعمليات المراقبة يُمكن تنظيم الأنظمة العنقودية إمّا بشكل متناظر، أو غير متناظر. في النظم العنقودية غير المتناظرة تكون هناك آلة واحدة في وضع استعداد (أي ساخن) تتمثل مهمتها فقط في مراقبة الخادم النشط في حين تقوم الأخرى بتنفيذ التطبيقات. في حالة تعطل هذا الخادم ستستلم العقدة الساخنة مهامه وتكون بذلك هي الخادم النشط، أمَّا في النظم العنقودية المتناظرة فستكون هناك عقدتين أو أكثر تُنفذ التطبيقات وتُراقب بعضها البعض. هذا التنظيم هو بوضوح أكثر كفاءة، وذلك لكونه يستخدم كل الموارد المتوفرة، إلَّا إنَّه يتطلب توفر عدة تطبيقات متاحة للتنفيذ.

ولأن النظم العنقودية تتضمن عدة عقد حاسوبية متصلة ببعضها عبر شبكة، فإن هذه الأنظمة يمكنها أن تُستخدم لتوفير محيط حوسبة عالي الأداء يتمتع بقوة حاسوبية أكبر من تلك التي تُوفرها الأنظمة أحادية المعالج، أو حتى الأنظمة التناظرية متعددة المعالجات، وذلك لإمكانية تشغيل تطبيق– ما– بشكل متزامن على كافة العقد الحاسوبية في النظام. هذا التطبيق يجب كتابته بشكل خاص لكي يتسنى الاستفادة من العقد العنقودية بشكل أفضل، الأمر الذي يتضمن استخدام تقنية التوازي، والتي يُقسم فيها البرنامج إلى مكونات منفصلة يُمكن تنفيذها بالتوازي على حواسيب فردية داخل نظام عنقودي. تُصمم هذه التطبيقات في العادة بحيث أنه بمجرد إنتهاء كل عقدة حاسوبية في العنقود من حل حصتها من المشكلة، تُجمَّع النتائج من كافة العقد في حل نهائي.

هناك أشكال أخرى من الأنظمة العنقودية تتضمن الأنظمة العنقودية المتوازية، والأنظمة العنقودية المرتبطة بشبكة موسعة. يسمح النوع الأول لعدة عقد بالوصول إلى نفس البيانات على مساحة تخزين مشتركة، ولأن معظم أنظمة التشغيل تفتقر إلى دعم يُمَكِّنها من الوصول إلى نفس البيانات في وقت واحد من قبل عدة عقد، فإن هذا النوع يتطلب استخدام نسخ خاصة من البرمجيات وإصدارات خاصة من التطبيقات.

بالمقابل تدعم الأنظمة العنقودية الموسعة عشرات العقد في العنقود الواحد، بالإضافة إلى عقد عنقودية منفصلة عن بعضها البعض بعدة كيلومترات ومتصلة عبر الشبكة الموسعة. الفضل في

التحسينات التى واكبت الأنظمة العنقودية يعود إلى مساحات التخزين الشبكية والتي هي عبارة عن تقنية تُتيح للعديد من الأنظمة أن ترتبط بمساحات تخزينية موحدة بواسطة الشبكة الموسعة. لذلك إذا كانت التطبيقات والبيانات مُخزنة على هذه المساحات التخزينية، فإنه يُمكن للبرنامج العنقودى أن يختار تطبيق– ما– للعمل على أي عقدة مرتبطة بالشبكة، وفي حالة فشلها فإنه بإمكان أي عقدة أخرى تعويضها، الأمر الذي سيُحقق زيادة كبيرة في الأداء والاعتمادية. يُوضح الشكل 1. 8 الهيكل العام للنظام العنقودي.



الشكل 1. 8: الهيكل العام للنظام العنقودي.

2.4.1 هيكلية التخزين

لتنفيذ أي برنامج في نظام الحاسوب، يجب على وحدة المعالجة المركزية تحميل التعليمات من الذاكرة، الأمر الذي يتطلب أولًا حفظ كل من هذه التعليمات والبيانات اللازمة للعمل فيها. لذلك تُنفذ أنظمة حاسوب الأغراض العامة معظم برامجها من ذاكرة قابلة لإعادة الكتابة تُسمى الذاكرة الرئيسية أو ذاكرة الوصول العشوائي، هذه الذاكرة عادة ما تُصمَّم من تقنية أشباه الموصلات وتُعرف بذاكرة الوصول العشوائي التفاعلية.

بالإضافة إلى ذلك تستخدم أنظمة الحاسوب أشكالًا أخرى من الذاكرة، منها على سبيل المثال، ذاكرة القراءة فقط، وذاكرة القراءة فقط القابلة للمسح والبرمجة كهربائيًا. ولأن محتوى ذاكرة القراءة فقط لا يمكن تغييره، يُخزن فيها فقط البرامج الثابتة مثل، برنامج استنهاض الحاسوب الذي سيُناقش في القسم 5.4.1، كما تُستخدم في خراطيش اللعب الإلكترونية.

بالمقابل يمكن تغيير محتوى النوع الثاني من ذاكرة القراءة فقط ولكن ليس بصورة متكررة، لذلك فهي تحتوي على البرامج التى تغلب عليها طبيعة البرامج الثابتة كتلك التي تتواجد في الهواتف الذكية والمستخدمة في تخزين البرامج المثَبتَة مسبقًا في أثناء التصنيع.

بغض النظر على نوعية الذاكرة فإن جميعها يُقدم مصفوفة من الثمانيات (المواقع) لكل منها عنوان خاص به يُتعامل معه بواسطة سلسلة من تعليمات التحميل أو التخزين لعناوين مواقع محددة في الذاكرة. فتعليمة التحميل **load** تنقل بايت أو كلمة من الذاكرة الرئيسية إلى سجل داخلي في وحدة المعالجة المركزية، في حين أن تعليمة التخزين **store** تنقل محتوى سجل إلى الذاكرة الرئيسية. وبصرف النظر عن التحميل والتخزين الصريح فإن وحدة المعالجة المركزية تُحمل تلقائيًا التعليمات من الذاكرة الرئيسية ومن ثم تُنفذها.

بناءً على نظام ببنية نموذج فون نيومان[3] تأخذ دورة تنفيذ التعليمة التقليدية عدة مراحل. أولًا: مرحلة جلب التعليمة من الذاكرة وتخزينها في سجل التعليمة، وثانيًا: مرحلة فك شفرتها، الأمر الذي قد يتسبب في جلب المعاملات من الذاكرة وتخزينها في سجل داخلي، وثالثًا: مرحلة تنفيذ التعليمة على المعاملات وتخزين النتيجة في الذاكرة. نموذجيًا من المفضل أن تتواجد البرامج والبيانات في الذاكرة الرئيسية بشكل دائم، إلَّا إنَّ هذا الترتيب غير ممكن عادة وذلك للسببين التاليين:

1. حجم الذاكرة الرئيسية صغير جدًا إذا ما قورن بحجم جميع البرامج والبيانات المراد حفظها بشكل دائم.
2. الذاكرة الرئيسية هي وحدة تخزين متطايرة أي تفقد محتواها عند إيقاف التغذية الكهربائية عنها.

بالتالي، فإن معظم أنظمة الحاسوب تمتلك وحدات تخزين ثانوية تُعتبر امتدادًا للذاكرة الرئيسية، تتمثل مهمتها الرئيسية في تخزين كميات كبيرة من البيانات وبشكل دائم.

---

[3] راجع المرجع **Null & Lobur, 2016** للتعرف على بنية فون نيومان (**Von Neumann**).

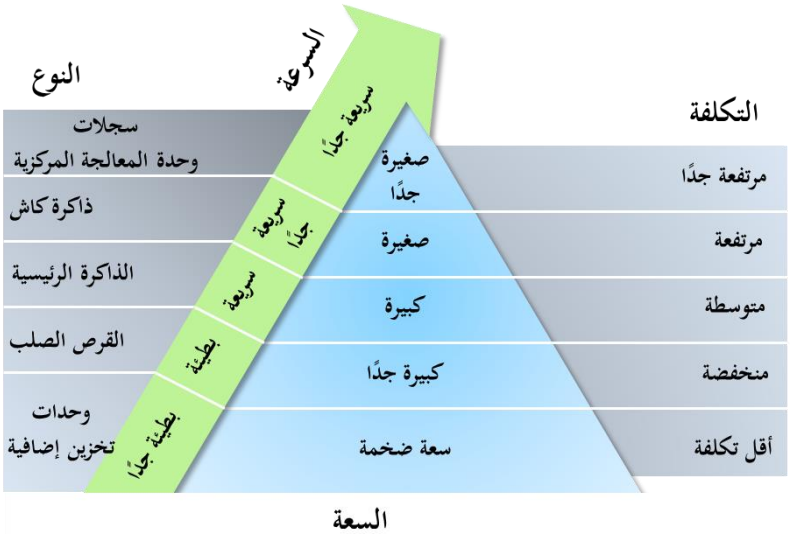
من أجهزة التخزين الثانوية الأكثر شيوعًا الأقراص الممغنطة، والتي يمكنها توفير مساحات تخزين كبيرة لكل من البرامج والبيانات. فمعظم البرمجيات (سواءً أكانت خاصة بالنظام أو بالتطبيقات) تُحفظ على الأقراص ومن ثم تُحمَّل في الذاكرة في أثناء فترة التنفيذ، يجعل هذا الأمر العديد من البرمجيات تستخدم القرص كمصدر للبيانات، ووجهة للحفظ خلال مراحل المعالجة. بالتالي فإن الإدارة الجيدة لوحدات التخزين الثانوية ذات أهمية محورية في نظام الحاسوب، كما سيتبين ذلك من خلال مناقشة هذا الموضوع لاحقاً في الفصل السادس.

بصورة أشمل، إن بنية التخزين التي ذُكرت إلى حد الآن (متضمنة السجلات، والذاكرة الرئيسية، والأقراص الممغنطة) هي واحدة من عدة أنظمة تخزين ممكنة. تشمل هذه الأنظمة الذاكرة السريعة، والأقراص الضوئية، والأشرطة الممغنطة، وغيرها، والذي يُوفر كل منها الوظائف الأساسية لتخزين البيانات وحفظها حتى يتم استرجاعها في وقت لاحق، إلَّا إنَّ أوجه الاختلافات الرئيسية بينها تكمن في السرعة، والحجم، والتكلفة، وكذلك الاستقرارية والاعتمادية، أي بقاء البيانات صحيحة لفترة زمنية طويلة.

يُمكن تنظيم هذه المجموعة المختلفة من أنظمة التخزين في شكل هرمي وفقًا للسرعة والتكلفة، وذلك كما هو موضح في الشكل 1. 9. المستويات العليا منها غالية التكلفة ولكنها سريعة، فكلما اتجهنا لأسفل التسلسل الهرمي تقل تكلفة كل خانة ثنائية ويزداد زمن الوصول بشكل عام. المفاضلة بين هذه الأنظمة تكون مقبولة، لأنه إذا كان هناك نظام تخزين– ما– أسرع وأقل تكلفة من نظام آخر ولكن متساويان في الخصائص الأخرى فلن يكون هناك أي سبب لاستخدام ذاكرة أبطأ وأكثر تكلفة. في الحقيقة العديد من أنظمة التخزين القديمة بما فيها الأشرطة الورقية أصبحت من الثرات الآن وذلك نتيجة لتفوق الوسائط الحديثة عليها من حيث السرعة والتكلفة.

بالإضافة إلى الفروقات في السرعة والتكلفة تتنوع وحدات التخزين في كونها متطايرة أو غير متطايرة، فالذاكرة المتطايرة تفقد محتواها عند نزع التغذية الكهربائية عنها وفي غيابها يجب حفظ البيانات في الذاكرة من نوع غير المتطايرة. في التسلسل الهرمي المبين في الشكل 1. 9، أول ثلاث نظم تخزين من الأعلى هي نظم متطايرة، بينما تمثل البقية نظم التخزين غير المتطايرة.

أخيرًا، إن تصميم أي نظام ذاكرة متكامل يجب أن يخلق نوع من التوازن بين جميع العوامل التي نُوقشت سابقًا، بمعنى يجب على نظام الحاسوب أن يستخدم قدرًا من الذاكرة المكلفة في الحالات الضرورية مع توفير أكبر قدر ممكن من الذاكرة غير المتطايرة والرخيصة، كما يمكنه استخدام الذاكرة السريعة لتحسين الأداء في الحالات التي يوجد فيها تفاوت كبير في وقت الوصول أو معدل النقل بين مكونات الحاسوب، سيتم الرجوع إلى مناقشة أسس إدارة الذاكرة بنوع من التفصيل في الفصل الخامس.



الشكل 1. 9: الشكل العام لهيكلية التخزين.

3.4.1 هيكلية الإدخال، والإخراج

ضمن نظام الحاسوب، هناك جزء كبير من الشفرة البرمجية لنظام التشغيل مُكرس لإدارة الإدخال، والإخراج، وذلك بسبب أهميتها بالنسبة لموثوقية وأداء النظام، وكذلك بسبب الطبيعة المختلفة لأجهزة الإدخال، والإخراج. يُقدم هذا القسم لمحة عامة عن هيكلية الإدخال، والإخراج.

يتكون نظام حاسوب الأغراض العامة في العادة من وحدات المعالجة المركزية، وذاكرة، وعدة متحكمات خاصة بالأجهزة الملحقة، والتي ترتبط جميعها بواسطة ناقل مشترك. تقع

مسؤولية كل متحكم على نوع معين من الأجهزة، وقد يُرفق به أكثر من جهاز واحد وذلك حسب طبيعة هذا المتحكم، كما هو الحال مع متحكم واجهة نظام الحاسوب الصغير، والذي يُمكنه إرفاق سبعة أجهزة أو أكثر به. غالبًا ما يتضمن متحكم الجهاز وحدة محلية للتخزين المؤقت، ومجموعة من السجلات ذات أغراض خاصة، كما تتمثل مهمته الرئيسية في نقل البيانات بين الأجهزة الطرفية التي يتحكم فيها ووحدة تخزينه المؤقتة. وللقيام بذلك عادة ما تمتلك أنظمة التشغيل برنامج تشغيل معنوي لكل متحكم جهاز يُعرف باسم مشغل الجهاز، والذي عن طريقه يُمكن لنظام التشغيل التخاطب والتفاهم مع متحكم الجهاز، كما يُوفر واجهة موحدة لكل بقية مكونات هذا النظام.

يلعب مشغل الجهاز دورًا مهمًا في بدء عملية الإدخال، والإخراج فهو يُحمِّل السجلات المناسبة في المتحكم، والذي بدوره يفحص محتوياتها لتحديد الإجراءات التي يجب اتخاذها مثل، قراءة حرف من لوحة المفاتيح. يبدأ المتحكم في نقل البيانات من الجهاز إلى وحدة التخزين المؤقت المحلية، وعند اكتمال عملية النقل، يُعْلِم المتحكم المشغل عن طريق المقاطعة بإنتهاء العمل المناط به، عندها يُرجِّع المشغل التحكم إلى نظام التشغيل، وربما أيضًا يُرجِّع بيانات أو مؤشر للبيانات، إذا كانت العملية قراءة، أمّا في حالة قيامه بعمليات أخرى فإنه يُرجِّع معلومات الحالة.

تُعرف العملية السابقة بالإدخال، والإخراج بالمقاطعة وهي مفيده في نقل كميات قليلة من البيانات، ولكنها غير مجدية عند استخدامها لنقل بيانات بكميات كبيرة كعملية الإدخال، والإخراج الخاصة بالقرص. لحل هذه المشكلة تُستخدم تقنية الوصول المباشر للذاكرة. فبعد إعداد وحدات التخزين المؤقت، والمؤشرات، وعدَّادات جهاز الإدخال، والإخراج، ينقل متحكم الجهاز قالب كامل من البيانات مباشرة إلى أو من وحدة التخزين المؤقتة المحلية إلى الذاكرة دون أي تَدَخل من قبل وحدة المعالجة المركزية. يتطلب هذا الأمر مقاطعة واحدة فقط لكل قالب، وذلك لإعلام مشغل الجهاز باكتمال عملية النقل، بدلًا من مقاطعة واحدة لكل بايت في حالة الأجهزة المنخفضة السرعة. تسمح هذه الآلية لوحدة المعالجة المركزية بإنجاز أعمال أخرى في أثناء انشغال متحكم الجهاز بعمليات النقل. سنعود للتطرق لهذا الموضوع بالتفصيل في الفصل الخاص بوحدات الإدخال، والإخراج (الفصل الثالث).

4.4.1 الناقلات

حسب نموذج فون نيومان لنظام الحاسوب يُعتبر ناقل النظام أحد أهم أجزاء نظام الحاسوب ويختص بربط مكوناته ويستخدمه لنقل مختلف الإشارات والبيانات فيما بينها. يتضمن هذا الناقل ناقل البيانات لنقل البيانات، وناقل العنونة لنقل العنوان الذى يُحدد وجهة أو مصدر البيانات، وناقل التحكم لنقل إشارات التحكم والسيطرة التى تتحكم في مكونات النظام حسب التعليمة المنفذة. تتنوع هذه الناقلات من حيث البنية، والسرعة، والإنتاجية، وكذلك طرق توصيلها. يُوضح الشكل 1. 5 أبسط التنظيمات الهيكلية للناقلات والمستخدمة في الحواسيب الصغيرة وكذلك الحاسوب الشخصي نوع **IBM**. ولكن مع تطور المعالجات وازدياد سرعة الذاكرة، وصلت قدرة الناقل البسيط في التعامل مع الكثافة المرورية للبيانات إلى نقطة الانهيار، الأمر الذي أوجب إضافة ناقلات إضافية وذلك لمواكبة أجهزة الإدخال، والإخراج السريعة وحركة النقل مابين المعالج والذاكرة. مع هذا التطور تغير نظام ناقل الحاسوب والمعتمد على معالجات عائلة إنتل **x86** إلى الناقلات الموضحة في الجدول 1. 1.

هذه الناقلات ذات معدلات نقل ووظائف مختلفة يجب على نظام التشغيل أن يكون مُلمًا بها كي يتمكن من تهيئتها وإدارتها. يُعتبر الناقل الرئيسي في هذه الناقلات والمدرج في الجدول 1. 1 هو ناقل الرابط السريع للمكونات الملحقة – **PCIe**.

الجدول 1. 1: ناقلات الحاسوب المعتمدة في معالجات عائلة إنتل x86.

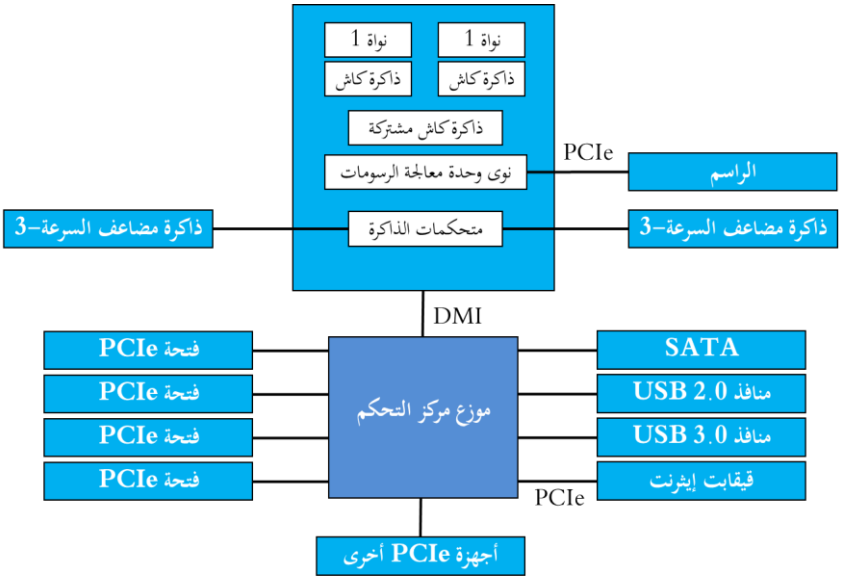
الناقل	الرمز	المصطلح
الرابط السريع للمكونات الملحقة	PCIe	Peripheral Component Interconnect Express
واجهة الوسائط المباشرة	DMI	Direct Media Interface
المرفق التقني التسلسلي المتقدم	SATA	Serial Advanced Technology Attachment
الناقل التسلسلي العام	USB	Universal Serial Bus
ناقل الذاكرة مضاعف السرعة–3	DDR3	Double Data Rate Type Three

صُمِّم ناقل الرابط السريع للمكونات الملحقة من قبل شركة إنتل خلفًا لناقل رابط المكونات الملحقة القديم، الناقل الجديد قادر على نقل عشرات القيقا بايت في الثانية الواحدة، وهو أسرع بكثير من سابقه إلَّا إنَّه يختلف عنه في طبيعته. وحتى العام 2004 الذي صُمم فيه هذا الناقل كانت معظم الناقلات متوازية ومشتركة. فالبنية المشتركة لناقل النظام تعني أن عدة أجهزة يمكنها التشارك في نفس الأسلاك لنقل البيانات. بالتالي، عندما يكون لهذه الأجهزة بيانات ترغب في إرسالها فإنها تحتاج إلى إشارة تحكم لتحديد مَن منها يُمكنه استخدام الناقل، أمَّا بنية الناقل المتوازي والمستخدمة في الرابط التقليدي للمكونات الملحقة فهي تُمكن النظام من إرسال كل كلمة من البيانات عبر عدة أسلاك متوازية، فمثلًا عبر هذا الناقل يمكن إرسال عدد مكون من 32 خانة ثنائية على 32 سلك متوازي.

بالمقابل يستخدم ناقل الرابط السريع للمكونات الملحقة وصلات نقطية مخصصة أي من نقطة إلى نقطة، الأمر الذي جعله يستخدم بنية الناقل التسلسلي والتي تُرسل كل الخانات الثنائية في رسالة عن طريق وصلة واحدة تُعرف باسم الممر، وذلك كما هو الحال في حزمة البيانات في الشبكات الحاسوبية. هذا الأمر يجعل العمل أكثر بساطة، لأنه ليس من الضرورى التأكد من أن جميع الخانات قد وصلت إلى وجهتها في نفس الوقت بالضبط. ومع هذا لازال التوازي مستخدمًا بحيث يمكن إرسال عدة ممرات في نفس الوقت كإرسال 32 ممر لنقل 32 رسالة في نفس الوقت، أي بالتوازي. ومع هذا كله، ولأن سرعة الأجهزة الطرفية مثل البطاقات الشبكية في ازدياد كبير، تُطور مقاييس الرابط السريع للمكونات الملحقة كل 3 إلى 5 سنوات. على سبيل المثال، الرابط السريع للمكونات الملحقة ذو الإصدار 2.0 له 16 ممر ويوفر سرعة نقل تصل إلى 64 قيقا بايت في الثانية، وبترقيته إلى الإصدار 3.0 ستتضاعف هذه السرعة، أمَّا الإصدار 4.0 سيُضاعف ذلك مرة أخرى.

بالرغم من هذا التطور، لازالت هناك عدة أجهزة حاسوبية تراثية تستخدم رابط المكونات الملحقة، كما هو مبين في الشكل 1. 10 والموصولة بمعالج محوري منفصل. كتطوير مستقبلي من الممكن توصيل جميع الأجهزة التي تستخدم رابط المكونات الملحقة القديم بموزع لربطهم بالموزع الرئيسي مما ينشأ عنه شجرة الناقلات، في هذه الحالة تتخاطب وحدة المعالجة المركزية مع الذاكرة عبر ناقل الذاكرة المضاعف السرعة–3 السريع، ومع جهاز الرسومات الخارجي عن

طريق الرابط السريع للمكونات الملحقة، ومع كافة الأجهزة الأخرى عبر موزع يتصل بناقل واجهة الوسائط المباشرة. هذا الموزع يربط بدوره جميع الأجهزة الأخرى مستخدمًا الناقل التسلسلي العام للتخاطب مع الأجهزة التسلسلية، وناقل المرُفق التقني التسلسلي المتقدم للتواصل مع الأقراص الصلبة ومشغلات الأقراص الضوئية دي في دي، ويُستخدم الرابط السريع للمكونات الملحقة لنقل الحزم الشبكية (الإيثرنت) بمعدل واحد قيقا خانة ثنائية، علاوة على ذلك، يُمكن من الشكل 1. 10 ملاحظة أن لكل نواة من النواتين ذاكرة سريعة خاصة به وذاكرة سريعة أكبر مشتركة بينهما، ولكل منهما ناقل إضافي.



الشكل 1. 10: هيكلية نظام x86 [Tanenbaum & Bos, 2015].

من ناحية أخرى أُضيف الناقل التسلسلي العام لغرض إلحاق كل أجهزة الإدخال، والإخراج البطيئة بالحاسوب مثل لوحة المفاتيح والفأرة، وهو يستخدم وصلة صغيرة لها أربعة أسلاك إلى إحدى عشرة سلكًا وذلك اعتمادًا على الإصدار الخاص بالناقل، البعض من هذه الأسلاك يُستخدم لإمداد الطاقة الكهربائية لأجهزة الناقل التسلسلي أو ربطها بالأرضي. كما يعمل الناقل التسلسلي كناقل مركزي بحيث يستخدمه الجهاز الرئيسي لاستفتاء كافة أجهزة الإدخال، والإخراج كل ملي ثانية لغرض معرفة ما إذا كان هناك أي عملية نقل. إضافة إلى ذلك فإن أي جهاز يدعم

تقنية الناقل التسلسلي العام يُمكن وصله بالحاسوب وسيعمل مباشرة دون الحاجة إلى إعادة تشغيله.

بالإضافة إلى ما سبق ذكره من الناقلات، هناك ناقل يُسمى ناقل واجهة نظام الحاسوب الصغير وهو ناقل عالي الأداء ومخصص للأقراص السريعة، والماسحات الضوئية، وغيرها من الأجهزة التي تحتاج إلى سعة نقل كبيرة، والتي نجدها غالبا في الخوادم، ومحطات العمل حيث يمكن تشغيلها بسرعة تصل إلى 640 ميقا بايت لكل ثانية.

أخيرًا، عند العمل في بيئة كالتي في الشكل 1. 10، فإنه يتحتم على نظام التشغيل معرفة ماهية الأجهزة الطرفية التى رُبطت بالحاسوب لغرض تهيئتها، هذا الشرط دفع بشركتى إنتل، وميكروسوفت– اعتمادًا على مفهوم مماثل تم إنجازه لأول مرة في آبل ماكنتوش– لتصميم نظام حاسوب شخصى سُمي بنظام 'وصل وشغل'. قبل استحداث هذه التقنية كان لكل بطاقة إدخال، وإخراج طلب مقاطعة بمستوى ثابت وعناوين ثابتة لسجلات الإدخال، والإخراج. على سبيل المثال، كان للوحة المفاتيح المقاطعة 1 وتستخدم عناوين إدخال، وإخراج من 0x60 إلى 0x64، ولمتحكم الأقراص المرنة المقاطعة 6 ويستخدم عناوين إدخال، وإخراج من 0x3F0 إلى 0x3F7، وللطابعة المقاطعة 7 وتستخدم عناوين إدخال، وإخراج من 0x378 إلى 0x37A، وهكذا.

قد تتسبب هذه الآلية في حدوث بعض من المشاكل وخصوصًا عند إضافة بطاقات جديدة للنظام– كبطاقتي الصوت، والاتصال الهاتفي– تستخدم نفس رقم المقاطعة، الأمر الذي سيؤدي إلى ظهور تعارض وعدم اشتغال هذه البطاقات، قد يكون الحل في تضمين مفاتيح مصغرة أو وصلات عبور لكل بطاقة إدخال، وإخراج، ومن ثم إرشاد المستخدم لتحديد مستوى المقاطعة وعناوين أجهزة الإدخال، والإخراج التي لا تتعارض مع غيرها في النظام المستخدم، ولكن هذه الطريقة قد لا تخلو هي الأخرى من المخاطر، لأن أي خطأ يمكن أن يؤدي إلى حالة من الفوضى.

ما تقوم به تقنية 'وصل وشغل' بالمقابل هو جمع المعلومات تلقائيًا حول أجهزة الإدخال، والإخراج ومن ثم تُعيِّن مركزيًا مستويات المقاطعة وعناوين الإدخال، والإخراج وإعلام كل بطاقة بماهية أرقام ومستوى مقاطعتها. هذا العمل يرتبط ارتباطًا وثيقًا باستنهاض الحاسوب والذي سيتم

تناوله في الجزء التالي.

5.4.1 استنهاض الحاسوب

عند الضغط على زر تشغيل الحاسوب، يُنفِّذ الحاسوب عدة خطوات تنتهي عادة بمحث النظام، أو ببرنامج الولوج، أو واجهة المستخدم الرسومية، وذلك تبعا لماهية نظام التشغيل المستخدم. تُعرف هذه الخطوات باستنهاض الحاسوب وهي تتلخص في الآتي: يحتوي كل حاسوب على لوحة إلكترونية تُسمى اللوحة الأم، هذه اللوحة تتضمن جزء معنوي يُطلق عليه اسم نظام الإدخال، والإخراج الأساسي. هذا النظام يحوي برنامج إدخال، إخراج من المستوى الدوني (أولي/ابتدائي)، يشمل إجراءات القراءة من لوحة المفاتيح، والكتابة على الشاشة، وعمليات الإدخال، والإخراج الخاصة بالقرص، بالإضافة إلى مهام أخرى. يُحفظ نظام الإدخال، والإخراج الأساسي في الحواسيب الحديثة في ذاكرة الوصول العشوائي الوميضية من نوع غير المتطايرة، وذلك لكي يمكن تحديثها من قبل نظام التشغيل عند العثور على خلل في هذا النظام.

عندما يَنهض الحاسوب، يشتغل نظام الإدخال، والإخراج الأساسي ويبدأ أولًا بفحص حجم ذاكرة الوصول العشوائي المثبتة على الحاسوب، ومن ثم يختبر ما إذا وُصِّلت كل من لوحة المفاتيح وبقية الأجهزة الأساسية الأخرى بالحاسوب، ومدى استجابتها بشكل صحيح. كل هذا يتم عن طريق فحص ناقل رابط المكونات الملحقة وناقل الرابط السريع للمكونات الملحقة وذلك للكشف عن كافة الأجهزة المتصلة بهما. إذا كانت الأجهزة الحالية المتصلة بالناقل تختلف عن تلك التي كانت متصلة بالحاسوب في آخر عملية استنهاض له فسيتم تهيئة الأجهزة الجديدة.

بعد ذلك يُحدد نظام الإدخال، والإخراج الأساسي الجهاز الرئيسي للاستنهاض من خلال استعراضه لقائمة الأجهزة المخزنة في ذاكرة إلكترونية خاصة والمستخدمة في حفظ خصائص الحاسوب. هذه القائمة يمكن تحديثها من قبل مستخدم الحاسوب عن طريق الولوج إلى برنامج تهيئة نظام الإدخال، والإخراج الأساسي مباشرة عند بدء عملية الاستنهاض وذلك بالضغط على أزرار محددة في لوحة المفاتيح.

ينطلق المسار الافتراضي لعملية الاستنهاض من مشغل وحدة الأقراص المدمجة (أو في بعض الأحيان من الناقل التسلسلي العام) بفرضية وجودهما. في حالة فشل هذه العملية، يبدأ

مسار الاستنهاض من القرص الصلب، في جميع الأحوال يُقْرأ القطاع الأول من جهاز الاستنهاض إلى الذاكرة ومن ثم يُنفَّذ. يحتوي هذا القطاع على برنامج يفحص في العادة جدول التجزئة الواقع في نهاية قطاع الاستنهاض وذلك لتحديد الجزء النشط، بعد ذلك يُقرأ مُحمل الاستنهاض الثانوي من ذلك الجزء والذي سيقرأ نظام التشغيل من الجزء النشط ومن ثم سيبدأ في تشغيله.

يستلم نظام التشغيل مهام تشغيل الحاسوب ويستعلم نظام الإدخال، والإخراج الأساسي للحصول على معلومات تهيئة الحاسوب. بعدها يتحقق نظام التشغيل مما إذا كان لدى كل جهاز متصل بالحاسوب برنامج تشغيل خاص به يُعرف بمشغل الجهاز. في حالة عدم توفر هذا البرنامج، يطلب نظام التشغيل من المستخدم إدراج القرص الخاص بمشغل الجهاز (المزود من قبل الشركة المصنعة للجهاز) أو تحميله من الشبكة المعلوماتية، وبمجرد توفر كافة مشغلات الأجهزة، يحُمِّلها نظام التشغيل جميعا في نمط النواة ومن ثم يُهيِّئ الجداول الخاصة به، ويُنشئ أي عملية خلفية يحتاجها، ويُنهي هذه المرحلة بإخراج شاشة الولوج إلى النظام أو واجهة المستخدم الرسومية، كما هو موضح في الشكل 1. 11.

الشكل 1. 11: شاشة الولوج إلى النظام: (اليمين) محث نظام 'إم إس دوس'– (اليسار) واجه نظام 'ويندوز'.

5.1 أنواع نظم التشغيل

أنتجت سلسلة التطورات التي مرت بها نظم التشغيل خلال أكثر من نصف قرن عدة نمادج لنظم التشغيل، بعض منها غير معروف بشكل واسع، وبالتالي فإن هذا القسم يستعرض بعض من هذه الأنواع بنوع من الإيجاز.

1.5.1 نظام تشغيل الدفعة

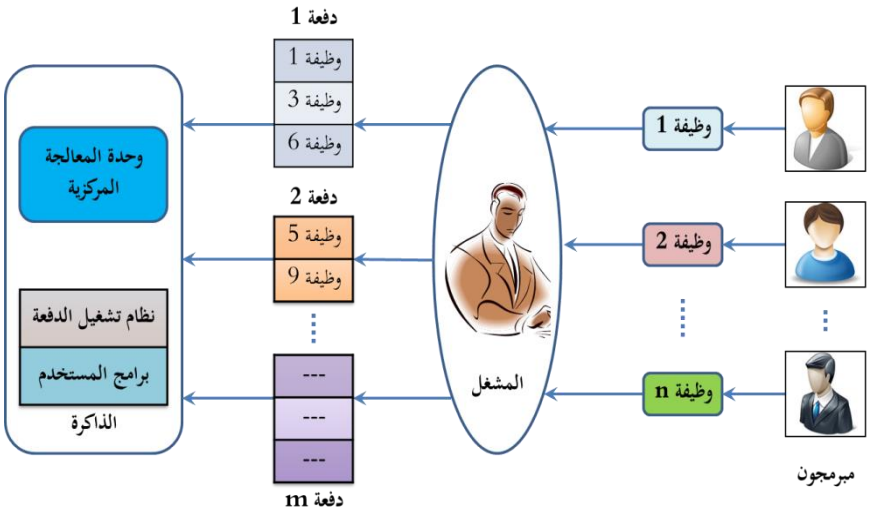
في فترة ظهور الجيل الثاني من الحواسيب المتمثل في الحواسيب المركزية والتى هى بحجم الغرف، كانت مسؤولية المبرمج تتمثل في إعداد البرنامج أو البيانات ليتم معالجتها وتثبيتها على البطاقات والتي تُسلم في نهاية المطاف إلى مشغل الحاسوب داخل الغرفة. في وقت لاحق، عندما ينتهي الحاسوب من مهمته، يسحب المشغل النتائج ومن ثم يُسلِّمها إلى المبرمج. تتسبب هذه الآلية في إهدار وقت كبير من زمن الحاسوب، نتيجة انشغال المشغل باستلام البطاقات وتسليم النتائج.

لاستغلال وقت الحاسوب بشكل أفضل، ظهر مفهوم نظام تشغيل الدفعة والذي تتمثل مهمته الرئيسية في نقل عنصر التحكم تلقائيًا من وظيفة إلى أخرى بعد فرز وتجميع مجموعة من الوظائف المتشابهة ذات الاحتياجات المماثلة من قبل المشغل في دُفعات. بعدها يُرسل المشغل هذه الدُفعات إلى وحدة المعالجة لتُعالجها تلقائيًا من خلال قراءة الوظيفة التالية من الشريط والبدأ في تشغيلها وهكذا إلى أن تنتهي منها جميعًا، كما هو موضح في الشكل 1. 12. هذا النوع من نظام التشغيل لا يتفاعل مع المستخدم مباشرة ويتطلب في العموم تقديم البرامج، والبيانات، وأوامر النظام المناسبة معًا في شكل وظيفة، لكي تُعالج على هيئة دُفعات. في كل مرة ينتهي فيها نظام التشغيل من معالجة أي دفعة تُنسخ النتائج على شريط ممغنط، لكي يُسلَّم فيما بعد إلى المبرمجين بعد الانتهاء من جميع الدفعات.

بالتالي فإن هذا النوع من النظم موجه بشكل كبير تجاه معالجة العديد من الوظائف في آن واحد، والتي في معظمها بحاجة إلى كميات هائلة من بيانات الإدخال، والإخراج، كتجهيز الطلبات في شركة تأمين، أو الإبلاغ عن المبيعات لسلسلة من المتاجر، أو معالجة أعداد كبيرة من الطلبات الصغيرة التي تُطلب في الثانية الواحدة، مثل معالجة الشيكات في مصرف، أو حجز تذاكر السفر في شركة طيران.

من مزايا نظام تشغيل الدفعة أنه يُوفِّر وقت الحاسوب الذي أُهدر مسبقًا، كما أنه يُسهِّل إدارة الأعمال المتكررة بشكل كبير حيث يمكن لعدة مستخدمين مشاركة أنظمة الدُفعات. إلَّا إنَّ من عيوبه ضرورة تمتع مشغلو الحاسوب بالخبرة الكافية للتعامل مع مثل هذه الأنظمة. إضافة إلى

ذلك تُواجه هذا النظام صعوبة تصحيح الأخطاء في حالة فشل أي وظيفة، الأمر الذي يتسبب في جعل الوظائف الأخرى تنتظر لفترات غير معروفة. كما أنه إذا طلبت الوظيفة من المستخدم إدخال البيانات في أثناء زمن التشغيل، فيجب عليه الانتظار حتى تُنفذ الوظائف الأخرى للدفعة وهو ما سيزيد أيضًا من زمن التنفيذ الكلي. نظرًا لأن جميع وظائف الدفعة الواحدة يتم تنفيذها بالتتابع، فإنه من غير الممكن في هذا النظام تنفيذ الوظائف تبعًا لأولويات محددة إذا ما تَعيَّن تنفيذ أي وظيفة على أساس عاجل. أخيرًا، وليس آخرًا، إذا تتطلبت إحدى وظائف الدفعة بعض من عمليات الإدخال، والإخراج، فيجب على وحدة المعالجة الانتظار إلى أن تنتهي هذه العملية وهو ما سيجعل وحدة المعالجة في حالة خمول وذلك للبطء الشديد الناتج عن التعامل مع أجهزة الإدخال، والإخراج. علمًا أن هذه الوحدة لا يمكنها في أثناء فترة الخمول تنفيذ أي وظيفة أخرى.



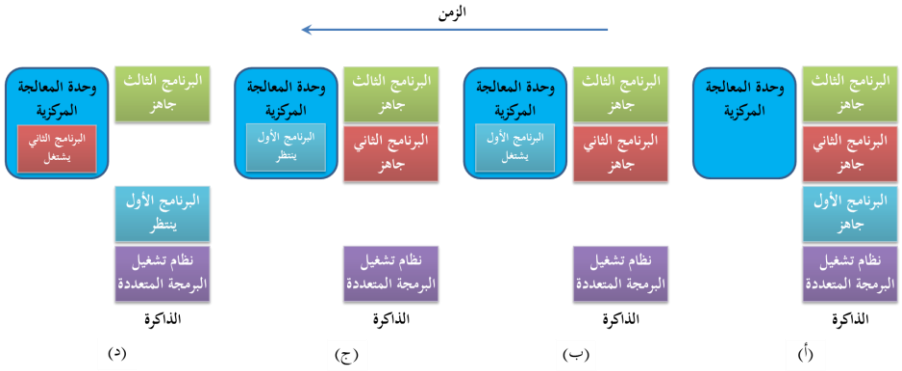
الشكل 1. 12: وصف لآلية نظام تشغيل الدفعة.

2.5.1 نظام تشغيل البرمجة المتعددة

لتحسين استغلالية وحدة المعالجة المركزية والذاكرة الرئيسية أي الاستفادة منهما بشكل أمثل، ظهر مفهوم نظام تشغيل البرمجة المتعددة، والذي قد يُعرف أيضا بنظام تشغيل الحواسيب الشخصية لشيوع استخدامه في هذا النوع من الأجهزة. يسمح هذا النوع من النظم بتشغيل عدة

برامج على نفس الحاسوب، بالتالي يُوفِّر دعم جيد للحواسيب ذات المستخدم الواحد التي تُستخدم على نطاق واسع في مجال برامج معالجة النصوص، وجداول البيانات، والوصول إلى الإنترنت.

يعتمد نظام تشغيل البرمجة المتعددة على التبديل السريع لاستخدام وحدة المعالجة المركزية بين عدة برامج، فعندما يحتاج برنامج قيد التنفيذ إلى التعامل مع وحدات الإدخال، والإخراج سيؤدي هذا الأمر إلى جعل وحدة المعالجة المركزية تنتظر إلى أن تنتهي عملية الإدخال أو الإخراج، هذا يعني هدر وقت وحدة المعالجة المركزية. للاستفادة من وحدة المعالجة يحتفظ نظام التشغيل بعدة برامج جاهزة في الذاكرة في وقت واحد، وبدلًا من الانتظار، يتم تنفيذ برنامج آخر جاهز. لذلك من الممكن للعديد من البرامج مشاركة وقت وحدة المعالجة المركزية، كما هو موضح في الشكل 1. 13.



الشكل 1. 13: مفهوم نظام تشغيل البرمجة المتعددة. أ) الحالة البدائية– ب) البرنامج الأول يشتغل– ج) البرنامج الأول ينتظر في عملية الإدخال أو الإخراج– د) استبدال البرنامج الثاني.

عند مقارنة عمليات الإدخال، والإخراج بالتعليمات التنفيذية الأخرى نجد أنها بطيئة للغاية، لذلك حتى ولو تضمن البرنامج عدد صغير جدًا منها، فإن معظم زمن تنفيذ هذا البرنامج سيُستهلك في القيام بهذه العمليات، بناءً عليه، فإن استغلال وقت انتظار وحدة المعالجة المركزية هذا والسماح لبرنامج آخر باستخدامها خلال ذلك الوقت سيزيد من استغلاليتها وسيُحسن من فاعليتها. من المهم الإشارة هنا إلى أن البرمجة المتعددة لا تعني تنفيذ البرامج في نفس الوقت،

ولكنها تعني أن هناك عددًا من البرامج الجاهزة داخل الذاكرة الرئيسية متاحة لوحدة المعالجة المركزية.

من مزايا نظام تشغيل البرمجة المتعددة أنها تُساهم في استغلال وحدة المعالجة المركزية بشكل عالي، وأنها تُعطي الإيحاء بأن هذه الوحدة مخصص لها عدة برامج في وقت واحد وتعمل بشكل متزامن في الذاكرة الرئيسية، إلّا إنَّه من عيوبها قد تكون أحجام البرامج مختلفة وهو ما يستوجب الحاجة إلى إدارة الذاكرة لاستيعابها فيها. كذلك قد تتواجد عدة برامج جاهزة للتنفيذ، مما يعني الاحتياج إلى جدولة وحدة المعالجة المركزية بشكل دقيق.

من ناحية أخرى، يختلف نظام تشغيل البرمجة المتعددة عن نظام تشغيل المشاركة الزمنية في كون الأول يُركز على الاستغلال الفعَّال لزمن وحدة المعالجة المركزية، من خلال السماح لعدة برامج في أن تتشارك في استخدام وحدة المعالجة المركزية في نفس الوقت، في حين تُركز المشاركة الزمنية على مشاركة مصادر الحاسوب بين العديد من المستخدمين في نفس الوقت، الأمر الذي سيُتيح لعدد كبير منهم فرصة العمل على نظام حاسوب واحد في نفس الوقت.

3.5.1 نظام تشغيل المشاركة الزمنية

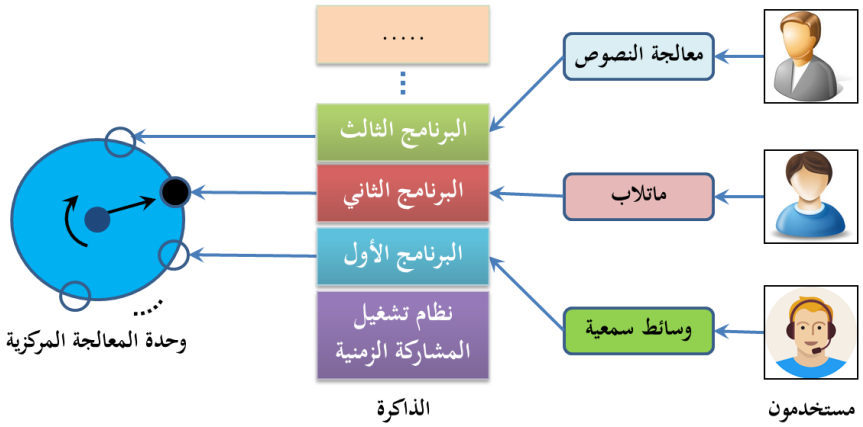
بظهور مفهوم البرمجة المتعددة ظهرة معه مشكلة إهدار وقت المبرمج عندما يكون هناك خطأ في بناء إحدى جمل البرنامج. يتسبب هذا الضياع أيضًا في إهدار زمن المعالج، بالتالي للاستفادة بشكل أمثل من المعالج ظهر نظام تشغيل المشاركة الزمنية الذي يتطلب تشغيلًا متزامنًا للبرامج في نماذج منسقة لوقت المعالج المركزي، تُعطى فيها فترة زمنية قصيرة لكل مستخدم لتنفيذ عملياته الخاصة به.

بناءً عليه تسمح نظم المشاركة الزمنية بتشغيل العديد من الوظائف لعدد كبير من المستخدمين على حاسوب واحد في نفس الوقت عن طريق التبديل بينها بشكل مستمر وسريع للغاية، يُمكِّن هذا المستخدمين من التفاعل مع كل برنامج في أثناء تشغيله دون إدراكهم بأن هناك مشاركة للنظام فيما بينهم، وأن كل مستخدم سيشعر بأنه مستخدم حصري لوحدة المعالجة المركزية، بالرغم من أنه من غير الممكن مشاركة وحدة واحدة بين عدة مستخدمين. كما أنه يسمح للعديد من المستخدمين بمشاركة مصادر الحاسوب في الفواصل الزمنية لعدة برامج في

وقت واحد كالذاكرة، ووحدة المعالجة المركزية، وما إلى ذلك، أو مثل الاستعلام عن قاعدة بيانات كبيرة. يظهر مفهوم نظام تشغيل المشاركة الزمنية في الشكل 1. 14.

كما أشرنا سابقًا، يُمكِّن نظام تشغيل المشاركة الزمنية المستخدم من الاتصال المباشر بينه وبين النظام، بحيث يُعطي المستخدم التعليمات إلى النظام أو إلى البرنامج مباشرة باستخدام جهاز إدخال ومن ثم يستجيب النظام لهذه التعليمات وإخراج النتائج مباشرة على أجهزة الإخراج، لذلك يستخدم هذا النوع من النظم عدة استراتيجيات جدولة لوحدة المعالجة المركزية لغرض تخصيص قدر ضئيل من الزمن لكل مستخدم تُعرف باسم الشريحة الزمنية ومن ثم تبادلها فيما بينهم. يتعرض الفصل الثاني لمناقشة هذه الاستراتيجيات بنوع من التفصيل.

بالرجوع إلى الشكل 1. 14، نلاحظ أن البرنامج النشط هو البرنامج الثاني، بينما يتواجد البرنامج الثالث في حالة انتظار، والبرنامج الأول في حالة الجهوزية، بمجرد اكتمال الشريحة الزمنية الخاصة بالبرنامج النشط، ينتقل عنصر التحكم إلى المستخدم الجاهز التالي أي البرنامج الأول وتستمر هذه العملية بنفس الطريقة، وهكذا. يُعتبر نظام التشغيل المركزي 'أو إس 390' والمنحدر من 'أو إس 360' أحد الأمثلة على هذا النوع من النظم، إلَّا إنَّه في الآونة الأخيرة بدأ استبداله تدريجيا بنسخ من نظام تشغيل 'يونيكس' مثل نظام 'لينكس' وأصبح يُعرف بنظام تعدد المهام.



الشكل 1. 14: وصف لآلية نظام تشغيل المشاركة الزمنية.

تتمثل مزايا نظام تشغيل المشاركة الزمنية في تساوي حظوظ كل مستخدم في حصوله على جزء من وقت المعالج، كما أنه يُقلل من زمن خمول وحدة المعالجة المركزية. بالإضافة إلى ذلك يُوفِّر هذا النظام استخدام تفاعلي لنظام الحاسوب بتكلفة معقولة ويُعد امتدادًا منطقيًا للبرمجة المتعددة التي تعمل في وضع تفاعلي مع وقت استجابة سريع، بالمقابل تتسم هذه الأنظمة بالتعقيد، لأنها تحتاج طرق معقدة لإدارة الذاكرة لغرض تسريع زمن الاستجابة بشكل معقول وتوفير حماية عالية.

4.5.1 نظام تشغيل المعالجات المتعددة

من الطرق الشائعة بشكل متزايد للحصول على حاسوب ذي قدرات عالية هو ربط عدة وحدات معالجة مركزية في نظام واحد، واعتمادًا على طريقة ربط هذه المعالجات وعلى المصادر المشتركة فيما بينها، فإن هذا النوع من الأنظمة يُسمى بأنظمة أجهزة الحواسيب المتوازية، أو الحاسبات المتعددة، أو المعالجات المتعددة والتي تتحد كلها في نظام واحد.

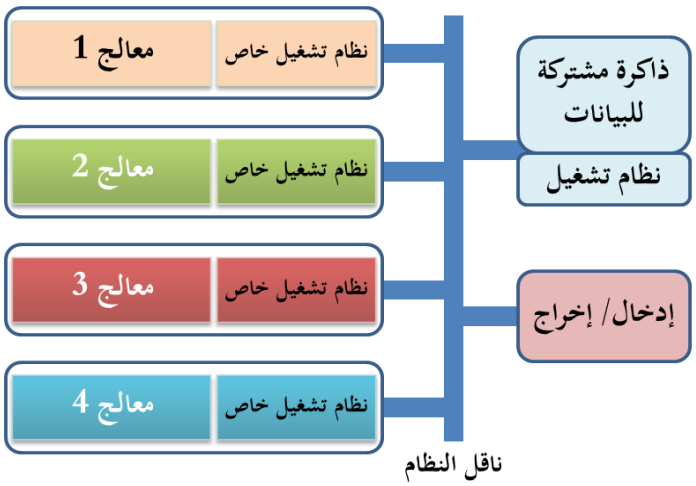
بظهور هذا النوع من الحواسيب ظهر نوع خاص من أنظمة التشغيل يُعرف بنظام تشغيل المعالجات المتعددة، التي يُعتبر وجهًا آخر لنظم تشغيل الخادم ولكن بسمات خاصة للاتصالات والربط، وكذلك التوافق. كما أنه يتعامل أيضًا مع استدعاءات النظام، ويُدير كل من الذاكرة وأجهزة الإدخال، والإخراج، ويُوفِّر كذلك خدمات نظام الملفات كتلك التي يُوفرها النظام الأحادي. يتسم هذا النظام بتعدد مهامه، لأنه يسمح بمعالجة عدة مهام في وقت واحد بواسطة معالجات مختلفة، كما أنه يُستخدم عندما تكون هناك حاجة إلى سرعة عالية جدًا لمعالجة حجم كبير من البيانات كتلك الأنظمة التي تعمل في بيئة مثل التحكم في الأقمار الصناعية، والتنبؤ بالطقس، وما إلى ذلك.

لنظام تشغيل المعالجات المتعددة عدة هيئات مختلفة ومعقدة مقارنة مع نظام تشغيل متعدد البرمجة يشتغل على حاسوب أحادي المعالج وذلك لأن النوع الأول يُنفذ المهام بشكل متزامن لزيادة أداء المعالجات، بالتالي بناءً على آلية التحكم وتنظيم الحاسوب، هناك ثلاثة أنواع أساسية من هذا النظام تتمثل في الآتي:

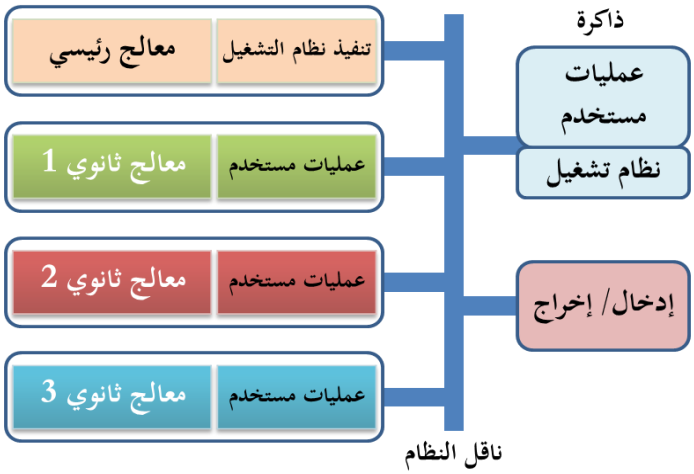
- نظام المُشرِف المنفصل: هذه الهيئة هي في الأساس أبسط طريقة لتنظيم نظام تشغيل

متعدد المعالجات وفيها تُقسم الذاكرة إلى عدة أحجام ثابتة على حسب عدد وحدات المعالجة المركزية المتواجدة في النظام. تتصرف كل وحدة بشكل مستقل وتحتوي على ذاكرتها الخاصة وعلى نظام تشغيل خاص يُدير أجهزة الإدخال، والإخراج المحلية ونظام الملفات والذاكرة. يُعاني هذا النموذج من صعوبة تنفيذ المهمة الواحدة بشكل متوازي، كما أنه غير فعّال، نتيجة الاحتفاظ المتماثل للنسخ الخاصة بالنواة والمشرف وبنية البيانات في كل وحدة. الشكل 1. 15 يوضح هذه البنية.

- نظام المُشرِف الرئيسي والمعالجات الثانوية: يكون في هذا النموذج معالج رئيسي يتحكم في النظام بأكمله، بينما تتصرف بقية المعالجات كوحدات ثانوية. يحتوي المعالج الرئيسي على بنية بيانات واحدة تتعقب العمليات الجاهزة ويتركز عمله على تنفيذ نظام التشغيل الذي يُجدْول المعالجات الثانوية ويتحكم في نشاطها لتنفيذ البرامج التطبيقية. يتميز هذا النظام بسهولة تطويره نسبيًا وبسماحه للتنفيذ المتوازي للمهمة الواحدة، إلَّا إنَّ توسعه محدود، لأن المعالج الرئيسي يُمثل عنق الزجاجة وبالتالي سيفشل في الاستفادة الكاملة من المعالجات الثانوية. يُوضح الشكل 1. 16 هذا النظام.

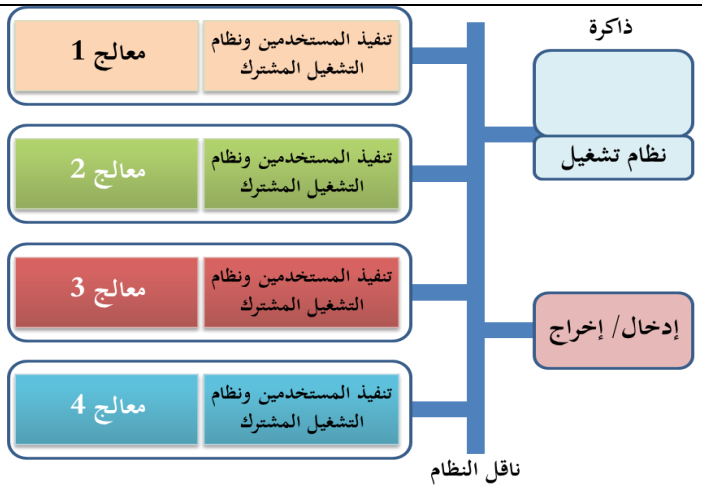


الشكل 1. 15: بنية نظام المشرف المنفصل.



الشكل 1. 16: نظام المشرف الرئيسي والمعالجات الثانوية.

- نظام متماثل المعالجات المتعددة: تتطابق وتتماثل في هذا النموذج جميع وحدات النظام ويتم التعامل معها على قدم المساواة بما في ذلك الوصول إلى جميع مصادره، فهي، أي الوحدات تتعامل مع نفس النسخة من نظام التشغيل والمحتفظ بها في الذاكرة. بمعنى آخر، هناك نسخة واحدة من نواة نظام التشغيل يمكن تنفيذها بواسطة جميع المعالجات بشكل متزامن، وللقيام بذلك يُتحكم في عملية التنفيذ برمتها بشكل دقيق من أجل تبادل الوصول إلى بنية البيانات النادرة والمصادر المتاحة بشكل سليم.
يتطلب هذا الأمر التعامل مع نظام التشغيل بالكامل كجزء حرج يُنفذ فقط من قبل معالج واحد في الوقت الواحد، بحيث يلعب هذا المعالج دورًا خاصًا ويعمل كمشرف لهذا النظام في هذا الوقت بالرغم من وجود معالجات أخرى. ولتنفيذ التطبيقات المختلفة بشكل متوازي يُحتفظ بقائمة انتظار للمعالجات الجاهزة في الذاكرة المشتركة. بعد ذلك يُخصص أول معالج متاح إلى أول عملية جاهزة وهكذا إلى أن تنشغل جميع المعالجات أو تَفرغ قائمة الانتظار. يُوازن هذا النموذج بين العمليات والذاكرة بشكل حيوي وهو موضح في الشكل 1. 17.



الشكل 1. 17: نظام متماثل المعالجات المتعددة.

5.5.1 نظام تشغيل الزمن الحقيقي

يُعتبر نظام تشغيل الزمن الحقيقي نوع آخر من نظم التشغيل، ويتميز هذا النظام بارتباطه بالزمن الحقيقي، الذي يُعتبر معلمة رئيسية فيه، فهو يُستخدم في البيئات التي يجب فيها استقبال عدد كبير من الأحداث الخارجية ومعالجتها في وقت قصير جدًا يسمى زمن الاستجابة، أو عندما تكون هناك متطلبات زمنية صارمة للغاية مثل أنظمة الصواريخ وأنظمة مراقبة الحركة الجوية. على سبيل المثال، في أنظمة التحكم المتعلقة بالعمليات الصناعية، تجمع أجهزة الحواسيب البيانات بشكل آني حول عملية الإنتاج في الزمن الحقيقي لوقوعها ومن ثم استخدامها للسيطرة على الآلات في المصنع. وكمثال على ذلك، عند مرور السيارة فوق خط التجميع في أثناء عملية التصنيع، فإن إجراءات معينة يجب أن تُتَّخذ في لحظات معينة من الزمن وإلَّا سيحدث هناك خلل في السيارة. لذلك ينبغي على نظام تشغيل الزمن الحقيقي أن يُوفر ضمانات مطلقة تضمن حدوث الإجراءات المناسبة في أزمنتها المحددة مسبقًا، وذلك لضمان سير عملية التصنيع على الشكل الصحيح.

لا يتطلب نظام تشغيل الزمن الحقيقي نتائج دقيقة فحسب، بل يشترط أيضًا أن تتوفر النتائج في وقتها المناسب، وإلَّا سيحدث خلل في النظام. بناءً عليه، هناك نوعان من نظم تشغيل الزمن

الحقيقي هما: نظام الوقت الحقيقي الصارم، ونظام الوقت الحقيقي المرن. تكون في النظام الأول قيود الوقت صارمة للغاية لدرجة أن أقصر فترة تأخير ممكنة تكون غير مقبولة، هذا النوع من النظم يمكن العثور عليه في تطبيقات النظم في مجالات التحكم الصناعي، والطيران كأنظمة إنقاذ الأرواح (المظلات الأوتوماتيكية)، ومجالات أخرى مماثلة. أما في النوع الثاني فتكون قيود الوقت أقل صرامة، أي أن هناك مرونة في التحكم في عملية وقوع الأحداث. بمعنى آخر قد لا يتسبب عدم وقوع الحدث في وقته المناسب أي ضرر دائم يؤثر على النتيجة النهائية للعملية، إنما قد يُؤثر في جودتها فقط. أنظمة الصوت أو الوسائط المتعددة الرقمية، والهواتف الرقمية هي خير مثال لتطبيقات هذا النوع من النظم.

أخيرًا، تتداخل فئات نظم تشغيل الحواسيب المحمولة، والأنظمة المدمجة، وأنظمة الوقت الحقيقي فيما بينها إلى حد كبير. في أنظمة الوقت الحقيقي والأنظمة المدمجة تُبرمج وتُحمل البرامج فقط من قبل مصممي النظام ولا يمكن للمستخدمين إضافة برمجيات خاصة بهم، الأمر الذي يُوفر حماية أفضل. من ناحية أخرى، صُممت نظم تشغيل الحواسيب المحمولة والأنظمة المدمجة لغرض المستهلكين، في حين أن أنظمة الوقت الحقيقي هي أكثر استخدامًا في المجالات الصناعية، ومع ذلك، لديهم قدر معين من القواسم المشتركة.

مقارنة بنظام تشغيل المشاركة الزمنية، فإن نظام تشغيل الزمن الحقيقي يميل إلى معالجة تطبيق واحد فقط في وقت واحد وتقديم الخدمات لمعالجة الأحداث الخارجية في الزمن المحدد. في المقابل، يتعامل نظام المشاركة الزمنية مع عدة تطبيقات مختلفة كما أنه يُتيح الاستخدام التفاعلي المتزامن لأنظمة الحواسيب من قبل العديد من المستخدمين عن طريق التبديل بينهم. من الاختلافات الأخرى هو أنه في نظام تشغيل الزمن الحقيقي إذا تأخرت الاستجابة عن موعدها المحدد فتسيترتب عليها نتائج كارثية، بينما في نظام المشاركة الزمنية لن يُسبب تأخر الاستجابة عواقب وخيمة.

6.5.1 نظام تشغيل الشبكات

بظهور شبكات الحاسوب ظهر معها نظام تشغيل الشبكات الذي يشتغل على مزودات الخدمات. فهو نظام تشغيل يُدير الأجهزة والمستخدمين في الشبكة ويتحكم بنشاطاتهم، وذلك

لأنه يتضمن البرامج اللازمة للاتصال وتبادل المعلومات على الشبكة. أيضًا يُوفر هذا النظام القدرة على إدارة المستخدمين والمجموعات، والبيانات، وأمن الشبكة وكذلك كافة الوظائف الأخرى للشبكات كخدمة الطباعة، وخدمة ويب، أو خدمة الملف. كما أن هذه المزودات إمَّا أن تكون حواسيب شخصية كبيرة، أو محطات عمل، أو حواسيب مركزية، تخدم العديد من المستخدمين في آن واحد عبر الشبكة، لكي تسمح لهم بمشاركة المصادر المادية والمعنوية للحواسيب بالاستعانة بنظام تشغيل الشبكات.

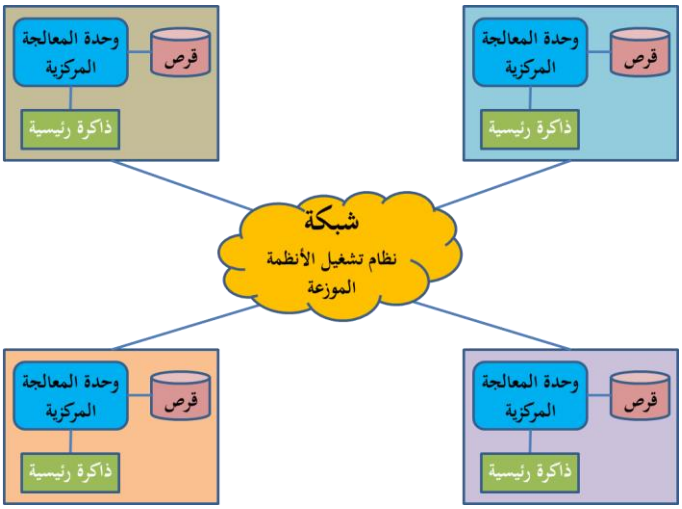
يتمثل أحد الجوانب الأكثر أهمية في نظام تشغيل الشبكات في كونه يُوفِّر خدمات باستمرار. فمن وجهة نظر المستخدم، يمكن أن تكون خدمات نظام تشغيل الشبكات بطيئة لكن من غير المقبول أن تكون غير متوفرة. كما أنه يتميز بوفرة الخدمات الأمنية من خلال الخوادم مع إمكانية دمج التقنيات الجديدة وترقية الأجهزة بسهولة في النظام. إلَّا إنَّ اعتماد المستخدم على موقع مركزي للحصول على معظم خدماته، واشتراط الصيانة والتحديثات الدائمة للنظام يعتبران من أهم التحديات التي تواجه نظام تشغيل الشبكات. تُعتبر أنظمة 'سُولارِس'، و'لينكس' وكذلك 'ويندوز سيرفير' 200x من الأمثلة النموذجية لمثل هذا النوع من أنظمة التشغيل.

7.5.1 نظام تشغيل الأنظمة الموزعة

مع تطور تكنولوجيا الحواسيب تطورت معه تقنيات الشبكات بحيث وفرت إمكانيات تواصل أجهزة حواسيب مستقلة ذاتيًا مع بعضها البعض باستخدام شبكة اتصال مشتركة إمَّا محلية أو عالمية. يمتلك كل نظام مستقل مكونات الحاسوب الاعتيادية من وحدة معالجة، وذاكرة رئيسية، وذاكرة ثانوية، إلخ. إلَّا إنَّها قد تختلف من حيث الحجم والوظيفة والنوع. يوضح الشكل 1. 18 هيكلية هذا النظام.

تطلب هذا التطور الحديث أيضًا وجود نظام تشغيل خاص عُرِف بنظام تشغيل الأنظمة الموزعة. نظام التشغيل هذا سمح للخدم وللزبائن بأن يشتغلوا على آلات مختلفة تتصل ببعضها البعض عن طريق الشبكة، بحيث لا يُدرك المستخدمون مكان تشغيل برامجهم، أو مكان وجود ملفاتهم، لأن ذلك يتم تلقائيًا وبكفاءة بواسطة نظام التشغيل. من الفوائد الرئيسية لهذا النظام هو أنه من الممكن دائمًا لأي مستخدم الوصول إلى الملفات أو البرامج ليس فقط الموجودة فعليًا

على نظامه بل حتى تلك الموجودة على بعض من الأنظمة الأخرى المتصلة بالشبكة. وكمثال على ذلك، تُشغِّل مزودي الإنترنت العديد من أجهزة الخادم، التي تدعم زبائنها، بينما تقوم مواقع الإنترنت باستخدام المزودين لتخزين صفحات الويب والتعامل مع الطلبات الواردة.



الشكل 1. 18: مفهوم نظام تشغيل الأنظمة الموزعة.

من خصائص نظام تشغيل الأنظمة الموزعة هو أنه غير شفاف، أي أنه يُخفي المكونات المادية وكذلك توزيع المصادر عن المستخدمين وبرامج التطبيقات، كما أنه يُوفِّر لهم الوسائل لمشاركتها على نطاق النظام. من ناحية أخرى، يتميز هذا النظام بأن فشل أي جهاز متصل بالشبكة لن يؤثر عليها، لأن جميع الأجهزة مستقلة عن بعضها البعض. كذلك يُساهم تشارك المصادر في تسريع الحسابات بشكل دائم وتقليل الحمل على الجهاز المضيف، أيضًا يسمح نظام تشغيل الأنظمة الموزعة بتطوير وإضافة العديد من الأنظمة بسهولة إلى الشبكة. بالمقابل، فشل الشبكة الرئيسية سيؤدي إلى إيقاف النظام بأكمله، كما أن هذا النوع من النظم ليس متاح بسهولة، لأنه مكلف.

8.5.1 نظام التشغيل المدمج

هناك العديد من التطبيقات التي تُستخدم فيها الحواسيب كأجهزة تحكم مثل أفران الميكروويف، والتلفزيونات الرقمية، وأجهزة الحاسوب في السيارات، والهواتف المحمولة،

وإشارات المرور، وأجهزة الصرَّاف الآلي، وأجهزة التحكم في الطائرة. يُعتبر هذا النوع من الحواسيب الخاصة ذات القدرات المحدودة والتي تحتاج إلى نوع خاص من نظم التشغيل يُعرف باسم نظام التشغيل المدمج. يتميز هذا النظام ببساطته، نتيجة لعدم وجود تعقيدات في البرامج ونتيجة لعدم وجود حاجة للحماية بين التطبيقات، ولتخليه عن العديد من الوظائف التي تُوفرها أنظمة تشغيل سطح المكتب التقليدية والتي قد لا تستخدمها التطبيقات المتخصصة التي تعمل بها. كل هذا ساهم في جعل نظام التشغيل المدمج فعَّال في استخدام المصادر وفي كونه أكثر موثوقية من النظم الأخرى.

تميل الأنظمة المصممة للأجهزة المدمجة إلى أن تكون ذات مُهِمات أكثر تخصصًا اعتمادًا على التطبيقات التي تعمل بها، كما أنها ذات مصادر محدودة إذا ما قورنت بالأنظمة غير المدمجة. بالتالي لا يتم تحميل البرامج فيها إلّا من قبل أشخاص متخصصين وتُحمَّل فقط في أثناء عملية التصنيع، فمثلًا لا يمكنك تحميل تطبيقات جديدة على فرن الميكروويف الخاص بك، وذلك لأن برامجه موجودة في ذاكرة القراءة فقط. أيضًا، تميل معظم أنظمة التشغيل المدمجة إلى أن تكون أنظمة تشغيل تعمل في الزمن الحقيقي ذات السمات التي تم التطرق لها في القسم 5.5.1.

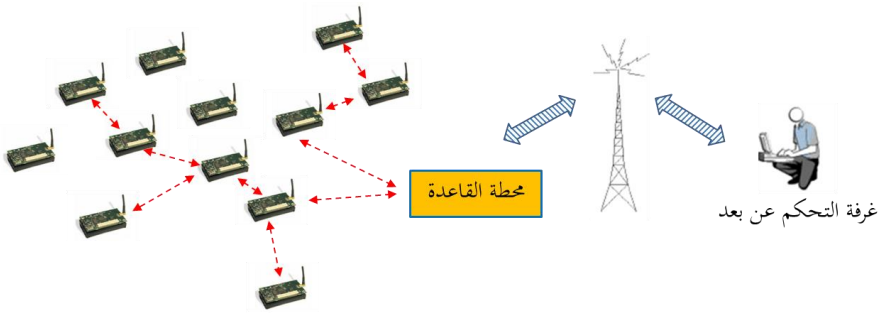
من أهم الفروقات الجوهرية بين نظم التشغيل التقليدية ونظم التشغيل المدمجة هو أنَّ الأولى تُمثل بيئة تفاعلية يتفاعل فيها المستخدم والحاسوب مع بعضهما البعض لأداء مجموعة كبيرة من المهام، بينما تقتصر مهمة الأخيرة على أداء مهمة واحدة فقط والتي غالبًا ما تقوم بها دون تدخل المستخدم بشكل قاطع أو أحيانًا بتدخل قليل منه. كذلك من أهم خصائص أنظمة التشغيل المدمجة هو اعتمادها على الاستخدام المباشر للمقاطعات وعلى ردات الفعل الآنية لها.

9.5.1 نظام تشغيل شبكات التحسس اللاسلكية

شبكات التحسس اللاسلكية عبارة عن عدد كبير من العقد الاستشعارية الصغيرة المتصلة ببعضها لاسلكيًا، كما هو موضح في الشكل 1. 19. هذه العقد هي عبارة عن أجهزة حواسيب صغيرة يتصل بعضها ببعض مع محطة القاعدة عن طريق الاتصالات اللاسلكية. تُستخدم هذه الشبكات في استشعار حماية المباني، وحراسة الحدود الوطنية، والكشف عن الحرائق في

الغابات، وقياس درجة الحرارة وهطول الأمطار، والتنبؤ بالطقس، وكذلك جمع معلومات حول تحركات العدو في ميادين القتال، وغيرها.

تعمل أجهزة الاستشعار هذه بالبطارية ومزودة بجهاز راديو يُستخدم للاتصالات اللاسلكية وهي حواسيب صغيرة لها وحدة معالجة مركزية ذات قدرات محدودة، وذاكرة عشوائية، وذاكرة قرءة فقط، ومتصل بها مجس أو أكثر تُستخدم للتحسس. أيضًا لها نظام تشغيل خاص بها يُعرف باسم نظام تشغيل شبكات التحسس اللاسلكية. يختلف هذا النظام عن نظام التشغيل التقليدي في كونه محدود المصادر، ومُتعذر الوصول إليه. لذلك فهو يعمل تحت ظروف قاسية تتمثل في ضعف إمكانيات المعالج، وحدودية حجم الذاكرة، وكذلك حدودية طاقة التغذية المتوفرة لدى النظام للقيام بمهامه. أيضًا هو عبارة عن نظام تشغيل بسيط قد يستجيب للأحداث الخارجية ليقوم بعملية القياس أو قد يقوم بعمليات قياس دورية مرتبطة بالنبضة الداخلية للجهاز. كما هو الحال مع أنظمة التشغيل المدمجة، يتم تحميل جميع البرامج مقدمًا وقد تُحمَّل من الإنترنت، الأمر الذي يُسهِّل عملية الاستخدام. يُعتبر نظام "تايني أو إس" من نظم التشغيل المعروفة لعقدة الاستشعار السالفة الذكر.



الشكل 1. 19: شبكات التحسس اللاسلكية.

10.5.1 نظام تشغيل الحواسيب المحمولة

استمرار التطور في تقنيات الدوائر المتكاملة أدى إلى ظهور أجهزة الحواسيب المحمولة أو أجهزة المساعد الرقمي الشخصي والتي عبارة عن جهاز حاسوب صغير يمكن حمله في راحة اليد

ويُستخدم بشكل شائع لتخزين المواعيد، وأرقام الهواتف، ودليل العناوين الإلكتروني، والمفكرة. مع ظهور الهاتف الذكي تم استبدال معظم أجهزة المساعد الرقمي الشخصي تقريبًا بالهواتف الذكية، والأجهزة اللوحية.

أغلب هذه الأجهزة تستند في الواقع في مجملها على وحدات المعالجة المركزية ذات 32 خانة مع الوضع المحمي التي تستخدم نظام تشغيل خاص ومتطور. هذا النوع من النظم يُعرف باسم نظام تشغيل الحواسيب المحمولة وهو نظام متطور على نحو متزايد له القدرة على التعامل مع الاتصالات الهاتفية، التصوير الرقمي، وغيرها من المهام المعروفة لدى مستخدمي هذه الأجهزة. منذ حوالي عقد من الزمن أصبح هناك تشابه بين هذه النظم وأنظمة تشغيل الحواسيب الشخصية، من أنظمة التشغيل المعروفة في هذا المجال هما نظاما 'سيمبيان' و'بالم'.

11.5.1 نظام تشغيل البطاقات الذكية

بظهور البطاقات الذكية ظهر معها نظام تشغيلها، والذي يُعتبر أصغر أنواع نظم التشغيل. فهو مصمم لخدمة أجهزة بطاقات الائتمان التي تحتوي على رقاقة وحدة المعالجة المركزية. هذا النوع من النظم عبارة عن أوامر تُحمَّل بشكل دائم في ذاكرة القراءة فقط والخاصة بالبطاقة الذكية. تُستخدم هذه الأوامر بشكل متكرر عند كل استخدام للبطاقة.

نظام تشغيل البطاقات الذكية يجب أن يشتغل تحت ظروف قاسية من حيث ضعف إمكانيات المعالج، وحدودية حجم الذاكرة، وكذلك قلة طاقة التغذية التي قد تتوفر عن طريق عملية الحث، بالتالي هذا الأمر قد يحد بشكل كبير من مهام مثل هذا النوع من النظم، عليه فإن إمكانيات هذا النوع من البطاقات تكون محدودة إمّا في التعامل مع وظيفة واحدة مثل عملية الدفع الإلكتروني، أو التعامل مع أكثر من وظيفة وهو ما يجعل مثل هذه البطاقات عالية التكلفة.

في حالة البطاقات الذكية متعددة التطبيقات فإن نظم تشغيل البطاقات الذكية تسمح بتطوير تطبيقات متعددة تعمل على بطاقة واحدة، بحيث يتم تحميل تطبيقات جافا (برامج صغيرة) إلى البطاقة التي ستُفسَّر من قبل مترجم آلة جافا الظاهرية، الأمر الذي سوف يجعل إدارة المصادر وحمايتها في هذه الحالة أمرًا صعبًا عند وجود تطبيقين أو أكثر في نفس الوقت، وهو ما يتطلب نظم تشغيل بطاقات ذكية أكثر تطورًا.

6.1 هيكلية نظم التشغيل

أنتجت سلسلة التطورات التي مرت بها نظم التشغيل– في حقيقة الأمر– العديد من الهيكليات لهذه النظم، التي قد يفتقد البعض منها لكثير من الهيكلية المنطقية وذلك لعدة أسباب منها:

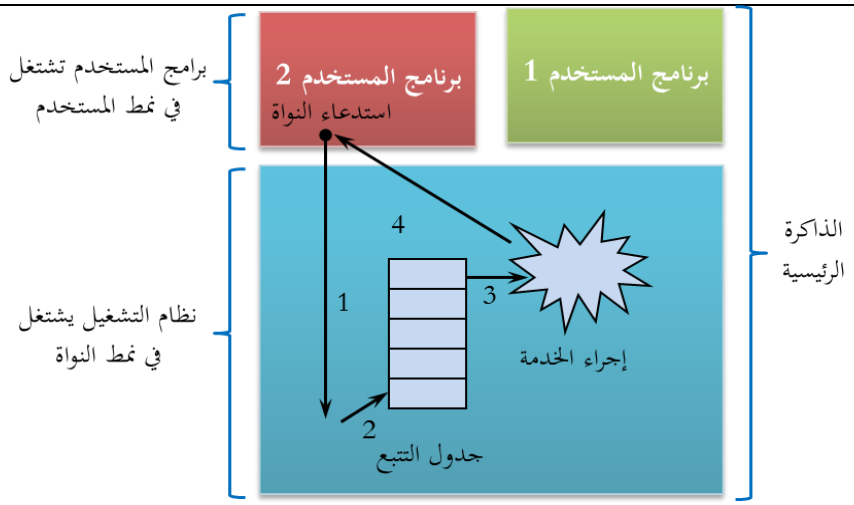
- صُممت معظم نظم التشغيل القديمة في وقت لم تكن فيه مبادئ تراكيب نظم التشغيل أوضح مما هي عليه الآن.
- الرغبة في وجود نظام تشغيل يؤدي كل ما هو مطلوب، مما يستلزم إضافة مهام جديدة كلما دعت الضرورة ودون مراعاة لأي هيكلية.

وفيما يلي سرد لبعض من هيكليات نظم التشغيل.

1.6.1 الهيكلية الأحادية

معظم نظم التشغيل القديمة لا تحتوي على أي هيكلية محددة، فهي عبارة عن برنامج واحد يُنفذ في نمط النواة، هذا يعني أن نظم التشغيل عبارة عن مجموعة من الإجراءات والوظائف تُعرف بالخدمات والتي تُجمع مع بعضها البعض لتُكوِّن برنامجًا ثنائيًا كبيرًا قابلًا للتنفيذ. تتمثل هذه الخدمات في إدارة الذاكرة، وإدارة الملفات، وإدارة العمليات، بالتالي باستطاعة أي إجراء أو وظيفة استدعاء أي إجراء آخر كلما دعت الحاجة إلى ذلك. إلَّا إنَّ إعطاء ألوف الإجراءات إمكانية استدعاء بعضها البعض بدون أي قيود في الغالب ما يؤدي إلى صعوبات في فهم النظام، الأمر الذي يجعل هذا النوع من النظم غير عملي.

لإنتاج البرنامج الهدفي الخاص بنظام التشغيل في هذه الهيكلية، يُترجم أحد البرامج كل الإجراءات المختارة ومن ثم يربطها مع بعض في ملف هدفي واحد باستخدام الرابط وذلك لإنتاج الملف القابل للتنفيذ. هذا يقودنا إلى أنه عند طلب أي خدمة من نظام التشغيل يجب تحميل بعض من المعاملات في سجلات معينة أو في ما يسمي بالمكدس ومن ثم تنفيذ أمر قفز خاص يُعرف باستدعاء النواة. هذا الأمر يجعل الآلة تنتقل من نمط المستخدم إلى نمط النواة الأمر الذي يجعل التحكم ينتقل إلى نظام التشغيل، كما هو موضح بالحدث رقم (1) في الشكل 1. 20.



الشكل 1. 20: الكيفية التي تتم بها عملية الاستدعاء: (1) برنامج المستخدم يقفز إلى نمط النواة، (2) نظام التشغيل يحدد رقم الخدمة، (3) نظام التشغيل يستدعي إجراء الخدمة، (4) ترجيع التحكم إلى نمط المستخدم.

بعد ذلك يبحث نظام التشغيل عن المعاملات ويُحدِّد استدعاء النظام المناسب الذي سوف يُنفذ (الحدث رقم (2) في الشكل 1. 20). بعدها يُحدَّد رقم الخدمة من داخل جدول التتبع، الذي يحتوي على عدد محدد من الإجراءات، لكي يُنفذ (الحدث رقم (3) في الشكل 1. 20). أخيرًا، يُرَجَّع التحكم إلى نمط المستخدم، وذلك كما هو موضح بالحدث رقم (4) في نفس الشكل. بشكل عام يمكن تلخيص هذه البنية في النقاط التالية:

1. يَحقن البرنامج الرئيسي طلب خدمة.
2. تُنفذ مجموعة من الإجراءات الخدمية استدعاءات النظام.
3. تُساعد مجموعة من الإجراءات المساعدة الإجراءات الخدمية.
4. يُرجَّع التحكم إلى البرنامج الرئيسي.

من مزايا هذه البنية أنها نسبيًا سريعة، لأنها تُمثل كيان واحد فقط قابل للتنفيذ، وتحتاج إلى شفرة أقل. بالمقابل ونظرًا لاعتماد هذه البنية على تجميع الإجراءات في برنامج واحد فهي تُعاني من كثرة العلاقات البرمجية والاعتمادية. بالتالي قد يتسبب أي خلل في أي جزء في تعطيل النواة

بالكامل وهو ما يعني أن هذه البنية ذات اعتمادية منخفضة. كما أنه من الصعب إدماج بنية النظم الأحادية في عمارات مختلفة للحاسوب، لأنها تحتاج إلى إعادة كتابة النواة بالكامل لكي يتم دعم هذه العمارات.

2.6.1 هيكلية الطبقات

كتعميم لمفهوم النظم الأحادية، السابقة الذكر، يمكن النظر إلى نظم التشغيل على أساس تسلسل هرمي لمجموعة من الطبقات بحيث تعتمد كل طبقة على الطبقة التي أسفلها. وفي هذا السياق، طُوِّرت أول هيكلية في سنة 1968 وأصبحت تُعرف بهيكلية THE نسبة إلى (Technisch Hogeschool Eindhoven) وهي إحدى الجامعات الهولندية. تعتمد هذه الهيكلية على استخدام طريقة نظام الدفعة. يوضح الشكل 1. 21 الهيكلية الخاصة بهذا النظام والتي تنقسم إلى ست طبقات، فيما يلي توضيح لكل طبقة منها.

الشكل 1. 21: هيكلية نظام التشغيل THE.

الطبقة رقم "0": تعالج هذه الطبقة عملية تخصيص المعالج وعملية الفتح بين العمليات عند حدوث أي مقاطعة، أو عند إنتهاء الوقت الخاص بكل عملية. توجد فوق هذه الطبقة سلسلة من العمليات التتابعية، كل واحدة منها يمكن برمجتها دونما أي اعتبار لحقيقة أن هناك العديد من العمليات تُنفذ من قبل معالج واحد. بمعنى آخر، تُوفر هذه الطبقة الأساس الذى تتم به عملية

البرمجة الدقيقة لوحدة المعالجة المركزية.

الطبقة رقم "1": تتمثل مهمة هذه الطبقة في إدارة الذاكرة. فهي التي تُخصص أماكن للعمليات في الذاكرة الرئيسية وفي أسطوانة التخزين الإضافية بحجم 512 كيلوبايت التي تستوعب أجزاء من العمليات التي لا توجد لها أماكن في الذاكرة الرئيسية في أثناء عملية التنفيذ، أيضًا تُخفي هذه الطبقة عن العمليات نفسها حقيقة ما إذا كانت العمليات موجودة في الذاكرة أو في الأسطوانة الإضافية، وذلك لكونها هي المسؤولة عن استحضارها إلى الذاكرة كلما دعت الحاجة.

الطبقة رقم "2": تُعتبر هذه الطبقة هي المسؤولة عن عمليات الاتصال بين كل من العملية ووحدة طرفية التحكم للمشغل والمتمثل في المستخدم.

الطبقة رقم "3": تُدير هذه الطبقة أجهزة الإدخال، والإخراج، وكذلك عملية التخزين اللحظي لسيل المعلومات المتدفق من وإلى هذه الأجهزة. كما تُسهل عملية التعامل معها على أساس أجهزة مجردة ذات خصائص بسيطة من دون الحاجة إلى الخوض في التعامل مع الخصوصيات المعقدة للأجهزة الحقيقية.

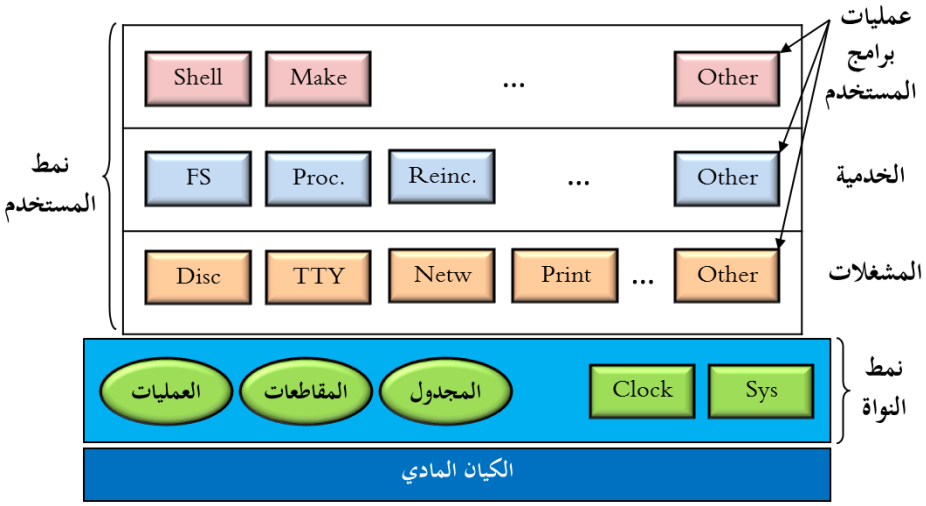
الطبقة رقم "4": تتواجد في هذه الطبقة برامج المستخدم التي لا تخص كل من العمليات، وإدارة كل من الذاكرة، ووحدات الإدخال، والإخراج.

الطبقة رقم "5": يوجد في هذه الطبقة عمليات تشغيل المشغل.

3.6.1 هيكلية النواة الدقيقة

مع بداية السبعينيات ظهر مفهوم يُعرف بالنواة الدقيقة، وهو مبني على مفهوم بنية نظم الطبقات ويعتمد في الأساس على تقليص حجم النواة وجعل عمليات التنفيذ تتم خارجها كلما أمكن ذلك، بمعنى آخر توفير الحد الأدنى من البرامج التي يُمكن أن تُوفر الآليات اللازمة لتنفيذ نظام التشغيل، تتضمن هذه الآليات إدارة فضاء العنونة منخفضة المستوى، وإدارة خيوط العمليات، وكذلك الاتصالات الداخلية بين العمليات، بينما تُنقل بقية خدمات النواة مثل إدارة كل من الشبكات، والملفات، وغيرها إلى نمط المستخدم، وذلك كما هو مبين في مثال هيكلية نظام

تشغيل 'مينيكس 3' والموضحة في الشكل 1. 22. تُمثل في هذا الشكل **Sys** النواة الدقيقة بالإضافة إلى مشغل مولد النبضة، وذلك لارتباطه الوثيق بالمجدول، بينما تشتغل بقية مشغلات الأجهزة على أساس عمليات مستخدم منفصلة.



الشكل 1. 22: بنية نظام 'مينيكس 3' [Tanenbaum & Bos, 2015].

توجد خارج نمط النواة ثلاث طبقات خاصة بالعمليات تشتغل في نمط المستخدم، وهي:

- طبقة المشغلات: وهي تحوي طبقات برامج مشغلات الأجهزة. ولأن هذه البرامج تشتغل في نمط المستخدم، فإنه لا يوجد وصول فعلي مباشر إلى منافذ الإدخال، والإخراج كما إنه لا يمكن إصدار أوامر مباشرة لوحدات الإدخال، والإخراج. وكبديل لذلك، تتم برمجة أي جهاز إدخال أو إخراج عن طريق إخبار المشغل المناسب بالقيم المراد كتابتها، وبمنفذ الإدخال، أو الإخراج المراد الوصول إليه ومن ثم تنفيذ استدعاء النواة لإخبار النواة بالقيام بعملية القراءة، أو الكتابة.
- الطبقة الخدمية: وهي تلي طبقة المشغلات والتي تقوم بمعظم مهام نظم التشغيل كعملية إدارة نظم الملفات عن طريق برامج الملفات الخدمية، وكذلك استحداث، وإدارة، وقتل العمليات من قبل مدير العمليات، ... إلخ. من الخدم المثيرين للاهتمام في هذه البنية هو خادم التناسخ، الذي تتمثل مهمته في معرفة ما إذا كان الخدم الآخرون يعملون بشكل

صحيح. في حالة اكتشاف عطل في خادم– ما– فسيتم استبداله آليًا، وبدون تدخل المستخدم.

- طبقة برامج المستخدم: وهي تتحصل على خدمات نظام التشغيل عن طريق بعث رسائل قصيرة إلى الخادم المناسب ليُلبي الطلب. مثلًا، أي عملية ترغب في القيام بعملية قراءة ستحتاج إلى بعث رسالة إلى أحد خدم الملفات لكي يقوم بعملية القراءة.

ساهم استخدام نموذج النواة الدقيق في عزل الأخطاء وذلك لأن عملية تشغيل مشغلات الأجهزة ونظم الملفات كعمليات مستخدم منفصلة تجعل أي خلل محلي في أي عملية تشغيل تؤثر فقط في مشغل ذلك الجهاز ولا تؤدي إلى توقيف النظام بالكامل. مثلًا، أي خلل في برنامج تشغيل الصوت سوف يُسبب في تشويه الصوت أو توقيفه، لكنه لن يُوقِف الحاسوب. في المقابل في النظم الأحادية، يمكن لخلل داخل برنامج تشغيل الصوت أن يؤدي بسهولة إلى الرجوع إلى عنوان ذاكرة غير صحيح وجلب النظام إلى حالة التوقف الفوري للنظام.

من المزايا الأخرى لنموذج النواة الدقيق تتمثل في كونه أكثر مرونة. حيث يُمكن أن تتواجد فيه عدة استراتيجيات مختلفة وعدة واجهات لبرمجة التطبيقات. أيضًا هناك حاجة أقل إلى إعادة ترجمة النواة عند إجراء تغييرات في نظام التشغيل، كما أنه لا حاجة لإعادة تشغيل الحاسوب عند إجراء تغيير خارج النواة. مثل هذه المزايا تجعل بنية النواة الدقيقة سهلة الاختبار والصيانة، لأن البرمجيات المحملة لا تتطلب إعادة تحميل النواة بالكامل، أخيرًا، هذه البنية يمكن تكييفها بسهولة للعمل تحت بيئة الأنظمة الموزعة.

بالمقابل تُعتبر إدارة هذه البنية عملية معقدة جدًا وأن أي خلل في الاتصالات الداخلية للعمليات قد يتسبب في تعطيل النظام بالكامل.

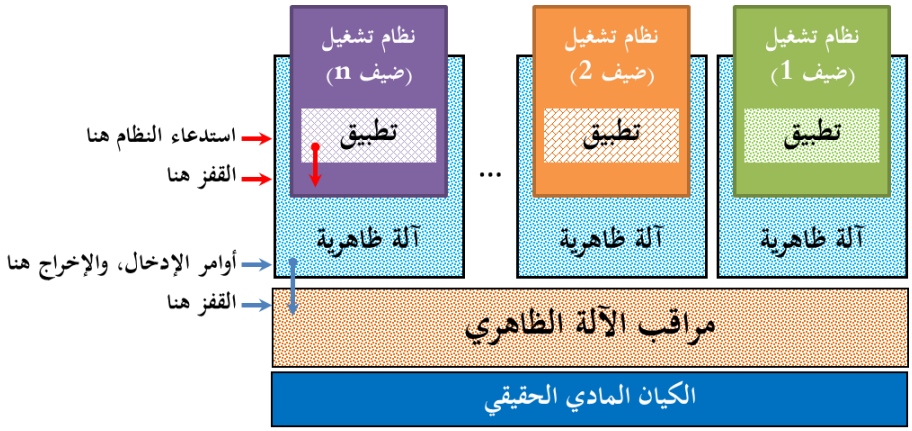
4.6.1 هيكلية الآلات الظاهرية

مع ظهور أجهزة الحواسيب المركزية باهظة الثمن الخاصة بالمؤسسات الكبيرة ظهرت معها الحاجة إلى تشغيل أنظمة تشغيل مختلفة على نفس الجهاز وفي نفس الوقت، وذلك لوجود بعض من التطبيقات تشتغل على نظام تشغيل محدد، والبعض الآخر على أنظمة تشغيل أخرى. طرحت شركة IBM مفهوم مراقب الآلة الظاهري وهو عبارة عن برنامج حاسوبي يُمثل قلب هذا النظام

ويقوم بإنشاء، وتشغيل، وإدارة الآلات الظاهرية، والتحكم فيها في بيئة تشغيل افتراضية على جهاز مضيف حقيقي.

كما هو موضح في الشكل 1. 23 وعلى وجه التحديد، يقع مراقب الآلة الظاهري بين واحد أو أكثر من أنظمة التشغيل، والكيان المادي، ويُعطي الوهم لكل نظام تشغيل بأنه يتفاعل مع هذا الكيان، ويتحكم فيه. إلَّا إنَّه خلف الكواليس من يتحكم فعليًا فيه هو هذا المراقب. بمعنى آخر يعمل مراقب الآلة الظاهري كنظام تشغيل لأنظمة التشغيل المتعددة، أي الضَيفة ويُعطي الإيحاء للتطبيقات بأن لكل واحد منهم وحدة معالجة مركزية خاصة به، وذاكرة افتراضية كبيرة، بينما في واقع الأمر يتم سرًا التبديل بين التطبيقات ومشاركة الذاكرة.

في نظم الآلات الظاهرية، يُسمى جهاز الحاسوب الذي يقوم بمراقبة وتشغيل الآلات الظاهرية بالآلة المستضيفة، بينما تُدعى الآلات الظاهرية بالآلات الضيفة، والتي تُشغِّل كل منها بدورها نظام تشغيل مستقل يُدعى النظام الضيف. هذه الآلات في الحقيقة عبارة عن نسخ محاكية للكيان المادي متضمنة نمطي المستخدم، والنواة، ومقاطعات الإدخال، والإخراج، وكذلك كل ما تمتلكه الآلة الحقيقية، الأمر الذي يُتيح إمكانية تشغيل عدة أنظمة تشغيلة مختلفة. على سبيل المثال، بإمكان تشغيل نظام 'أو إس 360' الخاص بالمعالجة بالدفعة على بعض هذه الآلات، وعلى البعض الآخر تشغيل تطبيقات 'ويندوز' على نظام 'آبل ماكنتوش' أو نظام 'لينكس'.



الشكل 1. 23: بنية نظام الآلات الظاهرية.

بالمثل، باستخدام نظم الآلات الظاهرية يستطيع خادم شبكة تشغيل تطبيقات مختلفة كُتِبت لأنظمة تشغيل مختلفة، الأمر الذي سيُقلل من التكلفة نتيجة لشراء عدد أقل من الخوادم. يتضمن هذا الأمر كذلك إمكانية توفير تكاليف الأجهزة، بالإضافة إلى تكاليف الصيانة، والدعم المستمر.

كما نلاحظ في الشكل 1. 23، عندما يُنفِّذ أي برنامج أمر استدعاء نظام، فإن هذا الاستدعاء سينتقل إلى نظام التشغيل الموجود في الآلة الظاهرية الخاص بها، كما لو كانت آلة حقيقية، بدلًا من الانتقال إلى مراقب الآلة الظاهري، بالتالي فإن الآلة الظاهرية ستقوم بإصدار أمر إدخال، أو إخراج لقراءة أي شيء قد تحتاجه، لكي تُحقق هذا الاستدعاء. عليه فإن أوامر الإدخال، والإخراج هذه سوف تنتقل إلى المراقب لكي يُنفذها كجزء من محاكاته للكيان المادي الحقيقي. للتعرف أكثر على الآلات الظاهرية يمكن الرجوع إلى جاريدو وآخرون (Garrido et al., 2011).

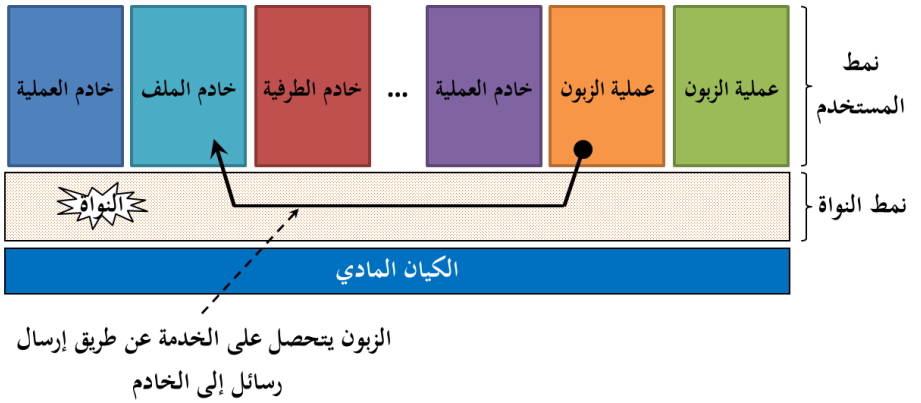
5.6.1 نموذج الخادم والزبون

بالرغم من أن هيكلية الآلات الظاهرية تعتبر من الهيكليات المتطورة التي نُقل فيها أجزاء كبيرة من نظم التشغيل إلى الطبقات العليا منها، إلَّا إنَّها مازالت تُعتبر من البرامج المعقدة، والسبب الرئيسي يرجع إلى أن عملية محاكاة عدد من الآلات الظاهرية ليست بالأمر الهين، هذا أدى إلى ظهور العديد من المحاولات في أنظمة التشغيل الحديثة من أجل تطبيق فكرة نقل أجزاء من التشغيل من الطبقات السفلى إلى الطبقات العليا كلما أمكن ذلك وترك أقل جزء في النواة.

التوجه في النظم الحديثة لعمل كل هذا يتمثل في تنفيذ معظم وظائف نظم التشغيل في نمط المستخدم أو الزبون ومن ثم الاستفادة من مفهوم النواة الدقيقة وإجراء بعض من التعديلات عليه. هذه التعديلات تتمثل في التفرقة ما بين عمليات الخدم الذين يوفرون الخدمات ويُعرفون بالخدم، والزبائن الذين يستفدون منها. هذه التعديلات أدت إلى ظهور ما أصبح يُعرف باسم نموذج الخادم والزبون.

يوضح الشكل 1. 24 بنية الخادم والزبون والتي قُسِّم فيها نظام التشغيل إلى مجموعة من الخدم والزبائن مثل: خادم الملف، وخادم العملية، وكذلك خادم الطرفية بالإضافة إلى عمليات الزبائن. حيث يُوفر كل خادم من هؤلاء الخدم عددًا من الوظائف، أو الخدمات، في حين تتركز

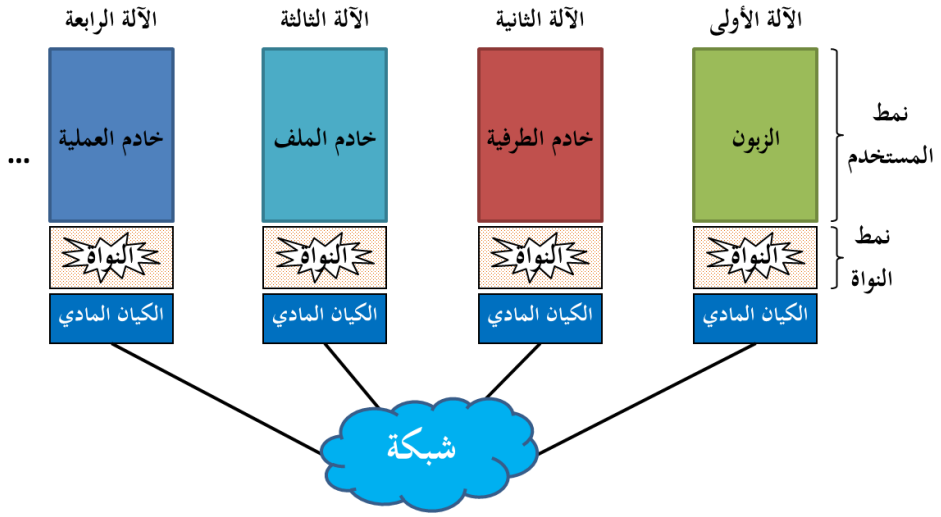
مهمة النواة في هذه البنية في القيام بعمليات الاتصال بين الزبون، والخادم، بالتالي لطلب أي خدمة مثل: قراءة مقطع من ملف يُرسل الزبون طلبه إلى خادم العملية المناسب الذي يتحقق بدوره من صحته، ومن ثم إنجازه وإرسال الرد إلى الزبون.



الشكل 1. 24: نموذج الخادم والزبون.

من مزايا هذا النموذج إمكانية استخدام هذه الهيكلية في الأنظمة الموزعة، وذلك من خلال جعل الخدم والزبائن يشتغلون على آلات مختلفة تتصل ببعضها، إمّا عن طريق شبكة محلية أو عالمية، كما هو موضح في الشكل 1. 25. في هذه الحالة إذا حدث اتصال بين الزبون والخادم عن طريق بعث رسالة فإنه ليس من الضروري أن يعرف الزبون ما إذا عُولجت رسالته محليًا على نفس الآلة أو أُرسلت الرسالة من خلال الشبكة إلى خادم آخر في آلة أخرى، الأمر الذي يهمه هنا هو بعث الطلب واستقبال الرد.

في الآونة الأخيرة نلاحظ أن هذا المفهوم أصبح مطبق بشكل واسع. على سبيل المثال، يمكن اعتبار الحواسيب الشخصية على أساس زبائن، بينما الآلات الكبيرة والموجودة في مكان– ما– على أساس خدم. كمثال لذلك قد يطلب حاسوب شخصي تحميل موقع إلكتروني من مزود خدمة– ما– ويقوم الأخير بتوفير هذا الطلب عبر شبكة الإنترنت.



الشكل 1. 25: نموذج الخادم والزبون داخل الأنظمة الموزعة.

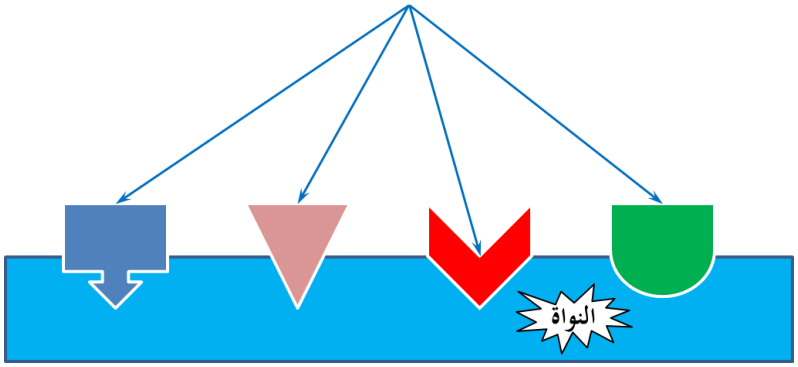
6.6.1 الوحدات القابلة للتحميل

في أنظمة التشغيل القديمة، عندما تكون هناك حاجة إلى إضافة وظائف جديدة إلى نظام التشغيل، فإنه يُضاف شفرات هذه الوظائف إلى نواة نظام التشغيل الأساسية. أي إضافة ملفاتها المصدرية إلى الشجرة المصدرية للنواة، بالتالي سيحتاج المستخدمون إلى إعادة ترجمة وتشغيل النواة الأساسية في كل مرة تُلحق فيها وظيفة جديدة. هذا من شأنه أن يؤدي إلى إهدار الوقت ومساحة الذاكرة المخصصة لهذه الوظائف، لأن معظم هذه الأنظمة لن تستخدمها بشكل كبير.

من هنا جاءت فكرة وحدات النواة القابلة للتحميل، فالطرق المعاصرة في تصميم نظم التشغيل– كأنظمة 'يونيكس' مثل 'سولاريس'، و'لينكس'، 'ماك أو إس إكس'، وكذلك 'ويندوز'– تتضمن استخدام وحدات النواة القابلة للتحميل، والتي هي عبارة عن ملفات هدفية تُضاف إلى النواة تفاعليًا (ديناميكيًا). تحتوي هذه الملفات على شفرة تسمح بتوسيع ودعم النواة الأساسية لنظام التشغيل أهمها دعم أي كيان مادي جديد يُلحق بالنظام (مشغلات الأجهزة)، ودعم مشغلات نظام الملف، وكذلك إضافة استدعاءات جديدة للنظام. بحيث تمتلك النواة مجموعة من المكونات الأساسية وترتبط بخدمات إضافية عبر وحدات مستقلة، سواءً أكان ذلك في أثناء استنهاض الحاسوب، أو في أثناء التنفيذ، من دون أن تكون هناك حاجة إلى إعادة ترجمة

النواة نفسها. كما أنه في حالة الاستغناء عن أي وحدة مستقلة، أي الاستغناء عن وظيفتها، يمكن إلغاء تحميلها لتحرير الذاكرة وغيرها من المصادر لا سيما أنها غير مرتبطة بشكل معقد مع النواة الأساسية. يُوضح الشكل 1. 26 فكرة وحدات النواة القابلة للتحميل.

وحدات النواة القابلة للتحميل



الشكل 1. 26: هيئة وحدات النواة القابلة للتحميل.

النتيجة العامة لبنية نظام الوحدات مشابهة لهيكلية النواة الدقيقة، حيث أنه ليس لدى الوحدة الأساسية سوى وظائف أساسية، ومَعْرِفة بكيفية تحميل الوحدات الأخرى والتواصل معها، ولكنه أكثر كفاءة في كون نظام الوحدات لا يحتاج إلى استخدام أسلوب تمرير الرسائل من أجل التواصل، هذه البنية هي أيضًا على غرار بنية نظام الطبقات من حيث أن كل جزء من النواة يتم تعريفه مع واجهات حماية له، إلَّا إنَّ نظام الوحدات أكثر مرونة من نظام الطبقات، وذلك لأن أي وحدة يمكنها استدعاء أي وحدة أخرى.

من الأمثلة الحديثة لبنية الوحدات القابلة للتحميل بنية نظام تشغيل 'سولاريس' الموضحة في الشكل 1. 27 والمصممة على شكل نواة أساسية محاطة بها سبعة أنواع من وحدات النواة القابلة للتحميل، وهي:

1. فئات الجدولة.
2. أنظمة الملفات.
3. استدعاءات النظام القابلة للتحميل.

4. الهيئات القابلة للتنفيذ.
5. وحدات STREAMS[4].
6. متنوعات.
7. مشغلات الجهاز الطرفي والناقل.

كما يُستخدم نظام 'لينكس' وحدات النواة القابلة للتحميل في المقام الأول لدعم مشغلات الأجهزة وأنظمة الملفات.

الشكل 1. 27: وحدات النواة القابلة للتحميل في نظام 'سولاريس'.

[4] وحدة STREAMS هي نموذج برمجة مرن عام لخدمات اتصالات نظام 'يونيكس' يُعرِّف الواجهات البينية القياسية لإدخال، وإخراج البيانات داخل النواة وبينها وبين بقية النظام.

لبنية نظام الوحدات عدة مزايا منها:

- مريحة: إضافة ميزات حديثة مباشرة إلى النواة، لا يتطلب إعادة ترجمة النواة الأساسية.
- كفؤة: لا حاجة لتمرير الرسائل كما هو الحال في بنية النواة الدقيقة.
- مرنة: يمكن لأي وحدة استدعاء أي وحدة أخرى على عكس بنية نظام الطبقات.
- غير مكلفة: الاستغناء عن أي وظيفة يؤدي إلى تحرير الذاكرة، وغيرها من المصادر.

إلَّا إنَّه من عيوب بنية الوحدات القابلة للتحميل هو تفتت النواة. هذا يعني أنه في كل مرة يتم فيها إضافة أو حذف وحدة مستقلة حديثة يحدث تفتت للنواة وهو ما يؤدي إلى ضعف في الأداء.

7.6.1 الأنظمة الهجينة

من الناحية العملية، هناك عدة أنظمة تشغيل تعتمد على الجمع بين هياكل مختلفة لأنظمة التشغيل والتي ينتج عنها أنظمة تُعرف بالأنظمة الهجينة. الغاية والغرض منها هو الاستفادة من مزايا مختلف الأنظمة المكونة لها والجمع بين أفضل الوظائف لها، لكي تعمل على معالجة مشاكل الأداء، والأمان، واحتياجات قابلية الاستخدام، وما إلى ذلك.

مثلًا، تُوفر بنية الطبقات تقسيمًا جيدًا للوظائف، إلَّا إنَّها تُعاني من ارتفاع عدد الطبقات الأمر الذي يؤثر سلبًا على إدارة النظام. بالمقابل تتميز بنية النواة الدقيقة بكفاءة عالية في عزل الوظائف الأساسية داخلها، لكنَّ الوظائف الأخرى الموجودة خارج النواة غير مدمجة بشكل صحيح، بالتالي الجمع بينهما في نظام هجين سوف يُمكنه من الاستفادة من مزاياهما الاثنيْن.

من أمثلة الأنظمة الهجينة كل من نظام 'سولارس' و'لينكس'. وهما يتشاركان في مزايا كل من الأنظمة الأحادية، لأن وجود نظام التشغيل في مساحة عنوان واحدة يُساهم في توفير أداءً فعَّالًا للغاية، ومزايا أنظمة الوحدات القابلة للتحميل، لأنه بالإمكان إضافة وظائف جديدة تفاعليًا إلى النواة. من الأنظمة الهجينة الأخرى نظام 'ويندوز'. فهو يُعتبر إلى حد كبير من الأنظمة الأحادية إلَّا إنَّه يحمل سمات بعض من السلوك النموذجي لأنظمة النواة الدقيقة، كما أنه يدعم أنظمة الوحدات القابلة للتحميل.

خلاصة القول: يسمح نظام التشغيل الهجين لأحد مكوناته بالوفاء بمجموعة من المتطلبات، كتوفير واجهة المستخدم، ومراقبة التطبيقات، ... إلخ، بينما يعالج المكون الآخر بقية المتطلبات، لأنه يتمتع بأداء عالي.

7.1 الأمن والحماية

عندما يتمتع نظام الحاسوب بخاصيتي تعدد المستخدمين والبرمجة المتعددة أي التنفيذ المتزامن لعدة عمليات، فإنه يتحتم على النظام تنظيم الوصول إلى البيانات وحمايتها من الوصول غير المصرح به، الأمر نفسه ينطبق ليس فقط على البيانات وإنما يشمل كذلك استخدام واستغلال أجزاء الذاكرة، ووحدة المعالجة المركزية، وكذلك بقية مصادر النظام. لذلك يجب أن تكون هناك آليات تضمن أن يكون التعامل معهم جميعًا فقط من خلال العمليات التي اكتسبت الإذن المناسب من نظام التشغيل، ومنع المستخدمين غير الموثوق فيهم من مشاركة مساحات أسماء منطقية، ومادية شائعة. فالكيان المادى الخاص بعنونة الذاكرة يعمل على ضمان تنفيذ العملية فقط ضمن فضاء العنونة المخصص لها. بالمثل، صُممت سجلات التحكم في الأجهزة بحيث تكون غير مرئية بالنسبة للمستخدمين، الأمر الذي يضمن حماية للأجهزة الطرفية المختلفة.

لضمان الوصول والاستخدام الآمن للمكونات المادية والمعنوية للحاسوب تُوفر نظم التشغيل تدابير أمنية تُعرف باسم الأمن والحماية واللذان يُستخدما بشكل متبادل في بعض الأحيان بالرغم من أنهما مختلفان إلى حد كبير، يوضح الجدول 1. 2 أهم هذه الاختلافات. تكمن الحماية في حقيقة أنها تتعامل مع التهديدات الداخلية، أي حماية برامج المستخدم وبياناته من تطفل المستخدمين المحليين المصرح لهم باستخدام نفس النظام. يُمكن تفسير الحماية من خلال مثال منظمة لها عدة إدارات يعمل بها العديد من الموظفين. هذه الإدارات المختلفة يمكنها أن تتشارك في المعلومات المشتركة فيما بينها. لكن الأمر سيكون مختلف بالنسبة للمعلومات الحساسة. لذلك، يتمتع هؤلاء الموظفون بحقوق وصول مختلفة ومميزة يمكنهم من خلالها الوصول إلى بيانات هذه المنظمة. عليه فإن مفهوم الحماية يُحتم على المنظمة إتخاد تدابير واضحة وصريحة لاعتراض الانتهاكات الشريرة والمتعمدة لقيود الوصول من قبل المستخدمين.

بالمقابل يتعامل مصطلح الأمن مع تهديدات المعلومات الخارجية في أنظمة الحاسوب، أي العمل على حماية برامج المستخدم وبياناته ومكونات النظام من التطفل من جانب مستخدمين خارجيين غير مصرح لهم باستخدام نفس النظام، ولكن مخولين باستخدام أنظمة أخرى. ولتوضيح مفهوم الأمن يمكن النظر إلى مؤسسة يُسمح فيها بالوصول إلى البيانات من قبل مختلف موظفيها، لكن غير مسموح لأي مستخدم غير عضو فيها أو مستخدم يعمل في مؤسسة أخرى بهذا الوصول. لذلك يجب على هذه المؤسسة أن تعمل بشكل حاسم على توفير بعض من آليات الأمان بحيث لا يمكن لأي مستخدم خارجي الوصول إلى بياناتها.

الجدول 1. 2: أهم الاختلافات بين مفهومي الأمن والحماية.

الأمن	الحماية
يُوفر الوصول إلى النظام للمستخدمين الشرعيين فقط.	تتحكم في الوصول إلى مصادر النظام.
مصطلح عام يُعالج مخاوف أكثر تعقيدًا.	تأتي الحماية تحت الأمن وتُعالج قضايا أقل تعقيدًا.
يصف الشخص المسموح له باستخدام النظام.	تُحدد المستخدم الذي يمكنه الوصول إلى مصدر معين.
ينطوي تحته التهديدات الخارجية للنظام.	تنطوي تحتها التهديدات الداخلية للنظام.
تقوم آلية الأمن بمصادقة وتشفير المستخدم أو العملية لغرض تكامل البيانات.	تستخدم إذن في آلية الحماية.

قد يكون لنظام الحاسوب قدر كافي من الحماية ولكنه لا يزال عرضة للفشل أو الاختراق. كمثال على ذلك سرقة معلومات ترخيص الدخول لمستخدم نظام– ما–. هذا الأمر قد ينتج عنه إمكانية نسخ أو مسح معلومات المستخدم بالرغم من تفعيل آلية حماية الذاكرة والملفات. لذلك ما سبق ذكره يُحتِّم على النظام نوع من الأمن لغرض الدفاع عن النظام من التهديدات الداخلية والخارجية والتي هي عبارة عن برامج ضارة بطبيعتها تؤدي إلى تأثيرات ضارة على النظام. هذه التهديدات تنتشر على نطاق واسع وتشمل الفيروسات، والديدان، وحصان طروادة، ومصيدة الباب، والهجمات الخاصة بحرمان المستخدمين الشرعيين من الخدمة وكذلك لصوص الهويّات. في بعض من النظم الحاسوبية، يُعتبر منع بعض من هذه الهجمات من وظيفة نظام التشغيل، وفي

البعض الآخر تُترك هذه المهمة إلى سياسات أو برمجيات إضافية. ونظرًا للارتفاع المفاجئ في الحوادث الأمنية، تُعتبر الخصائص الأمنية لنظام التشغيل مجال سريع النمو للأبحاث والإنجازات العلمية.

خلاصة القول: يتطلب مفهومَي الحماية والأمن من النظام أن يكون قادرًا على تمييز جميع المستخدمين، لذلك تحافظ أغلب نظم التشغيل على قائمة بأسماء المستخدمين ومعرِّفات المستخدم المرتبطة بها. في نظام تشغيل 'ويندوز'، يُطلق على هذه المعرِّفات معرِّف الأمن وهي عبارة عن معرِّفات رقمية أحادية، أي لكل مستخدم مُعرِّف فريد بحيث كلما وَلج مستخدم إلى النظام، فإن مرحلة المصادقة أو المطابقة ستُحدِّد المعرِّف المناسب لهذا المستخدم، والذي سيتم ارفاقه بجميع عمليات وخيوط المستخدم الخاص به، أما عندما يحتاج المعرِّف إلى أن يكون قابلًا للقراءة من قبل المستخدم، فسيُترجم اسم المستخدم عكسيًا عبر قائمة أسماء المستخدمين.

في بعض الحالات، قد يُميز المعرِّف مجموعة مستخدمين، أي مُعرِّف مجموعة وليس مستخدم واحد بحيث يُمكِّن هذه المجموعة من صلاحيات مشتركة. أخيرًا، قد لا يُكتفى بمُعرِّف المستخدم أو مُعرِّف المجموعة في بعض الأحيان بل يحتاج الأمر إلى منح المستخدم صلاحيات تصاعدية لكسب أذونات إضافية لغرض القيام بنشاطات محددة، كمنحه إذن الوصول إلى جهاز مقيد الاستخدام.

8.1 أنظمة التشغيل المتاحة

مرت نظم التشغيل خلال أكثر من نصف قرن بالعديد من النمادج، من أشهرها استخدامًا عالميًا عائلتي 'يونكس' وميكروسوفت 'ويندوز'، وهي تُمثل عائلتين مختلفتين من النظم الحديثة تم تطويرهما مع مرور السنين. بالتالي يستعرض هذا القسم هاتين العائلتين بنوع من الإيجاز.

1.8.1 نظام 'يونيكس'

في العام 1974 قام كل من دينس ريتشي وكين تومبسون من شركة معامل بيل (AT&T Bell Labs) بتقديم نظام التشغيل 'يونيكس' للعالم، الذي صُمِّم على حواسيب صغيرة وبهدفين أساسيين: صغر حجم برمجيات النظام، وقابلية التنقل، أي يمكن نقله من جهاز إلى

آخر، إلَّا إنَّه مع حلول عام 1980 ازداد عدد مستخدميه وكان أغلبهم من رواد الجامعات ومراكز الأبحاث، من أشهر نسخ هذا النظام: نظام 'يونيكس 5' مُقدم من معامل بيل، ونظام 'بي إس دي يونيكس' مُقدم من جامعة كاليفورنيا في بيركلي، لقد أصبح نظام المشاركة الزمنية 'يونيكس' نظام التشغيل الرئيسي في الحواسيب الصغيرة والكبيرة وهو يُعتبر أول نظام تشغيل كُتب بلغة السي الراقية.

من ضمن عائلة 'يونيكس' نظام 'لينكس'، الذي صُمِّم وأُنجِز من قبل لينوس تورفالس وآخرين في عام 1991، كما قام لينوس بنشر شفرة هذا النظام البرمجية على الشبكة العالمية للمعلومات وطلب من المبرمجين والمطورين المساهمة في تعديلها وتطويرها. فيما بعد أصبح 'لينكس' متوفرًا مجانًا ومن الأنظمة البرمجية التجارية المهمة كما أُستخدم على العديد من الحواسيب الصغيرة والكبيرة.

2.8.1 نظام 'ماك أو إس إكس'

يُعتبر نظام 'ماك أو إس إكس' في حقيقة الأمر من أنظمة التشغيل المتطورة جدًا والذي يستند في الأساس على نظام 'بي إس دي يونيكس'، ويُوفر الدعم للعديد من الأنظمة مثل: 'بوسكس'، و'لينكس'، ونظام واجهات التطبيقات البرمجية، كما قامت شركة آبل (المنتجة لأنظمة ماك) بتضمين العديد من خصائص النظام المجاني 'بي إس دي يونكس 5' مع نظام النواة الدقيقة ماك 3.0.

يدعم هذا النظام العديد من التقنيات المتطورة مثل 'يونيكس'، وجافا، وكذلك عدة برمجيات، ومواقع، وقواعد البيانات المفتوحة المصدر. كما يُقدم نظام 'ماك أو إس إكس' العديد من بيئات التشغيل المدمجة تحت بيئة سطح المكتب، بالإضافة إلى إطار للتطبيقات الشيئية وإجراءات واجهات التطبيقات البرمجية.

3.8.1 نظام ميكروسوفت 'ويندوز'

يُعتبر نظام ميكروسوفت 'ويندوز' من أكثر نظم التشغيل استخدامًا وهو من البرمجيات التى أطلقتها شركة ميكروسوفت. طُوِّر هذا النظام من أجل الحواسيب الشخصية التى تستخدم

معالجات إنتل. كما تشمل عائلة أنظمة 'ويندوز' عدة إصدارات منها 'ويندوز 95'، و'ويندوز 98'، و'ويندوز مي'، وكذلك 'ويندوز سي'، واللاتان يمكن اعتبارهما من أصغر نظم هذه العائلة. النظم الكبيرة والقوية من عائلة أنظمة 'ويندوز' تضم 'ويندوز إن تي'، و'ويندوز 2000'، و'ويندوز إكس بي'، و'ويندوز فيستا'، و'ويندوز 7'، وكذلك 'ويندوز 10'.

تتميز عائلة أنظمة 'ويندوز' الكبيرة بدعمها لنموذج حماية معقد وخصائص إضافية لدعم وظائف الشبكات، وإدارة عالية للذاكرة الظاهرية، وتنفيذ كامل لوظائف واجهات التطبيقات البرمجية ذات 32 خانة (**Win32 API**).

9.1 أنظمة التشغيل ذات 64 خانة

مع تطور التقنية وظهور المعالجات ذات 64 خانة ثنائية وأنظمة الحاسوب بمعمارية 64 خانة ثنائية ظهرت نظم تشغيل مخصصة بخاصية 64 خانة ثنائية، تتحدث الأقسام التالية عنها بنوع من الإيجاز.

1.9.1 نظام ميكروسوفت 'ويندوز 7'

مع ظهور نظام التشغيل 'ويندوز 7' ازداد عدد أجهزة الحاسوب القادرة على استخدام الطبعة ذات 64 خانة، وذلك لإمكانية عنونة أماكن أكثر في الذاكرة (أي زيادة حجمها). فباستطاعة إصدار 32 خانة من 'ويندوز' التعامل فقط مع ما يصل إلى 4 قيقا بايت من عناوين ذاكرة الوصول العشوائي، بينما يستخدم فعليًا فقط حوالي 3 قيقا بايت، لكون أن 1 قيقا محجوزة. في حين أنه يمكن لنظام التشغيل 64 خانة من الناحية النظرية التعامل مع حوالي 17 بليون قيقا بايت من ذاكرة الوصول العشوائي. في الممارسة العملية، فإن الإصدارات الأكثر تكلفة والمتقدمة من 'ويندوز 7' ذات 64 خانة يمكنها التعامل مع ما يصل إلى 192 قيقا بايت.

بطبيعة الحال لازالت أنظمة التشغيل ذات 64 خانة تواجه بعض من التحديات، فإصدارات 64 خانة من 'ويندوز 7' و'فيستا' تحتاج لمشغلات برمجية جديدة تتوافق مع الأجهزة ذات 32 خانة، وعلى الرغم من أن الشركات المصنعة تقوم بتطوير مشغلات برمجية ذات 64 خانة لأجهزتها الطرفية الحديثة، إلَّا إنَّ مستخدمي الطابعات، والماسحات الضوئية، والأجهزة الأخرى

القديمة يواجهون ظروفًا صعبة في محاولة العثور على مشغلات برمجية ذات 64 خانة تتوافق مع أجهزتهم الطرفية القديمة.

2.9.1 نظام 'ماك أو إس إكس'

يُعتبر نظام التشغيل 'ماك أو إس إكس' ليس فقط نظام التشغيل الأكثر تقدمًا في العالم بل وأيضًا آمنًا للغاية، ومتوافق، وسهل الاستخدام. فنظام 'ماك أو إس إكس سنو ليوبارد' يتضمن تكنولوجيات جديدة تُقدم تحسينات فورية مع إعدادات ذكية للمستقبل.

تُوفر أنظمة 64 خانة من 'ماك أو إس إكس' لجميع مستخدميها الأدوات اللازمة للاستفادة من إمكانيات 64 خانة في تسريع عمل كل شيء، بدءًا من التطبيقات اليومية المباشرة وصولًا إلى الحسابات العلمية الأكثر تعقيدًا. على الرغم من أن نظام التشغيل 'ماك أوس إكس' هو بالفعل نظام يتمتع بخاصية 64 خانة وله قدرات عالية من نواح عدة، إلَّا إنَّ نظام 'سنو ليوبارد' قام بخطوة كبيرة متقدمة من خلال إعادة كتابة ما يقرب من جميع تطبيقات النظام لتتوافق مع 64 خانة وتمكين نظام ماك من معالجة كميات هائلة من الذاكرة، الأمر الذي جعله على استعداد تام لدعم التطبيقات المستقبلية.

10.1 موجز الفصل

بدأت رحلة هذا الفصل بالتعرف على ماهية نظم التشغيل من خلال النظر إليها من وجهتي نظر مختلفتين، على أساس آلة موسعة، وعلى أساس مدير للموارد، ففي الحالة الأولى عُرِّفت نظم التشغيل بأنها الكيان الذي زَوَّد المستخدمين بآلة مجردة تكون أكثر ملاءمةً للاستخدام من الآلة الفعلية، بينما في الحالة الثانية عُرِّفت بأنها الكيان الذي يُدير الأجزاء المختلفة للنظام بكفاءة عالية. وانطلاقًا من هاذين التعريفين تعرض الفصل إلى عدة وظائف لهذه النظم منها تسهيل إجراء عمليات الإدخال، والإخراج، والسماح بمشاركة مكونات النظام بين المستخدمين دون أن تكون هناك تضاربات في الاستخدام.

يلي تعريف نظم التشغيل، قدَّم الفصل أيضًا مراحل التطور التي مرت بها هذه النظم تاريخيًا من ظهور الجيل الأول للحواسيب وحتى وقتنا الحالي. بدأت نظم التشغيل في الظهور في الفترة

التي حلت فيها النظم محل المشغل، وانتهاءً إلى أنظمة البرمجة المتعددة الحديثة. طيلة هذه الفترة كان من أبرز النظم التي ظهرت أنظمة الدفعة، وأنظمة البرمجة المتعددة، وأنظمة الحواسيب الشخصية، وكذلك الحواسيب المتنقلة.

ولأن تطور نظم التشغيل مرتبط بتطور المكونات المادية للحاسوب ومُتفاعل معها بشكل كبير، فإن دراسة بعض المفاهيم الأساسية عن هذه المكونات سوف يكون مساعدًا في فهم أنظمة التشغيل نفسها، حيث تشمل هذه المكونات المعالجات، والذواكر الرئيسية والثانوية، ووحدات الإدخال، والإخراج والتي تتواصل مع بعضها البعض عن طريق ناقل النظام. لذلك نجد أن المفاهيم الاساسية التي بُنيت عليها كافة نظم التشغيل هي: العمليات، وإدارة وحدات الإدخال، والإخراج، وحالة الجمود، وإدارة الذاكرة، وإدارة نظم الملفات التي سيتم التعامل مع كل من هذه المفاهيم في الفصول اللاحقة.

كما أن هذا التطور رافقه ظهور عدة أنواع من نظم التشغيل، ناقش منها هذا الفصل أحد عشرة نوعًا. ابتداءً من نظام تشغيل الدفعة، والبرمجة المتعددة ومرورًا بنظام المشاركة الزمنية، والمعالجات المتعددة، وغيرها، ووصولًا إلى نظام تشغيل البطاقات الذكية. تختلف هذه الأنواع باختلاف الهدف من التصميم، وطبيعة التطبيق المطلوب تحقيقه. فمنها، على سبيل المثال، ما يتماشى مع الأنظمة التفاعلية ومنها ما هو مناسب لأنظمة الوقت الحقيقي، وهكذا.

من خلال دراسة تطور نظم التشغيل نجد أنها مرت بعدة هيكليات تتراوح ما بين البسيط والمعقد. أكثر هذه الهيكليات شيوعًا ما يلي: الهيكلية الأحادية، وهيكلية الطبقات، وهيكلية النواة الدقيقة، وهيكلية الآلات الظاهرية، ونظم نموذج الخادم والزبون، والوحدات القابلة للتحميل، وأخيرًا، الهيكليات الهجينة. الغاية والغرض من هذا التطور في الهيكليات هو السعي للاستفادة من مزايا الأنظمة المكونة لها وتحقيق وظائف تعمل على معالجة مشاكل الأداء والأمان، واحتياجات قابلية الاستخدام، وما إلى ذلك.

أخيرًا، وليس آخرًا، تُوفر نظم التشغيل تدابير الأمن والحماية لضمان الوصول والاستخدام الآمن للمكونات المادية والمعنوية للحاسوب. حيث إن مفهوم الأمن يُعالج تهديدات المعلومات الخارجية في أنظمة الحاسوب، في حين تُوفر برامج الحماية التدابير الواضحة، والصريحة

لاعتراض الانتهاكات الشريرة والمتعمدة لقيود الوصول من قبل المستخدمين داخل النظام نفسه.

في نهاية هذا الفصل، قُدِّمت بنوع من الإيجاز أمثلة لبعض من نظم التشغيل الحديثة والمتاحة كنظامي ميكروسوفت 'ويندوز'، و'يونيكس'.

11.1 أسئلة للمراجعة

1. أذكر التعريفين الخاصين بنظم التشغيل، ومن ثم وضح الشروط الخاصة بتطوير هذه النظم.
2. ما الوظائف الرئيسية لنظم التشغيل؟
3. لماذا لم يُستخدم مفهوم المشاركة الزمنية مع حواسيب الجيل الثاني؟
4. هناك العديد من الأهداف وراء تصميم، وبناء نظم التشغيل، على سبيل المثال، الاستفادة المثلى من المصادر، قابلية التطور وغيرها. هات مثالين من أهداف التصميم التي قد يتعارض تحقيقهما مع بعضهما البعض.
5. ما أهم سمات الجيل الثاني لأنظمة التشغيل؟
6. في الفترة ما بين 1965–1980 تطور الحاسوب بحيث أصبح يحوي عدة طرفيات لوحدات الإدخال، والإخراج ساعدت في سرعة الوصول إلى المعلومات، إلَّا إنَّها مازلت أبطأ بكثير من سرعة المعالج، الأمر الذي أدَّى إلى عدم الاستفادة منه بشكل أمثل. ما هو الحل المقترح لمعالجة هذه الإشكالية؟ وما هي المشاكل التي تواجه؟
7. بين مع الشرح أهم المزايا التي تقدمها الأنظمة متعددة المعالجات.
8. بماذا تتميز الأنظمة العنقودية عن الأنظمة متعددة النوى؟
9. اذكر الفرق بين نمط النواة، ونمط المستخدم، ثم اشرح كيف يُساعد وجود هذين النمطين في تصميم نظم التشغيل.
10. وضح أيٍ من الأوامر التالية يجب السماح بها فقط في نمط النواة:
    أ. فصل المقاطعات.
    ب. قراءة الوقت اليومي للساعة.
    ج. ضبط الوقت اليومي للساعة.
    د. تغيير خريطة الذاكرة.

هـ. تغيير مداخل في جدول حالات الجهاز.

و. الوصول إلى أجهزة الإدخال، والإخراج.

11. بين ماذا يعني أمر القفز؟ ثم وضح استخدامه في نظم التشغيل.

12. ما الفرق الرئيسي بين أمر القفز والمقاطعة؟

13. عدد بعض من الفروق بين نظام تشغيل الدفعة، ونظام تشغيل البرمجة المتعددة.

14. يُعتبر نموذج الخادم والزبون من النظم المعروفة في النظم الموزعة، هل من الممكن استخدام هذا النموذج كذلك في نظام الحاسوب المفرد؟

15. ما الفرق بين أنظمة المشاركة الزمنية، وأنظمة البرمجة المتعددة؟

16. ما الفرق بين أنظمة المشاركة الزمنية، وأنظمة الزمن الحقيقي؟

17. ما الفرق بين أنظمة الزمن الحقيقي، وأنظمة التشغيل المدمجة؟

18. في التسلسل الهرمي لهيكلية التخزين المبين في الشكل 1. 9، لماذا تزداد التكلفة، كلما صغر حجم وحدة التخزين؟

19. يُبين الجدول 1. 1 عدة أنواع لناقلات الحاسوب، ما هي أهم الفروقات الجوهرية بين هذه الأنواع؟

20. عدد في نقاط أهم مراحل استنهاض الحاسوب.

21. في القسم 5.1 تعرفنا على عدة أنواع من نظم التشغيل، إعطي مثال لتطبيق لكل نوع من هذه الأنواع.

22. ما الخصائص المميزة لكل نوع من أنواع نظم التشغيل؟

23. قارن بين هيكليات نظم التشغيل المذكورة في القسم 6.1 من حيث المزايا، والعيوب.

24. وضح مع الرسم أهم الخطوات الأساسية لتنفيذ العمليات في إطار الهيكلية الأحادية لنظم التشغيل.

25. حدد في نقاط أهم وظائف مراقب الآلة الظاهري في هيكلية الآلات الظاهرية.

26. للآلات الظاهرية عدة مزايا جعلتها واسعة الانتشار، ولكن لا تخلو هي الأخرى من العيوب. أذكر على الأقل عيب واحد لها.

27. ما المقصود بالوحدات القابلة للتحميل؟ وما فائدتها؟

28. ما الغاية والغرض من الأنظمة الهجينة؟

29. اشرح مفهومي الأمن والحماية في إطار نظم التشغيل.

30. لماذا طُوِّرت أنظمة التشغيل ذات 64 خانة ثنائية؟

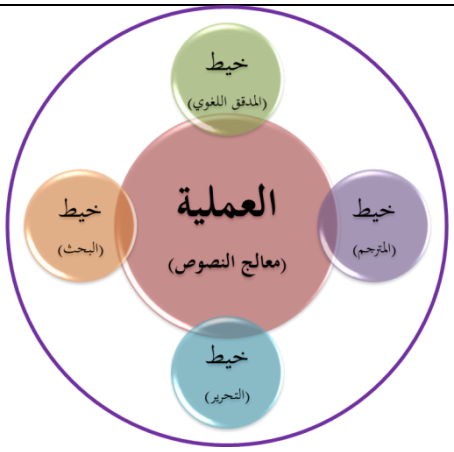
الفصل الثاني العمليات والخيوط

1.2 مدخل إلى العمليات والخيوط

بعدما قدَّم لنا الفصل الأول لمحة عامة عن نظم التشغيل، تأتي بقية الفصول لتشرع في تقديم دراسة مفصلة عن كيفية تصميم وبناء أنظمة التشغيل. وباعتبار العملية هي الركيزة أو اللبنة الأساسية لأي نظام تشغيل، فإنه من المهم جدًا لمصممي أو لطلبة نظم التشغيل أن يتكون لديهم فهم شامل لماهية العملية في وقت مبكر من عملية التصميم أو الدراسة يُساعدهم في فهم معظم إن لم يكن كل التفاصيل المتعلقة بنظم التشغيل.

بالعودة إلى الأجيال الأولى من الحواسيب نجد أنَّها كانت تملك القدرة على تنفيذ عملية واحدة فقط في كل مرة، لذلك كانت العمليات تدخل واحدة تلو الأخرى في قائمة انتظار، لتُنفذ بشكل تسلسلي، ولكن مع تطور علوم الحاسوب أصبح من الشائع قدرة الحواسيب الحديثة على تنفيذ عدة مهام في نفس الوقت، سواءً كان ذلك على التوازي الحقيقي، أو في صورة ما يُعرف بالتوازي الوهمي. فالحاسوب بإمكانه اليوم تحرير ملف نصي في أثناء حرق بيانات على القرص المدمج، ناهيك عن قدرته على إتاحة الفرصة للمستخدمين للتواصل عبر المواقع الإلكترونية أو تصفحها، بينما في الخلفية يُشغِّل الحاسوب العمليات غير المرئية المتعلقة بنظام التشغيل دون أن يشعر المستخدمون بتبادل تنفيذ العمليات أو حتى بفترات خمول البعض منها.

يحدث كل هذا التنفيذ من خلال تعاون العمليات فيما بينها من أجل القيام بالمهام المطلوبة قدر المستطاع على النحو الأمثل. الأمر أصبح متطرفًا إلى درجة أن العملية الواحدة أصبحت قادرة على تنفيذ عدة مهام من خلال تقسيمها إلى عدة خيوط قابلة لأن تُنفذ على التوازي تُعرف بالخيوط المتعددة، كما هو الحال مع برنامج معالج النصوص والذي يضم عدة خيوط– منها على سبيل المثال لا الحصر خيط المدقق اللغوي، وخيط المترجم، وخيط التحرير، وخيط البحث– تعمل في نفس فضاء عنونة العملية الأم، كما هو موضح في الشكل 2. 1. تُدار العمليات والخيوط من قبل نظم التشغيل، وبالأخص تحت إدارة العمليات. لذلك سيُخصص هذا الفصل لدراسة العمليات والخيوط والتعرف على الكيفية التي تُدار بهما من قبل هذه الإدارة، وعلى الاتصالات الداخلية لهما، وعلى المشاكل المصاحبة لمثل هذه الاتصالات، وسيكون ذلك من خلال التعرف أولًا على العمليات، ثم الخيوط، ومن ثم ننتقل إلى بقية مكونات الفصل كالمجدولات وأنواعها.



الشكل 2. 1: برنامج معالج النصوص وبعض من الخيوط الخاصة به.

2.2 العملية

بالنظر اليوم إلى أجهزة الحواسيب الحديثة عند بداية اشتغالها يُلاحظ المرء أنَّ هناك عدة عمليات تبدأ في الاشتغال في الخفاء، والتي غالبًا قد لا يُلاحظها المستخدم. على سبيل المثال، قد تبدأ عمليات البحث عن تحديثات للنظام، وأخرى لبرنامج مكافحة الفيروسات، ناهيك عن القيام بعملية الفحص الدوري لها، وعملية اختبار البريد الإلكتروني الوارد، في حين قد تُشغَّل عمليات مستخدم صريحة كمعالج النصوص، وحرق قرص مدمج، بينما يتصفح المستخدم المواقع الإلكترونية. كل هذه النشاطات تتطلب وجود إدارة عمليات محكمة ونظام متعدد البرمجة يدعم عمليات التنفيذ.

تُنفذ العمليات في نظم الحواسيب القديمة واحدة تلو الأخرى بشكل متتالي، ولكن مع ظهور مفهوم البرمجة المتعددة أصبحت وحدة المعالجة المركزية قادرة على أن تنتقل من تنفيذ برنامج إلى آخر لفترة زمنية قصيرة جدًا، بالتالي بإمكانها تنفيذ عدة برامج في فترة زمنية قدرها واحد ثانية، الأمر الذي يجعل عملية التنفيذ هذه تظهر للعيان بأنها تمت بشكل متوازي، وهو ما يُعرف بالتوازي الوهمي، بينما في حقيقة الأمر هناك جزء وحيد من برنامج واحد يُنفذ في أي لحظة زمنية. إنَّ تتبع وإدارة عدة نشاطات متوازية تُعتبر من أصعب المهام على مستخدم الحاسوب، بالتالي طَوَّر مصمميي نظم التشغيل على مدار عقود من الزمن مفهوم نموذج العمليات المتتابعة

لتسهيل عملية التعامل مع مفهوم التوازي.

1.2.2 نموذج العملية

العملية بشكلها التجريدي هي واحدة من أهم الأساسات التي تُوفرها أنظمة التشغيل، فهي تدعم القدرة على التنفيذ المتزامن لعدة عمليات حتى عندما تكون هناك وحدة معالجة مركزية واحدة، أي بمعنى تُحول وحدة المعالجة المركزية الواحدة إلى مجموعة وحدات ظاهرية للمعالجة المركزية، لذلك وجب تعريف العملية قبل الخوض في التعرف على نموذجها. في واقع الأمر، هناك عدة معاني خاصة بتعريف العملية أكثرها شيوعًا التعريف الذي ينص على أن العملية عبارة عن أي برنامج جاري تنفيذه، أي في أثناء التنفيذ. بعبارة أخرى، عندما تكتب برنامج بأي لغة برمجة، يُترجمه المترجم إلى شفرة ثنائية، بالتالي النص الأصلي، والشفرة الثنائية يُمثلان برنامج، أمَّا عندما تُحول الشفرة الثنائية إلى نسخة قابلة للتنفيذ وتُنفذ، فإن البرنامج يصبح عملية تؤدي جميع المهام المنوطة بالبرنامج.

عندما يُحمل البرنامج في الذاكرة ويصبح عملية نشطة، أي برنامج قيد التنفيذ، يُمكن تقسيمه إلى مجموعة من المكونات، كما هو موضح في الشكل 2. 2، والتي تشمل بعض من الأجزاء التالية:

- جزء البيانات: يتضمن المتغيرات العامة (العالمية) والثابتة.
- جزء المكدس: يحتوي على البيانات المؤقتة، مثل معلمات الدوال، وعناوين الرجوع، والمتغيرات المحلية.
- جزء النص: يُمثل النشاط الحالي الذي تمثله قيمة عدَّاد البرنامج، ومحتويات سجلات المعالج.
- جزء الكومة التخزينية: الذاكرة المخصصة ديناميكيًا للمعالجة في أثناء وقت التنفيذ.

بناءً على ما سبق ذكره يُمكن تعريف العملية بشكلها التجريدي على أساس أنها البرنامج لحظة التشغيل متضمنًا القيم الحالية للمكدس، والبيانات، والكومة التخزينية، وعدَّاد البرنامج، والسجلات، وكذلك كافة المتغيرات المتعلقة به. لذلك فإن جميع برمجيات الحاسوب القابلة للتنفيذ، وأحيانًا بما في ذلك أنظمة التشغيل، تُنظم على هيئة سلسلة من العمليات تُعرف باسم

العمليات التسلسلية وتُمثل نموذج العملية. من الناحية النظرية، كل عملية لديها وحدة معالجة مركزية ظاهرية خاصة بها، ولكن في الواقع وبطبيعة الحال، هناك وحدة معالجة مركزية حقيقية وحيدة تنتقل ذهابًا وإيابًا من عملية إلى أخرى لتُجسد مفهوم البرمجة المتعددة والذي نَوَّه عنه الفصل الأول.

الشكل 2. 2: مكونات العملية قيد التنفيذ والمحملة في الذاكرة.

ستُعامل العمليات في هذا الفصل على فرضية أن هناك وحدة معالجة مركزية وحيدة، إلَّا إنَّها في الواقع قد تكون غير صحيحة وذلك لوجود عدة شرائح حاسوبية حديثة قد تحوي اثنين، أو أكثر من وحدات المعالجة المركزية، ولكن ولدواعي التبسيط سنُركز على وجود وحدة معالجة مركزية وحيدة قادرة على تنفيذ عملية واحدة فقط في أي لحظة زمنية معطاة. أمَّا في حالة توفر أكثر من وحدة فسنفترض أن كل منها ستكون قادرة على تنفيذ عملية واحدة فقط في أي لحظة زمنية.

2.2.2 قالب التحكم في العملية

عند إنشاء أي عملية جديدة في النظام، ينشأ معها تركيبة بيانات تُسمى قالب التحكم في العملية أو مُوَصَّف العملية، والذي يستخدمه نظام التشغيل الداعم للبرمجة المتعددة في تتبع حالة جميع العمليات في أثناء التنفيذ داخل النظام. لذلك يُخزن في هذا القالب كافة البيانات المتعلقة بالعملية الجديدة كحالتها، وعدَّاد البرنامج، ومؤشر المكدس، بالإضافة إلى عدة عناصر أخرى من

البيانات أو الحقول موضحة في الجدول 2. 1. كل هذه المعلومات مطلوبة ومهمة ويجب حفظها وتحديثها داخل القالب الخاص بكل عملية لاستخدامها عند نقل العملية من حالة التشغيل إلى حالة جاهزة أو حالة موقوفة، بحيث يمكن الاستفادة من هذه المعلومات في إعادة تشغيل العملية لاحقًا كما لو أنها لم تتوقف على الإطلاق. بمعنى يُمثل قالب التحكم في العملية العملية نفسها داخل النظام وإذا ما إنتهت العملية فسيُلغى قالبها كذلك. كما أنه يجب الحفاظ على قالب التحكم في العملية في منطقة محمية من الوصول العادي للعملية، في بعض من نظم التشغيل يُوضع هذا القالب في بداية مكدس النواة للعملية.

الجدول 2. 1: بيانات قالب التحكم في العملية.

الحقل	الوصف
مُعرَّف العملية	قيمة صحيحة، وفريدة، تُسند للعملية بواسطة نظام التشغيل، وتُستخدم لتمييز العمليات عن بعضها وخصوصًا في إطار البرمجة المتعددة.
الحالة	تُخزن الحالة الخاصة بالعملية أو وضع العملية مثل، تشتغل، جاهزة، ...إلخ.
مالك العملية	المالك الفعلي للعملية (مستخدم– نظام). بالنسبة إلى عملية المستخدم فإن مالكها هو المستخدم نفسه، أمَّا فيما يخص عملية النظام فإن مالكها هو مشرف النظام أو خادمه الذي أنشأها.
المصادر	المصادر التي يمكن للعملية استغلالها.
فضاء العنونة	مجموعة من المعاملات التي تُشير إلى مواقع الذاكرة المخصصة للعملية.
الصلاحيات	صلاحيات وخصوصية العملية.
سجلات وحدة المعالجة المركزية	مثل، السجل التراكمي، وسجل الأساس، وسجلات للأغراض العامة.
مؤشر الأب	مؤشر للعملية الرئيسية.
قائمة العمليات الأبناء	الأبناء التابعين للعملية.
قائمة الخيوط	قائمة الخيوط التي يمكن للعملية استدعائها.
مؤشر المكدس	مطلوب حفظه عند نقل العملية من حالة إلى أخرى للاحتفاظ بالموضع الحالي لها.

الحقل	الوصف
معلومات حالة الإدخال، والإخراج	مثل، الأجهزة المخصصة للعملية، الملفات المفتوحة، ...إلخ.
معلومات جدولة وحدة المعالجة المركزية	مثل، الأولوية (قد يكون للعمليات المختلفة أولويات مختلفة)، كجدولة العملية الأقصر أولًا.
معلومات إدارة الذاكرة	تشمل معلومات حول نظام إدارة الذاكرة المستخدمة مثل، جدول الصفحة، وجدول المقاطع، وحدود الذاكرة، وما إلى ذلك.
معلومات حساب العملية	مثل، مقدار زمن وحدة المعالجة المركزية المستخدمة لتنفيذ العملية، الحدود الزمنية، مُعرَّف التنفيذ، ...إلخ.
مجموعة العمل	كل عملية لها حد أقصى، وأدنى من الصفحات تُدعى مجموعة العمل.
عدَّاد البرامج	يُخزن العدَّاد الذي يحتوي على عنوان التعليمات التالية التي ستُنفذ للعملية.
قائمة الملفات المفتوحة	تتضمن هذه المعلومات قائمة الملفات المفتوحة بالنسبة للعملية.

3.2.2 إنشاء وإنهاء العمليات

نحن نعلم أنه كلما نُفِّذ برنامج، ستُنشأ عملية جديدة، وستشتغل لفترة زمنية، ومن ثم تنتهي، إمَّا قصرًا أو طوعًا. في الأنظمة البسيطة جدًا كالتحكم في وحدة فرن الميكروويف، من الممكن أن تكون جميع العمليات حاضرة عند عملية التشغيل، ولكن ماذا لو احتجنا إلى إنشاء عمليات جديدة، أو رغبنا في جدولة مهمات مختلفة داخل البرنامج في أثناء اشتغال عمليات أخرى. هل يمكن تحقيق ذلك؟ في الأنظمة ذات الأغراض العامة، قد يحتاج المرء إلى بعض الوسائل لإنشاء وإنهاء العمليات إمَّا تلقائيًا أو كلما دعت الحاجة إلى ذلك في أثناء عملية التشغيل. سيتطرق هذا الجزء إلى بعض الجوانب الخاصة بهاتين العمليتين.

1.3.2.2 إنشاء العمليات

بشكل عام هناك أربعة أحداث رئيسية ينتج عنها إنشاء العمليات وهي موضحة في الشكل 2. 3. تُسمى العملية التي أُنشئت حديثًا بالعملية الفرعية أو العملية الابن، في حين تُسمى العملية التي أنشأتها (أو العملية المستحدثة عند بدء التنفيذ) بالعملية الأصلية أو العملية الأب، والتي بإمكانها توليد عدة عمليات أبناء في أثناء زمن التنفيذ بحيث تتكون الهيكلية الهرمية للعمليات.

في هذه الهيكلية هناك أب وحيد لكل عملية، في حين قد لا يوجد أي ابن للعملية الأب، وقد يكون هناك ابن، أو ابنان، أو أكثر، وذلك حسب متطلبات التنفيذ. بعد عملية الإنشاء يرتبط الأب والابن ببعضهما البعض بطرق محددة، ولهما نفس صورة الذاكرة، ونفس الملفات المفتوحة، ولكن ستكون لكل منهما مساحة عنوان مميزة خاصة، بحيث إذا غيَّرت إحداهما كلمة في مساحة العنوان الخاصة بها، فلن يكون التغيير مرئيًا للعملية الأخرى.

الشكل 2. 3: حالات إنشاء العمليات.

بالرجوع إلى أسباب إنشاء العمليات في الشكل 2. 3 نجد أنَّ إحدها يتمثل في تهيئة النظام عند بداية الاشتغال، في هذه الحالة عادةً ما تتولد عدة عمليات، بعضها يُعرف بالعمليات الأمامية أي العمليات التي تتفاعل مع المستخدمين وتُنفذ طلباتهم، والبعض الآخر تُعرف بالعمليات الخلفية وتكون لديها بعض من الوظائف المحددة، ولا ترتبط بشكل مباشر بالمستخدم، كعمليات قبول البريد الإلكتروني الوارد التي تكون في حالة نوم مستمر وتُوقظ فجأةً فقط عند ورود أي بريد إلكتروني، كذلك الحال بالنسبة لبرنامج مكافحة الفيروسات والذي يَستيقظ عند اكتشاف فيروسات، أو عند تحديثه. مثل هذه العمليات تتواجد عادة بالعشرات في النظم الكبيرة، حيث يمكن استخدام برنامج 'ps'، ومدير المهام لسرد مثل هذا النوع من العمليات الجارية في نظامي 'يونكس' والنوافذ، على التوالي.

من الحالات الأخرى لإنشاء العمليات والموضحة في الشكل 2. 3 هي بدء وظيفة دفعة والمتركزة في الحواسيب المركزية والتي عادة ما تستلم عدد من وظائف الدفعة من المستخدم. عندما يُقرر نظام التشغيل بأن لديه الموارد الكافية لتشغيل وظيفة ما، فسيُنشأ عملية جديدة ويُنفذ الوظيفة التالية في طابور انتظار الدخل.

بالإضافة إلى ما سبق ذكره هناك عمليات جديدة غالبًا ما تُستحدث بواسطة عملية جاري

تنفيذها عند بداية التنفيذ بواسطة استدعاءات النظام وذلك لغرض مساعدتها على القيام بعملها. يكون هذا الأمر مفيد بشكل خاص عندما يُصاغ العمل الذي يتعين القيام به في شكل عدة عمليات تفاعلية ذات صلة ببعضها، ولكن وفي نفس الوقت غير معتمدة على بعضها البعض. على سبيل المثال، إذا كانت هناك كمية كبيرة من البيانات يُراد استحضارها عبر شبكة لطباعتها لاحقًا، فقد يكون من المريح إنشاء عملية لجلب البيانات ووضعها في مخزن لحظي مشترك، في حين تقرأ عملية أخرى البيانات من المخزن، ومن ثم طباعتها بواسطة عملية ثالثة.

تتمثل بعض من الأمثلة الأخرى لإنشاء العمليات وخاصة في الأنظمة التفاعلية في تشغيل مستخدم الحاسوب برنامج محدد سواءً عن طريق كتابة أمر معين في محث النظام، أو بالنقر المزدوج على إيقونة البرنامج نفسه. في كلتا الحالتين قد تتولد نوافذ جديدة تُتيح للمستخدم إمكانية التفاعل مع البرنامج قيد التنفيذ والتحكم فيه حسب متطلباته.

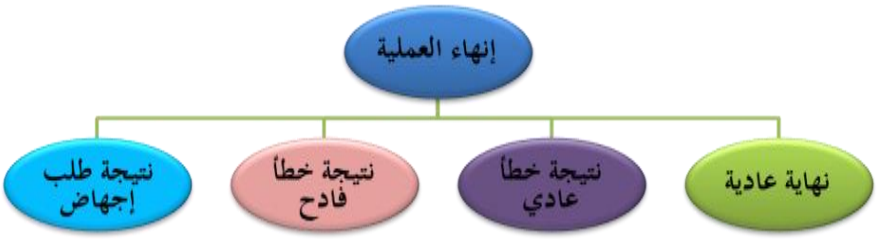
خلاصة الأمر تُنشأ عملية جديدة في جميع الحالات سالفة الذكر بواسطة عملية تُنفذ استدعاء نظام إنشاء عملية، هذه العملية قد تكون عملية مستخدم قيد التنفيذ، أو عملية نظام أُستدعيت بواسطة لوحة المفاتيح أو الفأرة، أو عملية إدارة الدفعة. في أي حالة من هذه الحالات عندما يُقرر نظام التشغيل استحداث عملية جديدة فإنه يقوم بالتالي:

- إسناد مُعرَّف عملية فريد للعملية الجديدة.
- تخصيص مساحة تخزينية لها.
- تهيئة قالب التحكم في العملية.
- ضبط الرابط المناسب لها في قائمة العمليات الجاهزة.
- إنشاء بعض هياكل البيانات الإضافية لها لغرض تقييم أدائها.

2.3.2.2 إنهاء العمليات

القاعدة العامة في إنهاء العمليات هو أنه بعد أن تُنفذ العملية وظيفتها المناطة بها، يجب إنهاؤها، لأنه لا شيء يدوم إلى الأبد ولا حتى العمليات، هذا الإنهاء قد يكون بشكل طوعي، إمَّا بخروج عادي، أو نتيجة لخطأ بسيط، أو قد يكون بشكل لا إرادي، إمَّا بخروج نتيجة خطأ فادح، أو إجهاض من قبل عملية أخرى. يُوضح الشكل 2. 4 باختصار هذه الحالات الأربع، والتي

ستُناقش تاليًا.



الشكل 2. 4: حالات إنهاء العمليات.

بشكل عام، تنتهي العمليات بإنتهاء تنفيذ المهمة الموكلة لها بعد أن تطلب من نظام التشغيل حذفها باستخدام استدعاء نظام الخروج. يُعرف هذا الخروج بالإنهاء الاختياري أو الطوعي والذي عادةً ما يتم بواسط أمر خروج العملية (**Exit Process**) في نظام النوافذ، وأمر الخروج (**Exit**) في نظام 'يونكس'، والذي يُنهي أيضًا أي خيوط مرتبطة بهذه العملية. تُرجِّع العملية في هذه الحالة قيمة الحالة (في الغالب عدد صحيح) إلى العملية الرئيسية (الأب) عبر استدعاء نظام الانتظار. بعد إنتهاء العملية، يُتلف مُوَصِّفها، ويُلغي نظام التشغيل تخصيص كافة مصادر العملية المخصصة لها مثل، الذاكرة الفعلية والظاهرية، والملفات المفتوحة، ومخازن الإدخال، والإخراج اللحظية.

هناك عدة وسائل متاحة في النظم التفاعلية تدعم الإنهاء الطوعي للبرامج كتلك التي تُوفرها برامج مثل معالج النصوص، وبرامج تصفح الإنترنت، وغيرها والمتمثلة في رمز أو عنصر قائمة بحيث يمكن استخدامه من قبل المستخدم عن طريق النقر عليه لإغلاق أي ملفات مفتوحة مؤقتًا، ومن ثم إنهاء البرنامج.

من الأسباب الأخرى لإنهاء العملية هو الإنهاء الاختياري الناتج عن اكتشاف نظام التشغيل أن العملية قد وصلت إلى نقطة لا يمكن أن تستمر منها، كمحاولتها الامتثال لأوامر غير قانونية مثل، الرجوع إلى موقع غير موجود في الذاكرة، أو القسمة على الصفر. في بعض من أنظمة التشغيل (مثل، 'يونكس') قد تُعطى للعملية فرصة إبلاغ نظام التشغيل برغبتها في معالجة بعض من الأخطاء بنفسها، ومن ثم تُقاطع العملية بدلاً من إنهائها عند حدوث أحد هذه الأخطاء. كذلك الحال بالنسبة لنظام 'ويندوز' فعمليات الشاشة التفاعلية لا تقوم عادةً بعملية الخروج عندما

يحدث خطأ بسيط، وبدلًا من ذلك تُولد شاشة تفاعلية جديدة تُعطي المستخدم فرصة أخرى لإعادة المحاولة.

