Operating Systems

Lecture # 1

Department of Computer

4th Class

Introduction to Operating Systems

By

Dr. Ahmed Khudhair Abbas

Collage of education for pure science

Computer department

المقدمة Introduction

يتكون نظام الحاسب الآلي من معالج أو عدة معالجات بالإضافة إلي الذاكرة الرئيسية والأقراص والطابعات ولوحة المفاتيح وجهاز العرض ومحولات الشبكة والتي تسمي مجتمعه بالمكونات المادية Hardware , وتشكل هذه المكونات مجتمعه نظاماً معقداً في التعامل , مما أستوجب كتابة برامج تتحكم في أدارة جميع هذه المكونات وتستخدمها استخداما صحيحاً , وتسمي هذه البرامج ببرامج النظام والتي من أهم وظائفها أدارة جميع هذه الأجهزة (المكونات المادية) بالإضافة إلي تقديم واجهة بسيطة للمستخدم لكي يتمكن من التعامل مع المكونات المادية.

ونظام التشغيل Operating System : هو مجموعة من البرمجيات المسؤولة عن إدارة الموارد(عتاد الحاسوب Hardware، وبرمجيات الحاسوب Software)، ويمثل وسيط بين المستخدم، وعتاد الحاسوب، وبتعريف آخر يمثل نظام التشغيل جسر لتشغيل برامج المستخدم، ويقوم بالمهام الأساسية مثل: إدارة وتخصيص موارد الحاسوب (الذاكرة، القرص الصلب، الوصول للأجهزة الملحقة إلخ...)، وترتيب أولوية التعامل مع الأوامر، والتحكم في أجهزة الإدخال، والإخراج مثل :لوحة المفاتيح، وكذلك لتسهيل التعامل مع الشبكات، وإدارة الملفات

طبقات الحاسب Computer layers

الطبقة الأولى: طبقة المكونات المادية Hardware

وتنقسم إلي:

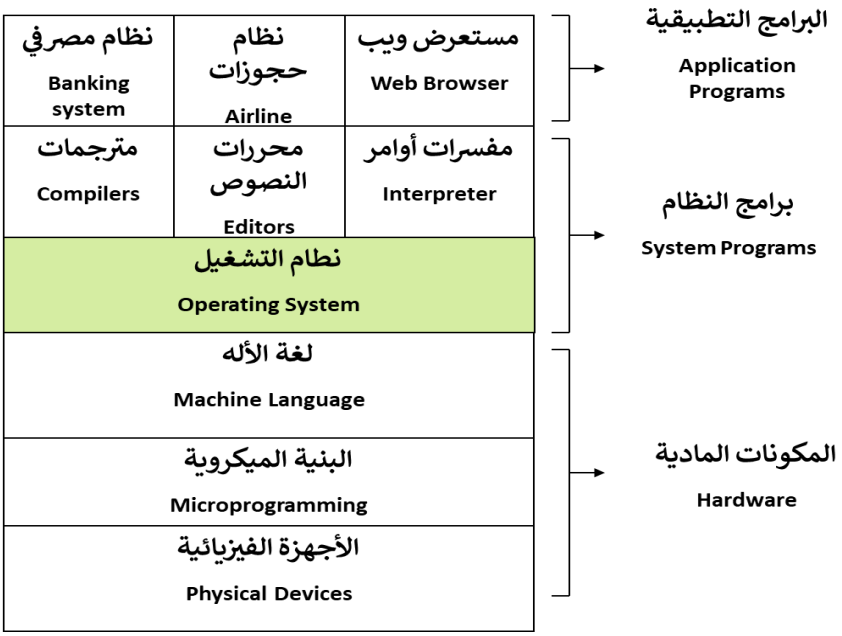
1. **المستوي الأول** الذي يتألف من الأجهزة الفيزيائية (شرائح دارات متكاملة وأسلاك ومزودات طاقة).
2. **المستوي الثاني** والذي يشتمل على البنية الميكروية والتي تجمع الأجهزة الفيزيائية مع بعض المسجلات الداخلية في المعالج وذلك، لتنفيذ مجموعة من التعليمات.
3. **المستوي الثالث** والذي يحتوي على لغة الآلة

الطبقة الثانية: هي طبقة برامج النظام Software

والتي تحتوي على نظام التشغيل والذي يهدف إلى إخفاء جميع التعقيدات، التي تظهر عند التعامل مع المكونات الفيزيائية وذلك من خلال مجموعة من الايعازات 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.

الطبقة الثالثة: هي طبقة البرامج التطبيقية Applications

وهي برامج تعمل في نمط المستخدم (User Mode) لكنها تساعد نظام التشغيل على القيام بالمهام



طبقات نظام الحاسب الالي

أنواع أنظمة التشغيل: OS Types

1. أنظمة التشغيل الأجهزة الكبيرة :Main Frame

تستخدم كخوادم ويب متطورة وخوادم لمواقع التجارة الالكترونية الواسعة. عموماً تهتم أنظمة تشغيل الأنظمة الكبيرة بمعالجة عدة مهام في وقت واحد وتحتوي هذه المهام في الغالب على كمية هائلة من عمليات الإدخال والإخراج بالإضافة إلى ذلك فان تلك الأنظمة تقدم نموذجياً لثلاثة أنواع من الخدمات الدفعية Batch و معالجة المناقلات Transaction processing ومشاركة الزمن Time sharing.

2. أنظمة تشغيل الخوادم: Servers

يأتي هذا النوع من أنظمة التشغيل في المرتبة الثانية والتي تعمل على المخدمات وهي عادة ما تتكون من أجهزة شخصية كبيرة جدا أو محطات عمل أو أجهزة كبيرة Main frame كما أنها تخدم عدة مستخدمين في نفس الوقت على الشبكة وتسمح للمستخدمين بالمشاركة في الموارد المادية والبرمجية. بالإضافة إلى أنها تقدم خدمات طباعة الملفات وخدمات الويب، كما أن مواقع الويب تستخدم المخدمات لتخزين صفحات الويب.

