بسم الله الرحمن الرحيم

جامعة السّودان المفتوحة

برنامج الحاسوب

مبادئ نظم التشغيل

رمز المقرر ورقمه: ٢٠٣٦

تأليف: د. السماني عبد المطلب أحمد

أ. الهادي سليمان الهادي

تحكيم علمي : د. سيف الدين فتوح عثمان

تصميم تعليمي: أ. منال محمد بشير التنقاري

التنضيد الطباعي: الهادي سليمان

التدقيق اللغوي: أ.الهَديْ عبد الله محمد

التصميم الفني: منى عثمان احمد النقة

منشورات جامعة السودان المفتوحة، الطبعة الأولى ٢٠٠٦

مقدمة المقرر

الحمد لله الذي أشرقت أنوار حقيقته على مطالع السعادة من آفاق قلوب أوليائـــه المقربين، فعاشوا مع الحق في قران مجيد أفضى بهم إلى فرقان نير فريد، فـــتح لهـــم آفاق السماء بروح الصفاء لتلقى مطالع المعارف لمن علت همته وكملت روحه من أهل الوفاء ، والصلاة والسلام على سيدنا الحبيب محمد الرسول الأكرم، صاحب لواء الحمد الأكرم، وعلى أله وأصحابه وتابعيهم ومن تبعهم بإحسان إلى يوم الدين.

أهلاً بك عزيزي الدارس في **المقرر الأول لدراسة نظم التشغيل** وهـــو خـــاص بالجزء النظري لمفهوم نظم التشغيل والأجزاء الأساسية المكونة لـــنظم التـــشغيل مثـــل المهمات والركود وأسبابه وطرق تفاديه والذاكرة الرئيسة والافتراضية ووحدات الإدخال و الإخراج.

حاولنا بقدر الإمكان اختيار الترجمة المناسبة والمتداولة في معظم الجامعات السودانية والتي تعطي المدلول التام للمصطلحات العلمية، مع المحاولة على إبقاء الكلمة الأنجليزية المقابلة للترجمة جنباً إلى جنب، أو تجدها في مسرد المصطلحات في نهاية كل وحدة، حتى يتسنى لك عزيزي الدارس القدرة للرجوع للمراجع الأنجليزية و المواقع على الانترنت

.

الأهداف العامة لهذا المقرر

عزيزي الدارس، بعد فراغك من دراسة هذا المقرر وحل جميع الواجبات الواردة فيه من تدريبات وأنشطة، ينبغي أن تكون قادراً على:

[١] شرح مفاهيم وأهمية نظم التشغيل وتطورها وكيفية عملها.

[٢] شرح وظائف نظم التشغيل وأنواع نظم التشغيل.

[٣] الإلمام بالأجزاء الأساسية لنظم التشغيل والعلاقات بينهما.

[٤] شرح مفهوم المهمات والركود وأسبابه وطرق حله.

[٥] شرح مفهوم الذاكرة وكيفية إدارتها ومفهوم الذاكرة الافتراضية.

[٦] توضيح الخوارزميات الأساسية لإدارة مصادر الحاسوب المختلفة.

محتويات المقرر

يشمل المقرر الوحدات التالية :


الوحدة الأولى
أساسيات نظم التشغيل

محتويات الوحدة

المقدمة

تمهيد

أهلاً بك عزيزي الدارس في الوحدة الأولى من مقرر "**نظم التشغيل**" الجزء الأول وهي بعنوان "**أساسيات نظم التشغيل**" .

تقدم لك هذه الوحدة عزيزي الدارس فكرة عن كيفية تنظيم الحاسوب .وأين موقع نظام التشغيل من مكونات الحاسوب، فضلاً عن التعريف بأهداف نظام التشغيل،كما تقدم لك هذه الوحدة عزيزي الدارس وظائف نظم التشغيل، والتطور التاريخي لنظم التشغيل عبر تبيان أجيال الحاسوب وسرد أنواع أنظمة التشغيل ! ونختم عزيزي الدارس الوحدة ببعض مبادئ المفاهيم الأساسية لنظام التشغيل والتي يجب على الدارس الإلمام بها حتى يكون مستعدا للدخول في تفاصيلها عبر الوحدات الأخرى من هذا المقرر.

تتألف هذه الوحدة من سبعة أقسام رئيسة لتحقيق الأهداف أعلاه، فالقسم **الأول** جاء تحت عنوان **تنظيم الحاسوب، وهو** يرتبط بالهدف الأول من قائمة الأهداف المشار إليها سابقاً i ونتناول فيه الوحدات الأساسية للحاسوب، و طبقات نظام الحاسوب. **الثاني** جاء تحت عنوان **تعريف نظام التشغيل،** ويعتبر مكملاً للقسم الأول في تحقيق الهدف الأول من قائمة الأهداف المشار إليها سابقاً ونتحدث فيه عن أهمية نظام التشغيل ونبين أن نظام التشغيل يقوم وظيفتين منفصلتين عن بعضهما تماماً وهما :

أولاً : آله ظاهرية Virtual Machine

ثانياً ــ مدير الموارد Resource Manger

الثالث جاء تحت عنوان **وظائف نظم التشغيل**، و يرتبط بالهدف الثاني من قائمة الأهداف المشار إليها سابقاً ونتحدث فيه عن أهم الوظائف التي يجب أن تتوفر في أي نظام تشغيل

الرابع جاء تحت عنوان **تاريخ نظم التشغيل**، **و** يرتبط بالهدف الثالث من قائمة الأهداف المشار إليها سابقاً ونتحدث فيه عن تاريخ أنظمة التشغيل وتطورها منذ ظهورها حتى الآن، من خلال نقسيم مراحل التطور إلى خمسة أجيال، كل جيل يقدم تجديدا وتطويرا على سابقه.

الخامس جاء تحت عنوان **تطور نظام التشغيل** ويعتبر مكملاً للقسم الرابع في تحقيق الهدف الثالث من قائمة الأهداف المشار إليها سابقاً ونتحدث فيه عن تطور مفاهيم نظم التشغيل **OS** منذ البداية عند طرح مفهوم النظم الدُفعية البسيطة إلى طرح المفاهيم المتقدمة مثل نظم الزمن الحقيقي ونظم المشاركة بالزمن.

السادس جاء تحت عنوان **أنواع نظم التشغيل**، يرتبط بالهدف الرابع من قائمة الأهداف المشار إليها سابقاً، نتحدث فيه عن عدة أنواع من الأنظمة المتنوعة بعضها معروف والبعض الأخر يستخدم في مجالات ضيقه .

السابع جاء تحت عنوان **مفاهيم نظم التشغيل الأساسية**، **و**يرتبط بالهدف الخامس من قائمة الأهداف المشار إليها سابقاً ونناقش فيه أهم مفاهيم نظم التشغيل، مثل: النواة والقشرة وإدارة الذاكرة وغيرها من المفاهيم.

أهداف الوحدة

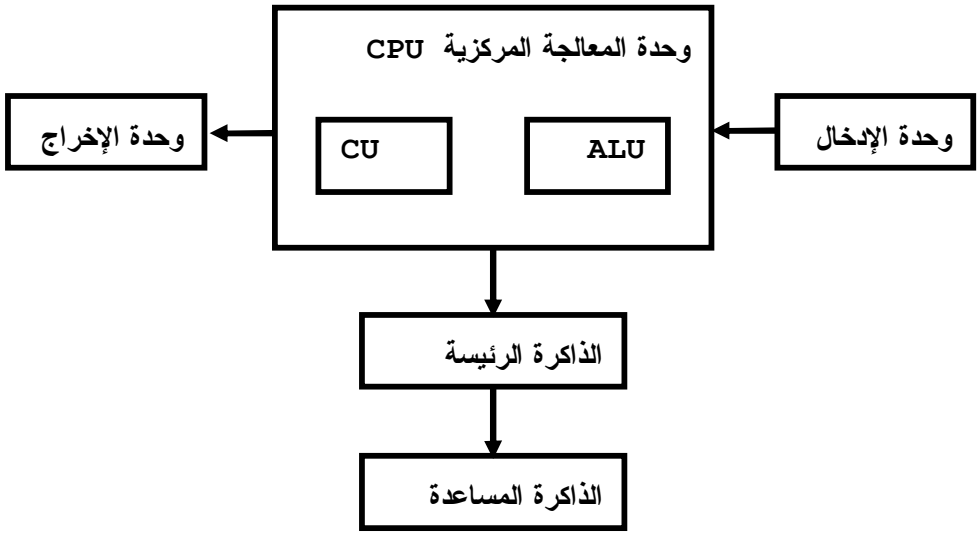
عزيزي الدارس،

بعد فراغك من دراسة هذه الوحدة وحل جميع الواجبات الواردة فيها من تدريبات وأنشطة، ينبغي أن تكون قادراً على أن:

- تعرف تنظيم الحاسوب وتبين موقع نظام التشغيل منه.
- تشرح وظائف نظم التشغيل.
- تسرد التطور التاريخي لنظم التشغيل وأجيال الحاسوب.
- تشرح أنواع نظم التشغيل.
- تعرف مبادئ المفاهيم الأساسية لنظام التشغيل .

١. تنظيم الحاسوب

يوجد العديد من الأشكال التي توضح المكونات الأساسية للحاسوب والشكل رقم (١-١) يبين المكونات الحاسوب الأساسية.



الشكل رقم (١-١)المكونات الأساسية للحاسوب

عموماً يتكون نظام الحاسوب من معالج أو عدة معالجات، بالإضافة إلي الذاكرة الرئيسية، والأقراص، والطابعات، ولوحة المفاتيح، وجهاز العرض، ومحولات الشبكة i والتي تسمى مجتمعه بالمكونات المادية **Hardware** ، وتشكل هذه المكونات مجتمعه نظاماً معقداً في التعامل، مما اتوجب كتابة برامج تتحكم في إدارة جميع هذه المكونات وتستخدمها استخداما صحيحاً، وتسمى هذه البرامج ببرامج النظام والتي من أهم وظائفها إدارة جميع هذه الأجهزة (المكونات المادية)، بالإضافة إلي تقديم واجهة بسيطة للمستخدم لكي يتمكن من التعامل مع المكونات المادية .

ينقسم نظام الحاسوب كما الشكل رقم (١-٢) إلي ثلاث طبقات على النحو التالي:

١,١.الطبقة الأولى

وهي طبقة المكونات المادية والتى تنقسم إلى ثلاثة مستويات

- **المستوى الأول** الذي يتألف من الأجهزة الفيزيائية والتي تشمل شرائح دارات متكاملة وأسلاكاً ومزودات طاقة وأجهزة فيزيائية أخرى .
- **المستوىالثاني** والذى يشتمل على البنية المكروية والتي تجمع الأجهزة الفيزيائية مع بعض المسجلات الداخلية في المعالج، وذلك لتنفيذ مجموعة من التعليمات .
- **المستوى الثالث** والذي يحتوي على لغة الآلة والتى تتكون من ٥٠ إلى ٣٠٠ تعليمة معظمها من أجل نقل البيانات بين أجزاء الحاسوب وإنجاز العمليات الحسابية ومقارنة القيم، بالإضافة إلى التحكم في أجهزة الإدخال والإخراج من خلال تحميل بعض القيم في المسجلات الخاصة بهذه الأجهزة.

وعلى سبيل المثال إذا أردنا تنفيذ أمر قراءة من القرص فيجب تحميل قيم عنوان القرص وعنوان الذاكرة الرئيسية وعدد البيانات المراد نقلها ونوع العملية (قراءة أم كتابة) في المسجلات الخاصة بذلك .

١,٢.الطبقة الثانية

من طبقات نظام الحاسوب هي طبقة برامج النظام والتي تحتوي على نظام التشغيل، والذي يهدف إلى إخفاء جميع التعقيدات، حيث يتكون نظام التشغيل من طبقة برمجيات تخفي التعقيدات التي تظهر عند التعامل مع المكونات الفيزيائية وذلك من خلال مجموعة من التعليمات المناسبة ، التي تجعل التعامل مع تلك المكونات من المهام السهلة ، كما توضع بقية برمجيات النظام فوق نظام التشغيل والتي تحتوي على مفسرات الأوامر ومحررات النصوص والمترجمات وغيرها من البرامج غير التطبيقية وهي ليست جزءاً من نظام التشغيل i ويظهر ذلك جلياً عندما نقول إن نظام التشغيل

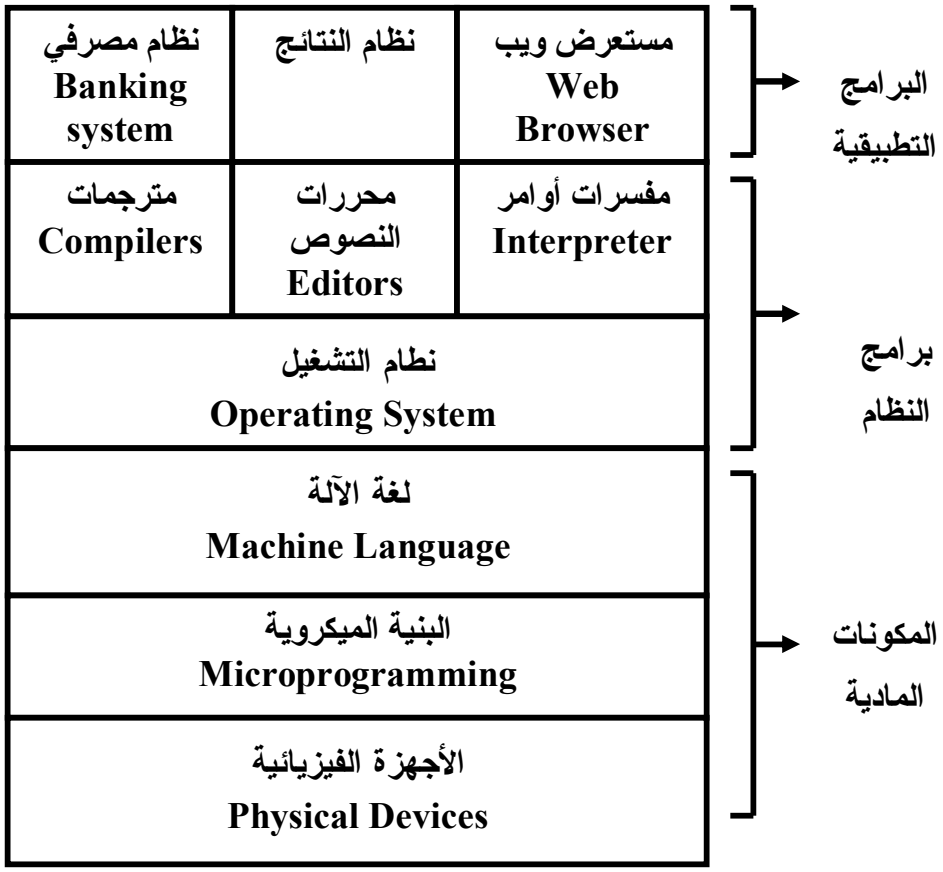
يعتبر جزء من البرمجيات التي تعمل في نمط النواة **(Kernel Mode)** K ولا يمكن للمستخدم تعديله. بينما تعمل محررات النصوص والمترجمات في نمط المستخدم **(User Mode)** وبالتالي يمكن للمستخدم تعديلها ، فمثلاً:

إذا لم يرغب المستخدم في التعامل مع مترجم معين فيمكنه كتابه مترجم خاص به واستخدامه بدلاً من المترجم السابق ، لكنه لا يستطيع تغيير معالج مقاطعه الساعة لأنه جزء من نظام التشغيل ويكون محمياً من قبل المكونات المادية من محاولات التعديل من قبل المستخدم .

١,٣. الطبقة الثالثة

من طبقات نظام الحاسوب والتي تمثل طبقة البرامج التطبيقية وهي برامج تعمل في نمط المستخدم **(User Mode)** لكنها تساعد نظام التشغيل على القيام بمهمة فمثلاً:

برنامج يسمح للمستخدم بتغيير كلمة المرور وهذا البرنامج ليس جزءاً من نظام التشغيل لأنه لا يعمل في نمط النواة (Kernel Mode) لكنه يقوم بمهمة حساسة ويجب حمايته بطريقة خاصة.



الشكل رقم (١-٢) طبقات نظام الحاسوب

نشاط

ناقش مع زملائك تعدد الأشكال التي توضح المكونات الأساسية للحاسوب

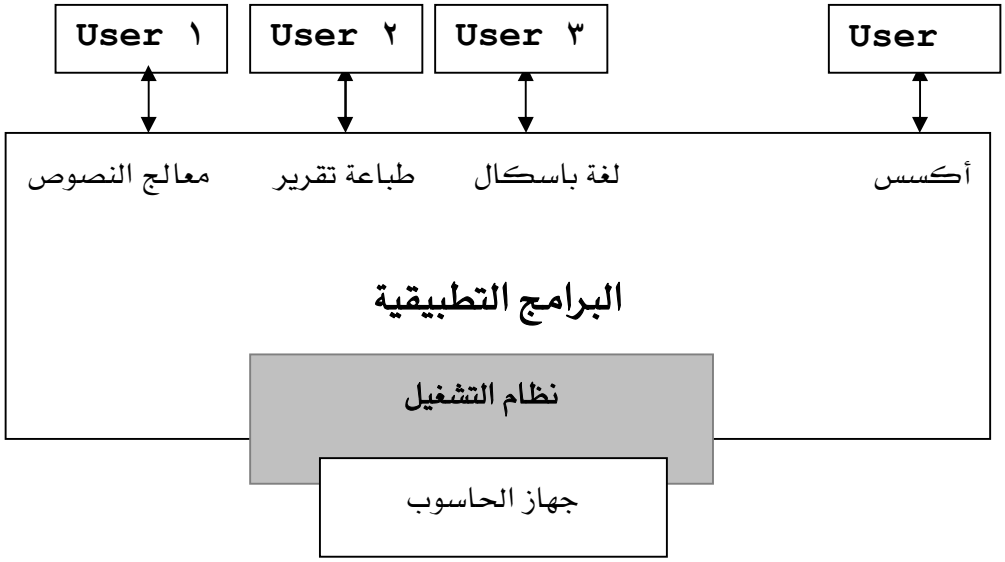
أسئلة تقويم ذاتي

١- اكر أهم وظائف نظام التشغيل .

٢- لماذا لا تعتبر المترجمات جزء من نظام التشغيل؟

٢. تعريف نظام التشغيل

نظام التشغيل هو برنامج يعمل بين مستخدم الحاسوب وجهاز الحاسوب وهو يمكن المستخدم من تنفيذ برمجياته بسهولة وبكفاءة عالية i ونلاحظ من الشكل (١-٣)



الشكل رقم (١-٣) نظام التشغيل ينظم طلبات المستخدمين

نجد إن نظام التشغيل ينظم أو يتحكم في طلبات المستخدمين.و يمكن أن نقول إن نظام التشغيل هو برنامج تحكم يساعد المستخدمين في تنفيذ أعمالهم باستخدام الحد الأدنى من إمكانية الحاسوب (الوقت والذاكرة).

يمكن تعريف نظام التشغيل أيضاً بأنه البرنامج الرئيسي لأي حاسوب i حديث و تتركز المهام الرئيسية لنظام التشغيل في كيفية إدارة :

- المكونات المادية و تهيئتها للعمل (تسمى العتاد **(Hardware).**
- البرمجيات **(Software).**

وبذلك تعطي أفضل كفاءة ممكنة، و تقلل من احتمالية حدوث أعطال و أخطاء. ومن أخطر وظائف نظام التشغيل هنا البلاغ عن الأخطاء غير القادر على تجاوزها، و إعلام المستخدم بها i أما الأخطاء القادر على تجاوزها فيتم تسجيلها للعودة إليها لاحقا و دراستها من قبل المستخدم.

يوجد نظام التشغيل حاليا في كثير من :

الإجهزة الالكترونية مثل **الهاتف النقال** (الموبايل).

- **أجهزة اللعب** الشهيرة مثل **PlayStation** .
- في بعض أنواع **الآلات الحاسبة** .

وتختلف التصميمات و طريقة العمل في كل من هذه الأجهزة إلا أنه لا يزال مبدأ إدارة العتاد و البرامج مطبقاً فيهما جميعا. وكما هو معروف لا يمكن لأي جهاز حاسوب أن يعمل إلا عند توفر نظام التشغيل (الذي يحّمل من الذاكرة الثانوية إلى الذاكرة الرئيسية) عند تشغيل الجهاز i ليبدأ بإدارة العمل في الجهاز ويتكون نظام التشغيل من مجموعة من البرامج المتكاملة تعمل كفريق كل منها يؤدي مهمات معينة. يسمى برنامج التحكم الرئيسي في نظام التشغيل بالمشرف**Supervisors**، أو بالمراقب **Monitors** ،أو المنفذ **Executive** وهو المسئول عن توجيه النشاطات لجميع أجزاء نظام التشغيل i وعند تشغيل الحاسوب لأول مرة فإن المشرف هو أول برنامج يحمل إلى الذاكرة الرئيسية.

إحدى المهمات الأساسية لنظام التشغيل، هي التحكم بالوصول إلى الأقراص ووسائط التخزين، وكان اسم أنظمة تشغيل الأجهزة الشخصية في أيامها الأولى، **DOS**، وهي اختصار لـــ**Disk Operating System** . وقد ظهر ذلك جلياً من خلال تسمية أنظمة التشغيل لعدد من الشركات العاملة في مجال الحاسوب i والجدول التالي يبين أسماء أنظمة التشغيل والشركات التي قامت بتصميمها.

MS-DOS	من شركة مايكروسوفت
PC-DOS	من شركة**IBM**
DR-DOS	من شركة **Digital Research**
AppleDOS	من شركة أبل النظام السابق لأبل ماكنتوش
AmigaDOS	من شركة كومودور، لأجهزة **Commodore Amiga**

بالرغم من أن معظم مستخدمي الحاسوب لهم بعض الخبرة في واحد أو أكثر من أنظمة التشغيل إلا أنه يصعب عليهم تحديد ما هو نظام التشغيل لما يقوم به من وظيفتين منفصلتين عن بعضهما تماماً وهما :

أولاً : آله ظاهرية Virtual Machine

بنية معظم الحاسوب يصعب برمجتها على مستوي لغة الآلة i وخاصة برمجة أجهزة الإدخال والإخراج i ويمكن توضيح ذلك بشرح كيفية إنجاز عمليات الإدخال والإخراج من القرص المرن. تحتاج كل من الأوامر الأساسية **Read & Write** إلي عدد من المعاملات(البارامترات) التي تحدد بعض الأشياء منها :

- ✓ عنوان كتلة القرص المرن المراد قراءتها.
- ✓ عدد القطاعات في المسار.
- ✓ نمط التسجيل المستخدم في الوسط الفيزيائي.
- ✓ مسافة الفجوة بين القطاعات.
- ✓ وغيرها .

تعتبر هذه الأشياء تعقيدات لا يحتاج إليها المبرمج i بل يحتاج إلي تجريد بسيط للتعامل مع تلك الأقراص i فمثلاً يحتوي القرص على مجموعة من الملفات فيحتاج المبرمج إلى طريقة لكي يفتح كل ملف ، ثم مكانية القراءة منه أو الكتابة فيه و إغلاقه بعد الانتهاء من عملية الكتابة أو القراءة ، أما التفصيل، فمثلاً ما هي حالة محرك القرص فيجب ألا يراها المستخدم .

يمكن القول بأن البرامج التي تخفي حقيقة المكونات المادية عن المبرمج أو المستخدم وتقدم له تجريداً بسيطاً وجميلاً في التعامل مع الملفات وإمكانية الكتابة فيها أو القراءة منها هي بالتأكيد نظام التشغيل. ومن هذه الزاوية يمكن القول بأن وظيفة نظام التشغيل هي آلة ظاهرية **Virtual Machine** تسهل استخدام المكونات المادية التي تحتها i أما كيفية تحقيق هذه الغاية فهي عملية طويلة. ونستخلص من ذلك بأن نظام التشغيل يقدم خدمات متنوعة تستطيع البرامج الحصول عليها باستخدام تعليمات خاصة تدعى استدعاءات النظام **System Calls**

ثانياً ــ مدير الموارد Resource Manger

عندما ننظر إلى نظام التشغيل من أعلى إلى أسفل يتضح لنا أنه يقدم واجهة مناسبة وسهلة للمستخدمين كما ذكرنا سابقاً. أما إذا نظرنا إليه من أسفل إلى أعلى فنجده يقوم بإدارة جميع أجزاء الحاسوب، ويمكن توضيح ذلك إذا تخيلنا أن هنالك ثلاث عمليات تعمل كلها على جهاز واحد i وكل عمليه تريد طباعة ملف معين في نفس الوقت وعلى الطابعة نفسها، ستكون السطور الأولى من مخرجات الطابعة من العملية الأولي بينما السطور التالية من العملية الثانية وبعض السطور من العملية الثالثة مما يحدث فوضي وتداخل بين المستخدمين وتصادم. يمكن لنظام التشغيل تنظيم هذه الفوضى وذلك من خلال تخزين بيانات العمليات الثلاثة المتجهة إلى الطابعة في مخزن وسيط **Buffer** موجود في القرص، فعندما تنتهي العملية الأولى من الطباعة يستدعي العملية الثانية وهكذا .

تزداد الحاجة لإدارة وحماية موارد الحاسوب عندما يكون للحاسوب عدة مستخدمين، لأنه يمكن أن تتداخل أعمال المستخدمين مع بعضها البعض، بالإضافة الي ذلك فقد يحتاج المستخدمون للتشارك ليس في المكونات المادية فقط بل بالمعلومات أيضاً (الملفات ، قواعد البيانات ، .... إلخ).

من هذا المنظور يمكن تحديد مهمة نظام التشغيل على أنه ينظم استخدام المستخدمين للموارد ، وذلك بمنح تلك الموارد لمن يطلبها ومراقبة استخدامها، ومنع التضارب في طلبات تلك الموارد من المستخدمين المختلفين. يتم توزيع مشاركة الموارد بطريقتين :

١,٢. التوزيع الزماني Time Multiplexing

يمكن لكل برنامج استخدام المعالج لفترة زمنية معينة، فمثلاً يقوم البرنامج الأول باستخدام المعالج في البداية ثم يأتي دور البرنامج الثاني وهكذا إلى أن يتم تنفيذ كل البرامج. في الأنظمة التي تحتوي على معالج واحد ويوجد عدة برامج يراد تنفيذها في هذه الحالة يقوم نظام التشغيل بتخصيص المعالج لأحد البرامج ثم ينتظر فترة زمنية كافية لكي يعمل قليلاً ، وبعد ذلك ينتقل التنفيذ لبرنامج آخر ثم بعد قليل للبرنامج الآخر وهكذا ينتقل الدور تدريجياً حتى يصل إل البرنامج الأول مره أخري . من الأمثلة الأخرى عل التوزيع الزمني التشارك بالطابعة .

٢,٢. التوزيع المكاني Space Multiplexing

فمثلاً تقسم الذاكرة الرئيسية عادة بين عدة برامج تعمل على الجهاز بحيث يستطيع كل منها أن تقيم في الذاكرة في نفس الوقت منتظرة دورها لاستخدام المعالج ، إذا افترضنا أن هنالك ذاكرة كافية لاحتواء عدة برامج ، فإن أنسب طريقة هي وضع عدة برامج في نفس الوقت بدلاً من جعل كل منها تأخذ الذاكرة كلها ، وخصوصاً إذا كانت تحتاج فقط إلي جزء من تلك الذاكرة .

من الموارد الأخرى الموزعة مكانياً القرص الصلب i وذلك لأن القرص الصلب في العديد من الأنظمة تخزن فيه ملفات تابعة لعدة مستخدمين في نفس الوقت، وتعد عملية تخصيص مساحة القرص ومعرفة من يستخدم الكتل المختلفة في القرص من مهام نظام التشغيل الأساسية .

وفقاً لما سبق يمكن أن نعرف نظام التشغيل على أنه عبارة عن برنامج :

- ✓ **يعمل كوسيط ما بين مستخدم الحاسوب ومكوناته المادية (العتاد) أي إن الغرض منه هو تزويدنا ببيئة عمل يستطيع المستخدم من خلالها تنفيذ برامجه.**
- ✓ **يتحكم في تنفيذ برامج المستخدم، لكي يمنع حدوث الأخطاء والاستخدام غير المناسب للحاسوب i وبشكل خاص التحكم بأجهزة الإدخال والإخراج I/O.**
- ✓ **ينفذ دائماً في الحاسوب ويسمى باللب (Kernel) وما عدا ذلك بالبرامج التطبيقية.**

وعلى الرغم من كل ما تقدم، لا يوجد تعريف كاف لنظام التشغيل ولا يوجد اتفاق عالمي على ما هو الجزء المنتمي أو غير المنتمي إلى نظام التشغيل OS

يمكن تقسيم أنظمة التشغيل من خلال

- ✓ **عدد المهمات التي يمكن أن تنجز آنياً، أي مهمة واحدة أو عدد من المهمات**
- ✓ **عدد المستخدمين الذين يمكنهم استخدام النظم آنياً i أي مستخدم وحيد أو عدد من المستخدمين**

الجدول رقم (١-١) يبين بعض الأمثلة لنظم التشغيل:

جدول رقم (١-١)

عدد المهام والمستخدمين في بعض أنظمة التشغيل

نظام التشغيل	المستخدمين	المهام	عدد المعالج
MS/PC DOS	**Single**	**Single**	**1**
Windows 3x	**Single**	**quasi-Multi**	**1**
Macintosh 7.*	**Single**	**quasi-Multi**	**1**
Amiga DOS	**Single**	**Multi**	**1**
Windows 9x	**Single**	**Multi ***	**1**
UNIX	**Multi**	**Multi**	***N***
Windows NT	**S/Multi**	**Multi**	***N***
Windows 2000	**Multi**	**Multi**	***N***

نشاط

ناقش مع زملائك، عبارة "يوجد نظام التشغيل حاليا في كثير من الهاتف النقال الأجهزة الالكترونية مثل (الموبايل) ".

أسئلة تقويم ذاتي

١- عرف نظام التشغيل

٢- كيف يمكن تقسيم نظام التشغيل؟

٣. وظائف نظم التشغيل

يقوم أي نظام تشغيل بالوظائف التالية:

١-تمكين الاتصال بين الحاسوب والمستخدم من خلال واجهة المستخدم التي تكون على شكل أوامر يعطيها المستخدم للجهاز، **Command based** أو على شكل واجهة رسومية وقوائم يختار منها المستخدم الأمر المطلوب **Graphical User Interface** كما هو الحال في برمجية. **Windows**

٢-وتوزيع المعدات المشتركة على المستخدمين في الشبكة وجدولة iاستخدامها فإذا كانت هناك طابعة واحدة مع عدة أجهزة مرتبطة مع بعضها البعض عن طريق الشبكة، وأراد عدد من المستخدمين طباعة وثائقهم باستخدام الطابعة في الوقت نفسه فإن نظام التشغيل يجدول عملية الطباعة حسب سياسة معينة بحيث يحصل كل مستخدم في النهاية علىوثيقةمطبوعة.

٣-يسهل الاتصال بين مكونات الحاسوب حيث يسهل حركة التعليمات الداخلية والبيانات بين الأجهزة الطرفية والمعالج والبرامج وأجهزة التخزين، أي أنه يسهل عمليات الإدخال والإخراج والتخزين الثانوية.

٤- الحماية من الأخطاء ومراقبة النظام وإخطار المستخدم في حال الفشل حيث يفحص نظام التشغيل معدات نظام الحاسوب بشكل مستمر ويتم إخطار المستخدم فورا في حالة حدوث أي مشكلةi فمثلا عند إعطاء أمر الطباعة لوثيقة ما والطابعة خالية من الورق تظهر رسالة تخطر المستخدم بعدم إمكانية الطباعة لخلو الطابعة من الورق.

٥- جدولة استخدام المصادر واستغلالها بشكل أمثل : حيث□ن نظام التشغيل يحدد المهام المطلوبة والمصادر المتوفرة من معالج وذاكرة وأجهزة في كل لحظة زمنية، ويوزع عليها المهام المطلوبة بطريقة تزيد من سرعة إنجاز العمل.

٦-يتعقب الملفات على الأقراص : فيسهل عمل النسخ الاحتياطية ومسح الملفات وتشكيل الأقراص وتهيئتها للتخزين عليها؛ كما يقوم بفتح الملفات وإغلاقها وتحميلها إلى الذاكرة الرئيسية؛ كما يتعقب نظام التشغيل جدول مواقع الملفات ويحدثه باستمرار.

٧-حماية النظام : يسمح نظام التشغيل أو يمنع وصول مستخدم معين إلى نظام الحاسوب أو أي ملف مخزن فيه حسب الصلاحيات المعطاة لهذا المستخدم.

أسئلة تقويم ذاتي

حدد وظائف نظام التشغيل الأساسية.

٤. تاريخ نظم التشغيل

شهدت أنظمة التشغيل تطوراً كبيراً فمرت بعدة مراحل منذ ظهورها حتى الآن، وذلك نتيجة التطوير والتجديد في الابتكارات العلمية التي تخدم في هذا المجال، ويمكن أن نقسم تلك المراحل إلى خمسة أجيال، كل جيل يقدم تجديدا وتطويرا على سابقه:

الجيل الأول (١٩٤٥ – ١٩٥٥م):

ومن أبرز ملامح هذا الجيل:

- ✓ يسمى بجيل الصمامات، واستخدمت فيه الصمامات الإلكترونية المفرغة وأنابيب أشعة المهبط بطاقة تخزينية تصل إلى ٢٠٠٠ كلمة.
- ✓ بداية ظهور الحاسوب بشكل تجاري في ١٤ يونيو ١٩٥١م بسعر يصل حوالي **1M$**، حيث اشترت مصلحة الإحصاءات الأمريكية أول جهاز من نوع **(Univac)** لاستخدامه في جدولة الإحصاءات السكانية.
- ✓ لا يوجد نظام تشغيل **(Operating System)** ويعمل الحاسوب من خلال تحكم يدوي.

- ✓ البرمجة تتم باستخدام لغة الآلة **(Language Machine)** حيث تكتب التعليمات للحاسوب على شكل سلسلة من الأرقام.
- ✓ كبيرة الحجم تصل المساحة التي تشغلها حوالي ٢٠٠ م^{٢} وتحتاج إلى تسخين قبل عملها، ما ينتج حرارة عند استخدامها، ويستلزم ذلك تغيير الصمامات بمعدل صمام في يوم.
- ✓ كان التركيز منصبًا على القدرة الحسابية.

<u>الجيل الثاني (١٩٥٦ – ١٩٦٤م):</u>

ومن أبرز ملامح هذا الجيل:

- ✓ يسمى بجيل الترانزستور حيث استخدم بدلاً من الصمامات المفرغة، ويتميز الترانزستور بصغر حجمه، وعدم حاجته إلى التسخين وعدم استهلاكه الطاقة بالسرعة العالية، وقد فتح استخدام الترانزستور آفاقًا جديدة في حقل الإلكترونيات عمومًا وفي مجال الحاسوب خصوصًا. أصبحت أجهزة الحاسوب أصغر حجمًا وأكثر كفاءة. .
- ✓ أصبحت البرمجة أقل تعقيدًا بعد ظهور لغة التجميع (**Assembly Language)** التي تستخدم مختصرات للحروف بدلاً من الأرقام مثل **(Sub)** وتعني **(Subtract)** اطرح وهكذا. كما تم استخدام لغات برمجة أخرى مثل Fortran, Cobol
- ✓ استخدام البطاقات المثقبة كوحدة إدخال للحاسوب والشريط الممغنط بصفتها وسيلة تخزين ذات قدرة تخزينية عالية ويمكن الوصول للبيانات المخزنة عليها بسرعة والطابعة
- ✓ تم استخدام نظام تشغيل بسيط (نظام دُفعي).
- ✓ استخدمت بطريقة أولية حزم البرمجيات الجاهزة وأنظمة التحكم في الإدخال والإخراج ومترجم البرامج **(Compiler)**.

- ✓ اقتصر استخدام الحاسوب على الجامعات والمنظمات الحكومية والأعمال التجارية، ولم يكن شائع الاستخدام.

الجيل الثالث (١٩٦٥ – ١٩٧١م):

من أبرز ملامح هذا الجيل:

- ✓ يسمى بجيل الدارات المتكاملة وذلك لظهور الدوائر الكهربائية المتكاملة **(Integrated Circuit)**، وهي عبارة عن دوائر إلكترونية متكاملة على شريحة صغيرة من السيلكون لا يجاوز حجمها ١سم مربع، وتحتوي على ملايين من المعدات الإلكترونية.
- ✓ ظهرت أجهزة حاسوب متوسطة أكثر سرعة وذات قدرة تخزينية أكبر .
- ✓ ظهور أنظمة تشغيل محسنة تدعم فكرة المشاركة في الوقت **(Sharing Time)** وهي عملية تنظيم مهام الحاسوب المختلفة من عمليات إدخال، ومعالجة الوصول إلى الاستخدام الأمثل لوحدة المعالجة المركزية، ما يساعد على سرعة استجابة الحاسوب، ويشعر كل مستخدم بأنه الوحيد الذي يتعامل والحاسوب مع وجود عدد كبير من المستخدمين، أي أنظمة تشغيل متعددة المستخدمين ومتعددة البرمجة.
- ✓ ظهرت شبكات الحاسوب **(Network Computer)**، وأصبح بالإمكان الاتصال بالحاسوب الرئيسي من طريق نهاية طرفية من مكان بعيد.

الجيل الرابع (١٩٧٢ – ١٩٨٠م):

ومن أبرز ملامح هذا الجيل:

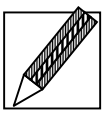
- ✓ ظهور الدوائر الكهربائية المتكاملة الكبيرة **Large Scale Integration** وهي عبارة عن دوائر تحتوي ملايين الترانزستورات موضوعة على شريحة من السيلكون.
- ✓ تطور وسائل اختزان البيانات كأقراص الليزر، والأقراص الممغنطة والأشرطة الممغنطة التي تصل سعة بعضها إلى **(Giga Byte)** أو $^{9}١٠$ بايت.
- ✓ ظهور أجهزة حاسوب متنوعة.**Mainframe, Minicomputer, PC** :

- ✓ ظهور أنظمة تشغيل متنوعة ومتطورة شبكية و موزعة مع تطور وسائل اتـصالات البيانات.
- ✓ ظهور أول معالج دقيق (**Processor Micro**) بجهود العالِم تيدهوف مثل **Intel 4004, 8080,8086, 80186,80286:** وأصبح بالإمكان استخدام هذا المعالج فـي صناعة الأجهزة كالساعات الرقمية، والـسيارات، وحاسـوب الجيـب، والأجهـزة المنزلية، والحاسوب الشخصي.
- ✓ ظهور لغات البرمجة للجيل الرابع، وقواعد البيانات والشرائح الممتدة.

الجيل الخامس (١٩٨٠ حتى وقتنا الحاضر):

ظهر هذا المصطلح من طريق اليابانيين، للتعبير عن أهدافهم الاسـتراتيجية فـي اختراع أجهزة حاسوب ذكية ذات قدرات عالية، وذلك بمواصلة الأبحاث العلميـة فـي مجالات الذكاء الاصطناعي والأنظمة الخبيرة واللغـات الطبيعيـة فـي التحـدث إلـى الحاسوب، واستثمر اليابانيون والأمريكيون على حد سواء بلايين الدولارات للأبحاث في هذه المجالات، ولا شك في أن لذلك ما يبرره، إذ إن الـسيطرة الاقتـصادية وغيرهـا ستكون بيد من يملك المعلومات أولاً.

تدريب(١)



١-حدد الجيل الذي أصبح بإمكانه الاتصال بالحاسـوب الرئيـسي مـن طريق نهاية طرفية من مكان بعيد.

٢- حدد أبرز ملامح الجيل الرابع.

٥. تطور نظم التشغيل

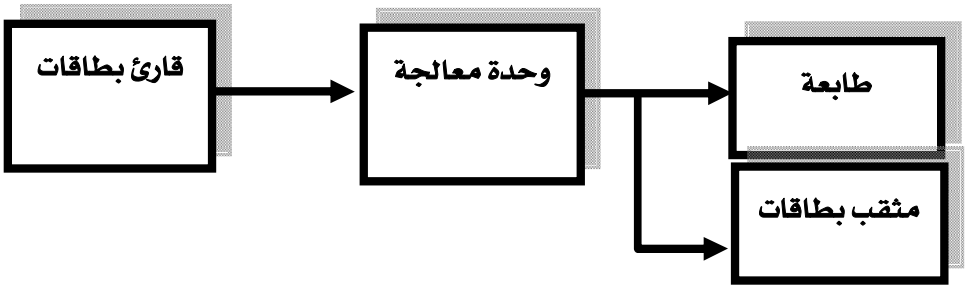
لفهم ماهية نظام التشغيل **OS**. يتوجب علينا أولاً فهم كيفية تطوره مع مرور الزمن منذ البداية إلى طرح المفاهيم مثل : البرمجة المتعددة والمشاركة بالزمن **(Multiprogramming & Time Sharing)** ..إخ

- النظم الدُفعية البسيطة Simple Batched Systems·
- النظم الدُفعية متعددة البرمجة.Multiprogramed Batched Systems ·
- نُظم تقاسم الزمن Time-Sharing System .
- نُظم الحواسيب الشخصية Personal-Computer Systems·
- النُظم التفرعيةparallel Systems .
- النُظم الموزعة Distributed Systems·
- نُظم الزمن الحقيقي Real -Time Systems.