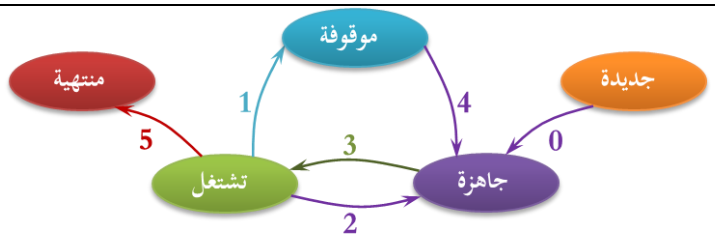
من الأسباب غير العادية لإنهاء العملية هو الخطأ الفادح كمحاولة ترجمة المستخدم لملف غير موجود، والذي سيجعل المترجم يُنهي العملية ببساطة. كذلك الحال بالنسبة لإجهاض العملية الناتج عن تنفيذ أمر استدعاء نظام مناسب مثل، الأمر **kill**، والأمر **TerminateProcess** في نظامي 'يونكس' و'ويندوز'، على التوالي، والذي يصدر من قبل العملية الرئيسية لإخبار نظام التشغيل بإجهاض عملية أخرى. هنا يجب ملاحظة أنه في كلتا الحالتين، يجب أن يكون لدى العملية الرئيسية التخويل اللازم والضروري لإصدار هذا النوع من الأوامر. في هذه الحالة يجوز للعملية الرئيسية (الأب) إنهاء أحد أبنائها لأسباب متنوعة منها:

- لم تعد المهمة المسندة إلى العملية الابن مطلوبة.
- تجاوز العملية الابن لاستخدامها للمصادر المخصصة لها.
- إذا إنتهت العملية الأب، فلن يسمح نظام التشغيل للعملية الابن بمواصلة تنفيذها.
- إذا ما إنتهت العملية إمَّا بشكل طبيعي أو غير طبيعي، فيجب أيضًا إنهاء جميع أبنائها.

تسمى هذه الظاهرة بالإنهاء المتتالي والذي يبدأ عادةً من قبل نظام التشغيل.

4.2.2 حالات العملية

على الرغم من أن كل عملية مستقلة بذاتها عن الأخرى إلَّا إنَّه في الغالب ما تحتاج العمليات إلى التفاعل والتواصل مع بعضها البعض مثلًا، قد تُولد عملية– ما– بعض من النتائج والتي قد تكون مدخلات لعملية أخرى تليها في التنفيذ، الأمر الذي سيجعل العملية الأخيرة تنتظر في هذه النتائج. في بعض الأحيان كذلك تُوقف عملية جاهزة وقادرة على الاشتغال بسبب قرار نظام التشغيل بتخصيص وحدة المعالجة المركزية إلى عملية أخرى. لذلك تمر العملية منذ إنشائها وحتى الإنتهاء من مهمتها بعدة حالات مختلفة، الحد الأدنى لعدد هذه الحالات هو خمس. يُوضح الشكل 2. 5 مخطط هذه الحالات وذلك عندما يكون بإمكان وحدة المعالجة المركزية تنفيذ أكثر من عملية. يُفصل الجدول 2. 2 بالمقابل سلسلة الحالات هذه والتي تعتمد على أحداث معينة.



0 ← بداية عملية جديدة.
1 ← تُوقف العملية انتظارًا لإدخال البيانات.
2 ← يختار المجدول عملية أخرى.
3 ← يُقرر المجدول تشغيل هذه العملية.
4 ← البيانات المراد إدخالها أصبحت جاهزة.
5 ← إنتهاء العملية.

الشكل 2. 5: مخطط حالات العملية.

الجدول 2. 2: حالات العملية.

الحالة	الوصف
جديدة	تُمثل الحالة الأولية عند إنشاء العملية لأول مرة من قبل نظام التشغيل وتحميلها في الذاكرة، وتخصيص قالب التحكم وبعض من المصادر الأخرى لها، بعدها تتغير حالة العملية إلى جاهزة.
تشتغل	الحالة التي تُخصص فيها وحدة المعالجة المركزية للعملية لتُنفذها فعليًا. إذا تواجدت وحدة معالجة مركزية واحدة فقط في النظام، فستكون عملية واحدة في هذه الحالة في أي فترة زمنية، لكن إذا تواجد n من المعالجات في النظام، فيُمكن تشغيل n من العمليات في نفس الوقت.
موقوفة	الحالة التي تكون فيها العملية غير قادرة على الاشتغال بسبب انتظارها لحدوث أحداث خارجية. يمكن أن يكون هناك عدة عمليات في هذه الحالة.
جاهزة	الحالة التي تكون فيها العملية قادرة على الاشتغال وتنتظر في تخصيص المعالج لها من قبل نظام التشغيل، ولكنها موقوفة لحظيًا للسماح لعملية أخرى بالاشتغال. يمكن أن يكون هناك عدة عمليات في هذه الحالة.
منتهية	تُمثل الحالة التي إنتهى فيها نظام التشغيل من تنفيذ العملية، وحذفها من الذاكرة.

من خلال الشكل 2. 5 يمكن ملاحظة أن هناك ستة احتمالات للانتقال ما بين حالات العملية الخمس وذلك على النحو الموضح في الجدول 2. 3. بعض من هذه الانتقالات تحدث

تحت سيطرة مجدول العمليات والذي سنتطرق له لاحقًا في هذا الفصل بنوع من التفصيل.

الجدول 2. 3: احتمالات الانتقال ما بين حالات العملية الخمس.

رقم الانتقال	من	إلى	الوصف
0	جديدة	جاهزة	يحدث هذا الانتقال عند دخول عملية جديدة مرحلة التشغيل، وهي تُمثل الحالات التي نُوقشت في القسم 1.3.2.2.
1	تشتغل	موقوفة	يحدث هذا الانتقال عندما تكتشف نظم التشغيل بأن العملية غير قادرة على مواصلة التنفيذ بسبب حاجتها إلى مدخلات خارجية.
2	تشتغل	جاهزة	يحدث هذا الانتقال بقرار من مجدول العمليات (والذي يُعتبر جزء من نظام التشغيل) عندما يرى أن العملية قد أخذت وقتها الكافي في التنفيذ، وأنَّه آن الأوان لتنفيذ عملية أخرى.
3	جاهزة	تشتغل	يتم هذا الانتقال تحت تحكم المجدول، وهو يحدث عندما يرى المجدول أن كل العمليات الأخرى قد أخذت وقتها في التنفيذ، وأنَّه آن الأوان من جديد لتنفيذ العملية الأولى مرة ثانية.
4	موقوفة	جاهزة	يحدث هذا الانتقال عندما تتوفر المدخلات الخارجية (مثل وصول بعض من المدخلات) والتي تنتظر فيها العملية. عندما لا تكون هناك عملية أخرى قيد التشغيل في تلك اللحظة، سيحدث الانتقال الثالث لحظيًا، والعملية سوف تبدأ مباشرةً في التنفيذ، وإلا فإنه قد تُضطر إلى الانتظار في حالة جاهزة لبعض من الوقت، حتى تتحرر وحدة المعالجة المركزية.
5	تشتغل	منتهية	يحدث هذا الانتقال عند إنهاء مرحلة التشغيل للعملية والمتمثلة في الحالات التي نُوقشت في القسم 2.3.2.2.

بعد ما تعرفنا على مفهوم حالات العملية، أصبح من السهل الآن تخيل ما يجري من أحداث داخل نظام التشغيل بخصوص العمليات. فبعض من هذه العمليات قد يكون قيد التنفيذ، ولكنها ستُوقف نظرًا لاحتياجها لأحدث خارجية، بالتالي ستنتقل إلى حالة موقوفة وستبقى هناك إلى أن تجهز الأحداث المطلوبة لتنتقل على إثرها إلى حالة جاهزة. يأتي الآن دور المجدول في اختيار إحدى العمليات القابعة في الحالة الأخيرة وفقًا لاستراتيجية خوارزمية الجدولة لغرض نقلها إلى

حالة تشتغل. يُناقش القسم التالي آلية تنفيذ العمليات.

5.2.2 تنفيذ العمليات

لكي يُنفذ نموذج العملية بشكل صحيح، يستخدم نظام التشغيل ما يُعرف باسم جدول العملية والمتكون من عدة مداخل تُناظر عدد العمليات الحالية بهذا النموذج. من ناحية منطقية يحوي هذا الجدول جميع قوالب التحكم للعمليات الحالية في النظام، والتي تضم ضمنيًا كل المعلومات المهمة حول حالة كل عملية، والمذكورة في الجدول 2. 1.

الجدول 2. 4: بعض من حقول مداخل جدول العملية النموذجي.

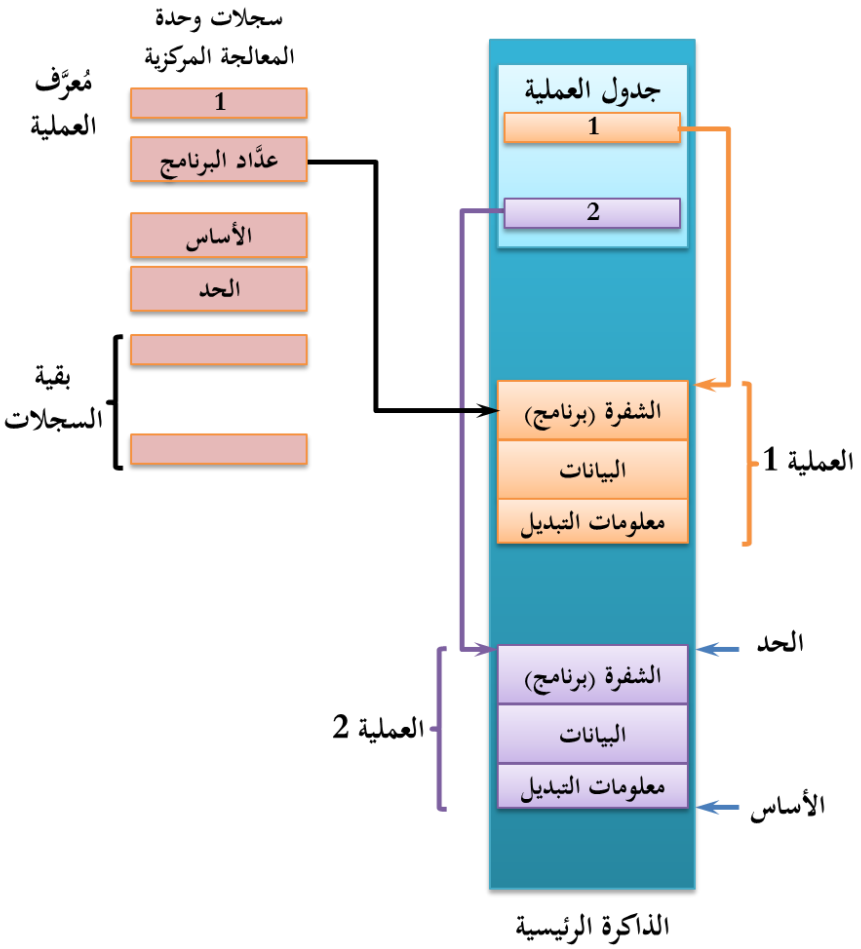
إدارة العملية	إدارة الذاكرة	إدارة الملف
السجلات	مؤشر معلومات مقطع النص	جذر الدليل
عدَّاد البرنامج	مؤشر معلومات مقطع البيانات	دليل العمل
كلمة حالات البرنامج	مؤشر معلومات مقطع المكدس	مُوصَّف الملف
مؤشر المكدس		مُعرَّف المستخدم
حالة العملية		مُعرَّف المجموعة
الأولوية		
معلمات الجدولة		
مُعرَّف العملية		
أب العملية		
مجموعة العملية		
الإشارات		
زمن بداية العملية		
الزمن المستخدم من وحدة المعالجة المركزية		
زمن وحدة المعالجة الخاص بالأبناء		
زمن المنبه التالي		

يُوضح الجدول 2. 4 بعض من الحقول الرئيسية لمداخل جدول العملية في نظام نموذجي. حيث يتعلق العمود الأول بإدارة العملية، والعمود الثاني بإدارة الذاكرة، أمَّا العمود الثالث فهو يتعلق بإدارة الملفات. تجدر الإشارة هنا إلى أنَّ الحقول الموضحة في هذا المثال قد تختلف من نظام إلى آخر، ولكن ما هو مذكور يُمثل فقط فكرة عامة عن أنواع المعلومات المستخدمة.

بعد إعطاء فكرة عامة عن جدول العملية، من الممكن التبحر قليلًا في شرح الطريقة التي يمكن من خلالها إدارة تسلسل تنفيذ العمليات المتعددة داخل وحدة المعالجة المركزية. في هذا السياق تُخصص كتلة من الذاكرة (فضاء عنونة) لكل عملية بالإضافة إلى جدول العملية الذي يُنشأ بواسطة نظام التشغيل لتتبع تنفيذ هذه العمليات، كما هو موضح في الشكل 2. 6. يتضمن هذا الجدول مؤشرًا إلى موقع كتلة الذاكرة التي تحتوي على العملية ولربما بعض من متطلبات التنفيذ الأخرى. من الملاحظ أيضًا من خلال هذا الشكل أنَّ عدَّاد البرنامج يُشير إلى التعليمات التالية في تلك العملية التي ستُنفذ، كما أن سجلي الأساس، والحد سيُحددان المساحة التي تشغلها العملية المعنية داخل الذاكرة، فسجل الأساس يُمثل عنوان بداية المساحة، بينما يُمثل سجل الحد حجمها، كما أنهما يُستخدمان لمنع التداخل بين العمليات، كما سيتم تبيانه بشكل تفصيلي في الفصل الخامس من هذا الكتاب.

يُبين المثال الموضح في الشكل 2. 6 أن العملية 1 حاليًا قيد التنفيذ لكون عدَّاد البرنامج يُشير إلى إحدى تعليماتها. إذا حدثت مقاطعة في أثناء ذلك من قبل العملية 2، وكانت ذات أهمية أعلى من العملية 1 فإن جميع ما يتعلق بهذه العملية من معلومات مثل عدَّاد البرنامج، وغيرها من البيانات المتعلقة بها يجب حفظها أولًا، لذلك ستُدفع إلى المكدس بواسطة الكيان المادي للمقاطعة، بعد ذلك يقفز النظام إلى العنوان المحدد في متجه المقاطعة والذي يحوي عنوان إجراء خدمة المقاطعة لكي يُنفذه.

في وقت لاحق ووفقًا لآليات جدولة العمليات، يُمكن لنظام التشغيل تبديل تنفيذ العملية واستئناف تنفيذ العملية الأولى. سيتطلب الأمر مرة ثانية حفظ معلومات وبيانات العملية 2 ودفعها في المكدس، بينما سيُحمل عدَّاد البرنامج بقيمة تُشير إلى مساحة برنامج العملية 1، وستُحمل السجلات وخريطة الذاكرة للعملية المتداولة الآن، ومن ثم استئناف التنفيذ تلقائيًا. سنعود إلى مناقشة موضوع المقاطعات بتفصيل أكثر في الفصل الثالث.



الشكل 2. 6: تنفيذ العمليات.

3.2 اتصالات العملية الداخلية

غالبًا ما تحتاج العمليات إلى التواصل مع بعضها البعض وذلك لغرض القيام ببعض من المهام التي تتطلب تشغيل أكثر من عملية. لذلك تُستخدم اتصالات العملية الداخلية (وهي مجموعة من واجهات البرمجة) لتنظيم الأنشطة بين العمليات المختلفة وتشغيلها بشكل متزامن في نظام التشغيل، ولتبادل البيانات بين خيوط العملية الواحدة، أو بين عمليات قيد التشغيل في نظام واحد، أو في عدة أنظمة متصلة بشبكة، وبالتالي تمكين مشاركة البيانات دون تداخل.

من الممكن أن تكون هذه الاتصالات ما بين العمليات ذات الصلة ببعضها (متعاونة) مثل، ما بين عمليات الأب والأبناء، والتي تتأثر بالعمليات التنفيذية فيما بينها، أو ما بين العمليات المستقلة، أي ما بين عمليتين مختلفتين أو أكثر، والتي لا تتأثر بالعمليات الأخرى المنفذة في النظام. على الرغم من أنه يمكن للمرء أن يعتقد أن العمليات المستقلة ستُنفذ بكفاءة عالية، ولكن من الناحية العملية، هناك عدة مواقف يمكن فيها استخدام الطبيعة التعاونية نذكر منها ما يلي:

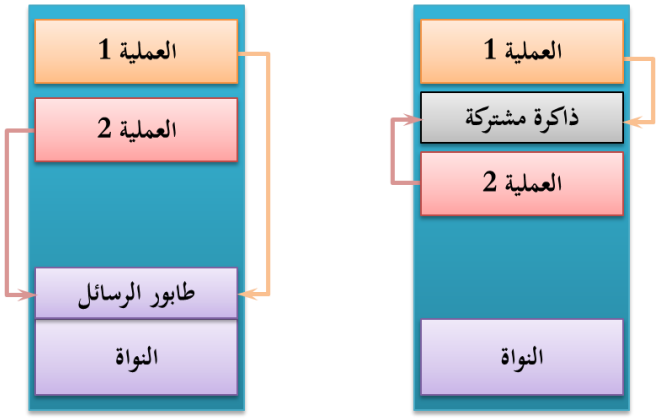
- تعجيل العمليات الحسابية: لزيادة سرعة تنفيذ وظائف محددة، يُمكن تقسيمها إلى عدة مهام فرعية بحيث تُنفذ على التوازي، شرط أن يكون لدى النظام القدرة على هذا النوع من التنفيذ.
- مشاركة المعلومات: من المهم أن يُوفر النظام إمكانية تسمح بالوصول المتزامن لنفس المعلومات (مثل، ملف مشترك)، الأمر الذي سيكون اقتصاديًا إلى حد كبير.
- الرفاهية: قد يرغب المستخدم الواحد في الحصول على تنفيذ متزامن لعدة مهام، كأن يقوم المستخدم بعمليات تحرير، وتهيئة، وطباعة، وكذلك ترجمة في آنٍ واحد.
- الحداثة: الرغبة في بناء أنظمة تعمل بطريقة معيارية تعتمد على مفاهيم تجزئة الوظائف إلى عمليات مصغرة أو عدة خيوط.

يمكن للعمليات التواصل مع بعضها البعض باستخدام عدة طرق نذكر منها الطريقتين التاليتين تمثيلًا لا حصرًا، والموضحتين في الشكل 2. 7.

1. الذاكرة المشتركة.
2. تمرير الرسالة.

تُنشأ منطقة من الذاكرة في نموذج الذاكرة المشتركة تتشارك فيها كل العمليات المتعاونة وتستخدمها لتبادل المعلومات بينها عن طريق قراءتها من– أو كتابتها في هذه المنطقة المشتركة، كما أنها تتطلب نوع من الحماية الخاصة عند حدوث الوصول المتزامن للعمليات. بالمقابل تتبادل العمليات ذات الصلة الرسائل في نموذج تمرير الرسائل عن طريق إرسالها إلى– أو استقبالها من طابور الرسائل، علمًا أن الرسالة المستلمة من قبل العملية الهدف ستُحذف من الطابور. في الأقسام التالية سوف يُنظر إلى بعض من القضايا والمشاكل المتعلقة بالاتصالات

الداخلية للعمليات.



الشكل 2. 7: طريقتي الذاكرة المشتركة، وتمرير الرسالة لاتصالات العملية الداخلية.

1.3.2 وضع التسابق

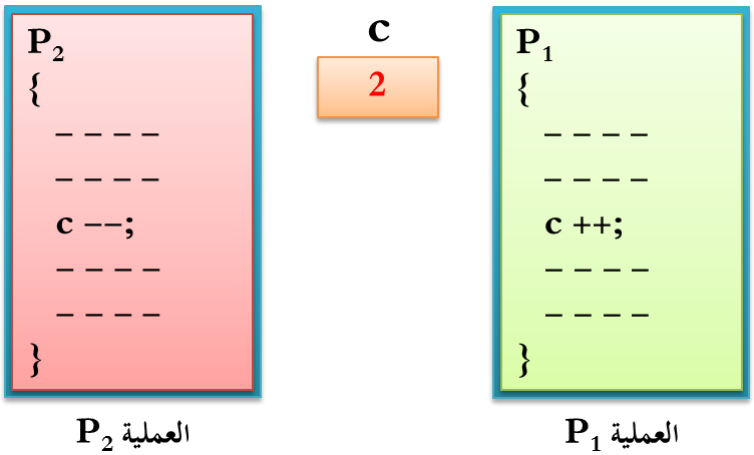
يُمكن للعمليات في نظم التشغيل أن تتشارك في بعض من مواقع التخزين وذلك لغرض قراءتها أو الكتابة فيها من قبل أي عملية. مواقع التخزين المشتركة هذه قد تكون مواقع في الذاكرة، أو قد تكون ملفات مشتركة في القرص، الأمر الذي يترتب عليه وجود اتصالات داخلية بين العمليات. على سبيل المثال، في الحواسيب المعتمدة على استخدام مفهوم الأنابيب[5]، تتواصل العمليات فيما بينها عن طريق تمرير ناتج العملية الأولى إلى العملية التالية في مراحل التنفيذ داخل الأنابيب، وهكذا مع بقية المراحل. يؤدي هذا الأمر بطبيعة الحال إلى ظهور عدة مشاكل.

[5] هي عبارة عن تقنية يمكن من خلالها للحاسوب معالجة الملايين من التعليمات في الثانية الواحدة. وفيها تُقسم عملية تنفيذ الأوامر إلى مجموعة من المراحل تُنفذ عبر أنابيب على التوازي في وقت واحد.

إحدى أهم هذه المشاكل هي مشكلة وضع التسابق وهي تحدث عندما تتعامل عمليتان أو أكثر في نفس الوقت مع بيانات مشتركة بحيث تعتمد النتيجة النهائية على أيُّهما نُفِّذ بدقة نظرًا لعدم وجود آليات التحكم المناسبة في التنفيذ. ولتوضيح هذه المشكلة بشكل تفصيلي، سنتطرق إلى مثالين بسيطين، لكنهما شائعان.

مثال المتغير العام المشترك

يفترض هذا المثال أن هناك عمليتان P_1 و P_2، كل منهما تتنافس في التعامل مع متغير عام قابل للمشاركة من قبل عدة عمليات. تتمثل مهمة إحدى هاتين العمليتين في زيادة قيمة المتغير العام بمقدار واحد، بينما تتمثل مهمة الأخرى في طرح القيمة واحد منه، كما هو موضح في الشكل 2. 8. من المفترض أن تبقى قيمة المتغير c في هذا الشكل ثابتة، أي 2 بعد تنفيذ هاتين العمليتين، لأن الأولى ستزيده بمقدار واحد، والثانية ستُقلله بنفس القيمة، كل هذا يحدث في حالة تنفيذ العمليتان بشكل متزامن، أي دون أن تقع مشكلة وضع التسابق بالنسبة للمتغير c.

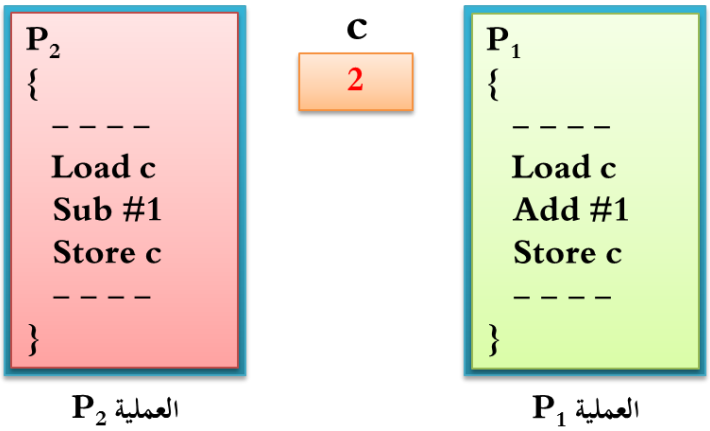


الشكل 2. 8: توضيح وضع التسابق من خلال مثال المتغير العام المشترك.

ولكن ما يحدث قد يكون مغايرًا تمامًا، أي قد نحصل على نتائج مختلفة بتنفيذ هاتين العمليتين عدة مرات في غياب التنفيذ التزامني. للتعرف أكثر على مثل هذه الحالات سيُترجم أمري الجمع، والطرح إلى تعليمات الآلة كما هو موضح في الشكل 2. 9. كنا قد نوهنا في السابق ولو بشكل موجز إلى أنه يمكن للعملية أن تكون قيد التشغيل، ولكنها قد تتوقف بسبب

انتظارها لأحداث خارجية، أو لأنها قد أخذت وقتها الكافي في التنفيذ، الأمر الذي قد يقود إلى اختيار عملية أخرى لكي تُنفذ.

والآن بالرجوع إلى النموذج الموضح في الشكل 2. 9 وتطبيق آلية التنفيذ سالفة الذكر، نجد أنه من الممكن أن تكون العملية P_1 قيد التنفيذ وبفرض أنها بالضبط في تنفيذ الأمر 'Load c' والذي سينتج عنه تحميل القيمة 2 في السجل المحلي لهذه العملية وغير المرئي بالنسبة للعملية P_2. على فرضية أن المجدول قرر في هذه اللحظة توقيف العملية الأولى وإفساح المجال أمام العملية الثانية لتبدأ في تنفيذ الجزء الخاص بها والتي ستُحمل قيمة المتغير c والتي لازالت 2 في سجلها المحلي وغير المرئي بالنسبة للعملية الأخرى، ولأن الزمن المخصص لهذه العملية لم ينتهي بعد ستطرح واحد، وستُخزن ناتج الطرح في المتغير c لتصبح قيمته 1. بإعطاء الفرصة من جديد بالنسبة للعملية P_1 والتي ستبدأ من آخر مكان توقفت فيه وهو في هذه الحالة تنفيذ أمر الجمع، ومن ثم أمر التخزين لينتج عنه أن قيمة المتغير c ستكون حاليًا 3، لأن آخر قيمة أُحتُفظ بها في سجلها المحلي هي 2. يُوضح الجدول 2. 5 خطوات التنفيذ هذه.



الشكل 2. 9: ترجمة أمري الجمع، والطرح إلى تعليمات الآلة.

بالمقابل يُبين الجدول 2. 6 تكرار للخطوات السابقة باستثناء أن العملية التي بدأت التنفيذ أولًا هي P_2، وأنها قد أُستوقفت لإعطاء الفرصة للعملية P_1، الأمر الذي نتج عنه أن القيمة النهائية للمتغير العام c هي 1. هذا الأمر يعكس حقيقة أن غياب التزامن في التنفيذ وخصوصًا عند التعامل مع البيانات المشتركة يُؤدي إلى اختلاف في نتائج التنفيذ المتكرر لنفس العمليات،

وهو ما أشرنا إليه سابقًا بوضع التسابق.

الجدول 2. 5: نتائج التنفيذ غير المتزامن بدءًا من العملية P_1.

المتغير العام c	العملية P_2		العملية P_1	
	السجل المحلي	التعليمة	السجل المحلي	التعليمة
2			2	Load c
2	2	Load c		
2	1	Sub #1		
1	1	Store c		
1			3	Add #1
3			3	Store c

الجدول 2. 6: نتائج التنفيذ غير المتزامن بدءًا من العملية P_2.

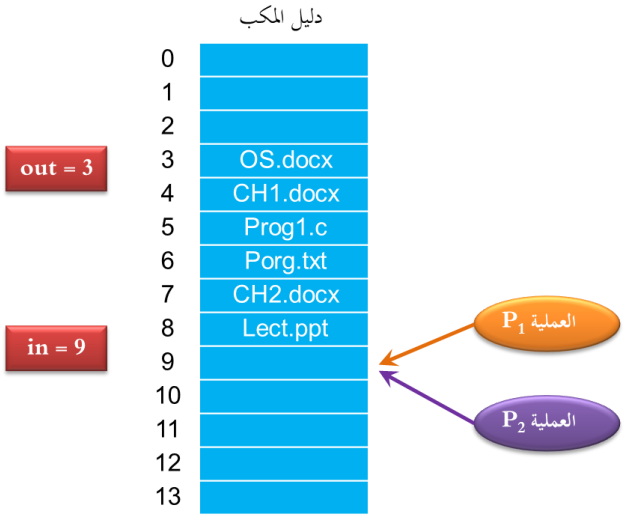
المتغير العام c	العملية P_2		العملية P_1	
	السجل المحلي	التعليمة	السجل المحلي	التعليمة
2	2	Load c		
2			2	Load c
2			3	Add #1
3			3	Store c
3	1	Sub #1		
1	1	Store c		

مثال مكب الطباعة

يتمثل المثال الشائع لتوضيح مشكلة وضع التسابق في مكب الطباعة، وهو عبارة عن مخزن لحظي يستقبل أسماء الملفات المراد طباعتها والمرسلة من قبل العمليات. لدى هذا المكب دليل يحتوي على عدة أماكن مرقمة من **0**، إلى الحد الأقصى لحجمه، والتي بإمكانها أن تحتفظ بأسماء هذه الملفات. من أجل تنظيم عملية الطباعة هناك عملية تُدعى الطابعة الخفية مهمتها الرئيسية فحص هذا الدليل دوريًا لغرض معرفة ما إذا كان هناك ملفات جاهزة للسحب أم لا، في حالة وجود ملفات ستسحبها على الطابعة، ومن ثم تحذف أسمائها من الدليل. لدى هذا المكب أيضًا متغيران قابلان للمشاركة: المتغير **out** والذي يُؤشر إلى الملف التالي في السحب، والمتغير **in** والذي يُؤشر إلى المكان الحر التالي داخل مكب الدليل، وذلك كما هو موضح في الشكل **2. 10**.

بفرض أن الأماكن الثلاثة الأولى في هذا المثال قد سُحبت ملفاتها على الطابعة، لذلك فهي حرة الآن، وأن الأماكن من 3 إلى 8 لازالت مشغولة، وبفرض أن هناك عمليتان P_1 و P_2 قد قررتا معًا في آن واحد إرسال ملفاتهما إلى دليل المكب كما هو موضح في الشكل 2. 10، لذلك ستتنافسان على قراءة المتغير **in** لمعرفة المكان الحر التالي. الخطوات التسلسلية للقيام بذلك تتمثل في التالي:

1- أي عملية ترغب في إرسال ملفها إلى مكب الطباعة، تقرأ أولًا قيمة المتغير **in**.
2- تُخزن العملية قيمة هذا المتغير في متغيرها المحلي.
3- تُرسل العملية اسم الملف إلى المكب.
4- تُضيف العملية القيمة واحد إلى متغيرها المحلي وتُخزنه في المتغير **out**.



الشكل 2. 10: العمليتان P_1 و P_2 ترغبان في الوصول إلى مكان مشترك في نفس الوقت.

مع فرضية أن العملية P_1 هي من قرأت المتغير **in** أولًا، ونفذت نفس الخطوات التسلسلية السابقة، هذا يعني أنها خزنت القيمة 9 في متغيرها المحلي الخاص بها كما هو الحال في المثال السابق. في أثناء ذلك نفترض أن المجدول قرر في هذه اللحظة مقاطعة هذه العملية لأنها قد أخذت وقتها الكافي في التنفيذ، وتخصيص وحدة المعالجة المركزية إلى العملية P_2، بالتالي ستقرأ هذه العملية أيضًا المتغير **in** مع تخزين القيمة 9 في متغيرها المحلي، ومن ثم إدخال اسم

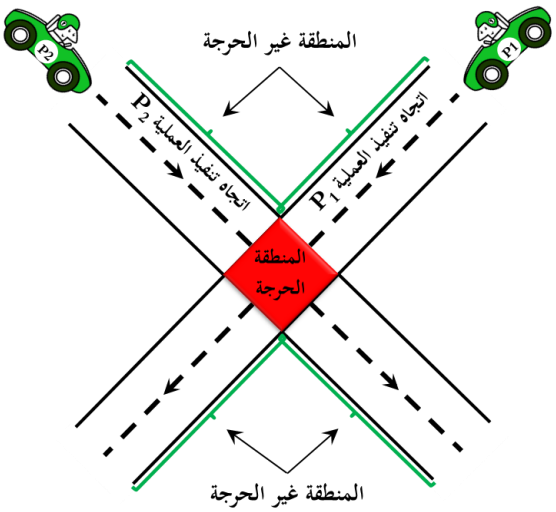
الملف المراد سحبه في المكان التاسع، بعدها ستُحدِّث قيمة المتغير **in** بالقيمة **10** حتى يُشير إلى المكان الحر التالي داخل دليل المكب، وبذلك تكون العملية P_2 قد أنهت مهمتها بنجاح.

أخيرًا، سيختار المجدول العملية P_1 للتنفيذ مرة ثانية، وستبدأ من آخر مكان توقفت فيه. هذا يعني أنها ستُرسل ملفها إلى الموقع **9**، لأنه يُعبر عن قيمة متغيرها المحلي والذي سينتج عنه مسح اسم ملف العملية P_2 والموجود في نفس المكان، ومن ثم تُضيف واحد إلى محتوى المتغير المحلي وتُخصص هذه القيمة إلى المتغير **in**. بهذه الوضعية سيكون دليل المكب على ما يُرام والطابعة الخفية لن تلاحظ أي خطأ، الأمر الذي سيترتب عنه عدم سحب أي نتائج للعملية P_2. الأمر نفسه سيتكرر مع العملية P_1، أي لن تتحصل على أي نتائج في حالة ما اختار المجدول العملية P_2 لتنفيذها أولًا، بالتالي ظهور مشكلة وضع التسابق.

2.3.2 المنطقة الحرجة

السؤال الذي يطرح نفسه الآن هو كيف يمكن منع حدوث مشكلة وضع التسابق؟ قبل الإجابة على هذا السؤال، سنُوضح أولًا مفهوم المنطقة الحرجة عن طريق الاستدلال بتقاطع طرق سيارات، كما هو موضح في الشكل 2. 11. تتحرك السيارتان في اتجاه تقاطع الطريقين (وهو اتجاه تنفيذ العمليتان) بحيث إذا وصلتا إلى منطقة التقاطع في نفس الوقت سيحدث تصادم بينهما وهو حدث غير مرغوب فيه البتة، لذلك تُمثل هذه المساحة المنطقة الحرجة بالنسبة للسيارتين، لأنها جزء لا يتجزأ من طريق كل سيارة (عملية). ما يجب فعله الآن هو البحث عن وسائل تمنع دخول السيارتين، أي العمليتين إلى منطقة التقاطع، أي المنطقة الحرجة في نفس الوقت، أي منعهما بالتبادل.

بإسقاط مفهوم المنطقة الحرجة على مشكلة وضع التسابق، نقول بأنه يُمكن منع هذه المشكلة عن طريق إيجاد طرق تمنع دخول العمليات إلى مناطقها الحرجة في نفس الوقت، أي منع حدوث قراءة أو كتابة بيانات مشتركة من قبل أكثر من عملية في نفس الوقت. بالتالي ما نحتاجه هو ما يُسمى بالمنع التبادلي، أي إيجاد طرق تضمن منع أي عملية من استخدام متغير أو ملف مشترك جاري استخدامه من قبل أحد العمليات وفي نفس الوقت. لذا فإن مشكلة منع وضع التسابق يُمكن تلخيصها بشكل آخر، وكما هو مبين في الشكل 2. 12.



الشكل 2. 11: تمثيل بسيط للمنطقة الحرجة عن طريق تقاطع طريقين.

يُفَصَّل هذا الشكل هيكلية العملية إلى جزء يُهيئ العملية لطلب الدخول إلى المنطقة الحرجة (والمشار إليه بالجزء **Entry section** في هذا الشكل)، وجزء يتعامل مع مهام حرجة (بيانات مشتركة) كتحميل، أو حفظ المتغير العام **c**، أو قراءة المتغير المشترك **in** في المثالين السابقين، على التوالي، والتي تؤدي بدورها إلى وضع التسابق، وهو يُدعى بالجزء الحرج أو المنطقة الحرجة. هناك جزء ثالث يُعرف بالمنطقة غير الحرجة وتشمل الجزء الذي يُجري الحسابات الداخلية وبعض من المهام الأخرى التي لا تؤدي بدورها إلى وضع التسابق مثل، إضافة وطرح القيمة واحد في المثال الأول، وتخزين قيمة المتغير **in** في المتغير المحلي للعملية.

لذلك لو تمكنا من تنسيق مجموعة العمليات التي تصل إلى المصادر المشتركة بحيث نمنع تواجد عمليتين أو أكثر داخل مناطقها الحرجة في نفس الوقت، فإنه بالإمكان المساهمة في منع حدوث مشكلة وضع التسابق. ولكن هذا وحده لا يكفي وخصوصًا في إذا ما كانت هناك رغبة في تنفيذ عمليات التوازي والتي تتطلب استخدام مصادر مشتركة بطريقة صحيحة وبكفاءة عالية، بل سيتطلب الأمر إضافة إلى ذلك تحقيق الشروط الأربعة التالية:

- المنع التبادلي، أي عدم السماح بتواجد أكثر من عملية في آن واحد داخل مقاطعها الحرجة.

- عدم السماح لأي عملية بالانتظار اختياريا لفترة زمنية طويلة لكي تدخل مقطعها الحرج.
- عدم السماح لأي عملية موجودة خارج مقطعها الحرج بإيقاف عمليات أخرى.
- عدم السماح بوضع فرضيات حول السرعة النسبية للعملية، أو عدد وحدات المعالجة المركزية.

```
while (TRUE){
    Entry section
        Critical section (e.g. Load c, Store c)
    Exit section
        Non-Critical section (e.g. Add #1, Sub #1)
}
```

الشكل 2. 12: تفصيل بنية العملية.

3.3.2 المنع التبادلي والانتظار المشغول

المنع التبادلي ما هو إلا وسيلة لضمان تواجد عملية واحدة فقط داخل منطقتها الحرجة ومنع العمليات الأخرى من الدخول في نفس الوقت، لكي لا تُسبب في حدوث مشكلة وضع التسابق. لذلك سنحاول في هذا القسم التعرف على بعض من الاقتراحات الخاصة بإنجاز هذه الوسيلة.

1.3.3.2 تعطيل المقاطعات

من الطرق الأكثر وضوحًا لتحقيق المنع التبادلي هو السماح للعملية الراغبة في الوصول إلى مصدر مشترك والدخول إلى منطقتها الحرجة بتعطيل كل المقاطعات مباشرةً بعد دخولها لها، ومن ثم إعادة تمكينها مباشرةً قبل مغادرتها إيَّاها. بهذه الآلية سوف لن تتمكن وحدة المعالجة المركزية من التنقل بين العمليات، الأمر الذي سيضمن عدم استخدام بقية العمليات لهذا المصدر في نفس الوقت.

على الرغم من أن تعطيل المقاطعات قد يبدو حلاً جيدًا، إلَّا إنَّ عيوبه تفوق بكثير مزاياه. من الناحية العملية، يُعتبر تعطيل المقاطعات من المهام الكبرى في الحاسوب، بالتالي إعطاء هذه القوة لعمليات المستخدم قد يجعل الجهاز في أحسن الأحوال غير قادر على خدمة المقاطعات لفترة طويلة (لأننا لا ندري ماذا ستفعل العملية داخل منطقتها الحرجة؟ وكم تحتاج من الوقت؟)، وفي أسوئها قد لا تتمكن العملية من إعادة تمكين المقاطعات مطلقًا، وبالتالي تعطل النظام بشكل نهائي. إذًا ليس من الحكمة أن تُعطى عمليات المستخدم القدرة على إيقاف المقاطعات والتحكم فيها.

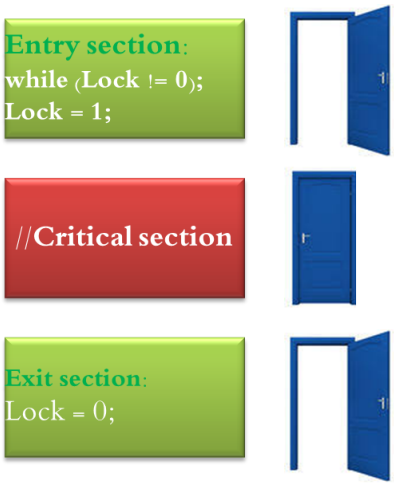
هذا الحل قد يتماشى فقط مع الأنظمة أحادية المعالج، لأنه في حالة امتلاك الحاسوب لأكثر من وحدة معالجة مركزية فإن إجراء تعطيل المقاطعات سيؤثر فقط على الوحدة التي تُنفذ أمر التعطيل، بينما ستُوَاصل الوحدات الأخرى عملها بشكل طبيعي، مما سيؤدي حتمًا إلى حدوث مشكلة وضع التسابق، وذلك لإمكانية وصول عملياتها إلى المصادر المشتركة. كذلك هذا الحل سيُعطِّل مفهوم البرمجة المتعددة، لأن تعطيل المقاطعات قد يمنع عمليات أخرى غير مهتمة بمنطقتها الحرجة من التنفيذ.

2.3.3.2 متغير القفل

كمحاولة أخرى لحل مشكلة وضع التسابق سنحاول في هذا القسم الاطلاع على حل معنوي يُنفذ في نمط المستخدم، أي لا يحتاج إلى دعم من نظام التشغيل. يستخدم هذا الحل متغير قابل للمشاركة يُدعى **lock** قيمته الاستهلالية صفر، وهي تعني أنه بإمكان العمليات الدخول إلى مناطقها الحرجة، بينما القيمة الأخرى المحتملة له هي واحد، والتي تعني أنَّ الباب مقفل، وأن ثمَّة عملية شاغرة لمنطقتها الحرجة وغير مسموح بدخول العمليات الأخرى لهذه المناطق، كما هو موضح في الشكل 2. 13.

عندما ترغب عملية– ما– في الدخول إلى منطقتها الحرجة تتفحص أولًا متغير القفل، إذا كانت قيمته صفر، فستُسند له القيمة واحد، ومن ثم تدخل إلى الجزء الحرج، أمَّا إذا كانت قيمته واحد فستدخل العملية في حلقة انتظار إلى أن يُسند له قيمة صفر من جديد وهو ما يُعبر عنه بالانتظار المشغول. هذا يعني أن القيمة صفر تدل على عدم تواجد عملية داخل المنطقة الحرجة،

بينما تعني القيمة واحد تواجد عملية داخل منطقتها الحرجة.



الشكل 2. 13: هيكلية خوارزمية حل متغير القفل.

يُوفر متغير القفل أبسط آلية مزامنة للعمليات، ولكن هناك بعض من النقاط الجديرة بالملاحظة. إحداها أنه ببساطة ينقل المشكلة من المتغير المشترك إلى متغير القفل، بمعنى أن هذا الحل هو الآخر عرضة لنفس المشكلة القاتلة والتي ظهرت في دليل المكب. لو افترضنا أن هناك عمليتان، قرأت إحداهما المتغير **lock** وكانت قيمته **0**، وقبل أن تَسند له القيمة واحد قرر المجدول إيقاف هذه العملية– نتيجة نفاذ وقتها– وتشغيل عملية أخرى والتي ستقرأ بدورها المتغير **lock** وستجد قيمته لازالت **0**، عندها ستُغير قيمتة، وستبدأ في الدخول إلى منطقتها الحرجة. في هذه اللحظة، بفرض أن المجدول أعطى الفرصة من جديد للعملية الأولى، عليه وكما أشرنا سابقًا ستبدأ من المكان التي توقفت فيه آخر مرة، أي ستُغير قيمة المتغير **lock** إلى **1**، وستدخل هي الأخرى إلى جزئها الحرج. وفقًا لشرط المنع التبادلي (يجب ألَّا تكون هناك أكثر من عملية واحدة داخل المنطقة الحرجة في نفس الوقت)، بالتالي لا يضمن حل متغير القفل تحقيق هذا الشرط. الأمر الآخر أن هذا الحل يُعاني من الانتظار المشغول وهو ما يهدر وقت المعالج، لذلك فهو ليس بالحل الأمثل.

يُعرف الانتظار المشغول بأنه دخول أي عملية في حلقة تكرارية واختبارها بشكل مستمر لمتغير– ما– وانتظاره حتى يأخذ قيمة محددة. مثل هذه الحلول والتي تُعاني من هذه المشكلة

يجب استبعادها، لكونها تهدر وقت وحدة المعالجة المركزية. ولكن عندما يكون هناك توقع معقول بأن الانتظار سيكون قصير، فمن الممكن استخدام هذا النوع من الحلول.

3.3.3.2 التناوب الدقيق

في حل متغير القفل كانت هناك مشكلة فعلية تكمن في حقيقة أن أي عملية كانت تدخل إلى منطقتها الحرجة فقط عندما تكون قيمة هذا المتغير 1، لذلك كان باستطاعة أكثر من عملية رؤية هذه القيمة في نفس الوقت وبالتالي عدم ضمان المنع التبادلي هناك. من هنا جاءت فكرة حل التناوب الدقيق والتي تكمن في جعل العملية تدخل منطقتها الحرجة فقط في الحالة التي تكون فيها قيمة مُعرِّفها مساوي تمامًا لقيمة المتغير **turn**، كما هو مفصل في الخوارزمية الموضحة في الشكل 2. 14. تُنفذ هذه الخوارزمية في نمط المستخدم وهي تستخدم المتغير الصحيح **turn** لغرض تتبع تعاقب دخول العمليات إلى مناطقها الحرجة عبر تغيير قيمته من قبلها في أثناء التنفيذ.

```
Initially turn = i

   Loop forever {                Loop forever {
      while (turn != j){}           while (turn != i){}
      Critical_section              Critical_section
      turn = i                      turn = j
      N-CS}                         N-CS}
```

(ب) (أ)

الشكل 2. 14: حل التناوب الدقيق المقترح لمشكلة المنطقة الحرجة– أ) العملية **i**– ب) العملية **j**. في كلا الحالتين، N–CS ترمز للمنطقة غير الحرجة.

يُمكن لقيمة المتغير **turn** أن تأخذ فقط إحدى القيمتين **i** أو **j**، بمعنى إذا لم تكن قيمته **i**، فمن المؤكد أنها ستكون **j** أو العكس. بالتالي يمكن استخدام هذا الحل فقط مع عمليتين هما P_i و P_j، واللتان ستشتركان في تعاقب تبديل قيمة هذا المتغير والذي قيمتة الاستهلالية **i**، وستكون الخطوات على النحو التالي: مبدئيًا تفحص العملية P_i المتغير **turn** وستجد قيمته تساوي **i** وهو ما سيعطيها الإذن بالدخول إلى منطقتها الحرجة، بينما سيجعل العملية P_j تدخل في تكرار محكم تختبر فيه دوريًا المتغير **turn** لمعرفة ما إذا أُسند له القيمة **j** من قبل العملية P_i أم لا. عندما تخرج العملية P_i من منطقتها الحرجة، تضبط المتغير **turn** على **j** مما

يسمح للعملية P_j بدخول منطقتها الحرجة. إذا حاولت في هذه الحالة العملية P_i الدخول إلى منطقتها الحرجة مرة أخرى، فستُحظر لأن المتغير **turn** لم تعد قيمته **i**.

مع أن هذا الحل قد يبدو للوهلة الأولى أنه حل مشكلة وضع التسابق، إلَّا إنَّه يُعاني من خلل كبير يتضح مع تأمل سلسلة الأحداث التالية. تشتغل العملية P_i وتدخل منطقتها الحرجة، وتخرج منها بعد أن تضبط المتغير **turn** على **j**، لتُعطي الإذن للعملية P_j بالدخول إلى جزئها الحرج. الآن العملية P_i في جزئها غير الحرج وبفرض أنه سيستغرق وقتًا طويلًا. من الممكن الآن تشغيل العملية P_j والتي لها الحق في الدخول إلى الجزء الحرج، بفرض أنها دخلت إليه وخرجت منه، لأنها أسرع بكثير من P_i، بمجرد تركها، ستضبط المتغير **turn** على القيمة **i**، بعدها ستُنفذ جزئها غير الحرج بسرعة كبيرة وتعود إلى أعلى الإجراء. الوضع الآن هو أن العملية P_i في جزئها غير الحرج والعملية P_j تنتظر في أن تُسند القيمة **j** إلى المتغير **turn**.

في الواقع، لا يوجد سبب يمنع العملية الأخيرة من الدخول إلى منطقتها الحرجة لأن العملية P_i مشغولة بمنطقتها غير الحرجة. بعبارة أخرى، عملية التناوب هذه ليست بالفكرة الجيدة، وخصوصًا عندما تكون إحدى العمليات أسرع بكثير من غيرها، لأن ما نراه هنا هو انتهاك للشرط الثالث المذكورة أعلاه، أي أنَّ هناك عملية خارج منطقتها الحرجة ومنعت عملية أخرى من الدخول إلى منطقتها الحرجة. بالعودة إلى دليل المكب السابق ذكره وربط المنطقة الحرجة بعملية القراءة من والكتابة في دليل المكب، نجد أن العملية P_j غير قادرة على طباعة ملف آخر بسبب انشغال العملية P_i بأعمال أخرى.

4.3.3.2 خوارزمية بيترسون

في سنة **1981** قدَّم العالم بيترسون خوارزمية بسيطة تُنفذ في نمط المستخدم قادرة على توفير أغلب متطلبات حل مشكلة وضع التسابق مثل المنع التبادلي، وغيره، إلَّا إنَّها تُعاني من مشكلة الانتظار المشغول، كما أنها قابلة للتطبيق لعمليتين فقط. يتكون هذا الحل من إجراءين موضحين في صيغة نمط لغة السي في الشكل 2. **15** ويستخدم المتغير **turn** والمصفوفة أحادية البعد **interested**. الإجراء الأول هو **Enter_CS**، وهو إجراء الدخول إلى المنطقة الحرجة، والذي تستدعيه أي عملية ترغب في الدخول إلى هذه المنطقة، أمَّا الإجراء

الثاني فهو **Exit_CS** ويُمثل إجراء الخروج من المنطقة الحرجة، أي تستدعيه العملية عندما ترغب في الخروج منها.

تتلخص فكرة هذه الخوارزمية في استدعاء العملية الراغبة في الدخول إلى منطقتها الحرجة لإجراء الدخول مع تمرير رقم العملية (**0** أو **1**). يترك هذا الاستدعاء العملية في حالة انتظار– إذا تتطلب الأمر ذلك– إلى حين تأمين الدخول إلى هذه المنطقة، أمَّا عند الإنتهاء من تنفيذ الجزء الحرج، فستستدعي نفس العملية إجراء الخروج لكي تخرج منه، وبالتالي تسمح للعمليات الأخرى بدخول مناطقها الحرجة إذا توفرت الرغبة لديهم.

```
... // required header files
#define FALSE  0
#define TRUE   1
#define N      2                    // Number of processes
int    turn;                        // Whose turn is it?
int    interested[N];               // All values initially 0

void Enter_CS(int process)          // Process ID: 0 or 1
{
   int other;                       // The process other ID
   other = 1 - process;             // The opposite process
   interested[process] = TRUE;      // This process is interested
   turn = process;                  // Set flag

while (interested[other] && turn == process); // Wait
}


void Exit_CS(int process)
{
   interested[process] = FALSE;  // Process leaves CS
}
```

الشكل 2. 15: طريقة خوارزمية بيترسون لإنجاز عملية المنع التبادلي.

في البداية، كلتا العمليتان خارج منطقتهما الحرجة والمصفوفة **interested** تحمل القيمة الاستهلالية **0**. بفرض أن العملية P_0 استدعت إجراء الدخول، بالتالي سيُضبط المتغير **other** على **1**، وستُسند القيمة 'صح' للموقع المناظر لهذه العملية داخل المصفوفة للدلالة على اهتمامها بالدخول إلى المنطقة الحرجة. بعد ذلك يُسند رقم العملية للمتغير **turn** قبل أن يأتي

الدور على تنفيذ الحلقة التكرارية. في هذه الحالة، بما أن الشرطين الواردين في الحلقة مستوفيان تمامًا بواسطة العملية $\mathbf{P_0}$، فإنها ستنجو من هذه الحلقة وتدخل إلى منطقتها الحرجة.

والآن إذا ما قامت العملية $\mathbf{P_1}$ باستدعاء إجراء الدخول، فسوف يُعلق طلبها إلى حين إسناد القيمة 'خطأ' إلى الموقع المناظر لهذه العملية داخل المصفوفة، وهذا سيحدث فقط عندما يُستدعى إجراء الخروج من قبل العملية $\mathbf{P_0}$. إذا وصلنا إلى حالة تستدعي فيها كلتا العمليتان إجراء الدخول إلى المنطقة الحرجة في نفس الوقت، كلاهما سوف يُخزِّن رقم العملية الخاص بها في المتغير **turn**، وبغض النظر عن من خزَّنت أولًا فإن هذا المتغير سيحوي رقم آخر عملية قامت بالتخزين، وبفرضية أنها العملية $\mathbf{P_1}$ فهذا يعني أن قيمته الآن هي **1**. عليه عندما تُنفذ العمليتان جملة الحلقة، سوف تختبر العملية $\mathbf{P_0}$ الشرط وستجده غير متحقق، وستدخل إلى منطقتها الحرجة، بينما سيتحقق الشرط بالنسبة للعملية $\mathbf{P_1}$ وهو ما يجعلها تدخل في تكرار محكم داخل الحلقة، بالتالي لن تستطيع الدخول إلى منطقتها الحرجة حتى تخرج العملية $\mathbf{P_0}$ من جزئها الحرج.

5.3.3.2 أمر اختبار وضبط القفل

في علوم الحاسوب من الممكن الاستعانة بالكيان المادي لحل مشكلة وضع التسابق من خلال مجموعة التعليمات الخاصة بالمعالج، وذلك لما يُقدمه من سهولة في البرمجة، وتحسين في فعالية النظام. في بيئة المعالج الواحد تمكنا من منع حدوث هذه المشكلة بواسطة تعطيل المقاطعات، إلَّا إنَّ هذا الحل غير ممكن في بيئة المعالجات المتعددة، لأن التعطيل سيؤثر فقط في المعالج الصادر لأمر التعطيل. من التعليمات الأخرى التي تستعين بالكيان المادي أمر اختبار وضبط القفل والذي يصلح لبيئة المعالجات المتعددة، على العكس من طريقة تعطيل المقاطعات، تتمثل مهمة هذا الأمر في قراءة محتويات موقع في الذاكرة وتخزينها في سجل، ومن ثم تخزين قيمة غير صفرية في عنوان ذلك الموقع، والتي يُشترط في تنفيذها أن تكون غير قابلة للتجزئة، أي أنه لا يمكن لأي عملية أخرى الوصول إلى موقع الذاكرة هذا، إلا بعد إنتهاء تعليمات هذا الأمر.

يَستخدم المعالج في أنظمة المعالجات المتعددة أمر اختبار وضبط القفل لقفل ناقل الذاكرة وذلك لغرض منع المعالجات الأخرى من الوصول إلى الذاكرة إلى حين إنهاء المعالج الأول

مهمته. في إطار حل مشكل وضع التسابق يمكننا استخدام هذا الأمر الخاص لتنسيق الوصول إلى الذاكرة المشتركة، وللتحكم في الدخول إلى المنطقة الحرجة بطريقة بسيطة نسبيًا باستخدام متغير مشترك **flag**. عندما تكون قيمتة **0**، فإنه بإمكان أي عملية إسناد **1** إليه باستخدام أمر اختبار وضبط القفل، ومن ثم قراءة أو كتابة الجزء المشترك، وبعد الإنتهاء من ذلك تسند إليه العملية القيمة **0** من جديد بواسطة الأمر **move**. البرنامج الموضح في الشكل 2. **16** والمكتوب بلغة التجميع يُبين الكيفية التي يُستخدم بها هذا الأمر لمنع تواجد أكثر من عملية داخل مناطقها الحرجة.

ينسخ أول أمر هنا المحتوى القديم للمتغير **flag** في السجل **R0** ويُسند **1** إلى المتغير، وذلك لإخفاء القيمة الأصلية له، بينما يُقارن الأمر التالي القيمة القديمة والمنسوخة في **R0** بالقيمة **0**. إذا كانت قيمته غير صفرية، فهذا يعني أن القفل مغلق والبرنامج سوف يذهب إلى البداية ويختبر المتغير **flag** من جديد، وهو ما يؤدي بالطبع إلى الانتظار المشغول. عاجلًا أم آجلًا سيُسند **0** إلى المتغير **flag** وذلك عندما تنتهي العملية من منطقتها الحرجة وسيتم الرجوع إلى البرنامج الرئيسي. كما أشرنا سابقًا يُستخدم أمر بسيط وهو **move** لمسح محتوى المتغير **flag** وذلك عن طريق استدعاء البرنامج الفرعي والموضح أيضًا أسفل الشكل 2. **16**.

```
ENTER_CS:
TSL  R0, flag     |Copy flag to R0 and set flag to 1
CMP  R0, #0       |Was flag 0?
JNZ  ENTER_CS     |If it was not zero, flag was set, so loop
RET               |Return and enter critical section

EXIT_CS:
MOVE flag ,#0     |Store a 0 in flag
RET               |Return
```

الشكل 2. **16**: حل مشكلة وضع التسابق باستخدام أمر اختبار وضبط القفل.

كبديل لأمر اختبار وضبط القفل يمكن استخدام أمر الاستبدال (**XCHG**) والذي يستبدل محتويات موقعي تخزين مثل، سجل، وموقع في الذاكرة. يُوضح الشكل 2. **17** البرنامج المكتوب بلغة التجميع والخاص باستخدام هذا الأمر في منع تواجد أكثر من عملية داخل مناطقها الحرجة. حيث يمكن ملاحظة أن هذا الحل مشابه تمامًا لحل طريقة أمر الاختبار. علمًا بأن جميع وحدات المعالجة المركزية نوع إنتل **x86** تستخدم تعليمة الاستبدال هذه لعمليات

المزامنة ذات المستوى المنخفض.

إن استخدام هذا النوع من الحلول في الحقيقة يُوفر حلا لمشكلة المنطقة الحرجة بطريقة صحيحة، إلَّا إنَّ مشكلة الانتظار المشغول لا تُساعد في الاستفادة بشكل أمثل من وحدة المعالجة المركزية، لأنها تُبقي هذه الوحدة مشغولة طوال الوقت في التحقق باستمرار من شرط الحلقة التكرارية، على الرغم من أنَّ كل ما تقوم به العملية في هذه الحالة هو انتظار الفرصة للدخول إلى منطقتها الحرجة، لذلك كان لزامًا البحث عن بدائل أفضل.

```
ENTER_CS:
MOVE R0          |Put a 1 in the register
XCGH R0, flag    |Swap the contents of the R0 and flag
CMP R0,#0        |Was flag 0?
JNE ENTER_CS     |If it was not zero, flag was set, so loop
RET              |Return and enter critical section

EXIT_CS:
MOVE flag,#0     |Store a 0 in flag
RET              |Return
```

الشكل 2. 17: الدخول والخروج من المقطع الحرج باستخدام أمر الاستبدال.

4.3.2 المنوم والمنبه

لاحظنا من خلال القسم السابق أن أغلب الحلول التي تعرضنا لها إلى حد الآن تُعاني من مشكلة الانتظار المشغول، والذي كما أشرنا سابقًا بأنه لا يسمح باستغلال وقت المعالج بشكل أمثل، بالرغم من أنها وفرت حلًا مقبولًا بالنسبة إلى مشكلة وضع التسابق. إضافةً إلى ذلك هناك مشكلة أخرى ترتبط بمثل هذه الحلول تُعرف بمشكلة انعكاس الأولوية. تحدث هذه المشكلة عندما تُجدول العمليات على أساس الأولويات وتكون هناك عمليتان، تتمتع الأولى بأولوية عالية، والثانية بأولوية منخفضة. في هذه الحالة يختار المجدول تنفيذ العملية الأولى بمجرد دخولها حالة جاهزة بغض النظر عن حالة العملية الثانية، والتي بفرض أنها في لحظة– ما– كانت موجودة داخل جزئها الحرج، وفي أثناء ذلك دخلت العملية الأولى إلى حالة جاهزة. من المفترض الآن أن المجدول سيختار هذه العملية للتنفيذ بحكم أولويتها، ولكن ولأن العملية الثانية مشغولة بجزئها الحرج، والمجدول لن يُراعها لكي تخرج منه، بالتالي ستبقي العملية الأولى إلى الأبد في حالة

الانتظار المشغول.

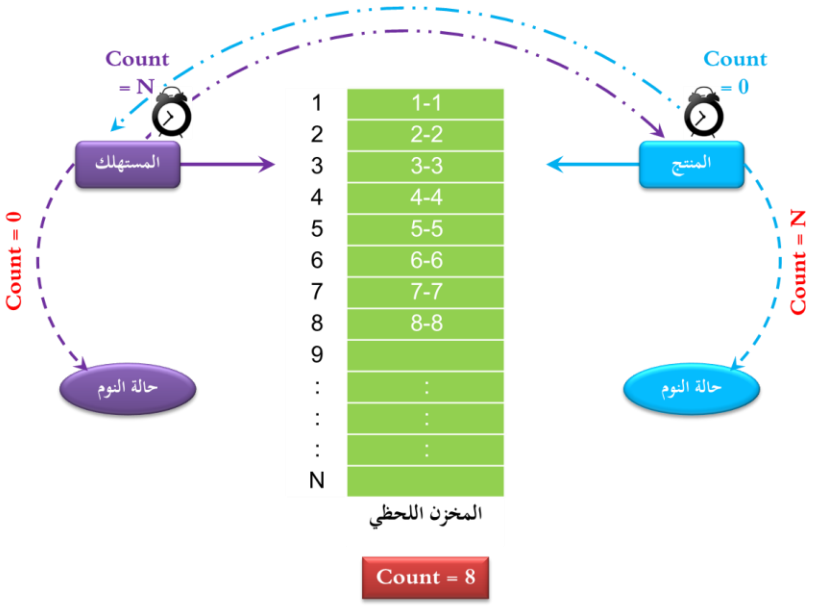
ولأن الانتظار المشغول أمرًا غير مرغوب فيه، وجب البحث عن بدائل أفضل لمشكلة المنع التبادلي، من هنا جاءت فكرة حل المنوم والمنبه، والتي تعتمد على منع العمليات من الدخول في الحلقات التكرارية التي تشغل المعالج، وبدلًا من ذلك تُوجه هذه العمليات إلى الذهاب إلى حالة جديدة تُعرف بحالة النوم، ومن ثم تُنبَّه عندما يأتي دورها للدخول إلى مناطقها الحرجة. يستخدم الحل الجديد إجرائين هما: المنوم، والمنبه. فالأول عبارة عن برنامج فرعي يستدعي نظام يُسبب في حظر المستدعي، أي جعله يذهب إلى حالة النوم ويبقى هناك إلى أن تُنبهه عملية أخرى خرجت للتو من منطقتها الحرجة، بينما يُمثل المنبه استدعاء النظام الذي يُنبه العملية التي دخلت في حالة النوم، لكي تستكمل عملها. لتوضيح فكرة عمل هذا الحل، سنحاول في هذا القسم التعرف أولًا على مشكلة المنتج والمستهلك.

مشكلة المنتج والمستهلك

كمثال توضيحي لهذه المشكلة، نفترض أن هناك مخزن لحظي يحوي عدد **N** من العناصر وله عمليتان تستخدمانه هما: المنتج، والمستهلك، كما هو موضح في الشكل 2. 18. تضع العملية الأولى البيانات داخل المخزن، بينما تُخرِجها الثانية منه، كما أن هناك متغير **Count** يُستخدم لتتبع عدد العناصر داخل هذا المخزن، والذي قيمته الحالية 8.

يُوضح الشكل 2. 19 الخوارزمية الخاصة بكل من المنتج والمستهلك لإضافة العناصر إلى المخزن وحذفها منه. عندما يرغب المنتج في إضافة عنصر إلى المخزن وكان هذا المخزن ممتلئ، أي **Count = N**، فلن يستطيع ذلك، بالتالي يتحتم عليه الآن الذهاب إلى النوم. إذا لم يمتلئ المخزن بعد، فبإمكانه إضافة العنصر إلى المخزن ومن ثم زيادة قيمة المتغير **Count** بمقدار واحد. عند محاولة المستهلك إخراج عنصر أو أكثر من المخزن ويكتشف أن المخزن ممتلئ، سيعتقد أن المنتج في حالة النوم، لذلك سيُرسل إليه إشارة إيقاظ، لكي يُنبهه، وم ثم يُكمل مهمته. بالمثل عندما يُحاول المستهلك إخراج عنصر من المخزن وكان هذا المخزن خالي من العناصر، سيُضطَّر إلى الذهاب هو الآخر إلى حالة النوم ويبقى هناك إلى أن يُرسل له المنتج إشارة إيقاظ عندما يجد أن قيمة المتغير **Count** تساوي **0**. أمَّا إذا كانت عكس ذلك، فسيقوم

بحذف العنصر من المخزن وإنقاص قيمة هذا المتغير بمقدار واحد.



الشكل 2. 18: توضيح فكرة حل المنوم والمنبه من خلال مشكلة المنتج والمستهلك.

للوهلة الأولى تبدو هذه الطريقة بسيطة بما فيه الكفاية، ولكنها لا تخلو هي الأخرى من التعقيدات. لنفترض أن المخزن خالي من العناصر، وقرأ المستهلك المتغير **Count** ليجد قيمته تساوي **0**، في هذه اللحظة قرر المجدول إيقاف المستهلك وتشغيل المنتج، والذي بدوره سيقوم بإضافة عنصر إلى المخزن وإضافة **1** إلى هذا المتغير. من خلال الاختبار المناسب سيلاحظ المنتج أن قيمة المتغير **Count** كانت صفر، الأمر الذي سيجعله يستدعي إجراء المنبه والذي سيُرسل بدوره إشارة إيقاظ إلى المستهلك. لسوء الحظ فإن المستهلك لم يدخل حالة النوم بعد، وهو ما سيتسبب في ضياع إشارة الإيقاظ هذه. عندما يُعطى المستهلك الفرصة من جديد ليشتغل، سيبدأ من آخر مكان توقف فيه، بالتالي سوف يذهب إلى النوم. عاجلًا أم آجلًا سيمتلئ المخزن من قبل المنتج وسيذهب هو الآخر إلى النوم وسيبقيا هناك إلى الأبد.

من خلال تتبع هذه الآلية نلاحظ أن جوهر المشكلة هنا يكمن في فقدان إشارة الإيقاظ، بالتالي فلو تمكنا من إيجاد طريقة لمنع ضياعها فسيكون كل شئ على ما يرام. من الحلول المقترحة لحل هذه المشكلة هو تعديل قواعد الإضافة بحيث تشمل إضافة خانة ثنائية تُسمى

خانة إنتظار الصحوة قيمتها الاستهلالية 0، وتُضبط عندما تُرسل إشارة إيقاظ على القيمة 1، بالتالي قبل أن تذهب أي عملية إلى النوم وكانت هذه الخانة تحمل القيمة 1، سيتحتم عليها إعادة ضبط هذه الخانة على الحالة 0، وستبقى يقظة لتُواصل عملها بالطريقة المعتادة.

```
Size of buffer, N
Initial value of Count = 0

producer( )
{
 Loop forever{
   Prepare next item
   If buffer is full, call sleep()
   Insert item in buffer
   Increment number of items in buffer
   Was buffer empty? wakeup(consumer)
 }
}

consumer( )
{
 Loop forever{
   If buffer is empty, call sleep()
   Remove item from buffer
   Decrement number of items in buffer
   Was buffer full? wakeup(producer)
   Consume items
 }
}
```

الشكل 2. 19: حل مشكلة المنتج والمستهلك باستخدام المنوم والمنبه.

هذا التعديل في الحقيقة لا يكفي لحل هذه المشكلة جذريًا، لكونه لا يتماشى مع الحالات التي يُتعامل فيها مع أكثر من عمليتين، كتصحيح مبدئي لهذا الحل يمكن إضافة خانة أخرى أو ربما 8 أو حتى 32 خانة، ولكن من حيث المبدأ المشكلة لا تزال قائمة، لذلك وجب البحث عن حل أفضل.

5.3.2 منظم دخول العمليات (الإشارة)

بالرجوع إلى تقاطع الطرق الموضح في الشكل 2. 11، نجد أن الحل الأمثل لمنع السيارتين من التصادم مع بعضهما البعض يَكمن في استخدام إشارة ضوئية للتحكم في دخول السيارات إلى المنطقة الحرجة. بناءً على هذا يُمكن توضيح فكرة عمل الإشارة، والتي اقترحها العالم ديكسترا (E. W. Dijkstra) في عام 1965، وتَكمن في استخدام متغير S صحيح وموجب بهدف تنظيم الدخول إلى المنطقة الحرجة الخاصة بالعمليات، وبربطه مع حل المنوم والمنبه، يُمكن القول أن هذا الحل يهدف إلى تتبع وحفظ عدد إشارات الإيقاظ، وذلك لغرض الفصل فيها لاحقًا، وهناك نوعان من متغير الإشارة S:

- متغير الإشارة الثنائي: والذي يحمل قيمتان فقط 0 أو 1، ويُستخدم للتحكم في تحقيق المنع التبادلي.
- متغير الإشارة العدَّاد: والذي يحمل قيمة صحيحة موجبة تزداد بمقدار 1 عند كل إصدار لإشارة إيقاظ، بمعنى لو كانت قيمته 0 فسيدل على عدم وجود إشارة إيقاظ مخزنة، بينما لو كانت قيمته 1 أو أكثر فسيدل ذلك على وجود إشارة إيقاظ أو أكثر مخزنة وغير مفصول فيها بعد.

كذلك هناك إجرائين يُمكن تطبيقهما على هاذين النوعين من متغير الإشارة، وهما تعميم لعملتي المنوم والمنبه السابقتين. تجدر الإشارة هنا إلى أن كل منهما يضم مجموعة من الخطوات التي يجب تنفيذها جميعًا كعملية واحدة ومن دون أي تجزئة، وذلك لضمان عدم استخدام أي عملية لهذا المتغير في أثناء استخدامه من قبل عملية أخرى. يتمثل هاذان الإجراءان في الآتي:

- إجراء down(S): تتمثل مهمة هذا الإجراء على متغير الإشارة S في اختبار قيمته، هل هي أكبر من 0 أو لا، إذا كانت أكبر من 0، فسيقوم الإجراء بإنقاصه بمقدار واحد، ومن ثم مواصلة إتمام العملية. أما إذا كانت قيمته صفر فسيؤدي هذا إلى ذهاب العملية إلى النوم.
- إجراء up(S): تقتصر مهمة هذا الإجراء في زيادة قيمة متغير الإشارة S، وفي حالة وجود عملية أو أكثر في حالة النوم وغير قادرة على إتمام إجراء down سابق، فإن إجراء up

سيتسبب أيضًا في اختيار إحدى هذه العمليات (على سبيل المثال، الموجودة في بداية طابور الانتظار) لكي تُنشَّط، وتُفعَّل من جديد لاستكمال عملها.

يُستخدم متغير الإشارة الثنائي كما أشرنا آنفًا في التحكم في الدخول إلى المنطقة المشتركة للعمليات، وللقيام بذلك تتبع كل عملية راغبة في الدخول إلى منطقتها الحرجة التسلسل التالي من التعليمات:

1. تُنفذ العملية إجراء **down** على متغير الإشارة الثنائي، والذي سيختبر قيمته، إذا كانت **1**، فسيتم إنقاصها وسيسمح للعملية بالمضُي قدمًا في الدخول إلى منطقتها الحرجة، أمَّا إذا كانت صفر، فستُعلق العملية وستُضاف إلى نهاية طابور انتظار الإشارة.
2. إذا سُمِح للعملية بالدخول إلى منطقتها الحرجة، فستُنفذها.
3. بعد الإنتهاء من تنفيذ المنطقة الحرجة، تستدعي العملية إجراء **up** الذي سيُضيف **1** إلى متغير الإشارة، وسيُعيد تنشيط العملية المنتظرة في مقدمة طابور انتظار الإشارة.
4. الاستمرار في تنفيذ التسلسل المعتاد للتعليمات بحيث تصبح العملية المعاد تنشيطها العملية الحالية.

كما أشرنا سابقًا تُنفذ الخطوات الخاصة بكل من إجراء **down**، وإجراء **up** بصورة غير مُجزئة وكعملية واحدة. الطريقة الإعتيادية لإنجاز ذلك تتمثل في تنفيذهما على أساس استدعاء نظام، مع جعل نظام التشغيل يُعطِّل ولفترة وجيزة كافة المقاطعات في أثناء اختباره وتحديثه لقيمة متغير الإشارة، ووضع العملية في حالة النوم إذا تطلب الأمر ذلك. تعطيل المقاطعات هذا لن يترتب عنه أي ضرر، لأنه يتطلب فقط بضعة تعليمات. أمَّا إذا كان الأمر متعلق باستخدام عدة وحدات معالجة مركزية، فيجب على النظام حماية كل متغير إشارة بواسطة متغير قفل مع استخدام أمر اختبار وضبط القفل، أو أمر الاستبدال للتأكد من أن عملية اختبار متغير الإشارة تتم فقط من قبل وحدة معالجة مركزية واحدة في كل مرة. فيما يلي سيُوضح منظم دخول العمليات من خلال مشكلة المنتج والمستهلك.

حل مشكلة المنتج والمستهلك باستخدام منظم دخول العمليات

لفهم حل مشكلة المنتج والمستهلك المشار إليها في الشكل 2. **18** باستخدام متغير

الإشارة، يُمكن الرجوع إلى الشكل 2. 20، والذي من خلاله نلاحظ أن هذا الحل يستخدم ثلاثة متغيرات من نوع متغير الإشارة: الأول يُدعى **emptyCount** ويُستخدم لتتبع الأماكن الخالية داخل المخزن، ويُهيأ وفقًا لحجم المخزن اللحظي، والثاني يُدعى **fullCount** ويُستخدم لعد الأماكن المشغولة داخل المخزن وقيمته البدائية صفر، وهما من متغير الإشارة العدَّاد، والمتغير الأخير من نوع متغير الإشارة الثنائي، ويُدعى **mutex** ويُستخدم لمنع المنتج والمستهلك من دفع أو إخراج العناصر إلى– أو من المخزن اللحظي في نفس الوقت، ويحمل واحد كقيمة بدائية. ولضمان تحقيق المنع التبادلي يتحتم على كل من المنتج والمستهلك القيام بعملية **down** مباشرةً قبل الدخول إلى الجزء الحرج، وعملية **up** مباشرةً بعد الخروج منه.

```
Producer:
{
 Loop forever{
   down(emptyCount)// Decrement number of free slots
   down(mutex)     // Enter critical section
   Insert item     // Critical section
   up(mutex)       // Leave critical section
   up(fullCount)   // Increment number of busy slots
 }
}

Consumer:
{
 Loop forever{
   down(fullCount)  // Decrement number of busy slots
   down(mutex)      // Enter critical section
   Remove item      // Critical section
   up(mutex)        // Leave critical section
   up(emptyCount)   // Increment number of free slots
 }
}
```

الشكل 2. 20: حل مشكلة المنتج والمستهلك باستخدام متغير الإشارة.

4.2 الخيوط

أشرنا في بداية هذا الفصل إلى أنَّ لكل عملية فضاء عنونة خاص بها وهي تُمثل وظيفةً أو برنامجًا يُمكن أن يُنفذه الحاسوب. فبالنظر إلى تطبيق معالج النصوص والذي يُمثل عملية يُمكن

تشغيلها على الحاسوب، نجد أنَّه قادر على القيام بعدة مهام في كل مرة يُنفذ فيها، كإبراز الكلمات التي بها أخطاء إملائية في أثناء الكتابة، أو الإكمال التلقائي للكلمات عند إدخالها، أو غيرها من المهام الأخرى. هذا يعني أن مثل هذه التطبيقات من الممكن تفكيكها إلى عدة عمليات خفيفة الوزن تعمل في نفس فضاء عنونة العملية الأم، والتي أُطلق عليها اسم الخيوط[6]، بحيث تشترك جميعها في التعليمات البرمجية، والمصادر الخاصة بالعملية الأم، كما هو موضح في الشكل 2. 21.

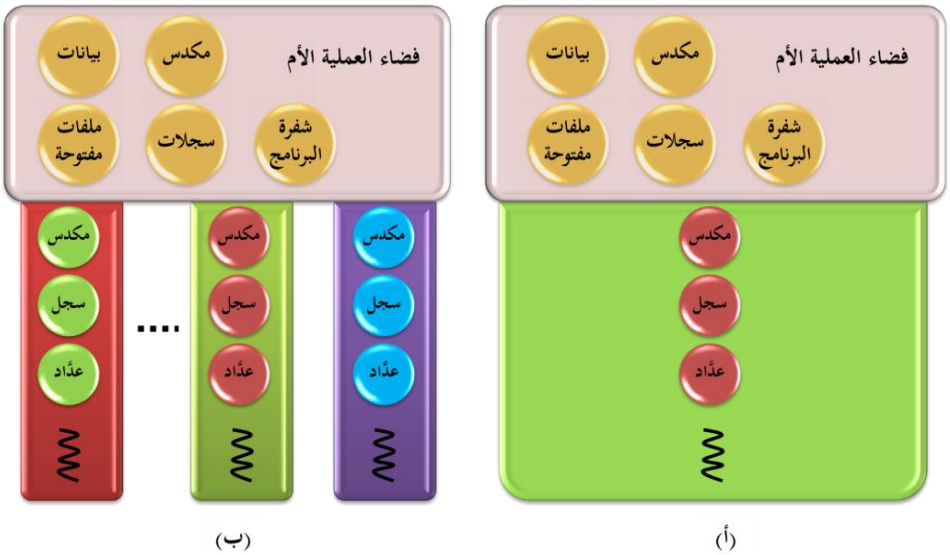
الشكل 2. 21: مشاركة الخيوط لجميع محتويات فضاء العنونة الخاصة بالعملية الأم.

ينتمي كل خيط إلى عملية واحدة فقط، ولا يمكن له أن يوجد خارجها، لذلك كما هو الحال مع العمليات، لدى كل خيط مُعرَّف فريد يُميزه عن بقية خيوط العملية، كما أنه يمتلك سمات

[6]السر وراء استخدام اسم الخيط هنا للدلالة على تداخل العمليات الصغيرة وتكاملها مع بعضها البعض، كما تتداخل خيوط النسيج مع بعضها البعض لتُكون النسيج المتكامل.

خاصة به مثل، حالة التنفيذ، ومكدس التنفيذ، ومؤشر لأب الخيط، وكذلك مساحة تخزينية خاصة به تُستخدم لمتغيراته المحلية. بالإضافة إلى ذلك لكل خيط تركيبة بيانية تُعرف باسم مُوصَّف الخيط، ويستخدمها نظام التشغيل لتخزين كل البيانات المتعلقة به، منها على سبيل المثال مُعرَّف الخيط، والعملية المالكة له، وقائمة بالخيوط المرتبطة معه، وأولويته، وكذلك قائمة المصادر الخاصة به، تُحفظ أغلب هذه المعلومات في ما يُعرف بجدول الخيط والذي تستخدمه العملية لتتبع الخيوط التابعة لها.

يُوضح الشكل 2. 22–أ عملية ذات خيط واحد لها سجلات نظام خاصة بها ومكدس وبيانات وملفات تساعدها في تنفيذ هذا الخيط. بالمقابل يوضح الشكل 2. 22–ب الحالات التي تتواجد فيها عدة خيوط تتحكم في نفس فضاء العنونة وتشتغل بطريقة شبه متوازية، كما لو كانت (تقريبًا) عمليات منفصلة (باستثناء فضاء العنونة المشترك). لاحقًا سيُرجع إلى هذا الموضوع بنوع من التفصيل.



الشكل 2. 22: أ) نموذج العملية ذات الخيط الواحد– ب) نموذج العملية ذات الخيوط المتعددة.

الاختلافات بين الخيوط والعمليات

بالرغم من أن العمليات والخيوط تعمل معًا، إلَّا إنَّه هناك الكثير من الاختلافات بينهما. فالعمليات عادةً ما تكون ثقيلة في التنفيذ إلى حد ما مثل معالج النصوص، بينما تكون الخيوط أخف، كخيار الحفظ الخلفي في أثناء عمل معالج النصوص. يُوضح الجدول 2. 7 بعض من هذه الاختلافات.

الجدول 2. 7: بعض من الاختلافات بين العمليات والخيوط.

العمليات	الخيوط
تحتاج إلى مصادر نظم التشغيل عندما تحدث عملية تبديل للعملية.	لا تحتاج إلى هذه المصادر عند حدوث عملية تبديل للخيط.
إذا حُظِرت أي عملية، تستمر بقية العمليات في عملها.	إذا حُظِر أي خيط على مستوى المستخدم، تُحظر جميع الخيوط الأخرى.
كل عملية تستخدم نفس الشفرة ولها ذاكرة منفصلة.	بإمكان كافة الخيوط مشاركة الملفات، والعمليات المستحدثة كأبناء، وكذلك الذاكرة.
يُعامل نظام التشغيل العمليات المختلفة بشكل مختلف.	يُتعامل مع جميع خيوط مستوى المستخدم كمهمة واحدة لنظام التشغيل.
العمليات مستقلة عن بعضها.	توجد الخيوط كمجموعات فرعية من العملية وغير مستقلة.
أي تطبيق له عدة عمليات سوف يستخدم مصادر النظام بشكل أكثر.	العمليات التي تستخدم عدة خيوط تستخدم مصادر النظام بشكل أقل.
تتطلب تبديل السياق.	نسبيًا، لا تتطلب تبديل السياق.
كل عملية تتعامل مع بياناتها الخاصة بها فقط.	بإمكان الخيوط الوصول إلى بيانات الخيوط الأخرى.
يتطلب الاتصال بين العمليات بعض من الوقت.	يتطلب الاتصال بين الخيوط وقتًا أقل من العمليات.

أنواع الخيوط

هناك نوعان عامان من الخيوط، يُستخدم النوع الأول على مستوى المستخدم، بينما يُستخدم النوع الثاني على مستوى النواة. يتناول هذا الجزء هذان النوعان بنوع من الإيجاز، جنبا إلى جنب مع مزايا وعيوب كل نوع، على أن يُفصلا لاحقًا.

أولًا: الخيوط على مستوى المستخدم

تُدار خيوط المستخدم على مستوى التطبيق دون تدخل من النواة باستخدام مكتبة الخيوط (سيُناقش القسم 3.4.2 هذا النوع من المكاتب)، لذا لا يحتاج تبديل الخيوط إلى استدعاء نظام التشغيل، والتسبب في مقاطعة النواة. يبدأ التطبيق في التنفيذ كعملية باستخدام خيط أساسي، ومن ثم يُمكن للعملية بعد ذلك إنشاء خيوط جديدة وتمرير التحكم إلى إحدها للتنفيذ، كما يُمكنها عبر هذه المكتبة استدعاء الوظائف المناسبة لمختلف مهام إدارة الخيوط مثل، تعليق الخيط، أو حتى إنهائه.

من المزايا الأكثر وضوحًا التي تقدمها خيوط المستخدم هي إمكانية تنفيذها حتى على نظام تشغيل لا يدعم الخيوط، لأنها لا تتطلب إحداث أي تغيير على مستوى هذا النظام. هذا يعني بساطة إدارة هذا النوع من الخيوط، لأن إنشاء الخيوط، والتبديل بينها، ومزامنتها يمكن القيام به دون تدخل النواة، الأمر الذي سينعكس إيجابًا على السرعة والفاعلية في القيام بذلك، لأن الأمر لن يكون أكثر كلفةً من استدعاء وظيفة.

بالمقابل يُعاني هذا النوع من الخيوط من قلة التنسيق بينها وبين نواة نظام التشغيل، بمعنى أن هذا النظام يتعامل مع العمليات من حيث تخصيص شريحة زمنية لها بنوع من التكافؤ بغض النظر عن عدد الخيوط التي تحويها كل عملية، فالأمر متروك هنا لكل خيط للتنسيق ونقل التحكم لبقية خيوطها. قد يحدث أيضًا حظر للعملية برمتها في النواة، حتى إذا كانت هناك خيوط قابلة

للتشغيل داخلها، لأن النواة غير مدركة حتى لوجودها، كحظر عملية بسبب حدوث خطأ الصفحة[7] الناتج عن أحد خيوطها. مثل هذه الحالات قد تؤدي إلى اتخاذ قرارت خاطئة من قبل نظام التشغيل مثل، جدولة عملية بها خيوط خاملة أو محظورة. لذلك قد يتطلب الأمر الاتصال بين النواة وإدارة الخيوط على مستوى المستخدم للتغلب على مثل هذه المشاكل.

ثانيًا: الخيوط على مستوى النواة

تحتفظ النواة في هذه الطريقة بجميع المعلومات الخاصة بالعملية وخيوطها في الموصَّفات التي سبق ذكرها، كما أنها تحتفظ بكل من جداول الخيوط، وجداول العمليات التقليدية لتتبع الخيوط، والعمليات على التوالي، وذلك لغرض إدارتهم من قبل النواة بالشكل الجيد. بالتالي عندما تحتاج العملية إلى خدمات الخيوط عليها طلب ذلك من خلال واجهة استدعاء النظام الخاصة بالخيوط والمتاحة في النواة.

تتمثل إحدى مزايا خيوط النواة في عدم حظر العملية بسبب أن أحد خيوطها قد حُظِر نتيجةً لحدث معين مثلما يحدث مع خيوط المستخدم، كما أنه إذا حُظِر أي خيط، فيمكن للنواة جدولة خيط آخر من نفس العملية. ميزة أخرى ونظرًا لأن لدى النواة معرفة كاملة بجميع خيوط العمليات، فقد يُقرر المجدول إعطاء فترات زمنية متفاوتة لتنفيذ العمليات استنادًا على عدد الخيوط التي تحويها كل عملية، أي إعطاء زمن أكثر للعمليات التي تحتوي على عدد أكبر من الخيوط، مقارنة بالعمليات التي تحتوي على عدد أقل، يُتيح هذا النوع من الخيوط إمكانية

[7] نوع من القفز يُخبر نظام التشغيل أن الصفحة ليس لها وجود في الذاكرة الفعلية. ستتم دراسة أخطاء الصفحة في الفصل الخامس. في الوقت الراهن، يكفي أن نعرف أنه يُمكن إعداد أجهزة الحواسيب بطريقة بحيث لا تتواجد كافة البرامج في الذاكرة الرئيسية في آن واحد. إذا استدعى البرنامج أو انتقل إلى تعليمة غير موجودة في الذاكرة، فسيحدث ما يُعرف بخطأ الصفحة وسيتواصل نظام التشغيل مع القرص ويجلب التعليمة غير المتوفرة منه. هذا يجعل العملية تتوقف ريتما تُحدد وتُقرأ التعليمة الضرورية.

جدولتها على عدة معالجات. أخيرًا، تتناسب خيوط النواة بشكل جيد مع التطبيقات التي يتكرر فيها حظر الخيوط.

من ناحية أخرى تُساهم خيوط النواة في زيادة حجم النواة وإضافة تعقيدات لها من حيث إدارة هذه الخيوط، لأنها تتطلب تحكم كامل في كل خيط الأمر الذي يحتاج إلى الحفاظ على معلومات حول كل الخيوط، ونتيجة لذلك، يكون هناك حمل عام كبير، كما أن الاحتياجات إلى استخدام استدعاءات النظام تجعل خيوط النواة بطيئة وغير فعالة مقارنة بالخيوط على مستوى المستخدم، والتي هي أسرع منها مئات المرات.

1.4.2 استخدامات الخيوط وفوائدها

في الواقع، هناك عدة أسباب وراء الرغبة في استخدام الخيوط لما لها من فوائد عدة. ذكرنا فيما سبق أن في العديد من التطبيقات، هناك عدة أنشطة تُنفذ في وقت واحد، بعض منها قد يتعرض للتوقف من وقت لآخر، بالتالي من خلال تحليل مثل هذه التطبيقات إلى عدة خيوط متتابعة تعمل بشكل شبه متوازي، يمكن أن يُساهم هذا في تبسيط نموذج البرمجة. بالتالي يُمثل كل خيط تدفق منفصل للتحكم في تنفيذ العمليات، وهو ما يجعل لدى مجموعة الخيوط هذه عدة فوائد نذكر منها ما يلي:

- تُوفر الخيوط أساسًا مناسبًا للتنفيذ المتوازي للتطبيقات على المعالجات المتعددة للذاكرة المشتركة.
- تعمل خيوط المعالجة على تقليل زمن تبديل السياق بينها مقارنة بسياق معالجة العمليات والتي تزيد من نفقات وحدة المعالجة المركزية العامة.
- يُوفر استخدام الخيوط المزامنة داخل العملية نفسها.
- تُسرِّع الخيوط استجابة البرامج، أي إذا فُكِّكت العملية إلى عدة خيوط وأكمل إحدها تنفيذه، فيمكن إرجاع مخرجاته على الفور.
- تسريع تنفيذ العملية من خلال الاستخدام الفعَّال لنظام المعالجات المتعددة، بحيث إذا امتلكت العملية عدة خيوط، فبالإمكان جدولتها على هذه المعالجات، وهو ما سيجعل التوازي الحقيقي أمرًا ممكنًا.

- تُمكن الخيوط من مشاركة المصادر بينها ضمن العملية الواحدة مثل، التعليمات البرمجية، والبيانات، والملفات.
- سهولة التواصل بين الخيوط لأنها تتعامل مع فضاء عنونة مشترك.
- تُساهم في تحسين إنتاجية النظام، لأنه إذا قُسِّمت أي عملية إلى عدة خيوط، وأُعتبِر كل منها وظيفة فذلك سيزيد من عدد الوظائف المنجزة في الوحدة الزمنية الواحدة.

أمثلة واقعية لاستخدامات الخيوط

من السهل جدًا أن نلمس أهمية الخيوط كذلك من خلال النظر في بعض الأمثلة الواقعية. يتمثل المثال الأول– والذي تعرضنا له سابقًا– في النظر إلى تطبيق معالج النصوص. عادة ما يعرض هذا التطبيق مستند جاري إنشاؤه على الشاشة بطريقة منسقة كتلك التي يظهر بها على الصفحة المطبوعة. على وجه الخصوص، جميع فواصل الأسطر، وفواصل الصفحات تكون في مواقعها الصحيحة والنهائية، بحيث يمكن للمستخدم تفحصها وإحداث تغيير في المستند إذا لزم الأمر.

إذا أحدث المستخدم أي تغيير في هذا المستند كإلغاء، أو إضافة كلمة، أو جملة، أو فقرة، أو أي جزء منه، فستُساعد الخيوط في إنجاز مهمة إعادة تنسيق المستند بالكامل بشكل أفضل. لنفترض أن معالج النصوص مكتوب كبرنامج متكون من خيطين اثنين. يتفاعل إحداها مع المستخدم، ويُنفذ الآخر مهمة إعادة التنسيق في خلفية التنفيذ. بمجرد ما يُحذف، أو يُضاف الجزء المراد حذفه، أو إضافته، يُخبر خيط التفاعل خيط إعادة التنسيق لإعادة تنسيق بقية المستند. وفي الوقت نفسه، لا يزال خيط التفاعل مستمر في التنصت على لوحة المفاتيح، والفأرة للإستجابة للأوامر الصادرة من المستخدم، في حين يُجري الخيط الآخر الحسابات بجنون في الخلفية.

بينما نحن نتعامل مع هذا الوضع، لماذا لا يُضاف خيط ثالث؟ فالعديد من معالجات النصوص لديها ميزة الحفظ التلقائي للملف بأكمله في القرص كل بضعة دقائق، لحماية المستخدم من فقدان عمله الذي قام به طيلة اليوم الواحد في حال وقوع حادث عرضي للبرنامج، أو للنظام، أو في حالة انقطاع التيار الكهربائي عن الجهاز. بالتالي يمكن للخيط الثالث التعامل

مع الحفظ التلقائي في القرص دون التداخل مع الخيطين الأولين.

استخدام الخيوط على هذا النحو سيؤثر بالتأكيد إيجابًا في تحسين أداء التطبيق وتسريعه، وعدم تعقيد النموذج البرمجي للتطبيق، على العكس مما لو كان البرنامج مبني على خيط واحد فقط. بالتالي فإن استخدام ثلاثة خيوط يجعل نموذج البرمجة أبسط: الخيط الأول يتفاعل فقط مع المستخدم، والخيط الثاني يُنسق المستند عندما يُطلب منه ذلك، أما الخيط الثالث فسيكون مسؤول عن الحفظ التلقائي بشكل دوري. يظهر هذا الوضع المكون من ثلاثة خيوط في الشكل 2. 23.

الشكل 2. 23: تطبيق معالج نصوص بثلاثة خيوط.

بناءً على ما سبق ذكره من الواضح أن استخدام الخيوط في مثل هذه الحالات يُساهم بشكل كبير في إنجاح مثل هذه المهمات، لأنها تشترك جميعها في نفس فضاء الذاكرة والذي

سيجعلها قادرة على العمل على نفس المستند المحرر. أمَّا في حالة استخدام عمليات منفصلة فالأمر لن يكون مجديًا، لأنه من غير الممكن التشارك في هذا المستند.

من الأمثلة الواقعية الأخرى التي يُستدل بها على فائدة الخيوط هي جداول البيانات الإلكترونية، والتي يحتوي جزء منها على بيانات مدخلة من قبل المستخدم، والجزء الآخر يُحتسب من العناصر المدخلة عن طريق استخدام صيغ رياضية. عندما يتغير أي عنصر من البيانات المدخلة، قد يتطلب الأمر إعادة حساب عدة عناصر أخرى. بالتالي من خلال وجود خيط خلفي يُعيد الحسابات، يمكن للخيط التفاعلي الأول أن يسمح للمستخدم بإجراء تغييرات على البيانات المدخلة، في حين أن الحسابات لازالت جارية. بالمثل، يمكن للخيط الثالث التعامل مع الحفظ التلقائي الدوري للقرص من تلقاء نفسه.

بالرغم من أن الخيوط غالبًا ما تكون مفيدة، إلَّا إنَّها هي الأخرى تُضيف بعض من التعقيدات إلى نموذج البرمجة. .بالنظر إلى الآثار المترتبة عن استدعاء الإجراءات التي ينشأ عنها تفرع العمليات. إذا كان لدى العملية الأب عدة خيوط، هل ينبغي على الابن أن يمتلكها كذلك؟ إن لم يكن كذلك، قد لا تؤدي العملية وظيفتها بشكل صحيح، لأن كل واحد منها قد يكون مهمًا. ومع ذلك، إذا تحصلت العملية الابن على عدد من الخيوط بنفس عدد خيوط الأب، فالأسئلة التي ستطرح نفسها: ماذا سيحدث إذا توقف خيط من خيوط الأب في استدعاء القراءة مثلًا من لوحة المفاتيح؟ هل سيتوقف الخيطين؟ عندما تُدخل البيانات، هل سيتحصل كلاهما على نسخة من ذلك؟ هل الأب فقط؟ هل الابن فقط؟

بعض من المشاكل الأخرى تتعلق بحقيقة أن الخيوط تتشارك في فضاء العنونة والعديد من المصادر الأخرى. بالتالي ماذا سيحدث إذا أغلق أحد الخيوط مصدر، بينما هناك خيط آخر لازال يقرأ منه؟ ماذا سيحدث إذا لاحظ أحد الخيوط أن هناك مساحة ذاكرة صغيرة جدًا وبَدأ في تخصيص المزيد منها، في أثناء ذلك أُستبدل الخيط، ولاحظ الخيط الجديد كذلك أن هناك مساحة ذاكرة صغيرة، وبَدأ أيضًا في تخصيص المزيد منها. من المحتمل هنا أن تُخصص الذاكرة مرتين. يمكن حل هذه المشاكل مع بعض من الجهد، ولكن هناك حاجة إلى التفكير والتصميم الدقيق لجعل برامج الخيوط المتعددة تعمل بشكل صحيح.

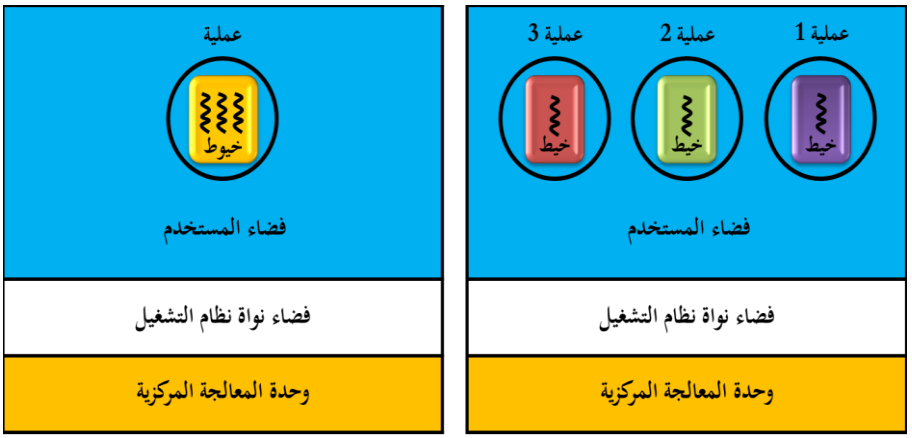
2.4.2 نموذج الخيط النمطي (التقليدي)

أشرنا في الجدول 2. 1 إلى أن العملية تحتوي على عدد من المصادر ذات الصلة مع بعضها البعض مثل، فضاء العنونة، والذي يضم الملفات المفتوحة، ومعلومات الحساب، بالإضافة إلى سجلات التحكم في التنفيذ كعدَّاد البرنامج، والمكدس، وقائمة الخيوط التابعة للعملية، بالتالي يُمكن النظر إلى العمليات على أنها مجموعة من المصادر ذات الصلة. يُساعد هذا النموذج في جعل العملية تُدار بسهولة كبيرة، ويسمح لها بالتحكم في الخيوط التابعة لها، والتي يمتلك كل واحد منها عدَّاد برنامج– لتتبع سير التعليمات التالية في التنفيذ– كما أن له سجلات لاستعاب متغيرات العمل الحالية، كذلك لديه مكدس يحتوي على تسلسل التنفيذ، مع إطار واحد لكل إجراء، كما هو موضح في الشكل 2. 22. وعلى الرغم من أن الخيط يجب تنفيذه في جزء من العملية، إلَّا إنَّ لدى الخيط والعملية مفاهيم مختلفة سُرِدت في الجدول 2. 7، فالعمليات تُستخدم لتجميع المصادر معًا، بينما تُمثل الخيوط الكيانات المجدولة للتنفيذ على المعالج.

الأمر الذي أضافته الخيوط إلى نموذج العملية هو السماح بالتنفيذ المتعدد ليأخذ مكانه في نفس بيئة العملية إلى درجة كبيرة من الاستقلالية عن بعضها البعض. بمعنى وجود عدة خيوط تعمل بصورة موازية داخل نفس العملية، وهو ما يُماثل وجود عدة عمليات تشتغل في نفس الوقت في نفس جهاز الحاسوب، إلَّا إنَّ التبديل بين الخيوط أقل تكلفة بكثير من التبديل بين العمليات المنفصلة. في الحالة الأولى، تتشارك الخيوط في فضاء العنونة وغيرها من المصادر ولا توجد حماية للذاكرة بينها، لأنه عادةً لا توجد بينها مشكلة أمنية تتطلب هذه الحماية، فالخيوط تتعاون فيما بينها لأنها تُمثل نفس المستخدم، أي نفس العملية، أمَّا في الحالة الثانية، فتتشارك العمليات في الذاكرة الفعلية، والأقراص، والطابعات، وغيرها من المصادر والتي تتطلب نوع من التنافس.

بناءً على ما تقدم ذكره يمكن أن يكون هناك نموذجين للعملية: النموذج التقليدي، والذي يكون فيه للعملية فضاء عنونة خاص بها يتواجد فيه خيط تحكم واحد، كما هو الحال في الشكل 2. 24–أ، والذي يُبين ثلاث عمليات تقليدية من هذا النوع، لكل منها خيط يعمل في فضاء منفصل عن الآخر. النموذج الآخر هو نموذج الخيوط المتعددة والموضح في الشكل 2. 24– ب، والذي يحوي عملية واحدة ذات ثلاثة خيوط للتحكم، تشترك جميعها في نفس فضاء العنونة. بالتالي يُستخدم النموذج الأول عندما تكون العمليات الثلاث غير مرتبطة ببعضها أساسًا،

في حين يُستخدم النموذج الأخير عندما تكون الخيوط الثلاثة جزء من نفس العمل وتتعاون بنشاط وبشكل وثيق مع بعضها البعض.



(أ) (ب)

الشكل 2. 24: أ) ثلاث عمليات، لكل واحدة منها خيط واحد– ب) عملية واحدة بثلاثة خيوط.

تُعتبر عملية تنفيذ عدة خيوط على نظام يمتلك معالج مركزي واحد مشابهة تمامًا لتناوب عدة عمليات على هذا المعالج، والذي سينتقل ذهابًا وإيابًا بشكل سريع بينها، الأمر الذي سيُعطي الوهم بأن العمليات التتابعية المنفصلة تُنفذ على التوازي. كذلك الأمر بالنسبة لخيوط العملية الواحدة، تنتقل وحدة المعالجة المركزي بينها بنفس الآلية، وبذلك تُعطي هي الأخرى الوهم بأنها تشتغل بشكل متواز، حتى ولو تم ذلك على وحدة معالجة مركزية منفصلة، التي هي أبطأ من الوحدة الحقيقية. بالتالي بوجود ثلاثة خيوط حسابية ضمن العملية الواحدة، سوف يُنفذ كل واحد منها من قبل هذه الوحدة بثلث سرعتها الحقيقية.

في إطار نموذج الخيوط المتعددة ترتبط الخيوط المختلفة مع بعضها ضمن العملية الواحدة على العكس من العمليات المختلفة التي غالبًا ما تكون مستقلة، السبب وراء ذلك يكمن في أن الخيوط تمتلك جميعها بالضبط نفس فضاء العنونة، الأمر الذي يعني كذلك أنها تشترك في نفس المتغيرات العالمية. هذا يعني أيضًا أن كل خيط يمكنه الوصول إلى كل عنوان في الذاكرة ضمن نفس هذا الفضاء، أي كل خيط يُمكنه قراءة، أو كتابة، أو حتى مسح أي جزء مشترك بينها، حتى

ولو كان ملك لخيط آخر. فالحماية هنا أمر غير مطلوب: أولًا لأنها مستحيلة، وثانيًا لاحتياج الخيوط إلى التعاون فيما بينها، لا أن تتخاصم من أجل القيام بالمهمة المناطة بهم، وثالثًا لأنها مملوكة لنفس المستخدم، على عكس العمليات المختلفة، التي قد تكون من مختلف المستخدمين، والتي قد تكون معادية لبعضها البعض.

كما هو الحال في العملية التقليدية (أي عملية من خيط واحد فقط)، تمر الخيوط بنفس الحالات التي تمر به العمليات والتي شاهدناها في الشكل 2. 5، أي يمكن للخيط أن يكون في أي حالة من عدة حالات: تشتغل، أو موقوفة، أو جاهزة، أو منتهية. كذلك الحال بالنسبة لانتقالات الخيوط بين هذه الحالات تحدث بنفس الكيفية المبينة في هذا الشكل، وعلى النحو الموضح في الجدول 2. 3. من المهم أن نُدرك أيضًا أن لكل خيط مكدسه الخاص به يستخدمه في التحكم في استدعاء إجراءات التنفيذ، ولأن لكل خيط الحق في أن يستدعي إجراءات مختلفة، سيكون لدى كل خيط تسلسل تنفيذ مختلف، وهو السبب وراء أن لكل خيط مكدس خاص به.

استدعاءات تنفيذ الخيوط

يبدأ تنفيذ العمليات بخيط واحد في نموذج الخيوط المتعددة، والذي سيكون له القدرة على استدعاء عدة إجراءات مكتبية صُمِّمت بالخصوص نسرد منها ما يلي:

- استدعاء إجراء **thread_create**: يُستخدم هذا الإجراء لإنشاء خيوط جديدة والتي ستعمل تلقائيًا في نفس فضاء العنونة للخيط المنشأ. قد يأخذ هذا الإنشاء في بعض الأحيان الشكل الهرمي، أي بمعنى أن هناك علاقة الأباء، والأبناء، وقد تتساوى في كثير من الأحيان كل الخيوط ولا وجود لمثل هذه العلاقات.
- استدعاء إجراء **thread_exit**: عندما يُنهي الخيط عمله، يستدعي هذا الإجراء، لكي يخرج، ومن ثم يختفي ولن يكون بعدها قابلًا للجدولة من جديد.
- استدعاء الإجراء **thread_join**: في بعض من أنظمة الخيوط، يمكن لإحدها الانتظار من أجل خروج خيط (محدد)، لذلك يُستخدم هذا الإجراء، والذي سيُوقِف الخيط الذي استدعاه إلى أن يخرج الخيط (المحدد).

- استدعاء إجراء **tread_yield**: يسمح هذا الإجراء للخيط بالتخلي طوعًا عن المعالج، وذلك للسماح لخيط آخر بالاشتغال. مثل هذا الاستدعاء مهم، لأنه لا توجد مقاطعات نبضية لفرض تطبيق البرمجة المتعددة كما هو الحال مع العمليات. بالتالي فإنه من المهم للخيوط أن تتنازل عن المعالج طوعًا من وقت لآخر لإعطاء الخيوط الأخرى فرصة الاشتغال.
- هناك استدعاءات أخرى تسمح لأحد الخيوط بانتظار خيط آخر لإنهاء بعض من الأعمال، كما تسمح للخيط بأن يعلن أنه قد أنهى بعض من الأعمال المناطة به، وهلم جرا.

3.4.2 خيوط POSIX

من أجل الاستفادة الكاملة من إمكانيات الخيوط، وجعلها قابلة للتطبيق على أكثر من نظام، أي خيوط قياسية، حدد نظام **IEEE**[8] معيارًا للخيوط هو معيار **1003.1c**، والذي أُطلق على حزمة خيوطه اسم بثريدس **Pthreads**، وهي تندرج تحت خيوط **POSIX** (وهي اختصار لجملة **Portable operating system interface** التي تعني واجهة نظام التشغيل المحمولة) وتُمثل مجموعة من المعايير التي نُفذت بشكل أساسي لأنظمة التشغيل المستندة إلى نظام 'يونكس'، على الرغم من أن بعض منها قابل للاشتغال على أنظمة 'ويندوز'. يُعرِّف هذا المعيار أكثر من **60** وظيفة استدعاء، يُمكن توفيرها إمَّا كمكتبة على مستوى المستخدم أو كمكتبة على مستوى النواة، يستعرض هذا القسم بعض منها. علمًا أن جميع استدعاءات خيوط **Pthreads** لها سمات معينة، وكل خيط يحتوي على المعلمات التالية المرتبطة به:

- مُعرَّف الخيط، الذي يُحدد الخيط بشكل فريد، ويتم إرجاعه عند إنشاء الخيط.
- مجموعة من السجلات (بما في ذلك عدَّاد البرنامج).
- مجموعة من الخصائص المخزنة في بنية الخيط، والتي تشمل حجم المكدس، معلمات الجدولة، وغيرها من العناصر اللازمة لاستخدامه.

[8] **IEEE** – أكبر منظمة في العالم مهنية فنية، ومكرسة لتطوير التكنولوجيا لصالح الإنسانية.

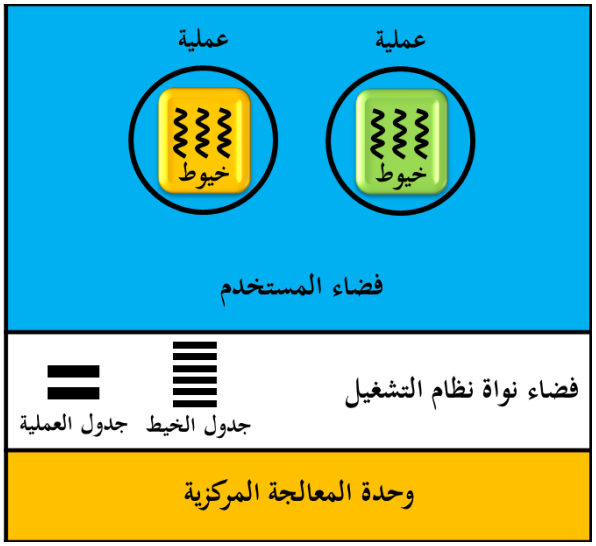
فيما يلي شرح موجز لبعض من هذه الاستدعاءات.

- استدعاء إجراء **pthread_create**: يُستخدم هذا الإجراء لإنشاء خيوط جديدة، ويُرجِّع مُعرَّف الخيط المنشأ حديثًا كقيمة وظيفة. هذا الاستدعاء مشابه إلى حد كبير استدعاء نظام **fork** (باستثناء المعلمات)، بحيث يكون لمعرَّف الخيط دورًا مهمًا كدور مُعرَّف العملية، وهو في الغالب تحديد الخيوط التي تم الرجوع إليها في الاستدعاءات الأخرى.
- استدعاء إجراء **pthread_exit**: عندما يُنهي الخيط عمله، يستدعي هذا الإجراء، والذي سيُنهي الخيط ويُحرر المكدس.
- استدعاء الإجراء **pthread_join**: إذا ما احتاج خيط– ما– قبل أن يُنهي عمله إلى الانتظار من أجل إنهاء خيط آخر لعمله، فبإمكانه استدعاء هذا الإجراء وتمرير مُعرَّف الخيط الخاص بالخيط الذي سيتم انتظاره كمعلمة لكي يخرج.
- استدعاء إجراء **pthread_yield**: أحيانًا يحدث أن خيط– ما– قد لا يُوقف منطقيًا، بالرغم من أنه قد أخذ وقته الكافي، لذلك يجب منح الفرصة لخيط آخر لكي يشتغل. هذا الخيط يمكنه تحقيق هذا الهدف من خلال استدعاء هذا الإجراء. مثل هذا الاستدعاء لا يوجد في حالة استخدام العمليات، لأن الافتراض هناك هو أن العمليات تتنافس بشراسة وكل واحدة منها تريد استغلال وقت وحدة المعالجة المركزية بالكامل كلما أمكنها ذلك. ومع ذلك، وبما أن خيوط العملية تعمل معًا وأن شفرتها كُتبت من قبل نفس المبرمج، يُريد المبرمج أحيانًا أن يُعطي كل خيط الفرصة للخيط الآخر.
- استدعاء إجراء **pthread_attr_init**: يُنشئ هذا الإجراء بنية الخاصية المرتبطة بالخيط ويُهيئه بالقيم الافتراضية مثل، الأولوية، والتي يمكن تغييرها عن طريق معالجة الحقول في بنية الخاصية.
- استدعاء إجراء **pthread_attr_destroy**: يمسح هذا الإجراء بنية خاصية الخيط ويُحرر الذاكرة المخصصة له. علمًا أن هذا الاستدعاء لا يُؤثر على الخيوط التي تستخدمه، بل يُتيح لهم الفرصة للاستمرار في الخروج.

4.4.2 إنجاز الخيوط في فضاء النواة

تفصيلًا لما سبق ذكره في مقدمة القسم **4.2** تتمثل فكرة إنجاز الخيوط في النواة في كونها

تُنفذ في النواة، وتُدار مباشرةً من هناك من قبل نظام التشغيل. تتطلب الإدارة في هذه الحالة معرفة كل ما يتعلق بالعملية والخيوط، لذلك تُلاحظ تواجد كل من جدول العمليات، وجدول الخيوط في النواة، كما هو موضح في الشكل 2. 25. فالجدول الأول تستخدمه النواة في تتبع تنفيذ العمليات وإدارتها ، أمَّا الثاني فتستخدمه في تتبع تنفيذ كافة الخيوط داخل النظام. بالتالي يجب على النواة توفير كافة الإجراءات المتعلقة بالخيوط، بحيث عندما يرغب أي خيط مثلًا في إنشاء خيط جديد، أو إنهاء خيط موجود فإنه يستدعي إجراء خدمة الإنشاء أو الإنهاء داخل النواة والذى بدوره يُحدِّث جدول الخيوط بها.



الشكل 2. 25: حزمة خيوط مدارة من قبل النواة.

كما أشرنا سابقًا، يحتفظ جدول الخيوط هذا بكل من سجلات الخيط، والحالة وغيرها من المعلومات الأخرى، وهي نفس المعلومات الموجودة على مستوى المستخدم، إلَّا إنَّها محفوظة في النواة بدلًا من فضاء المستخدم. هذه المعلومات هي مجموعة جزئية من المعلومات التي تحتفظ بها النواة عن العمليات التقليدية أحادية الخيط. في هذا السياق، عندما يحدث تبديل تنفيذ خيوط النواة نتيجة لتوقف خيط– ما– جراء احتياجه لأحداث خارجية، يمكن للنواة تنفيذ خيط آخر من نفس العملية (إذا كان جاهزًا) أو خيط من عملية آخرى مختلفة. في الحالتين تحفظ النواة بجميع معلومات الخيط الموقوف في قالب التحكم في الخيط المناظر تمامًا لقالب

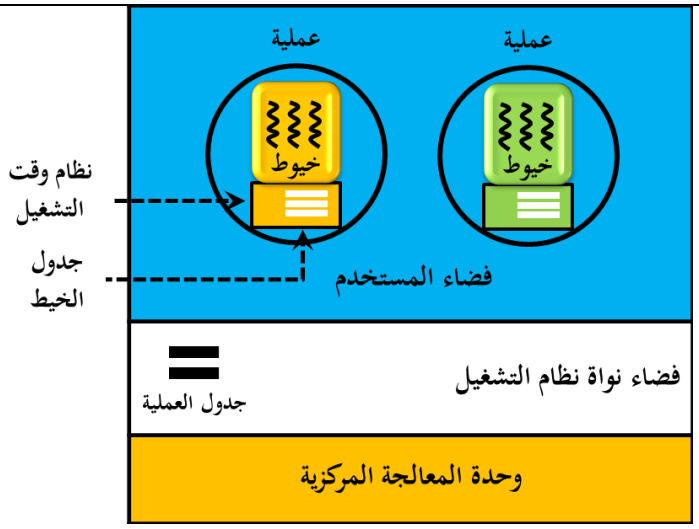
التحكم في العملية، بعدها تختار النواة الخيط الجديد للتنفيذ، وتُحمل كل بياناته اللازمة للتنفيذ، ومن ثم تقفز لمحتوى عدَّاد البرنامج الخاص بالخيط الجديد. هذه الخطوات مشابهة لخطوات تنفيذ العمليات.

من المساوي المأخوذة على تنفيذ الخيوط في النواة هي التعقيد النسبي لإنشاء وإنهاء الخيوط داخلها، لذلك تتجه بعض من النظم إلى التقليل من هذا التعقيد من خلال إعادة تدوير الخيوط الخاصة بها. فعندما ينتهي أي خيط وبدلًا من حذفه من النواة، يُبوَّب على أنه غير قابل للتنفيذ من دون أن تتأثر بنية بياناته فيها. في وقت لاحق وعوضًا عن إنشاء خيط جديد، يُعاد تفعيل واستخدام هذا الخيط.

5.4.2 إنجاز الخيوط في فضاء المستخدم

تتمثل الطريقة الثانية لإنجاز الخيوط في وضع حزمة الخيوط كليًا في فضاء المستخدم دون أي دعم من فضاء النواة، بينما تُدار العمليات بالشكل المعتاد في الفضاء الأخير على أساس عمليات ذات خيط واحد. يُوضح الشكل 2. 26 طريقة الإنجاز هذه. ولأن الخيوط تُدار في هذه الحالة على مستوى المستخدم، تحتاج كل عملية إلى جدول خيط خاص بها، وذلك لتتبع خيوطها. يُماثل هذا الجدول جدول العملية داخل النواة، باستثناء أنه يتتبع فقط الخصائص الخاصة بكل خيط داخل العملية وليس داخل النظام، مثل عدَّاد البرنامج، ومؤشر المكدس، والسجلات، وقالب صغير للتحكم في الخيوط.

تُدار الخيوط في هذه الهيكلية من قبل 'نظام وقت التشغيل' وهو بمثابة مكتبة المستخدم تحوي مجموعة من الإجراءات التي تُدير الخيوط فقط في فضاء المستخدم كإجراء **pthread_create** السابق ذكره. هذا يجعل إنشاء الخيوط، واستبدالها، وكذلك مزامنتها يحدث على مستوى فضاء المستخدم دون أي استدعاءات للنواة، الأمر الذي يجعل خيوط المستخدم أسرع من خيوط النواة مئات المرات، لأنه عندما يُنقل الخيط إلى حالة جاهزة أو حالة موقوفة، فإن المعلومات اللازمة لإعادة تشغيله تُخزن في جدول الخيط والموجود أصلًا في فضاء المستخدم، بنفس الطريقة التي تُخزن بها النواة معلومات العمليات في جدول العملية.



الشكل 2. 26: حزمة خيوط على مستوى المستخدم.

ذكرنا سابقًا بأن الخيوط تتميز بالتخلي الطوعي عن المعالج لأجل إعطاء الفرصة لخيط آخر، لذلك إذا أحدث الخيط أي حدث يستوجب توقفه، فعليه استدعاء أحد إجراءات "نظام وقت التشغيل"، لكي يتحقق هذا الإجراء من معرفة ما إذا كان من الضروري وضع الخيط في حالة التوقف. إذا كان الأمر كذلك، فإنه يُخزِّن بيانات الخيط الخاصة به في جدول الخيط، ومن ثم ينظر في الجدول للبحث عن ما إذا كان هناك خيط جاهز للاشتغال ليقوم إثر ذلك بتحميل بياناته. بمجرد تبديل مؤشر المكدس وعدَّاد البرنامج، تبدأ تلقائيًا مرحلة حياة الخيط الجديد. تجدر الإشارة هنا إلى أن فكرة إعادة تدوير الخيوط التي أشرنا إليها في القسم السابق أيضًا ممكنة على مستوى المستخدم، ولكن لأن تعقيدات إدارة الخيوط أقل بكثير منها على مستوى النواة، فإنه ليس هناك حافز للقيام بذلك.

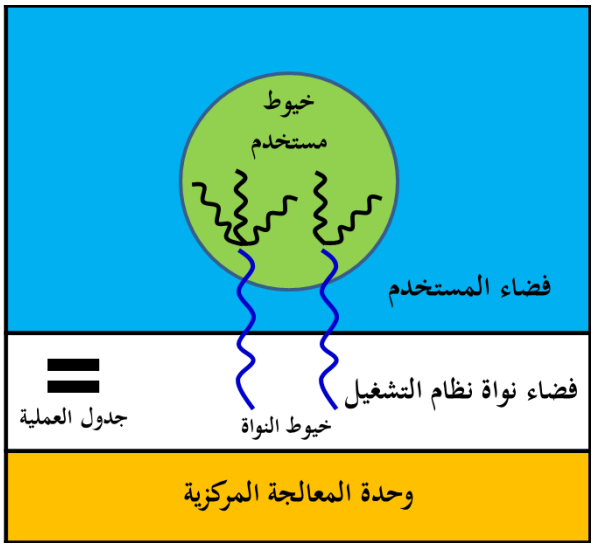
6.4.2 الإنجاز الهجين للخيوط

من الطرق الأخرى لإنجاز الخيوط هو محاولة الجمع ما بين الخيوط على مستوى المستخدم وعلى مستوى النواة، وذلك لغرض الاستفادة من مزايا الخيوط في كلتا الحالتين. تُتيح هذه الطريقة للمبرمج إمكانية تحديد عدد الخيوط الذي سيستخدمه على مستوى النواة وعدد الخيوط على مستوى المستخدم والتي سترتبط بكل منها، أي يُمكن أن تحتوي خيوط النواة

المختلفة في نفس العملية على أعداد مختلفة من خيوط عمليات المستخدم المخصصة والتي تتناوب على استخدامها، كما هو موضح في الشكل 2. 27، الأمر الذي يمنح النظام مرونة كبيرة في التنفيذ.

في هذا الأسلوب تكون النواة على دراية فقط بالخيوط التي على مستواها وبالتالي هي التي تُجدولها، أمَّا الخيوط التي على مستوى المستخدم فسيُتعامل معها من حيث الإنشاء، أو الإنهاء، أو الجدولة كتلك التي تعمل في نظام تشغيل لا يدعم إمكانية الخيوط المتعددة.

يتميز هذا الأسلوب في الإنجاز بسرعة التبديل بين خيوط المستخدم ضمن خيط النواة الواحد (بدون تبديل السياق) بنفس الحالة التي تعمل بها الخيوط على مستوى المستخدم فقط. كما أن التبديل بين خيوط النواة لنفس العملية يتطلب استدعاء النظام بنفس الطريقة المعمول بها في أسلوب الخيوط على مستوى النواة فقط. أيضًا استخدام الإنجاز الهجين للخيوط يُساعد في معالجة مشكلة حظر للعملية برمتها، في حالة توقف أحد خيوطها. مثلًا إذا حدث ومُنع خيط واحد فقط من خيوط النواة، فلا يزال بإمكان التطبيق متعدد الخيوط ككل العمل، لأن خيوط المستخدم التابعة للخيوط الأخرى على مستوى النواة لهذه العملية لا تزال قابلة للتنفيذ.

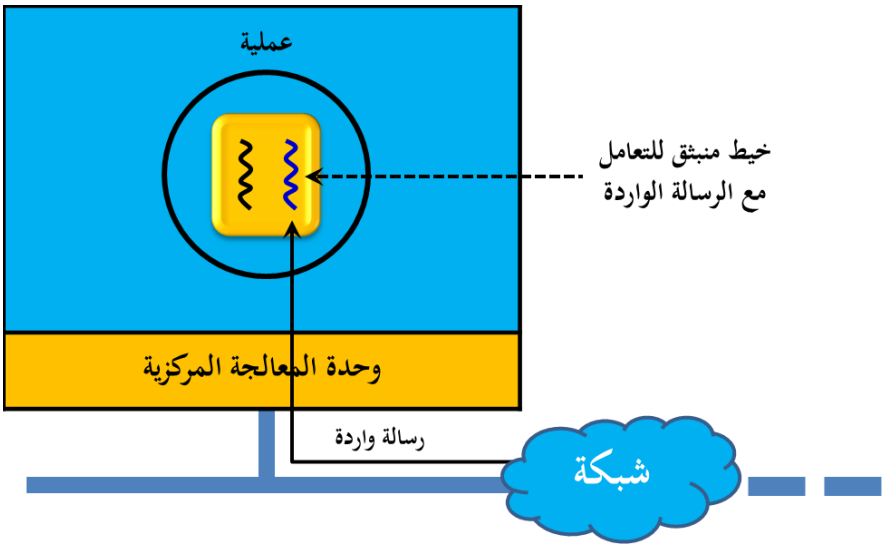


الشكل 2. 27: ربط خيوط المستخدم مع خيوط النواة.

7.4.2 الخيوط المنبثقة

في الطريقة الاعتيادية لأنظمة الشبكات الموزعة عادةً ما تكون هناك عملية استقبال الرسائل في حالة موقوفة تنتظر في حدث يتمثل في استدعا نظام الاستقبال والذي ينتظر في ورود أي رسالة إليه عبر الشبكة، عندما تصل هذه الرسالة وتستلمها العملية، تبدأ مراحل معالجة الرسالة، والمتمثلة في فك شفرتها، واختبارها، ومن ثم معالجتها. بتتبع استخدام هذه الطريقة يُمكن ملاحظة طول الفترة الزمنية ما بين وصول الرسالة وبدأ المعالجة نتيجةً لتبديل سياق العمليات، والذي يتطلب حفظ كل معلومات العملية القديمة، وتحميل معلومات العملية الجديدة.

لتحسين مستوى أداء هذه الطريقة يُمكن جعل الرسالة الواردة عبر الشبكة إلى النظام تتسبب في انبثاق خيط جديد من قبل النظام يتعامل مع هذه الرسالة. يُوضح الشكل 2. 28 فكرة تولد هذا الخيط، والذي يُسمى بالخيط المنبثق. يستلم هذا الخيط الرسالة الواردة وذلك لغرض القيام بخطوات معالجتها، بالتالي تقليل التأخير بين وصول الرسالة وبدء معالجتها بشكل كبير. السر وراء هذا التحسين في كون الخيوط المنبثقة جديدة، أي أن ليس لديها أي حالات أو معلومات سابقة يجب استعادتها، فكل منها يبدأ نشط، الأمر الذي يُساهم في سرعة إنشائها بشكل ملحوظ.



الشكل 2. 28: انبثاق خيط جديد عند وصول رسالة إلى النظام عبر الشبكة.

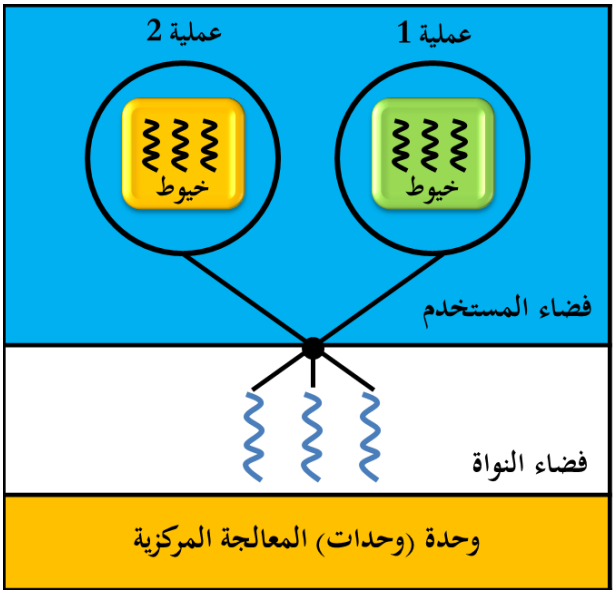
من الملاحظات المهمة التي يجب مراعاتها عند استخدام طريقة الخيوط المنبثقة هو أنها تحتاج إلى بعض من التخطيط المسبق يشمل تحديد العملية التي سيشتغل فيها الخيط المنبثق، وتحديد ما إذا كان هذا الخيط سيشتغل على مستوى النواة، أم على مستوى فضاء المستخدم. تُنفذ الخيوط المنبثقة غالبًا في فضاء النواة لكونها أسهل وأسرع من تنفيذها في فضاء المستخدم، لإمكانية الوصول بسهولة إلى جميع جداول النواة وأجهزة الإدخال، والإخراج والتي قد يُحتاج إليها في معالجة أي مقاطعة، إلَّا إنَّ الأمر قد لا يخلو من العقبات، لأنه بإمكان خيط نواة مشوب بالأخطاء أن يتسبب في أضرار أكثر بكثير من خيط مستخدم، كأن يستمر تنفيذ خيط–ما– لفترة طويلة جدًا من دون أن تكون هناك طريقة– ما– لكبحه، الأمر الذي قد يتسبب في فقدان البيانات الواردة بشكل نهائي.

8.4.2 نماذج الخيوط المتعددة

تُوفر بعض من نظم التشغيل ترابطًا مشتركًا بين الخيوط على مستوى فضاء المستخدم لعمليات المستخدم وعلى مستوى فضاء النواة لعمليات النواة مثلما هو الحال عليه في نظام 'سولاريس'، و'ويندوز'، و'ماك أو إس إكس' و'لينكس'. لهذا الترابط عدة مزايا أهمها أنه يجعل بالإمكان تنفيذ عدة خيوط لنفس التطبيق بالتوازي على عدة معالجات، كما أنه بإمكانه أن يأخذ عدة علاقات شائعة تشمل: علاقة عديد– لعديد، وعلاقة عديد– لواحد، وأخيرًا، علاقة واحد– لواحد.

علاقة عديد– لعديد

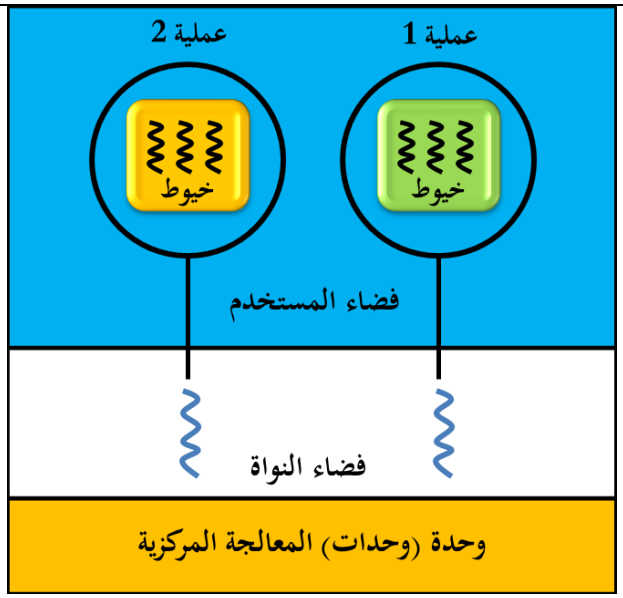
ترتبط خيوط عمليات المستخدم في نموذج علاقة عديد– لعديد بعدد مساو أو أصغر من خيوط عمليات النواة، كما هو موضح في الشكل 2. 29، بالتالي تسمح هذه العلاقة للمطورين بإنشاء عدة خيوط على مستوى المستخدم حسب الضرورة، كما يمكن لخيوط النواة المقابلة لها أن تشتغل بالتوازي على جهاز متعدد المعالجات، الأمر الذي سيزيد من كفاءة النظام. يُوفر هذا النموذج أداء جيد لمفهوم المزامنة بحيث عندما يستدعي خيط نظام التوقف (الحظر)، بإمكان النواة جدولة خيط آخر للتنفيذ.



الشكل 2. 29: نموذج عديد– لعديد.

علاقة عديد– لواحد

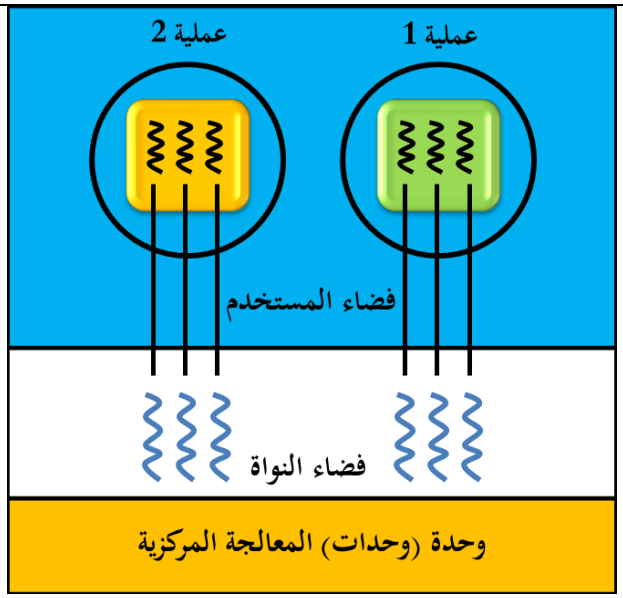
في علاقة عديد– لواحد تُخصص عدة خيوط على مستوى عمليات المستخدم لخيط واحد فقط على مستوى عمليات النواة، كما هو موضح في الشكل 2. 30. تُدار الخيوط في هذا النموذج من قبل مكتبة الخيوط على مستوى المستخدم، كما أنه بإمكان تشغيل خيط واحد على هذا المستوى بسبب وجود خيط واحد يُناظره على مستوى النواة، الأمر الذي يجعل أي عملية تتوقف برمتها إذا قام الخيط باستدعاء نظام التوقف. كذلك يمنع هذا الأمر تشغيل عدة خيوط على التوازي في الأنظمة متعددة المعالجات وهو ما جعل هذا النموذج مستخدم فقط من قبل عدد قليل جدًا من أنظمة التشغيل.



الشكل 2. 30: نموذج عديد– لواحد.

علاقة واحد– لواحد

يتم اسقاط أو تعيين خيط واحد على مستوى عمليات المستخدم في نموذج علاقة واحد– لواحد مقابل خيط واحد مناظر له، ومنفصل على مستوى عمليات النواة، كما هو موضح في الشكل 2. 31، الأمر الذي ينتج عنه مزامنة في التنفيذ بشكل أفضل مقارنة بالنموذج السابق، أي أن هذه العلاقة تسمح لعدد من الخيوط بأن تشتغل بشكل متوازي على عدة معالجات، كما أنه يُمكن السماح لخيط آخر بالاشتغال عند استدعاء خيط لنظام التوقف. تجدر الإشارة هنا إلى إن أغلب تنفيذات هذا النموذج تُقيد عدد الخيوط المولدة، لأن إنشاء خيط مستخدم يتطلب إنشاء خيط نواة مناظر له، وهذا يحد من أداء التطبيقات نتيجة لتعقيدات إنشاء خيوط النواة، وهو ما يُعد عيبًا في هذا النموذج. من أمثلة نظم التشغيل المطبقة لعلاقة واحد– لواحد نظام 'لينكس'، جنبًا إلى جنب مع مجموعة من عائلة أنظمة 'ويندوز'.



الشكل 2. 31: نموذج واحد– لواحد.

ملخص العلاقات الثلاثة

بالنظر إلى تأثير هذه التصاميم على المزامنة، أي التنفيذ على التوازي يُمكن ملاحظة أن نموذج عديد– لواحد يسمح للمطورين بإنشاء العديد من خيوط المستخدم حسب الرغبة، ولكن هذا لا يُفضي إلى المزامنة الحقيقية، وذلك لأن النواة في الوقت نفسه يمكنها جدولة خيط واحد فقط. بالمقابل يسمح نموذج واحد– لواحد بمزامنة أكثر، ولكن على المطورين أن يكونوا حريصين على عدم إنشاء عدة خيوط ضمن التطبيق الواحد لحدوديته في بعض الحالات. من جهة أخرى، لا يعاني نموذج عديد– لعديد من أوجه القصور السالفة الذكر، حيث يمكن للمطورين إنشاء العديد من خيوط المستخدم حسب الضرورة، ويمكن تشغيل خيوط النواة المناظرة بالتوازي على عدة معالجات. إضافة إلى ذلك، عند قيام خيط باستدعاء نظام توقف يمكن للنواة جدولة خيط آخر للتنفيذ.

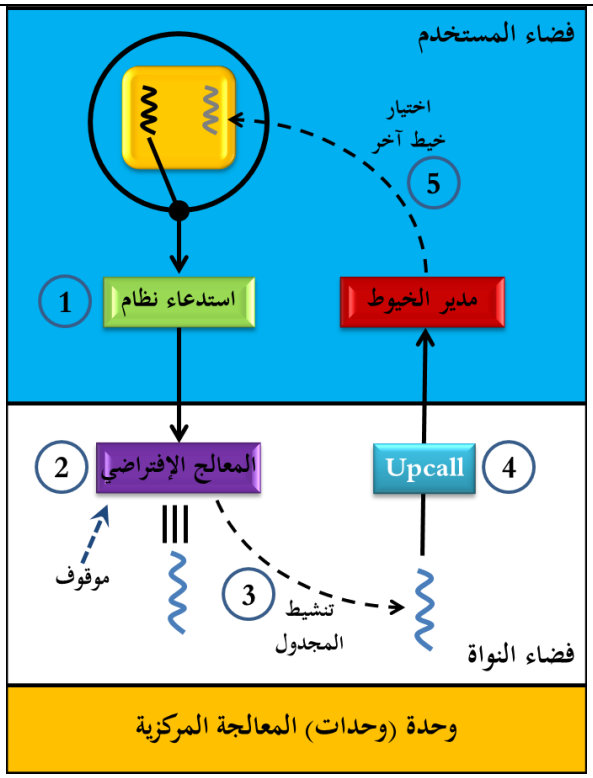
من الممكن دمج أكثر من علاقة في نموذج واحد كدمج نموذج عديد– لعديد مع نموذج واحد– لواحد ليسمح لخيوط المستخدم بأن تكون مقيدة بخيط نواة واحد فقط، يُشار إلى هذا التعديل أحيانًا بالنموذج ذي المستويين.

9.4.2 تنشيط المجدول

بينما يكون استخدام الخيوط على مستوى النواة أفضل من استخدامها على مستوى المستخدم في بعض من التطبيقات التي تستفيد من المعالجات المتعددة، إلَّا إنَّها تُعاني بشكل مؤكد من البطأ. لذلك ابتكر العالم آندرسون وآخرون في سنة 1992موضوع تنشيط المجدول، والمتمثلة أهدافه في محاكاة وظائف خيوط النواة، لكي تُحقق أداءً، ومرونةً أفضل في التعامل مع حزم الخيوط المنجزة على مستوى المستخدم. لتحقيق هذه المحاكاة في الأنظمة التي تستخدم إمَّا نموذج عديد– لعديد أو النموذج ذي المستويين تُوفر النواة للتطبيق مجموعة من المعالجات الافتراضية (في بعض من المراجع يُشار إليها بالعمليات خفيفة الوزن) التي تُمثل خيوط النواة، والتي تُتيح للتطبيق التحكم الكامل في الخيوط التي ستُشغل على هذه المعالجات. يُتحكم في عدد المعالجات الافتراضية داخل المجموعة بواسطة النواة، استجابة للمطالب المتنافسة للعمليات المختلفة في النظام.

في هذا الصدد، قد يتطلب تطبيق– ما– عدد محدد من المعالجات الافتراضية لضمان تشغيله بكفاءة، ولكن وبالنظر إلى تطبيق آخر مقيد بوحدة معالجة مركزية ذات معالج واحد، فإنه في اللحظة الواحدة لا يمكن تشغيل إلا خيط واحد فقط، لذلك يمكن الاكتفاء بمعاج افتراضي واحد. من جهة أخرى، إذا كان التطبيق يُجري عمليات إدخال، وإخراج كثيفة فقد يتطلب الأمر عدة معالجات افتراضية لتشغيله، لأنه عادة ما ينشأ معالج افتراضي جديد عند كل استدعاء متزامن لنظام التوقف. تُخطِر النواة في جميع الأحوال مدير الخيوط على مستوى المستخدم بأحداث النواة الهامة باستخدام الاستدعاء **Upcall**.

يوضح الشكل 2. 32 مثال لهذه الأحداث والمتمثل في استدعاء خيط لنظام التوقف والذي ستُخصص على إثره النواة خيط نواة جديد للعملية. بمعنى آخر إذا توقف خيط النواة بسبب انتظاره لاستكمال عملية إدخال، وإخراج، سيتوقف معه المعالج الافتراضي وهو ما سيؤدي أيضًا إلى توقف خيط المستخدم المرتبط بهذا المعالج. بناءً على ما سبق ذكره، يمكن توضيح طريقة عمل تنشيط المجدول على النحو التالي:



الشكل 2. 32: فكرة تنشيط المجدول.

يفترض المثال الموضح في الشكل 2. 32 أن هناك تطبيق يُنفذ على مستوى المستخدم، وأن النواة قد خصصت خيط واحد (معالج افتراضي) لإحدى عملياته والتي تحوي خيطين. يُجدول التطبيق هاذين الخيطين على المعالج الافتراضي على مستوى النواة. بفرض أن خيط المستخدم الجاري تنفيذه سيُضطر للانتظار لاستكمال عملية الإدخال، والإخراج، بالتالي سيستدعي نظام التوقف (الخطوة رقم 1)، استجابة لهذا الاستدعاء تحظر النواة كل من هذا الخيط، والمعالج الافتراضي الذي من المفترض أن يُنفذه (الخطوة رقم 2). يأتي الدور الآن على تنشيط المجدول، لذلك تقرر النواة تخصيص خيط جديد، أي معالج افتراضي جديد على مستوى النواة للعملية (الخطوة رقم 3). تستخدم النواة الآن إجراء مكتبة الخيوط **Upcall** لحفظ حالة الخيط الموقوف، ولكي تُعلم مدير الخيوط على مستوى المستخدم بالخيط الذي حُظِر على مستوى المستخدم، وبالخيط الجديد الذي أصبح حاليًا متاحًا على مستوى النواة (الخطوة رقم

4)، بعدها ينقل مدير خيوط المستخدم التحكم إلى بقية خيوط العملية على مستوى المستخدم لإعادة جدولة أحد الخيوط الجاهزة على المعالج الافتراضي الجديد (الخطوة رقم 5).

أخيرًا، عندما يَجهز الحدث الذي ينتظر فيه خيط المستخدم الموقوف سابقًا، تستدعي النواة إجراء **Upcall** آخر لإعلام مكتبة الخيوط بأن هذا الخيط قد أصبح الآن جاهزًا للتشغيل، ومن ثم ينتظر دوره لكي يُجدوله التطبيق من جديد على أي معالج افتراضي متاح إذا توفر، أو قد ينتزع له أحد معالجات خيوط المستخدم.

5.2 جدولة المعالجات

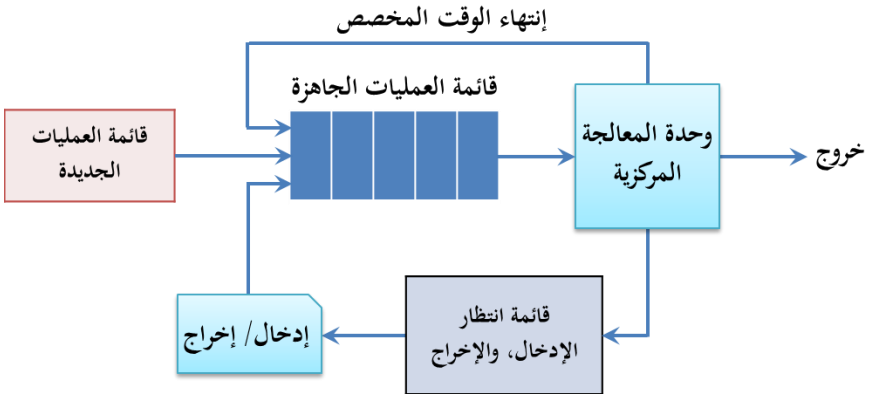
تُعتبر جدولة المعالجات (وحدات المعالجة المركزية) الأساس المبني عليه أنظمة تشغيل البرمجة المتعددة، فالمعالجات غالبًا ما تتنقل ذهابًا وإيابًا ما بين العمليات لغرض تنفيذها بطريقة تلبي أهداف أداء النظام. ونظرًا لأن المعالجات هي أهم مصادر الحاسوب، يجب على نظم التشغيل إدارتها بأكثر الطرق كفاءة وفعالية. وللقيام بذلك وبكفاءة عالية تُؤدي الجدولة دورًا غايةً في الأهمية في تنظيم تنافس العمليات أو الخيوط الموجودة داخل قائمة الحالة الجاهزة على هذه المعالجات في نفس الوقت.

من خلال مراجعة القسم 3.2 نتبين أنه بإمكان وحدة المعالجة المركزية تشغيل أكثر من عملية في آن واحد، وأن العمليات تتصل ببعضها البعض وأنها تتشارك في كثير من الأحيان في مجموعة مصادر النظام. كل هذا يُحتم على نظم التشغيل أن يكون لديها جزء مسؤول عن اتخاذ قرارات معينة بشأن تحديد أيٍ من العمليات التي يجب تشغيلها في لحظة معينة، وكذلك زمن تشغيلها. يُقدم هذا القسم بالتالي المفاهيم الأساسية لجدولة المعالجات والتي تتضمن أهداف الجدولة، وحالاتها، وأنواعها، كما يُقدم عدة خوارزميات خاصة بعملية الجدولة.

يُجسد الشكل 2. 33 نموذجًا مبسطًا لمفهوم جدولة وحدة المعالجة المركزية والذي يُحافظ فيه نظام التشغيل على قوائم انتظار جدولة العمليات المتمثلة في قائمة العمليات الجديدة، وقائمة العمليات الجاهزة الخاصة بمجموعة العمليات الجاهزة للتنفيذ والموجودة في الذاكرة الرئيسية، وقائمة انتظار الإدخال، والإخراج وتشمل العمليات المحظورة بسبب انتظارها لأحداث خارجية. في هذا النموذج تتطلب خدمة وحدة المعالجة المركزية التسلسل التالي من الإجراءات:

- تُنقل أي عملية جديدة إلى قائمة العمليات الجاهزة.
- تَتلقى العمليات الجاهزة خدمة وحدة المعالجة المركزية وفقًا لخوارزميات الجدولة والتي سنتعرض لها لاحقًا.
- تَنْظَم أي عملية قيد التنفيذ وتحتاج إلى أي حدث خارجي إلى قائمة انتظار الإدخال، والإخراج وتبقى هناك إلى أن يتوفر الحدث المطلوب، ومن ثم تُنقل إلى قائمة العمليات الجاهزة.
- إذا كانت هناك عملية قيد التنفيذ وبحاجة إلى مزيد من الخدمة، وإنتهى وقتها المخصص لها، فستُنقل إلى قائمة العمليات الجاهزة.
- تخرج العملية إذا ما إنتهت مهمتها.

من أجل تبسيط موضوع الجدولة، نفترض أن نظام الحاسوب يحتوي على معالج واحد فقط، وأن الجزء المسؤول على اتخاذ قرارات الجدولة والذي سيُقرر أيٍ من العمليات ستشتغل تاليًا يُدعى بالمجدول. فالمجدول هو الذي يُقرر أيضًا متى يبدأ تنفيذ العملية، ومتى تقف، ومتى يسمح لغيرها بالتنفيذ، وذلك حسب حالات العملية. الأمر نفسه ينطبق على جدولة الخيوط، ولكن بنسب تتفاوت. أيضًا تُدعى الخوارزميات المستخدمة لهذا الغرض بخوارزميات الجدولة، والتي تُعتبر عملية تصميمها من المهام الصعبة، لأنها تعتمد في واقع الأمر على عدة معايير وأهداف قد يؤدي تحقيق إحدها إلى التعارض مع تحقيق الآخر.



الشكل 2. 33: مفهوم مبسط لجدولة وحدة المعالجة المركزية.

1.5.2 أهداف ومعايير الجدولة

تختلف أهداف ومعايير الجدولة التي يجب على خوارزميات الجدولة تحقيقها باختلاف بيئة التشغيل. فبينما تُركز أنظمة الدفعة على كل من الإنتاجية، والفترة الزمنية المستغرقة في تنفيذ العمليات بدءًا من لحظة تشغيلها وحتى نهاية التشغيل، وكذلك استغلالية وحدة المعالجة المركزية، نجد أن الأنظمة التفاعلية تهتم بزمن الاستجابة والمحصور بين زمن إصدار أول أمر وزمن ظهور أول استجابة، وكذلك النسبية، أي تلبية توقعات المستخدمين. بعض من هذه المعايير يُمكن تعريفها على النحو التالي:

1. زمن الإنتهاء وهو الزمن الذي يكتمل فيه تنفيذ العملية.
2. الفترة الزمنية المستغرقة في التنفيذ والتي تُمثل فرق الزمن بين زمن الإنتهاء وزمن وصول العملية، أي أنها تساوي زمن الإنهاء مطروح منه زمن الوصول.
3. زمن الانتظار والذي يُشير إلى فرق الزمن بين الفترة الزمنية المستغرقة في التنفيذ ومدة تشغيل العملية، أي أنه يساوي الفترة الزمنية المستغرقة في التنفيذ مطروح منها مدة تشغيل العملية.

بالمقابل يُعتبر الإلتزام بالمواعيد النهائية للتنفيذ، وتجنب كل من فقدان البيانات، وتدهور الجودة في أنظمة الوسائط المتعددة أهم معايير أنظمة الوقت الحقيقي. ولكن هناك أيضًا بعض الأهداف العامة والتي من الأفضل تحقيقها في جميع الحالات والتي في مجملها يمكن تفصيلها تحت العدالة، والتوازن، أي الاستفادة من جميع مكونات النظام، بالإضافة إلى تطبيق السياسة المعلنة له. وفيما يلي تفصيل لأهداف ومعايير كل بيئة.

أهداف ومعايير جدولة أنظمة الدفعة

تُعتبر إنتاجية النظام كما أشرنا سابقًا من المعايير المهمة في جدولة أنظمة الدفعة، فهي تتمثل في عدد الوظائف التي تُعالج أو تُنفذ في الوحدة الزمنية الواحدة، والتي يجب أن تكون مرتفعة، بمعنى كلما زاد عدد الوظائف المنجزة في هذه الوحدة كلما تمتع النظام بإنتاجية أفضل. بينما تتمثل الفترة الزمنية المستغرقة في التنفيذ في المتوسط الإحصائي للزمن بدءًا من لحظة تشغيل العملية وحتى نهاية تنفيذها، أي منذ لحظة تقديم وظيفة الدفعة وحتى لحظة الإنتهاء من تشغيلها، وهي ما تقيس متوسط طول زمن انتظار المستخدم لمخرجات وظيفة الدفعة، بالتالي كلما قلّ هذا

المتوسط، كلما تحسَّن أداء النظام.

من ناحية أخرى يُعتبر معيار استغلالية وحدة المعالجة المركزية بالنسبة لأنظمة الدفعة ليس مهمًا جدًا إذا ما قورن بالمعيارين السابقين، إلَّا إنَّ وصول هذا المعيار إلى نسبة 100٪ قد تكون مفيدة في معرفة اللحظة التي يصل فيها النظام إلى أقصى حد لقوته الحوسبية. ولكن الحقيقة أيضًا أنه قد يتعارض أحيانًا تحقيق المعيارين السابقين بشكل مثالي. على سبيل المثال، في حالة وجود مزيج من الوظائف القصيرة والطويلة داخل النظام، وكانت قوانين الجدولة تنص على تنفيذ الوظائف الأقصر أولًا، بالتالي فإن هذا النظام سيُحقق إنتاجية عالية، ولكن على حساب معيار الفترة الزمنية المستغرقة في التنفيذ بالنسبة للوظائف الطويلة والتي ستكون سيئة جدًا.

أهداف ومعايير جدولة الأنظمة التفاعلية

نظرًا لأن الأنظمة التفاعلية تتفاعل مباشرةً مع المستخدم، نجدها تُركز قدر المستطاع على محاولة تقليل متوسط الزمن المحصور بين إرسال أي عملية مستخدم أمرًا أو طلبًا إلى نظام التشغيل وتلقي أول استجابة، أي الزمن المستغرق لبدء الاستجابة وليس لإخراجها وهو ما يُعرف بمتوسط زمن الاستجابة. بالتالي نجد أنه من المهم في الحواسيب الشخصية تقديم طلبات المستخدم المتمثلة مثلًا، في بدء برنامج معين، أو فتح أي ملف على عمل عمليات خلفية كاختبار وجود تحديثات لنظام الحماية من الفيروسات، أو قراءة وتخزين البريد الإلكتروني والذي بدوره سيُقلل من زمن الاستجابة. بمعنى آخر، يُعتبر تفضيل عمليات المستخدم على العمليات الخلفية من الخصائص الجيدة والتي يجب توفرها في الأنظمة التفاعلية.

من المهم أيضًا في الأنظمة التفاعلية توافق توقعات المستخدم لزمن تنفيذ أي مهمة، مع أزمنة التنفيذ الفعلية للمهام، وهو ما يندرج تحت مصطلح النسبية. فالأوامر البسيطة من المتوقع تنفيذها في زمن أقل من الأوامر المعقدة، بالتالي عند النقر على أيقونة برنامج إرسال فاكس، من المتوقع أن يأخذ الحاسوب زمن طويل في تنفيذه، وهو ما سيستقبله المستخدم بكل رحابة صدر، ولكن عندما ينظر المستخدم إلى الطلب على أساس أنه طلب بسيط ويجده قد كلف زمنًا طويلًا فسيؤدي ذلك إلى انزعاجه.

أهداف ومعايير جدولة أنظمة الزمن الحقيقي

ما يُميز أنظمة الزمن الحقيقي عن الأنظمة السابقة الذكر هو وجود مواعيد نهائية محددة يجب أو على الأقل ينبغي على المجدول تحقيقها والإلتزام بها. لذلك نجد أن أهداف مجدولات أنظمة الزمن الحقيقي تُركز على الاستجابة الفورية للأحداث التي تحدث في العالم الحقيقي، لكي يمكن السيطرة عليها والتفاعل معها. على سبيل المثال، إذا لم يعمل نظام الزمن الحقيقي للسيطرة على فرامل المركبة الآلية في الوقت المناسب، فقد يتسبب ذلك في وقوع كارثة. بالتالي لتحقيق هذا الهدف، يجب أن يُدعَّم النظام بعناية بالخيار المناسب من بين مجموعة متنوعة من خيارات الجدولة المتاحة في الوقت الحقيقي.

أشرنا في الفصل الأول إلى أن في نظام الوقت الحقيقي المرن لا يتسبب عدم وقوع الحدث في وقته المناسب أي ضرر دائم يؤثر على النتيجة النهائية للعملية، إلَّا إنَّه قد يؤدي إلى فقدان البيانات، وتدهور الجودة وخاصة في تلك الأنظمة التي تنطوي على الوسائط المتعددة، الأمر الذي من المحتمل أن يتسبب في عدم رضا المستخدم لذلك ولتجنب مثل هذه المشاكل، يجب أن تكون عملية الجدولة غاية في الدقة.

الأهداف العامة للجدولة

تحت جميع بيئات أنظمة التشغيل والمتمثلة في أنظمة الدفعة، والأنظمة التفاعلية، وكذلك أنظمة الوقت الحقيقي، تُعتبر العدالة أحد أهم الأهداف العامة لها وتحت أي ظرف. فمقياس العدالة يُشير إلى ما إذا تعامل النظام مع جميع العمليات بطريقة مماثلة، أي يجب أن يتأكد المجدول من أن كل عملية تحصل على نصيبها العادل من وحدة المعالجة المركزية ولا يمكن تأجيل أي عملية إلى أجل غير مسمى. لاحظ بالطبع أن العدالة قد لا تعني إعطاء وقت متساوٍ لجميع العمليات أو الخيوط، لأنه يجب في كثير من الأحيان مراعاة فئات محددة من العمليات ذات الأهميات الأعلى في التطبيقات مثل، عمليات التحكم والرقابة على السلامة داخل محطة نووية.

يتصل مفهوم العدالة أيضًا إلى حد ما بتطبيق السياسة المحلية للنظام، فإذا كانت هذه السياسة تنص على أن عمليات الرقابة على السلامة تتمتع بأولوية عالية، وأنها تبدأ في الاشتغال

كلما ظهرت للوجود، حتى لو كان ذلك على حساب عمليات أخرى، فيجب على المجدول التأكد من تنفيذ هذه السياسة بشكل دقيق.

من الأهداف العامة الأخرى للجدولة هو توفير مزيج متوازن من الوظائف للحفاظ على جميع أجزاء النظام تشتغل بشكل متزامن كلما أمكن ذلك. مثلًا، من المهم في نظام متعدد المعالجات موازنة حمل التنفيذ بينها، بحيث لا يكون إحداها مُثقل بعمليات التنفيذ، بينما يكون الآخر في وضع الخمول. كذلك إذا تمكن المجدول من تحقيق التوازن بين استغلال وحدة المعالجة المركزية وجميع أجهزة الإدخال، والإخراج طيلة الوقت، فهذا سيضمن القيام بمزيد من الأعمال في الثانية الواحدة. فتحميل مزيج متوازن من وظائف الإدخال، والإخراج وعمليات وحدة المعالجة في الذاكرة معًا في آن واحد سيكون أفضل من لو حُمِّلت وشُغِّلت جميع عمليات وحدة المعالجة المركزية أولًا، ومن ثم وبعد الإنتهاء من تنفيذها، تُحمَّل وتُشغَّل جميع وظائف الإدخال، والإخراج.

من مساوي استخدام الإستراتيجية الأخيرة هو تكدس العمليات على وحدة المعالجة المركزية وبالتالي إجهادها، بينما سيكون القرص خاملًا وغير مستغل. ولو كان العكس، فسيُجهد القرص بسبب وفرة العمليات، بينما ستكون وحدة المعالجة المركزية خاملة وغير مستغلة. لذا من الأفضل الحفاظ على درجة البرمجة المتعددة مستقرة داخل النظام، لكي يكون متوسط معدل إنشاء العملية مساويًا لمتوسط معدل مغادرة العمليات للنظام.

2.5.2 حالات الجدولة

من المسائل الرئيسية المتعلقة بالجدولة هي متى تُؤخذ قرارات الجدولة؟ من الواضح ومن خلال تتبع آلية تنفيذ العمليات، وبالرجوع إلى انتقالات العملية المسردة في الجدول 2. 3، نجد أن هناك مجموعة متنوعة من الحالات التي يُتخذ فيها قرارات جدولة وحدة المعالجة المركزية:

1. عند إنشاء عملية جديدة، يحتاج المجدول إلى اتخاذ قرار الجدولة سواءً بتشغيل العملية الأب أو العملية الابن.
2. عند إنتهاء تنفيذ أي عملية، يجب على المجدول اتخاذ قرار بالجدولة وذلك لأن هذه العملية لم تعد قادرة على الاشتغال.

3. عندما تُحظر أي عملية بسبب انتظارها لأحداث خارجية، أو لأي سبب آخر، فيجب على المجدول اختيار عملية أخرى للتشغيل.
4. عند حدوث مقاطعة إدخال، وإخراج، يمكن للمجدول اتخاذ قرار الجدولة لغرض تشغيل عملية توقفت بسبب المقاطعة، أو بعض من العمليات الجاهزة الأخرى.

3.5.2 أنواع الجدولة

في أنظمة الحواسيب يمكن اتخاذ قرار الجدولة عند كل مقاطعة أو عند كل عدد محدد من المقاطعات. الأمر الذي يترتب عليه تقسيم خوارزميات الجدولة إلى فئتين بناءً على كيفية التعامل مع المقاطعات المولدة.