4. أنظمة تشغيل المعالجات المتعددة: Multiprocessors

من الطرق الشائعة للحصول على طاقة حسابيه كبيرة توصيل عده معالجات CPU في نظام واحد وتسمي هذه الأنظمة أما بالحواسيب المتوازية Parallel Computers أو الحواسيب المتعددة Multi Computers أو المعالجات المتعددة Multi-Processor وذلك حسب طريقة أتصال 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.

4. أنظمة تشغيل الحواسيب الشخصية: Personal Computers

تنحصر مهمتها في تقديم واجهة جيده للمستخدم. كما تستخدم هذه الأنظمة بشكل واسع لمعالجة النصوص والجداول الممتدة والوصول للأنترنت ومن الأمثلة الشهيرة لهذا النوع هو نظام Linux & Windows

5. أنظمة التشغيل المضمنة: Embedded Systems

تعمل مع حواسيب تتحكم بأجهزة لا تصنف بشكل عام علي أنها حواسيب مثل أجهزة التلفاز، أفران الميكروويف والهواتف النقالة . كما أنها تمتاز في غالب الأحيان بنفس مزايا أنظمة الزمن الحقيقي لكنها لها قيود أخري بالنسبة للحجم ومتطلبات الذاكرة والقوة الحسابية مما يجعلها مميزة

6. أنظمة تشغيل الزمن الحقيقي: Real Time System

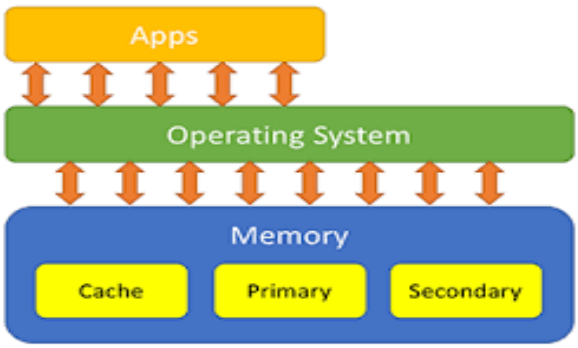
يمتاز هذا النوع بأنه يعتمد على الزمن كوسيط أساسي(أنظمة التحكم بالعمليات الصناعية), تقوم حواسيب الزمن الحقيقي بجمع جميع البيانات عن عمليه الإنتاج ثم بعد ذلك تستخدمها للتحكم في ألآت المصنع.

مهام نظام التشغيل: Operating System Functions

1. إدارة الذاكرة Memory Management
2. إدارة العمليات Processor Management
3. إدارة الأجهزة Device Management
4. إدارة الملفات File Management
5. الأمنية Security
6. اكتشاف الأخطاء Error detecting

إدارة الذاكرة: Memory Management

تشير إدارة الذاكرة إلى إدارة الذاكرة الأساسية أو الذاكرة الرئيسية. الذاكرة الرئيسية هي مجموعة كبيرة من الكلمات أو البايت حيث يكون لكل كلمة أو بايت عنوانها الخاص.



توفر الذاكرة الرئيسية تخزينًا سريعًا يمكن الوصول إليه مباشرة بواسطة وحدة المعالجة المركزية. لذلك لكي يتم تنفيذ البرنامج، يجب أن يكون في الذاكرة الرئيسية.

يقوم نظام التشغيل بالأنشطة التالية لإدارة الذاكرة:

1- يحتفظ بمسارات الذاكرة الأساسية (Keeps tracks of primary memory) أي جزء منها يستخدمه وما الجزء غير المستخدم.

2- في البرمجة المتعددة (multiprogramming) يقرر نظام التشغيل العملية التي ستحصل على الذاكرة و متى و كم .

3- يخصص الذاكرة (Allocates the memory) عندما تطلبها العملية القيام بذلك.

4- يلغي تخصيص الذاكرة (De-allocates the memory) عندما لا تعود العملية بحاجة إليها أو عندما يتم إنهاؤها

إدارة المعالجات Processor Management

في بيئة البرمجة المتعددة، يقرر نظام التشغيل العملية التي يحصل عليها المعالج ومتى وكم من الوقت. تسمى هذه الوظيفة جدولة العملية. يقوم نظام التشغيل بالأنشطة التالية لإدارة المعالج:

1- يحتفظ بمسارات المعالج وحالة العملية (Keeps tracks of processor and status of process) يُعرف البرنامج المسؤول عن هذه المهمة باسم مراقب حركة المرور.

2- يخصص المعالج وحدة المعالجة المركزية (CPU) لعملية ما (Allocates the processor (CPU) to a process)

3- يلغي تخصيص المعالج (De-allocates processor) عندما لا يكون المعالج مطلوبًا.

ادارة الجهاز Device Management

يقوم نظام التشغيل بإدارة اتصال الجهاز عبر برامج التشغيل الخاصة بهم. يقوم نظام التشغيل بالأنشطة التالية لإدارة الجهاز:

1- يحتفظ بتتبع جميع الأجهزة (Keeps tracks of all devices) يُعرف البرنامج المسؤول عن هذه المهمة باسم وحدة التحكم I / 0.

2- يقرر أي عملية يحصل عليها الجهاز ومتى وكم من الوقت (which process gets the device)

3- يخصص الجهاز بطريقة فعالة (Allocates the device in the efficient way)

4- يزيل تخصيص الأجهزة (De-allocates devices)

إدارة الملفات File Management

عادة ما يتم تنظيم نظام الملفات في دلائل لتسهيل التنقل والاستخدام. قد تحتوي هذه الأدلة على ملفات وتوجيهات أخرى. يقوم نظام التشغيل بالأنشطة التالية لإدارة الملفات:

-1 يتتبع المعلومات والموقع والاستخدامات والحالة وما إلى ذلك (Keeps track of information, location, uses, status etc) غالبًا ما تُعرف المرافقات الجماعية باسم نظام الملفات.