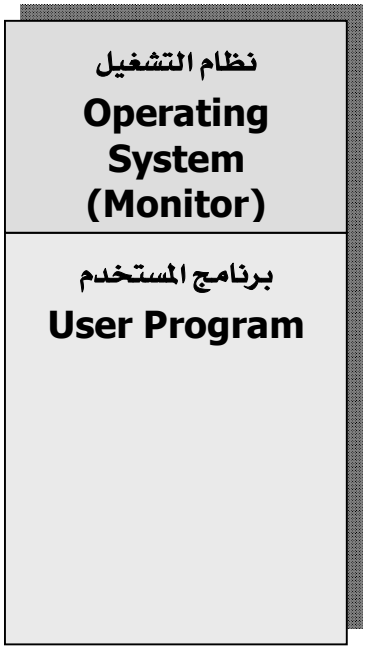
٥,١. النظم الدُفعية البسيطة Simple Batched Systems

تتميز أنظمة التشغيل في هذا الطور بالخصائص التالية:

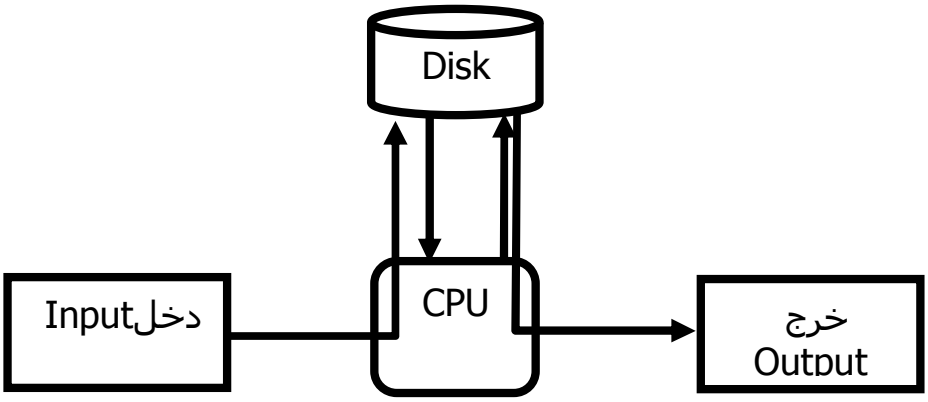
- ✓ تقوم بتنفيذ سلسلة من المهمات الواحدة تلو الأخرى.
- ✓ يوجد مشغِّل **operator** له خبرة وسرعة في تحميل البرامج وتشغيل النظام.
- ✓ يوجد مراقب في الذاكرة لتحميل المهمة ونقل التحكم إلى المهمة ثم عودة التحكم إلى المراقب مرة أخرى.



✓ محتويات الذاكرة



✓ تداخل عمليات الدخل/الخرج مع عملية الحساب لمهمات أخرى

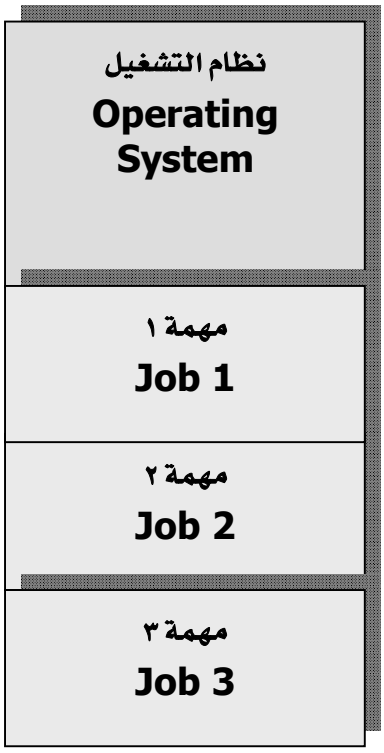


٢,٥. النظم الدُفعية متعددة البرمجة

Multiprogramming Batched Systems

تتميز أنظمة التشغيل في هذا الطور بالخصائص التالية:

- يقوم النظام بحفظ عدد من البرامج في الذاكرة.
- يختار النظام أحد المهمات ويقوم بتنفيذها إلى أن تنتهي أو تطلب عملية دخل/خرج.
- يحجز النظام جهاز دخل/الخرج لأحد المهمات.



٣,٥. نُظم تقاسم الزمن Time-Sharing System

تتميز أنظمة التشغيل في هذا الطور بالخصائص التالية:

- يتناوب على وحدة المعالجة عدة مهمات موجودة في الذاكرة أو على القرص.
- تُحصص وحدة المعالجة لمهمة ما إذا كانت موجودة في الذاكرة فقط، وتعطى حصة زمنية.
- يمكن أن تُنقَل المهمة بين القرص والذاكرة
- يسمح ذلك بالحصول على نظام تفاعلي، ويعطي إمكانية التقلية Debugging
- يسمح بإمكانية تعدد المستخدمين.

٤,٥. أنظمة الحاسوب الشخصي

Personal Computer Systems

لم تكن هذه الأنظمة متعددة المستخدمين والمهام، وهدفها الأول هو ملاءمة استخدام الحاسوب ببساطة ويسر، ولها الخصائص التالية:

- الحاسوب مخصص لمستخدم وحيد.
- تتلاءم مع حاجات المستخدم من سهولة الاستخدام وسرعة الاستجابة.
- لا تحتاج إلى استخدام متقدم لوحدة المعالجة.
- لا تحتاج إلى حماية قوية.

٥,٥. الأنظمة المتوازية Parallel Systems

تتميز أنظمة التشغيل في هذا الطور بالخصائص التالية:

- عدة معالجات تتشارك الذاكرة والساعة.
- مزايا
  - زيادة قوة المعالجة .
  - اقتصادية.
  - زيادة الوثوقية (تعطل معالج لا يؤدي إلى توقف النظام).

- تعدد معالجات متناظرة
  - كل معالج له نسخة مطابقة من نظام التشغيل.
  - كل معالج ينفذ إجراءً مستقلاً.
  - مشكلة: عدم توافق في تقاسم الأعباء (معالج مشغول والأخرى غير مشغولة.)
- تعدد معالجات غير متناظرة
  - هناك معالج سيدMaster ، ومعالجات تابعة.Slave .
  - يوزع المعالج السيد المهام على المعالجات الأخرى.

٦,٥. الأنظمة الموزعة Distributed Systems

تتميز أنظمة التشغيل في هذا الطور بالخصائص التالية:

- توزيع العمل على عدة معالجات لكل منها ذاكرته الخاصة .
- تتخاطب المعالجات فيما بينها عن طريق ممر خاص أو **شبكة.**
- فوائد
  - تشارك الموارد.
  - زيادة سرعة المعالجة وتقاسم الحمل (Load Sharing) .
  - مو ثوقية.
  - توفير وسائل اتصال بين البرامج.
- نظام التشغيل الشبكي:
  - يقدم خدمة مشاركة الملفات والطباعة.
  - يقدم خدمات اتصالات.
  - يعمل كل نظام على نحو مستقل عن غيره.
- نظام التشغيل الموزع:
  - نظام تشغيل شبكي.
  - استقلالية أقل .
  - يعطي الانطباع بأنه يوجد نظام تشغيل واحد يتحكم بالشبكة.

٧,٥. أنظمة الزمن الحقيقي Real-Time Systems

تتميز أنظمة التشغيل في هذا الطور بالخصائص التالية:

- تستخدم كأنظمة تحكم في تطبيقات خاصة (آلات صناعية، تجارب علمية،).
- شروط زمنية معرفة جيداً.
- نظم زمن حقيقي قاسية.
  - المهام الحرجة يجب أن تُنجز ضمن مدة محددة.
  - تخزن المعطيات في ذاكرة رئيسية (RAM, ROM)·
  - تتعارض مع نظم تقاسم الزمن.
- نظم زمن حقيقي لينة.
  - المهام الحرجة لها الأولوية على غيرها.
  - استخدام محدود في التحكم الصناعي والأتمتة (otomotic).
  - مفيدة في حالة تطبيقات تعدد الوسائط والحقيقة الافتراضية.

أسئلة تقويم ذاتي

بين الملامح الأساسية للنظم الدُفعية متعددة البرمجة.

٦. أنواع نظم التشغيل

بعدما تحدثنا عن تاريخ التطورات التي حدثت لأنظمة التشغيل يكون لدينا عدة أنواع من الأنظمة المتنوعة بعضها معروف والبعض الآخر يستخدم في مجالات ضيقه وسوف نتحدث في هذا الجزء عن بعض هذه الأنواع .

١,٦. أنظمة التشغيل للأجهزة الكبيرة Main Frame

تحتل أنظمة تشغيل الأجهزة الكبيرة المرتبة الأولى من حيث الحجم ، كما إنها مازالت تستخدم في مراكز بيانات الشركات الكبرى لأنها تمتاز عن الحواسيب الشخصية بسعة منافذ الإدخال i الإخراج فمثلاً يمكن أن يحتوي واحد من هذه الأجهزة الكبيرة على ١٠٠٠ قرص صلب تسع إلى آلاف الغيغا بايت من البيانات ، بينما الحواسيب الشخصية فأنها تحتوي عادةً علي سعة تخزينية أقل بكثير، كما إنها أصبحت تستخدم كملقمات ويب متطورة وملقمات لمواقع التجارة الإلكترونية واسعة النطاق وغيرها من الأعمال التجارية .

عموماً تهتم أنظمة تشغيل الأنظمة الكبيرة بشكل رئيسي بمعالجة عدة مهام في وقت واحد وتحتوي هذه المهام في غالب الأمر على كمية هائلة من عمليات الإدخال والإخراج، بالإضافة إلى ذلك فإن تلك الأنظمة تقدم نموذجاً لثلاثة أنواع من الخدمات الدفعية **Batch** ومعالجة الناقلات **Transaction processing** ومشاركة الزمن **Time sharing**.

٢,٦. أنظمة تشغيل المخدمات

يأتي هذا النوع من أنظمة التشغيل في المرتبة الثانية والتي تعمل على المخدمات وهي عادة تتكون من أجهزة شخصية كبيرة جدا، أو محطات عمل أو أجهزة كبيرة **Main frame**، كما إنها تخدم عدة مستخدمين في نفس الوقت على الشبكة، وتسمح للمستخدمين بالمشاركة في الموارد المادية والبرمجية .

بالإضافة إلى إنها تقدم خدمات طباعة الملفات وخدمات الويب ، كما إن مواقع الويب تستخدم المخدمات لتخزين صفحات الويبi ومن أنظمة تشغيل المخدمات النموذجية نظام **Windows2000 & Unix** . وظهر مؤخراً نظام **Linux** .

٣,٦.أنظمة تشغيل المعالجات المتعددة

من الطرق الشائعة للحصول على طاقه حسابيه كبيرة توصيل عدة معالجات **CPU** في نظام واحدأوتسمى هذه الأنظمة بالحواسيب المتوازية **Parallel Computers**، أو الحواسيب المتعددة **Multi Computers**، أو المعالجات المتعددة **Multi Processor** وذلك حسب طريقة اتصال المعالجات مع بعضها البعض والموارد المشتركة بينها. وتحتاج هذه الأنظمة إلى أنظمة تشغيل خاصة لكنها في العادة تكون أنظمة، تشغيل مخدمات معدله لها مزآيا خاصة لتحقيق الاتصال بين المعالجات .

٤,٦. أنظمة تشغيل الحاسوب الشخصي

النوع الآخر من أنظمة التشغيل هو أنظمة الحاسوب الشخصي التي تنحصر مهمتها في تقديم واجهة جيدة للمستخدم . كما تستخدم هذه الأنظمة بشكل واسع لمعالجة النصوص والجداول الممتدة والوصول للإنترنت. ومن الأمثلة الشهيرة لهذا النوع نظام **Linux & Windows2000 & Windows 98** .

٥,٦. أنظمة تشغيل الزمن الحقيقي

يمتاز هذا النوع من أنظمة التشغيل على أنه يعتمد على الزمن كوسيط أساسي. فمثلاً أنظمة التحكم بالعمليات الصناعية ، تقوم حواسيب الزمن الحقيقي بجمع جميع البيانات عند عمليه الإنتاج ثم بعد ذلك تستخدمها للتحكم في الآت المصنع. كما إن هذه الأنظمة في الغالب تحتوي على نقاط حرجة يجب الاهتمامأ بها فعلى سبيل المثال

اذا كان لدينا سيارة تتحرك ضمن خط إنتاج السيارات فإن هنالك عدة أفعال يجب القيام بها في لحظات زمنية معينة ، فمثلاً إذا بدأ روبوت اللحام عمله باكراً أو متأخراً فإن السيارة ستصبح تالفه .

وهذا يعني أن لدينا نظام زمن حقيقي صلب **Hard Real Time System** .

كما إن هنالك نوع آخر من أنظمة الزمن الحقيقي، وهي أنظمة الزمن الحقيقي المرنة **Soft Real Time System**، وهي التي تمكننا من تجاوز بعض الخطوط الحرجة دون حدوث أي مشاكل بهاi فعلي سبيل المثال

أنظمة الصوت الرقمي والوسائط المتعددة . ومن أشهر أنظمة تشغيل الزمن الحقيقي الشهيرة نظامي QNX & VXworks .

٦,٦. أنظمة التشغيل المضمنة

يعمل هذا النوع من أنظمة التشغيل مع حواسيب تتحكم بأجهزة لا تصنف بشكل عام على أنها حواسيبi مثل أجهزة التلفاز و أفران الميكروويف و الهواتف النقالة. كما أنها تمتاز في غالب الأحيان بنفس مزايا أنظمة الزمن الحقيقي لكن لها قيود أخرى بالنسبة للحجم ومتطلبات الذاكرة والقوة الحسابيةi مما يجعلها مميزةi ومن أمثلة هذه الأنظمة نظاما **Windows CE & Plam OS**.

٧,٦. أنظمة تشغيل البطاقات الذكية

تعمل أنظمة التشغيل الأصغر حجماً على البطاقات الذكية **iSmart Cards** وهي عبارة عن أجهزة بحجم البطاقة الائتمانية **Credit Card**، بالإضافة إلى أنها تحتوي على معالج **CPU** iكما أنها تمتاز بقيود قاسية على القوة الحسابية وحجم الذاكرةi لذلك فنجدها تقوم بوظيفة واحدة مثل الدفع الإلكتروني، وبعضها يستطيع القيام بعدة وظائف على نفس البطاقة الذكية .

٧. مفاهيم نظم التشغيل

يعتبر المختصون بنظام التشغيل يونكس، أن نظام التشغيل يتألف من ثلاثة أجزاء رئيسية، هي:

- النواة (Kernel) .
- القشرية (Shell) .
- نظام الملفات (File system) .

١,٧. النواة (Kernel)

بينما يتجه مستخدمو دوس/ويندوز إلى عدم استخدام المصطلح "النواة"، واستخدام مصطلح "القشرية"، أحياناً فقط. فالنواة هي مجموعة الوظائف الأدنى مستوىً في نظام التشغيل، والتي تحمّل إلى الذاكرة، كلما قمت بتشغيل الجهاز، وذلك مباشرة بعد أن تعمل بعض الوظائف الموجودة في بيوس.

٢,٧. القشرية (Shell)

القشرية هي الواجهة المرئية لنظام التشغيل، وهي عبارة عن برنامج يعمل في الطبقة العليا منه، ويسمح للمستخدمين بإصدار الأوامر إليه. يوجد لنظام يونكس عدد من القشريات، القشرية .القياسية لنظام التشغيل دوس، هي البرنامج الموجود في ملف **Commnad.com.**

٣,٧. المهمة Process

استخدمت هذه الكلمة تبادليا مع كلمة عملية ولها عدة تعريفات منها:

- برنامج تحت التنفيذ.
- نشاط غير متزامن.
- وحدة إنجاز الأعمال.
- الكينونة التي تعطى لها المعالجات.

لا يوجد تعريف متفق عليه عالميا لكن تعريف **البرنامج تحت التنفيذ هو أكثر استخداما** ، والبرنامج هو كينونة غير حية وفقط عندما يبدأ المعالج في العمل على هذا البرنامج تصبح هذه الكينونة حيةi وندعوها بالمهمة.

٤,٧. المقاطعة Interrupt

عملية القطع هي حدث يغير من التسلسل الذي تنفذ فيه الأوامر بواسطة المعالج وعادة يتم بواسطة مكونات فيزيائية في الحاسوبi وهنالك ست درجات للمقاطعة هي :

١-المقاطعة نتيجة استدعاء المشرف :

وهي تقنية تساعد في الحفاظ على أمن نظام التشغيل من المستخدمين حيث إن المستخدم يجب أن يطلب الخدمة ن خلال استدعاء المشرف. وبالتالي فإن نظام التشغيل(**OS**) قد يرفض طلباً معيناً إذا كان المستخدم لا يملك الصلاحية المناسبة .

٢-مقاطعات الإدخال والإخراج : I/O Interrupts

تحدث عندما تكتمل عملية إدخال أو إخراج iأو حدث خطأ في الإدخال والإخراج، أو عند تجهيز جهاز معين.

٣-المقاطعات الخارجية External Interrupts:

مثل انتهاء الكوانتوم (**Quantum**) على الساعة والتي تسبب مقاطعة (**Interrupt**) نظام التشغيل.

٤-مقاطعة الاستئناف Resume Interrupt :

تحدث عندما يضغط المشغل على زر الاستئناف للوحة التحكم، أو عند وصول أمر استئناف معالج بالإشارة إليه من معالج آخر في نظام المعالجات المتعددة.

٥-مقاطعات تدقيق البرامج: وتحدث هذه المقاطعات لأحد الأسباب التالية:

- مشاكل القسمة على الصفر.
- محاولة تنفيذ شفرة عملية غير صحيحة.
- محاولة الرجوع إلى مصدر محمي.

- محاولة الرجوع إلى موقع في الذاكرة خارج حدود الذاكرة الحقيقية.

٦-**مقاطعة تدقيق الآلة:** وتتم نتيجة لحدوث خلل في جهاز الحاسوب.

٥,٧. إدارة الذاكرة الرئيسة

إحدى وظائف نظام التشغيل هو إدارة موارد الحاسوبأو أهمها الذاكرة الرئيسية، وذلك لأنها المكان الوحيد الذي منه تستدعي وحدة المعالجة المركزية إيعازات البرامج والبيانات المراد تنفيذها. يسمى الجزء من نظام التشغيل الذي يتولى مهمة إدارة الذاكرة بمدير الذاكرة **Memory Manager** ومن أهم مهامه:

- ✓مراقبة حالة جميع مواقع الذاكرة من حيث الفراغ، وذلك لتسكين المهمات المراد تنفيذها أو الامتلاء من أجل تفريغ المواقع بعد انتهاء المهمات من التنفيذ .
- ✓تحديد الطريقة التي من خلالهايتم توزيع المواقع الفارغة للعمليات المراد تنفيذها، مع تحديد الأولويات في التسكين .
- ✓نقل العمليات التي تم تنفيذها من الذاكرة الرئيسية إلى الذاكرة الثانوية (القرص الصلب) أو العكس .

إن دارة وتنظيم الذاكرة الرئيسية يعني في المحل الأول كيف توزع المواقع الفارغة في الذاكرة أهل تخصص لمهمة واحده أم توزع على عده مهمات؟ وإذا اخترنا الخيار الثاني، فهل تقسم المواقع بالتساوي للمهمات أم تقسم حسب حاجة المهمة ؟، وللإجابة عن كل هذه التساؤلات تجدها عزيزي الدارس عندما ندرس الوحدة الرابعة بإذن الله.

٦,٧. إدارة الذاكرة الافتراضية

المقصود بمصطلح الذاكرة الافتراضية هو القدرة علي عنونة مواقع أكبر بكثير من الحيز الحقيقي المتاح في الذاكرة الرئيسية، مما يعطي إحساسا لدى المستخدم بإن برنامجه موجود بالكامل في الذاكرة الرئيسية، بينما هو في الحقيقة موجود في وسائط التخزين الثانوية(القرص الصلب) أولا يوجد في الذاكرة الرئيسية إلا الجزء الفاعل .

وتعتبر هذه الطريقة من أنجح الطرق أوالأساليب في نظام البرمجة المتعددةi كما إنه يتم استخدام هذه الطريقة بإحدى أسلوبين:

١. أسلوب التصفح **paging**.
٢. أسلوب القطاعات **portioning**.

٧,٧. الإدخال والإخراج INPUT/OUTPUT

إحدى الوظائف الأساسية لنظام التشغيل هي التحكم بكل طرفيات الـ **I/O** للحاسوب و قدرته على إرسال الأوامر إلى الأجهزة المحيطة، استقبال المقاطعات ومعالجة الأخطاء، أي إمكانية الاتصال مع الأجهزة الطرفية القياسية والأجهزة الصناعية الأخرى، ففي تطبيقات التحكم الصناعي، يمكن لمدخلات الحاسوب أن تكون مخرجات لمقياس الحجم أو لمقياس درجة الحرارة أو لجهاز مكتشف الحرائق ...إلخ. وبالمقابل مخرجات الحاسوب يمكن أن تكون الأوامر الرقمية، لأجل تغيير سرعة محرك أو فتح صمام أو التحريض على القراءة التالية لمقياس الحجم العددي. باختصار: إن الحاسوب ذا الاستخدام العام (العالمي) يجب أن يكون قادراً على التوافق مع الخصائص الكثيرة والمختلفة للأجهزة الطرفية في الأوساط المختلفة. سوف ندرس مبادئ الـ **I/O** في الوحدة الرابعة بإذن الله.

إدارة القرص التخزيني

يعتبر القرص التخزيني من أهم وحدات الحاسوب، حيث يتم فيه تخزين البيانات وإخراجها منه عند الحاجة ، لذلك فهي تحتاج إلى اهتمام وإدارة من قبل نظام التشغيل وخاصة في أنظمة التشغيل متعددة المستخدمين، حيث يكثر استخدام هذه الوحدة .

٨,٧. خصائص الملف File Attributes

عندما نتحدث عن الملف نعني أن يكون له اسم وبيانات، وأن يكون نظام التشغيل له معلومات عن الملفi مرتبطة به مثل التاريخ وزمن إنشاء الملف وحجمه. وهذه العناصر تسمى خصائص الملف وتختلف الخصائص من نظام

إلى نظام آخر. •القائمة أدناه توضـــــح بعض الخصائص المعـــــــروفة للملفـــــــاتi وليس بالضرورة أن تتوفر كل هذه الخصائص في نظام التشغيل الـــــــــــواحد.

Protection	**How can access the file &in what way**
Password	**Password needed to access the file**
Creator	**Id of person how created the file**
Owner	**Currant owner**
Hidden flag	**0 for normal , 1 for do not display in listing**
System flag	**0 for normal file, 1 for system file**
Key length	**Number of bytes in the key field**
Creation time	**date & time file was created**
Current size	**Number of bytes in the file**
Maximum size	**Maximum size file may grow to**

٩,٧. الحماية

بتطور نظام الحاسوب وسهولة استخدامه زادت الاعتماديةعليه. لهـذا بـرزت مـشكلة الحماية، أي الأمن بازدياد شديد.

أسئلة تقويم ذاتي

١- اذكر أهم وظائف مدير الذاكرة .

٢- بين الأجزاء الرئيسة التي تكون نظام يونيكس.

٣- ما هى درجات المقاطعة؟

الخلاصة

عزيزي الدارس، بعد أن قمت بدراسة الوحدة الأولى من هذا المقرر سنلخص ما درسناه فيها بالنقاط التالية:

- وضحنا المكونات الأساسية للحاسوبأو قسمنا نظام الحاسوب إلى ثلاث طبقات:
  الطبقة الأولى: طبقة المكونات المادية.
  الطبقة الثانية : طبقة برامج النظام.
  الطبقة الثالثة: طبقة البرامج التطبيقية.
- عرفنا نظام التشغيل بأنه برنامج يعمل بين مستخدم الحاسوب وجهاز الحاسوب وهو يمكّن المستخدم من تنفيذ برمجياته بسهولة وبكفاءة عالية.
- يقوم نظام التشغيل بوظفتين منفصلتين عن بعضهما وهما:
  آلة ظاهرية: تسهل استخدام المكونات المادية باستخدام استداعاءات النظام.
  مدير الموارد: إدارة جميع أجزاء الحاسوب.
- يتم توزيع مشاركة الموارد بطريقتين:
  التوزيع الزماني: يمكن لكل برنامج استخدام المعالج لفترة زمنية معينة.
  التوزيع المكاني: تقسم الذاكرة الرئيسية عادة بين عدة برامج تعمل على الجهاز بحيث تنفذ العمليات حسب أولوياتها.
- يقوم أي نظام تشغيل بالوظائف التالية:
  ١. تمكين الاتصال بين الحاسوب والمستخدم.
  ٢. توزيع المعدات المشتركة على المستخدمين في الشبكة.
  ٣. يسهل الاتصال بين مكونات الحاسوب.
  ٤. الحماية من الأخطاء ومراقبة النظام.
  ٥. جدولة استخدام المصادر واستقلالها.
  ٦. يتعقب الملفات على الأقراص.

٧. حماية النظام.

- تطورت أنظمة التشغيل تطوراً كبيراًوقد قسمنا مراحل التطور إلى خمسة أجيال.
- وفقاً لتطور نظم التشغيل فإن هناك عدة مفاهيم مثل:
  - النظم الدُفعية البسيطة Simple Batched Systems ·
  - الـــنظم الدُفعيـــة متعـــددة البرمجـــة Multiprogramed Batched Systems.
  - نُظم تقاسم الزمن Time-Sharing System·
  - نُظم الحواسيب الشخصية Personal-Computer Systems .
  - النُظم التفرعية Parallel Systems ·
  - النُظم الموزعة Distributed Systems ·
  - نُظم الزمن الحقيقي Real -Time Systems ·
- هنالك عدة أنواع من الأنظمة المتنوعة:
  ١. أنظمة التشغيل للأجهزة الكبيرة.
  ٢. أنظمة تشغيل المخدمات.
  ٣. أنظمة تشغيل المعالجات المتعددة.
  ٤. أنظمة تشغيل الحاسوب الشخصي.
  ٥. أنظمة تشغيل الزمن الحقيقي.
  ٦. أنظمة التشغيل المضمنة.
  ٧. أنظمة تشغيل البطاقات الذكية.
- يتألف نظام التشغيل من ثلاثة أجزاء رئيـــسية، هـــي: النـــواة، القـــشرية،نظام الملفات.
- عرفنا المهمة بأنها برنامج تحت التنفيذ، وعملية القطع بأنها حدث يغير مـــن التسلسل الذي تنفذ فيه الأوامر بواسطة المعالج.

- إحدى وظائف نظام التشغيل هو إدارة موارد الحاسوب و أهمها الذاكرة الرئيسية وذلك لأنها المكان الوحيد الذي منه تستدعي وحدة المعالجة المركزية إيعازات البرامج والبيانات المراد تنفيذها.
- مصطلح الذاكرة الافتراضية هو القدرة على عنونة مواقع أكبر بكثير من الحيز الحقيقي المتاح في الذاكرة الرئيسية.
- إحدى الوظائف الأساسية لنظام التشغيل هي التحكم بكل طرفيات الـ **I/O** للحاسوب، و قدرته على إرسال الأوامر إلى الأجهزة المحيطة، استقبال المقاطعات ومعالجة الأخطاء.
- نعني بالملف أن يكون له اسم وبيانات وأن يكون نظام التشغيل له معلومات عن الملف مرتبطة به مثل: التاريخ، وزمن إنشاء الملف، وحجمه. وهذه العناصر تسمى خصائص الملف.

لمحة مسبقة عن الوحدة التالية

عزيزي الدارس، في الوحدة التالية من هذا المقرر سنعطيك فكرة عن مفهوم وتعريف المهمات وحالاتها المختلفة وكيف ولماذا تنتقل المهمة بين هذه الحالات وكيف، يتم إنشاء كتلة تحكم المهمات، كما سوف نناقش مشكلة معقدة يقوم بها نظام التشغيل وهي عملية تخصيص الموارد أو ما يسمي بجدولة المهمات. أتمنى أن تتقدم في دراستها باجتهاد حتي تحقق الأهداف المرجؤة منك.

إجابات التدريبات

تدريب (١)

١-ظهرت شبكات الحاسوب وأصبح بالإمكان الاتصال بالحاسوب الرئيسي من طريــق نهاية طرفية من مكان بعيد في الجيل الثالث.

٢-من أبرز ملامح الجيل الرابع يتمثل في ظهور الآتي:

- ✓ الدوائر الكهربائية المتكاملة الكبيرة .
- ✓ أقراص الليزر، والأقراص الممغنطةi والأشرطة الممغنطة.
- ✓ أجهزة حاسوب متنوع.**Mainframe, Minicomputer, PC** ·
- ✓ أنظمة تشغيل متنوعة ومتطورة شبكية و موزعة·
- ✓ أول معالج دقيق·
- ✓ لغات البرمجة للجيل الرابع، وقواعد البيانات والشرائح الممتدة.

إجابات أسئلة التقويم الذاتي

تقويم ذاتي رقم (١)

١-

- إدارة جميع أجهزة المكونات المادية للحاسوب.
- تقديم واجهة بسيطة للمستخدم لكي يتمكن من التعامل مع المكونات المادية.

٢-تعمل المترجمات في نمط المستخدم وبالتالي يمكن للمستخدم تعديلها ، فمثلاً إذا لم يرغب المستخدم، في التعامل مع مترجم معين فيمكنه كتابه مترجم خاص به واستخدامه بدلاً من المترجم السابق ، لكنه لا يستطيع تغيير معالج مقاطعة الساعة لأنه جزء من نظام التشغيل ويكون محمياً من قبل المكونات المادية من محاولات التعديل من قبل المستخدم .

تقويم ذاتي رقم (٢)

١-يمكن أن نعرف نظام التشغيل على أنه عبارة عن برنامج :

- ✓ يعمل كوسيط ما بين مستخدم الحاسوب ومكوناته المادية (العتاد)، أي إن الغرض منه هو تزويدنا ببيئة عمل يستطيع المستخدم من خلالها تنفيذ برامجه.
- ✓ يتحكم في تنفيذ برامج المستخدم، لكي يمنع حدوث الأخطاء والاستخدام لغير المناسبi للحاسوب وبشكل خاص التحكم بأجهزة الإدخال والإخراج **I/O**،
- ✓ ينفذ دائماً في الحاسوب ويسمى باللب **(Kernel)** وما عدا ذلك بالبرامج التطبيقية.

٢-يمكن تقسيم أنظمة التشغيل من خلال

- ✓ عدد المهمات التي يمكن أن تنجز آنياًأ أي مهمة واحدة أو عدد من المهمات.
- ✓ عدد المستخدمين الذين يمكنهم استخدام النظم آنياً، أي مستخدم وحيد أو عدد من المستخدمين.

تقويم ذاتي رقم (٣)

راجع القسم الثالث "وظائف نظم التشغيل"

تقويم ذاتي رقم (٤)

تتميز النظم الدُفعية متعددة البرمجة بالخصائص التالية:

- يقوم النظام بحفظ عدد من البرامج في الذاكرة.
- يختار النظام أحد المهمات ويقوم بتنفيذها إلى أن تنتهي أو تطلب عملية دخل/خرج.
- يحجز النظام جهاز دخل/الخرج لأحد المهمات.

تقويم ذاتي رقم (٥)

١-يتكون نظام التشغيل يونيكس من ثلاثة أجزاء رئيسية، هي:

- النواة(**Kernel**) .
- القشرية(**Shell**) .
- نظام الملفات(**File system**) .

٢-هنالك ست درجات للمقاطعة هي :

- المقاطعة نتيجة استدعاء المشرف .
- مقاطعات الإدخال والإخراج :**I/O Interrupts**.
- المقاطعات الخارجية **External Interrupts**.

- مقاطعة الاستئناف **Resume Interrupt** ·
- مقاطعات تدقيق البرامج، وتحدث هذه المقاطعات لأحد الأسباب التالية:
- مقاطعة تدقيق الآلة وتتم نتيجة لحدوث خلل في جهاز الحاسوب.

٣-أهم مهام مدير الذاكرة هي:

- مراقبة حالة جميع مواقع الذاكرة·
- تحديد الطريقة التي من خلالها يتم توزيع المواقع الفارغة للعمليات المراد تنفيذها مع تحديد الأولويات في التسكين .
- نقل العمليات التي تم تنفيذها من الذاكرة الرئيسية إلي الذاكرة الثانوية (القرص الصلب) أو العكس .

مسرد المصطلحات

Hardware	المكونات المادية
Kernel Mode	نمط النواة
User Mode	نمط المستخدم
Interpreter	مفسر
Editors	محررات النصوص
Compilers	مترجمات
Operating System	نظام التشغيل
Machine Language	لغة الآلة
Physical Devices	الأجهزة الفيزيائية
Software	البرمجيات
Supervisor	المشرف
Monitor	المراقب
Executive	المنفذ
Virtual Machine	آلة ظاهرية
Resource Manger	مدير الموارد
System Calls	استدعاءات النظام
Time Multiplexing	التوزيع الزماني
Space Multiplexing	التوزيع المكاني
Assembly Language	لغة التجميع
Integrated Circuit	الدوائر الكهربائية المتكاملة
Sharing Time	المشاركة في الوقت
Network Computer	شبكات الحاسوب

Batched Systems	النظم الدُفعية
Multiprogramed	متعددة البرمجة
Personal-Computer	الحاسوب الشخصي
Distributed Systems	النُظم الموزعة
Real -Time Systems	نُظم الزمن الحقيقي
Smart Cards	البطاقات الذكية
Credit Card	البطاقة الائتمانية
Kernel	النواة
Shell	القشرية
File system	نظام الملفات
Process	المهمة
Interrupt	المقاطعة
Memory Manager	مدير الذاكرة
paging	التصفح
portioning	القطاعات
INPUT/OUTPUT	الإدخال والإخراج

المراجع

١-يمان اللبني وأسامة العبد الله، تصميم وتنفيذ نظم التشغيل الحديثة، شعاع للنشر والعلوم، سوريا-حلب ٢٠٠٥م.

٢-ج آرتشر هاريس (ترجمة أمين أيوبي)، أنظمة تشغيل الحاسوب، أكاديمياً، بيروت ٢٠٠٢م.

٣-عبد الرءوف الحلاق ومنتصر خاطر، أنظمة التشغيل، منشورات جامعة القدس المفتوحة، ١٩٩٦م.

٤-محمد أحمد فكرين، نظم تشغيل الحاسبات، دار المريخ ١٩٩٦م

٥- Silberschatz, A. and Galvin, P.B., "Operating System Concepts", Fifth edition Addison-Wesley, Reading, MA, 199٧.

6- Tanenbaum, Andrew S., "Modern Operating Systems", Prentice-Hall, Englewood Cliffs, NJ, 1992.

http://www.personal.kent.edu/~rmuhamma/OpSystems/os.html
http://www.cag.lcs.mit.edu/~rinard/osnotes/
http://williamstallings.com/OS4e.html


الوحدة الثانية
إدارة المهمات
Processes Management

محتويات الوحدة

المقدمة

تمهيد

أهلاً بك عزيزي الدارس في الوحدة الثانية من مقرر "**نظم التشغيل**" الجزء الأول وهي بعنوان "**إدارة المهمات** "

تقدم لك هذه الوحدة عزيزي الدارس فكرة عن مفهوم وتعريف المهمات وحالاتها المختلفة وكيف ولماذا تنتقل المهمة بين هذه الحالات ،وكيف يتم إنشاء كتلة تحكم المهمات، كما سوف تناقش مشكلة معقدة يقوم بها نظام التشغيل وهي عملية تخصيص الموارد أو ما يسمى بجدولة المهمات.

تتألف هذه الوحدة من ثلاثة أقسام تعتبر المرتكزات الأساسية لهذه الوحدة، فالقسم الأول يتحدث عزيزي الدارس عن المهمات بشكل عام، ونجد أن كل كلمة مهمة هي كلمة مرادفة لكلمة أعملية والقسم الثاني سوف نتحدث فيه عن كتلة تحكم المهمات Process Control Block (PCB) و الخيوط Threads وأخيرا في القسم الثالث سوف نتحدث عما يسمى بجدولة المهمة Process Scheduling وتغطي هذه الأقسام الأهداف الواردة في بداية هذه الوحدة.

أهداف الوحدة

عزيزي الدارس،

بعد فراغك من دراسة هذه الوحدة ، ينبغي أن تكون قادراً على أن:

- تشرح مفهوم المهمة .
- تبين حالات المهمة المختلفة وكيفية التنقل بينها.
- تنشئ كتلة تحكم المهمات.
- تشرح مفهوم فضاء العناوين والخيوط.
- تشرح أهمية جدولة المهمات.
- تذكر أنواع الأولويات.
- تذكر الخوارزميات المختلفة لجدولة المهمات.

١. مفهوم المهمات

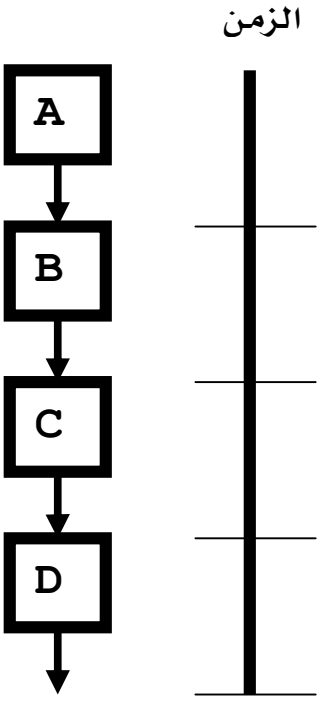
لابد إنك لاحظت عزيزي الدارس إن جميع الأجهزة الحديثة تقوم بتشغيل عدد من البرامج في نفس الوقت فمثلا نجد أثناء تشغيل برنامج معين يقوم الحاسوب بـــ :

- القراءة من القرص ·
- استخدام الطابعة ·

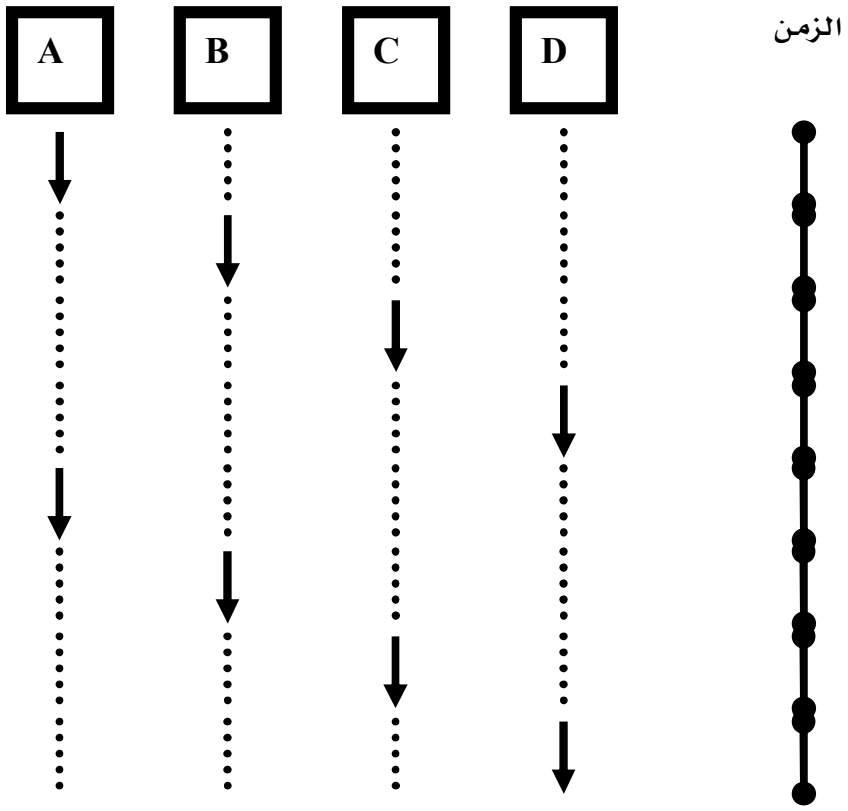
ويستطيع المعالج في متعدد البرمجة التبديل من برنامج إلى اخرi ومع العلم بأن المعالج في الواقع لا يعمل إلا مع برنامج واحد فقط في أي لحظة زمنية معينةi ولصغر الفترة الزمنية التي يشغلها كل برنامج (اقل من $\frac{1}{100}$ من ميلي ثانية)، يشعر المستخدم بأن هنالك معالجة متوازية لهذه المهماتi وإنما في الحقيقة هنالك معالجة متسلسلة لهذه المهمات وتعرف بالتوازي الشكلي **(Pseudo Parallelism)** ويمكن توضيح ذلك في الشكل رقم(٢-١) .أ و ب.

(أ)

برمجة متعددة لأربعة برامج في كل لحظة زمنية معينة يوجد فقط مهمة واحده تعمل على المعالج



(ب) أربع مهمات متسلسلة ومستقلة في كل لحظة زمنية معينة.
يوجد فقط مهمة واحده تعمل على المعالج



الشكل رقم (٢-١) : معالجة متسلسلة لأربعة مهمات (التوازي الشكلي)

مما سبق يجب تنظيم جميع البرامج القابلة للتنفيذ في شكل مهمات متسلسلة **Sequential Processes** i وتختصر بالمهماتi وتسمي أحياناً بالعمليات.

تعريف

يمكن تعريف المهمة بأنها برنامج قيد التنفيذ، له بيانات مدخله وإخراج ومؤشر يصف حالته.

تلاحظ عزيزي الدارس إن هنالك ثمة تشابه بين البرنامج والمهمة، نعم هنالك فرق بسيط يمكن توضيحه من خلال المثال التالي:

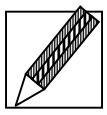
افترض أن أستاذ ماده نظم التشغيل دخل القاعة لتدريسها وقام أحد الطلاب يطلب من الأستاذ شرح السؤال رقم (٧) في التمرين السابق، وقام آخر يطلب من الأستاذ شرح جزئية محددة من المحاضرة السابقة .ولكن قرر الأستاذ البدء في المحاضرة وأرجأ طلبات الطلاب إلى نهاية المحاضرة. فأخرج المادة العلمية للمحاضرة بعدما شغل مكبر الصوت وجهاز الحاسوب والـ **Projector**.