تُمثل الفئة الأولى الجدولة الاستباقية والتي تسمح بمقاطعة عملية جاري تنفيذها حتى قبل أن تُنهي مهمتها وذلك لأحد الأسباب التالية:

- إنتهاء الفترة (الشريحة) الزمنية المخصصة لها.
- انتظارها لحدث خارجي (إدخال/ إخراج).
- وصول عملية أخرى ذات أولوية أعلى منها إلى قائمة العمليات الجاهزة.

في جميع هذه الحالات تُعلق العملية الأولى وتُنقل مرة أخرى إلى قائمة العمليات الجاهزة لتنتظر دورها من جديد. هذه الفئة من الجدولة تتطلب حدوث نبضة المقاطعة في نهاية الفترة الزمنية المحددة لكل عملية، وذلك لمنح المجدول السيطرة على وحدة المعالجة المركزية.

بالمقابل تُمثل الفئة الثانية الجدولة غير الاستباقية وفيها تُخصص وحدة المعالجة المركزية لعملية معينة وتحتفظ بها إلى أن تُنجز مهمتها حتى ولو بقت تشتغل لمدة ساعات، أو تُوقف بسبب انتظارها لأحداث خارجية ناتجة عن التعامل مع وحدات الإدخال، والإخراج. هذا يعني أنه خلال نبضة المقاطعة لن تكون هناك قرارات جدولة. أمَّا بعد إنتهاء نبضة المقاطعة، فستُستأنف العملية التي كانت تشتغل قبل المقاطعة، طالما لم تكن هناك عملية ذات أولوية أعلى منها.

أخيرًا، أشرنا آنفًا إلى أن هناك أنواع مختلفة من أنظمة التشغيل والتي تشتغل في بيئات

مختلفة تختلف باختلاف مجالات التطبيق تتمثل في أنظمة الدفعة، والأنظمة التفاعلية، وأنظمة الزمن الحقيقي. إذًا وكنتيجة طبيعية هناك حاجة أيضًا إلى أنواع مختلفة من خوارزميات الجدولة. بعبارة أخرى، ما يجب على خوارزميات الجدولة تحسينه يختلف من بيئة إلى أخرى.

4.5.2 الجدولة في أنظمة الدفعة

تتميز أنظمة الدفعة بأنها لا تتفاعل مباشرةً مع المستخدم، لذلك لا تجد في الغالب من ينتظر بفارغ الصبر في الحصول على استجابة سريعة لطلبات قصيرة. هذا يعني أنه بإمكان كل وظيفة أن تأخذ نصيبها الكافي من وحدة المعالجة المركزية من دون أن تكون هناك قيود زمنية على فترات تشغيل العمليات وهو ما يُحقق نوعًا ما مبدأ العدالة، بالتالي ستكون الجدولة الاستباقية أو غير الاستباقية ذات الفترات الزمنية الطويلة مقبولة لتنفيذ كل وظيفة في إطار هذه الأنظمة. يترتب على هذا الأمر قلة عمليات تبديل السياق بين الوظائف وبالتالي تحسين أداء النظام. فيما يلي سيتم النظر بشكل تفصيلي في بعض خوارزميات جدولة أنظمة الدفعة. تجدر الإشارة هنا إلى أنَّ بعض من هذه الخوارزميات يمكن استخدامها حتى في النظم التفاعلية والتي ستُناقش لاحقًا.

1.4.5.2 خوارزمية القادم أولًا يُخدم أولًا للجدولة

تُعتبر خوارزمية القادم أولًا يُخدم أولًا من أبسط خوارزميات الجدولة غير الاستباقية وفيها تُخصص الوظائف لوحدة المعالجة المركزية حسب ترتيب طلبهم من قبلها، أو وصولهم إليها. في الأساس، هناك طابور انتظار واحد يضم جميع الوظائف المتواجدة في حالة جاهزة. عند وصول أول وظيفة إلى الطابور، فستُنفذ فورًا إلى أن تنتهي، أي لن يُسمح بمقاطعتها حتى ولو اشتغلت لفترة زمنية طويلة جدًا، ولكن إذا ما احتاجت إلى أي حدث خارجي (تعامل مع وحدات الإدخال، والإخراج) فستُعلق هذه الوظيفة إلى حين توفره، ومن ثم تُجلب الوظيفة التالية من بداية الطابور، بينما ستُضاف الوظيفة الموقوفة إلى نهايته وستُعامل مثل أي وظيفة وصلت حديثًا إليه. في حالة وصول وظائف جديدة فستُرص في نهاية الطابور.

من أهم خصائص هذه الخوارزمية السهولة في الفهم والبرمجة. حيث يُمكن تمثيل جميع الوظائف الجاهزة على أساس طابور انتظار، تُسحب العمليات من مقدمته لغرض تنفيذها، في حين تُرفق الوظائف الجديدة أو العمليات المنتقلة من حالة موقوفة إلى حالة جاهزة إلى نهايته، ولكن

من عيوبها هو أنه بإمكان عملية واحدة– لا تقوم بأي عمليات إدخال، أو إخراج– الاستحواذ على المعالج طيلة فترة اشتغالها. يؤدي هذا الأمر إلى حظر عدة عمليات وخصوصًا تلك التي تحتاج إلى استخدام المصادر لفترة قصيرة وهو ما سيؤدي بشكل أساسي إلى عدم استغلال المصادر بشكل أمثل، وبالتالي ضعف الأداء بشكل عام.

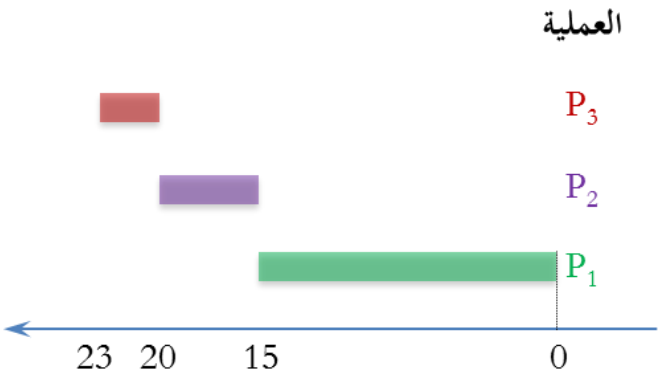
مثال توضيحي:

إذا كانت هناك ثلاث عمليات P_1، P_2، و P_3 تشتغل على حسب الفترات الزمنية 15، 5، و 3 ملّي ثانية على التوالي، وكان زمن وصولها إلى وحدة المعالجة على النحو التالي: 0، 2، 3 على التوالي، فأوجد قيمة كل من:

- متوسط الفترة الزمنية المستغرقة في التنفيذ،
- ومتوسط زمن الانتظار.

يُوضح الشكل 2. 34 الكيفية التي تعمل بها خوارزمية القادم أولًا يُخدم أولًا في تنفيذ هذه العمليات. من خلال كل من هذا الشكل، ومن العلاقات الحسابية الموضحة سابقًا، نجد أن:

- متوسط فترة التنفيذ = [(15) + (20 – 2) + (23 – 3)] ÷ 3 = 17.67
- متوسط زمن الانتظار = [(15 – 15) + (18 – 5) + (20 – 3)] ÷ 3 = 10



الشكل 2. 34: مثال لتنفيذ ثلاث عمليات باستخدام خوارزمية القادم أولًا يُخدم أولًا.

2.4.5.2 خوارزمية أقصر وظيفة أولًا للجدولة

تُعتبر خوارزمية أقصر وظيفة أولًا أكثر ملائمة لوظائف الدفعة وهي تندرج تحت خوارزميات الجدولة غير الاستباقية، وتتمحور فكرتها حول البدء في تنفيذ الوظائف ذات أقصر زمن تنفيذ أولًا، ومن ثم التي تليها في الكبر، لذلك فهي تتطلب مسبقًا معرفة زمن تشغيل كل وظيفة، والذي قد يصعب معرفته في الغالب. لهذا السبب يُفضل استخدام هذه الخوارزمية في البيئات التي يُمكن فيها التنبؤ بدقة للفترة الزمنية التي ستستغرقها أي وظيفة لإنجاز مهمتها داخل دفعة الوظائف. تنتج دقة التنبؤ هذه من الخبرة المكتسبة نتيجة لتكرار نفس العمل.

تتميز هذه الخوارزمية بأنها من أفضل الطرق لتقليل زمن الانتظار إذا ما قورنت ببقية الخوارزميات، لأن البدء بأقصر وظيفة أولًا سيؤثر بشكل أساسي في تقليل متوسط الفترة الزمنية المستغرقة في التنفيذ، وبالتالي في متوسط زمن الانتظار. على سبيل المثال بفرض أن هناك أربع وظائف بأزمنة تشغيل هي **a**، **b**، **c**، و **d**، على التوالي. الوظيفة الأولى ستنتهي عند **a**، والثانية ستنتهي عند **a + b**، والثالثة عند **a + b + c**، وهكذا. بالتالي يكون متوسط الفترة الزمنية المستغرقة في التنفيذ هو (**4a +3b +2c +d**) ÷ **4**. إذًا من الواضح أن **a** تُساهم بشكل كبير في المتوسط تليها بقية الأزمنة، لذلك ينبغي أن تُنفذ **a** أولًا، تليها **b**، ثم **c**، وأخيرًا **d** كأطول زمن. الحجة نفسها تنطبق أيضًا على قدم المساواة على أي عدد من الوظائف.

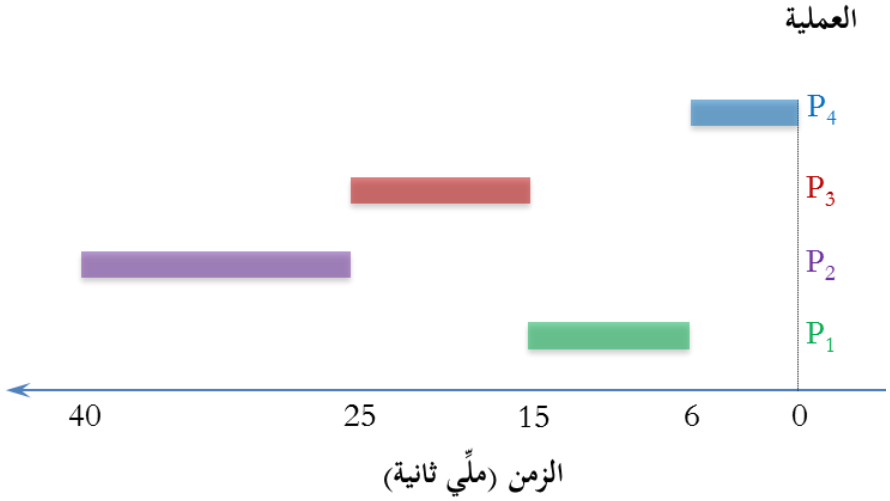
مثال توضيحي:

إذا كانت هناك أربع عمليات P_1، P_2، P_3، و P_4 تشتغل على حسب الفترات الزمنية **9**، **15**، **10**، و **6** ملِّي ثانية على التوالي، وصلت في نفس الوقت إلى مركز الحاسوب، فأوجد قيمة كل من:

- متوسط الفترة الزمنية المستغرقة في التنفيذ،
- ومتوسط زمن الانتظار.

يُوضح الشكل 2. **35** الكيفية التي تعمل بها خوارزمية أقصر وظيفة أولًا في تنفيذ هذه العمليات. من خلال هذا الشكل ومن خلال العلاقات الحسابية الموضحة سابقًا، نجد أن:

- متوسط فترة التنفيذ = (6 + 15 + 25 + 40) ÷ 4 = 21.50
- متوسط زمن الانتظار = [(6 – 6) + (9 – 15) + (10 – 25) + (15 – 40)] ÷ 4 = 11.50



الشكل 2. 35: مثال لتنفيذ أربع عمليات باستخدام خوارزمية أقصر وظيفة أولًا.

تجدر الإشارة إلى أن خوارزمية أقصر وظيفة أولًا تكون هي المثلى فقط عندما تتوفر كل الوظائف في وقت واحد. إذا كان وقت وصول الوظائف إلى مركز الحاسوب مختلفًا، فهذا يعني أن جميعها غير متاحة في قائمة الوظائف الجاهزة في الوقت 0، وقد تصل بعض منها بعد مرور بعض الوقت والتي قد يكون من ضمنها وظائف ذات أزمنة قصيرة، في مثل هذه الحالات ونظرًا لأن هذه الخوارزمية غير استباقية، فيتحتم على أي وظيفة ذات زمن أقصر من زمن الوظيفة الجاري حاليًا تنفيذها الانتظار برهةً من الزمن إلى أن تنتهي، ومن ثم تُختار الوظيفة الأقصر، كما هو موضح في الخوارزمية التالية

إذا حدث واستمرت الوظائف الأقصر في الظهور داخل قائمة العمليات الجاهزة في أثناء تواجد وظيفة ذات زمن طويل في حالة تشتغل، فسيؤدي ذلك إلى ظهور مشكلة المجاعة، أي لن تتمكن الوظيفة ذات الزمن الطويل من الحصول على حصتها من وحدة المعالجة المركزية. أحد الحلول الممكنة لهذه المشكلة تكمن في تجنب جدولة المزيد من الوظائف في طابور الوظائف

الجاهزة بمجرد امتلائها، قد يكمن الحل الآخر في استخدام مفهوم الشيخوخة، كما سيُوضح ذلك لاحقًا.

3.4.5.2 خوارزمية أقصر الأوقات المتبقية تاليًا للجدولة

لاحظنا في الخوارزمية السابقة أنه في حالة عدم توفر كل الوظائف في وقت واحد، ستبدأ الخوارزمية في تنفيذ أول وظيفة دخلت إلى قائمة الوظائف الجاهزة في الوقت 0، حتى ولو لم تكن أقصر الوظائف، الأمر الذي سوف يؤثر على كفاءتها. لذلك ظهرت نسخة استباقية من خوارزمية أقصر وظيفة أولًا عُرفت بخوارزمية أقصر الأوقات المتبقية تاليًا.

تُعنَى هذه الخوارزمية باختيار أي وظيفة للتنفيذ بناءً على كمية الزمن المتبقي من زمن تشغيلها، أي إذا كان الزمن المتبقي للوظيفة الجاري حاليًا تنفيذها أكبر من زمن التشغيل الكلي لأي وظيفة أخرى داخل قائمة الوظائف الجاهزة، فسيُعلق تنفيذ هذه الوظيفة وستختار الخوارزمية الوظيفة الجديدة ذات الزمن الأقل للبدء في تنفيذها مباشرةً، بالتالي تُتيح هذه الخوارزمية الفرصة للوظائف الجديدة ذات أزمنة التشغيل القصيرة للحصول على خدمات جيدة. مرة أخرى، يجب أن يكون زمن التشغيل لأي وظيفة معروف مسبقًا

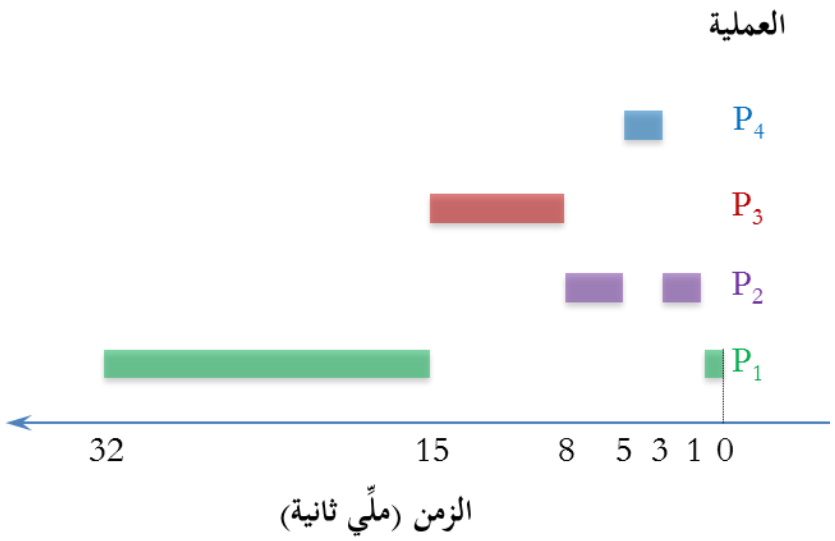
مثال توضيحي:

إذا كانت هناك أربع عمليات P_1، P_2، P_3، و P_4 تشتغل على حسب الفترات الزمنية 18، 5، 7، و 2 ملّي ثانية، ووصلت في أزمنة مختلفة إلى مركز الحاسوب على النحو 0، 1، 2، 3 على التوالي، بين آلية تنفيذ هذه العمليات في ظل استخدام خوارزمية أقصر الأوقات المتبقية تاليًا، ومن ثم أوجد قيمة كل من:

- متوسط الفترة الزمنية المستغرقة في التنفيذ،
- ومتوسط زمن الانتظار.

من خلال المعطيات السابقة نلاحظ أن العملية P_1 وصلت أولًا، بالتالي ستُنفذ مباشرةً، ولكن بعد مُضي 1 ملّي ثانية ستُعلق الخوارزمية تنفيذها، وذلك نظرًا لوصول العملية P_2 والتي زمن تنفيذها أقل من الزمن المتبقي للعملية الأولى، وبالتالي ستكون مرشحة للتنفيذ على النحو

الموضح في الشكل 2. 36. بعد ذلك تستمر الخوارزمية في تنفيذ العملية P_2 حتى بعد وصول العملية P_3، لأن زمن تنفيذ العملية الأخيرة أكبر من الزمن المتبقي من العملية P_2، ولكن الأمر سيكون مختلف بعد وصول العملية P_4 ذات زمن التنفيذ 2 ملِّي ثانية، نتيجة لذلك ستُعلق الخوارزمية الآن تنفيذ العملية P_2، وستبدأ في تنفيذ العملية P_4 إلى أن تنتهي، ومن ثم تستأنف الخوارزمية تنفيذ العملية P_2 إلى أن تنتهي، يليها P_3، وأخيرًا تستكمل الخوارزمية تنفيذ P_1.



الشكل 2. 36: مثال لتنفيذ أربع عمليات باستخدام خوارزمية أقصر الأوقات المتبقية تاليًا.

بالتالي من خلال كل من الشكل 2. 36 ومن العلاقات الحسابية الموضحة سابقًا، نجد أن:

- متوسط فترة التنفيذ = $[(0 - 32) + (1 - 8) + (2 - 15) + (3 - 5)] \div 4 = 13.50$
- متوسط زمن الانتظار = $[(18 - 32) + (5 - 7) + (7 - 13) + (2 - 2)] \div 4 = 5.50$

4.4.5.2 خوارزمية الأولوية للجدولة

بالنظر إلى خوارزمية القادم أولًا يُخدم أولًا للجدولة نلاحظ بأنها تفترض ضمنيًا أن جميع

الوظائف تتمتع بنفس القدر من الأهمية، وأنها تتعامل معهم على حسب ترتيب وصولهم، في حين تُعطي خوارزمية أقصر وظيفة أولًا للجدولة نوعًا من الأهمية لهذه الوظائف وفقًا لكمية الزمن المستغرق في التنفيذ، بمعنى تتمتع الوظائف ذات أقصر زمن بأهمية أعلى من تلك التي لها زمن أطول. ولكن وفي كثير من الأحيان وخصوصًا في الحواسيب المركزية ومتعددة المستخدمين، يحدث أن هناك تفاوت في أهمية مستخدمي هذا النوع من الأجهزة. على سبيل المثال، في المؤسسات الكبرى مثل الجامعات يتمتع العمداء بأعلى أهمية يليهم في ذلك الأساتذة، ومن ثم عمال النظافة، وأخيرًا الطلاب. أخذ مثل هذه العوامل الخارجية في عين الاعتبار يؤدي إلى ضرورو وجود ما يُعرف بجدولة الأولويات.

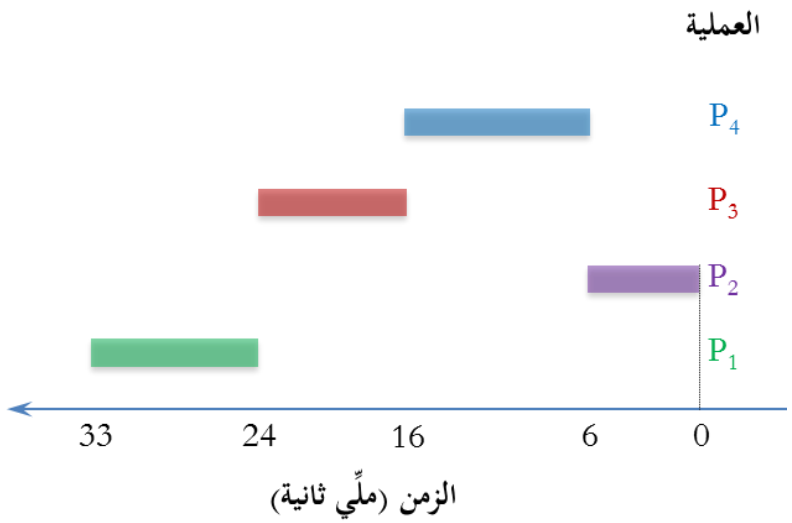
تتمثل الفكرة الأساسية في هذا النوع من الخوارزميات في إسناد أولوية لكل وظيفة بحيث تُنفذ الوظائف على حسب أولويتها المسندة لها، أي تُنفذ الوظائف ذات الأولويات العالية أولًا، بينما قد تُنفذ مجموعة الوظائف ذات الأولويات المتعادلة بآلية القادم أولًا، يُخدم أولًا، أو بأي آلية أخرى. في هذا الإطار تتمثل الأولويات بمدى محدد من الأرقام مثل، المدى من **0** إلى **9**، أو **0** إلى **1024**، بحيث يمكن أن تدل الأرقام الصغيرة عل الأولويات المنخفضة، والأرقام الكبيرة على الأولويات العالية، أو العكس. تُسند هذه الأولويات بناءً على متطلبات الذاكرة، أو متطلبات المستخدم، أو أي متطلبات أخرى للمستخدم.

مثال توضيحي:

إذا كانت هناك أربع عمليات وصلت إلى مركز الحاسوب تقريبًا في نفس الوقت هي: P_1 وتشتغل لمدة **9** ملِّي ثانية وبأفضلية **0**، و P_2 وتشتغل لمدة **6** ملِّي ثانية وبأفضلية **5**، و P_3 وتشتغل لمدة **8** ملِّي ثانية وبأفضلية **1**، و P_4 وتشتغل لمدة **10** ملِّي ثانية وبأفضلية **3**، علمًا أن **0** تُمثل أقل أفضلية، فأوجد قيمة كل من:

- متوسط الفترة الزمنية المستغرقة في التنفيذ،
- ومتوسط زمن الانتظار.

يُوضح الشكل 2. 37 الكيفية التي تعمل بها خوارزمية الأولوية في تنفيذ هذه العمليات. من خلال هذا الشكل ومن العلاقات الحسابية الموضحة سابقًا، نجد أن:



الشكل 2. 37: مثال لتنفيذ أربع عمليات باستخدام خوارزمية الأولوية.

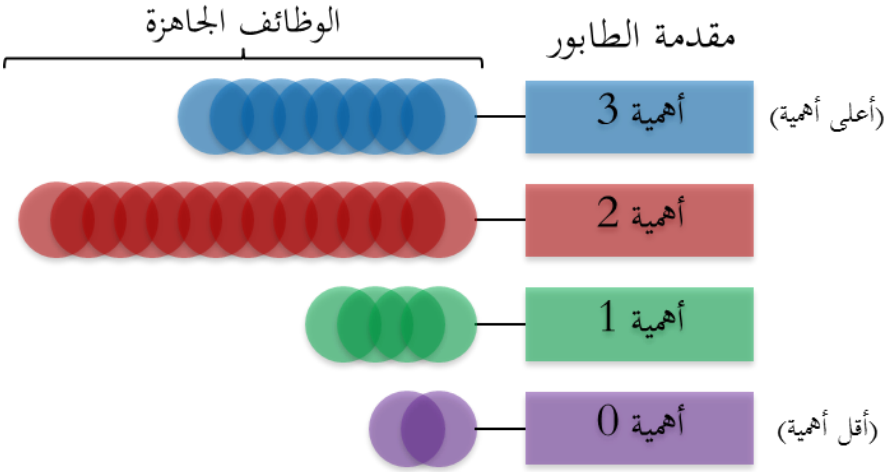
- متوسط فترة التنفيذ = (6 + 16 + 24 + 33) ÷ 4 = 19.75
- متوسط زمن الإنتظار = [(6 – 6) + (10 – 16) + (8 – 24) + (9 – 33)] ÷ 4 = 11.50

تُعتبر خوارزمية الأولوية من الخوارزميات البسيطة، والشائعة في الجدولة غير الاستباقية بحيث إذا وصلت وظيفة جديدة ذات أولوية أعلى من الوظيفة الجاري حاليًا تنفيذها، تُوضع هذه الوظيفة في مقدمة قائمة العمليات الجاهزة وتبقى هناك إلى أن تنتهي وحدة المعالجة من تنفيذ الوظيفة الحالية، ومن ثم تبدأ مرحلة تنفيذ الوظيفة الجديدة. من عيوب هذا النهج أنَّ الوظائف صاحبة الأولويات المنخفضة سوف تكون عرضةً للتجويع، وخصوصًا إذا كان هناك تدفق لسيل من الوظائف ذات أولويات عالية إلى قائمة الوظائف الجاهزة.

يكمن حل مشكلة المجاعة هنا في استخدام مفهوم الشيخوخة لغرض زيادة أولوية الوظائف تدريجيًا، بناءً على زمن انتظارها في قائمة العمليات الجاهزة. في هذه الحالة سيتغير نهج خوارزمية الأولوية غير الاستباقية إلى نهج الجدولة الاستباقية بحيث عند زيادة أهمية أي وظيفة، تُقارن أهميتها بأهمية الوظيفة الجاري حاليًا تنفيذها فإذا أصبحت أعلى منها، فستُوقف مباشرةً الوظيفة الحالية، وسينتقل التنفيذ إلى الوظيفة التالية في الأهمية، وهكذا. في هذا النوع من الجدولة، إذا

دخلت وظيفة جديدة في قائمة العمليات الجاهزة وكانت تتمتع بأهمية أعلى من الوظيفة قيد التنفيذ، فستتوقف وحدة المعالجة المركزية عن معالجتها، وسينتقل التنفيذ إلى الوظيفة الجديدة. كبديل لحل الشيخوخة، قد يتم تعيين فترة زمنية محددة لكل وظيفة كحد أقصى يُسمح فيه لها بالاشتغال، وعند نفاذ هذا الحد تُعطى الفرصة للوظيفة التالية في الأهمية.

غالبًا ما يكون من المناسب تجميع الوظائف في مجموعات بحيث تحمل كل مجموعة عدد من الوظائف ذات نفس الأهمية، كوجود عدد من الأساتذة في مجموعة الأساتذة، وعدد من الطلبة في مجموعة الطلبة، وهكذا، وذلك كما هو موضح في الشكل 2. 38. في هذا الشكل، يُستخدم مجدول الأولوية للتنسيق بين المجموعات بحيث تُنفذ الوظائف داخل المجموعة ذات الأهمية رقم 3 أولًا في حالة لم تكن خالية (باعتبارها أعلى رتبة) حتى تنتهي جميعها، ثم ينتقل التنفيذ إلى المجموعة ذات الأهمية رقم 2 لتنفيذ جميع وظائفها إذا لم تكن خالية، وهكذا حتى تنتهي كافة المجموعات، مع ملاحظة أنه ليس من الضروري أن تتواجد وظائف في كل المجموعات، وأن الوظائف داخل نفس المجموعة تُنفذ باستخدام أحد الخوارزميات الأخرى مثل، خوارزمية القادم أولًا، يُخدم أولًا.



الشكل 2. 38: خوارزمية الجدولة لأربع مجموعات ذات أهميات مختلفة.

5.5.2 الجدولة في الأنظمة التفاعلية

تُستخدم الجدولة الاستبقاية في بيئة الأنظمة التفاعلية لضمان الاستجابة السريعة للطلبات الجديدة الخاصة بالمستخدم التفاعلي. فالهدف هو الخدمة التفاعلية المباشرة والسريعة لطلبات الإدخال، والإخراج الاعتيادية مثل، لوحة المفاتيح، والفأرة مقابل الخدمات الطويلة التي تتطلب السيطرة على وحدة المعالجة المركزية لفترات زمنية طويلة وتحرم العمليات الأخرى من خدماتها. لذلك تسعى خوارزميات الجدولة الاستباقية إلى تقليل زمن الاستجابة للمستخدم، وإلَّا سيُعتبر النظام غير مجدي. يُقدم هذا القسم بعض من خوارزميات الجدولة في الأنظمة التفاعلية والتي عادة ما تجدها في الحواسيب الشخصية، مزودي الخدمات، وأنواع أخرى من الأنظمة.

1.5.5.2 خوارزمية راوند روبن للجدولة

تُستخدم خوارزمية راوند روبن (**Round-Robin**) في الغالب في أنظمة المشاركة الزمنية، وهي تُعتبر من أبسط، وأعدل، وأكثر الخوارزميات استخدامًا في هذا المجال. تعتمد فكرة هذه الخوارزمية على الاحتفاظ بقائمة للعمليات الجاهزة كتلك التي شاهدناها في الشكل 2. 33، وعلى تخصيص أو تحديد شريحة زمنية للعملية بحيث تُنفذ فيها. في حالة إنتهاء العملية قبل إنتهاء الفترة الزمنية المخصصة لها، أو في حالة دخولها إلى حالة موقوفة، فستُخصص وحدة المعالجة المركزية لأول عملية موجودة في مقدمة القائمة. أمَّا إذا إنتهت هذه الفترة الزمنية قبل إنتهاء العملية نفسها فستُحظر وتُنقل إلى نهاية هذه القائمة، ومن ثم تُنفذ العملية المتواجدة في مقدمة القائمة.

العامل المهم والمؤثر في هذه الخوارزمية هو طول الشريحة الزمنية، فقيمتها تكون في العادة لجميع العمليات أقل من قيمة زمن نبضة وحدة المعالجة، وأطول بكثير من زمن تبديل سياق العمليات، أي زمن الانتقال من عملية إلى أخرى، والذي يتطلب بعض من زمن وحدة المعالجة المركزية لغرض الإشراف على حفظ وتحميل السجلات، وخرائط الذاكرة الخاصة بكل عملية، وكذلك تحديث مختلف الجداول والقوائم الخاصة بذاكرة كاش، وهلم جرا. يتسبب هذا الزمن في ضياع وقت المعالج، بالتالي يجب التقليل من حالات تبديل السياق كلما أمكن ذلك من خلال اختيار القيمة المناسبة لهذه الشريحة.

على سبيل المثال، لو افترضنا أنَّ القيمة الزمنية لهذه الشريحة هو 4 ملِّي ثانية، وأنَّ زمن تبديل السياق يستغرق 1 ملِّي ثانية، فهذا يعني أن 20٪ من وقت وحدة المعالجة المركزية سوف يضيع في أعمال الإشراف، بينما ستكون فاعلية وحدة المعالجة المركزية فقط 80٪. لتحسين هذه الفاعلية نحتاج إلى زيادة قيمة الشريحة مثلًا، لو تم ضبطها على 100 ملِّي ثانية فستكون فاعلية وحدة المعالجة المركزية 99.9٪، أي أن الوقت الضائع من زمن الوحدة سيكون أقل من 1٪. ولكن، وبالنظر إلى ما سوف يحدث عندما تكون هناك 20 عملية جاهزة للاشتغال في قائمة الانتظار وكانت وحدة المعالجة غير منشغلة، نجد أن أول عملية تشتغل مباشرةً، بينما ستشتغل العملية الثانية بعد 100 ملِّي ثانية، في حين أن آخر عملية ستنتظر زمنًا وقدره ثانيتين من أجل الحصول على حصتها من وحدة المعالجة المركزية، كل هذا بفرضية أن كل العمليات السابقة استغلت شريحتها الزمنية بالكامل.

خلاصة القول، كلما طالت الشريحة الزمنية، كلما أثر ذلك سلبًا على زمن الاستجابة، وخصوصًا لأصحاب الطلبات الموجودة في نهاية القائمة، بينما كلما قصرت، كلما كانت الخدمات أفضل، ولكن على حساب زيادة حالات تبديل السياق ومن ثم تقليل الفاعلية. تستخدم معظم أنظمة التشغيل الحديثة في زمن يتراوح ما بين 10 إلى 100 ملِّي ثانية كطول للشريحة الزمنية، وهي غالبًا ما تكون حلًا وسطًا ومقبولًا. أمَّا فيما يخص زمن تبديل السياق فهو في الغالب أقل من 10 ميكرو ثانية، وهو ما يُمثل جزء صغير من الشريحة الزمنية. من مآخذ هذه الخوارزمية أنها تفترض في أن جميع المستخدين يتمتعون بنفس الأهمية.

مثال توضيحي:

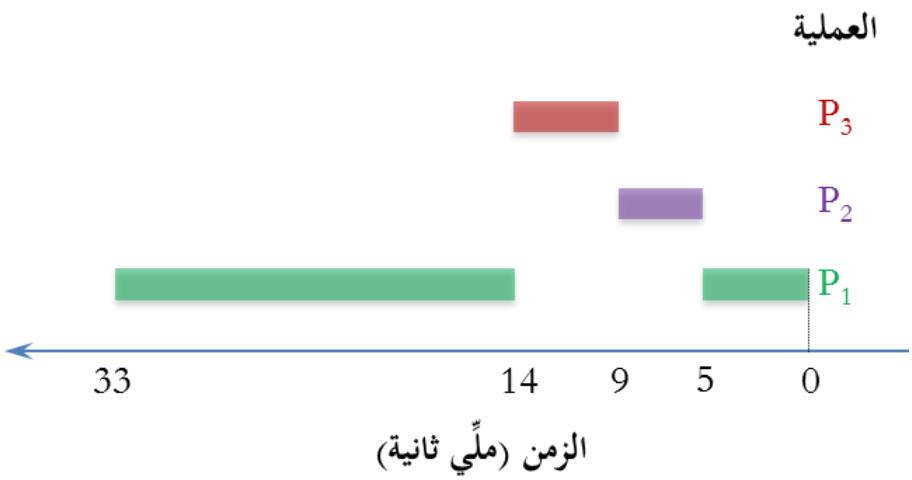
إذا كانت هناك ثلاث عمليات P_1، P_2، و P_3 وصلت إلى مركز الحاسوب تقريبًا في نفس الوقت وتشتغل على حسب الفترات الزمنية 24، 4، و 5 ملِّي ثانية على التوالي، وكانت قيمة الشريحة الزمنية تساوي 5 ملِّي ثانية، فأوجد قيمة كل من:

- متوسط الفترة الزمنية المستغرقة في التنفيذ،
- متوسط زمن الانتظار.

يوضح الشكل 2. 39 الكيفية التي تعمل بها خوارزمية راوند روبن في تنفيذ هذه العمليات.

من خلال هذا الشكل ومن خلال العلاقات الحسابية الموضحة سابقًا، نجد أن:

- متوسط الفترة الزمنية المستغرقة في التنفيذ = (33 + 9 + 14) ÷ 3 = 18.67
- متوسط زمن الإنتظار = [(33 – 24) + (9 – 4) + (14 – 5)] ÷ 3 = 7.67



الشكل 2. 39: مثال لتنفيذ ثلاث عمليات باستخدام خوارزمية راوند روبن.

2.5.5.2 خوارزمية الضامن للجدولة

تهدف خوارزمية الضامن للجدولة إلى تحقيق مبدأ الإنصاف بين العمليات من خلال حفظ وتتبع حصة كل عملية من وحدة المعالجة المركزية منذ إنشائها، ومن ثم تخصيصها وفقًا لذلك. بمعنى إذا كان هناك n من العمليات المسجلة في أثناء استخدام النظام، فإن كل منها يجب أن تتحصل على حوالي $\frac{1}{n}$ من زمن وحدة المعالجة المركزية– الأمر الذي يبدو عدلًا بما فيه الكفاية– تُسمى هذه الحصة من الزمن بالزمن الافتراضي للعملية.

أيضًا من السهل إلى حد ما حساب نسبة الزمن الفعلي لوحدة المعالجة المركزية المستهلك من العملية منذ إنشائها، والذي يُعرف بالزمن الفعلي للعملية. إذًا بالإمكان الآن حساب نسبة الزمن الفعلي إلى الزمن الافتراضي، بحيث تعني النسبة 0.5 أن العملية استهلكت فقط نصف حصتها الافتراضية، أي نصف ما كان ينبغي لها أخذه، وتعني نسبة 2.0 أن العملية أخذت ضعف وقتها الافتراضي، أي ضعف ما كان يحق لها. بالتالي تنص خوارزمية الضامن للجدولة على تشغيل

العملية ذات النسبة الأقل إلى أن تصبح نسبتها فوق أقرب منافسيها.

3.5.5.2 خوارزمية اليانصيب للجدولة

لدى خوارزمية جدولة اليانصيب نهجًا مختلفًا جدًا في عمليات الجدولة تُحاول من خلاله معالجة تعقيدات تنفيذ خوارزمية الضامن للجدولة، وإعطاء نتائج توقعية على نحو مماثل مع أكثر بساطة في التنفيذ. الفكرة الأساسية لهذه الخوارزمية تتمثل في إعطاء كل عملية ضمن قائمة العمليات الجاهزة بعض من تذاكر اليانصيب، ومن ثم تختار الخوارزمية فائزًا بناءً على رقم أُنشأ عشوائيًا. تُمنح العملية الفائزة جائزة تتمثل في الحصول على فترة زمنية للتنفيذ، والتي قد تكتمل فيها تنفيذ العملية، أو قد تنتهي صلاحية الفترة قبل اكتمال تنفيذها، لذلك تُرسل من جديد إلى قائمة العمليات الجاهزة. تُعاد الكرة من جديد لاختيار فائز آخر من القائمة، ولتلقي جائزته، ومن ثم تستمر الدورة.

على الرغم من أن كل عملية ستحصل على تذكرة واحدة على الأقل في كل مرة يُختار فيها فائز، إلَّا إنَّ أكثر العمليات أهمية يمكن إعطائها تذاكر إضافية، وذلك لزيادة فرصة اختيارها للفوز. مثلًا، إذا خُصصت 20 تذكرة من أصل 100 لأي عملية، ستكون لديها فرصة 20٪ للفوز في كل مرة، أي ستتحصل هذه العملية على نحو 20٪ من وحدة المعالجة المركزية، بينما لو خُصِّص لها 60 تذكرة، فستكون فرصتها 60٪. تُشبه هذه الطريقة جدولة الأولوية من حيث أنها تُعطي الأولوية لما يُعتقد أنها عمليات مهمة، إلَّا إنَّها أكثر إنصافًا، لأنها تمنح كل عملية على الأقل فرصة (وإن كانت صغيرة) للفوز بالفترة الزمنية، وبالتالي تجنب مفهوم المجاعة في جدولة اليانصيب.

جدولة اليانصيب لديها العديد من الخصائص المثيرة للإهتمام، فهي تتميز بالرشاقة مع زيادة عدد العمليات، بمعنى تؤثر إضافة العمليات أو حذفها على جميع النسب الأخرى، بغض النظر عن عدد التذاكر التي تمتلكها العملية. كما أنه إذا ظهرت عملية جديدة وتحصلت على بعض من تذاكر اليانصيب، فإنها من المحتمل أن تكون لديها فرصة للفوز في الجولة القادمة من اليانصيب بما يتناسب مع عدد التذاكر التي تمتلكها. يمكن للعمليات كذلك أن تتبادل تذاكر اليانصيب إذا رغبوا في ذلك. على سبيل المثال، عندما تُرسل عملية زبون رسالة إلى خادم العملية ثم تتوقف،

فبإمكانها منح جميع تذاكرها لخادم العملية، وذلك لزيادة فرصة اختياره للفوز في المرة القادمة، وعندما ينتهي، فسيُرجِّع التذاكر للزبون بحيث يمكنه الاشتغال مرة أخرى.

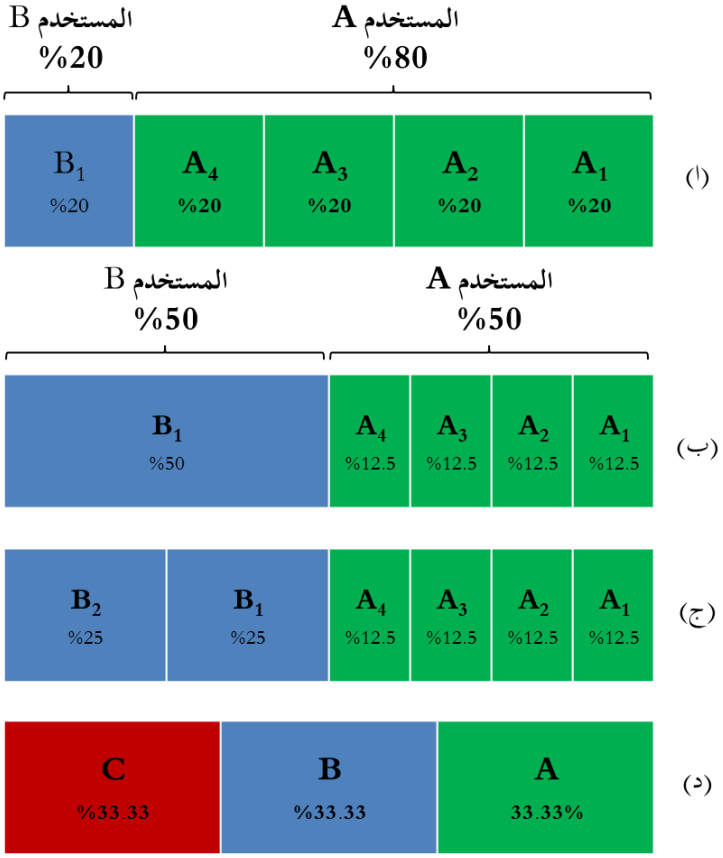
نظرًا للطبيعة العشوائية لجدولة اليانصيب، فإنه لا يُمكن التنبؤ بطبيعة الاختيار القادم، وهو ما يُعتبر عيبًا رئيسيًا فيها. بمعنى لا توجد طريقة لمصممي نظام التشغيل ليعرفوا من خلالها على وجه اليقين ماذا سيحدث في جدولة اليانصيب، الأمر الذي قد ينتج عنه نتائج غير مرغوب فيها. على سبيل المثال، من الممكن أن تتلقى أي عملية مرتبطة بوحدة المعالجة المركزية عدة شرائح زمنية متتالية، مما يسمح لعمليات الإدخال، والإخراج الأخرى بضياع حصصها للحصول على وحدة المعالجة المركزية. لهذا السبب، لا تُستخدم جدولة اليانصيب كثيرًا في تصميم نظم التشغيل الحديثة، على الرغم من قابليتها للتطبيق.

4.5.5.2 خوارزمية الحصة العادلة للجدولة

اعتمدت خوارزميات الجدولة السابقة على العملية نفسها دون النظر إلى صاحب أو مالك العملية. على سبيل المثال إذا كان هناك مستخدم **A** يمتلك أربع عمليات، وآخر **B** يمتلك عملية واحدة فقط في نظام يُطبق خوارزمية راوند روبن للجدولة، فسيتحصل المستخدم **A** على أربعة أضعاف الزمن الذي سيتحصل عليه المستخدم **B**، كما هو موضح في الشكل 2. 40-أ، وستكون سلسلة الجدولة التي تلبي جميع القيود المحددة على النحو $\mathbf{A_1B_1A_2B_1A_3B_1A_4B_1A_1B_1A_2B_1A_3B_1A_4}\ldots$ الأمر الذي قد لا يبدو عدلًا بما فيه الكفاية بالنسبة لهذا المستخدم.

لذلك طور جودي كاي، وبيرز لاودر في جامعة سيدني في الثمانينيات لأول مرة خوارزمية الحصة العادلة للجدولة والمتمثلة فكرتها في توزيع استخدام وحدة المعالجة المركزية بالتساوي بين مستخدمي النظام أو المجموعات بغض النظر عن عدد العمليات التي يمتلكها كل مستخدم، كما هو موضح في الشكل 2. 40-ب، على العكس من التوزيع المتساوي للشرائح الزمنية بين العمليات. بالتالي ستكون سلسلة الجدولة الجديدة على النحو $\mathbf{A_1A_2B_1A_3A_4B_1A_1A_2B_1A_3A_4}\ldots$ وبناءً عليه هناك العديد من الاحتمالات الممكن اتباعها لتحقيق العدالة المنشودة.

نُلاحظ في الشكل 2. 40–ب تَعادل حصة مستخدمي النظام بالرغم من أن المستخدم **B** يمتلك عملية واحدة، بينما يمتلك المستخدم **A** أربع عمليات تتوزع عليهم الشريحة الزمنية بنسبة 12.5٪ لكل واحدة منهم من إجمالي حصة وحدة المعالجة المخصصة لهذا المستخدم. بالمقابل يُوضح الشكل 2. 40–ج حالة النظام عندما بدأ المستخدم **B** في تشغيل عملية ثانية والتي ستتحصل على 25٪ من إجمالي شريحته الزمنية. من ناحية أخرى، إذا بدأ مستخدم ثالث في استخدام النظام، فسوف يُعيد المجدول توزيع دورات وحدة المعالجة المركزية المتاحة بحيث يتحصل كل مستخدم على ثلث هذه الدورات، كما هو مبين في الشكل 2. 40–د.



الشكل 2. 40: توزيع الحصص بين المستخدمين: أ) باستخدام جدولة روبن– ب–ج) باستخدام جدولة الحصة العادلة.

6.5.2 الجدولة في أنظمة الزمن الحقيقي

تختلف الجدولة في أنظمة الزمن الحقيقي عن الجدولة في النظم التفاعلية في كونها على علم مسبق بأن عمليات الزمن الحقيقي لن تعمل لفترات زمنية طويلة وعادة ما تُنهي عملها بسرعة. لذلك قد يتعارض استخدام خوارزميات الجدولة الاستباقية مع طبيعة عمل هذه النظم. فنظام الزمن الحقيقي هو أحد الأنظمة الذي يلعب دورًا أساسيًا في تعاملات الحواسيب الخارجية، كتلك التي تُولد محفزات للتحكم في أحداث معينة، ومن ثم يتحتم على الحاسوب التفاعل معها بشكل آني في غضون فترة محددة من الزمن. من أمثلة هذه الأنظمة نظام مراقبة المرضى في وحدة العناية المركزة في المستشفي، ونظام الطيار الآلي في الطائرات، ونظام التحكم الآلي في المصانع الآلية، وغيرها من الأنظمة الأخرى. في جميع هذه الحالات، حتى ولو توفرت الاستجابة الصحيحة، إلَّا إنَّ توفرها بعد فوات الأوان في كثير من الأحيان يكون كارثة. لذلك تُركز مجدولات هذا النوع من النظم على الوفاء بجميع المواعيد النهائية، كما أنها لا تهتم كثيرًا بالعدالة في تنفيذ العمليات أو بزيادة إنتاجية النظام.

تُصنف أنظمة الزمن الحقيقي بناءً على أهمية تطبيقاتها إلى أنظمة الزمن الحقيقي المرن، وإلى الثابت. فالأولى تعني الضياع العرضي لأي موعد نهائي غير مرغوب فيه، إلَّا إنَّه من الممكن تحمله، ولكنَّ الأداء قد يتدهور إذا تم تجاوز الكثير منها. من أمثلة هذا النوع من الأنظمة أنظمة الأجهزة المنزلية، والترفيهية. أمَّا الأنظمة الثابتة فتعني أن هناك مواعيد نهائية مطلقة يجب الوفاء بها ولا يجوز تجاوزها بأي حال من الأحوال، وإلَّا ستتسبب في حدوث كارثة وخسارة مالية كبيرة. يمكن استخدام هذا النوع في أنظمة الزمن الحقيقي الخاصة بالتحكم في الطائرات، ومحطات الطاقة.

يتحقق السلوك الزمني الحقيقي في كلتا الحالتين من خلال تقسيم برامج النظام الحقيقي إلى عدد من العمليات، كل منها لها سلوك زمني يُمكن التنبؤ به ومعروف مسبقًا. هذه العمليات عادة ما تكون قصيرة الأجل ويُمكن تشغيلها بسهولة إلى النهاية في زمن أقل من الثانية. عندما يكتشف النظام حدث خارجي، يأتي دور المجدول في جدولة العمليات بحيث يُستجاب لها، ويتم استيفاء جميع المواعيد النهائية الخاصة بها. تُقسم هذه الأحداث إلى دورية، تتكرر على فترات منتظمة، وإلى غير دورية، تحدث بشكل غير متوقع مرة واحدة فقط، أو قد تحدث عدة مرات بمرور

الوقت، وكذلك إلى أحداث متفرقة، يُمكن أن تظهر في كثير من الأحيان، ولكن ليس بطريقة ثابتة أو منتظمة.

ولأن الأحداث الدورية تتكرر بشكل منتظم، وجب على النظام أن يكون قادرًا على الاستجابة لهذه الأحداث كلما أمكن ذلك، وذلك اعتمادًا على مقدار الوقت الذي يتطلبه كل حدث للمعالجة، فأحيانًا قد لا يكون من الممكن التعامل معها جميعًا. مثال ذلك، إذا كان هناك عدد $\mathbf{m}$ من الأحداث الدورية، والحدث $\mathbf{i}$ يحدث في الدورة $\mathbf{P_i}$ ويتطلب $\mathbf{C_i}$ ثانية من وقت وحدة المعالجة المركزية للتعامل مع كل حدث، بالتالي لا يمكن التعامل مع هذا السيل من الأحداث إلَّا إذا كان نظام الزمن الحقيقي قابل للجدولة، بمعنى تحقق الشرط: $\sum_{i=1}^{m} C_i/P_i \leq 1$.

مثال توضيحي:

إذا افترضنا وجود نظام زمن حقيقي مرن يتعامل مع ثلاثة أحداث دورية ذات دورات زمنية قدرها 150، 300، و 450 ميلّي ثانية، على التوالي، وكانت هذه الأحداث تتطلب زمن وقدره 40، 50، و 120 ملّي ثانية من وقت وحدة المعالجة المركزية للحدث الواحد، على التوالي. فإن هذا النظام يُعتبر قابل للجدولة وذلك لأن $0.27 + 0.17 + 0.27 \geq 1$. إذا أُضيف حدث رابع بدورة زمنية قدرها 100 ملّي ثانية، فإن النظام سيبقى قابل للجدولة طالما هذا الحدث لا يحتاج إلى أكثر من 290 ملّي ثانية من وقت وحدة المعالجة المركزية. فُرِض في هذه الحسابات أن زمن عملية تبديل السياق صغير جدًا بحيث يُمكن إهماله.

أخيرًا، من الممكن أن تكون خوارزميات الجدولة في الوقت الفعلي ثابتة أو ديناميكية بناءً على توقيت اتخاذ قرارت الجدولة. يعتمد النوع الأول على اتخاذ هذه القرارات قبل بدء تشغيل النظام، بمعنى القيام بتحليل خصائص كل عملية، وتحديد الترتيب الدقيق لتنفيذها من قبل مصمم النظام بعد التوفر المسبق لكافة المعلومات المثالية حول العمل الذي يتعين القيام به، والمواعيد النهائية التي يجب الوفاء بها، والتي لن يتغير ترتيب تنفيذها في أثناء تنفيذ الجدولة. بالمقابل تعمل الجدولة الديناميكية على اتخاذ قرارات الجدولة في وقت التشغيل، أي بعد بدء عملية التنفيذ. ولعدم توفر المعلومات سالفة الذكر هناك مرونة في التقيد بالمواعيد النهائية، كما أنها

قابلة للتكيف حسب متطلبات التنفيذ، أي أن ترتيب تنفيذ العمليات المجدولة يتغير باستمرار.

7.5.2 مقارنة بين خوارزميات جدولة المعالجات

من خلال الاطلاع عل جدولة المعالجات، لاحظنا أن هذه العملية تتضمن العديد من خوارزميات الجدولة المختلفة. يُلخص الجدول 2. 8 مزايا وعيوب كل خوارزميات الجدولة التي درسناها حتى الآن.

الجدول 2. 8: مقارنة لمزايا وعيوب بعض من خوارزميات الجدولة.

الخوارزمية	النوع	المزايا	العيوب
القادم أولًا يُخدم أولًا	غير استباقية	بسيطة جدًا، سهلة التنفيذ، تُحقق العدالة.	متوسط انتظار طويل، استغلالية منخفضة، إنتاجية منخفضة.
أقصر وظيفة أولًا	غير استباقية	متوسط انتظار قصير، إنتاجية عالية.	تحتاج معلومات استباقية، تُعاني من المجاعة.
أقصر الأوقات المتبقية تاليًا	استباقية	تحسن في زمن الاستجابة، لا تُعاني من المجاعة.	تحتاج معلومات استباقية، توقيف عملية تكاد تكتمل واستبدالها بعملية حديثة ذات زمن تشغيل أقصر.
الأولوية	غير استباقية/ استباقية	تُوفر آلية جيدة عندما تكون الأهمية محددة بدقة.	تُعاني العمليات ذات الأولوية الأقل من المجاعة، صعوبة توزيع مستوى الأولويات.
راوند روبن	استباقية	حصة التنفيذ متساوية، لا توجد مجاعة.	متوسط انتظار طويل، نفقات عالية وكفاءة منخفضة بسبب قصر طول الشريحة، ضعف الاستجابة بسبب طول الشريحة.
الضامن	استباقية	تُحقق العدالة.	معقدة.
اليانصيب	استباقية	بساطة في التنفيذ،	صعوبة التنبؤ بطبيعة الاختيار القادم،

		أكثر عدلًا من الأولوية، لا توجد مجاعة، تسمح بتبادل التذاكر.	قد تتحصل عملية– ما– على عدة شرائح زمنية متتالية، صعوبة توزيع التذاكر.
الحصة العادلة	استباقية	تُحقق العدالة بالنسبة للمستخدمين، تحسن في زمن الاستجابة، تخسن في إنتاجية المصادر، تحسن في أداء النظام.	مساواة المستخدمين، بالرغم من عدم تكافؤهم من حيث عدد العمليات المملوكة لكل مستخدم.

6.2 موجز الفصل

بعد أن إنتهت رحلة المقدمة لهذا الكتاب، يأتي الفصل الثاني ليشرع في تقديم نموذج العملية باعتبارها الأساس التي ترتكز عليه جميع مكونات نظم التشغيل. بدأ هذا النموذج في بداية اختراع الحاسوب بالقدرة على التنفيذ المتتالي للعمليات، أي واحدة فقط في كل مرة، ومن ثم انتقل مع تطور الحاسوب إلى التوازي الوهمي، وإلى التوازي الحقيقي على حساب تعقيدات كل من الكيان المادي، والبرمجي للحاسوب. الأمر أصبح أكثر تعقيدًا إلى درجة أن العملية الواحدة فُكِّكت إلى مجموعة من الخيوط قابلة لأن تُنفذ بشكل متوازي. لذلك تتطرق هذا الفصل لدراسة العمليات والخيوط والتعرف على الكيفية التي تُدار بهما من قبل إدارة العمليات والخيوط، وعلى الاتصالات الداخلية لهما، وعلى المشاكل المصاحبة لمثل هذه الاتصالات.

بدأت هذه الإدارة بتقديم تعريف للعملية والذي نص على أنها تُمثل البرنامج لحظة تنفيذه متضمنةً بما في ذلك جميع الملحقات الخاصة بمرحلة التنفيذ، وبيَّنت أن هذه العملية يُمكن إنشاؤها، وإنهاؤها ديناميكيًا، وأنها تمر عبر سلسة من الحالات والانتقالات عُرفت بحالات العملية والتي شملت حالة جديدة، وجاهزة، وتشتغل، وموقوفة، وأخيرًا منتهية. كما أنها، أي الإدارة تستخدم جدول العملية والمتكون من جميع قوالب التحكم، وكافة المعلومات المهمة حول كل عملية من أجل تنفيذ العمليات الحالية بالنموذج والتحكم فيها بشكل صحيح.

تتعاون العميات وتتواصل فيما بينها في أثناء عمليات التنفيذ من أجل القيام بالمهام المنوطة بهم، إلَّا إنَّ هذا التعاون وهذا التواصل لا يخلوا من المشاكل، وخصوصًا عندما يتعلق الأمر بالبيانات المشتركة. تتمثل إحدى هذه المشاكل في مشكلة وضع التسابق والتي عُرِّفت بأنها الحالة التي تعتمد النتيجة النهائية فيها على دقة تنفيذ العمليات التي تستخدم أجزاء مشتركة. ولغرض مناقشة هذه المشكلة والتعرف على حلولها، قدَّم الفصل أولًا مفهوم المنطقة الحرجة عن طريق الاستدلال بتقاطع طرق السيارات، ومن ثم عرفها على أساس الجزء من البرنامج الذي يؤدي إلى حدوث مشكلة وضع التسابق، وبيَّن أن حل هذه المشكلة يرتكز في الأساس على منع تواجد أكثر من عملية واحدة داخل مناطقهم الحرجة في آن واحد، بالإضافة إلى ضرورة تحقق شرط المنع التبادلي، وشرط غياب المجاعة، وشرط عدم وجود طريق مسدود، وألَّا يكون الحل مبني على فرضيات حول السرعة النسبية للعملية، أو عدد وحدات المعالجة.

بعدها ناقش الفصل مجموعة من الحلول المقترحة لحل مشكلة وضع التسابق تمثلت في طرق تعتمد على مفهوم الانتظار المشغول منها تعطيل المقاطعات، ومتغير القفل، والتناوب الدقيق، وخوارزمية بيترسون، وأمر اختبار القفل، ثم انتقل إلى شرح مفهوم حل المنوم والمنبه، وكذلك منظم دخول العمليات من خلال مناقشته لحل مشكلة المنتج والمستهلك في كلتا الحالتين. جميع هذه الطرق تناولها الفصل بنوع من التفصيل الموضعي وبأمثلة توضيحية تناولت عدة جوانب من المشكلة.

بعد تقديم العملية، والاتصالات الداخلية لها، تتطرق الفصل إلى موضوع الخيوط وبَيَّن أنها عبارة عن عمليات خفيفة الوزن تعمل في نفس فضاء عنونة العملية الأم ولكل منها مُعرَّف فريد يُميزه عن بقية خيوط العملية وله عدة سمات وبيانات خاصة به مشابهة لبيانات العملية. تعرض الفصل كذلك بنوع من التفصيل لأهم الاختلافات بين الخيوط والعمليات، ولأنواعها المتمثلة في خيوط المستخدم، وخيوط النواة، ومن ثم عرَّج على استخداماتها وفوائدها والتي من بينها تحسين الإنتاجية، وزيادة نسبة التنفيذ المتوازي، وتقليل زمن تبديل السياق، وتحسين استجابة البرامج، كما اختتم الفصل هذه الجزئية بتقديم أمثلة واقعية لاستخدامات الخيوط.

يلي ذلك عرض الفصل مفهوم كل من نموذج الخيط التقليدي المتمثل في وجود خيط واحد فقط في فضاء العملية، ونموذج الخيوط المتعددة المتمثل في احتواء فضاء العملية على عدة

خيوط، والتي تتعاون وتتواصل مع بعضها من أجل تنفيذ المهمة المطلوبة، بالتالي فهي عرضة للمرور بنفس الحالات التي تمر بها العمليات. وكتكملة لهذه الجزئية تناول الفصل خيوط POSIX والتي تُمثل مجموعة من المعايير التي نُفذت بشكل أساسي لأنظمة التشغيل المستندة إلى نظام 'يونكس' كمثال لاستدعاءات الخيوط.

ومع تنوع الخيوط تنوعت معها آليات تنفيذها، لذلك فصَّل الفصل طرق إنجاز الخيوط إلى تنفيذها على مستوى فضاء النواة، وعلى مستوى فضاء المستخدم، ومن ثم عرَّج على الإنجاز الهجين لها. تُنفذ الخيوط في الطريقة الأولى في النواة، وتُدار مباشرةً من هناك من قبل نظام التشغيل، بينما تُوضع حزمة الخيوط في الطريقة الثانية كليًا في فضاء المستخدم، وتُدار العمليات بالشكل المعتاد في فضاء النواة على أساس عمليات ذات خيط واحد. أمَّا الطريقة الثالثة فهي تجمع بين الطريقتين السابقيتن لغرض الاستفادة من مزايا الخيوط في كلتا الحالتين. بالإمكان أيضًا الاستفادة من الخيوط بتطبيق فكرة الخيوط المنبثقة والتي تنبثق فقط عند الحاجة كتلك الخيوط التي تتعامل مع الرسائل الواردة عبر الشبكات.

بعد ذلك عاد الفصل ليعطي تفاصيل أكثر حول نماذج الخيوط المتعددة والتي تأخذ عدة علاقات شائعة تمثلت في: علاقة عديد– لعديد، وعلاقة عديد– لواحد، وأخيرًا علاقة واحد– لواحد. ترتبط خيوط عمليات المستخدم في العلاقة الأولى بعدد مساو أو أصغر من خيوط عمليات النواة، بينما تُخصص عدة خيوط على مستوى عمليات المستخدم في العلاقة الثانية لخيط واحد فقط على مستوى عمليات النواة، في حين يُعين خيط واحد على مستوى عمليات المستخدم في العلاقة الأخيرة مقابل خيط واحد مناظر له، ومنفصل على مستوى عمليات النواة. ولمعالجة بطأ الخيوط على مستوى النواة في بعض من التطبيقات ظهر مفهوم تنشيط الجدولة لغرض محاكاة وظائف خيوط النواة، ومن ثم تحسين أداء التعامل مع حزم خيوط المستخدم.

بعد أن تتطرق الفصل لموضوعي العمليات، والخيوط بشرح مفصل، انتقل ليتناول موضوع جدولة المعالجات والذي يُمثل حجر الأساس لأنظمة التشغيل بصفة عامة، ولأنظمة البرمجة المتعددة بصفة خاصة. بدأ هذا الفصل هذه الجزئية بتوضيح المفهوم العام للجدولة، ومن ثم قدَّم أهداف ومعايير الجدولة لثلاث بيئات تشغيل مختلفة هي: بيئة أنظمة الدفعة، وبيئة الأنظمة التفاعلية، وبيئة أنضمة الزمن الحيقيقي، أتبعها بتبيان أهم الأهداف العامة لهذه الجدولة والمتمثلة

في العدالة، وتحقيق التوازن لاستغلالية مكونات الحاسوب، بالإضافة إلى تطبيق السياسة المعلنة للنظام.

كما بين الفصل أيضًا الحالات التي يُتخذ فيها قرارات الجدولة، والتي ارتبطت بإنشاء، وإنهاء، وحظر العمليات، أو بمقطاعات الإدخال، والإخراج، ومن ثم بين أن الجدولة تنقسم إلى جدولة استباقية، وأخرى غير استباقية. تسمح الأولى بتبادل تنفيذ العمليات حتى ولو لم تنتهي من تنفيذ مهمتها بالكامل، بينما تشترط الثانية إنتهاء العملية من التنفيذ لكي تُستبدل بعملية أخرى، باستثناء العمليات المرتبطة بالإدخال، والإخراج والتي قد تظطر إلى تعليق مهمتها بسبب انتظارها لأحداث خارجية.

أخيرًا، قدَّم الفصل شرح مستفيض لمجموعة من خوارزميات جدولة المعالجات بوبها في ثلاثة أقسام، ودعمها بعدة أمثلة توضيحية. ضم القسم الأول منها الجدولة في أنظمة الدفعة وتناولت خوارزمية القادم أولًا يُخدم أولًا، وخوارزمية أقصر وظيفة أولًا، وخوارزمية أقصر الأوقات المتبقية تاليًا، وخوارزمية الأولوية، بينما عرض القسم الثاني الجدولة في الأنظمة التفاعلية وشملت خوارزمية راوند روبن، وخوارزمية الضامن، وخوارزمية اليانصيب، وخوارزمية الحصة العادلة، في حين تناول القسم الأخير الجدولة في أنظمة الزمن الحقيقي وبين أنها تتفرع إلى الجدولة المرنة، وإلى الجدولة الثابتة. فالأولى تعني الضياع العرضي لموعد– ما– غير مرغوب فيه، ولكن قد يتدهور الأداء إذا ما تم تجاوز الكثير منها، أمَّا الثانية فتعني أن هناك مواعيد نهائية مطلقة يجب الوفاء بها، وإلَّا لا فائدة منها. في النهاية أُختتم الفصل بمقارنة لكافة خوارزميات الجدولة التي تناولها الفصل.

7.2 أسئلة للمراجعة

1. في إطار معالجة وتنفيذ العمليات، ما المقصود بالتوازي الوهمي؟ ومالفرق بينه وبين التوازي الحقيقي؟
2. ما المقصود بقالب التحكم في العملية؟ وما فائدته؟
3. ما المعلومات الموجودة في قالب التحكم في العمليات؟ وما الغاية والغرض منها؟

4. عدد حالات إنشاء العمليات، وإنهائها.

5. لماذا يجوز للعملية الأب إنهاء أو إجهاض أحد أبنائها؟

6. عرف العملية، مع تبيان الحالات التي تمر بها العمليات، موضحًا اجابتك بالرسم وذكر أهم الانتقالات التي تحدث لها.

7. بالرجوع إلى مخطط حالات العملية، هل من الممكن انتقال العمليات من حالة جاهزة إلى حالة موقوفة؟ علل اجابتك؟

8. تتيح البرمجة المتعددة (أو المهام المتعددة) تنفيذ أكثر من عملية واحدة في وقت واحد. كيف يُمكن تحقيق ذلك على معالج أحادي؟

9. اشرح مشكلة وضع التسابق من خلال شرحك لعملية طباعة الملفات باستخدام مكب الطباعة.

10. عرف المنطقة الحرجة، وبين أهميتها في تفادي مشكلة وضع التسابق.

11. اذكر واشرح الطريقة التي عُولجت بها مشكلة الانتظار المشغول في كل من حل بيترسون، وأمر اختبار وضبط القفل.

12. هل عندما يكون بالإمكان منع العمليات من التواجد داخل مناطقها الحرجة في آن واحد، فإن ذلك سيكون كافيًا لمنع حدوث مشكلة وضع التسابق؟ وضح اجابتك.

13. تُعتبر طريقة تعطيل المقاطعات حلًا لتحقيق المنع التبادلي، ما هي مزاياه وعيوبه؟

14. لماذا لا يُوصف حل التناوب الدقيق بالحل المثالي لمشكلة وضع التسابق؟

15. اشرح بالتفصيل حل اختبار وضبط القفل، مع توضيح المشاكل التي يُعاني منها هذا الحل.

16. ما معنى مصطلح الانتظار المشغول؟ ولماذا هو حالة غير مرغوب فيها؟ وهل يمكن تجنبه تمامًا؟ اشرح اجابتك.

17. بَيَّن القسم 4.3.2 أن الوضعيات التي تتواجد فيها عمليات ذات أولويات عالية، وأخرى ذات أولويات منخفضة تكون عرضةً لحالة الانتظار المشغول. هل ستحدث نفس المشكلة

إذا أُستُخدم مجدول روبن بدلًا من مجدول الأولوية؟ ناقش ذلك.

18. اذكر مزايا استخدام منظم دخول العمليات لتحقيق المنع التبادلي مقارنة بحل خوارزمية بيترسون.

19. وضح كيف يمكن استخدام منظم دخول العمليات الثنائي لتنفيذ المنع التبادلي بين العمليات.

20. بَيِّن لماذا تُنفذ عمليتي منظم دخول العمليات **down** و **up** كعمليات غير قابلة للتجزئة.

21. وضح مشكلة المستهلك والمنتج، ثم أعطي مثالًا على حدوثها في أنظمة التشغيل.

22. عرف الخيط، ثم اذكر الفرق بينه وبين العملية؟ ومن ثم وضح بعض من فوائد الخيوط.

23. عدد الفروق بين كل من خيوط المستخدم وخيوط النواة.

24. ما مزايا وعيوب إنجاز الخيوط في كل من فضاء المستخدم وفضاء النواة؟

25. بَيِّن ثلاثة أمثلة لبرامج تطبيقية تُقدم فيها استخدام الخيوط المتعددة أداءً أفضل من استخدام خيط واحد.

26. صف الأحداث التي تُتخذ عندما تقوم النواة بتبديل السياق بين خيوطها.

27. تتشارك الخيوط في فضاء العنونة وغيرها من المصادر ولا توجد حماية للذاكرة بينها، لماذا لا نحتاج هنا إلى تحوطات الحماية؟

28. ما أهم معايير وأهداف كل من خوارزميات الجدولة في كل من أنظمة الدفعة، الأنظمة التفاعلية، وأنظمة الوقت الحقيقي؟

29. لماذا يُفضل استخدام الجدولة الاستباقية في النظم التفاعلية ولا يُفضل استخدامها في أنظمة الوقت الحقيقي؟

30. اذكر الفروق بين خوارزميات الجدولة الاستباقية وغير الاستباقية، وأيٍ منها يتناسب مع أنظمة المشاركة الزمنية؟

31. وضح مفهوم زمن تبديل السياق بالنسبة للعمليات، وناقش ماذا سيحدث إذا كلَّف زمنًا طويلًا؟

32. بفرض أن هناك خمس عمليات من P_1 إلى P_5 وصلت في نفس الوقت إلى مركز الحاسوب من أجل المعالجة، وكان زمن الاشتغال المتوقع لكل عملية هو 8، 10، 3، 15، و 5 جزء من الثانية، وكانت الأولويات المسندة لكل عملية على النحو التالي: 7، 9، 5، 3، و8 على التوالي (9 أعلى أهمية). احسب قيمة متوسط زمن الانتظار، ومتوسط الفترة الزمنية المستغرقة في التنفيذ في حالة استخدام كل من:

أ. خوارزمية راوند روبن (الشريحة الزمنية = 5)،

ب. خوارزمية الأولوية،

ج. خوارزمية أقصر وظيفة أولًا،

33. إذا كان هناك خمس وظائف تنتظر في التنفيذ والأزمنة المتوقعة للاشتغال كانت على النحو التالي: 8، 10، 5، 6، و x. ما هو الترتيب المناسب للتنفيذ والذي يُعطي أقل متوسط لزمن الانتظار؟ (تعتمد اجابتك على x.)

34. إذا كان هناك نظام زمن حقيقي مرن لديه أربع أحداث دورية ذات دورات زمنية 50، 100، 200، و 250 ملّي ثانية، على التوالي، وكانت هذه الأحداث تتطلب 35، 20، 10، و x ملّي ثانية من وقت وحدة المعالجة المركزية للحدث، على التوالي. ما هي أكبر قيمة تحملها x بحيث يصبح النظام قابل للجدولة؟

35. بفرض أن هناك ثلاث عمليات P_1، و P_2، و P_3 تشتغل على حسب الفترات الزمنية 7، 2، و 3 ملّي ثانية، ووصلت في في أزمنة مختلفة إلى مركز الحاسوب على النحو 0، 1، و 4 على التوالي، بين آلية تنفيذ هذه العمليات في ظل استخدام خوارزمية أقصر الأوقات المتبقية تاليًا، ومن ثم أوجد قيمة كل من:

- متوسط زمن الانتظار،
- متوسط الفترة الزمنية المستغرقة في التنفيذ.

36. بفرض أن هناك أربع عمليات P_1، و P_2، P_3، و P_4 تشتغل على حسب الفترات الزمنية 7، 4، 9، و 5 ملّي ثانية، ووصلت في في أزمنة مختلفة إلى مركز الحاسوب على النحو 0،

1، 3، و 4 على التوالي، بين آلية تنفيذ هذه العمليات في ظل استخدام خوارزمية أقصر الأوقات المتبقية تاليًا، ومن ثم أوجد قيمة كل من:

- **متوسط زمن الانتظار،**
- **متوسط الفترة الزمنية المستغرقة في التنفيذ.**

الفصل الثالث وحدات الإدخال، والإخراج

1.3 مدخل إلى إدارة وحدات الإدخال، والإخرج

تُلحق بالحواسيب أنواعًا كثيرةً من الأجهزة، تتلاءم في معظمها مع الفئات العامة لمعدات التخزين (الأقراص، الأشرطة)، معدات الإرسال، والاستقبال (توصيلات الشبكة، البلوتوث)، وأجهزة التفاعل مع الحاسوب (الشاشة، لوحة المفاتيح، الفأرة)، بالإضافة إلى أجهزة أخرى أكثر تخصصًا مثل، تلك المستخدمة في التحكم، وتوجيه الطائرة كعصا التحكم ودواسات القدم، والتي تُرسل للحاسوب أوامر تتسبب في تحريك الدفات والتحكم في كمية الوقود الموجهة إلى المحركات.

على الرغم من هذا التنوع الكبير في أجهزة الإدخال، والإخراج، إلَّا إنَّنا نحتاج فقط إلى بعض من المفاهيم لفهم كيفية توصيل الأجهزة وكيفية التحكم فيها من قبل نظام التشغيل، والذي يُعتبر من أهم وظائفه، فهو الذي يُصدر الأوامر إلى الأجهزة ويتتبع المقاطعات، ويتعامل مع الأخطاء، كما يُوفر الواجهة البينية بين أجهزة الإدخال، والإخراج وباقي أجهزة النظام، شرط أن تكون هذه الواجهة بسيطة وسهلة الاستخدام، بالإضافة إلى ذلك ينبغي أن تكون هذه الواجهة عامة، أي أن تكون مستقلة عن نوعية الجهاز المستخدم، أو ما يُحقق استقلالية الجهاز.

بالتالي سيكون موضوع هذا الفصل هو التعرف على كيفية إدارة نظم التشغيل لوحدات الإدخال، والإخراج من خلال دراسة مفاهيمها الأساسية من الناحية المادية والمعنوية، والتعرض لبعض من الأمثلة لهذه الوحدات، وكذلك لبعض من المشاكل المصاحبة لمثل هذه الأجهزة، كما سيُعَرِّج الفصل على بعض من الحلول المقترحة لمنع وتجنب حدوث هذه المشاكل.

2.3 المفاهيم المادية لوحدات الإدخال، والإخراج

تختلف النظرة إلى المكون المادي لوحدات الإدخال، والإخراج باختلاف مستخدمي الحاسوب، فالمهندس مثلًا، ينظر إلى هذه الوحدات على أساس دارات متكاملة، وأسلاك، ومغذيات قدرة، ومشغلات، وكافة المكونات الفيزيائية المكونة للكيان المادي، في حين أن المبرمج يراها من حيث الواجهة البينية للكيان المعنوي، والأوامر التي يتقبلها الكيان المادي، والوظائف التي تُنفذها وحدات الإدخال، والإخراج، وكذلك الأخطاء التي يمكن الإبلاغ عنها. في هذا الفصل، سيكون اهتمامنا ببرمجة أجهزة الإدخال، والإخراج وليس بتصميمها، أو ببنائها، أو

بصيانتها، بالتالي سيقتصر الاهتمام على كيفية برمجة الكيان المادي، بدلًا من التركيز على الكيفية التي يعمل بها داخليًا. في الأقسام التالية ستُعطى خلفية عامة بسيطة عن الكيان المادي لوحدات الإدخال، والإخراج من حيث صلتها بالبرمجة.

للتواصل مع وحدات الإدخال، والإخراج، نحتاج إلى استخدام نظام الإدخال، والإخراج الذي له هدفان رئيسيان:

- استلام طلب تطبيق الإدخال، والإخراج وإرساله إلى الجهاز الفعلي، ثم استقبال أي رد يأتي من الجهاز وإرساله إلى التطبيق.
- تحسين أداء طلبات الإدخال، والإخراج المختلفة.

بالإضافة إلى ذلك، يجب أن يُوفر نظام الإدخال، والإخراج آليات لتهيئة النظام على سبيل المثال، لإعلام نظام التشغيل بما هو مُوَصل بالضبط بالأجهزة، وكيفية التواصل معها.

3.3 أنواع وحدات الإدخال، والإخراج

تنقسم أجهزة الإدخال، والإخراج في العموم إلى نوعين هما: الأجهزة المقطعية، والأجهزة الحرفية. يتمثل النوع الأول في تلك الأجهزة التي تُخزن فيها المعلومات في مقاطع ذات أحجام ثابتة، وعناوين خاصة، ويتصل بها نظام التشغيل عن طريق إرسال كتل كاملة من البيانات. المدى الشائع لهذه الأحجام يتراوح من 512 إلى 32,768 خانة ثمانية، كما تتم عمليات النقل في الغالب بحجم مقطع أو عدة مقاطع متتالية. من أهم خصائص الأجهزة المقطعية هو إمكانية قراءة وكتابة كل مقطع مستقلًا عن الآخر، تُعتبر الأقراص بجميع أنواعها، وشرائح الناقل التسلسلي العام، وكاميراته من الأمثلة الشائعة لمثل هذا النوع من الأجهزة.

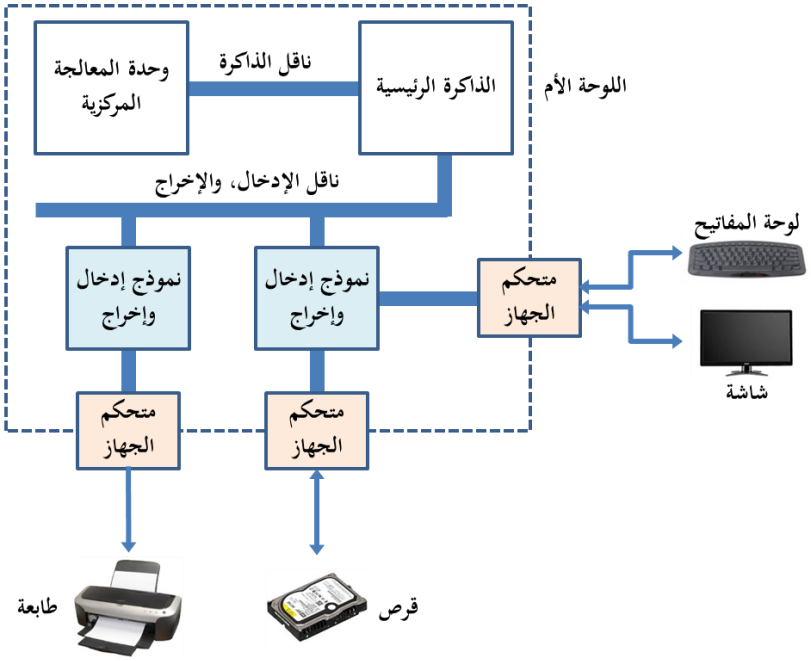
بينما يتمثل النوع الثاني من أجهزة الإدخال، والإخراج في تلك الأجهزة التي تُرسل وتَستقبل سلسلة من البيانات الحرفية من دون استخدام أي بنية مقطعية، ويتصل بها نظام التشغيل عن طريق إرسال وتلقي سلسلة من الأحرف المفردة. من الأجهزة التي تُصنف تحت الأجهزة الحرفية الطابعة، والمنافذ التسلسلية، والمنافذ المتوازية، وبطاقات الصوت.

هناك بعض من الأجهزة التي لا يمكن إدراجها تحت هذين النوعين مثل، مولد النبضة الذي

لا يملك مواقع تخزين معنونة، ولا يُولِّد أو يستقبل أي بيانات حرفية، وكل ما يقوم به هو توليد مقاطعات عند فترات زمنية محددة.

4.3 الكيان المادي لوحدات الإدخال، والإخراج

قبل مناقشة الكيفية التي يُعالج بها نظام التشغيل الإدخال، والإخراج، سيُراجع الفصل الكيفية التي تعمل بها هذه الوحدات. يظهر المخطط العام للحاسوب في الشكل 3. 1، المعلومات الأكثر أهمية في هذا المخطط هي أن لكل جهاز فعلي واجهة جهاز (تُعرف أيضًا باسم متحكم الجهاز)، تعمل واجهات الجهاز هذه على التحكم المباشر في الأجهزة لمساعدة وحدة المعالجة المركزية الرئيسية في القيام بمهامها، سيتم التعرف على متحكم الجهاز بنوع من التفصيل في القسم 2.4.3.



الشكل 3. 1: التركيبة العامة للحاسوب، وتوصيل وحدة المعالجة المركزية بمتحكمات الأجهزة والذاكرة.

من أجل التحكم في الأجهزة تحتوي معظم متحكمات الأجهزة على مستوى ذكاء معين

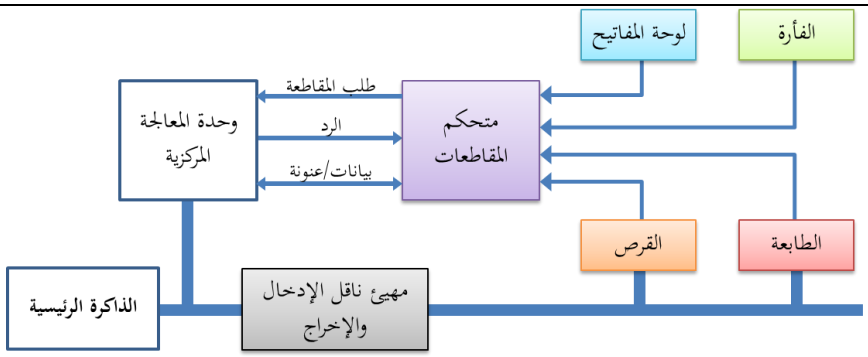
(بمعنى أنها تحوي في حد ذاتها حواسيب مصغرة)، إلَّا إنَّ مستوى الذكاء قد يختلف من متحكم إلى آخر بشكل كبير، ولكن تتفق جميعها في الاحتواء على سمة واحدة مهمة جدًا وهي القدرة على تشغيل الأجهزة في نفس الوقت الذي تقوم فيه وحدة المعالجة المركزية الرئيسية ببعض من المهام الأخرى. بالتالي تكون الخطوات العامة التي تعمل به متحكمات الأجهزة كما يلي:

1. تطلب وحدة المعالجة المركزية الرئيسية من متحكم الجهاز القيام بوظيفة– ما.
2. تستمر وحدة المعالجة في تنفيذ بعض من المهام الأخرى، بينما يُنفذ متحكم الجهاز الوظيفة المطلوبة.
3. عند اكتمال الوظيفة، يُقاطع متحكم الجهاز وحدة المعالجة لإعلامها باكتمالها.
4. مع اكتمال الوظيفة، يستجيب نظام التشغيل ويُخطِر العملية ذات الصلة بذلك.

من خلال دراسة متحكمات الأجهزة وعلاقتها بنظام التشغيل، يمكن ملاحظة الأتي:

1. نظام التشغيل قادر على تحسين أداء النظام الكلي، إذا تمكن من الحفاظ على الأجهزة المختلفة مشغولة قدر الإمكان.
2. من المهم أن يُعالج نظام التشغيل مقاطعات الجهاز بأسرع وقت ممكن لما لذلك من فوائد:
   - أ– بالنسبة للأجهزة التفاعلية (لوحة المفاتيح، أو الماوس، أو لاقط الصوت)، يمكن أن يجعل النظام أكثر استجابةً.
   - ب– بالنسبة لأجهزة الاتصالات (المودم، والإيثرنت، إلخ)، يمكن أن يؤثر ذلك على السرعة الفعالة للاتصالات.
   - ج– بالنسبة لأنظمة الوقت الحقيقي، يمكن أن يكون هذا هو الفرق بين النظام الذي يعمل بشكل صحيح والذي يتعطل.

بالإضافة إلى ذلك هناك عدة أجهزة تعمل في وقت واحد، لذلك من المحتمل أن عددًا منها سوف يرغب في إرسال المقاطعات إلى وحدة المعالجة المركزية في نفس الوقت، كما هو موضح في الشكل 3. 2، عليه يجب أن يكون نظام التشغيل مستعدًا للتعامل مع مثل هذه المواقف على النحو الموضح تاليًا.



الشكل 3. 2: الكيان المادي لمتحكم المقاطعات.

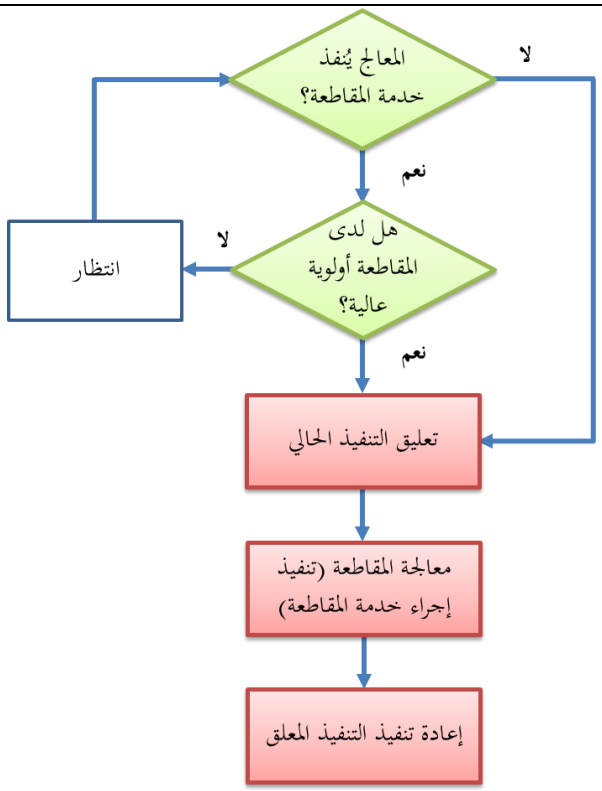
1.4.3 التعامل مع عدة أجهزة في وقت واحد

يحتوي نظام الحاسوب عادةً على عدة أجهزة تعمل في نفس الوقت، الأمر الذي يُؤدي في الغالب إلى حدوث مقاطعات متزامنة (أو شبه متزامنة) بسبب هذه الأجهزة المتعددة. السؤال الذي يطرح نفسه: كيف يتعامل نظام التشغيل مع هذه المقاطعات؟

على مستوى الكيان المادي أُعطيت مستويات في الأولوية للمقاطعات الصادرة من الأجهزة المختلفة وفقًا لسرعة احتياجات وحدة المعالجة المركزية للاستجابة إلى مقاطعة الجهاز، أمَّا على مستوى نظام التشغيل، فهو يحتاج إلى الرد بسرعة على المقاطعة لأحد السببين التاليين:

- للحصول على معلومات من متحكم الجهاز قبل فقدانها.
- لبدء العملية التالية قبل فوات الأوان.

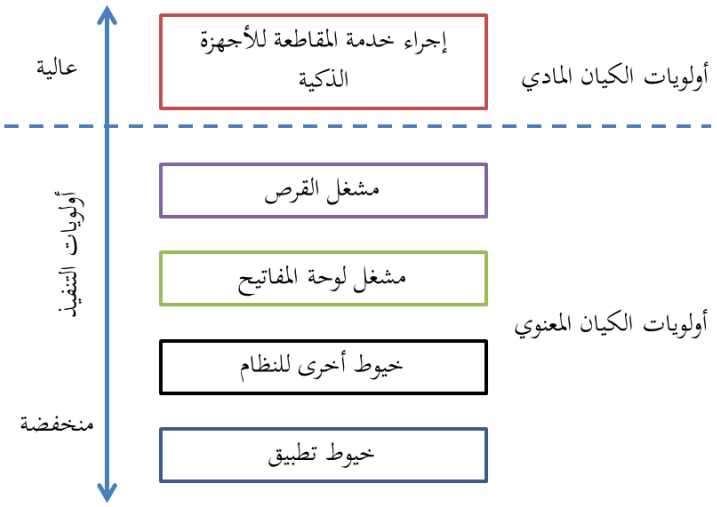
يعمل الكيان المادي في العادة على النحو الموضح في الشكل 3. 3، حيث تُعالج هذه الآلية أولوية إجراءات خدمة المقاطعة، ولأنها تعتمد على الأولوية، فإن إجراءات خدمة المقاطعة تتمتع فعليًا بأولوية تنفيذ عالية في النظام بأكمله، لذلك يجب كتابتها لتنفيذ الحد الأدنى فقط من العمل اللازم لتحقيق الهدفين المذكورين أعلاه، بعدها يجب تمرير الطلب إلى مشغل الجهاز.



الشكل 3. 3: إجراءات المقاطعة المادية.

في أي نظام نموذجي تُنفذ مشغلات الأجهزة كخيوط في عملية النظام مع منحها أولويات أعلى من خيوط العمليات الأخرى. بشكل عام سيكون لدى خيوط مشغلات الأجهزة التي تتعامل مع الأجهزة المهمة مثل، الأقراص أولوية أعلى من الخيوط التي تتعامل مع الأجهزة الأخرى مثل، لوحة المفاتيح. يُوضح الشكل 3. 4 أولويات تنفيذ النظام النموذجية.

نظرًا لأن مشغلات الأجهزة تُنفذ كخيوط، فبإمكانها تأمين مصادر النظام وتنفيذ أنواع أخرى من استدعاءات النظام. من ناحية أخرى تعمل إجراءات خدمة المقاطعة في مجالٍ نِصفهُ يعتمد على الكيان المادي، والنصف الآخر موجه لنظام التشغيل. على هذا النحو، تميل بيئة البرمجة الخاصة بإجراءات خدمة المقاطعة إلى أن تكون مقيدة، أي أنَّ معظم وظائف النظام العادية غير متوفرة، سنعود إلى مناقشة هذا الموضوع بشكل تفصيلي في القسم 5.3.



الشكل 3. 4: أولويات تنفيذ النظام.

2.4.3 متحكم الجهاز

يأخذ متحكم الجهاز في الحواسيب الشخصية، شكل رقاقة في اللوحة المطبوعة أو قد يكون على شكل بطاقة دائرة مطبوعة تُدرج في مجرى خاص بها في اللوحة الأم، فهو يحتوي على فتحة توصيل يتصل بها جهاز الإدخال أو الإخراج عن طريق سلك، مع العلم أن هناك متحكمات أجهزة يمكن توصيل أكثر من جهاز بها. يُستخدم ناقل النظام في أغلب الحواسيب الدقيقة من أجل تحقيق الاتصال بين وحدة المعالجة المركزية والمتحكم، وذلك كما هو موضح في الشكل 3. 1، كما يمتلك كل متحكم بعض من السجلات لغرض استخدامها في الاتصال بوحدة المعالجة المركزية.

من ناحية أخرى كثيرًا ما يكون التفاعل بين وحدة متحكم الجهاز ووحدة الإدخال، والإخراج نفسها على مستوى منخفض جدًا. فالقرص، قد يُهيَّأ بحيث يحوي مليوني قطاع لكل مسار، و 512 خانة ثمانية لكل قطاع، ولكن ومع ذلك ما يأتي من مشغل الجهاز هو تيار من الخانات الثنائية المتتالية، تبدأ بالدباجة، ثم 4096 خانة ثنائية للقطاع، وأخيرًا الجزء الخاص بشفرة تصحيح الأخطاء. تُكتب هذه الدباجة في أثناء تهيئة القرص، وتحتوي على رقم كل من الأسطوانة، والقطاع، وحجم القطاع، وبيانات أخرى مماثلة، وكذلك معلومات التزامن. ستناول موضوع تهيئة

القرص في القسم 2.9.3.

بالمقابل تتمثل مهمة المتحكم في تحويل تيار الخانات الثنائية المتتالية إلى قطاعات من الثُمانيات والقيام بعملية تصحيح الأخطاء إذا لزم الأمر. قطاعات الثُمانيات هذه عادة ما تُجمع خانة بخانة في مخزن لحظي داخل المتحكم، وبعد أن يتحقق النظام من صحتها والإعلان على أنها خالية من الأخطاء، تُنسخ في الذاكرة الرئيسية.

أمَّا فيما يخص التواصل مع وحدة المعالجة المركزية، يستخدم متحكم الجهاز بعض من سجلات التحكم التي عن طريق الكتابة فيها يمكن لنظام التشغيل إصدار أوامر إلى الجهاز، وذلك لغرض توصيل بيانات، أو قبول بيانات، وتشغيل أو إيقاف الجهاز نفسه، أو تنفيذ بعض من الإجراءات الأخرى، كما يمكن لنظام التشغيل– من خلال قراءة هذه السجلات– معرفة ماهية حالة الجهاز من حيث استعداده لقبول أمرًا جديدًا، أو غيره.

بالإضافة إلى سجلات التحكم، هناك عدة أجهزة لديها مخزن لحظي يُستخدم لحفظ البيانات بشكل مؤقت، كما يُمكن لنظام التشغيل القراءة من المخزن، أو الكتابة فيه. على سبيل المثال، الطريقة الشائعة لأجهزة الحواسيب لعرض بكسل على الشاشة تتم عن طريق استخدام ذاكرة فيديو (مخزن لحظي)، بحيث يُكتب فيها من قبل البرامج أو من قبل نظام التشغيل، لذلك من المهم أن يكون لدى وحدة المعالجة المركزية طريقة لتمرير المعلومات من وإلى جهاز الإدخال، والإخراج، هناك ثلاثة طرق متاحة للتواصل مع وحدة المعالجة المركزية والجهاز تتمثل في الآتي:

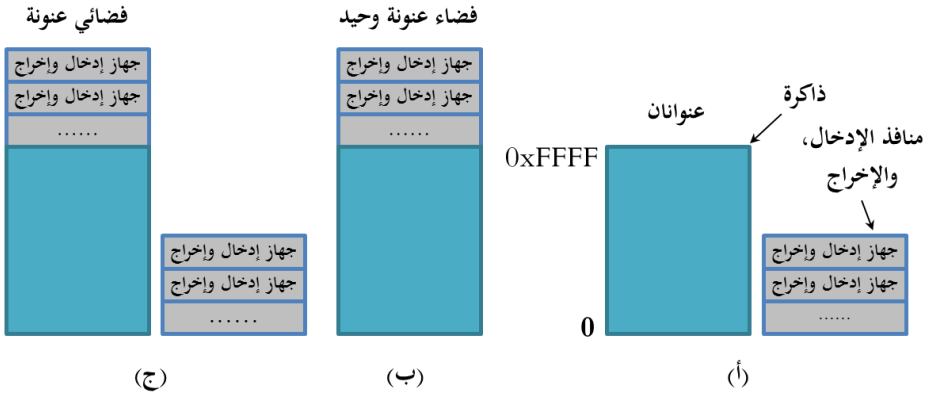
- استخدام المنافذ المستقلة للإدخال، والإخراج.
- استخدام الذاكرة المعنونة لوحدة الإدخال، والإخراج.
- استخدام الوصول المباشر للذاكرة.

1.2.4.3 المنافذ المستقلة للإدخال، والإخراج

تستخدم وحدة المعالجة المركزية في طريقة المنافذ المستقلة للإدخال، والإخراج تعليمات مصممة خصيصًا للتحكم في أجهزة الإدخال، والإخراج، تسمح المتحكمات عادةً بإرسال البيانات إلى الجهاز أو القراءة منه. بالتالي تُعيِّن هذه الطريقة رقم منفذ لكل سجل تحكم خاص

بوحدة الإدخال، والإخراج، وهو غالبًا ما يكون عدد صحيح يتكون من 8 أو 16 خانة. مجموعة المنافذ هذه تُشكل ما يعرف باسم فضاء المنافذ لوحدات الإدخال، والإخراج، وتكون محمية ولا يمكن لبرامج المستخدم العادي الوصول إليها، بينما يمكن لنظام التشغيل فعل ذلك. باستخدام تعليمة إدخال وإخراج خاصة مثل، **IN REG, PORT** يمكن لوحدة المعالجة المركزية قراءة سجل التحكم **PORT** وتخزين النتيجة في سجل وحدة المعالجة المركزية **REG**. بالمثل، باستخدام الأمر **OUT PORT, REG** يمكن لوحدة المعالجة المركزية كتابة محتوى السجل **REG** في سجل التحكم.

يختلف فضاء وحدات الإدخال، والإخراج في هذه الطريقة عن فضاء عنونة الذاكرة، وذلك كما هو موضح في الشكل 3. 5-أ، فالتعليمتان **IN R0,4** و **MOV R0,4** تختلفان في التصميم عن بعضهما بالكامل، الأولى خاصة بالفضاء الأول والثانية خاصة بفضاء عنونة الذاكرة.



الشكل 3. 5: أ) فضاء ذاكرة وفضاء إدخال، وإخراج منفصلان- ب) ذاكرة معنونة لوحدات الإدخال، والإخراج- ج) هجين.

2.2.4.3 الذاكرة المعنونة لوحدة الإدخال، والإخراج

تتمثل الطريقة الثانية التي تَبناها حاسوب PDP–11[8] في عنونة جميع سجلات التحكم في فضاء الذاكرة، وذلك كما هو موضح في الشكل 3. 5–ب، حيث خُصِّص عنوان ذاكرة فريد لكل سجل تحكم غير مسموح بتخصيصه لأي متغير آخر، وهو ما يُعرف بنظام الذاكرة المعنونة لوحدة الإدخال، والإخراج. عند استخدام هذه الطريقة تتم مشاركة فضاء العنوان نفسه بواسطة الذاكرة وأجهزة الإدخال، والإخراج، حيث يُوصَّل الجهاز مباشرةً بمواقع ذاكرة رئيسية معينة لكي يتمكن جهاز الإدخال، والإخراج من نقل كتلة البيانات من وإلى الذاكرة دون المرور عبر وحدة المعالجة المركزية. يُستخدم نظام الذاكرة المعنونة لوحدة الإدخال، والإخراج لمعظم أجهزة الإدخال، والإخراج عالية السرعة مثل الأقراص، وواجهات الاتصال.

ميزة هذه الطريقة هي أنه بالإمكان استخدام كل التعليمات التي يمكنها الوصول إلى الذاكرة لمعالجة الإدخال، والإخراج، الأمر الذي يُبسط البرمجة. بالمقابل يُقلل فضاء وحدات الإدخال، والإخراج في هذه الحالة من المساحة الإجمالية للذاكرة.

كما يمكن أيضًا استخدام مخطط هجين يجمع ما بين الطريقتين سابقة الذكر، وذلك كما هو مبين في الشكل 3. 5–ج، هذه العمارة مستخدمة في معالج البنتيوم.

السؤال الذي يطرح نفسه الآن هو كيف تعمل هذه المخططات؟ في جميع الأحوال، عندما ترغب وحدة المعالجة المركزية في قراءة كلمة، سواءً من الذاكرة أو من منفذ إدخال أو إخراج، ستُوضع العنوان المطلوب على ناقل العنونة، ومن ثم تحقن إشارة **Read** على ناقل إشارات التحكم، كما ستستخدم خط إشارة إضافي لتوضيح ما إذا كان المقصود هنا فضاء الإدخال، والإخراج أو فضاء الذاكرة. إذا كان المقصود فضاء الذاكرة، فستستجيب الذاكرة لهذا الطلب،

[8] PDP–11 هي سلسلة من الحواسيب المصغرة ذات 16 خانة، بيعت من قبل شركة المعدات الرقمية (DEC) في الفترة من عام 1970 وحتى 1990.

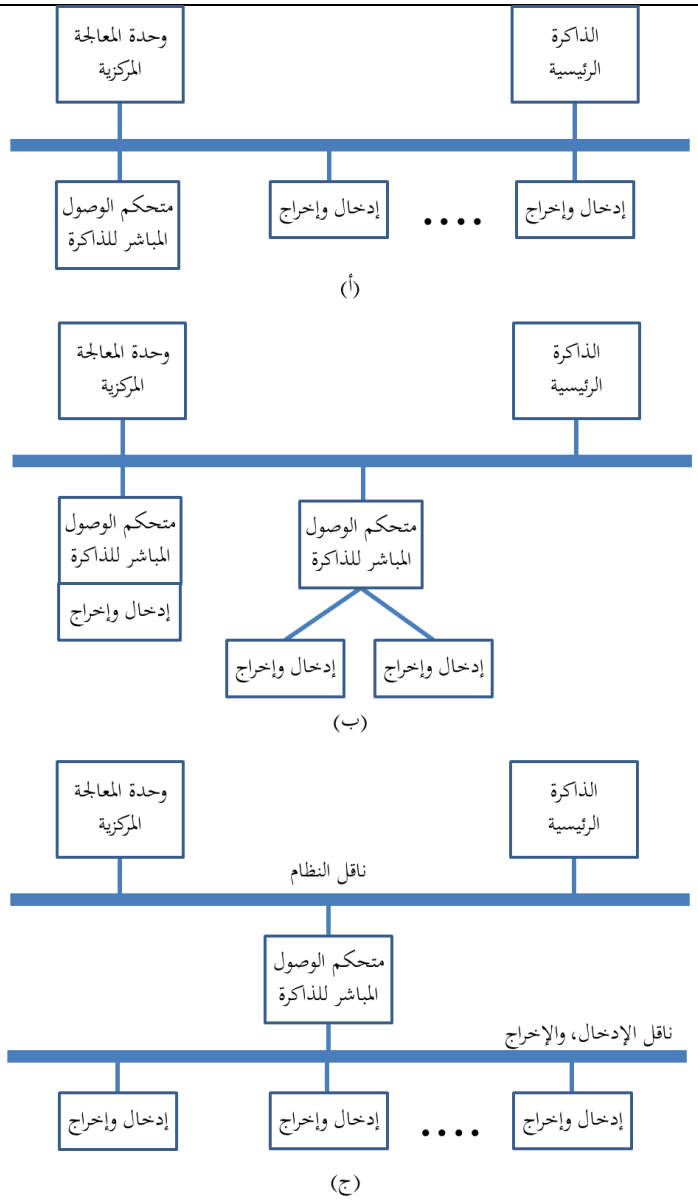
وإلا فسيستجيب جهاز الإدخال، والإخراج له. أمَّا إذا كان هناك فضاء الذاكرة فقط كما هو الحال في الشكل 3. 5–ب، فعلى كل من وحدة الذاكرة وجهاز الإدخال، والإخراج مقارنة العنوان بمدى العناوين المخصصة لخدمة كل واحد منهما، فإن وقع العنوان في المدى المحدد لأي منهما، فإن المعني سيستجيب لهذا الطلب، علمًا بأنه لن يكون هناك أي غموض في تحديد تبعية العنوان، لأنه لا يوجد عنوان مخصص لموقعين في آن واحد.

3.2.4.3 الوصول المباشر للذاكرة

تعمل الأجهزة البطيئة مثل، لوحات المفاتيح، والأجهزة السريعة مثل، الأقراص على إنشاء مقاطعة لوحدة المعالجة المركزية الرئيسية بعد نقل كل خانة ثمانية، وهو ما يجعل نظام التشغيل يقضي معظم وقته في معالجة هذه المقاطعات، الأمر الذي سيتسبب في ضياع وقت وحدة المعالجة المركزية. لذلك غالبًا ما يُستخدم نظام مختلف يُدعى الوصول المباشر للذاكرة لغرض تقليل هذا الحمل. يُعد هذا النظام أحد ميزات أنظمة الحواسيب التي تسمح لبعض من الأنظمة الفرعية للأجهزة بالوصول إلى الذاكرة الرئيسية مباشرةً، وباستقلالية عن وحدة المعالجة. هذه الميزة مفيدة أيضًا في الوقت الذي لا تستطيع فيه وحدة المعالجة مواكبة معدل نقل البيانات، أو عندما يكون نقل بيانات الإدخال، والإخراج بطيء نسبيًا.

إضافةً إلى ذلك لا يُمكن لنظام التشغيل استخدام متحكم الوصول المباشر للذاكرة، إلَّا إذا كان الكيان المادي للمتحكم مدعَّم بهذه التقنية، وهو ما تفعله معظم الأنظمة الحديثة. في بعض الأحيان يُدمج هذا المتحكم في وحدات تحكم القرص، ووحدات التحكم الأخرى، ولكن قد يتطلب مثل هذا التصميم متحكم منفصل للوصول المباشر للذاكرة لكل جهاز، إلَّا إنَّ الغالب في ذلك هو وجود متحكم واحد لتنظيم عمليات النقل إلى عدة أجهزة في وقت واحد، يُوضح الشكل 3. 6 الترتيبات المختلفة لمتحكم الوصول المباشر للذاكرة.

يُوضح الشكل 3. 6–أ الترتيب الذي يستخدم ناقل مفرد ومتحكم وصول مباشر للذاكرة مستقل، في هذه الهيئة يُستخدم الناقل مرتين عند كل عملية نقل، واحدة عند نقل البيانات من وحدة الإدخال، والإخراج إلى متحكم الوصول المباشر للذاكرة، والأخرى من المتحكم إلى الذاكرة، الأمر الذي يؤدي إلى تعليق وحدة المعالجة المركزية مرتين.



الشكل 3. 6: الترتيبات المختلفة لمتحكم الوصول المباشر للذاكرة– أ) ناقل وحيد ومتحكم وصول مباشر للذاكرة مستقل– ب) ناقل وحيد وإدخال وإخراج مدمج مع متحكم الوصول المباشر للذاكرة – ج) ناقل الإدخال، والإخراج.

في المقابل يُبين الشكل 3. 6–ب الترتيب الذي يُدمج فيه متحكم واحد للوصول المباشر للذاكرة مع نموذج واحد أو أكثر من وحدات الإدخال، والإخراج بدون تضمين ناقل النظام كجزء يعمل مع وحدة الإدخال، والإخراج أو كوحدة منفصلة تتحكم فيها. تستخدم هذه الهيئة الناقل مرة واحدة فقط عند كل عملية نقل والمتمثلة في نقل البيانات من متحكم الوصول المباشر للذاكرة إلى الذاكرة، وهو ما يؤدي إلى تعليق وحدة المعالجة المركزية مرة واحدة.

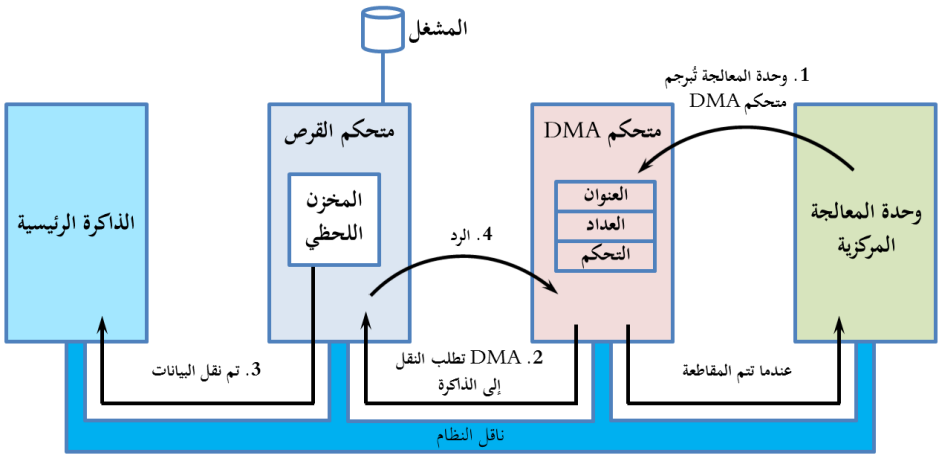
أمَّا الشكل 3. 6–ج فهو يُوضح البنية التي تدمج متحكم الوصول المباشر للذاكرة مع وحدات الإدخال، والإخراج من خلال ناقل الإدخال، والإخراج، والتي تُقلل من عدد واجهات الإدخال، والإخراج المطلوبة بين المتحكم ووحدة الإدخال، والإخراج. تستخدم هذه الطريقة أيضًا الناقل مرة واحدة فقط، وتُعلق فيها وحدة المعالجة المركزية مرة واحدة عند كل عملية نقل من المتحكم إلى الذاكرة.

بغض النظر عن المكان الفيزيائي لمتحكم الوصول المباشر للذاكرة، فإن لدى هذا المتحكم حق الوصول إلى ناقل النظام بشكل مستقل عن وحدة المعالجة المركزية، وذلك كما هو مبين في الشكل 3. 7. يحتوي هذا المتحكم على العديد من السجلات التي يُمكن الكتابة فيها وقراءتها من قبل وحدة المعالجة المركزية، وتشمل سجل عنوان الذاكرة، وسجل عدَّاد الخانات الثمانية، وسجل التحكم (واحد أو أكثر)، حيث تُستخدم السجلات الأخيرة في تحديد كل من منفذ الإدخال، والإخراج الذي سيُستخدم، واتجاه النقل (قراءة من جهاز الإدخال، والإخراج أو الكتابة فيه)، ووحدة النقل (بمعدل خانة ثمانية أو كلمة في الوحدة الزمنية الواحدة)، وعدد الخانات الثمانية المراد نقلها.

قبل توضيح فكرة عمل متحكم الوصول المباشر للذاكرة، سنوضح أولًا الكيفية التي تتم بها عملية قراءة القرص عندما لا يُستخدم هذا المتحكم والتي تكون على النحو التالي:

1. يقرأ متحكم الجهاز مقطع، أو مجموعة من المقاطع من القرص بشكل متسلسل (خانة بخانة) إلى أن يُحمَّل المقطع بأكمله في المخزن الداخلي لمتحكم الجهاز.
2. يتحقق المتحكم من عدم وجود أخطاء ناتجة عن عملية القراءة.
3. يُحدِث المتحكم مقاطعة تُسبب في بدء تشغيل نظام التشغيل ليدخل بدوره في حلقة تكرارية لقراءة محتوى مخزن المتحكم اللحظي، بحيث يقرأ في كل دورة خانة ثمانية واحدة، أو كلمة

من سجل المتحكم وتخزينها في الذاكرة الرئيسية.



الشكل 3. 7: استخدام متحكم الوصول المباشر للذاكرة في عملية النقل، ترمز DMA هنا الوصول المباشر للذاكرة [Tanenbaum & Bos, 2015].

أمَّا في حالة دعم متحكم الجهاز بمتحكم الوصول المباشر للذاكرة فإن الأمر سيكون مختلفًا، وستكون الخطوات على النحو التالي:

4. تُبرمج وحدة المعالجة متحكم الوصول المباشر للذاكرة من خلال ضبط سجلاته بحيث يُعْلَم المتحكم بعنوان المكان الذي سيُخزَّن المقطع فيه داخل الذاكرة، وبعدد الخانات الثمانية المراد نقلها، بالإضافة إلى عنوان المقطع داخل القرص (الخطوة 1 في الشكل 3. 7).
5. تُصدر وحدة المعالجة المركزية أيضًا أمرًا إلى متحكم القرص لقراءة البيانات من القرص إلى المخزن اللحظي الداخلي، وللتحقق من عدم وجود أخطاء في عملية القراءة. عندما يتأكد المتحكم من صحة البيانات في المخزن اللحظي لمتحكم القرص، يمكن لمتحكم الوصول المباشر للذاكرة أن يبدأ في العمل.
6. يبدأ متحكم الوصول المباشر للذاكرة في عملية النقل بإصدار طلب قراءة لمتحكم القرص (الخطوة 2 في الشكل 3. 7) مستخدمًا ناقل النظام. طلب القراءة هذا يشبه أي طلب قراءة آخر، ومتحكم القرص لا يعرف أو يهتم بما إذا كان هذا الطلب قد صدر من وحدة المعالجة المركزية أو من متحكم الوصول المباشر للذاكرة. عادةً، ما يكون عنوان الموقع

المراد الكتابة فيه داخل الذاكرة موجود على ناقل العنونة، بالتالي عندما يَبحث متحكم القرص عن الكلمة التالية في مخزنه الداخلي فإنه سيكون على دراية بموقع الكتابة. تحتاج عملية الكتابة هذه إلى دورة قياسية أخرى لناقل النظام (الخطوة 3 في الشكل 3. 7).

7. عندما تنتهي عملية الكتابة، يُرسل متحكم القرص إشارة الإقرار إلى متحكم الوصول المباشر للذاكرة مستخدمًا ناقل النظام (الخطوة 4 في الشكل 3. 7).
8. بعدها يَزيد المتحكم قيمة عنوان الذاكرة ويُنقص قيمة عدَّاد الخانات الثمانية، وذلك تبعًا لعدد الخانات الثمانية المنقولة، إذا كانت قيمة عدَّاد هذه الخانات لا تزال أكبر من صفر، فستُكرر الخطوتين السابقتين إلى أن تصل قيمته صفر.
9. يُقاطع متحكم الوصول المباشر للذاكرة وحدة المعالجة المركزية، لكي يُعلمها بإنتهاء عملية النقل. عند اشتغال نظام التشغيل من جديد سيجد أن المقطع المطلوب موجود بالفعل في الذاكرة.

ذكرنا سابقًا أن القرص يقرأ أولًا البيانات إلى المخزن الداخلي قبل أن يبدأ متحكم الوصول المباشر للذاكرة. هنا يمكن أن يتساءل المرء، لماذا لا يُخزن المتحكم الخانات الثمانية في الذاكرة الرئيسية بمجرد الحصول عليها من القرص؟ بعبارة أخرى، لماذا نحتاج إلى المخزن اللحظي الداخلي؟ السبب وراء ذلك يتمثل في الآتي:

- تُمكن فكرة استخدام التخزين اللحظي الداخلي متحكم القرص من التحقق من صحة البيانات قبل بدء عملية النقل، إذا كانت البيانات متضررة، فتُصدر إشارة خطأ وستفشل عملية النقل هذه.
- بمجرد أن تبدأ عملية النقل من القرص، ستبدأ البيانات في الوصول من القرص بمعدل ثابت سواءً أكان المتحكم جاهزًا لاستقبالها أم لا، وبناءً عليه إذا حاول المتحكم إرسال البيانات مباشرةً إلى الذاكرة، فسيتعين عليه استخدام ناقل النظام في كل مرة ينقل فيها كلمة أو خانة ثمانية، إذا كان ناقل النظام مزدحمًا بسبب استخدامه من قبل بعض من الأجهزة الأخرى، فيتحتم على المتحكم الانتظار إلى أن يتحرر ناقل النظام، الأمر الذي سيؤدي إلى وجوب تخزين البيانات في مكان– ما– (المخزن اللحظي).

هناك ميزة مهمة لها انعكاسات إيجابية على مشغل القرص، وهي إمكانية قيام المتحكم بالبحث في اثنين أو أكثر من مشغلات الأقراص وفي نفس الوقت. هذا يعني، بينما ينتظر كل من المتحكم والبرمجيات في إكمال عملية البحث في أحد المشغلات، فإنه بإمكان المتحكم البدء في عملية بحث في مشغل أقراص آخر، كما يمكن لعدة متحكمات القراءة من أو الكتابة في مشغل قرص واحد فقط في الوقت الذي تتم فيه عملية بحث عن مشغل آخر أو عدة مشغلات أقراص أخرى، باستثناء متحكم القرص المرن الذي لا يستطيع القراءة أو الكتابة على مشغلين في نفس الوقت. السبب في ذلك يرجع إلى أن القراءة أو الكتابة تتطلب من المتحكم نقل خانات ثنائية ضمن حدود الميكروثانية، لذلك فإن نقلة واحدة قد تستهلك أكثر من القدرة الحاسوبية لها. الأمر يختلف بالنسبة للأقراص الصلبة مع وجود متحكم متكامل، وكذلك مع إمتلاك النظام لأكثر من مشغل للأقراص الصلبة وقابلة للعمل في آن واحد على الأقل في أثناء فترة النقل بين القرص والمخزن اللحظي للمتحكم. من الجدير بالذكر أن القدرة على أداء عمليتين أو أكثر في نفس الوقت يمكن أن تُقلل من متوسط زمن الوصول بشكل كبير.

5.3 مراجعة المقاطعات

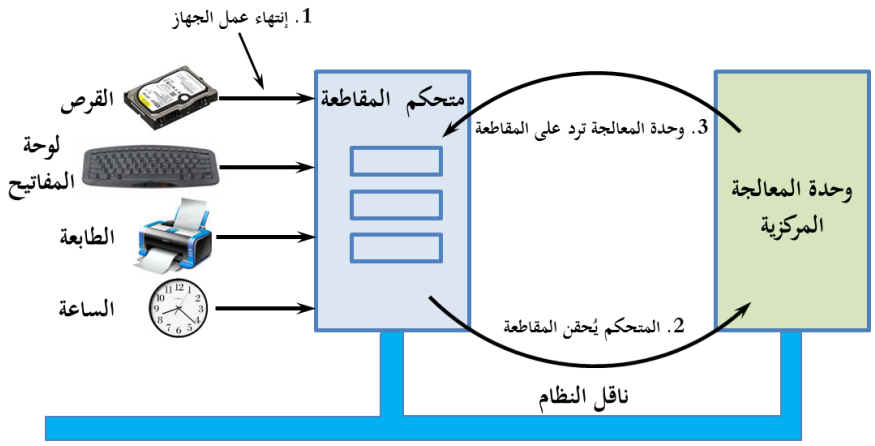
كنا قد قدمنا نبذة وجيزة عن المقاطعات في القسم السابق، بالتالي سنحاول في هذا القسم مراجعتها بنوع من التفصيل، وذلك لأهمية هذا الموضوع بالنسبة لوحدات الإدخال، والإخراج. فالمقصود بالمقاطعة هو التغيير في التنفيذ الذي يحدث بسبب أنواع مختلفة من الأحداث. تُقسم المقاطعات في الغالب إلى مقاطعات متزامنة وغير متزامنة، تُعرف الأولى أيضًا بالاستثناءات وهي تنتج بواسطة وحدة التحكم داخل وحدة المعالجة المركزية في أثناء تنفيذ التعليمات دون أي تخطيط مسبق، وتُسمى متزامنة لأنها تصدر من وحدة التحكم فقط بعد إنهاء تنفيذ التعليمات بسبب أخطاء في شفرة العملية، أو في التعليمات نفسها كالقسمة على صفر. أمَّا المقاطعات غير المتزامنة فهي تحدث في أزمنة عشوائية عندما يتعلق الأمر بالأحداث الخارجية (المنافذ التسلسلية، لوحة المفاتيح) كطلبات الحصول على خدمات نظام التشغيل، أو في أثناء تعامل وحدة المعالجة المركزية مع وحدات الإدخال، والإخراج. في كلتا الحالتين، تحتاج المقاطعات إلى معالجة فورية وقد تتسبب في وقف مؤقت لتنفيذ برنامج معين من أجل تنفيذ برنامج آخر له أولوية أعلى، أو أي سبب آخر، كما تتضمن المقاطعة تخزين مؤقت لبيانات البرنامج الموقوف

لاسترجاعها عند إنتهاء المقاطعة.

يُخصص لكل مقاطعة إجراء يُعرف بإجراء خدمة المقاطعة وهو المسؤول عن معالجتها. عند حدوث المقاطعة تُوقف وحدة المعالجة المركزية عملها وتنتقل فورًا إلى عنوان ثابت يحوي في معظم الحالات عناوين بداية إجراءات خدمة المقاطعة والذي يبدأ مباشرة في التنفيذ. في النهاية تُجدِّد وحدة المعالجة المركزية البرنامج الذي تَوَقَّف وتُحمَّل بياناته. يُوضح الشكل 3. 8 مثال لهيكلية مقاطعات الإدخال، والإخراج في نظام الحاسوب الشخصي. على مستوى الكيان المادي، تعمل المقاطعة على النحو التالي:

1. عندما ينتهي جهاز الإدخال أو الإخراج من العمل المناط به، سيُحدث مقاطعة (بفرض تمكين المقاطعات من قبل نظام التشغيل) وذلك من خلال حقن إشارة على خط ناقل النظام الذي عُيِّن بالخصوص.
2. يُكشف عن هذه الإشارة من قبل رقاقة متحكم المقاطعة الموجودة على اللوحة الأم والتي ستُقرر ماهية العمل الذي يجب القيام به بعد ذلك.
3. إذا لم تكن هناك مقاطعات أخرى معلقة، يُعالج متحكم المقاطعة مباشرةً العملية، أمَّا إذا كانت وحدة المعالجة المركزية مشغولة بمعالجة مقاطعة أخرى، أو طلب جهاز آخر إذن بالمقاطعة في آن واحد– وكان هذا الطلب ذو أولوية أعلى– فسيُتجاهل الجهاز الأول في الوقت الراهن على أن يستمر هذا الجهاز في التأكيد على إشارة المقاطعة حتى يُخدم من قبل وحدة المعالجة المركزية.
4. للتعامل مع المقاطعة يضع المتحكم رقم على ناقل العنونة والذي يُحدَّد من خلاله الجهاز الذي يرغب في مقاطعة وحدة المعالجة المركزية، إشارة المقاطعة هذه ستجعل وحدة المعالجة المركزية تتوقف عن مواصلة عملها الحالي وستبدأ في القيام بعمل آخر.
5. يُستخدم الرقم الموجود على ناقل العنونة كمؤشر إلى جدول يُسمى بمتجه المقاطعة، وذلك لجلب عدَّاد برنامج جديد يُشير إلى إجراء خدمة المقاطعة المناظرة لإشارتها.
6. بعد وقت قصير يبدأ تشغيل إجراء خدمة المقاطعة الذي بدوره يرد على المقاطعة عن طريق كتابة قيمة محددة بأحد متحكمات المقاطعة الخاصة بمنافذ الإدخال، والإخراج. هذا الرد يُخبر المتحكم بأنه حر في إصدار مقاطعة أخرى، كما يمكن تجنب مشكلة وضع التسابق

في حالة وجود عدد من المقاطعات (في آن واحد) عن طريق تأخير وحدة المعالجة المركزية لهذا الرد إلى أن تكون مستعدة للتعامل مع المقاطعة القادمة.



الشكل 3. 8: آلية حدوث مقاطعة الإدخال، والإخراج.

7. قبل البدء في تنفيذ إجراء الخدمة، يَحْفظ الكيان المادي معلومات معينة. ماهية هذه المعلومات، وتحديد مكان حفظها يختلف بشكل كبير من وحدة معالجة مركزية إلى أخرى. كحد أدنى، يجب حفظ عدَّاد البرنامج، بحيث يمكن استخدامه لإعادة تشغيل العملية الموقوفة، وكحد أقصى قد تُحفظ جميع السجلات العامة وعدد كبير من السجلات الداخلية.

القضية الأخرى المهمة في هذا السياق هي أين تُحفظ هذه المعلومات؟ يتمثل أحد الخيارات المتاحة في وضع هذه المعلومات في سجلات داخلية بحيث يمكن لنظم التشغيل قراءتها كلما كانت هناك حاجة إلى ذلك. المشكلة مع هذا النهج تكمن في أنه لا يمكن الرد على متحكم المقاطعة إلَّا بعد أن تُقراء جميع المعلومات ذات الصلة، خشية أن تحدث مقاطعة ثانية تؤدي إلى الكتابة فوق السجلات الداخلية قبل إنقاذ الوضع. تؤدي هذه الاستراتيجية إلى ضياع أزمنة طويلة، وخصوصًا عندما تُفصل المقاطعات التي قد تؤدي إلى فقد المقاطعات والبيانات معًا.

نتيجةً لما سبق ذكره فإن معظم وحدات المعالجة المركزية تحتفظ بهذه المعلومات في المكدس بالرغم من وجود مشاكل في هذا الطريقة. تتمثل المشكلة الأولى في مالك المكدس،

فقد يكون المكدس الحالي الذي قيد الاستخدام هو مكدس مُخصص لعملية مستخدم– ما–، بالتالي قد لا يكون مؤشر المكدس مؤشرًا قانونيًا. هذا المؤشر من شأنه أن يتسبب في خطأ فادح عندما يحاول الكيان المادي كتابة بعض من الكلمات في العنوان المشار إليه به، أيضًا قد يُشير المؤشر إلى نهاية الصفحة في الذاكرة، وخصوصًا بعد عدة محاولات للكتابة في الذاكرة، الأمر الذي قد يؤدي إلى تجاوز حدود الصفحة، وهو ما سيولد خطأ الصفحة[9]، علمًا بأن حدوث خطأ الصفحة في أثناء معالجة المقاطعة للكيان المادي ينشأ عنه مشكلة أكبر تتمثل في عدم وجود إمكانية لحفظ الحالة التي تتعامل مع خطأ الصفحة. الأمر قد يزداد تعقيدًا إذا ما نظرنا إلى حالة وحدات المعالجة المركزية الحديثة، وخصوصًا التي تعتمد على مفهوم الأنابيب أو السلّمي الفائق[10]، وذلك كما سيتم مناقشته تاليًا.

المقاطعات الدقيقة وغير الدقيقة: بعد الإنتهاء من تنفيذ كل تعليمة يستكشف برنامج مُصغر، أو الكيان المادي في النظم القديمة ما إذا كانت هناك مقاطعة مُعلقة. إذا كان الأمر كذلك، يدفع النظام كل من عدَّاد البرنامج، والسجلات الأخرى المستخدمة كسجل الحالة إلى المكدس، ويبدأ في تنفيذ خطوات المقاطعة. بعد الإنتهاء من ذلك، تبدأ العملية العكسية في التنفيذ من خلال إخراج محتويات المكدس، والرجوع إلى تنفيذ العملية السابقة.

تفترض الأجهزة القديمة ضمنيًا أنه في حالة حدوث مقاطعة مباشرةً بعد أي تعليمة، ستُنفذ جميع التعليمات بالكامل– بما في ذلك التعليمة نفسها– ولن تُنفذ أي تعليمة بعدها على الإطلاق، هذه الفرضية صحيحة دائمًا في هذا النوع من الأجهزة، ولكن قد يختلف الأمر في حالة وحدات المعالجة المركزية الحديثة مثل تلك التي تعتمد على مفهوم الأنابيب.

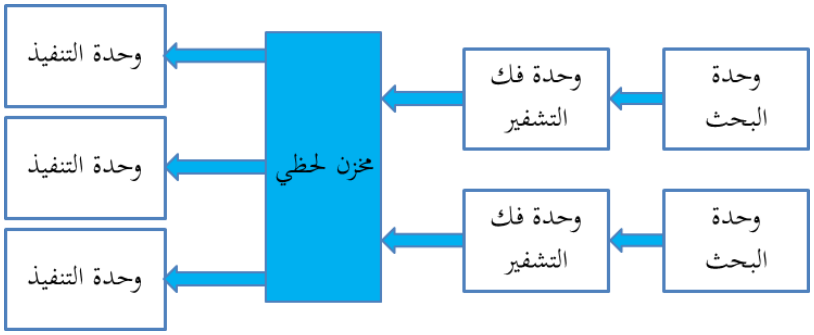
---

[9] سيتم التحدث عن موضوع خطأ الصفحة في الفصل الخامس بنوع من التفصيل.

[10] في السلّمي الفائق يُدمج أكثر من خط أنابيب في معالج واحد بحيث يصبح لدينا مثل ما يشبه خطوط إنتاج تعمل في نفس الوقت، أي بالتالي مضاعفة سرعة المعالج عدة مرات.

الآن ماذا سيحدث إذا أُصدرت مقاطعة– ما– في أثناء انشغال مراحل الأنابيب بالكامل (وهي الحالة المعتادة)؟ في مثل هذه الحالات ستكون هناك عدة تعليمات في مراحل مختلفة من عملية التنفيذ، بالتالي عند حدوث المقاطعة، قد لا تعكس قيمة عدَّاد البرنامج الحدود الصحيحة بين التعليمات المنفذة والتعليمات غير المنفذة. عمليًا، من المحتمل أن تكون هناك عدة تعليمات غير مُكتملة التنفيذ، مع خليط من التعليمات الأخرى، والتي قد تكون أكثر أو أقل اكتمالًا، لذلك على الأرجح أن يعكس عدَّاد البرنامج عنوان التعليمة التالية في البحث والتي يجب جلبها والدفع بها في خط الأنابيب بدلًا من عنوان التعليمة التي عُولجت مؤخرًا من قبل وحدة التنفيذ.

تزداد الأمور سوءًا في جهاز السلّمي الفائق والمبين هيكليته في الشكل 3. 9، فقد تتفكك التعليمات إلى عمليات مُصغرة تُنفذ من دون مراعاة أي ترتيب، وذلك اعتمادًا على توفر الموارد الداخلية مثل، الوحدات الوظيفية، والسجلات. بالتالي في أثناء حدوث المقاطعة قد تكون هناك بعض من التعليمات بدأت منذ فترة طويلة وتوقفت لسبب من الأسباب، ولربما لم تبدأ من جديد بعد، وقد تكون هناك تعليمات أخرى بدأت في الآونة الأخيرة وهي على وشك الإنتهاء تقريبًا. عليه عند نقطة انطلاق إشارة المقاطعة، سوف تكون هناك عدة تعليمات في حالات مختلفة من الإكتمال، وغير مرتبطة مع عدَّاد البرنامج.



الشكل 3. 9: مفهوم السلّمي الفائق لوحدة المعالجة المركزية الحديثة [Stallings, 2012].

بناءً على ما تقدم فإن المقاطعة التي تترك الجهاز في حالة محددة ومعرَّفة جيدًا، يُطلق عليها اسم المقاطعة الدقيقة، والتي تتمتع بالخصائص الأربع التالية:

1. يكون عدَّاد البرنامج محفوظ في مكان معروف.
2. تكون جميع التعليمات الصادرة قبل تعليمة المقاطعة قد نُفذت بالكامل.
3. جميع التعليمات التالية لتعليمة المقاطعة مازالت لم تُنفذ بعد.
4. تكون الحالة التنفيذية للتعليمة المشار إليها بعدَّاد البرنامج معروفة.

المشكلة في كل هذا هو أنه مثل هذه الخصائص قد تتعارض مع متطلبات خطوط الأنابيب أو غيرها من متطلبات الأداء السريع للمعالج، لذلك يتعين الرضا بمقاطعات أقل دقة من أجل الحصول على أقل نقص في الأداء. يوضح الوضع المبين في الشكل 3. 10–أ معنى المقاطعة الدقيقة، مع ملاحظة أنَّ جميع التعليمات قبل عدَّاد البرنامج (316) قد نُفذت بالكامل، بينما التعليمات الواقعة بعده غير منفذة بعد.

بالمقابل يُطلق على المقاطعة التي لا تفي بهذه المتطلبات بالمقاطعة غير الدقيقة، وهي ما يجعل الحياة أكثر صعوبةً وخصوصًا لمبرمج نظم التشغيل، الذي يجب عليه الآن معرفة ماذا حدث؟ وما الذي سيحدث؟ يُوضح الشكل 3. 10–ب هذا النوع من المقاطعات، والذي يظهر فيه مجموعة من التعليمات المختلفة، والتي تختلف في مراحل الإنجاز، مع ملاحظة أنه ليس من الضروري أن تكون التعليمات الأقدم هي الأكثر إنجازًا من التعليمات الحديثة.

الشكل 3. 10: أ) المقاطعات الدقيقة– ب) المقاطعات غير الدقيقة.

تدفع الأجهزة المدَعَّمة للمقاطعات غير الدقيقة في العادة بكمية كبيرة من الحالات الداخلية

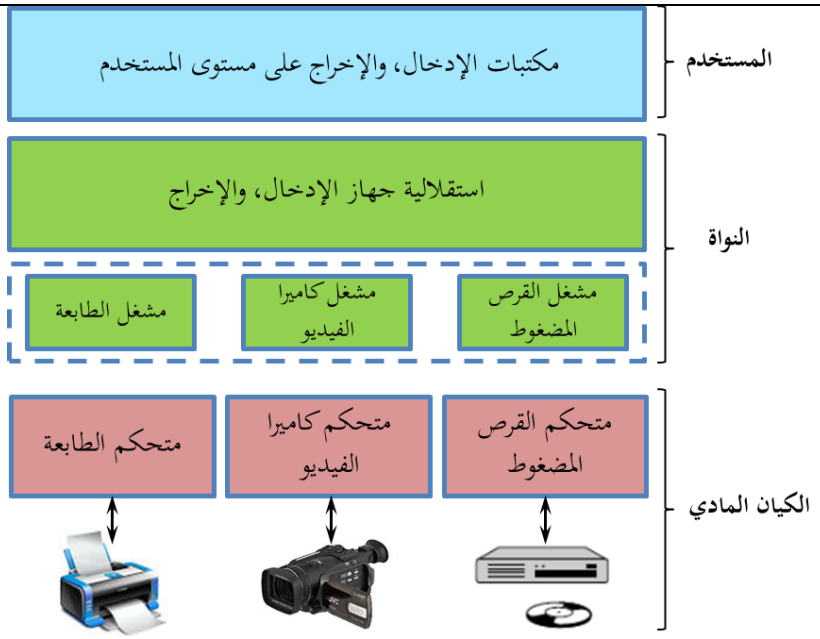
في المكدس، وذلك لإعطاء نظام التشغيل الفرصة لمعرفة المجرايات داخل النظام، يترتب على هذا الأمر حفظ كمية كبيرة من المعلومات في الذاكرة لكل عملية مقاطعة، وهو ما سيجعل المقاطعة بطيئة، وعملية الرجوع أسوأ. يؤدي مثل هذا الوضع إلى السخرية من وجود وحدات معالجة مركزية سريعة جدًا، ومدعمَّة بمفهوم السلّمي الفائق، لأنها وبسبب بطء المقاطعات في بعض الأحيان تكون غير صالحة للعمل في الوقت الحقيقي.

خلاصة القول نجد أن هناك بعض من أجهزة الحواسيب صُممت على أساس أن بعض من أنواع المقاطعات، والقفز تكون دقيقة والبعض الآخر لا. مثلًا، يمكن تصميم مقاطعات الإدخال، والإخراج بمفهوم المقاطعات الدقيقة، بينما عمليات القفز بمفهوم المقاطعات غير الدقيقة، وذلك بسبب أن أخطاء البرمجة القاتلة مثل القسمة على صفر لا تتطلب إعادة تشغيل العملية من جديد. الجانب السلبي لتصميم الحواسيب على أساس مفهوم المقاطعات الدقيقة هو أنه يتحتم على وحدة المعالجة المركزية تسجيل كل ما تقوم به بعناية والحفاظ على النسخ الاحتياطية من السجلات حتى يمكنها توليد مقاطعات دقيقة في أي لحظة، كل هذا الحمل له تأثير سلبي كبير على أداء النظام.

6.3 المفاهيم المعنوية لوحدات الإدخال، والإخراج

لنبتعد الآن عن الكيان المادي لأجهزة الإدخال، والإخراج، ولنتعرف على المفاهيم البرمجية للإدخال، والإخراج. تُنظم هذه البرمجيات في الغالب في ثلاث طبقات على النحو الموضح في الشكل 3. 11، والمفصلة كالتالي:

- طبقة المكتبات على مستوى المستخدم: تُوفر هذه المكتبات واجهات بسيطة لبرامج المستخدم للقيام بعمليات الإدخال، والإخراج، مثل مكتبة **stdio** والمستخدمة في لغتي البرمجة **C** و **C++**.
- طبقة نماذج النواة: وهي تُوفر مشغلات الأجهزة للتعامل مع متحكمات الأجهزة ونماذج الإدخال، والإخراج ذات السمة الاستقلالية والمستخدمة من قبل المشغلات.
- طبقة الكيان المادي: تحوي هذه الطبقة الكيان المادي الفعلي والمتحكمات التي تتفاعل مع مشغلات الأجهزة لتنشيط الكيان المادي.



الشكل 3. 11: طبقات برمجيات الإدخال، والإخراج.

1.6.3 أهداف برمجيات الإدخال، والإخراج

يُلقي هذا القسم نظرة على أهداف ومفاهيم برمجيات الإدخال، والإخراج الرئيسية التي يجب مراعاتها في تصميم هذه البرمجيات، وهي على النحو التالي:

- استقلالية الجهاز: يُعتبر هذا المفهوم من أهم الخصائص الرئيسية لبرمجيات الإدخال، والإخراج، وهو يعني أنه ينبغي أن يكون من الممكن كتابة البرامج المستخدمة للوصول إلى أي جهاز إدخال، أو إخراج دون الحاجة إلى تحديد أو معرفة تفاصيل الجهاز مقدمًا. فمثلًا، يجب أن يكون البرنامج المعد لقراءة ملف كمدخل قادرًا على قراءة ملف من القرص الصلب، أو القرص المضغوط، أو حتى من شريحة الناقل التسلسلي العام من دون أن تكون هناك حاجة إلى تعديل البرنامج من قبل المستخدم ليتلاءم مع كل نوع من هذه الأجهزة المختلفة. كذلك ينبغي للمرء أن يكون قادرًا على كتابة أمر مثل **sort <input >output**، بحيث يعمل مع كل من المدخلات القادمة من أي نوع من أجهزة الإدخال مثل القرص، أو لوحة المفاتيح، أو المخرجات المتجهة إلى أي نوع من

أجهزة الإخراج مثل القرص، أو الشاشة. الأمر متروك هنا لنظام التشغيل لرعاية المشاكل الناجمة عن حقيقة أن هذه الأجهزة مختلفة حقًا وتتطلب سلسلة من الأوامر المختلفة للقراءة أو الكتابة.

- التسمية الموحدة: المفهوم الثاني لأهداف برمجيات الإدخال، والإخراج والمرتبط ارتباطًا وثيقًا باستقلالية الجهاز هو التسمية الموحدة، فاسم أي ملف أو جهاز يجب أن يكون مجرد سلسلة حرفية، أو عددًا صحيحًا ولا يعتمد على نوع الجهاز بأي شكل من الأشكال. في نظام 'يونيكس' يمكن لجميع الأقراص أن تكون مُجمعة في التسلسل الهرمي لنظام الملفات بطرق مختلفة، وهو ما لا يُلزم المستخدم بأن يكون على بينة بالاسم الذي يتوافق مع كل جهاز. على سبيل المثال، يمكن تركيب شريحة الناقل التسلسلي العام على رأس الدليل **usr/ast/backup/** بحيث إذا نُسخ ملف إلى المسار **usr/ast/backup/monday/** فهذا يعني نسخه إلى هذه الشريحة. في هذه الطريقة، تُعنون جميع الملفات والأجهزة بواسطة اسم المسار.
- معالجة الأخطاء: بشكل عام يُعتبر هذا المفهوم من القضايا المهمة بالنسبة لبرمجيات الإدخال، والإخراج، لذلك ينبغي التعامل مع الأخطاء في طبقة الكيان المادي كلما أمكن ذلك، بحيث إذا اكتشف المتحكم خطأ– ما– في أثناء القراءة، يجب عليه أن يُحاول تصحيح الخطأ بنفسه، إذا كان بوسعه فعل ذلك. أمَّا إذا لم يتمكن المتحكم من تصحيحه، فيجب على مشغل الجهاز التعامل معه حتى ولو كلف ذلك قراءة المقطع مرة أخرى، فالعديد من الأخطاء قد تكون عابرة مثل، الأخطاء الناجمة عن بقع من الغبار على رأس القراءة التي سوف تزول– في كثير من الأحيان– إذا ما قُرأ المقطع من جديد. في حالة لم يتمكن الكيان المادي من التعامل مع الأخطاء فيجب عليه إخطار الطبقة التي تليه بهذا الخطاء، لكي يُعالج فيها إن أمكن ذلك. في كثير من الحالات، تُعالج الأخطاء في طبقة الكيان المادي. سنعود إلى مناقشة التعامل مع الأخطاء في القسم **6.9.3**.
- الأجهزة الحرفية والأجهزة المقطعية: من المفاهيم الأخرى لأهداف برمجيات الإدخال، والإخراج هو التفرقة بين الأجهزة الحرفية والأجهزة المقطعية. فالأولى تنقل الخانات الثمانية واحدة تلو الأخرى، بينما تنقل الثانية مجموعة من الخانات الثمانية كوحدة واحدة.

- الوصول التسلسلي أو العشوائي للأجهزة: الفرق يجب أن يكون جليًا بين الوصول التسلسلي أو العشوائي للأجهزة، فالجهاز التسلسلي ينقل البيانات بترتيب ثابت مُحدد من قبل الجهاز، في حين يُمكن لمستخدم جهاز الوصول العشوائي توجيه الجهاز للبحث عن أيٍ من مواقع تخزين البيانات المتاحة بشكل عشوائي.
- النقل المتزامن والنقل غير المتزامن: من القضايا الرئيسية الأخرى لبرمجيات الإدخال، والإخراج قضية النقل المتزامن أو المحجوب، والنقل غير المتزامن أو غير المحجوب. معظم عمليات الدخل والخرج تندرج تحت النوع الثاني من النقل الذي يُسمح فيه لعمليات أخرى بمواصلة عملها قبل إنتهاء عملية الإرسال، الأمر الذي يمنح الفرصة لوحدة المعالجة المركزية أن تبدأ النقل ومن ثم تتفرغ للقيام بمهام أخرى إلى أن تصل المقاطعة. مثال ذلك ما يحدث في متحكم الوصول المباشر للذاكرة. السبب وراء ذلك هو أن عمليات الإدخال، والإخراج يمكن أن تكون بطيئة للغاية مقارنة بمعالجة البيانات، فجهاز الإدخال، والإخراج يتضمن أجزاء ميكانيكية يجب تحريكها، مثل رأس القراءة والكتابة في القرص الصلب، الأمر الذي يتطلب زمنًا طويلًا مقارنة بزمن معالجة البيانات، لذلك وجب استغلال هذا الزمن.

  بالمقابل تندرج برامج المستخدم تحت النوع الأول من النقل الذي تُعلق فيه أو تُحجب عمليات الدخل والخرج تلقائيًا بعد بدئها إلى حين توفر البيانات في المخزن اللحظي. هذا يعني، عندما تتعدد عمليات الدخل والخرج فإن المعالج سيبقى فترة طويلة خاملًا إلى أن تنتهي هذه العمليات، وهو ما سيتسبب في ضياع وقت المعالج.
- التخزين اللحظي: في كثير من الأحيان البيانات التي تُرسل من جهاز الإدخال، والإخراج لا يمكن تخزينها مباشرة في وجهتها النهائية إلا بعد ما تُحفظ بشكل مؤقت في مخزن المتحكم اللحظي، وذلك لعدة أسباب نذكر منها: ضرورة التحقق من صحة البيانات قبل بدء عملية الإرسال، كذلك عند استقبال حزمة من الشبكة من قبل الحاسوب نجد أن نظام التشغيل لا يعرف المكان الذي ستُوضع فيه هذه الحزمة، إلَّا بعد تخزينها في المخزن اللحظي، ومن ثم فحصها واختبارها. بالإضافة إلى ذلك بعض من الأجهزة لديها قيود شديدة خاصة بالزمن الحقيقي (مثل، الأجهزة السمعية الرقمية)، لذا يجب وضع البيانات في المخزن اللحظي، لغرض ضمان التوافق بين المعدل الذي يُكتب به في المخزن اللحظي مع المعدل الذي يُفرغ به من أجل تجنب تجاوز الحد الأدنى. بالرغم من أهمية التخزين اللحظي إلَّا إنَّه غالبًا ما

ينتج عنه عدة عمليات نسخ، الأمر الذي يترتب عليه تأثر أداء الإدخال، والإخراج، سيتم الرجوع إلى مناقشة التخزين اللحظي بنوع من التفصيل في القسم 2.10.3.

- سرعة العملية: برمجيات الإدخال، والإخراج يجب أن تكون لها القدرة على التعامل مع السرعات المختلفة لأجهزة الإدخال، والإخراج والتي تتراوح سرعاتها من بضع خانات ثمانية في الثانية إلى بضع قيقا خانة ثمانية في الثانية.
- القراءة والكتابة، القراءة فقط، أو الكتابة فقط: من مفاهيم برمجيات الإدخال، والإخراج هو مفهوم التفرقة بين أجهزة القراءة والكتابة، القراءة فقط، أو الكتابة فقط. فبعض من هذه الأجهزة تقوم بإجراء كل من الإدخال والإخراج، بينما تدعم أجهزة أخرى اتجاه واحد فقط لنقل البيانات، إما الكتابة أو القراءة.
- الأجهزة العامة والأجهزة المخصصة: المفهوم الأخير لأهداف برمجيات الإدخال، والإخراج هو ما يعرف بقابلية الأجهزة للمشاركة (عامة) مقابل الأجهزة المخصصة، فبعض من أجهزة الإدخال، والإخراج، مثل الأقراص، مسموح بمشاركتها واستخدامها من قبل عدة مستخدمين في نفس الوقت. بالمقابل، بعض من الأجهزة الأخرى، مثل مشغلات أقراص الشريط الممغنط، يجب أن تكون مخصصة لمستخدم واحد طيلة استخدامها وحتى يتم الإنتهاء منها، بعدها يمكن لمستخدم آخر أن يستغلها. بالتالي فإن استعمال اثنين أو أكثر من المستخدمين لنفس الشريط الممغنط في نفس الوقت لغرض كتابة مقطع سوف لن يكون مجديًا. إن استخدام الأجهزة المخصصة يقود إلى مجموعة متنوعة من المشاكل، منها مشكلة الجمود، والتي سنتطرق لها في الفصل الرابع، بالتالي يجب أن يكون نظام التشغيل قادرًا على التعامل مع كل الأجهزة المشتركة والمخصصة بطريقة– ما– تُتيح تجنب مثل هذه المشاكل.

أخيرًا، عند استخدام الحاسوب لغرض الوصول إلى تطبيق– ما، يتم إخفاء العديد من هذه المفاهيم والاختلافات من قبل نظام التشغيل. وللقيام بذلك تُجمع الأجهزة في عدد قليل من الأنواع التقليدية وتُطور أنماط للوصول إلى الأجهزة تكون مفيدة وقابلة للتطبيق على نطاق واسع، على الرغم من أن إجراءات استدعات النظام الفعلية قد تختلف باختلاف نظام التشغيل.

فئات الأجهزة تكون قياسية إلى حد ما، وبالتالي تتضمن عدة طرق للوصول الرئيسي لهذه

الأجهزة منها: الوصول المقطعي والحرفي لوحدات الإدخال، والإخراج، والوصول المعنون لوحدات الإدخال، والإخراج، أيضًا تُوفر أنظمة التشغيل استدعات نظام خاصة للوصول إلى عدد قليل من الأجهزة الإضافية، مثل ساعة التوقيت اليومي وجهاز ضبط الوقت، والبعض الآخر يُوفر مجموعة من استدعات النظام الخاصة بالشاشة الرسومية والفيديو وأجهزة الصوت، القسم التالي يتعرض لطرق التفاعل مع أجهزة الإدخال، والإخراج.

2.6.3 طرق الإدخال، والإخراج

هناك ثلاثة طرق مختلفة جوهريًا يمكن بها إنجاز عمليات الإدخال، والإخراج. يتناول هذا القسم هذه الطرق بنوع من التفصيل، والتي تتضمن: الإدخال، والإخراج المبرمج، والإدخال، والإخراج بالمقاطعة، والإدخال، والإخراج باستخدام الوصول المباشر للذاكرة.

1.2.6.3 الإدخال، والإخراج المبرمج

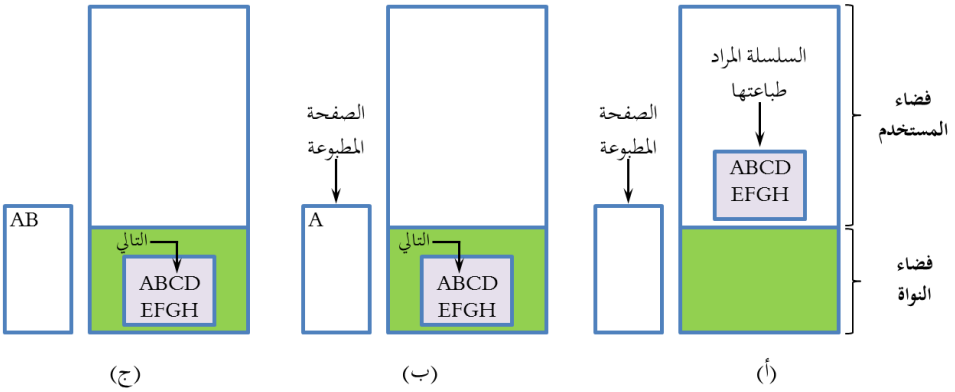
يُعتبر الإدخال، والإخراج المبرمج من أبسط أشكال الإدخال، والإخراج، وفيه تكون وحدة المعالجة المركزية هي المسؤولة عن كل أعمال الإدخال، والإخراج، حيث يُصدر المعالج أمر الإدخال، والإخراج– نيابة عن العملية– إلى وحدة الإدخال، والإخراج المناسبة وينتظر حتى تكتمل العملية، إلَّا إنَّ المعالج ستقع على عاتقه عملية التحقق من حالة وحدة الإدخال، والإخراج بشكل دوري، إلى أن يكتمل تنفيذ العملية، الأمر الذي قد يتسبب في ضياع وقت المعالج وخصوصًا إذا كان المعالج أسرع من وحدة الإدخال، والإخراج.

ولتوضيح فكرة عمل هذه الطريقة، يمكن تتبع المثال التالي: بفرض أن هناك عملية مستخدم ترغب في سحب سلسلة الحروف المكونة من ثمانية أحرف **'ABCDEFGH'** على الطابعة، بالتالي ستكون خطوات تنفيذ هذا المثال كما يلي:

1. تُجمَّع السلسلة في المخزن اللحظي في فضاء المستخدم،كما هو موضح في الشكل 3. **12**–أ.
2. تُقدِّم عملية المستخدم طلبًا من أجل اكتساب الطابعة لغرض السحب عن طريق إجراء أمر استدعاء نظام. إذا كانت الطابعة قيد الاستخدام بواسطة عملية أخرى، سيفشل هذا

الاستدعاء، وستُرجِع رسالة خطأ، أو سوف يُحجب إلى حين تحرر الطابعة، وذلك اعتمادًا على نظام التشغيل، والمعاملات الممرَّرَة في هذا الاستدعاء.

3. في حالة توفر الطابعة، تقوم عملية المستخدم باستدعاء نظام مرة ثانية، لكي تجعل نظام التشغيل يسحب السلسلة على الطابعة.



الشكل 3. 12: خطوات سحب سلسلة من الحروف [Tanenbaum & Bos 2015].

4. ينسخ نظام التشغيل (عادة) المخزن اللحظي متضمنًا السلسلة الحرفية إلى مصفوفة موجودة في فضاء النواة، لكي يتم الوصول إليها بسهولة، كما هو موضح في الشكل 3. 12–ب.
5. يتحقق النظام من التوفر (التحرر) الحالي للطابعة، إذا كانت الطابعة قيد الاستخدام من قبل عملية أخرى، فسينتظر النظام إلى أن تتحرر. بمجرد توفر الطابعة ينسخ نظام التشغيل الحرف الأول في سجل بيانات الطابعة باستخدام الذاكرة المعنونة لوحدة الإدخال، والإخراج، ينُشط هذا الحدث الطابعة، ولكن قد لا يجعلها تُظهر الحرف بعد، لأن بعض الطابعات قد تحجز خط أو صفحة قبل طباعة أي شيء. ومع ذلك، نرى في الشكل 3. 12–ب، أن الحرف الأول قد طُبع، وأن النظام قد وضع علامة على الحرف 'B' للدلالة على أنه سيُسحب في الخطوة التالية.
6. بمجرد نسخ الحرف الأول إلى الطابعة، يتحقق نظام التشغيل من معرفة مدى استعداد الطابعة لقبول حرف آخر. عمومًا، لدى الطابعة سجل ثانٍ يُوضح حالتها.
7. ينتظر نظام التشغيل الطابعة كي تَجهز مرة أخرى، عندما يحدث ذلك، يَسحب النظام الحرف

التالي، كما هو مبين في الشكل 3. 12-ج.

8. تستمر هذه الحلقة حتى تُسحب السلسلة بأكملها، بعدها تُرجَّع السيطرة إلى عملية المستخدم.

تتلخص الإجراءات التي اتبعها نظام التشغيل لكي يسحب سلسلة الحروف في البرنامج المبين في الشكل 3. 13، حيث يتضح في هذا الشكل الجانب الأساسي في الإدخال، والإخراج المبرمج. في البداية، تُنسخ البيانات إلى النواة، بعدها يدخل نظام التشغيل في حلقة تكرارية محكمة بحيث يسحب حرف في كل مرة، بعد سحب الحرف تستطلع وحدة المعالجة المركزية بشكل مستمر الجهاز لمعرفة ما مدى استعداده لقبول حرف آخر، يُسمى هذا السلوك بالانتظار المشغول والذي نُوقش في الفصل السابق.

بالرغم من أن طريقة الإدخال، والإخراج المبرمج بسيطة إلَّا إنَّها تُعاني من بعض من المساوئ المتمثلة في شغل وحدة المعالجة المركزية بالكامل إلى أن تنتهي عملية الإدخال أو الإخراج، وبالتالي إهدار كمية كبيرة من وقت المعالج مع تأثر أدائه. في نظم التشغيل المدمجة، التي ليس لدى المعالج فيها أي عمل آخر يقوم به، يُعتبر الانتظار المشغول مقبولًا، ولكن في الأنظمة المعقدة يكون لدى وحدة المعالجة المركزية الكثير من الأعمال الأخرى التي يجب القيام بها، بالتالي يُعتبر الانتظار المشغول أمرًا غير مقبول، إضافةً إلى ذلك، لا تُمكِن هذه الطريقة وحدة المعالجة المركزية من القيام بعمل آخر في أثناء اختبار الجهاز، كما أنه من الصعب القيام بعدة عمليات إدخال، وإخراج في وقت واحد. لذلك وجب البحث عن طرق إدخال وإخراج أفضل.

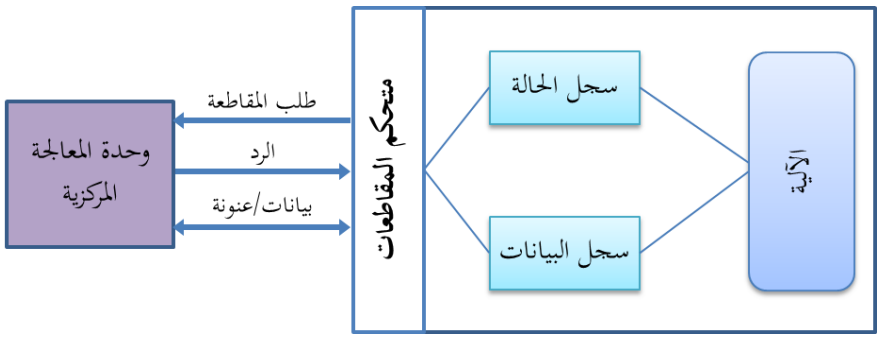
```
copy_from_user(buffer, p, count);        // p is the kernel buffer
for (i = 0; i < count; i++) {            // loop on every character
   while (*printer_status_reg != READY);// loop until ready
   *printer_data_register = p[i];        // output one character
}
return_to_user();
```

الشكل 3. 13: إرسال سلسلة حرفية إلى الطابعة باستخدام الإدخال، والإخراج المبرمج [Tanenbaum & Bos, 2015].

2.2.6.3 الإدخال، والإخراج بالمقاطعة

بالرغم من أن الإدخال، والإخراج المبرمج من أبسط أشكال الإدخال، والإخراج، إلَّا إنَّه يُسبب في ضياع وقت وحدة المعالجة المركزية. لتوضيح هذا يمكن النظر إلى مثال الطابعة التي تسحب البيانات من دون استخدام التخزين اللحظي للحروف، أي تسحب كل حرف بمجرد وصوله إليها. إذا كانت باستطاعة الطابعة سحب 100 حرف لكل ثانية، فإن كل حرف سيستغرق 10 ملِّي ثانية لسحبه، هذا يعني أن بعد كتابة كل حرف في سجل الطابعة، ستبقى وحدة المعالجة المركزية في حلقة الخمول لمدة 10 ملِّي ثانية في انتظار أن يُسمح بسحب الحرف التالي. في الواقع، هذا الوقت يكفي للقيام بعملية الفتح وتشغيل بعض من العمليات الأخرى تصل إلى عشرات الملايين.

لذلك تسعى طريقة الإدخال، والإخراج بالمقاطعة لمعالجة مشكلة ضياع وقت المعالج بجعل مسؤولية إعلام وحدة المعالجة المركزية بإنتهاء عملية الإدخال، والإخراج تقع على عاتق وحدة الإدخال، والإخراج نفسها، أي عند اكتمال العملية تُرسل هذه الوحدة إشارة مقاطعة إلى المعالج لإعلامه بهذا الإنتهاء، والذي سيستكمل التنفيذ الحالي الذي يقوم به في لحظة وصول الإخطار ثم يقف، بعدها يدفع عدَّاد البرنامج في المكدس وينتقل إلى إجراء خدمة المقاطعة لمعالجة مهمة الإدخال، والإخراج. يُوضح الشكل 3. 14 واجهة المقاطعة بالنسبة لوحدات الإدخال، والإخراج.



الشكل 3. 14: واجهة المقاطعة.

بالرجوع إلى مثال سحب سلسلة الحروف في الجزء السابق، يُمكن توضيح عمل هذه

الطريقة كما يلي: عندما يُستدعى النظام لسحب السلسلة، يُنسخ المخزن اللحظي في فضاء النواة ويُرسل الحرف الأول إلى الطابعة في أقرب فرصة تكون فيها على استعداد لقبول الحروف، في هذه اللحظة تستدعي وحدة المعالجة المركزية المجدول وتُشغل بعض من العمليات الأخرى. بالمقابل العملية التي طلبت سحب السلسلة ستدخل حالة موقوفة إلى أن تُسحب السلسلة بأكملها، يُوضح البرنامج المبين في الشكل 3. 15–أ طريقة عمل استدعاء النظام هذا.

عندما تُخرِج الطابعة الحرف وتَجْهز لقبول حرف آخر، فإنها ستُولد مقاطعة، هذه المقاطعة تُوقف العملية الحالية وتحفظ حالتها، ثم يُشغل إجراء خدمة المقاطعة المتعلقة بالطابعة. يُوضح الشكل 3. 15–ب نسخة من برنامج الطباعة هذا. داخل هذا البرنامج إذا لم يكن هناك مزيد من الحروف للسحب، فإن معالج المقاطعة سيقوم ببعض من الإجراءات لغرض إلغاء حظر المستخدم، وإلَّا فإنه سيحب الحرف التالي، ويرد على المقاطعة، ومن ثم يعود إلى العملية التي كانت تشتغل قبل المقاطعة، لتُواصل من حيث توقفت.

```
copy_frorn_user(buffer, p, count);
enable_interrupts();
while (*printer_status_reg != READY);
*printer_data_register = p[0];
scheduler();
```

(أ)

```
if (count == 0){
   unblock_user();
} else{
   *printer_data_register = p[i];
   count = count - 1;
   i = i + 1;
}
acknowledge_interrupt();
return_from_interrupt();
```

(ب)

الشكل 3. 15: إرسال سلسلة حرفية إلى الطابعة باستخدام الإدخال، والإخراج بالمقاطعة– أ) البرنامج المنفذ في نفس لحظة استدعاء نظام الطباعة– ب) إجراء خدمة المقاطعة الخاص بالطابعة [Tanenbaum & Bos, 2015].