-2 يقرر من يحصل على الموارد (Decides who gets the resources)

-3 يخصص الموارد (Allocates the resources)

-4 يزيل تخصيص الموارد (De-allocates the resources)

الأنشطة الهامة الأخرى (Other Important Activities):

1. الأمان **Security** : عن طريق كلمة المرور وتقنيات أخرى مماثلة ، تمنع الوصول غير المصرح به إلى البرامج والبيانات.
2. التحكم في أداء النظام **Control over system performance**: تسجيل التأخيرات بين طلب الخدمة والاستجابة من النظام.
3. حساب المهمة **Job accounting** : تتبع الوقت والموارد المستخدمة من قبل مختلف الوظائف والمستخدمين.
4. أدوات الكشف عن الأخطاء **Error detecting aids** : إنتاج عمليات تفريغ وتتبع ورسائل الخطأ وغيرها من أدوات تصحيح الأخطاء واكتشاف الأخطاء.
5. التنسيق بين البرامج والمستخدمين الآخرين **Coordination between other software and users** : التنسيق وتخصيص المجمعين Accumulator والمترجمين الفوريين Interpreters والبرامج الأخرى لمختلف مستخدمي أنظمة الكمبيوتر.

Operating Systems

Lecture # 2

Department of Computer

4th Class

Computer System Architecture

By

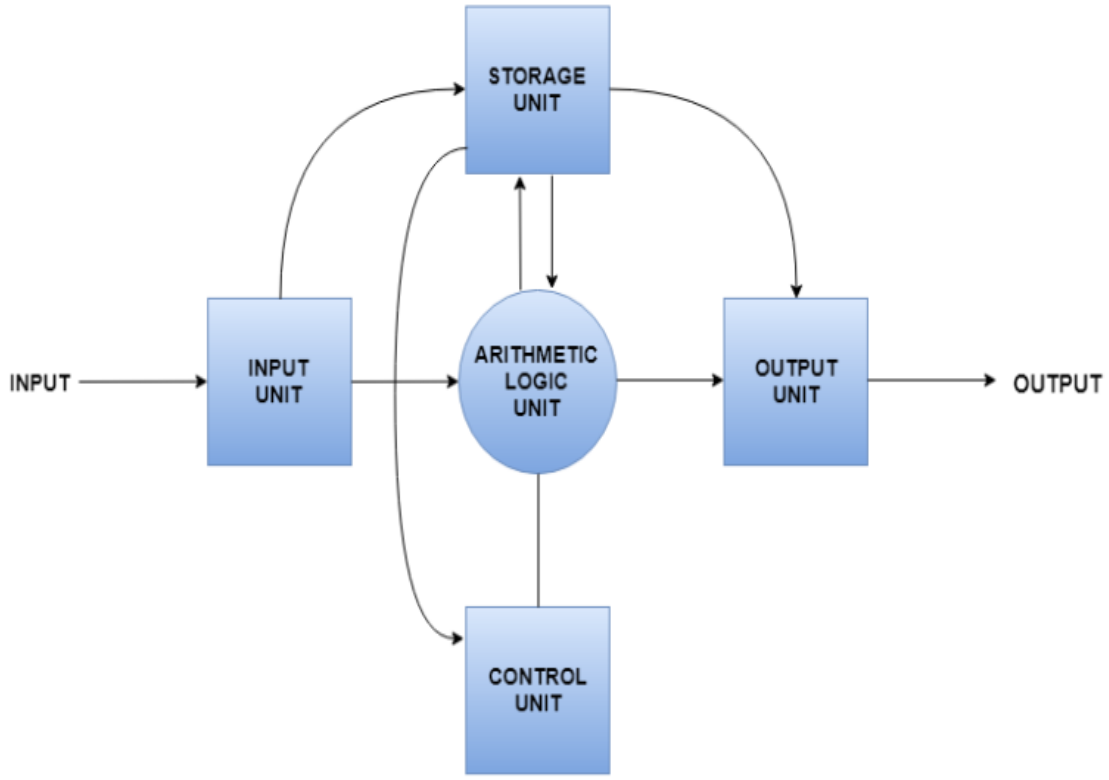
Dr. Ahmed Khudhair Abbas

Collage of education for pure science – Computer department

هيكلية نظام الحاسب Computer architecture

نظام الكمبيوتر هو في الأساس آلة تبسط المهام المعقدة وتزيد الأداء إلى أقصى حد وتقلل من التكاليف بالإضافة إلى استهلاك الطاقة. المكونات المختلفة في بنية نظام الكمبيوتر هي وحدة الإدخال ووحدة الإخراج ووحدة التخزين ووحدة المنطق الحسابي ووحدة التحكم وما إلى ذلك.

الرسم البياني الذي يوضح تدفق البيانات بين هذه الوحدات هو كما يلي :



تنتقل بيانات الإدخال من وحدة الإدخال إلى .ALU وبالمثل تنتقل البيانات المحسوبة من ALU إلى وحدة الإخراج. تنتقل البيانات باستمرار من وحدة التخزين إلى ALU وتعود مرة أخرى. وذلك لأن البيانات المخزنة يتم حسابها قبل تخزينها مرة أخرى. تتحكم وحدة التحكم في جميع الوحدات الأخرى بالإضافة إلى بياناتها.