في هذا المثال، .يمكن أن نقول

البرنامج	هو المادة العلمية
المعالج	هو الأستاذ
المهمة	هي فعالية تتألف من قراءة المادة العلمية واستغلال الموارد

وبعد مضي فترة زمنية يطرق طالب باب القاعة للدخول للمحاضرة يوقف الأستاذ المحاضرة ويتجه إلى الباب (مقاطعة) وبعد نقاش بينهما يسمح الأستاذ للطالب بالدخول ثم يواصل الأستاذ محاضرته من المكان الذي توقف عنده.

تدريب (١)



في المثال السابق، حدد الموارد المتاحة من وجهة نظر نظم التشغيل.

٢. حالات المهمة

هنالك كثير من الأحداث المتنوعة التي تدعو المهمة لتغير حالاتها، وبالتالي يمكن أن تكون في واحدة من الحالات التالية:

أ-مهمة جديدة **New Process**

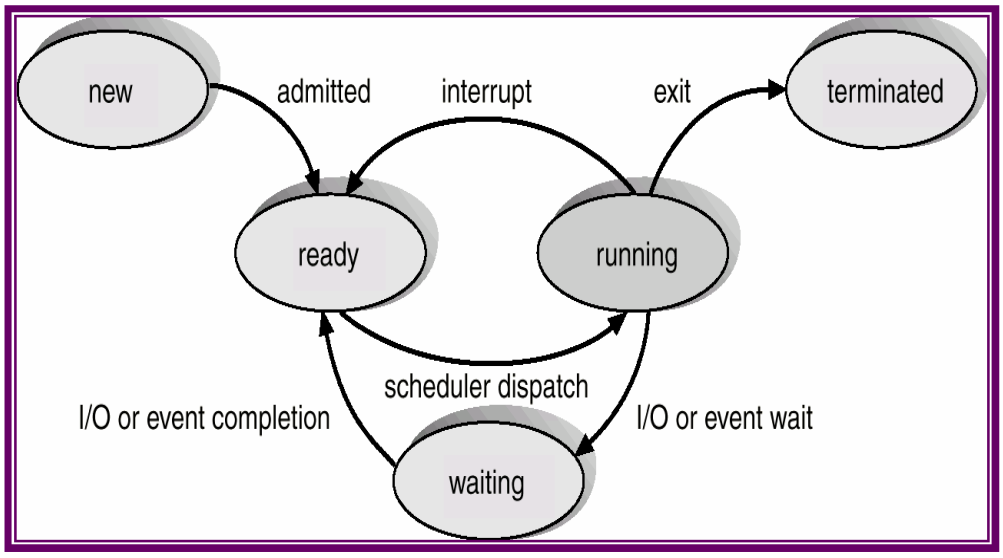
ب-مهمة شغالة **Running Process**

ج-مهمة جاهزة أو مستعدة **Ready Process**

د-مهمة متوقفة أو معاقة **Waiting Process**

هـ-إنهاء مهمة **Terminated Process**

يوضح الشكل رقم (٢-٢) هذه الحالات وطرق الانتقال من حالة إلى أخرى.



الشكل رقم (٢-٢) يوضح حالات المهمة وطرق الانتقال بينها [٤]

تتضمن أي مهمة الأتي:

- شفرة البرنامج.
- عداد البرنامج **(program counter)**.
- محتويات السجلات.

- مكدسة المهمة (البيانات المؤقتة مثل **Parameters**).
- قسم البيانات (مثل **Global Variables**.

وسوف نقوم عزيزي الدارس الآن بشرح هذه الحالات بالتفصيل

هنالك أربع حالات تستوجب إنشاء أو إنتاج مهمة جديدة وهي:

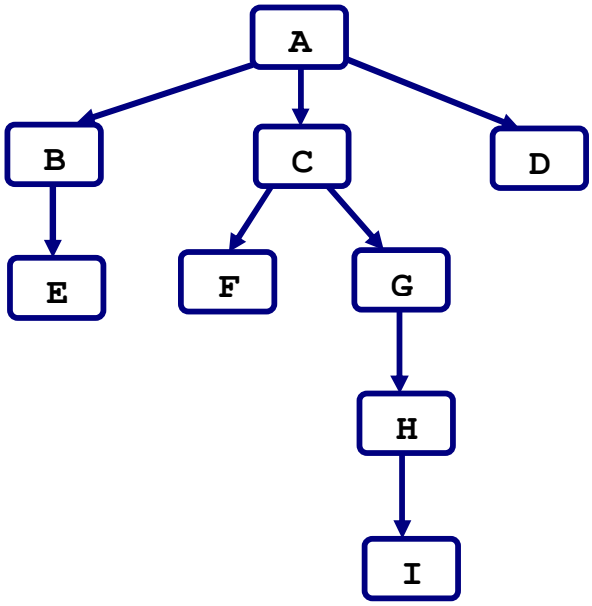
١,٢. مهمة جديدة New Process

- عند تهيئة النظام.
- بناءً على طلب من مستخدم.
- بناءً على طلب عملية قيد التنفيذ.
- بدء عمل دفعي (**batch job**) .

ويتطلب إنتاج مهمة أيضا عدة معاملات:

- تسمية المهمة.
- إدخالها في جدول المهمات المعروفة للنظام. ١
- تحديد الأولوية الابتدائية للمهمة.
- إنتاج كتلة تحكم المهمات.٢
- تخصيص الموارد الأولية للمهمة.

قد تنتج المهمة مهمة جديدة، تسمى المهمة المُنْتَجةُ بالمهمة الأصل أو الأب، بينما تسمى المهمات التي تم إنتاجها بالمهمات الفرعية أو الأبناءi وعليه نحتاج لأصل واحد لإنتاج الفرعiوهذا الإنتاج يشكل تركيباً هرمياً كما بالشكل رقم (2-3)



الشكل رقم (2-3): التركيب الهرمي لمهمة تنتج مهمات أخرى

٢,٢. مهمة شغالة Running Process

تستخدم المعالج بشكل فعلي في هذه اللحظة، أي إنها قيد التنفيذ في وحدة المعالجة المركزيةi وقد تنتقل المهمة من هذه الحالة إلى حالة:

- "**جاهزة** " عند حدوث مقاطعة.
- "**إنهاء**" عند انتهاء مهمتها والخروج **Exit** من المعالجة.
- "**توقف**" عند انتظارها لحدوث حدث خارجي مثل **Input/Output**.

٣,٢. مهمة جاهزة أو مستعدة Ready process

تكون المهمة في هذه الحالة جاهزة ومستعدة للعمل لكنها متوقفة مؤقتا للسماح بتشغيل عملية آخرى. ويمكن أن تكون هنالك أكثر من مهمة في وضع الاستعداد، وتسمى حينها قائمة المهمات المستعدة أو الجاهزة.

٤,٢. مهمة متوقفة أو معاقة Waiting Process

تكون المهمة في هذه الحالة متعطلة وغير قادرة على العمل(بعدما كانت في حالة التنفيذ) أي في حالة انتظار لحدوث حدث خارجي يمكنها متابعة التنفيذ وقد يحدث ذلك في اغلب الأحيان لاستكمال عملية دخل أو خرج، وهي بالتالي لا تحتاج إلى المعالجi وعندئذ تنتقل إلى حالة الاستعداد.

٥,٢. إنهاء مهمة Terminated Process

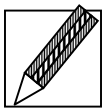
هنالك عدة أسباب تدعو لإنهاء المهمة ولكن معظمها تنتهي لأنها أنجزت عملهاi ومن الأسباب التي تسبب إنهاء المهمة هي:

- انتهاء طبيعي بعد إنجاز عملها·
- انتهاء بسبب خطأ ما أثناء تنفيذ العملية·
- انتهاء المهمة بسبب مهمة أخرى·

هنالك عدد من العمليات يمكن أن تتم على المهمةi ومن هذه العمليات:

- إنشاء مهمة·
- تدمير مهمة·
- تعليق مهمة·
- استئناف مهمة·
- تغيير أولوية مهمة·
- إعاقة مهمة·
- إيقاظ مهمة.
- إرسال مهمة·
- اتصال مهمة (تمكين مهمة من الاتصال بمهمة أخرى)·

تدريب(٢)



ما العدد الأقصى والأدنى للمهمات التي يمكن أن تكون في حالة تشغيل أو توقف ،أو أن تكون جاهزة إذا كان النظام يحتوي على N من وحدات المعالجة المركزية؟

أسئلة تقويم ذاتي

حدد الخدمات التي تطلبها المهمة عندما تكون في حالة إعاقة من وحدة المعالجة المركزية CPU.

٣. كتلة تحكم المهمات Process Control Block (PCB)

يتم إنشاء كتلة تحكم المهمات **PCB** لكل مهمة تم إنشاؤها وتعتبر وسيلة المتابعة الوحيدة للمهمة من وقت قبولها إلى أن يتم تنفيذها أوهي عبارة عن سجل أي إن كتلة تحكم المهمات تحوي سجلاً واحداً لكل عملية.

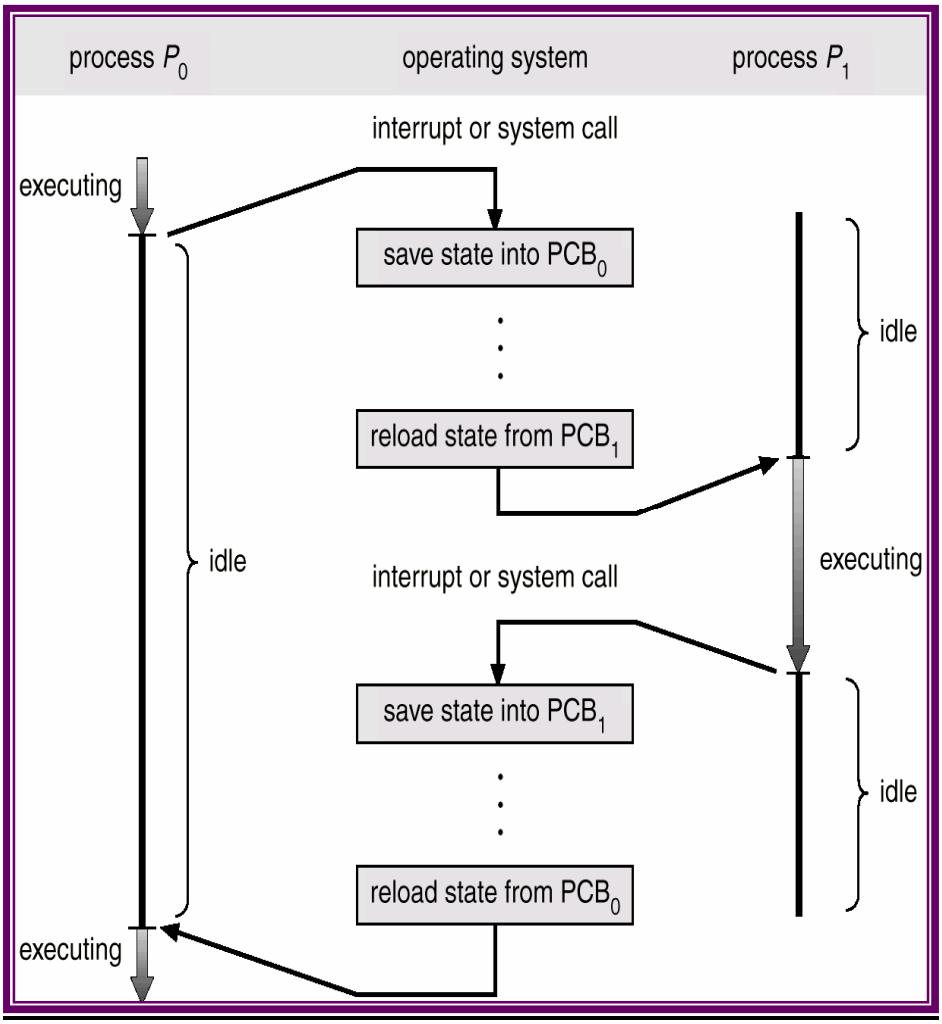
تخزن **PCB** على سجلات سريعة جدا أسرع كثيرا من سرعة عناصر الذاكرة الأساسية حتى يتمكن نظام التشغيل من متابعة حالات التنفيذ.وتحتوي **PCB** على كل البيانات اللازمة لنظام التشغيل لإدارة المهمة .ويبين الشكل (٢-٤) أهم الحقول في نظام تشغيل نموذجي.

رقم المهمة.
حاله العملية.
مؤشر المكدس.
عداد البرنامج.
المسجلات.
. . . . .
حقول عن إدارة الذاكرة.
حقول عن إدارة الملفات.

شكل رقم (٢-٤) : أهم الحقول في PCB في نظام تشغيل نموذجي

يبين الشكل رقم (٢-٥) كيفية استخدام .**PCBs** لكل مهمة، فنجد أن المهمة P_0 عندما بدأت في التنفيذ حدث لها مقاطعة من قبل نظام التشغيل ،وبالتالي قام نظام التشغيل بحفظ حالة المهمة في PCB_0 الخاص بها، وعندما أراد نظام التشغيل تنفيذ المهمة P_1 قام نظام التشغيل باسترجاع حالتها من PCB_1 الخاص بها.

هكذا يقوم نظام التشغيل بحفظ حالة المهمة في **PCB** الخاص بها عندما يقطع تنفيذها واسترجاع حالتها من **PCB** الخاص بها عند البدء في تنفيذها.



الشكل رقم (٢-٥) : كيفية استخدام .PCBs لكل مهمة [٤]

٤. الخيوط Threads

تسمى أحيانا بالمهمات الخفية **(light weight)** لأنها تمتلك كثيراً من خصائص المهمات ومعلوم إن لكل مهمة:

١,٤. فضاء العناوين

يشمل فضاء العناوين نص البرنامج والبيانات الخاصة به والموارد الخاصة به مثل الملفات وعمليات الأبناء ومعلومات التسجيل وغيرها من الأشياء ، وضعت جميع هذه الأشياء في فضاء واحد يسهل عمليه إدارة المهمة.

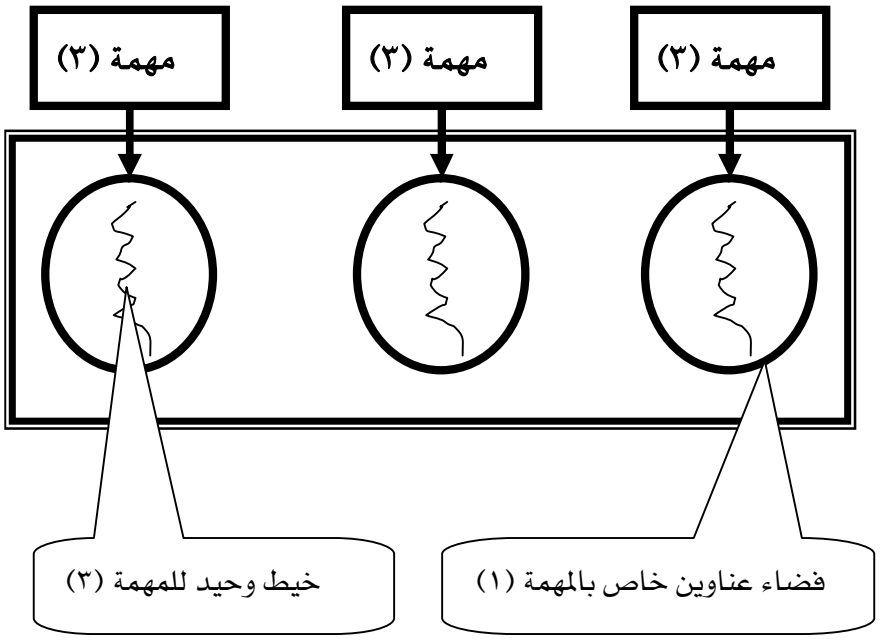
٢,٤. خيوط التنفيذ

في أنظمة التشغيل القديمة كان لكل مهمة خيط تنفيذ وحيد، ولكن في جميع الأنظمة الحديثة تمتلك المهمة عدداً من خيوط التنفيذ ويحتوي الخيط على معلومات حول عداد البرنامج الذي يحدد التعليمة التي تنفذ. ويحتوي الخيط أيضا على مسجلات ومكدس لتخزين خطوات التنفيذ السابقة وغيرها من الأشياء.

كما ذكرنا إن المهمة في أنظمة التشغيل القديمة تمتلك خيطا وحيدا ويكون الخيط في واحده من الحالات التالية:

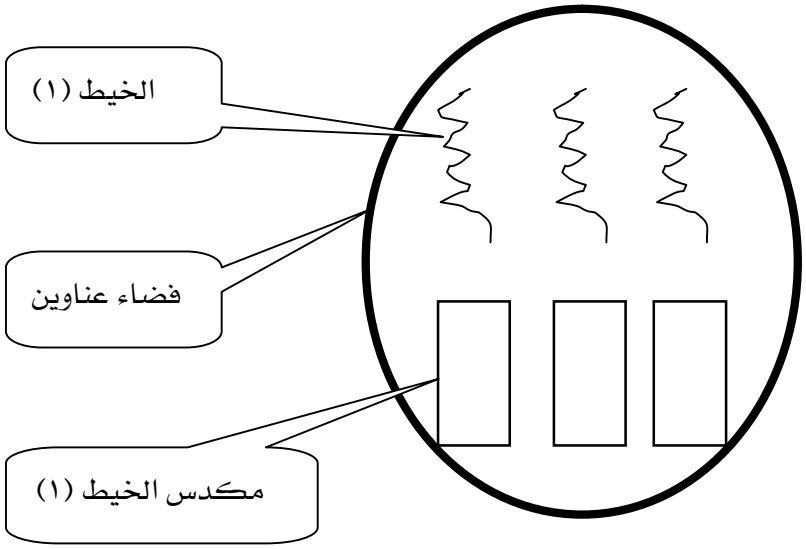
- شغال.
- متوقف.
- جاهز.
- منتهي.

والشكل رقم (٢-٦) يوضح ثلاث مهمات لكل مهمة فضاء عناوين خاص وخيطا وحيدا للتنفيذ.



شكل رقم (٢-٦) : المهمات وخيط التنفيذ في أنظمة التشغيل التقليدية

بينما يبين الشكل رقم (٢-٧) فضاء عناوين وحيد لثلاثة خيوط تنفيذ للمهمة .ولكل خيط مكدس" خاص" بها كما ذكرنا سابقا.



شكل رقم (٢-٧) المهمة وخيوط التنفيذ في أنظمة التشغيل الحديثة

لعل المستخدم لبرنامج **Word** يلاحظ فائدة استخدام الخيوط للمهمة الواحدة. فعندما تقوم بتحرير النص باستخدام لوحة المفاتيح أو الفأرة يقوم البرنامج أيضا بحفظ تلقائي للملف كل فترة وأخرى دون أن يعطل عمل تحرير النص ، لأن البرنامج في هذه الحالة يعمل علي خيطيين ،خيط يمكن من تحرير النص وخيط يمكن من حفظ الملفi وإذا كان البرنامج ذا خيط وحيد فإنه يتجاهل عملية تحرير النص حتى نهاية عملية الحفظ التلقائي ويصبح ذلك أمراً لا يطاق.

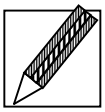
نشاط

في رأيك ومن خلال دراستك ما العمليات الضرورية التي يجـــب أن تتم حتى نتمكن من إنشاء مهمة؟

أسئلة تقويم ذاتي

ما محتويات PCB؟

تدريب (٣)



وضح أي من الأتي يعتبر مهمة Process وأيهما يعتبر خيطـــاً Thread

1. Program code.
2. Local or temporary data.
3. Global data.
4. Allocated resources.
5. Execution stack.
6. Memory management info.
7. Program counter.
8. Parent identification.
9. Thread state.
10. Registers.

٥. جدولة المهمة Process Scheduling

كما هو معلوم عزيزي الدارس، إن البرمجة المتعددة تستخدم لزيادة كفاءة النظامi عليه عندما تكون هنالك مهمه قد توقفت عن التنفيذ في انتظار متطلبات إدخال/إخراج مثلاً فإن مهمة أخرى تدخل حيز التنفيذi ولكن السؤال الذي يفرض نفسه هو

"كيف يتم اختيار هذه المهمة من مجموعة مهمات أخري" ؟

سياسة الجدولة هي التي تحل هذا الإشكال مع مراعاة عده معايير. وفيما يلي بعض المعايير شائعة الاستخدام:

[١] استخدام المعالج

وهو الزمن الفعلي لاستخدام المعالج ويعتبر من أهم المعايير في أنظمة المشاركة وغير ذي أهمية نسبياً في نظام المستخدم الواحد.

[٢] الطاقة الإنتاجية

هي عدد المهمات التي يمكن للنظام تنفيذها في فترة زمنية محددة أو بمعنى آخر هو معدل المهمات التي أنجزت. وسياسة الجدولة تسعى للحصول على أقصى طاقة إنتاجية ممكنة.

[٣] الوقت الكامل

وهو الزمن أو الفترة الزمنية الذي تقضيه المهمة في النظام منذ إنشائها وحتى نهاية تنفيذهاiوهو يتناسب عكسياً مع الإنتاجية.

[٤] زمن الاستجابة

هو الزمن المنقضي أو المستغرق بين استلام الطلب حتى ظهور النتائج

[٥] زمن الانتظار

هو متوسط الفترة الزمنية التي تقضيها المهمة في الانتظار .

[٦] العدل

ينبغي أن تكون طريقة الجدولة عادلة بحيث تحصل كل مهمة على فرصة متساوية للتنفيذ

[٧] الأولويات

كل مهمة تحمل عنواناً به إشارة تحدد أفضلية المهمة أو أولوية المهمة في التنفيذ، حيث تقوم طريقة الجدولة باختيار المهمات ذات الأولوية العالية قبل المهمات ذات الأولوية الأدنى.

حتى تحقق طريقة الجدولة أهدافها يجب أن تحدد هل المهمة تختص بالإدخال والإخراج؟ أم أنها خاصة بوحدة المعالجة المركزية وأولوية كل مهمة ومدي أهمية الاستجابة السريعة؟

٦. خوارزميات الجدولة

هي خوارزميات تكتب لتوضيح كيفية دخول المهمات إلى وحدة المعالجة المركزية وعادة ما تستخدم العلاقات الرياضية لتحديد زمن الدخول و زمن المعالجة. وقبل البدء عزيزي الدارس عليك الإلمام بالمصطلحات التالية حتى تستطيع أن تفهم ما هو المقصود

CPU utilization	جعل المعالج مشغولاً قدر الإمكان
Throughput	عدد المهمات التي يكتمل تنفيذها لكل وحده زمنية.
Turnaround time	كمية الوقت اللازم لتنفيذ مهمة معينة.
time Waiting	كمية الوقت التي تظل المهمة منتظرة في **ready queue** إلى أن يأتيها دورها في التنفيذ
Burst Time	هو الوقت الذي يستغرقه المعالج للانتهاء من المهمة.

١,٦ الأول في الوصول الأول في المعالجة

first come first serve(FCFS)

تعتبر خوإرزمية الأول في الوصول الأول في المعالجة **first come first serve(FCFS)** أبسط طريقة في الجدولة، حيث تعتمد على فكرة المهمة التي تصل أولاً تنفذ أولاًا وتسمى أيضا **first in first out** حيث يكون هنالك صف للمهمات الجاهزة للتنفيذ **(ready)**. عندما ينتهي المعالج من تنفيذ مهمة يتم اختيار المهمة التالية للتنفيذ من صف الانتظار، حيث يتم اختيار المهمة التي تكون في مقدمة الصف والتي تكون استغرقت أطول فترة انتظار، الجدول التالي يوضح عدداً من المهمات وزمن وصولها وزمن الخدمة لكل مهمة.

المهمة	زمن الوصول	زمن الخدمة
A	٠	٣
B	٢	٦
C	٤	٤
D	٦	٥
E	٨	٢

إذا نظرت للجدول أعلاه تجد إن تنفيذ المهام يتم بالترتيب حسب زمن الوصول. وهذه الطريقة جيده في حالات المهمات الطويلة ،ولكنها غير عادلة بالنسبة للمهمات الصغيرة فمثلا المهمة **E** في الجدول تحتاج للمعالجة لوحدتين زمنيتين لكنها تنتظر أطول فترة زمنية.

مثال

إذا كانت **P1, P2 and P3** عبارة عن ثلاث مهمات قادمات للمعالج بـــنفس الترتيـــب، يمكن لنا رسم مخطط يوضح لنا كيفية معالجة المهمات الثلاث وذلك من خلال مخطـــط **Gantt** التالي

يتضح لنا من المخطط أعلاه أن :

Waiting time for P1 = 0; P2 = 24; P3 = 27
Average waiting time: (0 + 24 + 27)/3 = 17

ثم تصل **P3** أولا ثم بعدها **P2**ولكن عزيزي الدارس ماذا يحدث عندما تصل **P1**أخيرا . هل يوجد فرق بينها وبين الترتيب السابق، نعم، يتضح لنا حسب مخطط **Gantt** التالي

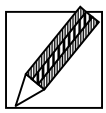
أن :

Waiting time for P1 = 6; P2 = 0; P3 = 3
Average waiting time: (6 + 0 + 3)/3 = 3

من المخطط السابق نلاحظ أن وقت الانتظار قل كثيرا عن السابقi بمعنى أنه إذا كانت هناك مهمة تحتاج مثلا إلى وقت طويل ثم مباشرة جاءت مهمة بعدها لا تحتاج إلا لوقت قصير فإنها تضطر إلى الانتظار في **ready queue** إلى أن ينتهي المعالج من المهمة التي معه .

والآن بما أن الطريقة الأخيرة جيده جدا، إذن لابد من إدخال بعض التعديلات عليها لتنتج طريقة جديدة تسمى أقصر المهمات أولاً (Shortest-Job-First (SJF.

تدريب (٤)



إذا كانت **A, B, C and D** عبارة عن أربعة مهمات قادمات للمعالج وفقاً للبيانات التالية

المهمة **Process**	زمن المعالجة **Burst Time**
A	**5**
B	**2**
C	**9**
D	**4**

بافتراض **all processes start running at the same time** إذا كان المعالج يعمل بجدولة الأول في الوصول الأول في المعالجة،

- ارسم مخطط **Gantt** ليوضح كيفية تنفيذها.
- احسب متوسط زمن الانتظار **Average waiting time**.
- احسب متوسط الزمن الكلي للتنفيذ **Average Turnaround time**.

يعتبر مهمة **Process** وأيهما يعتبر خيط **Thread**

٢,٦. أقصر المهمات الأول في المعالجة

Shortest Job First (SJF)

نعني بجدولة أقصر المهمات أولاً **Shortest Job First (SJF)** أن المهمة التي لا تستغرق وقتاً طويلاً في المعالج تدخل أولاً. أي أن هذه الطريقة تعتمد على فكرة أن يتم اختيار المهمة التي تحتاج لزمن تنفيذ أقصر (الزمن المقدر لانجازها اقل مقارنة ببقية المهمات الموجودة في صف الانتظار). وهذه الطريقة تعطي أفضلية للمهمات الصغيرة على حساب المهمات الكبيرة، وتنقسم هذه الطريقة إلى نوعين:

النوع الأول :غير إيقافي non preemptive

عند قدوم عدد من المهمات إلى المعالج يأخذ المهمة التي لا تأخذ وقتاً طويلاً أي أقل وقت، ولكن عند قدوم مهمة جديدة وكان الوقت اللازم لمعالجتها أقل من التي مع المعالج فإنه يتم تجاهلها ويكمل المعالج عمله إلى إن ينتهي.

النوع الثاني : إيقافي preemptive

نفس الطريقة السابقة ولكن الفرق هو عند قدوم مهمة ما وكان الوقت اللازم لمعالجتها أقل من التي مع المعالج فإن المعالج يوقف العملية ويضع المهمة التي كانت قيد التنفيذ في **ready queue** ويبدأ معالجة المهمة القادمة الجديدة وهكذا. ويسمى هذا النوع أيضاً **بجدولة أقصر وقت متبقي.(Shortest-Remaining-Time-First (SRTF**

مثال

إذا كانت **P1, P2, P3 and P4** عبارة عن أربعة مهمات قادمات للمعالج وفقاً للبيانات التالية

المهمة	زمن الوصول **Arrive Time**	زمن المعالجة **Burst Time**
P1	**0**	٧
P2	٢	٤
P3	٤	١
P4	٥	٤

إذا كان المعالج يعمل بجدولة أقصر المهمات أولاً، ارسم مخطط **Gantt** ليوضح كيفية تنفيذها إذا كان يعمل وفقاً لطريقة:

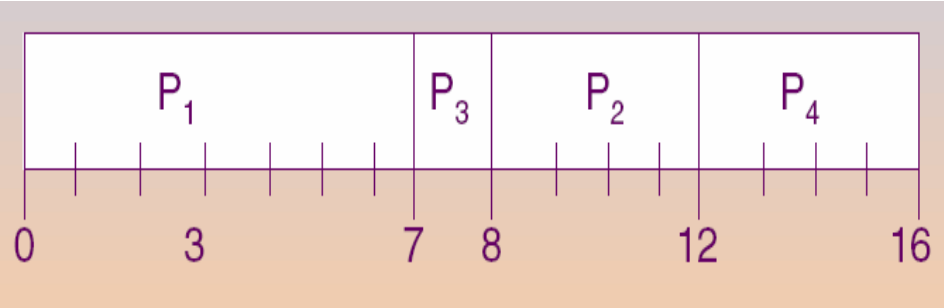
- غير الإيقاف non preemptive .
- الإيقاف preemptive.

ثم احسب متوسط زمن الانتظار **Average waiting time** لكل طريقة.

الحل

أولاً: غير الإيقاف non preemptive

- مخطط **Gantt**

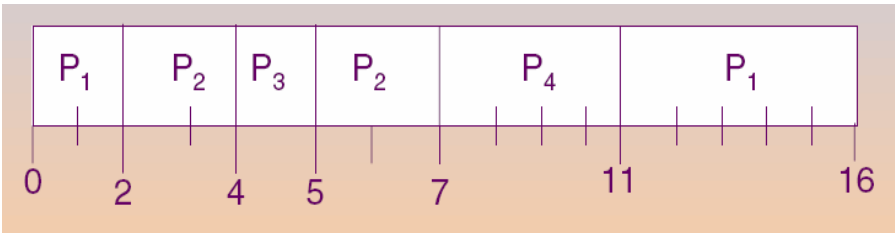


متوسط زمن الانتظار

Average waiting time=(0+6+3+7)/4=4

ثانياً: الإيقاف Preemptive

- مخطط **Gantt**



متوسط زمن الانتظار

Average waiting time=(9+1+0+2)/4=3

هذه الطريقة تقلل متوسط زمن الانتظار، ولكن تظهر مشكلة تسمى المجاعة **Starvation**: وهي تعني أن هنالك مهمة قد لا تنفذ إطلاقاً فمثلاً

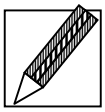
الجدول التالي يبين بعض البيانات

المهمة	زمن الوصول **Arrive Time**	زمن المعالجة **Burst Time**
P1	**0**	٢
P2	٠	١
P3	١	١
P4	٢	١

.	.	.
.	.	.
.	.	.
Pn	**n**	١

من خلال الجدول أعلاه يتضح لنا أن المهمة **P1** تحتاج لوحدتين للتنفيذ وبالتالي تنفذ بعد المهمة **P2** التي تحتاج إلى وحدة واحدة فقط أوبعد مضي وحدة واحدة تظهر مهمة جديدة هي**P3** وتحتاج إلى وحدة زمنية واحدة فقط وبالتالي تنفذ المهمة **P1** بعد المهمة **P3** وهكذا بعض مضي وحدة أخرى تظهر مهمة أخرى تحتاج إلى وحدة زمنية واحدة وهي بالتالي أحق بالتنفيذ من المهمة **iP1** وعليه تصبح المهمة **P1**غير قادرة على التنفيذ إطلاقاً. وهذا ما يعرف بعملية إشباع وحدة المعالجة المركزية **CPU saturation**. وعموماً تعتبر عملية المجاعة من الأشياء غير المحببة لوحدة المعالجة المركزية

تدريب (٥)



إذا كانت **A, B, C and D** عبارة عن أربعة مهمات قادمات للمعالج وفقاً للبيانات التالية

المهمة **Process**	زمن المعالجة **Burst Time**
A	**5**
B	**2**
C	**9**
D	**4**

بافتراض **all processes start running at the same time** إذا كان المعالج يعمل بجدولة أقصر المهمات أولاً **Shortest Job First (SJF)**،

١. ارسم مخطط **Gantt** ليوضح كيفية تنفيذها.
٢. احسب متوسط زمن الانتظار **Average waiting time**.
٣. احسب متوسط الزمن الكلي للتنفيذ **Average Turnaround time**.

ايهما يعتبر مهمة Process وأيهما يعتبر خيطاً Thread؟

٣,٦. التخصيص الدوري

تستخدم طريقة التخصيص الدوري **Round Robin (RR)** مفهوم المشاركة الزمنية حيث تعمل بمبدأ الـ**FIFO**، ولكن تعطى لكل مهمة فترة زمنية معينة(**quantum**) فإذا لم يتم إنجازها في هذه الفترة فإنه يتم ايقافها ليبدأ في تنفيذ المهمة التالية لها في صف الانتظار. وتوضع المهمة التي أنجزت في آخر صف الانتظار.

مثال

إذا كانت **P1, P2, and P3** عبارة عن أربعة مهمات قادمات للمعالج وفقاً للبيانات التالية

المهمة	زمن المعالجة**Burst Time**
P1	**24**
P2	**3**
P3	**3**

إذا كان المعالج يعمل بجدولة طريقة التخصيص الدوري **(RR)**،

ارسم مخطط **Gantt** ليوضح كيفية تنفيذها إذا كان **Quantum = 4**:

الحل

باستخدام **round robin** و **Quantum = 4** يكون تنفيذ المهمات أعلاه كالآتي

P1	**P2**	**P3**	**P1**	**P1**	**P1**	**P1**	**P1**
٠ ٤	٧	١٠	١٤	١٨	٢٢	٢٦	٣٠

مثال

إذا كانت **P1, P2, P3 and P4** عبارة عن أربعة مهمات قادمات للمعالج وفقاً للبيانات التالية

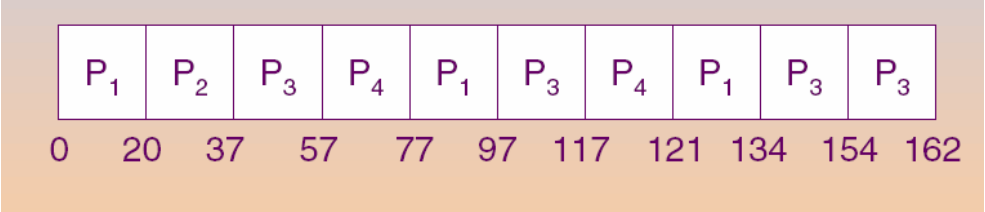
المهمة	زمن المعالجة **Burst Time**
P1	**53**
P2	**17**
P3	**68**
P4	**24**

إذا كان المعالج يعمل بجدولة طريقة التخصيص الدوري **(RR)**،

ارسم مخطط **Gantt** ليوضح كيفية تنفيذها إذا كانت الفترة الزمنية لكل مهمة تساوي **20** وحدة أي أن **Quantum = 20**

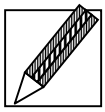
الحل

باستخدام **round robin** و **Quantum = 20** يكون تنفيذ المهمات أعلاه كالآتي



يجب أن تكون الفترة الزمنية الممنوحة لكل مهمه مناسبة لأنها إذاكانت كبيرة لكل المهمات فإن **round robin** تصبح **FIFO**، وإذا كانت قصيرة تكثر عمليات التبديل بين المهمات وهذا يؤثر على كفاءة النظام.

تدريب(٦)



إذا كانت **A, B, C and D** عبارة عن أربعة مهمات قادمات للمعالج وفقاً للبيانات التالية

المهمة **Process**	زمن المعالجة **Burst Time**
A	**5**
B	**2**
C	**9**
D	**4**

بافتراض **all processes start running at the same time** إذا كان المعالج يعمل بجدولة التخصيص الدوري **Round Robin (RR)**،إذا كان

أ- Quantum=3

- ارسم مخطط **Gantt** ليوضح كيفية تنفيذها.
- احسب متوسط زمن الانتظار **Average waiting time**.
- احسب متوسط الزمن الكلي للتنفيذ **Average Turnaround time**.

ب- Quantum=2

- احسب متوسط زمن الانتظار **Average waiting time**.
- احسب متوسط الزمن الكلي للتنفيذ **Average Turnaround time**.

٤,٦. الأولوية

هي خوارزمية لمعالجة نقاط الضعف في خوارزمية **SRTF** التي تنحاز للمهمات الكبيرة (التي تستغرق أطول فترة زمنية)، وتعرف جدولة الأولوية بجدولة أعلى معدل استجابة أيضاً**High Response Ratio Next (HRRN)**

وتتطلب جدولة الأولوية تخصيص مقدار أولوية لكل مهمة، وتصبح المهمة التي ستشغل المعالج هي المهمة ذات الأولوية القصوى.

مثال

إذا كانت **P1, P2, P3, P4 and P5** عبارة عن خمس مهمات قادمات للمعالج وفقاً للبيانات التالية

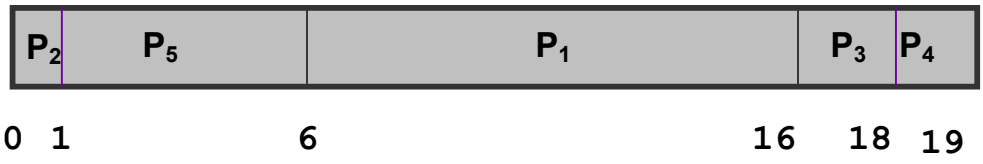
المهمة	الأولوية **Priority**	زمن المعالجة**Burst Time**
P1	٣	١٠
P2	١	١
P3	٤	٢
P4	٥	١
P5	٢	٥

إذا كان المعالج يعمل بجدولة الأولوية،

- ارسم مخطط **Gantt** ليوضح كيفية تنفيذها.
- احسب متوسط زمن الانتظار**Average waiting time**.

الحل

مخطط **Gantt**



زمن الانتظار للمهمات **P1=6, P2=0, P3=16, P4=18 and P5=1** وعليه يكون متوسط زمن الانتظار

Average waiting time=(6+0+16+18+1)/5 = 8.2

مثال

إذا كانت **P1, P2, P3 and P4** عبارة عن أربع مهمات قادمات للمعالج وفقاً للبيانات التالية

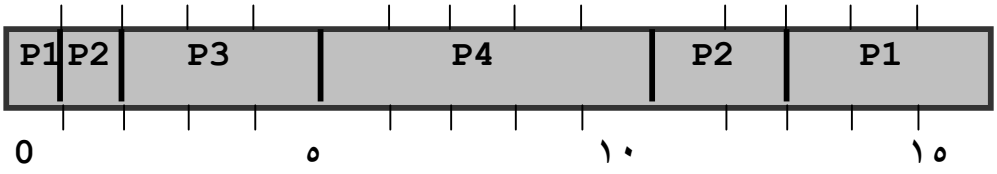
المهمة	زمن الوصول	الأولوية	زمن المعالجة
P1	٠	٣	٤
P2	١	٤	٣
P3	٢	٦	٣
P4	٣	٥	٥

إذا كان المعالج يعمل بجدولة الأولوية، العدد الأكبر للأولوية يملك الأولوية الأعلى ارسم مخطط **Gantt** ليوضح كيفية تنفيذ المهمات، إذا كان يعمل وفقاً لطريقة:

- غير الإيقاف non preemptive
- الإيقاف preemptive

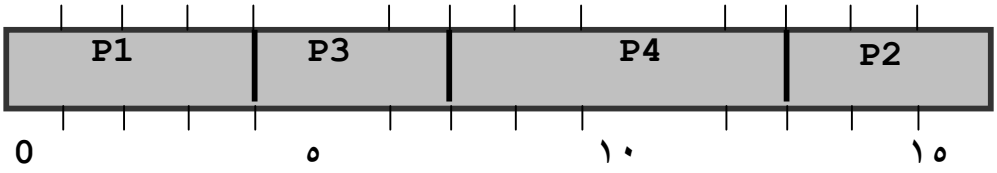
الحل

أولاً: غير الإيقاف non preemptive



تشتغل المهمة **P1** إلى أن تُستكمل وعندما تنتهي تكون المهمات الثلاث الأخرى في انتظار التشغيل وفقاً لأولوياتها **P2, P3 and P4**

ثانياً: الإيقاف preemptive



تشتغل المهمة **P1** لحين وصول المهمة **P2** بعد مرور وحدة زمنية واحدة وعندئذ يتم تنفيذ المهمة **P2** وفقاً لأولوياتها وبعد مرور وحدتين زمنيتين تصل المهمة **P3** ويتم تنفيذها لأنها تمتلك الأولوية الأعلى وهكذا.

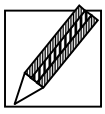
٥,٦. صفوف التغذية الراجعة ذات المستويات المتعددة

كل الخوارزميات السابقة للجدولة لا تنظر إلى خصائص المهمة وإنما تتعامل بطريقة متساوية لكل الوظائف ويعتبر ذلك خللاً واضحاً، لهذا جاءت هذه الخوارزمية لحل هذا الخلل .

نشاط

في رأيك ومن خلال ما درسته (تعاون مع المشرف)
كيف يمكن وصف خوارزمية صفوف التغذية الراجعة ذات المستويات المتعددة.؟

تدريب (٧)



١. إذا كانت A, B, C and D عبارة عن أربعة مهمات قادمات للمعالج وفقاً للبيانات التالية

المهمة	الأولوية	زمن المعالجة Burst Time
A	2	5
B	2	2
C	0	9
D	1	4

بافتراض all processes start running at the same time إذا كان المعالج يعمل بجدولة الأولوية Priority ،إذا كان Quantum=3

- ارسم مخطط Gantt ليوضح كيفية تنفيذها.
- احسب متوسط زمن الانتظار Average waiting time.
- احسب متوسط الزمن الكلي للتنفيذ Average Turnaround time.