من المزايا التي تُقدمها طريقة الإدخال، والإخراج بالمقاطعة هو أنه لا يتعين على المعالج الانتظار حتى اكتمال العملية، وبالتالي، فإنها تُقلل من زمن الوصول وتُحسن في سرعة الأداء العام، كما أنها تسمح بأن تكون هناك عدة عمليات إدخال، وإخراج في نفس الوقت. بالمقابل يُمكن أن تكون طريقة الإدخال، والإخراج بالمقاطعة صعبة البرمجة وخصوصًا إذا ما أُستخدمت لغة غير راقية، كما أنه من الصعب جعل المكونات المختلفة لعملية المقاطعة تعمل معًا بشكل

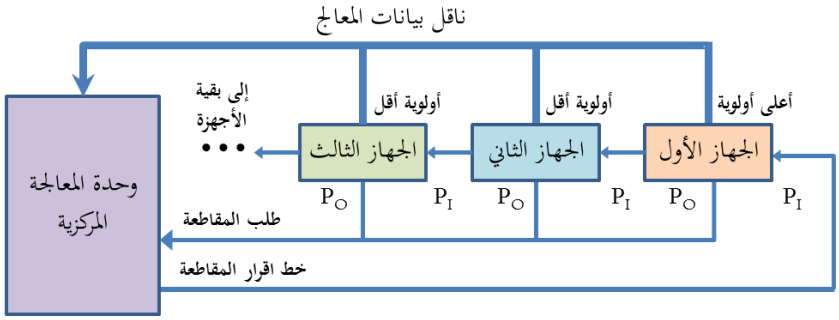
جيد، إلَّا بواسطة الشركة المصنعة للجهاز، أو مُطور نظام التشغيل.

هناك عدة طرق للتعامل مع المقاطعات منها: الاستطلاع، والأولوية المتوازية، وتسلسل ديزي، والتي من بينها تُعتبر الأولوية المتوازية الأكثر شيوعًا.

الأولوية المتوازية: المقاطعة ذات الأولوية المتوازية هي نظام يُقرر الأولوية التي سَتَخدِم بها وحدة المعالجة المركزية أجهزة الإدخال، والإخراج المختلفة، والتي تُولد إشارة المقاطعة في نفس الوقت. في هذا النوع من الأولويات يمتلك النظام سلطة تحديد الشروط المسموح بها لمقاطعة وحدة المعالجة المركزية في أثناء خدمة المقاطعات الأخرى، بشكل عام تتمتع الأجهزة ذات النقل السريع مثل الأقراص بأولوية أعلى من الأجهزة البطيئة مثل لوحات المفاتيح، عليه عند قيام جهازين أو أكثر بمقاطعة وحدة المعالجة المركزية في آنٍ واحد، تخدم الوحدة الجهاز صاحب الأولوية الأعلى أولاً.

تسلسل ديزي: تتمثل طريقة تحديد أولوية المقاطعة في التوصيل المتتالي لكافة الأجهزة التي تُنشئ إشارة مقاطعة كل حسب أولويته، أي يُوضع الجهاز ذو الأولوية الأعلى في أول التسلسل متبوعًا بالجهاز الذي يليه في الأولية، ثم الذي يليه، وهكذا مع بقية الأجهزة ذات الأولويات الأقل، حيث يكون الجهاز الذي يتمتع بأقل أولوية في آخر التسلسل، كما هو مُوضح في الشكل 3. 16.

في التوصيل التسلسلي لجميع الأجهزة بناءً على مفهوم ديزي هناك خط طلب مقاطعة مشترك لجميع الأجهزة. وظيفة هذا الخط هو نقل إشارة طلب المقاطعة من الأجهزة إلى وحدة المعالجة المركزية، والتي تستجيب بدورها في حالة استلامها لإشارة مقاطعة بتمكين خط إقرار المقاطعة، تُستلم إشارة الإقرار هذه أولًا بواسطة الجهاز الأول عند المدخل $\mathbf{P_I}$ الخاص به، إذا كان من تقدم بطلب للمقاطعة هو هذا الجهاز، فسيستجيب لهذا الإقرار ويختزلها، بمعنى يمنع مرورها إلى بقية الأجهزة التالية في التسلسل. بعدها يضع الجهاز متجه المقاطعة في ناقل بيانات وحدة المعالجة المركزية، ويضبط إشارة طلب المقاطعة على المستوى العالي للدلالة على أن الجهاز قد اهتم بالمقاطعة، أمَّا إذا لم يطلب الجهاز الأول المقاطعة، سيُمرر إشارة الإقرار إلى الجهاز التالي في التسلسل من خلال المخرج $\mathbf{P_O}$، والذي سيُكرر نفس الخطوات.



الشكل 3. 16: مخطط نظام أولوية تسلسل ديزي.

3.2.6.3 الإدخال، والإخراج باستخدام الوصول المباشر للذاكرة

من خلال تتبع طريقة المخرجات القائمة على المقاطعة يُمكن ملاحظة أن هذه الطريقة تُسبب أيضًا في ضياع وقت وحدة المعالجة المركزية وذلك مع حدوث المقاطعة عند سحب كل حرف، نتيجةً للوقت المستقطع لعملية المقاطعة. وكحل لهذه المشكلة يمكن استخدام متحكم الوصول المباشر للذاكرة، والذي يُعد طريقةً أخرى لإجراء الإدخال، والإخراج. الفكرة في هذه الطريقة تَكمن في السماح للمتحكم بتغذية الأحرف للطابعة في وقت واحد، دون إضاعة وقت وحدة المعالجة المركزية، هذا الأمر سيُقلل عدد مرات المقاطعة من واحدة عند إخراج كل حرف إلى واحدة عند سحب كل الحروف، وبالتالي الحفاظ على وقت المعالج. تُوضح الشفرة في الشكل 3. 17 طريقة الإدخال، والإخراج هذه باستخدام الوصول المباشر للذاكرة.

```
copy_from_user(buffer, p, count);
set_up_DMA_controller();
scheduler();
```

(أ)

```
acknowledge_interrupt();
unblock_user();
return_from_interrupt();
```

(ب)

الشكل 3. 17: طباعة سلسلة حرفية باستخدام الوصول المباشر للذاكرة– أ) البرنامج المنفذ في نفس لحظة استدعاء نظام الطباعة– ب) إجراء خدمة المقاطعة الخاص بالطابعة [Tanenbaum & Bos, 2015].

تناول القسم الثاني من هذا الفصل متحكم الوصول المباشر للذاكرة بنوع من التفصيل، بالتالي يكفي أن نُراجع هنا بعض من المفاهيم الخاصة بعملية الإدخال، والإخراج عندما يُستخدم

هذا المتحكم. إنَّ استخدام متحكم الوصول المباشر للذاكرة يؤدي إلى إزالة وحدة المعالجة المركزية من مسار النقل والسماح لجهاز الإدخال، والإخراج بإدارة ناقل الذاكرة مباشرةً، الأمر الذي سيؤدي إلى تحسين سرعة النقل.

وللقيام بذلك تُزود وحدة المعالجة المركزية متحكم الوصول المباشر للذاكرة بمعلومات التحكم الضرورية مثل القطاعات المطلوب قراءتها، وعنوان البداية، وحجم البيانات المراد نقلها، وعنوان المكان الذي سيُخزن فيه المقطع داخل الذاكرة، بعدها يبدأ المتحكم في نقل البيانات، بينما تتفرغ وحدة المعالجة المركزية للقيام بمهام أخرى. عند إنتهاء المتحكم من عملية نقل القطاعات، يُقاطع وحدة المعالجة المركزية لإعلامها بإنتهاء عملية النقل.

بالرغم من كفاءة طريقة الإدخال، والإخراج باستخدام متحكم الوصول المباشر للذاكرة في نقل البيانات بين الجهاز والذاكرة بشكل مستقل عن وحدة المعالجة المركزية، إلَّا إنَّه زاد من تكلفة الأجهزة الداعمة له، وأضاف تعقيدات برمجية، كما أنَّ عدم مواكبة هذا المتحكم لسرعة وحدة المعالجة المركزية ستُقلل من كفاءة هذا النقل وخصوصًا إذا لم يكن لدى وحدة المعالجة المركزية أي عمل تقوم به في أثناء انتظارها للمتحكم، الأمر الذي قد يجعل طريقة الإدخال، والإخراج بالمقاطعة، أو حتى الإدخال، والإخراج المبرمج أفضل.

7.3 طبقات برمجيات الإدخال، والإخراج

تُنظم برمجيات الإدخال، والإخراج الحديثة في أربع طبقات مجردة، بحيث تكون لكل طبقة وظيفة محددة بشكل جيد، ولها واجهة بينية مشتركة واضحة المعالم إلى الطبقات المجاورة تسمح بدرجة عالية من المرونة. من فوائد هذا التنظيم هو إمكانية تعديل الطبقات بشكل مستقل دون التأثير على الطبقات المجاورة، هذه الطبقات هي:

- برمجيات الإدخال، والإخراج لفضاء المستخدم.
- البرمجيات المستقلة لأجهزة الإدخال، والإخراج.
- مشغل الجهاز.
- معالجة المقاطعات.

تختلف الوظائف والواجهات من نظام إلى آخر، كما أنه– ونتيجةً لهذا التجريد– لن تكون

هناك حاجة لمعرفة كيفية عمل كل طبقة، المهم هو معرفة كيفية التفاعل مع هذه الطبقات وكذلك الواجهات، بالتالي فإن النقاش التالي سيتناول جميع الطبقات بدءًا من أسفل الطبقات إلى أعلاها، وهو ليس مخصصًا لآلة محددة.

1.7.3 طبقة معالجة المقاطعات

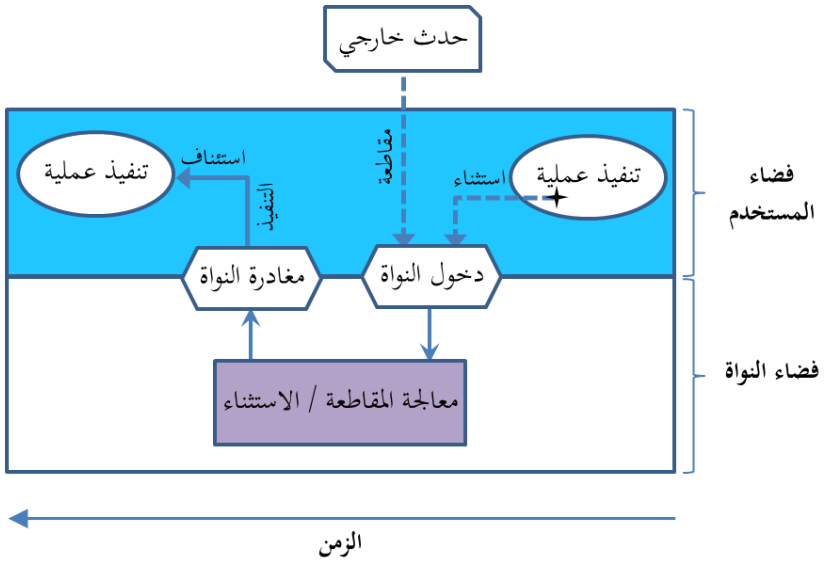
تقع طبقة معالجة المقاطعات مباشرةً فوق الكيان المادي، ومهمتها هي تحقيق التوافق بين العمليات في أثناء تنفيذها وخصوصًا تلك التي ترغب في القيام بعمليات الإدخال، والإخراج. كنا قد أشرنا في الفصل الثاني إلى أنَّ العمليات خلال تنفيذها تمر بمجموعة من الحالات منها تشتغل، وموقوفة، وجاهزة، لذلك تنقل هذه الطبقة العمليات من حالة تشتغل إلى حالة موقوفة عندما تُطظر العملية إلى انتظار أحداث خارجية متعلقة بوحدات الإدخال، والإخراج، أو حدوث استثناءات في تنفيذ العملية، ومن ثم إرجاعها إلى حالة جاهزة عند توفر هذه الأحداث لكي تُواصل تنفيذها فيما بعد.

يُوضح الشكل 3. 18 آلية حدوث المقاطعة أو الاستثناء، كما هو موضح في هذا الشكل عند حدوث أحدهما يحدث انتقال من وضع المستخدم إلى وضع النواة حيث تتم معالجتهما هناك، بعدها يُستأنف التنفيذ في فضاء المستخدم.

تُعتبر عمليات المقاطعة حقيقة غير مرغوب فيها لمعظم وحدات الإدخال، والإخراج، إلَّا إنَّه لا يمكن تجنبها لأهميتها، لذلك ينبغي أن تكون هذه العمليات مخفية في أعماق نظام التشغيل، ولا تتعامل معها إلَّا أجزاء قليلة من هذا النظام. وكأفضل طريقة لإخفائها هو أن يُوقف مشغل الجهار الذي بدأ عملية الإدخال، والإخراج إلى أن تتوفر الأحداث الخارجية وتكتمل العملية ومن ثم تحدث المقاطعة. هذا المشغل يُمكنه أيضًا إيقاف نفسه عن طريق القيام بعملية **Down** على متغير الإشارة أو الانتظار لتحقق متغيرًا شرطيًا.

عندما تحدث المقاطعة، يقوم إجراء المقاطعة بكل ما يتطلبه الأمر من أجل معالجة المقاطعة، وتحديث هياكل البيانات، وإيقاظ العملية التي كانت تنتظر في حدوث المقاطعة، أي نقل المشغل إلى حالة جاهزة لغرض مواصلة عمله المناط به. وللقيام بذلك يُنفذ إجراء المقاطعة في بعض الحالات عملية **Up** على متغير الإشارة، بذلك تكون النتيجة النهائية للمقاطعة هي: أن

المشغل الذي تم حظره سابقًا سيكون الآن قادرًا على الاشتغال، هذا النموذج يعمل بشكل أفضل إذا ما تمت هيكلة المشغلات كعمليات نواة لها حالاتها الخاصة، ومكادسها، وكذلك عدَّادات البرنامج الخاصة بها.



الشكل 3. 18: آلية المقاطعة / الاستثناء.

إنَّ معالجة المقاطعة ليست مجرد مسألة اتخاذ المقاطعة، والقيام بتنفيذ إجراء المقاطعة، والعودة منها إلى العملية السابقة. هناك عدة خطوات مناطة بنظام التشغيل والتي يجب تنفيذها كجزء من الكيان المعنوي بعد اكتمال المقاطعة المادية. تجدر الإشارة هنا إلى أن تفاصيل هذه الخطوات تعتمد على نوعية النظام نفسه، حتى إنَّ بعض من الخطوات المذكورة أدناه قد لا تكون مطلوبة على آلة معينة، وقد تكون هناك خطوات أخرى غير مدرجة، كما قد يختلف ترتيب هذه الخطوات باختلاف الأجهزة. بناءً على آلية حدوث المقاطعة أو الاستثناء الموضحة في الشكل 3. 18، تكمن تفاصيل خطوات المعالجة في النقاط التالية:

1. في أثناء الدخول إلى فضاء النواة، يجب أولاً حفظ حالة وحدة المعالجة المركزية، وقيم كافة سجلاتها الخاصة بعملية التنفيذ الحالية في الذاكرة والتي لم تُحفظ من قبل المقاطعة المادية، وذلك لغرض الاستعانة بها لاحقًا في استئناف تنفيذ العملية.

2. داخل فضاء النواة يُتعامل مع المقاطعة أو الاستثناء من حيث:

- تحديد سبب المقاطعة أو الاستثناء: هناك عدة أسباب وراء حقن المقاطعة أو الاستثناء، لذلك يحتاج المعالج إلى تحديد المسبب الرئيسي لها وذلك من خلال الاطلاع على المعلومات المحتفظ بها في سجلات مخصصة أو في عناوين محددة مسبقًا في الذاكرة.
- خدمة المقاطعة والاستثناء: بعد تحديد سبب المقاطعة يُنفَّذ إجراء خدمة المقاطعة المناسب للقيام بخدمة هذه المقاطعة. على سبيل المثال، إذا كانت المقاطعة بسبب لوحة المفاتيح، فسيتم الحصول على رمز المفتاح المضغوط عليه وتخزينه في مكان– ما–، أو اتخاذ بعض من الإجراءات المناسبة الأخرى.

3. عند الإنتهاء من معالجة المقاطعة أو الاستثناء، تقوم النواة بالخطوات التالية:

- تحديد عملية لاستئنافها: بعد الإنتهاء من معالجة المقاطعة، قد تختار النواة استئناف نفس العملية التي كانت تُنفذ قبل المعالجة أو قد يُستأنف تنفيذ أي عملية أخرى موجودة حاليًا في الذاكرة، وذلك استنادا على أولويات التنفيذ.
- استعادة حالة أو سياق العملية المختارة: يمكن الآن استعادة حالة وحدة المعالجة المركزية بالنسبة للعملية المختارة من خلال قراءة واستعادة جميع قيم السجلات من الذاكرة والتي حُفظت في الخطوة رقم 1.
- استئناف تنفيذ العملية المختارة: يجب استئناف العملية التي أُختيرت للاستئناف من النقطة نفسها التي توقفت عندها وذلك بالرجوع إلى عنوان هذه التعليمة والذي حُفظ من قبل الجهاز عند حدوث المقاطعة. الأمر يتعلق بمسألة الحصول على هذا العنوان وجعل وحدة المعالجة المركزية تستمر في التنفيذ انطلاقًا منه.

من خلال ما سبق يُمكن ملاحظة أن معالجة المقاطعة ليس بالأمر الهين، فهي تتطلب عددًا كبيرًا من تعليمات وحدة المعالجة المركزية ومن التعامل مع طبقة مشغل الجهاز والتي سنناقشها تاليًا.

2.7.3 طبقة مشغل الجهاز

تعرضنا سابقًا في هذا الفصل إلى متحكم الجهاز الذي من خلاله لاحظنا أنَّ لكل متحكم بعض من السجلات التي تُستخدم لإصدار الأوامر إليه والبعض الآخر يُستخدم في قراءة حالته.

عدد هذه السجلات وطبيعة الأوامر تختلف جذريًا من جهاز إلى آخر. مثلًا، على برنامج مشغل لوحة المفاتيح استقبال معلومات من اللوحة نفسها، تُخبره عن أيٍ من الأزرار التي ضُغطت حاليًا. في المقابل قد يحتاج برنامج مشغل القرص لمعرفة كل شيء عن القطاعات، والمسارات، والأسطوانات، والرؤوس، وأذرع الحركة، وكذلك جميع المكونات الأخرى للقرص التي تجعله يعمل بشكل صحيح. إذًا من الواضح أن برامج المشغلات هذه سوف تكون مختلفة عن بعضها البعض.

ونتيجة لذلك، يحتاج كل جهاز إدخال، وإخراج متصل بالحاسوب إلى بعض من البرامج الخاصة من أجل التحكم فيه، هذه البرامج تُعرف باسم مشغل الجهاز، الذي في العموم يُكتب من قبل الشركة المصنعة للجهاز، لكونها على دراية كاملة بخصوصياته. التعريف الشائع لمشغل الجهاز هو مجموعة من الملفات البرمجية التي تُمكّن نظام تشغيل الحاسوب من الاتصال بأي جهاز والذي من دونه لن يتمكن الحاسوب من إرسال البيانات إلى الأجهزة وتلقيها منها بشكل صحيح. من ناحية أخرى يُوفر مشغل الجهاز واجهة برمجية للأجهزة تُتيح لنظام التشغيل وبرامج الحاسوب الأخرى الوصول إلى وظائف الأجهزة دون الحاجة إلى معرفة التفاصيل الدقيقة الخاصة بالأجهزة المستخدمة.

ولأن كل نظام تشغيل يحتاج إلى مشغلات أجهزة خاصة به، فإن مصنعي الأجهزة عادةً ما يوفرون عدة نسخ من مشغلات أجهزة لكي تتوافق مع أنظمة التشغيل الشائعة، فكثيرًا ما يُعالج المشغل نوعًا واحدًا من الأجهزة، أو على الأكثر، فئة واحدة من الأجهزة المرتبطة ببعضها البعض إرتباطًا وثيقًا، لأنه من المفيد جدًا للنظام أن يكون هناك مشغل قرص واحد يدعم عدة أقراص ذات أحجام وسرعات مختلفة بدلًا من أنْ يُضمَّن مشغل لكل قرص.

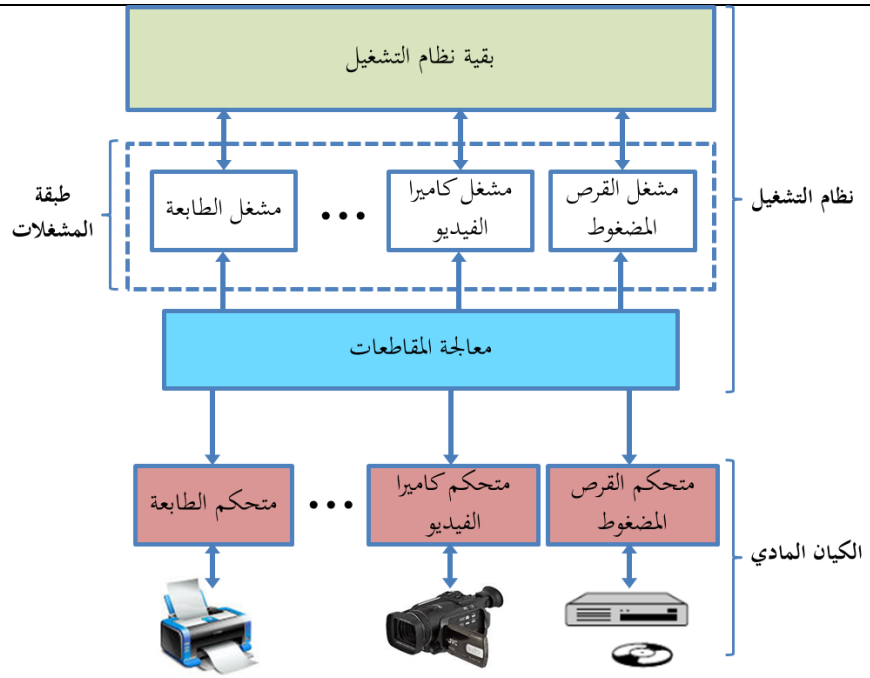
لبدء تشغيل عملية الإدخال، والإخراج، يُحمِّل مشغل الجهاز السجلات المناسبة داخل متحكم الجهاز والذي يفحص بدوره محتويات هذه السجلات لتحديد الإجراء المطلوب اتخاذه (مثل: قراءة حرف من لوحة المفاتيح). يبدأ المتحكم في نقل البيانات من الجهاز إلى مخزنه اللحظي وبمجرد اكتمال عملية النقل، يُبلِّغ مشغل الجهاز من خلال المقاطعة بإنتهاء العملية، ويُرجِّع بعدها التحكم إلى نظام التشغيل. من المحتمل إرجاع البيانات أو مؤشر إلى البيانات إذا كانت العملية قراءة. بالنسبة لبعض من العمليات الأخرى يقوم المشغل بإرجاع معلومات الحالة.

يوضح الشكل 3. 19 موقع مشغلات الأجهزة التي عادة ما تُثبت أسفل بقية نظام التشغيل.

هناك عدة أنواع من مشغلات الأجهزة، يختص كل منها بنوع مختلف من الإدخال، والإخراج، من هذه المشغلات مشغل الأجهزة المقطعية التي تحوي وسائط تخزين يمكن عنونتها فعليًا مثل، الأقراص. معظم المشغلات الأخرى تُعتبر مشغلات أجهزة حرفية مثل، لوحات المفاتيح، والطابعات، التي تُولد أو تستقبل تيارًا من الأحرف التتابعية. تُستخدم المشغلات الحرفية مع الناقلات، والمنافذ التسلسلية، والتي تتعامل مع البيانات حرفًا بحرف، الفأرة هي أيضًا جهاز تسلسلي ويحتوي على مشغل جهاز حرفي.

معظم نظم التشغيل تُعرِّف واجهة بينية قياسية تَدعم جميع مشغلات الأجهزة المقطعية، وواجهة بينية قياسية ثانوية تَدعم جميع مشغلات الأجهزة الحرفية. تتكون هذه الواجهات من عدد من الإجراءات التي يُمكن استدعاؤها من قبل بقية نظام التشغيل للحصول على مشغل يُنجز لها العمل، من الإجراءات النموذجية المستخدمة هنا هي تلك الإجراءات التي تقوم بقراءة مقطع (جهاز مقطعي) أو بكتابة سلسلة أحرف (جهاز حرفي).

كملخص لهذه الطبقة تُوفر طبقة مشغل الجهاز طريقة للتعامل مع أجهزة الإدخال، والإخراج المختلفة بغض النظر عن اختلافها من حيث الوظيفة، والنوع، بحيث يُوفر كل مشغل جهاز طريقة للتعامل مع جهاز معين أو مجموعة من الأجهزة مثال ذلك، وجود مشغل للفأرة يَدعم التعامل مع أكثر من نوع منها. في العموم تتمثل وظيفة هذه الطبقة في استقبال طلبات مجردة من طبقة برمجيات الأجهزة المستقلة، وذلك لغرض تنفيذها، في حالة استقبال هذه الطبقة لأمر– ما– وكان المشغل في نفس زمن وصول الطلبية غير مشغول فستقوم مباشرةً بتنفيذ الأمر، وإلا فسوف يدخل هذا الأمر في طابور الطلبيات ليُنفذه المشغل لاحقًا. هناك أيضًا عدد قليل من الوظائف الأخرى التي يجب على المشغل القيام بها منها: تهيئة الجهاز– إذا لزم الأمر–، وتفسير الأوامر الصادرة من نظام التشغيل، وجدولة العديد من الطلبات المعلقة، وقبول المقاطعات ومعالجتها، وإدارة نقل البيانات أو القيام بإدارة متطلبات الطاقة.



الشكل 3. 19: موقع مشغلات الأجهزة، ملاحظة: جميع الاتصالات بينها وبين المتحكمات تتم بواسطة ناقل النظام.

3.7.3 طبقة البرمجيات المستقلة لأجهزة الإدخال، والإخراج

على الرغم من أن بعض من برمجيات الإدخال، والإخراج (المشغلات) مخصصة لجهاز معين والبعض الآخر مستقل عنه، إلَّا إنَّ الحدود الفعلية بين المشغلات والبرمجيات المستقلة للأجهزة تعتمد على الجهاز نفسه، لأنَّ بعض من الوظائف التي من الممكن أن تُنفذ بطريقة مستقلة عن الأجهزة يمكن في الواقع القيام بها في المشغلات، وذلك لغرض تحقيق الكفاءة أو لأسباب أخرى. تُوضح النقاط التالية الوظائف التي غالبًا ما يُقام بها في البرمجيات المستقلة للأجهزة.

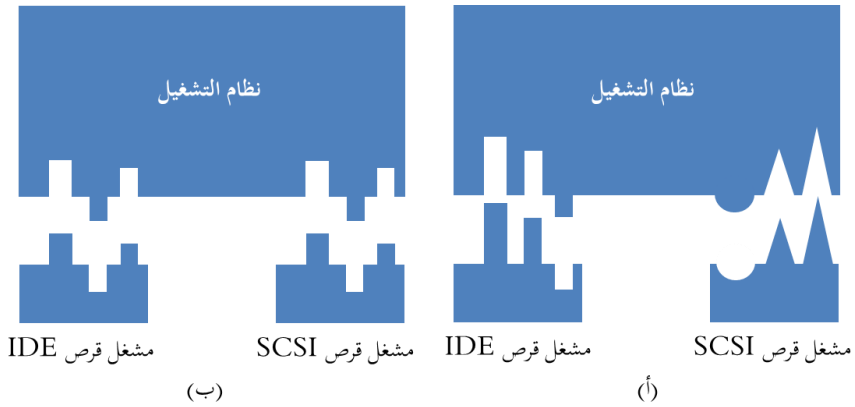
- توفير واجهة بينية مستقلة عن مشغلات الأجهزة.
- التخزين اللحظي.
- الإبلاغ عن الأخطاء.

- تخصيص وتحرير الأجهزة المخصصة.
- توفير حجم قطاع منتظم ومستقل عن الأجهزة.

فيما يلي سيتم النظر إلى المهام المذكورة أعلاه بمزيد من التفاصيل.

توفير واجهة بينية مستقلة عن مشغلات الأجهزة: من المسائل المهمة والرئيسية في نظام التشغيل وخاصة فيما يتعلق بأجهزة الإدخال، والإخراج هي كيفية جعلها هي والمشغلات تبدو إلى حد ما كأنها جهاز واحد، بالرغم من الاختلافات الجوهرية بينها. فلو افترضنا أن الأقراص، والطابعات، ولوحات المفاتيح جميعها مرتبطة بواجهات بينية مختلفة، فإنه في كل مرة يُضاف فيها جهاز جديد يجب تعديل نظام التشغيل ليتوافق مع الجهاز المضاف. الحاجة إلى تعديل نظام التشغيل بعد كل عملية إضافة جهاز هي في الواقع عمل مضني.

يُوضح الشكل 3. 20–أ الحالة التي تكون فيها مشغلات الأجهزة تحتوي على واجهات بينية مختلفة مرتبطة مع بقية نظام التشغيل، هذه الحالة تعني أن وظائف المشغلات المتاحة للنظام لغرض استدعائها تختلف من مشغل لآخر، وهو ما يعني أيضًا أن وظائف النواة التي يحتاجها المشغل تختلف باختلاف المشغل. وبأخذ كل هذا في عين الاعتبار، فإن إضافة مشغل جديد يتطلب الكثير من الجهد والوقت.



الشكل 3. 20: أ) واجهة بينية غير منظمة لمشغلات الأجهزة– ب) واجهة بينية منظمة لمشغلات الأجهزة.

في المقابل، يُوضح الشكل 3. 20–ب تصميمًا مختلفًا تتوحد فيه الواجهة البينية لكافة المشغلات، الأمر الذي يُسهل بشكل كبير إضافة مشغلات جديدة، شرط أن تتوافق مع هذه الواجهات، يسمح هذا المفهوم لبرامج التطبيقات بالتواصل مع أي جهاز بأسلوب مماثل بغض النظر عن صُنع الجهاز وطبيعته. هذا يعني أنه يجب على مبرمجي التطبيقات إدخال البيانات إلى أو إخراجها من جهاز منطقي، والذي يُقدم تمثيلًا افتراضيًا لأي جهاز متصل بنظام التشغيل، يجعل هذا الأمر المستخدم غير مهتمًا بالتنفيذ الداخلي، وتفاصيل الاتصالات منخفضة المستوى.

ولتوفير واجهة بينية منظمة ومستقلة لمشغلات الأجهزة يُحدد نظام التشغيل لكل فئة من الأجهزة– مثل الأقراص أو الطابعات– مجموعة من المهام التي يجب على المشغل تقديمها. ففي ما يتعلق بالقرص، تشمل هذه المهام القراءة والكتابة، وتزويد وفصل الطاقة، وعملية التهيئة، وبعض من المهام الأخرى. وللقيام بذلك يُزود المشغل– في كثير من الأحيان– بجدول مؤشرات لهذه المهام، بحيث يُسجل نظام التشغيل عنوان هذا الجدول عند تحميل برنامج المشغل، يُمكِّن هذا الأمر نظام التشغيل من استدعاء إحدى هذه المهام بطريقة غير مباشرة من خلال هذا الجدول، لذلك يُعتبر جدول مؤشرات المهام الوسيط أو الواجهة البينية المستقلة بين المشغل وبقية نظام التشغيل، والذي يجب على جميع الأجهزة من نفس الفئة المحددة التقيد به.

من جوانب الواجهة البينية المستقلة لمشغلات الأجهزة هو الاهتمام بالكيفة التي تُسمى بها أجهزة الإدخال، والإخراج، فالبرمجيات المستقلة لهذه الأجهزة تعتني بمواءمة الأسماء الرمزية للأجهزة مع المشغل الصحيح، كما أن هناك أمر آخر مرتبط ارتباطًا وثيقًا بالتسمية، وهو الحماية. فالسؤال المطروح هو كيف يمكن للنظام منع المستخدمين من الوصول إلى الأجهزة غير المسموح لهم باستخدامها؟ في كل من نظام 'ويندوز' و'يونيكس'، تظهر الأجهزة في نظام الملفات ككائنات اسميه، وهو ما يعني أن قواعد الحماية المعتادة للملفات تنطبق على أجهزة الإدخال، والإخراج، بالتالي يُمكن لمسؤول النظام تعيين الأذونات المناسبة لكل جهاز.

التخزين اللحظي: يُعتبر التخزين اللحظي بالنسبة للأجهزة المقطعية والحرفية من الأشياء المهمة، وذلك لعدة أسباب. يتمثل إحداها في الآتي: بفرض أن هناك عملية– ما– ترغب في قراءة بيانات من جهاز الإدخال، والإخراج، إحدى الاستراتيجيات الممكنة للتعامل مع الحروف الواردة من هذا الجهاز هي أن تُفعل عملية المستخدم استدعاء نظام القراءة، ومن ثم تتوقف

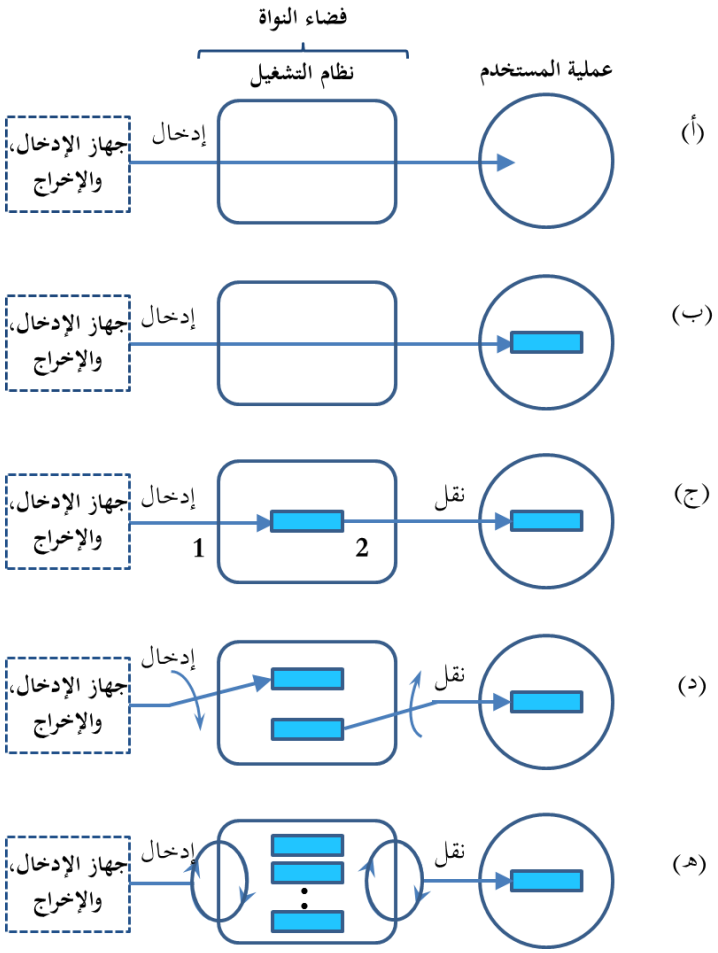
لانتظار وصول الحرف، عند وصوله تحدث المقاطعة ويُسلِّم إجراء خدمة المقاطعة على إثرها الحرف إلى عملية المستخدم ويُوقضها لمواصلة عملها. بعد وضع الحرف في مكان– ما،– تقرأ العملية حرفًا آخر، وتتوقف مرة أخرى، وتتكرر الخطوات نفسها. يُوضح الشكل 3. 21–أ نموذج هذه الآلية من القراءة. المشكلة مع هذه الطريقة تتمثل في أن عملية المستخدم يجب تشغيلها مع إدخال كل حرف، الأمر الذي يترتب عليه السماح للعملية بالاشتغال عدة مرات ولفترات زمنية قصيرة، الأمر الذي بدوره سيجعل هذا النموذج غير فعَّال وليس بالفكرة الجيدة.

يُظهر الشكل 3. 21–ب تحسينًا في النموذج السابق والمتمثل في وجود مخزن لحظي في فضاء المستخدم مُجهز من قبل عملية المستخدم، يسع هذا المخزن عددًا محددًا من الحروف، ويُستخدم من قبل هذه العملية لقراءة العدد نفسه من الحروف، يضع إجراء خدمة المقاطعة الحروف المدخلة في المخزن اللحظي إلى أن يمتلئ، عندها يُوقظ الإجراء عملية المستخدم. يُعتبر هذا التصميم أكثر كفاءةً من التصميم السابق، ولكنه قد يُعاني من مشكلة امتلاء المخزن في أثناء وصول الحروف، الأمر الذي يتطلب ترحيل المخزن اللحظي إلى الذاكرة وتأمينه هناك. ولكن إذا أمَّنت العديد من العمليات صفحاتها في الذاكرة، فإن مجموعة الصفحات المتوفرة سوف تتقلص، وهو ما سيؤدي بدوره إلى انخفاض مستوى الأداء.

من الاستراتيجيات الأخرى للتعامل مع الحروف الواردة من جهاز الإدخال، والإخراج هي تعيين مخزن لحظي داخل فضاء النواة، بحيث تُوضع الحروف المدخلة داخله من قبل إجراء معالجة المقاطعة، كما هو مبين في الشكل 3. 21–ج. عندما يمتلئ هذا المخزن، تُنقل صفحة مخزن المستخدم إلى فضاء المستخدم– إذا لزم الأمر ذلك– ويُنسخ المخزن اللحظي هناك في عملية واحدة، هذا المخطط هو أكثر كفاءةً من المخططين السابقين.

بالنظر إلى المخطط السابق يمكن ملاحظة أنه يُعاني من بعض القيود على سبيل المثال، ماذا سيحدث للأحرف التي تصل في أثناء جلب الصفحة الخاصة بالمخزن اللحظي للمستخدم من القرص؟ بما أن المخزن اللحظي ممتلئ بالكامل، فسوف لن تكون هناك إمكانية لإضافة أحرف جديدة به، يتمثل المخرج من هذه المشكلة في إضافة مخزن لحظي آخر في فضاء النواة بحيث تُدخل الأحرف إلى أحد المخازن اللحظية بينما يُفرغ نظام التشغيل المخزن الآخر في فضاء المستخدم، كما هو موضح في الشكل 3. 21–د، تُسمى هذه الطريقة بالتخزين اللحظي

المزدوج.



الشكل 3. 21: أ) الإدخال بدون التخزين اللحظي– ب) التخزين اللحظي في فضاء المستخدم– ج) التخزين اللحظي في فضاء النواة متبوعًا بنسخ في فضاء المستخدم– د) التخزين اللحظي المزدوج في فضاء النواة– هـ) التخزين اللحظي الدائري.

قد يكون التخزين اللحظي المزدوج غير ملائم، إذا كانت عمليات الإدخال، والإخراج تحدث بشكل سريع، بالتالي يمكن استخدام المخزن اللحظي الدائري والذي يستخدم أكثر من مخزنين لحظيين، بحيث يُمثل كل مخزن لحظي وحدة واحدة في المخزن اللحظي الدائري، كما

هو موضح في الشكل 3. 21–هـ، تُستخدم هذه الطريقة على نطاق واسع وتتألف من منطقة في الذاكرة ومؤشرين، أحدهما يُشير إلى الموقع الحر التالي، الذي يمكن وضع البيانات الجديدة فيه، والآخر يُشير إلى الموقع الأول من البيانات الموجودة في المخزن الدائري والتي لم تُنقل بعد.

أخيرًا، يُستخدم التخزين اللحظي على نطاق واسع لعدة أسباب منها:

- للتغلب على مشكلة عدم تطابق السرعة بين المنتج (أو المرسل) والمستهلك (أو المتلقي) لتيار البيانات.
- للمواءمة بين الأجهزة ذات الأحجام المختلفة لنقل البيانات.
- في بيئة البرمجة المتعددة، عندما يكون هناك مجموعة متنوعة من نشاطات الإدخال، والإخراج ونشاطات لعمليات متنوعة للخدمة، فإن التخزين اللحظي هو أحد الأدوات التي يمكن أن تزيد من كفاءة نظام التشغيل وأداء العمليات الفردية.

بالرغم من هذه المزايا، إلَّا إنَّ للتخزين اللحظي جانب سلبي متمثل في كون كثرة أو تعدد التخزين اللحظي للبيانات يؤدي إلى التأثير على كفاءة النظام بشكل عام.

الإبلاغ عن الأخطاء: الأخطاء هي أكثر شيوعًا في سياق التعامل مع وحدات الإدخال، والإخراج عنها مع باقي مكونات الحاسوب، بالتالي عند حدوثها، يجب على نظام التشغيل التعامل معها على أفضل وجه ممكن، فالعديد من هذه الأخطاء هي أخطاء خاصة بالجهاز نفسه، ويجب التعامل معها عن طريق المشغل المناسب، ولكن الإطار العام لمعالجتها هو في الواقع مستقل عن الجهاز.

إحدى فئات أخطاء الإدخال، والإخراج هي أخطاء البرمجة، هذا النوع من الأخطاء يحدث عندما تطلب عملية– ما– شيئًا مستحيلًا، مثل الكتابة إلى جهاز إدخال (لوحة المفاتيح)، أو القراءة من جهاز إخراج (الراسمة). هناك أخطاء أخرى تتمثل في توفر عنوان مخزن لحظي غير صالح، أو معلمة غير صحيحة، أو محاولة التعامل مع جهاز غير موجود (على سبيل المثال، تحديد قرص ثالث في حين أن النظام يمتلك قرصين فقط). وكردة فعل على مثل هذه الأخطاء يُبلَّغ عنها من خلال إرسال شفرة الخطأ المناسبة.

فئة أخرى من الأخطاء هي فئة أخطاء الإدخال، والإخراج الفعلية مثل، محاولة الكتابة في

مقطع قرص متضرر أو محاولة القراءة من كاميرا الفيديو وهي مقفلة. في مثل هذه الحالات الأمر متروك للمشغل لتحديد ما يجب القيام به، أمَّا إذا لم يعرف المشغل ماذا سيفعل، فإنه قد يُمرر المشكلة إلى البرمجيات المستقلة لأجهزة الإدخال، والإخراج.

طريقة معالجة البرماجيات للأخطاء يعتمد على بيئة وطبيعة الخطأ نفسه، فإذا كان خطأ قراءة بسيط، وكان هناك مستخدم تفاعلي متاح فسيظهر مربع حوار يحث فيه المستخدم على القيام بعمل محدد، يمكن أن تشمل الخيارات إعادة المحاولة عدة مرات، أو تجاهل الخطأ، أو قتل عملية الاستدعاء، أمَّا إذا لم يكن هناك أي مستخدم متاح، فلربما يكون الخيار الحقيقي الوحيد هو أن يتم إفشال استدعاء النظام مع تمرير شفرة الخطأ. ومع ذلك، هناك بعض من الأخطاء لا يمكن التعامل معها بهذه الطريقة، فقد يُدمر مثلًا أحد بنيان البيانات المهمة، مثل جذر الدليل، أو قائمة المقاطع الحرة، في هذه الحالة، قد يكون من الأجدى على النظام عرض رسالة خطأ وإنهاء البرنامج.

تخصيص وتحرير الأجهزة المخصصة: بعض من الأجهزة مثل، سواقة الأقراص المدمجة لا يمكن استخدامها إلا من قبل عملية واحدة في أي لحظة معطاة. في هذه الحالة الأمر متروك لنظام التشغيل لفحص طلبات استخدام الجهاز ومن ثم قبولها أو رفضها بناءً على ما إذا كان الجهاز المطلوب متاحًا أم لا، علمًا أنَّ هناك آلية خاصة لتتبع طلب أو تحرير الأجهزة المخصصة. فمحاولة الحصول على جهاز غير متاح ستُوقِف العملية المستدعية بدلًا من فشلها، الأمر الذي يترتب عليه وضع العمليات الموقوفة في طابور. عاجلًا أم آجلًا، سيتحرر الجهاز المطلوب وسيُسمح للعملية الأولى في الطابور بالحصول عليه ومتابعة التنفيذ.

توفير حجم قطاع منتظم ومستقل عن الأجهزة: اختلاف وتنوع الأقراص قد يؤدي بالطبيعة إلى اختلاف أحجام القطاعات، بالتالي الأمر متروك للبرمجيات المستقلة لأجهزة الإدخال، والإخراج لإخفاء هذه الحقيقة، وتوفير حجم قطاع موحد للطبقات العليا. قد يحدث هذا عن طريق معاملة قطاعات عديدة كقطاع منطقي واحد، وعلى الطبقات العليا التعامل فقط مع جميع الأجهزة المجردة التي تستخدم حجم القطاع المنطقي نفسه، بغض النظر عن الحجم الحقيقي للقطاع. بالمثل، قد تُوفر بعض من الأجهزة الحرفية بياناتها بمقدار خانة ثمانية واحدة في أي لحظة (مثل أجهزة المودم)، في حين أنه قد تُوفر أجهزة أخرى بياناتها في صورة وحدات أكبر

(مثل واجهات الشبكة)، بالتالي يتمثل دور البرمجيات المستقلة لأجهزة الإدخال، والإخراج في اخفاء هذه الاختلافات.

4.7.3 طبقة برمجيات الإدخال، والإخراج لفضاء المستخدم

تتضمن طبقة برمجيات الإدخال، والإخراج لفضاء المستخدم المكتبات التي تُوفر واجهات مبسطة للوصول إلى وظائف النواة أو للتفاعل مع مشغلات الأجهزة. تتمثل هذه المكتبات في كافة الإجراءات المستخدمة في تحديد شكل المدخلات، والمخرجات التي تُستدعى بواسطة برامج المستخدمين مثل الاستدعاء **putchar**، و **getchar**، و **printf**، و **scanf** وهي مثال على مكتبة **stdio** الخاصة بالإدخال، والإخراج على مستوى المستخدم والمتاحة في لغة البرمجة **C** والتي تُربط داخل البرنامج الرئيسي.

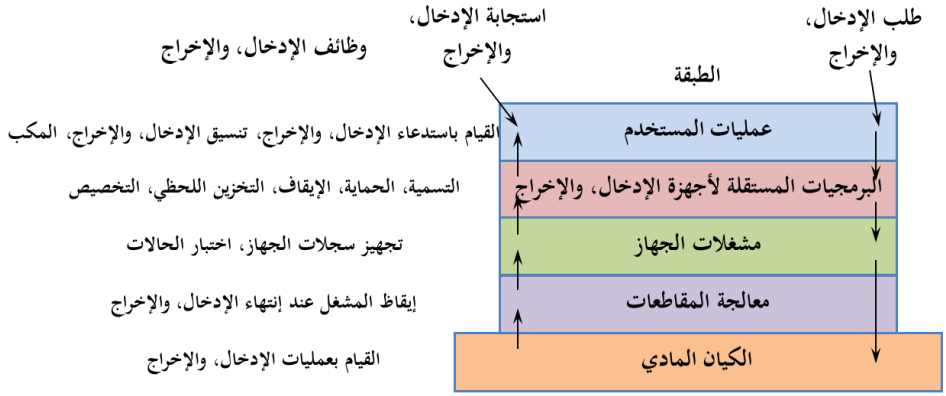
إضافةً إلى إجراءات المكتبة هناك فئة أخرى مهمة تُمثل التخزين المؤقت، وتُوفر للنظام وسيلة للتعامل مع أجهزة الإدخال، والإخراج المخصصة في نظم البرمجة المتعددة. فبالنظر إلى الطابعة، وهي إحدى أجهزة المكب النموذجية، نجد أنه على الرغم من سهولة فتح الملف الحرفي الخاص بالطابعة من قبل أي عملية مستخدم، إلَّا إنَّه من الممكن أن تمنع عملية– ما– عمليات أخرى من فتح هذا الملف وهو ما يعني منع العمليات الأخرى من استخدام الطابعة، السبب في ذلك يرجع إلى أن العملية الفاتحة للملف الحرفي قد تبقى مدة ساعات دون أن تسحب أي شيء بسبب الاستخدام الحصري والذي تمت مناقشته في الفصل الثاني.

وكحل لهذه المشكلة أُنشئت عملية خاصة– تسمى بالعملية المخفية–، ودليل خاص– يسمى دليل المكب– لسحب أي ملف، تضع العملية الراغبة في سحب الملف اسمه في دليل المكب، ومن ثم يُترك الأمر إلى العملية المخفية المخولة الوحيدة باستخدام الملف الحرفي الخاص بالطابعة لسحب الملفات المتواجدة في الدليل. من خلال حماية الملف الخاص من الاستخدام المباشر من قبل المستخدمين، سيتم التخلص من المشكلة سالفة الذكر.

فكرة استخدام المكب ليست محصورة في الطابعات، بل قابلة للاستخدام في حالات أخرى للإدخال والإخراج والمتمثل في نقل ملفات عبر شبكة– ما–، فلإرسال ملف إلى أي مكان، يضع المستخدم هذا الملف في دليل مكب الشبكة، ومن ثم تأخذ الشبكة المخفية في

وقت لاحق هذا الملف وتُرسله.

أخيرًا، يُلخص الشكل 3. 22 نظام الإدخال، والإخراج الذي تظهر فيه جميع الطبقات والوظائف الرئيسية لكل طبقة، بدءًا من القاع، هذه الطبقات هي الكيان المادي، ومعالجة المقاطعات، ومشغلات الجهاز، والبرمجيات المستقلة لأجهزة الإدخال، والإخراج، وأخيرًا عمليات المستخدم.



الشكل 3. 22: طبقات برمجيات الإدخال، والإخراج مع أهم الوظائف الخاصة بكل طبقة.

تُوضح الأسهم في الشكل 3. 22 انسياب التحكم الخاص بالإدخال، والإخراج، عندما يحاول برنامج المستخدم مثلًا، قراءة مقطع من ملف، يُحفِّز نظام التشغيل لتنفيذ استدعاء القراءة. تبحث البرمجيات المستقلة للأجهزة أولًا عن هذا المقطع في ذاكرة التخزين اللحظي، إذا لم يتوفر هناك، فسيُستدعى مشغل الجهاز لإصدار طلب إلى الكيان المادي لجلب المقطع من القرص، ومن ثم تُوَقَّف العملية إلى أن تنتهي عملية القرص. بعد الإنتهاء من جلب المقطع من القرص، يُولد الكيان المادي إشارة مقاطعة، عندها تشتغل طبقة معالجة المقاطعات لاكتشاف الحدث والتعرف على الجهاز الذي طلب الخدمة في الوقت الحالي، ومن ثم تستخلص حالته وتُوقظ العملية النائمة لاستكمال طلب الإدخال، والإخراج، والسماح لعملية المستخدم بمواصلة مهمتها.

8.3 الناقلات الذكية

تَدعم الأنظمة الحديثة غالبًا الناقلات الذكية مثل الناقل التسلسلي العام أو IEEE1394، والتي تتمتع بمستوى عالٍ من الذكاء وتُعرف بواجهات البرامج الموجَّهة،

كجزء من هذه الواجهات، تُوضع كافة الوظائف الخاصة بالجهاز في وحدة تحكم الجهاز، بالتالي لا توجد حاجة إلى برامج تشغيل خاصة بالجهاز في نظام التشغيل. ونظرًا لأن هذه الناقلات تَدعم عدة أجهزة في وقت واحد، فإنها تستخدم بروتوكولات اتصال بين وحدة التحكم المركزية ومتحكم الجهاز.

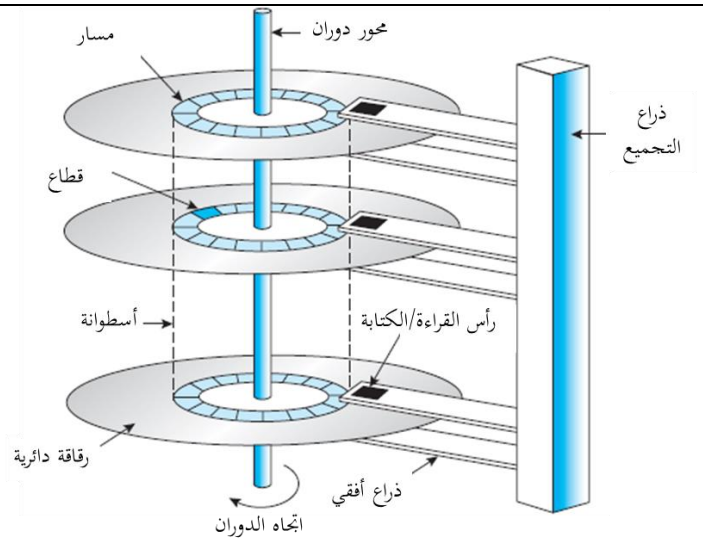
9.3 الأقراص

في هذا القسم ستُدْرس بعض من أجهزة الإدخال، والإخراج الحقيقية، وذلك من خلال تناول أحد الأمثلة، وهي الأقراص التي تُعتبر من الناحية النظرية بسيطة، ولكنها في الواقع من الأجهزة المهمة جدًا في عالم الحواسيب.

1.9.3 الكيان المادي للقرص

تأتي الأقراص في مجموعات متنوعة، أكثرها شيوعًا الأقراص الممغنطة (الأقراص الصلبة، والأقراص المرنة)، فهي تتميز بحقيقة أن عمليتي القراءة والكتابة تُنفذان بنفس السرعة على حد سواء، مما يجعلها مثالية للاستخدام كوسائط تخزين ثانوية. تُوفر الأقراص الممغنطة الجزء الأكبر من التخزين الثانوي لأنظمة الحواسيب الحديثة وقد تُستخدم صفائف منها في بعض الأحيان لتوفير وحدات تخزين ذات موثوقية عالية. ونتيجة لتنوع البرامج، والبيانات، والأفلام، هناك أنواع مختلفة ومهمة من هذه الأقراص. في الأقسام التالية سيُعرج على الأقراص الممغنطة من الناحية المادية والمعنوية كمثال توضيحي لهذه الأقراص.

الأقراص الممغنطة: تظهر البنية الفيزيائية لمحرك الأقراص النموذجي في الشكل 3. 23، حيث يحتوي كل قرص على رقاقة دائرية واحدة أو أكثر مثبتة من مركزها على محور دوران يعمل على تدوير كل الأقراص بنفس السرعة. يتراوح قطر الرقاقات من 1.8 إلى 3.5 بوصة ويُغطى سطحي الرقاقة بمواد قابلة للمغنطة، بحيث تُخزن المعلومات عن طريق تسجيلها مغناطيسيًا عليها. بالمقابل تُثبت رؤوس القراءة والكتابة على أذرع أفقية تمتد على كل من السطحين العلوي والسفلي لكل رقاقة دائرية، تتحرك هذه الأذرع بسرعة كبيرة ذهابًا وإيابًا بين مركز الرقاقات وحافتها الخارجية، الأمر الذي سينتج عنه إمكانية وصول رؤوس القراءة والكتابة إلى أي نقطة على سطح الرقاقات.

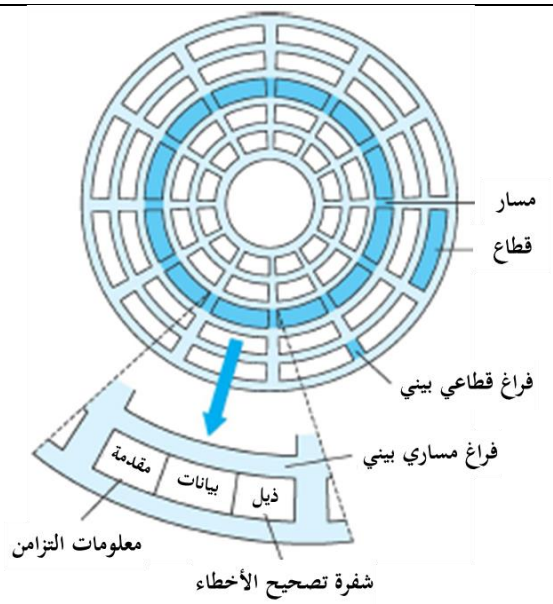


الشكل 3. 23: حركة رقاقات القرص الصلب [Silberschatz et al., 2018].

ينقسم سطح الرقاقة منطقيًا إلى عدة مسارات دائرية تعتمد على عدد رؤس القراءة والكتابة المصفوفة عموديًا، كما تُشكل مجموعة المسارات الموجودة في موضع ذراع واحد أسطوانة وقد يكون هناك الألوف من الأسطوانات المتمركزة في محرك الأقراص. من ناحية أخرى تنقسم المسارات إلى عدة قطاعات وهي أصغر كمية من البيانات التي يمكن قراءتها أو كتابتها فعليًا ويتراوح عددها ما بين 8 إلى 32 في الأقراص المرنة، وتصل إلى عدة مئات في الأقراص الصلبة، علمًا بأن كل القطاعات تحتوي على نفس العدد من الخانات الثمانية وأنه تُقاس سعة تخزين محركات الأقراص الشائعة بالقيقا خانة ثمانية. بالمقابل يتراوح عدد رؤوس القراءة والكتابة من رأس واحد إلى حوالي 16 رأسًا. يوضح الشكل 3. 24 نموذج لتهيئة مسارات وقطاعات القرص هذه.

لقراءة البيانات أو كتابتها، يجب أن يُحرك جهاز القرص الذراع إلى المسار الصحيح، يُسمى الزمن اللازم لتنفيذ ذلك بزمن البحث، بعدها يجب أن ينتظر جهاز القرص تدوير البيانات المطلوبة في موضع يكون أسفل الرأس، يُسمى زمن تنفيذ ذلك بزمن الاستجابة الدوراني. بالتالي يمكن حساب زمن الوصول إلى القرص على النحو التالي:

زمن الوصول إلى القرص = زمن البحث + زمن الاستجابة الدوراني



الشكل 3. 24: تهيئة قطاعات القرص [Garrido et al., 2011].

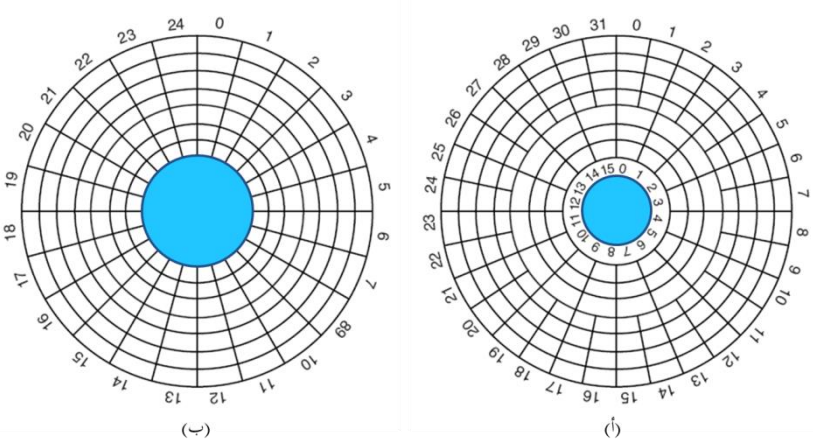
يستغرق زمن البحث على محرك الأقراص الحديث النموذجي حوالي 8 ملّي ثانية، وزمن الاستجابة الدوراني حوالي 4 ملّي ثانية. بالتالي، يمكن لوحدة المعالجة المركزية أن تُنفذ حوالي 10 إلى 30 مليون تعليمة في الزمن الذي تستغرقه في الوصول الفعلي إلى القرص الواحد، هذا الفرق كبير وله تأثير كبير على أداء النظام، سيُناقش هذا لاحقًا في هذا الفصل.

يقوم المتحكم في الأقراص القديمة بأغلب الأعمال الخاصة بالقرص، وتتدفق البيانات فيها على هيئة سلسلة بسيطة من الخانات الثنائية المتتالية، بينما في الأقراص الحديثة يحتوي مشغل القرص نفسه على متحكم دقيق يقوم بجزء كبير من العمل ويسمح للمتحكم الحقيقي بإصدار مجموعة من الأوامر الراقية، أي على المستوى العالي. يُوَصَّل محرك الأقراص بوحدة المعالجة المركزية بواسطة مجموعة من الأسلاك تُسمى ناقلات الإدخال، والإخراج وتتوفر عدة أنواع منها، بما في ذلك الناقل التسلسلي العام، وقناة الألياف، وملحق التكنولوجيا المتقدمة، وغيرها.

من الملاحظ أيضًا في الأقراص القديمة أن عدد القطاعات في المسار نفسه ثابت بالنسبة لجميع الأسطوانات، بينما تنقسم الأقراص الحديثة إلى مناطق، بحيث يزداد عدد القطاعات في

المناطق الخارجية عنه في المناطق الداخلية. يُوضح الشكل 3. 25–أ قرصًا صغيرًا مقسمًا إلى منطقتين، تحتوي مسارات المنطقة الخارجية على 32 قطاعًا، في حين تحتوي مسارات المنطقة الداخلية على 16 قطاعًا. بالنظر إلى أحد الأقراص الحقيقية مثل، WD 18300 نجد أنه مقسم إلى 16 المنطقة أو أكثر، بحيث يزداد عدد القطاعات في المسار بنحو 4٪ للمنطقة الواحدة انطلاقًا من المناطق الداخلية وصولًا إلى المناطق الخارجية.

لإخفاء تفاصيل عدد القطاعات لكل مسار، تمتلك معظم الأقراص الحديثة شكلًا هندسيًا ظاهريًا يُقدم لنظام التشغيل. تُصدر الأوامر إلى البرمجيات للتصرف كما لو أن هناك عدد x من الأسطوانات، و y من رؤوس القراءة والكتابة، و z من القطاعات لكل مسار، بالتالي يقوم المتحكم بمواءمة الطلب الافتراضي (z,y,x) مع أرقام الأسطوانات، والرؤوس، والقطاعات الحقيقية. يُبين الشكل 3. 25–ب الشكل الهندسي الظاهري المحتمل للقرص الحقيقي الموضح في الشكل 3. 25–أ، في كلتا الحالتين، لدى القرص 192 قطاعًا، يتمثل الاختلاف بينها في الترتيب المعلن عن الترتيب الحقيقي.



الشكل 3. 25: أ) الشكل الهندسي الحقيقي لقرص مقسم إلى منطقتين– ب) الشكل الهندسي الظاهري المحتمل لنفس القرص [Tanenbaum & Bos, 2015].

بالنسبة لأجهزة الحواسيب المستندة إلى تقنية بنتيوم، القيم القصوى لهذه المعايير الثلاثة هي في كثير من الأحيان على النحو 65535، 16، و 63، وذلك بسبب الحاجة إلى أن تكون

متوافقة مع القيود المفروضة على حاسوب IBM الأصلي، تتمثل هذه القيود في استخدام القيم 16 و 4 و 6 لتحديد الخانات الثنائية التي تُمثل المعايير الثلاثة، على التوالي مع ترقيم الأسطوانات والقطاعات ابتداءً من 1 وترقيم الرؤوس ابتداءً من 0، اعتمادًا على هذه المعايير، مع جعل حجم كل قطاع 512 خانة ثمانية، فإن حجم أكبر قرص ممكن هو 31.5 قيقا خانة ثمانية. وللالتفاف على هذا الحد، تدعم كافة الأقراص الحديثة نظامًا يُسمى عنونة الكتل المنطقية، التي تُرقم فيه قطاعات القرص بشكل متتالي ابتداءً من الصفر، ودون أيما اعتبار للشكل الهندسي للقرص.

2.9.3 تهيئة القرص

تُعنون الأقراص المغناطيسية الحديثة كمصفوفات كبيرة أحادية البعد للكتل المنطقية والتي تعتبر أصغر وحدة نقل. يكون حجم هذه الكتل في العادة 512 خانة ثمانية، على الرغم من أن بعض الأقراص يمكن تهيئتها بحيث يكون لها حجم كتلة منطقي مختلف مثل، 1024 خانة ثمانية. بعد تصنيع القرص يكون خاليًا من أي معلومات، لذلك قبل أن نتمكن من استخدامه لتخزين البيانات، يجب تقسيمه إلى قطاعات بحيث يُمكن لوحدة التحكم في القرص قراءتها والكتابة فيها، تُعرف عملية التقسيم هذه بعملية التهيئة أو التنسيق وهي تتم في مرحلتين:

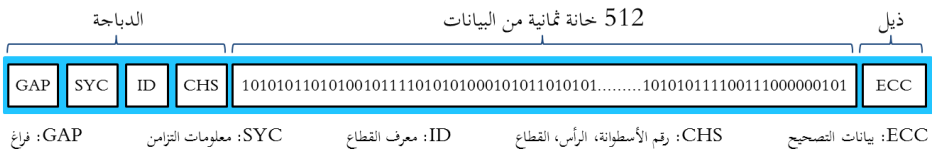
- مرحلة التهيئة ذات المستوى المنخفض.
- مرحلة التهيئة ذات المستوى العالي.

1.2.9.3 مرحلة التهيئة ذات المستوى المنخفض

تُنفذ هذه المرحلة من قبل برامج خاصة وينتج عنها سلسلة من المسارات متحدة المركز، كل منها يحتوي على عدد من القطاعات، مع وجود فجوات قصيرة بين القطاعات. القطاع 0 هو القطاع الأول من المسار الأول على الأسطوانة الخارجية، تستمر عملية التخطيط من خلال هذا المسار، ثم من خلال بقية المسارات في تلك الأسطوانة، ثم من خلال بقية الأسطوانات من الخارج إلى الداخل (أي باتجاه المركز).

ينتج عن هذه المرحلة بنية بيانات خاصة لكل قطاع تَملأ القرص بالكامل وهي تتكون عادة

من مقدمة (دباجة)، ومساحة البيانات الفعلية، وذيل، كما هو موضح في الشكل 3. 26. تبدأ الدباجة الموضحة في هذا الشكل بنمط معين، بحيث يسمح للأجهزة بالتعرف على بداية القطاع، كما أنها تحتوي على أرقام الأسطوانة، والرأس، والقطاع، وبعض من المعلومات الأخرى مثل، بيانات التزامن. يُحدد حجم جزء البيانات عن طريق برنامج التنسيق ذو المستوى المنخفض. بالمقابل يحتوي حقل الذيل شفرة تصحيح الخطأ، وهي معلومات يمكن استخدامها للتعافي من أخطاء القراءة، حجم ومحتويات هذا الحقل تختلف من مُصنع لآخر، وهو ما يتوقف في العادة على مقدار مساحة القرص التي باستطاعة المصمم التخلي عنها من أجل توفير درجة أعلى من الموثوقية، وعلى قدرة المتحكم في التعامل مع الشفرات المعقدة. علاوةً على ذلك، لدى كافة الأقراص الصلبة عدد من القطاعات الاحتياطية لاستخدامها كبديل للقطاعات المعطوبة في أثناء عملية التصنيع.



الشكل 3. 26: قطاع القرص المهيأ.

في أثناء كتابة البيانات في القطاع من قبل المتحكم خلال عمليات الإدخال، والإخراج العادية، تُحدَّث شفرة تصحيح الخطأ بقيمة تُحسب من الخانات الثمانية في مساحة البيانات الفعلية، بالتالي عند قراءة هذا القطاع، يُعاد حساب هذه الشفرة وتُقارن بالقيمة المخزنة. إذا لم يكن هناك تطابق فهذا يشير إلى أنّ مساحة البيانات بالقطاع قد تعرضت للتلف أو أنّ هذا القطاع قد يكون معطوبًا، في حالة تلف عدد قليل فقط من البيانات فإنه بإمكان المتحكم تحديد الخانات الثنائية التي تغيرت وحساب قيمها الصحيحة ثم الإبلاغ عن خطأ بسيط قابل للاسترداد، يُعالج المتحكم تلقائيًا شفرة تصحيح الخطأ عند كل عملية قراءة أو كتابة للقطاع.

بالنسبة لمعظم الأقراص الصلبة، تُنجَز مرحلة التهيئة ذات المستوى المنخفض في المصنع كجزء من عملية التصنيع، وذلك حتى تتمكن الشركة المصنعة من اختبار القرص ومواءمة أرقام الكتل المنطقية مع القطاعات الخالية من العيوب عليه. إضافة إلى ذلك يُمكن إخطار متحكم القرص في أثناء هذه التهيئة بعدد الخانات الثمانية التي يمكن تركها ما بين الدباجة والذيل لكل

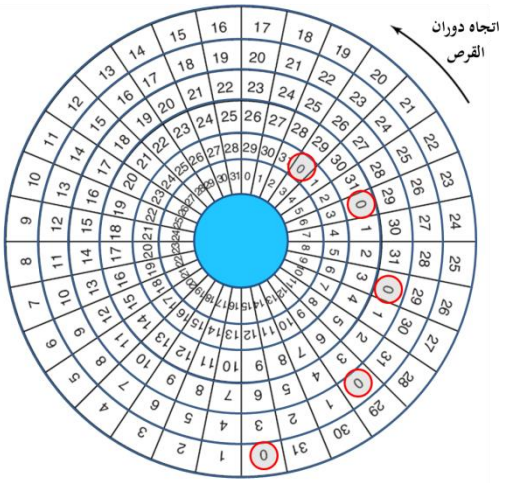
القطاعات، هذا العدد من الممكن اختياره من بين عدة أحجام، مثل 256 و 512 و 1024 خانة ثمانية، علمًا بأن تنسيق القرص باستخدام حجم قطاع كبير يعني احتواء عدد أقل من القطاعات لكل مسار، ولكنه يعني أيضًا كتابة عدد أقل من الدباجة والذيل على كل مسار وتوفير مساحة إضافية لبيانات المستخدم.

هنا يجب الإشارة إلى أنه في أثناء عملية التهيئة يُوضع القطاع 0 في كل مسار على مسافة ثابتة من المسار السابق تُعرف بانحراف الأسطوانة، وذلك لغرض تحسين الأداء. تتمثل الفكرة هنا في السماح للقرص بقراءة عدة مسارات في عملية واحدة مستمرة دون فقدان البيانات. للتعرف على طبيعة هذه المشكلة، يمكن النظر في الشكل 3. 25–أ. لنفترض أن هناك طلب يحتاج لقراءة ثمانية عشر قطاعًا تبدأ من القطاع رقم 0 في المسار الداخلي وتنتهي بالقطاع رقم 1 في المسار التالي، بالتالي فإنَّ قراءة القطاعات الستة عشر الأولى تأخذ دورة واحدة للقرص، ولكن هناك حاجة للتحرك نحو الخارج مسار واحد، للحصول على القطاع السابع عشر (القطاع رقم 0 في المسار التالي)، إلَّا إنَّه وبمرور الوقت اللازم لنقل رأس القراءة والكتابة مسار واحد، نجد أن القطاع 0 قد عبر الرأس، لذلك هناك حاجة لدورة كاملة لكي يأتي هذا القطاع مرة أخرى تحت الرأس. لذلك وللتخلص من هذه المشكلة تُنسق القطاعات بالطريقة الموضحة في الشكل 3. 27.

يعتمد طول مسافة انحراف الأسطوانة على التصميم الهندسي لمشغل الأقراص مثلًا، مشغل ذو 10،000 لفة في الدقيقة يحتاج إلى 6 ملّي ثانية لكي يُنهي دورته، بالتالي إذا كان المسار يحوي 300 قطاع، فإن أي قطاع جديد سيكون تحت رأس القراءة، والكتابة كل 20 ميكرو ثانية، وبناءً عليه إذا كان زمن الانتقال من مسار إلى آخر يأخذ 800 ميكرو ثانية، وهو ما يعني أن 40 قطاعًا مر في أثناء هذا الزمن، لذلك يجب أن تكون مسافة انحراف الأسطوانة 40 قطاعًا، بدلًا من القطاعات الثلاثة المبينة في الشكل 3. 27. من الجدير بالذكر أن التبديل بين رؤوس القراءة، والكتابة أيضًا يستغرق وقتا محددًا وهو ما يمثل انحراف الرأس، إلَّا إنَّ هذا الانحراف ليس بالكبير جدًا، بالتالي غالبًا ما يُهمل.

من خلال ما سبق ذكره ونتيجة لعملية التهيئة فإن سعة القرص تقل، وذلك اعتمادًا على أحجام الدباجة، والفجوة بين القطاعات، وكذلك حجم شفرة تصحيح الخطأ، فضلًا عن عدد

القطاعات الاحتياطية المحجوزة. في كثير من الأحيان السعة المهيئة للقرص تقل بنسبة 20٪ عن السعة غير المنسقة، علمًا بأن القطاعات الاحتياطية لا تُحسب من ضمن السعة المهيئة.



الشكل 3. 27: توضيح انحراف الأسطوانة [Tanenbaum & Bos, 2015].

تؤثر تهيئة القرص أيضًا على أداء النظام، ولتوضيح ذلك نفترض أن هناك قرصًا ذا 10،000 لفة في الدقيقة، وله مسار يحتوي على 300 قطاع، وكل قطاع يحوي 512 خانة ثمانية، بالتالي فإنه يحتاج إلى 6 ملِّي ثانية لقراءة 153،600 خانة ثمانية موجوة على المسار بمعدل 25،600،000 خانة ثمانية لكل ثانية أو 24.4 ميجا خانة ثمانية لكل ثانية، هذا الأمر يجعل النظام غير قادر على مجاراة المعدلات الأسرع من هذا المعدل، بغض النظر عن نوعية الواجهة المتصلة بالجهاز، حتى ولو كانت واجهة نظام الحاسوب الصغير SCSI[11] ذات معدل 80 ميجا خانة ثمانية لكل ثانية أو 160 ميجا لكل ثانية.

تتطلب القراءة بهذا المعدل وبشكل مستمر في الواقع وجود مخزن لحظي كبير الحجم في

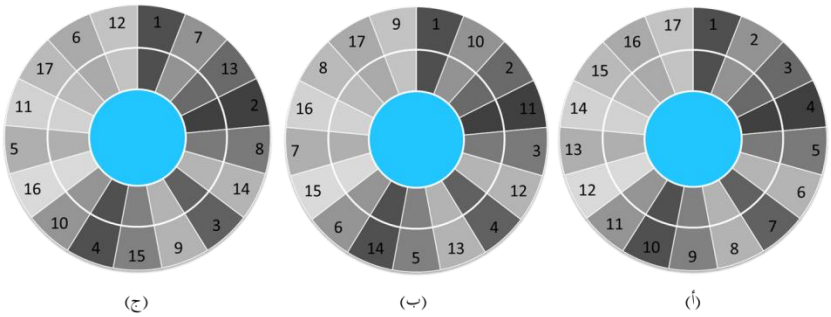
[11] SCSI, Small Computer System Interface

متحكم الجهاز، لكي تنجح عملية القراءة، ولتوضيح ذلك يمكن التمعن في المثال التالي: بفرض أن هناك متحكم جهاز يملك مخزن لحظي يسع قطاعًا واحدًا وأُصدر له أمر لقراءة قطاعين متتاليين، بعد الإنتهاء من قراءة القطاع الأول والقيام بحساب شفرة تصحيح الخطأ، يجب نقل البيانات من المتحكم إلى الذاكرة الرئيسية، في أثناء عملية النقل هذه، لن تتوقف رقاقات القرص أبدًا عن الدوران، وبمجرد الإنتهاء من قراءة القطاع بأكمله فليس أمامها سوى وقت قليل قبل أن تكون بداية القطاع الثاني تحت رأس القراءة، مباشرةً ستبدأ عملية إرسال محتويات هذا القطاع إلى المتحكم لوضعها في المخزن اللحظي. ولأن المتحكم منشغل بنقل البيانات إلى الذاكرة الرئيسية سيؤدي هذا إلى ضياع بيانات القطاع الثاني، بالتالي عندما تنتهي عملية النقل، سيضطر المتحكم إلى الانتظار بمقدار الوقت اللازم لدوران كامل القطاع الثاني والمجيء مرة أخرى تحت رأس القراءة، لكي يَقرأه مرة ثانية.

بالطبع الأمر نفسه سيتكرر مع قراءة القطاع الثالث. كل هذا الانتظار من شأنه أن يؤثر على الأداء مثلًا، إذا كان القرص يحتوي على سبعة عشر قطاعًا لكل مسار، فسيحتاج إلى سبعة عشر ضعفًا مقارنة بقراءة عشر قطاعات في الحالة المثالية.

يمكن القضاء على هذه المشكلة عن طريق ترقيم القطاعات بطريقة التعشيق عند تنسيق القرص في مرحلة التهيئة ذي المستوى المنخفض لغرض إعطاء المتحكم الوقت الكافي لنقل البيانات إلى الذاكرة. نرى في الشكل 3. 28–أ نمط الترقيم المعتاد (أي تجاهل انحراف الأسطوانة هنا) للأقراص الصلبة القديمة ذات 17 قطاعًا لكل مسار، في هذه الحالة سيتم ترقيم القطاعات على النحو 1، 2، 3، ...، 17 وهو ما سيتسبب في ظهور المشكلة الموضحة أعلاه، بدلًا من ذلك يمكن استخدام التعشيق المفرد وذلك كما هو موضح في الشكل 3. 28–ب. مع هذا الترتيب ستُرقم القطاعات كما يلي: 1، 10، 2، 11، 3، 12، 4، 13، 5، 14، 6، 15، 7، 16، 8، 17، 9، هذا يعني أنه في الوقت الذي يجري فيه معالجة القطاع 1 يمر القطاع 10 تحت رأس القراءة، ولذلك عندما يكون جهاز التحكم جاهزًا، فإن القطاع 2 يصل فقط تحت رأس القراءة، وهو ما يعني الاحتياج إلى دورتين من أجل قراءة المسار بأكمله، واحدة للقطاعات الفردية والأخرى للزوجية. إن عملية التهيئة على هذا الأساس تهدف إلى قراءة كافة القطاعات بطريقة متعاقبة مع المحافظة على تحقيق السرعة القصوى في القراءة التي تسمح بها الإمكانيات

المادية.



الشكل 3. 28: أ) بدون تعشيق– ب) التعشيق المفرد– ج) التعشيق المزدوج.

ماذا لو كان جهاز التحكم بطيئًا جدًا حتى مع استخدام التعشيق المفرد؟ إذا كان الأمر كذلك، يمكن استخدام التعشيق المزدوج والموضح في الشكل 3. 6–ج والذي تُرقم فيه القطاعات على النحو: 1، 7، 13، 2، 8، 14، 3، 9، 15، 4، 10، 16، 5، 11، 17، 6، 12. هذا من شأنه أن يُحسن الأداء إلى حد كبير إذا لم تستطع وحدة التحكم التعامل مع التعشيق المفرد. إذا امتلك المتحكم مخزنًا لحظيًا بسعة قطاع واحد، فإنه لا يهم ما إذا كانت عملية النسخ تتم من المخزن اللحظي إلى الذاكرة الرئيسية تحت إدارة المتحكم، أو تحت وحدة المعالجة المركزية الرئيسية، أو عن طريق شريحة الوصول المباشر للذاكرة. إلَّا إنَّه لتجنب الحاجة إلى التعشيق، ينبغي أن يكون المتحكم قادرًا على تخزين مسار كامل وهو ما تقوم به العديد من المتحكمات الحديثة. بعد استكمال تهيئة القرص على المستوى المنخفض، يأتي دور مرحلة التهيئة ذات المستوى العالي.

2.2.9.3 مرحلة التهيئة ذات المستوى العالي

تُنفذ هذه المرحلة في خطوتين: الخطوة الأولى تُدار من قبل برامج خاصة وتحت إشراف مستخدم الحاسوب وتُعرف بعملية التجزئة، فقبل أن يُستخدم القرص لاحتواء الملفات والبيانات، يُقسم إلى مجموعة واحدة أو أكثر من الأقسام. منطقيًا، يُمكن لنظام التشغيل التعامل مع كل قسم كما لو كان قرصًا منفصلًا، الأمر الذي سيسمح بتعايش عدة أنظمة تشغيل مختلفة داخل القرص الواحد مع تخصيص قسم لملفات المستخدم، في بعض الحالات الأخرى يُمكن استخدام هذه

الأقسام في عمليات المبادلة[12].

في معظم أجهزة الحواسيب يحتوي القطاع 0 على سجل الاستنهاض الرئيسي للحاسوب والذي يحتوي بدوره على بعض من شفرات الاستنهاض، بالإضافة إلى جدول الأقسام الذي يوضح القطاعات الخاصة ببداية كل قسم وكذلك حجمه (راجع موضوع استنهاض الحاسوب في الفصل الأول). من الممكن أن يحوي جدول الأقسام مساحة مخصصة لكل قسم. على سبيل المثال، إذا خُصصت أربع مساحات إلى نظام 'ويندوز'، فسوف يُطلق عليها :C، :D، :E، و:F، وسيتُعامل معها على أساس أقراص منفصلة، أمَّا إذا خُصصت ثلاثًا منها فقط لنظام 'ويندوز' وواحدة إلى نظام 'يونيكس'، فسيُطلق نظام 'ويندوز، عليها :C، :D، و:E، بينما سيُطلق على أول سواقة أقراص مدمجة :F. من الجدير بالذكر هنا يجب أن يُعلَّم أحد هذه الأقسام بالقسم النشط في جدول الأقسام، وذلك لكي يتمكن الحاسوب من الاستنهاض منه.

بالمقابل تُنفذ الخطوة الثانية من مرحلة التهيئة ذات المستوى العالي تحت إشراف نظام التشغيل، وتُمثل الخطوة النهائية في إعداد القرص للاستخدام وإنشاء نظام الملفات، فهي تتمثل في إجراء تهيئة رفيعة المستوى لكل قسم من أقسام القرص (كل على حدة)، بحيث تشمل إعداد قطاع الاستنهاض، وإدارة التخزين الخاصة بالمساحات الحرة والمخصصة (القوائم المرتبطة الحرة، أو خرائط الثنائية)، وإعداد الدليل الجذري، ونظام الملفات (سيُناقش هذا الموضوع بنوع من التفصيل في الفصل السادس)، كما تقوم هذه الخطوة أيضًا بوضع شفرة في مدخل جدول الأقسام تُوضح من خلالها نظم الملفات المستخدم في كل قسم، وذلك بسبب وجود العديد من أنظمة التشغيل الداعمة لأنظمة ملفات مختلفة وغير متوافقة. لزيادة الكفاءة، تُجمع معظم أنظمة الملفات القطاعات معًا في مجموعات كبيرة، تُسمى في كثير من الأحيان بالعناقيد.

كما أشرنا سابقًا في الفصل الأول عند تشغيل الحاسوب، يُنفذ في البداية نظام الإدخال،

[12] سيُناقش موضوع المبادلة في الفصل الخامس.

والإخراج الأساسي ثم يُقرأ سجل الاستنهاض الرئيسي للحاسوب ويقفز إليه. يتعرف برنامج الاستنهاض هذا على القسم النشط، ومن ثم يقرأ قطاع الاستنهاض منه ويُنفذه. يحتوي قطاع الاستنهاض على برنامج صغير يقوم بتحميل برنامج يُعرف باسم مُحمل التمهيد، وهو عبارة عن برنامج يتواجد في ذاكرة القراءة فقط، أو ذاكرة القراءة فقط القابلة للمسح والبرمجة، أو أي ذاكرة أخرى غير متطايرة ويبحث في نظام الملفات للعثور على نظام التشغيل، ومن ثم تحميله في الذاكرة وتنفيذه، وبذلك تبدأ حياة الحاسوب. يلي التعرف على آلية تهيئة القرص، يُقدم الفصل طرق جدولة ذراعه.

3.9.3 خوارزميات جدولة ذراع القرص

تتمثل إحدى مسؤوليات نظام التشغيل في استخدام أجهزة الإدخال، والإخراج بكفاءة، لذلك تستلزم تلبية هذه المسؤولية بالنسبة للأقراص المغناطيسية توفير زمن وصول سريع وعرض نطاق كبير للقرص. فالأول له مكونان رئيسيان، كما هو مذكور في القسم 1.9.3:

- زمن البحث: وهو الزمن اللازم لتحريك ذراع القرص إلى الأسطوانة التي تحتوي على القطاع المطلوب.
- زمن الاستجابة الدوراني: وهو الزمن الإضافي للقرص واللازم لتدوير القطاع المطلوب إلى أن يصل رأس القراءة، والكتابة.

أما عرض النطاق الترددي للقرص فهو إجمالي عدد وحدات الخانات الثمانية التي نقلت مقسومًا على إجمالي الزمن بين أول طلب للخدمة وإكمال آخر عملية نقل. بالتالي يُمكن تحسين كل من زمن الوصول وعرض النطاق الترددي عن طريق إدارة الطلب الذي تُعالج فيه طلبات الإدخال، والإخراج بالنسبة للقرص، هذه الطلبات من الممكن أن تأتي من نظام التشغيل، أو من كل من عمليات النظام، أو المستخدم.

إذا حدث طلب خدمة إدخال، أو إخراج وكان محرك القرص المطلوب ومتحكمه متوفرين، فسيُلبى هذا الطلب على الفور، أمَّا إذا كانا مشغولان، فسيُوضع أي طلب جديد للخدمة في قائمة انتظار الطلبات المعلقة الخاصة بمحرك القرص. بالنسبة لأنظمة البرمجة المتعددة ستكون هناك عدة عمليات وقد تحتوي قائمة انتظار الطلبات على عدة طلبات معلقة، وبالتالي عند اكتمال أي

طلب، يختار نظام التشغيل طلب آخر لخدمته. يُناقش هذا القسم الكيفية التي سيختار بها نظام التشغيل أحد هذه الطلبات المعلقة من خلال مناقشة عدد من الخوارزميات الشائعة لجدولة ذراع القرص والمتمثلة في:

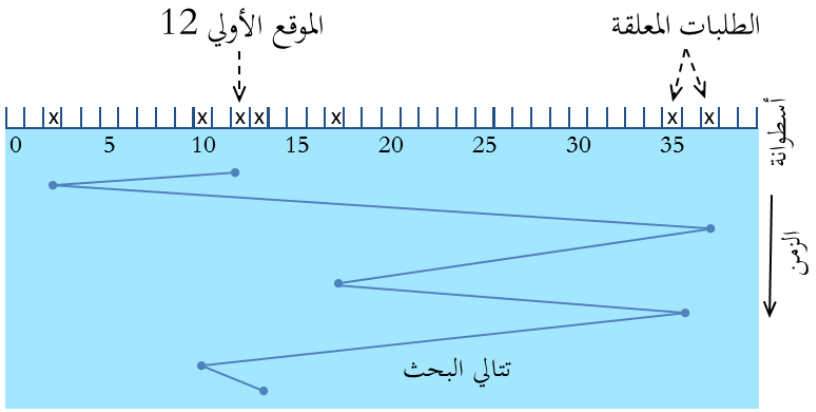
- خوارزمية القادم أولًا يُخدم أولًا.
- خوارزمية الأقصر زمن بحث أولًا.
- خوارزمية المسح.
- خوارزمية المسح الدائري.
- خوارزمية الجدولة الناظرة.

1.3.9.3 خوارزمية القادم أولًا يُخدم أولًا

تُعتبر هذه الخوارزمية من أبسط خوارزميات جدولة الأقراص وهي تعمل بآلية من يأتي أولًا يُخدم أولًا. أي تُعالج الطلبات بنفس الترتيب الذي تصل به إلى قائمة انتظار الطلبات المعلقة. من مزايا هذه الخوارزمية هو أنَّ كل طلب سيحصل على فرصة عادلة، وأنه لن يُؤجل أي طلب إلى أجل غير مسمى، إلَّا إنَّها لا تُساهم بشكل كبير في تحسين زمن البحث، ولا تُقدم أفضل خدمة ممكنة كما سنلاحظ من خلال الأمثلة اللاحقة. مع ذلك، يمكن الاستفادة منها عندما يكون القرص مُحملًا بشكل مكثف، وذلك عن طريق السماح بإنشاء طلبات قرص آخر من قبل عمليات أخرى في أثناء بحث ذراع القرص عن أحد الطلبات. وللقيام بذلك، تحتفظ العديد من مشغلات الأقراص بجدول مفهرس بأرقام الأسطوانات، بحيث تُربط كافة الطلبات المعلقة لكل أسطوانة معًا في قائمة مرتبطة تُعنون بمدخلها في الجدول.

إن استخدام القوائم المرتبطة يُساعد في تحسين أداء هذه الخوارزمية وذلك كما هو موضح في المثال التالي. بفرض أن هناك قرصًا يحتوي على 40 أسطوانة، وكانت هناك طلبية بخصوص قراءة مقطع موجود في الأسطوانة رقم 12. في خلال فترة البحث داخل هذه الأسطوانة، استقبل مشغل القرص طلبات جديدة تخص الأسطوانات 2، 37، 17، 35، 10، وكذلك 13 بنفس هذا الترتيب، والتي ستَدخل في جدول الطلبات المعلقة، مع تحديد قائمة مرتبطة منفصلة لكل أسطوانة، بالتالي سيقوم مشغل القرص بتلبية هذه الطلبات على النحو الموضح في الشكل 3.

.29



الشكل 3. 29: مثال توضيحي لخوارزمية القادم أولًا يُخدم أولًا لجدولة ذراع القرص.

عند الإنتهاء من الطلب الحالي (أي الأسطوانة 12)، سيكون لدى مشغل القرص الخيار لتلبية الطلب التالي، تبعًا لهذه الخوارزمية سيكون التنفيذ موجهًا إلى الأسطوانة 2 ثم 37 يليها 17، 35، 10، وأخيرًا الأسطوانة 13، بالتالي فإن مجموع حركة ذراع رأس القراءة يمكن حسابه من العلاقة:

- مجموع حركة ذراع رأس القراءة = 10 + 35 + 20 + 18 + 25 + 3 = 111 أسطوانة

على افتراض أن زمن الانتقال من أسطوانة إلى أخرى هو 5 ملّي ثانية، فإن زمن البحث يمكن حسابه من الصيغة:

- زمن البحث = مجموع حركة ذراع رأس القراءة * زمن الانتقال من أسطوانة إلى أخرى

= 111 * 5 = 555 ملّي ثانية

2.3.9.3 خوارزمية الأقصر زمن بحث أولًا

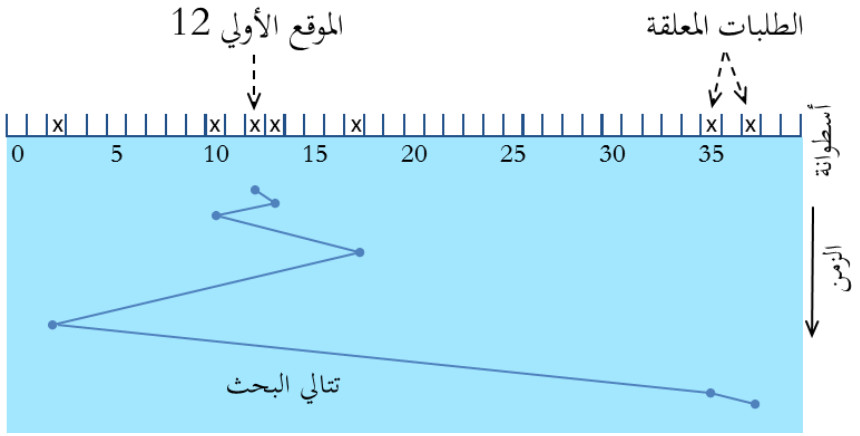
لخفض قيمة زمن البحث في الخوارزمية السابقة هناك العديد من المحاولات أو الطرق التي يمكن أن يستخدمها مشغل القرص، من هذه الخوارزميات ما يُعرف باسم خوارزمية الأقصر زمن بحث أولًا، وهي تعني البدء بأقصر زمن بحث. لذلك، يتم حساب زمن كل طلب مقدمًا في طابور

الانتظار لجدولته وفقًا لزمن البحث المحسوب، ونتيجة لذلك، سيُنفذ الطلب الذي بالقرب من ذراع القرص أولاً، وذلك على النحو الموضح في الشكل 3. 30. هذه الخوارزمية، وغيرها تتطلب حفظ كل الطلبات القادمة إلى مشغل القرص داخل جدول معنون بأرقام الأسطوانات حتى يُمكن تتبع هذه الطلبات.

من مزايا هذه الخوارزمية هو انخفاض متوسط زمن الاستجابة وزيادة الإنتاجية، إلَّا إنَّها تُعاني من النفقات العامة لحساب زمن البحث مقدمًا، كما أن هناك تباين كبير في زمن الاستجابة نتيجة لتفضيلها بعض الطلبات.

بتطبيق آلية تنفيذ هذه الخوارزمية على البيانات المعطاة في المثال السابق سيكون التنفيذ موجهًا إلى الأسطوانة 13 ثم 10 يليها 17، 2، 35، وأخيرًا الأسطوانة 37. بالتالي فإن مجموع حركة ذراع رأس القراءة يمكن حسابه على النحو التالي:

- مجموع حركة ذراع رأس القراءة = 1 + 3 + 7 + 15 + 33 + 2 = 61 أسطوانة



الشكل 3. 30: مثال توضيحي لخوارزمية الأقصر زمن بحث أولًا لجدولة ذراع القرص.

على افتراض أن زمن الانتقال من أسطوانة إلى أخرى هو 5 ملِّي ثانية، فإن زمن البحث يمكن حسابه من الصيغة:

- زمن البحث = مجموع حركة ذراع رأس القراءة * زمن الانتقال من أسطوانة إلى أخرى

= 61 * 5 = 305 ملّي ثانية

وهو ما يُقلل حركة الذراع إلى النصف مقارنة بخوارزمية القادم أولًا يُخدم أولًا. هذا لا يعني بأن هذه الخوارزمية هي الحل الأمثل، وذلك بسبب وجود المشكلة التالية:

بفرض أن هناك طلبات جديدة وصلت إلى مشغل القرص في أثناء معالجته للطلبات الأولى، وبفرض أن هناك طلبية وصلت بعد الإنتهاء من الأسطوانة 17 وكان هذا الطلب يخص الأسطوانة 9، بالتالي فإن رأس القراءة سيتجه إلى الأسطوانة 9 بدلًا من الذهاب إلى الأسطوانة 2، بعد الإنتهاء من 9، جاءت طلبية أخرى تخص الأسطوانة 14 هذا سيجعل التنفيذ دائمًا محصورًا في الوسط ما بين الأرقام المتقاربة ويجعل فرصة تنفيذ الأرقام البعيدة ضئيلة وهو ما سيؤثر سلبًا على زمن الاستجابة لكل طلب وتحقيق العدالة ما بين الطلبات.

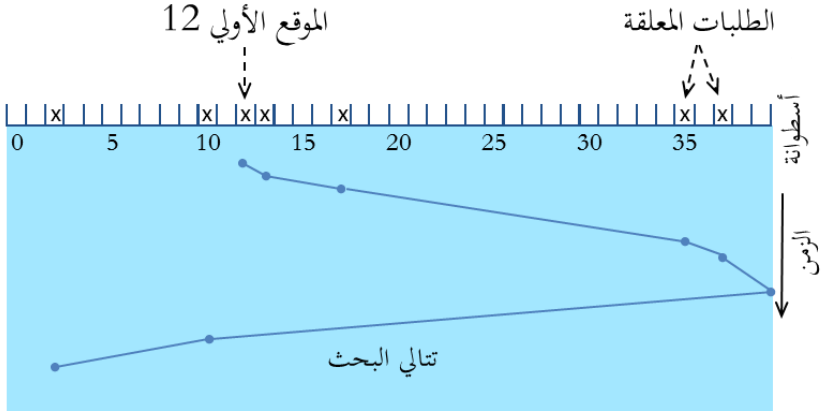
3.3.9.3 خوارزمية المسح

تُعْرَف هذه الخوارزمية أيضًا باسم خوارزمية المصعد وذلك للتشابه الكبير بين آلية تنفيذها وآلية استخدام المصعد. الطلبات الخاصة بالمصعد تأتي في البنايات باستمرار من الطوابق (الأسطوانات) وبشكل عشوائي، والحاسوب المشغل للمصعد قادر على تتبع تسلسل الضغط على زر المصعد من قبل الزبائن وخدمة هذه الطلبات باستخدام أيٍ من الخوارزميتين السابقتين، إلَّا إنَّ استخدامهما في هذه الحالة قد لا يُحقق مبدأَيْ الكفاءة والعدالة.

للتوفيق بين الطلبات المتضاربة مع بعضها ومحاولة تحقيق أفضل كفاءة وعدالة بالنسبة للطلبات الواردة للمشغل، يُمكن استخدام خوارزمية المسح، في هذه الخوارزمية يتحرك ذراع القرص في اتجاه معين ويُلبي كافة الطلبات المعلقة في هذا الاتجاه وصولًا إلى نهاية القرص، ومن ثم يعكس الاتجاه ليُلبي أيضًا كافة الطلبات المعلقة في الاتجاه الجديد ووصولًا إلى الحد الأقصى له. يُبين الشكل 3. 31 طريقة عمل خوارزمية المسح باستخدام نفس الطلبات الموضحة سابقًا، يفترض هذا الشكل أن اتجاه الحركة إلى أعلى.

تتطلب خوارزمية المسح وجود خانة ثنائية لتتبع اتجاه حركة الرأس إلى أعلى، أو إلى أسفل. عندما تأتي الطلبية يختبر مشغل القرص هذه الخانة، فإذا كانت الحركة إلى أعلى فسيُنفذ في هذا الاتجاه إلى أن يصل إلى نهاية القرص، أمَّا إذا لم تكن هناك طلبات في هذا الاتجاه فسيعكس

الاتجاه إلى أسفل عن طريق تغيير قيمة الخانة وسيُنفذ الطلبات المعلقة في الاتجاه العكسي إن وجدت. تجدر الإشارة هنا أنه إذا وصل طلب جديد إلى قائمة الانتظار وكان موقعه أمام الرأس مباشرةً، فسيخدمه على الفور، وسيتعين على الطلب الجديد الذي يصل إلى القائمة خلف موقع الرأس الانتظار حتى يتحرك الذراع إلى نهاية القرص ويعكس اتجاهه ويعود لخدمته في الاتجاه الجديد.



الشكل 3. 31: مثال توضيحي لخوارزمية المسح لجدولة ذراع القرص.

من مزايا هذه الخوارزمية هو أنها ذات إنتاجية عالية وتباين منخفض في متوسط زمن الاستجابة، إلَّا إنَّ الطلبات التي تصل خلف ذراع القرص ستعاني من الانتظار لفترة أطول.

باستخدام نفس طلبات مثال خوارزمية القادم أولًا يُخدم أولًا، سيكون التنفيذ على النحو 13، 17، 35، 37، بعدها يصل إلى الحد الأقصى للإسطوانات (39) ومن ثم يعكس الاتجاه ليقوم بتلبية الطلب 10، وأخيرًا 2. بالتالي فإن مجموع حركة ذراع رأس القراءة يمكن حسابه على النحو التالي:

- مجموع حركة ذراع رأس القراءة = 1 + 4 + 18 + 2 + 2 + 29 + 8 = 64 أسطوانة

على افتراض أن زمن الانتقال من أسطوانة إلى أخرى هو 5 ملِّي ثانية، فإن زمن البحث يمكن حسابه من الصيغة:

- زمن البحث = مجموع حركة ذراع رأس القراءة * زمن الانتقال من أسطوانة إلى أخرى

= 64 * 5 = 320 ملّي ثانية

وهو ما قَلل حركة الذراع تقريبًا إلى النصف مقارنةً بخوارزمية القادم أولًا يُخدم أولًا. هناك خاصية واحدة لطيفة تتمتع بها خوارزمية المسح، وهي أن الحد الأعلى للحركة الإجمالية لأي مجموعة من الطلبات ثابت ويساوي ضعف عدد الأسطوانات.

من المهم أن نُدرك أن كل خوارزميات جدولة القرص السابقة تفترض ضمنيًا أن هندسة القرص الحقيقي هي نفسها الهندسة الافتراضية، إذا لم تكن هذه الفرضية موجودة، فإن جدولة طلبات القرص تكون لا معنى لها، لأن نظام التشغيل لا يمكن أن يقرر حقًا ما إذا كانت الأسطوانة 40 أو 200 أقرب إلى الأسطوانة 39.

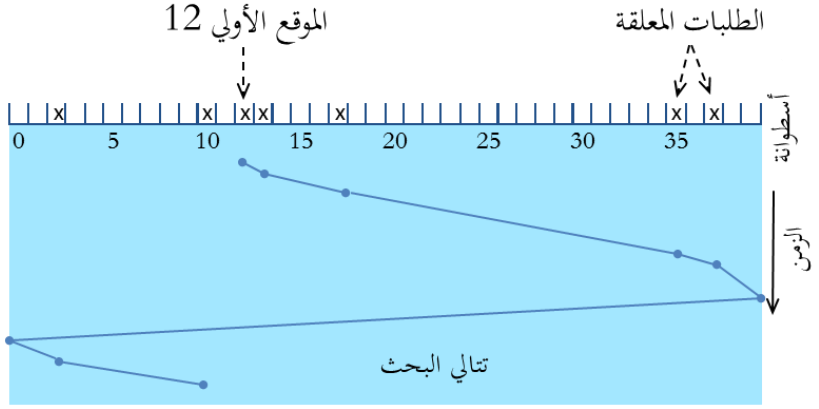
4.3.9.3 خوارزمية المسح الدائري

هناك تعديل طفيف من الممكن أن يطرأ على خوارزمية المسح للحالات التي لديها تباين صغير في أوقات الاستجابة. نتج عن هذا التعديل خوارزمية المسح الدائري والتي تتشابه مع خوارزمية المسح، باستثناء أن الذراع لا يتوقف لكي يتعامل مع الطلبات في أثناء التحرك في الاتجاه العكسي. بدلاً من ذلك، يتحرك الذراع بالكامل في اتجاه الطرف الآخر من القرص إلى الحد النهائي له ويبدأ في خدمة الطلبات في اتجاه الذهاب من جديد. السبب في هذا يرجع إلى أنه من الممكن أن يكون هناك عدد كبير من الطلبات ينتظر في الطرف الآخر أو قد لا يتواجد أي طلب أو يكون هناك عدد قليل من الطلبات في المسار الذي تم مسحه مؤخرًا. من مزايا هذه الخوارزمية توفير زمن انتظار منتظم مقارنة بخوارزمية المسح.

يبين الشكل 3. 32 طريقة عمل خوارزمية المسح الدائري باستخدام نفس الطلبات الموضحة سابقًا، يفترض هذا الشكل أن اتجاه الحركة إلى أعلى، بالتالي سيكون التنفيذ على النحو 13، 17، 35، 37، بعدها يصل إلى الحد النهائي للإسطوانات (39) ومن ثم يعكس الاتجاه ليصل إلى الإسطوانة 0 ويقوم بتلبية الطلب 2، وأخيرًا 10.

هذا سيجعل حركة الذراع على النحو 1، 4، 18، 2، 2، 40، 1 و 8. بالتالي فإن مجموع حركة ذراع رأس القراءة يمكن حسابه على النحو التالي:

- مجموع حركة ذراع رأس القراءة = 1 + 4 + 18 + 2 + 4 + 39 + 2 + 8 = 78 أسطوانة



الشكل 3. 32: مثال توضيحي لخوارزمية المسح الدائري لجدولة ذراع القرص.

على افتراض أن زمن الانتقال من أسطوانة إلى أخرى هو 5 ملّي ثانية، فإن زمن البحث يمكن حسابه من الصيغة:

- زمن البحث = مجموع حركة ذراع رأس القراءة * زمن الانتقال من أسطوانة إلى أخرى

= 78 * 5 = 390 ملّي ثانية

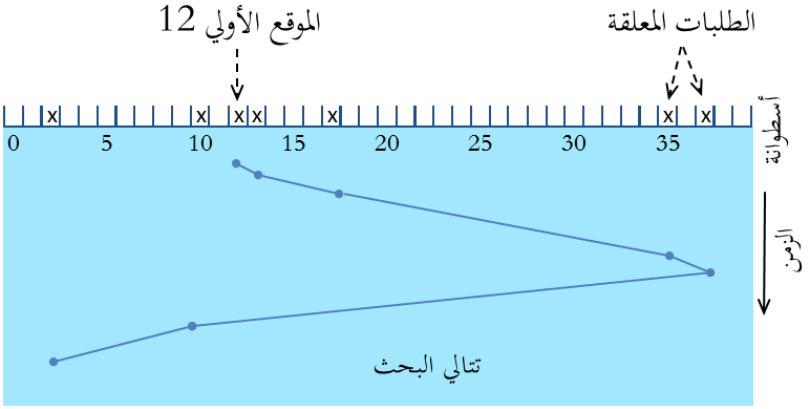
بناءً على معطيات المثال الموضح في الشكل 3. 32 يُمكن ملاحظة أن المسح الدائري لن يكون بنفس كفاءة خوارزمية الأقصر بحث أولًا أو خوارزمية المسح.

5.3.9.3 خوارزمية الجدولة الناظرة

ذُكر سابقًا أن كل من خوارزمية المسح، والمسح الدائري تُحركا ذراع القرص عبر العرض الكامل له. من الناحية العملية، في الغالب لا يتم تنفيذ هاتان الخوارزميتان بهذه الطريقة. الأكثر شيوعًا، هو جعل الذراع يتحرك فقط بقدر الطلب النهائي في كل اتجاه ومن ثم يعكس الذراع الاتجاه على الفور، دون الذهاب إلى نهاية القرص. يُسمى هذا الإصدار من الخوارزميات بخوارزمية الجدولة الناظرة، لأنها تنظر إلى الطلب قبل الاستمرار في التحرك في اتجاه معين.

يُوضح الشكل 3. 33 طريقة عمل خوارزمية الجدولة الناظرة باستخدام نفس الطلبات الموضحة سابقًا، يفترض هذا الشكل أن اتجاه الحركة إلى أعلى، بالتالي سيكون التنفيذ على النحو 13، 17، 35، 37، بعدها يعكس الاتجاه مباشرةً ليُلبي الطلب 10، وأخيرًا 2. هذا سيجعل حركة الذراع على النحو 1، 4، 18، 2، 27 و 8 وبمجموع قدره 60 أسطوانة وبزمن بحث قدره 300 ملّي ثانية.

بعد الاطلاع على بعض من خوارزميات جدولة ذراع القرص، يمكن القول بأن لكل خورزمية طريقتها في خدمة طلبات الإدخال، والإخراج وأنّ الأداء الإجمالي لكل منها يعتمد على عدد ومواقع الطلبات داخل القرص.



الشكل 3. 33: الجدولة الناظرة.

4.9.3 تحسين التأخير الدوراني

نظرًا لأن التأخير الدوراني قد يستغرق وقتًا طويلًا عند تحريك ذراع القرص، فقد يبدو من الطبيعي أن يُحسن ذلك بوضع هدفين في الاعتبار:

1. وضع الطلبات لنفس المسار معًا في طابور الطلبات.
2. تُطلب هذه الطلبات حسب موضع الدوران للقطاع المطلوب نسبة إلى الموضع الحالي للقرص.

لسوء الحظ أن العديد من وحدات تحكم القرص الحديثة لا تُعْلِم نظام التشغيل بموضع

الدوران الحالي، لذلك لا يمكن لنظام التشغيل تنفيذ الجزء الثاني من هذا التحسين. ومع ذلك، تستقبل وحدات تحكم القرص هذه عادةً عدة طلبات متزامنة وتستخدم ذاكرة التخزين المؤقت الكبيرة الخاصة بها للتعامل معها، وبالتالي تُحقق نفس التأثير.

5.9.3 تفاعل جدولة القرص ووظائف النظام الأخرى

من خلال ما سبق ذكره يُمكن ملاحظة تأثر أداء القرص بشكل كبير بمدى حركة الذراع. بالتالي، من السهل مشاهدة أنه إذا كانت الملفات قريبة من بعضها البعض على القرص، سيكون هناك حركة ذراع أقل مما لو كانت الملفات متباعدة، هذا يقود إلى ملاحظتين يمكن القيام بهما حول تخصيص مساحة القرص التي ستُناقش في الفصل السادس:

- من الأفضل تخصيص مساحات القرص القريبة لبعضها لأجزاء الملف الواحد من قبل نظام التخصيص، بدلًا من تخصيص مساحات متناثرة حول القرص.
- من المهم جدًا التخلص من التفتت الخارجي للقرص بشكل دوري عن طريق تجميع البيانات المجزئة (المبعثرة) على القرص ليعمل بصورة أكثر فاعلية.

هناك عدد من الملفات وهياكل بيانات القرص التي يصل إليها النظام بشكل متكرر مثل ملفات المبادلة وهياكل إدارة مساحة القرص، لذلك إذا خُصصت هذه العناصر على مسارات المركز، فستكون حركة الذراع أقل مما لو خُصصت في أحد أطراف القرص. في الواقع، يمكن تعميم هذه الملاحظة والقول أنه يمكننا تحسين الأداء الكلي من خلال تخصيص المساحة بدءًا من مركز القرص والاتجاه إلى الخارج، بدلًا من البدء من أحد جوانب القرص والتحرك عبر ذلك.

6.9.3 التعامل مع الأخطاء

تدفع الشركات المصنعة للأقراص باستمرار باتجاه الاستفادة من التقنية المتاحة لزيادة الكثافة الخطية للخانات الثنائية، فبالنظر إلى مسار في منتصف قرص 5.25 بوصة نجد أن محيطه حوالي 300 ملم، إذا احتوى هذا المسار 300 قطاع، وكان كل قطاع يحوي 512 خانة ثمانية، فستكون الكثافة الخطية المسجلة حوالي 5000 خانة لكل ملم، مع الأخذ بعين الاعتبار حقيقة أن هناك بعض المساحات المفقودة بسبب الدباجة، وشفرة تصحيح الأخطاء، وكذلك

الفجوات المتروكة بين القطاعات. عملية تسجيل 5000 خانة لكل ملم تتطلب وجود طبقة منتظمة للغاية وطلاء مؤكسد ناعم جدًا. لسوء الحظ، ليس من الممكن تصنيع قرص بمثل هذه المواصفات من دون وجود أخطاء، فبمجرد تحسن تقنية التصنيع إلى الحد الذي من الممكن أن تعمل فيه من دون عيب في مثل هذه الكثافة، نجد أن مصممي الأقراص سينتقلون إلى كثافات أعلى، وذلك لغرض زيادة القدرات التخزينية، وهو ما من شأنه على الأرجح تكرار العيوب من جديد.

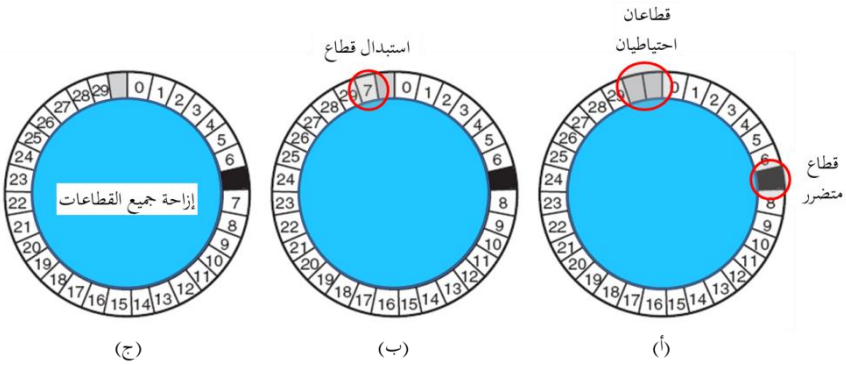
ينتج عن عيوب التصنيع قطاعات متضررة، وهي تلك القطاعات التي لا تُعطي بشكل صحيح القيم المكتوبة فيها عند قراءتها. إذا كان الضرر بسيط جدًا، ومتمثل في عدد قليل من الخانات الثنائية، فمن الممكن استخدام القطاع المتضرر وسيُترك الأمر لخانات شفرة تصحيح الأخطاء لتصحيح الأخطاء في كل مرة، أمَّا إذا كان الضرر كبيرًا، فلن تكون هناك إمكانية لتصحيح الخطأ، وبالتالي لن يُستخدم القطاع المتضرر. في العموم هناك حالتان عامتان للتعامل مع القاطعات المتضررة، إما داخل المتحكم، أو داخل نظام التشغيل.

1.6.9.3 الحالة الأولى: داخل المتحكم

قبل أن يُشحن القرص من المصنع، يتم اختباره، وتُكتب قائمة بالقطاعات المتضررة على القرص، ومن ثم تُعوض بقطاعات أخرى سليمة، للقيام بعملية التعويض هذه يمكن استعمال أكثر من طريقة. الشكل 3. 34-أ يُوضح أحد مسارات قرص به 30 قطاعًا للبيانات وقطاعان احتياطيان، القطاع السابع هو قطاع متضرر. في الطريقة الأولى سيقوم المتحكم بإعادة رسم خريطة أحد القطاعات الاحتياطية، ليحل محل القطاع السابع، وذلك كما هو مبين في الشكل 3. 34-ب. أمَّا الطريقة الثانية فتتمثل في إزاحة جميع القطاعات بمقدار قطاع واحد، كما هو مبين في الشكل 3. 34-ج.

في كلتا الطريقتين يجب أن تكون لدى المتحكم معرفة بماهية القطاعات المتضررة والمستبدلة، وللقيام بذلك يحتفظ المتحكم بهذه المعلومات عن طريق جداول داخلية (جدول لكل مسار)، أو عن طريق إعادة صياغة الدباجات لإعطاء أرقام القطاعات المعاد رسم خريطتها. إن استخدام الطريقة الأخيرة والمبينة في الشكل 3. 34-ج يتطلب جهدًا إضافيًا نتيجةً لإعادة صياغة 32 دباجة، ولكن في نهاية المطاف يُعطي أداء أفضل، لأن المسار بأكمله لا يزال من

الممكن قراءته في دورة واحدة.



الشكل 3. 34: أ) مسار قرص به قطاع متضرر– ب) استبدال قطاع متضرر بآخر احتياطي– ج) إزاحة جميع القطاعات لتمرير قطاع متضرر [Tanenbaum & Bos, 2015].

يمكن لأخطاء القرص أن تحدث أيضًا في أثناء التشغيل العادي، أي بعد تثبيت محرك الأقراص واستخدامه. أحد هذه الأخطاء قد يتمثل في عدم القدرة على معالجة بيانات شفرة تصحيح الأخطاء، وكخط دفاعي أولي تُعاد محاولة القراءة مرة أخرى، لأن بعض من أخطاء القراءة قد تكون عابرة وناجمة عن بقع من الغبار تكون موجودة تحت رأس القراءة، وسوف تنجلي بإعادة المحاولة، أمَّا إذا لاحظ المتحكم أن هذه الأخطاء متكررة في إطار قطاع معين، فيمكنه تعويض هذا القطاع مبكرًا بقطاع احتياطي قبل تضرر القطاع بالكامل، بهذه الطريقة سوف لن يكون هناك فقد في البيانات، ولن يلاحظ نظام التشغيل المستخدم أي مشكلة. غالبًا ما تُستخدم طريقة الاستبدال باعتبار أن القطاعات الأخرى قد تحتوي على بيانات في لحظة الاستبدال، لأن استخدام طريقة الإزاحة ستؤدي ليس فقط إلى إعادة صياغة الدباجة بل أيضًا إلى نسخ كافة البيانات الموجودة في القطاعات.

2.6.9.3 الحالة الثانية: داخل نظام التشغيل

في حالة عدم تمكن المتحكم من معالجة الأخطاء ومن إعادة رسم خريطة القطاعات بشفافية، يجب على نظام التشغيل القيام بهذه المهمة، ولكن في إطار البرمجيات. هذا يعني أنه يجب على نظام التشغيل الحصول أولًا على قائمة القطاعات المتضررة، وذلك إما عن طريق

قراءتها من القرص، أو ببساطة اختبار القرص نفسه، بالتالي بمجرد أن يُحدد نظام التشغيل القطاعات المتضررة، يمكن له إعادة بناء خارطة الجداول من جديد. إذا كان لنظام التشغيل الرغبة في استخدام الطريقة المبينة في الشكل 3. 34–ج، فإنه يجب عليه إزاحة البيانات في القطاعات من 7 إلى 29 بمقدار قطاع واحد.

إذا عالج نظام التشغيل خرائط جديدة، يجب عليه أن يتأكد من عدم وجود قطاعات متضررة في جميع الملفات، وكذلك في قائمة القطاعات الحرة، أو في خرائط الثنائية. للقيام بذلك يُنشأ ملف مشفر يحوي جميع القطاعات المتضررة عن طريق تشفير القرص بأكمله[13]، ولكي لا تكون هناك فرصة لقراءة هذا الملف فيجب ألَّا يدخل هذا الملف في نظام الملفات.

ومع ذلك لا تزال هناك مشكلة أخرى تتمثل في النسخ الاحتياطي عند نسخ الملفات ملف بملف، في هذه الحالة من المهم جدًا على وسائل النسخ الاحتياطي عدم محاولة نسخ ملف القطاعات المتضرر، ولمنع ذلك يجب على نظام التشغيل إخفاء الملف المشفر بشكل جيد بحيث لا يمكن العثور عليه من قبل وسائل النسخ الاحتياطي، أمَّا إذا جرت عملية النسخ الاحتياطي للقرص قطاعًا بقطاع، فسوف يكون من الصعب– إن لم يكن مستحيلا– منع أخطاء القراءة في أثناء النسخ الاحتياطي، الأمل الوحيد في هذه الحالة هو أن برنامج النسخ الاحتياطي لديه ما يكفي من الذكاء للتخلي عن عملية القراءة للقطاع المتضرر بعد 10 محاولات فشل ليواصل قراءة القطاع التالي.

القطاعات المتضررة ليست هي المصدر الوحيد للأخطاء، هناك أخطاء أخرى ناتجة عن مشاكل ميكانيكية في الذراع تتمثل في أخطاء البحث، والناتجة من تتبع المتحكم لموضع الذراع داخليًا. للقيام بعملية البحث يُصدر المتحكم سلسلة من النبضات إلى محرك الذراع بمعدل نبضة

[13] يتمثل تشفير القرص بأكمله في استخدام برنامج تشفير لتحويل كافة البيانات الموجودة على أحد الأقراص إلى نموذج غير قابل للقراءة إلا بواسطة مفتاح التشفير.

واحدة لكل أسطوانة لغرض تحريك الذراع بمقدار أسطوانة واحدة، عندما يصل الذراع إلى وجهته المحددة، يقرأ المتحكم الرقم الفعلي للأسطوانة من دباجة القطاع، إذا وجد الذراع نفسه في المكان الخطأ، فسيتولد خطأ بحث.

تسعى معظم متحكمات القرص الصلب لتصحيح أخطاء البحث تلقائيًا، بينما تُضبط معظم متحكمات الأقراص المرنة (بما في ذلك بنتيوم) خانة الخطأ وتَترك الباقي للمشغل، يُعالج المشغل هذا الخطأ عن طريق إصدار أمر إعادة المعايرة، لغرض نقل الذراع بقدر المستطاع إلى أبعد نقطة، ومن ثم إعادة ضبط السجل الداخلي للمتحكم والخاص بالأسطوانة الحالية على صفر. يَحل هذا الإجراء في العادة هذه المشكلة، أمَّا في حالة عدم حدوث ذلك، فمن الأجدى إصلاح محرك الأقراص.

10.3 تحسين أداء الإدخال، والإخراج

تَذكر أنَّه في الحواسيب الحديثة بإمكان وحدة المعالجة المركزية تنفيذ عشرة مليون تعليمة في نفس الوقت الذي يستغرقه قرص لتنفيذ وظيفة واحدة خاصة بالإدخال، والإخراج، وبالتالي فإن كل عملية إدخال، وإخراج فعلية مطبقة على القرص لها تأثير كبير على أداء النظام، بناءً عليه من المهم جدًا تحسين أداء الإدخال، والإخراج، وللقيام بذلك هناك ثلاثة طرق تُساهم في هذ التحسين وهي:

1. تقليل عدد طلبات الإدخال، والإخراج.
2. تنفيذ التخزين اللحظي.
3. جدولة طلبات الإدخال، والإخراج.

1.10.3 تقليل عدد طلبات الإدخال، والإخراج

إن أفضل طريقة لتحسين أداء الإدخال، والإخراج تتمثل في تقليل عدد طلبات الإدخال،

والإخراج التي يقوم بها برنامج التطبيق. لتوضيح هذه الفكرة يمكن النظر إلى المثال البسيط التالي والذي يُمثل جزء من شفرة SQL[14]:

```
SELECT Name from STUDENT_RECORDS
SELECT Address from STUDENT_RECORDS
```

تُمثل هذه الشفرة أمران، الأول لاختيار أسماء من سجل الطلبة، والثاني لاختيار عناوين من نفس السجل. فبدلًا من كتابة هذه الشفرة بهذه الطريقة، يمكن دمج الأمران في أمر واحد واستخدام الأسلوب التالي:

```
SELECT Name, Address from STUDENT_RECORDS
```

كلا المثالين سيتحصلا على قائمة بأسماء الطلاب والعناوين، إلَّا إنَّ الأسلوب الأول سيؤدي بمشغل قاعدة البيانات إلى قراءة جدول الطلبة من القرص مرتين، بينما يتطلب النهج الثاني قراءة الجدول مرة واحدة فقط، الأمر الذي سيُوفر قدرًا كبيرًا من الوقت.

في الواقع غالبًا ما يقضي المبرمجون وقتًا طويلًا في تحسين التعليمات البرمجية في برنامج– ما–، الأمر الذي قد ينتج عنه تَولد كمية كبيرة من طلبات الإدخال، والإخراج، لذلك يُمكن تحسين أداء مثل هذه البرامج بشكل كبير عن طريق تقليل عدد طلبات الإدخال، والإخراج المتعلقة بالقرص، علمًا بأن إزالة طلب إدخال، وإخراج– في العادة– أسهل من التخلص من عشرة مليون تعليمة. إذا كان هناك تطبيقات تدعم عدة طلبات في وقت واحد (على سبيل المثال، خدمة ويب)، فإن ذلك سيؤثر بشكل كبير على مدى جودة التطبيق في دعم العديد من المستخدمين، وهذا بدوره سيؤثر على حجم الإنفاق الذي يجب أن يُنفق على جهاز الحاسوب.

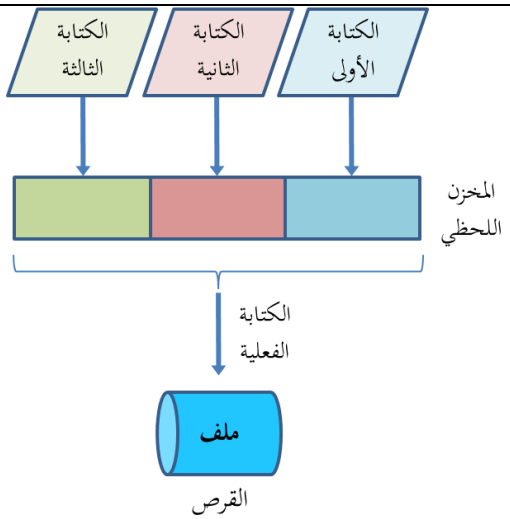
[14] SQL هي اختصار لجملة Structured Query Language وتعني لغة الاستعلامات البنائية وهي لغة تُستخدم للتواصل مع قاعدة البيانات، وإدارتها.

2.10.3 التخزين اللحظي

فكرة التخزين اللحظي هي فكرة بسيطة: إذا كان ليس بالإمكان القضاء على الإدخال، والإخراج بشكل كامل، فيجب جعل طلبات الإدخال، والإخراج الفعلية تجلب أكبر قدر ممكن من البيانات، وهو ما سيؤدي ضمنيًا إلى تقليل عدد الطلبات الفعلية بواسطة عامل التخزين اللحظي المستخدم والذي يتطلب وجود مخزن مؤقت كبير.

عمليًا تحتوي بعض من وحدات تحكم القرص الحديثة على كمية كبيرة من الذاكرة المؤقتة، لذلك يمكن تنفيذ تحسين الأداء التالي: بمجرد نقل الرأس إلى المسار الصحيح، وفي أثناء الانتظار إلى أن يَتَدرج القطاع المطلوب في موضعه الصحيح تحت رأس القراءة والكتابة، بالإمكان قراءة القطاعات الأخرى في المخزن اللحظي تِبعًا لظهورها تحت الرأس. لاحقاً، إذا طلبت وحدة المعالجة المركزية تلك القطاعات، فبإمكان متحكم القرص أن يُوفرها على الفور من ذاكرة التخزين المؤقت، بدلاً من الاضطرار إلى الوصول الفعلي للقرص مرة أخرى.

في هذه الحالة يجب التفريق بين طلبات الإدخال، والإخراج الفعلية والمنطقية. فالأولى يُقصد بها الوصول الحقيقي إلى الكيان المادي لوحدة الإدخال، والإخراج، بينما يُقصد بالأخيرة الوصول إلى المخزن اللحظي، لذلك إذا خُزِّنت ثلاثة طلبات منطقية مقابل كل طلب فعلي، فسيكون عدد طلبات الإدخال، والإخراج الفعلية ثلث الرقم المطلوب بدون تخزين لحظي. مثلًا، قد يرغب تطبيق– ما– في تنفيذ طلب إدخال، وإخراج منطقي لكل سطر في ملف نصي، فإنه بالإمكان زيادة أداء النظام عن طريق تنفيذ طلب إدخال، وإخراج فعلي واحد لمجموعة كبيرة من الأسطر. يوضح الشكل 3. 35 ثلاث كتابات منطقية مقابل كتابة واحدة فعلية عن طريق استخدام التخزين المؤقت. لمزيد من المعلومات حول هذا الموضوع يمكن الرجوع إلى غاريدو وآخرون (**Garrido et al., 2011**).



الشكل 3. 35: الكتابة عن طريق استخدام التخزين اللحظي.

3.10.3 جدولة طلبات الإدخال، والإخراج

بالنسبة لمعظم أجهزة الإدخال، والإخراج من المناسب استخدام خوارزمية جدولة القادم أولًا يُخدم أولًا في جدولة طلبات الإدخال، والإخراج، لكونها تتماشى معها في آلية التنفيذ. مثلًا، تُنفذ مقاطع ملف موسيقى بنفس الترتيب التسلسلي لعملية جلبها من القرص. ومع ذلك، قد لا تتناسب هذه الخوارزمية مع بعض من الأجهزة الأخرى (الأقراص خاصة)، لأن الترتيب الذي تُعالج به الطلبات لا يُقيد بطبيعة خصائص الجهاز. مثلًا، في الأنظمة النموذجية تكون هناك طلبات إدخال، وإخراج خاصة بالقرص قيد الانتظار آتية من عدة عمليات مختلفة، الأداء الصحيح لهذه العمليات لا يعتمد في العادة على الترتيب الذي تحدث فيه عمليات الإدخال، والإخراج الخاصة بالقرص فعليًا، بالتالي من الأفضل جدولة الطلبات بطرق أخرى تُساهم في تحسين أداء أجهزة الأقراص.

اختيار الجدولة المناسبة لطلبات الإدخال، والإخراج من شأنه أن يُقلل من زمن التبديل الخاص بالقرص الصلب (أي الزمن اللازم للتغيير من طلب إلى آخر) والذي هو في الأساس مرتفع للغاية خاصة مع مقارنته بزمن التبديل في سياق وحدة المعالجة المركزية. فتحريك ذراع القرص (مع رأس القراءة، والكتابة) والانتظار إلى أن يلف القرص ليصل إلى موضعه الصحيح، يترتب عليه أن مدة زمن التبديل أكبر بكثير من عملية القراءة، والكتابة الفعلية نفسها. يمكن

تشبيه هذا الأمر بحالة مماثلة تتمثل في الشخص الذي يخدم عدة عملاء موزعون في مناطق جغرافية مختلفة، بالتالي فإن متوسط الزمن الذي يستغرقه الشخص الخدمي للسفر من موقع خدمة إلى آخر أكبر بكثير من متوسط زمن الخدمة نفسها في كل موقع.

بالإضافة إلى ذلك، بالنسبة إلى معظم الأجهزة يكون زمن تنفيذ الأمر مستقلاً عن الأمر السابق، إلَّا إنَّ عمليات الإدخال، والإخراج الخاصة بالقرص لها خاصية أن الزمن المستغرق في تنفيذ عملية إدخال، وإخراج كاملة يعتمد على العملية السابقة (أي، أين تركت العملية السابقة الذراع؟). يُقال أن هذا الجانب من معالجة الإدخال، ولإخراج يعتمد على الحالة ويؤثر بشكل أساسي على زمن التبديل، لذلك يُساهم اختيار أسلوب جدولة مناسب في تقليل هذا التأثير.

11.3 موجز الفصل

تُعتبر وحدات الإدخال، والإخرج من أهم المعدات التي تُلحق بالحاسوب، كما أنها تتنوع بتنوع التطبيقات الحاسوبية في معظم إن لم يكن في شتّى مجالات الحياة، لذلك خُصص هذا الفصل لموضوع مهم في نظم التشغيل وهو إدارة وحدات الإدخال، والإخرج. تضمنت هذه الإدارة التعرف على كل من المفاهيم المادية، والمعنوية لوحدات الإدخال، والإخراج، وعلى أنواعها، وعلى أهم مكوناتها المادية، وهو متحكم الجهاز والذي يُمثل الوسيط بينها وبين وحدة المعالجة المركزية، كما يُوفر آلية الوصول إليها من خلال منافذ إدخال، وإخراج مستقلة، أو باستخدام الذاكرة المعنونة لوحدة الإدخال، والإخراج، أو عن طريق الوصول المباشر للذاكرة.

ولأهمية المقاطعات بالنسبة لوحدات الإدخال، والإخراج، قدَّم الفصل مراجعة تفصلية لهذا الموضوع تناولت تعريف المقاطعات والمتمثل في التغيير في تتالي تنفيذ العمليات نتيجةً لأحداث قد تكون عرضية من داخل وحدة المعالجة المركزية أو خارجية تُعبر عن تفاعلات مع وحدات الإدخال، والإخراج، سُميت هذه الأحداث بالمقاطعات المتزامنة، وغير المتزامنة على التوالي. كما تناول التقديم التعريف بإجراء خدمة المقاطعة، وسرد لأغلب الخطوات المتبعة لمعالجة المقاطعات، إضافةً إلى ذلك تتطرقت المراجعة إلى مفهومي المقاطعات الدقيقة، وغير الدقيقة، فالأول يُقصد به ترك الجهاز في حالة محددة ومعرَّفة جيدًا بعد حدوث المقاطعة وهو ما نجده في الأنظمة القديمة، أمَّا المفهوم الثاني فنجده في الأنظمة الحديثة وخصوصًا تلك التي تعتمد على

تقنية الأنابيب والسلم الفائق، والذي تكون فيه التعليمات في مراحل مختلفة من عملية التنفيذ، وهو ما يعني ترك الجهاز في حالة غير محددة عند حدوث المقاطعة.

يلي هذه المراجعة تعرض الفصل للمفاهيم المعنوية لوحدات الإدخال، والإخراج من خلال شرح البنية الأساسية لها، وكذلك الأهداف الرئيسية لهذه المفاهيم، نذكر منها على سبيل المثال لا الحصر استقلالية الجهاز، والتسمية الموحدة، ومعالجة الأخطاء، والنقل المتزامن وغير المتزامن، والتخزين اللحظي، والوصول التسلسلي أو العشوائي للأجهزة، والتي تتمحور في مجملها في إخفاء تعقيدات، واختلافات، وتنوع وحدات الإدخال، والإخراج من قبل نظام التشغيل عن المستخدم وتقديم أنواع تقليدية، وأنماط مطوَّرة للوصول إلى الأجهزة تكون مفيدة وقابلة للتطبيق على نطاق واسع.

ومع اختلاف وحدات الإدخال، والإخراج تنوعت معه أيضًا طرق إنجاز عمليات الإدخال، والإخراج تضمنت الإدخال، والإخراج المبرمج، والإدخال، والإخراج بالمقاطعة والإدخال، والإخراج باستخدام الوصول المباشر للذاكرة، والتي تناولها الفصل بنوع من التفصيل أوضح من خلاله المفاهيم الأساسية لكل طريقة، وكذلك مزاياها وعيوبها. خلص هذا التوضيح إلى أنَّه بالرغم من كفاءة طريقة الإدخال، والإخراج باستخدام متحكم الوصول المباشر للذاكرة في نقل البيانات بين الذاكرة ووحدة الإدخال، والإخراج بشكل مستقل عن وحدة المعالجة المركزية، إلاَّ إنَّ عدم مواكبة المتحكم لسرعة المعالج سيُقلل من هذه الكفاءة وخصوصًا إذا لم يُسْتَفاد من المعالج طيلة فترة انتظاره للمتحكم، الأمر الذي قد يجعل الطريقة الأولى، أو الثانية بديلًا أفضل.

ولكي تكون هذه الطرق ذات جدوى، وقابلة للتطبيق على نطاق واسع، نُظمت برمجيات الإدخال، والإخراج الحديثة في أربع طبقات مجردة ساهمت في مرونة البرمجيات بدرجة عالية، وتلخصت في برمجيات الإدخال، والإخراج لفضاء المستخدم، والبرمجيات المستقلة لأجهزة الإدخال، والإخراج، ومشغل الجهاز، وأخيرًا طبقة معالجة المقاطعات. الغاية والغرض من هذا التجريد كما أوضح الفصل هو إتاحة الفرصة لتطوير، وتعديل هذه الطبقات بشكل مستقل دون أن تُؤثر كل منها على الأخرى، كما لخص الفصل أغلب الوظائف الرئيسية لهذه الطبقات.

ولأن وحدات الإدخال، والإخراج ترتبط في مجملها مع وحدة المعالجة المركزية عن طريق ناقل النظام، عرَّج الفصل بشكل موجز على الناقلات الذكية، مثل الناقل التسلسلي العام والتي

عُرفت أيضًا بواجهات البرامج الموجَّهة لتمتعها بمستوى عالٍ من الذكاء ساهم في جعل كافة الوظائف الخاصة بالجهاز تَتَواجد في وحدة تحكم الجهاز، بدلًا من إلحاقها بنظام التشغيل. انتقل بعد ذلك الفصل إلى مناقشة الأقراص كأحد الأمثلة لهذه الوحدات، تناول فيها الكيان المادي الخاص بالأقراص الممغنطة، والشكل العام للتقسيم المنطقي لها من حيث الأسطوانات، والمسارات، وكذلك المقاطع والتي تُمثل أصغر كمية من البيانات التي يمكن قراءتها أو كتابتها فعليًا، يلي هذا تعرضت هذه المناقشة إلى عملية تهيئة القرص والتي تتم على مرحلتين هما: مرحلة التهيئة ذات المستوى المنخفض، ومرحلة التهيئة ذات المستوى العالي، واللاتان تضمنت كل واحدة منهما مجموعة من الخطوات والمهام الضرورية لتنسيق القرص وتجهيزه لحفظ البيانات بكفاءة عالية، كما أشارت هذه المناقشة إلى جملة من التحديات الخاصة بعملية التهيئة والحلول التي وُضعت لها.

وكتكملة لمناقشة الأقراص تطرق الفصل إلى آليات توفير زمن وصول سريع، وعرض نطاق كبير للقرص من خلال إدارة طلبات الإدخال، والإخرج الخاصة به باستخدام عدة خوارزميات عُرِفت باسم خوارزميات جدولة ذراع القرص، تمثلت في خوارزمية القادم أولًا يُخدم أولًا، وخوارزمية الأقصر زمن بحث أولًا، وخوارزمية المسح، وخوارزمية المسح الدائري، وكذلك خوارزمية الجدولة الناظرة. من خلال الأمثلة التوضيحة لمجموعة الخوارزميات هذه لاحظنا تفاوت فاعلياتها في تحسين زمن البحث، أي الزمن اللازم لتحريك ذراع القرص إلى الأسطوانة الخاصة بالقطاع المطلوب، الأمر الذي بيَّن أنّ الأداء الإجمالي لكل منها يعتمد على عدد ومواقع الطلبات داخل القرص.

من الأمور الأخرى المهمة لتحسين كفاءة جدولة ذراع القرص هو تقليل التأخير الدوراني لذراع القرص والذي يمكن معالجته بوضع الطلبات لنفس المسار معًا في طابور الطلبات، وطلبها حسب موضع الدوران للقطاع المطلوب نسبة إلى الموضع الحالي للقرص، والذي قد يُستعاض عنه باستقبال عدة طلبات متزامنة تُحفظ في ذاكرة تخزين مؤقت كبيرة. إضافةً إلى ذلك يُساهم كل من تخصيص مساحات القرص القريبة من بعضها البعض لأجزاء الملف، والتخلص من التفتت الخارجي للقرص في زيادة هذا التحسين.

أخيرًا، فيما يخص القرص تُعتبر معالجة أخطاء القرص أمرًا في غاية الأهمية والتي تَكْمن في

التعرف على القطاعات المتضررة وعلى أخطاء البحث الناتجة من تتبع موضع الذراع داخليًا من قبل المتحكم، ومن ثم جبر الضرر أو على الأقل محاولة تقليله. لذلك ناقش الفصل عدة مفاهيم لتصحيح مثل هذه الأخطاء، منها ما هو في إطار المتحكم، ومنها ما هو داخل برمجيات نظم التشغيل.

أوضح هذا الفصل في نهايته الأساليب التي تُساعد في تحسين الأداء العام لوحدات الإدخال، والإخراج من خلال تقليل عدد طلبات الإدخال أو الإخراج، لأن إزالة مثل هذه الطلبات قد يُوفر زمنًا يكفي لتنفيذ عشرات الملايين من التعليمات داخل وحدة المعالجة المركزية، أيضًا إن لم يكن ممكنًا القضاء على الإدخال، والإخراج بشكل كامل، فيجب على أقل تقدير جعل طلبات الإدخال، والإخراج الفعلية تجلب أكبر قدرًا ممكنًا من البيانات وتضعها في مخزن مؤقت كبير. أخيرًا، من الأفضل جدولة طلبات القرص بطرق مختلفة بما يتناسب مع التطبيق المستخدم، لأن ذلك من شأنه أن يُساهم أيضًا في تحسين أداء أجهزة الأقراص.

12.3 أسئلة للمراجعة

1. ما مفهوم إدارة وحدات الإدخال، والإخراج؟
2. ما الوظائف الرئيسية لإدارة وحدات الإدخال، والإخراج في إطار نظم التشغيل؟
3. ما أنواع وحدات الإدخال، والإخراج؟ وما المقصود بكل نوع؟
4. ما الخطوات العامة التي تعمل به متحكمات الأجهزة؟
5. لماذا يتوجب على نظم التشغيل الرد بسرعة على المقاطعات؟
6. لماذ نحتاج إلى التخزين اللحظي في متحكمات الجهاز؟
7. عدد الطرق التي من الممكن أن تتواصل بها وحدات الإدخال، والإخراج مع وحدة المعالجة المركزية، مع شرح إحداها بنوع من الإيجاز.
8. وضح فكرة عمل طريقة الوصول المباشر للذاكرة.
9. يُوضح الشكل 3. 6 الترتيبات المختلفة لمتحكم الوصول المباشر للذاكرة، ناقش هذه الترتيبات من حيث المزايا والعيوب.
10. عرف كل من المقاطعات المتزامنة، وغير المتزامنة، مع ذكر مثال لكل نوع.
11. بين أهم الخطوات التي تعمل بها المقاطعة.

12. اشرح كل من المقاطعات الدقيقة وغير الدقيقة، وأين يتواجد كل نوع؟
13. ما خصائص المقاطعة الدقيقة؟
14. اشرح عملية المفاضلة بين المقاطعات الدقيقة وغير الدقيقة في إطار تقنية الأنابيب.
15. عدِّد الأهداف الرئيسية لبرمجيات الإدخال، والإخراج، مع شرح ثلاثة أهداف.
16. ما المقصود باستقلالية الجهاز في إطار نظم التشغيل؟
17. ما الفرق بين الإدخال، والإخراج المبرمج، والإدخال، والإخراج بالمقاطعة؟
18. ما هو تسلسل ديزي؟
19. لماذا قُسمت برمجيات الإدخال، والإخراج إلى مجموعة من الطبقات؟
20. في أيٍ من طبقات برمجيات الإدخال، والإخراج تُنفذ الوظائف التالية:
    أ- القيام باستدعاء الإدخال، والإخراج، تنسيق الإدخال، والإخراج، المكب.
    ب- التسمية، الحماية، الإيقاف، التخزين اللحظي، التخصيص.
    ج- تجهيز سجلات الجهاز، اختبار الحالات.
    د- إيقاظ المشغل عند إنتهاء الإدخال، والإخراج.
21. ما الوظائف التي تُوفرها البرمجيات المستقلة للأجهزة؟
22. ما العوامل المؤثرة في زمن قراءة أو كتابة مقطع قرص؟
23. ما المقصود بالتعشيق؟ وما المشكلة التي يُحاول حلها؟ وما أنواعه؟
24. إذا ما أُستخدم التعشيق المزدوج مع القرص، فهل من الضروري استخدام انحراف الأسطوانة، لكي لا يحدث فقد للبيانات عند الانتقال من مسار إلى آخر. ناقش إجابتك؟
25. ما المقصود بانحراف الأسطوانة؟ وما المشكلة التي يُحاول حلها؟
26. ما مسافة انحراف الأسطوانة اللازمة لمشغل أقراص ذو **7200** لفة في الدقيقة، إذا كان زمن الانتقال من مسار إلى آخر يأخذ **1000** ميكرو ثانية؟ علمًا بأن المسار يحوي **200** قطاع، وكل قطاع يسع **512** خانة ثمانية؟
27. ما أهم الخطوات التي تُتخذ في مرحلة التهيئة ذات المستوى العالي؟ وماذا يحدث في كل خطوة؟
28. هل هناك أي خوارزمية أخرى من خوارزميات ذراع القرص باستثناء خوارزمية القادم أولًا يُخدم أولًا تتناسب مع بيئة المستخدم الواحد؟ علل اجابتك؟

29. علل لماذا تميل خوارزمية الأقصر زمن بحث أولًا إلى خدمة الأسطوانات الوسطية بدلًا من الأسطوانات الداخلية، والخارجية؟

30. بفرض أن هناك طلبية تخص قراءة الأسطوانات 21، 14، 2، 30، 26، وكذلك 40 بنفس هذا الترتيب، وكان زمن الانتقال من أسطوانة إلى أخرى يأخذ 2 ملّي ثانية، وكان ذراع القرص مبدئيًا عند الأسطوانة 12. فما هو زمن البحث اللازم لقراءة كل هذه الأسطوانات باستخدام:

أ- خوارزمية القادم أولًا يُخدم أولًا.

ب- خوارزمية الأقصر زمن بحث أولًا.

ج- خوارزمية المسح، علمًا أن الاتجاه إلى أسفل هو الاتجاه المبدئي لعملية المسح.

31. بفرض أنَّ هناك قرص به 4000 أسطوانة مرقمة من0 إلى 3999. يخدم المشغل حاليًا طلب عند الأسطوانة 1150، والطلب السابق كان عند الأسطوانة 805. أمَّا الطلبات المعلقة فوصلت على النحو التالي: 1069، 212، 1296، 1800، 555، 618، 360، 523، 3965، 2681. انطلاقًا من الموضع الحالي لرأس القراءة، ما هو إجمالي المسافة (معبرًا عنها بالأسطوانات) التي يتحركها ذراع القرص لتلبية هذه الطلبات في حالة استخدام خوارزميات جدولة ذراع القرص التالية:

أ- خوارزمية القادم أولًا يُخدم أولًا.

ب- خوارزمية الأقصر زمن بحث أولًا.

ج- خوارزمية المسح.

د- خوارزمية المسح الدائري.

هـ- خوارزمية الجدولة الناظرة.

32. لماذ تُرسل الملفات المرد طباعتها إلى مكب الطباعة في القرص، بدلًا من إرسالها مباشرة إلى الطابعة؟

33. كيف يمكن تحسين التأخير الدوراني؟

34. لماذا من المهم جدًا تحسين أدء الإدخال، والإخراج؟

35. وضح لماذا يتحسن أداء القرص إذا ما خُصِّصت مساحة القرص انطلاقًا من المركز باتجاه الخارج، وليس العكس؟

36. لماذا يُساعد القضاء على التفتت الخارجي للقرص على جعل القرص يعمل بصورة أكثر

فاعلية؟

37. ما الأخطاء التي من الممكن أن تحدث في القرص؟ وكيف يُمكن معالجتها؟

38. كيف يُمكن للتخزين اللحظي أن يُساهم في تحسين أداء الإدخال، والإخراج؟ وضح بمثال.

39. أيٍ من خوارزميات جدولة ذراع القرص تتلاءم بشكل أفضل مع قرص صوتي مدمج؟ علل اجابتك؟

الفصل الرابع حالة
الجـــــمـود

1.4 مدخل إلى حالة الجمود

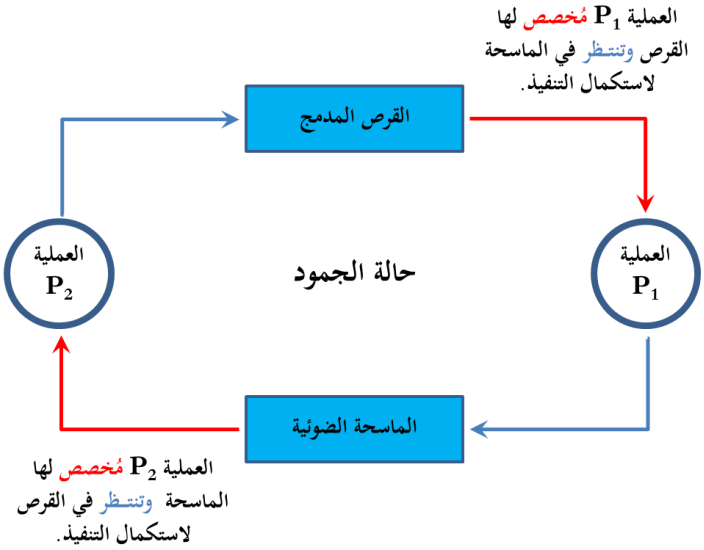
في بيئة البرمجة المتعددة تتنافس عدة عمليات للحصول على عدد محدود من المصادر من خلال تَقدُمها بطلبات للحصول على هذه المصادر، إذا لم يتوفر لأي عملية مصدرها في لحظة الطلب، فستدخل في حالة الانتظار، في بعض الأحيان، قد يستمر الانتظار إلى الأبد، لأن المصدر الذي طلبته مستخدم من قبل عملية أخرى مقيدة في حالة انتظار أخرى.

تعرضنا خلال الفصول السابقة إلى بعض من المصادر التي بإمكان أكثر من عملية استخدامها في آن واحد كالأقراص الصلبة، وإلى مصادر أخرى غير قابلة للاستخدام في أي لحظة إلَّا من قبل عملية واحدة فقط. فالطابعة مثلًا، لا يمكن استخدامها في نفس الوقت من قبل أكثر من عملية، وفي حالة توفر الرغبة لدى أكثر من عملية في استخدامها فيجب تحقيق ذلك عن طريق الاستعمال القصري لها، بمعنى استخدامها من قبل العملية الأولى حتى النهاية، ثم السماح للعملية الثانية باستخدامها حتى النهاية، وهكذا.

للأسف إن الاستعمال القصري للمصادر محفوف بالمخاطر، ولتوضيح ذلك يمكن التمعن في المثال التالي: بفرض أن هناك حاسوب لديه ماسحة ضوئية وقرص مدمج، وبفرض أن هناك عمليتان، طلبت الأولى استخدام القرص المدمج، في حين طلبت الثانية استخدام الماسحة الضوئية وتمت تلبية الطلبين. والآن، بفرض أن العملية الأولى احتاجت الماسحة الضوئية لكي تُكْمِل عملها فطلبتها دون ترك استخدام القرص المدمج، الأمر نفسه تكرر مع العملية الثانية، حيث طلبت استخدام القرص المدمج– لكي تُنهي عملها– مع احتفاظها بالسيطرة على الماسحة الضوئية، كما هو موضح في الشكل 4. 1. في هذه الحالة لن تستطيع أيٍ منهما إنهاء عملها، لأنه لا يمكن تلبية الطلبين الأخيرين وهو ما سيؤدي إلى ذهاب العمليتان إلى الانتظار (السكون) الأبدي. هذا الموقف يُعرف بحالة الجمود، وهو موضوع هذا الفصل.

سنتعرف في هذا الفصل أولًا على المصادر وأنواعها، ومن ثم آلية استخدامها، كما سيُنظر إلى كيفية حدوث حالة الجمود، ودراسة بعض من الطرق لمنعه، أو تفاديه، والتعرف على أنواع أخرى له، فحالة الجمود هذه تحدث في سياق أنظمة التشغيل، إلّا إنَّها تتواجد كذلك في سياق نظم قواعد البيانات، وعدد من الأنظمة الأخرى في علوم الحاسوب، لذلك فهي قابلة للتطبيق في

الواقع على مجموعة واسعة من أنظمة العمليات المتعددة. من ناحية أخرى، كُتب الكثير عن حالة الجمود، كما ظهر العديد من المراجع حول هذا الموضوع في سياق أنظمة التشغيل منها: نيوتن، (Newton, 1979)، وزوبل، (Zobel, 1983). ورغم أن هذين المرجعين من المراجع القديمة، إلَّا إنَّ تناولهما لموضوع الجمود كان جيدًا، لذلك فهما لا يزالان مفيدان.



الشكل 4. 1: حالة الجمود.

2.4 تعريف حالة الجمود

من خلال ما سبق ذكره يُمكن ملاحظة أنه في حالة الجمود يكون التقدم في تنفيذ العمليات صعبًا إن لم يكن مستحيلًا، وأنه يحدث عندما تنتظر عمليتان أو أكثر إلى أجل غير مسمى لحدث لا يمكن أن يحدث إلا بإحدى العمليات المشاركة في عملية الانتظار هذه، بالتالي يُمكن تعريف الجمود على النحو التالي:

يمكان مجموعة من العمليات الدخول إلى حالة الجمود عندما تنتظر كل عملية من هذه العمليات حدث– ما– لا يحدث إلَّا بسبب عملية أخرى داخل نفس المجموعة، ونظرًا لأنَّ جميع العمليات تنتظر، فلن يتسبب أيٍ منها في أي حدث يُمكن أن يُوقظ أي عملية أخرى داخل هذه المجموعة، الأمر الذي سيجعل كل العمليات تنتظر إلى الأبد.

بُني هذا التعريف على فرضية أنَّ لكل عملية خيط واحد، وأنه لا توجد مقاطعات بإمكانها إيقاظ عملية موقوفة (لمنع إيقاظ عملية موقوفة بسبب الجمود) قد ينتج عنها أحداث تؤدي إلى تحرر عمليات أخرى داخل المجموعة.

في معظم الأحيان، الحدث الذي تنتظر فيه كل عملية هو: تحرير بعض من المصادر المملوكة حاليًا لعمليات أخرى داخل نفس المجموعة. بعبارة أخرى، كل عضو في مجموعة من العمليات الموقوفة بسبب الجمود ينتظر في مصدر يمتلكه عضو آخر ينتمي إلى المجموعة نفسها، أي لا يمكن لأي عملية أن تشتغل، ولا أن تُحرر أي مصدر، ولا أن تُوقظ عملية أخرى. هنا يجب التنويه إلى أن عدد العمليات، وعدد ونوع كل من المصادر المملوكة، وطلبات اكتسابها ليست بالأمور المهمة. بالتالي هذه النتيجة صالحة لأي نوع من المصادر، بما في ذلك المصادر المادية والمعنوية. يُسمى هذا النوع من الجمود بجمود المصادر، والذي ربما هو الأكثر شيوعًا، ولكنه ليس بالنوع الوحيد، عليه سندرس أولًا جمود المصادر بالتفصيل، ومن ثم نُعرِّج بشكل موجز على الأنواع الأخرى منه مع نهاية هذا الفصل.

3.4 المصادر

كما لاحظنا سابقًا أن الجمود يحدث عندما يكون هناك استعمال قصري من قبل العمليات للأجهزة، أو لسجلات البيانات، أو للملفات. لذلك لجعل مناقشة حالة الجمود عامة، سوف نُشير إلى هذه الأجزاء بالمصادر، فالمصدر إمَّا أن يكون جهاز (مثل، محرك الأقراص) أو جزء من المعلومات (مثل، سجل في قاعدة البيانات). باختصار، المصدر شيء يجب اقتناؤه، استخدامه على مدى فترة زمنية محددة، ومن ثم تحريره أو إعادته، علمًا بأن للحاسوب (النظام) الحق في امتلاك عدة مصادر مختلفة.

لغرض مناقشة الجمود يمكن نمذجة النظام على أساس مجموعة من المصادر المحدودة، والتي يمكن تقسيمها إلى فئات مختلفة لغرض تخصيصها لعدد من العمليات لكل منها احتياجات مختلفة. قد تشمل فئات المصادر الذاكرة، الطابعات، وحدات المعالجة المركزية، والملفات المفتوحة، وما إلى ذلك، بحيث تكون جميع المصادر داخل الفئة الواحدة متكافئة مثل، امتلاك الحاسوب لثلاث طابعات، وهو ما يعني أن أي واحد منها يمكن أن يُستخدم لتلبية أي طلب

للمصدر. إذا لم يكن هذا هو الحال (أي إذا كان هناك بعض من الاختلاف بين المصادر ضمن الفئة الواحدة)، فيجب حينئذ تقسيم هذه الفئة إلى فئات منفصلة على سبيل المثال، قد تحتاج الطابعات إلى فصلها إلى طابعات ليزرية، وطابعات حبرية ملونة. بصفة عامة تأتي مصادر الحاسوب في نوعين: مصادر قابلة للسحب، ومصادر غير قابلة للسحب.

1.3.4 المصادر القابلة للسحب

تُعرَّف المصادر القابلة للسحب بأنها المصادر التي يُمكن سحبها من العملية المالكة لها دون حدوث أي آثار سيئة، ومثال ذلك الذاكرة. فبالنظر، إلى نظام يملك ذاكرة مستخدم بحجم 512 ميجابايت، وطابعة واحدة، وعمليتين بحجم 512 ميجابايت لكل عملية، وكانت كل منهما ترغب في طباعة ملفها باستخدام الطابعة. تطلب العملية الأولى الطابعة وستتحصل عليها، ثم تبدأ في حساب القيم المراد طباعتها، بفرض أن الفترة الزمنية المخصصة للعملية قد أُستنفذت قبل أن تنتهي هذه العملية من حساباتها، الأمر الذي سيؤدي إلى عملية مبادلتها.

العملية الثانية تشتغل الآن وستحاول، دون جدوى، الحصول على الطابعة، على الأرجح، لدينا الآن حالة الجمود، وذلك لأن لدى الأولى الطابعة ولدى الثانية الذاكرة، ولا يُمكن لأي منهما أن تمضي دون أن تتحصل على المصدر الآخر والمحجوز من قبل العملية الأخرى. لحسن الحظ، من الممكن سحب الذاكرة من العملية الثانية عن طريق المبادلة ومبادلتها بالعملية الأولى، والتي بإمكانها الآن أن تُواصل عملها وتطبع بياناتها ومن ثم تحريرها، الأمر الذي لن يؤدي إلى حدوث حالة الجمود.

2.3.4 المصادر غير القابلة للسحب

تتمثل المصادر غير القابلة للسحب في تلك المصادر، التي سحبها من المالك الحالي لها سيؤدي إلى فشل العمليات الحسابية أو حدوث أخطاء مثلًا، إذا بدأت عملية في حرق بيانات على القرص المدمج وفجأةً سُحبت سواقة الأقراص المدمجة من هذه العملية وخُصصت إلى عملية أخرى، سيؤدي ذلك حتمًا إلى تشوه القرص وحدوث أخطاء. مثل هذه المصادر تُعتبر من أمثلة المصادر غير القابلة للسحب. بشكل عام، تشمل حالة الجمود التعامل مع المصادر من هذا النوع، بينما يُمكن حل حالة الجمود الناتجة عن المصادر القابلة للسحب عن طريق إعادة توزيع

المصادر بين العمليات. بناءً عليه، سيتم التركيز على الجمود الناتج عن المصادر غير القابلة للسحب.

بالنظر إلى تسلسل الأحداث اللازمة لاستخدام أي مصدر في النظام فإنه يجب اتباع الخطوات التالية:

1. طلب المصدر: إذا توفر المصدر، يُخصص مباشرة للعملية الطالبة له، وإلَّا ستنتظر تحرره.
2. استخدام المصدر: في حالة حصول العملية على المصدر، يجب عليها استخدامه.
3. تحرير المصدر: فور إنتهاء العملية من تنفيذ مهمتها، يجب عليها تحرير المصدر، أي إعادته.

في حالة عدم توفر المصدر لحظة طلبه، فإنه يتحتم على العملية الطالبة له الانتظار إلى أن يتحرر المصدر المطلوب، ومن ثم تُنفذ، في بعض من نظم التشغيل تُوقف العملية تلقائيًا، بينما في البعض الآخر تصدر رسالة خطأ، أو قد يبقى الخيار متاحًا للعملية بين الانتظار قليلًا وبين محاولة الكرة من جديد إلى حين تحرر المصدر المطلوب، وذلك من خلال دخولها في حلقة محكمة تتبادل فيها حالتي الانتظار وطلب المصدر.

3.3.4 إدارة استخدام المصادر

يُترك الأمر لعمليات المستخدم في إدارة استخدام المصادر بنفسها وخاصةً لبعض من أنواع المصادر كالسجلات في نظام قاعدة البيانات. تتمثل هذه الإدارة في تخصيص متغير إشارة[15] لكل مصدر بحيث تُهيأ مبدئيًا كافة الإشارات بالقيمة 1 وتُستخدم في تنفيذ الخطوات الثلاث (طلب، واستخدام، وتحرير المصدر) كعملية واحدة غير قابلة للمقاطعة. هذه الخطوات موضحة في الشكل 4. 2–أ.