- **وحدة الإدخال Input Unit**

توفر وحدة الإدخال البيانات لنظام الكمبيوتر من الخارج. لذلك فهو يربط بشكل أساسي البيئة الخارجية بالكمبيوتر. يأخذ البيانات من أجهزة الإدخال ويحولها إلى لغة الآلة ثم يقوم بتحميلها في نظام الكمبيوتر وتعتبر لوحة المفاتيح والماوس وما إلى ذلك هي أجهزة الإدخال الأكثر استخدامًا.

- **وحدة الإخراج Output Unit**

توفر وحدة الإخراج نتائج عمليات الكمبيوتر للمستخدمين أي أنها تربط الكمبيوتر بالبيئة الخارجية. معظم بيانات الإخراج هي شكل صوت أو فيديو او نص . أجهزة الإخراج المختلفة هي الشاشات والطابعات ومكبرات الصوت وسماعات الرأس وما إلى ذلك.

- **وحدة التخزين Storage Unit**

تحتوي وحدة التخزين على العديد من مكونات الكمبيوتر التي تُستخدم لتخزين البيانات. يتم تقسيمها تقليديًا إلى تخزين أساسي وتخزين ثانوي. يُعرف التخزين الأساسي أيضًا باسم الذاكرة الرئيسية وهي الذاكرة التي يمكن الوصول إليها مباشرة بواسطة وحدة المعالجة المركزية. لا يمكن الوصول إلى وحدة التخزين الثانوية أو الخارجية مباشرة بواسطة وحدة المعالجة المركزية. يجب إحضار البيانات من التخزين الثانوي إلى وحدة التخزين الأساسية قبل أن تتمكن وحدة المعالجة المركزية من استخدامها. يحتوي التخزين الثانوي على كمية كبيرة من البيانات بشكل دائم.

- **وحدة المنطق الحسابية Arithmetic Logic Unit**

يتم تنفيذ جميع العمليات الحسابية المتعلقة بنظام الكمبيوتر بواسطة وحدة المنطق الحسابي. يمكنه إجراء عمليات مثل الجمع والطرح والضرب والقسمة وما إلى ذلك. تنقل وحدة التحكم البيانات من وحدة التخزين إلى وحدة المنطق الحسابي عند الحاجة إلى إجراء العمليات الحسابية. تشكل وحدة المنطق الحسابي ووحدة التحكم معًا وحدة المعالجة المركزية.

- **وحدة التحكم Control Unit**

تتحكم هذه الوحدة في جميع الوحدات الأخرى لنظام الكمبيوتر ومن ثم تُعرف باسم الجهاز العصبي المركزي. ينقل البيانات عبر الكمبيوتر كما هو مطلوب بما في ذلك من وحدة التخزين إلى وحدة المعالجة المركزية والعكس صحيح. تحدد وحدة التحكم أيضًا كيفية تصرف الذاكرة وأجهزة الإدخال والإخراج ووحدة المنطق الحسابي وما إلى ذلك.

الوصول المباشر للذاكرة Direct Memory Access (DMA)

الوصول المباشر للذاكرة (DMA) هو طريقة تسمح لجهاز الإدخال / الإخراج (I / O) بإرسال أو استقبال البيانات مباشرة إلى أو من الذاكرة الرئيسية متجاوزًا وحدة المعالجة المركزية لتسريع عمليات الذاكرة وتتم إدارة العملية بواسطة شريحة تعرف باسم وحدة تحكم.DMA (DMAC)

DMA Controller هو جهاز يسمح لأجهزة الإدخال / الإخراج بالوصول المباشر إلى الذاكرة بمشاركة أقل من المعالج. يحتاج جهاز التحكم DMA إلى نفس الدوائر القديمة للواجهة للتواصل مع وحدة المعالجة المركزية وأجهزة الإدخال / الإخراج. يوضح الشكل أدناه مخطط الكتلة لوحدة تحكم.DMA