أسئلة تقويم ذاتي

١. ما الفرق بين الجدولة الايقافية وغيرالإيقافية ؟
٢. ما المقصود بالإنتاجية Throughput؟
٣. كيف تقاس الأولوية في جدولة SJF.؟

الخلاصة

لقد أوضحنا عزيزي الدارس خلال الوحدة

- إن المهمة هي عبارة عن برنامج قيد التنفيذ له بيانات مدخله وإخراج ومؤشر يصف حالته وبالتالي هنالك ثمة تشابه بين البرنامج والمهمة.
- إن كثيراً من الأحداث المتنوعة تدعو المهمة لتغيير حالاتها، وبالتالي يمكن أن تكون المهمة جديدة، شغالة، مستعدة، متوقفة أو في حالة إنهاء.
- يتم إنشاء كتلة تحكم المهمات **PCB** لكل مهمة تم إنشاؤها وتعتبر وسيلة المتابعة الوحيدة للمهمة من وقت قبولها إلى أن يتم تنفيذها و تخزن **PCB** على سجلات سريعة جدا أسرع كثيرا من سرعة عناصر الذاكرة الأساسية حتى يتمكن نظام التشغيل من متابعة حالات التنفيذ.وتحتوي **PCB** على كل البيانات اللازمة لنظام التشغيل لإدارة المهمة .
- في أنظمة التشغيل القديمة كان لكل مهمة خيط تنفيذ وحيد ولكن في جميع الأنظمة الحديثة تمتلك المهمة عدد من خيوط التنفيذ ويحتوي الخيط على معلومات حول عداداً البرنامج الذي يحدد التعليمة التي تنفذ ويحتوي الخيط أيضا على مسجلات ومكدس لتخزين خطوات التنفيذ السابقة وغيرها من الأشياء.
- سياسة الجدولة والتي من خلالها يتم اختيار المهمة من مجموعة مهمات أخرى مع مراعاة عدة معايير أهمها: استخدام المعالج، الطاقة الإنتاجية، الوقت الكامل، زمن الاستجابة، زمن الانتظار، العدل، و الأولويات. وهنالك عدد من خوارزميات الجدولة تكتب لتوضيح كيفية دخول المهمات إلى وحدة المعالجة المركزية وعادة ما تستخدم العلاقات الرياضية لتحديد زمن الدخول و زمن المعالجة

لمحة مسبقة عن الوحدة التالية

عزيزي الدارس في الوحدة التالية من مقرر "**نظم التشغيل**" ، وهي بعنوان "**الاتصال بين المهمات والمزامنة**".سندرس مفهوم الاتصال بين العمليات، وهو يعني أن تتصل مهمة مع مهمات أخرى حتي لا تقف إحدهما في طريق الأخرى.وسنتعرف على مصطلح السيمافور و مشكلة الاستعصاء، وكيف تحدث، والموارد وأنواعها، بالإضافة إلي كشف الإستعصاء وحلوله.أرجو أن تهتم بدراستها كما عهدناك دائماً.

إجابات التدريبات

تدريب رقم (١)

مكبر الصوت وجهاز الحاسوب والـ **Projector**.

تدريب رقم (٢)

العدد الأقصى للمهمات

لا يوجد حد محدد لعدد المهمات التي يمكن أن تكون في حالة جاهزية أو في حالة توقفi وعلى الأكثر يمكن أن تكون هنالك **N** من المهمات في حالة تشغيل

العدد الأدنى للمهمات

يمكن أن يكون هنالك صفر من المهمات في عدد من الحالات الثلاثi فمثلاً

- ✓ تكون جميع المهمات في حالة توقف أثناء انتظار عمليات دخل/خرجi وعندئذ يكون عدد المهمات في حالة التشغيل وحالة الاستعداد (جاهزة) تساوي صفراً.
- ✓ تكون جميع المهمات في حالة استعداد (جاهزة) أو في حالة تشغيل وعندئذ يكون عدد المهمات في حالة توقف تساوي صفراً.

تدريب رقم (٣)

Program code.	**Process**
Local or temporary data.	**Thread**
Global data.	**Process**
Allocated resources.	**Process**
Execution stack.	**Thread**
Memory management info.	**Process**
Program counter.	**Thread**
Parent identification.	**Process**
Thread state.	**Thread**
Registers.	**Thread**

تدريب رقم (٤)

مخطط **Gantt**

0 5 7 16

----A------ --B-- -----C---------- ----D-----

زمن التحويل من تنفيذ مهمة إلى مهمة أخرى يسمى **contact switch time** وإذا رمزنا له بالرمز **C** (في جميع الأمثلة الواردة في هذه الوحدة فرضنا قيمة **C=0**)

Process	**Waiting Time**	**T_{TRND} Time**
A	**0**	**5**
B	**5 + C**	**7 + C**
C	**7 + 2C**	**16 + 2C**
D	**16 + 3C**	**20 + 3C**
Average	**7 + 1.5C**	**12 + 1.5C**

تدريب رقم (٥)

مخطط **Gantt**

0 2 6 11

--B--- ----D---- -----A---- -------C---------

Process	**Waiting Time**	**T_{TRND} Time**
A	**6 + 2C**	**11 + 2C**
B	**0**	**2**
C	**11 + 3C**	**20 + 3C**
D	**2 + C**	**6 + C**
Average	**4.75 + 1.5C**	**9.75 + 1.5C**

تدريب رقم (٦)

أ- Quantum=3

مخطط Gantt

0	**3**	**5**	**8**	**11**	**13**	**16**	**17**
--A---	**-B--**	**--C---**	**--D---**	**--A--**	**--C---**	**-D-**	**-C---**

زمن التحويل من تنفيذ مهمة إلى مهمة أخرى يسمى **context switching time** وإذا رمزنا له بالرمز **C** (في جميع الأمثلة الواردة في هذه الوحدة فرضنا قيمة **C=0**)

Process	**Waiting Time**	**T_{TRND} Time**
A	**0**	**13 + 4C**
B	**3 + C**	**5 + C**
C	**5 + 2C**	**20 + 7C**
D	**8 + 3C**	**17 + 6C**
Average	**4 + 1.5C**	**12.75 + 4.5C**

ب- Quantum=2

Average Waiting Time = 3 + 1.5C
Average TTRND Time = 13.25 + 6C

تدريب رقم (7)

Quantum=3

We schedule the most important processes first, and do round robin for processes with the same priority level

مخطط Gantt

0	**3**	**6**	**9**	**12**	**13**	**16**	**18**
--C--	**--C--**	**--C--**	**--D--**	**-D-**	**--A--**	**--B--**	**--A--**

زمن التحويل من تنفيذ مهمة إلى مهمة أخرى يسمى **contact switch time** وإذا رمزنا له بالرمز **C** (في جميع الأمثلة الواردة في هذه الوحدة فرضنا قيمة **C=0**)

Process	Waiting Time	T_{TRND} Time
A	**13 + 5C**	**20 + 7C**
B	**16 + 6C**	**18 + 6C**
C	**0**	**9 + 3C**
D	**9 + 3C**	**13 + 4C**
Average	**9.5 + 3.5C**	**15 + 5C**

إجابات أسئلة التقويم الذاتي

تقويم ذاتي رقم (١)

لا تتطلب عمليات الإعاقة خدمات من وحدة المعالجة المركزية **CPU** لأنه لا يمكن متابعة تنفيذها لحين استكمال الحدث.

تقويم ذاتي رقم (٢)

انظر الشكل رقم (٢-٤) الذي يبين أهم الحقول في PCB في نظام تشغيل نموذجي

تقويم ذاتي رقم (٣)

١-**الجدولة غير الإيقافية** هي عند قدوم مهمة جديدة وكان الوقت اللازم لمعالجتها أقل من التي مع المعالج فإنه يتم تجاهلها ويكمل المعالج عمله إلى إن ينتهي، بينما إذا كانت **الجدولة إيقافية** يوقف المعالج العملية ويضع المهمة التي كانت قيد التنفيذ في **ready queue** ويبدأ معالجة المهمة القادمة الجديدة .

٢-**الإنتاجية** هي عدد المهمات التي يكتمل تنفيذها لكل وحده زمنية.

٣- **تقاس الأولوية في جدولة SJF** بأقصر زمن للمعالجة.

مسرد المصطلحات

مهمة	**Process**
مهمة جديدة	**New Process**
شغالة	**Running**
حالة الاستعداد أو الجاهزية	**Ready State**
حالة متوقفة أو معاقة	**Waiting State**
عداد البرنامج	**Program Counter**
خيط	**Thread**
كتلة تحكم المهمات	**Process Control Block (PCB)**
عدد المهمات التي نفذت لكل وحده زمنيه.	**Throughput**
كمية الوقت اللازم لتنفيذ مهمة معينة.	**Turnaround Time**
زمن الانتظار	**Time Waiting**
الزمن الذي يستغرقه المعالج للانتهاء من مهمة.	**Burst Time**
المجاعة	**Starvation**
إيقافي (يحق إخلاؤها)	**Preemptive**
الأولوية	**Priority**
كمية	**Quantum**
الجدولة	**Scheduling**
زمن التحويل	**Context Switching Time**
التغذية الراجعة	**Feed Back**
معدل أعلى استجابة	**High Response Ratio**
إشباع وحدة المعالجة المركزية	**Saturation CPU**

المراجع

[١] ١-يمان اللبني وأسامة العبد الله، **تصميم وتنفيذ نظم التشغيل الحديثة**، شعاع للنشر والعلوم، سوريا-حلب ٢٠٠٥م.

[٢] ٢-ج آرتشر هاريس (ترجمة أمين أيوبي)، **أنظمة تشغيل الحاسوب**، أكاديمياً، بيروت ٢٠٠٢م.

[٣] ٣-عبد الرءوف الحلاق ومنتصر خاطر، **أنظمة التشغيل**، منشورات جامعة القدس المفتوحة، ١٩٩٦م.

[٤] ٤-محمد أحمد فكرين، **نظم تشغيل الحاسبات**، دار المريخ ١٩٩٦م

[5] ٥- Silberschatz, A. and Galvin, P.B., "**Operating System Concepts**", Fifth edition Addison-Wesley, Rea106ding, MA, 199٧.

[6] 6- Tanenbaum, Andrew S., "**Modern Operating Systems**", Prentice-Hall, Englewood Cliffs, NJ, 1992.

[7] http://www.personal.kent.edu/~rmuhamma/OpSystems/os.html

[8] http://www.cag.lcs.mit.edu/~rinard/osnotes/

[9] http://williamstallings.com/OS4e.html


الوحدة الثالثة
الاتصال بين المهمات والمزامنة

محتويات الوحدة

المقدمة

تمهيد

أهلاً بك عزيزي الدارس في الوحدة الثالثة من مقرر "**نظم التشغيل**" الجزء الأول، وهي بعنوان "**الاتصال بين المهمات والمزامنة**".

سندرس في القسم الأول من هذه الوحدة مفهوم الاتصال بين العمليات، وهو يعني أن تتصل مهمة مع مهمات أخرى حتي لا تقف إحدهما في طريق الأخرى ويكون هناك تسلسل منطقي للمهمات.

نتعرف في القسم الثاني على مصطلح السيمافور، وهو متغير بيانات مجرد يستخدم للتحكم في عملية التزامن، ويضبط الوصول إلى المسار الحرج.

أماالقسم الثالث سندرس فيه مشكلة الاستعصاء، وكيف تحدث، والموارد وأنواعها، بالإضافة إلي كشف الإستعصاء وحلوله.

أرجو عزيزي الدارس أن تدرس هذه الوحدة باجتهاد، وتناقش ما درسته مع زملائك و مشرفك الأكاديمي، وتحل الأسئلة الواردة فيها.

أهداف الوحدة

عزيزي الدارس،

بعد فراغك من دراسة هذه الوحدة ينبغي أن تكون قادراً على أن:

- توضح مفهوم الاتصال بين العمليات، ولماذا يحدث.
- تعرف الإقصاء أو الاستثناء المتبادل وكيف يتحقق.
- تعرف السيمافور
- تشرح كيف نتجنب حالات السباق.
- تعرف الموارد وأنواعها وكيفية الحصول عليها.
- تبين كيف تنشأ مشكلة الاستعصاء وكيف نحلها.

١. الاتصال بين العمليات

Interprocceses Communication (IPC)

كما ذكرنا سابقا بأن أحد العمليات التي يمكن أن تتم على المهمة هي تمكين مهمة من الاتصال مع مهمة أو مهمات أخرى وذلك من أجل:

١. تمرير معلومات إلي مهمة أخرى.
٢. التأكد من أن مهمتين أو أكثر لن تقف إحدهما في طريق الأخرى
٣. التأكد من التسلسل المنطقي للمهمات.

اذن لابد من وجود حل لـ:

١. منع أن تقف إحدى المهمات في طريق الأخرى. مثلاً يحاول عدد من المهمات الحصول على مورد محدد أو الحصول على الميغا بايت الاخير في الذاكرة

٢. تأمين التسلسل الصحيح للعمليات مثلا اذا كانت المهمه A تقوم بتوليد اعداد عشوائية ووضعها في مصفوفة، بينما تقوم المهمة B بايجاد فراغ المهمه A أنظر الى البرنامج التالي

S٠: cin>>a >> b>> c>>d;

S١ : x = a + b;

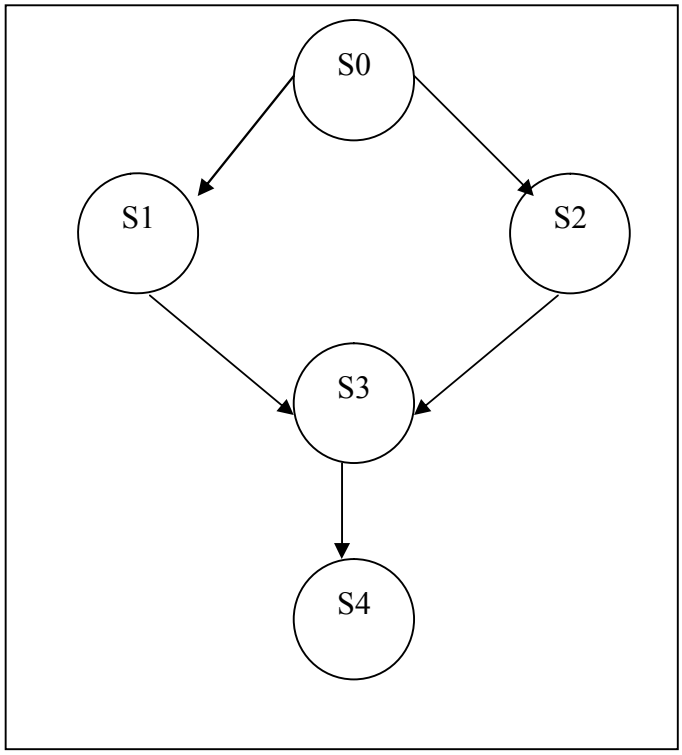
S٢ : y = c + d;

S٣ : z = x + y;

S4 : w = z + 10;

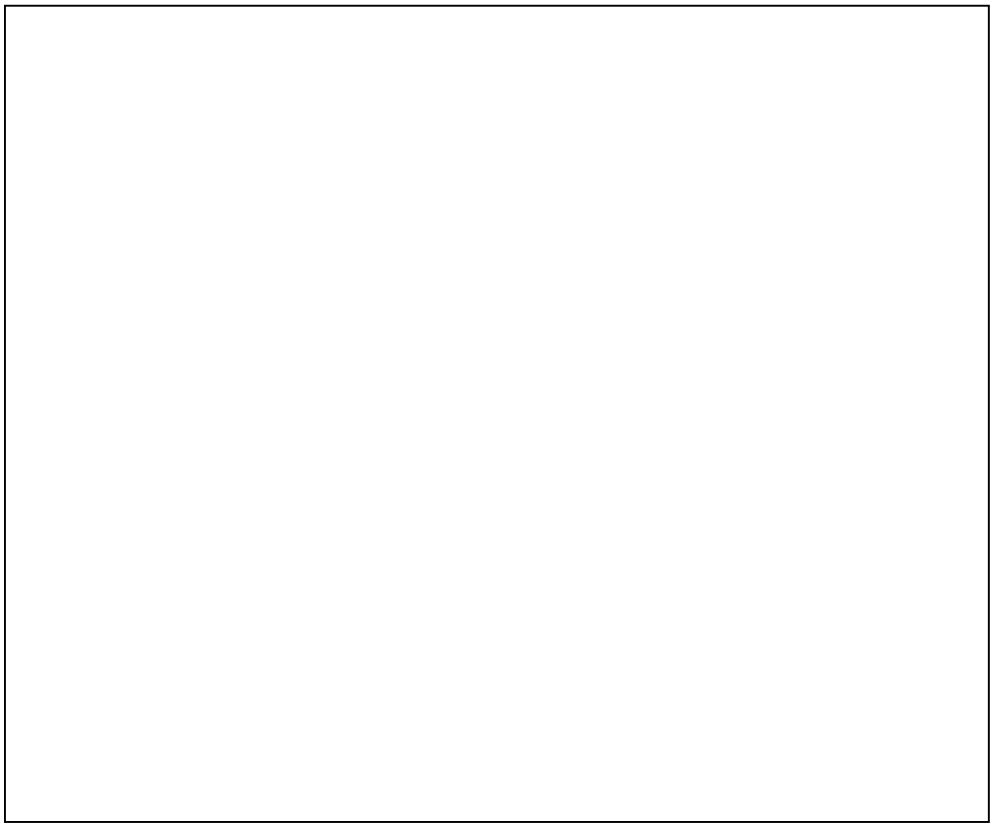
من الواضح أنه لايمكن حساب قيمة z قبل حساب قيمتي x,y، وكذلك w لايمكن أن تحسب قبل إيجاد قيمه z ولكن يمكن تنفيذ الجملة S1 والجملة S2 متزامنتين حيث إنهما لا تعتمدان على بعض.

يمكن تمثيل هذه الأسبقية من المخطط التالي



شكل رقم (٢-٦) مخطط الأسبقية

تتيح عملية الاتصال بين المهمات التشارك في منطقة التخزين سواء أكان على الـذاكرة او الملفات من أجل القراءة أو الكتابة ولكن تبقى النتيجة متعلقة بمن يبدأ العمـل ومتـي وينتج من هذه الحالات مايسمى بحالات السباق (Race condition) وأبسط مثال لـذلك هو عملية إدارة طابعة لعدد من الملفات المراد طباعتها والـوارده مـن عـدة جهـات (مهمات). يتم تبديل المهمات وفقا لزمن محدد تقوم الطابعة بطباعة الملف الموجود فـي مقدمة الصف من خلال المؤشر front بينما يتم إضافة ملف جديد لصف الطباعة من خلال مؤخرة الصف أي المؤشر rear .



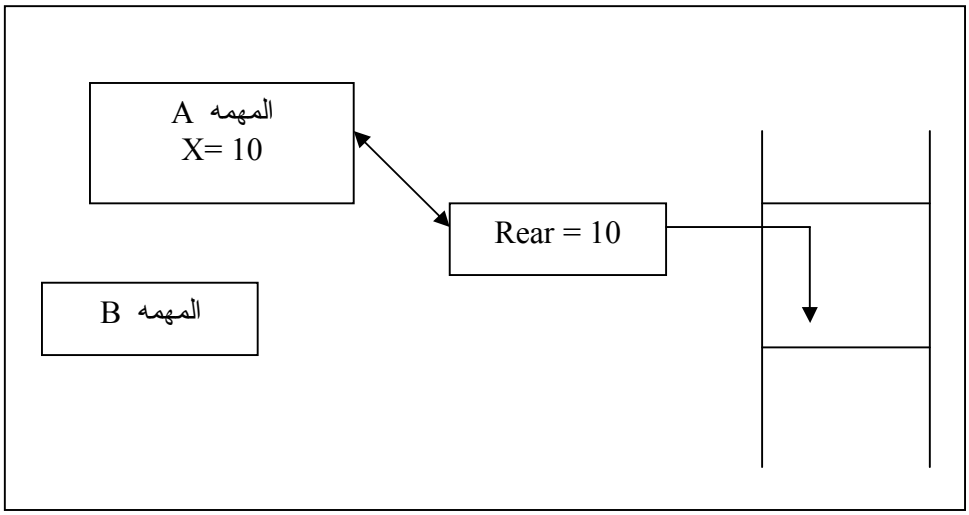
شكل رقم (٢-٧) صف الطباعة (الشكل العام)

عندما تقوم مهمة بارسال ملف للطباعة تقوم بقراءة المؤشر rear وتخزنه في متغير محلي، ثم تقوم بإرسال الملف ومعرفة مكان الملف في الصف.

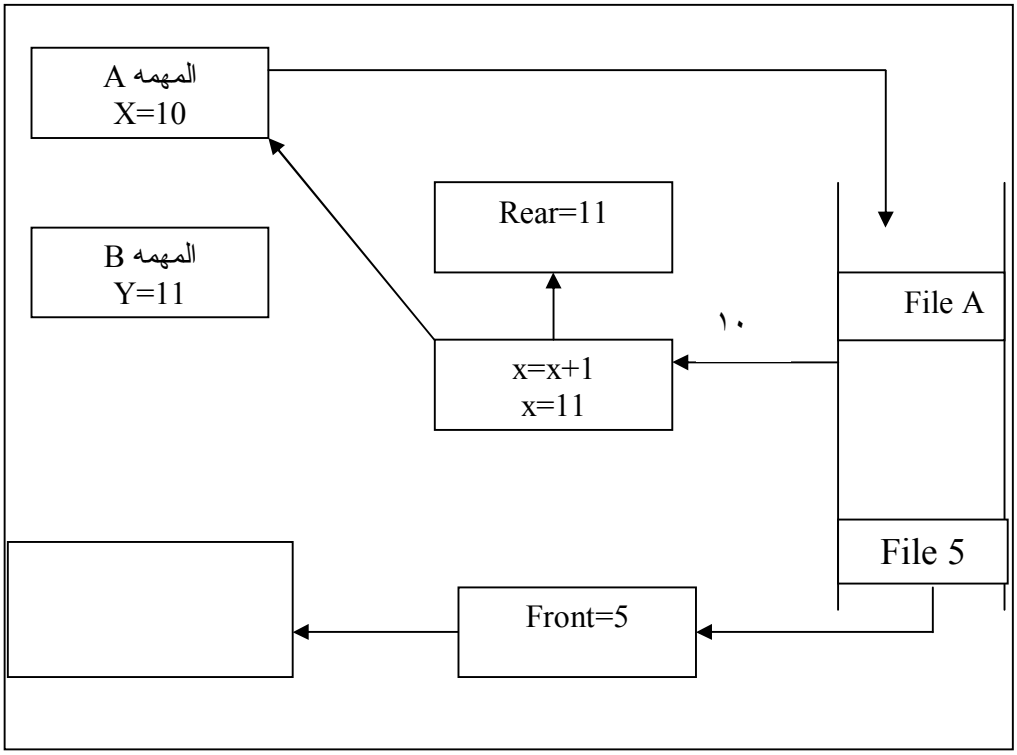
إذا أرادت المهمة A إرسال ملف تقوم بقراءة المؤشر rear الذي يساوي مثلاً ١٠ وتقوم بتخزين في المتغير المحلي الخاص بها مثلاً x، أي ان قيمة x تساوي ١٠، وبعد ذلك مباشرة وقبل انتهاء فترة عمل المهمة A وينتقل لتنفيذ المهمة B

تقوم المهمة B بقراءة المؤشر rear الذي يساوي ١٠ ويتم تخزينه في متغير محلي خاص y ، ثم يتم إرسال الملف إلي الصف .

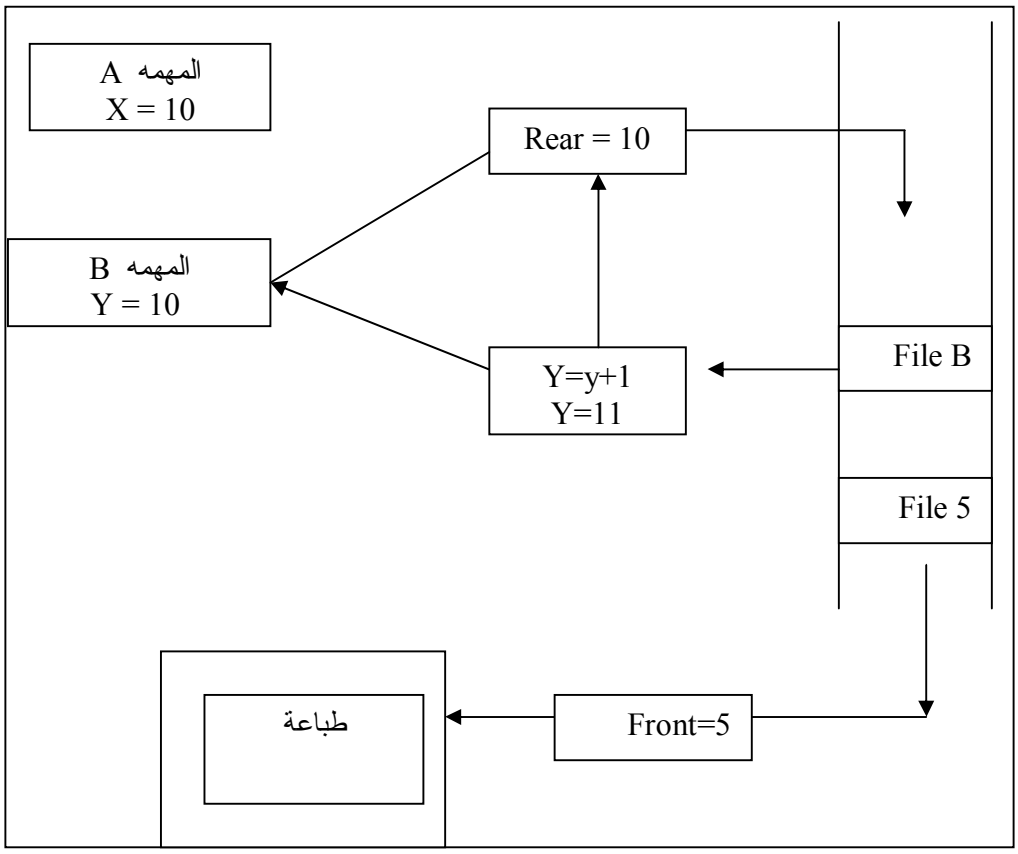
يستقبل برنامج الطباعة الملف ومعه المتغير الخاص بموقعه، مثلا y وهـو ١٠، وبعـد ذلك تزيد قيمة y بواحد، أي تصبح y=11 ويضع هذه القيمه في المؤشر rear
بعد فترة تعود المهمةB الي استئناف العمل بناء على برمجة المعالج وتواصل من نفـس النقطة التي توقفت عندها .وتقوم بارسال الملف ومعه قيمه المتغير x التي تـساوي ١٠ ويقوم برنامج الطباعة بوضع الملف في الموقع ١٠ ويقوم بزيـاده المتغيـر بمقـدار ١ ليصبح ١١ ويضعه في المؤشر rear .
لايلاحظ برنامج الطباعة أي خلل في ذلك، وسوف يكون صف الطباعة في هذه اللحظة متناسقا بعدما تم مسح الملف الخاص بالمهمة B



شكل رقم (٢-٨) المرحلة الأولى من صف الطباعة



شكل رقم(٢-٩) المرحلة الثانية من صف الطباعة



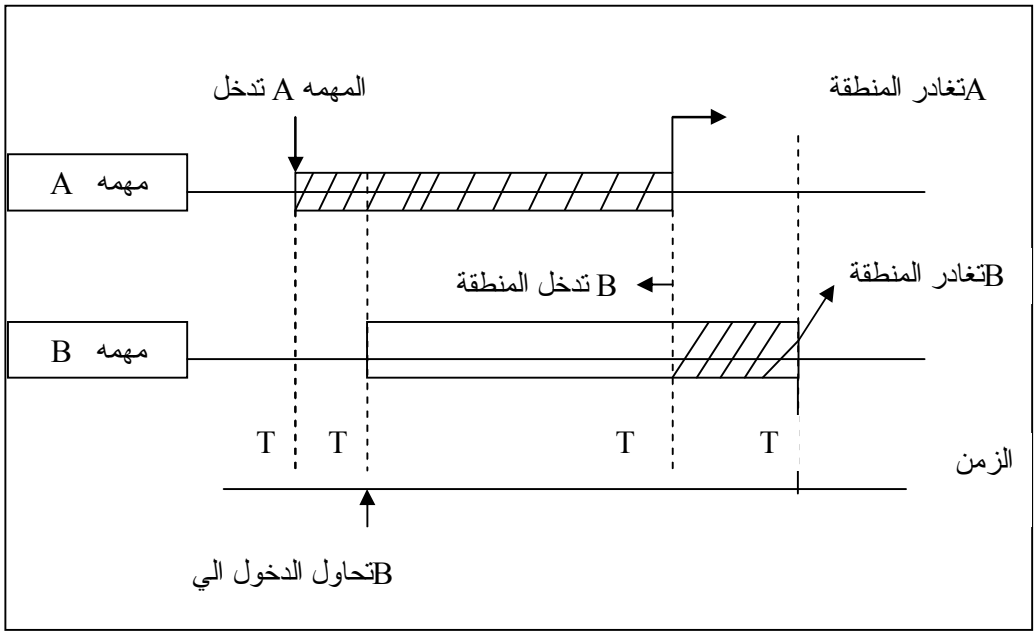
شكل رقم (٢-١٠)المرحلة الثالثة من صف الطباعة

هكذا تم مسح الملف file B ولن تحصل المهمة Bعلى طباعة الملف الخاص بها.

١,١. الإقصاء أو الاستثناء المتبادل Mutual Exclusion

يمكن حل المشكلة السابقة وكافه المشاكل التي تتعلق بالمشاركة في منطقة تخزين واحده بالإقصاء أو الاستثناء المتبادل (mutual exclusion) ويمكن تعريفه على انه الطريقـــة التي تمنع أو تتقصى أي مهمة من العمل إذا كانت هنالك مهمة تعمل على منطقة تخزين

مشتركة تسمى منطقة التخزين المشترك بالمنطقة الحرجة(critical section). والشكل التالي يبين فكرة الاستثناء المتبادل باستخدام المناطق الحرجة



شكل رقم(٢-١١) الاستثناء المتبادل باستخدام المنطقة الحرجة

عموما نحتاج إلى بعض المتطلبات لكي يعمل حل المنطقة الحرجة بصورة صحيحة وبكفاءة عالية، وهذه المتطلبات هي:

١.عدم وجود أي مهمتين في مناطقهما الحرجة في نفس الوقت
٢.لا يمكن افتراض أي فرضيات عن سرعات المهمات بالنسبة لبعضهما البعض، وكذلك عدد المعالجات.
٣.لاتستطيع المهمات التي تعمل خارج منطقتها الحرجة أن تعمل على منع المهمات الاخرى من الدخول إلي مناطقها الحرجة الخاصة بها.
٤.يجب أن تكون هنالك فترة محدده لدخول المهمات الاخرى لدخول مناطقها الحرجة أي لا يوجد انتظار ابدي للمهمة للدخول إلى منطقتها الحرجة.

هنالك عده طرق تحقق عملية الاستثناء المتبادل هي:

الطريقة الأولى : تعطيل المقاطعات

عندما تدخل مهمة منطقتها الحرجة تقوم بتعطيل جميع المقاطعات. وهي بالتـــالي تقـــوم بتعديل الذاكرة المشتركة دون الخوف من دخول أي مهمة أخرى إلي هذه المنطقة، وذلك لأنه معلوم لدينا ان المعالج ينقل التنفيذ من مهمة إلى أخرى من خلال مقاطعة الـــساعة التي تم إيقافها .عندما تخرج المهمة من منطقتها الحرجة تقوم بتفعيل المقاطعات.

نشاط

ناقش مع زملائك إخفاقات الطريقة الأولى.

الطريقة الثانية : الانتظار المشغول busy waiting

جعل المهمات في هذه الطريقة تشترك في متغير واحد يسمي turn الذي يأخذ القيمـــة ٠ أو ١ .اذاكانت turn =i فإن المهمة Ai هي التي تعمل في المنطقة الحرجـــة الخاصـــة بها. الشكل التالي يبين بنية المهمة Ai

```
Do
{
    While (turn != i)   no-opration;
              Critical_section();
             Turn = 1-I;
          Remander_section();
}while(ture);
```

شكل (٢-١٢): شفرة المهمة Ai حسب الطريقة الثانية

يلاحظ أن المهمات تدخل مناطقها الحرجة بالتناوب حيث تـدخل المهمـة Ai المنطقـة الحرجة اولا فإذا كانت المهمه الثانية A_{1-i} مستعده لدخول منطقتها الحرجة فإنها تجبـر على الانتظار، وبعد انتهاء المهمة الأولى Ai يجب أن تدخل المهمة A_{1-I} منطقتها حتى ولوكانت غير مستعده.

الطريقة الثالثة

يتم تعديل مشكلة الطريقة الثانية من خلال استخدام متغير flag[i] صـحيحة اذا كانـت المهمه A_i داخل منطقتها الحرجة، وبعد الخروج من منطقتها تعدل قيمـة flag[i] إلـى false والشكل رقم (٢-١٣)يبين الشفرة الخاصة بالمهمه A_i من خلال هذه الطريقة.

```
Do
{

    Flag[i] = true ;
   While(flag[1-i]) no-opration;
   Critical_section();
    Flag[i] = false;
   Remainder_section();
}while (true);
```

شكل (٢-١٣) شفرة المهمة A_i حسب الطريقة الثالثة

الطريقة الرابعة Peterson

تقوم فكرة (طريقة) Peterson بدمج الطريقة الثانية والثالثة معاً لتحقيق الحل الـصحيح لمشكلة المنطقة الحرجة، وذلك بتحقق الشروط الاربعة الخاصة بالحل الصحيح والأكثـر كفاءه لمشكلة المنطقة الحرجة كما ذكرنا سابقا.

تشترك كل المهمات في متغيرين هما

```
Boolean flag[z];
 Int   turn;
```

حيث تأخذ turn القيمة ٠ أو ١ بينما تأخذ flag[i] و flag[0] القيمة الأوليـة .false . والشكل رقم(٢-١٤) يبين الشفرة الخاصة بالمهمة Ai من خلال طريقة Peterson .

```
Do
   {
        Flag[i] = true;
       Turn = 1-i;
        While(flag[1-i]&&turn=1-i) no-operation;
          Critical_section();
          Flag[i]=false;
        Remainder_section();
 }while(true);
```

شكل (٢-١٤)شفرة المهمه A_i حسب طريقة Peterson

يمكننا ايضا التغلب على مشكلة المنطقة الحرجة من خلال معدات الحاسوب (hardware) اذا تحتوي العديد من المعالجات على التعليمة test and set lock(TSL) حيث يقوم المعالج الذي ينفذ التعليمة TSL بقفل ممر الذاكرة لمنع المعالجات الاخري من الوصول الي الذاكرة إلى ان ينتهي. والشكل التالي يوضح الشفرة البرمجية للتعليمة TSL

```
Boolean  TSL(Boolean  target)
    {
          TSL = target;
         target = true;
     }
```

شكل رقم(٢-١٤)تعريف الدالة TSL

```
Var
       J: 0..n-1;
      Key  Boolean;
Repeat
     Waiting[i]:=true;
     Key:=true;
     While waiting[i] and key do
      Key := TSL(lock);
```

```
    Waiting[i]:=flase;
    Critical_section();
      J:=(i+1)mod n;
   While (j ≠ i) and (not waiting[i]) do
    J:=(j+1)mod n;
   If j=I then lock :=false;
       Else waiting[i] := false;
  Remainder_section();
Until false;
```

شكل رقم(٢-١٥) : الشفرة الخاصة بالمهمه A_i حسب TSL

2. السيمافور Semaphores

Remember:
sleep-waiting

- Semaphores
- Monitors
- Message passing

السيمافور هو متغير بيانات مجرد abstract data type يستخدم للتحكم في عملية التزامن، ويضبط الوصول إلى المسار الحرج، وهو متغير برقم صحيح موجب وهــو الذي تتم فيه عمليتان أساسيتان وهما:

Vو P وهما الحرفان الأوائل من كلمتين من اصل ألماني. وهما بالترتيب:

PROBEREN (to test)
VERTOGEN (to increment)

أي الأولى للاختيار والثانية للزيادة. والبرنامج التالي يوضح ذلك:

```
struct semaphore
    {
        int count;
        processqueue queue;
```

```
        };
void p (semaphore     s)
        {
        if (s.count > 0)
                s.count = s.count – 1;
        else
                s.queue.insert(); // block this process.
        }
void v (semaphore     s)
        {
        if (s.queu.empty() )
                s.count = s.count  + 1;
        else
                s.queue.remove (); // schedule a process, if any, blocked
                on  //‘s’.
}
```

أيضا هاتان العمليتان قد يطلق عليهما أحيانا

{Wait& signal . down &up}.

بالقيمة ١ للسيمافور فإننا نتأكد من وجود استثناء متبادل في الوصول إذا كانتP نفذت قبل الدخول للقسم الحرج وv عند الانتهاء .

عند وضع قيمة السيمافور الابتدائية (n) فان P,V تستخدم لإتاحة n عملية للدخول لقسمها الحرج.

وهناك نوعان من السيمافورات:

- السيمافور الثنائي Binary Semaphore : والسيمافور الثنائي وهو الذي تأخذ فقط العمليتين صفر وواحد.
- السيمافور العام General Semaphore: السيمافور العام هو الذي يأخذ قيمه صحيحة غير سالبة .

بالتالي فان الاستثناء المتبادل Mutual Exclusion يمكن الحصول عليــه بإعطــاء القيمة الابتدائية للسيمافور بواحد (١) . وبتنفيذ العملية p قبل الوصول إلى القسم الحرج في v عند مغادرة المسار الحرج :

```
Shared semaphore  s=1;
//..
P(s);
// Critical section
V(s);
```

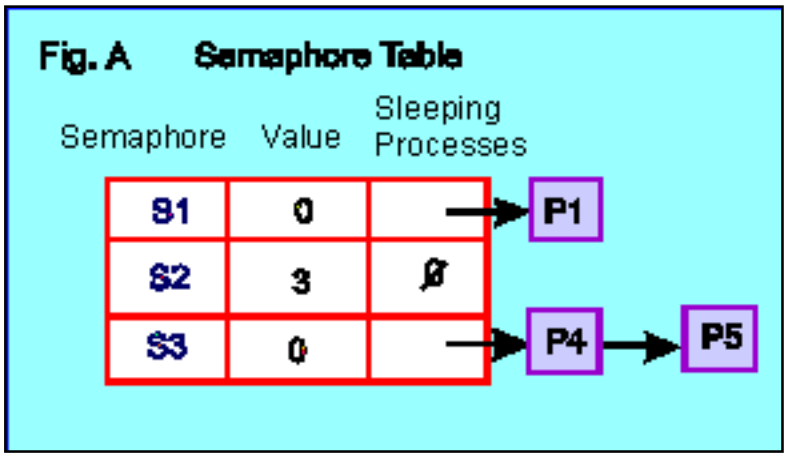
مثال:

لدينا ثلاثة سيمافورات s3, s2, s1

s1 قيمتها صفر . P1 ←——— blocked. في s1.

S2 قيمتها (٣) : هذا يعني أن هناك ثلاث عمليات يمكن أن تنفذ العمليــة down دون أن تنتقل العملية S2 ←—— sleep.

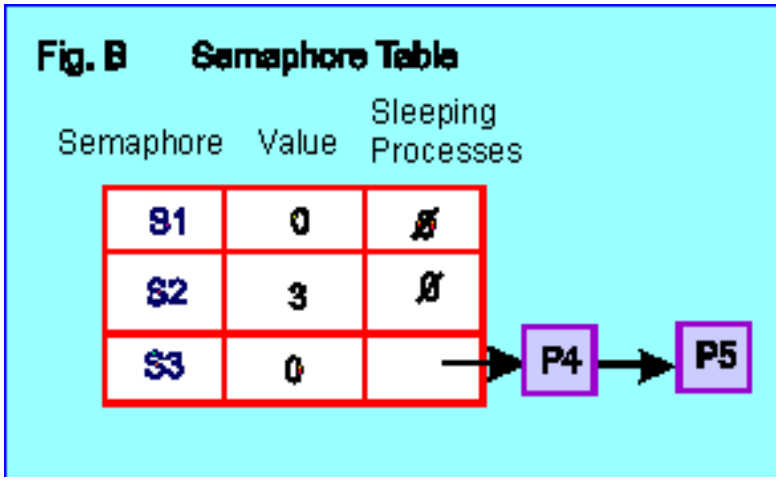
S3 قيمتها صفر : وبها عمليتان P4 , P5 ←——— BLOCK.



تخيل أن الآتي حدث بعد ذلك :

S1 يتم تنفيذ العملية up عليها. وعليه فإن p1 نشطت ثم نفذت العمليــة down علــى S1 نفسها ودخلت مسارها الحرج .

عليه فإن الشكل بعد ذلك سيكون كآلاتي :



المناطق الحرجة والمناطق الحرجة المشروطة:

CRITICAL REGIONS&ADDITIONAL CRITICAL REGIONS

تجمع العمليات فى قطاعات بواسطة برنامج، بعضها تحتاج إلى مصادر مشتركة وتسمى القطاعات الحرجة، وبعضها لايحتاج.

وحتى نتجنب مشكلة الـ RACE CONDITION لابد من آلية لتجنب هذه المشكلة، لذلك نحتاج لخاصية تزامن التنفيذ داخل المناطق الحرجة، وهذه هي الآلية التي يمكن من خلالها التحكم في الوصول للمناطق الحرجة.

١,٢. برنامج المراقبة MONITOR

هو تركيب متزامن يشتمل على عدد من البيانات والأوامر اللازمة لتخصيص أحد أو بعض المصادر المشتركة والتي يمكن استخدامها عدة مرات. وفى حالة احتياج العملية إلى أي من المصادر المشتركة فإنها ترسل إشارات للمراقب للسماح لها بالدخول، ولكن في حالة وجود عملية أخرى داخل المراقب فإنه يجعل العملية طالبة الدخول فى حالة انتظار.