[15] راجع الفصل الثاني، القسم 5.3.2 لمطالعة متغير الإشارة.

أما إذا احتاجت العملية إلى استخدام مصدرين أو أكثر، يمكنها الحصول عليهما بشكل تتابعي واحدًا تلو الآخر، كما هو مبين في الشكل 4. 2–ب. هذه الآلية تجعل الأمور تسير بشكل جيد، طالما هناك عملية واحدة فقط، لأنه لا يوجد أي منافس لها.

```
typedef int semaphore;
semaphore resource_1;
semaphore resource_2;

void process_A(void} {
     down(&resource_1);
     down(&resource_2);
     use_both_resources();
     up(&resource_2);
     up(&resource_1);
}
```

(ب)

```
typedef int semaphore;
semaphore resource_1;

void process_A(void) {
     down(&resource_1);
     use_resource_1();
     up(&resource_1);
}
```

(أ)

الشكل 4. 2: استخدام متغير الإشارة لاكتساب المصادر– أ) مصدر واحد– ب) مصدران [Tanenbaum & Bos, 2015].

بالمقابل يصف الشكل 4. 3 طريقتين للتعامل مع الوضعيات التي تتواجد فيها عمليتين P_1 و P_2، مع وجود مصدرين. في الشكل 4. 3–أ تطلب كلتا العمليتان المصدرين بنفس الترتيب، وبترتيب مختلف في الشكل 4. 3–ب. قد يبدو هذا اختلافًا بسيطًا، إلاَّ إنَّه في الحقيقة ليس كذلك. نلاحظ في الشكل 4. 3–أ أن إحدى العمليتين ستكتسب المصدر الأول قبل الأخرى، وهي نفس العملية التي ستكتسب بنجاح المصدر الثاني وستقوم بعملها، وإذا ما حاولت العملية الأخرى الحصول على المصدر الأول قبل أن يُفرج عنه، فستُوَقَّف ببساطة إلى حين تحرر هذا المصدر.

يختلف الوضع في الشكل 4. 3–ب، فقد يحدث أن تستحوذ إحدى العمليتان على المصدرين، الأمر الذي يؤدي إلى توقف العملية الأخرى بالفعل إلى أن تُنهي العملية التي استحودت على المصدرين عملها. ومع ذلك، فإنه من الممكن أن تستحوذ العملية P_1 على المصدر الأول والعملية P_2 على المصدر الثاني، هذا يعني الآن: أنَّ لكل واحدة منهما القدرة على إيقاف الأخرى عندما تُحاول أيٍ منهما الحصول على المصدر الآخر، سيؤدي هذا الوضع

حتمًا إلى عدم اكتمال عملهما، وإلى ما يُعرف بحالة الجمود.

```
typedef int semaphore;
   semaphore resource_1;
   semaphore resource_2;

   void process_A(void) {
      down(&resource_1);
      down(&resource_2);
      use_both_resources();
      up(&resource_2);
      up(&resource_1);
   }

   void process_B(void) {
      down(&resource_2);
      down(&resource_1);
      use_both_resources();
      up(&resource_1);
      up(&resource_2);
   }
```

(ب)

```
typedef int semaphore;
    semaphore resource_1;
    semaphore resource_2;

    void process_A(void) {
       down(&resource_1);
       down(&resource_2);
       use_both_resources();
       up(&resource_2);
       up(&resource_1);
    }

   void process_B(void) {
      down(&resource_1);
      down(&resource_2);
      use_both_resources();
      up(&resource_2);
      up(&resource_1);
   }
```

(أ)

الشكل 4. 3: أ) شفرة خالية من حالة الجمود– ب) احتمالية حدوث حالة الجمود

.[Tanenbaum & Bos, 2015]

هنا نرى كيف أن اختلاف بسيط في أسلوب اكتساب المصادر قد يؤدي إلى عمل البرنامج أو توقفه، لذلك ولأن الجمود يمكن أن يحدث بسهولة، هناك الكثير من الأبحاث تطرقت لهذه المشكلة وإلى كيفية التعامل معها.

4.4 توصيف حالة جمود المصادر

من البديهي أن عملية الجمود حالة غير مرغوب فيها، لأنها لا تُمكِّن العمليات من إنهاء تنفيذها مطلقًا، وتتسبب كذلك في جعل مصادر النظام محجوزة إلى الأبد، وتمنع وظائف أخرى من البدء في العمل. كل هذا يجعل من المهم جدًا توصيف هذه الحالة بشكل واضح والتعرف على البيئة الملائمة لها، والتي تساعد في حدوثها، وهو ما سيُساهم بشكل مباشر في وضع الحلول المناسبة لهذه المشكلة، أو على الأقل تقليل الأضرار الناجمة عنها.

1.4.4 الشروط الضرورية لحدوث حالة جمود المصادر

في عام 1971 بين كوفمان وآخرون (Coffman et al., 1971) أن هناك بيئة ملائمة لحدوث حالة الجمود تتمثل في أربعة شروط ضرورية لتحقق جمود المصادر، وأن تعذر تحقق أي واحدٍ منها سيمنع حدوثه. لاحقًا سنرى كيف يمكن منع حدوث هذه الحالة من خلال محاولة منع تحقق بعض من هذه الشروط، والمتمثلة في:

- شرط المنع التبادلي: أي مصدر إمَّا أن يكون حاليًا مخصص بالضبط لعملية واحدة بشكل حصري، أو أن يكون متاحًا للعمليات الأخرى.
- شرط الاحتجاز والانتظار: العمليات التي حاليا تمتلك مصادر– مسندة لها في وقت سابق– لها الحق في طلب مصادر جديدة.
- شرط عدم حق السحب: يُمنع سحب المصادر المسندة لإحدى العمليات قسرًا، وهي الوحيدة المخولة بتحريرها وإعادتها إلى النظام.
- شرط الانتظار الدائري: أي أن تكون هناك حلقة دائرية مكونة من عمليتين أو أكثر من العمليات، كل منها تنتظر في مصدر مسنود إلى العملية التي تليها في الحلقة.

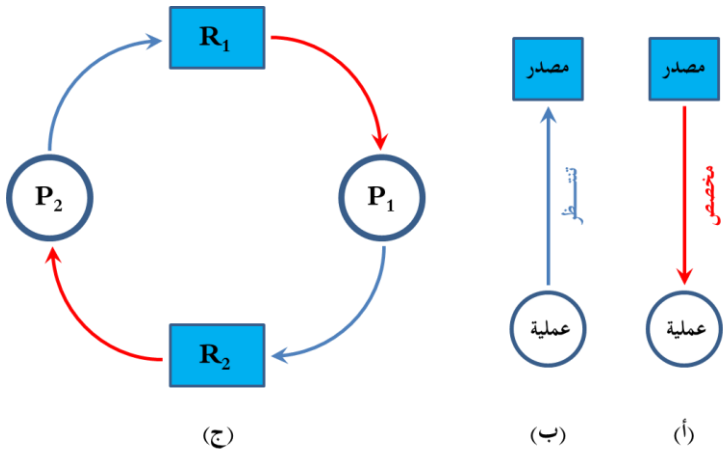
من الجدير بالذكر أن كل شرط من هذه الشروط الأربعة يتعلق بالسياسة المتبعة للنظام، بمعنى هل لدى النظام رؤية واضحة للأسئلة التالية:

- هل يمكن إسناد مصدر– ما– لأكثر من عملية في وقت واحد؟
- هل يمكن لعملية– ما– مُسند لها مصدر، طلب مصدر آخر؟
- هل يمكن سحب المصدر من العملية قسراً؟
- هل يمكن تحقق الانتظار الدائري؟

كل هذه الأسئلة وغيرها من المواضيع ذات الصلة بحالة الجمود ستُناقش في بقية هذا الفصل.

2.4.4 نمذجة حالة الجمود

لتوضيح حالة الجمود بشكل أفضل أُستخدمت الرسوم البيانية الموجهة[16] لنمذجة الشروط الأربعة السالفة الذكر. ففي عام 1972 أظهر هولت (Holt 1972) الكيفية التي يُمكن بها تمثيل هذه الحالة باستخدام هذه الرسوم، والتي احتوت على ثلاثة أشكال هندسية، وذلك كما هو موضح في الشكل 4. 4. يُمثل المربع في هذا الشكل المصدر، والدائرة العملية، بينما يعني السهم المتجه من المصدر إلى العملية: أن المصدر طُلب من قبل العملية مسبقًا، وتمت الموافقة على الطلب وهو حاليًا مخصص لها (الشكل 4. 4–أ).



الشكل 4. 4: نمذجة الجمود– أ) إسناد المصدر– ب) طلب المصدر– ج) حلقة الجمود.

أما السهم المتجه من العملية إلى المصدر فهو يعني: أن العملية قد تقدمت بطلب لاكتساب المصدر وقد رُفِض الطلب، لأنه مخصص لعملية أخرى وهي الآن موقوفة وتنتظر في تحرير ذلك المصدر، كما هو مبين في الشكل 4. 4–ب. بينما نرى بوضوح في الشكل 4. 4– ج حلقة حالة الجمود، والتي من خلالها نلاحظ أن العملية P_1 مُخصص لها المصدر R_1 وتنتظر

[16] الرسم البياني الموجه هو رسم بياني تكون فيه اتجاهات الأسهم محددة وفي اتجاه واحد فقط.

في تحرر المصدر R_2 والمسنود حاليًا للعملية P_2، والتي بدورها تنتظر في تحرر المصدر R_1. بالتالي فإن العمليتين ستنتظران إلى الأبد. في العموم تواجد أي حلقة داخل الرسم البياني الموجه لكل من العمليات، والمصادر يدل على وجود حالة جمود المصادر، (على افتراض أن هناك مصدر واحد من كل نوع)، في هذا المثال تكون الحلقة على هيئة $P_1 - R_1 - P_2 - R_2 - P_1$.

أمثلة توضيحية:

تُوضح الأمثلة التالية مجموعة من حالات النظام المتعددة والممثلة بتخصيص المصادر عن طريق الرسومات البيانية الموجهة لغرض التعرف على كيفية استخدامها في تحديد إمكانية حدوث حالة جمود المصادر من عدمها.

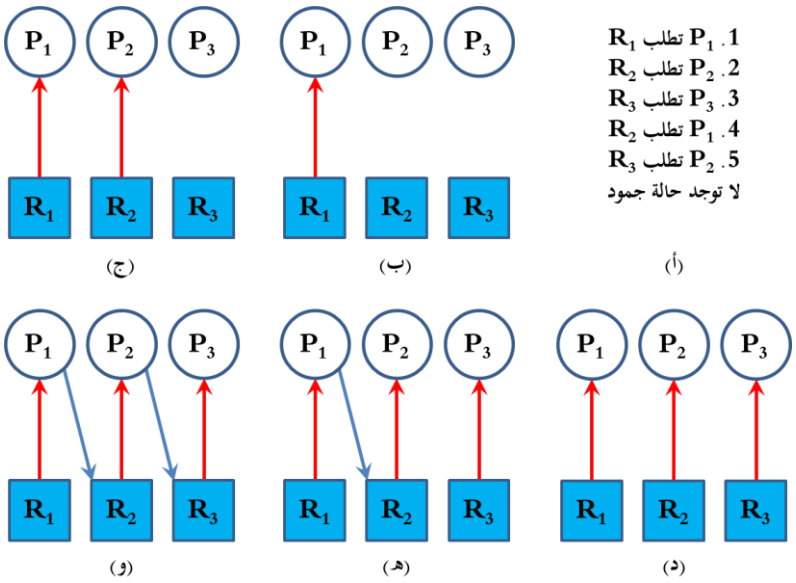
المثال الأول: بفرض أن هناك ثلاث عمليات P_1، P_2 و P_3، وثلاثة أنواع من المصادر R_1، R_2 و R_3، وكان طلب وتحرير هذه المصادر على النحو الموضح في الشكل 4. 5، وكان لنظام التشغيل الحرية في اختيار تنفيذ أي عملية في أي لحظة وبشكل تتابعي. عند اختيار تنفيذ هذه العمليات بحيث تُنفذ P_1 أولًا حتى النهاية، ثم P_2 ثانيًا حتى النهاية، وأخيرًا P_3 حتى النهاية، فلن تكون هناك أي حالة جمود، لأنه لا يوجد أي تنافس على المصادر، كما أنه ليس هناك أي توازي في التنفيذ على الإطلاق.

P_1	P_2	P_3
تطلب R_1	تطلب R_2	تطلب R_3
تطلب R_2	تطلب R_3	تطلب R_1
تُحرر R_1	تُحرر R_2	تُحرر R_3
تُحرر R_2	تُحرر R_3	تُحرر R_1
(أ)	(ب)	(ج)

الشكل 4. 5: التنفيذ التتابعي للعمليات.

إلَّا إنَّ تشغيل العمليات بشكل تتابعي، لا يُساعد في استغلال وحدة المعالجة المركزية من قبل عمليات أخرى طالما هناك من ينتظر في عمليات إدخال، أو إخراج، بالتالي قد لا يكون تنفيذ العمليات على نحو تتابعي بالحل الأمثل، إلَّا إنَّه في بعض من الظروف إذا لم تكن هناك عمليات إدخال، أو إخراج، فإن تنفيذ جميع العمليات بالتتابع أفضل وسيلة.

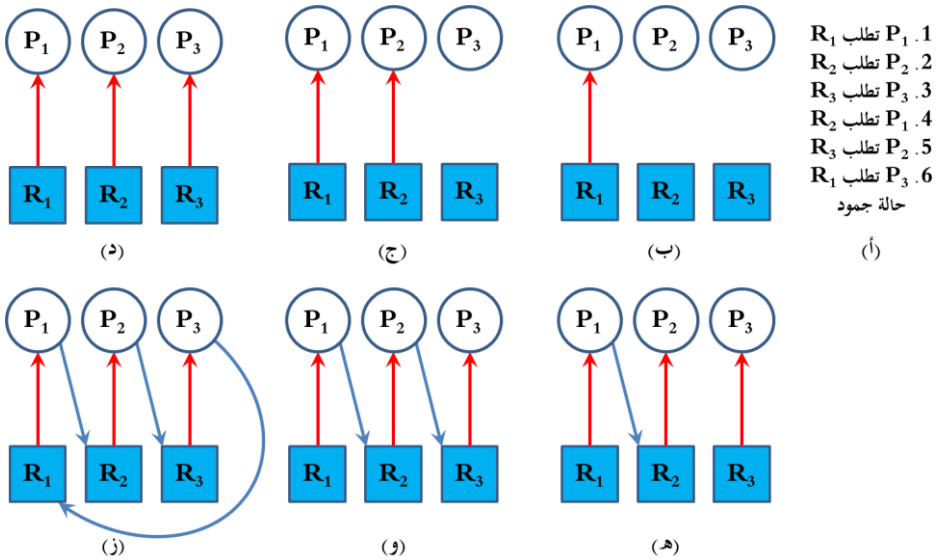
المثال الثاني: بفرض أن طلب العمليات الثلاث السابقة للمصادر R_1، R_2 و R_3 كان على النحو الموضح في الشكل 4. 6-أ. في البداية قامت العملية P_1 بطلب المصدر R_1 وخُصص لها، بالمثل قامتا العمليتان P_2 و P_3 بطلب المصدرين R_2 و R_3، على التوالي وخُصص لهما. ولكي تُكمل كل من العملية P_1 و P_2 عملهما احتاجتا إلى المصدرين R_1 و R_3، على التوالي، بالتالي سيتقدما بطلب إلى نظام التشغيل لاكتسابهما، ولأن هذان المصدران مُخصصان لعمليات أخرى، فسيُرفض طلبهما ويُعلقا إلى حين تحرر هذين المصدرين. هذه الطلبات ممثلة في باقي مكونات الشكل 4. 6. نلاحظ في الحالة المعروضة في الشكل 4. 6- و أنَّ النظام لم يدخل في حالة جمود المصادر وأنَّ العملية P_3 ليست في حالة الاحتجاز والانتظار، على العكس من P_1 و P_2، وهو ما يعني أنَّ العمليات الثلاث لم تُحقق شرط الانتظار الدائري.



الشكل 4. 6: استخدام الرسومات البيانية الموجهة في اكتشاف حدوث جمود المصادر من عدمه.

نظرًا لأن عدد المصادر محدود وغير كافي لتنفيذ العمليات الثلاث في وقت واحد في حالة النظام الموضحة في هذا المثال، عليه يجب إيجاد آلية لتنفيذ هذه العمليات. بما أن العملية P_3 لا تطلب أي مصدر آخر، فإنه يُمكن المضي قُدمًا في تنفيذها على الفور، عاجلًا أم آجلًا

ستنتهي، وستُحرر المصدر R_3، وعندها بالإمكان تخصيصه للعملية P_2، لكي تُكمل مهمتها وتُحرر المصدر R_2. يأتي الآن الدور على العملية P_1 لتنفيذها بالكامل بعد اكتسابها لجميع المصادر المطلوبة.

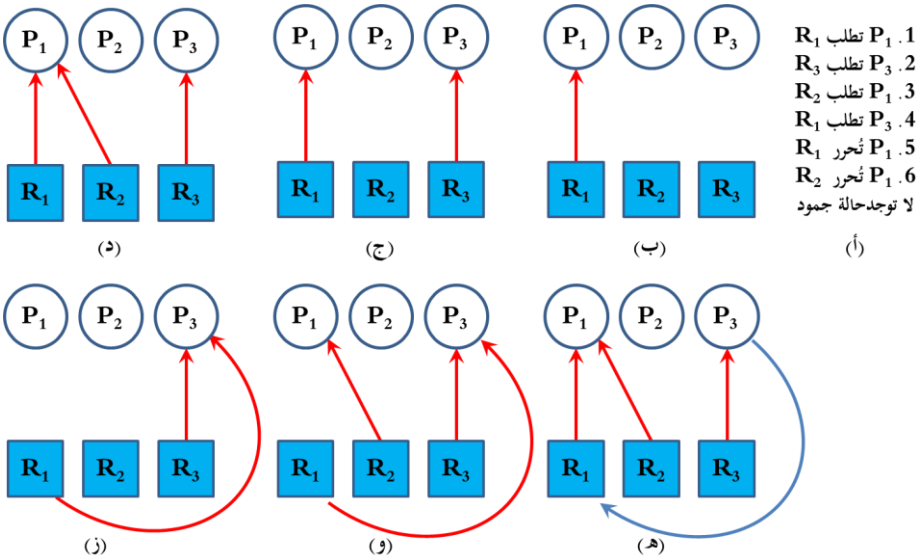


الشكل 4. 7: توضيح حالة حدوث جمود المصادر.

المثال الثالث: يُبين الشكل 4. 7 مثالًا آخرًا لحالة نظام تختلف قليلًا عن الحالة الموضحة في المثال السابق. يتمثل الاختلاف الرئيسي في قيام العملية P_3 بطلب المصدر R_1. بالتالي إذا نُفذت هذه العمليات بالترتيب الموضح في الشكل 4. 7–أ، فسيؤدي ذلك إلى المرور بعدة حالات ممثلة بالرسوم البيانية للمصادر المبينة في الشكل 4. 7–ب إلى –ز، بعد أن يُلبى الطلب الرابع، ستتوقف P_1 انتظارًا للمصدر R_2، كما هو مبين في الشكل 4. 7–هـ بالمثل، في الخطوتين التاليتين ستتوقف كل من P_2 و P_3، الأمر الذي يؤدي– في نهاية المطاف– إلى إنشاء حلقة جمود المصادر، مثلما هو موضح في الشكل 4. 7–ز. يكون في هذا الشكل شرط الانتظار الدائري صحيحًا للعمليات الثلاث نظرًا لوجود حلقة في الرسم البياني تبعًا للأسهم التي تُمثل عمليات تخصيص المصادر وطلبها.

المثال الرابع: يُوضح هذا المثال محاولة منع حدوث جمود المصادر من قبل نظام التشغيل

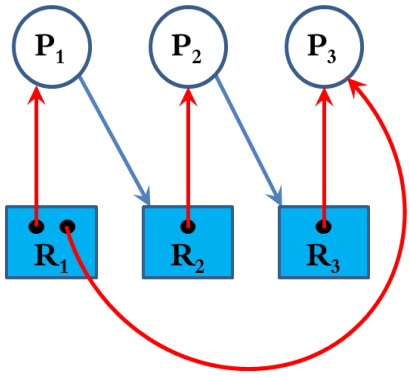
عن طريق اختيار تسلسل تنفيذ العمليات بطريقة مغايرة للمثال السابق. على وجه الخصوص، إذا لُوحظ أن تنفيذ طلب معين قد يؤدي إلى إنشاء حلقة الجمود، فلنظام التشغيل الأحقية في تعليق هذا الطلب (أي، عدم جدولة العملية الطالبة للمصدر) إلى أن تصبح الحالة آمنة. في الشكل 4. 7، يتجه النظام إلى تعليق طلب P_2 بدلًا من منحها R_2، وتنفيذ كل من طلب P_1 و P_3 على النحو المبين في الشكل 4. 8-أ. يؤدي هذا التسلسل في التنفيذ إلى الرسوم البيانية للمصادر الممثلة في الشكل 4. 8-ب إلى -ز، التي لا تؤدي إلى الجمود. بعد الخطوة (ز)، يمكن منح المصدر R_2 للعملية P_2، لأن P_1 أنهت عملها و P_3 أخذت كل ما تحتاجه.



الشكل 4. 8: تغيير تسلسل تنفيذ العمليات لتجنب حدوث جمود المصادر.

المثال الخامس: في الأمثلة السابقة أُفترِض أن هناك مصدر واحد من كل نوع. ولكن، ماذا لو امتلك النظام عدة مصادر من نفس النوع، كما هو موضح في هذا المثال. يُبين الشكل 4. 9 مثالًا لحالة نظام يحتوي على مصدرين من R_1 ومصدر واحد من كل من R_2 و R_3. نلاحظ وجود حلقة في هذا الشكل ولكن لن تؤدي إلى حدوث حالة الجمود مقارنة بالشكل 4. 7-ز، يتمثل الاختلاف الرئيسي هنا في أنه عند قيام العملية P_3 بطلب المصدر R_1 فسيُخصص في هذه الحالة المصدر الثاني من R_1 إلى هذه العملية دون أي انتظار لكونه حر. بالتالي، يُمكن بدء تنفيذ العملية P_3 على الفور لتُنهي بعد ذلك عملها ويتحرر المصدر R_3. هذا الأمر سيؤدي إلى

تكسير الحلقة وعدم حدوث حالة الجمود وسيكون تسلسل بقية الأحداث مشابه إلى حد كبير مع تلك التي نُوقشت سابقًا. للاطلاع أكثر على كيفية التعامل مع عدة مصادر من نفس النوع يُمكن مراجعة هولت (Holt 1972).



الشكل 4. 9: امتلاك النظام لعدة مصادر من نفس النوع يُساهم في تجنب حدوث حالة جمود المصادر.

سندرس لاحقًا في هذا الفصل بنوع من التفصيل بعض من الخوارزميات الخاصة باتخاذ قرارات التخصيص التي لا تؤدي إلى جمود المصادر. يكفي في الوقت الحالي أن نعرف أن الرسوم البيانية للمصادر هي أداة تُتيح لنا معرفة ما إذا كان تتبع الطلب وتحرير المصدر يؤدي إلى جمود المصادر أم لا، الأمر متعلق فقط بتنفيذ الطلبات وتحرير المصادر خطوة بخطوة، وبعد كل خطوة نتحقق من الرسم البياني، لمعرفة ما إذا كان الرسم يحتوي على أية حلقة. إذا كان الأمر كذلك، فهذا يعني أن هناك حالة جمود المصادر، وإذا لم يكن كذلك، فلا وجود له.

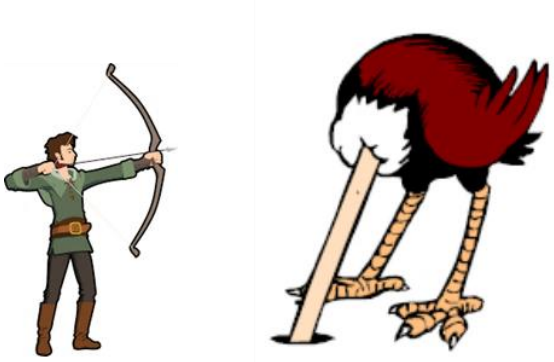
بشكل عام، هناك أربع استراتيجيات يمكن من خلالها التعامل مع حالة جمود المصادر:

1. فقط تجاهل المشكلة: وذلك استنادًا إلى فكرة: لو تجاهلت المشكلة، ستتجاهلك هي.
2. الاكتشاف والاسترداد: أي السماح بحدوث جمود المصادر، واكتشافه، ومن ثم التعافي منه.
3. التجنب الحيوي لجمود المصادر: أي تخصيص المصادر بعناية.
4. المنع: أي محاولة عدم السماح بتحقق أحد الشروط الأربعة السابقة.

وسنتطرق الآن من خلال الأقسام الأربعة التالية إلى هذه الطرق بنوع من التفصيل.

5.4 خوارزمية النعامة

أبسط طرق التعامل مع حالة الجمود تتمثل في تجاهلها والتظاهر بعدم وجودها على الإطلاق استنادًا إلى فكرة ما تقوم به النعامة من وضع رأسها في الرمال وإهمال كل ما يحدث حولها، كما هو موضح في الشكل 4. 10، هذا المفهوم هو في الأساس استرتيجية أكثر منه خوارزمية. على الرغم من أن هذه الاسترتيجية قد لا تبدو نهجًا قابلًا للتطبيق لمشكلة الجمود، إلَّا إنَّها مستخدمة في معظم أنظمة التشغيل، بما في ذلك 'لينكس' و'ويندوز'، لكن سيضل الأمر متعلق بمطور التطبيق لكتابة البرامج التي تتعامل مع حالة الجمود.



الشكل 4. 10: استراتجية النعامة.

تجاهل احتمال وجود حالة الجمود قد يكون أرخص الاستراتجيات المذكورة سابقًا، وخصوصًا عندما تكون حالة الجمود ناذرة الحدوث (مثلًا، مرة في السنة، أو بمتوسط مرة واحدة كل خمس سنوات)، كذلك عندما لا تكون هناك عواقب وخيمة ناتجة من حدوثه، أو عندما يكون من الصعب اكتشافه وأن تكلفة ذلك ستكون باهضة، كل هذه الأسباب تجعل تطبيق سياسة خوارزمية النعامة هو الحل الأفضل.

استجابة الناس لهذه الاستراتيجية في الحقيقة تتفاوت من فئة إلى أخرى، حيث تجد عالم الرياضيات يرى أن هذا الحل غير مقبول البتة، ويقول أنَّ حالة الجمود يجب منعها بأي ثمن، في حين يتساءل المهندس، كم مرة من المتوقع حدوث المشكلة؟ كم مرة تعطَّل النظام لأسباب أخرى؟ ما مدى خطورة الجمود؟ بالتالي إذا كان تعطل النظام بسبب الجمود بمتوسط أقل بكثير من انهيار النظام بسبب أعطال أخرى مثل، فشل في العتاد المادي، أو حدوث أخطاء برمجية في

شفرة نظام التشغيل، فإن أغلب المهندسين لا يملكون الرغبة في دفع غرامة كبيرة من أجل القضاء على هذه الحالة.

هنا، يجب الأخذ في عين الاعتبار أنه في حالة غياب خوارزميات للكشف عن حالة الجمود والتعافي منه قد يصل النظام إلى حالة التوقف التام، وأنه لا توجد وسيلة للتعرف على ما حدث. سيتسبب الجمود غير المكتشف في هذه الحالة في تدهور أداء النظام، لأن المصادر بدأت تحتفظ بها العمليات التي لا يمكن تشغيلها، ولأن المزيد والمزيد من العمليات ستتقدم بطلبات للحصول على هذه المصادر، مما يعني مشاركتها في حالة الجمود، وسيتوقف النظام عن العمل في نهاية المطاف، وسيتَعيَّن إعادة تشغيله يدويًا.

6.4 استراتجية الاكتشاف والتعافي من الجمود

تتمثل الطريقة الثانية المستخدمة في حل مشكلة الجمود في الاكتشاف والتعافي منه، هذا يعني أن النظام لن يحاول منع حدوث الجمود، ولكن بدلًا من ذلك سيسمح بحدوثه، ومن ثم يحاول اكتشافه إذا حدث، بالتالي يجب على النظام أن يُوفر:

- طرق لتفحص حالة النظام، وتحديد ما إذا كان الجمود قد حدث أم لا.
- طرق للتعافي من الجمود، واسترداد حالة النظام.

سنناقش هذين المطلبين في القسمين التاليين في إطار امتلاك النظام لنسخة واحدة فقط من كل نوع من أنواع المصادر، بالإضافة إلى الأنظمة ذات الحالات المتعددة لكل نوع من أنواع المصادر. في هذه الإستراتجية يُمكن ملاحظة أن نظامي الاكتشاف والاسترداد يتطلبا نفقات إضافية تشمل كل من تكاليف زمن التشغيل للحفاظ على المعلومات الضرورية وتنفيذ خوارزمية الكشف، وكذلك الخسائر المحتملة الكامنة في التعافي من حالة الجمود.

1.6.4 اكتشاف الجمود في وجود مصدر واحد من كل نوع

لتبسيط مفهوم اكتشاف حالة الجمود سنبدأ مع أبسط الحالات والمتمثلة في الأنظمة التي تتعامل مع مصدر واحد من كل نوع، أي تلك التي تمتلك مثلًا، ماسح ضوئي، ومسجل للأقراص المضغوطة، وراسمة، ومحرك للأقراص الممغنطة، أي مصدر واحد من كل فئة من فئات المصادر

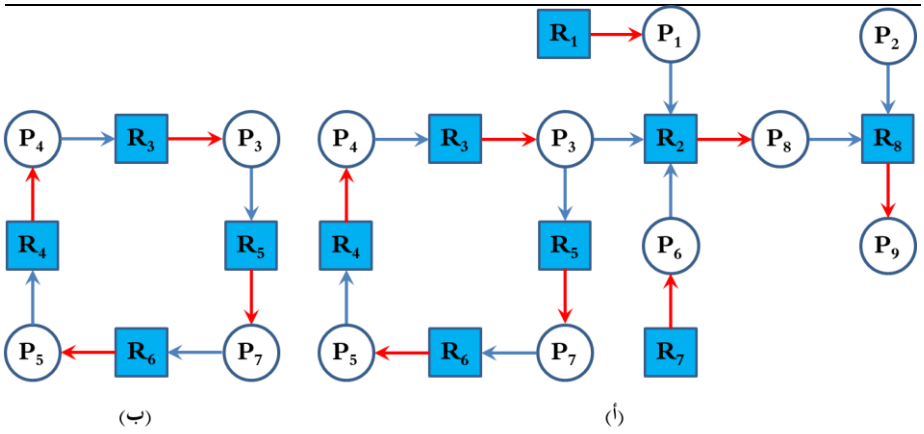
المتنوعة. بعبارة أخرى سنستثني في هذا القسم الأنظمة المتصلة مثلًا، بطابعتين، أو راسمتين والتي سنتعامل معها في وقت لاحق باستخدام تقنيات مختلفة.

في الأنظمة التي تتعامل مع مصدر واحد من كل نوع يُمكننا بناء رسم بياني موجه للمصادر من النوع المبين في الشكل 4. 11 (هنا يجب أن نتذكر أننا افترضنا وجود مصدر واحد من كل نوع وهو غالبًا ما يكون غير صحيح، لأن العديد من الأنظمة تحوي– على سبيل المثال– عدة طابعات، وأنَّ تقديم طلب للطباعة ليس مقصود به طابعة معينة). بالعودة إلى الرسم البياني، إذا احتوى الرسم حلقة أو أكثر، فهذا يعني وجود حالة جمود، وأنَّ أي عملية مشاركة في هذه الحلقة، فهي عملية موقوفة بسببه، بينما خلاف ذلك يعني لا وجود لأي حالة جمود.

كمثال على نظام أكثر تعقيدًا من تلك الأمثلة التي تعرضنا لها في السابق، يمكن النظر إلى نظام ذو تسع عمليات، P_1 إلى P_9، وثمانية مصادر، R_1 إلى R_8، حالة المصادر التي حاليًا مخصصة، وتلك التي حاليًا مطلوبة هي كما يلي:

1. العملية P_1 مُخصص لها R_1 وترغب في الحصول على R_2.
2. العملية P_2 ليس مُخصص لها أي مصدر، لكنها ترغب في الحصول على R_8.
3. العملية P_3 مُخصص لها R_3، وترغب في الحصول على R_2 و R_5.
4. العملية P_4 مُخصص لها R_4 وترغب في الحصول على R_3.
5. العملية P_5 مُخصص لها R_6 وترغب في الحصول على R_4.
6. العملية P_6 مُخصص لها R_7 وترغب في الحصول على R_2.
7. العملية P_7 مُخصص لها R_5 وترغب في الحصول على R_6.
8. العملية P_8 مُخصص لها R_2 وترغب في الحصول على R_8.
9. العملية P_9 مُخصص لها فقط R_8.

السؤال المطروح هو: هل وصل هذا النظام إلى حالة الجمود؟ إذا كان الأمر كذلك، أيٍ من هذه العمليات تتشارك فيه؟



الشكل 4. 11: أ) رسم بياني موجه للمصادر– ب) حلقة مستخلصة من أ.

للإجابة على هذا السؤال، يُمكننا بناء الرسم البياني الموجه للمصادر، كما هو موضح في الشكل 4. 11–أ. يحتوي هذا الرسم على حلقة واحدة يُمكن ملاحظتها عن طريق الفحص البصري، والمبينة في الشكل 4. 11–ب. من هذه الحلقة، يُمكن ملاحظة أن العمليات P_3، P_4، P_5 و P_7 موقوفة بسبب حالة الجمود، بينما لن تتوقف بقية العمليات بسببه، لأن هناك مجال لتخصيص المصادر المطلوبة لها وإتمام عملها المناط بها.

على الرغم من بساطة عملية انتقاء العمليات الموقوفة بالعين من الرسم البياني الموجه البسيط، إلَّا إنَّنا نحتاج في النظم الفعلية إلى خوارزمية متخصصة للكشف عن حالة الجمود. هناك عدة خوارزميات للكشف عن الحلقات في الرسوم البيانية الموجهة، أدناه سوف نعطي مثالًا بسيطًا لإحدى الخوارزميات التي تتفقد الرسم البياني للمصادر، وتنتهي إمَّا بوجود حلقة أو عندما يتبين لها عدم وجودها.

تستخدم هذه الخوارزمية قائمة بيانات ديناميكية– L–، وقائمة للعقد، فضلًا عن قائمة للأسهم. خلال تنفيذ الخوارزمية، ستُوضع علامة على الأسهم التي فُحصت بالفعل، وذلك لمنع عمليات الاختبار المتكررة. تعمل الخوارزمية من خلال تنفيذ الخطوات التالية على النحو المحدد:

1. لكل عقدة N في الرسم البياني، نفذ الخطوات الخمس التالية، علمًا بأن N هي عقدة

البداية.

2. هيئ **L** على أساس قائمة خالية، وعين جميع الأسهم على أساس غير معلمة.
3. أضف العقدة الحالية إلى نهاية **L**، وتحقق لمعرفة ما إذا ظهرت هذه العقدة مرتين في **L**. إذا كان الأمر كذلك، فهذا يعني أن الرسم البياني يحتوي على حلقة (مدرجة في **L**)، وستتوقف الخوارزمية.
4. انطلاقًا من أي عقدة معطاة، تَعرف على ما إذا كان هناك أي سهم صادر لا يحمل علامة. إذا كان الأمر كذلك، انتقل إلى الخطوة **5**، إذا لم يكن كذلك، انتقل إلى الخطوة **6**.
5. اختر عشوائيًا أحد الأسهم الصادرة وضع علامة عليه، ثم إتَّبِعه باتجاه العقدة الجديدة، وانتقل إلى الخطوة **3**.
6. إذا كانت هذه العقدة هي العقدة الأولية، فهذا يعني أن الرسم البياني لا يحتوي على أي حلقة، وستنتهي الخوارزمية. خلاف ذلك، نجد أننا وصلنا الآن إلى نهاية قاتلة، قم بإزالتها، ثم ارجع إلى العقدة السابقة، واجعلها العقدة الحالية، وانتقل إلى الخطوة **3**.

الشيئ الذي تفعله هذه الخوارزمية هو أخذ كل عقدة بالتبادل وجعلها جذر لما تأمل أن تُكوِّن شجرة، ثم تُطبق البحث الأولي المتعمق عليها، إذا مرت الخوارزمية بعقدة واجهتها في السابق، فهذا يعني أن هناك حلقة. إذا وُضِعت علامة على كافة الأسهم الصادرة من أي عقدة، فسترجع الخوارزمية إلى العقدة السابقة، علمًا أنه إذا تم الرجوع إلى الجذر ولم يكن بالإمكان الذهاب إلى أبعد من ذلك، فسيدل ذلك على عدم وجود حلقة داخل هذا الرسم البياني الثانوي، الذي وُصِل إليه من العقدة الحالية، في حالة صلاح هذه النتيجة لكافة العقد، فسيكون الرسم البياني خالي من أي حلقة، بالتالي لن تكون هناك حالة الجمود داخل هذا النظام.

والآن لنرى الكيفية التي تعمل بها الخوارزمية من الناحية العملية من خلال تطبيقها على الرسم البياني الموجه الموضح في الشكل **4. 11**-أ. ترتيب معالجة هذه العقد سيتم بشكل اختياري، لذلك ستُفحص من اليسار إلى اليمين، من أعلى إلى أسفل، وسيبدأ تنفيذ الخوارزمية من المصدر $\mathbf{R_1}$، ثم تباعًا $\mathbf{P_1}$، $\mathbf{P_2}$، $\mathbf{P_4}$، $\mathbf{R_3}$، $\mathbf{P_3}$، $\mathbf{R_2}$، $\mathbf{P_8}$، $\mathbf{R_8}$، وهكذا، علمًا بأنَّ الخوارزمية ستتوقف إذا واجهت أي حلقة في أثناء عملية الفحص.

كما أشرنا سابقًا، ستبدأ الخوارزمية بالمصدر $\mathbf{R_1}$ وستُهيأ القائمة **L**، ويُضاف إليها $\mathbf{R_1}$، ثم

تنتقل إلى الاحتمال الوحيد، P_1، وتُضيفه إلى L، الأمر الذي يجعل $L = [R_1, P_1)$، من P_1 تذهب الخوارزمية إلى R_2، لتعطي $L = [R_1, P_1, R_2]$، ومن ثم تذهب إلى P_8 ومنها إلى R_8 و P_9، لتعطي $L = [R_1, P_1, R_2, P_8, R_8, P_9]$، نلاحظ أنه ليس هناك أسهم صادرة من P_9، لذلك فهي نهاية قاتلة تُجبر الخوارزمية على التراجع إلى P_1. ولأن P_1 لا يوجد لها أسهم صادرة تحمل علامات، ستتراجع إلى R_1، وستنتهي عملية الفحص الخاصة بالمصدر R_1. سيُعاد الآن تشغيل الخوارزمية بدءًا من P_1، ويُعادة تهيئة L، وسيتوقف هذا البحث، وستبدأ الخوارزمية من جديد انطلاقًا من P_2، وسيتوقف هذا البحث بسرعة. الآن يجب على الخوارزمية عشوائيًا اختيار بداية جديدة، والتكن P_4 ومنها ينطلق البحث الآن وتستمر الخوارزمية في اتباع الأسهم الصادرة وستُحدث الخوارزمية L لتحصل على $L = [P_4, R_3, P_3, R_5, P_7, R_6, P_5, R_4, P_4]$. عند هذه النقطة ستكون هناك حلقة وستتوقف الخوارزمية، لأنها مرت بالعقدة P_4 مرتين.

هذه الخوارزمية هي أبعد ما تكون عن المثالية، ومع ذلك فهي تُوضح لنا أنه هناك إمكانية لاكتشاف وجود حالة الجمود. يُمكنك الرجوع إلى إيفن (Even, 1979) للتعرف على خوارزميات أفضل.

2.6.4 اكتشاف الجمود في وجود عدة مصادر من كل نوع

هناك حاجة إلى استخدام طرق مختلفة للكشف عن الجمود عند وجود عدة نسخ من بعض المصادر. سيتعرض هذا القسم لخوارزمية تعتمد على استخدام المصفوفات للكشف عن حالة الجمود لعدد n من العمليات هي P_1 إلى P_n. بفرض أن عدد فئات المصادر هو m، وأن عدد المصادر في الفئة الأولى E_1، وفي الفئة الثانية E_2، وعمومًا في الفئة i يكون E_i، بحيث $(1 <= i <= m)$، وبحيث يُمثل المتجه E متجه المصادر الموجودة، وهو يُعطي– في أي لحظة– العدد الإجمالي للمصادر الحالية لكل مصدر موجود. مثلًا، إذا كانت الفئة الأولى هي فئة مشغلات الطابعات، وكانت $E_1=3$ فهذا يعني أن النظام لديه ثلاث طابعات.

في أثناء اشتغال النظام وفي أي لحظة، تكون هناك بعض من المصادر مخصصة والبعض الأخر متاح. بفرض أن A تُمثل متجه المصادر المتاحة، بحيث تُعطي A_i عدد الحالات من

المصدر $\mathbf{i}$ المتوفرة حاليًا (أي غير المخصصة) وبحيث إذا خُصصت كافة المصادر من النوع $\mathbf{i}$، فستكون $\mathbf{A_i}$ تساوي صفر.

للمضي قُدمًا في هذه الخوارزمية، نحتاج الآن إلى تعريف مصفوفتين، $\mathbf{C}$ وهي مصفوفة التخصيص الحالية، و $\mathbf{R}$ وهي مصفوفة الطلبات، حيث يُمثل الصف $\mathbf{i}$ من المصفوفة الأولى عدد الحالات لكل مصدر تمتلكه حاليًا العملية $\mathbf{P_i}$، بالتالي يُمثل $\mathbf{C_{ij}}$ عدد حالات المصدر $\mathbf{j}$ الذي بحوزة العملية $\mathbf{P_i}$. بالمثل، يُمثل $\mathbf{R_{ij}}$ عدد الحالات من المصدر $\mathbf{j}$ الذي تحتاجه $\mathbf{P_i}$. هذه الهيكليات الأربعة للبيانات تظهر جليًا في الشكل 4. 12.

تتمتع هذه الهياكل الأربعة بميزة ثابتة وهي– على وجه الخصوص– أن كل مصدر إمَّا أن يكون مخصص لعملية– ما– أو متاح وهو ما يعني أن: $\sum_i^m \boldsymbol{C_{ij}} + \boldsymbol{A_j} = \boldsymbol{E_j}$. بعبارة أخرى، إذا أُضيف عدد حالات المصادر المخصصة من النوع $\mathbf{j}$ إلى عدد كافة حالات المصادر المتاحة من النوع نفسه، ستكون النتيجة هي عدد حالات المصادر الموجودة من تلك الفئة من المصادر.

تعتمد خوارزمية الكشف عن حالة الجمود هذه على المقارنة بين المتجهات، بالتالي فإن تعريف العلاقة الرياضية ($\mathbf{B} \geq \mathbf{A}$) على المتجهين $\mathbf{A}$ و $\mathbf{B}$ تعني أن كل عنصر من عناصر $\mathbf{A}$ هو أقل من أو يساوي العنصر المماثل له من $\mathbf{B}$. رياضيًا، هذه العلاقة تكون صحيحة فقط وإذا كان فقط ($\mathbf{B_i} \geq \mathbf{A_i}$) لجميع قيم $\mathbf{i}$ المحققة للشرط ($\mathbf{m} \geq \mathbf{i} \geq \mathbf{1}$).

تُعامل كل عملية في البداية على أساس أنها غير معلمة، وكلما تقدمت الخوارزمية سيتم وضع علامات عليها، للدلالة على أنها قادرة على الاستكمال، بالتالي فهي ليست موقوفة بسبب الجمود. عند إنتهاء الخوارزمية فإن أي عملية غير معلمة ستُعتبر موقوفة بسبب الجمود. تفترض هذه الخوارزمية السيناريو الأسوأ، وهو أن كل العمليات تحتفظ بجميع المصادر المكتسبة إلى أن تنتهي العملية، وهي تتلخص في الخطوات التالية:

1. ابحث عن عملية غير معلمة $\mathbf{P_i}$ المناظرة للصف $\mathbf{i}$ في المصفوفة $\mathbf{R}$ والذي يكون أقل من أو يساوي $\mathbf{A}$.
2. إذا وجدت مثل هذه العملية، أضف الصف $\mathbf{i}$ من المصفوفة $\mathbf{C}$ إلى $\mathbf{A}$، ثم علِّم هذه العملية وانتقل إلى الخطوة 1.

3. **في حالة عدم وجود عملية من هذا القبيل، فستنتهي الخوارزمية.**

متجه المصادر الموجودة E

E_1	E_2	E_3	…	E_m

متجه المصادر المتاحة A

A_1	A_2	A_3	…	A_m

مصفوفة التخصيص الحالية C

C_{11}	C_{12}	C_{13}	..	C_{1m}	P_1
C_{21}	C_{22}	C_{23}	..	C_{2m}	P_2
:	:	:		:	:
C_{n1}	C_{n2}	C_{n3}	..	C_{nm}	P_n

الصف n حاليا مخصص للعملية n

مصفوفة الطلبات R

R_{11}	R_{12}	R_{13}	..	R_{1m}	P_1
R_{21}	R_{22}	R_{23}	..	R_{2m}	P_2
:	:	:		:	:
R_{n1}	R_{n2}	R_{n3}	..	R_{nm}	P_n

الصف n هو ما تحتاجه العملية n

الشكل 4. 12: الهياكل الأربعة للبيانات المطلوبة من قبل خوارزمية اكتشاف حالة الجمود.

عند إنتهاء الخوارزمية، تكون جميع العمليات غير المعلمة– إن وجدت– موقوفة بسبب الجمود. تبحث هذه الخوارزمية في الخطوة الأولى عن العملية التي يُمكن تشغيلها إلى النهاية، مثل هذه العملية تتميز بأنها تمتلك طلبات للمصادر يمكن تحقيقها من خلال المصادر المتاحة حاليًا في النظام، وهو ما يعني أن العملية المختارة ستستمر في التنفيذ إلى أن تنتهي، ومن ثم تُرجع المصادر المخصصة لها إلى مجموعة المصادر المتاحة، بعدها يتم وضع علامة على العملية للدلالة على أنها مكتملة. إذا كانت جميع العمليات قادرة على الاشتغال إلى النهاية، فلن تصل أيٍ منها إلى حالة الجمود، وإذا كان بعض منها لا يمكنه أبدًا الإنتهاء، فستكون هناك حالة الجمود. على الرغم من أن الخوارزمية غير حتمية[17]، إلَّا إنَّ النتيجة ستكون هي نفسها دائمًا عند تشغيل العمليات في أي ترتيب محتمل.

[17] في علم الحاسوب، الخوارزمية غير الحتمية هي خوارزمية تُظهر سلوكيات مختلفة بالنسبة لنفس المدخلات عند تنفيذها عدة مرات، وهي مقابل الخوارزمية الحتمية.

مثال توضيحي:

يوضح الشكل 4. 13 مثال لكيفية عمل خوارزمية الكشف عن حالة الجمود. في هذا المثال، لدينا ثلاث عمليات، وأربع فئات للمصادر معنونة اختياريا على النحو:مشغلات الأقراص المدمجة، وماسحات، وراسمات مشغل الأقراص الصلبة. العملية P_1 تملك ماسحة واحدة، والعملية P_2 لها مشغلي أقراص صلبة، ومشغل أقراص مدمجة، والعملية P_3 لديها راسمة واحدة وماسحتين ضوئيتين، كل عملية تحتاج إلى مصادر إضافية، كما هو مبين في مصفوفة الطلبات **R**.

أقراص مدمجة	ماسحات	راسمات	أقراص صلبة	
2	3	2	3	E= متجه المصادر الموجودة

أقراص مدمجة	ماسحات	راسمات	أقراص صلبة	
1	0	1	1	A= متجه المصادر المتاحة

0	0	1	0	P_1
2	0	0	1	P_2
0	1	2	0	P_3

مصفوفة التخصيص الحالية C

2	0	1	2	P_1
1	0	1	0	P_2
1	1	0	0	P_3

مصفوفة الطلبات R

الشكل 4. 13: مثال توضيحي لخوارزمية اكتشاف حالة الجمود.

لتشغيل خوارزمية الكشف عن حالة الجمود، تتطلع الخوارزمية أولًا إلى العملية التي يُمكن تلبية طلباتها، فبالنظر إلى العملية P_1 نجد أنه لا يمكن تحقيق طلبها، لأنه لا يوجد مشغلين للأقراص المدمجة في مصفوفة المتاح، ولا تتوفر ماسحة ضوئية حرة. الأمر نفسه بالنسبة للعملية P_2، بسبب عدم توفر ماسحة ضوئية حرة. لحسن الحظ، يُمكن تلبية رغبة العملية P_3، الأمر الذي يجعل تنفيذ هذه العملية ومعالجة طلباتها ممكنًا، بعد إنهاء العملية P_3 عملها ستُعيد جميع مصادرها، الأمر الذي سيجعل مصفوفة المصادر المتاحة على النحو: (1 2 2 1) = **A**. يُمكن في هذه الحالة تلبية طلبات العملية P_2 بالتالي تشغيلها، ومن ثم استعادة مصادرها، لينتج: **A** = (2 2 2 3). أخيرًا سيتم تشغيل العملية P_1، وسوف لن تكون هناك أي حالة جمود في النظام.

بإجراء تغيير طفيف على الشكل 4. 13 وجعل العملية P_3 تحتاج إلى كل من مشغل الأقراص المدمجة، ومشغلي الأقراص الصلبة، وكذلك الماسحة، سيجعل النظام غير قادر على

تحقيق أيٍ من هذه الطلبات، وهو ما يؤدي إلى حالة الجمود.

من خلال ما سبق ذكره، يمكننا القول أنه بالإمكان معرفة الكيفية التي يُكشف بها عن حالة الجمود (على الأقل في حالة استخدام المصادر الثابتة والمعرَّفة مسبقًا)، ولكن السؤال المطروح الآن: ماذا عن الحالات التى تكون فيها المصادر متغيرة بتغير مجريات التنفيذ وغير محددة مسبقًا؟

أحد الاحتمالات الممكنة هو أن تُختبر حالة الطلبات في كل مرة يُطلب فيها أي مصدر. من المؤكد أن هذا الأمر سيساعد في التعرف على حالة الجمود مبكرًا، ولكن من المحتمل أن تكون هذه الإجراءات مكلفة من حيث إهدار زمن وحدة المعالجة المركزية.

كاستراتيجية بديلة لهذا الحل، يُجرى الاختبار دوريًا بعد فترة زمنية محددة، أو ربما فقط عندما تنخفض استغلالية وحدة المعالجة المركزية إلى ما دون المستوى المفروض. السبب في أخذ استغلالية وحدة المعالجة المركزية في عين الاعتبار، هو أنه إذا توقفت عدة عمليات بسبب الجمود، فسيكون هناك فقط عدد قليل من العمليات قادرًا على الاشتغال، الأمر الذي سيجعل وحدة المعالجة المركزية خاملة في كثير من الأحيان.

3.6.4 التعافي من الجمود

بفرض أن خوارزمية الكشف عن حالة الجمود نجحت في الكشف عنه، فما هي الخطوة التي تلي ذلك؟ في الواقع، هناك عدة بدائل للتعافي من حالة الجمود واسترداد حالة النظام وجعله يعمل مرة أخرى من جديد، يتمثل إحدى هذه البدائل في إخطار المشغل بحدوث حالة الجمود، والسماح له بالتعامل معها يدويًا، بينما يكمن البديل الآخر في السماح للنظام بالتعافي من حالة الجمود تلقائيًا. لذلك يعرض هذا القسم بعض من خيارات كسر حالة الجمود بالرغم من عدم كفاءتها العالية، وهي:

- التعافي بإجهاض العمليات.
- التعافي بسحب المصادر.
- التعافي باستخدام نقاط الفحص.

1.3.6.4 التعافي بإجهاض العمليات

تتمثل أسوأ، ولكن أبسط طريقة للتعافي من حالة الجمود في إجهاض العمليات، والتي إمَّا أن تتمثل في إجهاض عملية واحدة أو في مجموعة من العمليات. في كلتا الحالتين ستُسحب جميع المصادر المخصصة للعملية أو للعمليات المجهضة.

- إجهاض عملية واحدة في كل مرة حتى تُفك حلقة الجمود: إجهاض عملية واحدة داخل حلقة الجمود قد يؤدي مع قليل من الحظ إلى كسر هذه الحلقة وجعل العمليات الأخرى قادرة على الاستمرار، إذا لم تُفك هذه الحلقة، تُجهض عملية أخرى وهكذا حتى تُكسر الحلقة بالكامل. تُستخدم مثل هذه الحلول في الحواسيب المركزية الكبيرة، التي يُعتبر فيها إجهاض عملية وإعادة تشغيلها أمرًا مقبولًا وخصوصًا في أنظمة الدفعة. من مساوي هذا الأسلوب أنه يتسبب في زيادة مقدار الحمل بشكل كبير، لأنه بعد إجهاض أي عملية، يجب استدعاء خوارزمية الكشف عن حالة الجمود لتحديد ما إذا كان لا يزال هناك عمليات أخرى موقوفة بسبب حالة الجمود.
- إجهاض جميع العمليات الموقوفة: من الواضح أن هذه الطريقة ستعمل على كسر حلقة الجمود، ولكن على حساب التكلفة. قد يحدث أن حسابات العمليات الموقوفة بسبب حالة الجمود أخذت فترة طويلة، وبالتالي إجهاضها يعني التخلص من نتائجها الجزئية والتي من المحتمل إعادة حسابها لاحقًا.

بدلًا من إجهاض عملية مشاركة في حلقة الجمود، يمكن اختيار ضحية أخرى خارج الحلقة من أجل تحرير مصادرها، يستوجب هذا الأمر اختيار العملية الضحية بعناية، التي تحتجز مصادر مطلوبة من قِبل بعض من العمليات داخل الحلقة. مثلًا، قد تمتلك إحدى العمليات طابعة وتحتاج إلى راسمة، في حين تحتجز عملية أخرى راسمة وترغب في الحصول على الطابعة، الأمر الذي سيؤدي إلى حالة الجمود، ولكن قد تكون هناك عملية ثالثة تمتلك طابعة، وراسمة أخرتين مماثلتين للمصدرين السابقين، على التوالي بالتالي إجهاض هذه العملية سيؤدي إلى تحريرهما، الأمر الذي سيُساهم في كسر حلقة حالة الجمود.

في العموم، قد لا يكون إجهاض أي عملية أمرًا سهلًا، لأنه من المهم جدًا معرفة أنه من

الأفضل إجهاض عملية يمكن إعادة تشغيلها من البداية دون أن تتسبب في آثار سيئة. مثلًا، يمكن دائمًا إعادة تشغيل عملية الترجمة، لأن كل ما تفعله هو قراءة ملف مصدري وإنتاج ملف هدفي، بالتالي إجهاض هذه العملية وهي في منتصف طريق تنفيذ الجولة الأولى ليس له أي تأثير على تنفيذ الجولة الثانية. بالمقابل، إذا كانت العملية في منتصف عملية تحديث أي قاعدة بيانات، فإن إجهاضها وإعادة تشغيلها من جديد لا يمكن أن يخلو من المخاطر، لأنه إذا أضافت هذه العملية أي قيمة إلى بعض من الحقول في أحد جداول قاعدة البيانات هذه، فإن إعادة تشغيلها قد تُضيف مرة أخرى نفس القيمة من جديد إلى نفس الحقول، الأمر الذي سيكون غير صحيح. بالمثل، إذا كانت العملية في وسط سحب البيانات على الطابعة، فيتوجب على النظام إعادة ضبط الطابعة على الحالة الصحيحة قبل طباعة المهمة التالية.

خلاصة الأمر، إذا أُستخدمت طريقة الإنهاء الجزئي، فيجب أن يتم تحديد أي عملية (أو عمليات) موقوفة بسبب حالة الجمود حتى يتم إجهاضها. يُعتبر هذا القرار قرارًا يتعلق بسياسة النظام والذي يُشابه قرارات جدولة وحدة المعالجة المركزية. السؤال في الأساس اقتصادي، بمعنى يجب إجهاض العمليات التي سيُكلف إجهاضها الحد الأدنى من التكلفة. لسوء الحظ، فإن مصطلح الحد الأدنى للتكلفة ليس دقيقًا، لأن هناك عدة عوامل تؤثر على العملية التي ستُختار، نذكر منها:

1. ما أولوية العملية؟
2. ما المدة التي استغرقتها العملية في الحساب؟ وما المدة التي لازالت تحتاجها حسابات العملية قبل إتمام المهمة المحددة لها؟
3. كم عدد المصادر التي تستخدمها العملية؟ وما أنواعها (على سبيل المثال، ما إذا كانت المصادر سهلة السحب)؟
4. كم عدد المصادر التي تحتاجها العملية حتى تنتهي؟
5. كم عدد العمليات التي سيحتاج النظام إلى إجهاضها؟
6. ما إذا كانت العملية المختارة تفاعلية أو دفعة؟

2.3.6.4 التعافي بسحب المصادر

من أجل التعافي من حالة الجمود قد يكون من الممكن– في بعض الحالات– سحب بعض من المصادر بشكل مؤقت بعيدًا عن مالكها الحالي وتخصيصها إلى عملية أخرى حتى تنكسر حلقة الجمود، وفي كثير من الحالات الأخرى، قد تكون هناك حاجة للتدخل اليدوي، خصوصًا في أنظمة تشغيل المعالجة بالدفعة التي تعمل على الحواسيب المركزية.

على سبيل المثال، لسحب طابعة ليزرية بعيدًا عن مالكها، يمكن للمشغل أن يجمع كافة الأوراق المطبوعة مسبقًا ويضعها في الرف، ومن ثم تُعلق هذه العملية (مع وضع علامة على أساس أنها جاهزة)، بعدها يُمكن تخصيص هذه الطابعة إلى عملية أخرى تساعد في فك حلقة الجمود. عند إنتهاء العملية الأخيرة من الطباعة، يتم وضع كومة الأوراق المسحوبة مسبقًا مرة أخرى في علبة إخراج الطابعة واستئناف العملية الأصلية.

القدرة على سحب مصدر– ما– بعيدًا عن عمليته المالكة له، ومن ثم استخدامه من قبل عملية أخرى، ثم تخصيصه لها من جديد دون ملاحظة ذلك يعتمد إلى حد كبير على طبيعة المصدر، بالتالي التعافى بهذه الطريقة في كثير من الأحيان صعبًا إن لم يكن مستحيلًا، لأن اختيار تعليق عملية– ما– يتوقف بشكل كبير على أيٍ من العمليات التي لديها مصادر يمكن– في الواقع– سحبها بسهولة. عمومًا إذا كان سحب المصادر أمرًا مطلوبًا للتعافي من حالة الجمود، فيجب معالجة ثلاث قضايا:

1. اختيار الضحية: عندما يُتخذ قرار بسحب المصادر للتعافي من حالة الجمود يجب تحديد أيٍ من المصادر أو العمليات التي ستُسحب. كما هو الحال في إنهاء العملية، يجب تحديد ترتيب عمليات السحب لكي تُقلل التكلفة. قد تتضمن عوامل التكلفة مَعْلمات مثل عدد المصادر التي تمتلكها العملية المساهمة في حالة الجمود، وكذلك مقدار الوقت الذي استغرقته العملية حتى الآن.
2. الرجوع: إذا كان بالاستطاع سحب مصدر من أي عملية، فما الذي يجب فعله بهذه العملية؟ من الواضح أنه لا يمكن الاستمرار في التنفيذ الاعتيادي للعملية بسبب حدوث حالة الجمود، لذلك يجب إعادة العملية إلى حالة آمنة وإعادة تشغيلها من هناك.

بشكل عام، من الصعب تحديد الحالة الآمنة، لذلك فإن أبسط حل يتمثل في التراجع الكلي: أي وقف العملية ومن ثم إعادة تشغيلها، إلَّا إنَّ الأكثر فاعلية هو الرجوع بالعملية بقدر ما يلزم لكسر حالة الجمود، وهو ما يتطلب من النظام الاحتفاظ بالمزيد من المعلومات حول حالة كافة العمليات قيد التشغيل.

3. المجاعة: يستند اختيار الضحية في أي نظام في المقام الأول على عوامل التكلفة، كما أشرنا سابقًا، ولكن قد يحدث أن تُختار دائمًا نفس العملية كضحية، وهو ما يُعرف بالمجاعة، ونتيجة لذلك، لن تُكْمل هذه العملية مهمتها المحددة لها أبدًا، بالتالي يجب على أي نظام عملي معالجة مثل هذه الحالات. من الحلول المقترحة هنا هو التأكيد على اختيار العملية كضحية فقط لعدد (قليل) من المرات، ولكن الحل الأكثر شيوعًا هو تضمين عدد مرات الرجوع في عامل التكلفة.

3.3.6.4 التعافي باستخدام نقاط الفحص

إذا كان هناك معرفة باحتمالية حدوث حالة الجمود فإنه بالإمكان التعافي منها من خلال وضع ترتيبات محددة لتواجد نقاط فحص يُحتَفظ فيها بحالة النظام بشكل دوري. نقاط الفحص هذه عبارة عن هياكل بيانات تحتفظ على الأقل بحالة العملية، وحالة الذاكرة، وزمن حدوث الفحص، وحالة كل مصدر، أي أيٍ من المصادر مُخصص حاليًا ولأيٍ من العمليات. تُحفظ هذه الحالات في ملفات خاصة بحيث يُمكن الرجوع إليها لاحقًا ومن ثم إعادة تشغيلها. لذلك لكي تكون نقاط الفحص أكثر فاعلية، ينبغي على أي نقطة فحص جديدة عدم الكتابة على البيانات القديمة، بل ينبغي عليها أن تُدون ذلك في ملفات جديدة، بحيث يُجمَّع تسلسل كافة خطوات التنفيذ في أثناء تنفيذ هذه العملية.

عندما يتم الكشف عن حالة الجمود يلجأ النظام إلى نقاط الفحص هذه والتي من خلالها سيكون من السهل التعرف على المصادر التي تحتاجها العمليات، بالتالي للقيام بعملية التعافي من حالة الجمود يُعاد تنفيذ العملية المالكة للمصدر الذي احتاجته عملية أخرى بتنفيذ أحد نقاط الفحص التي تسبق اللحظة التي خُصص فيها هذا المصدر، وهو ما سيؤدي إلى فقدان العمل المنجز منذ زمن حدوث نقطة الفحص. في حالة إرجاع العملية التي سُحب منها المصدر إلى اللحظة التي لم يكن فيها المصدر مخصصًا لها– والذي خُصص الآن إلى إحدى العمليات

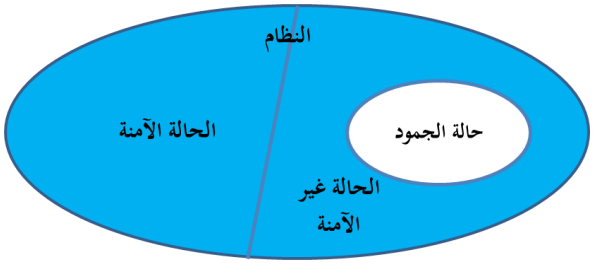
الموقوفة بسبب حالة الجمود– وقامت هي نفسها بمحاولة الحصول على المصدر مرةً أخرى، فسوف تضطر إلى الانتظار إلى حين تحرره.

7.4 تجنب حدوث الجمود

يهدف مفهوم تجنب حدوث حالة الجمود إلى السماح للنظام بتغيير الحالة عن طريق تخصيص المصادر، فقط عندما يكون من المؤكد أن عملية التخصيص إذا ما تمت لن ينتج عنها حالة الجمود. يتأتى هذا من خلال فحص حالة النظام بشكل مستمر وتحليل حالة تخصيص المصادر الحالية لمعرفة الحالات الآمنة وغير الآمنة. سنعود إلى التطرق لمفهوم هاتين الحالتين بنوع من التفصيل في القسم 2.7.4، ولكن يكفينا هنا أن نعرف أن الحالة الآمنة هي الحالة التي لا يؤدي فيها تخصيص المصادر إلى حالة الجمود، وعلى العكس من ذلك فالحالة غير الآمنة هي الحالة التي إذا ما خُصصت فيها المصادر فحتمًا سيؤدي ذلك إلى حالة الجمود.

ولإجراء فحص الحالة الآمنة، يجب على العملية أن تُخطر نظام التشغيل بالحد الأقصى لعدد المصادر من كل نوع والتي من المحتمل أن تطلبها العملية لإكمال تنفيذها. المقصود بالحد الأقصى لمطالبة المصادر هو إجمالي عدد مثيلات المصادر من كل نوع التي ستطلبها أي عملية.

يُوضح الشكل 4. 14 النظام بحالتيه الآمنة وغير الآمنة، كما يُمكن ملاحظة أن حالة الجمود مرتبطة فقط بالحالة الأخيرة، علمًا بأن حالة الجمود هي حالة غير آمنة، إلَّا إنَّ الحالة غير الآمنة ليس بالضرورة أن تُمثل حالة جمود. يُمكن للنظام أيضًا أن ينتقل من الحالة الآمنة إلى الحالة غير الآمنة في حالة الموافقة على طلبات المصادر الحرجة، من الناحية المثالية، يجب على النظام التحقق من كل طلب للمصادر، لمعرفة ما إذا كان النظام لا يزال آمنا عند تحقيقه.



الشكل 4. 14: النظام بحالتيه الآمنة وغير الآمنة.

خلاصة الأمر من خلال مناقشة الكشف عن الجمود، كنا نفترض ضمنيًا أنه عندما تطلب عملية– ما– مصادر محددة، فإنها تقوم بطلبهم دفعة واحدة (المصفوفة **R** من الشكل 4. 12)، إلَّا إنَّ طلب المصادر يكون في معظم نظم التشغيل بشكل متتالي، لذلك يجب على النظام أن يكون قادرًا على معرفة ما إذا كان تخصيص المصدر آمنًا أم لا، وعلى اتخاذ قرار التخصيص فقط عندما يكون الوضع آمنًا، وبناءً عليه فإن السؤال الذي يطرح نفسه: هل هناك خوارزمية قادرة دائمًا على تجنب حالة الجمود من خلال اتخاذ القرار الصحيح في جميع الأوقات؟ الجواب هو نعم، أي يمكننا تجنب حالة الجمود، ولكن فقط إذا أُتيحت لنا بعض من المعلومات في وقت مبكر. سيتطرق هذا القسم إلى بعض من خوارزميات تجنب حدوث حالة الجمود التي تعتمد على التخصيص الدقيق للمصادر.

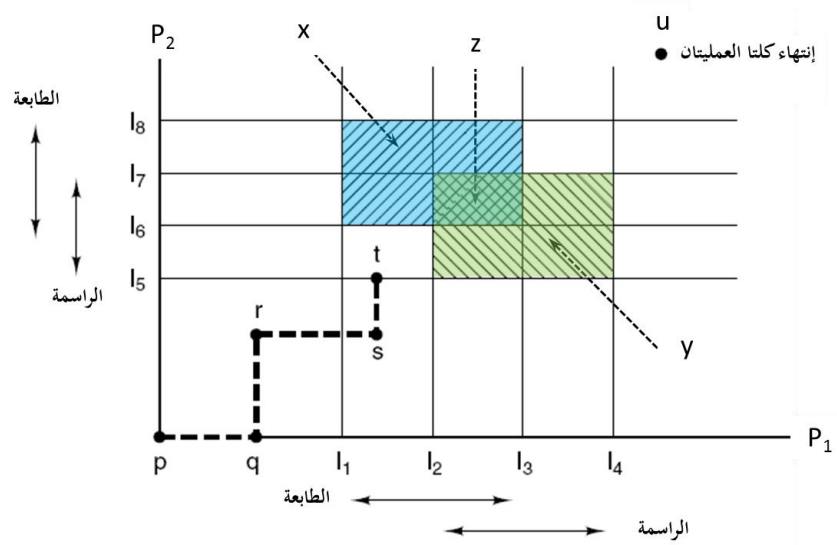
1.7.4 مسارات المصادر

تستند الخوارزميات الرئيسية الخاصة بتجنب حالة الجمود على مفهوم الحالة الآمنة، بالتالي قبل وصف هذه الخوارزميات، سوف ننظر بشكل طفيف إلى مفهوم الحالة الآمنة باستخدام طريقة الرسم لسهولة فهمها، بالرغم من أنها لا تُترجم بشكل مباشر إلى خوارزمية قابلة للاستخدام، وإنما تُعطي شعورًا جيدًا بطبيعة المشكلة.

تُعرف هذه الطريقة بمسارات المصادر والتي تفترض أن هناك عمليتان $\mathbf{P_1}$ و $\mathbf{P_2}$ وأن كلاهما بحاجة إلى استخدام الطابعة والراسمة (يتوفر مصدر واحد من كل منهما)، كما هو موضح في الشكل 4. 15. يُمثل المحوران الأفقي والرأسي عدد التعليمات التي تُنفذهما $\mathbf{P_1}$ و $\mathbf{P_2}$ على التوالي، بينما يُمثل الخط المتقطع مسار المصدر والذي ينشأ عند تنفيذ المجدول لكل من هاتين العمليتين، في حين تُمثل النقاط على هذا المسار حالات الفصل بين العمليتين. بفرض أن **p** هي نقطة الانطلاق الأولى وأن النقطة **u** هي الهدف المراد الوصول إليه. للوصول إلى هذا الهدف يجب أن تمر العملية $\mathbf{P_1}$ بتنفيذ الأوامر $\mathbf{I_1}$ و $\mathbf{I_2}$ و $\mathbf{I_3}$ و $\mathbf{I_4}$، والعملية $\mathbf{P_2}$ بتنفيذ الأوامر $\mathbf{I_5}$ و $\mathbf{I_6}$ و $\mathbf{I_7}$ و $\mathbf{I_8}$.

في أثناء تنفيذ العملية $\mathbf{P_1}$ للأمر $\mathbf{I_1}$ طلبت الطابعة، بينما بتنفيذها للأمر $\mathbf{I_2}$ احتاجت إلى الراسمة ومن ثم أعادت المصدرين عند كل من $\mathbf{I_3}$ و $\mathbf{I_4}$ على التوالي. بنفس الطريقة، احتاجت

P_2 الراسمة من الأمر I_5 إلى I_7، والطابعة من الأمر I_6 إلى I_8.



الشكل 4. 15: مسارات المصادر الخاصة بالعمليتين P_1 و P_2 [Tanenbaum & Bos, 2015].

مبدئيًا عند النقطة p لم تُنفذا العمليتان P_1 و P_2 أي أمر، وبفرض أن المجدول قام أولًا باختيار العملية P_1، سنصل بالتالي إلى النقطة q التي عندها تكون العملية قد نفذت بعض من الأوامر التابعة لها، بينما لم تُنفذ بعد العملية P_2 أي تعليمة. عند هذه النقطة قرر المجدول تنفيذ العملية P_2 ليصبح المسار أفقي وسنصل إلى النقطة r، وهكذا ثم إلى s مرورًا بالأمر I_1 والذي سينتج عنه تخصيص الطابعة للعملية P_1، عندما يصل المسار إلى النقطة t تطلب العملية P_2 الراسمة، هنا يجب ملاحظة أنه في حالة امتلاك النظام لمعالج واحد فقط، ستكون جميع المسارات إما أفقية أو عمودية، كما أن الحركة ستكون دائمًا في إتجاه الشمال، أو الشرق، ولن تكون في إتجاه الجنوب، أو الغرب، لأن العمليات لا يمكن تشغيلها– بطبيعة الحال– زمنيًا إلى الوراء.

تُمثل المناطق المظللة مناطق مستحيلة وهي تحضى باهتمام خاص، فالمساحة x تُمثل الحالة التي ستُستخدم فيها الطابعة من قبل العمليتين، وحسب قواعد المنع التبادلي سيكون من

المستحيل الدخول إلى هذه المنطقة. بالمقابل تُمثل المساحة **y** الوضعية التي ستُستخدم فيها الراسمة من قبل P_1 و P_2، والتي هي الأخرى وحسب قواعد المنع التبادلي سيكون من المستحيل الدخول إليها، أمَّا المساحة **z** ستُمثل منطقة حالة الجمود.

الملاحظة المهمة هنا هي– في الواقع– عند النقطة **t**، لتفادي حالة الجمود يجب على المجدول جدولة العملية P_1 وتنفيذها حتى تنتهي من الأمر I_4 قبل أن يسمح بتنفيذ العملية P_2 مرة أخرى، وذلك لتفادي دخول النظام إلى المنطقة غير الآمنة والتي ستؤدي في نهاية المطاف إلى حالة الجمود. بعد ذلك يُمكن التحرك في أي مسار إلى أن نصل إلى النقطة **u**، التي سينتهي عندها تنفيذ كلتا العمليتان من دون أن يمر المسار عبر إحدى المناطق المظللة.

2.7.4 الحالات الآمنة وغير الآمنة

أشرنا في القسم السابق بأن الحالة الآمنة تعني إمكانية جدولة العمليات بحيث يمكن تنفيذها كافة إلى النهاية حتى لو طلبت جميعها على الفور وبشكل مفاجئ كافة مصادرها (بما في ذلك الحد الأقصى المحدد لها) دون التعرض لحالة الجمود. بشكل أكثر وضوحًا، تتوفر الحالة الآمنة إذا كان هناك تسلسل آمن لتنفيذ العمليات $\{P_0، P_1، ...، P_n\}$ بحيث يُمكن منح جميع طلبات المصادر للعملية P_i باستخدام المصادر المخصصة حاليًا لها ولجميع العمليات P_j حيث **j** أقل من **i**، (بمعنى آخر، إذا إنتهت جميع العمليات التي قبل العملية P_i وحررت مصادرها، فستكون العملية P_i نفسها قادرة على الإنتهاء أيضًا باستخدام المصادر التي حُررت)، أما في حالة عدم وجود هذا التسلسل الآمن فسيكون النظام في حالة غير آمنة، مما قد يؤدي إلى حالة الجمود.

مثال توضيحي:

لتوضيح مفهومي الحالة الآمنة وغير الآمنة باستخدام مصدر واحد من كل نوع يُمكن تتبع المثال المعطى في الشكل 4. 16–أ، علمًا بأن الخوارزميات المستخدمة في تجنب حالة الجمود تستخدم البيانات المعطية في الشكل 4. 12. في أي لحظة زمنية في هذا الشكل، هناك حالة حالية تتكون من المصفوفات **E** ، **A**، **C**، و **R**. بالعودة إلى هذا المثال نجد أن لدينا حالة تمتلك فيها العملية P_1 أربع عينات من مصادر مختلفة والحد الأقصى لها يصل إلى

ثمانية مصادر، كذلك العملية P_2 ثلاثة مصادر وتحتاج في وقت لاحق إلى ستة مصادر مع بعض كحد أقصى. بالمثل، لدى العملية P_3 مصدرين، وقد تحتاج إلى سبعة مصادر إضافية. مجموع هذه المصادر اثنتا عشر، منها تسعة مصادر مخصصة بالفعل، وثلاثة مازالت متاحة.

بتتبع الحالة المبينة في الشكل 4. 16–أ نجد أنها حالة آمنة، لأن هناك إمكانية لتلبية كافة المطالب، الأمر الذي سيُتيح لجميع العمليات فرصة الاستمرار في التنفيذ حتى النهاية. هذه الإمكانية تتمثل ببساطة في جدولة P_2 حصريًا، ومماطلة كافة الطلبات الأخرى إلى أن تطلب بقية احتياجاتها من المصادر والحصول عليها، الأمر الذي سيؤدي إلى الحالة الموضحة في الشكل 4. 16–ب، علمًا أنه عندما تنتهي هذه العملية ستُعيد كافة مصادرها لتنتج لدينا الحالة الممثلة في الشكل 4. 16–ج، والتي تُتيح لنا فرصة جدولة تشغيل P_2، مما تؤدي في النهاية إلى الشكل 4. 16–د، مع إنتهاء العملية الأخيرة نحصل على الشكل 4. 16–هـ. الآن يُمكن للعملية P_3 الحصول على مصادرها السبعة التي تحتاجها واستكمال مهمتها، بالتالي هكذا حالة والمعطية في الشكل 4. 16–أ هي حالة آمنة، لأن النظام– من خلال تحديد مواعيد دقيقة للتنفيذ– يمكنه تجنب حدوث حالة الجمود.

بالمقابل بفرض أن العملية P_1– في الشكل 4. 16–أ– قد طلبت مصدر آخر وتحصلت عليه لتنتج لنا الحالة المعطاة في الشكل 4. 17. السؤال المطروح الآن: هل يمكننا ايجاد تسلسل آمن لتنفيذ العمليات يضمن لنا تجنب حدوث حالة الجمود؟ دعونا نحاول!

العملية	المستخدم	الحد الأقصى
P_1	4	8
P_2	3	6
P_3	2	9

المتاح: 3 (أ)

العملية	المستخدم	الحد الأقصى
P_1	4	8
P_2	6	6
P_3	2	9

المتاح: 0 (ب)

العملية	المستخدم	الحد الأقصى
P_1	4	8
P_2	0	-
P_3	2	9

المتاح: 6 (ج)

العملية	المستخدم	الحد الأقصى
P_1	8	8
P_2	0	-
P_3	2	9

المتاح: 2 (د)

العملية	المستخدم	الحد الأقصى
P_1	0	-
P_2	0	-
P_3	2	9

المتاح: 10 (هـ)

الشكل 4. 16: توضيح أن الحالة في (أ) حالة آمنة.

بالتمعن في المثال الموضح في الشكل 4. 17 نجده مثالًا لحالة غير آمنة تفتقر إلى وجود

إمكانية إيجاد تسلسل آمن لتنفيذ العمليات يضمن لنا تجنب حدوث حالة الجمود. في هذه الحالة يجب على النظام التأني في اتخاذ قراراته، لأن النظام لا يمتلك حاليًا سوى مصدرين متاحين للاستخدام، وأقل عملية نشطة تحتاج إلى ثلاثة مصادر. هذا يعني أنه ليس هناك أي تسلسل آمن للتنفيذ يضمن تنفيذ العمليات إلى النهاية. بالتالي فإن قرار التخصيص الذي ينقل النظام من الشكل 4. 16–أ إلى الشكل 4. 17 سيُمثل التحول من الحالة الآمنة إلى الحالة غير الآمنة، إذًا من المهم جدًا لاحقًا ألَّا يتحقق طلب العملية P_1 في الشكل 4. 16–أ.

العملية	المستخدم	الحد الأقصى
P_1	5	8
P_2	3	6
P_3	2	9

المتاح: 2

الشكل 4. 17: حالة غير آمنة.

من الجدير بالذكر أن الحالة غير الآمنة لا تعني حدوث حالة الجمود، كما هو الحال مع المثال المبين في الشكل 4. 17–أ، أي أنه يُمكن تشغيل النظام لفترة من الزمن، بل من الممكن تشغيل إحدى العمليات حتى النهاية. علاوةً على ذلك، من المحتمل أن تُعيد العملية أحد مصادرها قبل أن تطلب أي مصدر آخر، الأمر الذي قد يعطي الفرصة للعملية P_3 باستكمال وظيفتها، ومن ثم تجنب حالة الجمود. وهكذا فإن الفرق بين الحالة الآمنة والحالة غير الآمنة أنه في الأولى يمكن للنظام أن يضمن إنتهاء تنفيذ كافة العمليات، بينما في الحالة الأخيرة لا يمكنه إعطاء مثل هذه الضمانة.

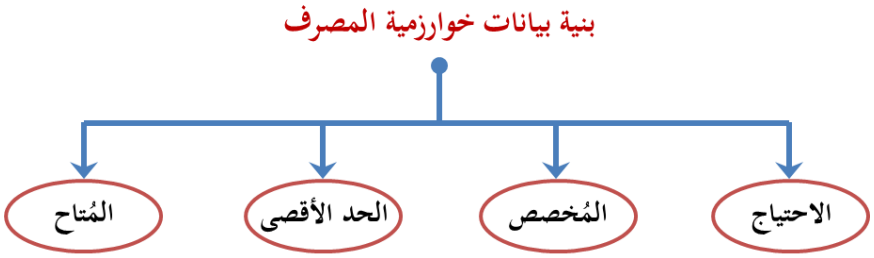
3.7.4 خوارزمية المصرف

تعرضنا في الأقسام السابقة إلى خوارزميات استخدام الرسم البياني لتخصيص المصادر لاكتشاف حالات الجمود، والتي لا تتلائم مع أنظمة التخصيص في الحالات التي تتعدد فيها المصادر من كل نوع. خوارزمية المصرف هي إحدى الخوارزميات التي تتلائم بشكل أفضل مع مثل هذه الحالات، إلَّا إنَّها أقل فاعلية من مخطط الرسم البياني لتخصيص المصادر. هذه

الخوارزمية هي امتداد لخوارزميات اكتشاف حالات الجمود. يُناقش هذا القسم كل من خوارزمية المصرف الخاصة بمصدر واحد من كل نوع، والخاصة بعدة مصادر من نفس النوع.

1.3.7.4 خوارزمية المصرف الخاصة بمصدر واحد من كل نوع

ما تفعله هذه الخوارزمية هو التحقق من ما إذا كانت حالة تخصيص المصدر الحالية هي حالة آمنة، إذا كانت كذلك فسيُخصص، وإلَّا سيُرفض الطلب، لأن الحالة ستكون غير آمنة. تُعرف حالة تخصيص المصادر من خلال إجمالي عددها في النظام، وعدد المصادر المخصصة، والحد الأقصى للمصادر الذي تحتاجه كل عملية. هذه المعلومات (الحالة) تُخزن في بنية البيانات الموضحة في الشكل 4. 18. سنعود إلى مناقشة هذا المخطط بنوع من التفصيل في القسم التالي.



الشكل 4. 18: مخطط بنية بيانات خوارزمية المصرف.

مثال توضيحي:

يُمكن توضيح فكرة هذه الخوارزمية من خلال تمثيلها بمصرف قرية صغير يتعامل مع مجموعة من العملاء مُنِحوا تسهيلات ائتمانية. يُعطي الشكل 4. 19-أ مثال توضيحي لخوارزمية المصرف نُشاهد فيه أربعة زبائن، يونس، ويوسف، وآدم، وجنان. كل واحد منهم مُنِح عدد محدد من وحدات الائتمان (حيث تناظر كل وحدة مثلًا، 1000 دينار)، ولأن المصرف على دراية بأن زبائنه لن يحتاجو إلى الحد الأقصى للائتمانات الخاصة بهم على الفور، قرر الاحتفاظ فقط باثنتي عشرة وحدة لخدمتهم، بدلًا من ثلاث وعشرين وحدة (يُمثل في هذا السياق الزبون العملية، والوحدة المصدر مثل الطابعة، وأخيرًا يُمثل المصرف نظام التشغيل).

الزبون	المستخدم	الحد الأقصى
يونس	0	8
يوسف	0	4
آدم	0	6
جنان	0	5

المتاح: 12

(أ)

الزبون	المستخدم	الحد الأقصى
يونس	5	8
يوسف	2	4
آدم	1	6
جنان	2	5

المتاح: 2

(ب)

الزبون	المستخدم	الحد الأقصى
يونس	5	8
يوسف	2	4
آدم	1	6
جنان	3	5

المتاح: 1

(ج)

الشكل 4. 19: حالات تخصيص المصادر– أ) الحالة الاستهلالية– ب) آمنة– ج) غير آمنة.

يطلب الزبائن كما هو معتاد في المصارف من وقت لآخر قروض من المصرف (أي طلب مصدر)، بالتالي بعد فترة زمنية يمكن أن تأخذ وضعية القروض النمط الموضح بالشكل 4. 19– ب. بالتمعن في هذا الشكل نجد أن هذه الحالة هي حالة آمنة، لأنه مع توفر وحدتين كباقي ضمن المتاح، يُمكن للمصرف أن يُلبي طلبات الزبون يوسف، لأنه قادر على تحقيقها، ويُماطل في طلبات التخصيص الأخرى. عند إنتهاء حاجة يوسف من الوحدات الأربع المخصصة له سيُعيدها، مع توفر أربع وحدات كمتاح، يُمكن للمصرف الآن السماح إمَّا للزبون يونس، أو جنان بأن يمتلك الوحدات اللازمة لمواصلة العمل.

بفرض أن الزبونة جنان قد طلبت قرضًا وتمت تلبيته، لتنشأ الوضعية الموضحة في الشكل 4. 19–ج. الحالة الناشئة الآن هي حالة غير أمنة، وذلك لأن عدد الأرصدة المتبقي لا يكفي لتحقيق أي طلب في حالة ما طلب أي زبون كافة أرصدته، الأمر الذي سيؤدي بدوره إلى حدوث حالة الجمود. إنَّ حالة غير آمنة كهذه لا تعني في الواقع حدوث الجمود للمصرف، لأنه لم يلبي الطلب بعد، وإنما هي عبارة عن حالة تؤدي إلى تأجيل تحقيق الطلب إلى حين آخر.

تتعامل خوارزمية المصرف مع الطلبات كل على حدا، وتتحقق في كل مرة من أن منحه (أي تخصيصه) سيؤدي إلى حالة آمنة، إذا كانت كذلك، يُمنح الطلب، وإلَّا فسيُؤجل إلى وقت لاحق. لمعرفة ما إذا كانت الحالة آمنة، يتحقق المصرف من معرفة ما إذا كان لديه ما يكفي من المصادر

لخدمة بعض من الزبائن، إذا كان الأمر كذلك، يفترض المصرف أن تلك القروض ستُسدد، والزبائن الآن أقرب إلى تحقيق رغباتهم. إذا تأكد المصرف– في نهاية المطاف– من أنه قادر على سداد جميع القروض، فالحالة تعتبر آمنة، بالتالي يمكنه منح الطلب الأولي.

2.3.7.4 خوارزمية المصرف الخاصة بعدة مصادر من نفس النوع

خوارزمية المصرف يمكن تعميمها للتعامل مع عدة مصادر من نفس النوع. بالعودة إلى مخطط بنية بيانات هذه الخوارزمية والمبين في الشكل 4. 18 واستخدام البيانات المعطية في الشكل 4. 12، نجد أن خوارزمية المصرف الخاصة بعدة مصادر من نفس النوع يمكن تنفيذها باستخدام البيانات التالية مع فرضية أن عدد العمليات هو n وعدد أنواع المصادر m.

متجه المتاح **Ava**: وهو متجه بطول m ويُمثل عدد المصادر المتاحة من كل نوع بحيث إذا كانت $Ava_j = k$ فهذا يعني أن هناك k من المصادر المتاحة من نوعية R_j.

مصفوفة الحد الأقصى **Max**: وهي عبارة عن مصفوفة ذات بعد $n \times m$ تُمثل الحد الأقصى لعدد المصادر من كل نوع التي يمكن أن تطلبها العملية لإكمال تنفيذها بحيث إذا كانت $Max_{ij} = k$، فهذا يعني أن العملية P_i بإمكانها طلب على الأغلب k من المصادر من نوعية R_j.

مصفوفة التخصيص **C**: وهي عبارة عن مصفوفة ذات بعد $n \times m$ تُمثل عدد المصادر من كل نوع التي حاليا مخصصة لكل عملية بحيث إذا كانت $C_{ij} = k$، فهذا يعني أن العملية P_i حاليًا مخصص لها k من المصادر من نوعية R_j.

مصفوفة الإحتياج **Need**: وهي عبارة عن مصفوفة ذات بعد $n \times m$ تُمثل عدد المصادر المتبقية التي تحتاجها كل عملية بحيث إذا كانت $Need_{ij} = k$، فهذا يعني أن العملية P_i مازالت تحتاج k من المصادر من نوعية R_j لإنهاء مهمتها، علما أن:

$$Need_{ij} = Max_{ij} - C_{ij}.$$

أمثلة توضيحية:

تُوضح الأمثلة التالية مجموعة من حالات النظام المتعددة والممثلة بتخصيص المصادر

لغرض التعرف على كيفية استخدام خوارزمية المصرف في تحديد الحالة الآمنة للنظام من عدمها.

المثال الأول: يوضح الشكل 4. 20 المعطيات الخاصة بهذا المثال. تُمثل المصفوفة التي على اليمن عدد المصادر المخصصة حاليًا لكل من العمليات الخمس، والتي في الوسط تبين عدد المصادر التي لازلت تحتاجها كل عملية من أجل إنهاء مهمتها، تُمثلان هاتان المصفوفتان المصفوفتين C و R في الشكل 4. 12، في حين تُمثل المصفوفة التي على اليسار مصفوفة الحد الأقصى لعدد المصادر من كل نوع التي يُمكن أن تطلبها العملية لإكمال تنفيذها.كما هو الحال بالنسبة للمصدر الواحد، يجب على العمليات تحديد المصادر الخاصة بها، التي تحتاجها قبل أن تبدأ في التنفيذ، بحيث يُمكن للنظام حساب مصفوفة الاحتياج في أي لحظة.

تُوضح المتجهات الثلاثة في أقصى يسار الشكل متجه المصادر المتوفرة، والمصادر المخصصة، والمصادر المتاحة الذي يُمثل عدد المصادر المتاحة من كل نوع. من متجه المتوفر يمكن ملاحظة أن النظام لديه ستة مشغلات أشرطة ممغنطة، وثلاث راسمات، وأربع طابعات، ومشغلان للأقراص المدمجة، مخصص منها حاليًا خمسة مشغلات أشرطة ممغنطة، وثلاث راسمات، وطابعتان، ومشغلان للأقراص المدمجة. هذه الحقيقة يمكن ملاحظتها من خلال إضافة أعمدة المصادر الأربعة في المصفوفة اليمنى، بينما متجه المصادر المتاحة هو ببساطة الفرق بين ما يملكه النظام وما هو قيد الاستخدام حاليًا.

العملية	أشرطة ممغنطة	راسمات	طابعات	أقراص مدمجة
P_1	3	0	1	1
P_2	0	1	0	0
P_3	1	1	1	0
P_4	1	0	1	1
P_5	0	0	0	0

مصفوفة التخصيص

(أ)

العملية	أشرطة ممغنطة	راسمات	طابعات	أقراص مدمجة
P_1	1	1	0	0
P_2	0	1	1	2
P_3	3	1	0	0
P_4	0	0	1	0
P_5	2	1	1	2

مصفوفة الاحتياج

(ب)

العملية	أشرطة ممغنطة	راسمات	طابعات	أقراص مدمجة
P_1	4	1	1	1
P_2	0	2	1	2
P_3	4	2	1	0
P_4	1	1	1	1
P_5	2	1	1	0

مصفوفة الحد الأقصى

(ج)

متجه المتوفر: (2 4 3 6)

متجه المخصص: (2 2 3 5)

متجه المتاح: (0 2 0 1)

الشكل 4. 20: معطيات المثال الأول لخوارزمية المصرف الخاصة بعدة مصادر من نفس النوع.

والآن يُمكن تلخيص خوارزمية التأكد من أن الحالة آمنة كما يلي:

1. ابحث عن أي صف في مصفوفة الاحتياج الذي كل مصادره أصغر من أو يساوي متجه المتاح، في حالة عدم وجود ذلك، فإن النظام– في نهاية المطاف– سيتوقف بسبب حالة الجمود، لأنه لا يمكن تشغيل أي عملية إلى النهاية (على افتراض أن العمليات تحتفظ بجميع مصادرها إلى أن تنتهي).
2. افترض أن العملية المناظرة لهذا الصف قامت بطلب كل المصادر التي تحتاجها (وهو حق مكفول لها) ومن ثم أنهت عملها، أَشِّرْ على هذه العملية على أساس أنها منتهية، وأضف كافة مصادرها إلى متجه المتاح.
3. كرر الخطوتين السابقتين إلى أن يتم وضع علامات إنهاء على جميع العمليات (هذه الوضعية تعني أن الحالة الأولية هي حالة آمنة)، أو إلى أن تجد عملية لا يمكن تلبية طلباتها (الأمر الذي سيُمثل حالة غير آمنة).

إذا كانت هناك عدة عمليات مؤهلة ليتم اختيارها في الخطوة الأولى، فإنه لا يهم أي منها سيتم اختيارها: فمتجه المصادر المتاحة إمَّا أن يكبر، أو في أسوأ الأحوال، يبقى على ما هو عليه.

بالعودة إلى المثال في الشكل 4. 20، الحالة الراهنة هي حالة آمنة. ولكن بفرض أن العملية P_2 طلبت الآن الطابعة وتمت تلبية الطلب، لأن الحالة الناتجة لا تزال آمنة (العملية P_4 يمكنها الإنتهاء، ومن ثم العملية P_1 أو P_5، تليها البقية).

ولنتخيل الآن أنه بعد منح العملية P_2 إحدى الطابعتين المتبقيتين، طلبت العملية P_5 الطابعة الأخيرة، تحقيق هذا الطلب سيجعل متجه المصادر المتاحة يكون عل النحو (0 0 0 1)، الأمر الذي سيؤدي إلى حالة الجمود، لذلك يجب تأجيل طلب P_5 لفترة من الوقت.

المثال الثاني: يوضح الشكل 4. 21 معطيات نظام ذو معالج واحد له ثلاثة أنواع من المصادر متوفر منها خمس وحدات من كل نوع وتشترك جميعًا في ثلاث عمليات. تُمثل المصفوفة في الشكل 4. 21–أ مصفوفة التخصيص، بينما تُمثل المصفوفة في الشكل 4. 21–ب مصفوفة الاحتياج. السؤال: أيٍ من العمليات الثلاث ستنتهي كآخر عملية؟

العملية	ماسحات	راسمات	طابعات
P_1	1	2	1
P_2	2	0	1
P_3	2	2	1

مصفوفة التخصيص

(أ)

العملية	ماسحات	راسمات	طابعات
P_1	3	0	1
P_2	2	1	0
P_3	0	2	1

مصفوفة الاحتياج

(ب)

الشكل 4. 21: معطيات المثال الثاني لخوارزمية المصرف الخاصة بعدة مصادر من نفس النوع.

استنادًا على معطيات المثال نجد أن:

- مجموع المصادر المتوفرة للنظام = [ماسحات راسمات طابعات] = [5 5 5]
- مجموع المصادر المخصصة = [ماسحات راسمات طابعات] = [3 4 5]
- المصادر المتاحة = الفرق بين المجموعتين السابقتين = [2 1 0]

الخطوة الأولى:

- بناءً على متجه المصادر المتاحة لا يمكن الآن تنفيذ إلَّا العملية P_2 بالتالي ستُخصص المصادر لها.
- تُنفذ العملية P_2 بالكامل وستُحرر مصادرها ليصبح متجه المصادر المتاحة = [3 1 2].

الخطوة الثانية:

- بناءً على متجه المصادر المتاحة لا يمكن الآن تنفيذ إلا العملية P_1 بالتالي ستُخصص المصادر لها.
- تُنفذ العملية P_1 بالكامل وستُحرر مصادرها ليصبح متجه المصادر المتاحة = [4 3 3].

الخطوة الثالثة:

- الآن بناءً على متجه المصادر المتاحة يمكن تنفيذ العملية P_3 بالتالي ستُخصص المصادر لها.

- تُنفذ العملية P_3 بالكامل وستُحرر مصادرها ليصبح متجه المصادر المتاحة = [5 5 5].

خلاصة الأمر أن النظام في حالة آمنة والتسلل الآمن للتنفيذ سيكون على النحو P_2، P_1، P_3 بالتالي ستكون P_3 هي آخر عملية تُنفذ.

المثال الثالث: نظام تشغيل يستخدم خوارزمية المصرف لتجنب حدوث حالة الجمود من خلال إدارة تخصيص ثلاثة أنواع من المصادر لثلاث عمليات. يُوضح الشكل 4. 22 حالة النظام الحالية والممثلة باستخدام مصفوفة التخصيص (الشكل 4. 22–أ) ومصفوفة الحد الأقصى (الشكل 4. 22–ب). يحوي متجه المصادر المتاحة على ثلاث طابعات، وراسمتين، وماسحتين. الحالة الحالية للنظام هي حالة آمنة. آخذا في عين الإعتبار الطلبين المستقلين التاليين:

- الطلب الأول: العملية P_1 تطلب فقط ماسحتين إضافيتين.
- الطلب الثاني: العملية P_2 تطلب فقط طابعتين إضافيتين.

أيٍ من الجمل التالية صحيحة؟

1. يمكن فقط السماح بالطلب الأول.
2. يمكن فقط السماح بالطلب الثاني.
3. يمكن السماح بالطلبين.
4. لا يمكن السماح بأي طلب.

العملية	ماسحات	راسمات	طابعات
P_1	1	0	0
P_2	0	2	3
P_3	1	1	2

مصفوفة التخصيص

(أ)

العملية	ماسحات	راسمات	طابعات
P_1	3	4	8
P_2	0	2	6
P_3	3	3	3

مصفوفة الحد الأقصى

(ب)

الشكل 4. 22: معطيات المثال الثالث لخوارزمية المصرف الخاصة بعدة مصادر من نفس النوع.

استنادًا على معطيات المثال نجد أن:

- متجه المصادر المتاحة = [3 2 2].
- تُحسب مصفوفة الاحتياج من خلال $Need_{ij} = Max_{ij} - C_{ij}$ المبينة في الشكل 4. 23.
- من معطيات المسألة أن النظام في حالة آمنة (يُمكن أيضًا إثبات ذلك).

العملية	ماسحات	راسمات	طابعات
P_1	2	4	8
P_2	0	0	3
P_3	2	2	1

مصفوفة الاحتياج

الشكل 4. 23: مصفوفة الاحتياج الخاصة بالمثال الثالث.

التأكد مما إذا كان الطلب الأول يمكن السماح به بشكل مطلق: بما أن متجه المصادر المتاحة = [3 2 2] وطلب العملية هو [2 0 0]، فهذا يعني أنه بإمكان النظام تحقيقه، بالتالي ستفترض خوارزمية المصرف أن هذا الطلب سيُنفذ وسينتج عن ذلك الحالة الموضحة في الشكل 4. 24.

العملية	ماسحات	راسمات	طابعات
P_1	3	0	0
P_2	0	2	3
P_3	1	1	2

مصفوفة التخصيص

(أ)

العملية	ماسحات	راسمات	طابعات
P_1	3	4	8
P_2	0	2	6
P_3	3	3	3

مصفوفة الحد الأقصى

(ب)

العملية	ماسحات	راسمات	طابعات
P_1	0	4	8
P_2	0	0	3
P_3	2	2	1

مصفوفة الاحتياج

(ج)

الشكل 4. 24: نتائج السماح بتخصيص طلب العملية P_1.

- الآن أصبح متجه المصادر المتاحة = [0 2 3].
- يلي هذا قيام خوارزمية المصرف باختبار حالة النظام بعد عملية التخصيص المفترضة للتأكد من أنَّها حالة آمنة أم لا؟
- إذا كانت الحالة آمنة، فبإمكان النظام السماح بتحقيق الطلب الأول، وإلَّا فلا.

الخطوة الأولى:

- بناءً على متجه المصادر المتاحة لا يمكن الآن تنفيذ إلَّا العملية P_2 بالتالي ستُخصص المصادر لها.
- تُنفذ العملية P_2 بالكامل وستُحرر مصادرها ليصبح متجه المصادر المتاحة = [0 4 6].
- الآن أصبح من المستحيل تلبية أي طلب لأي عملية، مما يعني الدخول إلى حالة الجمود وهي حالة غير آمنة.
- بالتالي لن يُسمح بتحقيق الطلب الأول.

التأكد مما إذا كان الطلب الثاني يمكن السماح به بشكل مطلق: بما أن متجه المصادر المتاحة = [2 2 3] وطلب العملية هو [0 0 2]، فهذا يعني أنه بإمكان النظام تحقيقه، بالتالي ستفترض خوارزمية المصرف أن هذا الطلب سيُنفذه وسينتج عن ذلك الحالة الموضحة في الشكل 4. 25.

العملية	ماسحات	راسمات	طابعات
P_1	1	0	0
P_2	0	2	5
P_3	1	1	2

مصفوفة التخصيص

(أ)

العملية	ماسحات	راسمات	طابعات
P_1	3	4	8
P_2	0	2	6
P_3	3	3	3

مصفوفة الحد الأقصى

(ب)

العملية	ماسحات	راسمات	طابعات
P_1	2	4	8
P_2	0	0	1
P_3	2	2	1

مصفوفة الاحتياج

(ج)

الشكل 4. 25: نتائج السماح بتخصيص طلب العملية P_2.

- الآن أصبح متجه المصادر المتاحة = [1 2 2].

- يلي هذا قيام خوارزمية المصرف باختبار حالة النظام بعد عملية التخصيص المفترضة للتأكد من أنها حالة آمنة أم لا؟
- إذا كانت الحالة آمنة، فبإمكان النظام السماح بتحقيق الطلب الثاني، وإلَّا فلا.

الخطوة الأولى:

- بناءً على متجه المصادر المتاحة لا يمكن الآن تنفيذ إلَّا العملية P_2 بالتالي سيُخصص المصادر لها.
- تُنفذ العملية P_2 بالكامل وستُحرر مصادرها ليصبح متجه المصادر المتاحة = [6 4 2].

الخطوة الثانية:

- بناءً على متجه المصادر المتاحة لا يمكن الآن تنفيذ إلا العملية P_3 بالتالي سيُخصص المصادر لها.
- تُنفذ العملية P_3 بالكامل وستُحرر مصادرها ليصبح متجه المصادر المتاحة = [8 5 3].

الخطوة الثالثة:

- بناءً على متجه المصادر المتاحة لا يمكن الآن تنفيذ إلا العملية P_1 بالتالي سيُخصص المصادر لها.
- تُنفذ العملية P_1 بالكامل وستُحرر مصادرها ليصبح متجه المصادر المتاحة = [8 5 4].

خلاصة الأمر أن النظام في حالة آمنة والتسلل الآمن للتنفيذ سيكون على النحو P_2، P_3، P_1 بالتالي سيُسمح بتحقيق الطلب الثاني.

خلاصة القول: نُشرت خوارزمية المصرف لأول مرة من قبل ديكسترا في عام 1965، ومنذ ذلك الوقت لا يكاد يوجد أي كتاب يتناول أنظمة التشغيل إلَّا وتحدث عنها بنوع من التفصيل، كذلك هناك العديد من الأوراق التي لا حصر لها تناولت جوانب مختلفة منها. للأسف، عدد قليل من الكتاب أشاروا بجرأة أنه على الرغم من أن هذه الخوارزمية من الناحية النظرية رائعة، إلَّا إنَّها– من الناحية العملية– في الغالب غير مجدية، لأن العمليات نادرًا ما تكون على دراية مسبقة

بالحد الأقصى للمصادر التي تحتاجها. بالإضافة إلى ذلك، عدد العمليات غير ثابت ويتغير بتسجيل مستخدمين جدد، أو بخروج أحدهم. علاوةً على ذلك، يمكن للمصادر التي يُعتقد أنها متاحة الآن أن تختفي فجأةً (كأن أن يَعطل مشغل الأشرطة الممغنطة)، بالتالي في الممارسة العملية، وللأسباب السالفة الذكر هناك عدد قليل من النظم القائمة– إن وجدت– تستخدم خوارزمية المصرف لتجنب حالة الجمود.

8.4 منع حدوث الجمود

بعد أن رأينا أنَّ تجنب حدوث حالة الجمود أمرًا مستحيلًا في الأساس، لأنه يتطلب الحصول على معلومات حول الطلبات المستقبلية غير المعروفة، بالتالي السؤال الذي يطرح نفسه: هل من الممكن منع حدوث حالة الجمود؟ الجواب يتمثل في العودة إلى الشروط الأربعة– التي ذُكرت في القسم 4.4– لمعرفة ما إذا كان بالإمكان أنْ تُعطي أدنى فكرة حول عملية المنع، لأنه– على الأقل– في حالة ضمان عدم تحقق أحد هذه الشروط، فإن حدوث حالة الجمود سيكون أمرًا مستحيلًا. يتعرض هذا القسم لمناقشة منع حدوث حالة الجمود عن طريق منع حدوث على الأقل أحد هذه الشروط.

1.8.4 عدم السماح بمبدأ المنع التبادلي

تتمثل المحاولة الأولى لمنع حدوث حالة الجمود في منع تحقق شرط المنع التبادلي، بحيث إذا مُنِع تخصيص أي مصدر بشكل حصري لعملية واحدة فقط، فسوف لن تكون هناك حالة الجمود. المصادر القابلة للمشاركة لا تحتاج إلى هذا الشرط، بالتالي لن تكون عرضةً لحدوث هذه الحالة، من الأمثلة النموذجية لمثل هذه المصادر هي الملفات القابلة للقراءة فقط والتي بإمكان عدة عمليات طلب فتحها في نفس الوقت وكذلك ضمان وصول العمليات إليها من قبل النظام في وقت متزامن دون أي انتظار، ومن دون حدوث أي حالة جمود.

ولكن لا يمكن في العموم منع حدوث حالة الجمود عن طريق منع تحقق شرط المنع التبادلي، لأن بعض من المصادر غير قابلة للمشاركة على الاطلاق، فمن الواضح أن السماح لعمليتين باستخدام الطابعة في نفس الوقت سيؤدي إلى الفوضى، إلَّا إنَّه مع وجود التخزين اللحظي لمخرجات الطابعة (أي استخدام المكب) يُمكن لعدة عمليات توليد مخرجات في نفس

الوقت. في هذه الحالة، فإن العملية الوحيدة التي لها حق استخدام الطابعة الفعلية هي الطابعة الخفية (راجع القسم 1.3.2)، وبما أنها لا تطلب أية مصادر أخرى، بالتالي يُمكننا القضاء على حالة الجمود بالنسبة للطابعة الحقيقية.

هنا يجب ملاحظة التالي: إذا بُرمجت الطابعة الخفية بحيث تبدأ الطباعة حتى قبل أن تخرج كافة المخرجات إلى المكب، فإن ذلك سيُعرِّضها للخمول، وخصوصًا إذا قررت عملية الإخراج الانتظار لعدة ساعات بعد إخراج أول دفعة من المخرجات. لهذا السبب، عادة ما تُبرمج الطابعة الخفية للقيام بعملية الطباعة فقط بعد توفر ملف الإخراج بالكامل. ومع ذلك، يُمكن لهذا القرار في حد ذاته أن يؤدي إلى حالة الجمود. ماذا سيحدث لو أنَّ هناك عمليتين قامت كل منهما بشغل نصف حجم المكب المتاح بمخرجاتهما، الأمر الذي لن يُتيح لهما الفرصة لإكمال مخرجاتهما بالكامل، في هذه الحالة لن تستطيع العمليتان إنهاء عملهما، وهو ما سينتج عنه حالة الجمود بالنسبة للقرص. لتفادي هذا يُمكن تجنب تخصيص المصادر إلَّا في الحالات الضرورية على الاطلاق، ومحاولة التأكد من أن عدد قليل من العمليات يمتلك المصادر فعليًا.

2.8.4 عدم السماح بمبدأ الإمساك والانتظار

المنهجية الثانية المتبعة في منع حدوث حالة الجمود تتمثل في عدم السماح بتحقق مبدأ الإمساك والانتظار، بمعنى يجب أن نضمن أنه عندما تطلب العملية مصدرًا، فإنَّها لا تحتفظ بأية مصادر أخرى، وهو ما يُمكن تحقيقه من خلال أحد الطرق التالية:

- جعل العملية تطلب كافة مصادرها قبل بدء تنفيذها.
- جعل العملية تُحرر ولو بشكل مؤقت كافة المصادر التي تمتلكها حاليًا قبل أن تحاول الحصول على مصادر جديدة.

تُؤثر الطريقتان سلبيًا على كل من أداء نظام التشغيل، والاستفادة من المصادر، ومتوسط زمن انتظار العملية، وكذلك انتاجية النظام، كما أنَّ هناك احتمالية حدوث المجاعة، لأن أي عملية ذات أولوية منخفضة قد تكون عرضة للانتظار الأبدي، فيما يلي سيتم التطرق لهاتين الطريقتين بنوع من الإيجاز.

تتمثل الطريقة الأولى في جعل جميع العمليات تطلب كافة مصادرها قبل البدأ في التنفيذ،

إذا كان كل شيء متوفر، فسيُخصص كل ما تحتاجه العملية لها ومن ثم تُشغل حتى النهاية، أمَّا إذا لم يتوفر أي مصدر فلن يُخصص أي شيء، والعملية ستدخل في حالة الانتظار إلى أن تتوفر جميع مصادرها التي تحتاجها. تُعتبر هذه الطريقة من الناحية العملية أكثر ملاءمة لأنظمة الدفعة.

بالمقابل تتمثل الطريقة الثانية لكسر شرط الإمساك والانتظار في جعل العملية الراغبة في الحصول على أي مصدر تُفرج أولًا وبشكل مؤقت عن كل المصادر المخصصة حاليًا لها ، ومن ثم تحاول الحصول على كل ما تحتاجه في مرة واحدة، هذه الطريقة أكثر شيوعًا في الأنظمة التفاعلية.

لتوضيح الفرق بين هاتين الطريقتين، يُمكن النظر إلى عملية تنسخ البيانات من محرك أقراص الدي في دي إلى وحدة تخزين القرص، وتفرزها، ومن ثم تسحب النتائج على الطابعة. في الطريقة الأولى، يجب طلب جميع المصادر قبل بداية تنفيذ العملية، يعني يجب مبدئيًا طلب محرك أقراص الدي في دي، والقرص، والطابعة. في هذه الحالة سيُحتفظ بالطابعة إلى أن تُنفذ العملية بالكامل، على الرغم من أنها تحتاج الطابعة في النهاية فقط.

الطريقة الثانية تسمح للعملية بأن تطلب مبدئيًا فقط مشغل أقراص الدي في دي، ووحدة تخزين القرص لكي تتم عملية النسخ، بعد ذلك تُحرر العملية المصدرين، ومن ثم يجب عليها طلب القرص، والطابعة، عند تحقق ذلك تُنسخ البيانات من القرص إلى الطابعة وبعد الإنتهاء من ذلك تقوم بتحريرهما.

كل من هاتين الطريقتين لا يخلوَان من العيوب، فالأولى تعتمد على طلب كافة المصادر قبل بدء عملية التنفيذ، ولكن العديد من العمليات لا تعرف عدد المصادر التي تحتاجها إلَّا بعد البدء في الاشتغال. في الحقيقة، لو كانت العمليات تعلم ذلك، لأمكن استخدام خوارزمية المصرف.

المشكلة الثانية– وهي شائعة في الطريقتين– تتمثل في عدم الاستفاذة من المصادر بشكل أمثل. لتوضيح ذلك يمكن النظر إلى المثال التالي: بفرض أن هناك عملية تقرأ بيانات من لوحة المفاتيح، ومن ثم تُحللها لمدة ساعة، بعدها تُرسل النتائج إلى كل من شريط المخرجات، والراسمة، بالتالي إذا كان لا بدَّ من طلب جميع المصادر مسبقًا، فعلى هذه العملية الاحتفاظ بكل من مشغل الأقراص، والراسمة لمدة ساعة.

أخيرًا، كلتا الطريقتان عرضة لمشكلة المجاعة، فقد تضطر العملية التي تحتاج إلى عدة مصادر شائعة الاستخدام إلى الانتظار إلى أجل غير مسمى، لأن أحد المصادر الذي تحتاجه قد يُخصص دائمًا لبعض من العمليات الأخرى.

3.8.4 عدم السماح بمبدأ عدم حق السحب

شرط عدم حق السحب يعني أنه لا يمكن مقاطعة العملية أو اجبارها على تحرير أي مصدر تملكه، لذلك منع تحقق هذا الشرط قد يجعل منع حدوث حالة الجمود أمرًا ممكنًا.

- تكمن أحد الطرق المستخدمة لتحقيق هذا في جعل العملية– التي أُجبرت على انتظار مصدر جديد– تُحرر ضمنيًا كل المصادر التي خُصصت لها في السابق، ومن ثم اجبارها على طلبهم دفعة واحدة مرة ثانية بما في ذلك المصدر الجديد، وستكون إعادة تنفيذ العملية فقط عندما تتمكن من الحصول على مصادرها القديمة وكذلك المصادر الجديدة التي احتاجتها.
- هناك طريقة أخرى وهي أنه عند طلب أي مصدر من قبل عملية– ما– وكان هذا المصدر غير متوفر، فإن النظام يتطلع إلى معرفة ما إذا كانت هناك عملية أخرى تملكه وهي نفسها حاليًا في حالة انتظار لبعض من المصادر الأخرى، إذا عُثر على مثل هذه العملية، فقد يُسحب هذا المصدر منها ويُضاف إلى قائمة المصادر التي تنتظرها العملية الأولى، أمَّا فيما يخص العملية الأخيرة فلن يُعاد تنفيذها إلَّا عند تخصيص المصادر الجديدة التي تطلبها واسترداد أي مصدر سُحب منها في أثناء انتظارها.

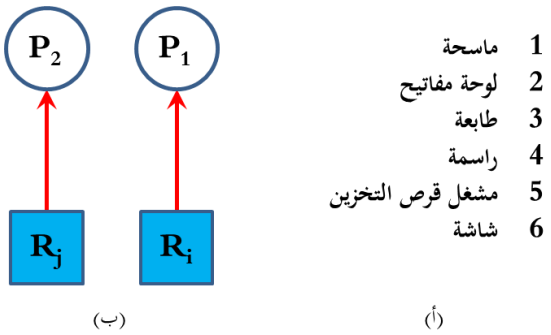
قد يكون من الممكن تطبيق أيٍ من هاتين الطريقتين على المصادر التي تُحفظ، وتُستعاد حالاتها بسهولة، مثل سجلات وحدة المعالجة المركزية، والذواكر، لكنها لا تنطبق في العموم على الأجهزة الأخرى مثل، الطابعات، ومحركات الأشرطة. ولكن إذا خُصصت الطابعة لعملية– ما– وفي منتصف الطباعة سُحبت منها قسرًا، لأنها احتاجت مثلًا الراسمة غير المتوفرة حاليًا، يُعتبر أمرًا صعبًا في أحسن الأحوال، ومستحيلًا في أسوئها. ومع ذلك، يُمكن محاكاة بعض من المصادر لتجنب هذا الوضع مثل، استخدام مكب الطباعة على القرص لاستقبال مخرجات عملية الطباعة، والسماح فقط للطابعة الخفية للوصول إلى الطابعة الحقيقية، الأمر الذي سيَحُد من حالة

الجمود الناتجة عن الطابعة.

4.8.4 عدم السماح بمبدأ الانتظار الدائري

يَتَمثل عدم السماح بمبدأ الانتظار الدائري في عدم تهيئة الفرصة لنشوء حلقة انتظار، الأمر الذي يُمكن تحقيقه باستخدام عدة طرق، أبسطها تتمثل في الآتي: من حق أي عملية استخدام مصدر واحد فقط، وفي حالة ما احتاجت إلى مصدر آخر، يجب عليها إعادة المصدر الأول. يُعتبر هذا الحل تقييدًا غير مقبول بالنسبة لعملية تحتاج لنسخ ملف ضخم من وحدة تخزين القرص إلى الطابعة.

تفترض الطريقة الأخرى لضمان عدم وجود شرط الانتظار الدائري بشكل دائم ترقيم كافة المصادر بترتيب عددي لجميع أنواعها كما هو موضح في الشكل 4. 26–أ، بالتالي من حق كل عملية أن تطلب أي مصدر في أي وقت تشاء، ولكن شرط أن يتم ذلك حسب الترتيب العددي للمصادر. مثلًا، بإمكان أي عملية طلب الطابعة أولًا ثم الراسمة، أما العكس فغير مسموح به. وبعبارة أخرى، إذا كان المصدر R_j مُخصص لعملية– ما– ورغبت في طلب أي مصدر R_i يجب أن تكون رتبته i أقل من j.



الشكل 4. 26: أ) الترقيم العددي للمصادر– ب) الرسم البياني للمصدر.

باستخدام هذه القاعدة لا يمكن أن تنشأ أي حلقة في الرسم البياني لتخصيص المصادر. المثال التالي يوضح صحة هذه الفكرة في حالة تطبيقها على عمليتين (مع العلم بأنها قابلة للتطبيق على أكثر من ذلك). في الشكل 4. 26–ب، يُمكن أن تحدث حالة الجمود فقط في حالة طلبت كل من العملية P_1 المصدر R_j، والعملية P_2 المصدر R_i، على افتراض أن R_i و

R_j هما مصدران مختلفان، أي لهما رقمان مختلفان، بالتالي لو كانت **i** أكبر من **j**، فإنه لن يُسمح للعملية P_1 بطلب R_j، لأن رتبته أقل من رتبة المصدر المخصص لها بالفعل، بينما إذا كانت **j** أكبر من **i**، فلن يُسمح للعملية P_2 بطلب R_i، لأن رتبته أقل مما هو لديها بالفعل، عليه سيستحيل في الحالتين حدوث حالة الجمود.

هناك تغيير بسيط في هذه الخوارزمية يتمثل في إسقاط شرط أن يتم الحصول على المصادر في تسلسل متزايد، واستبداله بأن لا يحق للعملية الطالبة للمصدر أن تطلب مصدر أقل رتبة من المصدر المالكة له بالفعل. مثلًا، إذا طلبت عملية– ما– في البداية المصدرين **9** و **10**، ثم حررتهما، فإن هذه العملية ستبدأ بشكل فعلي من جديد، وسوف لن يكون هناك أي سبب يمنعها من طلب المصدر **1**.

على الرغم من أن الترقيم العددي للمصادر حل مشكلة حالة الجمود، إلَّا إنَّه قد يكون من المستحيل العثور على ترقيم نسبي للمصادر يُرضي الجميع. عندما تشمل المصادر جداول العمليات، ومساحة المكب على القرص، وسجلات قاعدة البيانات المؤمنة، وكذلك المصادر المجردة الأخرى، سيكون عدد المصادر والطرق المختلفة لاستخداماتها كبيرًا جدًا بحيث يكون من الصعب الحصول على ترقيم مناسب.

9.4 مواضيع أخرى

في هذا القسم سوف نناقش بعض من القضايا المتنوعة والمتعلقة بحالة الجمود التي تشمل تأمين المرحلتين، وحالة الجمود لغير المصادر، وكذلك التجويع.

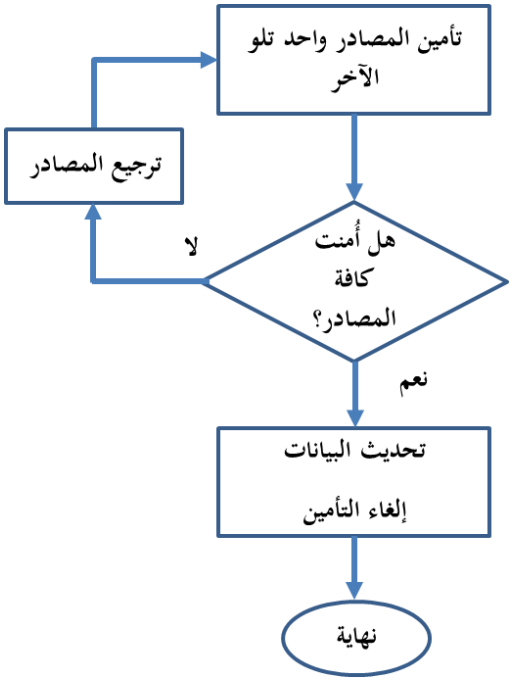
1.9.4 خوارزمية تأمين المرحلتين

تُستخدم خوارزمية تأمين المرحلتين لاكتساب المصادر المشتركة من دون أن تكون هناك احتمالية لحدوث حالة الجمود، فهي تقنية بسيطة، وخاصة ومُعَدة لتطبيقات محددة في مجال أنظمة قاعدة البيانات بسبب عدم كفاءة طرق كل من تجنب، ومنع حالة الجمود. تنقسم هذه الخوارزمية إلى مرحلتين، لذلك يُطلق عليها تأمين المرحلتين وهي تتمثل في:

- المرحلة الأولى وتشمل نشاط اكتساب تأمين السجلات (البيانات).

- المرحلة الثانية وتشمل نشاط تحديث (تعديل) السجلات، ونشاط إلغاء حالة التأمين.

تُنفذ المرحلة الأولى بالدخول في حلقة تُحاول فيها الخوارزمية تأمين كافة السجلات التي ترغب في مشاركتها واحدا تلو الآخر، إذا لم تنجح الخوارزمية في تأمين أي سجل مطلوب، فستُلغي جميع حالات التأمين للسجلات الخاصة بها، ومن ثم تُعيد هذه المرحلة من جديد (تشبه هذه الطريقة إلى حد ما عملية الطلب المسبق لجميع المصادر اللازمة)، أمَّا في حالة نجاح هذه المرحلة، فستبدأ المرحلة الثانية والمتمثلة في تحديث (تعديل) السجلات، وإلغاء حالة التأمين، كما هو موضح في الشكل 4. 27.



الشكل 4. 27: خوارزمية تأمين المرحلتين.

إذا تحققت عملية تأمين المصادر بالكامل فهذا يعني أنه لا توجد أي عملية تحتفظ ببعض من المصادر المشتركة وتنتظر قيام عملية أخرى بتحرير مصدر مشترك ترغب في امتلاكه، هذا يعني أن الجمود لا يمكن أن يحدث بسبب التنازع على المصادر. في بعض من الإصدارات لهذه الطريقة قد لا يتم تحرير وإعادة استخدام سجل مؤمن خلال المرحلة الأولى، الأمر الذي قد يجعل

حدوث حالة الجمود ممكنًا.

من الملاحظ أنه بالإمكان أن تكون سياسة 'إلغاء جميع حالات التأمين، وإعادة المحاولة'– التي سبق ذكرها– مشكلة في الأنظمة الموزعة، لأنه قد يكون من غير المضمون إتاحة الوصول إلى المصادر المطلوبة في غضون فترة زمنية محدودة، الأمر الذي قد يؤدي إلى حالة مجاعة العملية، أي عدم تمكن عملية واحدة بشكل أبدي من تأمين جميع مصادرها اللازمة لها لكي تستمر في تنفيذ مهمتها.

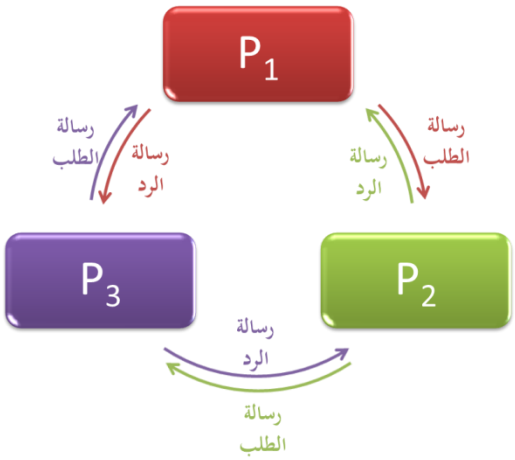
هذا الأمر يُمثل أيضًا مشكلة حقيقية لأنظمة الوقت الحقيقي، وأنظمة التحكم في العمليات. بالتالي لا يمكن استخدام خوارزمية تأمين المرحلتين في التطبيقات ذات طابع الزمن الحقيقي غير المرن، كما أنه من غير المقبول إنهاء عملية وصلت إلى منتصف التنفيذ، وتشغيلها من جديد بسبب عدم توفر أحد المصادر. كذلك ليس مقبولًا أن تبدأ عملية من جديد بعد أن قامت بقراءة أو كتابة رسائل على الشبكة، أو قامت بتحديث ملفات، أو أي شيء آخر لا يمكن تكراره بأمان. هذه الخوارزمية تعمل فقط في الحالات التي يُرتب فيها المبرمج الأحداث بعناية فائقة، بحيث يمكن إيقاف البرنامج في أي وقت في أثناء المرحلة الأولى وإعادة تشغيله، علمًا بأن العديد من التطبيقات لا يمكن تنظيمها بهذه الطريقة.

2.9.4 جمود الاتصالات

أشرنا في بداية هذا الفصل إلى أنَّ هناك عدة أنواع لحالة الجمود منها حالة جمود المصادر وهي أكثرهم شيوعًا، بالإضافة إلى حالة جمود الاتصالات والتي تحدث في الغالب في الشبكات. فمن المعروف أن التواصل بين العمليات في إطار الشبكات يتم عن طريق بعث الرسائل، كما هو مبين في الشكل 4. 28، بحيث تُرسل العملية $\mathbf{P_1}$ رسالة طلب إلى العملية $\mathbf{P_2}$، ومن ثم تتوقف إلى أن تبعث العملية $\mathbf{P_2}$ رسالة الرد، الأمر نفسه يحدث مع بقية العمليات داخل الشبكة. لنفترض أن رسالة الطلب قد ضاعت ولم تصل إلى $\mathbf{P_2}$، هذا الأمر يجعل $\mathbf{P_1}$ تتوقف انتظارًا منها لرسالة الرد من $\mathbf{P_2}$ التي ستتوقف هي الأخرى بسبب انتظارها للطَّلب المرسل من $\mathbf{P_1}$، تؤدي هذه الحالة إلى دخول العمليتين إلى حالة الانتظار الأبدي، أي حالة جمود الاتصالات.

هنا يجب ملاحظة أنه في حالة جمود الاتصالات لا يوجد أي نوع من المصادر على

الإطلاق، فالعملية P_1 لا تمتلك أي مصدر تحتاجه العملية P_2، والعكس بالعكس، وإنَّما سُمِّي بجمود الاتصالات بناءً على تعريف حالة الجمود، الذي يتضمن وجود مجموعة من العمليات، كلها موقوفة بسبب انتظارها لحدث يحدث فقط بسبب عملية أخرى داخل نفس المجموعة. على هذا الأساس لا يُمكن منع أو تجنب هذا النوع من الجمود بنفس الطرق المتبعة مع حالة جمود المصادر، إلَّا إنَّه من الممكن استخدام مؤقت لفكه عن طريق مراقبة زمن الإرسال والاستقبال. في معظم أنظمة الاتصالات الشبكية، من المتوقع الرد على أي رسالة أُرسلت من عملية إلى أخرى ضمن فترة زمنية محددة، لذلك إذا إنتهت المهلة المحددة قبل وصول الرد، فسيفترض المرسل أن الرسالة فُقدت، وبناءً عليه سيُعيد إرسالها مرة أخرى، بل مرات أخرى، إذا لزم الأمر، وبهذه الطريقة سيتم منع حالة الجمود.



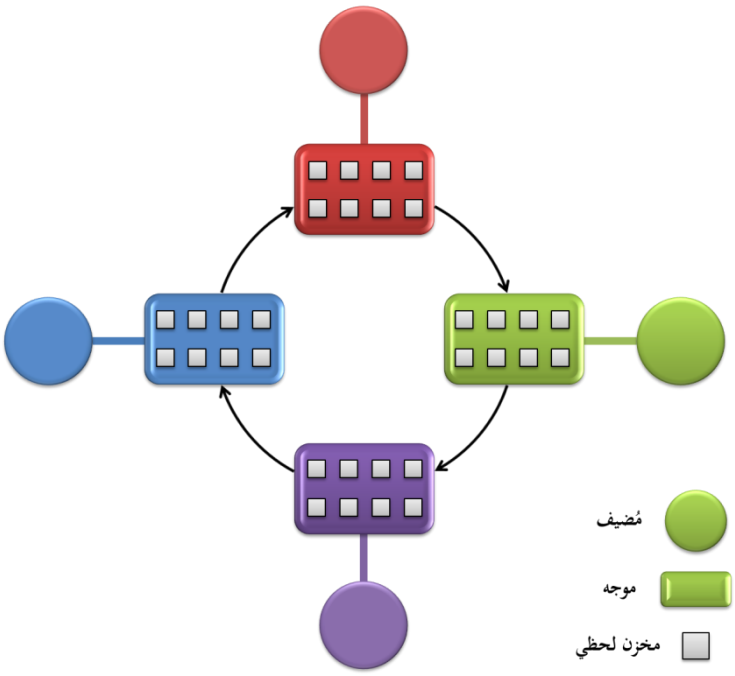
الشكل 4. 28: حالة جمود الاتصالات.

قد ينتج عن هذا الحل استلام المرسل لرسالة الرد مرتين أو أكثر، وربما ينتج عنه عواقب غير مرغوب فيها وخصوصًا عندما يتعلق الأمر بنظام الدفع الإلكتروني الذي قد تحوي الرسالة فيه على تعليمات خاصة بالدفع، بالتالي لا ينبغي تكرار وتنفيذ هذه الرسالة عدة مرات لمجرد أن الشبكة بطيئة، أو زمن المؤقت قصير جدًا.

حالة جمود المصادر هي الأخرى قابلة للحدوث في نظم الاتصالات، فبالنظر إلى الشبكة الممثلة في الشكل 4. 29، نجد أنها تتكون من جهازين رئيسين هما: المضيف، والموجه.

فالأول يُمثل حاسوب مستخدم يُولِج من خلاله إلى الشبكة ويُقدم الخدمات لروادها، أمَّا الموجه فهو جهاز اتصالات متخصص في توجيه حزم البيانات من المصدر إلى الهدف داخل مسار الشبكة.

يستخدم الموجه عددًا من المخازن اللحظية للتحكم في عملية توجيه الحزم عبر الشبكة، تُوضع هذه الحزم عند وصولها إلى الموجه في المخزن اللحظي لكي يُرسِلها لاحقًا إلى الموجه الذي يليه في المسار، ثم إلى الذي يليه، وهكذا حتى تصل إلى الهدف المحدد. تُمثل هذه المخازن المصادر، وهي ذات سعة، وعدد محدودين. بالرجوع إلى الشكل 4. 29، بفرض أنَّ لدى كل موجه ثمانية مخازن، وأنَّ هناك حاجة لتوجيه جميع الحزم من الموجه الأول إلى الثاني، ومنه إلى الثالث، ومنه إلى الرابع، ومنه إلى الموجه الأول من جديد، هذه الوضعية لا تُتيح فرصة توجيه أي حزمة من أي موجه إلى آخر، لأنه لا يوجد مخزن لحظي فارغ في الطرف الآخر، بالتالي سنواجه هنا حالة جمود المصادر الكلاسيكية.



الشكل 4. 29: حالة جمود المصادر في نظم الاتصالات الشبكية.

3.9.4 المجاعة

تطرقنا فيما سبق ذكره في هذا الفصل لأكثر من مرة إلى مصطلح المجاعة، والتي هي من الظواهر المرتبطة بشكل وثيق مع حالة الجمود، والتي تحدث في الأنظمة الديناميكية في عدة صور منها ما هو مرتبط مع طلب المصادر، ومنها ما هو مرتبط بخوارزميات جدولة العمليات وخصوصًا مجدول الأولوية. ففي الحالة الأولى تكون هناك حاجة إلى عدة سياسات خاصة باتخاذ قرارات تخصيص المصادر للعمليات، إلَّا إنَّ البعض منها قد يُؤدي– على ما يبدو– إلى عدم حصول بعض من العمليات على أي خدمة أو أي مصدر، على الرغم من أنها ليست ضمن حالة الجمود.

فبالنظر إلى إحدى السياسات المتبعة في تخصيص الطابعة للعمليات عندما تتنافس مجموعة منها في نفس الوقت على الحصول على هذا المصدر، نجد أنها تعتمد على فلسفة تخصيص الطابعة أولًا للعملية صاحبة أصغر ملف ومن ثَم لصاحبة الملف التالي في الحجم، وهكذا مع فرضية أنَّ معلومات حجم كل ملف متوفرة. للوهلة الأولى تبدو أن هذه السياسة عادلة وخصوصًا إذا ما كان هناك دفق مستمر من العمليات ذات الملفات المتقاربة في الحجم، ولكن ماذا سيحدث عندما تظهر عملية ذات ملف كبير؟ بناءً على الفلسفة المتبعة سيختار النظام دائمًا العمليات ذات الملفات الأصغر حجمًا، وسيؤجل العملية مالكة الملف الكبير ربما إلى أجل غير مسمى على الرغم من أنها غير موقوفة بسبب حالة الجمود، الأمر الذي قد يؤدي ببساطة إلى تجويع هذه العملية حتى الموت. من الممكن تجنب هذا النوع من المجاعة عن طريق تخصيص المصادر تبعًا لسياسة من يأتي أولًا، يُخدم أولًا، والتي ستخدم كافة الطلبات بشكل عادل.

أمَّا بخصوص حالة المجاعة المتعلقة بخوارزمية مجدول الأولوية فهي تتمثل في الحالات التي يكون فيها النظام الحاسوبي مُحمَّل بكثافة مع وجود عملية ذات أهمية منخفضة وجاهزة للتنفيذ، لكنها موقوفة بسبب تدفق تيار من العمليات ذات أهميات أعلى منها الأمر الذي سيمنعها لفترة زمنية طويلة وغير محددة من أخذ حصتها من المعالج. بالمقابل يتمثل حل هذا النوع من المجاعة في استخدام مبدأ الشيخوخة، أي في الزيادة التدريجية لأولوية العمليات التي انتظرت لفترة زمنية طويلة داخل النظام. على سبيل المثال، إذا كان مدى الأولوية من **0** (أعلى أهمية) إلى **100** (أقل أهمية)، فيُمكن زيادة أهمية أي عملية بمقدار وحدة واحدة كلما انتظرت

عشر دقائق، هذا يعني أن العملية ذات أقل أهمية (100) لن تستغرق أكثر من 17 ساعة حتى تصبح أهميتها 0.

يتلخص الفرق بين حالة الجمود وحالة المجاعة في إطار نظم التشغيل في الجدول 4. 1.

الجدول 4. 1: أهم الفروقات بين حالة الجمود وحالة المجاعة.

حالة الجمود	حالة المجاعة
تحدث عندما لا تستطيع أي عملية داخل مجموعة عمليات مواصلة تنفيذها بسبب احتجاز المصادر من قبل عمليات أخرى داخل نفس المجموعة.	تحدث عندما تنتظر عملية لفترة طويلة وغير محدودة لاكتساب مصدر تحتاجه.
يتوقف استخدام المصادر من قبل العمليات.	يستمر استخدام المصادر من قبل العمليات عالية الأهمية.
تنتظر كل عملية في الأخرى لكي تُكمل عملها، ولن تُنفذ أيٍ منها.	يستمر تنفيذ العمليات عالية الأهمية، بينما تتوقف العمليات ذات الأهميات المنخفضة.
تتسبب في وقف النظام بالكامل.	تستمر عمليات النظام في العمل، باستثناء العملية الضحية.
يُمكن منعها بمنع تحقق شروطها.	يُمكن منعها بمبدأ الشيخوخة، أو بمبدأ من يأتي أولًا، يُخدم أولًا.

10.4 موجز الفصل

بدأت رحلة هذا الفصل بمدخل لحالة الجمود قدَّم فيه الفصل بعض من الأسباب الكامنة وراء هذه الحالة والتي من أهمها الاستعمال القصري، أي الحصري للمصادر، بمعنى استخدامها من قبل العملية الأولى حتى النهاية، ثم التالية حتى النهاية، وهكذا. يلي هذا المدخل تناول الفصل التعريف التفصيلي لحالة جمود المصادر والذي بين أنها تحدث عندما تتوقف كافة أعضاء مجموعة من العمليات بسبب انتظارها لحدث يحدث فقط من قبل عملية أو عمليات أخرى في نفس المجموعة، الأمر الذي يؤدي إلى انتظار جميع العمليات إلى الأبد، عادة ما يتمثل هذا الحدث في تحرير أحد المصادر المملوك من عضو آخر في المجموعة.

وللدور المهم الذي تلعبه المصادر في حالة جمود المصادر، خُصِّص لها قسم في هذا

الفصل تناول تعريفها، وأنواعها المتمثلة في المصادر القابلة للسحب، والأخرى غير القابلة للسحب. فالأولى يُقصد بها المصادر التي يمكن سحبها من العملية المالكة لها دون حدوث أي آثار سيئة، ولا تنتج عنها حالة جمود المصادر، أمَّا الثانية فهي تلك المصادر، التي إذا ما سُحبت من مالكها الحالي سيؤدي إلى فشل العملية، أو حدوث أخطاء، وقد تسبب بشكل أساسي في حدوث حالة جمود المصادر. لذلك من المهم جدًا أن تكون هناك إدارة تُعنى بآليات طلب، واستخدام، وتحرير المصادر بشكل دقيق أملًا في تجنب حدوث حالة الجمود.

على هذا الأساس قدَّم الفصل توصيفًا لحالة جمود المصادر من خلال التعريف بالبيئة الملائمة لها، أي الشروط التي تساعد في حدوثها، من أجل المساهمة في حل هذه المشكلة، أو على الأقل تقليل الأضرار الناجمة عنها. تمثلت هذه الشروط في شرط المنع التبادلي، وشرط الاحتجاز والانتظار، وشرط عدم حق السحب، وأخيرًا شرط الانتظار الدائري، والتي نُمذِجت بواسطة رسوم بيانية موجهة لتسهيل عملية التوصيف ونمذجة حالة الجمود. لذلك ضرب الفصل عدة أمثلة توضيحية لمجموعة من حالات النظام المتعددة لغرض التعرف على كيفية استخدامها في تحديد إمكانية حدوث حالة جمود المصادر من عدمها.

يلي ذلك تناول الفصل أربع استراتجيات خاصة بالتعامل مع حالة جمود المصادر تمثلت في تجاهل المشكلة، والاكتشاف والتعافي منها، والتجنب الحيوي لها، وأخيرًا منعها. تُعرف الاستراتيجية الأولى بسياسة النعامة، أي التظاهر بعدم وجود مشكلة والتصرف كأن شيء لم يحدث، بينما تتمحور الاستراتيجية الثانية حول السماح بحدوث المشكلة، ومن ثم محاولة اكتشافها أولًا، والتعافي منها ثانيًا، بالتالي عرض الفصل طرق لاكتشاف حالة الجمود في وجود مصدر واحد من كل نوع، وفي وجود عدة مصادر من كل نوع، كما ناقش عدة آليات للتعافي منها إمَّا عن طريق إجهاض العمليات، أو سحب المصادر، أو استخدام نقاط الفحص.

أمَّا فيما يخص استرتيجية التجنب الحيوي لحالة الجمود فقد وضَّح الفصل هذا المفهوم من خلال استراتيجية مسارات المصادر لتقديم مفهومي الحالة الآمنة وغير الآمنة، واللَّتان تعتمد عليهما آلية التجنب بشكل كبير، فالأولى يُقصد بها الحالة التي توجد فيها سلسلة من الأحداث تضمن تنفيذ جميع العمليات حتى النهاية دون أن تحدث حالة الجمود، أمَّا الثانية فتعني الحالة التي لا توجد فيها مثل هذه الضمانات، وأن الاستمرار في التنفيذ سيؤدي حتمًا إلى حالة الجمود.

ولزيادة فهم هاتين الحالتين قدَّم الفصل أمثلة توضيحية بالخصوص.

مفهوم الحالة الآمنة، وغير الآمنة بُنيت عليه خوارزمية المصرف والتي هي امتداد لخوارزميات اكتشاف حالة الجمود وتعمل على تجنبها من خلال عدم تحقيق أي طلب، طالما سيُعرض النظام للخطر وينقله إلى الحالة غير الآمنة. بناءً عليه ناقش الفصل كل من خوارزمية المصرف الخاصة بمصدر واحد من كل نوع، والخاصة بعدة مصادر من نفس النوع.

ولأنَّ تجنب حدوث حالة الجمود قد يكون أمرًا مستحيلًا لعدة أسباب منها استحالة توفر المعرفة المسبقة للحد الأقصى لمتطلبات كل عملية، وتغير عدد العمليات بتسجيل مستخدمين جدد، أو بخروج أحدهم، ناقش الفصل أيضًا استراتيجية منع حدوث حالة الجمود عن طريق منع تحقق على الأقل أحد شروطها والمتمثلة في: شرط المنع التبادلي والذي من الممكن منعه من خلال استخدم فكرة المكب لكل شيء، وشرط الإمساك والانتظار والمتمثل منعه في السماح لأي عملية بامتلاك مصدر واحد فقط في أي لحظة معطاة، أو الطلب المبدئي لكل المصادر التي تحتاجها العملية قبل البدأ في تنفيذها، وشرط عدم حق السحب والذي يُعالج بسحب المصدر بعيدًا عن العملية، وأخيرًا شرط الانتظار الحلقي الذي لن يتحقق بترقيم المصادر عدديًا وطلبها وفق تسلل دقيق حسب تصاعد الأرقام.

أخيرًا، يعرض الفصل بعض من القضايا المتنوعة والمتعلقة بحالة الجمود شملت تأمين المرحلتين، وحالة جمود الاتصالات، وكذلك المجاعة. تُستخدم خوارزمية تأمين المرحلتين لاكتساب المصادر المشتركة في مجال أنظمة قاعدة البيانات من دون أن تكون هناك احتمالية لحدوث حالة الجمود، وهي تتم على مرحلتين، تشمل الأولى نشاط اكتساب تأمين السجلات، والثانية نشاطي تحديث السجلات، وإلغاء حالة التأمين. أمَّا حالة جمود الاتصالات فهي تحدث عندما تتواصل مجموعة من العمليات عن طريق الرسائل، بحيث تنتظر كل عملية في رسالة، في حين أنَّ قناة التواصل فارغة، غالبًا ما تُعالج هذه الحالة بتحديد مهلة زمنية مناسبة عن طريق مؤقت. من ناحية أخرى تأخذ المجاعة عدة صور منها المرتبط بطلب المصادر والتي من الممكن التغلب عليها بسياسة التخصيص وفقًا لآلية من يأتي أولًا يُخدم أولًا، ومنها المرتبط بمجدول الأولوية والتي يُمكن تجنبها من خلال زيادة الأولوية تدريجيًا للعمليات التي انتظرت لفترة زمنية طويلة داخل النظام وهو ما يُعرف بمبدأ الشيخوخة.

11.4 أسئلة للمراجعة

1. عرف حالة الجمود، مع ذكر بعض من الأسباب المؤدية إلى حدوثها.
2. عرف المصادر واذكر أنواعها، وما المقصود بكل نوع؟
3. يُخصص متغير إشارة في إدارة استخدام المصادر لضمان تنفيذ خطوات طلب، واستخدام، وتحرير المصدر كعملية واحدة غير قابلة للمقاطعة، علل ذلك؟
4. هناك بيئة ملائمة لحدوث حالة الجمود تتمثل في أربعة شروط ضرورية لتحقق جمود المصادر، أذكرها.
5. ما فائدة نمذجة حالة الجمود؟
6. ما الاستراتجيات التي يُمكن التعامل بها مع حالة الجمود؟
7. بفرض أن هناك ثلاث عمليات وثلاثة مصادر، وكان طلب وإعادة هذه المصادر من قبل العمليات على النحو الموضح بالكيفية التالية:

(1)	(2)
1. P_1 تطلب R_2	1. P_1 تطلب R_2
2. P_2 تطلب R_3	2. P_3 تطلب R_3
3. P_3 تطلب R_1	3. P_1 تطلب R_1
4. P_1 تطلب R_3	4. P_3 تطلب R_2
5. P_2 تطلب R_1	5. P_2 تطلب R_1
6. P_3 تطلب R_3	6. P_3 تُحرر R_3

باستخدام نمذجة حالة الجمود، بين بالرسم هل سيؤدي تخصيص هذه الطلبات إلى حدوث حالة الجمود أم لا؟

8. استجابة الناس لاستراتيجية النعامة في إطار حالة الجمود تتفاوت من فئة إلى أخرى، ناقش ذلك.
9. بفرض أن هناك نظام ذو سبع عمليات، $\mathbf{P_1}$ إلى $\mathbf{P_7}$، وستة مصادر، $\mathbf{R_1}$ إلى $\mathbf{R_6}$، حالة المصادر التي حاليًا مخصصة، وتلك التي حاليًا مطلوبة هي كما يلي:
   أ. العملية $\mathbf{P_1}$ مُخصص لها $\mathbf{R_1}$ وترغب في الحصول على $\mathbf{R_2}$.
   ب. العملية $\mathbf{P_2}$ ليس مُخصص لها أي مصدر، لكنها ترغب في الحصول على $\mathbf{R_3}$.

ج. العملية P_3 ليس مُخصص لها أي مصدر، لكنها ترغب في الحصول على R_2.

د. العملية P_4 مُخصص لها R_4 وترغب في الحصول على R_2 و R_3.

هـ. العملية P_5 مُخصص لها R_3 وترغب في الحصول على R_5.

و. العملية P_6 مُخصص لها R_6 وترغب في الحصول على R_2.

ز. العملية P_7 مُخصص لها R_5 وترغب في الحصول على R_4.

هل وصل هذا النظام إلى حالة الجمود؟ إذا كان الأمر كذلك، أيٍ من هذه العمليات تتشارك فيه؟

10. بفرض أن هناك نظام يتكون من أربع عمليات P_1 إلى P_4، وخمسة أنواع من المصادر R_1 إلى R_5، وكانت حالته على النحو التالي:

متجه المصادر الموجودة

$E = (2\ 4\ 1\ 4\ 4)$

مصفوفة التخصيص الحالية C

0	1	1	1	2	P_1
0	1	0	1	0	P_2
0	0	0	0	1	P_3
2	1	0	0	0	P_4

متجه المصادر المتاحة

$A = (0\ 1\ 0\ 1\ 2\ 1)$

مصفوفة الطلبات R

1	1	0	2	1	P_1
0	1	0	2	1	P_2
0	2	0	3	1	P_3
0	2	1	1	0	P_4

باستخدام خوارزمية كشف حالة الجمود في وجود عدة مصادر من كل نوع، بين أن هناك حالة جمود في هذا النظام، وحدد العمليات الموقوفة بسببها.

11. لنفترض أن في الشكل 4. 12، $C_{ij} + R_{ij} > E_j$ لبعض قيم **i**. ما الآثار المترتبة عن هذا على النظام؟

12. كيف يُمكن تجنب حالة الجمود؟

13. ما المقصود بالحالة الآمنة، والحالة غير الآمنة؟ وأيٍ منهما تؤدي إلى حالة الجمود؟

14. نظام لديه أربع عمليات وخمسة مصادر مخصصة، التخصيص الحالي والاحتياجات القصوى للمصادر كما يلي:

	مصفوفة المخصص					مصفوفة الحد الأقصى					متجه المصادر المتاحة				
P_1	1	2	0	1	1	P_1	3	1	2	1	1	1	1	x	0	0
P_2	0	1	1	0	2	P_2	0	1	2	2	2					
P_3	0	1	0	1	1	P_3	0	1	3	1	2					
P_4	0	1	1	1	1	P_4	1	2	2	1	1					

ما أصغر قيمة يمكن أن تأخذها x تجعل هذا النظام في حالة آمنة؟

15. **ما الخيارات الممكنة للتعافي من حالة الجمود؟ وأيُّهما أفضل من وجهة نظرك؟ وضِّح اجابتك.**
16. **ما الفرق الرئيسي بين النموذج الموضح في الشكل 4. 15، ووصف الحالة الآمنة وغير الآمنة في الجزء 2.7.4، وما النتائج المترتبة عن هذا الاختلاف؟**
17. **بفرض أن هناك نظام له ست عمليات P_1 إلى P_6 ، وثلاثة مصادر R_1 إلى R_3. هل سيكون تنفيذ العمليات التالية في حالة آمنة؟ وضِّح اجابتك.**

	مصفوفة المخصص				مصفوفة الحد الأقصى				متجه المصادر المتاحة		
	R_3	R_2	R_1		R_3	R_2	R_1		R_3	R_2	R_1
P_1	0	2	1	P_1	2	2	2	P_1	0	1	0
P_2	0	0	1	P_2	0	1	1	P_2			
P_3	1	1	1	P_3	3	4	1	P_3			
P_4	1	1	0	P_4	1	1	1	P_4			
P_5	1	0	0	P_5	2	2	1	P_5			
P_6	0	0	1	P_6	1	5	1	P_6			

18. **جميع المسارات في الشكل 4. 15، إمَّا أفقية أو عمودية. هل يمكنك تصور أي ظرف من الظروف يمكن للمسارات أن تكون تحته قطرية؟**
19. **هل يمكن لمخطط مسار المصادر في الشكل 4. 15 أن يُستخدم كذلك لتوضيح مشكلة حالة الجمود لثلاث عمليات وثلاثة مصادر؟ إذا كان الأمر كذلك، كيف يمكن القيام بذلك؟ إذا لم يكن كذلك، لماذا لا؟**
20. **بإلقاء نظرة دقيقة على الشكل 4. 19–ب، إذا قام يوسف بطلب مصدر آخر، هل سيؤدي**

ذلك إلى حالة آمنة أو غير آمنة؟ ماذا لو جاء الطلب من يونس بدلًا من يوسف؟

21. نظام لديه عمليتين وثلاثة مصادر متطابقة، كل عملية تحتاج كحد أقصى إلى مصدرين، هل من الممكن حدوث حالة الجمود؟ وضح إجابتك.

22. بفرض أن العملية **A** في الشكل **4**. **20** طلبت آخر مشغل للأشرطة الممغنطة، هل سيؤدي هذا إلى حالة الجمود؟

23. لماذا تجنب حدوث حالة الجمود يُعتبر أمرًا صعبًا إن لم يكن مستحيلًا؟

24. كيف يُمكن عدم السماح بمبدأ المنع التبادلي في إطار منع حدوث حالة الجمود؟

25. بفرض أن هناك نظام يملك مصدرين R_1 و R_2 وإجمالي عدد مثيلات المصدر الأول هو **6**، والثاني هو **3**. هل من الممكن تنفيذ العمليات التالية دون أن تحدث حالة الجمود؟

مصفوفة المخصص

R_2	R_1	
1	1	P_1
0	1	P_2
0	1	P_3
1	0	P_4
1	2	P_5

مصفوفة الحد الأقصى

R_2	R_1	
2	2	P_1
2	4	P_2
2	3	P_3
1	1	P_4
3	6	P_5

26. بين كيف يُمكن للترقيم العددي للمصادر من أن يُساعد في عدم السماح بمبدأ الانتظار الدائري.

27. تُستخدم خوارزمية تأمين المرحلتين لاكتساب المصادر المشتركة من دون أن تكون هناك احتمالية لحدوث حالة الجمود، إلَّاإنَّها قد تُعاني من حالة مجاعة العملية، ناقش ذلك.

28. ما المقصود بجمود الاتصالات؟ وكيف يُمكن التعامل معه؟

29. وضِّح كيف تختلف حالة المجاعة عن حالة الجمود؟

30. بفرض أن هناك نظام له خمس عمليات P_1 إلى P_6 ، وثلاثة مصادر R_1 إلى R_3، وإجمالي عدد مثيلات المصدر الأول هو **2**، والثاني هو **5**، والثالث هو **4**. هل من الممكن تنفيذ العمليات التالية دون أن تحدث حالة الجمود؟

مصفوفة المخصص

R_3	R_2	R_1	
1	1	0	P_1
0	1	0	P_2
1	0	0	P_3
1	2	1	P_4
1	0	1	P_5

مصفوفة الحد الأقصى

R_3	R_2	R_1	
3	2	1	P_1
0	2	2	P_2
1	1	0	P_3
3	5	3	P_4
2	1	1	P_5

الفصل الخامس إدارة الذاكرة

1.5 مدخل إلى الذاكرة

تعرفنا خلال الفصل الأول على المكونات المادية لنظام الحاسوب والتي من أهمها مكون الذاكرة الرئيسية أو ما يُعرف باسم ذاكرة الوصول العشوائي، والتي هي عبارة عن مصفوفة كبيرة من الخانات الثمانية، لكل خانة منها عنوان خاص بها تُجلب الأوامر منها بطلب من وحدة المعالجة المركزية بناءً على العنوان المحمَّل في عدَّاد البرنامج. هذه الأوامر قد تتسبب في تحميل، أو حفظ بيانات، أو أوامر إضافية من وإلى الذاكرة، على التوالي تبعًا لمجريات حلقة تنفيذ الأوامر.

تُعتبر الذاكرة لذلك من إحدى أهم المصادر التي يجب أن تُدار بعناية في إطار نظم التشغيل، من أجل الحصول على أفضل أداء. حيث نجد اليوم أن متوسط حجم ذاكرة الحاسوب الشخصي قد تضاعف بأكثر من $10,000$ ضعفًا مقارنةً بحجم ذاكرة حواسيب أوائل الستينيات. بالمقابل ازداد حجم البرامج بشكل أكبر، وأسرع من أحجام الذواكر. في الواقع نجد أن كل مبرمج يتمنى أن يمتلك ذاكرة خاصة ذات حجم وسرعة لا نهائيتين، كما أنها لا تفقد محتوياتها عند فصل الطاقة الكهربائية عنها، فضلًا على أنها غير مكلفة. لسوء الحظ، التقنيات المتاحة حاليًا لا تُوفر مثل هذه الآماني في الوقت الحاضر، ربما يتحقق ذلك في المستقبل. بالتالي يتطلب هذا الأمر الوقوف عنده وتناوله بشكل تفصيلي، عليه سندرس في هذا الفصل الكيفية التي تُدير بها نظم التشغيل الذاكرة الرئيسية.

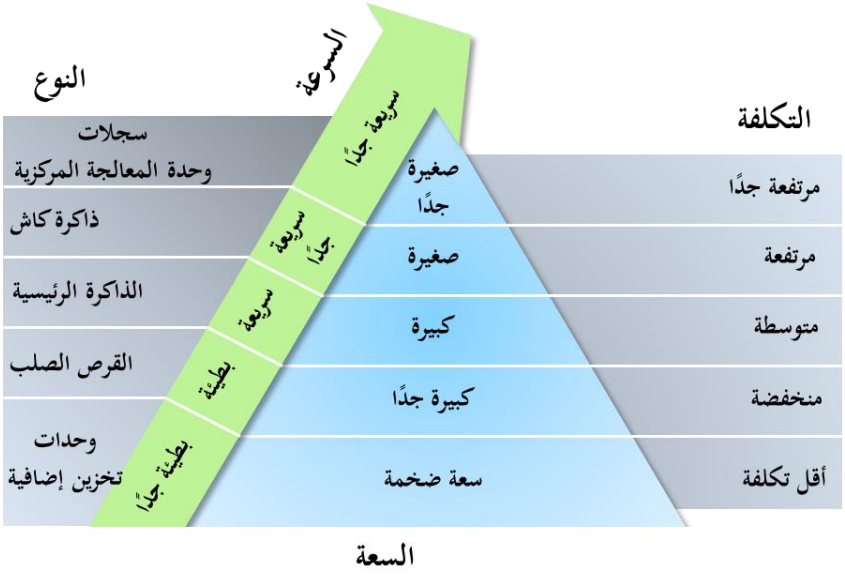
2.5 التسلسل الهرمي للذاكرة

على مر السنين من تاريخ اختراع الحاسوب تكَون مفهوم التسلسل الهرمي للذاكرة المبين في الشكل 5. 1. يمتلك الحاسوب في هذا التسلسل ذاكرة كاش أو الذاكرة المخبأ التي تتميز بصغر حجمها وسرعتها الفائقة وتكلفتها العالية. يلي ذلك الذاكرة الرئسية ذات السرعة، والتكلفة المتوسطتين، بينما يُمثل قرص التخزين الصلب الذاكرة الثانوية، الذي يتميز بسعته الكبيرة، وببطئه، ورخص ثمنه. ناهيك عن وسائط التخزين القابلة للإزالة مثل، أقراص الدي في دي، وشرائح الناقل التسللي العام.

باختصار، أسرع هذه الأنواع وأقلها حجمًا– كما نلاحظ من خلال هذا الشكل– هي السجلات تليها ذاكرة كاش ثم الذاكرة الرئيسية وأخيرًا في قاع الهرم توجد الذاكرة الثانوية والتي

تُعتبر أكبرهم حجمًا وأبطأهم سرعةً. تُدير نظم التشغيل كل هذه الأنواع، حيث يُسمى الجزء المسؤول عن إدارة الذاكرة "بمدير الذاكرة" الذي تتمثل مهمته في الآتي:

- تحديد وتتبع الأجزاء قيد الاستخدام، والأجزاء غير المستخدمة من الذاكرة لغرض تسكين العمليات الجديدة المراد تنفيذها.
- تخصيص الذاكرة للعمليات عندما تكون في حاجة إليها، واسترجاعها منها بعد الإنتهاء من مهمتها مع تحديد الأولويات في التسكين.
- إدارة عملية المبادلة بين الذاكرة والذاكرة الثانوية عندما يكون حجم الذاكرة لا يكفي لاستيعاب كافة العمليات.



الشكل 5. 1: الشكل الهرمي العام لهيكلية التخزين.

بالتالي سنحاول في هذا الفصل التعرض لعدة برامج مختلفة لإدارة الذاكرة تتراوح من بسيطة جدًا إلى متطورة للغاية. ولأن إدارة ذاكرة كاش عادة ما تتم من قبل الكيان المادي، سيكون تركيز هذا الفصل فقط على إدارة الذاكرة الرئيسية. في البداية، سننظر إلى أبسط الطرق الممكنة لإدارتها ومن ثم نتقدم تدريجيًا إلى الأكثر تطورًا.