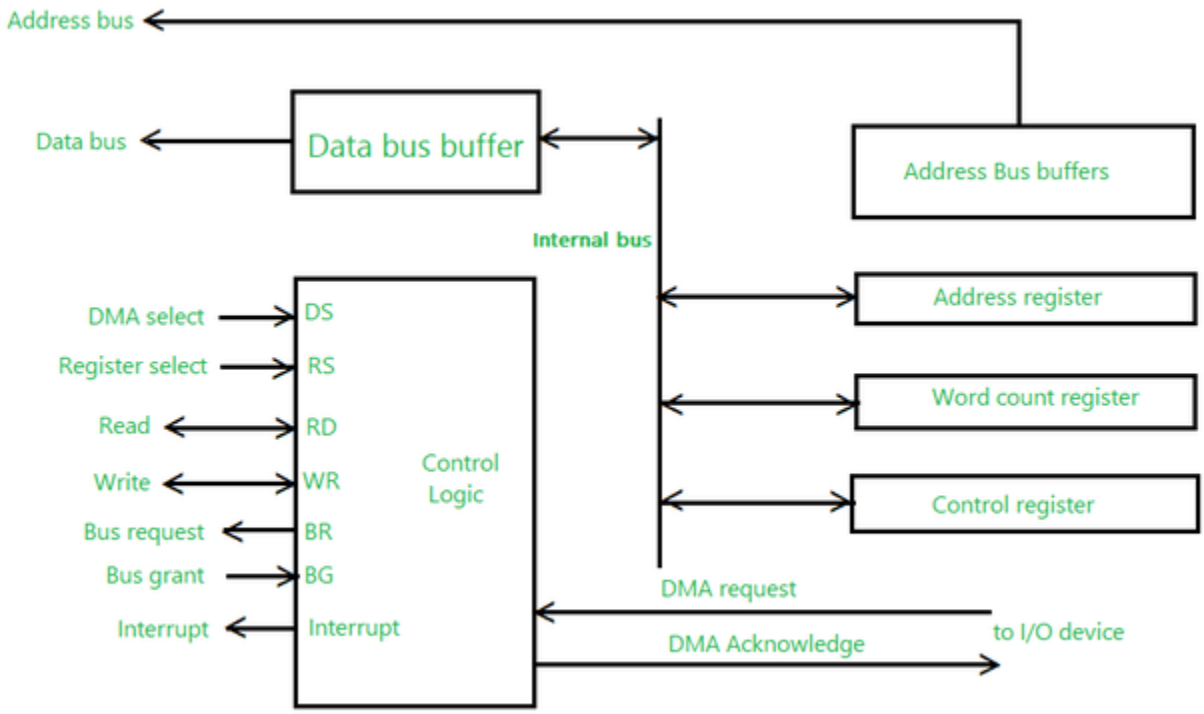
سجلات تحكم DMA

تحتوي وحدة تحكم DMA على ثلاثة سجلات :

- سجل العنوان Address register يحتوي على العنوان لتحديد الموقع المطلوب في الذاكرة.

- سجل عدد الكلمات Word count register يحتوي على عدد الكلمات المراد نقلها.
- سجل التحكم **Control register** يحدد وضع النقل.

تظهر كافة السجلات في DMA إلى وحدة المعالجة المركزية كسجلات واجهة الإدخال / الإخراج. لذلك يمكن لوحدة المعالجة المركزية القراءة والكتابة في سجلات DMA تحت تحكم البرنامج عبر ناقل البيانات.



DMA Block Diagram

توضيح Explanation

تقوم وحدة المعالجة المركزية (CPU) بتهيئة DMA عن طريق إرسال المعلومات المعطاة عبر ناقل البيانات.

•عنوان بداية كتلة الذاكرة حيث تكون البيانات متاحة (للقراءة) أو مكان تخزين البيانات (للكتابة).

•يرسل أيضًا عدد الكلمات وهو عدد الكلمات في الذاكرة المراد قراءتها أو كتابتها.

•التحكم في تحديد طريقة النقل مثل القراءة أو الكتابة.

•عنصر تحكم لبدء نقل.DMA

بمعنى اخر انه يتم استخدام جزء محدد من الذاكرة لإرسال البيانات مباشرة من جهاز طرفي إلى اللوحة الأم دون إشراك المعالج الدقيق ، بحيث لا تتداخل العملية مع التشغيل الكلي للكمبيوتر. وفي أجهزة الكمبيوتر القديمة ، تم ترقيم أربع قنوات 0,1,2,3 الى DMA عندما تم تقديم 16-bit industry standard architecture (ISA) تمت إضافة القنوات

5 و 6 و 7. وكان ISA عبارة عن معيار ناقل كمبيوتر لأجهزة الكمبيوتر المتوافقة مع IBM ، مما يسمح للجهاز ببدء المعاملات (ناقل رئيسي) بسرعة أكبر. وتحتوي وحدة تحكم ISA DMA على 8 قنوات DMA ، كل واحدة منها مرتبطة بعنوان 16 بت.

بعدها تم استبدال ISA بمنفذ رسومات accelerated graphics port (AGP) وبطاقات peripheral component interconnect (PCI) وهي أسرع بكثير. وينقل كل DMA حوالي 2 ميغابايت من البيانات في الثانية وتُستخدم أدوات موارد نظام الكمبيوتر للاتصال بين الأجهزة والبرامج. الأنواع الأربعة لموارد النظام هي:

- عناوين I / O.
- عناوين الذاكرة.
- أرقام طلب المقاطعة.(IRQ)
- قنوات الوصول المباشر للذاكرة.(DMA)

تُستخدم قنوات DMA لتوصيل البيانات بين الجهاز المحيطي وذاكرة النظام. تعتمد جميع موارد النظام الأربعة على خطوط معينة في الحافلة. تُستخدم بعض الخطوط على الناقل لـ IRQs ، وبعضها للعناوين (عناوين الإدخال / الإخراج وعنوان الذاكرة) وبعضها لقنوات.DMA

تمكّن قناة DMA الجهاز من نقل البيانات دون تعريض وحدة المعالجة المركزية لحمل عمل زائد. بدون قنوات DMA ، تنسخ وحدة المعالجة المركزية كل جزء من البيانات باستخدام ناقل طرفي من جهاز الإدخال / الإخراج. يشغل استخدام ناقل طرفي وحدة المعالجة المركزية أثناء عملية القراءة / الكتابة ولا يسمح بتنفيذ أي أعمال أخرى حتى تكتمل العملية.

باستخدام DMA ، يمكن لوحدة المعالجة المركزية معالجة المهام الأخرى أثناء إجراء نقل البيانات. يبدأ نقل البيانات أولاً بواسطة وحدة المعالجة المركزية ويمكن نقل كتلة البيانات من وإلى الذاكرة بواسطة DMAC بثلاث طرق:

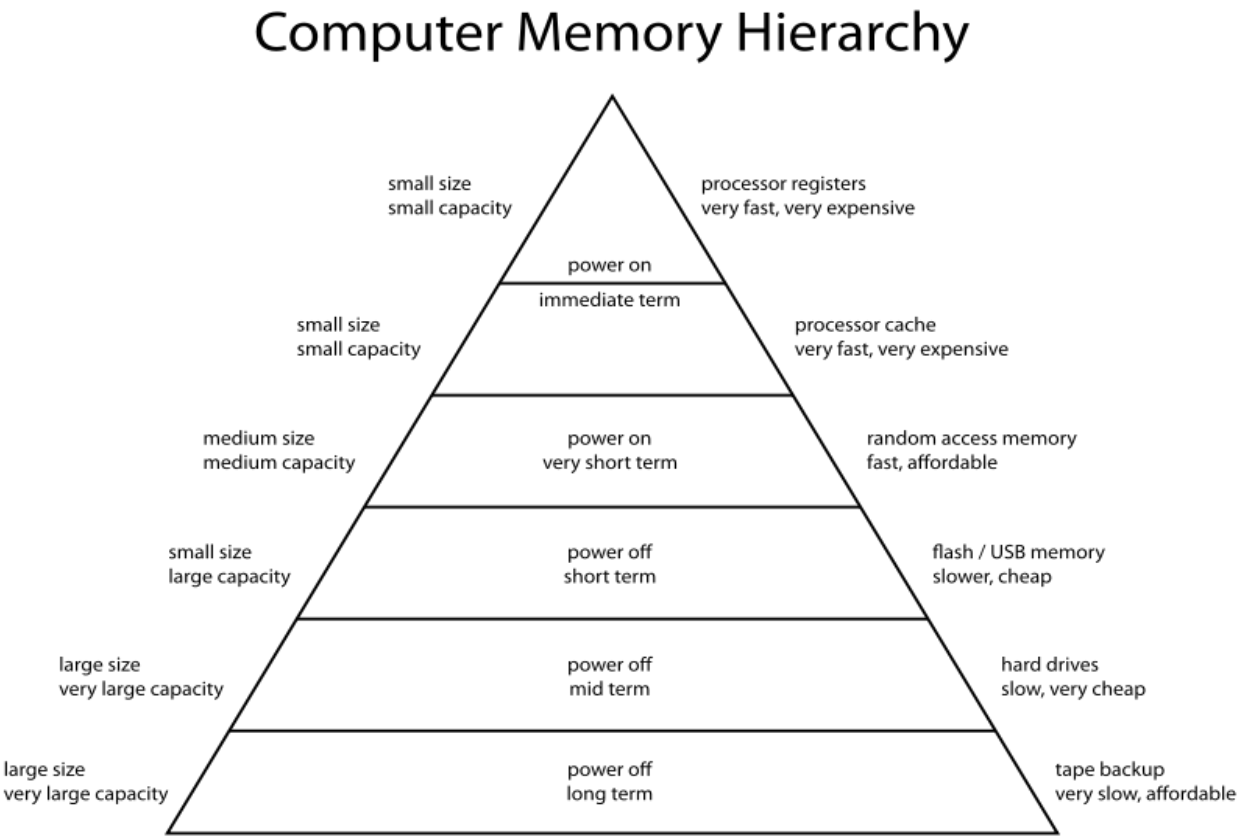
- **Burst mode :** لا يتم تحرير ناقل النظام إلا بعد اكتمال نقل البيانات.
- **Cycle stealing :** أثناء نقل البيانات بين قناة DMA وجهاز الإدخال / الإخراج ، يتم التخلي عن ناقل النظام لبضع دورات على مدار الساعة حتى تتمكن وحدة المعالجة المركزية من أداء مهام أخرى. عند اكتمال نقل البيانات ، تتلقى وحدة المعالجة المركزية طلب مقاطعة من وحدة تحكم .DMA
- **Transparent mode** : يمكن لـ DMAC تولي مسؤولية ناقل النظام فقط عندما لا يطلبه المعالج.

هرمية الذاكرة Memory Hierarchy

في معمارية الحاسوب، تقسم هرمية ذاكرة تخزين الحاسوب إلى تسلسل هرمي وفقًا لوقت الاستجابة نظرًا لارتباط وقت الاستجابة والسعة ويمكن أيضًا تمييز المستويات عن طريق أدائها وتقنيات التحكم بها وتؤثر هرمية الذاكرة على الأداء في التصميم المعماري للحاسوب ويتطلب التصميم عالي الأداء مراعاة قيود هرمية الذاكرة أي حجم وقدرات كل مكون يمكن عرض كل مكون من المكونات المختلفة كجزء من هرمية الذاكرة بحيث يكون كل طرف أصغر وأسرع من الطرف الأعلى في التسلسل الهرمي للتقليل من الانتظار بمستويات أعلى ويستجيب المستوى الأدنى عن طريق ملئ الذاكرة المؤقتة ثم التأشير لتنشيط النقل.

توجد أربعة مستويات تخزين رئيسية:

- داخلي- سجلات المعالج وذاكرة التخزين المؤقت.
- رئيسي- نظام ذاكرة الوصول العشوائي وبطاقات وحدة التحكم.
- تخزين كمي متصل- تخزين ثانوي.
- تخزين إجمالي خارجي- التخزين الثلاثي والخارجي.



رسم بياني لهرمية ذاكرة الحاسوب

أوضاع نظام التشغيل Operating System Modes

يوجد وضعان للتشغيل في نظام التشغيل للتأكد من أنه يعمل بشكل صحيح. هذه هي وضع المستخدم User mode وضع النواة Kernel mode .