ولكن إذا طلبت إحدى العمليات الدخول إلى المراقب، وكان المصدر المطلـــوب ممنوحا من قبل فإن البرنامج الفرعي التابع للمراقب سيعمل على ترتيب عملية انتظارها خارجا حتى يرجع ذلك المصدر المطلوب.

وبعد أن تنتهي إحدى العمليات من المصدر الممنوح لها فإنها تقوم بمناداة أحد مـــدخلات المراقب لإرجاع المصدر، فيقوم هذا المدخل يتسلم المصدر وينتظر المناداة من عمليـــة أخرى لمنحها هذا المصدر.

ولضمان حصول العمليات المنتظرة للمصادر فان المراقب يعطي أولوية أعلـــى لها عن تلك العمليات التى تأتى حديثا .

وهناك شرط مستقل لكل سبب مميز تحتاجه أي عمليه لتنتظر عليه خارج نظـــام المراقبة وهذا المتغير الشرطي ينشئ صفا خاصا به لتنتظر فيه العمليات ويعمـــل علـــى مبدا (FIFO) المرتبط بنظام الـــ (MONITOR).

تخصيص المصادر البسيطة بواسطة المراقب تتم بواسطة **برامج المراقبة SIMPLE RESOURCE ALLOCATION WITH MONITORS:**

عمليه تنظيم مصادر البيانات بواسطة المراقب تتم بواســـطة المراقـــب، وتـــتم بواسطته عمليتين متعاكستين ، وبهذا تشبه السيمافورات الثنائية إحـــدهما عمليـــة GET RESOURCEوتـــشبه P OPERATION والأخـــرى عمليـــه RETURN RESOURCE وتشبه عمليه R OPERATION ، وبهذا يمكن استخدامه في بنـــاء السيمافورات. وهذا يبين ان قوه المراقب لا تقل عن السيمافورات .

ونلاحظ في البرنامج التالي عملية إعطاء قيم أوليـــة للمتغيـــرات قبـــل أن تبـــدأ العمليات في استخدام المراقب لذلك أعطى المتغيـــر RESOUREEINUSE القيمـــة (FALSE) ليبين أن المصدر جاهز للمنح:

```
MONITOR RESOURCE ALLOATOR;
VAR
RESOURCINUSE: BOOLEAN;
RESOURCEISFREE: CONDITION;
PROCEDURE GETRESOURCE;
```

```
BEGIN
IF RESOURCEINUSE THEN
WAIT RESOURCEISFREE;
RESOURCEINUSE:=TRUE;
END;
PROCEDURE RETURNRECSOURCE;

BEGIN
RESOURCEINUSE:=FALSE;
SIGNAL RESOURCEISFREE;
END;
RESOURCEINUSE:=FALSE;
END;
```

أمثله على المراقب

a) الموقع الحلقي

نظام التشغيل غالبا ما يحدد منطقه تخزين معزولة لإيجاد وسيله للتراسل بين منتج البيانات ومستقبلها. ويتم عمل ذلك بواسطة مصفوفة معينة وحجم معين، حيث يقوم المنتج للبيانات بوضع البيانات في مواقع متتالية في المصفوفة بشكل دائري، ويقوم المستقبل بتفريغ المصفوفة بالترتيب الذي أدخلت به .

b) القارئات والكاتبات

القارئات هي عبارة عن عدد من العمليات تقوم بقراءة البيانات، أما الكاتبات فهي عبارة عن عدد من العمليات تقوم بكتابة البيانات .

بما أن القارئات لا تغير من محتوى البيانات فإنه من الممكن لعدد من القارئات أن تعالج البيانات في وقت واحد .

وهذا على عكس الكاتبات لأنها تقوم بتعديل البيانات لذا لابد أن تقوم بمعالجة البيانات تحت شروط معينة.

القارئات والكاتبات الخاصة بالمراقب تستخدم لتنظيم التحكم بالنظام الكلي لقاعدة البيانات ، وفي كل الحالات فإنه يسمح لواحدةٍ فقط من تلك الكاتبات بأن تقوم بمعالجه البيانات

في وقت معين ، وفي هذه الحالة يأخذ المتغير المنطقي SOMEONEISWRITING القيمة TRUE.

ويستخدم متغير خاص يسمى READERS لعمليات القراءة، و آخر يسمى WRITER، وذلك للتنسيق بين العمليتين، فالأول يشير لعدد عمليات القراءة النشطة والتي تتناقص بمقدار واحد حتى تصل إلى الصفر، وبالتالي فإن واحدة من عمليات الكتابة المنتظرة تصبح نشطة. وتتم العملية بإرفاق أحد المتغيرين مع أحد العمليات (القراءة والكتابة) المنتظرة .

وعندما تبدأ إحدى العمليات بقراءة البيانات فإنها تطلب من المراقب ما يسمى بمعالجه المراقب BEGIN READING لبدء القراءة، وعند انتهائها تطلب من المراقب أيضا ما يدعى بمعالجة المراقب الـ FINISH READING لانتهاء القراءة ويتكرر الشئ نفسه مع عملية القراءة .

٢,٢. التعابير المنطقية Path Expression

هو تركيب يعطي المقدرة على ترتيب عملية تنفيذ البرامج الفرعية ومنها :

١- التعبير المنطقي PATH READ END:-

وهو يشير إلى أن أمر القراءة يستدعي من قبل فعالية متفردة (SINGLE ACTIVITY) ، وعند إنهاء عمليه القراءة قد تستدعى نفس الفعالية أو فعالية أخرى أو القراءة .

٢-التعبير المنطقي : ATH BEGIN READING ,READ ,FINISHED , READING,END

وهذا التعبير يشير إلى أن استدعاء أي من هذه الأوامر لابد أن يتم بالترتيب والتتابع الموضح في التعبير المنطقي .

٣- التعبير المنطقي : PATH (READ) END

يشير إلى أن أمر القراءة قد يتغير من وقت إلى آخر بأي عدد من الفعاليات المختلفة وعندما تنتهي كلها يسمح باستخدام التعبير المنطقي مره أخرى .

٤- التعبير المنطقي :- PATH READ – WRITE END

وهذا التعبير يعني انه يمكن اما لفعاليه ان تقرأ او لفعاليه ان تكتب ولكن ليس معا، والاوليه للقادم أولا .

٣. الاستعصاء DEADLOCK

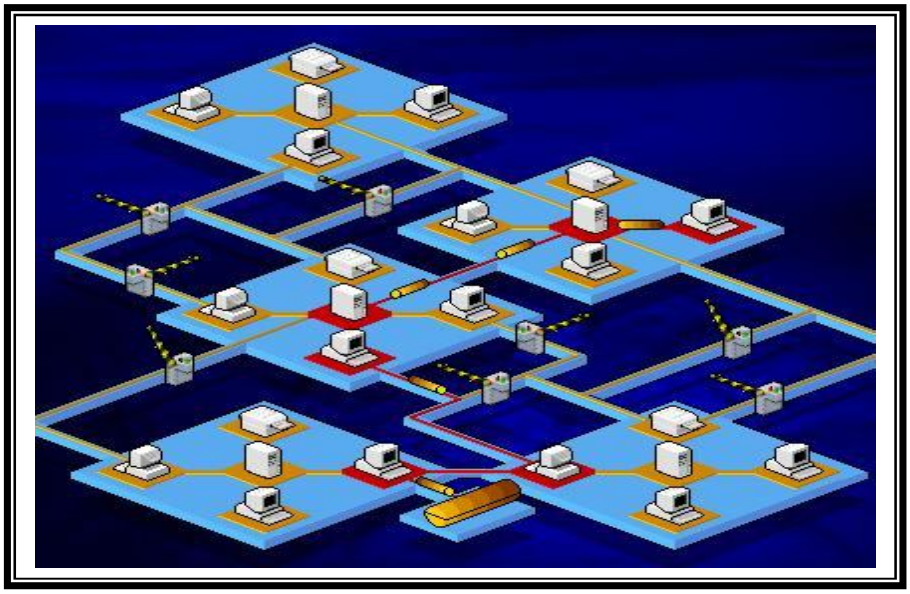
كما ذكرنا سابقاً فإن انظمة الحاسب الآلي تتكون من مجموعه من الموارد (الطابعات ، الأقراص ، ... ، الخ) ، وأنه يمكن لأي عمليه استخدام واحد أو أكثر من تلك الموارد لأنجاز عملها .

فمثلاً اذا كان لدينا عمليتان C & D وتريدان قراءة ملف عبر الماسحه الضوئيه ثم تقومان بعد ذلك بنسخه إلى قرص مدمج .

أولاً تطلب العملية C استخدام الماسحة الضوئية ويتم لها ذلك ، وبعد ذلك تطلب العملية D أستخدام ناسخه الأقراص المدمجه ويتم لها ذلك أيضاً ثم تطلب العمليه C ناسخه الأقراص المدمجه لكن طلبها يرفض حتى تتخلى العملية D عن ناسخة الأقراص المدمجة .

وبنفس الطريقة تطلب العملية D استخدام الماسحه الضوئية ولكن للأسف يرفض طلبها حتى تتخلى العمليه C عن الماسحة الضوئية، وبذلك ستبقي كلتا العمليتين في حالة توقف للأبد، وهذا مايسمى بالاستعصاء DEADLOCK .

ونفس الطريقة يمكن ان يحدث الأستعصاء DEADLOCK على المستوى المادي (الأجهزة) ، فمثلاً اذا كان لدينا مجموعة من الحاسبات متصله مع بعضها البعض عن طريق شبكه محليه وكان لدينا مجموعه من الأجهزة (طابعات ، ماسحات ضوئيه ، ... ، الخ) تستخدم كموارد مشتركه لجميع الحاسبات فيمكن ان يحدث الاستعصاء DEADLOCK حول ثلاثة أو أربعه من تلك الموارد من قبل عدة مستخدمين.



وكما أنه يمكن أن يقع الاستعصاء على مستوي قواعد البيانات. وعلى سبيل المثـــال اذا افترضنا انه يوجد لدينا نظام قواعد بيانات، وكانت هنالك عمليتان تريدان التعـــديل فـــي قاعدة البيانات وبأفتراض ان العمليتين هما: P1 & P2 والسجل المراد التعـــديل فيـــه هو: R2 & R1

فإذا قامت العمليه P1 بقفل السجل R1 لأجراء التعديل فيـــه وفـــي نفـــس الوقت قامت العمليه P2 بقفل السجل R2 لنفس الشئ ، ثم بعد ذلـــك أرادت كلتا العمليتين P1 & P2 قفل سجل الأخرى وفي هذه الحاله ســـوف يحـــدث لدينا ما يسمى بالأستعصاء DEADLOCK .

عموماً يمكن أن يحدث الاستعصاء على كل مستويات الموارد (الماديـــة ، البرمجيـــه) وبذلك يمكن أن نعرف **المورد بإنه أي شئ يمكن أن تستخدمه عملية ما في لحظه مـــن الزمن .**

١,٣. أنواع الموارد

وكما ذكرنا سابقاً فإن المورد هو أي شئ يمكن لعمليه ما استخدامه في لحظه من الزمن ، ومن هذا التعريف يمكننا ان نقسم الموارد إلى نوعين هما:

١. موارد قابله للاستيلاء Preemptable Resource .

٢. موارد غير قابلة للاستبلاء NON Preemptable Resource.

أولاً: الموارد القابله للاستيلاء Preemptable Resource

يمكن أن تعرف الموارد القابلة للأستيلاء بأنها تلك الموارد التي يمكن أن تنتزع من العملية دون أن يسبب ذلك أي مشكله ومن أمثلة الموارد القابله للأستيلاء الذاكرة .

فمثلاً اذا كان لدينا ذاكرة متاحه بحجم 16 MB وكانت هنالك عمليتان P1 & P2 كل منهما تحتاج 16 MB من الذاكرة لطباعة ملف معين على الطابعه .

تقوم أولاً العمليه P1 بعملية الطباعه وذلك بحساب القيم التي تريد طباعتها وقبل ان تنجز عملها يتم أستبدالها بالعمليه P2 وذلك لتجاوزها الزمن المخصص لها ، وتطلب العمليه P2 الطابعه لطباعة ملفها ولكن طلبها يرفض لأن الطابعه في حوزة العملية P1 وظاهرياً يبدو لنا أنه سوف يحدث استعصاء، وذلك لأن العملية P1 تملك الطابعة والعملية P2 تملك الذاكرة، وكل منهما لا تستطيع متابعة عملها دون الحصول على المورد الذي بحوزة الأخرى .

ولكن يمكننا ان نستولي على الذاكرة من العملية P2 واعطاءها للعملية P1 بدون حدوث أي مشكلة وذلك بإدخال العملية P1 إلى الذاكرة مرة أخري لأكمال عملها وتحرر الطابعه وبذلك نكون قد منعنا حدوث الأستعصاء DEADLOCK.

ثانياً: موارد غير قابله للأستيلاء NON Preemptable Resource

وايضاً يمكن ان تعرف الموارد الغير قابله للأستيلاء على انها تلك الموارد التي لا يمكن انتزاعها من صاحبها دون حدوث أي مشكلة ، ومن أمثلة الموارد الغير قابلة للأستيلاء ناسخة الأقراص المدمجة .

فعموما يحدث الأستعصاء DEADLOCK في الموارد غير القابلة للاستيلاء ، أما الأستعصاء الظاهري والذي يحدث في الموارد القابلة للأستيلاء فيمكن حله، وذلك بإعادة ترتيب الموارد بإخذها من عملية واعطائها لعملية أخرى .

٢,٣. الحصول على الموارد

يتم الحصول على المورد وذلك بتنفيذ عدة خطوات وهي

١. طلب العملية للمورد .

٢. استخدام العملية للمورد .

٣. تحرير العملية للمورد .

وبعض أنظمة التشغيل تقوم بإدارة المورد، وذلك بربط كل مورد بمسير Semaphore وتعطي جميع المسيرات القيمه ١ مبدئياً ثم بعد ذلك تقوم بتحقيق الخطوات الثلاثة السابقة مع كل عملية .

فتنفذ الاستدعاء Down على المسير من اجل الحصول على المورد المحدد، ثم تنفيذ الأستدعاء USE لأستخدام المورد، وعند الانتهاء تقوم بتنفيذ الاستدعاء Up من اجل تحرير ذلك المورد .

```
Typedef  Int Semaphore
Semaphore     Resource _ A  ;
Void   Process _ P ( Void ) {
        Down ( & Resource _ A ); // A مناداة الدالة Down لطلب المورد
        Use _Resource _ A ( ); // A مناداة الدالة Use لاستخدام المورد
        Up ( & Resource _ A ); // A مناداة الدالة Up لتحرير المورد
        }
```

١,٢,٣. استخدام مسير لحماية مورد واحد

```
Typedef  Int Semaphore
Semaphore      Resource _ A  ;
Semaphore      Resource _ B  ;
Void   Process _ P ( Void ) {
        مناداة الدالة Down لطلب المورد A // ;( & Resource _ A ) Down
        مناداة الدالة Down لطلب المورد B // ;( & Resource _ B ) Down
        مناداة الدالة Use لاستخدام المورد A // ;( ) Use _Resource _ A
        مناداة الدالة Use لاستخدام المورد B // ;( ) Use _Resource _ B
        مناداة الدالة Up لتحرير المورد A // ;( & Resource _ A ) Up
        مناداة الدالة Up لتحرير المورد B // ;( & Resource _ B ) Up
}
```

أستخدام مسير لحماية موردين

كما لاحظنا في الشكل (b) فغالباً ما تحتاج العملية إلى موردين أو أكثر ولكن ليس هنالك أي مشكلة طالما ان هنالك عملية واحدة ولا توجد تنافس للحصول على تلك الموارد .

أما اذا كان لدينا عمليتان أو أكثر، ونفترض ان لدينا العمليتان P1 & P2 وكان لدينا موردان A & B في هذه الحالة يكون لدينا احتمالان .

أما ان تطلب كل من العمليتين P1 & P2 الموردين بنفس الترتيب دون أي مشاكل وذلك لأن أحدي العمليتان سوف تحصل على المورد الأول قبل الأخري وبنفس الطريقه سوف تحصل على المورد الثاني ، وفي هذه الحاله ستتوقف العملية الأخرى ريثما يصبح المورد المطلوب متوفراً .

```
Typedef  Int Semaphore

Semaphore     Resource _ A  ;
Semaphore     Resource _ B  ;
Void   Process _ P1 ( Void ) {
   Down ( & Resource _ A );
   Down ( & Resource _ B );
   Use _Resource _ A ( );
   Use _Resource _ B ( );
   Up ( & Resource _ A );
   Up ( & Resource _ B );
}
```

```
Typedef  Int Semaphore

Semaphore     Resource _ A  ;
Semaphore     Resource _ B  ;
Void   Process _ P2 ( Void ) {
   Down ( & Resource _ A );
   Down ( & Resource _ B );
   Use _Resource _ A ( );
   Use _Resource _ B ( );
   Up ( & Resource _ A );
   Up ( & Resource _ B );
}
```

(a) طلب العمليتين P1 & P2 الموردين بنفس الترتيب

أما الاحتمال الثاني وهو أن تقومان بطلب الموردين بترتيب مختلف. وهنا سوف تحـــدث مشكلة، وهي ان كل عملية سوف تكون حصلت على مورد واحد وتحتاج للمورد الـــذي بحوزة الأخري مما يسبب اعاقه للعمليتين. فستتوقف كل منهما عندما تحاول الحـــصول على المورد الأخر، وتصبح كل منهما غير قابلة للتنفيذ ابداً ويظهر لـــدينا الاستعـــصاء DEADLOCK .

```
Typedef  Int Semaphore

Semaphore     Resource _ A  ;
Semaphore     Resource _ B  ;
Void   Process _ P1 ( Void ) {
   Down ( & Resource _ A );
   Down ( & Resource _ B );
   Use _Resource _ A ( );
   Use _Resource _ B ( );
   Up ( & Resource _ A );
   Up ( & Resource _ B );
}
```

```
Typedef  Int Semaphore

Semaphore     Resource _ A  ;
Semaphore     Resource _ B  ;
Void   Process _ P2 ( Void ) {
   Down ( & Resource _ B );
   Down ( & Resource _ A );
   Use _Resource _ B ( );
   Use _Resource _ A ( );
   Up ( & Resource _ B );
   Up ( & Resource _ A );
}
```

(b) طلب العمليتان P1 & P2 الموردين بترتيب مختلف

من هذا كله يمكن ان نقول ان هنالك مجموعة من العمليات في حالة أستعصاء اذا كانت كل عملية من هذه المجموعة تحتاج إلى حدوث حدث لا يمكن تحقيقة الا من قبل عمليـــة أخري من هذه المجموعة نفسها .

وفي أغلب الاحيان يكون ذلك الحدث هو عبارة عن تحرير مورد تمتلكه عمليه أخـــري حالياً وبالتالي لا يمكن تحرير أي عمليه مستعصيه ويبقي هذا الوضع مستمراً إلى مـــا لا نهاية .

٣,٣ شروط الاستعصاء

قدم العالم كوفمان عام ١٩٧١ م أربعة شروط أساسية يجب توفرها لكي نقــول إن هــذه العملية مستعصية. وهي على النحو التالي :

١) **شرط منع التبادل**

وهو أن يكون كل مورد مرتبطاً ارتباطاً حصرياً بعملية واحدة .

٢) **شرط الاحتجاز والانتظار**

هو ان تكون كل عمليه قد أمتلكت مورداً معيناً ولكنها تحتــاج لمــورد أخــر لأنجاز عملها .

٣) **شرط عدم الاستيلاء**

وهو ألا يحدث انتزاع للمورد من عمليه معينه قسرياً وانما تقوم العملية نفــسها بعد أكتمال عملها بتحرير ذلك المورد .

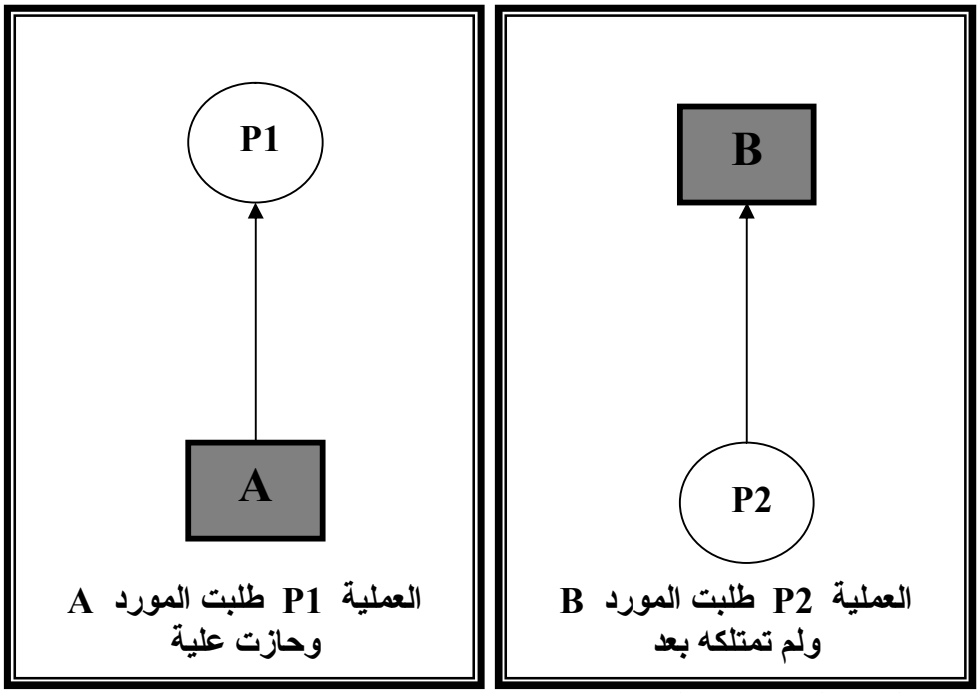
٤) **شرط الأنتظار الدائري**

وهو ان تكون هنالك سلسلة دائرية من العمليات، كل واحــده تحتــاج لمــوردٍ تمتلكه العملية التي تليها في السلسلة .

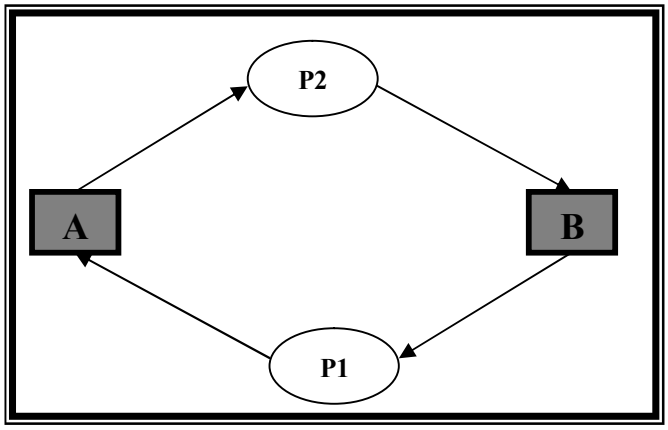
٤,٣ . نمذجة الاستعصاءات

قام العالم هولت عام ١٩٧٢م بشرح كيف يمكننا نمذجة تلك الشروط الأربعــة؟ وذلــك بإستخدام ما يعرف بالغرافات الموجهة . حيث قام بتمثيل المورد بعقــدة تحمــل شــكل المربع بينما مثل العملية بعقده تحمل شكل الدائرة ، أما القوس الذي يبدأ من عقده المورد (المربع) ويتجه إلى عقده العملية (الدائرة) فيعني هذا ان ذلك المورد قد طلبته تلــك العملية وحازت عليه .

وأما القوس الذي يبدأ من عقدة عمليه (دائرة) ويتجه إلى عقدة مورد (مربع) فيعنــي ان العمليه قد طلبت ذلك المورد ولم تمتلكه بعد .



على سبيل المثال يمكن تمثيل استعصاء بسيط حدث بين عمليتين P1 & P2 حيث كانت العملية P1 تحتاج إلى المورد A والذي كان بحوزة العملية P2 ، بينما كانت العملية P2 تحتاج إلى المورد B والذي كان بحوزة العملية P1 .

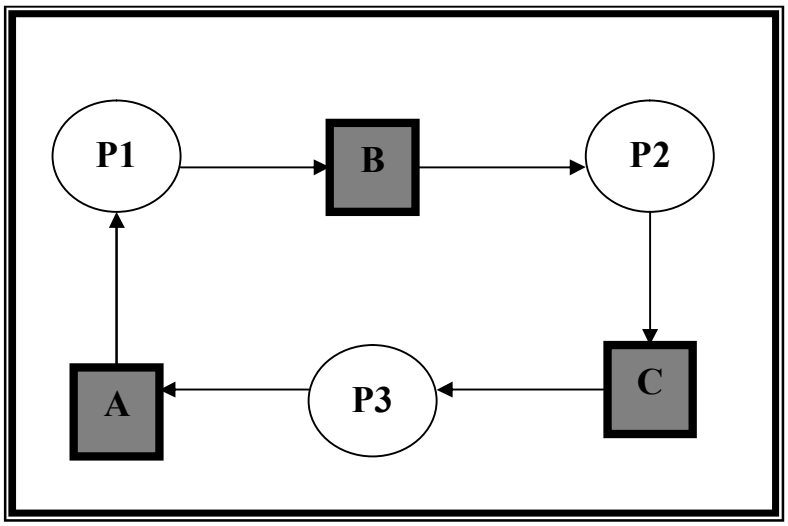


مثال أخر :-

اذا كانت لدينا العمليات P1 & P2 & P3 وكانت لدينا الموارد التالية A & B & C وكانت حالة الموارد المستخدمه حالياً والموارد المطلوبه كما يلي

١) العملية P1 تمتلك المورد A وتحتاج إلى المورد B .
٢) العملية P2 تمتلك المورد B وتحتاج إلى المورد C .
٣) العملية P3 تمتلك المورد C وتحتاج إلى المورد A .
استخدمً الغرافات الموجهة لتمثيل ذلك الأستعصاء .

الحل



والآن وبعد أن تحدثنا عن الاستعصاء وماهي الشروط التي يجب توفرها ليحدث، وكيف تتم نمذجة ذلك، نقوم بعرض الطرق التي تمكننا من التعامل مع الاستعصاءات، وبشكل عام توجد أربعة خطط عمل من أجل التعامل مع الاستعصاءات وهي :-

١/ الخطة الأولى

فكرة هذه الطريقة أن نقوم بتجاهل المشكلة تماماً . ومن أمثلة هذه الطريقة خوارزمية النعامة (دس رأسك في الرمل) وهي أبسط الطرق بحيث تتظاهر بإنه لاتوجد أي مشكلة اطلاقاً (وفي الواقع النعامة لا تقوم بذلك بل أنها تستطيع الركض بسرعه ٦٠ كلم / ساعه بالأضافة إلى أنها يمكنها قتل عدوها برفسه واحده منها) .
ويختلف الناس حول طرق التعامل مع هذه الطريقة ، فمثلاً الرياضيون يجدونها غير مقبولة أطلاقاً ويقولون يجب منع الاستعصاءات مهما كان الثمن .

بينما المهندسون يقولون اذا كانت الأستعصاءات تحدث بمعدل أستعـصاء كـل خمـس سنوات فيما ينهار النظام بسبب أسباب أخري مثل فشل في المكونات أو أخطاء في شفرة النظام بمعدل مره كل أسبوع فإنهم لا يرغبون ببذل جهودهم والتضحيه بالأداء من أجـل أزالة الاستعصاءات .

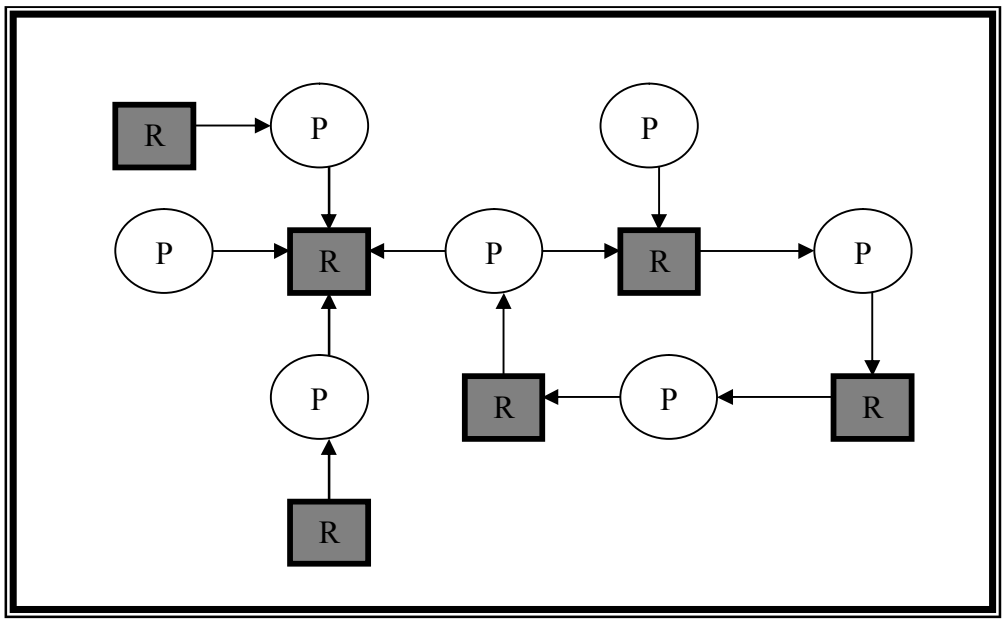
٢/ الخطة الثانية

فكرة هذه الطريقة تتلخص في ترك الاستعصاءات أن تحدث ثم بعد ذلك نقوم بإكتشافها واتخاذ الإجراءات المناسبة لحلها .

٥,٣. كشف الأستعصاءات و حلها

نفترض أن لدينا نظام يحتوي على ماسحة ضوئية واحدة وناسخة أقراص واحدة وراسمة واحدة أي ان هنالك مورد واحد من كل صنف وكان النظام يحتوي على سـبع عمليـات من P1 إلى P7 وسته موارد من R1 إلـى R6 ، وتعطـي حالـة المـوارد المستخدمة حالياً والموارد المطلوبه على النحو التالي :

١) العملية P1 تمتلك R1 لكنها تحتاج R2 .
٢) العملية P2 لا تمتلك شيئاً لكنها تحتاج R3 .
٣) العملية P3 لا تمتلك شيئاً لكنها تحتاج R2 .
٤) العملية P4 تمتلك R4 لكنها تحتاج R2 & R3 .
٥) العملية P5 تمتلك R3 لكنها تحتاج R5 .
٦) العملية P6 تمتلك R6 لكنها تحتاج R2 .
٧) العملية P7 تمتلك R5 لكنها تحتاج R4 .



بالرغم من أننا يمكننا معرفة العمليات المستعصية من خلال غراف بسيط الا أننا نحتـــاج إلى خوارزمية رسمية لكشف الاستعصاءات وذلك لاستخدامها في الأنظمة الفعليه .

كما أنه يوجد العديد من الخوارزميات المعروفة لكشف الحلقات في الغرافات الموجهـــة من بين هذه الخوارزميات سوف ندرس خوارزمية بسيطه والتـــي تقـــوم أولاً بفحـــص الغراف وتنتهي أما عند وجود حلقه أو تبرهن على عدم وجود حلقه ، كما انها تـــستخدم بنيه بيانات عبارة عن قائمة من العقد وتقوم الخوارزمية بتعليم الأقواس الخارجه من كل عقدة في الغراف للدلاله على انها قد تم فحصها وذلك لتجنب الفحص المتكرر ، وتتبـــع الخوارزمية الخطوات التالية :

١. اجعل N هي عقدة البداية (نختار عشوائياً أي عقدة من الغراف ولتكن N) .
٢. اجعل القائمة ولتكن L فارغه .
٣. اجعل جميع الأقواس الخارجه من الغراف غير معلمه .
٤. أضف العقدة N إلى نهاية القائمة L .

٥. اختبر العقدة N هل هي موجودة مرتين في القائمة؟ أذا كانت الاجابة نعم فإن الغراف يحوي حلقة وتنتهي الخوارزمية والا أذهب إلى الخطوة رقم ٦ .

٦. اختبر الأقواس الخارجة من العقدة N هل هنالك أي قوس غير معلم فإذا كانت الأجابة نعم أذهب إلى الخطوة رقم ٧ والا أذهب إلى الخطوة رقم ٨ .

٧. أختر أحد الأقواس عشوائياً وعلمه ثم اجعل العقدة N هي العقدة التي يشير اليها ذلك القوس ثم أذهب إلى الخطوة رقم ٤ .

٨. وصلنا إلى نهاية ميته لذلك سوف نقوم بالرجوع إلى العقدة السابقة للعقدة N ونجعلها N

٩. نختبر العقدة N هل هي عقدة البداية؟ فإذا كانت الاجابة بنعم فإننا قد وصلنا إلى النهاية ولا توجد أي حلقة، والا أذهب إلى ٤ .

والآن سوف نطبق تلك الخوارزمية على المسأله السابقة ، أولاً نقوم بترتيب العقد عشوائياً ولتكن

R1 , P1 , P2 , P3 , R2 , P4 , R3 , P5 , P6 , R4 , R6 , P7 , R5 .

فإذا وجدت الخوارزمية حلقة سوف تنتهي ، والأن سنبدأ خطوات الخوارزمية :

نجعل L قائمة فارغه ثم نختار عقدة البداية ولتكن R1 (العقدة الأولي من جهة اليسار في الترتيب) ، نضيف R1 إلى القائمة L ثم ننتقل إلى العقدة التالية لها (العقدة التي يتجه قوس من R1 اليها) ، وهي P1 ونقوم بإضافتها إلى القائمة فتصبح L = [R1 , P1]

ثم بعد ذلك ننتقل إلى R2 ونضيفها إلى القائمة فتصبح L = [R1 , P1 , R2]

ونلاخظ انه لا يوجد أي قوس خارج منها (R2) فهي نهاية ميته لذلك نعود إلى العقدة P1 وبما انها ليس لها قوس خارج منها أخر سوف نعود إلى R1 وبذلك يكون قد تم فحص العقدة R1 ولا يوجد لديها أي حلقة .

والأن سوف نعيد الخوارزمية بإستخدام العقدة P1 (وهي العقدة الثانية في الترتيب العشوائي الأولى) .

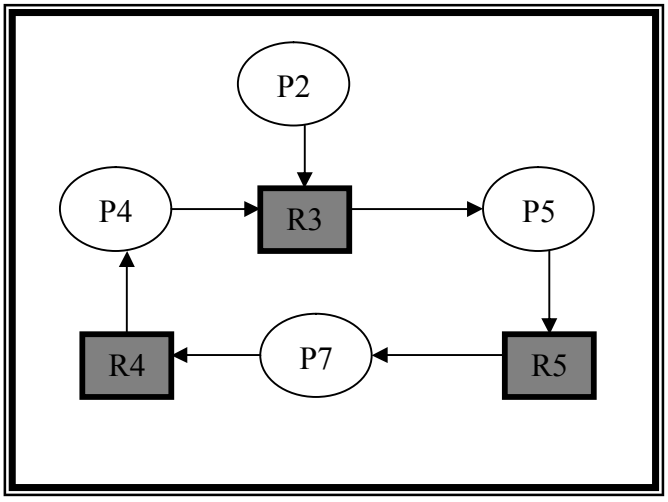
ثم نجعل L فارغه ونضيف P1 إلى القائمة ولكن البحث سوف ينتهي بسرعه لوجود النهاية الميته السابقة ، لذلك سوف نبدأ الخوارزمية من عقدة أخري وهي P2 (العقدة الثالثة في الترتيب) ، ونتابع الأقواس الخارجه من P2 حتي نصل إلى العقدة P4 وعندها تكون القائمة هي :

L = [P2 , R3 , P5 , R5 , P7 , R4 , P4]

ثم نختار أي قوس خارج من P4 عشوائياً ، فإذا قمنا بأختيار القوس المتجه إلى العقدة R2 فسوف نصل إلى نهاية ميته لذلك سوف نعود إلى P4 ونختار القوس المتجه إلى العقدة R3 ، وبذلك تصبح القائمة على النحو التالي :

L = [P2 , R3 , P5 , R5 , P7 , R4 , P4 , R3]

وفي هذه اللحظه نكتشف أن العقدة R3 قد تكررت، وبذلك يكون هنالك حلقه وتنتهي الخوارزمية بإكتشاف الحلقة التالية :



وبعد ما تعرفنا على الاستعصاء (بوجود حلقة في الغراف) سوف نقوم بدراسه عدة حلول لحل ذلك الأستعصاء وهي :

١) التخلص القسري من الأستعصاء

فكرة هذا الحل هو أخذ المورد المسبب للاستعصاء من العمليـــة وإعطائـــه لعملية أخرى دون حدوث أي مشكلة للعملية الأولى، ثم أعادته اليها بعد أنتهـــاء العمليـــه الثانيه منه ، ولكن هذا الحل عادة ما يكون صعباً أو مستحيلاً، وذلك لانه يعتمـــد بـــشكل كبير على طبيعية المورد والعمليات التي يمكن إعادة الموارد إليها بسهولة دون حـــدوث أي مشكلة .

٢) التخلص من الأستعصاء عن طريق التراجع

فكرة هذا الحل أن يكون هنالك نقاط اختبار دورية لكل عملية. وتحتوي تلك النقاط علـــى الموارد المخصصه لهذه العمليه عند كل نقطة.

وبذلك يكون لدينا تسلسل تراكمي من نقاط الاختبار فعند حدوث الاستعصاء يكون مـــن السهل لنا معرفة المورد المسبب، ثم بعد ذلك نقوم بإعادة العملية التي تملك ذلك المـــورد إلى النقطة التي تسبق عمليه حجز المورد ، واذا أرادت العمليه التي تراجعت الحـــصول على المورد مرة أخرى فإنها ستضطر للأنتظار حتى يصبح متاحاً مره أخرى.

٣) التخلص من الاستعصاء عن طريق قتل العملية

فكرة هذه الطريقة تتلخص في قتل إحدى العمليات المشتركة بالحلقة، وبـــذلك يمكن لجميع العمليات الأخرى متابعة عملها ، واذا لم يفد ذلك يمكن أختيار عملية أخرى من بين العمليات المشتركه وقتلها .

لذلك يجب المراعاه عند أختيار العملية المراد قتلها بعناية، بحيث تكون تمتلك المـــوارد التي تحتاجها عملية أو اكثر في الحلقه .

كما يفضل بقدر الإمكان قتل عملية يمكن إعادة تشغيلها مرة أخري دون حدوث أي آثار جانبية ضارة فعلى سبيل المثال يمكن إعادة عملية ترجمة من البداية بعد قتلها لأنه كـــل ما قامت به قبل القتل عباره عن قراءة ملف المصدر وأنتاج ملف الهدف وبـــذلك يكـــون التشغيل الأول ليس له أي علاقة بالتشغيل الثاني .

ولكن لا يمكن قتل عملية تريد إضافة مثلا الرقم ١ إلى سجل معين في قاعدة البيانات، وذلك لأن قتلها ثم تشغيلها مرة أخرى يؤدي إلى إضافة الرقم ٢ إلى السجل المعني (أضافة الرقم ١ قبل قتلها ثم إضافته مرة أخرى عند تشغيلها مرة أخرى) وهي نتيجة غير صحيحة.

٣/ الخطة الثالثة

معظم العمليات تطلب الموارد واحداً تلو الآخر لذلك يجب على النظام معرفة هل منح المورد لعملية معينة لا يسبب أستعصاء أم لا . أي أن هذه الخطه تهدف إلى تجنب الأستعصاء قبل حدوثه ، كما انه هنالك عدة خوارزميات نستطيع من خلالها تجنب الاستعصاءات، وذلك من خلال تخصيص المورد بحرص شديد .

وسوف ندرس الآن واحدة من هذه الخوارزميات وهي :

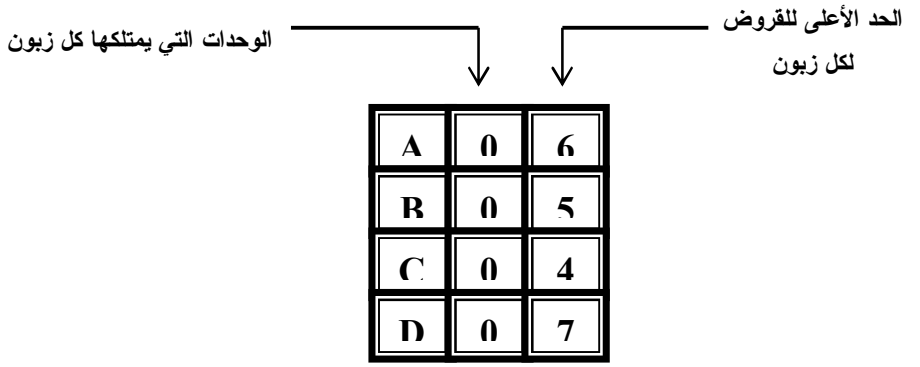
خوارزمية المصرفي Banker

قام العالم ديكاسترا في عام ١٩٦٥م بكتابة خوارزمية نستطيع من خلالها تجنب الاستعصاءات قبل حدوثها. وعرفها بخوارزمية المصرفي Banker وهي في الحقيقة عبارة عن توسيع لخوارزميات أكتشاف الاستعصاءات كما أنها تعتمد في عملها على الطريقه التي يتعامل بها موظف المصرف مع مجموعه من زبائنه الذين يمتلكون حسابات أقتراض في المصرف .

بأفتراض أن لدينا الزبائن A , B , C , D كل منهم لديه عدداً من وحدات الأقتراض كما ان المصرفي يظن ان الزبائن لن يحتاجون للحد الأعلى من قروضهم في دفعه واحدة، لذلك فقد حجز ١٠ وحدات فقط لخدمتهم مع العلم بإن مجموعه قروض الزبائن مجتمعه تساوي ٢٢ وحدة .