3.5 إدارة الذاكرة المفردة

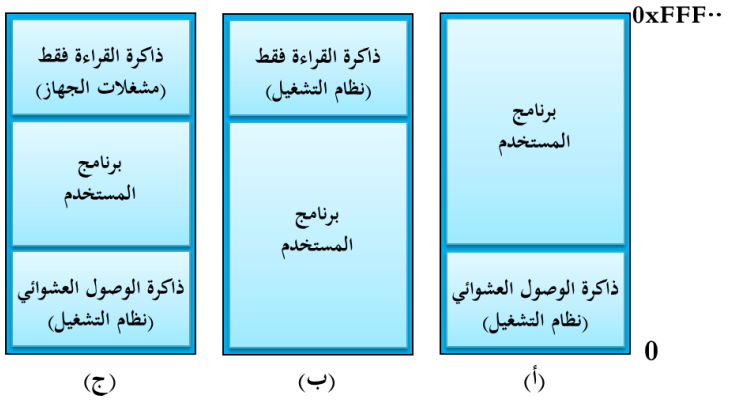
تُعتبر إدارة الذاكرة المفردة من التقنيات السهلة والبسيطة إلَّا إنَّها بدائية للغاية في إدارتها. فالحواسيب المركزية الأولية (جيل الخمسينيات)، ونظم الحواسيب المتوسطة الأولية (جيل الستينيات)، وأجهزة الحواسيب الشخصية الأولية (جيل السبعينيات) كلها تمتلك ذواكر من هذا النوع، فهي تستوعب برنامج واحد فقط مهما كان حجمه صغيرًا عليها، أو مهما كان حجمها كبيرًا على البرنامج. في ظل هذه الظروف، كان من المستحيل تحميل أكثر من برنامج في الذاكرة وتنفيذهم في نفس الوقت، أي أنَّ كل برنامج يُحمل، ويُنفذ إلى النهاية ومن ثم يَحل محله برنامج آخر، وهكذا. هذا يعني أن كل برنامج ببساطة يرى فقط الذاكرة الفعلية، أي عندما تُنفذ تعليمات مثل:

```
MOV REG, 2000
```

سينقل الحاسوب محتويات موقع الذاكرة الفعلي **2000** للسجل **REG**. بالتالي، فإن نموذج الذاكرة المقدم إلى المبرمج هو ببساطة الذاكرة الفعلية، أي مجموعة من العناوين تتراوح قيمها ما بين صفر والحد الأقصى لها، كل عنوان تُقابله خلية تحتوي على عدد من الخانات الثنائية، في العادة ثمان خانات.

يعرض الشكل 5. 2 ثلاثة خيارات ممكنة لنظام التشغيل داخل هذا النوع من الإدارة. الخيار الأول أُستخدِم بشكل رسمي في كل من الحواسيب المركزية والمتوسطة، وفيه يتواجد نظام التشغيل في الجزء السفلي من ذاكرة الوصول العشوائي، كما هو مبين في الشكل 5. 2–أ. الخيار الثاني أُستخدم من قبل بعض من أجهزة الحواسيب المحمولة والأنظمة المدمجة، وفيه يتواجد نظام التشغيل في ذاكرة القراءة فقط في الجزء العلوي من الذاكرة، كما هو مبين في الشكل 5. 2–ب. أما الخيار الثالث فيتمثل في تواجد مشغلات الجهاز في الجزء العلوي من الذاكرة داخل ذاكرة القراءة فقط، وبقية النظام في ذاكرة الوصول العشوائي في الجزء السفلي منها، كما هو مبين في الشكل 5. 2–ج. أُستخدم النموذج الأخير من قبل أجهزة الحواسيب الشخصية الأولية، ويُطلق على الجزء الموجود في ذاكرة القراءة فقط اسم نظام الإدخال، والإخراج الأساسي. من عيوب النموذجين الأول، والثالث هو أن أي خلل في برنامج المستخدم

يُمكن أن يقضي على نظام التشغيل، ويؤدي إلى نتائج كارثية.



الشكل 5. 2: خيارات تنظيم الذاكرة بوجود نظام تشغيل وبرنامج مستخدم واحد.

أخيرًا، عندما تُدار الذاكرة بهذه الطريقة، سيكون هناك برنامج واحد فقط يشتغل في أي لحظة زمنية، وبمجرد أن يُعطي المستخدم أمرًا للحاسوب، يُنفذ نظام التشغيل هذا الأمر بعد نسخ البرنامج المعني من القرص إلى الذاكرة. عندما ينتهي تنفيذ هذا البرنامج، يعرض نظام التشغيل محث النظام وينتظر إصدار أوامر جديدة له، عند تلقيه لإحداها، فإنه سيُحمل البرنامج المناظر له في الذاكرة، ليكتبه فوق البرنامج السابق.

بالإضافة إلى بساطة نموذج إدارة الذاكرة المفردة، تتميز هذه الإدارة بسهولة تحميل وتشغيل البرامج وسهولة نقل البرامج من وسائط التخزين إلى الذاكرة والعكس، كما أنها تُوفر حماية مطلقة للبيانات من التداخل والتضارب. مع هذا لا يخلو الأمر من العيوب، فهذا النوع من الإدارة لا يتوافق مع الحواسيب الكبيرة ويُسبب في إهدار الوقت والجهد كما أنه بطيء في تنفيذ البرامج، لأنه يعتمد على التنفيذ المتتالي وليس المتوازي.

هناك طريقة وحيدة تضمن الحصول على بعض من التوازي في تنفيذ العمليات في حالة الذاكرة المفردة تتمثل في البرمجة عن طريق الخيوط المتعددة، وذلك بفرضية أن جميع خيوط العملية ترى في نفس صورة الذاكرة. على الرغم من صحة هذه الفرضية، إلَّا إنَّ استخدامها محدودًا جدًا، بسبب رغبة الناس– في أغلب الأحيان– في تنفيذ برامج على التوازي في نفس الوقت تكون غير مرتبطة ببعضها البعض، الأمر الذي لا تُوفره الخيوط.

تشغيل عدة برامج دون تجزئة الذاكرة

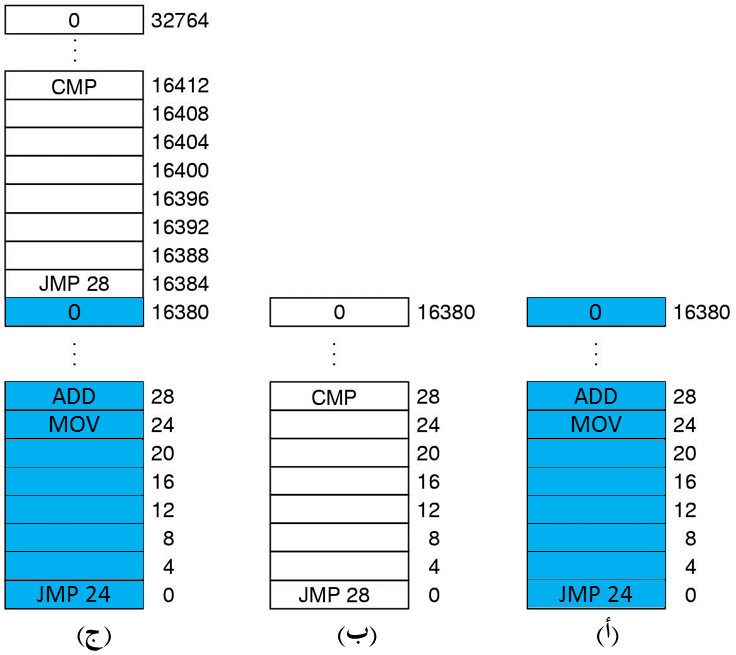
من المهم الإشارة هنا إلى إنه حتى مع استخدام الذاكرة المفردة، من الممكن تشغيل عدة برامج في نفس الوقت. هذا يتأتى من خلال استخدام مفهوم المبادلة وجعل نظام التشغيل يحتفظ بمحتويات الذاكرة في ملف داخل القرص ومن ثم جلب وتشغيل برنامج آخر. سوف يُناقش مفهوم المبادلة لاحقًا بنوع من التفصيل في القسم 5.5.

يُمكن كذلك الاستغناء عن تقنية المبادلة وتشغيل عدة برامج في وقت واحد باستخدام طرق خاصة بالحواسيب. مثلًا، قامت النماذج الأولية من جهاز **IBM 360** بتقسيم الذاكرة إلى مجموعة من المقاطع بحجم 2 كيلوبايت لكل مقطع وتخصيص مفتاح حماية لكل منها بطول أربع خانات ثنائية، يُحتفظ بهذه المفاتيح في سجلات خاصة داخل وحدة المعالجة المركزية، بحيث تحتاج آلة تملك ذاكرة ذات سعة واحد ميجابايت فقط إلى 512 سجل، أي بما مجموعه 256 بايت لحفظ المفاتيح. تُستخدم هذه المفاتيح لمحاصرة أي محاولة من قبل أي عملية قيد التنفيذ من الوصول إلى الذاكرة، وبمأن لنظام التشغيل فقط الحق في تغيير مفاتيح الحماية، ستُمنع عمليات المستخدم من التداخل مع بعضها البعض من جهة ومع نظام التشغيل نفسه من جهة أخرى.

ومع ذلك، فإن لهذا الحل عيب جوهري مبين في الشكل 5. 3. في هذا الشكل لدينا برنامجين، كل منهما حجمه 16 كيلوبايت موضحان في الشكل 5. 3–أ و ب. ضُلِّل الأول للإشارة إلى أن لديه مفتاح ذاكرة مختلف عن البرنامج الأخير. يبدأ البرنامج الأول من خلال القفز إلى العنوان 24، الذي يحتوي على التعليمة **MOV**، بينما يبدأ البرنامج الثاني من خلال القفز إلى العنوان 28، الذي يحتوي على التعليمة **CMP**، علمًا أن التعليمات التي لا صلة لها بهذه المناقشة لم يتم اظهارها. عندما يُحمَّل كلاهما على التوالي في الذاكرة بِدءًا من العنوان 0، ستظهر لدينا الحالة المعطية في الشكل 5. 3–ج. في هذا المثال، أُفتُرِض أن نظام التشغيل موجود في الجزء العلوي من الذاكرة، بالتالي لن يظهر كذلك.

بعد تحميل البرنامجين، يُمكنهما الآن الاشتغال. ولأن لدى كل واحد منهما مفتاح ذاكرة مختلف، لا يُمكن لأي منهما إتلاف الآخر، ولكن المشكلة هنا ذات طبيعة مختلفة. عندما يبدأ

البرنامج الأول، فإنه سيُنفذ التعليمة **JMP 24**، التي ستقوم بعملية القفز كما هو متوقع وبالتالي يعمل هذا البرنامج بشكل طبيعي.



الشكل 5. 3: توضيح مشكلة إعادة التخصيص [Tanenbaum & Bos, 2015].

ومع ذلك، بعد أن شُغِّل البرنامج الأول لفترة كافية، قد يُقرر نظام التشغيل تنفيذ البرنامج الثاني الذي حُمِّل أعلى البرنامج الأول عند العنوان **16384**. التعليمة الأولى التي ستُنفذ هي **JMP 28**، التي ستقفز إلى تعليمة **ADD** في البرنامج الأول، بدلًا من التعليمة **CMP** والتي من المفترض الانتقال إليها، هذا الأمر سيجعل البرنامج على الأرجح ينهار في أقل من ثانية.

المشكلة الأساسية هنا تتمثل في أن كلتا البرنامجين يستخدمان العنوان المطلق للذاكرة الفعلية، وهو ما لا نريده على الإطلاق. بالتالي يجب على كل برنامج أن يُشير إلى مجموعة من العناوين المحلية الخاصة به. لحل هذه المشكلة يُعدِّل جهاز **IBM 360** البرنامج الثاني في أثناء تحميله في الذاكرة باستخدام تقنية تُعرف باسم إعادة التخصيص الثابت، تُحمِّل هذه التقنية البرنامج في العنوان **16384** بعد أن يُضاف ثابت قدره **16384**إلى كافة عناوين هذا البرنامج في أثناء عملية التحميل. فالتعليمة **JMP 28** سيناضرها بعد الإضافة التعليمة **JMP 16412**.

على الرغم من أن هذه الآلية تعمل بشكل جيد، إلَّا إنَّها ليست بالحل العام، كما أنها تُبطئ عملية التحميل، نتيجةً لعمليات الإضافة. علاوة على ذلك، فإن الأمر يتطلب الحصول على معلومات إضافية خاصة بجميع البرامج القابلة للتنفيذ، لتوضيح ما إذا كانت تحتاج إلى عمليات الإضافة أم لا. مثال ذلك القيمة 28 في التعليمة **MOV REG, 28** التي تنقل العدد 28 إلى السجل **REG**، يجب ألَّا يُعاد تخصيصها، لأنها تُمثل تابث. هذا الأمر يجعل المحمَّل يحتاج إلى بعض من الطرق لمعرفة أيٍ من مثل هذه القيم عناوين، وأيٍ منها ثوابت.

أخيرًا، وكما أشرنا في الفصل الأول، التاريخ يُكرر نفسه كذلك في عالم الحواسيب. فبينما العنونة المباشرة للذاكرة الفعلية ليست سوى ذكرى بعيدة وُجدت على الحواسيب المركزية، والمتوسطة، وأجهزة الحواسيب المكتبية، والمحمولة، لا تزال الذاكرة المفردة شائعة في النظم المدمجة، وأنظمة البطاقات الذكية. فالأجهزة مثل الراديو، والغسالات، وأفران الميكروويف كلها تحتفظ– في هذه الأيام– ببرامجها في ذاكرة القراءة فقط، وفي معظم الحالات تُعنون الذاكرة من قبل البرمجيات بالعناوين المطلقة. هذا يجعل مثل هذه الأجهزة تعمل بشكل صحيح، لأن جميع البرامج معروفة مسبقًا، والمستخدم ليس حرًا في تشغيل البرمجيات الخاصة به على مثل هذا النوع من الأجهزة.

في حين أن النظم المدمجة الراقية (مثل الهواتف المحمولة) لديها أنظمة تشغيل متطورة ومنجزة بعناية، تجد النظم البسيطة منها لا تملك هذه الخاصية. في بعض الحالات، هناك نظام تشغيل، ولكن هو مجرد مكتبة ترتبط مع البرنامج التطبيقي وتُوفر استدعاءات النظام للقيام بعمليات الإدخال، والإخرج، وكذلك بعض من المهام الأخرى المشتركة. من الأنظمة المعروفة في هذا المجال نظام التشغيل **e–cos**.

4.5 إدارة الذاكرة المجزءة

من الشائع في الحواسيب الشخصية أن تكون هناك عدة برامج مفتوحة دفعة واحدة (معالج النصوص، برنامج البريد الإلكتروني، ومتصفح الويب، وغيرها) مع التركيز على واحد منها في أي لحظة زمنية، بينما تُنشط البقية عن طريق النقر بفأرة الحاسوب. هذا الأمر يصعب تحقيقه عند استخدام الذاكرة المفردة. لذلك من المهم جدًا البحث عن بدائل للذاكرة المفردة حتى يتسنى

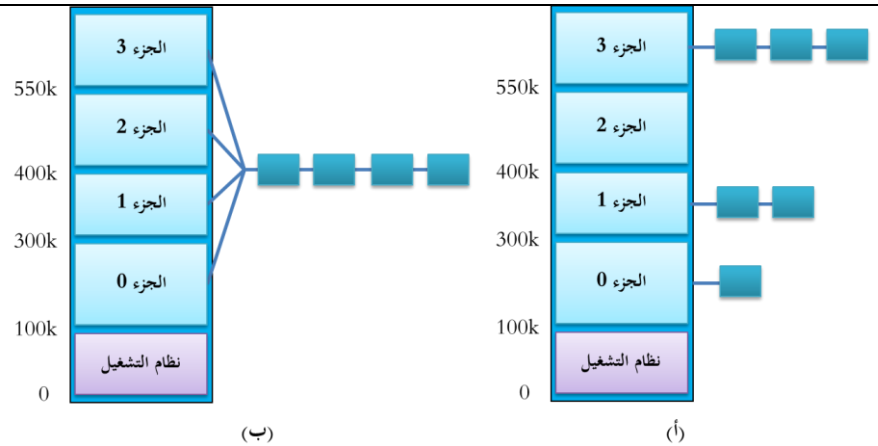
احتواء الذاكرة على أكثر من برنامج في آن واحد وتنفيذهم في نفس الوقت بكفاءة أعلى، مع التقليل من إهدار وقت وحدة المعالجة المركزية. السؤال المطروح الآن هو كيف يُمكن القيام بهذا لتحقيق هذا الهدف؟

تكمن أبسط هذه الطرق في تقسيم الذاكرة إلى مجموعة من الأجزاء ثابتة الحجم ولكن ليس من الضروري أن تكون متساوية، بحيث عندما يصل أي برنامج إلى الذاكرة، يَدخل مباشرة في الطابور الخاص بأصغر جزء يكفي لاستعابه، وذلك حتى لا يضيع جزء كبير من هذا الجزء. هذا سيؤدي إلى إضافة تعقيدات في عملية إدارة الذاكرة مما يُحتم على مدير الذاكرة التمتع بمرونة وديناميكية أعلى من ما عليه في الحالة السابقة، بالتالي ستُضاف إلى مدير الذاكرة الوظائف التالية:

1. تجهيز قوائم انتظار للبرامج المراد تحميلها في الذاكرة.
2. وضع طرائق للمفاضلة في ترتيب أولوية تحميل البرامج في الذاكرة.
3. تجهيز قائمة بأجزاء الذاكرة وأحجامها المتاحة للاستغلال.
4. تحديد آلية لتخصيص الجزء المناسب من الذاكرة لكل برنامج.
5. حماية البرامج من التداخل في أثناء التنفيذ.

يُوضح الشكل 5. 4–أ ذاكرة مجزءة إلى مجموعة من الأقسام بالإضافة إلى طوابير الإدخال الخاصة بكل قسم. ولكن استخدام الطوابير بهذا الشكل (أي لكل جزء طابور خاص به) سوف يكون غير مجدي في حالة ما إذا كان هناك ازدحام للطوابير على بعض من الأجزاء وفراغ على البعض الآخر، كالإزدحام على الجزء رقم 3 وخلو الجزء رقم 2 من أي طابور. وكبديل لهذا التنظيم يُمكن استخدام طابور واحد لكل الأجزاء كما في الشكل 5. 4–ب، بحيث يُخصص أي جزء ينتهي استخدامه إلى البرنامج الأقرب إلى مقدمة الطابور ويكون حجمه قريب جدًا من حجم الجزء لكي يتسنى تنفيذه.

بالرغم من سهولة إدارة الذاكرة المجزءة ذات الأجزاء ثابتة الحجم وتحميلها لنفقات إضافية بسيطة لنظام التشغيل، إلَّا إنَّ استخدام الذاكرة الرئيسية قد يكون غير فعَّال بسبب التفتت الداخلي والذي يحدث عندما يُخصص جزء أكبر من حجم البرنامج الفعلي ولا تستخدم منه إلا جزء صغير، فالباقي يُعد مساحة ضائعة، الأمر الذي أفضى إلى مفهوم المبادلة.



الشكل 5. 4: أ) الذاكرة المجزأة وطوابير الإدخال المستقلة– ب) الذاكرة المجزأة وطابور الإدخال الموحد.

5.5 المبادلة

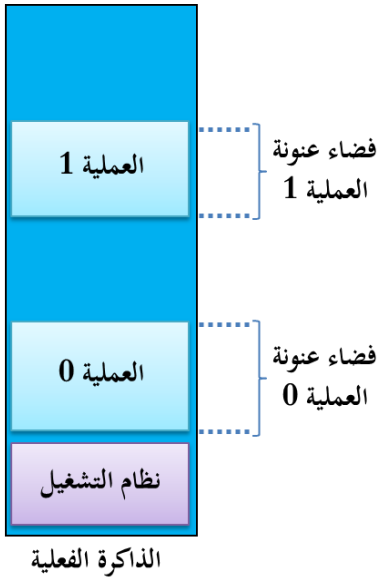
إن تقسيم الذاكرة إلى مجموعة من الأجزاء ثابتة الحجم مناسب جدًا لأنظمة الدفعة، ولكن في بعض الأحيان يكون حجم العملية غير معروف، بالتالي لا يُمكننا تخصيص أي جزء ثابت الحجم لها. أحيانًا يكون أيضًا (كما في أنظمة المشاركة الزمنية) عدد المستخدمين أكبر من أن يُستوعب عملياته داخل الذاكرة مما يُحتم علينا تخزين العمليات المتواجدة في حالة جاهزة أو متوقفة مؤقتًا في القرص ومن ثم استرجاعها منه إلى الذاكرة في حالة دخول العملية إلى حالة تشتغل. إن نقل العمليات من الذاكرة إلى القرص ثم من القرص إلى الذاكرة يُسمى بالمبادلة والتي سنناقشها في هذا القسم. في البداية سنتطرق إلى فكرة فضاء العنونة، وسجلي الأساس والحد، ومن ثم سنتناول مفهوم المبادلة بنوع من التفصيل.

1.5.5 فكرة فضاء العنونة

للسماح لعدة تطبيقات بالتواجد في الذاكرة في نفس الوقت دون أن يحدث أي تداخل بينها، لابد من حل مشكلتين رئيسيتين هما: الحماية، وإعادة التخصيص. لقد تعرضنا في السابق إلى حل بدائي أُستخدم على جهاز IBM 360 يتمثل في تعليم أجزاء من الذاكرة بمفاتيح حماية، ومقارنة مفتاح العملية المنفذة مع مفتاح الجزء المراد الوصول إليه في الذاكرة. ومع ذلك،

فإن هذا النهج في حد ذاته لا يحل مشكلة إعادة التخصيص، على الرغم من أنه يُمكن حلها عن طريق إعادة تخصيص البرامج في أثناء عملية التحميل، وهو ما يُعتبر حلًا بطيئًا ومعقدًا.

يتمثل الحل البديل والأفضل في تجزئة الذاكرة واستخدام فضاء العنونة، الذي يُمثل مجموعة من العناوين يُمكن لأي عملية استخدامها لعنونة الذاكرة، بحيث يكون لكل عملية فضاء عنونة خاص بها، ومستقل عن تلك الفضاءات التي تنتمي إلى العمليات الأخرى (ما عدا في بعض من الظروف الخاصة التي تتشارك فيها العمليات فضاءات العنونة الخاصة بها)، كما هو موضح في الشكل 5. 5.



الشكل 5. 5: فضاء عنونة العمليات.

إن مفهوم فضاء العنونة هو مفهوم عام جدًا ويحدث في العديد من التطبيقات. مثلًا، بالنظر إلى أرقام الهواتف، نجد أن رقم الهاتف المحلي عادة ما يتكون من عدد محدد من الخانات، مثلًا سبعة. بالتالي يمتد فضاء العنونة لأرقام الهواتف من 0000000 إلى 9999999، على الرغم من أن بعض منها غير مستخدم، كتلك التي تبدأ مع 000. مع نمو الهواتف المحمولة، وأجهزة الاتصالات الأخرى، أصبحت هذه الفضاءات صغيرة جدًا، الأمر الذي يتطلب استخدام خانات أكثر.

فضاءات العنونة لا يجب أن تكون رقمية فقط، فمجموعة مجالات الإنترنت 'com.' يُمثل كذلك فضاء عنونة يتكون من كافة السلاسل الحرفية بطول يتراوح من حرفين إلى 63 حرفًا، التي يُمكن أن تحتوي على حروف، وأرقام، وواصلات يليها اسم المجال. إلى حد ما تُعتبر هذه الفكرة بسيطة، ولكن الأصعب هنا هو كيف يُمكن إعطاء كل برنامج فضاء عنونة خاص به، لأنَّ أي عنوان في فضاء عنونة العملية 0 في الشكل 5. 5– مثلًا– يعني موقع فعلي يختلف عن نفس العنوان في فضاء عنونة العملية 1. أدناه سوف نناقش طريقة بسيطة وعامة، لكنها لم ترتقي لأن تكون موجودة داخل رقائق وحدات المعالجة المركزية الحديثة بسبب عدم الرغبة في استعمال مخططات معقدة، والرغبة الحقيقية في إيجاد بدائل أفضل.

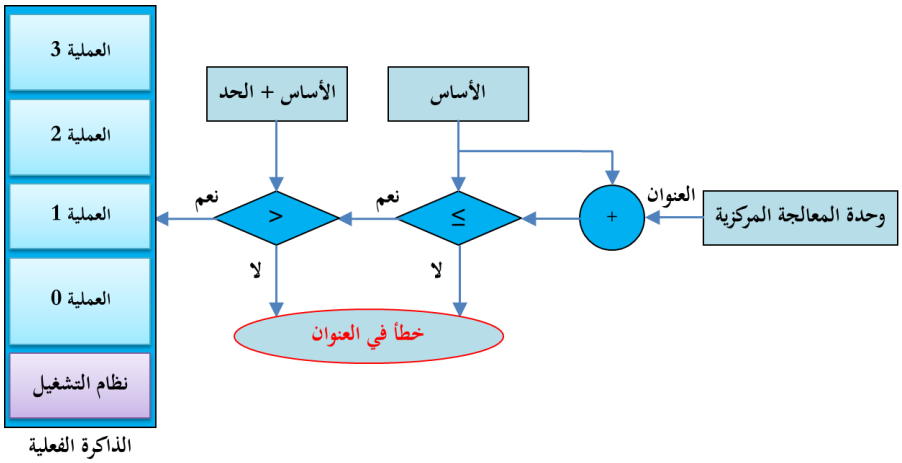
2.5.5 سجلا الأساس والحد

بالرغم من وجود حل بسيط– يستخدم صيغة مبسطة من إعادة التخصيص المتغير– يُعيَّن فضاء عنونة لكل عملية في جزء مختلف من الذاكرة الفعلية بطريقة بسيطة، إلَّا إنَّ الحل الكلاسيكي– الذي أُستخدم في الأجهزة الممتدة من جيل CDC 6600 (أول حاسوب عملاق في العالم) إلى Intel 8088 (قلب حاسوب IBM الشخصي الأصلي)– يتمثل في تزويد كل وحدة معالجة مركزية بسجلين ماديين خاصين هما الأساس والحد لاستخدامهما في حماية حقوق الوصول إلى الذاكرة الرئيسية، أي منع العملية من الوصول إلى الذاكرة التي لم تُخصص لها، بالتالي المساهمة في منع حدوث خلل داخل العملية يُؤثر على العمليات الأخرى، أو على نظام التشغيل نفسه.

عند تشغيل برنامج– ما– يُحمَّل سجل الأساس بالموقع الفعلي لبداية هذا البرنامج في الذاكرة، بينما يُحمَّل سجل الحد بطول البرنامج. بمعنى آخر، هاذان السجلان يُعرِّفا فضاء عنونة البرنامج، أي العملية. كمثال لهذه القيم يُمكن الرجوع إلى الشكل 5. 3–ج، والذي نجد فيه أن قيم الأساس والحد التي ستُحمل في هذه السجلات المادية عند تشغيل البرنامج الأول هي 0 و 16384، على التوالي، بينما القيم المستخدمة عند تشغيل البرنامج الثاني هي 16384 و 32764، على التوالي.

يُوضح الشكل 5. 6 فكرة استخدام سجلي الأساس والحد في حل مشكلة الحماية. في كل

مرة يتم فيها الرجوع إلى الذاكرة من قبل أي عملية– إمَّا لجلب تعليمات أو قراءة أو كتابة كلمة بيانات– تُضيف وحدة المعالجة المركزية تلقائيًا قيمة الأساس إلى العنوان الناتج عن العملية قبل إرسال العنوان على متن ناقل العنونة الخاص بالذاكرة، في نفس الوقت، تتحقق من أن العنوان الناتج أكبر من أو يساوي قيمة الأساس وأقل من مجموع كل من الأساس وقيمة الحد، وإلَّا فستُولد رسالة خطأ وستُحبط عملية الوصول إلى الذاكرة.



الشكل 5. 6: الفكرة الأساسية في استخدام سجلي الأساس والحد.

باستخدام سجلي الأساس والحد أصبح من السهل تخصيص فضاء عنونة خاص بكل عملية، لأنَّ أي عنوان ذاكرة مُوَلد تلقائيًا يُضاف إليه محتوى سجل الأساس قبل إرساله إلى الذاكرة. في العديد من التنفيذات، تتم حماية سجلي الأساس، والحد بحيث لا يُعدَّلا إلَّا عن طريق نظام التشغيل فقط.

من عيوب استخدام هاذين السجلين هو الحاجة إلى إجراء عمليات الجمع والمقارنة عند كل عملية رجوع إلى الذاكرة. قد تتم عمليات المقارنات بسرعة، لكن عمليات الإضافة ستكون بطيئة، بسبب زمن حساب المحمول، الذي يُمكن تقليله في حالة ما إذا أُستخدمت دوائر جمع مميزة.

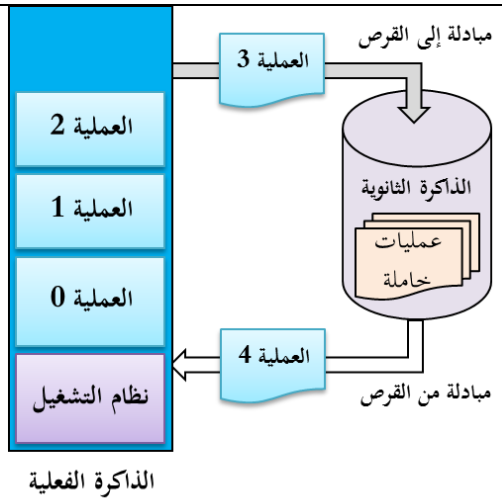
3.5.5 مفهوم المبادلة

إذا كانت الذاكرة الفعلية للحاسوب كبيرة بما يكفي لاحتواء جميع العمليات، فإن

المخططات التي وُصفت إلى حد الآن تفي– نوع ما– بالغرض، ولكن من الناحية العملية، فإن إجمالي ما تحتاجه العمليات من ذاكرة الوصول العشوائي– في كثير من الأحيان– أكثر بكثير مما يُمكن استيعابه في الذاكرة. في نظام 'ويندوز' أو 'لينكس' النموذجي على سبيل المثال، يُمكن لقرابة 40–60 عملية أو أكثر أن تبدأ في الاشتغال في بداية تشغيل الحاسوب. مثلاً، عند تثبيت برنامج تطبيقي تحت بيئة 'ويندوز'، غالبًا ما تصدر الأوامر بحيث عند تشغيل الحاسوب من جديد تُنفذ عملية مهمتها الأساسية البحث عن التحديثات الأخيرة لهذا التطبيق. هذه العملية يُمكن أن تشغل ببساطة مساحة قدرها من 5 إلى 10 ميجابايت من الذاكرة، ناهيك عن العمليات الخلفية الأخرى كاختبار البريد الوارد، واتصالات الشبكة الواردة، وأشياء أخرى كثيرة. كل هذا يحدث قبل بدء أول برنامج للمستخدم. كذلك بعض من البرامج التطبيقية بإمكانها أن تَشغل– في وقتنا الحاضر– من 50 إلى 200 ميجابايت أو أكثر من ذلك من مساحة الذاكرة. وكنتيجة لذلك، يتطلب حفظ كافة العمليات في الذاكرة في جميع الأوقات ذاكرة ذات سعة كبيرة، الأمر الذي لا يُمكن أن يتم، إلَّا إذا كانت هناك ذاكرة كبيرة وكافية لاستعاب هذه الأحجام.

لمعالجة الاستغلال المفرط للذاكرة لمثل هذه الحالات يُستخدم مفهوم المبادلة وهي آلية يُمكن فيها تبديل العملية مؤقتًا خارج الذاكرة الرئيسية (أو النقل) إلى الذاكرة الثانوية (القرص) وإتاحة تلك الذاكرة للعمليات الأخرى. في وقت لاحق، يُبدل النظام العملية من الثانوية إلى الرئيسية، كما هو موضح في الشكل 5. 7. على الرغم من أن أداء النظام يتأثر عادة بعملية المبادلة، إلَّا إنَّها تُساعد في تشغيل عمليات متعددة وكبيرة بشكل متواز.

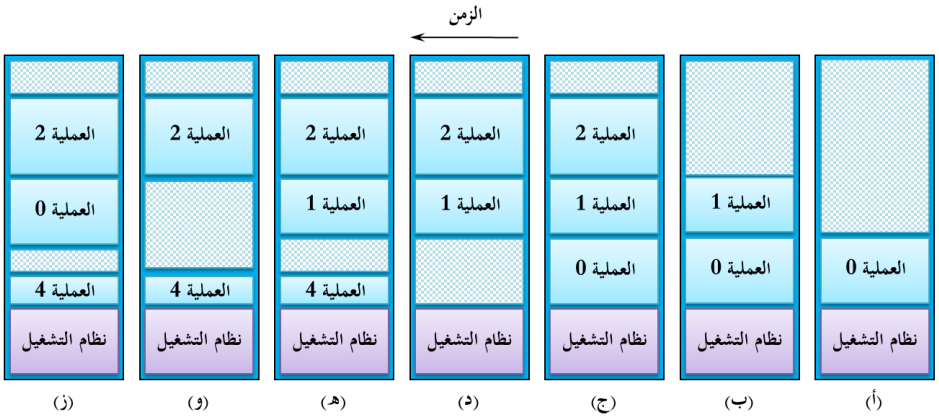
تُجلب كل عملية في الشكل 5. 7 في مجملها من القرص (مثل العملية 4)، وتُنَفذ لفترة من الزمن، ثم تُعاد إلى القرص (مثل العملية 3). تُخزن العمليات الخاملة في الغالب في القرص، حتى لا تستغل أي جزء من الذاكرة عندما لا تشتغل (على الرغم، من أن البعض منها يستيقظ بشكل دوري للقيام بعمله، ثم يذهب إلى النوم مرة أخرى).



الشكل 5. 7: مبادلة العمليات ما بين الذاكرة الرئيسية والذاكرة الثانوية.

يُمكن تتبع المثال الموضح في الشكل 5. 8 لغرض توضيح تشغيل نظام المبادلة. في البداية، العملية 0 موجودة فقط في الذاكرة، بعد ذلك حُمِّلت أو بُدِّلت كل من العملية 1 و 2 من القرص. في الشكل 5. 8–د أُستُرجعت المساحة المخصصة للعملية 0، لأنها قد إنتهت أو تمت مبادلتها من جديد إلى القرص ثم حُمِّلت العملية 4 ورُجِّعت العملية 1. أخيرًا، حُمِّلت العملية 0 مرة أخرى ولأنها الآن في موقع مختلف، فالعناوين الحالية لها لابد من إعادة تخصيصها، إمَّا عن طريق الكيان المعنوي عند حدوث المبادلة أو (على الأرجح) من قبل الكيان المادي في أثناء تنفيذ البرنامج، مثلًا باستخدام طريقة سجلي الأساس والحد التي تعمل بشكل جيد هنا.

نتيجةً لعمليات المبادلة السابقة الذكر تنشأ عدة فراغات في الذاكرة قد تتسبب في عدم استغلالها بشكل جيد. هذه الظاهرة تُعرف باسم التفتت الخارجي للذاكرة التي من الممكن القضاء عليه بتجميع الفراغات جميعها في جزء واحد كبير عن طريق نقلها إلى أبعد حد ممكن أسفل الذاكرة. تُعرف هذه التقنية باسم ضغط الذاكرة التي– عادة ما– تتطلب الكثير من وقت وحدة المعالجة المركزية، الأمر الذي يؤدي إلى عدم الرغبة في استخدامها. مثلًا، جهاز يملك ذاكرة بحجم واحد قيقابايت ويُمكنه نسخ أربعة بايت في 20 نانو ثانية سوف يستغرق حوالي خمس ثواني لضغط الذاكرة بأكملها.



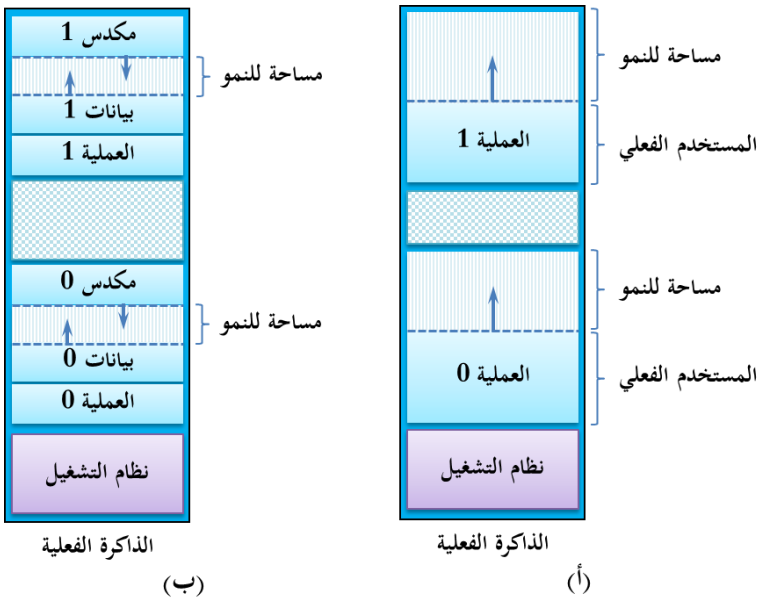
الشكل 5. 8: تغيرات تخصيص الذاكرة تبعاً لعمليات المبادلة (المساحات المضللة هي ذاكرة غير مستخدمة).

بعدما تعرفنا على مفهوم المبادلة يظل هناك سؤال جوهري يستحق أن يُؤخذ في عين الاعتبار: وهو ما حجم المكان الذي سيُخصص للعملية عندما تحدث المبادلة أو عندما تُنشأ عملية جديدة؟ في حالة إنشاء عمليات ذات حجم ثابت لا يتغير أبدًا، ستكون عملية التخصيص بسيطة، حيث يُخصص نظام التشغيل ما هو مطلوب بالضبط، ولكن ماذا لو كانت العمليات متغيرة الحجم؟

يقول واقع الحال أنّ العمليات عادة ما تكون غير ثابتة الحجم وهي عرضة للنمو في أثناء عمليات التنفيذ وذلك حسب متطلبات العملية، كما هو الحال في العديد من لغات البرمجة. لذلك فإنه من الصعوبة بمكان تقدير الحجم المناسب لهذه العملية وتخصيص حجم ثابت لها. ولحل هذه الإشكالية يُخصص فراغ إضافي يكون متاخم لمكان العملية حتى تستطيع النمو فيه، أما في حالة ما إذا كانت العملية محاذية لعملية أخرى بحيث تحد من عملية النمو، فستُنقل إلى مكان آخر يستوعبها، أو تتم مبادلة عمليات أخرى إلى القرص، حتى يتوفر فراغ كاف لهذه العملية. إذا لم تستطع العملية النمو في الذاكرة بسبب عدم وجود فراغ وكانت مساحة المبادلة على القرص محجوزة بالكامل، فستُجهض العملية أو تُعلق حتى تتحرر مساحة كافية لها.

أمَّا إذا كان من المتوقع أن معظم العمليات ستنمو في أثناء عملية التنفيذ، فسيكون من الأفضل تخصيص حجم أكبر من حجم العملية بقليل لكي تنمو فيه، وذلك للحد من الجهد

المرتبط بنقل أو مبادلة العمليات. ومع ذلك، عند مبادلة العمليات إلى القرص، يتم فقط مبادلة الجزء المستخدم فعليًا من الذاكرة، لأن مبادلة الجزء الإضافي سيكون إهدارًا للوقت. في الشكل 5. 9–أ نرى هيئة الذاكرة التي خُصصت فيها مساحة للنمو للعمليتين 0 و 1.



الشكل 5. 9: أ) تخصيص مساحة إضافية إلى جزء البيانات المتنامي– ب) تخصيص مساحة إضافية إلى كل من جزئي البيانات والمكدس المتناميين.

أحيانًا لا ينمو من البرنامج مقطع البيانات فقط، بل هناك مقاطع أخرى تنمو في أثناء عملية تنفيذ البرنامج، مثل مقطع المكدس. في هذه الحالة، سيُخصص فراغ إضافي لكليهما، وذلك كما موضح في الشكل 5. 9–ب. نلاحظ من خلال هذا الشكل أن هناك عمليتان، كل منهما لها مساحة نمو مقسمة بين مقطعي المكدس والبيانات، الجزء الأول موجود في قمة الذاكرة المخصصة لهذه العملية وسينمو إلى أسفل، أما مقطع البيانات فهو متاخم لنص البرنامج وسينمو إلى أعلى، الأمر الذي يؤدي إلى إمكانية استغلال مساحة النمو من قبل أحدهما أو كلاهما. إذا لم تكن هذه المساحة كافية فسيتم–كما أشرنا سابقًا– تخصيص مكان آخر، أو مبادلة عمليات أخرى أو إجهاض هذه العملية. وسنتطرق في الأقسام التالية إلى الطرق المستخدمة في إدارة، وتتبع استخدام الذاكرة.

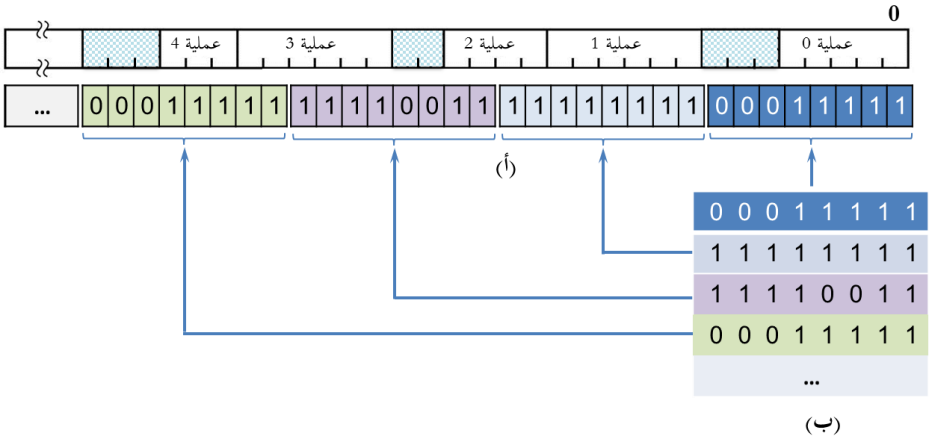
4.5.5 إدارة فراغ الذاكرة

عندما تُستخدم المبادلة ويكون تخصيص الذاكرة بأحجام متغيرة، يجب على نظام التشغيل إدارة عمليات التخصيص بعناية. بشكل عام، هناك أكثر من طريقة لتتبع استخدام الذاكرة نعرض منها خرائط الخانات الثنائية، والقوائم المرتبطة، فيما يلي سنتطرق إلى هاتين الطريقتين بتفصيل أكثر.

1.4.5.5 إدارة الذاكرة عن طريق خرائط الخانات الثنائية

عند استخدام خرائط الخانات الثنائية في إدارة الذاكرة وتتبع استغلالها، تُقسم الذاكرة إلى مجموعة من الوحدات، كما هو موضح في الشكل 5. 10-أ، والتي قد تكون ذات أحجام صغيرة بمعدل مجموعة كلمات، أو أحجام كبيرة بمعدل عدة كيلوبايت. يُناظر كل وحدة من هذه الوحدات خانة ثنائية داخل خرائط الخانات الثنائية المفصلة في الشكل 5. 10-ب، بحيث تكون قيمة هذه الخانة صفرًا عندما تكون الوحدة المناظرة لها غير مستخدمة، وتساوي واحدًا إذا كانت الوحدة المناظرة لها مستخدمة، (أو العكس بالعكس).

من العوامل المهمة في استخدام هذه الطريقة هو حجم الوحدة، بحيث كلما صغر حجمها كلما أُستغلت الذاكرة بشكل أفضل، وكلما زاد حجم خرائط الخانات الثنائية، والذي سيؤدي بدوره إلى بطء سرعة تنفيذ العملية، عند الاحتياج إلى البحث عن n من الأماكن الخالية والمتتالية داخل خرائط الخانات الثنائية لغرض استغلالها داخل هذه الذاكرة. هذا الأمر يجعل استخدام خرائط الخانات الثنائية في إدارة الذاكرة نادرًا. مثلًا، لو كان حجم الوحدة صغير بمقدار أربع خانات ثمانية، فإن 32 خانة ثنائية من الذاكرة تتطلب خانة واحدة فقط داخل الخريطة الثنائية، بالتالي ذاكرة بسعة n*32 سوف تحتاج إلى خريطة ثنائية بحجم n خانة وهو ما يجعل الخريطة الثنائية تأخذ 33/1 من حجم الذاكرة الفعلية والذي لا يُمكن استغلاله في خدمة العمليات. بالمقابل، كبر حجم الوحدة سيؤدي إلى صغر حجم خرائط الخانات الثنائية، ولكن سيؤدي ذلك إلى عدم استغلال الذاكرة بشكل أمثل وخصوصًا إذا كان حجم العملية ليس من مضاعفات حجم وحدة التخصيص.



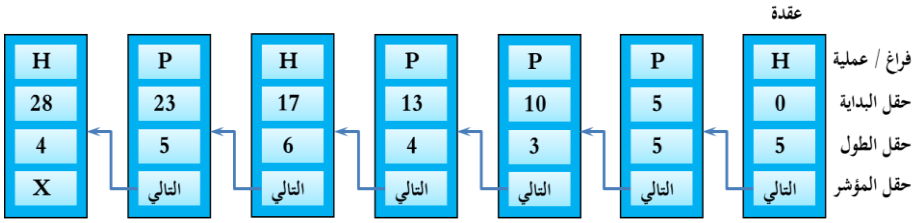
الشكل 5. 10: أ) جزء من الذاكرة به خمس عمليات، وثلاثة أماكن غير مستخدمة (المناطق المضللة)– ب) شكل خرائط الخانات الثنائية المناظرة لهذه الذاكرة.

2.4.5.5 إدارة الذاكرة عن طريق القوائم المرتبطة

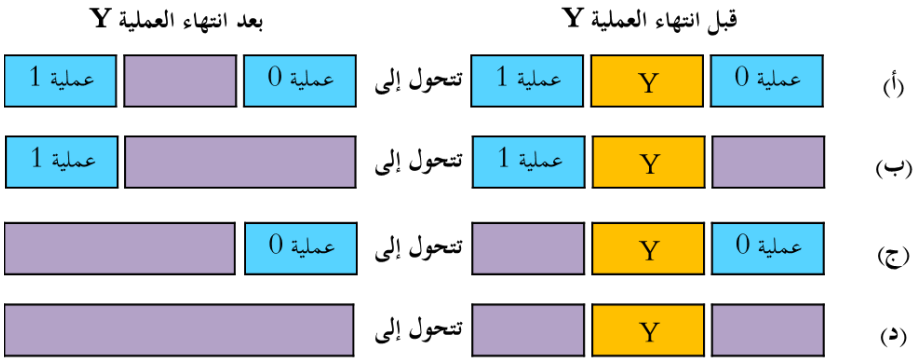
تتمثل الطريقة الثانية المستخدمة في إدارة وتتبع استغلال الذاكرة في استخدام القوائم المرتبطة وهي أحد هياكل البيانات التي تُوفر إمكانية استغلال وسائط التخزين بشكل أمثل، تتكون هذه القوائم من مجموعة من العقد، كما تتكون كل عقدة من عدد محدد من الحقول أهمها حقل المعلومات، وحقل المؤشر الذي يُشير إلى العقدة التالية في القائمة. تُستخدم هذه القوائم لحفظ أجزاء الذاكرة المخصصة والحرة على النحو المبين في الشكل 5. 11. يُسمى كل جزء من هذه القائمة عقدة، وهي إمَّا أن تُمثل عملية أو تُكون عقدة فارغة بين عمليتين. كل عقدة في القائمة تحتوي على حقل الفراغ (H) أو العملية (P)، حقل عنوان البداية، وحقل طول الجزء وحقل مؤشر الإدخال التالي.

يوضح الشكل 5. 11 مثال لجزء من الذاكرة يحوي أربع عمليات وثلاثة فراغات غير مستخدمة. أول عقدة من هذه القائمة تُمثل فراغ يبدأ عند العنوان 0 وبطول خمس وحدات، بينما تُمثل العقدة التالية عملية تبدأ عند العنوان 5 وبطول خمس وحدات، وهكذا مع بقية العقد، علمًا أن حقل مؤشر آخر عقدة سوف يحوي العلامة X للدلالة على نهاية القائمة. الاحتفاظ بالقائمة المرتبطة مرتبة بهذا الشكل أي حسب العنوان يجعل تحديث القائمة سهلًا ومباشرًا. فعندما تنتهي

عملية أو تتم مبادلتها عادة ما تجاورها عقدتين أُخرتين (باستثناء عندما تكون العملية في بداية، أو نهاية القائمة) وهي إمَّا أن تكون فراغات، أو عمليات، وذلك كما هو موضح في الشكل 5. 12. بالتالي عند تحديث القائمة فستُحدَّث حسب الاحتمالات الأربعة الموضحة بنفس الشكل. في الشكل 5. 12–أ، تحديث القائمة يتطلب استبدال العملية بواسطة فراغ. في الشكل 5. 12–ب والشكل 5. 12–ج، دُمج جزئين ليكونا فراغ واحد، بينما في الشكل 5. 12–د دُمجت ثلاثة أجزاء في فراغ واحد، الأمر الذي يجعل القائمة تتقلص بمقدار عقدتين.



الشكل 5. 11: قائمة مرتبطة تُمثل جزء من الذاكرة بها ثلاث عمليات وثلاثة فراغات غير مستخدمة.



الشكل 5. 12: الاحتمالات الأربعة لجيران العملية المنتهية.

عندما يُحتفظ بالعمليات والفراغات في قائمة مرتبطة مرتبة على النحو المشار إليه آنفًا، فستكون هناك عدة خوارزميات تُستخدم لتخصيص الذاكرة لعملية أُنشئت حديثًا (أو عملية موجودة تمت مبادلتها مؤخرًا من القرص). تاليًا سنتناول هذه الخوارزميات بنوع من التفصيل، مع ملاحظة أننا نفترض هنا أن مدير الذاكرة على دراية بحجم الذاكرة المراد تخصيصها للعملية.

3.4.5.5 خوارزميات تخصيص أجزاء الذاكرة

عند إنشاء عملية جديدة أو مبادلة عملية– ما– يجب تخصيص مكان أو جزء لها في الذاكرة، ولكن قد يكون هناك أكثر من جزء متاح لاستيعاب طلب هذه العملية. السؤال المطروح حاليًا ما هي الآلية التي سيُختار بها أجزاء الذاكرة لتخصيصها لمثل هذه العمليات؟ لاختيار جزء معين، هناك حاجة إلى خوارزميات تخصيص أجزاء الذاكرة، علمًا أن أفضل هذه الخوارزميات هي الخوارزمية التي تُقلل من التفتت الداخلي للذاكرة. فيما يلي طرح لبعض من هذه الخوارزميات.

خوارزمية البحث عن أول فراغ مناسب

تُعتبر خوارزمية البحث عن أول فراغ مناسب أبسط وأسرع هذه الخوارزميات على الإطلاق، وذلك لقصر عملية البحث فيها. حيث يمسح مدير الذاكرة جميع عقد القائمة المرتبطة إلى أن يجد فراغًا كبيرًا يكفي لاستعاب العملية، عندها تقف عملية البحث. بعد ذلك يُقسم هذا الفراغ إلى جزئيين. يُخصص جزء للعملية والجزء الآخر يُعلَّم على أساس أنه غير مستخدم، إلَّا في حالة ما إذا كان من المرجح إحصائيًا أن حجم الفراغ يتناسب بالضبط مع حجم العملية.

مثال توضيحي: في أحد نظم التشغيل التي تستخدم المبادلة في إدارة الذاكرة، كانت هذه الذاكرة بها خمسة أجزاء فارغة بأحجام مرتبة على النحو 100، 500، 200، 300، 600. ما هو الجزء التي سيُختار في حالة البحث عن مكان للعمليات ذات الأحجام 212، 417، 112، و 426 داخل هذه الذاكرة، وذلك عند استخدام هذه الخوارزمية؟

الحل: استنادًا على مفهوم هذه الخوارزمية سيكون تخصيص العمليات عى النحو الموضح في الجدول 5. 1.

الجدول 5. 1: مثال توضيحي لخوارزمية البحث عن أول فراغ مناسب.

رقم العملية	حجم العملية	رقم الجزء المخصص
1	212	2
2	417	5
3	112	3
4	426	لم يُخصص لها أي جزء

من خلال هذا المثال يُمكن ملاحظة أنه قد يحدث أن هناك عملية لم يُسمح لها بالتخصيص، بالرغم من أنه كانت هناك إمكانية لعملية التخصيص. فالعملية 4 ذات الحجم 426 لم يُخصص لها أي جزء، بالرغم من تواجد فراغات ذات أحجام كافية لاستعاب هذه العملية، إلَّا إنَّها أُستغلت من قبل عمليات أخرى، هذه المعضلة من الممكن حلها بواسطة الخوارزمية التالية.

خوارزمية البحث عن أفضل فراغ مناسب

خوارزمية البحث عن أفضل فراغ مناسب من الخوارزميات الأخرى المعروفة والمستخدمة على نطاق واسع في إدارة تخصيص فراغات الذاكرة. تمسح هذه الخوارزمية القائمة بالكامل من البداية إلى النهاية للبحث عن أصغر فراغ يتلائم بشكل أفضل مع حجم العملية المبدَّلة، بدلًا من تقسيم فراغ كبير قد تكون هناك حاجة إليه في وقت لاحق.

بالعودة إلى المثال التوضيحي السابق فسيكون تخصيص العمليات على النحو الموضح في الجدول 5. 2.

الجدول 5. 2: مثال توضيحي لخوارزمية البحث عن أفضل فراغ مناسب.

رقم العملية	حجم العملية	رقم الجزء المخصص
1	212	4
2	417	2
3	112	3
4	426	5

خوارزمية البحث عن أفضل فراغ مناسب أبطأ من خوارزمية البحث عن أول فراغ مناسب، لأنه يجب في الأولى بحث القائمة بأكملها في كل مرة تُستدعى فيه. إلى حد ما من المستغرب كذلك، أن تؤدي هذه الخوارزمية إلى ضياع في الذاكرة بشكل أكبر مما عليه الحال في خوارزمية البحث عن أول فراغ مناسب لأنها تميل إلى ملء الذاكرة بفراغات صغيرة، وعديمة الفائدة، على العكس من الخوارزمية السابقة التي تُولد في المتوسط فراغات كبيرة.

خوارزمية البحث التالي عن الفراغ المناسب

من جهة أخرى تعمل خوارزمية البحث التالي عن الفراغ المناسب بنفس الكيفية التي تعمل

بها خوارزمية البحث عن أول فراغ مناسب، إلَّا إنَّها كلما وجدت فراغ مناسب لحجم العملية فإنها تحتفظ بعنوان هذا الفراغ، لتبدأ منه الخوارزمية عندما تُستدعى مرة أخرى، بدلًا من إعادة المسح من بداية القائمة. تُعتبر هذه الخوارزمية من الخوارزميات سريعة البحث حيث أنها أسرع نسبيًا من الخوارزميتين السابقتين لإدارة تخصيص فراغ الذاكرة. إلَّا إنَّه في الواقع هناك عدة أبحاث وتجارب عملية بَيَّنت أن هذه الخوارزمية أسوء بقليل من خوارزمية البحث عن أول فراغ مناسب.

مثال توضيحي: في أحد نظم التشغيل التي تستخدم المبادلة في إدارة الذاكرة، كانت بهذه الذاكرة ثلاثة أجزاء فارغة بأحجام مرتبة على النحو 5، 10، 20. ما هو الجزء التي سيُختار في حالة البحث عن مكان للعمليات ذات الأحجام 10، 20، و 30 داخل هذه الذاكرة، وذلك عند استخدام هذه الخوارزمية؟

الحل: استنادًا على مفهوم هذه الخوارزمية سيكون تخصيص العمليات عى النحو الموضح في الجدول 5. 3.

الجدول 5. 3: مثال توضيحي لخوارزمية البحث التالي عن الفراغ المناسب.

رقم العملية	حجم العملية	رقم الجزء المخصص
1	10	2
2	20	3
3	30	لم يُخصص لها أي جزء

خوارزمية البحث عن أكبر فراغ مناسب

تلتف خوارزمية البحث عن أكبر فراغ مناسب على مشكلة تقسيم الفراغ إلى جزء يتلائم إلى حد كبير مع حجم العملية وجزء آخر صغير يُمثل فراغ. فهي تقوم بهذا عن طريق البحث عن أكبر فراغ مناسب متاح داخل القائمة ومن ثم تقسيمه لينتج عن ذلك فراغ يكون كبير بحيث يُمكن الاستفادة منه لاحقًا. ومع هذا فإن نتائج المحاكاة أظهرت كذلك أن هذه الخوارزمية ليست بالفكرة الجيدة.

مثال توضيحي: بالرجوع إلى المثال التوضيحي الأول فسيكون تخصيص العمليات على النحو الموضح في الجدول 5. 4.

الجدول 5. 4: مثال توضيحي لخوارزمية البحث عن أكبر فراغ مناسب.

رقم العملية	حجم العملية	رقم الجزء المخصص
1	212	5
2	417	2
3	112	4
4	426	لم يُخصص لها أي جزء

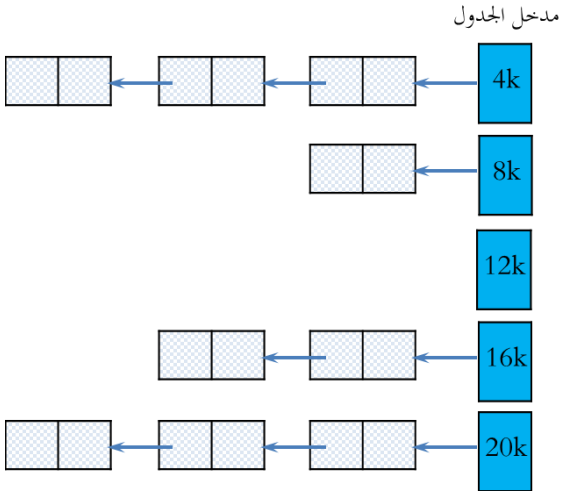
الخوارزميات الأربع السالفة الذكر يُمكن الإسراع في تنفيذها عن طريق الاحتفاظ بقوائم منفصلة لكل من العمليات والفراغات. بهذه الطريقة، ستُكرس جميع الخوارزميات طاقاتها الكاملة لفحص قائمة الفراغات فقط دون العمليات. الثمن الحتمي الذي سيُدفع على حساب تسريع التخصيص هو تعقيد إضافي وتباطؤ في عملية تحرير أجزاء الذاكرة، لأن الجزء المحرر منها لابد من إزالته من قائمة العمليات وإدراجه في قائمة الفراغات.

إذا أُستخدمت قوائم منفصلة للعمليات والفراغات، فإنه بالإمكان فرز قائمة الفراغات على أساس الحجم، الأمر الذي يجعل خوارزمية البحث عن أفضل فراغ مناسب أسرع. عندما تبحث هذه الخوارزمية في قائمة فراغات مرتبة تصاعديًا (أي من الأصغر إلى الأكبر)، فإنها بمجرد أن تجد فراغ مناسب، فستعرف أن هذا الفراغ هو أصغر فراغ من شأنه أن يقوم بهذه المهمة، بالتالي ليس هناك حاجة إلى مزيد من البحث، كما هو الحال مع نظام القائمة الواحدة. كذلك، عندما تكون قائمة الفراغات مرتبة حسب الحجم، فستتساوى كلتا الخوارزمتين البحث عن أول فراغ مناسب، والبحث عن أفضل فراغ مناسب من حيث السرعة، بينما ستكون خوارزمية البحث عن الفراغ المناسب التالي غير ذات جدوى.

من الممكن كذلك إضافة بعض التحسينات البسيطة عندما يُحتفظ بقوائم منفصلة للفراغات عن العمليات. فبدلًا من وجود مجموعة منفصلة من هياكل البيانات للحفاظ على قائمة الفراغات، كما هو الحال في الشكل 5. 11، يُمكن تخزين المعلومات في الفراغات نفسها مثلًا، الكلمة الأولى من كل فراغ يُمكن أن تكون حجم الفراغ، والكلمة الثانية مؤشر الإدخال التالي، نتيجة لذلك فإن عقد قائمة الشكل 5. 11، التي تتطلب ثلاث كلمات وخانة واحدة (P أو H)، لم تعد هناك حاجة لها.

خوارزمية البحث السريع عن الفراغ المناسب

من خوارزميات التخصيص الأخرى خوارزمية البحث السريع عن الفراغ المناسب، التي تحتفظ بقوائم منفصلة لبعض من الفراغات المطلوبة ذات أحجام شائعة الاستخدام بشكل كبير. مثلًا، قد يكون لديك جدول به عدد n من المداخل: المدخل الأول يُشير إلى رأس قائمة تتكون من فراغات ذات حجم أربعة كيلوبايت، المدخل الثاني هو مؤشر إلى قائمة لفراغات بحجم ثمانية كيلوبايت، المدخل الثالث هو مؤشر إلى قائمة لفراغات بحجم 12 كيلوبايت، كما هو موضح بالشكل 5. 13. الفراغات بحجم 21 كيلوبايت مثلًا، يُمكن وضعها إمَّا في قائمة 20 كيلوبايت أو على قائمة خاصة من الفراغات ذات الأحجام الفردية.



الشكل 5. 13: توضيح فكرة خوارزمية البحث السريع عن الفراغ المناسب.

باستخدام خوارزمية البحث السريع عن الفراغ المناسب سيكون العثور على فراغ من الحجم المطلوب سريع للغاية، لكنها تُعاني من نفس العيب الذي تُعاني منه جميع الطرق التي تعتمد الترتيب حسب حجم الفراغ، أي عندما تنتهي العملية أو تتم مبادلتها، فسيكون العثور على جيرانها لمعرفة ما إذا كان الدمج ممكنًا أم لا مكلفًا للغاية. إذا لم يتم الدمج، فستتجزأ الذاكرة بسرعة إلى عدد كبير من الفراغات الصغيرة التي لا تتناسب مع أي عملية.

6.5 الذاكرة الظاهرية

بتطور تقنيات الحاسوب أصبح بإمكانه معالجة ذاكرة أكبر من الذاكرة المثبتة فعليًا على النظام وذلك عن طريق استخدام ما يُعرف باسم الذاكرة الظاهرية (الافتراضية) وهي جزء من القرص الصلب الذي أُعدَّ لمحاكاة ذاكرة الوصول العشوائي للحاسوب. الميزة الرئيسية للذاكرة الظاهرية هي السماح للبرامج بأن يكون حجمه أكبر من الذاكرة الفعلية، بالتالي فهي تخدم غرضين أساسيين.

أولًا: السماح بتوسيع استخدام الذاكرة الفعلية عن طريق القرص.

ثانيًا: تُساعد في حماية الذاكرة، لأن كل عنوان ظاهري يُترجم إلى عنوان فعلي.

يتناول هذا القسم موضوع الذاكرة الظاهرية بنوع من التفصيل.

1.6.5 تمهيد

بالرغم من أن سجلي الأساس، والحد مكنا من تجزئة فضاء العنونة، إلَّا إنَّ تزايد أحجام البرمجيات أكثر بكثير من تزايد أحجام الذاكرة وهو ما يُعتبر عقبة رئيسية أمام الاستفادة من سرعة الحاسوب والبرمجة المتعددة. في الثمانينيات، كانت هناك عدة جامعات تُوفر نظام المشاركة الزمنية للتعامل مع عشرات المستخدمين، الذين يشتغلون في نفس الوقت. في وقتنا الحاضر تُوصي شركة ميكروسوفت بضرورة وجود ما لا يقل عن 512 ميجابايت لمستخدم واحد لنظام 'ويندوز فيستا' لتشغيل تطبيقات بسيطة، وواحد قيقابايت إذا كان التعامل مع تطبيقات معقدة، أما الاتجاه نحو الوسائط المتعددة فيتطلب المزيد والمزيد من الذاكرة.

نتيجة لهذا التزايد، ظهرت الحاجة الماسة لتشغيل برامج كبيرة جدًا أكبر من ما تستوعبه الذاكرة، وهناك بالتأكيد كذلك حاجة لأنظمة يُمكنها أن تدعم عدة برامج تعمل في نفس الوقت، كل منها قد يتناسب وحجم الذاكرة، ولكنها فعليًا قد تتجاوز مجتمعة حجم الذاكرة الفعلي. هنا يُمكن استخدام مبدأ المبادلة إلَّا إنَّها ليست بالخيار الجذاب، لأنه لأي قرص لديه معدل نقل 100 ميجابايت لكل ثانية، قد تُكلف مبادلة برنامج بحجم واحد قيقابايب على الأقل عشر ثواني لمبادلته من الذاكرة إلى القرص، وعشر أخرى لمبادلته من القرص إلى الذاكرة.

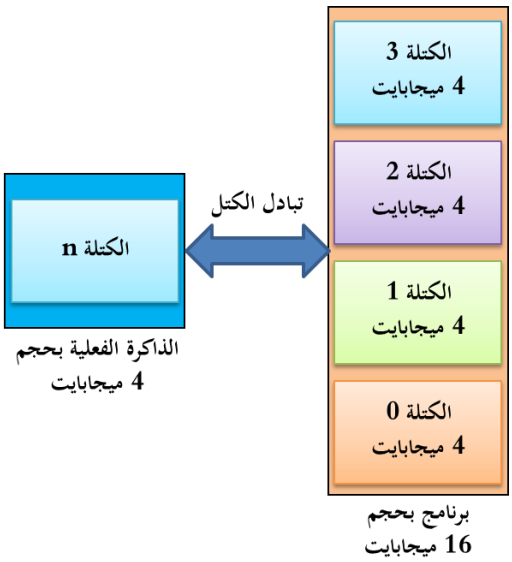
معضلة البرامج الكبيرة مع الذاكرة بدأت مع بداية الحوسبة وإن كان ذلك في تخصصات محدودة، مثل العلوم والهندسة. في الستينيات، كان الحل المعتمد لهذه المشكلة يتمثل في تقسيم البرامج إلى قطع صغيرة، تُدعى بالشرائح. عندما يبدأ البرنامج في التنفيذ، تُحمَّل الشرائح في الذاكرة من قبل مديرها، والذي بدوره سيُحمِّل ويُنفِّذ على الفور الشريحة 0، بعد الإنتهاء من ذلك يُحمِّل الشريحة 1، إما فوق الشريحة 0 (إذا لم تتوفر هناك أي مساحة) أو أعلاها في الذاكرة (إذا كانت هناك مساحة لذلك). جميع هذه الشرائح تكون محفوظة في القرص وتُبادل مع الذاكرة من قبل مدير الشرائح.

على الرغم من إنجاز العمل الفعلي لمبادلة الشرائح ما بين الذاكرة والقرص يتم من قبل نظام التشغيل تحت إشراف مدير الشرائح، إلاَّ إنَّ المبرمج هو من يُقسم البرنامج يدويًا إلى شرائح. عملية التقسيم هذه هي في الحقيقة عمل مضني وخصوصًا للمبرمج، لأنها عرضةً للخطأ، وكذلك تستغرق الكثير من الوقت والجهد. بالتالي لم يمضي وقت طويل حتى جاء من أحال هذه العملية بالكامل إلى جهاز الحاسوب.

في 1961 قدَّم فوترنقهام (Fotheringham, 1961) الذاكرة الظاهرية، وهي نوع من تقسيم البرنامج– الذي لا يُمكن استعابه في الذاكرة لكبر حجمه– إلى أجزاء قابلة إلى أن تُحمل في الذاكرة كل حسب الحاجة إليه. الفكرة الأساسية وراء ذلك تتمثل في أن كل برنامج له فضاء عنونة خاص به مُقسم إلى أجزاء تُسمى صفحات أو مقاطع وأن السلوك النموذجي للبرنامج لا يتطلب وجود جميع كتل البرنامج في وقت واحد في أثناء تنفيذه داخل الذاكرة، كما هو موضح في الشكل 5. 14، والذي يُبين كيف يمكن لبرنامج حجمه 16 ميجابايت من أن يُنفذ على آلة ذات ذاكرة فعلية بحجم 4 ميجابايت عن طريق اختيار الكتلة المناسبة بعناية لكي تُحمَّل في الذاكرة ومبادلتها مع الكتل الأخرى في القرص كل حسب الحاجة، أي أن هذه الكتل لا تحتاج إلى تخصيص منفصل للذاكرة الرئيسية وهو ما سينتج عنه أن مساحة العنوان الفعلي للعملية أصغر من مساحة العنوان المنطقي الخاص به.

خلاصة القول أنه بإمكان تمثيل كتل البرنامج بالصفحات بحيث تُمثل كل صفحة مجموعة متجاورة من العناوين تُحمَّل في الذاكرة الفعلية، علمًا أنه ليس من الضروري أن تُحمَّل كل الصفحات لكي يشتغل البرنامج. كذلك عندما يتم الرجوع من قبل البرنامج إلى جزء من فضاء

العنونة الخاص به داخل الذاكرة الفعلية، يُربط الكيان المادي اللازم على الفور. أما في حالة الرجوع إلى جزء من فضاء العنونة الخاص بهذا البرنامج وكان هذا الفضاء غير موجود في الذاكرة الفعلية، فسيُنبه نظام التشغيل لاستحضار الفضاء المفقود، ومن ثم إعادة تنفيذ التعليمات التي فشلت.



الشكل 5. 14: تبادل كتل البرنامج من القرص مع الذاكرة.

أخيرًا، تُعد إدارة الذاكرة في نظام التشغيل وظيفة أساسية تسمح بتخصيص الذاكرة للعمليات في أثناء التنفيذ وتُلغي تخصيصها عند عدم الحاجة إلى العملية سواءً بإنتهائها أو مبادلتها. تاليًا، سنناقش مفهوم كل من التصفح والتقطيع لإدارة الذاكرة وتوضيح الفروقات الأساسية بينهما.

2.6.5 التصفح

تُعتبر تقنية التصفح إحدى مخططات إدارة الذاكرة وهي تلعب دورًا مهمًا في تنفيذ الذاكرة الظاهرية، تسمح هذه التقنية بأن يُحتفظ بالعملية في أماكن غير متجاورة مما يعني تخزين أجزاء مختلفة من نفس العملية في أماكن مختلفة داخل الذاكرة الرئيسية، الأمر الذي يُساهم في حل مشكلة التفتت الخارجي للذاكرة. ولتنفيذ التصفح يُقسم فضاء العنونة الظاهري (المنطقي) إلى كتل من نفس الحجم تُسمى صفحات وهي أصغر جزء يُقسم الرنامج إليه، بينما يُقسم فضاء

العنونة الفعلي إلى كتل مناظرة للصفحات في الحجم داخل الذاكرة الفعلية تُعرف بإطارات الصفحة، والتي تحمل جميعها نفس الحجم (النظم الحقيقية تستخدم في أحجام صفحات من 512 بايت إلى 64 كيلوبايت).

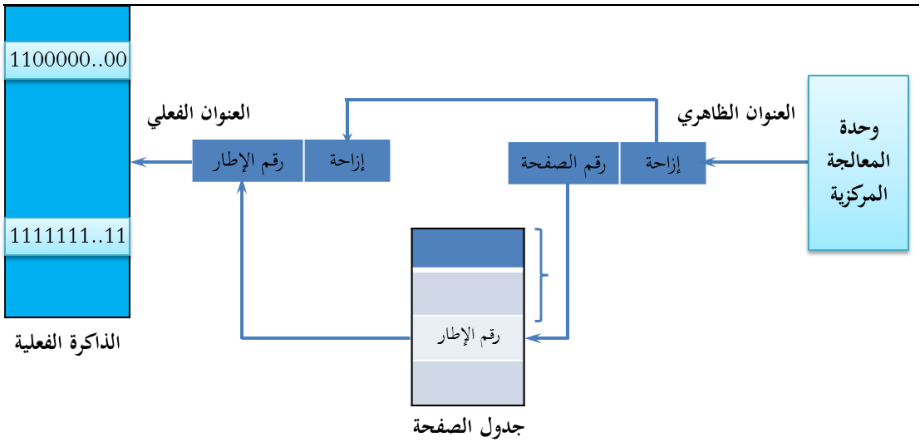
عندما يحتاج البرنامج إلى التنفيذ، تُحمَّل صفحاته من فضاء الذاكرة المنطقية في إطارات فضاء عنوان الذاكرة الفعلية. للوصول إلى إطار الصفحة يُقسم العنوان المولَد بواسطة وحدة المعالجة المركزية إلى جزأين هما: رقم الصفحة، والإزاحة داخل الصفحة، كما هو موضح في الشكل 5. 15. من خلال رقم الصفحة سيُحدد رقم إطار الصفحة ومن ثم ربطه مع هذه الإزاحة لتكوين العنوان الفعلي، حسب المعادلتين التاليتين:

عنوان الصفحة (العنوان المنطقي) = رقم الصفحة + الإزاحة

عنوان الإطار (العنوان الفعلي) = رقم الإطار + الإزاحة

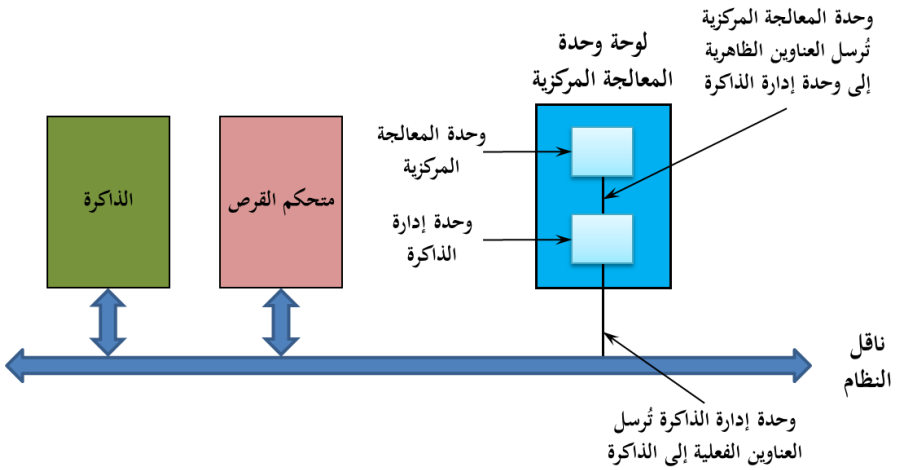
يستخدم الشكل 5. 15 جدول الصفحة في تتبع حساب العنوان الفعلي من خلال احتفاظه بالعنوان الأساسي للصفحة المخزنة في إطار مساحة الذاكرة الفعلية، والذي يُستخدم في تتبع العلاقة ما بين صفحة العملية وإطارها في الذاكرة عن طريق دمج العنوان الأساسي المحدد بجدول الصفحة مع الإزاحة داخل الصفحة لتعريف رقم الإطار في الذاكرة المخزنة فيها الصفحة. لاحقًا ستُوضح هذه العملية بأمثلة تفصيلية، بعدها سيُقدم القسم التالي مفهوم جدول الصفحة بنوع من التفصيل.

للتعرف أكثر على آلية تنفيذ تقنية التصفح على جهاز الحاسوب، سنتعرف أولًا على آلية توليد فضاء العنونة الظاهري، بالإضافة إلى مكون وحدة إدارة الذاكرة التي تُحوِّل العناوين الظاهرية إلى عناوين فعلية، كما هو موضح في الشكل 5. 16. تُفهرس البرامج على أي جهاز حاسوب مجموعة من عناوين الذاكرة الخاصة بها، بالتالي عند تنفيذ أي برنامج لتعليمة مثل: **MOV REG, 500** ستنسخ هذه التعليمة محتويات عنوان الذاكرة 500 إلى السجل **REG**، عناوين الذاكرة هذه يُمكن أن تتولد باستخدام الفهرسة، وسجلات الأساس، أو بواسطة عدة طرق أخرى.



الشكل 5. 15: مفهوم التصفح.

تُسمى العناوين المولَدة هنا بالعناوين الظاهرية وهي تُشكل فضاء العنونة الظاهري. في الحواسيب التي لا تدعم مبدأ الذاكرة الظاهرية، يُوضع العنوان الظاهري مباشرةً على ناقل الذاكرة ليتسبب في قراءة أو كتابة كلمة من أو إلى الذاكرة الفعلية بنفس العنوان، على التوالي، أمَّا في حالة استخدام الذاكرة الظاهرية، فستُرسل هذه العناوين مباشرةً إلى وحدة إدارة الذاكرة بدلًا من ناقل الذاكرة لتُترجمها بدورها إلى عناوين فعلية.

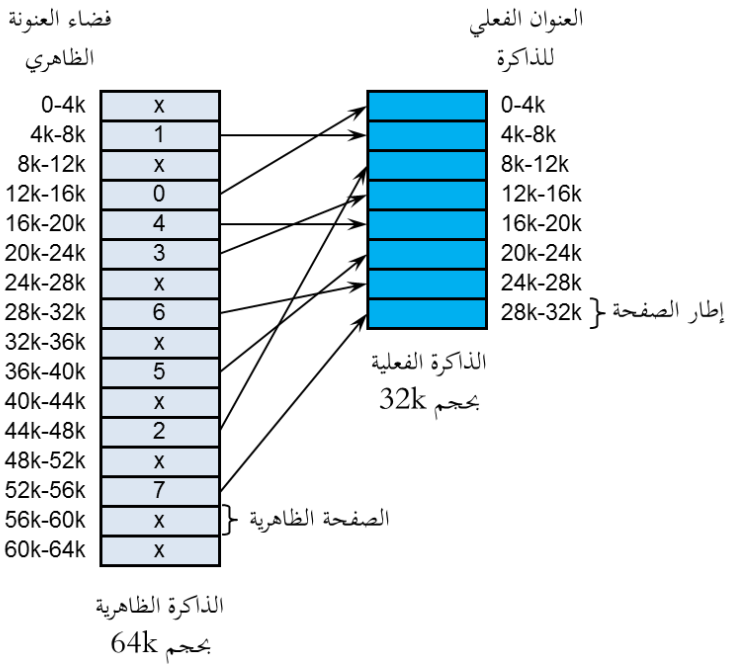


الشكل 5. 16: موقع، ووظيفة وحدة إدارة الذاكرة بالنسبة للوحة وحدة المعالجة المركزية.

كمثال توضيحي بسيط لكيفية عمل الربط ما بين كل من العنوان الفعلي والظاهري، يُمكن

النظر إلى الشكل 5. 17. في هذا المثال، لدينا جهاز حاسوب يُولِّد عناوين ظاهرية بطول خانتين ثمانيتين، أي من 0 إلى 64 كيلوبايت، إلَّا إنَّ حجم ذاكرته الفعلية هو 32 كيلوبايت. بالتالي وعلى الرغم من أنه يُمكن كتابة برنامج بحجم 64 كيلوبايت، نجد أنه من المستحيل تحميله في المجمل في هذه الذاكرة الفعلية ومن ثم تشغيله. لذلك يجب أن تكون هناك نسخة كاملة من الصورة الأساسية للبرنامج– ما يصل حجمه إلى 64 كيلوبايت– موجودة على القرص لكي تُجلب كتل البرنامج كلًا حسب الحاجة.

في المثال الموضح في الشكل 5. 17، أُستخدمت صفحات وإطارات صفحة بسعة 4 كيلوبايت، بالتالي مع وجود فضاء عنونة ظاهري بحجم 64 كيلوبايت، وذاكرة فعلية بسعة 32 كيلوبايت، سيكون لدينا 16 صفحة ظاهرية و8 إطارات صفحة وستتم دائما المبادلة ما بين الذاكرة والقرص على هيئة صفحات كاملة.



الشكل 5. 17: العلاقة ما بين العناوين الظاهرية والفعلية للذاكرة موضحة بجدول الصفحة، كل صفحة تبدأ بمضاعفات 4096 وتنتهي بزيادة 4095 عنوان.

التدوين المستخدم في الشكل 5. 17 يُمكن توضيحه بالآتي: المدى 0–4k يعني أن العناوين الظاهرية أو الفعلية في تلك الصفحة هي 0 إلى 4095، أما المدى 4k–8k فهو يُشير إلى مجموعة العناوين 4096–8191، وهلم جرا. كل صفحة تحتوي بالضبط على 4096 عنوان، بدءًا من مضاعفات 4096، ونهايةً بزيادة 4095 عنوان. ولنرى الآن الكيفية التي يُحسب بها العنوان الفعلي من العنوان الظاهري عن طريق وحدة إدارة الذاكرة.

حساب العنوان الفعلي من العنوان الظاهري

عندما يُحاول أي برنامج الوصول مثلًا إلى العنوان 5000، باستخدام التعليمة: MOV REG, 5000 يُرسل هذا العنوان الظاهري إلى وحدة إدارة الذاكرة، التي عندها ستجد أنه يقع في الصفحة رقم 1 أي في مدى العناوين (4096–8191)، وذلك حسب المخطط الموضح في الشكل 5. 17. هذا الرقم سيُناظر إطار الصفحة رقم 1 ذات مدى العناوين (4096–8191)، بالتالي ستُحول الوحدة العنوان الظاهري 5000 إلى عنوان فعلي محصور بين هذا المدى، وتضعه على ناقل العنونة. هذا يعني أن وحدة إدارة الذاكرة ستُحول كافة العناوين الظاهرية المحصورة بين 4096 و 8191 إلى العناوين الفعلية المحصورة ما بين 4096 و 8191.

بالمثل، فإن التعليمة: MOV REG, 12288 تتحول بالفعل إلى: MOV REG, 0، لأن العنوان الظاهري 12288 (في الصفحة الظاهرية 3) سيُناظر العنوان 0 (في إطار الصفحة الفعلية 0). كمثال ثالث، العنوان الظاهري 20500 يزيد بمقدار 20 خانة ثمانية عن بداية الصفحة الظاهرية 5 (العناوين الظاهرية 20480–24575) الذي سيُناظر العنوان الفعلي 12288 + 20 = 12308.

تجدر الإشارة هنا إلى أن القدرة على مواءمة الست عشرة صفحة ظاهرية على أي من إطارات الصفحة الثمانية من خلال وضع خريطة مناسبة لوحدة إدارة الذاكرة لا يحل المشكلة المتمثلة في أن فضاء العنونة الظاهرية أكبر من الذاكرة الفعلية. ولأن هناك فقط ثمانية إطارات صفحة فعلية، ستُربط في الشكل 5. 17 فقط ثمان صفحات من الصفحات الظاهرية مع الذاكرة الفعلية، بقية الصفحات معلمة بعلامة X للدلالة على أنها غير مرتبطة، كما هو موضح في هذا الشكل. بالمقابل تُستخدم خانة الحضور، والغياب في الكيان المادي الفعلي لتتبع الصفحات

الموجودة فعليًا في الذاكرة. السؤال الذي يطرح نفسه الآن هو: ماذا سيحدث إذا رجع البرنامج إلى عناوين غير مرتبطة مع الذاكرة الفعلية؟

عند استخدام التعليمة: **MOV REG, 1000** نجد أن العنوان الظاهري **1000** يقع في مدى الصفحة الظاهرية **0** التي تبدأ عند **0**. في هذه الحالة ستُلاحظ وحدة إدارة الذاكرة أن هذه الصفحة غير مرتبطة مع الذاكرة الفعلية، وذلك من خلال خانة الحضور، والغياب الخاصة بكل صفحة. تتسبب هذه الحالة في إشعار نظام التشغيل من قبل وحدة المعالجة المركزية بهذا الغياب، يُسمى هذا الإشعار بخطأ الصفحة[18] ويتسبب في جعل نظام التشغيل يختار إطار صفحة نادر الاستخدام ويكتب محتوياته في القرص (إذا لم تكن فعلاً هناك)، بعد ذلك يجلب النظام فقط الصفحة التي أُشير إليها مؤخرًا، ويضعها في إطار الصفحة الذي حُرِّر، ومن ثم تُحدَّث خارطة وحدة إدارة الذاكرة، ويُعاد تشغيل التعليمات المسببة لخطأ الصفحة هذا.

مثلًا، إذا قرر نظام التشغيل إعادة تخصيص إطار الصفحة رقم **1**، فعليه تحميل الصفحة الظاهرية رقم **6** في العنوان الفعلي **8192** وإحداث تغييرين على خارطة وحدة إدارة الذاكرة.

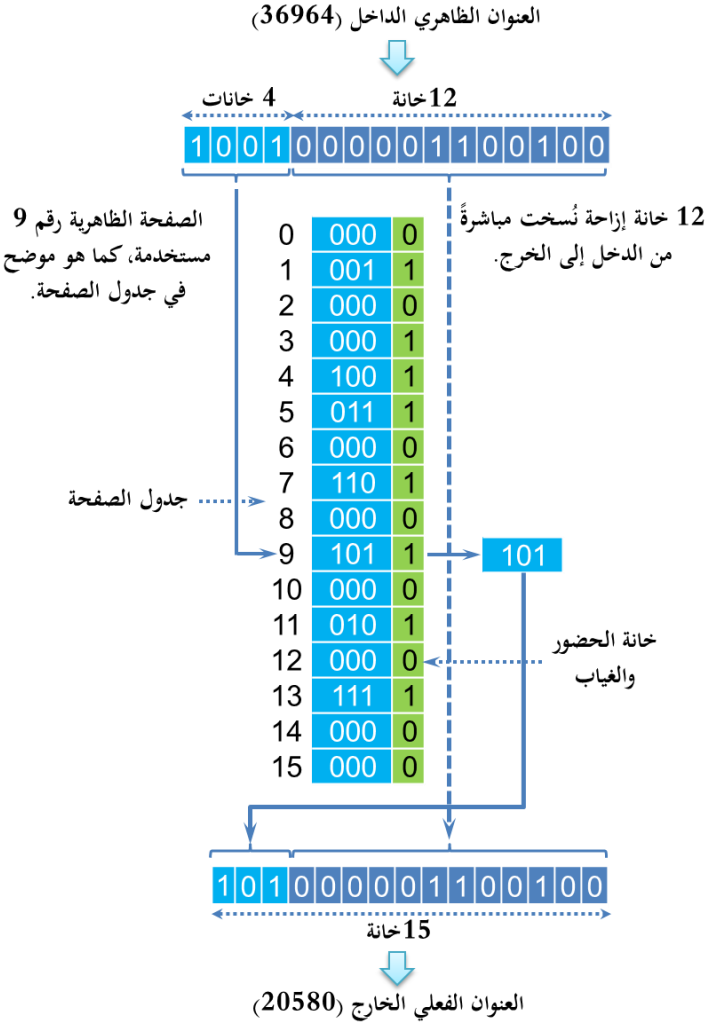
أولًا: تعليم مدخل الصفحة الظاهرية رقم **1** بأنها غير مرتبطة (أي تغيير خانة الحظور، والغياب)، لكي يتسبب مستقبلًا في خطأ الصفحة عند محاولة الوصول إلى أي من العناوين الظاهرية ما بين **4096** و **8191**.

ثانيًا: استبدال العلامة **X** في مدخل الصفحة الظاهرية رقم **6** برقم **1**، بحيث عندما يُعاد تنفيذ التعليمة المسببة لخطأ الصفحة، سيُربط العنوان الظاهري **24626** بالعنوان الفعلي **4146** (**4096 + 50**).

أخيرًا، يوضح المثال المبين في الشكل **5. 18** الطريقة التي تتبعها وحدة إدارة الذاكرة في حساب العنوان الفعلي من العنوان الظاهري، وذلك من خلال تتبع تحويل العنوان الظاهري

[18] نوع من القفز يُخبر نظام التشغيل أن الصفحة ليس لها إطار صفحة صحيح في الذاكرة الفعلية.

36964– وهو ما يناظر القيمة **1001000001100100** بالنظام الثنائي. باستخدام مفهوم التصفح الموضح في الشكل 5. 15 وخارطة وحدة إدارة الذاكرة المعطية في الشكل 5. 17 تُقسم هذه الوحدة القيمة الثنائية المناظرة للعنوان الظاهري الداخل ذو الست عشرة خانة ثنائية إلى 4 خانات تُمثل رقم الصفحة، وإلى 12خانة تُمثل مقدار الإزاحة داخل الصفحة نفسها، لذلك يُمكن تمثيل 16 صفحة، وعنونة 4096 خانة ثمانية داخل الصفحة الواحدة، على التوالي.



الشكل 5. 18: العمليات الداخلية لوحدة إدارة الذاكرة لحساب العنوان الفعلي.

يُستخدم رقم الصفحة كمؤشر داخل جدول الصفحة، ليُسفر عنه رقم إطار الصفحة المقابل لتلك الصفحة الظاهرية. إذا كانت قيمة خانة الحضور، والغياب صفرًا، فسيتسبب ذلك في إشعار نظام التشغيل بخطأ الصفحة. أما إذا كانت قيمتها واحدًا، فسيُنسخ رقم إطار الصفحة الموجود في جدول الصفحة في أعلى ثلاث خانات بسجل الإخراج، جنبًا إلى جنب مع خانات الإزاحة التي عددها 12 خانة. هذه الخانات تُنسخ من العنوان الظاهري الداخل إلى وحدة إدارة الذاكرة كما هي وبدون أي تعديل لتُشكل معًا العنوان الفعلي المتكون من 15 خانة والذي قيمته في هذا المثال 20580، يُوضع هذا العنوان على ناقل الذاكرة ليُمثل عنوان الذاكرة الفعلي.

كملخص لتقنية التصفح، عندما يُخصص أي إطار لصفحة– ما– يُترجم العنوان المنطقي إلى عنوان فعلي ويُنشأ مدخل له في جدول الصفحة، لكي يُستخدم طيلة تنفيذ البرنامج. بالتالي عندما تكون هناك عملية جاهزة للتنفيذ، تُحمل صفحاتها في إطارات الذاكرة المتاحة. في حالة عدم توفر أي اطار حر، ينقل نظام التشغيل الصفحات الخاملة أو غير المرغوب فيها حاليًا من الذاكرة إلى القرص لتحرير مساحة منها لعمليات أخرى، ومن ثم يجلب هذه الصفحات عندما يحتاجها البرنامج مرة ثانية. يستمر التنفيذ في هذه العملية حتى يُنفذ البرنامج بالكامل، بينما يحرص نظام التشغيل على نقل الصفحات الخاملة من الذاكرة الرئيسية إلى وسائط التخزين الثانوية، ومن ثم جلبها من جديد عندما يحتاجها البرنامج.

لتقنية التصفح عدة مزايا منها البساطة في التنفيذ، والكفاءة في إدارة الذاكرة، كما أنها تُلغي تَقَيُّد حجم العمل المراد تنفيذه بحجم الذاكرة المتوفر، كذلك تدعم الاتجاهات الحديثة لتطوير الحواسيب والبرمجيات، وخصوصًا تلك التي تدعم تعدد كل من المهام والمستخدمين. تُساهم أيضًا في زيادة عدد البرامج التي تُشغل آليا من خلال التحميل الجزئي للبرنامج في الذاكرة الرئيسية، ونظرًا لتساوي حجم كل من الصفحات وإطاراتها فإن عملية المبادلة ستكون سهلة، كذلك تعمل تقنية التصفح على تقليل التفتت الخارجي للذاكرة، إلَّا إنَّها تعاني من التفتت الداخلي لها.

تُعتبر هذه التقنية بالمقابل غير ملائمة للأنظمة ذات أحجام الذاكرة الصغيرة، نتيجةً لاحتياج جدول الصفحة إلى مساحة إضافية داخل الذاكرة نفسها، كما أنها تحتاج إلى عدة مكونات مادية إضافية تتطلب مكانًا في الذاكرة، ووقتًا من وحدة المعالجة المركزية لمتابعتها وتعديل محتوياتها

عند اللزوم. إضافة إلى ذلك يؤدي احتساب العنوان الفعلي (الفيزيائي)، وإجراء عملية التصفح، والبحث عن الصفحة إلى زيادة في زمن المعالجة.

3.6.5 جدول الصفحة

لاحظنا من خلال ما سبق ذكره أنَّ جدول الصفحة يلعب دورًا مهمًا في ربط العناوين الظاهرية مع العناوين الفعلية عن طريق تقسيم العنوان الظاهري إلى رقم الصفحة الظاهري (الخانات ذات المقام العالي) وإلى الإزاحة (الخانات ذات المقام المنخفض). مثلًا، مع عنوان بطول 16 خانة وحجم صفحة بمقدار 4 كيلوبايت، يُمكن للخانات الأربع العليا تحديد واحدة من الصفحات الظاهرية الست عشرة داخل جدول الصفحة، وبالاثنتي عشرة خانة تحديد الإزاحة (0–4095) داخل الصفحة نفسها.

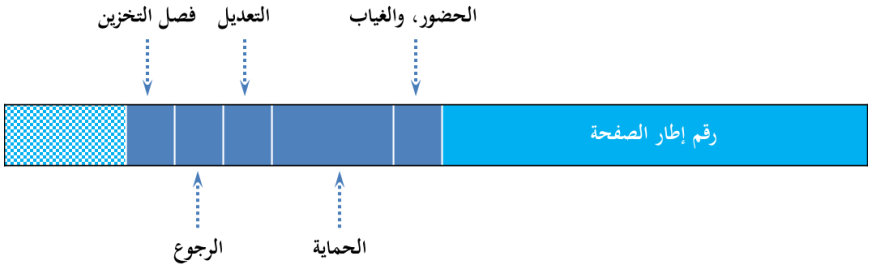
كما أشرنا سابقًا يُستخدم رقم الصفحة الظاهري داخل وحدة إدارة الذاكرة كمؤشر في جدول الصفحة للعثور على مدخل تلك الصفحة الظاهرية، الذي منه يتم العثور على رقم إطار الصفحة (إن وُجد). يُرفق هذا الرقم إلى نهاية الجزء الأعلى من الإزاحة ليحل محل رقم الصفحة الظاهري، ولتشكيل العنوان الفعلي الذى يُمكن إرساله إلى الذاكرة. هذا الأمر يجعل الغرض من جدول الصفحة يتمثل في ربط الصفحات الظاهرية مع إطارات الصفحة.

بنية مدخل جدول الصفحة

تعتمد البنية الدقيقة لمدخل جدول الصفحة بدرجة عالية على الآلة المستخدمة، إلَّا إنَّ البيانات الخاصة به تكاد تكون هي نفسها لجميع المداخل بمختلف الآلات. الأمر نفسه بالنسبة لحجم هذا المدخل غير أن الحجم الشائع له هو 32 خانة. يوضح الشكل 5. 19 عينة لأحد مداخل جدول الصفحة، أهم الحقول في هذا المدخل هو رقم إطار الصفحة، والذي يُحدد قيمته كنتيجة لربط الصفحة. إلى جانب ذلك هناك خانة الحضور، والغياب، والتي إذا كانت قيمتها تساوي واحد، فهذا يعني أن المدخل صحيح ويُمكن استخدامه، أما إذا كانت تساوي صفرًا، فالصفحة الظاهرية التي ينتمي لها المدخل غير موجودة حاليًا في الذاكرة، والرجوع إليها سيتسبب في خطأ الصفحة.

من الحقول الأخرى كذلك خانات الحماية التي تُستخدم للتحكم في أنواع الوصول

المسموح به للصفحة. أبسط صورة لهذا الحقل تتضمن خانة واحدة بحيث القيمة **0** تعني قراءة أو كتابة، بينما تعني**1** قراءة فقط. المداخل الأكثر تعقيدًا قد تحتوي على ثلاث خانات، خانة لكل من تمكين القراءة، الكتابة، وتنفيذ الصفحة.



الشكل 5. 19: البنية النموذجية لمدخل جدول الصفحة.

من الحقول الأخرى المهمة هما حقلي التعديل، والرجوع واللاذان يُستخدما لتعقب استخدام الصفحة. فعند الكتابة في الصفحة المعنية، يضبط الكيان المادي تلقائيًا خانة التعديل والتي تظهر أهميتها في الحالات التي يُقرر فيها نظام التشغيل إعادة تخصيص إطار الصفحة، بحيث إذا كانت الصفحة الموجودة فيه معدلة، فيجب تحديثها في القرص، أمَّا إذا لم تُعدل، فإنه يُمكن ببساطة التخلي عنها، لأن نسخة القرص لا تزال صالحة.

تُضبط خانة الرجوع في المقابل عند أي عملية رجوع إلى الصفحة، وذلك إما للقراءة أو للكتابة. تُساعد هذه القيمة نظام التشغيل في اختيار الصفحة التي ستُستبدل عند حدوث خطأ الصفحة. فالصفحات التي لم تُستخدم هي أفضل المرشحين للاستبدال من الصفحات التي تم الرجوع إليها. هذا الأمر له دورًا مهمًا في العديد من خوارزميات استبدال الصفحات، التي سنناقشها لاحقًا في هذا الفصل.

أخيرًا، تسمح خانة فصل التخزين بفصل التخزين المؤقت بالنسبة للصفحة. هذه الميزة مهمة للصفحات التي ترتبط مع سجلات الجهاز بدلًا من الذاكرة، إذا دخل نظام التشغيل في حلقة محكمة انتظارًا لبعض من أجهزة الإدخال، والإخراج للاستجابة لأمر صُدر له، فمن الضروري أن تستمر الأجهزة في جلب البيانات من الجهاز وعدم استخدام نسخة من تلك المخزنة سابقًا. إذًا من خلال هذه الخانة، من الممكن تمكين، وفصل التخزين المؤقت، لأنَّ الأجهزة التي لها فضاء

إدخال، وإخراج منفصل ولا تستخدم الذاكرة المعنونة لوحدة الإدخال، والإخراج قد لا تحتاج إلى هذه الخانة.

لاحظ أن عنوان القرص المستخدم للاحتفاظ بالصفحة عندما لا تكون في الذاكرة ليس جزءًا من جدول الصفحة، السبب وراء ذلك بسيط ويتمثل في كون جدول الصفحة يحوي فقط المعلومات التي يحتاجها الكيان المادي لترجمة العنوان الظاهري إلى عنوان فعلي، بينما يُحتفظ بالمعلومات التي يحتاجها نظام التشغيل للتعامل مع أخطاء الصفحات في جداول البرمجيات داخل نظام التشغيل.

4.6.5 تعجيل التصفح

من خلال التعرض إلى أساسيات الذاكرة الظاهرية والتصفح، لاحظنا أنه من المهم التعامل مع كل من قضية سرعة مواءمة العنوان الظاهري مع العنوان الفعلي، وكذلك العلاقة الطردية بين فضاء العنونة الظاهري وجدول الصفحة، أي أن كبر حجم فضاء العنونة الظاهري يؤدي إلى كبر حجم جدول الصفحة.

القضية الأولى هي نتيجة لحقيقة أن المواءمة بين العناوين الظاهرية والفعلية يجب أن تحدث عل مستوى كل عملية رجوع للذاكرة لكون جميع التعليمات يجب أن تأتي في نهاية المطاف من الذاكرة، والكثير منها قد يستخدم كذلك معاملات موجودة في الذاكرة، سيؤدي هذا الأمر إلى ضرورة الرجوع لجدول الصفحة الخاص بكل تعليمة لمرة، أو مرتين، أو في بعض الأحيان لأكثر من ذلك. فإذا كان تنفيذ تعليمة– ما– يأخذ مثلًا، واحد نانو ثانية، فيجب أن يُبحث في جدول الصفحة في أقل من 2 نانو ثانية، وذلك لتجنب جعل عملية المواءمة عقبة رئيسية من حيث السرعة.

الحاجة إلى مواءمة عدد كبير من الصفحات وبشكل سريع، تُعتبر شرطًا مهمًا في التصاميم التي تُبنى بها أجهزة الحواسيب. أبسط هذه التصاميم (على الأقل نظريًا) هو أن يكون هناك جدول صفحة واحد يحتوي على مصفوفة من السجلات المادية السريعة، التي تُمثل مداخل الصفحات الظاهرية، مفهرسة بأرقامها، كما هو مبين في الشكل 5. 18.

عندما تبدأ عملية– ما– في الاشتغال، يُحمل نظام التشغيل هذه السجلات بالإضافة إلى

جدول صفحة العملية المأخوذ من نسخة محفوظة في الذاكرة الرئيسية، وذلك لكي يُقلل من عمليات الرجوع إلى الذاكرة من أجل جدول الصفحة في أثناء تنفيذ العملية. من مزايا هذه الطريقة أنها واضحة، وصريحة، ولا تحتاج إلى الرجوع إلى الذاكرة في أثناء عملية المواءمة، أما عيبها فهي أنها مكلفة إلى حد كبير وخصوصًا إذا كان جدول الصفحة كبير. إضافة إلى ذلك الحاجة إلى تحميل جدول الصفحة بالكامل في سياق كل تبديل سيضر بأداء الحاسوب.

كتصميم آخر متطرف يُمكن أن يكون جدول الصفحة بالكامل في الذاكرة الرئيسية، بالتالي كل ما يحتاجه الكيان المادي هو سجل واحد يُشير إلى بداية هذا الجدول. هذا التصميم يجعل عملية المواءمة بين العناوين الظاهرية، والفعلية تتغير مع سياق كل تبديل عن طريق إعادة تحميل سجل واحد فقط. بطبيعة الحال، لدى هذا التصميم عيب يتمثل في الحاجة إلى الرجوع لمرة واحدة أو أكثر إلى الذاكرة لقراءة مداخل جدول الصفحة في أثناء تنفيذ كل تعليمة، مما يجعل هذه الطريقة بطيئة للغاية.

القضية الثانية متعلقة بالعلاقة الطردية بين فضاء العنونة الظاهري وجدول الصفحة، والتي انبثقت من حقيقة أن كافة أجهزة الحواسيب الحديثة تستخدم عناوين ظاهرية بطول لا يقل عن 32 خانة ثنائية، مع العلم أن طول 64 خانة ثنائية أصبح هو الآخر شائعًا على نحو متزايد. فبالنظر إلى صفحة بحجم أربعة كيلوبايت، وفضاء عنونة بطول 32 خانة ثنائية، سيكون لدينا مليون صفحة، أمَّا إذا كان فضاء العنونة بطول 64 خانة ثنائية، فسيكون الرقم أكبر من أن نفكر فيه. مع مليون صفحة في فضاء العنونة الظاهري، يجب أن يكون لدى جدول الصفحة كذلك مليون مدخل وهو ما يجعل العلاقة طردية بينهما. هنا يجب أن نتذكر بأن كل عملية تحتاج إلى جدول صفحة خاص بها، لأنها تمتلك فضاء عنونة ظاهري خاص بها، الأمر الذي سيزيد من بطء التصفح.

1.4.6.5 المترجم الجانبي للمخزن اللحظي

دعونا الآن ننظر إلى بعض الخطط التي نُفذت على نطاق واسع لتعجيل التصفح، وللتعامل مع فضاءات العنونة الظاهرية الكبيرة. نقطة الانطلاق لمعظم تقنيات التحسين تتمثل في وجود جدول الصفحة في الذاكرة، إلَّا إنَّ هذا التصميم من المحتمل أن يكون له تأثير كبير على الأداء.

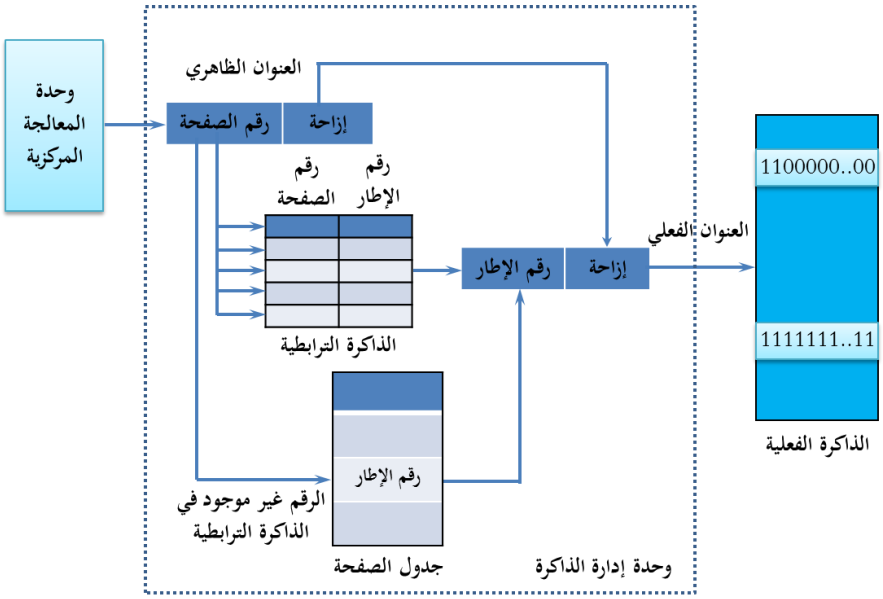
فبالنظر مثلًا، إلى تعليمة بحجم بايت تنسخ سجل– ما– إلى آخر، نجد أنها– في غياب التصفح– تحتاج إلى عملية رجوع واحدة فقط إلى الذاكرة لجلب التعليمات، بينما مع استخدام التصفح، ستكون هناك حاجة إضافية– ولو لمرة واحدة على الأقل– للرجوع إلى الذاكرة، لغرض الوصول إلى جدول الصفحة. ولأن– في العموم– سرعة التنفيذ محدودة بالمعدل الذي تستطيع به وحدة المعالجة المركزية الحصول على التعليمات والبيانات من الذاكرة، فإن إجراء عمليتي رجوع إلى الذاكرة سيُقلل الأداء بمقدار النصف، بالتالي في ظل هذه الظروف، لا يُمكن لأحد استخدام التصفح.

لقد عرف مصممي الحواسيب هذه المشكلة منذ سنوات، وقد وضعوا حل يستند على ملاحظة أن معظم البرامج تميل إلى جعل عدد كبير من عمليات الرجوع محصورة في عدد قليل من الصفحات، وليس العكس، وبالتالي جزء صغير فقط من مداخل جدول الصفحة تُقرأ بشكل مكثف، والبقية بالكاد تُستخدم على الإطلاق.

الحل القياسي لهذه المعضلة يتمثل في استخدام جهاز مادي صغير لمواءمة العناوين الظاهرية بالعناوين الفعلية دون المرور عبر جدول الصفحة. يُدعى هذا الجهاز بالمترجم الجانبي للمخزن اللحظي، وهو عبارة عن ذاكرة ترابطية سريعة جدًا. هذا الجهاز عادة ما يكون داخل وحدة إدارة الذاكرة، ويحتوي على عدد قليل من المداخل يتراوح ما بين 32 و 1024 مدخل، كل منها يحتوي على مفتاح يُمثل عنوان ظاهري ومعلومات حول صفحة واحدة، وخانة تُضبط عندما تُعدل هذه الصفحة، ورمز الحماية (قراءة/كتابة/تنفيذ)، وإطار الصفحة الفعلية المخصص لها. هذه الحقول تتناظر بشكل مماثل مع حقول جدول الصفحة، باستثناء رقم الصفحة الظاهري، الذي ليس له حاجة في جدول الصفحة، هناك حقل آخر (صحيح) يُشير إلى ما إذا كان المدخل صالح (أي قيد الاستخدام) أم لا.

تعمل الذاكرة الترابطية مع جدول الصفحة على النحو الموضح في الشكل 5. 20. تحتوي هذه الذاكرة على مجموعة قليلة من مداخل جدول الصفحة، بالتالي عندما يصل عنوان ظاهري إلى الذاكرة الترابطية لغرض موائمته، يقارنه الكيان المادي بجميع المفاتيح في وقت واحد (أي، بالتوازي). إذا وُجد تطابق وكان الوصول لا ينتهك خانة الحماية، فسيُرجَّع رقم إطار الصفحة مباشرةً دون المرور بجدول الصفحة واستخدامه في الوصول إلى الذاكرة الفعلية. إذا كان رقم

الصفحة الظاهري موجود داخل الذاكرة الترابطية وكانت التعليمة تحاول الكتابة على صفحة معدة للقراءة فقط، فستُولد رسالة خطأ حماية.



الشكل 5. 20: تسريع التصفح باستخدام المترجم الجانبي للمخزن اللحظي.

تتمثل الحالة المثيرة للاهتمام في الحدث الذي يكون فيه رقم الصفحة الظاهري غير موجود في الذاكرة الترابطية، عندما تكتشف وحدة إدارة الذاكرة غياب الرقم عن المترجم الجانبي للمخزن اللحظي، تبحث الوحدة عنه في جدول الصفحة، إن وجدته فستستخدمه للوصول إلى الذاكرة الفعلية بنفس الخطوات السابقة، كما ستُضيفه هو وكل ما يتعلق به إلى المترجم الجانبي للمخزن اللحظي للعثور عليه بسرعة في حالة ما تم الرجوع إله مرة ثانية.

أما إذا كان المترجم الجانبي ممتلئ، فيجب اختيار أحد مداخله كضحية لطرده واستبداله بمدخل جدول الصفحة الحديث. عملية الاختيار هذه يُمكن القيام بها باستخدام إحدى خوارزميات استبدال الصفحات التي سنناقشها لاحقًا. عندما يُحذف مدخل، تُنسخ خانة التعديل في مدخل جدول الصفحة داخل الذاكرة، القيم الأخرى ستكون موجودة بالفعل هناك، باستثناء خانة الرجوع. في حين عندما يُحمل مدخل من جدول الصفحة إلى المترجم الجانبي تُأخذ كافة الحقول من الذاكرة.

2.4.6.5 إدارة برمجيات المترجم الجانبي للمخزن اللحظي

العديد من الأجهزة الحديثة تُدير كل ما يتعلق تقريبًا بالتصفح في إطار البرمجيات، حيث تُحمل مداخل المترجم الجانبي بشكل صريح من قبل نظام التشغيل، بالتالي عندما تحدث حالة المترجم المفقود وهي الحالة التي لا يُعثر فيها على الصفحة المطلوبة داخل المترجم الجانبي، تُولد وحدة إدارة الذاكرة إشعار بهذه الحالة وترمي بالمشكلة في حضن نظام التشغيل، وذلك بدلًا من أن تذهب هي نفسها إلى جداول الصفحة لإيجاد وجلب الصفحة المطلوبة التي تم الرجوع إليها مؤخرًا. في هذه الحالة، يجب على النظام العثور على الصفحة، وحذف مدخل من المترجم الجانبي واستبداله بالمدخل الجديد، ومن ثم إعادة تشغيل التعليمة التي تسببت في هذا الخطأ. بطبيعة الحال، كل هذا يجب أن يتم بعدد قليل من التعليمات، لأن تكرار خطأ حالة المترجم المفقود يحدث أكثر بكثير من أخطاء الصفحات.

هناك عدة استراتيجيات مختلفة طُورت لغرض تحسين أداء الأجهزة التي تُدير المترجم الجانبي للمخزن اللحظي عن طريق البرمجيات، فللحد من أخطاء المترجم، يُمكن لنظام التشغيل– أحيانًا– استخدام الحدس لمعرفة أيٍ من الصفحات من المرجح أن تُستخدم لاحقًا، ومن ثم تحميل مداخلها مسبقًا في المترجم مثلًا، عندما تُرسل عملية عميل رسالة إلى عملية خادم على نفس الجهاز، من المحتمل جدًا أن الخادم سوف يشتغل قريبًا، بالتالي في أثناء معالجة القفز للقيام بالإرسال، يُمكن للنظام التحقق من مكان شفرة الخادم، والبيانات، وصفحات المكدس، ومن ثم مواءمتها قبل أن تحين الفرصة للتسبب في أخطاء المترجم.

للإطلاع على تفاصيل أكثر بخصوص برمجيات إدارة المترجم الجانبي للمخزن اللحظي يُمكن الرجوع إلى أولج وآخرون (**Uhlig et al., 1994**)، كما أنه يُمكن الرجوع إلى بالا وآخرون (**Bala et al., 1994**) للتعرف على إحدى الطرق المستخدمة للحد من خطأ المترجم وكذلك تكلفته.

أخيرًا، عند إدارة برمجيات المترجم الجانبي للمخزن اللحظي، من الضروري أن نفهم الفرق بين نوعين من أنواع أخطاء المترجم. الخطاء البسيط: ويحدث عندما لا تتواجد الصفحة المرجوع إليها في المترجم، بينما تكون موجودة في الذاكرة. في هذه الحالة، كل ما هو مطلوب هو فقط

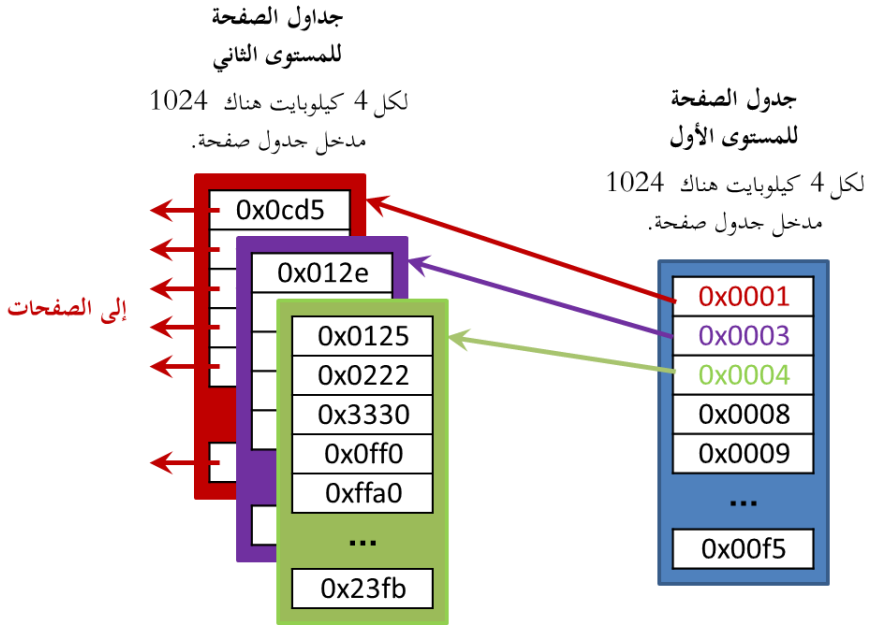
تحديث المترجم من دون أن تكون هناك حاجة إلى الرجوع إلى القرص. في العادة هذا النوع من الأخطاء يأخذ من عشرة إلى عشرين تعليمة آلة للتعامل معه، ويُمكن أن يكتمل في غضون بضعة نانو ثانية. في المقابل، يحدث الخطاء المعقد عندما تكون الصفحة نفسها غير موجودة في الذاكرة (وبطبيعة الحال، غير موجودة في المترجم)، هنا سيكون من الضروري الوصول إلى القرص لجلب الصفحة، الأمر الذي يتطلب عدة ملِّي ثانية، وهو أبطأ مليون مرة من الخطاء البسيط.

5.6.5 جداول الصفحة للذواكر الكبيرة

تناول القسم السابق استخدام المترجم الجانبي للمخزن اللحظي لمعالجة إحدى مشاكل تقنية التصفح وهي بطء عملية ترجمة العنوان الظاهري إلى العنوان الفعلي عن طريق جدول الصفحة الموجود في الذاكرة. من المشاكل الأخرى التي تواجهها هذه التقنية هي مشكلة التعامل مع فضاءات العنونة الظاهرية الكبيرة جدًا. فيما يلي ستُناقش طريقة كل من جداول الصفحة متعددة المستويات، وجداول الصفحة المعكوسة للتعامل مع هذه المشكلة.

1.5.6.5 جداول الصفحة متعددة المستويات

تتمثل طريقة جداول الصفحة متعددة المستويات في استخدام جدول صفحة يتميز بتعدد مستوياته، كما هو موضح في الشكل 5. 21. السر وراء طريقة جداول الصفحة متعددة المستويات يكمن في تجنب الاحتفاظ بكافة جداول الصفحة في الذاكرة في جميع الأوقات، بالذات تلك التي ليست هناك حاجة بها لأن تبقى حاليًا في الذاكرة، الأمر الذي سيؤدي إلى توفير في مساحة الذاكرة. على سبيل المثال، بالنظر إلى آلة ذات 32 خانة ثنائية ولها صفحات بحجم أربعة كيلوبايت، نجد أن عدد مداخل جدول الصفحة يصل إلى واحد ميجابايت (32 خانة – 12 خانة (إزاحة الصفحة) = 20 خانة، أي 2^{20} = 1 ميجابايت) ومع تخصيص أربع خانات ثمانية لكل مدخل سيكون حجم جدول الصفحة أربعة ميجابايت، وهو حجم معقول، ولكن مع وجود مئة برنامج يشتغل سيصل إجمالي حجم جداول الصفحة إلى 400 ميجابايت، لأن كل برنامج يحتاج إلى جدول صفحة خاص به للتعامل مع فضاء العناوين الظاهرية، وهو ما سيزيد الأمر تعقيدًا، ويجعل مبادلة الجداول أمرًا صعبًا.

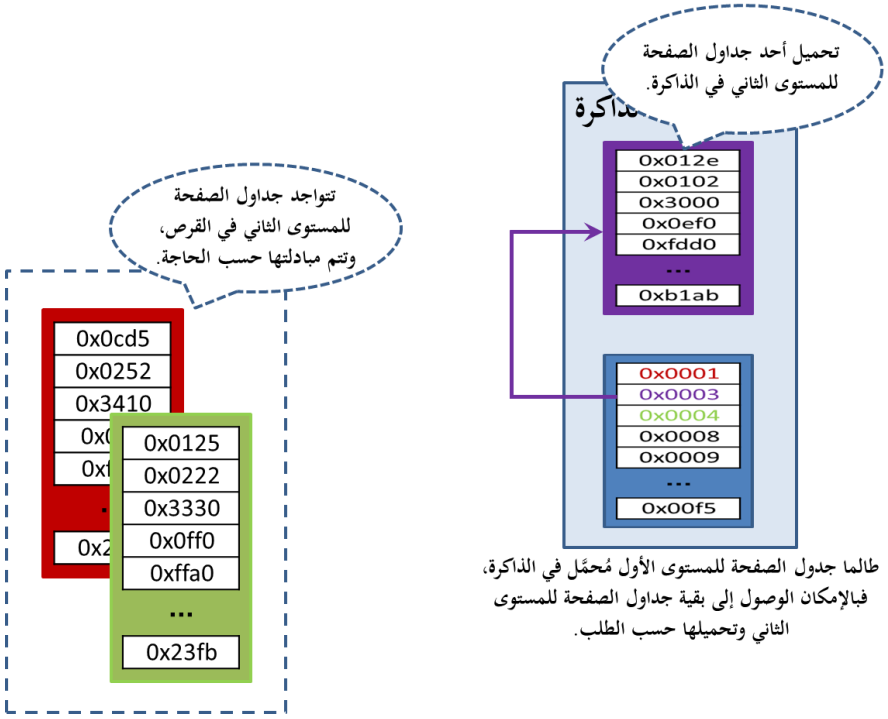


الشكل 5. 21: جداول الصفحة متعددة المستويات.

الأمر سيكون مختلفًا عند استخدام جداول الصفحة متعددة المستويات. نرى في الشكل 5. 21 مثالًا يُوضح كيفية عمل جدول صفحة ذو مستويين. على اليمين لدينا جدول الصفحة للمستوى الأول، الذي يحتوي على 1024 مدخل، والتي تُشير إلى جداول الصفحة للمستوى الثاني كل واحد منها يُمثل 4 ميجابايب، لأن فضاء العنوان الظاهري بالكامل بسعة 4 قيقابايت (32 خانة ثنائية) جُمِّع في مجموعات بطول 4 كيلوبايت.

بهذه الطريقة أصبح ليس من الضروري تحميل كل الجداول في الذاكرة وإنما سيُكتفى مبدئيًا فقط بتحميل جدول الصفحة للمستوى الأول، بينما تتواجد كافة جداول الصفحة للمستوى الثاني في القرص، وتتم مبادلتها حسب متطلبات تنفيذ البرامج، أو التطبيقات، كما هو موضح في الشكل 5. 22، لأنه مع تواجد جدول الصفحة للمستوى الأول في الذاكرة، باستطاعة نظام التشغيل الوصول إلى بقية جداول الصفحة للمستوى الثاني وتحميلها حسب الطلب، وذلك لكي يستخدمها في ترجمة العناوين الظاهرية إلى فعلية كما سيُوضح تاليًا. في هذه الحالة سيكون أقل حجم من بيانات جدول الصفحة تحتاج إلى حفظه في الذاكرة لتطبيق 32 خانة ثنائية هو أربعة

كيلوبايت لجدول المستوى الأول بالإضافة إلى أربعة أخرى لأحد جداول المستوى الثاني، أي بإجمالي ثمانية كيلوبايت، وهو أقل بكثير من أربعة ميجابايت لكامل التطبيق.

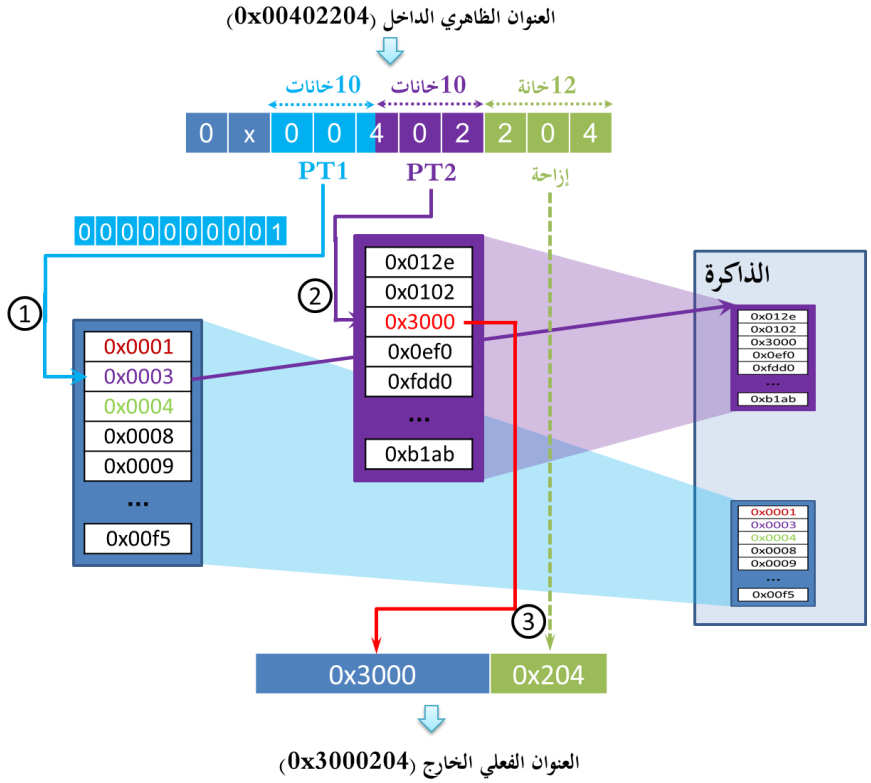


الشكل 5. 22: تحميل جدول الصفحة للمستوى الأول مع أحد جداول المستوى الثاني في الذاكرة.

ولنتعرف الآن على الآلية المستخدمة في طريقة جداول الصفحة متعددة المستويات لترجمة العناوين الظاهرية إلى فعلية. في المثال المبين بأعلى الشكل 5. 23 لدينا عنوان ظاهري (0x00402204) بطول 32 خانة ثنائية مُقسم إلى حقل PT1 وحقل PT2 بطول عشر خانات ثنائية لكل واحد منهما، وحقل إزاحة بطول 12 خانة ثنائية. ولأن طول الإزاحة هو 12، فسيكون حجم الصفحة أربعة كيلوبايت، وسيكون هناك ما مجموعه 2^{20} صفحة.

عندما يُرسل هذا العنوان الظاهري إلى وحدة إدارة الذاكرة، تستخلص هذه الوحدة أولًا حقل PT1 لاستخدامه كمؤشر في جدول الصفحة للمستوى الأول، والذي ينتج عنه عنوان أو رقم

إطار الصفحة الخاص بجدول الصفحة للمستوى الثاني وهو المدخل 1، كما هو موضح في الشكل 5. 23– الخطوة رقم 1. من ثم تستخدم PT2 لفهرسة جدول الصفحة للمستوى الثاني– التي حصلت عليه للتو– واستخلاص المدخل 2 (الخطوة رقم 2 في نفس الشكل)، هذا المدخل يحتوي الآن على رقم إطار الصفحة الذي يحتوي على العنوان الظاهري. إذا كانت هذه الصفحة غير موجودة في الذاكرة، فإن خانة الحضور والغياب ستكون صفرًا داخل مدخل جدول الصفحة، الأمر الذي سيتسبب في خطأ الصفحة، أمَّا إذا كانت الصفحة في الذاكرة، فسيُجمع رقم إطار الصفحة المأخوذ من جدول الصفحة للمستوى الثاني مع الإزاحة المتكونة من 12 خانة لبناء العنوان الفعلي لينتج عن ذلك العنوان 0x3000204، كما هو موضح في الخطوة 3 من نفس الشكل، بعدها يُوضع هذا العنوان على ناقل العنونة، ومن ثم يُرسل إلى الذاكرة.



الشكل 5. 23: آلية ترجمة عنوان ظاهري بطول 32 خانة ثنائية إلى عنوان فعلي لجداول الصفحة ذات المستويين.

الشيء المثير للاهتمام والذي يُمكن ملاحظته حول الشكل 5. 22 أو الشكل الذي يليه هو أنه بالرغم من أن فضاء العنونة يحتوي على أكثر من مليون صفحة، هناك حاجة فعلية لجدول المستوى الأول، وبعض من جداول المستوى الثاني التي حاليًا مطلوبة، بقية مداخل جدول الصفحة للمستوى الثاني تُضبط خانة الحضور والغياب الخاصة بها على القيمة 0، وذلك لإصدار خطأ الصفحة إذا ما تم الوصول إليها في أي وقت. إذا حدث هذا، فإن نظام التشغيل سيلاحظ أن عملية– ما– حاولت الرجوع إلى الذاكرة في الوقت الذي ليس من المفترض أن ترجع إليها، بالتالي يجب عليه اتخاذ الإجراءات المناسبة، مثل إرساله إشارة بالخصوص أو قتل العملية.

نظام جدول الصفحة ذو المستويين الموضح في الشكل 5. 21 يُمكن توسيعه ليشمل ثلاث وأربع مستويات، أو أكثر من ذلك. علمًا إن استخدام مستويات إضافية يُعطي مزيدًا من المرونة، إلَّا إنَّه ستكون هناك تعقيدات إضافية في حالة اعتماد أكثر من ثلاث مستويات.

2.5.6.5 جدول الصفحة المعكوس

بظهور حواسيب 64 خانة ثنائية وشيوعها بشكل كبير بدأت تظهر مشكلة كبر حجم جدول الصفحة. فمع وجود عنوان ظاهري بطول 64 خانة ثنائية يكون فضاء العنونة 2^{64} بايت، ومع استخدام صفحات بحجم 4 كيلوبايت، فستكون هناك حاجة إلى جدول صفحة به 2^{52} مدخل، وإذا كان طول كل مدخل 8 بايت، فسيكون حجم الجدول أكبر من 30 مليون قيقابايت. الأمر الذي سيجعل مقدار الحمل الإضافي لتخزين جدول الصفحة في الذاكرة كبيرًا جدًا، لذا وجب البحث عن حل مختلف للتعامل مع فضاءات العنونة الظاهرية الكبيرة.

تتمثل الطريقة البديلة في جعل فهرسة جداول الصفحة تتم عن طريق رقم الإطار بدلًا من رقم الصفحة المنطقي، الأمر الذي يُشار إليه بتقنية التصفح المقلوب، أو جدول الصفحة المعكوس. ولتوضيح فكرة هذه الطريقة سنفترض أنَّ هناك عملية لها فضاء عنونة حجمه 2^{52} صفحة، والتي سيناظرها نفس العدد من المداخل في جدول الصفحة لإحتواء كافة المعلومات الخاصة بالصفحات، بالتالي هناك دائمًا مساحة مخصصة لهذه الصفحات داخل جدول الصفحة بغض النظر عن وجودها في الذاكرة الرئيسية أم لا، علمًا بأن هذه الذاكرة لا يُمكنها استعاب إلَّا عددًا محدودًا من هذه الصفحات وفقًا لعدد الإطارات الموجودة بها، وهو ما يعني إهدار جزء من

المساحة المخصصة للجدول وخصوصًا إذا كانت الصفحة غير محمَّلة في الذاكرة.

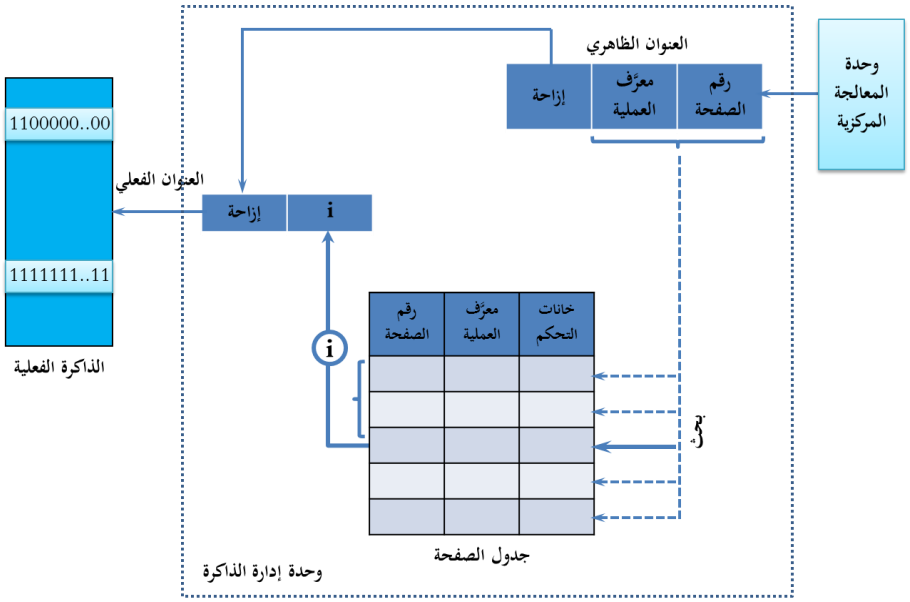
لتخفيف حمل التخزين الإضافي واستغلال المساحة المهدورة، يُفهرس جدول الصفحات برقم الإطار، بالتالي تتساوي عدد الإدخالات بعدد الإطارات في الذاكرة الرئيسية، وسوف لن تكون هناك حاجة لتخزين جدول صفحة مقابل كل عملية، وإنما ستكون هناك حاجة فقط لجزء ثابت من الذاكرة لتخزين معلومات الترحيل الخاصة بجميع العمليات معًا، الأمر الذي يُساهم بشكل فعَّال في التغلب على عيوب جداول الصفحة الكبيرة. مثلًا، مع فضاء عنونة ظاهري بطول 64 خانة ثنائية، وصفحة بحجم 4 كيلوبايت، وذاكرة وصول عشوائي بحجم واحد قيقابايت، يتطلب جدول الصفحة المعكوس فقط 262144 مدخل، كل منها يتتبع أيٍ من العمليات والصفحات الظاهرية الواقعة في إطار الصفحة.

مع استخدام هذه الطريقة في إدارة الذاكرة، نجد أنَّ عدد المداخل في جدول الصفحة المعكوس يتساوى مع عدد الإطارات في مساحة العنوان الفعلي، وأنَّ كل مدخل في جدول الصفحة سوف يحتوي على الحقول التالية:

- معرِّف العملية: حيث أنَّ جدول الصفحة المعكوس يحتوي على معلومات مساحة العنوان لكافة العمليات قيد التنفيذ، كان لزامًا تخزين معرِّف عملية لكل منها في جدول الصفحة المعكوس، وذلك لتحديد مساحة عنوانها بشكل فريد– باستخدام معرِّفها ورقم الصفحة– في أثناء احتواء عمليتين مختلفتين على مجموعة متماثلة من العناوين الافتراضية.
- رقم الصفحة: يُستخدم لتحديد نطاق رقم الصفحة الخاص بالعنوان المنطقي.
- خانات التحكم: تُستخدم هذه الخانات للاحتفاظ بمعلومات إضافية ذات الصلة بالتصفح، وتشمل خانة الصحة، وخانة المرجعية، وخانات أخرى تتضمن معلومات الحماية والقفل.
- مؤشر الربط: عندما تتشارك عمليتين أو أكثر في جزء من الذاكرة الرئيسية، يكون من المهم مواءمة صفحتين أو أكثر من الصفحات المنطقية في نفس مدخل جدول الصفحة، لذلك يُستخدم مؤشر الربط لمواءمة تفاصيل هذه الصفحات مع جذر جدول الصفحة.

يُجسِّد الشكل 5. 24 آلية عمل أبسط طريقة لجدول الصفحة المعكوس والمستخدمة في جهاز IBM RT. يُقسم العنوان الظاهري الذي أُنشأ بواسطة وحدة المعالجة المركزية في هذه

الطريقة إلى ثلاثة أجزاء تُمثل رقم الصفحة، ومعرَّف العملية، والإزاحة. عند حدوث حالة رجوع إلى الذاكرة، يُقارن الجزئين الأولين من هذا العنوان بواسطة وحدة إدارة الذاكرة مع فهرس جدول الصفحة المعكوس للبحث عن مطابقة لرقم الإطار المقابل. إذا وُجد تتطابق في المدخل i، يُرسل كعنوان حقيقي إلى الذاكرة الفعلية بعد أن تُضاف إليه الإزاحة، أمَّا إذا لم يكن هناك أي تتطابق، فسيُعلن عن خطأ صفحة.



الشكل 5. 24: طريقة عمل مفهوم جدول الصفحة المعكوس.

على الرغم من أنَّ جدول الصفحة المعكوس وفَّر كميات كبيرة من مساحة التخزين المطلوبة لتخزين جداول الصفحات – على الأقل عندما يكون فضاء العنونة الظاهري أكبر بكثير من الذاكرة الفعلية– إلَّا إنَّه يُعاني من التعقيد الكبير في ترجمة العنوان الظاهري إلى عنوان فعلي، ومن طول زمن البحث للعثور على المدخل المناسب، وكذلك من صعوبة تطبيق الذاكرة المشتركة، لكون أنه يُخزن مدخلًا واحدًا لكل إطار، الأمر الذي استوجب استخدام مؤشر الربط لتعيين أكثر من عنوان افتراضي للمدخل المحدد بترتيب رقم الإطار.

يتمثل بالمقابل السبيل للخروج من معضلة طول زمن البحث في جدول الصفحة المعكوس في استخدام المترجم الجانبي للمخزن اللحظي الذي تعرضنا له سابقًا. يُمكن القول أنه إذا كان

بإمكان هذا المترجم استعاب كل الصفحات المستخدمة بشكل مكثف، فستحدث الترجمة بنفس السرعة التي نتحصل عليها عند استخدام جداول الصفحة العادية. أمَّا في حالة حدوث خطأ المترجم لابد من البحث عن جدول الصفحة المعكوس في مجال البرمجيات. في حالة عدم كفاءة هذه الطريقة يُمكن استخدام جدول هاش لتوليد مفاتيح للمداخل وحفظها في جدول التقطيع، ومن ثم استخدام دالة هاش لمواءمة المفاتيح مع محتوياتها، وإرجاع الجزء الذي سيُضاف إلى الإزاحة لتوليد العنوان الفعلي للذاكرة الرئيسية.

من الأمثلة المستخدمة لهذه النوع من الإدارة أنظمة **PowerPC** و **UltraSPARC** و **IA-64**، كما أنه من الممكن الرجوع إلى طرق أخرى للتعامل مع الذواكر الظاهرية الكبيرة في تالوري وآخرون (**Talluri et al., 1995**).

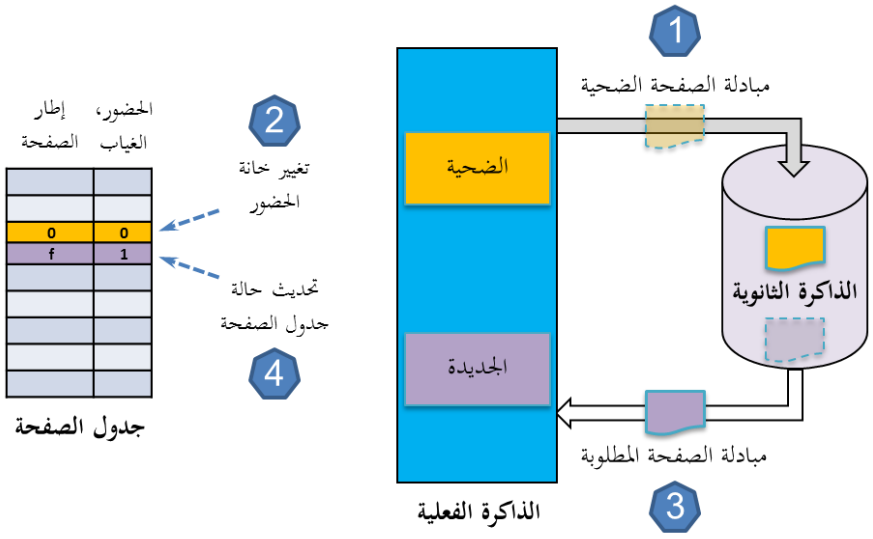
7.5 خوارزميات استبدال الصفحة

أشرنا في القسم الخامس إلى مفهموم المبادلة الذي ينص على تبديل العمليات بشكل مؤقت ما بين الذاكرة الرئيسية والذاكرة الثانوية، الغرض من ذلك هو إتاحة الفرصة إلى تلك العمليات التي لم تجد لها مكان في الذاكرة الرئيسية في أثناء تشغيلها، وبناءً عليه يجب في هذه الحالة إنشاء أماكن فارغة داخل الذاكرة الرئيسية وذلك من خلال اختيار إحدى الصفحات غير المرغوب فيها كضحية وطردها (أي إزالتها من الذاكرة)، لغرض إفساح المجال أمام الصفحة الواردة. إذا عُدِّلت الصفحة المراد إزالتها في أثناء وجودها في الذاكرة، فإنه من الضروري تحديث نسختها في القرص، أمَّا إذا لم يحدث أي تغيير عليها فلن تكون هناك حاجة إلى تحديثها، أي لن يتطلب الأمر إعادة كتابتها في القرص.

من الجدير بالذكر أيضًا أنَّ مفهوم استبدال الصفحة يحدث كذلك في مجالات أخرى من تصميم الحاسوب. مثلًا، لدى معظم أجهزة الحواسيب ذاكرة كاش واحدة أو أكثر تحتوي على مجموعة من مقاطع الذاكرة المستخدمة مؤخرًا. عند امتلاء هذه الذاكرة، يجب اختيار بعض من هذه المقاطع لغرض حذفها وهو ما يتوافق بالضبط مع آلية استبدال الصفحة، إلَّا إنَّ الفرق الجوهري يكمن في قصر الفترة الزمنية (الأمر الذي ينبغي القيام به في غضون بضعة نانو ثانية، وليست بضعة ملِّي ثانية، كما هو الحال مع استبدال الصفحة). السبب وراء القصر الزمني في

كون مقاطع ذاكرة كاش المفقودة تُجلب من الذاكرة الرئيسية التي لا تتطلب زمنًا للبحث ولن يكون هناك تأخير ناتج عن دوران الأقراص.

يُمكن تلخيص أحداث استبدال الصفحات في العموم في خطوات عامة موضحة في الشكل 5. 25، بعد اختيار الضحية تُبادل إلى القرص كخطوة أولى. في الخطوة الثانية، تُغير خانة الحضور، والغياب للصفحة الضحية داخل جدول الصفحة. في الخطوة الثالثة، تُبادل الصفحة المطلوبة من القرص إلى الذاكرة، أخيرًا تُحدَّث حالة جدول الصفحة بالنسبة للصفحة الجديدة.



الشكل 5. 25: مفهوم استبدال الصفحات.

يُمكن تفصيل هذه الخطوات بشكل أكثر وضوحًا إلى النقاط التالية:

1. تحديد موقع الصفحة المطلوبة على القرص.
2. البحث عن إطار حر لها في الذاكرة.
   أ. إذا توفَّر هذا الإطار، فسيُستخدم.
   ب. في حالة عدم توفر إطار حر، فستُستخدم إحدى خوارزميات استبدال الصفحة لاختيار الضحية وإنشاء إطار حر.
   ج. إذا أُجري أي تعديل على الصفحة الضحية، فيجب تحديث نسختها على القرص.

3. جلب الصفحة الجديدة وتحميلها في الإطار الجديد مع تحديث جدول الصفحة.
4. الاستمرار في تنفيذ عملية المستخدم من اللحظة التي حدث فيها خطأ الصفحة.

قد يكون من الممكن اختيار صفحة عشوائية كضحية لطردها عند حدوث خطأ الصفحة، ولكن قد يكون من الأفضل بكثير بالنسبة لأداء النظام اختيار الصفحة نادرة الاستخدام، لأنه إذا ما أُختيرت صفحة مستخدمة بكثرة فإنه على الأرجح ستُجلب من جديد بسرعة مما يؤدي إلى أعمال إضافية. السؤال المطروح هنا: ما هي الكيفية التي ستُنفذ بها عملية استبدال هذه الصفحات، وكيف سيتُعرف على الصفحات غير المرغوب فيها حاليًا؟ بمعنى آخر كيف يتم التعرف على الصفحات نادرة الاستخدام، حتى تتسنى عملية مبادلتها لاحقًا؟ في واقع الأمر، لقد أُنجز الكثير من العمل في موضوع خوارزميات استبدال الصفحات، سواءً أكان ذلك نظريًا أو تجريبيًا، أدناه سنشرح بعض من أهم هذه الخوارزميات.

ملاحظة: لجميع خوارزميات استبدال الصفحة، من المهم جدًا تحديد تسلسل رجوع كل عملية إلى صفحاتها الخاصة بها في أثناء عملية التنفيذ، يُسمى هذا التسلسل بتيار مرجع الصفحة **R** وهو يُعرَّف على النحو التالي:

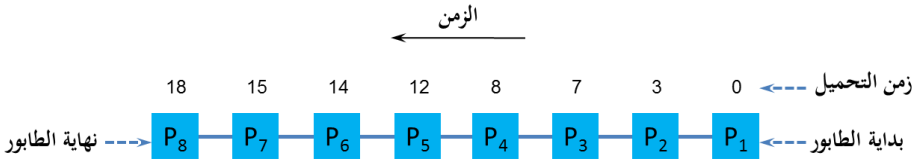
$$R = [r_1, r_2, r_3, \ldots, r_k]$$

حيث تعني r_i الروجوع إلى الصفحة **i**، يُمثل التسلسل: **[5، 2، 0، 3، 1، 4، 6، 8]** مثال لتيار مرجع الصفحة، والذي يُمثل سلوك العملية في أثناء عملية التنفيذ.

1.7.5 خوارزمية الداخل أولًا يخرج أولًا لاستبدال الصفحات

تُعتبر خوارزمية الداخل أولًا يخرج أولًا لاستبدال الصفحات من أبسط خوارزميات الاستبدال، فهي تعتمد على حفظ الصفحات الموجودة حاليًا داخل الذاكرة في طابور، بحيث تتواجد أقدم الصفحات وصولًا في مقدمته، بينما تكون أحدثها في مؤخرته، كما هو موضح في الشكل 5. 26، والذي يوضح الصفحات من P_1 إلى P_8 محفوظة في طابور مرتبط، ومرتبة تبعًا لزمن وصولها إلى الذاكرة. عند حدوث خطأ الصفحة ولم يكن هناك إطار حر داخل الطابور، يستبدل نظام التشغيل الصفحات عن طريق حذف الصفحة المتواجدة في مقدمته وإضافة الصفحة الحديثة من ناحية مؤخرته. طريقة الحذف هذه لا تضع أي قيود على الصفحات المحذوفة حتى

ولو كانت كثيرة الاستخدام، الأمر الذي يترتب عليه إعادة تحميلها عدة مرات، وهو ما يؤدي إلى ضياع الوقت والجهد. لهذا السبب يندر استخدام هذه الخوارزمية بشكلها الحالي، الحسنة الوحيدة لها هي بساطتها وسهولة تنفيذها.



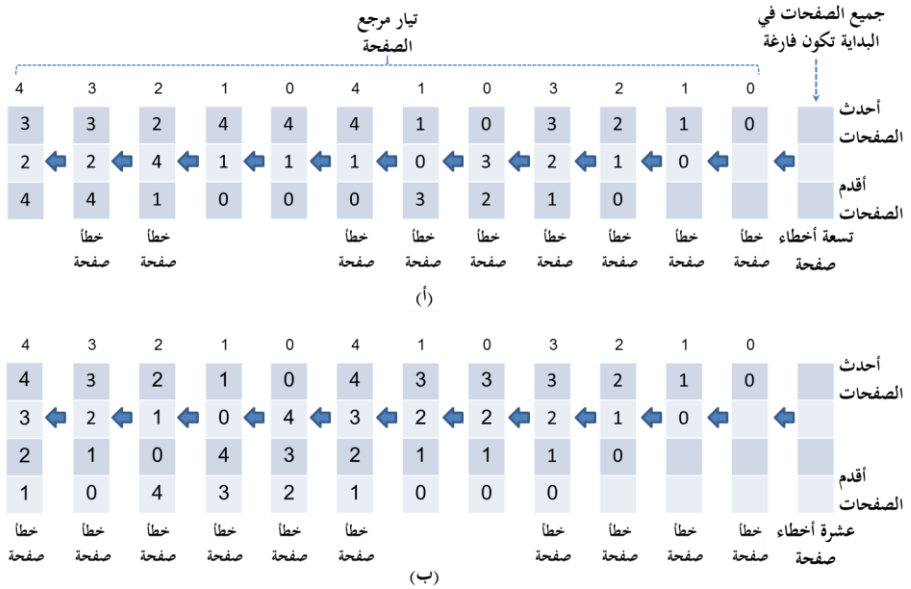
الشكل 5. 26: طابور خوارزمية الداخل أولًا يخرج أولًا لاستبدال الصفحات.

من خلال تتبع آلية عمل خوارزمية الداخل أولًا يخرج أولًا لاستبدال الصفحات، قد يبدو للوهلة الأولى أنه كلما زاد حجم المساحة المخصصة للعملية أي زاد عدد إطارات الصفحة داخل الذاكرة، كلما قلَّ تكرار حدوث خطأ الصفحة، ولكن هذا ليس صحيحًا بالمطلق. في سنة **1969** قدَّم العالم بلادي (**Belady**) مثالًا أوضح من خلاله عدم صحة هذا الاعتقاد، وأصبح يُعرف فيما بعد بشذوذ بلادي، المثال التالي يوضح هذه الظاهرة.

يُمثل الشكل 5. 27-أ برنامج يتكون من خمس صفحات ظاهرية أرقامها **0**، **1**، **2**، **3**، و **4**، وكان تيار مرجع الصفحة على النحو التالي:

[4 ،3 ،2 ،1 ،0 ،4 ،1 ،0 ،3 ،2 ،1 ،0]

في حين يحوي الطابور على ثلاثة إطارات. بتتبع تحقيق الطلبات من خلال هذا الشكل، نجد أن عدد أخطاء الصفحة يصل إلى تسعة أخطاء، أمَّا في حالة استخدام طابور بحجم أربع إطارات ولنفس المعطيات السابقة، فإننا سنجد أن عدد أخطاء الصفحة قد أصبح عشرة، وذلك كما هو موضح في الشكل 5. 27-ب.

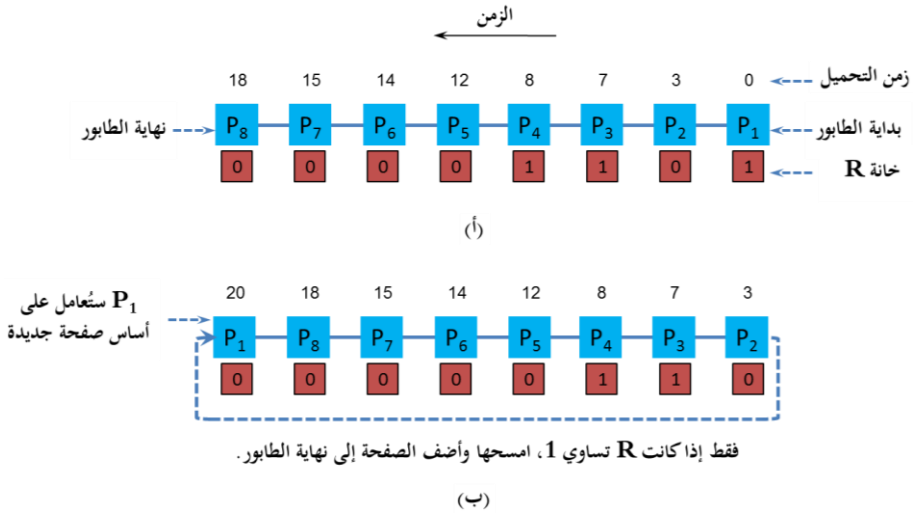


الشكل 5. 27: توضيح شذوذ بلادي.

2.7.5 خوارزمية الفرصة الثانية لاستبدال الصفحات

تعتمد فكرة خوارزمية الفرصة الثانية في الأساس على خوارزمية الداخل أولًا يخرج أولًا لاستبدال الصفحات، إلَّا إنَّها تتجنب حذف الصفحات المستعملة بشكل مكثف من خلال إجراء تعديل بسيط عليها يتمثل في إضافة خانة حالة مرتبطة بكل صفحة. تُسمى هذه الخانة بخانة الرجوع **R** وتُضبط باستخدام الكيان المادي عند حدوث أي عملية رجوع (كتابة أو قراءة) للصفحة المرتبطة بها. بالتالي تُفحص هذه الخانة والخاصة بالصفحة القديمة المراد حذفها قبل عملية الحذف، إذا كانت قيمتها **0**، فهذا يعني أن الصفحة المرتبطة بها على حد سواء قديمة وغير مستخدمة وستُستبدل على الفور، أمّا إذا ساوت قيمتها **1**، فستُعطى للصفحة فرصة ثانية وسيُعاد ضبط الخانة على **0**، ومن ثم يُضيف نظام التشغيل هذه الصفحة من جديد إلى مؤخرة طابور الصفحات، وستُعامل كما لو أنَّها وصلت للتو إلى الذاكرة، ثم يستمر البحث. هذا يعني أن أي عملية مُنحت لها فرصة ثانية سوف لن تُستبدل حتى تُستبدل جميع الصفحات في الطابور (أو تُعطى لها فرصة ثانية)، كذلك أي صفحة تُستخدم بكثرة وتُضبط خانة **R** دائمًا على **1** سوف لن تُستبدل أبدًا.

للتعرف أكثر على هذه الخوارزمية يُمكن تتبع المثال المبين في الشكل 5. 28. في الشكل 5. 28–أ نرى الصفحات من P_1 إلى P_8 محفوظة في طابور مرتبط، ومرتبة تبعًا لزمن وصولها إلى الذاكرة، ومُقترنةً بخانة الرجوع.



الشكل 5. 28: خوارزمية الفرصة الثانية– أ) الصفحات مرتبة زمنيًا بنظام الطابور– ب) طابور الصفحات بعد حدوث خطأ الصفحة، وكانت خانة R بالنسبة للصفحة P_1 تساوي 1.

لنفترض أن خطأ الصفحة حدث في الزمن 20، أقدم الصفحات هي P_1، التي وصلت في الزمن 0، أي عندما بدأت العملية. إذا كانت قيمة الخانة R الخاصة بالصفحة P_1 تساوي 0، فستُطرد هذه الصفحة من الذاكرة مباشرةً، إمَّا عن طريق كتابتها في القرص (إذا كانت معدلة)، أو فقط التخلي عنها (إذا كانت نظيفة). من ناحية أخرى، إذا كانت قيمة R تساوي 1، فستُوضع الصفحة P_1 في نهاية الطابور، وسيُضبط زمن التحميل على الوقت الحالي وهو 20، كما سيُعاد كذلك ضبط الخانة R على 0، ثم يستمر البحث عن الصفحة المناسبة انطلاقًا من P_2، كما هو مُفصل في الشكل 5. 28–ب.

ما تقوم به خوارزمية الفرصة الثانية هو البحث عن صفحة قديمة لم يُرجع إليها في الفترة النبضية الأخيرة. أمَّا إذا ما تم الرجوع إلى كل الصفحات، فستتساوى خوارزمية الفرصة الثانية مع طريقة الخوارزمية السابقة. على وجه التحديد، بفرض أن خانة R لكل الصفحات في الشكل 5.

28–أ تساوي 1، الأمر الذي سيجعل نظام التشغيل ينقل الصفحات واحدة تلو الأخرى إلى نهاية الطابور، ومن ثم إعادة ضبط الخانة R على 0 في كل مرة يُلحق به صفحة إلى نهايته. في نهاية المطاف، يرجع إلى الصفحة P_1، التي خانتها R الآن تساوي 0، والتي سيحذفها نظام التشغيل– في هذه الحالة–، وهو ما سيجعل الخوارزمية تنتهي دائمًا.

3.7.5 خوارزمية الاستبدال الأمثل للصفحات

من نتائج اكتشاف شذوذ بلادي هو البحث عن خوارزمية الاستبدال الأمثل للصفحات والتي لها أقل معدل خطأ صفحة، ولن تعاني أبدًا من هذه الظاهرة. تُعتبر هذه الخوارزمية من أسهل الخوارزميات وصفًا، ولكن من أصعبها تطبيقًا، فهي تسعى إلى استبدال الصفحة الأقل استخدامًا، أو نادرة الاستخدام.

تتطلب هذه الخوارزمية تَوفر معلومات مستقبلية عن تيار مرجع الصفحة. فعند حدوث خطأ الصفحة ولم تكن هناك إطارات حرة للاستخدام، تَفحص الخوارزمية تيار مرجع الصفحة، لمعرفة الرجوع المستقبلي للصفحات المتواجدة حاليًا في الذاكرة، بعدها تستخدم هذه المعرفة لاختيار الضحية، أي الصفحة الأمثل لكي تُحذف من الذاكرة.

مثلًا، بالتمعن في تيار مرجع الصفحة قد يكون هناك مجموعة من الصفحات داخل الذاكرة، منها من قد يُرجع إليها من قبل التعليمة القادمة (وهي الصفحة التي تحوي التعليمة الحالية)، ومنها من قد يُرجع إليها بعد تنفيذ 100 تعليمة، أو حتى بعد 500 أو 1000 تعليمة. عليه في أثناء حدوث خطأ الصفحة تُحذف الصفحات ذات الأرقام الكبيرة (وهي الصفحات الأمثل)، لأنها تُمثل في واقع الأمر الصفحات نادرة الاستخدام. أيضًا، إذا لم تُستخدم صفحة– ما– إلَّا بعد 5 مليون تعليمة وصفحة أخرى إلَّا بعد مليون تعليمة، فطرد الأولى من شأنه أن يؤجل خطأ الصفحة– في المستقبل– إلى أقصى حد ممكن.

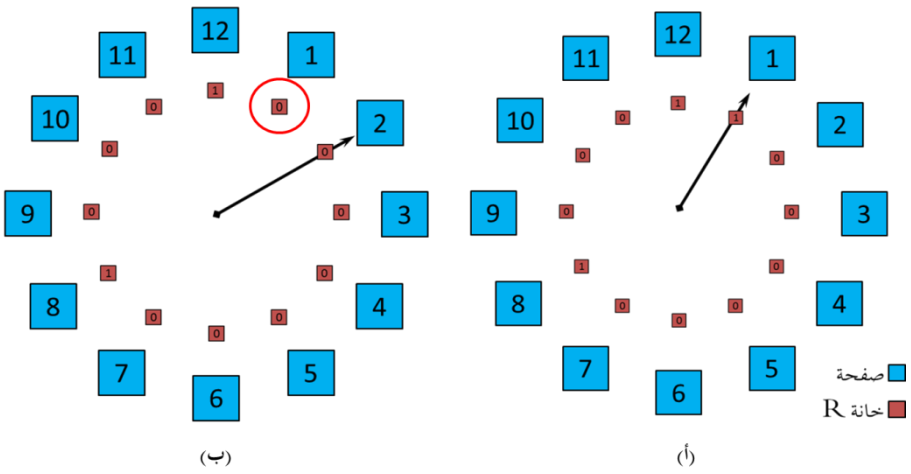
لسؤ الحظ، هذه الخوارزمية غير قابلة للتحقيق من الناحية العملية (أي صعوبة تنفيذها)، يرجع ذلك لعدم قدرة نظام التشغيل– في أثناء حدوث خطأ الصفحة– على إيجاد وسيلة لمعرفة المعلومات المستقبلية عن تيار مرجع الصفحة، أي أيٍ من الصفحات سيُرجع إليها تاليًا، هذه الحالة مشابهة لما شاهدنا مع خوارزمية أقصر وظيفة أولًا، وهي أنه كيف يُمكن للنظام معرفة

المعلومات المستقبلية الخاصة بأقصر الوظائف.

هناك مخرج من هذه الإشكالية تتمثل في تشغيل برنامج محاكاة لتتبع عمليات الرجوع إلى الصفحات في أثناء عملية التنفيذ والاستفادة منها في الجولة الثانية للتنفيذ باستخدام معلومات مرجعية الصفحة التي جُمِّعت خلال الجولة الأولى، وهو ما سيُتيح إمكانية تنفيذ خوارزمية الاستبدال الأمثل للصفحات في خلال الجولة الثانية. هذه العملية مفيدة في تقييم خوارزميات استبدال الصفحات، إلَّا إنَّها غير ذات جدوى في النظم العملية، بالتالي سندرس أدناه بعض من الخوارزميات المفيدة في الأنظمة الحقيقية.

4.7.5 خوارزمية الساعة لاستبدال الصفحات

على الرغم من أن خوارزمية الفرصة الثانية قد تفي بغرض الاستبدال، إلَّا إنَّه ليس من الضروري أن تكون فعَّالة بالشكل المطلوب، وذلك بسبب استمرارية تحريك الصفحات بشكل ثابت حول طابور الصفحات. كبديل أفضل يُمكن الاحتفاظ بجميع إطارات الصفحة على هيئة طابور دائري على شكل ساعة، كما هو مبين في الشكل 5. 29–أ، بحيث يُشير عقرب الساعة إلى أقدم الصفحات وهي في هذا الشكل الصفحة 1.



الشكل 5. 29: خوارزمية الساعة لاستبدال الصفحات.