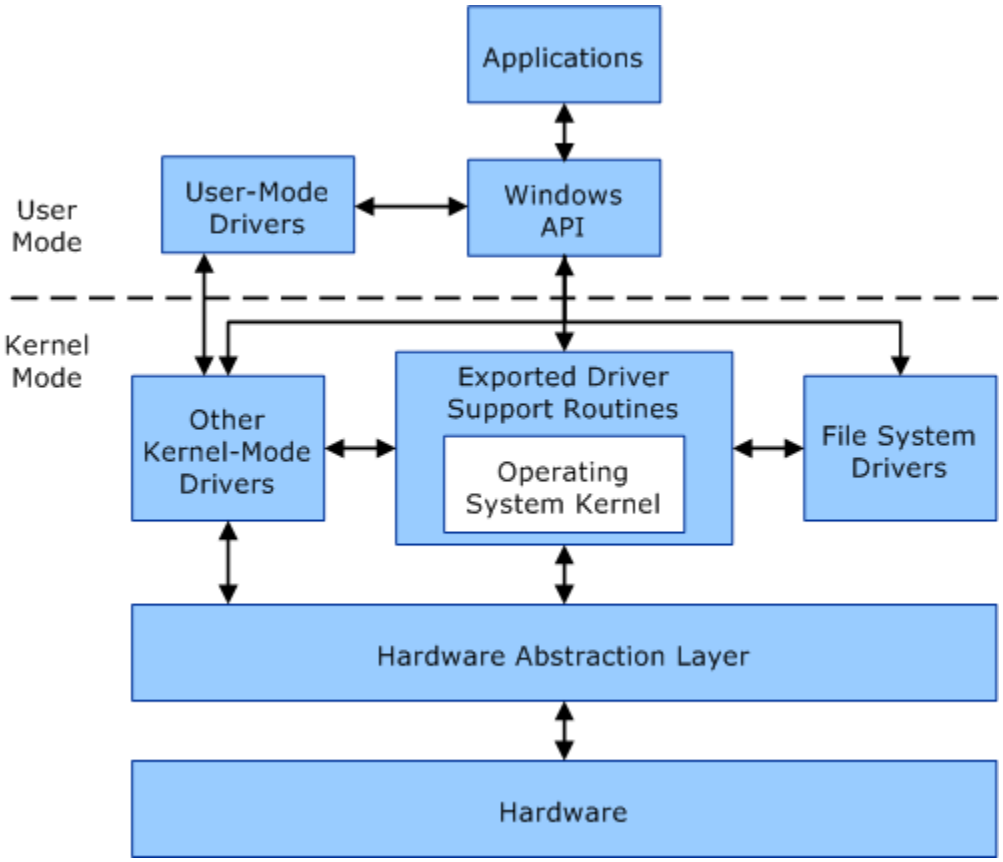
وضع المستخدم User mode

يكون النظام في وضع المستخدم عندما يقوم نظام التشغيل بتشغيل تطبيق مستخدم مثل التعامل مع محرر نصوص. يحدث الانتقال من وضع المستخدم إلى وضع kernel عندما يطلب التطبيق مساعدة نظام التشغيل أو تحدث مقاطعة أو مكالمة نظام ويتم ضبط بت الوضع على 1 في وضع المستخدم ويتم تغييره من 1 إلى 0 عند التبديل من وضع المستخدم إلى وضع.kernel

وضع لنواة Kernel mode

يبدأ النظام في وضع kernel عند بدء التشغيل وبعد تحميل نظام التشغيل يقوم بتنفيذ التطبيقات في وضع المستخدم. هناك بعض التعليمات المميزة التي لا يمكن تنفيذها إلا في وضع kernel. مثل تعليمات المقاطعة ، وإدارة المدخلات والمخرجات وما إلى ذلك وإذا تم تنفيذ التعليمات ذات التعليمات في وضع المستخدم User mode فهذا غير قانوني .

يتم تعيين بت الوضع على 0 في وضع kernel. يتم تغييره من 0 إلى 1 عند التبديل من وضع kernel إلى وضع المستخدم والصورة ادناه توضح الانتقال من وضع المستخدم إلى وضع kernel والعودة مرة أخرى .



يتم تنفيذ عملية المستخدم في وضع المستخدم حتى يتلقى استدعاء نظام.System call ثم يتم إنشاء اعتراض النظام وتعيين بت الأسلوب على الصفر. يتم تنفيذ استدعاء النظام في وضع kernel. بعد اكتمال التنفيذ ، يتم إنشاء اعتراض النظام مرة أخرى وتعيين بت الوضع إلى 1. يعود التحكم في النظام إلى وضع kernel ويستمر تنفيذ العملية.

هناك ضرورة كبير للوضع المزدوج (وضع المستخدم ووضع النواة) في نظام التشغيل ويمكن أن يتسبب عدم وجود وضع مزدوج ، أي وضع المستخدم User ووضع kernel في نظام التشغيل ، في حدوث مشكلات خطيرة حيث يمكن لبرنامج مستخدم قيد التشغيل محو نظام التشغيل بطريق الخطأ عن طريق الكتابة فوقه ببيانات المستخدم.

Operating Systems

Lecture # 3

Department of Computer

4th Class

Types Of Operating Systems

By

Dr. Ahmed Khudhair Abbas

Collage of Education for pure science

Computer department

يُقسَم نظام التشغيل من حيث قدرته على تشغيل أكثر من برنامج للمستخدم إلى قسمين :

- نظام متعدد المهام Multi-Tasking : هنا يُتيح للمستخدم التعامل مع أكثر من برنامج في الوقت نفسه.
- نظام أحادي المهام Single-Tasking : هذا النظام لا يُسمَح للمستخدم بتشغيل أكثر من برنامجٍ واحدٍ في الوقت ذاته.

كما تنقسم الأقسام أعلاه إلى عدة أقسام كذلك

Single User Single Task

أبسط أنواع نظم التشغيل تخدم مستخدما واحدا في الوقت الواحد وهو منفرد المهمة (Single-Tasking) بمعنى آخر يمكنه أن ينفذ برنامجا واحدا فقط في الوقت الواحد ومن الأمثلة عليه نظام MS-DOS

Single User Multi Tasks

هذا النظام لديه القدرة على تنفيذ أكثر من برنامج واحد بشكل متزامن، حيث تنتقل وحدة المعالجة المركزية (CPU) بين المهمات بسرعة كبيرة. ومن أمثلته Windows

Multi User Single task

يسمح هذا النظام لعدد من الأشخاص بتنفيذ كل منهم برنامجاً واحداً في الوقت نفسه. ويزود كل مستخدم بمحطة إدخال وإخراج تتصل مع الحاسوب المركزي ويسمى هذا التنظيم بنظام المشاركة الزمنية (Timesharing) وذلك لأن نظام التشغيل يأمر الحاسوب بالانتقال بسرعة كبيرة بين المستخدمين بعد إعطائهم فترات زمنية ثابتة لاستخدام CPU وتسمى هذه الفترات بالشرائح الزمنية (Time Slices) وهي قصيرة جداً بحيث يتوهم كل مستخدم أنه يمتلك الحاسوب بشكل كلي ومن امثلته Windows NT.