نلاحظ أن العالم قام بنمذجة العمليات والموارد المطلوبة والمتاحة ونظام التشغيل، وذلك بتمثيل العمليات بالزبائن، بينما قام بتمثيل الموارد المتاحة والمطلوبة بوحدات القروض، وكما أنه مثل نظام التشغيل الذي يدير كل هذه الموارد والعمليات بالمصرفي .

وكل زبون يقوم بإدارة عمله بشكل خاص، وذلك بطلب بعض القروض من حين إلى أخر. فمثلاً إذا كانت الحاله الابتدائية للمصرفي والزبائن A , B , C , D كما هوموضح في الشكل أدناه



الوحدات المتاحه لدي المصرفي = ١٠

واذا أفترضنا ان في لحظه ما قام الزبائن بطلب الوحدات التالية من المصرفي

$A = 1$, $B = 1$, $C = 2$, $D = 4$

يكون لدينا الشكل التالي :

A	١	6
B	١	5
C	٢	4
D	٤	7

الوحدات المتاحه لدي المصرفي = ٢ وحدة

وهذه الحالة يمكن أعتبارها حالة آمنه لأن لدى المصرفي وحدتان، وبذلك يمكن تجميـــد كل العمليات ما عدا العملية C مما يسمح لـــ C بإتمام عملها وتحرير جميع الوحدات الأربعه كما في الشكل التالي :

A	١	6
B	١	5
C	٠	4
D	٤	7

الوحدات المتاحة لدى المصرفي = ٤ وحدات

وبذلك يكون لدى المصرفي ٤ وحدات متاحة، ويستطيع بعد ذلك إعطاء الزبون D أو B الوحدات اللازمة له لإكمال عمله وهكذا .

أما إذا قمنا بمنح وحدة واحدة أخرى للزبون B وكما موضح في الشكل أدناه

A	١	6
B	٢	5
C	٢	4
D	٤	7

الوحدات المتاحه لدي المصرفي = ١ وحدة

فينتج لدينا حاله غير أمنه وذلك لأن المصرفي لأ يملك سوي وحدة واحدة وهي غير كافيه لتلبيه طلب أحد الزبائن الأربعه بالرغم من أن الزبون يمكن ألاّ يحتاج إلى الحد الأعلى من القروض دفعه واحدة ولكن يجب على المصرفي أن لا يعتمد على ذلك .

فعموماً تقوم خوارزمية المصرفي بدراسة كل طلب لدى وقوعه، وتختبر هل منحه سوف يؤدي إلى حالة أمنه فإذا كان كذلك يتم منح الطلب والأ سوف يؤجل هذا الطلب إلى ما بعد .

وتكون الحالة أمنه اذا كانت لدي المصرفي وحدات كافية لتلبيه أحد الزبائن وعندها يفترض أن هذه الوحدات قد تم سدادها وبعد ذلك يفحص الزبون الأقرب إلى الحد الأعلى من الوحدات وهكذا ، فإذا كانت جميع القروض يمكن سدادها في النهاية نقول إن الحاله أمنه وبعد ذلك يتم منح الطلب الأبتدئي .

٤) الخطة الرابعة

وفي الحقيقة لا يمكن عملياً تجنب الاستعصاءات، وذلك بسبب الحاجة لوجود معلومات كافية لجميع الطلبات المستقبلية والتي لا يمكن التنبؤ بها، لذلك كانت فكرة الخطة الرابعة تتلخص في أننا يمكننا منع الاستعصاءات وذلك بعدم تحقق أحد الشروط الأربعه على الأقل وبذلك يصبح الأستعصاء مستحيلاً من الناحية البنوية.

والآن سوف ندرس كل شرط على حدة ومحاولة عدم تحقيقه وبذلك يكون قد منعنا الاستعصاء .

١. مهاجمة شرط منع التبادل

يمكن مهاجمة الشرط الأول وذلك بعدم تخصيص أي مورد بشكل حصري على عملية واحدة، ولكن هذا يمكن أن يحدث أخطاء. فعلى سبيل المثال إذا قامت عمليتان بالكتابة إلى الطابعه في نفس الوقت سيؤدي إلى طباعة نص غير مفهوم .

ولكن يمكننا أستخدام جدولة الطباعة (Spooling) ، وبذلك يمكن لعدة عمليات من أستخدام الطابعه في نفس الوقت دون حدوث أي مشاكل لأن العمليه الوحيده التي تطلب الطابعه فعلياً هي مشغل الطابعه (Printer Demon) ، وبما أن ذلك المشغل لا

يستخدم أي موارد أخرى سوى الطابعة فإننا بذلك نكون قد تخلصنا من استعصاءات الطابعة، ولكن لا يمكن أستخدام هذه الطريقه مع جميع الأجهزة .

٢. مهاجمة شرط الإمساك والأنتظار

يمكن مهاجمة هذا الشرط بإجبار جميع العمليات على طلب جميع الموارد التي سوف تحتاجها قبل بدء التنفيذ، فإذا كانت كل الموارد متوفرة فسيتم تخصيص كل ما تحتاجه العملية من موارد، وتستطيع بالتالي التنفيذ حتى النهاية. وأما إذا كان أحد الموارد أو أكثر غير متاحٍ فلن يتم تخصيص أي مورد وبذلك سوف تضطر العملية للانتظار .

ولكن هذا الحل لا يمكن تنفيذه عملياً لأنه لا يمكن معرفة الموارد التي تحتاجها كل عملية قبل بداية عملها ، كما أنه اذا كان الأمر كذلك فيمكننا أستخدام خوارزمية المصرفي بالإضافه إلى أنه توجد مشكلة أخرى وهي عدم أستغلال الموارد بشكل أمثل وعلى سبيل المثال اذا كان لدينا عمليه تقرأ البيانات من قرص ثم تحلله لمدة ساعه وبعد ذلك تقوم بكتابة النتائج في قرص أخر وتحتاج في عملها إلى الماسحه فأذا أستوجب طلب كل الموارد مقدماً فإن العملية ستحجز الماسحة لمدة ساعه دون استخدامها .

٣. مهاجمة شرط عدم الاستيلاء

إن مهاجمة هذا الشرط أقل فائدة لأنه إذا أفترضنا أن لدينا عمليه تمتلك الطابعه وكانت في منتصف عملها، ثم قمنا بإخذ الطابعه منها قسرياً وذلك لعدم وجود راسمه لأزمة لهذه العمليه يمكن أن يكون صعباً في أحسن الأحوال أو مستحيلاً في أسوئها .

٤. مهاجمة شرط الانتظار الدائري

يمكننا مهاجمة هذا الشرط وذلك بعدة طرق، ويمكن تمثيل إحدى الطرق ببساطة بوضع القاعدة التاليه:

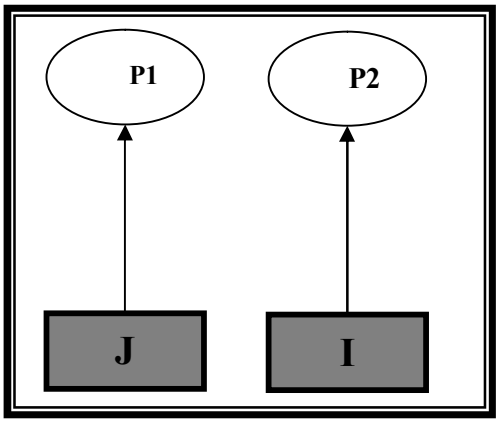
لكل عملية يجب أن تستخدم مورداً واحداً فقط في أي لحظة، فإذا أحتاجت مورداً آخر يجب عليها أن تحرر المورد الأول أولاً.

ويمكن أن تكون هذه الطريقة غير مقبوله في بعض الأحيان. فمثلاً إذا كانت لدينا عمليه تريد نسخ ملف ضخم من شريط إلى الطابعة.

كما أنه توجد طريقة أخري لمهاجمة شرط الأنتظار الدائري وتتمثل في ترقيم جميع الموارد وتستطيع كل عمليه طلب الموارد كما تحلو لها ولكن بشرط أن تكون جميع الطلبات في تسلسل رقمي معين فمثلاً أذا كانت لدينا الموارد بالترقيم التالي:

١. الماسحه . ٢. الطابعه . ٣. سواقه الشريط .

فإن أي عملية يمكنها طلب الطابعة أولاً ثم سواقة الشريط، ولكنها لا تستطيع طلب الطابعه أولاً ثم الماسحه وبذلك لا يمكن أن يحتوي غراف الموارد على أي من الحلقات ، وعلى سبيل المثال أذا كان لدينا العمليتان P2 , P1 وتطلبان الموردين I , J على التوالي وبإفتراض أن I , J هما موردان متمايزان (مختلفين) فأذا كان I > J هذا يعني أن العمليه P2 لا يمكنها طلب المورد J لانه أقل من المورد الذي بحوزتها حالياً ، فأما أذا كان I < J هذا يعني أن العمليه P1 لا يمكنها طلب المورد I لأنه أقل من المورد الذي بحوزتها حالياً ، وفي كلا الحالتين الاستعصاء مستحيلاً .



الخلاصة

عزيزي الدارس لنقم بتلخيص ما درسناه في هذه الوحدة، بدأنا وحدتنا بتعريف مفهوم الاتصال بين المهمات الذي يحدث لعدة أسباب هي:

- تمرير معلومات إلى مهمة أخرى.
- التأكد من أن مهمتين أو أكثر لا تقف إحداهما في طريق الأخرى.
- التأكد من التسلسل المنطقي للمهمات.

وشرحنا كيف تنتج حالات السباق. وهي المشاركة في منطقة التخزين. و وضحنا حلها بالإقصاء أو الإستثناء المتبادل وهو منع أو إقصاء أي مهمة من العمل إذا كانت هناك مهمة تعمل في منطقة تخزين مشتركة. وفهمنا أربع طرق لتحقيق عملية الاستثناء المتبادل.

عرفنا السيمافور بأنه متغير بيانات مجرد يستخدم للتحكم في عملية التزامن وضبط الوصول إلى المسار الحرج. وتتم فيه عمليتان أساسيتان هما P الاختيار و V الزيادة.

ناقشنا كيف نتجنب مشكلة حالات السباق بإستخدم برنامج المراقبة ومثلنا للمراقب بالموقع الحلقي و القارئات والكاتبات، عرفنا التعابير المنطقية وهي تركيب يعطي المقدرة على ترتيب عملية تنفيذ البرامج الفرعية.

أوضحنا كيف تحدث عملية الاستعصاء على كل مستويات الموارد و عرفنا أنواع الموارد وهي موارد قابلية للاستلاء و موارد غير قابلة للاستيلاء، وعرفنا أنه يتم الحصول على المورد بتنفيذ عدة خطوات وهي:

- طلب العملية للمورد.
- استخدام العملية للمورد.
- تحرير العملية للمورد.

شرحنا شروط الاستعصاء الأربعة وهي:

- شرط منع التبادل.
- شرط الاحتجاز و الانتظار.
- شرط عدم الاستيلاء.
- شرط الانتظار الدائري.

وأوضحنا كيف تتم نمذجة الاستعصاءات وبالتالي كشفها وحلها، ولـــدينا أربعـــة خطـــط لحلها شرحناها داخل الوحدة بإسهاب.

أتمنى عزيزي الدارس أن تكون قد أفدت من هذه الوحدة وفهمتها فهمـــاً جيـــداً. فهـــي تحتوي على عدد كبير من المفاهيم الهامة لعمليات الاتصال بين المهمات ومزامنتها.

لمحة مسبقة عن الوحدة التالية

عزيزي الدارس الوحدة التالية فكرة عن إحدى الوظائف الهامة لنظام التشغيل وهو إدارة الذاكرة، و فيها نستعرض أساليب تقسيم الذاكرة، وطرق و استراتيجيات تسكين العمليات المراد تنفيذها، وكيف تتم معالجة القطاعات الفارغة و القطاعات المشغولة .

مسرد المصطلحات

- **حالات السباق Race condition**

هي حالات وجود أكثر من مهمة في منطقة تخزين مشتركة.

- **بالإقصاء أو الاستثناء المتبادل (mutual exclusion)**

هو الطريقة التي تمنع أو تقصي أي مهمة من العمل إذا كانت هنالك مهمة تعمل على منطقة تخزين مشتركة.

- **المنطقة الحرجة critical section**

هي منطقة التخزين المشترك

- **السيمافور Semaphore**

هو متغير بيانات مجرد abstract data type يستخدم للتحكم في عملية التزامن ويضبط الوصول إلى المسار الحرج، وهو متغير برقم صحيح موجب والذي تتم فيه عمليتان أساسيتان وهى:

V و P وهما الحرفان الأوائل من كلمتين من أصل ألماني وهما بالترتيب:

PROBEREN (to test)
VERTOGEN (to increment)

- **السيمافور الثنائي Binary Semaphore**

وهو الذي يأخذ فقط العمليتين صفر وواحد.

- **السيمافور العام General Semaphore**

هو الذي يأخذ قيمة صحيحة غير سالبة .

- **القارئات والكاتبات**

القارئات هي عبارة عن عدد من العمليات تقوم بقراءة البيانات أما الكاتبات فهي عبارة عن عدد من العمليات تقوم بالكتابة البيانات .

- **الاستعصاء DEADLOCK**

إذا قامت العمليه P1 بقفل السجل R1 لإجراء التعديل فيه وفي نفس الوقت قامت العملية P2 بقفل السجل R2 لنفس الشئ ، ثم بعد ذلك أرادت كلتا العمليتان P1 & P2 قفل سجل الأخري وفي هذه الحاله سوف يحدث لدينا ما يسمي بالإستعصاء.

- **الموارد القابله للأستيلاء**

هي الموارد التي يمكن أن تنتزع من العملية دون أن يسبب ذلك أي مشكلة، ومن أمثلة الموارد القابلة للأستيلاء على الذاكرة .

- **الموارد غير القابله للأستيلاء**

هي الموارد التي لا يمكن انتزاعها من صاحبها دون حدوث أي مشكلة ، ومن أمثلة الموارد غير القابلة للأستيلاء ناسخة الأقراص المدمجة .

المراجع

١-يمان اللبني وأسامة العبد الله، تصميم وتنفيذ نظم التشغيل الحديثة، شعاع للنشر والعلوم، سوريا، حلب ٢٠٠٥م.

٢-ج آرتشر هاريس (ترجمة أمين أيوبي)، أنظمة تشغيل الحاسوب، أكاديمياً، بيروت ٢٠٠٢م.

٣-عبد الرءوف الحلاق ومنتصر خاطر، أنظمة التشغيل، منشورات جامعة القدس المفتوحة، ١٩٩٦م.

٤-محمد أحمد فكرين، نظم تشغيل الحاسبات، دار المريخ ١٩٩٦م

٥- Silberschatz, A. and Galvin, P.B., "Operating System Concepts", Fifth edition Addison-Wesley, Reading, MA, 199٧.

6- Tanenbaum, Andrew S., "Modern Operating Systems", Prentice-Hall, Englewood Cliffs, NJ, 1992.

http://www.personal.kent.edu/~rmuhamma/OpSystems/os.html
http://www.cag.lcs.mit.edu/~rinard/osnotes/
http://williamstallings.com/OS4e.html


الوحدةالرابعة
إدارة الذاكرة

محتويات الوحدة

المقدمة

تمهيد

أهلاً بك عزيزي الدارس في الوحدة الرابعة من مقرر "**نظم التشغيل**" الجزء الأول وهي بعنوان "**إدارة الذاكرة**" .

تقدم لك هذه الوحدة عزيزي الدارس فكرة عن أحدى الوظائف الهامة لنظام التشغيل وهو إدارة الذاكرة، و فيها نستعرض أساليب تقسيم الذاكرة، وطرق و استراتيجيات تسكين العمليات المراد تنفيذها، وكيف تتم معالجة القطاعات الفارغة و القطاعات المشغولة، كما سنتعرف على الذاكرة الأفتراضية و كيف تتم تقنية التصفيح، ونعرفك على بعض خوارزميات استبدال الصفحات. أتمنى عزيزي الدارس أن تدرس هذه الوحدة و تناقشها مع زملائك و مشرفك الأكاديمي، كما أنصحك بحل التدريبات وأسئلة التقويم الذاتي الواردة فيها.ومرحباً بك مرة أخرى .

أهداف الوحدة

عزيزي الدارس،

بعد فراغك من دراسة هذه الوحدة وحل جميع الواجبات الواردة فيها من تدريبات وأنشطة، ينبغي أن تكون قادراً على أن:

- تعرف أنواع نظم التشغيل ومميزات وعيوب كل منها.
- تبين طرق و استراتجيات تسكين العمليات المراد تنفيذها .
- تبين طرق مراقبة القطاعات الفارغة و القطاعات المشغولة في الذاكرة.
- تعرف الذاكرة الأفتراضية وسبب ا ستخدامها.
- تشرح تقنية التصفيح.
- تعرف بعض خوارزميات ا ستبدال الصفحات.

إحدى وظائف نظام التشغيل إدارة موارد الحاسب والتي من أهمها الذاكرة الرئيسية وذلك لأنها المكان الوحيد الذي منه تستدعي وحدة المعالجة المركزية إيعازات البرامج والبيانات المراد تنفيذها .

ويسمى الجزء من نظام التشغيل الذي يتولى مهمة إدارة الذاكرة بمدير الذاكرة Memory Manager والذي من أهم مهامه :

١) مراقبة حالة جميع مواقع الذاكرة من حيث الفراغ، وذلك لتسكين العمليات المراد تنفيذها أو الامتلاء من أجل تفريغ المواقع بعد انتهاء العمليات من التنفيذ .
٢) تحديد الطريقة التي من خلالها يتم توزيع المواقع الفارغة للعمليات المراد تنفيذها مع تحديد الأولويات في التسكين .
٣) نقل العمليات التي تم تنفيذها من الذاكرة الرئيسية إلى الذاكرة الثانوية (القرص الصلب) أو العكس .

وإن إدارة وتنظيم الذاكرة الرئيسية يعني في المحل الأول كيف توزع المواقع الفارغه في الذاكرة هل تخصص لعملية واحده أم توزع على عدة عمليات، وإذا اخترنا الخيار الثاني فهل تقسم المواقع بالتساوي للعمليات أم تقسم حسب حاجة العملية ؟ ولإجابة على كل هذه التساؤلات يجب أن نتطرق إلى أنواع نظم التشغيل .

١. أنواع نظم التشغيل

تنقسم نظم التشغيل إلى نوعين

١) نظم التشغيل أحادية البرامج Monoprogramming

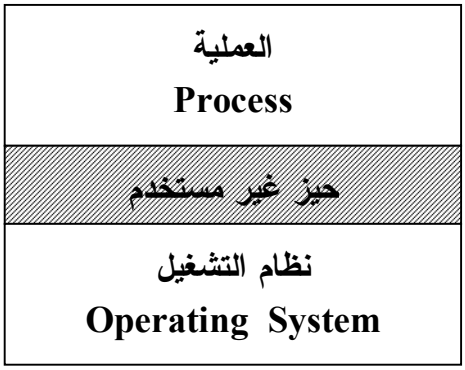
٢) نظم التشغل متعددة البرامج Multiprogramming

١,١. نظم التشغيل أحادية البرامج

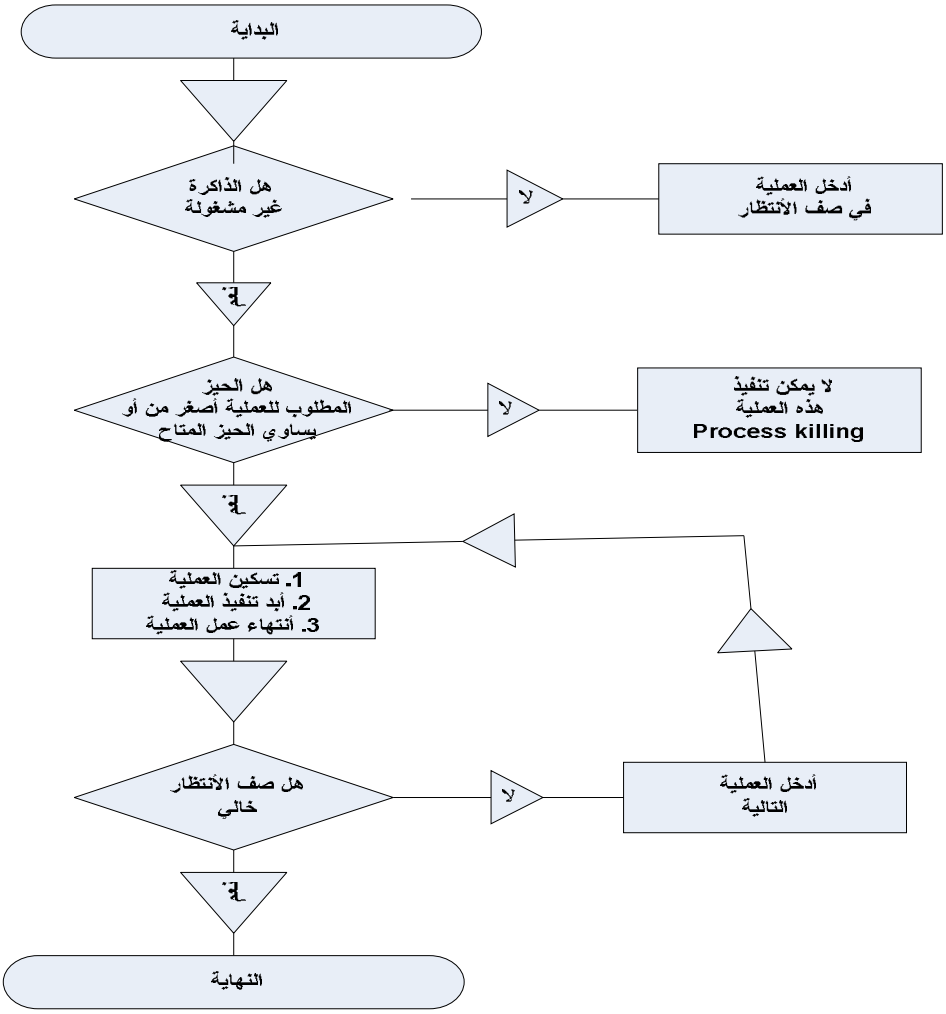
Monoprogramming

يعتبر هذا النظام من أقدم أنظمة التشغيل، وكان يستخدم مع بـــدايات الحاســبات ومازال مستخدماً في بعض المواقع .

وتكون إدارة الذاكرة في هذا النوع عملية بسيطه جداً، وذلك لأن الأسلوب المتبع فـــي هذا النظام هو إتاحة جميع موارد الحاسب لعملية واحدة ، بحيث تكون باقي العمليـــات في حالة خمود حتى تنتهي تلك العملية من التنفيذ ، وذلك يعني أن يتم تسكين عمليـــة واحدة في الذاكرة الرئيسية، بحيث لايزيد حيز العملية عن الحيز المتاح فـــي الـــذاكرة أوعلى هذا الأساس يتم تقسيم الذاكرة الرئيسية إلى ثلاثة أجزاء : جزء خاص بالعملية المراد تنفيذها، وجزء خاص بنظام التشغيل ، وعادة يكون هنالك حيز غير مـــشغول ويعتبر هذا الحيز الجزء الثالث .



رسم يوضح حالة الذاكرة



خوارزمية توضح عمل مدير الذاكرة في أنظمة التشغيل أحادية البرامج

عيوب نظم التشغيل أحادية البرامج

١) عدم استغلال حيز الذاكرة الا ستغلال الأمثل بحيث يكون في أغلب الأحيان هنالك حيز من الذاكرة غير مشغول .
٢) عدم استغلال قدرة وحدة المعالجة المركزية الاستغلال الأمثل، وذلك لأنها تتوقف عن العمل وتتفرغ لمراقبة وحدات الإدخال والإخراج .
٣) طول مدة انتظار العمليات الأخرى، وذلك لأن العملية تكون في حالة أنتظار لدورها

٢,١. نظم التشغل متعددة البرامج Multiprogramming

تعتبر أنظمة التشغيل متعددة البرامج قد عالجت القصور الواضح في أنظمة التشغيل أحادية البرامج، وذلك بدفع عدة عمليات للتنفيذ في نفس الوقت، بحيث تقوم وحدة المعالجة المركزية بالانتقال بينهما مما يؤدي إلى الاستغلالية المثلى لوحدة المعالجة المركزية ولموارد الحاسب .

أسئلة تقويم ذاتي

؟

١/ عرّف أنواع نظم التشغيل.
٢/ اذكر عيوب نظم التشغيل أحادية البرامج.

٢. إدارة الذاكرة

وبما أن هنالك عدة عمليات يتم تنفيذها في نفس الوقت تجدأن هذا يتطلب تقسيم الـذاكرة إلى عدة قطاعات لكي يتم تسكين العمليات المراد تنفيذها أوهنالك نوعين مـن أسـاليب تقسيم الذاكرة، وهما على النحو التالي :نوعان

٢,١. التقسيم الثابت Fixed Partition

وهو تقسيم الذاكرة إلى قطاعات ثابتة محددة علـى أن يخـصص لكـل عمليـه القطاع المناسب لها. ولكن يمكن أن يكون هنالك مجموعة من القطاعات غير مـشغولة، وذلك لعدم وجود عملية بنفس الحجم. بالرغم من أنه يوجد مجموعه من العمليـات فـي صفوف الأنتظار . وأيضاً يمكن أن يتم تسكين عملية في قطاع أكبر من حجمهـا لعـدم وجود الحجم المناسب لها، وبذلك يكون هنالك مساحات غير مشغولة داخل القطاع، وهذا مايسمى بالفراغ الداخلي Internal Fragmentation .

والحل الوحيد لهذه المشكلة هو أن تصتف جميع العمليات في صف واحد حسب أولويتها أو حسب قدومها (FIFO) First In First Out ، وبعد ذلك يقـوم مـدير الـذاكرة بمسح جميع القطاعات غيرالمشغولة ثم يختار أنسب قطاع لهذه العملية من حيث الحجم.

ويمكن أن تكون هنالك عملية ذات حجم صغير بحيث لايوجد قطاع مناسـب لحجمهـا، وفي هذه الحالة يقوم مدير الذاكرة بتخطي تلك العملية إلى عملية أخرى.

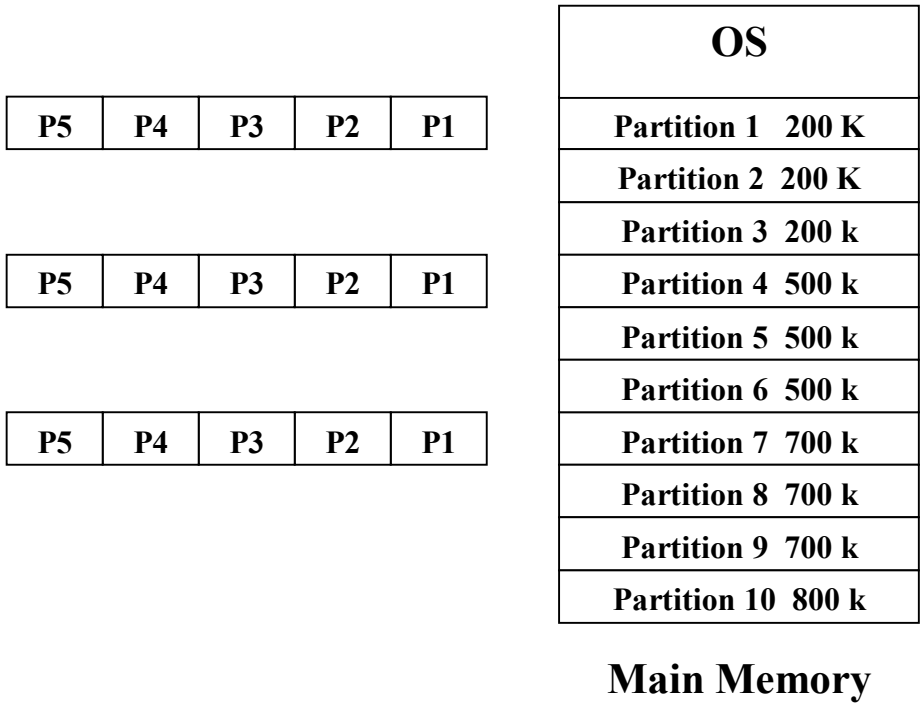
ثم العودة لها مرة أخرى. وهذه المشكلة تسمى بالانتظار الأبدي للعملية (انتظار العملية فترة طويلة لحين تنفيذها) .

ولحل هذه المشكلة يجب أن يوضع حد معين لتجاوز أي عملية لعملية أخري وذلك يعني أ لاّ يتم تجاوز عملية أكثر من n مرة .

وكما هو معلوم فإن العملية تنتج عملية أخرى، وأنها أيضاً يمكن أن تحتاج إلى بيانـات أثناء التنفيذ، وبالتالي فإن حجم العملية غير ثابت. فماذا نفعل إذا زاد حجم العملية عـن حجم القطاع الذي توجد به ؟

هنالك خياران لتفادي تلك المشكلة، وهما على النحو التالي :

- ✓ الخيار الأول وهو عن تسكين العمليةوفيه يجب مراعاة أن حجم العملية غير ثابت، أي تسكين العملية في قطاع أكبر بقليل من حجمها .
- ✓ الخيار الثاني هو إذا زاد حجم العملية عن حجم القطاع الخاص بها يتم البحث عن قطاع آخر مناسب، فإذا لم نجد نقوم بنقل العملية من الذاكرة الرئيسية إلى الذاكرة الثانوية واذا لم نجد أيضاً مكان مناسب في القرص الصلب تلجأ بعض أنظمة التشغيل الي مايسمى بقتل العملية Process Killing وذلك لتفادي حدوث مايسمى الجمود Deadlock للنظام ككل



نظام عدة صفوف

OS
Partition 1 200 K
Partition 2 200 K
Partition 3 200 k
Partition 4 500 k
Partition 5 500 k
Partition 6 500 k
Partition 7 700 k
Partition 8 700 k
Partition 9 700 k
Partition 10 800 k

P5	P4	P3	P2	P1

Main Memory

نظام الصف الواحد

وأبرز عيوب هذا التقسيم عدم كفاءة استغلال الذاكرة في الحالات التالية :

١) وجود قطاع غير مشغول لعدم وجود عملية تنتظر التنفيذ .

٢) وجود عمليات مؤجلة لعدم وجود الحجم المناسب لها .

٣) وجود فراغات موزعة غير مشغولة لاقيمة لها من حيث توافقها مع أحجام العمليات المنتظره .

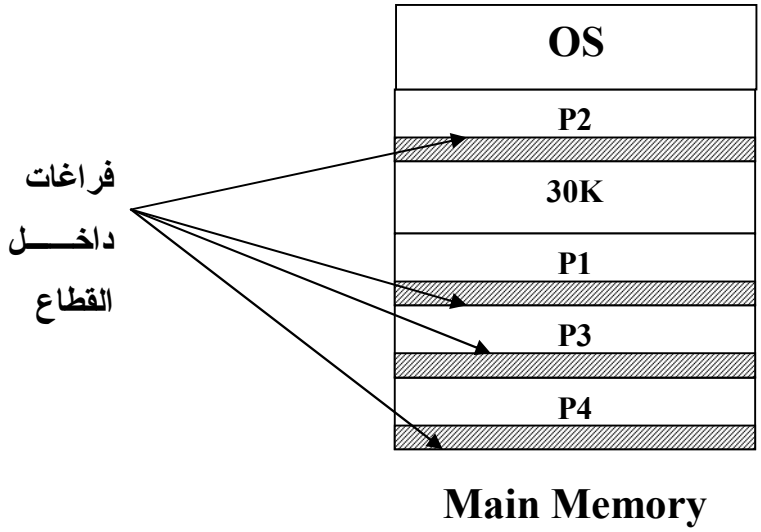
مثال :

إذا كان لدينا العمليات التالية P1 , P2 , P3 , P4 وكانت تلك العمليات تحتاج إلى المساحات 80K , 20K , 120K , 150K على التوالي ، المطلوب بالرسم وضح

الكيفية التي يقوم بها مدير الذاكرة بتسكين تلك العمليات علماً بأن الحجم الكلـــي للـــذاكرة الرئيسية 500K مقسمة إلى القطاعات التالية :-

200K , 140K , 50K , 50K , 30K , 30K

الحل :-



رسم يوضح حالة الذاكرة بعد تسكين العمليات

و ّ يأتي

السؤال ماهي الطريق التي يستخدمها مدير الذاكرة لتحديد القطاع المعين لتسكين العمليات ؟

هنالك عدة استرتيجيات يستخدمها مدير الذاكرة لتحديد القطاع المعين لتسكين العملية ونلخصها في الاتي :

١) **الاستراتيجية الأول**

أستراتيجية المكان الأول First Fit

تعمل هذه الأستراتيجية على مبدأ تسكين العملية في أول قطاع غير مـــشغول في الذاكرة يسع العملية بغض النظر عن حجم القطاع .

٢) **الاستراتيجية الثانية**

استراتيجية المكان التالي Second Fit OR Next Fit

تعمل هذه الاستراتيجية على مبدأ تسكين العملية في أول قطاع غير مشغول يسع العملية بصرف النظر عن حجم القطاع، ويكون هذا البحث بعد آخر قطاع تم التسكين فيه .

٣) **الاستراتيجية الثالثة**

استراتيجية المكان الأحسن Best Fit

تعمل هذه الاستراتيجية على مبدأ تسكين العملية في أنسب قطاع غير مشغول في الذاكرة من حيث الحجم .

٤) **الاستراتيجية الرابعة**

استراتيجية المكان الأكبر Worst Fit

تعمل هذه الاستراتيجية على مبدأ تسكين العملية في أكبر قطاع غير مشغول في الذاكرة بصرف النظر عن حجم العملية .

٥) **الاستراتيجية الخامسة**

استراتيجية Quick Fit

تعمل هذه الاستراتيجية عل مبدأ الاستراتيجية الثالثة! الفرق بينهما وهو أن في هذه الاستراتيجية يقوم مدير الذاكرة بإعداد جدول به جميع عناوين القطاعات غير المشغولة، وبذلك تكون عمليه البحث عمليه أسهل بكثير .

والسؤال الان ما الاستراتيجية الأفضل في تسكين العمليات ؟

تتوقف الاجابة عن هذا السؤال على عدة اعتبارات وهي :

أ) باعتبار الرغبة في تحديد القطاع المناسب للتسكين .

ب) باعتبار الرغبة في تسكين العملية بسرعة .

فإذا كانت باعتبار الرغبة في تحديد القطاع المناسب للتسكين فإن استراتيجية المكان الأفضل Best Fit هي الأفضل والأنسب .

أما إذا كانت الرغبة في تسكين العملية بسرعة فإن أفضل استراتيجية هي استراتيجية المكان الأول First Fit

مثال :

إذا كان لدينا العمليات التالية P1 , P2 , P3 , P4 , P5 وكانت تلك العمليات تحتاج إلى المساحات 100K , 70K , 150K , 50K , 30K على التوالي ، المطلوب توزيع العمليات على القطاعات وفق الاستراتيجيات السابقة علماً بأن حجم الذاكرة الكلي 930K مقسمه على النحو التالي:

الحل

١) باستخدام الاستراتيجية الأولى First Fit

OS
Process 1
Process 4
Process 2
Process 5
Process 3
Partition 6 30 k
Partition 7 80 k
Partition 8 180 k

Main Memory

٢) باستخدام الاستراتيجية الثانية Next Fit

OS
Process 1
Partition 2 60 K
Process 2
Partition 4 100 k
Process 3
Partition 6 30 k
Process 4
Process 5

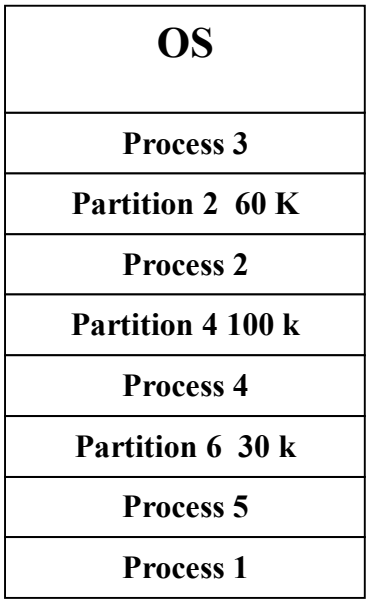
Main Memory

٣) باستخدام]

OS
Partition 1 160 K
Process 4
Partition 3 170 k
Process 1
Process 3
Process 5
Process 2
Partition 8 180 k

Main Memory

٤) باستخدام الاستراتيجية الرابعة Worst Fit



Main Memory

٢,٢. التقسيم الديناميكي للذاكرة Variable Partition

وهي تقسيم الذاكرة إلى قطاعات حسب حجم العمليات، بحيث تعطي كل عمليه قطاعاً أكبر بقليل من المساحة المطلوبة وذلك تجنباً لنمو العملية ، لأن العملية أثناء تنفيذها يمكن أن تحتاج إلى بيانات أو يمكن أن تنتج من العملية عملية أخرى .

وكذلك توجد في هذه الطريقة مشكلة الفراغات حيث توجد فراغات بين القطاعات المشغولة وتسمى هذه الفراغات بالفراغات الخارجية External Fragmentation ولكن هذه الفراغات تكون موجودة لفترة بسيطة لأن عند أنتهاء العملية من التنفيذ يقوم مدير الذاكرة بدمج هذا الفراغ مع الجزء الذي كانت تشغله العملية .

وأبرز نظم التشغيل التي تستخدم هذه الطريقة هو نظام تشغيل IBM المعروف باسم OS / MTV

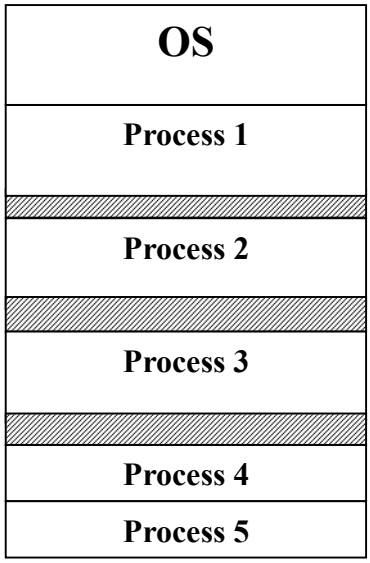
مثال :

يوضح الشكل التالي قائمة العمليات وزمن تشغيلها ، المطلوب رسم حالة الذاكرة عن انتهاء كل عملية من التنفيذ مستخدماً في تسكين العمليات طريقة التقسيم الديناميكي للذاكرة .