عندما يحدث خطأ الصفحة، تُفحص الصفحة المشار إليها بعقرب الساعة. الإجراء المتبع

هنا يعتمد على الخانة **R**: إذا كانت قيمة خانة **R** الخاصة بهذه الصفحة تساوي **0**، ستُحذف الصفحة مباشرةً، وستُدرج صفحة جديدة في مكانها داخل الساعة، ومن ثم تتقدم عقرب الساعة بمقدار خطوة واحدة، أمَّا إذا كانت قيمة **R** تُساوي **1** فستُمسح **R** وتتقدم عقرب الساعة بمقدار خطوة واحدة ليُشير إلى الصفحة التالية، كما هو موضح في الشكل 5. 29–ب. تتكرر هذه العملية حتى يُعثر على صفحة تكون قيمة خانة **R** فيها تساوي **0**، لذلك ليس من المستغرب، تسمية هذه الخوارزمية بخوارزمية الساعة.

5.7.5 خوارزمية استبدال الصفحات غير المستخدمة مؤخرًا

تُضيف بعض من أنظمة الحواسيب وخصوصًا تلك التي تدعم الذاكرة الظاهرية لكل مدخل في جدول الصفحات خانتي حالة مرتبطتان بكل صفحة، لكي يستخدمهما نظام التشغيل لتجميع معلومات إحصائية مفيدة حول الصفحات من حيث الاستخدام، وذلك كما هو موضح في الشكل 5. 19. الخانة الأولى هي **R**، وتُضبط باستخدام الكيان المادي عند حدوث أي عملية رجوع (كتابة أو قراءة) للصفحة المرتبطة بها، أما الخانة الثانية فهي **M**، وتُضبط كذلك باستخدام الكيان المادي فقط عند حدوث أي عملية كتابة (أي تعديل على الصفحة) داخل الذاكرة. من المهم أن ندرك أن هاتين الخانتين يجب تحديثهما عند كل عملية رجوع إلى الذاكرة، لذلك من الضروري أن يُضبطا من قبل الكيان المادي، فإذا ما ضُبطت أيٍ منهما على **1**، فستبقى كذلك حتى يُعيد نظام التشغيل ضبطها من جديد.

كما أسلفنا سابقًا بأن هاتين الخانتين مفيدتان في جمع معلومات إحصائية تساعد كثيرًا في عملية استبدال الصفحات. بالتالي يُمكن تلخيص آلية اشتغال هذه الخوارزمية على النحو التالي: عند بدء اشتغال أي عملية، تُضبط كلتا الخانتين من قبل نظام التشغيل على صفر. دوريًا (مثلًا، عند كل مقاطعة نبضية) تُمسح الخانة **R** حتى تستطيع الخوارزمية التمييز ما بين الصفحات التي لم يُرجع إليها في الآونة الأخيرة من تلك التي أُستُخدمت مؤخرًا. عند حدوث خطأ الصفحة، يَفحص نظام التشغيل جميع الصفحات، ويُقسمها إلى أربع فئات، وذلك حسب القيم الحالية لكل من **R** و **M**، على النحو التالي:

- الفئة **0**: لم يُرجع إليها، ولم تُعدل – وهي أفضل فئة للحذف.

- الفئة 1: لم يُرجع إليها، وعُدلت – وهي ليست مفضلة بشكل كبير للحذف، لأنه يجب تحديث نسختها في القرص قبل عملية الحذف.
- الفئة 2: رُجع إليها، ولم تُعدل – احتمال أنها سوف تُستخدم قريبًا.
- الفئة 3: رُجع إليها، وعُدلت – احتمال أنها سوف تُستخدم قريبًا، وأنه يجب تحديث نسختها في القرص قبل عملية الحذف.

على الرغم من أن الفئة 1 للصفحات تبدو للوهلة الأولى أنها من المستحيل أن تتولد، إلَّا إنَّها تحدث عندما تُمسح خانة R لصفحات الفئة 3 من قبل المقاطعة النبضية، مع العلم بأن هذه المقاطعة لا تمسح خانة M، وذلك لاستخدامها في التعرف على الحاجة إلى إعادة كتابة الصفحة على القرص من عدمها. وبالتالي فمسح خانة R وترك خانة M يؤدي إلى الفئة 1 من الصفحات.

بعد ذلك تحذف هذه الخوارزمية، أو تختار صفحة عشوائية من داخل أصغر فئة (تكون غير خالية) من هذه الفئات. هذه الخوارزمية تضمن كذلك فكرة أنه من الأفضل إزالة الصفحة المعدَّلة التي لم يُرجع إليها على الأقل خلال ضربة نبضية واحدة، بدلًا من إزالة صفحة نظيفة جاري استخدامها بشكل مكثف. عوامل الجذب الرئيسية لهذه الخوارزمية هي أنه من السهل فهمها، وتتمتع بكفاءة متوسطة في التنفيذ، وتُعطي كذلك أداء قد يكون كافي، ولكنه بالتأكيد ليس الأمثل.

6.7.5 خوارزمية استبدال الصفحات المستخدمة مؤخرًا بقلة

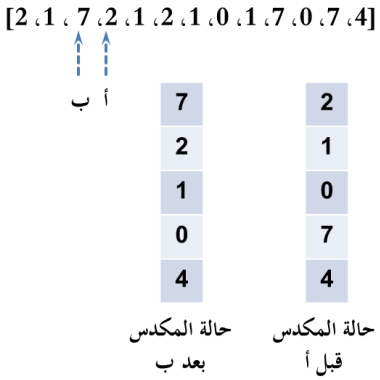
تتخلص خوارزمية استبدال الصفحات المستخدمة مؤخرًا بقلة من الصفحات التي لم تُستخدم مؤخرًا لفترة زمنية طويلة. فهي على العكس من خوارزمية الاستبدال الأمثل للصفحات في كونها تستفيذ من الإحصائيات الزمنية السابقة للصفحات لاختيار الضحية. الفكرة الأساسية لهذه الخوارزمية مبنية على أساس أن الصفحات التي أُستخدمت بكثافة من قبل مجموعة التعليمات القليلة الماضية، من المحتمل أن تُستخدم بشكل كبير مرة أخرى من قبل مجموعة التعليمات القادمة. على العكس من ذلك، فإن الصفحات التي لم تُستخدم منذ فترة زمنية طويلة، ربما تظل غير مستخدمة لفترة طويلة أخرى. عليه عندما يحدث خطأ الصفحة ولم تكن هناك إطارات حرة،

يتم التخلص من الصفحات التي لم تُستخدم مؤخرًا لفترة زمنية طويلة، أي المستخدمة مؤخرًا بقلة.

بالرغم من أن هذه الخوارزمية قابلة للتحقيق من الناحية النظرية وتُعتبر جيدة نوعًا ما، إلَّا إنَّها مكلفة وتُعاني من صعوبة التنفيذ. لأنه من أجل تنفيذ هذه الخوارزمية بالكامل، من الضروري الاحتفاظ بقائمة مرتبطة لكافة الصفحات في الذاكرة، بحيث تكون أكثر الصفحات المستخدمة مؤخرًا في مقدمتها، والأقل استخدامًا مؤخرًا في نهايتها، تكمن الصعوبة هنا في أن هذه القائمة يجب تحديثها عند كل عملية رجوع إلى الذاكرة. كذلك، العثور على الصفحة في القائمة وحذفها ومن ثم تحريكها إلى المقدمة هي عمليات تستغرق وقتًا طويلًا جدًا، حتى في حالة استخدام الكيان المادي. ومع ذلك، هناك طرق أخرى (منها ما هو مادي ومنها ما هو معنوي) يُمكن استخدامها لتنفيذ هذه الخوارزمية، نذكر منها ما يلي:

- العدَّادات: تُعتبر طريقة العدَّادت أبسط هذه الطرق وهي تتطلب دعم من الكيان المادي بعدَّاد بطول 64 خانة ثنائية يزداد تلقائيًا بعد تنفيذ كل تعليمة، كما يُزوَّد كل مدخل في جدول الصفحة بحقل كبير بما يكفي لاحتواء قيمة هذا العدَّاد. بعد كل عملية رجوع إلى الذاكرة، تُخزن قيمة العداد الحالية في الحقل الخاص بالصفحة التي رُجع إليها. بالتالي يفحص نظام التشغيل عند حدوث أي خطأ صفحة جميع العدَّادات المخزنة داخل جداول الصفحة، وذلك لغرض إيجاد أقل قيمة لها والتي ستناظر الصفحة الأقل استخدامًا والأكثر عرضة للحذف.
- المكدس: تتمثل الطريقة الثانية في تنفيذ هذه الخوارزمية في استخدام المكدس أحد هياكل البيانات والذي يحوي أرقام الصفحات، عند الرجوع إلى أي صفحة يُحذف رقمها من المكدس ويُضاف إلى قمته. في هذه الحالة، ستكون آخر أكثر صفحة استخدامًا في قمة المكدس، بينما ستكون الصفحة المستخدمة مؤخرًا بقلة دائمًا في قاعه، كما هو موضح في الشكل 5. 30.

من مساوي هذه الطريقة هو ضرورة تحريك محتوى المكدس، لذلك هناك توجه لتنفيذها باستخدام القوائم المرتبطة المزدوجة مع وجود مؤشر للمقدمة ومؤشر لذيل القائمة.



الشكل 5. 30: استخدام المكدس لتحديد الصفحة المستخدمة مؤخرًا بقلة.

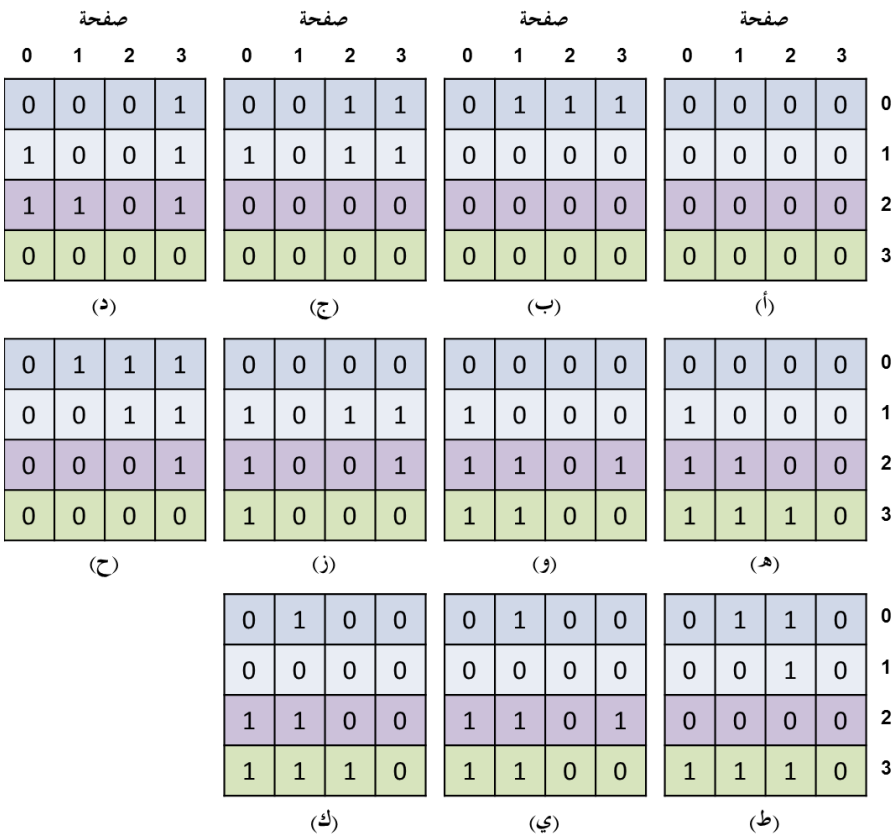
- المصفوفات: تعتمد الطريقة الثالثة لتطبيق هذه الخوارزمية كذلك على الكيان المادي، وفيها تُستخدم مصفوفة ذات بعدين بحجم N*N، حيث تُمثل N عدد إطارات الصفحة المستخدمة. مبدئيًا تحمل جميع خانات هذه المصفوفة القيمة صفر، وفي أثناء الرجوع إلى الصفحة رقم k يضبط نظام التشغيل (باستخدام الكيان المادي) جميع خانات الصف k على 1 وجميع خانات العمود k على 0. في اللحظة التي يحدث فيها خطأ الصفحة ولم تكن هناك إطارات حرة، تُحسب القيم الثنائية للصفوف، والصف الأقل قيمة سوف يُناظر الصفحة الأقل استخدامًا وهكذا.

مثال توضيحي: يوضح الشكل 5. 31 فكرة عمل هذه الخوارزمية من خلال التعامل مع برنامج يتكون من الصفحات الظاهرية 0 1 2 3، التي سيُرجع إليها على حسب تيار مرجع الصفحة التالي:

[0، 1، 2، 3، 2، 1، 0، 3، 2، 3].

الشكل 5. 31–أ يوضح الحالة الاستهلالية للمصفوفة، بينما يوضح الشكل 5. 31–ب الحالة التي تم الرجوع فيها إلى الصفحة 0. كذلك وبعد ما رُجع إلى الصفحة 1، أصبح لدينا الحالة المبينة في الشكل 5. 31–ج، وهكذا مع بقية الأشكال إلى أن وصلنا الحالة المعطية في الشكل 5. 31–ك. والآن بفرض أنه حدث خطأ صفحة وكانت جميع الإطارات مستغلة فما هي الصفحة التي ستُحذف؟ بناءً على آلية عمل الخوارزمية ستُحسب القيم الثنائية للصفوف، والصفحة المناظرة لأقل قيمة ستكون هي الصفحة الضحية وهي في هذه

الحالة الصفحة رقم 1.



الشكل 5. 31: مثال توضيحي لاستراتيجية الصفحات المستخدمة مؤخرًا بقلة باستخدام المصفوفات.

7.7.5 محاكاة استبدال الصفحات المستخدمة مؤخرًا بقلة باستخدام البرمجيات

على الرغم من أن خوارزميتي استبدال الصفحات السابقتين هما (من حيث المبدأ) قابلتان للتحقيق، إلَّا إنَّه هناك قلة من الآلات الداعمة للاحتياجات المطلوبة لتنفيذهما. عليه وبدلًا من ذلك، هناك حاجة إلى حل يُمكن تنفيذه في مجال البرمجيات. أحد هذه الحلول يُسمى بخوارزمية الصفحات غير المستخدمة بشكل متكرر والذي يتطلب عدَّاد معنوي مرتبط بكل صفحة قيمته البدائية صفر يُستخدم لتتبع عدد المرات التي رُجع فيها لكل صفحة. عند كل مقاطعة نبضية،

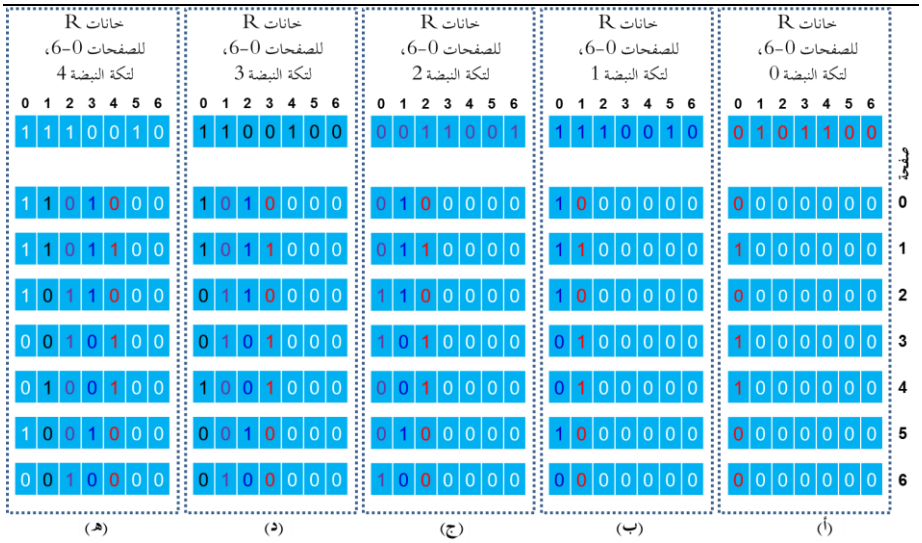
يمسح نظام التشغيل جميع الصفحات في الذاكرة ويُضيف قيمة خانة R– التي إما 0 أو 1– إلى عدَّاد الصفحة. عندما يحدث خطأ الصفحة وكانت جميع إطارات الصفحة محجوزة، تُختار الصفحة التي تحتوي على أقل قيمة للعدَّاد، وذلك لغرض استبدالها.

تُعاني طريقة عمل هذه الخوارزمية من مشكلة عدم النسيان، بمعنى تأثر الإحصائيات الحديثة بالإحصئيات القديمة للصفحات. مثلًا، في الحالات التي تتم فيها عملية الترجمة على عدة مراحل (تمريرات)، ربما تُؤثر الصفحات التي أُستخدمت بكثافة في أثناء التمريرة الأولى على التمريرات اللاحقة، لكونها لا تزال تحتفظ بقيم عالية للعدَّادات، وخصوصًا إذا كان زمن تنفيذ التمريرة الأولى أطول من زمن بقية التمريرات. هذا الأمر يجعل قيمة عدَّادات الصفحات الحاوية على شفرة التمريرات اللاحقة دائمًا أقل من قيمة عدَّادات صفحات التمريرة الأولى، نتيجةً لذلك قد يحذف نظام التشغيل صفحات مفيدة، بدلًا من صفحات لم تعد مؤخرًا قيد الاستخدام.

للخروج من المعظلة سالفة الذكر، يُمكن إجراء تعديل بسيط على هذه الخوارزمية يجعلها قادرة على محاكاة خوارزمية استبدال الصفحات المستخدمة بقلة مؤخرًا بشكل جيد. يتكون هذا التعديل من جزئين:

أولًا: تُزاح العدَّادات بمقدار خانة واحدة إلى اليمين قبل إضافة قيمة R إليها.
ثانيًا: تُضاف قيمة R إلى الخانات في أقصى اليسار بدلًا من في أقصى اليمين.

مثال توضيحي: يبين الشكل 5. 32 مثالًا لكيفية عمل الخوارزمية المعدَّلة والمعروفة باسم خوارزمية الشيخوخة. يفترض هذا المثال أنه بعد ضربة النبضة الأولى كانت خانات R للصفحات 0 إلى 6 تحتوي على القيم 0، 1، 0، 1، 1، 0، 0 على التوالي (خانة R للصفحة 0 تحمل 0، وللصفحة 1 تحمل 1، وللصفحة 2 تحمل 0، وهكذا). بعبارة أخرى، بين ضربة النبضة 0 والنبضة 1، تم الرجوع إلى الصفحات 1، 3، و 4، وهو ما جعل الخانة الخاصة بكل منها تُضبط على1، في حين لا تزال بقية الخانات الأخرى 0. بعد أن تُزاح العدَّادات السبع المناظرة للصفحات وتُضاف قيم الخانة R من جهة اليسار، ينتج لنا القيم المبينة في الشكل 5. 32–أ. الأعمدة الأربعة المتبقية تُبين لنا حالة هذه العدَّادات بعد ضربات النبضات الأربع المقبلة.



الشكل 5. 32: خوارزمية الشيخوخة لمحاكاة خوارزمية الصفحات غير المستخدمة بشكل متكرر. الموضح هو سبع صفحات على مدار خمس ضربات نبضية والممثلة من أ إلى هـ.

عندما يحدث خطأ الصفحة وكانت كافة الإطارات مشغولة، تُزال الصفحة المناظرة لأقل قيمة عدَّاد. مثلًا من الواضح أن الصفحة التي لم يُرجع إليها لمدة أربع ضربات نبضية، سيحتوى العدَّاد المناظر لها على أربعة أصفار من ناحية اليسار، مما يجعل قيمته أقل من قيمة العدَّاد المناظر للصفحة التي لم يُرجع إليها لمدة ثلاث ضربات نبضية، والذي سيحتوي على ثلاثة أصفار من ناحية اليسار.

هذه الخوارزمية تختلف عن الخوارزمية السابقة من ناحيتين. بالنظر إلى الصفحتين 3 و 6 في الشكل 5. 32–هـ، نلاحظ أنه لم يُرجع إليهما لمدة ضرباتان نبضيتان، ولكن رُجِع إليهما في الضربة السابقة لذلك. وفقًا لقواعد الخوارزمية السابقة، إذا كان من الضروري استبدال صفحة، فيجب اختيار إحدى هاتين الصفحتين، المشكلة هنا أننا لا نعرف من منهما رُجِع إليها مؤخرًا في الفترة الفاصلة بين الضربة 1 والضربة 2. يُلاحظ أنه عن طريق تسجيل خانة ثنائية واحدة فقط لكل فترة زمنية، سنفقد القدرة على التمييز بين عمليات الرجوع التي حدثت في وقت مبكر خلال فترة النبضة عن تلك التي تحدث مؤخرًا. كل ما يُمكننا القيام به هنا هو إزالة الصفحة 6، لأن الصفحة 3 رُجِع إليها كذلك في وقت سابق قبل ضربتين، بينما لم يحدث ذلك للصفحة 6.

الفرق الثاني بين الخوارزمية السابقة وخوارزمية الشيخوخة هو أن العدَّادات في الخوارزمية الأخيرة لديها عدد محدود من الخانات الثنائية (6 في هذا المثال) مما يحد من استخداماتها في الأفق الماضي. مثلًا، لنفترض أن هناك صفحتين لكل واحدة منهما عدَّاد قيمته 0، كل ما يُمكننا القيام به هو اختيار واحدة منهما عشوائيًا. في الواقع، قد تكون واحدة منهما رُجِع إليها في آخر سبع ضربات ماضية، بيمنا الأخرى في آخر 500 ضربة. لذلك ليس لدينا وسيلة في هذه الطريقة لرؤية هذا الفرق، ولكن في الممارسة العملية بشكل عام، استخدام ثمان خانات ثنائية كافية إذا كانت ضربة النبضة في حدود 20 ملي ثانية، لأنه إذا لم يُرجع إلى أي صفحة في حدود 160 ملِّي ثانية، فهذا يعني أنها ليست بتلك الأهمية.

8.7.5 خوارزمية صفحات مجموعة العمل لاستبدال الصفحات

عندما تبدأ أي عملية في التنفيذ لأول مرة تكون الذاكرة الفعلية خالية من أي صفحة لها، لذلك عند محاولة وحدة المعالجة المركزية جلب التعليمة الأولى، سيحدث خطأ الصفحة، وسيجعل هذا الأمر نظام التشغيل يجلب الصفحة التي تحتوي على هذه التعليمة، بعدها تتوالى بسرعة بقية أخطاء الصفحات الأخرى الخاصة بكل من المتغيرات الخارجية والمكدس. بعد فترة من الزمن، سيكون لدى العملية معظم الصفحات التي تحتاجها وستبدأ في التنفيذ مع حدوث عدد قليل– نسبيًا– من أخطاء الصفحات. هذه الإستراتيجية تُسمى بالتصفح عن طريق الطلب، لأن الصفحات تُحمَّل فقط عند الطلب، وليس في وقت مبكر.

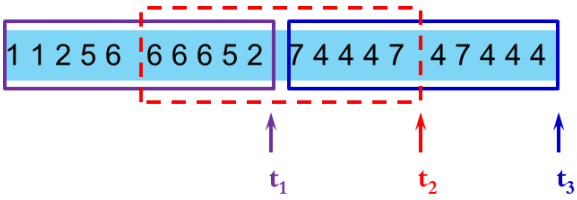
لحسن الحظ، تُظهر معظم العمليات نوع من المرجعية المحلية، وهو ما يعني أنه في أثناء أي مرحلة من مراحل التنفيذ، ترجع العملية فقط إلى جزء صغير نسبيًا من صفحاتها، بالتالي لو تواجد هذا الجزء بأكمله في الذاكرة، لأمكن تشغيل هذه العملية من دون أن تتسبب العملية في أخطاء كثيرة، على الأقل إلى أن ينتقل التنفيذ إلى مرحلة أخرى (مثلًا، التمريرة المقبلة للمترجم).

أطلق بيتر دينينج (**Denning, 1968a; Denning, 1980**) على هذا الجزء الصغير من الصفحات اسم مجموعة العمل الخاصة بالعملية، والتي تُمثل الحد الأدنى لمجموعة الصفحات التي من المتوقع تحميلها في الذاكرة الفعلية حتى تعمل العملية بكفاءة وتستخدمها خلال فترة زمنية معينة، لكي تتجنب أخطاء الصفحات غير الضرورية وجعلها تنخفض بشكل كبير،

يُمكن تعريف مجموعة العمل هذه بشكل تفصيلي على النحو التالي:

مجموعة العمل $ws(t, w)$ عند الزمن t: هي تلك الصفحات التي رُجِع إليها من قبل عملية– ما– في الفترة الزمنية $(t, t-w)$، حيث تُمثل t الزمن و w نافذة مجموعة العمل، بحيث أي صفحة تنتمي إلى ws يجب أن تُحقق شرط الانتماء إلى هذه النافذة، بمعنى أن تكون الصفحة ضمن الصفحات التي رُجِع إليها خلال آخر نافذة، مع العلم بأن مجموعة العمل هذه تتغير على طول فترة حياة العملية.

مثال توضيحي: بفرض أن $w = 10$ وأن تيار مرجع الصفحة مُعطى بالسلسلة المرجعية التالية:



هذا التيار موضح عليه ثلاث نوافذ عند ثلاثة أزمنة متباينة t_1، و t_2، و t_3، بالتالي فإن مجموعات عمل العملية عند هذه الأزمنة هي:

$$ws(t_1) = \{1,2,5,6\}$$
$$ws(t_2) = \{6,5,2,7,4\}$$
$$ws(t_3) = \{7,4\}$$

في هذا السياق يُقاس الزمن بوحدات الرجوع إلى الذاكرة وبشكل منفصل لكل عملية، بالتالي $t = 1024$ تعني حقًا الزمن الذي رَجعت فيه هذه العملية إلى الذاكرة رقم 1024. كما يُشير المصطلح 'صفحة' في هذا التعريف إلى الصفحات الفعلية، وليس إلى الصفحات الظاهرية، لأن الصفحات الفعلية الموجودة في الذاكرة الرئيسية ولو لمرة واحدة على الأقل خلال تلك الفترة الزمنية هي جزء من مجموعة العمل لهذه العملية.

هنا يجب الإشارة إلى أن مفهوم مجموعة العمل يتطلب ضمنيًا من العملية تحديد الصفحات التي تحتاجها حتى تُنفذ العملية من دون أن تحدث الكثير من أخطاء الصفحات، وأنْ تتواجد هذه

الصفحات في الذاكرة الرئيسية قبل البدء في تنفيذها. ولكن، هل باستطاعة هذه العملية معرفة عناصر مجموعة عملها لكي تُحمل في الذاكرة قبل التنفيذ؟ للوهلة الأولى قد يبدو هذا مستحيلًا، لأنه يتطلب التنبؤ بالمستقبل، إلَّا إنَّ الفكرة تكمن هنا في استخدام الاحتياجات الحديثة لعملية– ما– للتنبؤ باحتياجاتها المستقبلية، بمعنى أن المستقبل القريب يُقرَّب بشكل جيد من خلال الماضي القريب.

بالعودة إلى فكرة عمل خوارزمية استبدال الصفحة استنادًا على مجموعة العمل، نجد أنها تتمثل في البحث عن صفحة غير موجودة في مجموعة عمل العملية، ومن ثم حذفها. يُوضح الشكل 5. 33 جزء من جدول صفحة لآلة– ما–. في هذه الخوارزمية، ستكون الصفحات الوحيدة الموجودة في الذاكرة هي المرشحة للحذف، بينما ستُتجاهل الصفحات الغائبة عنها. يحتوي كل مدخل (على الأقل) عنصرين رئيسين من المعلومات: الزمن التقريبي لآخر استخدام للصفحة، وخانة الرجوع **R**.

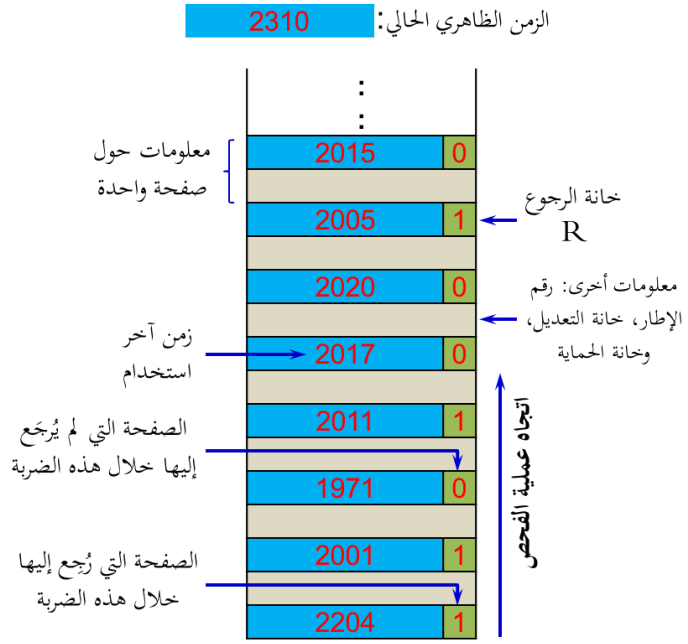
تعمل الخوارزمية على النحو التالي: من المفترض أن الكيان المادي يضبط الخانتين **R** و **M**، وذلك كما هو موضح في السابق، بالمثل، تتسبب المقاطعة النبضية الدورية في جعل البرمجيات تشتغل وتمسح خانة **R** عند كل ضربة نبضية، كذلك يُفحص جدول الصفحة للبحث عن صفحة مناسبة للحذف عند كل خطأ صفحة.

تُفحص جميع الصفحات عند حدوث خطأ الصفحة، ويُعالج كل مدخل لفحص خانة **R** على النحو التالي:

- إذا كانت **R** تساوي 1، يُكتب الوقت الظاهري الحالي في حقل زمن آخر استخدام داخل جدول الصفحة، للدلالة على أن الصفحة كانت قيد الاستخدام في وقت حدوث الخطأ، بمأن الصفحة رُجِع إليها خلال ضربة النبضة الحالية، فمن الواضح أنها تنتمي إلى مجموعة العمل، بالتالي فهي ليست مرشحة للحذف.
- إذا كانت **R** تساوي 0، فهذا يعني أن الصفحة لم يُرجَع إليها خلال ضربة النبضة الحالية وربما ستكون مرشحة للحذف. بالتالي لمعرفة ما إذا كان ينبغي حذف هذه الصفحة أم لا، يُحتسب أولًا عمرها:

عمر الصفحة = الوقت الظاهري الحالي – زمن آخر استخدام

ومن ثم يُقارن عمر الصفحة بالزمن الماضي من الزمن الظاهري τ.



جدول الصفحة

الشكل 5. 33: توضيح فكرة خوارزمية صفحات مجموعة العمل لاستبدال الصفحات.

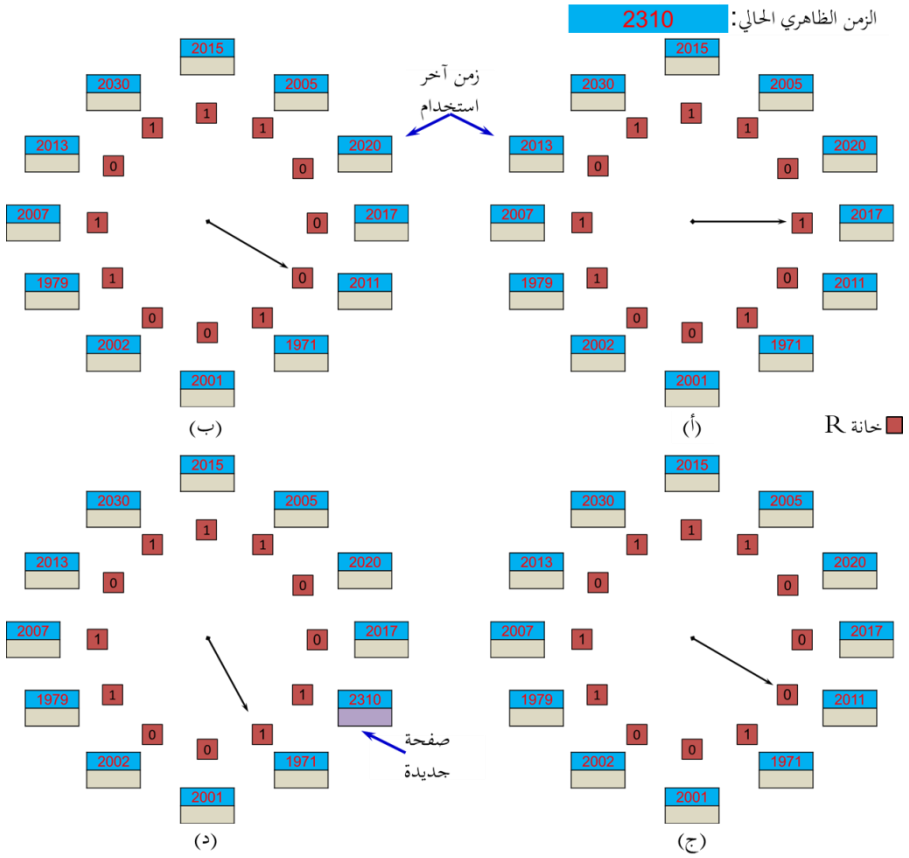
- إذا كان العمر أكبر من τ، فهذا يعني أن الصفحة لم تعد تنتمي إلى مجموعة العمل وستُستبدل بصفحة جديدة، بعدها يستمر مسح الصفحات المتبقية وتحديث المداخل.
- إذا كان العمر أقل من أو يساوي τ، فهذا يعني أن الصفحة لا تزال ضمن مجموعة العمل، إلَّا إنَّها بمنأى عن الحذف مؤقتًا، وستُلاحظ الصفحة ذات أكبر عمر، أي الحاملة لأصغر قيمة لزمن آخر استخدام.
- أخيرًا، إذا مُسح الجدول بأكمله دون العثور على مرشح للطرد، فهذا يعني أن كل الصفحات ضمن مجموعة العمل. في هذه الحالة، إذا عُثر على صفحة واحدة أو أكثر وكانت قيمة **R** تساوي **0**، ستُطرد الصفحة ذات العمر الأكبر. أمَّا إذا رُجِع إلى كل الصفحات خلال ضربة النبضة الحالية وهو أسوأ الحالات (بالتالي ستكون **R** لجميع الصفحات تساوي **1**)، سيختار النظام عشوائياً صفحة لحذفها، إلَّا إنَّه يُفضل أن تكون صفحة نظيفة إن وُجِدت.

9.7.5 خوارزمية ساعة مجموعة العمل لاستبدال الصفحات

لاحظنا من خلال تتبع آلية تنفيذ خوارزمية مجموعة العمل أنه من الضروري تَفحص جدول الصفحة بالكامل عند حدوث أي خطأ صفحة، من أجل تحديد مرشح مناسب للحذف وهو ما يُرهق عمل هذه الخوارزمية. في سنة 1981 أدخل كل من كار وهينيسي (Carr and Hennessey, 1981) تحسينات على هذه الخوارزمية لتفادي هذا العيب. يتمثل هذا التعديل في الجمع بين فكرة خوارزمية مجموعة العمل وبين مفهوم خوارزمية الساعة، عُرفت بخوارزمية ساعة مجموعة العمل، ونظرًا لبساطة التنفيذ والأداء الجيد للخوارزمية الجديدة، أُستخدمت على نطاق واسع في الممارسات العملية.

تُربط إطارات الصفحة في هذه الخوارزمية في شكل قائمة دائرية كما هو الحال في خوارزمية الساعة، وكما هو مبين في الشكل 5. 34–أ. كل مدخل من مداخل هذه القائمة يحتوي على حقل زمن آخر استخدام والمأخوذ من خوارزمية مجموعة العمل الأساسية، فضلًا عن خانة الرجوع R (موضحة في الشكل) وخانات أخرى غير ظاهرة فيه. تُضبط R بواسطة الكيان المادي بنفس الآلية المستخدمة في خوارزمية استبدال الصفحات غير المستخدمة مؤخرًا. تكون في البداية هذه القائمة فارغة، وعند تحميل الصفحة الأولى، تُضاف إلى القائمة، وبإضافة المزيد من الصفحات، تبدأ الساعة في التكون.

كما هو الحال مع خوارزمية الساعة، عند حدوث أي خطأ صفحة ولم تكن هناك إطارات حرة، تُختبر أولاً الصفحة المشار إليها بعقرب الساعة. إذا كانت خانة الرجوع R تساوي 1، فستكون هذه الصفحة مستخدمة خلال الضربة النبضية الحالية، بالتالي لن تكون مرشحًا مثاليًا للحذف، بعدها تُضبط R على 0 ويتحرك عقرب الساعة إلى الأمام ليُشير إلى الصفحة التالية، وتتكرر نفس الخطوات لهذه الصفحة. يُظهر الشكل 5. 34–ب سلسلة الأحداث هذه.



الشكل 5. 34: خوارزمية ساعة مجموعة العمل– أ) و ب) يعطيان مثال توضيحي عن ما يحدث عندما R تساوي 1– بينما ج) و د) يعطيان مثال توضيحي عن ما يحدث عندما R تساوي 0.

قد يظن المرء أن هذه الخوارزمية ستستمر تقريبًا بنفس استراتيجية استبدال الصفحات المستخدمة مؤخرًا بقلة وذلك باختيار الصفحة ذات الزمن الأقل لكي تُستبدل، ولكن بدلًا من ذلك، تعتمد هذه الخوارزمية على استخدام معلمة ثابتة هي τ والتي يضبطها مصمم نظام التشغيل بحيث تتلائم إلى حد ما مع الممارسة العملية. بالتالي عندما تكون قيمة خانة R للصفحة المشار إليها تساوي 0، كما هو الحال في الشكل 5. 34–ج، وكان عمر الصفحة أكبر من τ وكانت نظيفة (أي غير معدَّلة)، فهذا يعني أنها، أي الصفحة لا تنتمي إلى مجموعة العمل وأن نسخة منها لاتزال صالحة على القرص، بالتالي سيُحدد إطارها ويُوضع صفحةً جديدةً به، كما هو مبين في

الشكل 5. 34-د. تتمثل ميزة القيام بهذا في أنه ليس من الضروري البحث في جميع الصفحات النشطة للعثور على أكبر الصفحات عمرًا، بل من الممكن التوقف عندما تكون هناك صفحة عمرها أكبر سنًا من τ. إذا لم يتحقق ذلك، فستُختار الصفحة ذات الحقل الزمني الأقدم للاستبدال.

في أغلب الأحيان يُفضل استبدال صفحة نظيفة بدلًا من صفحة معدَّلة، لأن الصفحات النظيفة لا يجب تحديثها على القرص، لذلك يُفضل البحث حتى تتوفر صفحة نظيفة عمرها أقدم من τ. إذا لم يكن الأمر كذلك، فستُستخدم صفحة معدَّلة عمرها أقدم من τ، ثم يستمر البحث من حيث توقف مؤشر الساعة، مع الأخذ في عين الاعتبار أن جدولة عملية التحديث على القرص يجب أن تتم في نفس الوقت الذي يتحرك فيه عقرب الساعة إلى الأمام، وذلك لاستغلال الوقت بشكل أمثل، ومن ثم تحسين الكفاءة. من حيث المبدأ، يُمكن جدولة جميع الصفحات للمبادلة مع القرص في دورة واحدة على مدار الساعة، ولكن قد يتطلب الأمر وضع حد محدد يسمح بالكتابة العكسية لعدد n من الصفحات كحد أقصى، وإذا ما أُجتُيز هذا الحد، فلن تُجدول أي كتابات جديدة، وذلك لغرض تقليل الازدحام على القرص.

السؤال الذي يطرح نفسه الآن: ماذا سيحدث إذا رجع عقرب الساعة إلى نقطة البداية من جديد؟ إذا حدث ذلك، فهذا يعني أنه قد حدث أحد الاحتمالين التاليين:

1. جُدولت على الأقل كتابة واحدة على مدار الدورة.
2. لم تُجدول أي كتابة على مدار الدورة.

يبقي عقرب الساعة في الحالة الأولى يتحرك للبحث عن صفحة نظيفة، ولأنه قد جُدولت عملية كتابة واحدة أو أكثر، ستكتمل في نهاية المطاف إحدى هذه الكتابات وستُعلم صفحتها على أساس نظيفة، بالتالي ستُحذف أول صفحة نظيفة عُلِّمت عندما تواجهها الخوارزمية من جديد. هذه الصفحة ليست بالضرورة أن تكون ذات علاقة بأول عملية كتابة تمت جدولتها، لأن مشغل القرص قد يُعيد ترتيب الكتابات من أجل تحسين أداء القرص.

في الحالة الثانية، إذا إنتهت الدورة ولم تُجدول أي كتابة، فهذا يعني أن كل الصفحات تنتمي إلى مجموعة العمل، ولكن وبالرغم من الافتقار إلى معلومات مساعدة، أبسط شيء يُمكن

فِعله: هو المطالبة بأي صفحة نظيفة واستخدامها، إذا لم يتوفر ذلك، تُختار الصفحة الحالية كضحية وتُكتب مرة أخرى في القرص.

10.7.5 ملخص ومقارنة لخوارزميات استبدال الصفحة

تُستخدم خوارزميات استبدال الصفحة لمنع التخصيص الزائد للذاكرة عن طريق تعديل برنامج خدمة خطأ الصفحة لتشمل استبدال الصفحة، تستخدم هذه الخوارزميات بيانات إحصائية حول الصفحة (لتتبع عملية الرجوع إلى الصفحة وتعديلها)، وذلك لغرض التقليل من مبادلة الصفحات المهمة ما بين الذاكرة الفعلية والقرص. إن استخدام خوارزميات استبدال الصفحة يُمكِّن من توفير ذاكرة ظاهرية كبيرة بمساعدة ذاكرة فعلية أصغر، إذا حدث خطأ صفحة وكان هناك إطار صفحة غير مستخدم في الذاكرة الفعلية، فستُجلب الصفحة المطلوبة وتُحمل فيه، إذا لم يكن هناك إطار حر، فإن مهمة خوارزمية استبدال الصفحة هو البحث عن ضحية لاستبعادها وفتح المجال أمام الصفحة الجديدة لكي تُحمل بدلًا من الضحية. تتفاوت هذه الخوارزميات في آلية اختيار الضحية، إلَّا إنَّ أفضلها الخوارزمية التي تُحقق أدنى معدل خطأ صفحة. يستعرض بقية هذا القسم ملخص لهذه الخوارزميات، في حين يستعرض الجدول 5. 5 مقارنة موجزة لها.

- تحتفظ خوارزمية الداخل أولًا يخرج أولًا لاستبدال الصفحات بترتيب الصفحات تبعًا لزمن التحميل في الذاكرة من خلال استخدامها لقائمة مرتبطة، وهو ما يُسهل التعرف على أقدم الصفحات، بالتالي إزالتها، ولكن تلك الصفحة قد تكون لا تزال قيد الاستخدام، وهو ما سيُعتبر اختيار سيئ.
- خوارزمية الفرصة الثانية لاستبدال الصفحات ما هي إلا نسخة معدَّلة من خوارزمية الداخل أولًا يخرج أولًا لاستبدال الصفحات لمعالجة الاختيار السيئ للصفحة الضحية. تتحقق هذه الخوارزمية من ما إذا كانت الصفحة لازلت قيد الاستخدام قبل إزالتها، إذا كانت كذلك، فسيُحتفظ بها، وإلَّا ستُستبعد. هذا التعديل أدخل تحسينًا كبيرًا على أداء الخوارزمية. من ناحية أخرى، خوارزمية الساعة هي مجرد تنفيذ مختلف من خوارزمية الفرصة الثانية ولديها نفس خصائص الأداء، لكنها تحتاج إلى وقتًا أقل في التنفيذ وهو أمر جيد.
- تحذف خوارزمية استبدال الصفحة الأمثل الصفحة التي من المتوقع الرجوع إليها في المستقبل البعيد. ولكن وللأسف، هناك صعوبة كبيرة في إيجاد وسيلة لتحديد هذه الصفحة.

بالتالي في الممارسة العملية، لا يُمكن استخدام هذه الخوارزمية، إلَّا إنَّها مفيدة من كونها تُمثل مرجع يُمكن قياس كفاءة الخوارزميات الأخرى بها.

- تعتمد خوارزمية استبدال الصفحات غير المستخدمة مؤخرًا على استخدام خانتي **R** و **M** في تقسيم الصفحات إلى أربع فئات تبعًا لحالة كل منها، بعدها تختار صفحة عشوائية من أدنى فئة مرقمة. هذه الخوارزمية سهلة التنفيذ، وبسيطة جدًا، لكن هناك خوارزميات أخرى أفضل منها.
- تُعتبر خوارزمية استبدال الصفحات المستخدمة مؤخرًا بقلة من الخوارزميات الممتازة، ولكن لا يُمكن تنفيذها إلَّا بواسطة كيان مادي خاص، وفي حالة عدم توفره سيكون من الصعب استخدامها. بالمقابل خوارزمية الصفحات غير المستخدمة بشكل متكرر هي محاولة لتقريب الخوارزمية السابقة، وهي ليست بالمحاولة الجيدة، إلَّا إنَّ تقريب خوارزمية الشيخوخة أفضل بكثير، كما يُمكن تنفيذها بكفاءة، بل هي خيار جيد.
- تستخدم آخر خوارزميتان مجموعة العمل التي أعطت أداءً معقولًا، لكنهما مكلفتان بعض الشيء من حيث التنفيذ، إلَّا إنَّ خوارزمية ساعة مجموعة العمل لا تعطي أداء جيد فقط، بل تتمتع أيضًا بكفاءة في التنفيذ.

الخلاصة، أفضل خوارزميتان هما الشيخوخة، وخوارزمية ساعة مجموعة العمل، لأنهما تستندان إلى خوارزمية استبدال الصفحات المستخدمة مؤخرًا بقلة ومجموعة العمل، على التوالي. كلتاهما تعطي أداء جيد في التصفح، كما يُمكن تنفيذها بكفاءة. هناك عدد قليل من الخوارزميات الجيدة الأخرى، ولكن هاتين هما على الأرجح الأكثر أهمية من الناحية العملية.

الجدول 5. 5: مقارنة خوارزميات استبدال الصفحة.

الخوارزمية	المزايا	العيوب
الداخل أولًا، يخرج أولًا	– سهلة الفهم. – سهلة التنفيذ. – تستخدم هياكل بيانات معقدة قليلًا.	– ليست فعَّالة بالقدر الجيد. – يحتاج النظام إلى تتبع كل إطار. – تُعاني من شذوذ بلادي. – ربما تطرد صفحات مهمة.
الفرصة الثانية	– أدخلت تحسين أفضل في الأداء.	– تستخدم هياكل بيانات معقدة.

الخوارزمية	المزايا	العيوب
الأمثل	– أدنى معدل خطأ صفحة. – لا تُعاني البتة من شذوذ بلادي. – تستخدم هياكل بيانات بسيطة. – مفيدة كمرجع.	– من الصعب تنفيذها. – تحتاج إلى التنبؤ، أي المعرفة المستقبلية.
الساعة	– تحتاج إلى وقتًا أقل في التنفيذ. – واقعية (قابلة للتحقيق).	– تستخدم هياكل بيانات معقدة.
غير المستخدمة مؤخرًا	– سهلة التنفيذ. – بسيطة للغاية.	– يتطلب تنفيذها مساعدة الأجهزة المادية.
المستخدمة بقلة مؤخرًا	– نوعًا ما كفؤة. – من السهل إجراء التحليل الإحصائي الكامل. – لا تعاني البتة من شذوذ بلادي.	– تنفيذها ليس بالأمر السهل. – يتطلب تنفيذها مساعدة الأجهزة المادية.
المستخدمة بشكل متكرر	– إلى حد ما تقريب بسيط لخوارزمية المستخدم بقلة مؤخرًا.	– إلى حد ما مكلفة في التنفيذ.
الشيخوخة	– فعَّالة تُقرِّب خوارزمية المستخدم بقلة مؤخرًا بشكل جيد. – تعطي أداءً جيدًا في التصفح. – يُمكن تنفيذها بكفاءة.	– إلى حد ما مكلفة في التنفيذ.
مجموعة العمل	– تعطي أداءً نسبيًا جيد.	– إلى حد ما مكلفة في التنفيذ. – تحتاج إلى المعرفة المستقبلية. – كل عملية تحتاج إلى مساحة إضافية لحفظ مجموعة العمل. – تستخدم هياكل بيانات معقدة.
ساعة مجموعة العمل	– تعطي أداءً جيدًا. – ذات كفاءة جيدة في التنفيذ.	– مكلفة بعض الشيء من حيث التنفيذ. – تحتاج إلى المعرفة المستقبلية. – تستخدم هياكل بيانات معقدة.

8.5 التقطيع

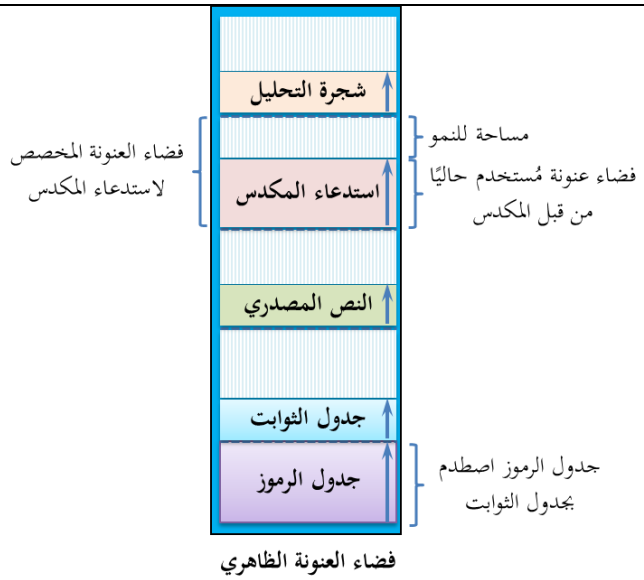
من خلال تتبع آلية عمل الذاكرة الظاهرية (راجع القسم 6.5) سيُلاحظ المرء أن رؤية كل من المستخدم والمبرمج لها ليست مماثلة لواقع الذاكرة الفعلية. فهما يرونها أنها مجموعة من الكائنات المختلفة في الحجم، وليس كمصفوفة خطية للخانات الثمانية أو كصفحات متساوية الحجم. هذا الأمر يؤثر سلبًا على واقع التعامل مع الذاكرة من حيث خصائصها المادية لكل من نظام التشغيل والمبرمج. لذلك اتجهت الأنظار إلى البحث عن آلية يستطيع من خلالها الكيان المادي توفير طريقة لإدارة الذاكرة الفعلية بشكل تتلاءم فيه مع وجهة نظر المبرمج.

مع توفر هذه الآلية سيكون للنظام حرية أكبر في إدارة الذاكرة، بينما يتمتع المبرمج ببيئة برمجة أكثر ارتياحية. أيضًا، إنَّ استخدام التصفح كأسلوب لإدارة الذاكرة هو أقرب إلى نظام التشغيل منه إلى المستخدم. كما أنه يُقلل من كفاءة النظام، لأنَّ بإمكانه تقسيم نفس الوظيفة إلى صفحات مختلفة قد تُحمَّل أو لا تُحمَّل في الذاكرة في نفس الوقت.

تُعتبر الذاكرة الظاهرية كذلك أحادية البعد، لأن العناوين الظاهرية تتراوح قيمها ما بين صفر وقيمة الحد الأقصى للعنوان، أي أنَّ العناوين تتسلسل واحد بعد الآخر. هذا الأمر لا يتناسب مع عدة تطبيقات وخصوصًا تلك التي تتطلب العديد من العناوين الظاهرية والمفصولة عن بعضها البعض. المترجمات هي خير مثال لذلك، فهي تحتاج إلى العديد من الجداول والتي تُستحدث في أثناء عملية الترجمة، منها على سبيل المثال:

1. جدول النص المصدري.
2. جدول الثوابت، الذي يحوي جميع الثوابت الصحيحة، والعشرية المستخدمة.
3. جدول شجرة التحليل، الذي يحوي التحليل التنظيمي للبرنامج.
4. جدول الرموز، الذي يحوي أسماء، وخصائص المتغيرات المستخدمة.
5. المكدس والمستخدم من أجل استدعاء الإجراءات داخل المترجم نفسه.

كل جدول من الجداول الأربعة الأولى يزداد حجمه في أثناء عملية الترجمة، بينما قد ينمو، أو يتقلص المكدس بطريقة غير متوقعة خلال الترجمة. هذا النمو قد يجعل الجداول تصطدم مع بعضها البعض عندما تُحمَّل في الذاكرة الظاهرية أحادية البعد، مثلما موضح في الشكل 5. 35.



الشكل 5. 35: نمو الجداول داخل الذاكرة الظاهرية أحادية البعد قد ينتج عنه اصطدامها ببعضها.

بالنظر إلى ما يحدث عندما يكون عدد المتغيرات لبرنامج– ما– أكبر مما هو معتاد، وبقية أجزاء البرنامج تأخذ حجمها الطبيعي، نجد أنَّ الجزء المخصص لجدول الرموز قد يُستغَل بالكامل، ويُؤدي إلى الاصطدام مع جزء الثوابت، في نفس الوقت قد تكون هناك مساحات أخرى حرة في بقية الجداول. هذا الأمر يجعل المترجم يُرسل رسالة مفادها أنه لا يُمكن مواصلة الترجمة، بسبب وجود عدد كبير من المتغيرات داخل الذاكرة، بالرغم من وجود أماكن خالية بها، وهو ما يجعل هذا الأمر غير مقبول.

للوهلة الأولى قد يكمن الحل المبدئي لهذه المشكلة في سحب أماكن من الأجزاء الأخرى وتخصيصها إلى الأجزاء صغيرة الحجم، إلَّا إنَّه لن يكون مجديًا، لأنه عاجلًا أم آجلًا ستتكرر المشكلة من جديد، بالتالي ما نحتاجه هو طريقة– ما– تُحرر المبرمج من إدارة نمو وتقلص الجداول، بنفس الطريقة التي حَدَّثْ بها الذاكرة الظاهرية من الاهتمام بعملية تنظيم البرنامج في شرائح (راجع القسم **6.5**).

لتفادي ما سبق ذكره من مآخذ على الذاكرة الظاهرة والتصفح، يَكمن الحل الجدري والعام

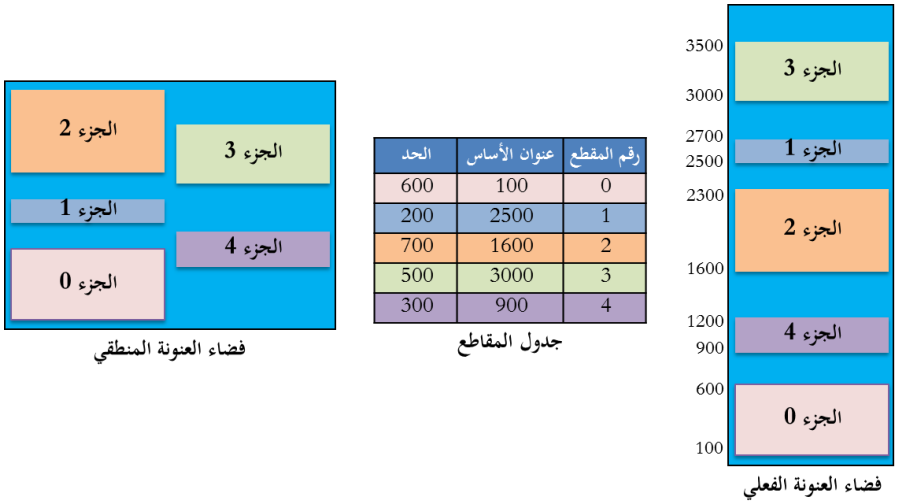
لجميع الملاحظات السابقة في تزويد الحاسوب بمجموعة من فضاءات العنونة، المستقلة عن بعضها استقلالًا تامًا. يُطلق على هذه الفضاءات اسم مقاطع، وهي ناتجة من تقطيع الذاكرة الفعلية إلى مجموعة من الأجزاء كل منها يُمثل فضاء منفصل عن الآخر، بحيث يحتوي كل مقطع على سلسلة من العناوين الخطية المتتالية، المبتدئة من صفر وحتى الحد الأقصى لكل مقطع، كما هو موضح في الشكل 5. 36. هذه المقاطع قد تكون ذات أطوال مختلفة أو متساوية، ومن الممكن أن تنمو أو تتقلص في أثناء تنفيذ البرنامج. مثلًا، يزداد طول مقطع المكدس عند إضافة أي عنصر ويتقلص بحذفه.

الشكل 5. 36: الذاكرة المقطعة تسمح باستقلالية نمو أو تقلص كل جدول عن الآخر.

لأن كل مقطع له مساحة عنوان منفصل، يُمكن للمقاطع المختلفة أن تنمو، أو تتقلص بشكل مستقل، دون أن تُؤثر في بعضها البعض. مثلًا، إذا كان المكدس في مقطع معين واحتاج إلى فضاء عنونة إضافي لينمو فيه، فيُمكنه الحصول عليه، لأنه لا يوجد أي شيء آخر في ذلك الفضاء ليصطدم به. بالطبع، يُمكن أن يمتلئ المقطع بالكامل، إلَّا إنَّ هذه المقاطع كبيرة جدًا، لذلك هذا الأمر عادة ما يكون نادر الحدوث.

لتحديد عنوان في الذاكرة المقطعة (ثنائية البعد)، يجب على البرنامج توليد عناوين منطقية ذات بعدين، الأول يُمثل رقم المقطع والثاني الإزاحة (العنوان) داخل المقطع نفسه. كما هو الحال في التصفح، لترجمة العنوان المنطقي إلى عنوان خطي فعلي يُستخدم جدول المقاطع

الموضح في الشكل 5. 37.

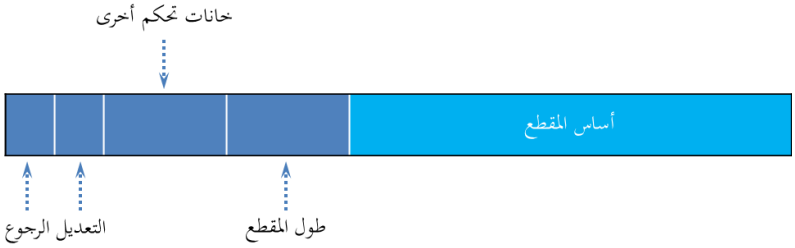


الشكل 5. 37: المظهر المنطقي للتقطيع.

1.8.5 جدول المقاطع

يحتفظ جدول المقاطع بالمعلومات الخاصة بكل مقطع وهو يُستخدم لمواءمة العنوان المنطقي ذو البعدين إلى عنوان فعلي ذو بعد واحد. يتكون هذا الجدول من عدة مداخل تبعًا لعدد مقاطع العملية. كل مدخل في هذا الجدول له البنية المبينة في الشكل 5. 38 والذي يحتوي على عدة حقول أهمها:

- عنوان الأساس: يُمثل بداية العنوان الفعلي لموقع المقطع في الذاكرة.
- الحد: يُمثل طول المقطع.



الشكل 5. 38: بنية مدخل جدول المقاطع.

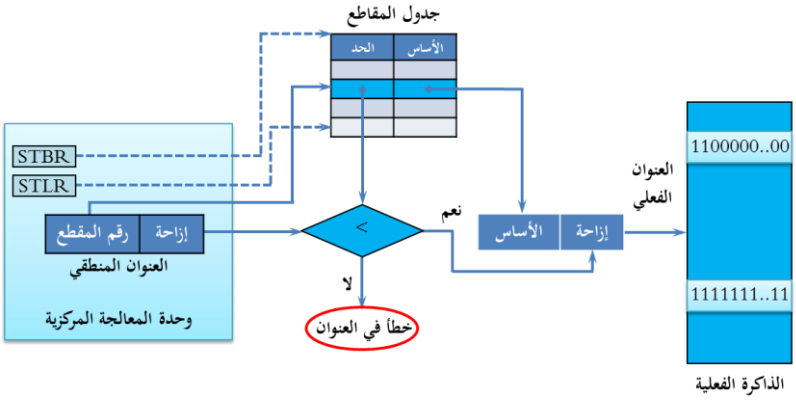
ولأن لكل برنامج فضاء عنونة منطقي خاص به، فإن لكل عملية جدول مقاطع خاص بها كذلك. يُحتفظ بهذا الجدول في الذاكرة في أثناء تنفيذ عمليته، ويُرجع إليه عن طريق سجلين ماديين هما: سجل أساس جدول المقاطع وسجل طول جدول المقاطع. في بعض من عمارات الحاسوب، إذا كان جدول المقطع صغير بما فيه الكفاية فيُحفظ في سجلات وحدة المعالجة المركزية.

استخدام جدول المقاطع

عندما يُولَّد العنوان المنطقي يُترجم إلى عنوان فعلي باستخدام جدول المقاطع على النحو التالي:

الخطوة الأولى: يُتحقق من أن العنوان المنطقي ضمن حدود المقطع المحدد بسجل أساس جدول المقاطع، وسجل طوله، إذا كان رقم المقطع خارج الحدود، فإن العنوان غير صالح وستصدر رسالة خطأ بالخصوص.

الخطوة الثانية: يُتحقق من أن الإزاحة أقل من الحد، إذا كانت أكبر فستصدر رسالة خطأ في العنوان، وإلَّا فسيكون العنوان صالح وسيُحدد العنوان الفعلي بواسطة الإزاحة والأساس المأخوذين من جدول المقاطع الموضح في الشكل 5. 39.



الشكل 5. 39: ترجمة العنوان المنطقي إلى عنوان فعلي. STBR هي اختصار لجملة Segment table base register وتعني سجل أساس جدول المقاطع، بينما STLR هي اختصار لجملة Segment table length register وتعني سجل طول جدول المقاطع.

مثال توضيحي: من خلال تتبع جدول المقاطع المعطَى في الجدول 5. 6، أيٍ من العناوين المنطقية التالية ستُولد خطأ عنوان؟

أ. 0، 430

ب. 1، 11

ت. 2، 100

ث. 3، 425

ج. 4، 95

إذا لم يحدث خطأ عنوان فاحسب العنوان الفعلي.

الجدول 5. 6: مثال توضيحي.

المقطع	الأساس	الحد
0	1219	700
1	2300	14
2	90	100
3	1327	580
4	1952	96

الحل:

من خلال مخطط التقطيع، نجد أن العنوان المنطقي يتكون من رقم المقطع وإزاحته، كما أنه من المعروف أن إزاحة المقطع يجب أن تقع دائما في المدى [0، الحد الأقصى−1]، بالتالي إذا كانت الإزاحة أكبر من أو تساوي الحد الخاص بالمقطع فسينتج عن ذلك خطأ العنوان.

الفقرة الأولى: 0، 430

من خلال المعطيات، نجد أن رقم المقطع = 0، وإزاحته = 430 وأن حده = 700. بالتالي إزاحة هذا المقطع يجب أن تقع في المدى [0، 699].

بناءً على ما سبق تقع إزاحة هذا المقطع في المدى المحسوب، الأمر الذي سوف لن ينتج

عنه خطأ عنوان وأنَّ العنوان الفعلي سيكون 1219 + 430 = 1649

الفقرة الثانية: 1، 11

من خلال المعطيات، نجد أن رقم المقطع = 1، وإزاحته = 11 وأن حده = 14. بالتالي إزاحة هذا المقطع يجب أن تقع في المدى [0، 13].

بناءً على ما سبق تقع إزاحة هذا المقطع تقع في المدى المحسوب، الأمر الذي سوف لن ينتج عنه خطأ عنوان وأنَّ العنوان الفعلي سيكون 2300 + 11 = 2311

الفقرة الثالثة: 2، 100

من خلال المعطيات، نجد أن رقم المقطع = 2، وإزاحته = 100 وأن حده = 100. بالتالي إزاحة هذا المقطع يجب أن تقع في المدى [0، 99].

بناءً على ما سبق نجد أن إزاحة هذا المقطع لا تقع في المدى المحسوب، الأمر الذي سوف ينتج عنه خطأ عنوان وأنَّ هذا الطلب غير صحيح، ولا يُمكن توليد عنوان فعلي صحيح.

الفقرة الرابعة: 3، 425

من خلال المعطيات، نجد أن رقم المقطع = 3، وإزاحته = 425 وأن حده = 580. بالتالي إزاحة هذا المقطع يجب أن تقع في المدى [0، 579].

بناءً على ما سبق نجد أن إزاحة المقطع 3 تقع في المدى المحسوب، الأمر الذي سوف لن ينتج عنه خطأ عنوان وأنَّ العنوان الفعلي سيكون 1327 + 425 = 1752

الفقرة الخامسة: 4، 95

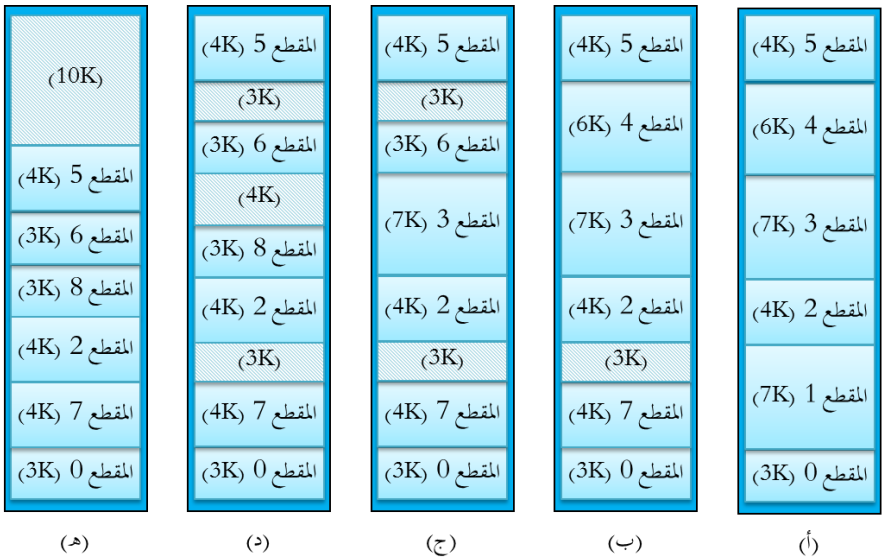
من خلال المعطيات، نجد أن رقم المقطع = 4، وإزاحته = 95 وأن حده = 96. بالتالي إزاحة هذا المقطع يجب أن تقع في المدى [0، 95].

بناءً على ما سبق نجد أن إزاحة المقطع 4 تقع في المدى المحسوب، بالتالي سوف لن ينتج خطأ عنوان وأنَّ العنوان الفعلي سيكون 1952 + 95 = 2047

2.8.5 تنفيذ التقطيع

تختلف عملية تنفيذ التقطيع عن عملية التصفح من ناحية جوهرية، وذلك كون أن التصفح ثابت الحجم، في حين أن التقطيع متغير. يوضح الشكل 5. 40–أ مثالًا لذاكرة رئيسية تحتوي مبدئيًا على ستة مقاطع، في حين يُمثل الشكل 5. 33–ب نفس الذاكرة بعد استبدال المقطع الأول بالمقطع السابع، ونتيجةً لأن حجم المقطع الأخير أصغر من حجم المقطع الأول، سيؤدي ذلك إلى ظهور فراغ (غير مستخدم) بين المقطع السابع والمقطع الثاني. بنفس الكيفية أُستبدل المقطع الرابع بالمقطع السادس، والمقطع الثامن بالمقطع الثالث لينشأ عن ذلك الشكل 5. 33–ج.

بعد وهلة من الزمن نلاحظ، كما هو الحال في الشكل 5. 33–د، وجود أماكن مستخدمة، وأخرى غير مستخدمة، أي أماكن تحوي مقاطع وأخرى تحتوي على فراغات. هذه الظاهرة تُسبب في ضياع مساحة من الذاكرة، وتُعرف باسم التفتت الخارجي للذاكرة الذي من الممكن القضاء عليه بواسطة عملية تُعرف بالضغط، وذلك كما هو مبين في الشكل 5. 33–هـ.



الشكل 5. 40: أ–د) تطور ظاهرة التفتت الخارجي للذاكرة– هـ) القضاء عليها عن طريق الضغط.

3.8.5 مزايا الذاكرة المقطعة

من خلال دراسة التقطيع نجد أن لدى الذاكرة المقطعة عدة مزايا إلى جانب تبسيط التعامل مع هياكل البيانات التي تنمو أو تتقلص حسب حاجة البرنامج، تتمحور هذه المزايا في الآتي:

- تُساهم إمكانية تحميل الإجراءات في مقاطع مستقلة تبدأ مع العنوان 0 في جعل ربط الإجراءات المترجمة بشكل منفصل بسيط إلى حد كبير. هذا الأمر يجعل تعديل أو إعادة ترجمة أي إجراء في أي مقطع لا يحتاج إلى تغيير أي إجراءات أخرى، لأنه لا حاجة إلى تعديل عناوين البداية، وإنما فقط تغيير رقم المقطع. أمَّا في حالة استخدام ذاكرة ذات بعد واحد فسيكون الأمر مختلف، لأن تغيير حجم إجراء واحد يُمكن أن يُؤثر على عنوان بداية الإجراءات الأخرى، غير ذات الصلة. وهذا، بدوره، يتطلب تعديل جميع إجراءات العملية التي تستدعي إجراءات منقولة، وذلك من أجل إدراج عناوين البداية الجديدة، وهو ما سيجعل هذه العملية مكلفة.
- يُساعد التقطيع العمليات في تشارك الإجراءات، أو البيانات كالمكتبات المشتركة بين عدة عمليات. فمحطات العمل الحديثة التي تعمل عليها نظم النوافذ المتقدمة تمتلك– في كثير من الأحيان– مكتبات رسومية كبيرة للغاية تُترجم، وتُجمع تقريبًا في كل برنامج. في الأنظمة الداعمة للتقطيع، يُمكن وضع هذه المكاتب في أحد المقاطع، بالتالي مشاركتها مع عدة عمليات سيكون سهلًا. يُلغي هذا الأمر الحاجة إلى وجودها في فضاء العنونة الخاص بكل عملية، مع العلم أنَّه من الممكن كذلك مشاركة هذه المكتبات في أنظمة التصفح النقية، إلَّا إنَّ الأمر سيكون أكثر تعقيدًا. في الواقع، هذه الأنظمة تفعل ذلك عن طريق محاكاة التقطيع.
- يُساهم التقطيع أيضًا في توفير خاصية الحماية لمقاطع الذاكرة وتوفير أنواع مختلفة منها. مثلًا، يُمكن تحديد خاصية التنفيذ فقط لمقطع يحوي إجراء محدد، بالتالي حظر محاولات القراءة منه أو التخزين فيه. كما يُمكن تحديد خاصية القراءة/الكتابة لمصفوفة موجودة في مقطع آخر بالتالي حظر عملية التنفيذ.

السؤال هنا: لماذا تُعتبر الحماية أمر منطقي في الذاكرة المقطعة، وغير ممكنة في الذاكرة المقسمة إلى صفحات أي ذات البعد الواحد؟ كما أشرنا سابقًا، في الذاكرة المقطعة، عادة ما يكون المستخدم على علم بما هو موجود في كل مقطع، ولأن كل مقطع يحتوي

على نوع واحد فقط من الكيانات، يُمكن أن يخُصص له نوع مناسب من الحماية، وهو ما لا يتأتى في أنظمة التصفح.

9.5 مقارنة بين التصفح والتقطيع

تستخدم نظم التشغيل الحديثة تقنيات متقدمة لتخصيص الذاكرة تشمل التصفح والتقطيع. كلاهما يندرج تحت التخصيص اللامتجاور للذاكرة، أي لا يشترطا أن تكون مواقع التخصيص متتالية، إلاَّ إنَّ التقنية الأولى وُجدت للحصول على فضاء عنونة خطي كبير دون الحاجة إلى اقتناء ذاكرة فعلية إضافية. هذه التقنية تسمح بإدارة الذاكرة من خلال تقسيمها إلى كتلة ثابتة الحجم تُعرف بالصفحة، ولا يتدخل المستخدم (أو المبرمج) في عملية تقسيم البرنامج إلى صفحات، كما أن حجم الصفحة يتحدد مسبقًا بناءً على معطيات الكيان المادي. في حين وُجدت تقنية التقطيع للسماح للبرامج والبيانات بأن تتوزع على فضاءات عنونة منطقية ومستقلة، وكذلك لإسعاف المشاركة والحماية، وهي تعتمد على تقسيم الذاكرة إلى مقاطع بأحجام متغيرة بناءً على رغبة المستخدم. يعرض الجدول 5. 7 مقارنة موجزة بين هاتين التقنيتين.

الجدول 5. 7: مقارنة موجزة بين كل من التصفح والتقطيع.

التصفح	التقطيع
أقرب إلى نظام التشغيل وهو المسؤول عنه.	أقرب إلى المستخدم، والمترجم هو المسؤول عن العمل.
يتحكم الكيان المادي في حجم الصفحة.	يتحكم المستخدم في حجم المقطع.
لا يُمكن احتضان الجداول ذات الأحجام المتغيرة بسهولة.	يُمكن القيام بذلك بكل سهولة.
تُخزن معلومات الصفحة في جدول الصفحة.	تُخزن المعلومات حول المقاطع في جدول المقاطع.
بإمكان مجموع فضاءات العنونة تجاوز حجم الذاكرة الفعلية.	بإمكان مجموع فضاءات العنونة تجاوز ذلك.
يحتوي مدخل جدول الصفحة على رقم إطار الصفحة وبعض من الخانات الأخرى التي تُمثل تفاصيل الصفحات.	يحتوي مدخل جدول المقاطع على العنوان الأساسي للمقطع وبعض من خانات الحماية له.
حجم جدول الصفحة أكبر من حجم جدول المقاطع.	حجم جدول المقاطع أصغر من حجم جدول الصفحة.
يُعاني من التفتت الداخلي للذاكرة بدلًا من الخارجي.	يُعاني من التفتت الخارجي بدلًا من الداخلي.

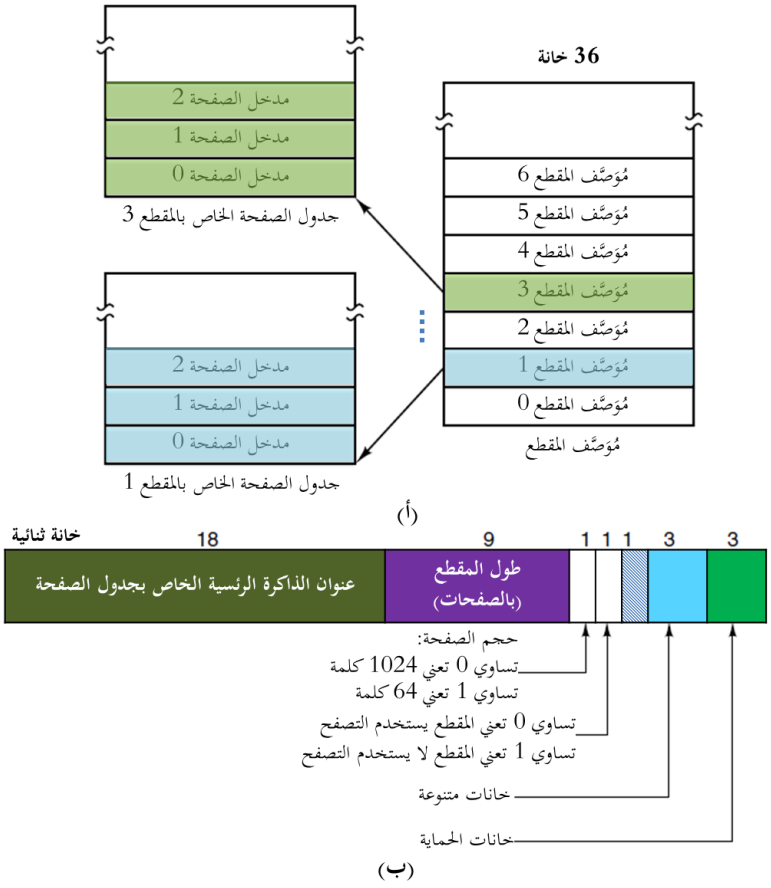
التصفح	التقطيع
هناك فضاء خطي واحد.	هناك عدة فضاءات.
ينقسم العنوان المنطقي إلى رقم الصفحة وإزاحة الصفحة.	ينقسم العنوان المنطقي إلى رقم المقطع وإزاحة المقطع.
أسرع من التقطيع.	أبطأ من التصفح.
من غير الممكن مشاركة الإجراءات بين المستخدمين.	من الممكن القيام بذلك.
لا يحتاج المبرمج إلى أن يكون على دراية بأن هذه التقنية مستخدمة.	يحتاج المبرمج إلى ذلك.
لا يُمكن التمييز بين– وحماية كل من الإجراءات والبيانات بشكل منفصل.	يُمكن التمييز بين– وحماية كل من الإجراءات والبيانات بشكل منفصل.
يحتاج إلى وصول واحد لجلب التعليمات من الذاكرة.	يحتاج إلى وصولين لجلبها من الذاكرة (أي مكلف زمنيًا).
العنونة: أحادية البعد.	العنونة: ثنائية البعد.

10.5 دمج التقطيع والتصفح

من خلال المقارنة المبينة في القسم السابق يُمكن للمرء أن يُلاحظ مجموعة المزايا التي يُقدمها كل من التصفح والتقطيع. بالتالي للاستفادة أكثر من هاتين التقنيتين يُمكن الجمع بينهما في نموذج واحد يُساعد بشكل كبير في حل الكثير من المعضلات التي تواجه عالم الحاسوب منها على سبيل المثال صعوبة (أو استحالة) الاحتفاظ بالمقاطع بالكامل في الذاكرة الرئيسية عندما تكون كبيرة جدًا، ولكن من خلال هذا الدمج يُمكن الاحتفاظ فقط بتلك الصفحات التي فعليًا هناك حاجة لأن تكون في الذاكرة.

من الأمثلة الداعمة لهذه الفكرة نظام: **MULTiplexed Information and Computing Service, MULTICS**) والذي أُستخدم على آلات هوني ويل **6000** (**Honeywell 6000**) وعلى الأجيال التالية التي أُستنسخت منه. هذه الآلات كانت تُوفر لكل برنامج ذاكرة ظاهرية تصل إلى 2^{18} مقطع (أي أكثر من **250،000**)، كل منها بطول **65،536** كلمة (**36** خانة ثنائية). ولكي يُستفاد من مزايا تقنية التصفح عُومل كل مقطع كذاكرة ظاهرية، لكي لا تكون هناك ضرورة لوجود المقطع بالكامل في الذاكرة في حالة استخدام أي مقطع منها.

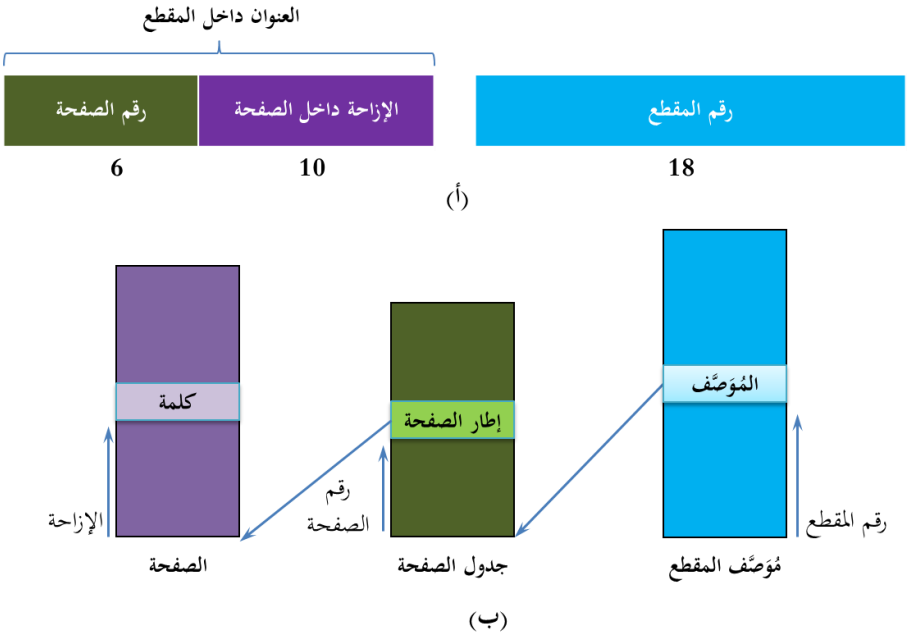
تعتمد الفكرة الأساسية لهذا النظام على تخصيص جدول مقاطع لكل برنامج، ومُوَصَّف واحد لكل مقطع. ولأنه من الممكن أن يتواجد أكثر من ربع مليون مدخل في الجدول الواحد، فالجدول نفسه يتُعامل معه على أساس مقطع منفصل ويُطبق عليه التصفح، لكي لا يُحمل بالكامل في الذاكرة. من جهة أخرى يحتوي مُوَصَّف المقطع على علامة للدلالة على ما إذا كان هذا المقطع موجود في الذاكرة الرئيسية أم لا. إذا حدث وكان أي جزء من هذا المقطع موجود في الذاكرة، فسيُعتبر المقطع وجدول الصفحة الخاص به موجودان في الذاكرة، في هذه الحالة، سيحتوي مُوَصَّفه على مؤشر يُشير إلى جدول الصفحة، كما هو موضح في الشكل 5. 41–أ.



الشكل 5. 41: الذاكرة الظاهرية لنظام MULTICS– أ) مُوَصَّف المقطع يُشير إلى جداول الصفحة– ب) بنية مُوَصَّف المقطع [Tanenbaum & Bos, 2015].

لأن العناوين الفعلية بطول 24 خانة ثنائية والصفحات في حدود 64 خانة ثمانية، فإن الست خانات ذات الرتبة المنخفضة من عناوين الصفحات هي في الواقع أصفار، لذلك مؤشر المُوَصَّف سيكون بطول 18 خانة ثنائية فقط، لكي يتتبع عنوان جدول الصفحة. يحتوي المُوَصَّف كذلك على حجم المقطع، خانات الحماية، وبعض من علامات التحكم الأخرى، يوضح الشكل 5. 41–ب مُوَصَّف مقطع نظام MULTICS، تُبين الأرقام في هذا الشكل أطوال الحقول.

للتعرف أكثر على مكونات العنوان الظاهري في نظام MULTICS يُمكن النظر إلى الشكل 5. 42–أ. يتكون هذا العنوان من جزئين هما: رقم المقطع، والعنوان داخله والذي ينقسم كذلك إلى رقم الصفحة، وإلى الإزاحة داخل الصفحة نفسها، بالتالي عند الرجوع إلى الذاكرة، تنفذ الخوارزمية التالية:



الشكل 5. 42: أ) العنوان الظاهري لنظام MULTICS بطول 34 خانة ثنائية– ب) ترجمة هذا العنوان إلى عنوان فعلي [Tanenbaum & Bos, 2015].

1. يُستخدم رقم المقطع للعثور على مُوَصَّفه.
2. يُتحقق من أنَّ جدول الصفحة لهذا المقطع موجود في الذاكرة. إذا كان كذلك، يُحدد

الموقع، إذا لم يكن كذلك، يحدث خطأ المقطع، أيضًا إذا كان هناك انتهاك للحماية، سيحدث خطأ الحماية.

3. يتم فحص مدخل جدول الصفحة الخاص بالصفحة الظاهرية المطلوبة. إذا كانت الصفحة نفسها غير موجودة في الذاكرة، فيُحفز خطأ الصفحة. أما إذا كانت الصفحة في الذاكرة، فيُستخلص عنوان الذاكرة الرئيسي لبداية الصفحة من مدخل جدول الصفحة.
4. تُضاف الإزاحة إلى الصفحة الأصلية لإعطاء عنوان الذاكرة الرئيسية، الذي تقع فيه الكلمة.
5. أخيرًا تحدث عملية القراءة أو الكتابة.

يُوضح الشكل 5. 42–ب هذه الخطوات، ما يحدث حقًا هنا هو أن سجل مُوَصَّف الأساس يُستخدم لتحديد مُوَصَّف المقطع الخاص بجدول الصفحة، والذي بدوره سيُشير إلى صفحات مُوَصَّف المقطع، وبمجرد ما يُعثر على مُوَصَّف المقطع المطلوب، يتم بكل سهولة الحصول على العنوان.

مما لا شك فيه الآن، أنه إذا نُفذت الخوارزمية السابقة بالفعل من قبل نظام التشغيل عند كل تعليمة سوف يُؤثر ذلك بشكل سلبي على سرعة تنفيذ البرامج. وكحل لهذه المشكلة يستعين الكيان المادي لأجهزة MULTICS بالمترجم الجانبي للمخزن اللحظي (بطول 16 كلمة) بحيث يُمكنه بحث كافة مداخله بشكل متوازي لأي مفتاح محدد، الأمر الذي سيساعد في تعجيل عملية التنفيذ. يُوضح الشكل 5. 43 نسخة من هذا المترجم. عندما يُعرض عنوان على جهاز الحاسوب، يختبر الكيان المادي للعنونة أولًا ما إذا كان العنوان الظاهري موجود في المترجم، إذا كان الأمر كذلك، فإنه يحصل على رقم إطار الصفحة مباشرةً من المترجم، ويُشكل العنوان الفعلي للكلمة المشار إليها دون الحاجة إلى النظر في مُوَصَّف المقطع أو جدول الصفحة.

من الخطوات الإضافية للتعجيل يحتفظ المترجم كذلك بعناوين أحدث ستة عشر صفحة رُجِع إليها في الآونة الأخيرة، بالتالي بإمكان البرامج التي مجموعة عملها أصغر من حجم المترجم التواجد كاملة في المترجم، الأمر الذي يجعلها تعمل بكفاءة. إذا لم تكن الصفحة المطلوبة موجودة في المترجم، فسيتم الرجوع فعلًا إلى المَوَصَّف وجداول الصفحة للعثور على عنوان إطار الصفحة، بعدها يُحدَّث المترجم ليتضمن هذه الصفحة بعد أن تُطرد الصفحة الأقل استخدامًا

مؤخرًا، ولتحديد هذا النوع من الصفحات يُستخدم حقل العمر.

رقم المقطع	الصفحة الظاهرية	إطار الصفحة	الحماية	العمر	هل المدخل مستخدم؟
	حقل المقارنة				
4	2	6	قراءة/كتابة	14	1
8	0	7	قراءة فقط	9	1
					0
3	3	5	تنفيذ فقط	8	1
7	5	12	قراءة فقط	3	1
2	1	0	قراءة/كتابة	6	1
					0
....					

الشكل 5. 43: نسخة مبسطة من المترجم الجانبي للمخزن اللحظي الخاص بنظام MULTICS.

11.5 موجز الفصل

ناقش هذا الفصل إدارة الذاكرة، واستهلها بمدخل أوضح من خلاله أهمية هذا المصدر، ومن ثم انتقل لشرح التسلسل الهرمي للذاكرة والذي بَيَّن العلاقة الرباعية لخصائص الذاكرة والمتمثلة في النوع، والسعة، والسرعة، وكذلك التكلفة، ولخصها في أنَّ أسرع هذه الأنواع وأقلها حجمًا هي السجلات تليها ذاكرة كاش، ثم الذاكرة الرئيسية، وأخيرًا في قاع الهرم توجد الذاكرة الثانوية والتي تُعتبر أكبرهم حجمًا وأبطأهم سرعةً. تُدار كل هذه الأنواع من قبل نظم التشغيل عن طريق مدير الذاكرة وتتلخص مهمته في تحديد وتتبع الأجزاء قيد الاستخدام، والأجزاء غير المستخدمة من الذاكرة، وتخصيص الذاكرة للعمليات عندما تكون في حاجة إليها، وإدارة عمليات المبادلة بين الذاكرة والقرص.

كما بَيَّن الفصل كيف أن أبسط أنظمة إدارة الذاكرة تلك التي لا تستخدم كل من المبادلة أو التصفح على الإطلاق وتُعرف بإدارة الذاكرة المفردة. في مثل هذه الأنظمة، إذا حُمِّل البرنامج في

الذاكرة، فسيظل هناك إلى أن ينتهي، وهو ما يُمثل عيب في هذا النوع من الإدارة. ومع هذا هناك بعض من أنظمة التشغيل التي قد تسمح في كل مرة بتواجد عملية واحدة فقط في الذاكرة، وهناك البعض الآخر الذي قد يدعم البرمجة المتعددة والتي أدت إلى ظهور مفهوم إدارة الذاكرة المجزءة. في إطار هذه الإدارة قُسِّمت الذاكرة إلى عدة أجزاء وأصبح بالإمكان تحميل عدة برامج فيها وفي نفس الوقت، الأمر الذي أضاف مهام جديدة إلى مدير الذاكرة شملت تجهيز قوائم انتظار للبرامج المراد تحميلها في الذاكرة، والمفاضلة بينها من حيث أولويات التحميل في الذاكرة، وكذلك حماية البرامج من التداخل في أثناء التنفيذ.

بعد ذلك تطرق الفصل إلى موضوع المبادلة، والتي عُرِّفت بعملية نقل العمليات من الذاكرة إلى القرص ثم العكس، والتي عند استخدامها، يُمكن للنظام التعامل مع المزيد من العمليات تصل إلى أكثر مما تستوعبه المساحة المخصصة لها في الذاكرة. كما أدى هذا المفهوم إلى اعتماد فضاءات العنونة، وحدي الأساس والحد لاستخدامهما في حماية حقوق الوصول إلى هذه الفضاءات كل حسب العملية المخصص لها. ولأن عملية المبادلة تتطلب تخصيص أماكن للعمليات في الذاكرة، ناقش الفصل كذلك مقدار حجم الجزء الذي سيُخصص للعملية عند إنشائها والذي تدرج من الثابت إلى المتغير، وإلى تخصيص مساحات إضافية تَسُدْ في الغالب حاجة العملية عندما تنمو في أثناء التنفيذ، وذلك للحد من الجهد المرتبط بإجهاض العملية، أو نقلها، أو تعليقها، أو مبادلة عمليات أخرى في حالة شَغِل المساحة المحجوزة بالكامل.

ولأن تخصيص أماكن للعمليات داخل الذاكرة مرتبط بإدارة المساحات الحرة فيها، تعرض الفصل إلى سبل إدارة وتتبع فراغ الذاكرة والتي أورد منها طريقة خرائط الخانات الثنائية، وطريقة القوائم المرتبطة. تُقسم الذاكرة في الطريقة الأولى إلى مجموعة من الأجزاء بحيث يُناظر كل جزء منها خانة ثنائية داخل الخريطة الثنائية تُعبر عن حالته من حيث الاستخدام. أمَّا الطريقة الثانية فهي تعتمد على تقسيم الذاكرة إلى عقد تحوي كل منها عدد محدد من الحقول أهمها حقل المعلومات، وحقل الدلالة على استخدام العقدة من عدمها، وحقل المؤشر الذي يُشير إلى العقدة التالية في القائمة. لتخصيص هذه العقد للعمليات تناول الفصل خوارزميات تخصيص أجزاء الذاكرة تمثلت في خوارزمية البحث عن أول فراغ مناسب، وخوارزمية البحث عن أفضل فراغ مناسب، وخوارزمية البحث التالي عن الفراغ المناسب، وخوارزمية البحث عن أكبر فراغ مناسب،

وخوارزمية البحث السريع عن الفراغ المناسب.

يلي خوارزميات التخصيص هذه استطرق الفصل إلى أحد أهم مواضيع إدارة الذاكرة والمستخدم في أغلب الحواسيب الحديثة وهو موضوع الذاكرة الظاهرية. تلعب تقنية التصفح دورًا مهمًا في تنفيذ الذاكرة الظاهرية، وهي في أبسط صُوَرِها أن كل عملية لها فضاء عنونة ظاهري خاص بها مقسم إلى مقاطع ذات أحجام موحدة تُعرف بالصفحات، يُمكن تحميلها في أي إطار صفحة متاح داخل الذاكرة الفعلية مستعينةً في ذلك بجدول الصفحة لكونه يحوي فقط المعلومات التي يحتاجها الكيان المادي لترجمة العنوان الظاهري إلى عنوان فعلي.

من خلال التعرض إلى تقنية التصفح، بيَّن الفصل أهمية تسريع مواءمة العنوان الظاهري مع العنوان الفعلي، وكذلك معالجة العلاقة الطردية ما بين حجم فضاء العنونة الظاهري وحجم جدول الصفحة. لذلك عرَّج الفصل على مفهوم المترجم الجانبي للمخزن اللحظي، والكيفية التي يُدار بها من أجل تعجيل التصفح، وعلى مفهوم جداول الصفحة للذواكر الكبيرة من أجل التعامل مع فضاءات العنونة الكبيرة من خلال شرح كل من جداول الصفحة متعددة المستويات، وجدول الصفحة المعكوس.

كتكملة لمفهموم المبادلة أسرد الفصل مجموعة من خوارزميات استبدال الصفحات المستخدمة في اختيار إحدى الصفحات غير المرغوب فيها كضحية وطردها من الذاكرة، لغرض إفساح المجال أمام الصفحات الواردة عندما لا يوجد لها مكان فيها. تتفاوت هذه الخوارزميات في آلية اختيار الضحية، إلَّا إنَّ أفضلها الخوارزمية التي تُحقق أدنى معدل خطأ صفحة لذلك خلص ملخص مقارنة هذه الخوارزميات إلى أن أفضلها خوارزمتي الشيخوخة وساعة مجموعة العمل.

لمعالجة قصور التصفح من حيث عدم قدرته على التعامل مع هياكل البيانات التي تتغير أحجامها في أثناء التنفيذ، وتبسيط عمليات الربط والمشاركة، وكذلك توفير حمايات مختلفة للأجزاء المختلفة، قدَّم الفصل التقطيع والذي أفضى إلى تقطيع الذاكرة الفعلية إلى مقاطع كل منها يُمثل فضاء منفصل عن الآخر، مستعينًا في ذلك بجدول المقاطع والذي يحتفظ بالمعلومات الخاصة بكل مقطع ويُستخدم لمواءمة العنوان المنطقي ذو البعدين إلى عنوان فعلي ذو بعد واحد.

أخيرًا، قدَّم الفصل مقارنة بين تقنيتي التصفح والتقطيع والتي خلصت إلى أن التقنية الأولى وُجدت للحصول على فضاء عنونة خطي كبير دون الحاجة إلى اقتناء ذاكرة فعلية إضافية، كما أنها سمحت بإدارة الذاكرة من خلال تقسيمها إلى صفحات دون أي تدخل من المبرمج، وأنَّ حجم الصفحة يتحدد مسبقًا بناءً على معطيات الكيان المادي، بينما تعتمد تقنية التقطيع على تقسيم الذاكرة إلى مقاطع بأحجام متغيرة بناءً على رغبة المستخدم، وتسمح للبرامج والبيانات بأن تتوزع على فضاءات عنونة منطقية ومستقلة، لغرض إسعاف مفهومي المشاركة والحماية. وللاستفادة من مزايا تقنيتي التصفح والتقطيع، أُختتم الفصل بعرض نقاش مفصَّل لنموذج يجمع بينهما في نظام واحد يُعرف بنظام MULTICS.

12.5 أسئلة للمراجعة

1. ما المقصود بالذاكرة؟ وما التسلسل الهرمي لها؟
2. ما أهم الوظائف الرئيسية لمدير الذاكرة؟
3. ما الخيارات الممكنة لنظام التشغيل في إدارة الذاكرة المفردة؟
4. ما مزايا وعيوب إدارة الذاكرة المفردة؟
5. كيف يُمكن ضمان بعض من التوازي في تنفيذ البرامج في ظل إدارة الذاكرة المفردة؟
6. في ظل استخدام الذاكرة المجزءة ما الوظائف الإضافية لمدير الذاكرة؟
7. ما المقصود بالتفتت الداخلي للذاكرة؟ وكيف ينشأ؟ وكيف يُعالج؟
8. ما أهمية مفهوم فضاء العنونة؟
9. ما أهمية سجلي الأساس والحد في إطار مفهوم المبادلة؟ وهل يمثلان حلًا مثاليًا؟
10. ما المقصود بالتفتت الخارجي للذاكرة؟ وكيف ينشأ؟ وكيف يُمكن القضاء عليه؟
11. ما الطرق المتاحة لإدارة فراغ الذاكرة وأيها أمثل؟
12. بالنظر إلى نظام مبادلة تحتوي الذاكرة فيه على فراغات بالأحجام التالية وبنفس الترتيب: 15، 5، 20، 22، 8، 13، 22 و 15، أيٍ من هذه الفراغات سوف يُخصص لطلبات الأجزاء المتعاقبة التالية:
    أ. 13
    ب. 8

ج. 14

عند استخدام خوارزمية البحث عن أول فراغ مناسب؟ الآن كرر نفس السؤال لكل من خوارزمية البحث عن أفضل فراغ مناسب، والبحث عن أسوأ فراغ مناسب، والبحث عن الفراغ المناسب التالي.

13. وضِّح مفهوم الذاكرة الظاهرية.

14. ما الفرق بين العنوان الفعلي والعنوان الظاهري؟

15. ما المقصود بخطأ الصفحة؟

16. عدِّد مزايا وعيوب ازدياد حجم الصفحة.

17. باستخدام جدول الصفحة الموضح في الشكل 5. 17، بين العنوان الفعلي المناظر لكل من العناوين الظاهرية التالية:

أ. 20

ب. 5855

ج. 16000

18. وضِّح باختصار مفهوم تقنية التصفح، وعدَّد المشاكل التي يُعاني منها، والكيفية التي عُولجت بها؟

19. ما وظيفة خانة فصل التخزين في بنية مدخل الصفحة؟

20. ما المقصود بالمترجم الجانبي للمخزن اللحظي؟ وما استخداماته؟

21. ما المقصود بخطأ المترجم، أو المترجم المفقود؟ وكيف يعُالج هذا الخطأ؟ وما أنواعه؟

22. ما المقصود بجداول الصفحة متعددة المستويات؟ وما الغاية والغرض منها؟

23. يَستخدم حاسوب تتشكل عناوينه من 32 خانة ثنائية جدول صفحة بمستويين، العناوين الظاهرية فيه مقسمة إلى حقل خاص بجدول الصفحة للمستوى العلوي بطول 9 خانات ثنائية، وحقل خاص بجدول الصفحة للمستوى الثاني وبطول 11 خانة ثنائية، وحقل خاص بالإزاحة. ما هو كبر الصفحات؟ وكم هناك صفحة في فضاء العنونة هذا؟

24. إذا كان هناك حاسوب يُنفذ 100 تطبيق باستخدام جدول صفحة بمستويين، بناءً على معطيات الشكل 5. 21، كم حجم المساحة التي يجب حفظها في الذاكرة الرئيسية طيلة فترة التنفيذ حتى يتمكن من الوصول إلى كل بيانات التطبيقات؟

25. عند استخدام جدول الصفحة متعدد المستويات، ما هي أصغر كمية من بيانات جدول الصفحة لغرض حفظها في الذاكرة الرئيسية من أجل تطبيق بطول 32 خانة ثنائية؟ (راجع معطيات الشكل 5. 21)

26. بفرض أن العنوان الظاهري الذي طوله 32 خانة ثنائية قُسِّم إلى أربعة حقول أ، ب، ج، د. تُستخدم الثلاثة الأولى منها لنظام جدول الصفحة بثلاثة مستويات، ويُمثل الحقل الرابع د الإزاحة. هل عدد الصفحات يعتمد على أحجام الحقول الأربعة؟ إن لم يكن كذلك أيٍ منها يؤثر؟ وأيٍ منها لا يؤثر؟

27. وضِّح مفهوم جدول الصفحة المعكوس، وبين لماذا نحتاجه؟

28. ما الغاية والغرض من خوارزميات استبدال الصفحة؟ وأين يُمكن تطبيقها؟

29. إذا كان هناك حاسوب له خمس إطارات صفحة، وكان زمن تحميل، وزمن آخر استخدام، وكذلك قيمة كل من خانتي الرجوع والتعديل لكل صفحة كما هو موضح في الجدول التالي (الزمن مقاس بالتكات النبضية):

الصفحة	زمن التحميل	زمن آخر رجوع	M	R
0	120	179	1	0
1	225	270	0	1
2	200	262	0	0
3	160	280	1	1
4	150	167	1	0

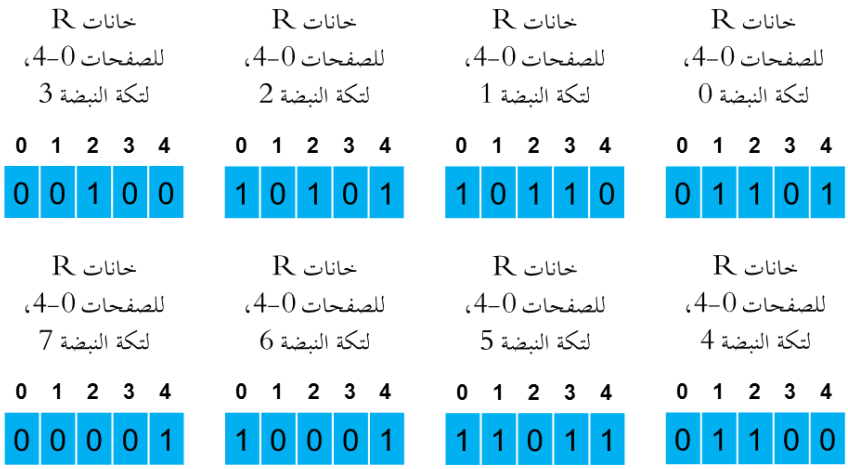
فما هي الصفحة التي ستُستبدل في حالة استخدام كل من خوارزمية الداخل أولًا يخرج أولًا، واستبدال الصفحات غير المستخدمة مؤخرًا، واستبدال الصفحات المستخدمة مؤخرًا بقلة، والفرصة الثانية لاستبدال الصفحات؟

30. إذا أُستخدمت خوارزمية الداخل أولًا يخرج أولًا لاستبدال الصفحات بوجود أربعة إطارات للصفحة، وثماني صفحات. كم عدد أخطاء الصفحات الذي سيحدث في حالة ما إذا كانت سلسلة الرجوع لهذه الصفحات على النحو **0176232752103** (البداية من اليمين)، علمًا أن إطارات الصفحة الأربعة في البداية فارغة؟ كرر هذه المسألة مع خوارزمية استبدال

الصفحات المستخدمة بقلة مؤخرًا.

31. اشرح ظاهرة شذوذ بلادي من خلال مثال تضعه أنت.

32. جهاز حاسوب يملك خمسة إطارات للصفحة، وحالات خانات **R** كانت على النحو المبين أدناه. إذا أُستخدمت خوارزمية الشيخوخة مع عدَّاد بطول 8 خانات ثنائية، استنتج قيم العدَّادات الخمسة بعد آخر تكة.



33. ما فائدة دمج التصفح مع التقطيع في نظام **MULTICS**؟ وما مهمته الرئيسية في هذا النظام؟

34. بالنظر في تسلسل الصفحات في الشكل 5. 28–ب، وبفرض أن خانة **R** للصفحات من P_1 إلى P_8 هي **10110111**، على التوالي. أي من هذه الصفحات سيتم حذفها في حالة استخدام خوارزمية الفرصة الثانية؟

35. بفرض أن τ= 300 في الشكل 5. 34، ما هي الصفحة التي سيتم حذفها؟

36. ما المقصود بمجموعة عمل العملية؟

37. وضِّح مفهوم التقطيع، مع ذكر الأسباب التي أدت إلى ظهوره.

38. ما أهم مزايا التقطيع مقارنة بتقنية التصفح؟

39. قارن بين تقنيتي التصفح والتقطيع من حيث العيوب.

40. يُساعد التقطيع في إسعاف المشاركة والحماية بين العمليات، وضِّح ذلك بالتفصيل.

41. عندما تُدمج كل من التجزئة والتصفح، كما هو الحال في نظام **MULTICS**، يجب أولًا النظر في مُوَصَّف الجزء، ومن ثم مُوَصَّف الصفحة. هل يعمل كذلك مترجم المخازن اللحظية الجانبية بنفس هذه الطريقة عندما يكون هناك مستويين من البحث؟

الفصل السادس نظم الملف

1.6 مدخل إلى نظم الملف

تحتاج كافة تطبيقات الحاسوب إلى تخزين المعلومات والرجوع إليها في أثناء التنفيذ. فالعمليات مثلًا، بإمكانها تخزين حجم معين من المعلومات في أثناء اشتغالها داخل الذاكرة المخصصة لها، هذا الحجم يتحدد أو يكون محدود بحجم الجزء المخصص لكل عملية، إلَّا إنَّه قد يتناسب في بعض من التطبيقات مع كمية المعلومات المخزنة، وقد لا يتناسب في البعض الآخر، كما هو الحال في برامج حجز التذاكر، وأنظمة المصارف، وغيرها. الأمر الذي يُمثل عيبًا جوهريًا في عملية الحفظ.

من العيوب الأخرى لهذه العملية هو أن حفظ البيانات داخل الجزء المخصص للعملية في الذاكرة مرتبط بالعملية نفسها، بمعنى إنتهاء العملية أو إجهاضها يؤدي إلى ضياع البيانات نفسها وهو ما سوف يكون غير منطقي، لأنه من المفترض أن تبقى هذه البيانات لفترة زمنية طويلة وخصوصًا إذا كانت مهمة ومستخدمة من قبل عدة عمليات أخرى.

يتمثل العيب الآخر في كون البيانات المخزنة داخل الأجزاء المخصصة للعملية تتعارض مع مفهوم المعالجة المتعددة من حيث احتياج العمليات إلى مشاركة نفس المعلومة في نفس الوقت، الأمر الذي قد لا يكون ممكنًا بسبب القيود التي يفرضها ويستخدمها مدير الذاكرة لحماية الأجزاء المخصصة للعملية داخل الذاكرة من التداخل أو الوصول غير المخول في أثناء التنفيذ.

كل ذلك يجعل بالضرورة بمكان البحث عن طرق بديلة لتخزين البيانات والتي من شأنها أن تُساهم في تفادي العيوب سالفة الذكر، وذلك من خلال تمكين التخزين المستمر للمعلومات والذي– كما سنلاحظ– سوف يحتاج إلى ثلاثة متطلبات هي:

- ضرورة وجود إمكانية تخزين كميات كبيرة من المعلومات.
- ضرورة توفر وبقاء المعلومات بشكل دائم حتى بعد الإنتهاء من استخدامها من قبل العمليات.
- في حالة استخدام المعالجة المتعددة يجب أن تكون هناك إمكانية لمشاركة البيانات، أي الوصول إلى نفس المعلومة وفي نفس الوقت من قبل العمليات.

هذه المتطلبات وغيرها يمكن تحقيقها من خلال تخزين المعلومات على وحدات التخزين

الثانوية (مثل، الأقراص) على شكل أحجام مختلفة أو متساوية تُعرف باسم الملفات بحيث يُمكن للعمليات قراءتها، أو كتابتها، أو حتى إلغائها عند عدم الحاجة إليها، مع العلم أن أي ملف لا يُمكن إلغاؤه إلَّا من قبل العملية المالكة له. هذا النوع من التخزين يتطلب نوع من الإدارة والمتمثلة في إدارة نظم الملف. بالتالي فإن هذه الإدارة تُمثل الطريقة التي يُدير بها نظام التشغيل آليات حفظ وتتبع البيانات أو الوصول إليها في صورة ملفات داخل مجلدات على وحدة التخزين، ومن ثم استعادتها أو تحديثها. الشكل 6. 1 يُوضح نظرة عامة حول مفهوم إدارة نظم الملف في إطار نظم التشغيل.

الشكل 6. 1: إدارة نظم الملف.

بناءً على هذا التعريف يمكن استخلاص بعض من المهام الأساسية لنظم الملف.

1. إدارة العمليات الخاصة بتسمية الملفات والمجلدات وسيماتها وقواعد الوصول إليها.
2. السماح لنظام التشغيل بإجراء العمليات الأساسية على الملفات والمجلدات مثل، الحذف، وتغيير الاسم، والنسخ، وغيرها.
3. تحديد وتتبع المساحات المستخدمة، وغير المستخدمة من الذاكرة الثانوية.
4. تتبع الملفات الموجودة في الذاكرة الثانوية باستخدام مجلد الملفات وجداول توزيعها.
5. استعادة المساحات المخصصة للملفات بعد حذفها من أجل إعادة تخصيصها إلى ملفات أخرى.
6. تنظيم الملفات والمجلدات في الذاكرة الثانوية بحيث يكون الوصول إليها واسترجاعها سريع وبشكل صحيح.
7. تحديد سياسة متابعة أجزاء الملفات في الذاكرة الثانوية وكيفية الوصول إليها.

2.6 الملفات

يُعتبر الملف الكيان الأساسي في أنظمة الحوسبة لتخزين المعلومات بشكل دائم في مختلف وسائط التخزين الثانوية. فهو عبارة عن آلية مجردة (بنية بيانات) تُوفر إمكانية تخزين المعلومات داخل هذه الوسائط ومن ثم قراءتها عند الطلب. هذه الإمكانية يجب أن تتوفر من دون الحاجة إلى إدخال المستخدم في التفاصيل الخاصة بمكان تخزين المعلومة، وبكيفية عمل وسائط التخزين، لأن ما يهمه هو الكيفية التي تُمكنه من استخدام الملف.

من خلال ذلك نستخلص بأن الملف هو عبارة عن مجموعة من المعلومات أو البيانات ذات الصلة ببعضها والتي يُعطَى لها اسم محدد لكي يتم الرجوع إليها من خلاله، فهو مجرد حاوية للبيانات والتي قد تكون نصية، أو ثنائية، أو صوتية، أو مرئية، أو صور، أو ما إلى ذلك من البيانات. هناك أيضا ملفات قابلة للتنفيذ، وهي عبارة عن إرشادات يمكن فهمها بواسطة وحدة المعالجة المركزية. بالتالي، سنحاول في هذا القسم تسليط الضوء على الملفات من حيث:

- طريقة تسمية وحماية الملفات.
- طريقة هيكلة الملفات.
- شرح العمليات المطبقة على الملفات.

1.2.6 تسمية الملفات

لكي يُستخدم الملف بشكل صحيح ووفق قواعد محددة، يُعطَى للملف عند إنشائه اسم خاص به من قبل العملية الناشئة له. قواعد منح هذا الاسم تختلف من نظام تشغيل إلى آخر، كما أن كل منها يعتمد على طرق خاصة في تسمية ملفاته قبل تخزينها على وسائط التخزين. اسم الملف وكذلك محتواه سوف يبقيان حتى بعد إنتهاء العملية الناشئة له، وبإمكان أي عملية أخرى استخدامه عن طريق إسمه المسنود له.

هناك العديد من القواعد الخاصة بتسمية الملفات والتي تعتمد على نوعية النظام. مثلًا، يعتمد نظام "إم إس دوس"– وهو من أوائل أنظمة التشغيل التي عرفها مستخدمو الحواسيب– في تسمية الملفات على إعطاء كل ملف اسم لا يتجاوز الثمانية أحرف والتي قد تكون حروف، أو

هجين ما بينها وبين الأرقام والرموز، كما أنه ليس حساسًا لحالة الحرف، أي لا يُفرِّق بين الحروف الكبيرة والصغيرة عند التسمية.

بالمقابل، يُمكن في نظام 'ويندوز'تسمية الملفات بأكثر من ثمانية أحرف تصل في بعض الأحيان حتى إلى 255 حرفًا، وهو كذلك لا يفرق بين حالة الحرف. في حين تُفرق بعض من الأنظمة الأخرى مثل، نظام 'يونكس' بين حالة الحروف الصغيرة والكبيرة، وهو يستخدم في التسمية 254 حرفًا. فمثلًا، الأسماء file.c، وFILE.c، وكذلك File.c، هي مسميات لثلاث ملفات مختلفة ولا تُعبر عن نفس الاسم.

يُلحق في العادة بالملفات جزء يُدعَى بالامتداد، مع ملاحظة أنه قد يكون أحيانًا غير مطلوب لمعظم الملفات، ولكنه أكثر راحة للإشارة إلى محتوى الملف أو نوعه، هذا الامتداد قد يتكون من حرف أو حرفين شرط ألَّا يتجاوز في الأغلب الأربعة أحرف، أيضًا قد يكون هناك أكثر من امتداد كما هو الحال في الاسم file.c.Z، حيث يرمز الحرف Z هنا إلى أن هذا الملف هو ملف مضغوط باستخدام أحد برامج الضغط. يُوضح الجدول 1.6 بعض من الامتدادات الشائعة وكذلك معانيها.

الجدول 6. 1: بعض من الامتدادات الشائعة في مجال الحاسوب.

الامتداد	المعنى
.c	للدلالة على أن هذا الملف مصدري مكتوب بلغة السي.
.txt	للدلالة على أن هذا الملف يُمتل ملف نصي.
.zip	للدلالة على أن هذا الملف مضغوظ.
.docx	للدلالة على أن هذا الملف يُمتل ملف نصي مكتوب ببرنامج معالج النصوص.
.exe	للدلالة على أن هذا الملف يُمتل نسخة تنفيذية.
.bat	للدلالة على أن هذا الملف نصي قابل للتنفيذ.
.html	للدلالة على لغة توصيف النص التشعبي.
.bak	للدلالة على أن هذا الملف هو نسخة احتياطية.
.pdf	للدلالة على تنسيق المستندات المحمولة.

استنادًا على امتداد الملف، يمكن للمستخدم معرفة نوع الملف ومن ثم اختيار البرنامج المناسب للتعامل مع هذا الملف، أمّا بالنسبة لنظام التشغيل، فهو يُمكِّنه من تحديد القواعد

الخاصة بكيفية تنظيم وحدات الملف وتفسيرها بطريقة مجدية.

2.2.6 سمات الملف

كما ذكرنا آنفًا، عند إنشاء أي ملف يُسند له اسم خاص به، لكي يُرجع إليه من خلاله. إضافةً إلى ذلك يُلحق نظام التشغيل بعض من السمات الخاصة بكل ملف، وهي عبارة عن بيانات أولية مرتبطة بملفات الحاسوب تُحدد سلوك نظم الملف من حيث منح أو رفض حقوقًا محددة لكيفية وصول كل من المستخدم أو نظام التشغيل إلى هذه الملفات مثل، سمات الحماية، وأذونات الوصول، والقراءة، والأرشفة، وتواريخ وأزمنة إنشاء الملفات، وكذلك أحجامها. يوضح الجدول 6. 2 بعض من هذه السمات الخاصة بالملفات والتي تتغير من نظام إلى آخر.

الجدول 6. 2: بعض من السمات الشائعة في أنظمة التشغيل والخاصة بالملفات.

السمة	المعنى
الأرشيف	تسمح بالنسخ الاحتياطي وأرشفت الملف.
مخفي	تسمح بإخفاء الملف عند استعراض محتويات وسائط التخزين.
نظام	تعني أن الملف يتبع ملفات النظام.
التنفيذ	تسمح بتنفيذ الملف من قبل المستخدمين أو نظام التشغيل.
الكتابة	تسمح بكتابة الملف وإجراء تعديلات عليه.
القراءة فقط	تسمح فقط بقراءة الملف، أي من دون أي تعديلات.
الحماية	تُحدد من له حق الوصول إلى الملف وإلى أي مدى.
كلمة السر	تُبين أن هناك حاجة لاستخدام كلمة السر.
ناشئ الملف	تُعرف ناشئ الملف.
مالك الملف	تُحدد مالك الملف.
زمن إنشاء الملف	تُوضح زمن إنشاء الملف.
زمن آخر تعديل	تُوضح زمن آخر تعديل للملف.
زمن آخر وصول	تُوضح زمن آخر وصول للملف.
الحجم الحالي	تُوضح الحجم الحالي للملف بالبايت.
الحجم الأقصى	تُوضح الحجم الأقصى والمتوقع لنمو الملف بالبايت.
النوع	تُحدد نوع الملف.

تُعد سمات الملفات شرطًا ضروريًا يُلحق بالملفات أو المجلدات والتي من الممكن ضبطها أو إلغاؤها، أي إيقاف تشغيلها في أي وقت، فهي لا تُؤثر في الواقع على الملفات ولا على المجلدات أو تُغير فيها عند تطبيق هذه السمات عليها أو إلغاؤها، إلَّا إنَّها تُساهم في جعل نظام التشغيل والبرامج الأخرى تتعامل بطرق مختلفة مع الملفات والمجلدات.

3.2.6 أنواع الملفات

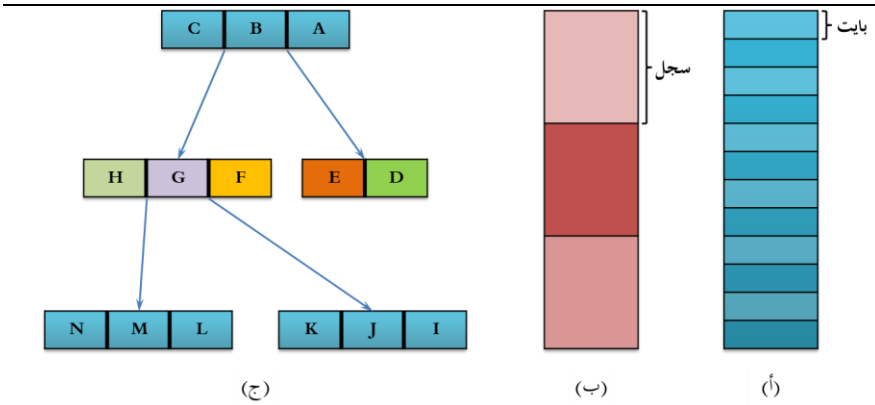
تدعم نظم التشغيل العديد من الملفات المتنوعة بتنوع البرامج والتطبيقات الخاصة بها، بالتالي يمكن تقسيم الملفات إلى المجموعات التالية:

- الملفات العادية أو الملفات المنتظمة: وهي تُمثل فئة الملفات التي أُنشئت بواسطة المستخدم وتحتوي على بياناته مثل الملفات الثنائية أو تلك التي تحتوي على نصوص، أو قاعدة بيانات، أو صور، أو أي نوع آخر من البيانات التي تخص المستخدم. لذلك يُمكن القول أن الملفات العادية تُستخدم لتخزين المعلومات الخاصة ببرامج المستخدم بحيث يمكنه إجراء عمليات مختلفة عليها. مثل، إضافة، أو تعديل، أو حذف أو حتى مسح الملف بأكمله.
- ملفات المجلد: وهي نوع خاص من الملفات لا يحتوي على أي برامج نصية، أو برامج قابلة للتنفيذ، وإنّما يُمثل فئة الملفات الموجودة في مجلد محدد، فهو يحتوي على قائمة بأسماء الملفات، وأسماء المجلدات، والمعلومات الأخرى المتعلقة بها مثل، الموقع على الجهاز، والتخزين، وحجم الملفات، وملكيتها. مثال ذلك، مجلد يُسمى نظام التشغيل ويحتوي على عدة ملفات تخص هذا النظام. بالتالي جميع هذه الملفات تُعرف بملفات المجلد.
- الملفات الخاصة أو ملفات الجهاز: وهي تُمثل فئة الملفات التي أُنشئت بواسطة النظام والضرورية لتشغيله. هذه الملفات تنقسم إلى نوعين:
  - الملفات الحرفية الخاصة: والتي يُتعامل فيها مع البيانات حرفًا بحرف كما في حالة وحدات الإدخال والإخراج الحرفية مثل، الطابعات، ولاقطات الصوت.
  - الملفات المقطعية الخاصة: والتي تُخزن فيها البيانات ويُتعامل معها ككتل أو مقاطع، كما هو الحال مع الأقراص، والأشرطة الممغنطة.

4.2.6 بنية الملفات

تأخذ بنية الملفات عدة حالات تبعًا لأنواع الملفات، بحيث تُبنى وفقًا لتنسيق محدد يفهمه نظام التشغيل، والذي يتطلب أن تكون البنية مُحددة، أي يمكنه تحديد مكان تحميل الملف في الذاكرة ومَعرفة موقع أول جزء منه. تمتد هذه الفكرة في بعض من نظم التشغيل لتشمل كل من مجموعة البنيات التي يدعمها النظام، ومجموعات العمليات الخاصة بالتعامل مع ملفات هذه البنيات. من ناحية أخرى، إنّ تعدد البنيات التي قد يدعمها نظام التشغيل تعني في الواقع إرهاقه بمزيد من الشفرات الداعمة لها. لذلك تدعم بعض من هذه الأنظمة الحد الأدنى من هذه البنيات مثل، 'يونكس'، و'إم إس دوس'.الشكل 6. 2 يُوضح ثلاث بنيات شائعة في نظم الشغيل.

1. بنية سلسلة الخانات الثمانية: يُخزن الملف بهذه الطريقة والموضحة في الشكل 6. 2–أ على هيئة سلسلة من الخانات الثمانية، بحيث تُعنون كل خانة لوحدها بإزاحة من بداية أو نهاية الملف. تتميز هذه البنية ببساطتها، إلَّا إنَّها لا تسمح لنظام التشغيل بتوفير إمكانيات جيدة لمعالجة الملفات، فنظام التشغيل لا يهتم بما هو موجود في الملف، لأن ما يراه هو تيار الخانات الثمانية والتي يُحدد معناها بواسطة البرامج على مستوى المستخدم. تُعتبر هذه الطريقة الأكثر شيوعًا في نظامي 'ويندوز'، و'يونكس'، وأغلب النظم الحديثة.
2. بنية سلسلة السجلات: يتألف الملف في هذه الطريقة كما هو موضح في الشكل 6. 2–ب من سلسلة من السجلات من نفس النوع وبطول ثابت، بحيث تكون لها تركيبية داخلية خاصة تجعل قراءة أو كتابة البيانات تتم في صورة سجل كامل، بمعنى أن العمليات المطبقة على الملف تتم على مستوى السجلات، لغرض تسريع عملتي القراءة والكتابة مقارنة بالطريقة السابقة.
3. بنية الشجرة: يُوضح الشكل 6. 2–ج هذه البنية الشجرة والتي يتكون فيها الملف أساسًا من سجلات مرتبة في صورة شجرة، والتي ليس من الضروري أن تكون بنفس الطول ولكل منها حقل مفتاح مُخصص له موقع ثابت في السجل يُستخدم في تسريع عملية البحث عن سجل معين. تُعتبر هذه الطريقة من الطرق المستخدمة بكثرة في نظام تشغيل الحواسيب المركزية.



الشكل 6. 2: بنيات الملفات الشائعة: أ) سلسلة الخانات الثمانية– ب) سلسلة السجلات– ج) الشجرة.

5.2.6 عمليات الملفات

أشرنا سابقًا بأن الأهداف الأساسية من الملفات تتمثل في حفظ البيانات بشكل دائم وتسهيل عمليات استخدامها والرجوع إليها في أثناء الحاجة. لذلك يتحتم على نظام التشغيل توفير بعض من الأوامر لإدارة الملفات حتى يتمكن المستخدم من التعامل معها بشكل صحيح، كتلك الأوامر المتعلقة بإنشاء الملفات، وكتابتها، وقراءتها، وإعادة ضبط موضعها، وحذفها، واقتطاعها وغيرها من العمليات الأخرى. يعرض الجدول 6. 3 بإيجاز مجموعة من العمليات الشائعة الاستخدام في التعامل مع الملفات.

الجدول 6. 3: بعض من عمليات الملفات الأكثر شيوعًا.

العملية	المعنى	الوظيفة
Creating a file	إنشاء ملف	إنشاء أنواع مختلفة من الملفات بطرق مختلفة.
Writing a file	كتابة ملف	كتابة بيانات في الملف، قد يكون في نهايته أو في أي موقع يتم تحديده من قبل المستدعي داخل الملف.
Reading a file	قراءة ملف	قراءة بيانات من الملف وذلك بداية من الموقع المحدد من المستدعي.
Deleting a file	مسح ملف	مسح الملف عند إنتهاء الحاجة منه، لتوفير مساحة خالية

العملية	المعنى	الوظيفة
		داخل القرص.
Truncating a file	اقتطاع ملف	حذف محتويات الملف مع الاحتفاظ بالسمات كما هي.
Opening a file	فتح ملف	فتح الملف قبل استخدامه.
Closing a file	إغلاق ملف	إغلاق الملف لتحرير مساحة الجدول الداخلية.
Appending a file	إلحاق الملف	إلحاق (أي، إضافة) البيانات فقط في نهاية الملف.
Seeking a file	تحديد مكان	تُستخدم في الوصول العشوائي، لأجل تحديد المكان الذي سوف تبدأ منه القراءة أو تتم فيه الكتابة.
Renaming a file	إعادة تسمية	إعادة تسمية ملف موجود.
Get attribute	استخلاص سمة	التعرف على سمات الملفات حتى يتسنى القيام بالمهام على الطريقة الصحيحة.
Set attribute	ضبط سمة	تعيين أو ضبط سمات الملف حتى بعد إنشائه.
Moving a file	نقل ملف	نقل ملف من مكان إلى آخر، مع إمكانية تغيير اسمه اختياريًا في نفس الوقت.
Copying a file	نسخ ملف	نسخ ملف في نفس المكان بمسمى جديد أو في مكان آخر، مع إمكانية تغيير اسمه اختياريًا في نفس الوقت.

6.2.6 آليات الوصول إلى الملفات

بعد أن تُحفظ الملفات في الذاكرة الثانوية، هناك حاجة إلى الوصول إليها لغرض تطبيق عدة عمليات عليها منها قراءة محتوياتها أو الكتابة فيها. بالتالي تُشير آلية الوصول إلى الملفات إلى الطريقة التي يمكن بها الوصول إلى سجلات الملف، وهو ما يمكن تحقيقه من خلال عدة طرق تتمثل في التالي:

- الوصول المتسلسل.
- الوصول المباشر (العشوائي).

- فهرسة الوصول المتسلسل.

1. الوصول المتسلسل: هو الوصول إلى محتويات الملف بشكل تسلسلي، متمثلًا في قراءة كافة الخانات الثمانية أو السجلات المكونة للملف من البداية إلى النهاية وبنفس الترتيب التسلسلي لهذه المكونات، دون وجود إمكانية قفز أي جزء، أي أن المعلومات الموجودة في الملف تُعالج بالترتيب، واحدة تلو الأخرى. لذلك يحتفظ نظام التشغيل بمؤشر يشير مبدئيًا إلى العنوان الأساسي للملف وبعد قراءة الجزء الأول منه تزداد قيمته بمقدار جزء واحد لتتم قراءة الجزء التالي، وتستمر هذه العملية حتى نهاية الملف. طريقة الوصول هذه هي الطريقة الأكثر بدائيةً، لكنها الأكثر استخدامًا بسبب حقيقة أن معظم الملفات (الملفات النصية، والصوتية، وملفات الفيديو، وغيرها) تحتاج إلى الوصول التسلسلي، والذي هو مريح للغاية وخصوصًا عندما يتمثل وسيط التخزين في الشريط الممغنط. تتميز هذه الطريقة ببساطة التنظيم، وسهولة قراءة الملفات. إلَّا إنَّها بطيئة للغاية، وتحديث أي سجل سيكون أمرًا صعبًا وسيستغرق وقتًا طويلًا، كما أنها تتطلب أن يكون لكافة سجلات الملف نفس الحجم.
2. الوصول المباشر (العشوائي): بظهور الأقراص ظهر معها نوع آخر من الوصول عُرف باسم الوصول المباشر أو العشوائي. هذا الوصول أتاح فرصة الوصول إلى محتويات الملف وقراءة الخانات الثمانية أو السجلات بطريقة عشوائية لا تشترط الترتيب، أي أن الوصول المباشر لا يشترط تسلسل محتويات الملف ولا أن تكون في مواقع متجاورة على وسيط التخزين، الأمر الذي تطلب توفر عنوان خاص لكل سجل داخل الملف، لكي يتم من خلاله الوصول المباشر للقراءة أو الكتابة.

إن استخدام هذه الطريقة يتناسب جدًا مع الكثير من التطبيقات وخصوصًا في مجال قواعد البيانات، لأن الوصول التسلسلي سيكون بطيئًا للغاية وغير فعّال في مثل هذه الحالات. يُعتبر نظام حجز التذاكر خير مثال على ذلك، فعندما يرغب مسافر في حجز تذكرة على رحلة معينة فإنه ليس من الضروري قراءة كل الرحلات الأخرى.

من مزايا هذا الأسلوب القدرة على قراءة البيانات أو كتابتها بشكل مباشر بدءًا من أي موقع في الملف، وهو ما يترتب عنه سرعة الوصول نظرًا لتحديد الموقع بدقة، هذا الأمر يترتب عليه أيضًا سرعة تحديث الملفات. ولكن، يُعتبر هذا التنظيم معقد نوعًا ما إذا ما قورن بالوصول المتسلسل نتيجةً لنفقات الفهرسة والتي ستُقلل من إجمالي مساحة التخزين.

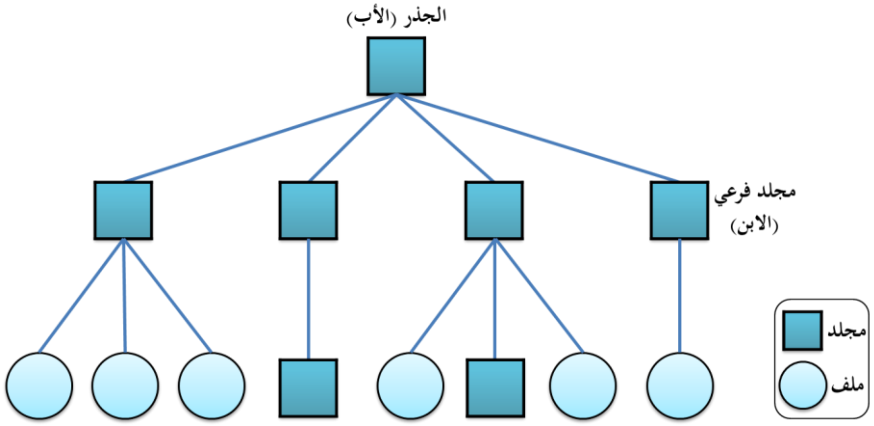
3. فهرسة الوصول المتسلسل: ظهر هذا النوع من الوصول للجمع بين كل من الوصول التسلسلي والوصول المباشر لأجل الاستفادة من مزاياهما. فهو يسمح بالوصول إلى السجلات تتابعيًا (بنفس الترتيب) أو عشوائيًا باستخدام الفهرس والذي يُخزن في الملف ويُقرأ في الذاكرة عند فتحه. في فهرسة الوصول المتسلسل، تُخزن السجلات بشكل عشوائي مرفقة معها مفاتيح فهرسة أساسية، هذه المفاتيح إما أن تكون أبجدية أو عددية يتم بواسطتها ترتيب السجلات والوصول إليها.

تتمتع آلية فهرسة الوصول المتسلسل بسرعة وسهولة البحث في قاعدة البيانات الضخمة بشرط تنظيم جدول الفهرس بشكل صحيح، وبإمكانية إدراج السجلات في منتصف الملف، كما أنها تدعم الاسترجاع الجزئي للسجلات، لكنها بحاجة إلى وجود مساحة إضافية في القرص لتخزين قيمة الفهرس، وبحاجة إلى مفاتيح فريدة وإعادة تنظيم دورية عند إدراج سجلات جديدة. أخيرًا، إذا ما حُذِف أي سجل، فإنه ينبغي تحرير المساحة المستخدمة من قبله وإلَّا فسينخفض أداء قاعدة البيانات.

3.6 المجلدات

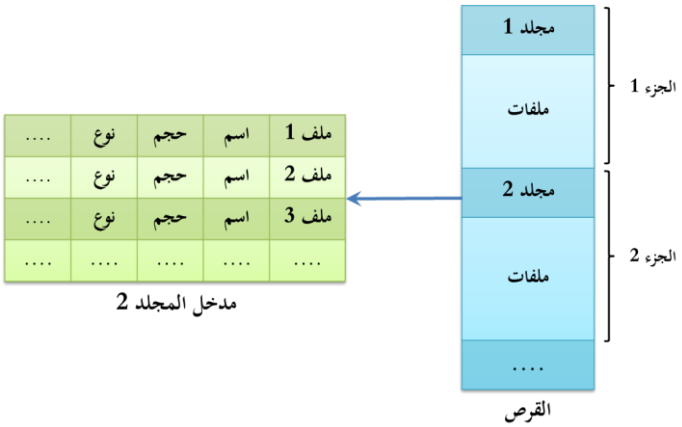
تُوفر نظم الملف ما يُسمي بالمجلدات أو الأدلة من أجل تتبع حفظ الملفات والتي تُعتبر في حد ذاتها ملفات عند كثير من الأنظمة. فالمجلد عبارة عن مكان يُخزن به الملفات ذات الصلة بما في ذلك السمات الخاصة بكل ملف، وذلك لغرض تنظيمها وتسهيل عملية الوصول إليها من خلال إجراءات إدارة الملفات المختلفة. تُدار معظم هذه المعلومات– خاصة تلك المتعلقة بالتخزين– بواسطة نظام التشغيل.

يُمكن إنشاء عدة مجلدات داخل القرص الثابت لغرض تنظيم عملية حفظ الملفات، كما يُمكن إنشاء عدة مجلدات داخل مجلد آخر تُعرف بالمجلدات الفرعية لينشأ عن ذلك شجرة المجلدات ومفهوم المجلد الأب والمجلد الابن، كما هو مبين في الشكل 6. 3، والذي يُوضح الهيئة العامة لشجرة المجلدات.



الشكل 6. 3: البنية العامة لشجرة المجلدات.

ولأنّ لدى كل ملف مجموعة من السمات التي يجب أن تُحفظ معه، نجد أن كل مجلد يحتوى على عدد من المداخل وفقًا لعدد الملفات التي يحتويها، يتكون كل مدخل من مجموعة من السمات الخاصة بكل ملف، كما هو مبين في الشكل 6. 4. من مزايا استخدام المجلدات الكفاءة من حيث المساعدة في إيجاد موقع الملف بشكل أسرع، كما أنها تسمح بتسمية ملفات مختلفة بنفس الاسم سواءً لنفس المستخدم أو لمستخدمين مختلفين، أيضًا تُوفر إمكانية تجميع الملفات حسب الخصائص مثل، تجميع برامج الألعاب، أو برامج لغة السي، ... إلخ.



الشكل 6. 4: عرض المجلد كملف يحتوي على بيانات التعريف الخاصة بمجموعة من الملفات.

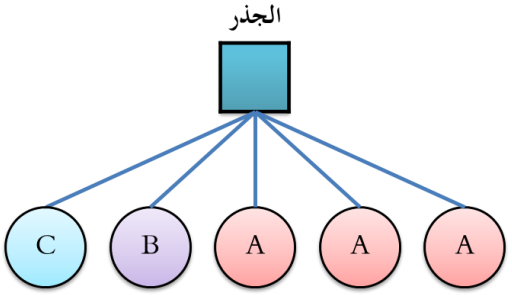
1.3.6 بنية المجلد

للمجلد عدة بنيات يمكن أن يستخدمها المستخدم تتدرج ما بين البسيطة والمعقدة، والتي يستعرض هذا القسم مجموعة منها.

1. بنية المجلدات ذات المستوى الواحد: تُعتبر هذه البنية من أبسط بنيات المجلدات المستخدمة في الأجيال الأولى من الحواسيب الشخصية، ولا زالت تُستخدم في بعض من الأجهزة المدمجة البسيطة مثل الكاميرات الرقمية. تتمثل هذه البنية في وجود مجلد واحد يحتوى على جميع الملفات الخاصة بجميع المستخدمين مما يجعل من السهل دعمها وفهمها، كما هو موضح في الشكل 6. 5. في هذا الشكل هناك ثلاثة مستخدمين، أحدهم يمتلك ثلاثة ملفات.

ونظرًا لأن هذه الطريقة تسمح لجميع المستخدمين بالتشارك في نفس المجلد، فإنه يتحتم عليهم عدم تسمية الملفات بنفس الاسم حتى لا يحدث تضارب قد يجعل النظام غير قادر على العمل. وهو ما يُمثل في واقع الأمر عيب يجعل المستخدم يغط الطرف عن هذه الطريقة، ويُحتم على نظم الملف إيجاد طرق أخرى لتفادي هذا العيب.

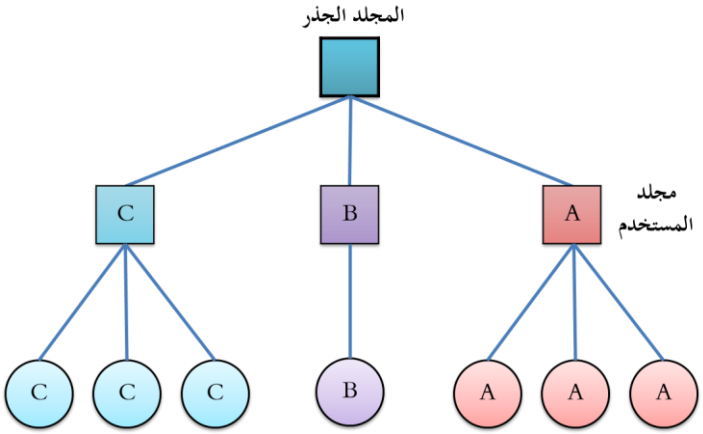
تتمتع هذه البنية بسرعة البحث وخصوصًا إذا كان حجم المجلد صغير، وبسهولة تنفيذها وإجراء العمليات مثل الإنشاء، والحذف، والتحديث. ولكن من سلبياتها صعوبة تجميع نفس النوع من الملفات حسب الحاجة، وعدم القدرة على تسمية الملفات بنفس الاسم.



الشكل 6. 5: بنية المجلدات ذات المستوى الواحد.

2. بنية المجلدات ذات المستويين: لتفادي عيوب البنية السابقة، أُجريت تحسينات عليها تتمثل في تخصيص دليل منفصل لكل مستخدم، كما هو موضح في الشكل 6. 6. في هذه

البنية، أصبح بالإمكان تسمية الملفات بنفس الاسم، ولكن لمستخدمين مختلفين. وللوصول إلى هذه الملفات يُستخدم اسم المسار الكامل، مثل /اسم المستخدم/اسم المجلد/.



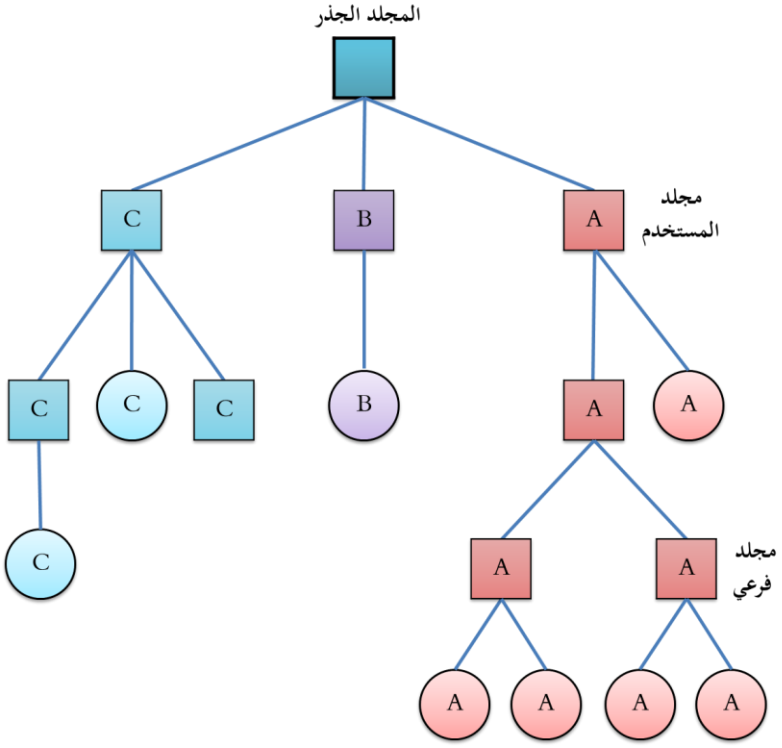
الشكل 6. 6: بنية المجلدات ذات المستويين.

من مزايا هذه البنية أنه أصبح البحث عن الملفات أكثر سهولة بسبب اسم المسار، وكذلك أصبح بالإمكان تجميع ملفات كل مستخدم في مجلد خاص به، لكنها لا زالت لا تسمح للمستخدم بمشاركة الملفات مع مستخدمين آخرين أو تجميع ملفين من نفس النوع معًا في نفس مجلد المستخدم الواحد.

3. بنية شجرة المجلدات: من خلال ملاحظة الشكل 6. 6 نجد أنّ بنية المجلدات ذات المستويين فعلًا قد حلت مشكلة التضارب في تشابه الأسماء، إلَّا إنَّها ما زالت تُعانى من مشكلة عدم وجود إمكانية تسمح للمستخدمين بتجميع ملفاتهم بطريقة منطقية. بمعنى تجميع الملفات ذات العلاقة الواحدة داخل مجلد واحد. فعلى سبيل المثال، باستخدام هذه الطريقة لا يمكن فصل الملفات الخاصة بالطلبة عن الملفات الخاصة بالأساتذة، وهكذا. لذلك ظهر الاحتياج الواضح لشجرة المجلدات والتي تسمح للمستخدم بامتلاك أكثر من مجلد فرعي، كما هو موضح في الشكل 6. 7.

بنية شجرة المجلدات هي الأكثر شيوعًا واستخدامًا، فهي تحتوي على جذر رئيسي، ولكل ملف في النظام مسار فريد. من مزايا هذه البنية أنها قابلة للتطوير وخالية تقريبًا من احتمالية تضارب الأسماء، إلَّا إنَّ هذه الطريقة لا تسمح بمشاركة الملفات، وأحيانًا قد تكون غير

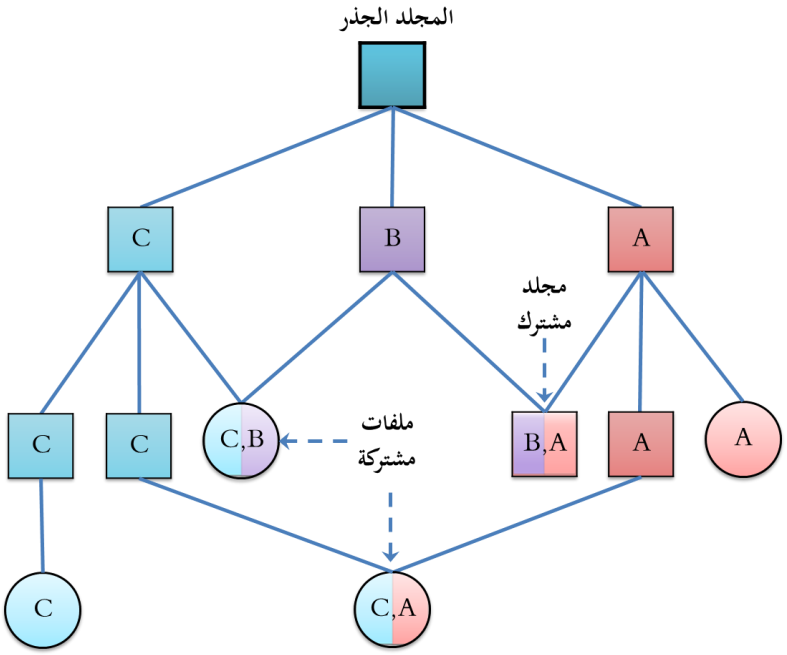
فعَّالة لكون الوصول إلى أي ملف قد يتطلب العبور عبر عدة مجلدات.



الشكل 6. 7: استخدام المجلدات الفرعية داخل شجرة المجلدات.

4. بنية مجلدات الرسم البياني اللاحلقي: تُمثل مشاركة الملفات بين المستخدمين مصدر قلق كبير في البنيات السابقة. لذلك أُستحدت بنية الرسم البياني اللاحلقي لمعالجة هذا القصور، كما هو موضح في الشكل 6. 8. في هذه البنية، قد يتشارك مجلد أو أكثر مع نفس الملف أو المجلد الفرعي عن طريق روابط إما منطقية أو مادية. من خلال تتبع هذا الشكل، يمكن ملاحظة تعدد مسارات الوصول إلى نفس الملف.

من فائدة بنية مجلدات الرسم البياني اللاحلقي هو استخدامه في الحالات التي يعمل فيها مستخدمان على مشروع مشترك ويحتاجان إلى مشاركة بعض من المجلدات الفرعية أو الملفات فيما بينهما، بحيث يرى كل مستخدم الإضافات أو التغييرات التي يُحدثها المستخدم الآخر.



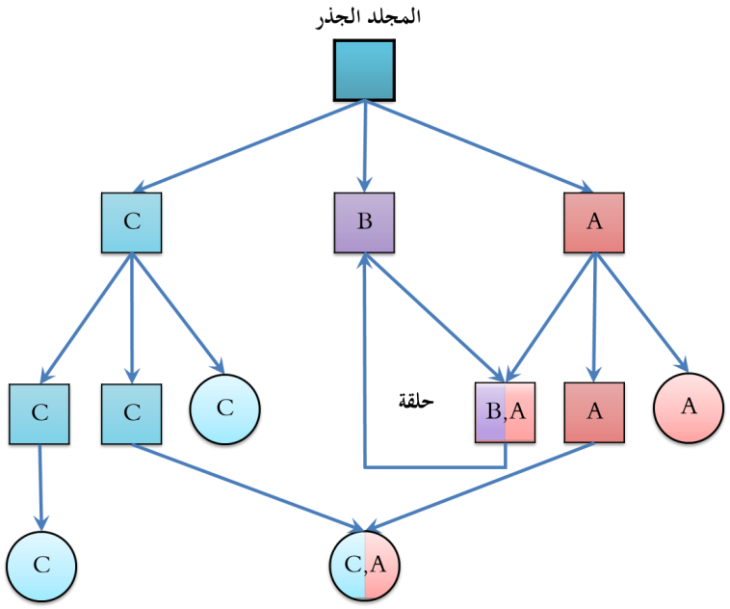
الشكل 6. 8: بنية مجلدات الرسم البياني اللاحلقي.

تتميز طريقة الرسم البياني اللاحلقي بالسماح لعدة مجلدات باحتواء نفس الملفات، وبسهولة البحث وذلك لتعدد مسارات الوصول إلى الملفات. إلَّا إنَّ المسارات المكررة قد تؤدي إلى تعقيد البنية ومهمة النسخ الاحتياطي، كما أن حذف الملفات المشتركة يمثل معضلة في هذه البنية. لذلك إذا كانت الروابط بينها منطقية فسيُحذف الملف ويبقى مؤشره، أما إذا كانت مادية فيتعين حذف الملف الفعلي فقط في حالة حذف جميع المراجع المرتبط به.

5. بنية مجلدات الرسم البياني العامة: رغم أن الرسم البياني اللاحلقي قدَّم آلية لمشاركة الملفات بين المستخدمين، إلَّا إنَّ هناك تخوف كبير من عدم المحافظة على الطبيعة المنظمة للأشجار نتيجةً لإضافة روابط المشاركة والتي قد ينشأ عنها حلقات داخل النظام الشجري. عليه إذا ما تكونت هذه الحلقات، فسيؤدي ذلك إلى تدمير بنية الأشجار وتَكَون بنية مجلدات الرسم البياني العامة والموضحة في الشكل 6. 9.

يسمح هذا النوع من بنية المجلدات بالمرونة العالية في عملية مشاركة الملفات، إلَّا إنَّها أكثر تكلفة من بنية المجلدات الأخرى. إضافة إلا ذلك، هناك مشكلتان رئيسيتان تتمثلان

في حساب الحجم الكلي الذي خُصِّص للملفات والمجلدات، وفي استعادة المساحة غير المستخدمة من القرص عندما تنقطع السبل للوصول إلى أيٍ من مكونات الرسم البياني بعد مسح رابطه.



الشكل 6. 9: بنية مجلدات الرسم البياني العامة.

2.3.6 عمليات المجلد

العمليات المستخدمة والمطبقة على المجلدات تختلف باختلاف النظام المستخدم وذلك من حيث الوظيفة التي تقوم بها كل عملية. وكعينة من هذه العمليات يُوضح الجدول 6. 4 بعض من العمليات المستخدمة في نظام 'يونكس'.

الجدول 6. 4: بعض من عمليات المجلدات.

العملية	المعنى	الوظيفة
Create	إنشاء	إنشاء مجلد جديد.
Delete	مسح	مسح المجلد عند إنتهاء الحاجة منه.
Opendir	فتح مجلد	فتح المجلد قبل استخدامه وذلك لعرض محتوياته.

العملية	المعنى	الوظيفة
Closedir	إغلاق مجلد	إغلاق المجلد بعد الإنتهاء من قراءته لتحرير مساحة الجدول الداخلية.
Rename	إعادة تسمية	تُستخدم هذه العملية لإعادة تسمية المجلد.
Search	بحث	البحث في المجلد الحالي.
Readdir	قراءة مجلد	تُرجع المدخل التالي في مجلد مفتوح.
Link	حلقة الوصل	ربط الملف لكي يظهر في أكثر من مجلد.
Unlink	فك ارتباط	إزالة ارتباط المجلد.

3.3.6 أسماء المسار

عندما يُنظم نظم الملف بناءً على بنيات المجلد المستعرضة في القسم السابق، ستكون هناك حاجة إلى آلية– ما– لتحديد أسماء الملفات والوصول إليها. تُعرف هذه الآلية باسم المسار وهو عبارة عن تسمية موقع مجلد الملف– في أثناء وجوده في نظام التشغيل– من خلال تتبع التسلسل الهرمي لشجرة المجلدات المعبّر عنه بسلسلة من الأحرف تُمثل مكونات المسار. يُفصل بين هذه المكونات بحرف محدد يختلف باختلاف النظام المستخدم إلَّا إنَّه غالبًا ما تُستخدم الشَرطة المائلة (’\‘) أو الشرطة المائلة العكسية (’/‘) أو النقطتين (’:‘). يُوضح الجدول 6. 5 أمثلة لمسارات مختلفة لبعض من نظم التشغيل.

الجدول 6. 5: أسماء مسارات مختلفة لبعض من نظم التشغيل.

النظام	اسم المسار
’ويندوز‘	C:\usr\docs\test.txt
’يونكس‘	/home/usr/docs/test.txt
’إم إس دوس‘	C:\usr\docs\test.txt
’ماكنتوش‘	HD:documents:test
’مَلتِكس‘	>usr>docs>test.txt
’سيمبيان‘	\usr\docs\test.txt

كما هو ملاحظ في الجدول 6. 5، يبدأ اسم المسار دائمًا من مجلد الجذر يعقبه أسماء

المجلدات الفرعية مفصولًا فيما بينها بأحد الحروف سابقة الذكر. مثلًا، في نظام 'ويندوز' بعد تسمية المجلد الجذر يُفصَل كل مجلد فرعي بالشَرطة المائلة، في حين في نظام 'يونكس' يُفصَل بينها بالشَرطة المائلة العكسية، أمّا في بيئة 'ماكنتوش' فيُصَل بين المجلدات برمز الشارحة.

من ناحية أخرى هناك طريقتان شائعتان لاستخدام اسم المسار، تتمثل الطريقة الأولى في المسار المطلق والذي ينطلق دائمًا من المجلد الجذر مرورًا بالمجلدات الفرعية ووصولًا إلى الملف. مثال ذلك المسار **\usr\docs\test.txt** والذي يعني أن المجلد الجذر ('\') يحتوي على المجلد الفرعي **usr** والذي يحوي المجلد الفرعي **docs** والذي بدوره يحتوي على الملف **test.doc**.

تُعرف الطريقة الثانية لاستخدام اسم المسار باسم المسار النسبي، وهي تعتمد على الاقتران بمفهوم مجلد العمل أو المجلد الحالي الذي يُعيِّنه المستخدم، في هذه الحالة يبدأ المسار نسبةً إلى مجلد العمل وليس كما هو معهود في الطريقة الأولى. مثلًا، إذا كان مجلد العمل معرَّف في نظام 'يونكس' على النحو التالي: **usr/docs/test.doc** فإن استخدام أمر النسخ **cp test.doc test.c** سيقوم بنفس مهمة الأمر **cp /usr/docs/test.doc /usr/docs/test.c** عندما يُستخدم المسار المطلق. السؤال الذي يطرح نفسه الآن هو متى تُستخدم هاتين الطريقتين؟

تعتمد إجابة هذا السؤال على الحالة التطبيقية للبرامج. مثلًا، عندما يحتاج البرنامج إلى الوصول إلى ملف معين دون النظر إلى ماهية مجلد العمل، فإنه يجب عليه دائمًا استخدام اسم المسار المطلق. على سبيل المثال، يحتاج برنامج المدقق الإملائي إلى الوصول إلى **/usr/lib/dictionary** للقيام بعمله. لذلك استخدام اسم المسار المطلق يكون هو الخيار الأمثل، لأن برنامج المدقق لا يعرف في هذه الحالة ماهية مجلد العمل عند استدعائه. كما أنه من الممكن استخدام استدعاءات النظام لتغيير مجلد العمل في حالة ما إذا احتاج المدقق الإملائي إلى عدة ملفات من داخل المكتبة، أي من **usr/lib/**، لذلك هذا الأمر سيُحتم استخدام المسارات النسبية كبديلًا للمسار المطلق.

4.6 إرفاق تحميل نظام الملف وإلغائه

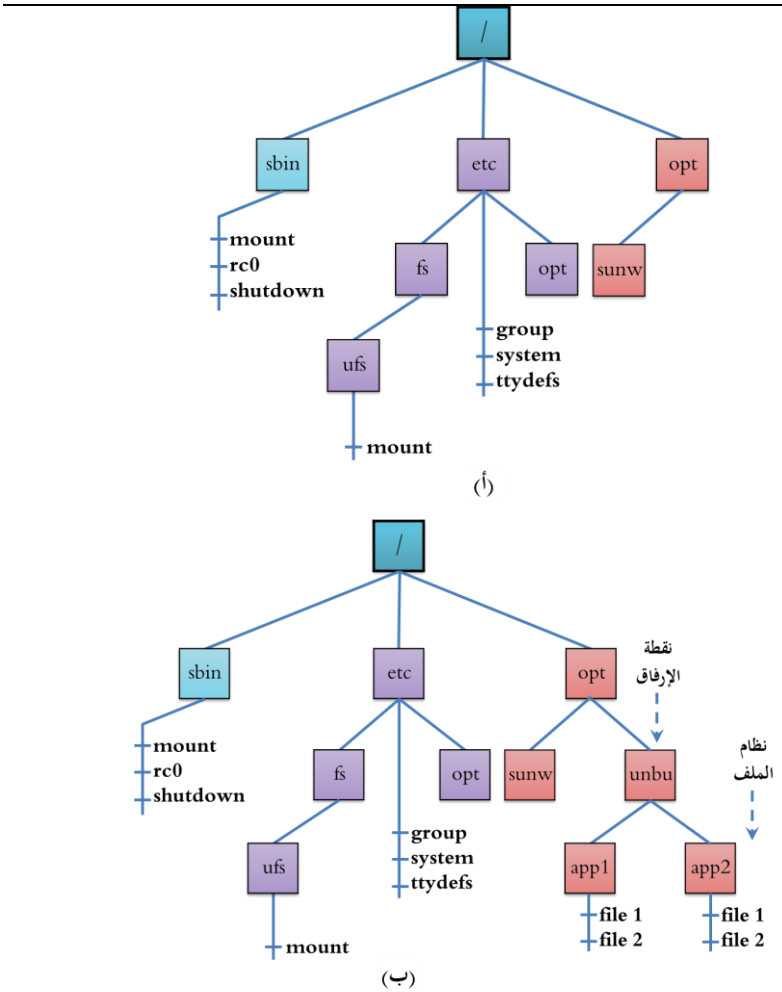
من المعروف أن الملفات التطبيقية كالملفات النصية لا يُمكن قراءتها والكتابة عليها إلا بعد إنشائها وفتحها، كذلك هو الحال مع نظام الملف والذي يجب إرفاقه وتثبيته أولًا في النظام، ومن ثم إتاحته للعمل عليه. مثلًا، نظام الملف (الجذر) هو نظام دائم التثبيت يمكن إلحاق أي نظام ملف آخر به والعمل عليه، كما يُمكن فصله عند إنتهاء الحاجة منه، تُعرف هذه العملية بإرفاق وإلغاء تحميل نظام الملف.

تختلف عملية إرفاق وإلغاء تحميل نظام الملف من نظام تشغيل لآخر، هناك أنظمه تشغيل لها تسلسل شجري يُمثل كل قسم في نظامها وتُرفق الملفات حسب تسلسل التخزين الموجود عند نقطة تُسمى نقطة الإرفاق، بينما يُوجد في بعض من الأنظمة الأخرى مثل، نظام 'لينكس' تسلسل شجري واحد فقط يضم جميع ملفات أدوات التخزين، والأقسام الموجودة في الجهاز ويستخدم نقطه الإرفاق لربط أي نظام ملف جديد.

على سبيل المثال يعرض الشكل 6. 10–أ نظام ملفات محلي، يبدأ من نظام ملفات الجذر (/) مرورًا بالمجلدات الفرعية **sbin**، و **etc**، و **opt**. بفرض أن هناك رغبة في الوصول إلى نظام ملفات محلي عن طريق نظام الملف **opt**/ الذي يحتوي على مجموعة من الملفات غير المرفقة.

للقيام بذلك، يجب أولًا إنشاء مجلد لاستخدامه كنقطة إرفاق لنظام الملف المطلوب تحميله، على سبيل المثال المجلد **opt/unbu**/. بمجرد إنشاء نقطة التحميل، يمكن تحميل نظام الملف (باستخدام الأمر **mount**، مما يجعل جميع الملفات والمجلدات الموجودة في **opt/unbu**/ متاحة، كما هو موضح في الشكل 6. 10–ب.

في نظام 'لينكس' لإضافه قرص مرن إلى الجهاز يجب إرفاقه أولًا من أجل استخدامه عن طريق الأمر التالي: **$mount /dev/fd0/mnt/floppy**، ولفك هذا الربط عند الإنتهاء منه يُستخدم الأمر **umount** على النحو **$umount /mnt/floppy**، بينما لا توجد صعوبة في نظام 'ويندوز' في تعريف أدوات التخزين الجديدة، لأنه بمجرد تثبيتها في الحاسوب سيتعرف عليها النظام تلقائيًا.



الشكل 6. 10: إرفاق نظام الملف: أ) نظام ملفات محلي– ب) تحميل نظام ملفات جديد.

5.6 تنفيذ نظم الملف

حاولنا في الأقسام السابقة النظر إلى الملفات من وجهة نظر المستخدم، أي إلى المواضيع التي تهمه من حيث تسمية الملفات وخصائصها، والعمليات المسموح بها عليها، بالإضافة إلى شجرة المجلدات، وإلحاق وفصل نظام الملف، بينما سنحاول في الأقسام التالية التعرف على وجهة نظر المصمم أو المنفذ لنظم الملف من حيث الكيفية التي تتم بها تخزين الملفات

والمجلدات، وكذلك الكيفية التي تُدار بها مساحة الأقراص من أجل استغلالها بكفاءة وموثوقية.

1.5.6 تنفيذ الملفات

إن أهم شيء في تنفيذ الملفات وتخزينها هو الكيفية التي تتبع بها الأجزاء الخاصة بكل ملف. في الواقع هناك عدة طرق مستخدمة لهذا الغرض، سنحاول في هذا القسم التعرض لبعض منها.

1.1.5.6 التخصيص المتجاور

تُعتبر هذه الطريقة من أبسط الطرق المستخدمة في تتبع الأجزاء الخاصة بكل ملف، فهي تتمثل في تخصيص أماكن متجاورة لتخزين الملف فيها على هيئة مجموعة من الأجزاء، بحيث إذا كان حجم كل جزء هو واحد كيلوبايت، فإن ملف حجمه خمسون كيلو سوف يحتاج إلى خمسين جزء متجاور. تتطلب هذه الطريقة مدخل واحد لكل ملف يُوضح جزء البداية وطول الملف. من هنا يمكن تعريف التخصيص المتجاور بواسطة عنوان أول جزء من الملف داخل القرص، وطوله معبرًا عنه بعدد الأجزاء المكونة له. بمعنى أن أي ملف يبدأ من الموقع **c** بطول **n** فسوف يشغل الأماكن **c**، **c + 1**، **c + 2**، ...، **c + n – 1**. الشكل **6. 11** يُوضح محتويات قرص به خمسة ملفات موزعة تبعًا لما هو مبين بمدخل المجلد.

تتميز هذه الطريقة ببساطتها في التنفيذ، وذلك لأن عملية تتبع الأجزاء الخاصة بكل ملف سوف تتقلص فقط في متابعة ومعرفة عنوان أول جزء في المجموعة. كما تتميز أيضًا بكفاءتها العالية لأنه بالإمكان قراءة محتوى الملف في عملية قراءة واحدة فقط.

لسوء الحظ، يحتوي نظام التخصيص المتجاور على عيبين رئيسيين هما:

- ضرورة معرفة حجم الملف مسبقًا، لكي يتسنى فيما بعد تخصيص العدد المناسب له من الأجزاء قبل إنشائه وهو ما يُعتبر صعب المنال.
- هذه الطريقة عرضةً للتفتت الخارجي لمساحة القرص بسبب الإنشاء والحذف المستمر للملفات، والتي ستسبب بدورها في ضياع لهذه المساحة. هذا يجعل من الصعب العثور على أجزاء متجاورة من مساحة القرص بطول كاف. هذا التفتت يمكن معالجته بإعادة نقل

الملفات من أجل توحيد المساحات الصغيرة، الحرة والمبعثرة داخل القرص، إلَّا إنَّ هذه العملية قد تستغرق وقتًا طويلًا للغاية، ولكن من الممكن تنفيذها بشكل أفضل عندما لا يكون النظام مشغولًا بعمل آخر.

محتويات القرص

11	10	9	8	7	6	5	4	3	2	1	0
23	22	21	20	19	18	17	16	15	14	13	12
35	34	33	32	31	30	29	28	27	26	25	24
47	46	45	44	43	42	41	40	39	38	37	36
59	58	57	56	55	54	53	52	51	50	49	48
...	70	69	68	67	66	65	64	63	62	61	60

مدخل المجلد

الملف	البداية (c)	الطول (n)
mail	0	5
list	8	8
fs	24	9
tree	38	11
os	62	6

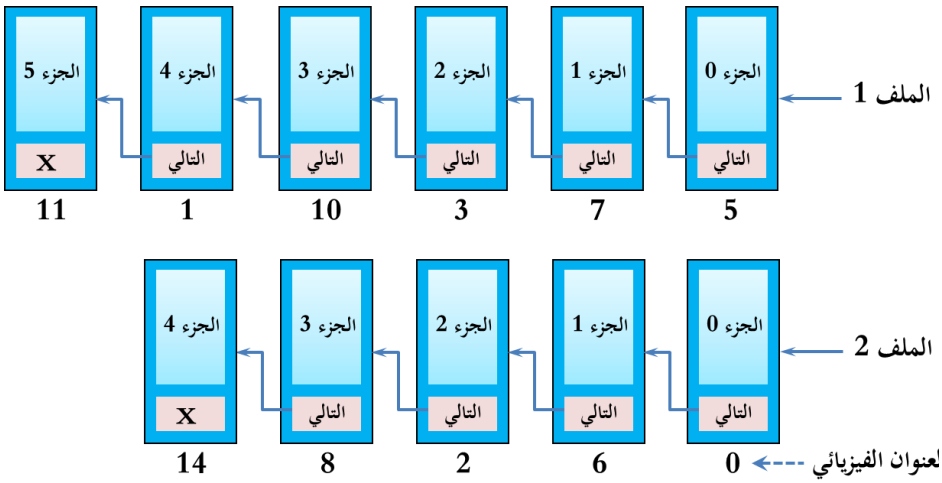
الشكل 6. 11: التخصيص المتجاور للملفات.

2.1.5.6 التخصيص باستخدام القوائم المرتبطة

قد يبدو للوهلة الأولى أن الحل الأمثل للتخلص من ظاهرة التفتت الخارجي يتمثل في عملية الضغط لتحرير مساحة إضافية على القرص، إلَّا إنَّه سوف لن يكون مجديًا في حالة ما إذا كان حجم القرص كبير، لذا وجب البحث عن طرق أخرى تختفي فيها مثل هذه العيوب. تتمثل إحدى هذه الطرق في استخدام القوائم المرتبطة في عملية التخصيص، وذلك من خلال تمثيل الملف على هيئة قائمة من العقد مرتبطة ببعضها عن طريق المؤشرات، كما هو موضح في الشكل 6. 12. تُقسم كل عقدة إلى جزئين، أحدهما بحجم كلمة ويُستخدم لحفظ مؤشر العقدة التالية في القائمة، بينما تُحفظ في الجزء الآخر البيانات نفسها.

في هذه الطريقة نحتاج فقط إلى معرفة عنوان أول عقدة في القائمة المَمَثِّلة للملف، وذلك لأن بقية العناوين سوف يتم التعرف عليها من خلال القائمة نفسها. يتكون الملف الأول في الشكل 6. 12 من ست عقد، تحمل العقدة الأولى منها العنوان الفيزيائي خمسة، تليها العقدة المعنونة بالعنوان سبعة، ثم العقد ذات العناوين ثلاثة، وعشرة، وواحد، أمَّا بالنسبة لآخر مكون لهذا الملف فيقع عند العنوان 11 والذي يحوي مؤشره العلامة X للدلالة على نهاية الملف الأول. الأمر نفسه بالنسبة للملف الثاني والذي يقع أجزاؤه عند العناوين 0، و6، و2، و8،

وأخيرًا 14.



الشكل 6. 12: التخصيص المرتبط عن طريق القوائم المرتبطة، تُشير العلامة X إلى نهاية القائمة.

باستخدام هذا النوع من التخصيص يمكن حذف وإضافة أي عقدة من وإلى القائمة بكل سهولة، بالتالي يُتيح هذا الأمر فرصة استغلال مساحة القرص بشكل جيد بحيث يمنع حدوث ظاهرة التفتت الخارجي، كما أنه لا يشترط الإعلان المسبق عن حجم الملف عند إنشائه لكي يُخصص له العدد المناسب من الأجزاء، وإنما يمكن للملف أن يُنشأ ويُخصص له العقد ببساطة حسب الحاجة.

بالمقابل، يُعاني التخصيص باستخدام القوائم المرتبطة من احتمالية حدوث التفتت الداخلي وخصوصًا في الجزء الأخير من الملف. كما أنه يدعم فقط الوصول التسلسلي للملفات، أي لن تكون هناك إمكانية للوصول المباشر لأي جزء. مثلًا، للوصول إلى الجزء السادس يجب البدء من بداية الملف والانتقال عبر الأجزاء باستخدام المؤشرات إلى أن يتم الوصول إلى هذا الجزء، ولأن هذه الطريقة تعتمد على استخدام المؤشرات بشكل كبير، فإن الحفاظ عليها وتتبعها يُساهم في تعقيدها، بالإضافة إلى أنه في حالة فقد أي مؤشر لأي جزء، سيؤدي إلى اقتطاع الملف.

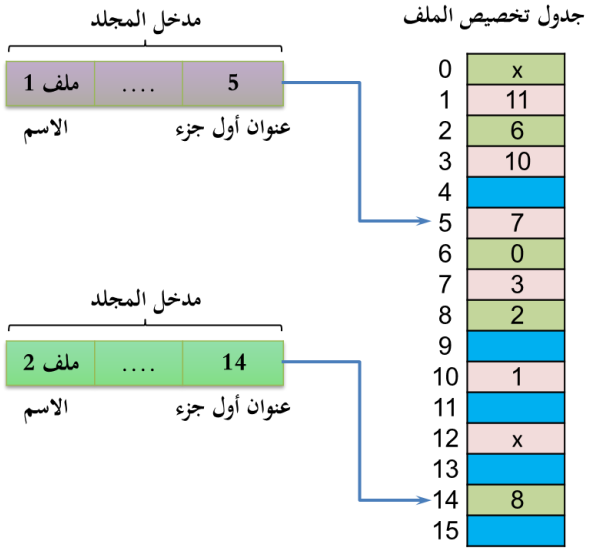
3.1.5.6 التخصيص باستخدام فهرسة القوائم المرتبطة

من خلال تتبع آلية التخصيص باستخدام القوائم المرتبطة، نجد أن جزء المساحة المحجوز للمؤشر سيُمثل مساحة ضائعة من إجمالي مساحة للقرص. مثلًا، إذا اقتطع المؤشر مساحة قدرها 8 خانات ثمانية من إجمالي مساحة الجزء والتي بفرض أنها تساوي 1024 خانة ثمانية، فسيتم استخدام 0.78٪ من القرص للمؤشرات بدلًا من المعلومات. أيضًا هذا يجعل حجم الجزء المتبقي 1016 خانة ثمانية وهو ما لا يتماشى مع قاعدة الأس المرفوع للقوة 2 والذي سيجعل حجم الجزء لا يتناسب مع البرامج، وذلك لكونها تقوم بقراءة وكتابة الأجزاء على أساس هذه القاعدة.

من هنا جاء مفهوم التخصيص باستخدام فهرسة القوائم المرتبطة والذي يتمثل في سحب المؤشرات من المساحات المخصصة لها داخل أجزاء الملف ووضعها داخل جدول في الذاكرة يُعرف باسم جدول تخصيص الملف، كما هو مبين في الشكل 6. 13. يُوضح هذا الشكل الكيفية التي سوف يكون عليها هذا الجدول من خلال تتبع الملفين المذكورين في الشكل 6. 12، نلاحظ أيضًا من خلال الشكل 6. 13 أنه عن طريق مدخل المجلد يمكن التعرف على اسم الملف وعنوان أول جزء منه والذي سيؤدي إلى التعرف على بقية مكوناته من خلال تتبع المؤشرات.

أُستخدمت هذه الطريقة في نظام 'إم إس دوس' ولا تزال مدعومة بالكامل من قبل كافة إصدارات 'ويندوز'، فهي تتميز بإمكانية الوصول العشوائي لمحتويات الملف، بالرغم من أنه يجب إتباع تسلسل المؤشرات للقيام بذلك، إلَّا إنَّ هذا التتبع يحدث بالكامل في الذاكرة، وبالتالي يمكن متابعتها دون الحاجة للوصول إلى القرص. كما يكفي أن يتواجد رقم واحد صحيح (يُمثل بداية القائمة) في مدخل المجلد لتحديد بقية عناصر الملف بغض النظر عن حجمه مثلما هو مستخدم في الطريقة السابقة.

العيب الأساسي لهذه الطريقة يتمثل في ضرورة وجود جدول تخصيص الملف بالكامل في الذاكرة بشكل دائم لجعله يعمل، مما سيجعله يشغل حيزًا من الذاكرة وخصوصًا إذا ما كان حجم هذا الجدول كبير.



الشكل 6. 13: فهرسة القوائم المرتبطة عن طريق جدول تخصيص الملف المحمَّل في الذاكرة.

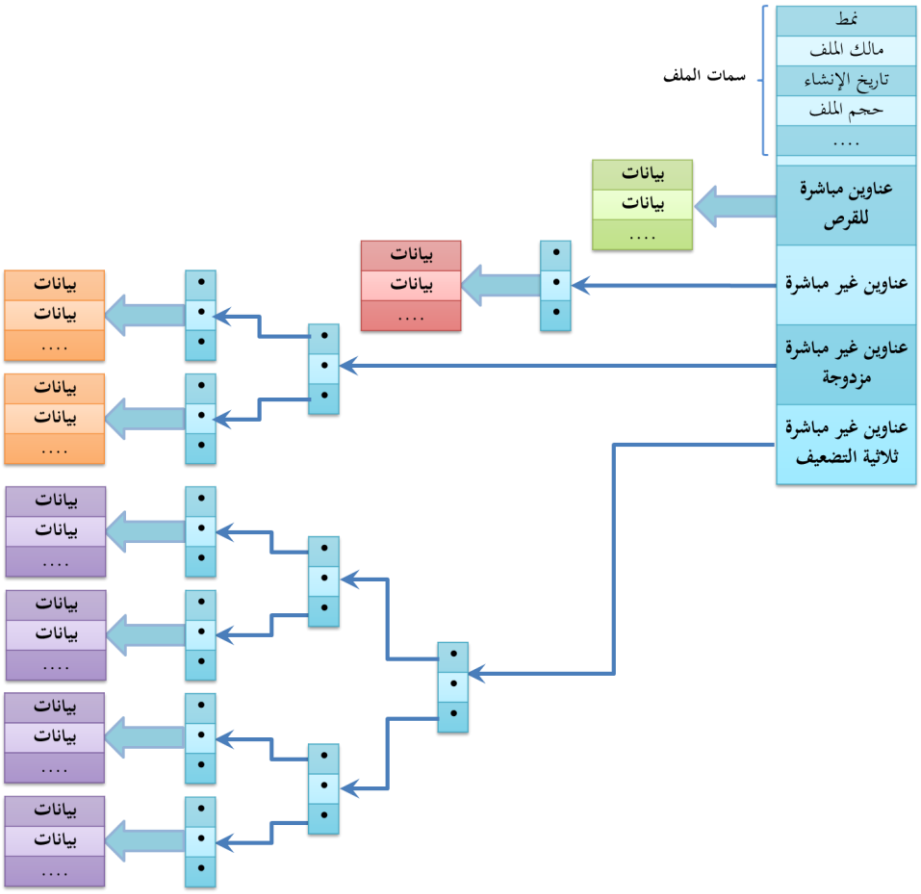
4.1.5.6 عقدة الفهرسة

يُفهرس نظام تشغيل 'يونكس' كل ملف بواسطة بنية بيانات تُعرف بعقدة الفهرسة، وهي عبارة عن جدول صغير يُنشأ عند إنشاء نظام الملف ويُلحق بكل ملف جديد، يحوي هذا الجدول سمات الملف مثل ملكية المستخدم، والمجموعة، ووضع الوصول (قراءة، كتابة، تنفيذ)، وحجم الملف، وجميع المعلومات المطلوبة لوصف بياناته، وعناوين أجزائه، وكذلك تخطيطه على القرص، كما هو موضح في الشكل 6. 14.

نلاحظ من خلال تتبع الشكل 6. 14 أنه بإمكان عقدة الفهرسة احتواء بعض من العناوين المباشرة للقرص داخل بنيتها لغرض عنونة الملفات الصغيرة (التي لا تزيد عن 12 جزء)، والتي سوف تُحمل من القرص إلى الذاكرة عند فتح الملف، ومع فرضية أن حجم الجزء هو 4 كيلوبايت، فمن الممكن الوصول إلى ما يصل إلى 48 كيلوبايت من البيانات مباشرةً.

أما عندما يكون حجم الملف أكبر من ذلك فإن عقدة الفهرسة تُوفر ثلاثة مؤشرات خاصة إضافية تُشير إلى عناوين إضافية تُستخدم لعنونة ملفات أكبر. يُعنون المؤشر الأول عناوين غير مباشرة، وهي عبارة عن كتل فهرسة لا تحتوي على أي بيانات لكنها تُمثل عناوين أجزاء للبيانات.

إذا لم تكفي هذه العناوين لتغطية حجم الملف، فسيُستخدم المؤشر الثاني والذي يُشير إلى عناوين غير مباشرة مزدوجة، تحتوي هذه العناوين على عنوان كتل فهرسة تحتوي على عناوين لكتل فهرسة أخرى تحتوي على مؤشرات إلى أجزاء البيانات الفعلية. أخيرًا، إذا كان حجم الملف كبير بحيث لا يمكن استعابه عن طريق العناوين غير المباشرة المزدوجة، فسيُلجأ إلى استخدام مؤشر العناوين غير المباشرة ثلاثية التضعيف، والذي يُضاعف مساحات العنونة بشكل كبير.



الشكل 6. 14: عقدة الفهرسة.

عندما يحاول المستخدم الوصول إلى الملف أو أي معلومات متعلقة بالملف، فإن رقم عقدة الفهرسة الخاص به يظل هو مفتاح العثور على هذه البيانات، لذلك على مستوى المستخدم يتم

التعامل مع اسم الملف للقيام بذلك، ولكن داخليًا يقوم نظام الملف باستخلاص رقم العقدة أولاً مع جدول عقدة الفهرسة ومن ثم استخدامه للوصول إلى عقدة الفهرسة المقابلة لهذا الملف والتي بدورها ستؤدي إلى التعرف على مكوناته، كما سيتم توضيحه في القسم التالي.

تتمثل مميزات عقدة الفهرسة في صغر مساحة الذاكرة التي يشغلها جدول العقدة عند فتح الملف المناظر له. فإذا افترضنا أن كل عقدة تحتاج إلى عدد **n** من الخانات الثمانية، وأن أقصى عدد للملفات التي يمكن فتحها في آنٍ واحد هو **m** فإن كمية المساحة المستغلة من قبل المصفوفة التي تستوعب عقد الملفات المفتوحة والذي يُحجز مسبقًا في الذاكرة هو **n*m** خانة ثمانية. بالمقابل، يُعتبر حدودية عدد العناوين المباشرة للقرص داخل بنية عقدة الفهرسة من أهم معوقات هذه الطريقة، إلَّا إنَّ استخدام العناوين غير المباشرة، وغير المباشرة المزدوجة، وثلاثية التضعيف يُساهم في حل هذه المعضلة، ولكن على حساب التعقيد وزيادة زمن الوصول.

2.5.6 تنفيذ المجلدات

لكي يُتعامل مع الملفات بشكل مثالي بعد إنشائها يجب حفظها أولًا على وسائط التخزين الثانوية مثل، الأقراص الصلبة، والضوئية، وغيرها، والتي أي، الملفات عادة ما يكون هناك الملايين، بل المليارات منها داخل الحاسوب. لذلك من الممكن استخدام وسيط التخزين في مجمله لنظام ملفات واحد، أو تقسيمه إلى عدة أجزاء بحيث يكون لكل جزء نظام منفصل.

يُعد التقسيم مفيدًا في تحديد أحجام أنظمة الملفات الفردية، أو في وضع عدة أنواع من الأنظمة على نفس الجهاز، أو في ترك جزء متاح للاستخدامات الأخرى، مثل المبادلة أو تخزين عدة أنظمة تشغيل مختلفة، الأمر الذي يُتيح للمستخدم تحميل أكثر من نظام تشغيل واحد على نفس الجهاز.

ولتنظيم عملية حفظ الملفات بشكل فعّال، تُخزن هذه الملفات في الغالب في مجلدات مرفقة بمعلومات حولها، بحيث تُحفظ هذه المعلومات في مداخل المجلدات وتشمل الاسم، والموقع، والحجم، والنوع، وغيرها من سمات الملفات. لذلك عند استخدام أي ملف، يجب فتحه أولًا لاستخلاص كافة المعلومات المتعلقة به والتي من أهمها الموقع، وآلية الوصول إلى أجزائه، وللقيام بذلك تستخدم نظم التشغيل اسم المسار (المعطى من قبل المستخدم) لتحديد

مدخل الدليل الخاص بالملف المراد فتحه وذلك لاحتوائه على كافة المعلومات المطلوبة، واعتمادًا على أي نظام مستخدم فقد تكون هذه المعلومات عبارة عن رقم يُمثل الجزء الأول (سواءً في القوائم الخطية أو المرتبطة) أو رقم عقدة الفهرسة. في جميع الأحوال تتمثل الوظيفة الرئيسية لنظم الملف في تحليل اسم الملف لتحديد كافة المعلومات الخاصة بإيجاد البيانات المتعلقة به.

في الواقع، هناك عدة طرق مستخدمة لتنفيذ المجلدات وتخزين بيانات الملف داخل كل مجلد، ولكن اختيار الآلية المناسبة قد يؤثر بشكل كبير على كفاءة نظام الملف، وأدائه، وموثوقيته. في العموم، تُصنف آليات التنفيذ بناءً على هيكلية البيانات المستخدمة والتي تنحصر في الطريقتين التاليتين.

1.2.5.6 تنفيذ المجلدات في صورة قائمة خطية

تتمثل أبسط طريقة لتنفيذ المجلدات في الاحتفاظ بجميع معلومات الملفات داخل المجلد على هيئة قائمة خطية، كما هو موضح في الشكل 6. 15. تتكون القائمة الخطية في هذا التصميم البسيط من مداخل الملفات الموجودة في المجلد بحيث يُناظر كل مدخل ملف منفصل، هذا المدخل يتضمن سمات مختلفة للملفات، مثل مالكه، ووقت إنشائه، كما يحتوي على اسم الملف، وعنوان أول جزء من أجزائه والذي يُستخدم من قبل نظام الملف في الوصول إلى الملف داخل القرص.

السمات

اسم ملف	المالك	نوعه	....	عنوان أول جزء
اسم ملف	المالك	نوعه	....	عنوان أول جزء
اسم ملف	المالك	نوعه	....	عنوان أول جزء
اسم ملف	المالك	نوعه	....	عنوان أول جزء
....	....	....	....	....

الشكل 6. 15: تنفيذ المجلدات في صورة قائمة خطية.

لتوضيح تفاصيل مدخل المجلد يمكن النظر إلى مدخل المجلد المستخدم في نظام 'إم إس دوس' والمبين في الشكل 6. 16. يتكون هذا المدخل من 32 خانة ثمانية موزعة على سمات الملف حسب ما هو موضح في هذا الشكل. يحتوي هذا المدخل على اسم الملف، وسماته، وكذلك عنوان أول جزء من مكوناته والذي يُستخدم كمؤشر لجدول تخصيص الملف الموضح في الشكل 6. 13.

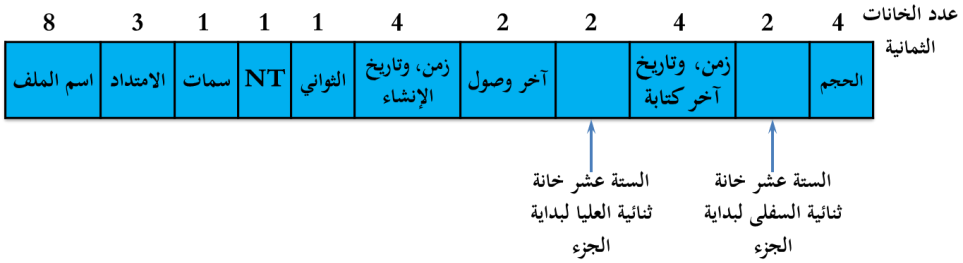
تنفيذ المجلدات بالنسبة لنظام الملف الخاص بالإصدار الأصلي لنظام التشغيل 'ويندوز 95' مطابق لنظام ملفات 'إم إس دوس'، بينما يدعم الإصدار الثاني منه ملفات بأسماء أطول وحجم أكبر. من ناحية أخرى يدعم نظام ملفات 'ويندوز 98' بنيتين لمدخل المجلدات. الأولى خاصة بنظام الملف الداعمة لجدول تخصيص الملف بطول 32 خانة ثنائية (FAT-32)، والثانية خاصة بالملفات ذات الأسماء الطويلة، في هذا القسم سنتعرض فقط للبنية الأولى.

الشكل 6. 16: مدخل مجلد نظام 'إم إس دوس'.

يُمثل المدخل المبين في الشكل 6. 17 مدخل المجلد الأساسي لنظام ملفات 'ويندوز 98'، والذي يُمثل امتداد لمدخل نظام ملفات 'إم إس دوس'. يحوي هذا المدخل خمسة حقول إضافية مكان الخانات العشر غير المستخدمة، بحيث يُستخدم حقل NT للتوافق مع نظام التشغيل 'ويندوز إن تي' من حيث عرض أسماء الملفات بشكل صحيح بناءً على حالة الحرف. كما يُوفر حقل 'الثواني' خانات إضافية لحفظ زمن الإنشاء بدقة تصل إلى 10 مللي ثانية، لكون هذه الدقة غير ممكنة في حالة استخدام فقط 16 خانة ثنائية. من الاختلافات الأخرى في هذه

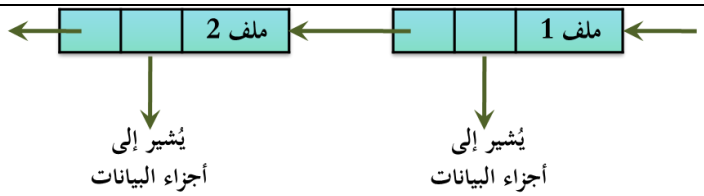
البنية استحداث حقل آخر وصول والذي يتتبع تاريخ آخر وصول إلى الملف. أخيرًا، الانتقال إلى نظام الملف ذو جدول تخصيص الملف بطول 32 خانة ثنائية جعل أرقام الأجزاء تتكون من 32 خانة ثنائية، لذلك هناك حاجة إلى حقل إضافي لتخزين 16 خانة ثنائية العلوية من عنوان جزء بداية الملف.

تتميز طريقة تنفيذ المجلدات في صورة قائمة خطية بسهولتها في البرمجة، إلَّا إنَّها تستغرق وقتًا طويلًا في التنفيذ من حيث البحث عن ملف، أو إضافة ملف جديد، أو تحديث ملف قديم، أو حذفه. مثلًا، لإنشاء ملف جديد، يجب التأكد أولًا من أن المجلد لا يحتوي على أي ملف يحمل نفس الاسم، بعدها يُضاف المدخل الجديد في نهاية المجلد، أمَّا في حالة الحذف فسيتم أولًا البحث عن الملف المعني داخل المجلد، ومن ثم تُحرر المساحة المخصصة له.



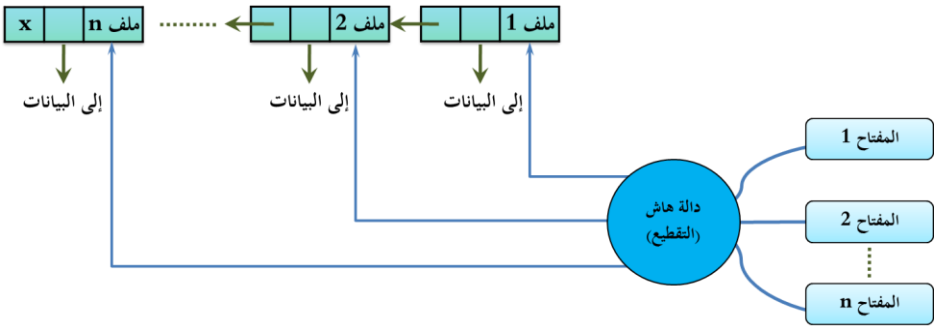
الشكل 6. 17: المدخل الأساسي لمجلد نظام 'ويندوز 98'.

من الطرق المستخدمة لتقليل متوسط زمن البحث، أو الوقت اللازم للإضافة، أو الحذف هو الحفاظ على القائمة الخطية مرتبة، إلَّا إنَّ ذلك سيزيد من تعقيدات عملية إنشاء الملفات وحذفها نظرًا لأنّ هذا الأمر يتطلب نقل كميات هائلة من بيانات المجلد من أجل الحفاظ عليه مرتب، أدى ذلك إلى الاستعانة ببنية القوائم المرتبطة في تنفيذ المجلدات، وذلك كما هو موضح في الشكل 6. 18. تُحفظ كافة مداخل المجلد في هذه الطريقة على صورة قائمة مرتبطة بحيث يحوي كل جزء مؤشر إلى أجزاء الملف، ومؤشر إلى الملف التالي في المجلد.



الشكل 6. 18: تنفيذ المجلدات على شكل قائمة مرتبطة.

بالرغم من مساهمة القوائم المرتبطة في تذليل الصعوبات سالفة الذكر، إلَّا إنَّ عمليات مثل سرد الملفات، أو تتبع المسارات، أو التحقق من وجود ملف قد تستغرق وقتًا أطول مع زيادة عدد المداخل في المجلد، فمعظم أنظمة الملفات الحديثة لا تحد من عدد الملفات التي يمكن تخزينها في المجلد الواحد. لذلك، وبناءً على نوع نظام الملف، تتجه الأنظمة الحديثة إلى تنفيذ المجلدات باستخدام جدول هاش (جدول التقطيع)، كما هو موضح في الشكل 6. 19، هذه الطريقة تسمح بتخزين عدد كبير من الملفات مع الحفاظ على مستوى عالٍ من الأداء في أثناء الوصول.



الشكل 6. 19: تنفيذ المجلدات باستخدام جدول هاش.

تعتمد طريقة تنفيذ المجلدات عن طريق جدول هاش على توليد مفاتيح للملفات وحفظها في جدول التقطيع، ومن ثم استخدام دالة هاش لمواءمة مفاتيح الملفات مع أسمائها، وإرجاع مؤشر إلى اسم الملف في القائمة الخطية أو المرتبطة والذي سيُستخدم لتحديد أو الوصول إلى الملف المطلوب. باستخدام هذه الطريقة سيُصبح البحث فعّالًا بسبب حقيقة أنه لن يُبحث في القائمة بأكملها في كل عملية وصول وإنَّما ستُفحص مداخل جدول هاش فقط باستخدام المفتاح، إذا عُثِر عليه، فسيُجلب الملف المقابل له. من مآخذ هذه الطريقة أنها تُعاني من التصادمات نتيجةً

لتَولِّد أكثر من مفتاح لنفس الملف والتي يجب استيعابها بطريقة ما.

2.2.5.6 تنفيذ المجلدات باستخدام عقدة الفهرسة

يُعتبر تنفيذ المجلدات باستخدام عقدة الفهرسة أمرًا شائعًا في أنظمة 'يونكس'، حيث تتكون عقدة الفهرسة بالنسبة للمجلد من اسم الملف، ورقم عقدة الفهرسة، كما هو موضح في الشكل 6. 20. في حين تحتوي عقدة الفهرسة نفسها على بقية خصائص الملف، والتي ستساعد نظام الملف في تحليل اسم المسار، وفتح الملف، وتحديد أجزاء القرص الخاصة به.

يُمكن في الشكل 6. 20 رؤية أول مدخلين وهما: (.) و (..) واللذان يمكن مشاهدتهما متى عُرضت محتويات المجلد، يُستخدم المدخل الأول لتغيير المجلد إلى المجلد الحالي نفسه (مما يعني أنه لا يفعل شيئًا)، أمَّا المدخل الثاني فيُستخدم للرجوع للمجلد السابق أو ما يُطلق عليه المجلد الأب للمجلد الحالي.

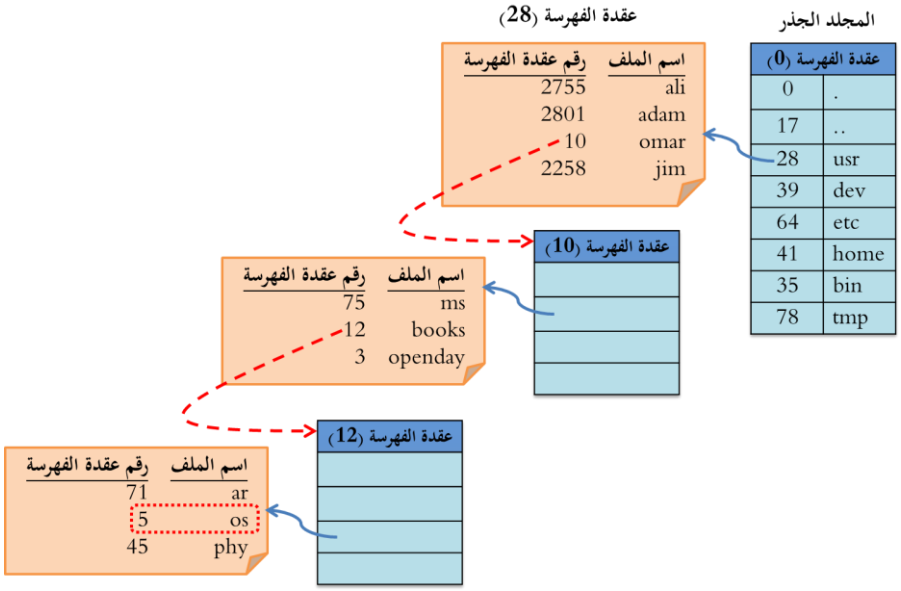
رقم عقدة الفهرسة (330033)	
330033	.
330017	..
330028	مجلد 1
330039	ملف 1
330064	ملف 2
330041	مجلد 2
330035	ملف 3
330078	ملف 4
.....	.....
عقد الفهرسة المناظرة لهذه الأسماء	أسماء الملفات والمجلدات

الشكل 6. 20: تنفيذ المجلدات باستخدام عقدة الفهرسة.

للتعرف أكثر على آلية تنفيذ المجلدات في أنظمة 'يونكس'، يُمكن النظر إلى المثال التالي الخاص بتوضيح الكيفية التي يُحدد بها نظام 'يونكس' مقاطع القرص الخاصة بالملف من خلال تحليل إسم المسار **usr/omar/books/os**/. خطوات هذا التحليل موضحة في الشكل

6. 21 وهي تكمن في الآتي:

- أولًا: يُحدد نظام الملف المجلد الجذر / من خلال عقدة الفهرسة الخاصة به.
- ثانيًا: يقرأ النظام مجلد الجذر وينظر في أول عناصر المسار وهو **usr**، لكي يُحدد عقدة الفهرسة الخاصة بالمجلد **usr**/، وهي العقدة رقم 28.
- ثالثًا: من خلال العقدة السابقة يُحدد مدخل المجلد **omar**/، ومن ثم تُقرأ عقدة الفهرسة الخاصة به، وهي العقدة رقم 10.
- رابعًا: تُحدد عقدة الفهرسة الخاصة بالمجلد **books**/ لتحديد موقع المجلد **os**، تُحمل عقدة هذا المجلد (العقدة رقم 5) في الذاكرة وتبقى هناك إلى حين إغلاق الملف المناظر لها.



الشكل 6. 21: خطوات تحليل إسم المسار **usr/omar/books/os**/ في نظام 'يونكس'.

6.6 إدارة فراغ القرص

كما نعلم أن مساحة التخزين في القرص محدودة وأنه مثلما يجب إدارة المساحة المخصصة

للملفات، فإنه من الطبيعي عند تخزين الملفات داخل القرص يجب البحث عن طرق لإدارة الأماكن الحرة حتى يسهل تتبعها فيما بعد. فالمقصود بالأماكن الحرة هي تلك التي لم تُخصص لأي ملف، أو لأي مجلد، أو المستردة منهما بعد حذفهما. عند إنشاء أيٍ منهما، يُتحقق أولًا من المساحة الحرة لمعرفة مدى ملاءمتها لاستعاب مقدار المساحة المطلوبة للملف أو المجلد الجديد، إذا توفرت هذه المساحة، فستُخصص للعنصر المراد إنشاؤه وستُلغى من إجمالي المساحة الحرة، كما أنه كلما حُذِف ملف أو مجلد، فستُضاف المساحة المحررة إلى قائمة المساحة الحرة.

يُعد نظام الملف أولًا وأخيرًا مسؤولًا عن إدارة، وتخصيص، واسترداد المساحات الحرة ويَستخدم لذلك عدة طرق أو أساليب تتمثل في التالي:

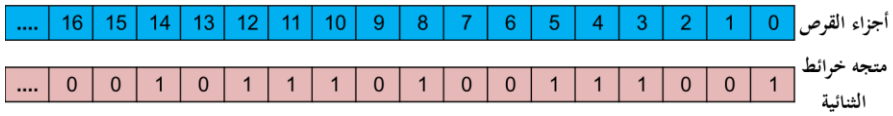
1. الخرائط الثنائية.
2. القوائم المرتبطة.
3. التجميع.
4. العد.

وفيما يلي شرح لهذه الأساليب بنوع من الإيجاز.

1.6.6 طريقة الخرائط الثنائية

تَتَمثل طريقة الخرائط الثنائية في استخدام متجه يحتوي على خانات ثنائية بعدد الأجزاء المكونة للقرص، حيث تُناظر كل خانة جزء وحيد من هذه الأجزاء، إذا كانت قيمة هذه الخانة تساوي واحد، فسيدل ذلك على أنَّ الجزء المناظر لها مستخدم، وإذا كانت قيمتها تساوي صفرًا، فهذا يعني أنَّ ذلك الجزء حر أي غير مشغول. هذا يعني أن قرص يحتوي على n من الأجزاء سوف يحتاج إلى متجه يتكون من عدد n من الخانات الثنائية. مبدائيًا تحمل جميع هذه الخانات القيمة صفر ويتم تغييرها تِبعًا لحالة استخدام أجزاء القرص. يوضح الشكل 6. 22 الفكرة الأساسية لهذه الطريقة، حيث يُبين المثال الموضح في هذا الشكل أنّ الأجزاء الحرة هي: 1، 2، 6، 7، 9، 13، 15، و 16.

تُستخدم خرائط الخانات الثنائية في أنظمة "آبل ماكنتوش" وهي تتمتع ببساطتها وبميزة أنه من السهل نسبيًا العثور على مجموعة متجاورة من الأجزاء الحرة، إلَّا إنَّها ليست ذات كفاءة عالية إلّا إذا أُحتُفِظ بكامل الخريطة في الذاكرة الرئيسية، ولكن قد لا يكون من الممكن الاحتفاظ بها في الذاكرة وخصوصًا إذا ما كان حجم الخريطة كبيرًا جدًا، كما أنها، أي الخريطة تحتاج إلى مساحة إضافية على القرص.



الشكل 6. 22: خرائط الثنائية لإدارة فراغ القرص.

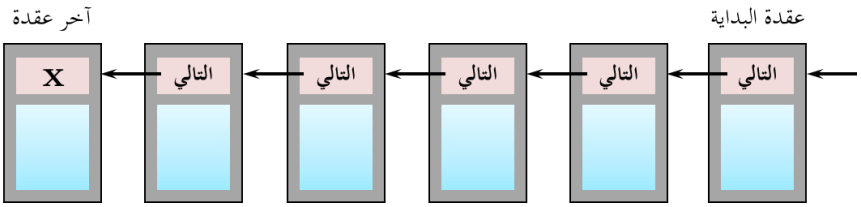
على سبيل المثال، بفرض أن حجم الجزء الواحد هو أربعة كيلوبايت، وأن حجم نظام الملف هو واحد قيقابايت، بالتالي سيكون حجم الخريطة الثنائية يصل إلى 32 كيلوبايت، وهو ما يمكن استعابه في الذاكرة بسهولة، ومسحه بسرعة للعثور على مساحة خالية، أمَّا بالنسبة لنظام ملفات حجمه واحد تيرابايت، فسيبلغ حجم الخريطة 32 ميقابايت والذي لا يزال قابل لاستعابه في الذاكرة، ولكنه سيؤثر سلبًا على سرعة البحث. هذا الحجم سيصل إلى 32 قيقابايت في حالة ما وصل حجم نظام الملف إلى واحد بيتابايت، وهو ما سيكون كارثي ولن يتلائم مع الذاكرة في معظم الأجهزة. يقود هذا الأمر إلى حفظ مثل هذه الخرائط في الذاكرة الثانوية، ولكن قراءتها من هناك يعني زيادة البطأ في عملية البحث.

2.6.6 طريقة القوائم المرتبطة

يتمثل الأسلوب الثاني المستخدم في إدارة المساحة الحرة في ربط جميع الأجزاء الحرة داخل القرص عن طريق قائمة مرتبطة، كما هو موضح في الشكل 6. 23. في هذه القائمة المرتبطة هناك مؤشر يُشير إلى أول عقدة في القائمة يُحفظ في موقع خاص على القرص، تحتوي عقدة البداية هذه على مؤشر للعقدة التالية في القائمة والتي بدورها تحتوي على مؤشر آخر يُشير إلى العقدة التي تليها وهكذا مع بقية العقد، بهذه الطريقة ترتبط جميع عقد القائمة وتنتهي بوضع العلامة x في آخر عقدة.

عندما تحتاج نظم الملف مساحة حرة، تسحب هذه المساحة من القائمة المرتبطة وتُخصصها إلى الملف المراد إنشاؤه، كما أنه في حالة الاستغناء عنها فستُرجَّع المساحة المحررة

إلى القائمة المرتبطة للمساحات الحرة. من مزايا هذه الطريقة بساطة استخلاص العقدة الأولى من قائمة المساحة الحرة وتخصيصها إلى الملف، ونقل مؤشر عقدة البداية إلى العقدة الحرة التالية في القائمة الحرة.

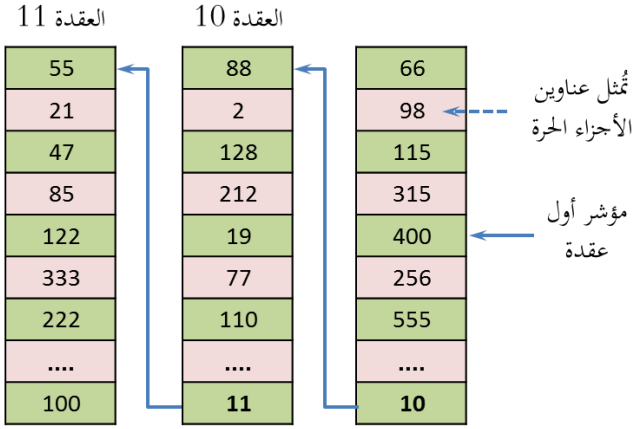


الشكل 6. 23: حفظ المساحات الحرة في شكل قائمة مرتبطة.

3.6.6 طريقة تجميع العناوين

من الطرق الأخرى لإدارة المساحات الحرة داخل القرص هي استخدام منهجية التجميع. في هذا النهج، تُحمَّل عناوين جميع الأجزاء الحرة داخل بعض من عقد القوائم (المرتبطة)، بحيث تستوعب كل عقدة أكبر عدد ممكن من هذه العناوين، وذلك كما هو موضح في المثال المعطَى في الشكل 6. 24. في هذا المثال، حُفظت عناوين الأجزاء الحرة في كل من العقدة الأولى، والثانية، والثالثة، كما أنّ آخر جزء في كل عقدة يُشير إلى العقدة التالية. أيضًا، الأرقام المبينة في هذا المثال تُمثل عناوين الأجزاء الحرة داخل القرص.

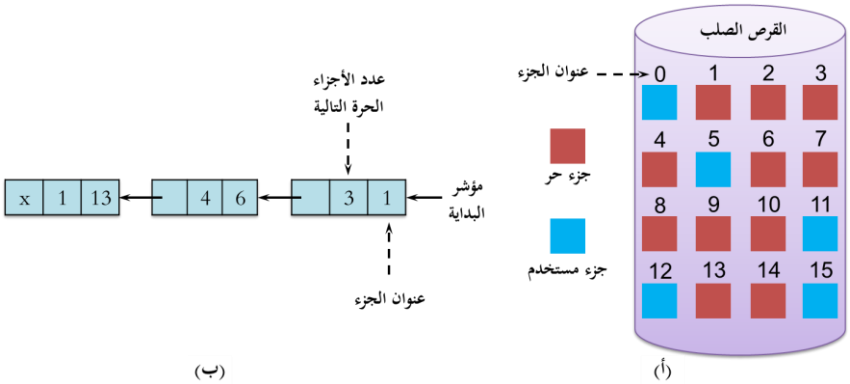
في هذه الطريقة، يُلاحظ أن هناك جزء من مساحة القرص يُخصص لحفظ هذه العناوين، بالتالي بفرض أن هناك قرص حجمه واحد تيرابايت يحوي تقريبًا واحد بليون جزء، سوف يحتاج إلى ما يُقارب أربعة مليون جزء بسعة 255 عنوان للاحتفاظ بكافة عناوين الأجزاء. تعتمد هذه الفرضية على أساس أن حجم كل جزء هو واحد كيلوبايت وأن كل عنوان بطول 32 خانة ثنائية. تتمثل ميزة هذا الأسلوب في أنه يمكن العثور بسهولة على عناوين مجموعة من الأجزاء الحرة للقرص.



الشكل 6. 24: حفظ عناوين الأجزاء الحرة داخل القوائم.

4.6.6 طريقة العد

يعتمد نهج طريقة العد في الأساس على حقيقة أنه يمكن تخصيص عدة أجزاء متجاورة أو تحريرها في وقت واحد وخصوصًا إذا ما أُستخدمت خوارزمية التخصيص المتجاور الموضحة في بداية هذا القسم. بالتالي، بدلًا من الاحتفاظ بعناوين كل الأجزاء الحرة في قوائم– كما هو الحال في طريقة التجميع–، يمكن الاحتفاظ فقط بمعلومتين هما: عنوان أول جزء حر، وعدد الأجزاء الحرة التي تليه. يُبين الشكل 6. 25–أ حالة افتراضية للقرص، في حين يُوضح الشكل 6. 25–ب القائمة المرتبطة لتتبع الأجزاء الحرة المناظرة لهذا القرص.



الشكل 6. 25: طريقة العد: أ) حالة القرص– ب) قائمة تتبع الأجزاء الحرة.

على سبيل المثال، في الشكل 6. 25–ب سيكون الإدخال الأول لقائمة المساحة الحرة عند العنوان واحد وبطول ثلاثة أجزاء، والإدخال الثاني لقائمة المساحة الحرة عند العنوان ستة وبطول أربعة أجزاء، وهكذا. نلاحظ هنا أن كل إدخال يتطلب مساحة أكبر من عنوان القرص لوحده، إلَّا إنَّ القائمة الإجمالية ستكون أقصر، طالما أن عدد الأجزاء أكبر من واحد. ومع ذلك، يُعتبر استخدام القوائم المرتبطة لتتبع هذا المخطط أمرًا غير فعَّال، لأن قراءة أي جزء منها يتطلب عبور القائمة، الأمر الذي يتطلب وقت إدخال، وإخراج كبير. لحسن الحظ، لا يُعد عبور قائمة المساحة الحرة إجراءً متكررًا، كما أنه من الممكن حفظ هذه الإدخالات باستخدام هياكل الأشجار بدلًا من القوائم المرتبطة. وبالتالي فإن عمليات البحث، والحذف، والإدراج ستكون أكثر فاعلية.

7.6 مشاركة الملفات

أشرنا في القسم 3.6 إلى أن مشاركة الملفات بين المستخدمين أمرًا مرغوبًا فيه وخصوصًا للذين يبحثون عن التعاون فيما بينهم لتحقيق أهداف محددة. تتمثل هذه المشاركة في المشاركة العامة أو الخاصة لمساحات التخزين سواءً الموجودة على الحاسوب أو في شبكة ذات مستويات مختلفة من امتيازات الوصول، يُستثنى من ذلك مشاركة الملفات عن طريق تبادلها باستخدام وسائط التخزين الثانوية كالأقراص المدمجة أو عن طريق الرسائل البريدية.

من ناحية أخرى، تُمكن مشاركة الملفات المستخدمين من استخدام نفس الملف أو ملفات أخرى من حيث القراءة، والكتابة، والتعديل، والنسخ، أو الطباعة. لذلك، يجب أن تُتيح نظم التشغيل آليات لمشاركة الملفات بالرغم من الصعوبات المتأصلة. يُناقش هذا القسم عدة جوانب لموضوع مشاركة الملفات كتلك التي تتعلق بالمشاكل العامة التي تَنشأ عندما يتشارك عدة مستخدمين في الملفات، والتحديات الكامنة في توسيع المشاركة إلى أنظمة الملفات المتعددة، بما في ذلك أنظمة الملفات عن بُعد. أخيرًا، يتناول هذا القسم الإجراءات المتبعة للتعامل مع الأحداث المتعارضة في استخدام الملفات المشتركة من قبل عدة مستخدمين، كالكتابة إلى ملف وآليات تنظيم ذلك.

1.7.6 تعدد المستخدمين

مع تعدد المستخدمين داخل النظام الواحد، تصبح سمتي تسمية الملفات وحمايتها من الاستخدام أو الوصول غير المخول ضرورة أساسية في نظام التشغيل سواءً أكان ذلك في نفس النظام، أو عبر أي شبكة. لذلك يجب على نظام التشغيل توفير آليات لحماية الملفات من التطفل وتنظيم عملية مشاركة الملفات، هذا الأمر قد يُنجز بإحدى الطريقتين: السماح للمستخدم من قبل النظام بالوصول إلى ملفات المستخدمين الآخرين بشكل افتراضي، أو قد يُطلب منه منح حق الوصول إلى ملفاته بشكل خاص.

مقارنة بنظام المستخدم الواحد، فإن تعدد المستخدمين في النظام يتطلب الاحتفاظ بسمات للملفات والمجلدات أكثر مما هو مطلوب، وذلك لتحقيق مشاركة الملفات وحمايتها بشكل فعّال. على هذا الأساس تطورت معظم النظم لتشمل استخدام مفهومي: المالك والمجموعة. فالأول هو المستخدم الذي يُنشِأ الملف والذي يتمتع بأكبر قدر من التحكم فيه من حيث تغيير السمات ومنح إذن الوصول إليه، أمّا مفهوم المجموعة فيُعطي الحق لمجموعة من المستخدمين لمشاركة الوصول إلى ملف– ما– وتنفيذ مجموعة فرعية واحدة من العمليات عليه، بينما يمكن لأعضاء مجموعة أخرى تنفيذ مجموعة فرعية أخرى من هذه العمليات، ما هية هذه العمليات التي يمكن تنفيذها بواسطة أعضاء المجموعة الواحدة والمستخدمون الآخرون تُحدد بالضبط من قبل مالك الملف.

أُشير في الفصل الأول تحت مفهوم الأمن والحماية إلى أن أنظمة التشغيل تحتفظ بمعرِّفات للمستخدمين وكذلك للمجموعات ضمن سمات الملف أو المجلد، والتي تُستخدم من قبل النظام لمصادقة أو لمطابقة حقوق الاستخدام. بالتالي عند طلب المستخدم القيام بأي عملية على أحد الملفات، يُقارن أولًا معرِّف هذا المستخدم بمعرِّف مالك الملف وتحديد إمكانية تنفيذ العملية من عدمها، الأمر نفسه يتكرر مع معرِّف المجموعة.

2.7.6 أنظمة الملفات عن بُعد

ساهم ظهور شبكات الحاسوب في تطور مشاركة الملفات بين المستخدمين بحيث أصبحت تتم عن بعد عبر شبكة محلية، أو واسعة، أو أي إتصال شبكي، وفي التعامل مع الملفات كما لو

كانت ملفات محلية يُمكن فتحها، وقراءتها، وكتابتها، وإغلاقها وما إلى ذلك من الإجراءات المتعلقة بها.

هذا يعني أن أنظمة الملفات عن بُعد تُمكن أي تطبيق يعمل على حاسوب عميل– ما– من الوصول إلى الملفات المخزنة على حاسوب آخر داخل الشبكة، والتي قد تشمل أيضًا المصادر المادية كالطابعات وغيرها. غالبًا ما يُشار إلى أنظمة الملفات هذه على أنها أنظمة ملفات شبكة أو أنظمة ملفات موزعة.

بالتالي تُعد مشاركة الملفات عن بُعد نوعًا من تقنية نظام الملف الموزعة والتي طُورت في عام 1980 وضُمِّنَت لأول مرة ضمن إصدار نظام 'يونكس' الخامس. كما تضمنت إصدارات 'ويندوز 2000'، و'إكس بي'، و'سيرفير 2003' نظام الملف عن بعد الأصلي المسمى بنظام شبكات ميكروسوفت، والذي وفَّر الوصول عن بعد إلى الملفات بالإضافة إلى الطابعات، والراسمات. مؤخرًا أُستخدمت الحوسبة السحابية على نحو متزايد لمشاركة الملفات.

3.7.6 دلالات توافق مشاركة الملفات

من أهم معايير تقييم نظام الملف الذي يدعم مشاركة الملفات هو دلالات توافق هذه المشاركة. فهي التي تُحدد كيفية وصول عدة مستخدمين لنظام– ما– إلى ملف مشترك في وقت واحد، كما أنها تُنسق بين طرق عرض الملفات المشتركة على نظام شبكي عندما يقوم أحد المستخدمين بإجراء تغييرات على الملف المشترك. على وجه الخصوص، تُحدد دلالات توافق مشاركة الملفات اللحظة التي ستظهر فيها هذه التغييرات للمستخدمين الآخرين.

يبدو للوهلة الأولى أن دلالات توافق مشاركة الملفات تتضمن جميع مشكلات المزامنة التي نُوقشت في الفصل الثاني، فهي ترتبط ارتباطًا مباشرًا بخوارزميات تزامن العمليات الخاصة بذلك الفصل، ولكن وللأسف فإن التأخيرات الطويلة التي تنطوي عليها عمليات الشبكة ومعدلات النقل البطيئة للأقراص تمنع تنفيذ العمليات كسلسلة غير قابلة للتجزئة والتي يجب أن تُنفذ إما ككتلة واحدة، أو لا يحدث شيء، وذلك منعًا لإحداث تغييرات غير مكتملة تؤثر سلبًا في دقة البيانات.

لتوضيح مفهوم دلالات توافق مشاركة الملفات، ستُقدم دلالات كل من نظام 'يونكس'، والجلسة، والملفات المشتركة غير القابلة للتغيير، والأقفال.

- **دلالات نظام 'يونكس'**

يستخدم نظام ملفات 'يونكس' الدلالات التالية:

- تظهر الكتابة إلى أي ملف مفتوح من قبل المستخدم على الفور لأي مستخدم آخر يعمل على نفس الملف المفتوح.
- لا يمكن لعدة مستخدمين إجراء عمليات القراءة والكتابة على نفس الملف بشكل متزامن.
- نظرًا لأن الملف يحتوي على صورة واحدة، فقد يؤدي ذلك إلى تأخير عمليات الوصول إلى الملف.
- هناك خيار لمشاركة مؤشر الموقع الحالي داخل الملف من قبل المستخدمين بحيث إذا ما قُدِّم المؤشر من قبل أي مستخدم، فسيتأثر بذلك بقية المستخدمين.

- **دلالات الجلسة**

هذه الدلالات مستخدمة في نظام 'أندرو' وهي تتمثل في التالي:

- لا تظهر الكتابة إلى أي ملف مفتوح من قبل أي مستخدم على الفور للمستخدمين الآخرين الذين لديهم نفس الملف المفتوح.
- فقط عندما يُغلق الملف، تصبح التغييرات التي أُجريت عليه مرئية فقط للمستخدمين الذين يفتحون الملف في وقت لاحق.
- وفقًا لهذه الدلالات، يمكن ربط الملف مؤقتًا بعدة طرق عرض– ربما مختلفة– في نفس الوقت. بالتالي يمكن لعدة مستخدمين أداء كل من القراءة والكتابة بشكل متزامن، كما أنه لا تُفرض أي قيود على جدولة عمليات الوصول، ولا يُأخر أي مستخدم في قراءة أو كتابة نسخته الشخصية من الملف.
- يمكن الوصول إلى أنظمة ملفات 'أندرو' بواسطة أنظمة الملفات الأخرى.

- **دلالات الملفات المشتركة غير القابلة للتغيير**

تتمثل هذه الدلالات في التالي:

- وفقًا لهذه لطريقة، يعني الإعلان عن مشاركة الملف من قبل مُنشئِه أن محتوياته ثابتة وغير قابلة للتعديل.
- لا يمكن إعادة استخدام الاسم لأي مصدر آخر، وبالتالي تصبح مشاركة الملف متمثلة في القراءة فقط.

- دلالات الأقفال

تتمثل هذه الدلالات في التالي:

- مشاركة الملفات تعني التأمين لإتاحة الوصول الحصري.
- منع الآخرين من القراءة في أثناء عمليات تحديث الملف.
- دلالات الأقفال يمكن أن تشمل الملف بالكامل أو جزء محدد منه.

8.6 نظام الملف الظاهري

مع تعدد أنواع نظم التشغيل تتعدد كذلك أنواع نظم الملف، فقد نجد العديد من أنظمة الملف المختلفة قيد الاستخدام داخل نظام التشغيل الواحد وذلك لعدة أسباب منها: الحاجة إلى استخدام الأنظمة القديمة بالرغم من وجود أنظمة حديثة، والحاجة إلى استخدام أنظمة خاصة، وتواجد عدة أنواع من الأجهزة وربما عدة أقسام في القرص الواحد، ولكل منها نظام ملف مختلف. مثلًا، يحتوي نظام تشغيل 'ويندوز' على عدة أنظمة ملفات منها **FAT-16**، و **FAT-32**، و **NTFS**[19]، وغيرها. بينما يحوي نظام تشغيل 'يونكس' أنظمة ملفات مثل، **Ext2**[20]، و **Ext3**، و **Ext4**، وغيرها. هنا يتبادر إلى أذهاننا أن نظام الملف الذي يعمل في نظام 'ويندوز' مختلف عنه في نظام 'يونكس'، بالتالي، السؤال الذي يطرح نفسه الآن: هل من

[19] نظام ملفات التكنولوجيا الجديدة أو 'إن تي إف إس' (**New Technology File System**).

[20] نظام الملف الموسع الثاني أو 'إ إكس تي 2' (**Second Extended File System**).

الممكن الجمع بين أنظمة ملفات مختلفة لنظم تشغيل مختلفة كالجمع مثلًا بين NTFS و Ext4 داخل نظام التشغيل الواحد؟

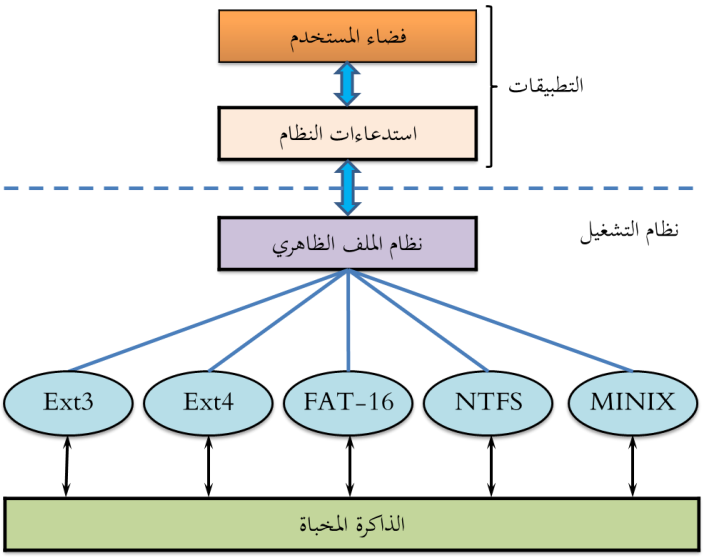
بناءً على ما سبق ذكره، من الواضح أن أنظمة التشغيل الحديثة يجب أن تدعم عدة أنواع من نظم الملف في نفس الوقت. ولكن كيف يمكن لهذه الأنظمة القيام بذلك بحيث تكون نظم الملف متكاملة في بيئة واحدة؟ وكيف يمكن للمستخدم التنقل بسلاسة بين هذه الأنواع المختلفة في أثناء تعامله مع مساحة نظم الملف دون الاهتمام بنوعية الجهاز المتواجد عليه الملف، أو نظام الملف المستخدم على هذا الجهاز؟

ببساطة ما يهم المستخدم هو القدرة على استخدام كل من الملف وطرق الوصول إلى المجلد التي نُوقشت سابقًا. لذلك تتضمن نظم التشغيل الحديث نظام الملف الظاهري لتوفير هذا التجريد. المناقشة التالية لا تقتصر على نظام تشغيل محدد، وإنما تُمثل نظرة عامة لكيفية عمل أنظمة الملفات الظاهرية.

إن دمج عدة أنظمة ملفات مختلفة في نظام تشغيل واحد يُعتبر أمرًا محتملًا وقابلًا للتنفيذ وخصوصًا مع ظهور مفهوم نظام الملف الظاهري والموضح في الشكل 6. 26، والذي يُبين علاقة التطبيقات مع نظام الملف الظاهري وأنظمة الملفات الأساسية. يتكون هذا المفهوم من ثلاث طبقات، تُوفر الطبقة الأولى وهي طبقة التطبيقات عمليات المستخدم والتي تستخدم هياكل بيانات وإجراءات لعزل وظائف استدعاءات النظام الأساسية عن تفاصيل التنفيذ، كما أن هذه الطبقة تُمثل الواجهة البينية لنظام الملف التي تُوفر استدعاءات مثل، فتح، وقراءة، وكتابة، وغلق الملفات وكل ما يتعلق بها، بالتالي عندما يقوم أحد التطبيقات بإجراء استدعاء لعملية إدخال أو إخراج، ينتقل الطلب إلى نظام الملف الظاهري والذي يُحدد بدوره الجهاز الذي يتواجد عليه الملف المطلوب وتوجيه الطلب إلى نظام الملف المناسب.

تُسمى الطبقة الثانية طبقة نظام الملف الظاهري، وهي طبقة مجردة موجودة فوق أنظمة الملفات المختلفة يمكن من خلالها لتطبيقات المستخدم الوصول إلى هذه الأنظمة، كما أنها تُمثل قلب هذا النظام وتحتوي على واجهتين متميزتين: واجهة عليا للتعامل مع عمليات المستخدم، وأخرى سفلية للتعامل مع أنظمة الملفات الحقيقية والتي تمثل الطبقة الثالثة. لدى طبقة نظام الملف الظاهري مهمتين أساسيتين: تتمثل المهمة الأولى في فصل العمليات العامة

للنظام عن تنفيذها من خلال تحديد واجهة نظام الملف الظاهري، تجدر الإشارة هنا إلى أنه من الممكن أن تتعايش عدة تطبيقات لواجهة نظام الملف الظاهري على نفس الجهاز، الأمر الذي سيسمح بالوصول إلى أنواع مختلفة من أنظمة التشغيل المثبّتة محليًا، بالإضافة إلى ذلك تتمثل المهمة الثانية في توفير آلية لتمثيل الملف بشكل فريد عبر أي شبكة وذلك لدعم أنظمة ملفات الشبكات.



الشكل 6. 26: مفهوم نظام الملف الظاهري.

أخيرًا، يجب على نظام الملف الظاهري إدارة جميع أنظمة الملفات المختلفة التي تُحمل في أي وقت، وإعطاء المستخدم الوهم بأنه يتفاعل معها ويتحكم فيها، وللقيام بذلك يحتفظ هذا النظام بهياكل البيانات التي تصف نظام الملف الظاهري بالكامل وأنظمة الملفات الحقيقية المحمَّلة.

9.6 اعتمادية نظام الملف

تتواجد بيانات مستخدمو الأنظمة الحاسوبية وملفاتهم على الوسائط التخزينية بشكل مستمر إلى أن تُغيَّر من قبلهم بشكل صريح، لذلك إن من أهم وظائف نظم التشغيل هو المحافظة على حياة وصحة هذه الملفات والبيانات طيلة فترة الحفظ، فتلف الحاسوب قد يُستعاض عنه مثلًا

بحاسوب آخر، ولكن ضياع الملفات أو إنهيارها أمرٌ في غاية الخطورة إن لم يكن الكارثة في حد ذاتها، وخصوصًا عندما يتعلق الأمر بالبيانات المهمة.

تُسهم الأحداث المختلفة التي تتعرض لها الأنظمة الحاسوبية في جعل أنظمة الملفات تفشل في تحقيق توقعات المحافظة على الملفات بسبب الإنهاء المفاجئ لنظام التشغيل بالطرق التالية:

- انقطاع التيار الكهربائي.
- التعطل العرضي للنظام أو لأحد مكوناته.
- إيقاف تشغيل النظام دون إتباع إجراءات الإيقاف الصحيحة.
- الأخطاء البشرية.

لذا من الضرورة بمكان إيجاد طرق هدفها الرئيسي أو الأساسي المحافظة على بيانات وملفات النظام والمستخدمين.

10.6 النسخ الاحتياطي

في الوقت الذي لا يُمكن لنظام الملف أن يُوفر أي حماية ضد تلف المعدات المادية للحاسوب ووسائط التخزين، بإمكانه أن يُوفر وسائل تساعد في حماية المعلومات. تتمثل أبسط هذه الوسائل في النسخ الاحتياطي أي في عمل نسخ احتياطية من الملفات في قرص آخر، متمثلًا في الأشرطة المغناطيسية، والأقراص ذات السعات الكبيرة، وذلك لضمان حفظ بيانات المستخدم في النظام في المقام الأول وكذلك حفظ حالة النظام أو حالة تشغيله، الأمر الذي سيُساعد في استعادة النظام إلى آخر حالة حُفِظت عند حدوث أعطال بالنظام إلى جانب استعادة جميع بيانات النسخ الاحتياطي المحددة.

عملية النسخ الاحتياطي لن تكون مرة واحدة فقط بل ستكرر عدة مرات، وذلك حسب الفترات الزمنية المحددة والتي قد تكون يوميًا، أو شهريًا، أو حتى سنويًا، وذلك حسب المتطلبات وحساسية الملفات. لذلك لا يبدو هذا الأمر بهذه البساطة، لوجود عدة اعتبارات يتمثل بعض منها في الآتي:

- هل ينبغي نسخ نظام الملف بالكامل أو قد يُكتفى فقط بأجزاء منه؟ في العادة، من المناسب

نسخ مجلدات محددة بكافة محتوياتها بدلًا من النظام بأكمله.

- لتجنب إهدار الوقت والجهد، ينبغي جعل عملية النسخ تتم دوريًا فقط للملفات التي غُيِّرت أو أُستحدِث مؤخرًا، هذا الأمر يقود إلى مفهوم النسخ الاحتياطي التزايدي.
- لضخامة حجم البيانات، ينبغي ضغطها قبل إجراء عملية النسخ الاحتياطي، ولكن هذا الأمر محفوف بمخاطر عدم القدرة على استرجاع الملفات المضغوطة بسبب أخطاء في آليات الضغط.

لمزيد من التفاصيل حول هذه الاعتبارات وغيرها، يمكن الرجوع إلى تانن باوم وآخرون (Tanenbaum et al., 2015). الأقسام التالية تُناقش بعض من الاستراتجيات المستخدمة في عمل النسخ الاحتياطية لنظام الملف وكذلك المشكلات الفنية التي تنطوي عنها.

1.10.6 النسخ الاحتياطي الفعلي

ينسخ النسخ الاحتياطي الفعلي كل بيانات الملفات الفعلية بالضبط من القرص الرئيسي إلى قرص الإخراج بنفس الترتيب بدايةً من أول مقطع فيه ووقوفًا عند نسخ آخر مقطع منه، قد يتم هذا النسخ في إحدى الحالتين التاليتين: في الحالة الأولى، يُسمح للمستخدم بتعديل قاعدة البيانات في أثناء النسخ الاحتياطي السريع، وللقيام بذلك تُحفظ ملفات سجل التغييرات التي أُجريت في أثناء النسخ الاحتياطي، لكي تُستخدم في مزامنة قاعدة البيانات والنسخة الاحتياطية، أمَّا في الحالة الثانية، لا يُمكن للمستخدم تعديل قاعدة البيانات في أثناء النسخ الاحتياطي، لذلك تُزامَن بشكل دائم قاعدة البيانات والنسخة الاحتياطية.

من العيوب التقنية التي يعاني منها النسخ الاحتياطي الفعلي هو أن عملية النسخ هذه تشمل كذلك المقاطع غير المستخدمة، الأمر الذي يُعد إهدارًا للوقت والجهد. مع ذلك، إذا تمكن برنامج النسخ من التعرف على المقاطع الحرة، فبإمكانه تجنب نسخ هذه المقاطع، ولكن قد يتسبب ذلك في عدم الحفاظ على ترتيب المقاطع. بمعنى أنه لم يعد صحيحًا أن المقطع $\mathbf{k}$ في النسخة الاحتياطية هو نفسه المقطع $\mathbf{k}$ في القرص الرئيسي. لمعالجة هذا الأمر يمكن كتابة رقم كل مقطع أمامه في النسخة الاحتياطية.

مصدر القلق الثاني في عملية النسخ الاحتياطي الفعلي هو أنها لا تستثني أيضًا المقاطع

المعطوبة. فمن المعروف أنه من المستحيل تصنيع الأقراص الكبيرة دون أي مقاطع معطوبة، بعضها قد يُكشف عنه ويُستبدل بمقاطع احتياطية محفوظة في نهاية كل مسار عند إجراء التنسيق منخفض المستوى والذي قد ينتج عنه هو الآخر عطب مقاطع أخرى، بالتالي إذا أُعيد تعيين كافة المقاطع المعطوبة بواسطة وحدة تحكم القرص وإخفائها عن نظام التشغيل، فإن النسخ الاحتياطي الفعلي يعمل بشكل جيد.

من ناحية أخرى، إذا بقت المقاطع المعطوبة مرئية بالنسبة لنظام التشغيل وحُفظت في ملف واحد أو أكثر من ملفات المقاطع المعطوبة، فمن الضروري للغاية أن يحصل برنامج النسخ الاحتياطي الفعلي على هذه المعلومات لكي يتجنب نسخها، وذلك منعًا لحدوث أخطاء قراءة القرص التي لا تنتهي في أثناء محاولة عمل نسخة احتياطية من ملف المقاطع المعطوبة.

أخيرًا، تتميز آلية النسخ الاحتياطي الفعلي ببساطتها في التنفيذ وسرعتها في التشغيل، إلَّا إنَّها تعاني من عدم القدرة على استثناء مجلدات محددة، واسترجاع الملفات بشكل فردي حسب الطلب، وتطبيق مفهوم النسخ الاحتياطي التزايدي.

2.10.6 النسخ الاحتياطي المنطقي

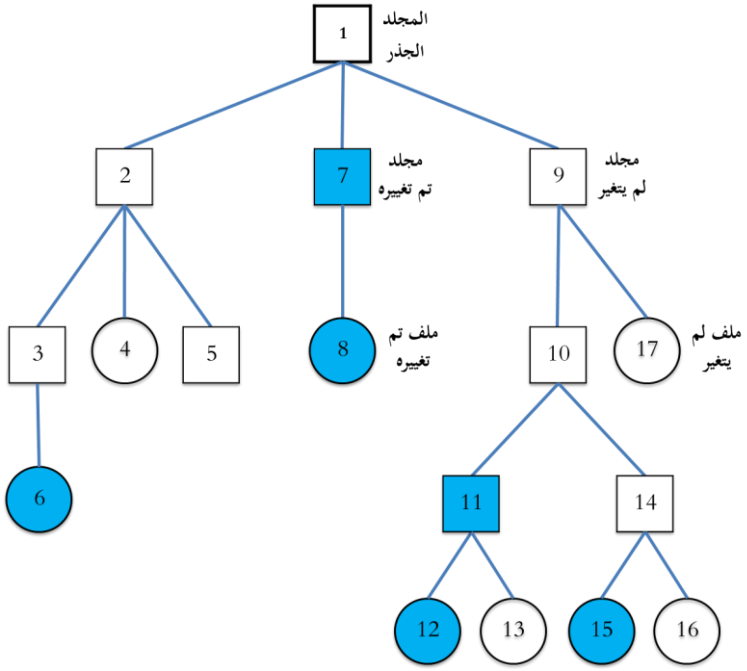
يُعتبر النسخ الاحتياطي المنطقي من الطرق الأكثر شيوعًا في هذا المجال والمستخدم من قبل معظم أنظمة 'يونكس'. فهو يبدأ من أي مجلد محدد أو من عدة مجلدات محددة وينسخ بشكل مستمر جميع الملفات والمجلدات الموجودة هناك والتي تغيرت منذ تاريخ آخر عملية نسخ احتياطي. هذا يجعل قرص الاحتياط يتحصل على سلسلة من المجلدات والملفات المحددة مسبقًا بعناية، مما يؤدي إلى سهولة استعادة أي ملف أو مجلد معين عند الطلب، على العكس من الطريقة السابقة.

لتوضيح طريقة النسخ الاحتياطي المنطقي بشكل مفصَّل، يُقدِّم هذا القسم خوارزمية نسخ احتياطي شائعة الاستخدام في معظم أنظمة 'يونكس'. يعرض الشكل 6. 27 مثال لشجرة مجلدات تحوي عدة ملفات ومجلدات مفهرسة بأرقام عقد الفهرسة، في هذا الشكل تُمثل المربعات المجلدات بينما تُمثل الملفات بالدوائر، كما تُعبر العناصر المظللة عن البيانات التي تحتاج إلى عملية النسخ الاحتياطي، أي العناصر التي تغيرت منذ تاريخ آخر عملية نسخ، بينما لا

تحتاج بقية العناصر لذلك.

تُجري خوارزمية النسخ الاحتياطي المنطقي أيضًا عملية النسخ الاحتياطي لكافة العناصر (حتى تلك غير المعدلة) التي تقع على مسار العناصر المعدلة وذلك لسببين:

- يتمثل السبب الأول في إتاحة استعادة الملفات والمجلدات التي نُسخت إلى نظام ملفات جديد مُحمَّل على حاسوب آخر. هذا الأمر يُتيح إمكانية استخدام برامج النسخ الاحتياطي والاستعادة في نقل أنظمة الملفات بالكامل بين أجهزة الحواسيب.



الشكل 6. 27: النسخ الاحتياطي المنطقي لنظام الملف.

- يتمثل السبب الثاني في إتاحة الفرصة لاستعادة ملف واحد بشكل تدريجي. مثلًا، بفرض أنه قد نُسخ نظام ملف بالكامل وأُلحق بنسخ تزايدي، بعد ذلك أُزيل المجلد **/usr/bou/proj/os1** بالإضافة إلى جميع المجلدات والملفات التحتية، بعد فترة من الزمن رغب المستخدم في استعادة الملف **/usr/bou/proj/os1/ch2/abstract**. ولكن، من الملاحظ أنه لا يمكن استعادة

الملف **abstract**، فقط لأنه لا يوجد مكان يُوضع فيه. لذلك، يجب أولًا استعادة المجلدين **os1** و **ch2** للحصول على كل السمات المتعلقة بهما.

تعتمد خوارزمية النسخ الاحتياطي المنطقي على خرائط الثنائية لتتبع عملية النسخ من خلال تخصيص عدة خانات ثنائية لكل عقدة فهرسة لتمثيل رقمها داخل هذه الخرائط. تُضبط وتُمسح هذه الخانات مع استمرار الخوارزمية في التنفيذ والذي يتم على أربع مراحل.

- المرحلة الأولى: تبدأ هذه المرحلة من المجلد الجذر وتفحص جميع المداخل الموجودة فيه بحيث تُوضع علامة مناظرة داخل الخريطة لكل عقدة فهرسة عُدِّل ملفها. كما يُعلَّم كل مجلد سواءً عُدِّل أم لم يُعدَّل، ومن ثم يُفحص بشكل متكرر. بنهاية هذه المرحلة تكون كافة الملفات المعدَّلة والمجلدات قد عُلِّمت داخل الخريطة الثنائية، كما هو موضح بالخانات المضللة في الشكل 6. 28–أ.

(أ)	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
(ب)	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
(ج)	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
(د)	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

الشكل 6. 28: خرائط الثنائية المستخدمة من قبل خوارزمية النسخ الاحتياطي المنطقي.

- المرحلة الثانية: في هذه المرحلة تُعبر شجرة المجلدات مرة ثانية لغرض إلغاء تعليم أي مجلد لا يحتوي على ملفات معدَّلة سواءً أكانت بداخله أو ضمن مجلدات فرعية موجودة تحته. الخريطة الثنائية الموضحة في الشكل 6. 28–ب هي نتاج هذه المرحلة، كما يُلاحظ في هذا الشكل قد أُلغي تعليم المجلد 5، لأنه لا يحتوي على أي شيء معدَّل، وبالتالي سوف لن يتم نسخه احتياطيًا. بالمقابل سوف يتم نسخ المجلدين 9، و 10 بالرغم من عدم تعرضهم للتعديل، ولكن لاحتياجهما في عملية استعادة الملفات في جهاز آخر. من خلال هذا الشكل أصبح واضحًا أيٍ من المجلدات والملفات يجب نسخها احتياطيًا.
- المرحلة الثالثة: في هذه المرحلة تُستعرض الخريطة الثنائية وتُنسخ كل المجلدات المعلمة للنسخ والممثلة في الخريطة الثنائية الموضحة في الشكل 6. 28–ج. هنا يجب أن يلاحظ

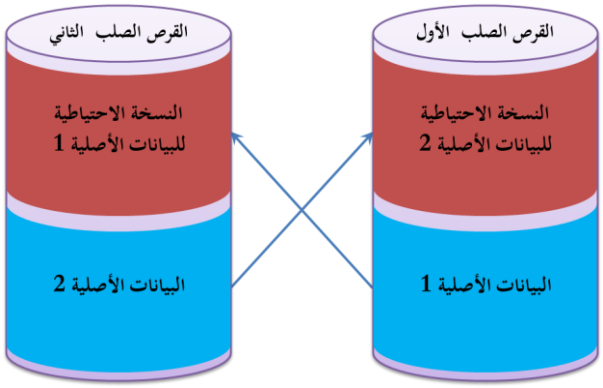
أن كل مجلد يُلحق به سماته الخاصة به لغرض الاستعانة بها في استعادته.

- المرحلة الرابعة: أخيرًا، في هذه المرحلة تُنسخ الملفات المعلَّمة في الخريطة الثنائية الموضحة في الشكل 6. 28–د، إضافةً إلى السمات الخاصة بها. هذا الأمر يُنهي عملية النسخ الاحتياطي المنطقي.

بعد الإنتهاء من عملية النسخ الاحتياطي لنظم الملف، تأتي الحاجة إلى عملية استعادتها من الأقراص الاحتياطية، هذه العملية تتم بشكل مباشر من قرص الاحتياط، وتبدأ بإنشاء نظام ملف جديد على القرص واسترجاع آخر نسخة احتياطية كاملة. وباعتبار المجلدات هي أول من يظهر في القرص الاحتياطي، ستُرجَّع أولًا لكي تُعطي الشكل الظاهري لنظام الملف، يلي هذه الخطوة استعادة الملفات نفسها إلى أن تكتمل النسخة الاحتياطية الكاملة، بعدها تُسترجع أول نسخة تزايدية تليها النسخ التالية، على التوالي إلى أن تكتمل جميع النسخ.

3.10.6 طرق أخرى للنسخ الاحتياطي

تواجه الحلول السابقة بعض من العيوب والمتمثلة في استهلاك وقت كبير عندما يُنسخ ملف ذو حجم كبير عدة مرات. يمكن حل هذه المشكلة عن طريق تزويد كل حاسوب بقرصين بدلًا من قرص واحد، ومن ثم تقسيم كل قرص إلى جزئين، كما هو موضح في الشكل 6. 29، بحيث يكون هناك جزء خاص بالبيانات الأصلية، وآخر خاص بالنسخة الاحتياطية للبينات نفسها. هذه الآلية تسمح بكتابة النسخة الاحتياطية مباشرة في نفس زمن كتابة النسخة الأصلية.



الشكل 6. 29: تزويد الحاسوب بقرصين من أجل النسخ الاحتياطي.

11.6 تناسق نظام الملف

من خلال تتبع آلية عمل نظم الملف في الأقسام السابقة، لاحظنا أن هذه النظم تعتمد على مجموعة داخلية من الجداول والسمات لتتبع المساحات المستخدمة، والمساحات المتاحة للاستخدام، وكل ما يتعلق بالملفات والمجلدات. لذلك عندما لا تتزامن هذه الجداول الداخلية أو البيانات الوصفية لنظم الملف بشكل صحيح مع البيانات الحقيقية الموجودة على القرص، يحدث نوعًا من عدم التناسق (التوافق) داخل أنظمة الملفات يُقلل من موثوقيتها.

إضافة إلى ذلك، تتوزع الجداول الداخلية والبيانات الوصفية لنظم الملف على عدة أماكن مختلفة داخل النظام، وبناءً على العمليات المطبقة على الملفات، يحتاج النظام إلى تحديثها من حين إلى آخر وفقًا لمجريات تنفيذ هذه العمليات. على سبيل المثال، تتطلب عملية إضافة بيانات إلى أي ملف ثلاثة تحديثات للبيانات الوصفية:

1. العثور على مساحة تخزينية غير مستخدمة لغرض تخصيصها للبيانات الجديدة.
2. ربط هذه المساحة مع قائمة مجموعات البيانات التابعة للملف.
3. إعادة كتابة سمات الملف مثل، الحجم، وزمن آخر تعديل، ... إلخ.

نظريًا: مثل هذه الخطوات يجب أن تُنفذ بشكل غير منفصل وكعملية واحدة، ولكن عمليًا: قد تحدث بعض من الأعطال كتلك التي أُشير إليها في القسم 9.6 تحول دون تنفيذ الخطوات كعملية واحدة، الأمر الذي ينتج عنه بالتأكيد حالة غير متناسقة. لذلك، هناك عدة تطبيقات تسعى لمعالجة مثل هذه الحالات، يُلقي هذا القسم نظرة على بعض من الأساليب التي تضمن تعزيز موثوقية نظم ملفات القرص.

لدى معظم الحواسيب برامج خدمية لفحص تناسق أنظمة الملفات مثل، برنامج **fsck** في نظام 'يونكس'، وبرنامج **chkdsk** أو **scandisk** في نظام 'ويندوز'، تشتغل مثل هذه البرامج في وقت الاستنهاض وخاصةً إذا لم يُغلق نظام الملف بشكل صحيح، حيث تبحث هذه الأدوات على حالات عدم التناسق داخل أنظمة الملفات والمتمثلة في:

- الحالات المتعلقة بمقاطع الملفات
  - وجود مقاطع للقرص غير مخصصة للملفات وفي نفس الوقت غير مدرجة في قائمة المقاطع الحرة.
  - وجود مقاطع للقرص مخصصة لملفات وفي نفس الوقت مُدرجة في قائمة المقاطع الحرة.
  - وجود مقاطع للقرص مخصصة لأكثر من ملف واحد في نفس الوقت.
  - عدد مقاطع القرص المخصصة لملف– ما– غير متوافق مع الحجم المحدد للملف.
- الحالات المتعلقة بالمجلدات
  - وجود اسمين متطابقين أو أكثر للملفات في نفس المجلد.
  - وجود ملفات أو عقد فهرسة مخصصة بشكل صحيح، ولكن لا تظهر في أي مدخل من مداخل المجلد.
  - عدد الارتباطات لعقدة الفهرسة لا يتطابق مع عدد المراجع التي تشير إلى هذه العقدة داخل بنية المجلد.
  - وجود مجلدات مرتبطة بشكل غير قانوني، أو وجود ملفات/مجلدات لا يمكن الوصول إليها من جذر شجرة المجلدات.

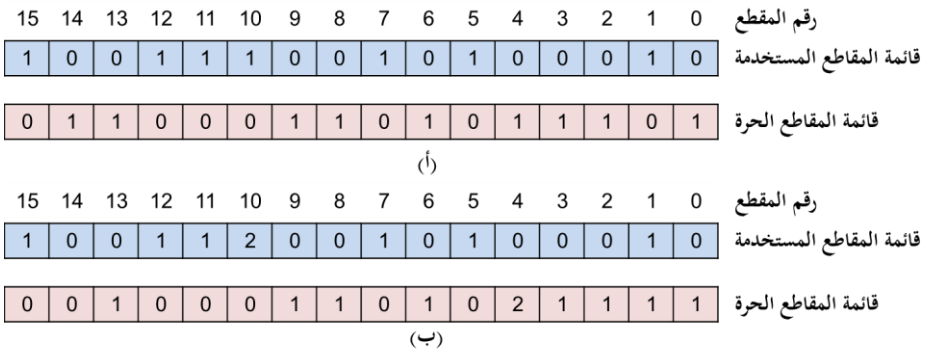
فيما يلي سيُقدم بعض من الأمثلة لتوضيح الكيفية التي تكتشف بها البرامج الخدمية في نظام 'يونكس' حالات اللاتوافق الخاصة بمقاطع الملفات داخل أنظمة الملفات.

لفحص توافق المقاطع يُنْشِأ برنامج الفحص جدولين، يحتوي كل منهما على عدَّاد خاص بكل مقطع، الغرض من عدَّادات الجدول الأول هو تتبع عدد المرات التي يظهر فيها المقطع المعني داخل الملف المحدد، بينما تتبع عدَّادات الجدول الثاني عدد المرات التي يظهر فيها المقطع داخل قائمة المقاطع الحرة، تُضبط مبدئيًا جميع هذه العدَّادات على صفر.

يبدأ البرنامج بقراءة جميع عقد الفهرسة، وانطلاقًا من أي عقدة يكون بالإمكان بناء قائمة بأرقام المقاطع المستخدمة داخل الملف، وعند قراءة أي رقم يزداد العدَّاد المناظر له داخل الجدول الأول، بعد ذلك يقوم البرنامج بقراءة قائمة المقاطع الحرة، وبنفس الطريقة يزداد العدَّاد المناظر لكل مقطع يجده داخل الجدول الثاني. في النهاية تُقارن محتويات الجدولين، فإذا وَجد

البرنامج أن لكل مقطع هناك 1 فقط يناظره في أحد الجدولين كما هو الحال في الشكل 6. 30–أ، فهذا يعني أن نظام الملف متوافق، أي متناسق.

بالمقابل، عندما يكون هناك خلل في نظام الملف فستكون هناك عدة احتمالات لوجوده، كما هو موضح في الشكل 6. 30–ب. يتمثل الاحتمال الأول في ظهور المقطع مُقيد في كلا الجدولين، كما هو الحال بالنسبة للمقطع رقم واحد. هذه الحالة تعني أن هذا المقطع مخصص لأحد الملفات وفي نفس الوقت لازال مقيدًا بقائمة المقاطع الحرة، تُعتبر مثل هذه الأخطاء من الأخطاء البسيطة وحلها يكمن في إعادة ضبط القائمة الحرة.



الشكل 6. 30: حالات نظام الملف أ) حالة متناسقة، ب) حالات غير متناسقة.

الاحتمال الثاني يتمثل في ظهور المقطع أكثر من مرة في قائمة المقاطع الحرة، كما هو الحال بالنسبة للمقطع رقم أربعة والذي ظهر مرتين. هذه الحالة تظهر فقط في حالة تمثيل القائمة الحرة باستخدام القوائم المرتبطة، أما في حالة استخدام خرائط الثنائية فمن المستحيل ظهور ذلك. إن حل هذا الخلل يتمثل في إعادة بناء هذه القائمة من جديد.

الاحتمال الأسوأ في حالات عدم التناسق يتمثل في ظهور نفس المقطع في أكثر من ملف، كما هو الحال بالنسبة للمقطع رقم عشرة والذي ظهر مرتين، بمعنى ظهر في ملفين. إذا مُسح أحد الملفين، فسيُوضع هذا المقطع في القائمة الحرة وبالتالي سينتج الاحتمال الأول وهو ظهور نفس المقطع في كل من القائمة الحرة والمستخدمة. أمَّا في حالة مسح هذين الملفين، فسينتج الاحتمال الثاني وهو تكرار ظهور نفس المقطع في القائمة الحرة. يكمن الحل هنا في عمل نسخة من هذا المقطع داخل أحد المقاطع غير المستخدمة وإدراجه في أحد الملفين. بهذه الطريقة لن

يتغير محتوى الملف وسيبقى نظام الملف متناسق، ولكن على حساب الملف الثاني والذي بالتأكيد سيتضرر. في هذه الحالة، من الضروري إخطار المستخدم بهذ الخلل، لكي يتخذ الإجراءات المناسبة.

أبسط احتمالات عدم التوافق في نظام الملف هو ظهور ما يُعرف باسم المقطع المفقود أي عدم ظهور المقطع في أيٍ من القائمتين، كما هو الحال بالنسبة للمقطع رقم 14. في الواقع لا يُعتبر هذا الخلل أمرًا خطيرًا، إلَّا إنَّه سيتسبب في ضياع مساحة من القرص، وكحل لهذا الخطاء يُضاف هذا المقطع إلى قائمة المقاطع الحرة.

بالإضافة إلى فحص المقاطع السالفة الذكر، فإنه بالإمكان أيضا فحص المجلدات وذلك عن طريق تكوين جدول من العدَّادات الخاصة بالملفات المتواجدة داخل كل مجلد، يبدأ الفحص من المجلد الجذر ويتكرر مع بقية المجلدات الفرعية في شجرة المجلدات حتى يُفحص كل مجلد في نظام الملف. لأي عقدة فهرسة داخل كل مجلد، يزداد العدَّاد وفقًا لظهور الملفات بها. هنا يجب أن ننوه إلى أنه بسبب الارتباطات المادية– راجع القسم 1.3.6– قد يظهر ملف في مجلدين أو أكثر وأنه لا يتم احتساب الارتباطات المنطقية ولا تتسبب في زيادة العدَّاد للملف الهدف.

في نظام الملف المتوافق، سيتوافق كلا العدَّادين، وإلَّا سيكون هناك خلل– ما– من الممكن ظهوره، قد يتمثل في أن يكون العدَّاد يحتوي على قيمة إمَّا أكبر أو أقل من العدد الفعلي للملفات داخل المجلد. إذا كانت قيمة العدَّاد المحسوب أكبر من عدد الملفات الفعلية، فحتى إذا تمت إزالة جميع الملفات من المجلدات، فسيظل العدد غير صفري ولن تُزال عقدة الفهرسة. هذا الخطأ ليس مؤثر، لكنه يتسبب في هدر مساحة من القرص، ولإصلاحه يجب تحديث عدد الملفات في عقدة الفهرسة بالقيمة الصحيحة.

الخطأ الأكثر خطورةً يتمثل في كون العدَّاد يحمل قيمة أقل من العدد الفعلي للملفات. فمثلًا، إذا كانت النتائج تُشير إلى أن عقدة الفهرسة تحوي ملفًا واحدًا فقط، في حين أن العدد الفعلي يدل على وجود ملفين، بالتالي عند حذف أي مدخل من المجلد، ينتقل عدَّاد عقدة الفهرسة إلى الصفر، وهو ما يعني أن نظام الملف سيُعلِّم هذه العقدة على أساس أنها غير مستخدمة، وسيُحرِّر كافة المقاطع الخاصة بها. هذا الإجراء ينتج عنه أن أحد المجلدات يُشير

الآن إلى عقدة فهرسة غير مستخدمة، والتي قد يُخصص مقاطعها قريبًا إلى ملفات أخرى. مرة أخرى، يكمن الحل هنا في ضبط عدد الارتباطات في عقدة الفهرسة بالعدد الفعلي لمداخل المجلد.

12.6 أمثلة لنظم الملف الواقعية

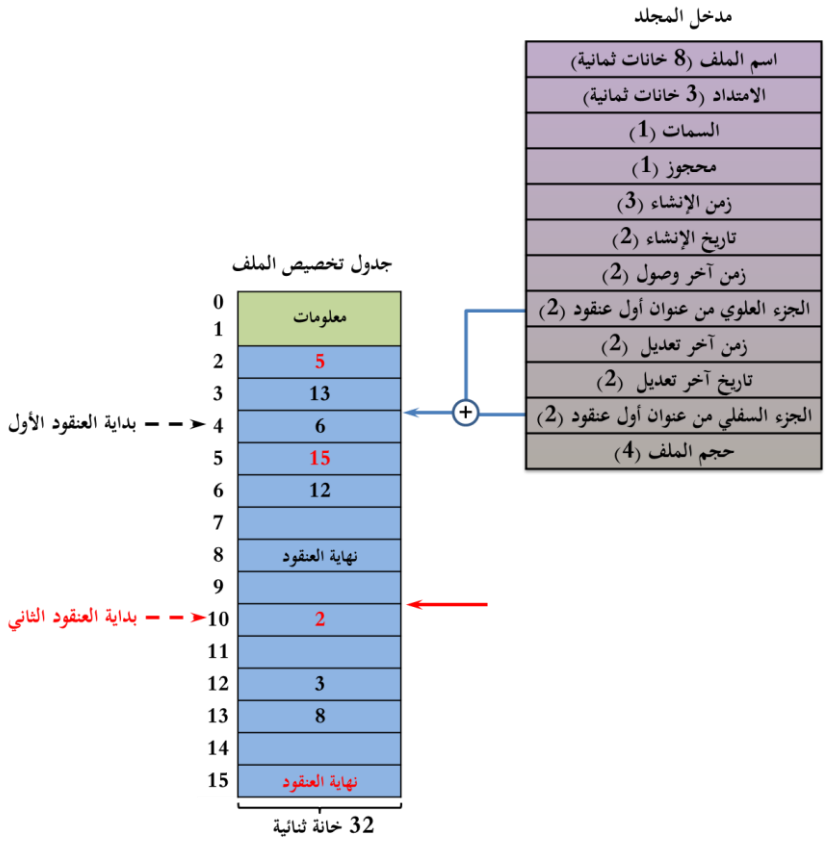
عادة ما يُقسم محرك الأقراص إلى عدة أجزاء بطريقة منظمة، لغرض الاستفادة منه بشكل أمثل. على سبيل المثال هناك مقطع الاستنهاض مُخصص له عنوان ثابت ويحتوي على برامج الاستنهاض المبدئي للنظام، وهناك مقطع التحكم في الأجزاء، والذي يصف الأجزاء التي تليه، أيضًا هناك جزء أو أكثر لاحتواء أنظمة تشغيل مختلفة، يُمثل أحدها الجزء النشط ويحتوي على نظام التشغيل المراد تمهيده. تسمح هذه البنية بتثبيت عدة أنظمة تشغيل على نفس محرك القرص الواحد، ولكن كل منها في جزء منفصل. تسمح هذه البنية أيضًا بالتحكم في كل جزء عن طريق نظام ملفات مختلف يستخدم استراتيجيات مختلفة ويُوفر عدة ميزات تختلف باختلاف النظام. يُقدم هذا القسم أمثلة لبعض من أنظمة الملفات هذه.

1.12.6 نظام جدول تخصيص الملفات الخاص بميكروسوفت

يُعتبر نظام جدول تخصيص الملفات ويُرمز له بالاختصار **FAT16** أو **FAT** من أبسط أنظمة الملفات، وأقدمها، وأكثرها شيوعًا، والذي صُمِّم من قبل شركة ميكروسوفت ليخدم نظام 'إم إس دوس'. أُنشأ هذا النظام في أوائل الثمانينيات بسعة تقديرية لحجم القرص كحد أقصى 2 قيقابايت، ويستخدم عناوين بطول 16 خانة ثنائية لتحديد مواقع الملفات، وهناك نسخة تستخدم 12 خانة ثنائية، ولكن سرعان ما تجاوزت الشركات المصنعة لمحركات الأقراص هذا الحجم في منتصف التسعينيات، الأمر الذي حتَّم على شركة الميكروسوفت إنتاج عدة نسخ منه تتمثل في **FAT32**، و **exFAT** من أجل مواكبة السعات المتزايدة لأحجام الأقراص، والتطورات المتلاحقة لأنظمة تشغيل 'ويندوز'.

يتوافق نظام جدول تخصيص الملفات مع أغلب أنظمة تشغيل ميكروسوفت ويُستخدم لغرض تتبع مكان العثور على الملفات على القرص، كما هو موضح في الشكل 6. 31، أي عن طريق استخدام القوائم المرتبطة (العناقيد المرتبطة) بحيث يُشير مدخل المجلد إلى مدخل– ما– في

جدول تخصيص الملفات والذي بدوره يُشير إلى أول عنقود. إذا كان هناك عنقود ثاني، فإن المدخل الأول في جدول تخصيص الملفات يُشير إلى المدخل الثاني والذي سيُشير بدوره إلى العنقود الثاني، وهكذا. بنفس هذه الآلية يُدير نظام جدول تخصيص الملفات الأماكن غير الشاغرة على القرص.



الشكل 6. 31: مثال لجدول تخصيص الملفات.

يتواجد جدول تخصيص الملفات في بداية وحدة التخزين، ويُحتفظ بنسخة احتياطية منه لاستخدامها في حال تعطلت النسخة الأولى، هذا بالإضافة إلى ضرورة تخزين كل من هذا الجدول والمجلد الجذر في موقع ثابت، حتى يُمكن تحديد موقع ملفات الاستنهاض للنظام بشكل صحيح.

يشتغل جدول تخصيص الملفات تحت قيود شديدة تتمثل من جهة في انحصار أسماء جميع الملفات الموجودة على النظام في ثمانية أحرف وامتداد مكون من ثلاثة أحرف، ومن جهة أخرى في عدم قدرته على دعم محركات الأقراص ذات السعات الأكبر من 2 قيقابايت (64 كيلوبايت من المداخل وبحجم 32 كيلوبايت لكل منهما). لذلك توقف استعمال نظام FAT16 مع نظام 'ويندوز 95'حيث أن الإصدار الأخير منها اعتمد على نظام FAT32.

لتفادي مثل هذه العيوب طُوِّر FAT16 إلى FAT32، والذي يستخدم 32 خانة ثنائية لعنونة الملفات وبإمكانه التعامل مع محركات أقراص يُمكن أن تصل أحجام كبيرة، أمَّا في حالة تعدي الحجم النهائي فيتحتم على نظام الملف تقسيم القرص إلى عدة أجزاء شرط ألَّا يزيد حجم أي جزء منها عن الحد الأقصى المسموح به. يُوضح الجدول 6. 6 أهم الفروقات بين هاذين النظامين.

الجدول 6. 6: أهم الفروقات بين نظامي FAT16 و FAT32.

الخاصية	نظام FAT16	نظام FAT32
الاستخدام	حجم القرص من صغير إلى كبير	حجم القرص من متوسط إلى كبير جدًا
عناوين الملفات	16 خانة ثنائية	32 خانة ثنائية
العدد الأقصى للعناقيد	65520	4177918
حجم العنقود	من 512 بايت إلى 64 كيلوبايت	من 512 بايت إلى 16 كيلوبايت
الحجم الأقصى	2 قيقابايت ويصل إلى 4 قيقا في بعض من نظم التشغيل.	32 قيقابايت ويصل إلى 2 تيرا في بعض من نظم التشغيل.
الحجم الأقصى للملف	2 قيقابايت	4 قيقابايت
طول اسم الملف	8 أحرف وقابل أن يصل إلى 255	يصل إلى 255

هناك أيضًا إصدار متقدم وحديث من نظام FAT32 يدعم تخزين المزيد من الملفات في كل مجلد مقارنةً به يتمثل في نظام exFAT، وهو يعني جدول تخصيص الملفات الموسَّع، والذي صُمِّم لدعم بطاقات الذاكرة، وأجهزة الناقل التسلسلي العام، بالإضافة إلى أجهزة الوسائط

مثل مشغلات الوسائط المحمولة، وأجهزة التلفزيون ذات الشاشات المسطحة، وما إلى ذلك.

بالرغم من الجهود المبذولة في تطوير نظام جدول تخصيص الملفات، إلَّا إنَّه يُعاني من مشكلة ضياع في المساحة التخزينية نتيجةً لظاهرة التفتت الداخلي والخارجي لها، كما يُعاني أيضًا من عدم قدرة النظام على توفير الحماية بالنسبة للملفات، ففي هذا النظام لا توجد أي إمكانية لمنع المستخدميين من الوصول إلى كافة الملفات على محرك الأقراص حتى لو اختلفت الحسابات الخاصة بهم (مشرف أو عادي)، كما أنه لا يُوفر خاصية الضغط بالنسبة للملفات.

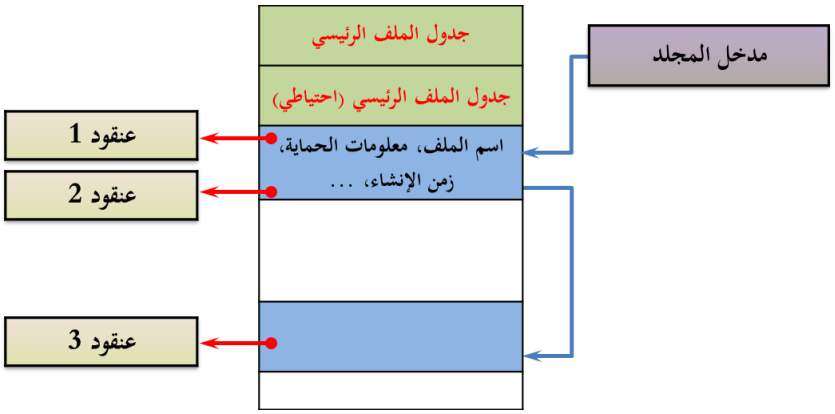
2.12.6 نظام ملفات التقنية الجديدة

لمعالجة قصور أنظمة جداول تخصيص الملفات من حيث حدودية حجم الملفات، وعدم دعمها لمحركات أقراص ذات سعات كبيرة جدًا، وضِعف ميزات الأمن والحماية، طُوِّر نظام ملفات التكنولوجيا الجديدة أو نظام 'إن تي إف إس' كجزء من تطور نظام 'ويندوز إن تي' والذي دُعِّمت به أغلب أنظمة 'ويندوز' المتلاحقة مثل، 'ويندوز 2000'، و'إكس بي'، ... إلخ.

يتكون فضاء (جزء من القرص) نظام ملفات التقنية الجديدة من ملفات، ومجلدات، وخرائط الثنائية والتي يستخدمها لتتبع العناقيد الحرة، إضافةً إلى بعض من هياكل البيانات الأخرى، والذي يتمثل أهمها في جدول الملف الرئيسي وهو يُعتبر قلب هذا النظام ويحتوي على مجموعة من المداخل (يُطلق عليها سجلات) المتتالية بحجم ثابت مقداره واحد كيلوبايت لكل مدخل، حيث يتحدد عدد المداخل بناءً على عدد الملفات والمجلدات المستحدثة في القرص. يصف المدخل الأول من هذا الجدول جدول الملفات الرئيسية نفسه ويُلحق بمدخل احتياطي له يَستخدمه نظام التقنية الجديدة كبديل في حالة تضرر المدخل الأول. في هذه الحالة يستخدم النظام النسخة الاحتياطية عند التشغيل التالي للجهاز ويُنشئ تلقائيًا نسخة جديدة وهو ما يضمن حفظ البيانات من الضياع أو التلف.

يصف جدول الملف الرئيسي أيضًا جميع الملفات، وبالتالي جميع الكائنات المخزنة على القرص عن طريق عدة حقول للبيانات تتضمن سمات الملف، ونوعه، وحجمه، وزمن وتاريخ إنشائه، ومعلومات الحماية، وكذلك قائمة بعناوين القرص المخصصة للعناقيد والتي ستُشير إلى

مكان وجود الملف، كما هو موضح في الشكل 6. 32. يوضح هذا الشكل عينة لبنية نظام التقنية الجديدة لنظام الملف، إذا كان حجم الملف صغير، فستكون محتوياته فعليًا ضمن حدود المدخل، أما إذا كان حجمه كبير جدًا، فسيكون من الضروري استخدام أكثر من مدخل للاحتفاظ بقائمة جميع مقاطعه، في هذه الحالة يُعرف المدخل الأول بالمدخل الأساسي ويُشير إلى بقية مداخل جدول الملف الرئيسي.



الشكل 6. 32: عينة لبنية نظام التقنية الجديدة لنظام الملف.

يُوضح الشكل 6. 33 نسخة مبسطة لبنية مدخل جدول الملف الرئيسي والتي تتكون من عدة حقول قابلة للتوسع وتشمل سمات مُعرَّفة من قِبل المستخدم، تتمثل في الآتي:

- السمات الأساسية: تتضمن هذه السمة المعلومات الأساسية في نظام 'إم إس دوس' كأذونات القراءة والكتابة، وزمن الإنشاء، وزمن التعديل الأخير، ... إلخ.
- سمة اسم الملف: تصف اسم الملف في مجموعة من الأحرف قد تُمثل أسماء ملفات متعددة تحتوي على روابط متعددة، أو أسماء قصيرة كما هو الحال في نظام 'إم إس دوس'.
- سمة واصف الحماية: تُوضح هذه السمة مالك الملف ومن له حق الوصول إليه (وكيف يمكن الوصول إليه).
- حقل البيانات: كما أشرنا سابقًا يحتوي هذا الحقل إمَّا على بيانات الملف الفعلي في حالة ما إذا كان الملف صغير، أو على مؤشرات للبيانات، بدلًا من البيانات نفسها في حالة الملفات الكبيرة.

سمات أساسية	اسم الملف أو المجلد	سمات واصف الحماية	البيانات أو مؤشرات

الشكل 6. 33: نسخة مبسطة لمدخل جدول الملف الرئيسي.

من المزايا التي يُقدمها نظام ملفات التقنية الجديدة أنه يُؤمِّن درجة أعلى من الحماية عن طريق تشفير الملفات، وخاصية استردادها والتي تُساعد في سرعة استعادة المعلومات عند حدوث عطل مفاجئ في الطاقة، أو عند حدوث مشكلة أخرى في النظام، لكونه يُخزن معلومات عناقيد الملفات ضمن كل عنقود.

إضافةً إلى ما سبق ذكره يُحقق نظام ملفات التقنية الجديدة كسبًا جيدًا في أداء النظام على العكس من نظام جدول تخصيص الملفات، بسبب استخدامه لخوارزمية بحث متطورة تُساعد بدرجة كبيرة في الوصول إلى الملفات على تقسيمات من فئة عدة قيقابايت وبسرعة كبيرة، وهو ما يُعتبر مفيدًا بالنسبة لقواعد البيانات الضخمة، كما يُتيح إمكانية ضغط الملفات دون أن تكون هناك حاجة إلى ضغط القرص بالكامل. إضافةً إلى كل هذا باستطاعة هذا النظام تعيين أذونات للملفات من شأنها أن تُحدد آلية الوصول إليها سواءً على مستوى المستخدمين أو المجموعات، وكذلك تحديد مستوى الوصول المسموح به. ينطبق هذا الأمر على كل من مستخدمي الجهاز الواحد وعلى مستخدمي الشبكة عندما يكون الملف في مجلد مشترك.

من ناحية أخرى وبالرغم من أن نظام ملفات التقنية الجديدة هو نظام حديث ومُدَعم من قبل أنظمة 'ويندوز' الحديثة، إلَّا إنَّه غير مناسب عندما تكون أغلب الملفات الموجودة على وحدات التخزين صغيرة، الأمر الذي قد يتسبب في حمل إضافي في إدارة النظام يؤدي في الغالب إلى انخفاض طفيف في الأداء مقارنةً بنظام جدول تخصيص الملفات. أيضًا يتطلب نظام ملفات التقنية الجديدة قدرًا كبيرًا من المساحة التخزينية مُخصصة لنظام الملف نفسه قد تصل إلى 4٪ أو أكثر من مساحة محرك الأقراص بسبب احتياجاته، الأمر الذي سيُقلل من إجمالي مساحة التخزين الخاصة بالملفات الفعلية. بالتالي قد تؤدي محاولة تحميل نظام ملفات التقنية الجديدة على أجهزة تخزين صغيرة كمحرك أقراص الناقل التسلسلي العام، ومحركات الأقراص القديمة مثل الأقراص المرنة إلى ضعف الأداء أو عدم كفاية مساحة التخزين. إضافةً إلى ذلك يُعاني نظام التقنية الجديدة من عدم توافقه مع الإصدارات القديمة لأنظمة تشغيل ميكروسوفت التي سبقت

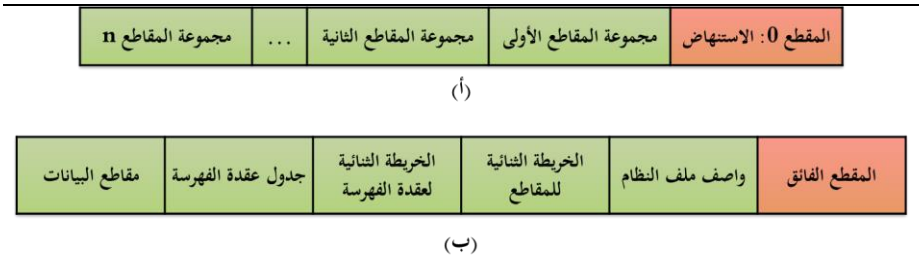
'ويندوز 2000'.

3.12.6 نظام الملف الموسَّع الثاني، والثالث 'إِ إكس تي 2، 3' الخاص 'بلينُكس'

استَخدمت الإصدارات الأولى من نظام 'لينُكس' نظام ملفات متوافق مع نظام 'مينيكس'، ولكن بسبب حدودية كل من طول أسماء الملفات (14 حرف)، وحجم النظام (64 ميقابايت) سرعان ما أُستبدل بأول نظام ملفات موسَّع في عام 1992م، والذي سمح باستخدام أسماء ملفات أطول وصلت إلى 255 حرفًا، وأحجام ملفات أكبر وصلت إلى 2 قيقابايت، إلَّا إنَّ عدم كفاءة أدائه من حيث عدم توفر السمات الزمنية كزمن الإنشاء والتعديل، ومعاناته من ظاهر التفتت أدَّيا إلى إعادة تصميمه لتحسين أدائه وإضافة بعض الميزات المفقودة، الأمر الذي أفضى إلى ظهور النظام الموسَّع الثاني، والذي لا يزال يُستخدم على نطاق واسع.

نظام الملف الموسَّع الثاني صُمم في البداية بواسطة ريمي كارد (Rémy Card) في سنة 1993، وأصبح أول نظام ملفات تجاري لنظام 'لينُكس' مُعتمد كنظام افتراضي للعديد من نسخ هذا النظام وقادرًا على إدارة الملفات بحجم يصل إلى 4 تيرابايت، وهو ما سمح باستخدام أقراص ذات أحجام كبيرة دون أن تكون هناك حاجة إلى إنشاء العديد من الأقسام، كما أنه سمح باستخدام مداخل مجلدات متغيرة الطول تصل إلى 255 حرفًا، مع قابلية التمديد إلى 1012 إذا لزم الأمر.

يُقسم نظام الملف الموسَّع الثاني إلى مقاطع تُجْمَع في مجموعات ولا ترتبط بالتخطيط المادي لمقاطع القرص، لكي يتسنى لمحركات الأقراص الحديثة تحسين الوصول التسلسلي وإخفاء هندستها المادية عن نظام التشغيل. يُخصص المقطع صفر لبرنامج استنهاض الحاسوب وتتوالى بعده هذه المجموعات، كما هو الحال في الشكل 6. 34–أ. يحتوي الملف الكبير على ألوف المقاطع والتي عادةً ما تُضمَّن بياناته داخل مجموعة واحدة من المقاطع كلما أمكن ذلك، وذلك لغرض تقليل عدد مرات البحث داخل القرص عند قراءة كميات كبيرة من البيانات المتجاورة.



الشكل 6. 34: أ) تخطيط القرص لنظام الملف الموسَّع الثاني– ب) بنية مجموعة المقاطع.

من ناحية أخرى تحتوي كل مجموعة مقاطع على نسخة من المقطع الفائق، وجدول لوصفها، ونسخة من الخريطة الثنائية للمقاطع، وأخرى لعقد الفهرسة، بالإضافة إلى جدول لعقد الفهرسة، وأخيرًا مقاطع البيانات الفعلية، كما هو موضح في الشكل 6. 34–ب. فيما يلي سيُقدم شرح موجز لهذه المكونات بدايةً من اليمين إلى اليسار.

- المقطع الفائق: يحتوي هذا المقطع على بيانات التحكم الهامة– حول تخطيط نظام الملف– والضرورية لتشغيل نظام التشغيل، تتمثل في عدد عقد الفهرسة، وعدد مقاطع القرص، وبداية قائمة المقاطع غير المستخدمة داخل القرص. لذلك نجد أن هذا المقطع يتكرر في جميع مجموعات المقاطع كنسخ احتياطية، لزيادة الموثوقية وتسهيل عملية الاسترداد لنظام الملف، إلَّا إنَّه في العادة ما تُستخدم النسخة الموجودة في مجموعة المقاطع الأولى في عملية التمهيد.
- واصف ملف النظام: يندرج هذا المقطع أيضًا تحت معلومات التحكم والتي تُوضح موقع الخريطة الثنائية لكل من المقاطع وعقد الفهرسة، وبداية جدول عقد الفهرسة لكل مجموعة مقاطع. إضافةً إلى ذلك يحتوي هذا الواصف على عدد كل من المقاطع غير المستخدمة، عقد الفهرسة، وكذلك المجلدات داخل المجموعة.
- الخريطة الثنائية للمقاطع، وعقد الفهرسة: الغاية والغرض من هاتين الخريطتين هو تتبع المقاطع، والعقد غير المستخدمة، على التوالي. يؤدي تحديد حجم هاتين الخريطتين إلى تحديد حجم مجموعة المقاطع، والعقد في المجموعة التي تنتمي إليها، مع استخدام خريطة بحجم واحد كيلوبايت سيكون الحد الأقصى لعدد المقاطع هو 8192 وللعقد 8192، أمّا في حالة استخدام خريطة بحجم 4 كيلوبايت، فسيتضاعف العدد بمقدار أربع مرات.

- جدول عقد الفهرسة: يُستخدم جدول عقد الفهرسة لتتبع كل مجلد، أو ملف، أو أي ارتباط رمزي، ويتكون من سلسلة من المقاطع المتتالية، والذي يحتوي كل منها على عدد محدد مسبقًا من عقد الفهرسة. تُرقم هذه العقد من 1 إلى الحد الأقصى لعددها، طول كل عقدة هو 128 خانة ثمانية وتصف بالضبط ملف واحد، كما أنها تتضمن بيانات حول حجم الملف أو الدليل، وإذن ملكيته، بالإضافة إلى معلومات كافية لتحديد موقع كافة مقاطع القرص التي تحتفظ ببيانات الملف.
- مقاطع البيانات: نظام الملف الموسَّع الثاني مبني على فرضية حفظ البيانات في هذه المقاطع والتي تحوي كافة المجلدات والملفات، لذا هذه المقاطع نفس الطول، إلَّا إنَّه قد يختلف باختلاف إصدارات النظام، كما أنه لا يشترط التخزين المتجاور لمقاطع الملف الواحد أو المجلد الواحد إذا احتاجا لأكثر من مقطع واحد، لذلك قد تجد مقاطع الملفات الكبيرة منتشرة في جميع أنحاء القرص.

كما هو الحال في الأنظمة السابقة الذكر، تتطور أنظمة الملفات الموسَّعة أيضًا لتوفر ضمانات مختلفة، ولتواكب التطور في كل من أحجام الأقراص، وأنظمة التشغيل، وخصوصًا مع وجود بيئة المصدر المفتوح لنظام 'لينُكس'. هذا التطور أنتج عدة أنظمة من بينها نظام الملف الموسَّع الثالث، والرابع والتي تُعتبر من أكثر الأنظمة الشائعة والمستقرة في بيئة خوادم 'لينُكس'، والتي صُممت بمبادئ مختلفة لتوفير تنسيقات، وخيارات إرفاق متنوعة. يُوضح الجدول 6. 7 مقارنة بسيطة لهذه الأنظمة.

لمزيد من التفاصيل حول نظام الملف الموسَّع الثالث، والرابع، يُمكن الرجوع إلى تانن باوم وآخرون (Tanenbaum et. al, 2015)، وزلبرشاتس وآخرون (Silberschatz et. al, 2018).

الجدول 6. 7: مقارنة بسيطة لأنظمة الملفات الموسَّعة.

نظام الملف الموسَّع الثاني	نظام الملف الموسَّع الثالث	نظام الملف الموسَّع الرابع
طُوِّر في سنة 1993	طُوِّر في سنة 2001	طُوِّر في سنة 2008
الحد الأقصى لحجم الملف يتراوح ما بين 16 قيقا و 2	الحد الأقصى لحجم الملف يتراوح ما بين 16 قيقا و 2	الحد الأقصى لحجم الملف يتراوح ما بين 16 قيقا و 16 تيرابايت.

نظام الملف الموسَّع الثاني	نظام الملف الموسَّع الثالث	نظام الملف الموسَّع الرابع
تيرابايت.	تيرابايت.	
إجمالي حجم نظام الملف ما بين 2 تيرا و 32 تيرابايت.	إجمالي حجم نظام الملف ما بين 2 تيرا و 32 تيرابايت.	إجمالي حجم نظام الملف يصل إلى 1024 بيتابايت.

4.12.6 أنظمة ملفات أخرى

لا تقتصر أنظمة الملفات فقط على ما سبق ذكره بل تتعدى لتشمل عدة أنظمة أخرى، نذكر منها على سبيل المثال لا الحصر ما يلي:

- نظام ملفات الشبكة: يسمح نظام ملفات الشبكة (والأنظمة المشابهة الأخرى) بمشاركة المجلدات والملفات بين الحواسيب عبر شبكة، بحيث يمكن للمستخدمين والبرامج الوصول إلى الملفات على الأنظمة البعيدة كما لو كانت على الحاسوب الشخصي للمستخدم، فهو لا يهتم بتخصيص مساحات القرص، ولكنه يُركز على توفير آليات مختلفة للوصول للملفات والمجلدات عن بعد.
- أيزو 9660: يسمح هذا النظام بتنسيق البيانات الموجودة على الأقراص المدمجة (المضغوطة) بشكل معياري بما يتوافق مع أنظمة التشغيل المختلفة، بحيث يسمح بقراءة البيانات بكل سهولة ويسر، سواءً أكان ذلك على حاسوب شخصي، أو جهاز 'ماك'، أو أي نظام حاسوبي رئيسي آخر، فهو يُخصص مساحة متجاورة لكل ملف على القرص.
- نظام ملفات 'يونكس': يُعرف أيضًا بنظام الملف السريع وهو من أكثر الملفات شيوعًا في أنظمة تشغيل 'بي إس دي'، و'سولاريس'، و'يونكس'، كما أن له إصدارات مختلفة مدعومة من قبل جميع عائلة نظام التشغيل 'يونكس'. حديثًا تميل تقنيات الحاسوب إلى استبدال هذا النظام بأنظمة تشغيل مختلفة مثل، ZFS في نظام 'سولاريس'، و JFS في أنظمة الملفات المشتقة من نظام 'يونكس'.

13.6 موجز الفصل

بدأ هذا الفصل بمدخل لنظم الملف ناقش فيه العيوب الكامنة وراء تخزين البيانات في العمليات متمثلةً في حدودية مساحة التخزين، وعدم بقاء البيانات حية لمدة طويلة، بالإضافة إلى

عدم مواكبة هذا التخزين لمتطلبات البرمجة المتعددة، ومن ثم عرَّج على متطلبات الحل المنشود لهذه العيوب والتي أوجزها في ضرورة توفير إمكانية لتخزين كميات كبيرة من المعلومات مع بقاؤها لفترات زمية طويلة، وأن تكون قابلة للمشاركة من قبل أكثر من معالج وفي نفس الوقت.

أفضت هذه المتطلبات إلى ضرورة تخزين البيانات على هيئة ملفات في وسائط التخزين الثانوية (الأقراص الصلبة، الأقراص المدمجة، ... إلخ.) والتي بدورها تطلبت وجود إدارة لتتبع آليات التخزين وتسهيل عمليات الوصول إلى هذه البيانات. في إطار نظم التشغيل عُرِفَت هذه الإدارة بإدارة نظم الملف وانبثقت منها عدة مهام أساسية تمحورت في مجملها حول إدارة كافة العمليات المتعلقة بالملفات والمجلدات، وتتبع كل من المساحات الحرة وإجراءات تخصيصها، والمستخدمة وإجراءات استرجاعها داخل وسائط التخزين الثانوية.

لذلك تعرض هذا الفصل إلى الكيفية التي يرى بها كل من مستخدمي، ومصممي الملفات الخصائص المتعلقة بالملفات، والعمليات المطبقة عليها بما في ذلك بنية الملفات، وطُرق الوصول إليها، كما ناقش أيضًا المجلدات وبنيتها، والعمليات المطبقة عليها، وكذلك توضيح مفهوم المسار من خلال تتبع التسلسل الهرمي لشجرة المجلدات. إضافة إلى ذلك، تناول الفصل مفهوم إرفاق وإلغاء تحميل نظام الملف والمستخدم في تثبيت أنظمة ملفات إضافية داخل النظام أو فصلها منه، والتي قد تتم تلقائيًا كما هو الحال في نظام 'ويندوز'، أو تحت إشراف المستخدم باستخدام أوامر محددة مثلما هو مطبق في نظام 'لينُكس'.

بعدها تطرق الفصل إلى آليات تنفيذ نظم الملف من خلال عرض طرق تنفيذ كل من الملفات والمجلدات، حيث ناقش التنفيذ الأول بالتفصيل آلية التخصيص المتجاور، والتخصيص باستخدام القوائم المرتبطة أو باستخدام فهرسة القوائم المرتبطة، وكذلك التخصيص عن طريق عقدة الفهرسة، أمَّا في ما يخص تنفيذ المجلدات فقد ناقش الفصل تنفيذها باستخدام كل من القائمة الخطية وعقدة الفهرسة.

مثلما هو الحال في إدارة تخصيص الملفات، فإنه من الطبيعي البحث عن طرق لإدارة الأماكن غير المخصصة حتى يسهل تتبعها فيما بعد، لذلك ناقش الفصل عدة طرق لإدارة فراغ القرص تمثلت في خرائط الثنائية، والقوائم المرتبطة، وتجميع العناوين، وكذلك العد.

من ناحية أخرى، وبما أن مشاركة الملفات بين المستخدمين تلعب دورًا مهمًا في تحقيق الأهداف المشتركة بينهم، ناقش هذا الفصل هذه الجزئية من خلال توضيحه لآليات المشاركة، والمشاكل لمتعلقة بها، والتحديات الكامنة في توسيع المشاركة إلى أنظمة الملفات المتعددة، بما في ذلك أنظمة الملفات عن بُعد. في النهاية أُختُتِمت هذه الجزئية بتوضيح أهم دلالات توافق مشاركة الملفات المتمثلة في دلالات كل من نظام "يونكس"، والجلسة، والملفات المشتركة غير القابلة للتغيير، وأخيرًا دلالات الأقفال.

يلي مناقشة دلالات توافق مشاركة الملفات تطرق الفصل إلى نظام الملف الظاهري والذي أتاح إمكانية الجمع بين أنظمة ملفات مختلفة لنظم تشغيل مختلفة داخل نظام التشغيل الواحد، الأمر الذي مَكَّن المستخدمين من التنقل بسلاسة بين هذه الأنواع المختلفة. بعد ذلك تعرض الفصل إلى مفهوم اعتمادية نظام الملف معرجًا على النسخ الاحتياطي والذي يُساعد في حفظ كل من بيانات المستخدم داخل النظام، وحالة النظام أو حالة تشغيله. في إطار هذا الموضوع ناقش الفصل عدة أساليب للقيام بعملية النسخ منها النسخ الاحتياطي الفعلي، والمنطقي.

من المواضيع الأخرى التي تناولها الفصل والتي تُساهم في زيادة اعتمادية نظام الملف وموثوقيتها هو توافقية نظام الملف. من هذه التوافقية ما هو متعلق بالملفات ومنها ما هو متعلق بالمجلدات، لذلك عندما لا تتوافق الجداول الداخلية أو البيانات الوصفية الخاصة بكليهما مع ما هو موجود فعليًا على القرص، يتأثر سلبًا تناسق أنظمة الملفات وبالتالي تنخفض موثوقيتها، ولتعزيز هذه الموثوقية طُوِّرت عدة تطبيقات تعمل على تَفحُّص الجداول الداخلية والبيانات الوصفية لغرض تحديد حالات عدم التناسق ومن ثم معالجتها كل حسب متطلباته.

أخيرًا، عرض هذا الفصل بعض من الأمثلة لنظم الملف الواقيعة، تمثلت في نظام جدول تخصيص الملفات، نظام ملفات التقنية الجديدة، ونظام الملف الموسَّع، بالإضافة إلى بعض من الأنظمة الأخرى.

14.6 أسئلة للمراجعة

1. عدد الأسباب التي وراء الحاجة إلى تخزين البيانات في الملفات.
2. ما هو نظام الملف وما الوظائف الرئيسية له؟

3. هل من الممكن تسمية الملفات بأي اسم؟ علل إجابتك؟
4. ما الغاية والغرض من إلحاق الملفات والمجلدات بسمات خاصة ؟
5. من المستحسن أن يدعم نظام التشغيل معظم بنيات الملفات، ولكن واقعيًا لا يتم تحقيق ذلك، لماذا؟
6. قارن بين مزايا وعيوب البنيات الثلاث الشائعة في نظم الشغيل.
7. ما الفرق بين سمة كتابة الملف وسمة إلحاق الملف؟
8. قارن بين مزايا وعيوب طرق الوصول إلى الملفات.
9. لماذا وُجدت الملفات؟ ولماذا تعددت بنياتها؟
10. ما فائدة اسم المسار، وما أنواعه؟
11. اشرح مفهوم إرفاق وإلغاء تحميل نظام الملف.
12. هناك عدة طرق مستخدمة في تتبع الأجزاء الخاصة بكل ملف، أذكرها مع شرح إحداها بنوع من الإيجاز، مع ذكر مزايا وعيوب هذه الطريقة.
13. لماذا سُحِبت المؤشرات من المساحات المخصصة لها داخل أجزاء الملف عند تمثيله باستخدام القوائم المرتبطة ووضعها داخل جدول في الذاكرة يُعرف باسم جدول تخصيص الملف؟
14. لماذا تُخزن عناوين المباشرة للمقاطع داخل عقدة الفهرسة نفسها؟
15. نظام ملفات يمتلك مقاطع بحجم 512 خانة ثمانية وله عناوين قرص بطول 4 خانات ثمانية. ما هو أقصى حجم يدعمه نظام الملف هذا، في حالة امتلاكه لعقدة فهرسة لها 10 عناوين مباشرة، ولعنوان غير مباشر مفرد، ولعنوان غير مباشر مزدوج، ولعنوان غير مباشر ثلاثي التضعيف؟
16. بماذا تمتاز المجلدات المنفذة باستخدام القوائم المرتبطة عن تلك المنفذة عن طريق القوائم الخطية؟
17. لماذا تتجه الأنظمة الحديثة إلى تنفيذ المجلدات باستخدام جدول هاش؟ وما الذي أدخله من تحسينات على المجلدات؟ وما هي عيوبه؟
18. إسوةً بالآلية الموضحة في الشكل 6. 21 بين مع الرسم خطوات تحليل المسار **/usr/ali/data/ds** في نظام "يونكس". ملاحظة: يمكنك استخدام أرقام عقد افتراضية

لمساعدتك في الحل.

19. عدد طرق إدارة فراغ القرص وقارن بينها من حيث المزايا والعيوب.

20. لماذا نحتاج إلى مشاركة الملفات بين المستخدمين؟ وضِّح بمثال.

21. ما المقصود بدلالات توافق مشاركة الملفات؟ ولماذا نحتاجها؟

22. تحدث بإسهاب عن نظام الملف الظاهري، موضحًا أهم وظائف طبقة نظام الملف الظاهري.

23. وضح كيف تسمح طبقة نظام الملف الظاهري لنظام التشغيل بدعم عدة أنواع من نظم الملف بسهولة.

24. ما المقصود باعتمادية نظام الملف؟

25. ما وظيفة النسخ الاحتياطي؟ وما الفرق بين النسخ الاحتياطي الفعلي والمنطقي؟

26. بفرض أن الملف رقم 8 في الشكل 6. 27 لم يتم تعديله منذ آخر عملية نسخ احتياطي، في هذه الحالة ما هي الصور التي ستكون عليها خرائط الثنائية الأربعة في الشكل 6. 28.

27. ما المقصود بتوافق نظام الملف؟

28. يوضح الشكل التالي أحد نتائج برامج اختبار توافقية نظم الملف:

رقم المقطع	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
قائمة المقاطع المستخدمة	2	1	0	1	1	0	0	1	1	0	0	1	0	0	1	0
قائمة المقاطع الحرة	0	0	1	0	0	0	1	1	0	1	2	0	1	1	0	1

من خلال الاطلاع على هذا الشكل، هل هناك عدم توافق في نظم الملف؟ في حالة الإجابة بنعم بين ما إذا كانت هذه الأخطاء بسيطة أم معقدة، وما هو الحل الذي يتناسب مع كل خطأ؟

29. عدد أهم الفروقات بين نظام جدول تخصيص الملفات ونظام ملفات التقنية الجديدة.

30. ما مزايا وعيوب نظام ملفات التقنية الجديدة؟

31. تحدث باختصار عن مكونات بنية مجموعة المقاطع في نظام الملف الموسَّع الثاني، موضحًا التحسينات التي أدخلتها مجموعة المقاطع على هذا النظام مقارنة بنظام لملفات البسيط.

32. يوضح الشكل التالي الحالة البدائية لخريطة ثنائية لقسم قرص بعد أول تهيئة له، المقطع الأول منها خُصص إلى المجلد الجذر.

رقم المقطع	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
الخريطة	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

بعد تخصيص مساحة بطول 6 مقاطع للملف 1، أصبح شكل الخريطة على النحو التالي:

رقم المقطع	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
الخريطة	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

وضح صورة الخريطة التي ستكون عليها بعد تنفيذ الأحداث التالية:

أ. كتابة الملف 2 بطول 5 مقاطع.

ب. مسح الملف 1.

ج. كتابة الملف 3 بطول 8 مقاطع.

د. مسح الملف 2.

33. لماذا يجب الاحتفاظ بخرائط الثنائية الخاصة بتخصيص الملفات في الذاكرة الثانوية، وليس في الذاكرة الرئيسية؟

34. بالنظر إلى نظام يدعم استراتيجيات التخصيص المتجاور، والقوائم المترابطة، وفهرسة القوائم المرتبطة. ما هي المعايير التي ينبغي استخدامها لتحديد الاستراتيجية المناسبة لأي ملف؟

معجم المصطلحات

معجم المصطلحات

المصطلح	الترجمة
ا	
'آبل أي أو إس'	Apple's iOS
'آبل ماكنتوش'	Apple Mac
اتصالات العملية الداخلية	Interprocess Communication
إجراء معالجة المقاطعة	Interrupt handler procedure
إجراءات خدمة المقاطعة	Interrupt service routines, ISRs
أجزاء	Partitions
أجهزة حرفية	Character devices
أجهزة مقطعية	Block devices
اختبار وضبط القفل	Test and set lock, TSL
إدارة فراغ القرص	Disk space management
إدخال، وإخراج بالمقاطعة	Interrupt-Driven I/O
إدخال، وإخراج مبرمج	Programmed I/O
إرفاق وإلغاء تحميل نظام الملف	Mounting and unmounting file system
استجابة	Response
استدعاء المكدس	Stack call
استدعاء النظام	System call
استدعاء نواة	Kernel call
استغلالية	Utilization
استقلالية الجهاز	Device independence
استنهاض الحاسوب	Computer booting
أسطوانة تخزين إضافية	Drum
اسم المسار المطلق	Absolute path name
اسم المسار النسبي	Relative path name
اسم مسار	Path name

Semaphore	إشارة
Acknowledgement	إشارة الإقرار أو الرد
Page frame	إطار صفحة
Relocation	إعادة تخصيص
Re-enable interrupt	إعادة تمكين المقاطعة
File system reliability	اعتمادية نظام الملف
Shortest remaining time next	أقصر الأوقات المتبقية تاليًا
Shortest process next	أقصر عملية تاليًا
Shortest job first	أقصر وظيفة أولًا
Java virtual machine	آلة جافا الظاهرية
Virtual machine	آلة ظاهرية
Extended machine	آلة موسعة
Physical device	إلكترونيات
MS-Dos	'إم إس دوس'
Extension	امتداد
Instruction	أمر
Shell command	أمر شيل
Trap instruction	أمر قفز
Security	أمن
Throughput	إنتاجية
Busy waiting	انتظار مشغول
Process creation	إنشاء عملية
Batch system	أنظمة الدفعة
Multiprocess systems	أنظمة العمليات المتعددة
Real time systems	أنظمة الوقت الحقيقي
Mainframe operating systems	أنظمة تشغيل الحواسيب المركزية
Interactive systems	أنظمة تفاعلية

Symmetric multiprocessing, SMP	أنظمة تناظرية متعددة المعالجات
Clustered systems	أنظمة عنقودية
Parallel clusters	أنظمة عنقودية متوازية
Clustering over a wide-area network	أنظمة عنقودية مرتبطة بشبكة موسعة
Multiprocessor systems	أنظمة متعددة المعالجات
Multicore systems	أنظمة متعددة النوى
Parallel systems	أنظمة متوازية
Remote file system	أنظمة ملفات عن بُعد
Hybrid systems	أنظمة هجينة
Normal termination	إنهاء عادي (طوعي)
Process termination	إنهاء عملية
Abnormal termination	إنهاء غير عادي (لا إرادي)
Priority	الأولوية
ISO 9660	'أيزو' 9660
Icon	إيقونة

ب

Byte	بايت/ خانة ثمانية
Depth-first search	بحث أولي متعمق
Microprogramming	برمجة دقيقة
Multiprogramming	برمجة متعددة
User-space I/O software	برمجيات الإدخال، والإخراج لفضاء المستخدم
Audio driver	برنامج تشغيل الصوت
Microprogram	برنامج مُصغر
MS Word	برنامج معالج النصوص
Object program	برنامج هدفي
Punch card	بطاقة تخريم
Printed circuit card	بطاقة دائرة مطبوعة

Smart card	بطاقة ذكية
Pixel	بكسل
Pentium	بنتيوم
File structure	بنية الملف
POSIX	'بوسكس'
	ت
Release	تحرير
Double buffering	تخزين لحظي مزدوج
Linked list allocation	تخصيص باستخدام القوائم المرتبطة
Linked list allocation using an index	تخصيص باستخدام فهرسة القوائم المرتبطة
Contiguous allocation	تخصيص متجاور
Compilation	ترجمة
Process hierarchies	تسلسل هرمي للعملية
Running	تشتغل
Full-disk encryption, fde	تشفير القرص بأكامله
Paging	تصفح
Interleaving	تعشيق
Suspend	تعليق
External fragmentation	تفتت خارجي
Internal fragmentation	تفتت داخلي
Segmentation	تقطيع
Enable interrupt	تمكين المقاطعة
Strict alternation	تناوب دقيق
Portable document format	تنسيق مستندات محمولة
Implementing directories	تنفيذ الأدلة
High-level format	تهيئة ذو مستوى عالي
Low-level format	تهيئة ذو مستوى منخفض

Pseudo-parallelism	توازي وهمي
Consistency	توافق
File system consistency	توافقية نظام الملف
Terabyte	تيرابايت
Dual-core	ثنائي النواة
ج	
Ready	جاهزة
Page tables	جداول الصفحة
Constant table	جدول الثوابت
Symbol table	جدول الرموز
Page table	جدول الصفحة
Inverted page table	جدول الصفحة المعكوس
Master file table, MFT	جدول الملف الرئيسي
File-allocation table, FAT	جدول تخصيص الملف
Extended file allocation table	جدول تخصيص الملفات الموسع
Process table	جدول عملية
Scheduling	جدولة
Preemptive scheduling	جدولة الاستباقية
Non-preemptive scheduling	جدولة غير الاستباقية
Fetch	جلب
Tablet	جهاز لوحي
Google's Android	جوجل أندرويد
ح	
Personal computer	حاسوب شخصي
Status	حالات
Process States	حالات العملية
Deadlock	حالة الجمود

Deterministic	حتمي
Periodic event	حدث دوري
Aperiodic event	حدث غير دوري
Sporadic event	حدث متفرق
Denial of service	حرمان من الخدمة
Trojan horse	حصان طروادة
Tight loop	حلقة محكمة
Protection	حماية
Multiple users computers	حواسيب متعددة المستخدمين
Mainframes	حواسيب مركزية
Cloud computing	حوسبة سحابية
خ	
Reincarnation server	خادم تناسخ
Control bits	خانات التحكم
Modified bit	خانة التعديل
Present/absent bit	خانة الحضور، والغياب
Referenced bit	خانة الرجوع
Valid bit	خانة الصحة
Wakeup waiting bit	خانة انتظار الصحوة
Bit	خانة ثنائية/ بت
Web service	خدمة ويب
Bitmaps	خرائط الخانات الثنائية
Page fault	خطأ الصفحة
Bug	خلل
Page replacement algorithms	خوارزميات استبدال الصفحة
The least recently used (LRU) page replacement algorithm	خوارزمية استبدال الصفحات المستخدمة مؤخرًا بقلة
The not recently used page replacement	خوارزمية استبدال الصفحات غير المستخدمة مؤخرًا

algorithm	
The optimal page replacement algorithm	خوارزمية الاستبدال الأمثل للصفحات
Shortest seek time first, SSTF algorithm	خوارزمية الأقصر زمن بحث أولًا
algorithm Next fit	خوارزمية البحث التالي عن الفراغ المناسب
algorithm Quick fit	خوارزمية البحث السريع عن الفراغ المناسب
algorithm Best fit	خوارزمية البحث عن أفضل فراغ مناسب
algorithm Worst fit	خوارزمية البحث عن أكبر فراغ مناسب
algorithm First fit	خوارزمية البحث عن أول فراغ مناسب
Scheduling algorithm	خوارزمية الجدولة
LOOK scheduling algorithm	خوارزمية الجدولة الناظرة
Fair-share scheduling algorithm	خوارزمية الحصة العادلة للجدولة
The first-in, first-out (FIFO) page replacement algorithm	خوارزمية الداخل أولًا يخرج أولًا لاستبدال الصفحات
The clock page replacement algorithm	خوارزمية الساعة لاستبدال الصفحات
algorithm. Not frequently used, NFU	خوارزمية الصفحات غير المستخدمة بشكل متكرر
Guaranteed scheduling algorithm	خوارزمية الضامن للجدولة
The second-chance page replacement algorithm	خوارزمية الفرصة الثانية لاستبدال الصفحات
First-come first-served, FCFS algorithm	خوارزمية القادم أولًا يُخدم أولًا
SCAN algorithm	خوارزمية المسح
Circular SCAN (C-SCAN) algorithm	خوارزمية المسح الدائري
Lottery scheduling algorithm	خوارزمية اليانصيب للجدولة
The wsclock page replacement algorithm	خوارزمية ساعة مجموعة العمل لاستبدال الصفحات
The working set page replacement algorithm	خوارزمية صفحات مجموعة العمل لاستبدال الصفحات
Thread	خيط
د	
Precise	دقيق

Session semantics	دلالات الجلسة
Consistency semantics	دلالات توافق
Spooler directory	دليل المكب
Large scale integration circuits	دوائر تكاملية كبيرة
Worms	ديدان
ذ	
Read only memory, ROM	ذاكرة القراءة فقط
Electrically erasable programmable, EEPROM	ذاكرة القراءة فقط القابلة للمسح والبرمجة كهربائيا
Memory-mapped I/O	ذاكرة المعنونة لوحدة الإدخال، والإخراج
Random-access memory, RAM	ذاكرة الوصول العشوائي
Dynamic, DRAM	ذاكرة الوصول العشوائي التفاعلية
Associative memory	ذاكرة ترابطية
Virtual memory	ذاكرة ظاهرية
Physical memory	ذاكرة فعلية
Memory cache	ذاكرة كاش (مخبأة)
ر	
Linker	رابط
Disk head	رأس القرص
Connectivity	ربط
ز	
Responsetime	زمن الاستجابة
Context time	زمن السياق (الفتح)
Context switching time	زمن تبديل السياق
س	
Register	سجل
Segments table base register, STBR	سجل أساس جدول المقاطع
Base register	سجل الأساس

Limit register	سجل الحد
Special device register	سجل خاص بالجهاز
Segments table length register, STLR	سجل طول جدول المقاطع
Program status word, PSW	سجل كلمة حالات البرنامج
Superscalar	سلّمي فائق
Attributes	سمات
File attributes	سمات الملف
Tape drive	سواقة الشريط
ش	
Virtual machine monitor	شاشة الآلة الظاهرية
Microsoft networks	شبكات ميكروسوفت
Local area network, LAN	شبكة محلية
Wide area network, WAN	شبكة واسعة
Quasi-parallel	شبه توازي
Parse tree	شجرة التحليل
Kernel source tree	شجرة مصدرية النواة
Time quantum	شريحة زمنية
Magnetic tape	شريط ممغنط
Error-correcting code, ECC	شفرة تصحيح الأخطاء
ص	
Page	صفحة
ض	
Clock tick	ضربة نبضية
Compaction	ضغط
Memory compaction	ضغط الذاكرة
ط	
Printer daemon	طابعة خفية

طلب	Request
ع	
عدَّاد البرنامج	Program counter, PC
عدالة	Fairness
عقدة فهرسة	I-node (index-node)
علاقة عديد- لعديد	Many-to-Many relationship
علاقة عديد- لواحد	Many-to-One relationship
علاقة واحد- لواحد	One-to-One relationship
عمليات أمامية	Foreground processes
عمليات تسلسلية	Sequential processes
عمليات خلفية	Background processes
عمليات صغيرة	Miniprocesses
عملية	Process
عملية الفتح	Process switch
عملية خفيفة الوزن	Lightweight process
عنقود	Cluster
عنوان ذاكرة فعلي	Physical memory address
عنوان ظاهري	Virtual address
عنوان فعلي	Physical address
غ	
غير حتمي	Nondeterministic
غير دقيق	Imprecise
غير دوري	A periodic
غير متزامن	Asynchronous
غير متناظر	Asymmetrically
غير محجوب	Non-blocking
ف	

فترة زمنية مستغرقة في التنفيذ	Turnaround time
فريد	Unique
فصل	Disabling
فصل المقاطعة	Disable interrupt
فضاء عنونة	Address space
فضاء عنونة ظاهري	Virtual space address
فهرسة وصول متسلسل	Indexed sequential access
فيروس	Virus
ق	
قارئ البطاقات	Card reader
قالب التحكم في الخيط	Thread control block, TCB
قالب التحكم في العملية	Process Control Block, PCB
قرص	Disk
قطاع	Sector
قَفل المرحلتين	Two-Phase locking
قناة الألياف	Fibre channel, FC
قوائم مرتبطة	Linked list
قيقابايت	Gigabyte
ك	
كلمة	Word
كيان مادي	Hardware
كيان معنوي	Software
ل	
لب دقيق	Micro-kernel
لغة الآلة	Machine language
'لينُكس'	Linux
م	

Mac OS X	'ماك أو إس إكس'
Swapping	مبادلة
Swap out	مبادلة إلى
Swap in	مبادلة من
Interrupt vector	متجه مقاطعة
Device controller	متحكم الجهاز
Compiler	مترجم
Translation look-aside buffer, TLB	مترجم جانبي للمخزن اللحظي
Multipass compiler	مترجم متعدد التمريرات
Synchronous	متزامن
Variable	متغير
Lock variable	متغير القفل
Symmetric multiprocessors	متماثل المعالجات المتعددة
Symmetrically	متناظر
Blocked	متوقفة
Starvation	مجاعة
Scheduler	مجدول
Current directory	مجلد حالي
Working directory	مجلد عمل
Directory	مجلد/ دليل
Folder	مجلد/ دليل
Acyclic graph directories	مجلدات رسم بياني لاحلقي
Simulating LRU in software	محاكاة استبدال الصفحات المستخدمة مؤخرًا بقلة باستخدام البرماجيات
Blocking	محجوب
Editor	محرر
Base station	محطة القاعدة
Workstation	محطة عمل

Modern workstations	محطة عمل حديثة
Secondary boot Loader	مُحمل الاستنهاض الثانوي
Bootstrap loader	مُحمل التمهيد
Buffer	مخزن لحظي
Entry	مدخل
Page table entry	مدخل جدول الصفحة
Memory manager	مدير الذاكرة
Resources manager	مدير المصادر
Network interface control	مراقب واجهة شبكات
Concurrently	مزامنة، في نفس الوقت
Storage area networks	مساحات التخزين الشبكية
Tracks	مسارات
Personal digital assistant, PDA	مساعد رقمي شخصي
Windows explorer	مستكشف 'ويندوز'
Record	مسجل
Time sharing	مشاركة زمنية
Master-slave multiprocessors	مشرف رئيسي معالجات ثانوية
Device driver	مشغل الجهاز
Producer-consumer problem	مشكلة المنتج والمستهلك
Resource	مصدر
Non-preemptable resource	مصدر غير قابل للسحب
Preemptable resource	مصدر قابل للسحب
Trap door	مصيدة الباب
Virtual processor	معالجات افتراضية
Multiprocessor	معالجات متعددة
Interrupt handlers	معالجة المقاطعات
Batch processing	معالجة بالدفعة

Parameters	معاملات
Identifier	مُعرِّف
Thread identifier	مُعرِّف الخيط
Process ID	معرِّف العملية
Security IDs, SIDs	مُعرِّف أمن
User identifiers, user IDs	مُعرِّفات مستخدم
Command interpreter	مفسر الأوامر
Process data segments	مقاطع بيانات العملية
Interrupt	مقاطعة
Precise interrupt and Imprecise interrupt	مقاطعة دقيقة وغير دقيقة
Clock interrupt	مقاطعة نبضية
Segment	مقطع
Data segment	مقطع البيانات
Stack segment	مقطع المكدس
Critical section	مقطع حرج
Superblock	مقطع فائق
Missing block	مقطع مفقود
Block	مقطع/كتلة
Print spooler	مكب الطباعة
Print spooler	مكب الطباعة
Stack	مكدس
Advanced technology attachment, ATA	ملحق التكنولوجيا المتقدمة
Lane	ممر
Terminated	منتهية
Platform controller hub	منصة مركز التحكم
Mutual exclusion	منع تبادلي
Sleep and wakeup	منوم ومنبه

مؤشر الربط	Chained pointer
مؤشر مكدس	Stack pointer, SP
مُوَصِّف	Descriptor
مُوَصِّف الخيط	Thread descriptor
مُوَصِّف العملية	Process descriptor
ميجابايت	Megabyte
'مينيكس 3'	MINIX 3
'مينيكس'	MINIX

ن

ناقل إشارات التحكم	Control bus
ناقل البيانات	Data bus
ناقل العنونة	Address bus
ناقل النظام	System bus
ناقل تسلسلي عام	Universal serial bus, USB
نبضة وحدة المعالجة	CPU burst
نسبية	Proportionality
نسخ احتياطي	Backup
نسخ احتياطي تزايدي	Incremental backup
نسخ احتياطي فعلي	Physical backup
نسخ احتياطي منطقي	Logical backup
نص مصدري	Source text
نظام 'آبل أي أو إس'	Apple's iOS
نظام 'أندرو'	Andrew system
نظام 'بي إس دى يونكس'	BSD, Berkeley system distribution, UNIX
نظام 'تايني أو إس'	TinyOS system
نظام 'سنو ليوبارد'	Snow Leopard system
نظام الإدخال، والإخراج الأساسي	Basic input output system, BIOS

Second and third extended file system, ext2, ext3	نظام الملفات الموسَّع الثاني، والثالث 'إ إكس تي 2، 3'
Hard real-time system	نظام الوقت الحقيقي الثابت
Soft real-time system	نظام الوقت الحقيقي المرن
Operating system	نظام تشغيل
Palm OS	نظام تشغيل 'بالم'
RIM's Blackberry operating system	نظام تشغيل 'بلاك بيري رِم'
Solaris operating system	نظام تشغيل 'سولاريس'
Symbian operating system	نظام تشغيل 'سيمبيان'
Smart card operating system	نظام تشغيل البطاقات الذكية
Batch operating system	نظام تشغيل الدفعة
Real-time operating system	نظام تشغيل الزمن الحقيقي
Conversational monitor system	نظام شاشة تحاورية
Uniprocessor system	نظام معالج وحيد
Virtual file system	نظام ملف ظاهري
New technology file system, NTFS	نظام ملفات التكنولوجيا الجديدة 'أو إن تي إف إس'
Network file system, NFS	نظام ملفات الشبكة
System V APis	نظام واجهات التطبيقات البرمجية
Run time system	نظام وقت التشغيل
Monolithic systems	نظم أحادية
Layered systems	نظم الطبقات
Networks operating systems	نظم تشغيل شبكات
Wireless sensor networks operating systems	نظم تشغيل شبكات التحسس اللاسلكية
Embedded operating systems	نظم تشغيل مدمجة
Distributed operating systems	نظم تشغيل موزعة
Mount point	نقطه إرفاق
User mode	نمط المستخدم

Kernel mode	نمط النواة
Client-server model	نموذج الخادم والزبون
Core	نواة
µ-kernel	نواة دقيقة
هـ	
Smartphone	هاتف ذكي
Mobile phone	هاتف محمول أو متنقل
و	
Software-oriented interfaces	واجهات البرامج الموجهة
Graphical user interface, GUI	واجهة المستخدم الرسومية
Small computer system interface, SCSI	واجهة نظام الحاسوب الصغير
Loadable kernel modules, LKM	وحدات النواة القابلة للتحميل
Memory management unit, MMU	وحدة إدارة الذاكرة
Central processing unit, CPU	وحدة المعالجة المركزية
Plug and Play	وصل وشغل
Sequential access	الوصول التتابعي/ التسلسلي
File accesses	وصول إلى ملف
Direct/Random access	وصول مباشر (عشوائي)
Direct memory access, DMA	وصول مباشر للذاكرة
Race condition	وضع التسابق
Job	وظيفة
Batch job	وظيفة دفعة
Windows Server 200x	’ويندوز سيرفير‘ 200x
Windows	’ويندوز‘
ي	
Unix V	’يونكس 5‘
Unix	’يونكس‘

المراجع

Anderson, T. E., Bershad, B. N., Lazowska, E. D., and Levy, H. M., Scheduler Activations: Effective Kernel Support for the User-Level Management of Parallelism, ACM Transaction on Computer Systems, vol. 10, pp. 53–79, Feb. 1992.

Bala, K., Kaashoek, M. F., and Weihl, W., Software Prefetching and Caching for Translation Lookaside Buffers, Proceeding First Symposium on Operating Systems Design and Implementation, USENIX, pp. 243–254, 1994.

Brinch-Hansen, P., Operating System Principles, Englewood Cliffs, Prentice-Hall, 1973.

Carr, R. W., and Hennessy, J. L., WSClock-A Simple and Effective Algorithm for Virtual Memory Management, Proc. Eighth Symposium on Operating Systems Principles, ACM, pp. 87–95, 1981.

Coffman, E. G., Elphick, M. J., and Shoshani, A., System Deadlocks, Computing Surveys, vol. 3, pp. 67–78, June 1971.

Denning, P. J., The Working Set Model for Program Behavior, Commun. of the ACM, vol. 11, pp. 323–333, 1968a.

Denning, P. J., Working Sets Past and Present, IEEE Trans. on Software Engineering, vol. SE-6, pp. 64–84, Jan. 1980.

Dijkstra, E.W., Co-operating Sequential Processes, in Programming Languages, Genuys, F. (Ed.), London: Academic Press,

1965.

Fotheringham, J., Dynamic Storage Allocation in the Atlas Including an Automatic Use of a Backing Store, Commun. of the ACM, vol. 4, pp. 435–436, Oct. 1961.

Garrido, J. M., Schlesinger, R., and Hoganson, K., Principles of modem operating systems, 2nd ed., Jones and Bartlett Publishers, 2011.

GeeksforGeeks, (May 2020), Available: https://www.geeksforgeeks.org/

Goldberg, R. P., Survey of Virtual Machine Research, IEEE Computer, Volume 7, Number 6. A terrific survey of a lot of old virtual machine research, Walker and Cragon, 1995.

Holt, R.C., Some Deadlock Properties of Computer Systems, Computing Surveys, vol. 4, pp. 179–196, Sept. 1972.

jvatpoint, (May 2020), Available: https://www.javatpoint.com/

Mana, S. C., Recourse Management Using a Fair Share Scheduler, International Journal of Computer Science and Security (IJCSS), vol. 6, pp. 29-33, 2012.

Newton, G., Deadlock Prevention, Detection, and Resolution: An Annotated Bibliography, ACM SIGOPS Operating Systems Rev., vol. 13, pp. 33–44, April 1979.

Null, L., and Lobur, J., The Essentials of Computer Organization and Architecture, 4th ed., William Brottmiller, 2016.

Peterson, G.L., Myths about the Mutual Exclusion Problem, Information Processing Letters, vol. 12, pp. 115–116, June 1981.

Puhan, J., Operating Systems, Embedded Systems, and Real-Time Systems, 1st ed., FE Publishing, 2015.

Silberschatz, A., Galvin, P. B., and Gagne, G., Operating Systems Concepts, 6th ed. New York: Wiley, 2002.

Silberschatz, A., Galvin, P. B., and Gagne, G., Operating Systems Concepts, 10th ed., Hoboken, NJ: John Wiley & Sons, 2018.

Stallings, W., Computer Organization and Architecture Designing for Performance, 9th ed., 2012.

Stallings, W., Operating Systems Internals and Design Principles, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2001.

Stallings, W., Operating Systems: Internals and Design Principles, 7th ed., Upper Saddle River, NJ: Prentice Hall, 2011.

studytonight, (May 2020), Available: https://www.studytonight.com/

Talluri, M., Hill, M.D., and Khalidi, Y.A., A New Page Table for 64-Bit Address Spaces, Proc. 15th Symp. on Operating Systems Principles, ACM, pp. 184–200, 1995.

Tanenbaum, A. S., and Van Steen, M. R., Distributed Systems, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2007.

Tanenbaum, A. S., Woodhull, A. S., The Minix book: Operating Systems: Design and Implementation, 3rd ed., Prentice Hall,

2006.

Tannenbaum, A. S. and Herbert, B., Modern Operating Systems, 4th ed., Upper Saddle River, NJ: Prentice Hall, 2015.

Tannenbaum, A. S., and Wetherall, D. J., Computer Networks, 5th ed., Upper Saddle River, NJ: Prentice Hall, 2010.

Tannenbaum, A. S., and Woodhull, A. S., Operating Systems: Design and Implementation, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 2006.

Tannenbaum, A. S., Modern Operating Systems, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2001.

Tanner, M., Practical Queuing Analysis, New York: McGraw-Hill, 1995.

tutorialspoint, (May 2020), Available: https://www.tutorialspoint.com/

Uhlig, R., Nagle, D., Stanley, T., Mudge, T., Secrest, S., and Brown, R., Design Tradeoffs for Software-Managed TLBs, ACM Transaction on Computer Systems, vol. 12, pp. 175–205, Aug. 1994.

Zobel, D., The Deadlock Problem: A Classifying Bibliography, ACM SIGOPS Operating Systems Rev., vol. 17, pp. 6–16, Oct. 1983.

ما يُميز هذا الكتاب

حرص المؤلف على إعداد هذا الكتاب باللغة العربية وبأسلوب سلس اعتمد على ترابط مكوناته لتسهيل قراءته وفهمه. كما اجتهد المؤلف في تقليل استخدام المصطلحات الإنجليزية داخل النصوص قدر المستطاع واكتفى باستخدامها فقط في الحالات الضرورية الذي يتطلبه سياق الشرح. وحتى يكون الفهم واضحًا ومكتملًا ضمَّن المؤلف في نهاية هذا الكتاب معجم ترجمة باللغة العربية لأغلب المصطلاحات العلمية الواردة فيه، بوبها أبجديًا لتسهيل عملية البحث عنها داخل هذا المعجم. ومن باب حرص المؤلف في هذا الخصوص اعتنى باختيار الترجمة الدقيقة لأي مصطلح أُستُخدم في الكتاب استنادًا إلى الترجمات الشائعة لهذه المصطلحات في مجال علوم الحاسوب، أو في بعض الأحيان إلى الاجتهادات الفردية له، والتي يأمل من الله أن يكون قد وُفِّق فيها.

من المزايا الأخرى لهذا الكتاب هو تقديم العديد من الأمثلة التوضيحية كلما استلزم الأمر ذلك، واختتام كل فصل منه بأسئلة مرجعية تُساهم في جعل الطالب يفهم مادة الكتاب ويتفاعل معها بشكل إيجابي، كما أنه اعتمد على تضمين الألوان داخل النصوص، والجداول، والأشكال التوضيحية لما لذلك من دلالات على تبسيط المعاني وتقريبها وتذكرها بشكل أفضل.

أخيرًا، أُعدَّ الكتاب كحصيلة لعدة سنوات تدريسية لمادة نظم التشغيل استعان فيها المؤلف بعدة مقالات، وصفحات متخصصة في مجالات نظم التشغيل، وكتب مرجعية كان من أهمها مؤلفات تانن باوم، وزلبرشاتس، وجاريدو والتي ساهمت بشكل كبير في وضع أسس هذا الكتاب.

ختمامًا أدعو الله العلي القدير أن يُثري هذا الكتاب المكتبة العربية ويسد فجوةً فيها، وأن يستفيد منه كل من اطَّلع عليه.

د. عمر المبروك أبوزيد

غريان–ليبيا : أغسطس 2023م