العملية	المساحة	زمن المعالجة
Process 1	60 K	10 M/Sec
Process 2	100 K	5 M/Sec
Process 3	30 K	15 M/Sec
Process 4	70 K	8 M/Sec
Process 5	50 K	20 M/Sec

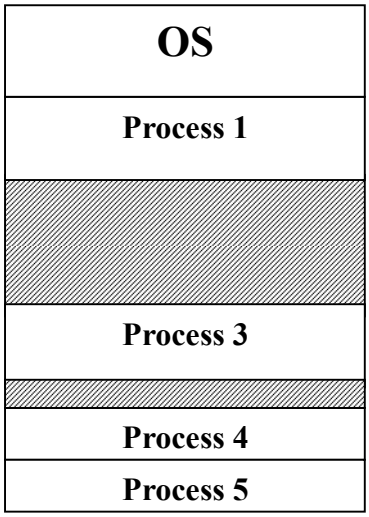
الحل

١) حالة الذاكرة عن تسكين العمليات



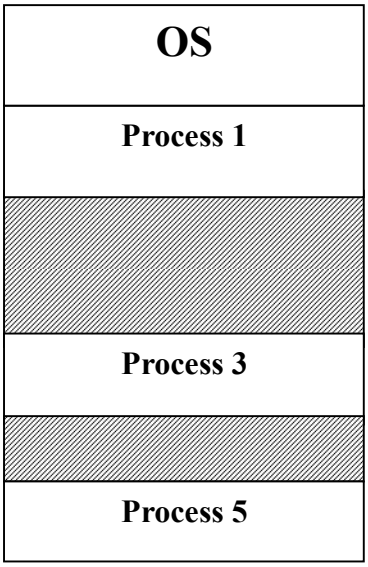
Main Memory

٢) حالة الذاكرة بعد انتهاء العملية P2



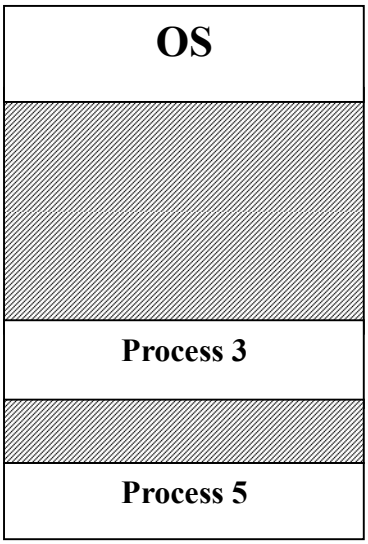
Main Memory

٣) حالة الذاكرة بعد انتهاء العملية P4



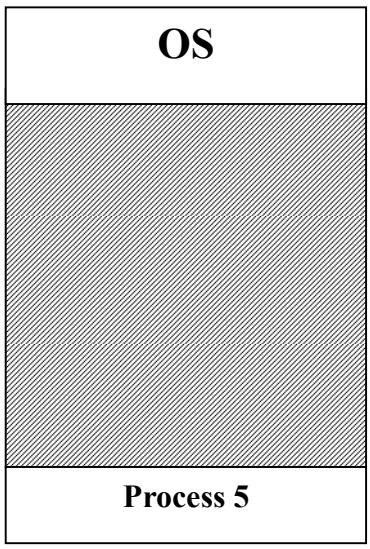
Main Memory

٤) حالة الذاكرة بعد انتهاء العملية P1



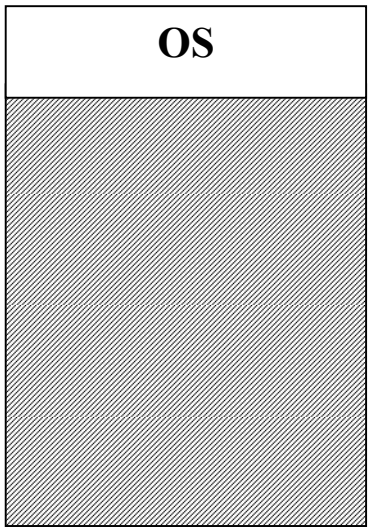
Main Memory

٥) حالة الذاكرة بعد انتهاء العملية P3



Main Memory

٦) حالة الذاكرة بعد انتهاء العملية P5



Main Memory

وهنالك مجموعة من الطرق التي من خلالها يقوم مدير الذاكرة بمراقبة القطاعات الفارغة (كيف يعرف أنها فارغة) والقطاعات المشغولة (كيف يعرف أنها مشغوله) وهي على النحو التالي :

٣,٢. مراقبة الذاكرة

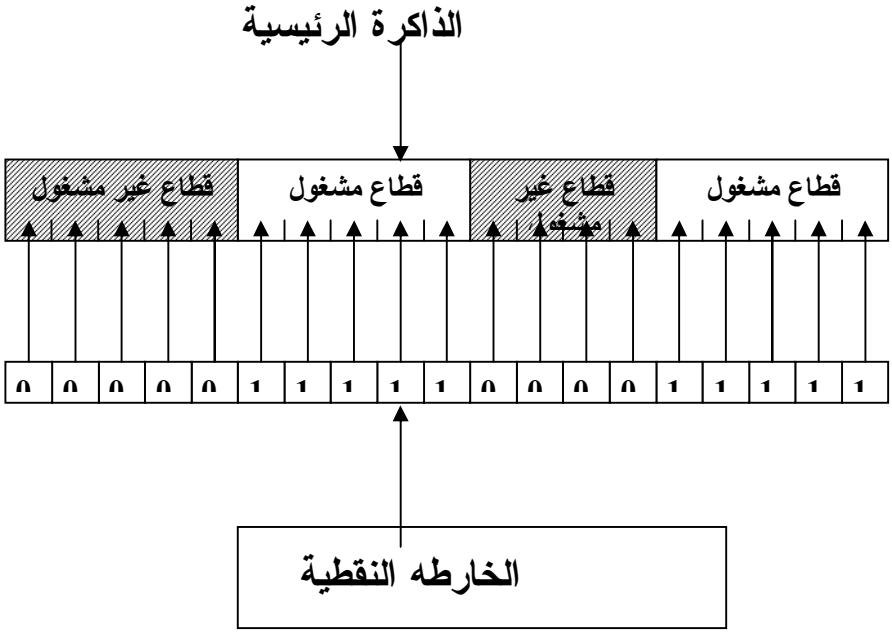
١,٣,٢. مراقبة الذاكرة باستخدام الخارطه النقطية

Memory Management With Bit Map

تتلخص هذه الطريقة بتقسيم الذاكرة إلى قطعة صغيرة تسمى كل قطعه Allocation Bit بحيث تكون هنالك خريطة من المربعات تمثل هذه القطع الصغيرة، وذلك بتمثيل القطعة الصغيرة المشغوله في الذاكرة بمربع يوجد بداخلة الرقم 1 في الخارطه النقطية،والقطعة غير المشغولة في الذاكرة بمربع يوجد بداخله الرقم 0 في الخارطة،

وبذلك يقوم مدير الذاكرة بإدارة الذاكرة عن طريق الخارطة النقطية أوعنـــد انتهـــاء أي عملية من التنفيذ يقوم بتحويل المربعات الخاصه بها في الخارطه من 1 إلى 0

مثال :



٢,٣,٢. مراقبة الذاكرة باستخدام القوائم المتصلة

Memory Management With Linked List

فكرة هذه الطريقة تتمثل في تمثل قطاعات الذاكرةسواء أكانت مشغولة أو غيـــر مشغولة بقوائم متصله مع بعضها البعض بحيث تكون كل قائمة من هذه القـــوائم تمثـــل قطاعاًمعيناً في الذاكرة أوتتكون كل قائمه من أربع أجزاء وهي على النحو التالى :-

١) الجزء الأول

الجزء الأول من القائمة يوضح حالة القطاع هل هو فارغ أم مشغول، بحيـــث يرمز للقطاع الفارغ بالرمز H للقطاع المشغول بالرمز P .

٢) الجزء الثاني

الجزء الثاني من القائمة يوضح عنوان أول قطعة صغيرة في القطاع .

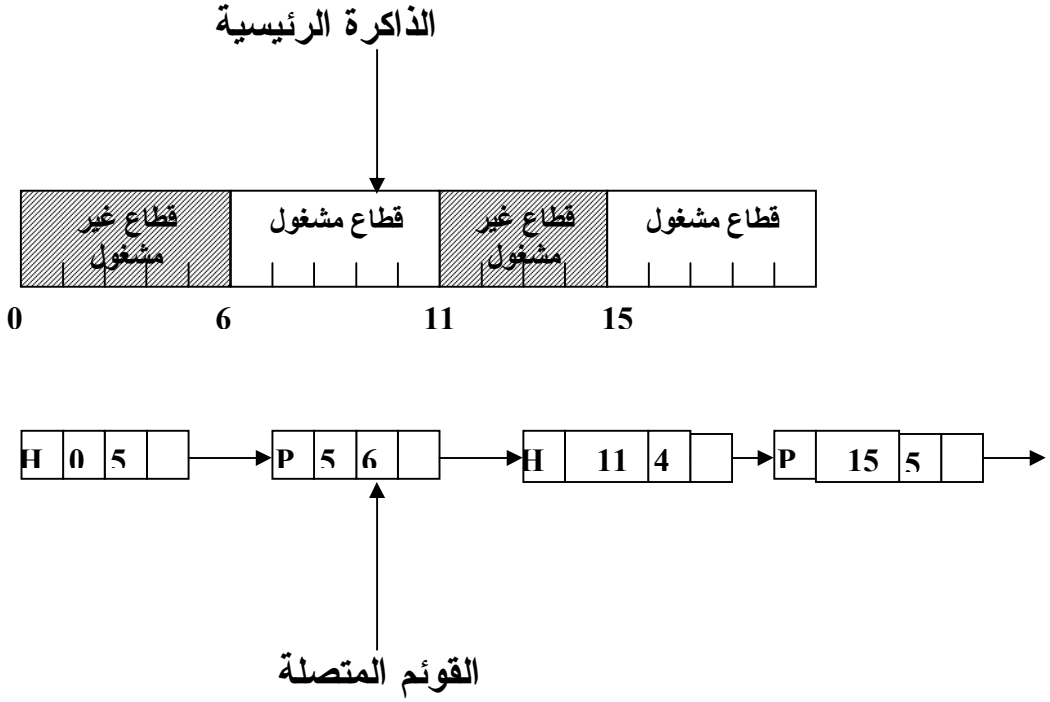
٣) الجزء الثالث

الجزء الثالث من القائمة يوضح الحجم الكلي للقطاع .

٤) الجزء الرابع

الجزء الرابع من القائمة عبارة عن مؤشر يشير إلى القائمة التالية لهذه القائمة .

مثال :



مثال :

إذا كان لدينا العمليات التالية P1 , P2 , P3 ،وكانت تلك العمليات تحتاج إلى المساحات 20K , 10K , 5K على التوالي ، مستخدماً استراتيجية المكان الأول First Fitتحتاج المطلوب بالرسم توضيح حالة الذاكرة بعد تسكين كل عملية ، علماً بأن الذاكرة المتاحه موضحه بالرسم التالي .

10K	10K	20K	30K	10K	20K	20K	20K

حالة الذاكرة الابتدائية

الحل

١) حالة الذاكرة بعد تسكين العملية الأولى

10K	10K	20K	P1	10K	20K	20K	20K

٢) حالة الذاكرة بعد تسكين العملية الثانية .

10K	P2	20K	P1	10K	20K	20K	20K

٣) حالة الذاكرة بعد تسكين العملية الثالثة .

10K	P2	20K	P1	10K	P3	20K	20K

مثال :

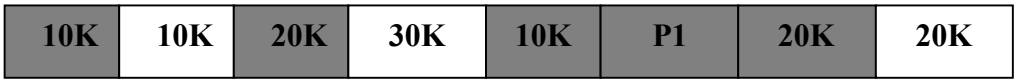
إذا كان لدينا العمليات التالية P١ , P2 , P3 ، وكانت تلك العمليات تحتاج إلى المساحات 20K , 10K , 5K على التوالي ، مستخدماً استراتيجية المكان الأحسن Best Fit المطلوب بالرسم توضيح حالة الذاكرة بعد تسكين كل عملية ، علماً بأن الذاكرة المتاحة موضحة بالرسم التالي .

10K	10K	20K	30K	10K	20K	20K	20K

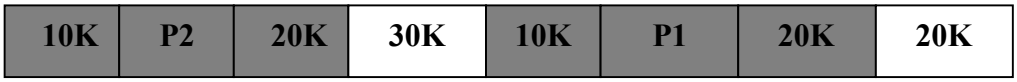
حالة الذاكرة الابتدائية

الحل:

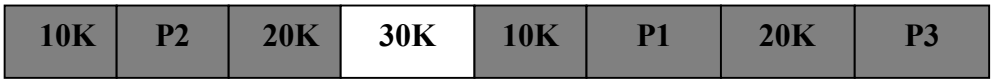
٤) **حالة الذاكرة بعد تسكين العملية الأولى**



٥) **حالة الذاكرة بعد تسكين العملية الثانية .**

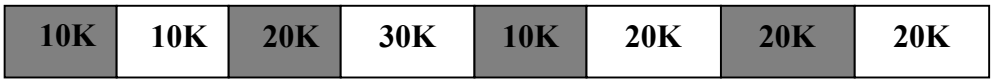


٦) **حالة الذاكرة بعد تسكين العملية الثالثة .**



مثال:

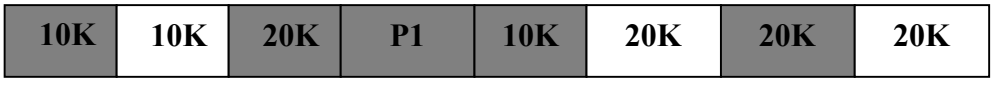
إذا كان لدينا العمليات التالية P1 , P2 , P3 ،وكانت تلك العمليات تحتاج إلى المساحات 20K , 10K , 5K على التوالي ، مستخدماً استراتيجية المكان الأكبر Worst Fit المطلوب بالرسم توضيح حالة الذاكرة بعد تسكين كل عملية ، علماً بأن الذاكرة المتاحه موضحه بالرسم التالي .



حالة الذاكرة الابتدائية

الحل

٧) **حالة الذاكرة بعد تسكين العملية الأولى**



٨) **حالة الذاكرة بعد تسكين العملية الثانية .**

10K	10K	20K	P1	10K	P2	20K	20K

٩) **حالة الذاكرة بعد تسكين العملية الثالثة .**

10K	10K	20K	P1	10K	P2	20K	P3

مثال :

إذا كان لدينا العمليات التالية P1 , P2 , P3 ،وكانت تلك العمليات تحتاج إلى المساحات 20K , 10K , 5K على التوالي ، مستخدماً استراتيجية المكان التالى Next Fit .المطلوب بالرسم توضيح حالة الذاكرة بعد تسكين كل عملية ، علماً بأن الذاكرة المتاحه موضحه بالرسم التالي .

10K	10K	20K	30K	10K	20K	20K	20K

حالة الذاكرة الابتدائية

الحل

١) **حالة الذاكرة بعد تسكين العملية الأولى**

10K	10K	20K	P1	10K	20K	20K	20K

٢) **حالة الذاكرة بعد تسكين العملية الثانية .**

10K	10K	20K	P1	10K	P2	20K	20K

10K	10K	20K	P1	10K	P2	20K	P3

تدريب (١)

١) إذا كان لدينا العمليات التالية P1 , P2 , P3 ،وكانت تلك العمليات تحتاج إلى المساحات 10K , 25K , 20K على التوالي ، مستخدماً استراتيجية المكان التالى Next Fit المطلــوب بالرسم توضيح حالة الذاكرة بعد تسكين كل عملية ، علماً بأن الذاكرة المتاحه موضحة بالرســم التالي .

10K	30K	20K	15K	10K	25K	30K	50K

حالة الذاكرة الابتدائية

٢) إذا كان لدينا العمليات التالية P1 , P2 , P3 ،وكانت تلك العمليات تحتاج إلى المساحات 10K , 25K , 20K على التوالي ، مستخدماً استراتيجية المكان الاول First Fit ،المطلــوب بالرسم توضيح حالة الذاكرة بعد تسكين كل عملية ، علماً بأن الذاكرة المتاحة موضحه بالرســم التالي .

10K	30K	20K	15K	10K	25K	30K	50K

حالة الذاكرة الأبتدائية

٣) إذا كان لدينا العمليات التالية P1 , P2 , P3 ،وكانت تلك العمليات تحتاج إلى المساحات 10K , 25K , 20K على التــوالي ، مــستخدماً اســتراتيجية المكــان الأكبــر Worst Fit ،المطلوب بالرسم توضيح حالة الذاكرة بعد تسكين كل عملية ، علمــاً بــأن الــذاكرة المتاحــة موضحة بالرسم التالي .

10K	30K	20K	15K	10K	25K	30K	50K

إذا كان لدينا العمليات التالية P1 , P2 , P3 ،وكانت تلك العمليات تحتاج إلى المساحات 10K , 25K , 20Kعلى التوالي ، مستخدماً استراتيجية المكان التالى Next Fit المطلوب بالرسم توضيح حالة الذاكرة بعد تسكين كل عملية ، علماً بأن الــذاكرة المتاحــه موضــحة بالرسم التالي .

10K	30K	20K	15K	10K	25K	30K	50K

اسئلة تقويم ذاتي

١/ لماذا نقسم الذاكرة؟
٢/ ما أساليب تقسيم الذاكرة؟
٣/ أذكر عيوب التقسيم الثابت.
٤/ كيف يقوم مدير الذاكرة بمراقبة القطاعات الفارغة والقطاعات المشغولة؟

٣. الذاكرة الافتراضية

Virtual Memory

قبل عدة سنوات واجهت مستخدمي الحاسوب مشكلة في البرامج تتمثل في احتياج البرنامج إلى مساحة تخزينية أكبر من المساحة المتاحة .

وكان الحل هو تجزئة البرنامج من قبل المبرمج إلى أجزاء تسمى **بالطبقات** ،بحيث تبدأ التنفيذ من الطبقة الأولى أولاً ،وعندما تنتهي تستدعي طبقة أخرى ،ويتم المبادلة بينهما إلى داخل و خارج الذاكرة من قبل نظام التشغيل حسب الحاجة .

ولكن ظهرت بعض المشاكل .عندما يكون البرنامج كبيراً بحيث تكون عملية تجزئته من قبل المبرمج عملية مملة وتستهلك وقتاً طويلاً ، لذلك ادى هذا الأمر إلى التفكير عن وسيلة لرمي العمل كله على عاتق الحاسب .

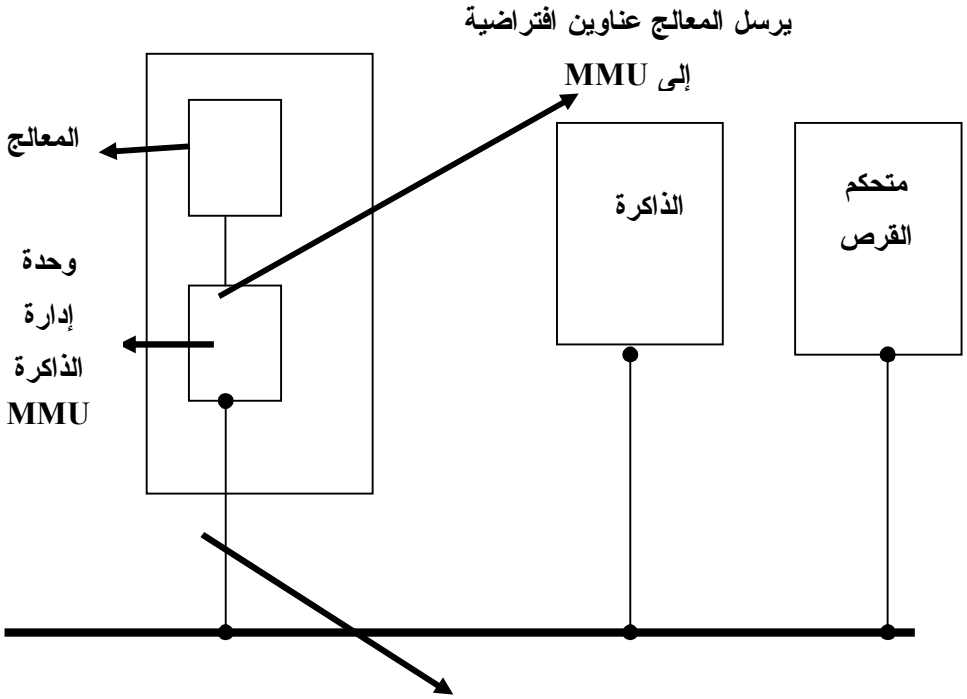
ولم يمضى وقت طويل حتى اخترع Foheringham عام ١٩٦١م طريقة لرمي العمل كله على عاتق الحاسب والتي سميت بالذاكرة الافتراضية Virtual Memory

والفكرة الأساسية لهذه الطريقة هي أن الحجم الكامل للبرنامج (بيانات + مكدسة) قد يزيد عن حجم الذاكرة المتوفرة ، بحيث يحتفظ نظام التشغيل بأجزاء البرنامج المستخدمة حالياً في الذاكرة الرئيسية، ويترك باقي البرنامج في القرص .

فمثلاً يمكن تشغيل برنامج بحجم 32MB على جهاز به ذاكرة 8MB باختيار 8MB المراد تنفيذها الان كي تبقى في الذاكرة ،ومبادلة أجزاء البرنامج بين القرص والذاكرة حسب الحاجة .

١,٣. التصفيح

تستخدم معظم أنظمة الذاكرة الافتراضية تقنية تسمى التصفيح PAGING ،وهي إمكانية توليد عناوين افتراضية للذاكرة أكبر من العناوين الحقيقة، بحيث يتم بعد ذلك تحويل العناوين الافتراضية إلى عناوين حقيقة بواسطة وحدة إدارة الذاكرة MMU Memory Management Unit



ترسل MMU عناوين فيزيائية إلى الذاكرة

يتم تقسيم العناوين الأفتراضية إلى وحدات تسمى بالصفحات Pages حسب طول العناوين الافتراضية المولده من قبل المعالج، بينما تسمى الوحدات المقابلة لهـذه الصفحات في الذاكرة الفيزيائية بإطارات الصفحات Page Frames

فمثلاً إذا كان لدينا ذاكرة فيزيائية بحجم 32KB، وكان المعالج يولد عناوين افتراضـية بحجم 64KB، يكون لدينا 16 صفحة افتراضية ويقابلها 8 إطارات صفحات .

الشكل أعلاه يوضح العلاقة بين العناوين الافتراضية والعناوين الفيزيائية بواسطة جدول الصفحات

بهذه الطريقة نكون قد قمنا بحل المشكلة السابقة . ولكن ربما تظهر مشكلة أخـرى، إذا حاولت عملية استخدام صفحة غير موجودة. أي ليس لها إطار صفحة .

(في الشكل أعلاه الصفحات الموضحة بعلامة X ليس لها ذاكرة مقابلة) .

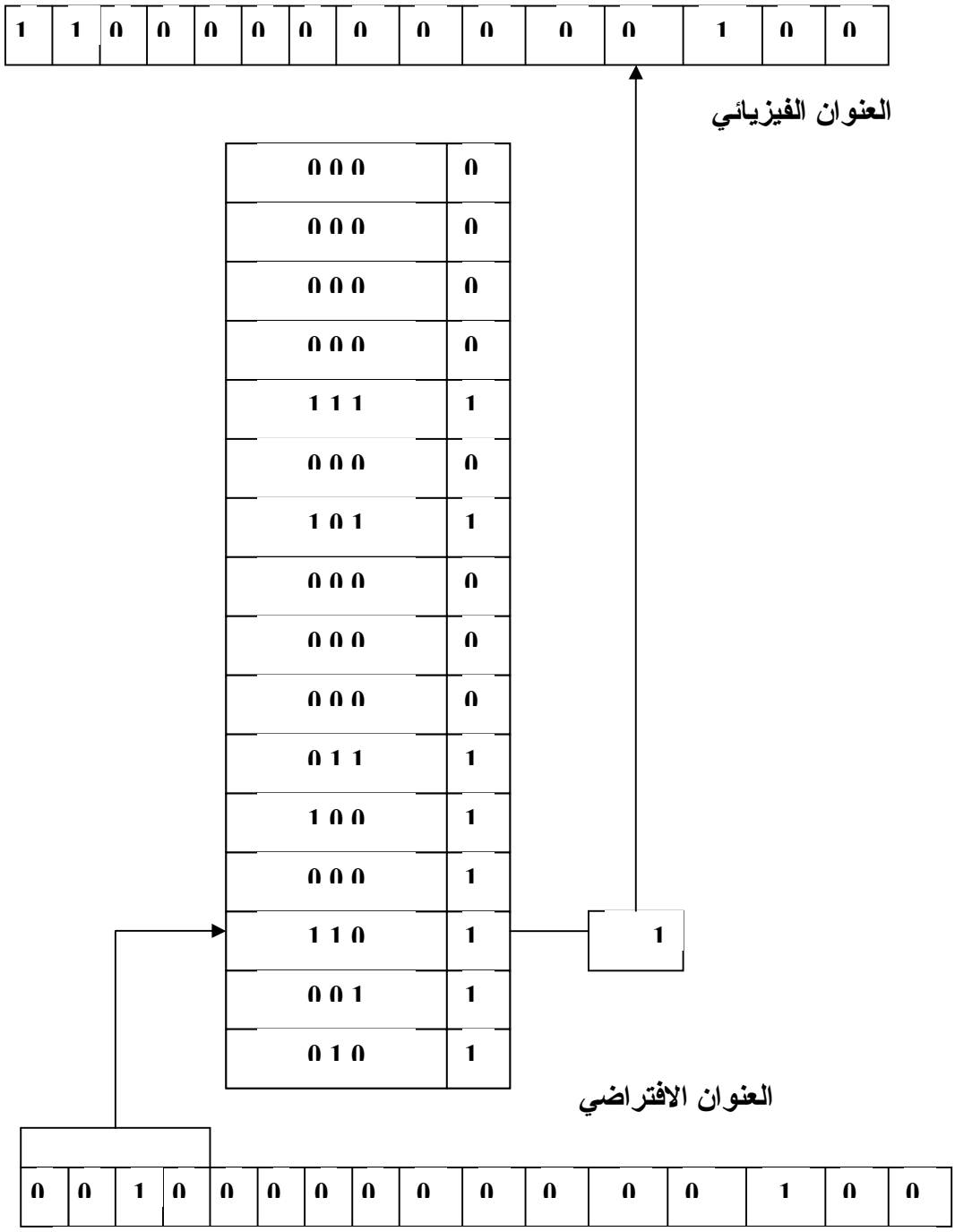
وهذا يؤدي إلى ما يسمى بخطأ الصفحة Page Fault . وعلى سبيل المثال إذاقامت عملية باستخدام العنوان الافتراضي ٣٢٧٨٠ والذي يقع فـي الـصفحة الافتراضـية ٨ وتلاحظ وحدة إدارة الذاكرة أن هذه الصفحة ليس لها إطار، يؤدي ذلك إلى جعل المعالج ينتقل إلى نظام التشغيل والذي بدورة يقوم باختيار إطار صفحة قليل الاستخدام ويكتـب محتوياتها في القرص، ثم يأتي بالصفحة التي أشارت إليها العملية ويضعها فـي إطـار الصفحة التي تم نقل محتوياتها إلى القرص ، ثم بعد ذلك يقوم بتعديل خارطـة MMU، ويعيد تنفيذ العملية التي تسببت بالخطأ .

فمثلاً يقرر نظام التشغيل بتحرير الإطار رقم ١ فينقل محتوياته إلـى القـرص ويقـوم باستبدال الإشارة X الموجودة بالصفحة رقم ٨ بالقيمة ١ التي تشير إلـي رقـم إطـار الصفحة وتحرير الصفحة الافتراضية التي كانت تحمل القيمة ١ بالإشارة X .

الآن سوف نتطرق لشرح الطريقة التي تستخدمها وحدة إدارة الذاكرة لتحويـل العنـوان الافتراضي إلى عنوان فيزيائي. أولاً تقومMMU بتحويل العنوان الافتراضي العـشري إلى عنوان افتراضي ثنائي بطول ١٦ بت، ثانياً يقسم العنوان الافتراضي الوارد ذو ١٦ بت إلى رقم الصفحة بطول ٤ بت وإزاحة بطول ١٢بت. وعن طريـق رقـم الـصفحة يمكن معرفة رقم إطار الصفحة المقابلة للصفحة الافتراضية المحـددة. فمـثلاً إذا قـام المعالج بتوليد العنوان الافتراضي ٨١٩٦ ومرره إلى وحدة إدارة الذاكرة تقـومMMU بتحويل هذا العنوان إلى السلسة الثنائية المقابلة وهي ٠٠١٠٠٠٠٠٠٠٠٠١٠٠

ثمً يقسم هذه السلسة إلى ٤ بت تمثل رقم الصفحة الافتراضية و ١٢ بت تمثل الإزاحـة فيكون رقم الصفحة الافتراضية المقابلة للعنوان الافتراضي هي ٠٠١٠ والتي تمثل الرقم ٢ فتنظر وحدة إدارة الذاكرة داخل الصفحة الافتراضية رقم ٢. لمعرفـة رقـم إطـار الصفحة وليكن ١١٠، ثم تضيف هذه السلسلة إلى سلسلة الإزاحة فيصبح العنوان الجديد

هو ١١٠٠٠٠٠٠٠٠٠١٠٠ والذي يقابل الرقم ٢٤٥٨٠ عشرياً وهو يمثـــل العنـــوان الفيزيائي. وبذلك تكونMMU قد قامت بتحويل العنوان الافتراضي ٨١٩٦ إلى العنوان الفيزيائي ٢٤٥٨٠ الذي سوف يمرر إلى الذاكرة .



الشكل أعلاه يوضح طريقة عمل وحدة إدارة الذاكرة في تحويل العنوان الافتراضي إلى عنوان فيزيائى

٢,٣. خوارزميات استبدال الصفحات

عند حدوث خطأ الصفحة Page Fault يقوم نظام التشغيل باختيار صفحة لإزالتها من الذاكرة لإتاحة المجال للصفحة التى ستجلب إلى الذاكرة.

على الرغم من أن اختيار الصفحة يمكن أن يكون عشوائياً، إلا أن أداء النظام يمكن أن يتحسن كثيراً إذا تم اختيار صفحة قليلة الاستخدام من أن نقوم بإزالة صفحة كثيرة الاستخدام، لأنها سوف تجلب مره أخرى إلى الذاكرة خلال فترة قصيرة مما يؤدي إلى عبء إضافي.

لذلك كانت هنالك الكثير من الدراسات التي أجريت في مجال خوارزميات استبدال الصفحات من الناحيتين العملية والنظرية .وسوف نقوم بشرح بعض الخوارزميات الأكثر أهمية ومنها :

١) خوارزمية استبدال الصفحة المثلى

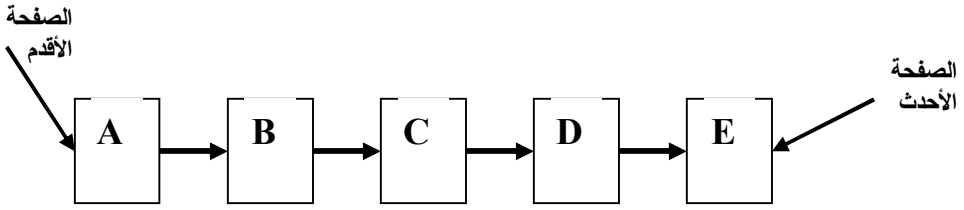
فكرة هذه الخوارزمية تتمثل في الاتي، عند لحظة حدوث خطأ الصفحة يقوم نظام التشغيل بوضع لافتة على كل صفحة تدل على أن هذه الصفحة سوف يتم استخدامها بعد N تعليمة. ثم بعد ذلك يختار نظام التشغيل الصفحة ذات اللافتة الأعلى لإزالتها.

فمثلاً إذا كان لدينا صفحة لن تستخدم قبل ٥٠٠ تعليمة وصفحة أخرى لن تستخدم قبل ٢٠٠ تعليمة فإن إزالة الصفحة الأولى سوف تؤخر حدوث خطأ الصفحة.

من عيوب هذه الخوارزمية أنها نظرية وليست عملية، أي أنها مستحيلة التنفيذ لأنه عند حدوث خطأ الصفحة لا يملك نظام التشغيل أي طريقة لمعرفة متى سيشار إلى كل صفحة مستقبلاً.

٢) خوارزمية استبدال الصفحة الداخله أولاً تخرج أولاً
(First In First Out) FIFO

من الاسم يتضح فكرة الخوارزمية والتي تتمثل في أن ينظم نظام التشغيل قائمة متصلة من جميع الصفحات، بحيث تكون في رأس القائمة الصفحة الأقدم من حيث الاستخدام وفي ذيل القائمة الصفحة الأحدث. وعند حدوث خطأ الصفحة يقوم نظام التشغيل بإزالة الصفحة الموجودة في رأس القائمة ويضيف الصفحة الجديدة إلى ذيل القائمة.

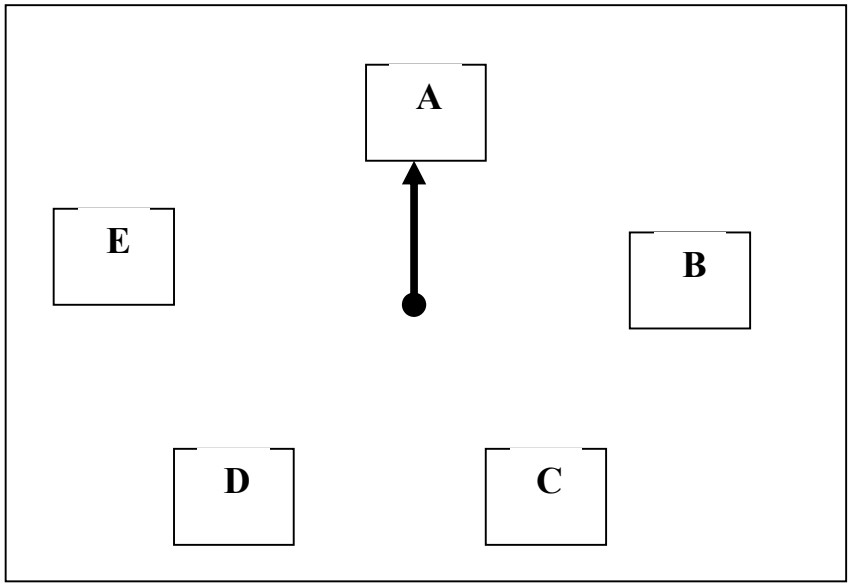


٣) خوارزمية الساعة لاستبدال الصفحات

تتمثل فكرة هذه الخوارزمية بأن يكون هنالك مؤشر في كل صفحة يحمل إحدى القيمتين صفر، وتشير إلي أن هذه هي الصفحة المراد إزالتها أو العكس إذا كانت تحمل القيمة واحداً .

وينظم نظام التشغيل جميع الصفحات في قائمة دائرية في شكل ساعة ويشير عقرب الساعة إلى الصفحة الأقدم، وعند حدوث خطأ الصفحة يتم اختبار عقرب الساعة إذا كان يشير إلى صفحة مؤشرها يساوي صفراً يقوم بازالة هذه الصفحة. ويضع الصفحة الجديدة في مكان الصفحة التى تم إزالتها، ويقدم عقرب الساعة إلى الصفحة التالية، أما إذا كان عقرب الساعة يشير إلى صفحة مؤشرها لا يساوي صفراً يقوم بتصفير مؤشر هذه الصفحة، ثم يقدم عقرب الساعة إلى الصفحة التالية حتى الوصول إلى الصفحة التي يحمل مؤشرها القيمة صفراً .

مثال



٤) خوارزمية استبدال الصفحة الأقل استخداماً مؤخراً

Least Recently Used LRU

بنيت فكرة هذه الخوارزمية على أن الصفحات المستخدمة بكثرة في التعليمات القليلة السابقة ستستخدم بكثرة في التعليمات القليلة التالية ، وبهذا تكون الصفحات التي لم تستخدم لفترة طويلة ستبقى كذلك لفترة طويلة على الأغلب ، فعند حدوث خطأ الصفحة يقوم نظام التشغيل بازالة الصفحة التي لم تستخدم لأطول فترة من الذاكرة .

وعلى الرغم من أنه يمكن أن تحقق LRU نظرياً الا إنها تتطلب وجود قائمة خطية لكل الصفحات في الذاكرة بحيث تكون الصفحة الأكثر استخداماً مؤخراً موجودة في الأمام، والصفحة الأقل استخداماً موجودة في الخلف ، وتحديث هذه القائمة عند كل إشارة إلى الذاكرة تتطلب وقتاً، لذلك فهي صعبه التحقيق .

وعندما يكون عدد الصفحات كبيراً يقوم نظام التشغيل بتنظيم مصفوفة $N \times N$

بت وتكون عناصرها تحمل القيمة صفراً في البداية ، وعندما يتم الإ شارة إلى الصفحة K يقوم نظام التشغيل أولاً بجعل جميع عناصر الصف الذي يحمل الرقم K تساوي ١، ثم يجعل كل عناصر العمود الذي يحمل نفس القيمة تساوي صفراً .

وفي أي لحظة يكون الصف ذو القيمة الثنائية الأدنى هو الأقل استخداما مؤخراً .

فمثلاً إذا كان لدينا ٤ صفحات وتم الأشارة إليها بالترتيب التالي ١٢٣٤٣٢١٤، وضـــح شكل المصفوفة بعد كل إشارة.

1

	1	**2**	**3**	**4**
1	0	1	1	1
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

2

	1	**2**	**3**	**4**
1	0	0	1	1
2	1	0	1	1
3	0	0	0	0
4	0	0	0	0

3

	1	**2**	**3**	**4**
1	0	0	0	1
2	1	0	0	1
3	1	1	0	1
4	0	0	0	0

4

	1	**2**	**3**	**4**
1	0	0	0	0
2	1	0	0	0
3	1	1	0	0
4	1	1	1	0

5

	1	**2**	**3**	**4**
1	0	0	0	0
2	1	0	0	0
3	1	1	0	1
4	1	1	0	0

6

	1	**2**	**3**	**4**
1	0	0	0	0
2	1	0	1	1
3	1	0	0	1
4	1	0	0	0

7

	1	**2**	**3**	**4**
1	0	1	1	1
2	0	0	1	1
3	0	0	0	1
4	0	0	0	0

8

	1	**2**	**3**	**4**
1	0	1	1	0
2	0	0	1	0
3	0	0	0	0
4	1	1	1	0

٥) خوارزمية WSCLOCK لاستبدال الصفحة

WORKING SET CLOCK

تعتبر من الخوارزميات الأكثر استخدماً من الناحية العملية لبساطة تحقيقها وأدائها الجيد ، وهي شبيهة بخوارزمية الساعة .

تتمثل فكرة هذه الخوارزمية بأن يكون لكل صفحة بنية بيانات تتكون من حقلين ،حقل يمثل زمن آخر استخدام للصفحة والحقل الآخر به مؤشراً يحمل إحدى القيمتين صفر، وتشير إلى إن هذه هي الصفحة المراد إزالتها أو العكس. لذا كانت تحمل القيمة واحداً .

وتوضع كل هذه البنيات في شكل دائرة بحيث تكون القائمة في البداية فارغة. وعند تحميل الصفحة الأولى إلى الذاكرة تضاف البنية الخاصة بها بحيث يكون حقل المؤشر دائماً في بداية تحميل أي صفحة يحمل القيمة ١ ، وعقرب الساعة يشير إلى أول صفحة تم تحمليها الى الذاكرة .

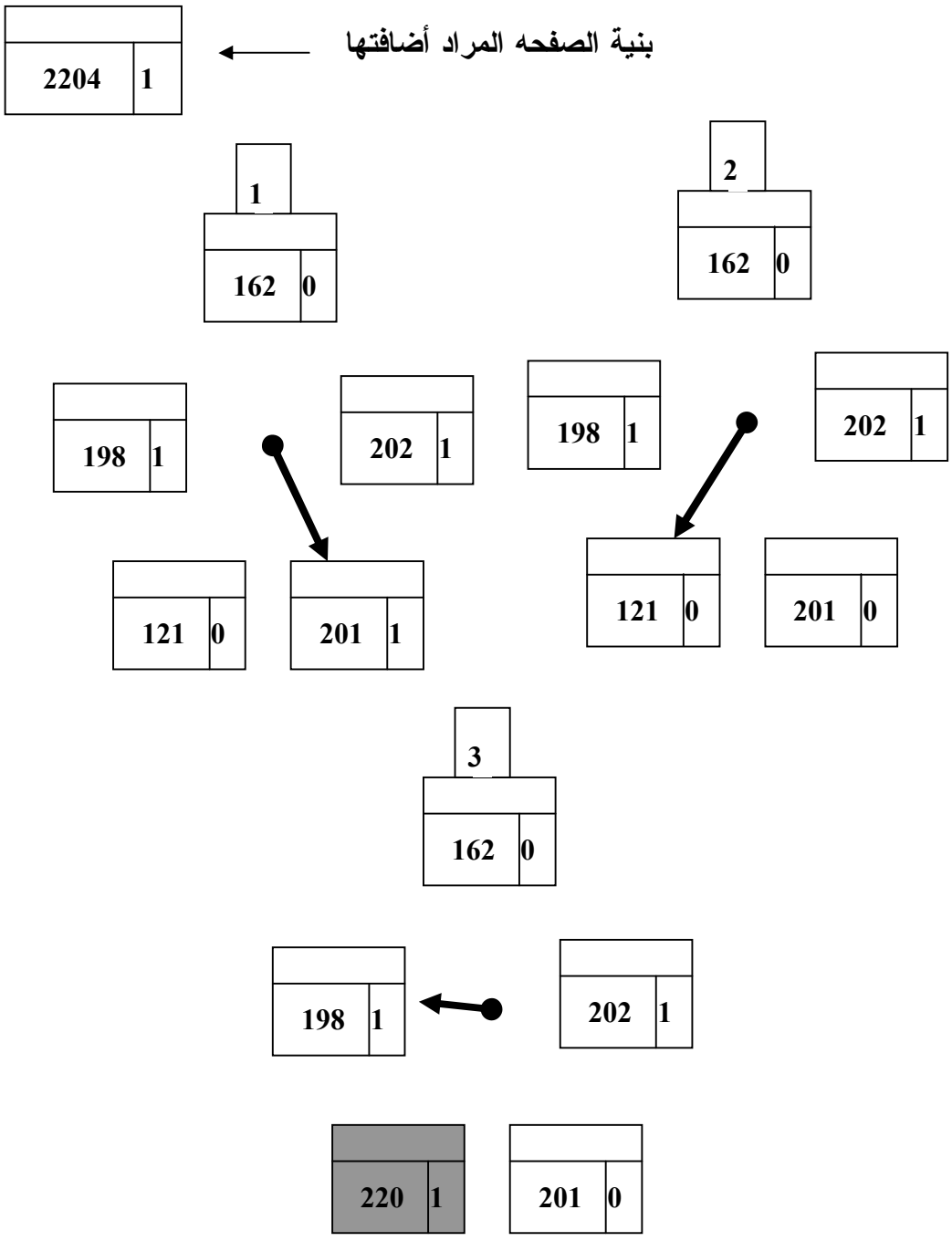
وعند حدوث خطأ الصفحة يقوم نظام التشغيل بأختبار البنية التي يشير إليها عقرب الساعةفإذا كان المؤشر الموجود داخل هذه البنية يساوي ١ يقوم نظام التشغيل بتصفيره وينقل عقرب الساعة إلى البنية التالية .

أما إذا كان المؤشر يحمل القيمة صفراً ينظر بعد ذلك إلى حقل زمن آخر استخدام فإذا كان الزمن الظاهري الحالي ناقصاً زمن آخر أستخدام (عمر الصفحه) أكبر من T ميلي ثانية سابقة يتم إزالة تلك الصفحة ووضع الصفحة الجديدة في مكانها وتحريك عقرب الساعة الي الصفحه التالية .

أما إذا كان المؤشر يساوي صفراً و الزمن الظاهري الحالي ناقصاً زمن آخر استخدام أقل من أو يساوي T ميلي ثانية سابقة نحرك عقرب الساعة إلى التالية فإذا تم تحريك عقرب الساعة حتى إشارة إلى نفس الصفحة أي أنها قامت بدورة كاملة هذا يعني أن كل الصفحات حديثة الاستخدام، وفي هذه الحالة يختار نظام التشغيل الصفحة ذات العمر الأكبر (أكبر قيمة للزمن الظاهري الحالي ناقصاً زمن آخر استخدام) .

علماً بأن T تمثل مقياساً لمعرفة أي الصفحات أقدم. ويتم تحديدها مسبقاً .

مثال


بنية الصفحه المراد أضافتها
2204 1
1
162 0
2
162 0
198 1
202 1
198 1
202 1
121 0
201 1
121 0
201 0
3
162 0
198 1
202 1
220 1
201 0

أسئلة تقويم ذاتي

١/ كيف أنشئت فكرة إستخدام ذاكرة افتراضية.؟

٢/ عرّف تقنية التصفيح Paging .

٣/ ما الخوارزمية الأكثر استخداماً لاستبدال الصفحة.؟

الخلاصة

عزيزي الدارس، لنقم سويا بمناقشة ما درسناه في هذه الوحدة، قمنا في جزئها الأول بتعريف أنواع نظم التشغيل وهي نظم التشغيل، أحادية البرامج وهو نظام إتاحة جميع موارد الحاسب لعملية واحدة حتى تنتهي من التنفيذ، والنوع الثاني هو نظم التشغيل متعددة البرامج، وقد عالجت القصور في الأنظمة أحادية البرامج.

تحدثنا عن إدارة الذاكرة وأساليب تقسيمها إلى تقسيم ثابت أي قطاعات ثابتة محددة ويخصص لكل عملية القطاع المناسب لها. ولكن يمكن أن يكون هنالك مجوعة من القطاعات غير المشغولة ومساحات غير مشغولة داخل القطاع. يستخدم مدير الذاكرة عدة استراتيجيات لتحديد القطاع المعين لتسكين العملية وهي:

- إستراتيجية المكان الأول.
- إستراتيجية المكان التالى.
- إستراتيجية المكان الأحسن.
- إستراتيجية المكان الأكبر.
- إستراتيجية المكان الأسرع.

ووصفنا التقسيم الديناميكي للذاكرة بأنه تقسيم الذاكرة حسب حجم العمليات. وقد أنتجت أيضاً مايسمى بالفراغات الخارجية.

ناقشنا كيفية مراقبة القطاعات الفارغة والمشغولة في الذاكرة باستخدام الخارطة النقطية، ثم باستخدام القوائم المتصلة.

في القسم الثالث تحدثنا عن حل هام وهو إستخدام الذاكرة الافتراضية بتقسيم البرنامج إلى طبقات. ويتم تنفيذه بأجزاء يتم تبادلها بين الذاكرة الرئيسية والذاكرة الافتراضية. وشرحنا تقنية التصفيح وهي تقسيم العناوين الإفتراضية الي وحدات تسمى الصفحات. وتسمى الوحدات المقابلة لها في الذاكرة بإطارات الصفحات. شرحنا بعض خوارزميات استبدال الصفحات مثل خوارزمية استبدال الصفحة المثلى، وخوارزمية الصفحة الداخلة أولاً تخرج أولاً وخوارزمية الساعة لاستبدال الصفحات، وخوارزمية استبدال الصفحة الأقل استخداماً مؤخراً، وخوارزمية WSClock.

لمحة مسبقة عن الوحدة التالية

عزيزي الدارس، في الوحدة التالية سنناقش إدارة أجهزة الإدخال والإخراج كواحدة من أهم الوظائف الرئيسية التي يقوم بها نظام التشغيل، حيث نناقش أنواع البرمجيات التي تحدد عمل أجهزة الإدخال والإخراج وطرق إنجازها

مسرد المصطلحات

- **مدير الذاكرة Memory Manager**

هو جزء من النظام يقوم بمهام تنفيذ إدارة موارد الحاسب.

- **نظم التشغيل أحادية البرامج Monoprogramming**

هو نظام إتاحة جميع موارد الحاسب لعملية واحدة، وتكون إدارة الذاكرة في هذا النوع عملية بسيطه جداً.

- **نظم التشغل متعددة البرامج Multiprogramming**

هي النظم التي تقوم بدفع عدة عمليات للتنفيذ في نفس الوقت.

- **الفراغ الداخلي Internal Fragmentation .**

مساحات غير مشغولة داخل القطاع عندما يتم تسكين عملية في قطاع أكبر من حجمها لعدم وجود الحجم المناسب لها.

- **قتل العملية Process Killing**

هو إذا زاد حجم العملية عن حجم القطاع الخاص بها يتم البحث عن قطاع آخر مناسب، فإذا لم نجد نقوم بنقل العملية من الذاكرة الرئيسية الي الذاكرة الثانوية، واذا لم نجد أيضاً مكاناً مناسباً في القرص الصلب تلجأ بعض أنظمة التشغيل إلى قتلهاوذلك لتفادي حـــدوث مايسمى **الجمود Deadlock** للنظام ككل.

- **الفراغات الخارجية External Fragmentation**

فراغات موجودة بين القطاعات المشغولة.

المراجع

١-يمان اللبني وأسامة العبد الله، تصميم وتنفيذ نظم التشغيل الحديثة، شعاع للنشر والعلوم، سوريا-حلب،٢٠٠٥م.

٢-ج آرتشر هاريس (ترجمة أمين أيوبي)، أنظمة تشغيل الحاسوب، أكاديمياً، بيروت ٢٠٠٢م.

٣-عبد الرءوف الحلاق ومنتصر خاطر، أنظمة التشغيل، منشورات جامعة القدس المفتوحة، ١٩٩٦م.

٤-محمد أحمد فكرين، نظم تشغيل الحاسبات، دار المريخ ١٩٩٦م.

٥- Silberschatz, A. and Galvin, P.B., "Operating System Concepts", Fifth edition Addison-Wesley, Reading, MA, 199٧.

6- Tanenbaum, Andrew S., "Modern Operating Systems", Prentice-Hall, Englewood Cliffs, NJ, 1992.

http://www.personal.kent.edu/~rmuhamma/OpSystems/os.html
http://www.cag.lcs.mit.edu/~rinard/osnotes/
http://williamstallings.com/OS4e.html


الوحدة الخامسة
إدارة أجهزة الإدخال
والإخراج

محتويات الوحدة

المقدمة

تمهيد

أهلاً بك عزيزي الدارس في الوحدة الخامسة والأخيرة من مقرر "**نظم التشغيل**" وهي بعنوان "**إدارة أجهزة الإدخال والإخراج** "، ويمكن اعتبار إدارة أجهـزة الإدخـال والإخراج واحدة من أهم المهام الرئيسية التي على نظام التشغيل القيام بهـا ، إذ عليـه إرسال الأوامر إلى تلك الأجهزة واكتشاف ومعالجة الأخطاء إن وجدت وخلاف ذلك من المهام.

وسنناقش في القسم الأول من هذه الوحدة أجهزة الإدخال و الإخراج وأنواعهـا وبعض المفاهيم العامة لها، أما في قسمنا الثاني سنشرح طرق إنجاز عمليـات الإدخـال والإخراج، ونتعرف في القسم الثالث عل معمارية القرص التخزينية، ونضيف في القسم الرابع مفهوم إدارة الفراغات التخزينية، ونختم الوحدة بالتعرف علىي جدولـة القـرص التخزينية، عزيزي الدارس أنصحك بدراسة هذه الوحدة باهتمام وحل الأسـئلة الـواردة فيها، وفقك الله.

أهداف الوحدة

عزيزي الدارس،

بعد فراغك من دراسة هذه الوحدة ينبغي أن تكون قادراً على أن:

- تتعرف على أجهزة الإدخال والإخراج و أهدافها.
- تشرح مستويات برمجيات أجهزة الإدخال و الإخراج.
- تعرف الطرق الأساسية الثلاثة لإنجاز عمليات الإدخال و الإخراج.
- تشرح معمارية القرص التخزينية.
- تفهم خوارزميات جدولة القرص التخزينية.

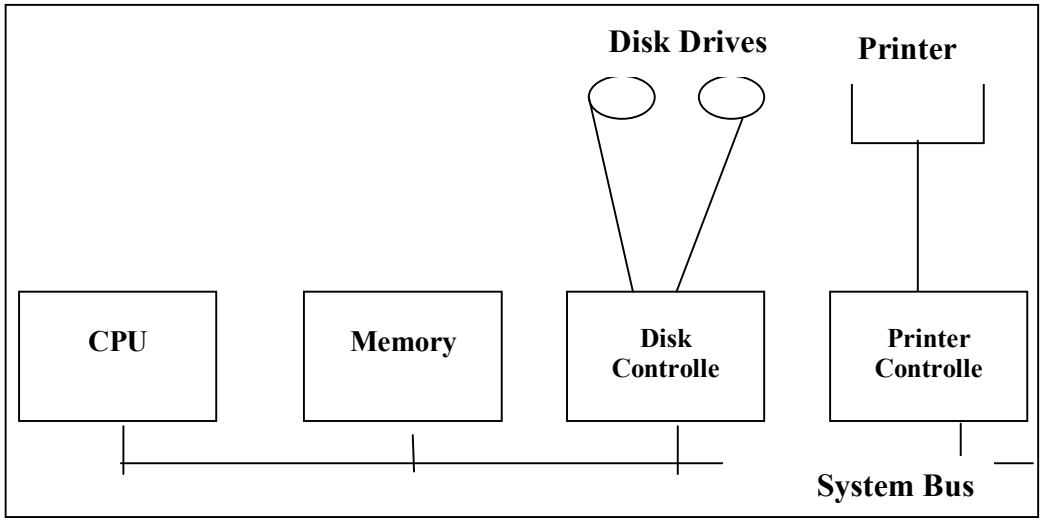
١. أجهزة الإدخال والإخراج

تنقسم أجهزة الإدخال والإخراج إلى فئتين: فئة معنونه (كتلة Block Devices) وفئة غير معنونة (Character Devices).

الفئة الأولى هي عبارة عن أجهزة يمكن تخزين المعلومات بها في كتل ثابتة الحجم وكل كتلة لها عنوان خاص، بالإضافة إلى إمكانية إجراء عمليات القراءة والكتابة والبحث والانتقال. وتمثل الأقراص واحدة من الأجهزة المعنونة الأكثر شيوعاً.

أما **الفئة الثانية** فتسمى بالأجهزة غير المعنونة. وهي ترسل وتستقبل سلسة من الإشارات. كما أنها لايمكن تخزين البيانات بها بالإضافة إلى عدم إمكانية إجراء أي من عمليات القراءة أو الكتابة أو البحث أو الأنتقال. وتعتبر الطابعات واحدة من الأجهزة غير المعنونة الأكثر شيوعاً.

عموماً فإن أي جهاز إدخال وإخراج يتكون من جزءين ، جزء ميكانيكي ويعتبر المكون المادي وجزء إلكتروني ويسمى بالمتحكمات Device Controller والذي يوفر التفاعل المنطقي مع الحاسب بحيث يتلقى الأوامر من وحدة المعالجة المركزية ثم بعد ذلك يقوم بترتيبها وتنفيذها.



الشكل(١) أجهزة الإدخال والإخراج

١. ١. أهداف برمجيات أجهزة الإدخال والإخراج

بوجود المكونات المادية لأجهزة الإدخال والإخراج مثل أجهزة التحكم Control Unit لابد من وجود مكونات برمجية لتساهم في أداء المكونات المادية.

١. ١. ١. الاستقلالية عن الأجهزة

تعتبر الاستقلالية عن الأجهزة (Device Independence) من المفاهيم الأساسية في تصميم برمجيات أجهزة الإدخال والإخراج ، ونعني بذلك إمكانية كتابة برنامج يستطيع الوصول إلى أي جهاز إدخال وإخراج دون تحديد نوع الجهاز بشكل مسبق.

فمثلاً إذا كان لدينا برنامج مكتوب لقراءة ملف، يجب أن يتمكن البرنامج من قراءة الملف من أي جهاز تخزين ، أي يمكنه قراءة الملف من القرص المرن أو القرص الصلب أو القرص المدمج أو غيرها من وسائط التخزين دون التعديل في البرنامج.

٢. ١. ١. معالجة الأخطاء

تعتبر معالجة الأخطاء (Error Handling) من المواضيع الهامة في تصميم برمجيات أجهزة الإدخال والإخراج ، بحيث يجب معالجة الأخطاء قريباً من الجزء المادي لأجهزة الإدخال والإخراج بقدر الإمكان ، إذ على المتحكمات Device Controller اكتشاف الأخطاء ومحاولة إصلاحها إن أستطاعت ، أما إذا لم تستطع عندها تقوم بمناداة نظام التشغيل لمعالجة الوضع، وذلك بعرض المشكلة على الطبقة الأعلي من المتحكمات Device Controller فإن لم تستطع الطبقة العليا معالجة الأخطاء يقوم نظام التشغيل بعرضها على الطبقة الأعلى وهكذا الي أن تصل إلى المستخدم في شكل رسائل الخطاء.

٣. ١. ١. التزامن

من المواضيع الهامة أيضاً في تصميم برمجيات أجهزة الإدخال والإخراج هو إمكانية التعامل مع دوائر التحكم بالصورتين المتزامنة Synchronous وغير المتزامنة Asynchronous. فإن معظم الدخل / الخرج الفيزيائي غير متزامن. فمثلاً يبدأ المعالج بتنفيذ عملية ثم يتضح بأن هذه العملية تحتاج إلى دخل / خرج، فيقوم المعالج بتنفيذ عملية أخرى حتى تصل دخل / خرج للعملية السابقة ، لذلك لابد أن يكون هنالك إ مكانية إرجاع البرنامج تلقائياً ريثما تتوفر البيانات ، كما يجب على نظام التشغيل جعل العمليات التي تكون في الواقع مقادة بالمقاطعات (غير متزامنة) تبدو كأنها متزامنه بالنسبة للمستخدم.

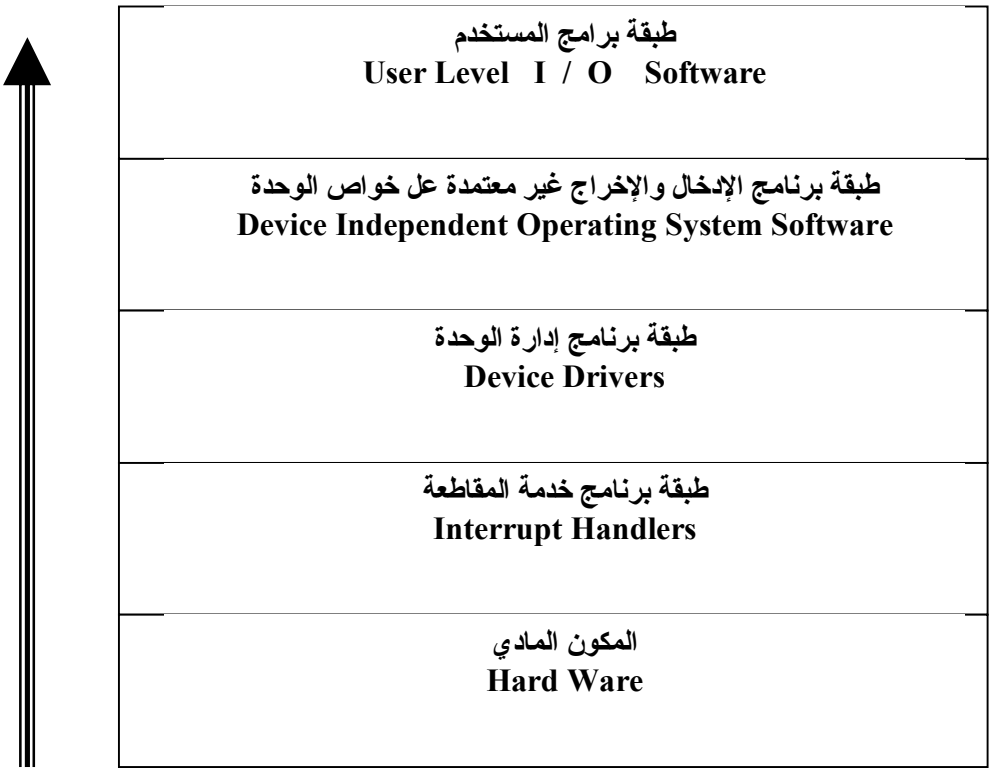
٤. ١. ١. مفاهيم هامة

من المفاهيم الهامة في تصميم برمجيات أجهزة الإدخال والإخراج هي إمكانية التعامل مع الأجهزة التي يمكن المشاركة فيها، وتلك التي لايمكن المشاركة فيها في مستوى داخلي لا يظهر للمستخدم.

فمثلاً يمكن استخدام الأقراص من قبل عده مستخدمين في نفس الوقت، أي يجب ان ألاّ يكون هنالك مشكلة إذا قام عدة مستخدمين بفتح عدة ملفات على نفس القرص في نفس الوقت.

١,٢. مستويات برمجيات أجهزة الإدخال و الإخراج

كما أن هنالك أجهزة أخرى مثل سواقات الأشرطة يجب أن تخصص لمستخدم واحد حتي انتهائه من استخدامها. عندها يستطيع مستخدم آخر استخدام سواقة الشريط ، وتم تحقيق تلك الأهداف في أربعة مستويات برمجية للتعامل مع وحدات الإدخال والإخراج وهي كما يلي :



الشكل (٢)مستويات برمجيات أجهزة الإدخال والإخراج

١,١,٢. طبقة برنامج خدمة المقاطعة

Interrupt Handlers

تعتبر المقاطعات واقعاً مفروضاً ولا يمكن تجنبها، ويجب دائماً إخفاؤها بعيداً عن المستخدم. بحيث لايدري بها إلا جزء صغير من نظام التشغيل. والطريقة الأفضل لأخفائها هي جعل برنامج التشغيل الذي يبدأ عملية الإدخال / الإخراج يتوقف ريثما تنتهي عملية الدخل / الخرج.

وقد تكون هذه العملية لأي وحدة من الوحدات. فبالتالي تكون هناك أنواع متعددة للقطع. وكل قطع له برنامج خدمة محدد موجود ضمن خدمات نظام التشغيل أو

النظام الأساسي لإدخال ولإخراج (BIOS) ، ويتم الرجوع لهذا البرنامج بعد تحديد نوعية القطع وموضع البرنامج الذي يقوم بهذه الخدمة.

٢,١,٢. طبقة برنامج إدارة الوحدة Device Driver

يعتبر هذا البرنامج جزءاً من نظام التشغيل الذي يتولى توجيه أوامر وحدةالتحكم، كما أنه يحتفظ بعدد من السجلات داخل وحدة التحكم وعناوينها.

فالوظيفة الأساسية لهذا البرنامج هي استقبال الطلبات من برامج الإدخال في الطبقة البرمجية العليا، ثم بعد ذلك تنفيذ هذه الطلبات بالتنسيق مع وحدة التحكم. فمثلاً قد يكون الطلب قراءة بلوك معين من القرص التخزيني فيتبع هذا البرنامج عدة خطوات للتحقق من كفاءة القرص التخزيني. موضع البلوك أولاً ، كما أنه يقوم بكتابة الأوامر في ذاكرة وحدة التحكم لحين معالجتها بواسطة الوحدة.

٣,١,٢. طبقة برنامج الإدخال والإخراج غير المعتمدة على خواص الوحدة

Device Independent Operating System Software

وتعتبر الطبقة البرمجية الأعلى من طبقة برنامج إدارة الوحده، بحيث تتكامل في مهامها مع برنامج طبقة إدارة الوحده. وتتلخص مهام هذه الطبقة في الاتي :

أ) تعتبر طبقة تداخل بين طبقة المستخدم وطبقة إدارة الوحدة.

ب) تسمية المكونات المادية Device Naming.

ج) حجز الذاكرة المؤقته في الوحدة Buffering.

د) حجز سعة تخزينية في وحدات الإدخال والإخراج.

هـ) تحديد الأخطاء.

٤,١,٢. طبقة برامج المستخدم

User Level I / O Software

وهي الطبقة الأخيرة من الطبقات العاملة والتي تقدم خدمات للمستخدم لاستخدام أجهـزة الإدخال والإخراج المختلفة في الحاسب، بحيث يقوم المستخدم بطلب الخدمة من بـرامج هذه الطبقة والتي بدورها تحول هذا الطلب إلى الطبقات البرمجية الأعلـى ثـم الأعلـى وهكذا إلى أن يتم تنفيذ الخدمة.

أسئلة تقويم ذاتي

١) عرف الفئتين: المعنونة وغير المعنونة.
٢) اشرح أهداف برمجيات الإدخال والإخراج.
٣) بالرسم، وضح مستويات برمجيات الإدخال و الإخراج.

٢. طرق إنجاز عمليات الإدخال والإخراج

يوجد ثلاث طرق أساسية لإنجاز عمليات الإدخال والإخراج وهي على النحو التالي:

١,٢. الإدخال / الإخراج المبرمج

Programmed I / O

تتلخص فكرة هذه الطريقة في جعل المعالج يقوم بالعمل كله ، فمثلاً إذا افترضنا أنـه توجد عمليه تريد طباعة سلسلة من الحروف تقوم العملية أولاً بتجميـع السلـسلة فـي مخزن

مؤقت Buffer، ثم بعد ذلك تستدعي نظام التشغيل الذي بدوره ينسخ المخزن المؤقـت Buffer الذي يحوي السلسلة المراد طباعتها في مصفوفة ، ثم يفحص اذا ما كانـت الطابعة متاحة حالياً ، فإذا لم تكن متاحه فإنه ينتظر حتي تصبح متاحة وعنـدها ينـسخ الحرف الأول إلى مسجل البيانات الخاص بالطابعة، وبعد طباعة، الحرف الأول ، يقوم

بفحصها مرة أخرى فإذا كانت الطابعه جاهزة لاستقبال الحرف التالي وذلك عن طريق قراءة حالة الطابعة بواسطة المسجل الخاص بذلك، ويصبح نظام التشغيل في انتظار الطابعة كي تصبح جاهزة مرة أخرى، وعندها يقوم بطباعه الحرف التالي وتستمر هذه الحلقة حتي طباعه السلسلة بأكملها ، ثم بعد ذلك يعود التنفيذ إلى عملية أخرى.

٢. ٢. الإدخال والإخراج المقاد بالمقاطعات

إذا كانت الطابعة تستطيع طباعة ١٠٠ حرف / ثانية مثلاً ، هذا يعني أن كل حرف يستغرق ١٠ ميلي ثانية لطباعته، أي أنه بعد كتابة كل حرف في مسجل بيانات الطابعة سيبقى المعالج خاملاً لمده ١٠ ميلي ثانية حتي يسمح له بإرسال الحرف التالي وهذا وقت كاف لإجراء تبديل وتشغيل عمليه أخرى أثناء فترة ١٠ ميلي ثانية عوضاً عن هدرها.

هذه الطريقة تتيح للمعالج تنفيذ عمليات أخرى أثناء انتظاره للطابعة كي تصبح جاهزة ، فيقوم المعالج بنسخ الحرف الأول إلى الطابعة وبعد ذلك يستدعي المجدول ويتم تنفيذ عملية. أخري وعندما تطبع الطابعة الحرف وتصبح جاهزة لاستقبال الحرف التالي يقوم نظام التشغيل بتوليد مقاطعه تؤدي إلى إيقاف العملية الحالية وتخزين حالتها.

فإذا لم يكن هنالك أي حروف أخرى للطباعة يقوم المعالج باستئناف عمل العملية المتوقفة والاّ فإنه يقوم بطباعة الحرف التالي ، ثم يرسل إشعاراً للمقاطعه ثم يعود إلى العملية التي كانت تعمل قبل حدوث المقاطعة وإلتى تتابع التنفيذ من النقطة التي توقفت عندها.

٣,٢. الإدخال والإخراج باستخدام تقنية DMA Using Direct Memory Access

هنالك مشكلة واضحة في الطريقة السابقة، وهي أن المقاطعه تحدث عند كل حرف ، وكما هو معلوم فإن المقاطعات تستهلك وقتاً، لذلك تعتبر الطريقة السابقة مضيعة لوقت المعالج. فالحل الأفضل هو استخدام تقنية DMA.

وتتلخص فكرة هذه الطريقة في جعل المتحكم Device Controller يغذي الطابعة بالحروف واحداً تلو الآخر دون إزعاج المعالج، أي أن يقوم المتحكم بالعمل عوضاُ عن المعالج الرئيسي.

الفائدة الكبرى عند أستخدام تقنية DMA (وهي عملية نقل المتحكم Device Controller للبيانات مباشرة من المخزن المؤقت Buffer الموجود بداخله إلى الذاكرة الرئيسية) هي تقليل عدد المقاطعات من مقاطعة لكل حرف إلى مقاطعه واحدة لكل مخزن مؤقت يطبع ، فإذا كانت الحروف المراد طباعتها كثيرة وكانت المقاطعات بطيئة يؤدي استخدام تقنية DMA إلى تحسين كبير في الأداء.

وأما إذا كان متحكم DMA غير قادر على قيادة الجهاز بسرعته القصوى، أو إذا لم يكن للمعالج عمل آخر سوى انتظار مقاطعة DMA فإن أستخدام إحدى الطريقتين السابقتين قد يكون أفضل من استخدام تقنية DMA.

يعتبر القرص التخزيني من أهم وحدات الحاسب حيث يتم فيه تخزين البيانات وإخراجها منه عند الحاجة ، لذلك فهي تحتاج إلى اهتمام وإدارة من قبل نظام التشغيل، وخاصة في أنظمة التشغيل المتعددة المستخدمين، حيث يكثر استخدام هذه الوحدة.

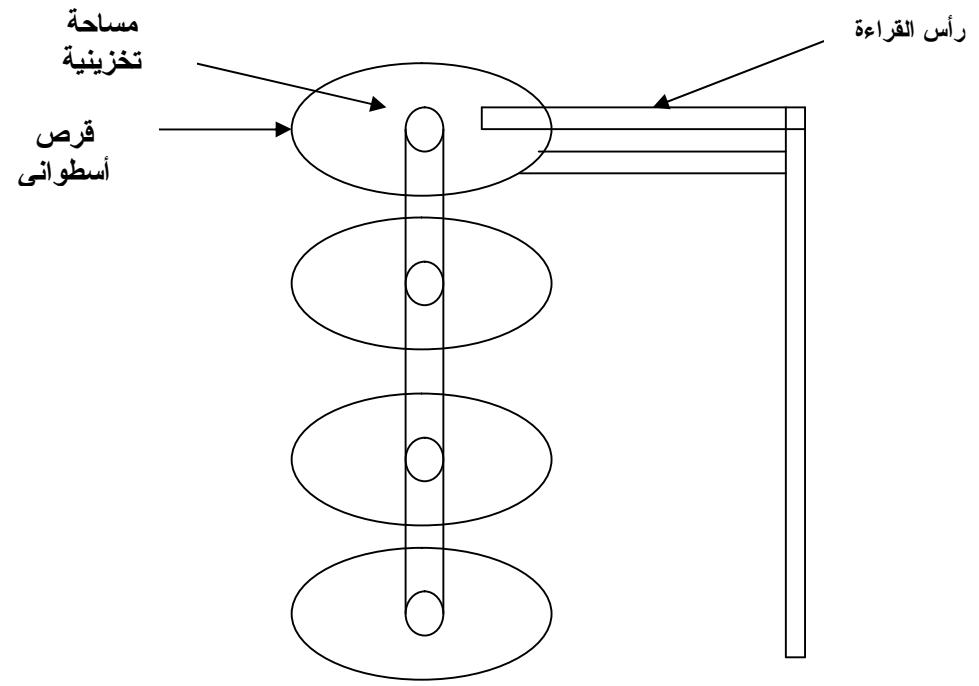
أسئلة تقويم ذاتي

١- اشرح طريقة الإدخال/ الإخراج المبرمج.
٢- لماذا نستخدم تقنية DMA ؟

٣. معمارية القرص التخزيني

يتكون القرص التخزيني من مجموعة أقراص أسطوانية، يحتوي كل قرص على سطحين ممغنطين، ويحتوى السطح الواحد علي مجموعة من المساحات الدائرية Tracksبحيث تقسم كل مساحة تخزينية إلى مجموعة من المربعات التخزينية Blocks ذات الحجم الثابت ، كما تسمى هذه المربعات التخزينية فيزيائياً بالمقاطع التخزينية Sectors.

وتثبت هذه الأقراص المغنطيسية بواسطة رأس ميكانيكي عند كل قرص كما يمكن أن تكون هنالك عده رؤوس قراءة لتساعد في عملية القراءة أو الكتابة من قرص إلى آخر بسرعة.



الشكل (٣) معمارية القرص التخزيني

والان وبعد أن تحدثنا عن معمارية القرص التخزيني نتطرق إلى التحدث عن مكوناته المادية فيتكون القرص التخزيني من مكونين مادين هما :-

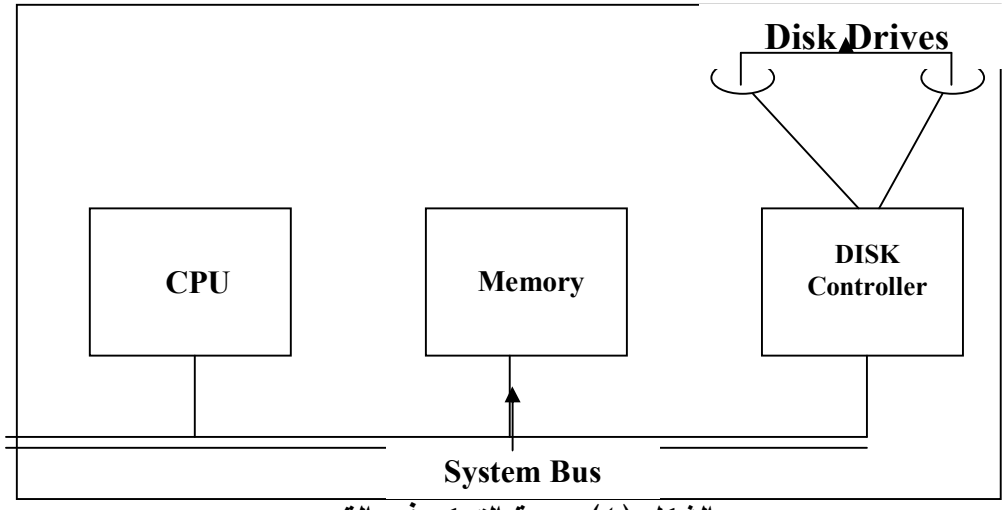
١,٣. سواقة القرص Disk Drives

وتعتبرسواقة القرص المكون المادي التخزيني، فهي تحتوي على القرص و الرؤوس والسواقة وكل الأجزاء الميكانيكية الأخرى.

٢,٣. وحدة التحكم في القرص Disk Controller

وتمثل الوحدة الإلكترونية التي توفر التفاعل المنطقي مع الحاسب بحيث تقوم هذه الوحدة بتلقي الأوامر من وحدة المعالجة المركزية. وبعد ذلك تقوم بترتيبها وتحويلها للسواقة.

فعلى سبيل المثال إذا أردنا استخراج معلومة من القرص التخزيني نحتاج إلى تحديد عدة أشياء وهي رقم السواقة والسطح والمساحة التخزينية Track بالإضافة إلى المقطع Block. ثم بعد ذلك يقوم رأس القراءة بالتحرك نحو المقطع، ويسمى الزمن المستغرق في تحديد المقطع **بزمن التحديد Seek Time** أما الزمن المستغرق حتى دوران القرص التخزيني حول القارئ المتحرك **بزمن الانتظار Latency Time**.



الشكل (٤) وحدة التحكم في القرص

أسئلة تقويم ذاتي

١) وضح مع الرسم معمارية القرص التخزيني.
٢) صف ما يحدث إذا أردنا استخراج نعلومة من القرص التخزيني.

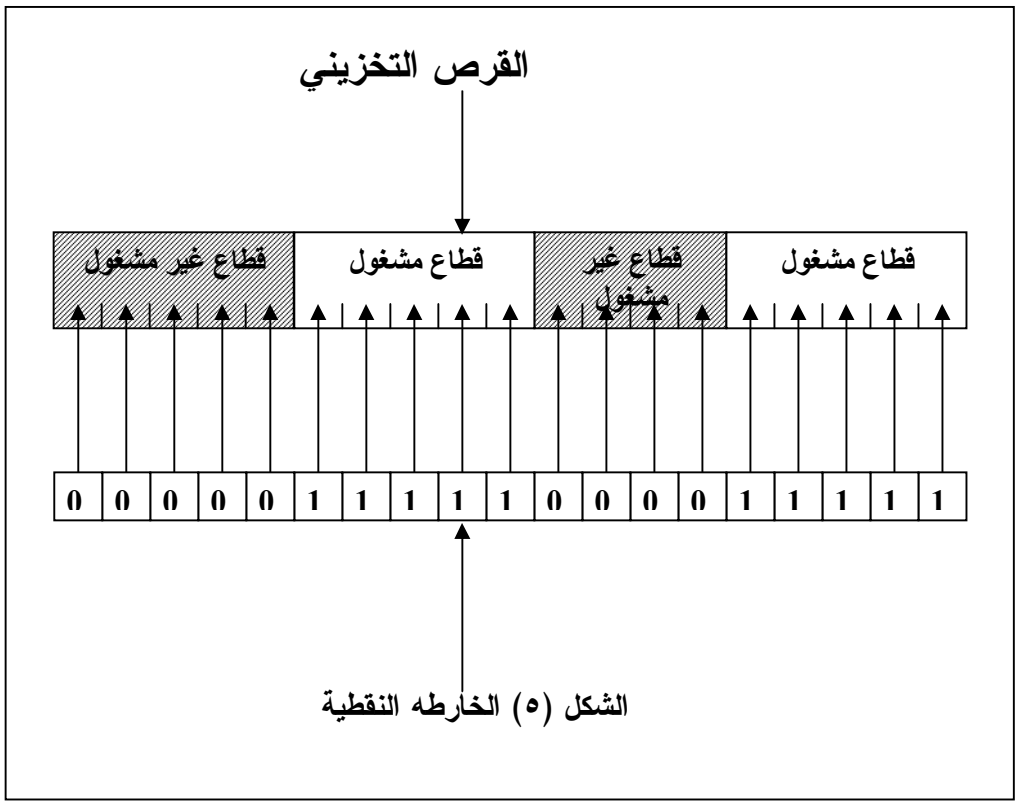
٤. إدارة الفراغات التخزينية

Free Space Management

تستوجب عملية التعامل مع القرص التخزيني عملية إدارة للفراغات في القرص حتى يتسنى حصر وتحديد المواقع الفارغة بصورة دائمة من قبل نظام التشغيل، ثم بعد ذلك يتم تحديد المساحة المطلوبة لملف محدد ، فهنالك عدة طرق لإدارة الفراغات التخزينية وهي على النحو التالي :

١,٤. الخارطة النقطية Bit Map

تتلخص هذه الطريقة بتقسيم القرص التخزيني إلى قطع صغيرة تسمى كل قطعة Allocation Bit، بحيث تكون هنالك خريطة من المربعات تمثل هذه القطع الصغيرة وذلك بتمثيل القطعة الصغيرة، المشغوله في القرص التخزيني بمربع يوجد بداخلة الرقم 1 في الخارطة النقطية، والقطعة غير المشغولة في القرص التخزيني بمربع يوجد بداخله الرقم 0 في الخارطه، وبذلك يمكن إدارة القرص عن طريق الخارطة النقطية.



الشكل (٥) الخارطه النقطية

٢,٤. القوائم المتصلة Linked List

في هذه الطريقة يتم ربط المربعات الفارغة بواسطة مؤشرات، بحيث يحتوي أول مربع فارغ على عنوان المربع الفارغ التالي له مباشرة، وهكذا حتى الوصول إلى جميع المربعات الفارغة.

٣,٤. التجميع Grouping

هي نفس فكرة القوائم المتصلة ولكن في هذه الطريقة يتم تجميع عناوين مجموعة من المربعات الفارغة في أول مربع فارغ، بحيث يحتوي آخر مربع فارغ على عنوان مجموعة أخري من المربعات الفارغة وهكذا ، وتتميز هذه الطريقة بسهولة الحصول على مجموعة من المربعات الفارغه.

أسئلة تقويم ذاتي

١) اشرح طريقة الخارطة النقطية في إدارة الفراغات التخزينية.
٢) ما الفرق بين طريقة القوائم المتصلة وطريقة التجميع ؟

٥. جدولة القرص التخزيني

Disk Scheduling

عندما تطلب عملية موقعاً تخزينياً تقوم العملية باستدعاء نظام التشغيل بواسطه مايسمى بنداء النظام System Call بحيث يتم تحديد عدة عوامل وهي :

أ) نوع العملية (إدخال أم إخراج).
ب) عنوان الموقع التخزيني (سواقة ، أسطوانية ، سطح ، قطاع).
ج) عنوان الذاكرة المراد تحميل البيانات إليها أو منها.
د) حجم البيانات المراد نقلها.

فهنالك عده خوارزميات يتبعها نظام التشغيل لجدولة وخدمة هذه الطلبات وسوف نتطرق إلى خوارزميتين هما :

٥,١. خوارزمية القادم أول يخدم أولاً

First Come First Service (FCFS)

فكرة هذه الخوارزمية مبنية على وضع كل الطلبات القادمة في فترة زمنية متقاربة في صف خدمة واحد ، وبعد ذلك يتم خدمة العملية حسب أولوية قدومها في الصف.

نلاحظ أن من عيوب هذه الخوارزمية أنه إذا كانت كل عملية تحتاج إلى مربعات تقع في أسطوانات تخزينية مختلفة عند بقية العمليات، مما يستدعي رأس القراءة إلى الأنتقال المستمر من أسطوانية الى إخرى عند خدمته كل عملية، مما يقتضي زيادة زمن التحديد.

٢,٥. خوارزمية أقل زمن تحديد أولاً

Short Seek Time First (SSTF)

واضح من اسم هذه الخوارزمية بأنها تعمل على تفادي عيب الخوارزمية السابقة ، فيتم خدمة العملية التي تستغرق أقل زمن تحديد.

فعلى سبيل المثال إذا كان رأس القراءة موجوداً في أسطوانية معينة، وهنالك عدةعمليات تطلب مواقع تخزينية مختلفة وفي أسطوانات مختلفة، عندها يبحث نظام التشغيل عن العملية التي تطلب معلومة موجودة في نفس الأسطوانية الحالية أو الأقرب وهكذا.

أسئلة تقويم ذاتي

١) ما العوامل التي يستدعيها النظام عندما تطلب عملية موقعاً تخزينياً ؟

٢) كيف تعمل خوارزمية القادم أولاً يخدم أولاً ؟

الخلاصة

عزيزي الدارس، بعد أن درسنا هذه الوحدة سنقوم بتلخيص ما درسناه فيها لتسهيل مراجعتها. بدأنا الوحدة بتقسيم أجهزة الإدخال والإخراج إلى فئتين وهما: الفئة المعنونة، والفئة غير المعنونة. وعرفنا أهداف برمجيات الإدخال والإخراج وهي:

- الاستقلالية عن الأجهزة.
- معالجة الأخطاء.
- التزامن.
- إمكانية التعامل مع الأجهزة في مستوى داخلي.

وتتحقق هذه الأهداف على أربعة مستويات وهي:

- طبقة برنامج خدمة المقاطعة.
- طبقة برنامج إدارة الوحدة.
- طبقة برنامج الإدخال و الإخراج غير المعتمدة على خواص الوحدة.
- طبقة برامج المستخدم.

وتوجد ثلاثة طرق أساسية لإنجاز عمليات الإدخال والإخراج وهي:

- الإدخال و الإخراج المبرمج.
- الإدخال والإخراج المقاد بالمقاطعات.
- الإدخال و الإخراج باستخدام تقنية DMA.

شرحنا معمارية القرص التخزيني حيث يتكون من مجموعة من الأقراص الأسطوانية. يحتوي كل قرص علي سطحين ممغنطين، ويحتوي السطح الواحد على مجموعة من المساحات الدائرية. كما يوجد عدة رؤوس للقراءة و الكتابة. وعرفنا وحدة التحكم في القرص وهي التي توفر التفاعل المنطقي مع الحاسب.

عملية التخزين تستوجب عملية لإدارة الفراغات التخزينية. وذكرنا ثلاثة طرق منها:

- الخارطة النقطية.

- القوائم المتصلة.
- التجميع.

هناك خوارزميات يتبعها النظام لجدولة وخدمة العمليات منها:

- خوارزمية القادم أولاً يخدم أولاً.
- خوارزمية أقل زمن تحديد أولاً.

أتمنى عزيزي الدارس أن تكون قد أفدت من هذه الوحدة، وهي تحتاج منك الرجوع إلى المراجع والمصادر لزيادة فهمها والتعرف على المعرفة الحديثة في مجـــال برمجيـــات الإدخـــــــــــــــال والإخـــــــــــــــراج، وفقـــــــــــــــك الله.

مسرد المصطلحات

- **فئة معنونة Block Devices**

أجهزة يمكن تخزين المعلومات بها في كتل ثابتة الحجم.

- **فئة غير معنونة Character Devices**

أجهزة ترسل وتستقبل سلسلة من الإشارات ولا يمكن تخزين بيانات بها.

- **استقلالية الأجهزة Device Independence**

إمكانية كتابة برنامج يستطيع الوصول إلى أي جهاز إدخال وإخراج دون تحديد نوع الجهاز بشكل مسبق.

- **معالجة الأخطاء Error Handling**

أحد مهام برمجيات الإدخال و الإخراج.

- **سواقة القرص Disk Drivers**

المكون المادي التخزيني للقرص.

- **زمن التحديد Seek Time**

الزمن المستغرق في تحديد المقطع.

- **زمن الانتظار Latency Time**

الزمن المستغرق حتى دوران القرص التخزيني حول القارئ المتحرك.

المراجع

١-يمان اللبني وأسامة العبد الله، تصميم وتنفيذ نظم التشغيل الحديثة، شعاع للنشر والعلوم، سوريا،حلب ٢٠٠٥م.

٢-ج آرتشر هاريس (ترجمة أمين أيوبي)، أنظمة تشغيل الحاسوب، أكاديمياً، بيروت ٢٠٠٢م.

٣-عبد الرءوف الحلاق ومنتصر خاطر، أنظمة التشغيل، منشورات جامعة القدس المفتوحة، ١٩٩٦م.

٤-محمد أحمد فكرين، نظم تشغيل الحاسبات، دار المريخ ١٩٩٦م

٥- Silberschatz, A. and Galvin, P. B. , "Operating System Concepts", Fifth edition Addison-Wesley, Reading, MA, 199٧.

6- Tanenbaum, Andrew S. , "Modern Operating Systems", Prentice-Hall, Englewood Cliffs, NJ, 1992.

http://www. personal. kent. edu/~rmuhamma/OpSystems/os. html
http://www. cag. lcs. mit. edu/~rinard/osnotes/
http://williamstallings. com/OS4e. html