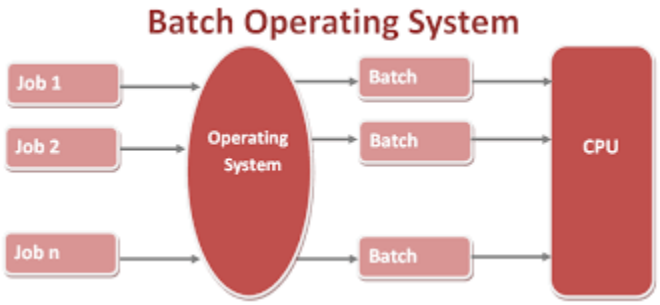
Multi User Multi Tasks

هو عباره عن نظم التشغيل الحديثة حيث تتيح للمستخدم تشغيل عدة برامج حتى إن كان تعدد المعالجات (CPU) غير كافية لذلك. تقوم نظم التشغيل بتوزيع وقت المعالج بين هذه البرامج بحيث يأخذ كل برنامج وقت محدد من المعالج من ثم يقوم بإيقاف مؤقت للبرنامج وإعطاء برنامج اخر هذا الوقت. هذا يعني ان خلال اللحظة الواحدة برنامج واحد يعمل على المعالج ونظام التشغيل يقوم بالتغيير بسرعة كبيرة جداً كأجزاء من الثانية. عملية توزيع وقت المعالج تسمى بالجدولة (scheduling) حيث يحتفظ نظام التشغيل بقائمة من البرامج التي قام المُستخدم بتشغيلها وتقوم عملية الجدولة بتوزيع وقت لكل برنامج موجود في هذه القائمة ليستفيد من المُعالج في هذا الوقت.

ويمكن تصنيف الأنواع المختلفة لأنظمة من حيث طريقة العمل وفيما يلي بعض الأنواع المهمة من أنظمة التشغيل الأكثر استخدامًا.

1. نظام التشغيل بالدفعات Batch operating system

لا يتفاعل مستخدمو نظام التشغيل الدفعي مع الكمبيوتر بشكل مباشر. يقوم كل مستخدم بإعداد وظيفته على جهاز غير متصل بالإنترنت مثل البطاقات المثقوبة وإرسالها إلى مشغل الكمبيوتر. لتسريع المعالجة ، يتم تجميع المهام ذات الاحتياجات المماثلة معًا وتشغيلها كمجموعة. وهكذا ، ويترك المبرمجون برامجهم مع المشغل ويقوم المشغل بعد ذلك بفرز البرامج إلى دفعات بمتطلبات مماثلة .



فيما يلي مشاكل أنظمة الدُفعات :

- عدم وجود تفاعل بين المستخدم والوظيفة.
- غالبًا ما تكون وحدة المعالجة المركزية خاملة ، لأن سرعات أجهزة الادخال / الاخراج الميكانيكية تكون أبطأ من وحدة المعالجة المركزية .
- من الصعب إعطاء الأولوية المطلوبة .

2. أنظمة تشغيل مشاركة الوقت Time-sharing operating systems

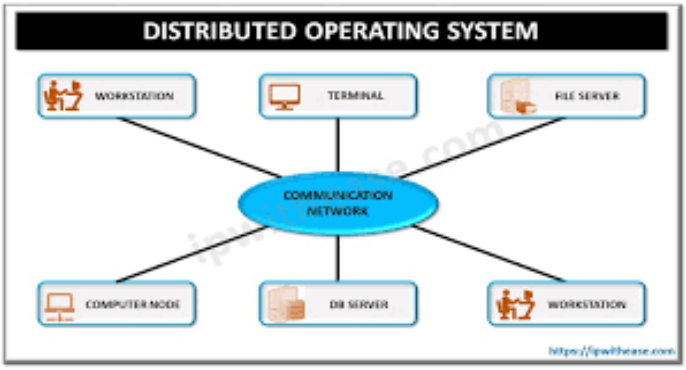
تعد مشاركة الوقت تقنية تمكن العديد من الأشخاص الموجودين في محطات طرفية مختلفة من استخدام نظام كمبيوتر معين في نفس الوقت. تعد مشاركة الوقت أو تعدد المهام امتدادًا منطقيًا للبرمجة المتعددة ويُطلق على وقت المعالج الذي يتم مشاركته بين عدة مستخدمين في وقت واحد اسم مشاركة الوقت.

يتم تنفيذ مهام متعددة بواسطة وحدة المعالجة المركزية (CPU) عن طريق التبديل بينها ، لكن التبديل يحدث كثيرًا. وبالتالي ، يمكن للمستخدم تلقي استجابة فورية . يقوم المعالج بتنفيذ كل برنامج مستخدم في دفعة قصيرة أو كمية من الحساب. هذا إذا كان n من المستخدمين موجودين ، يمكن لكل مستخدم الحصول على الوقت الكمي

3. نظام التشغيل الموزع Distributed operating System

تستخدم الأنظمة الموزعة معالجات مركزية متعددة لخدمة تطبيقات متعددة في الوقت الفعلي ومستخدمين متعددين. يتم توزيع وظائف معالجة البيانات بين المعالجات وفقًا لذلك يمكن للمرء أداء كل وظيفة بكفاءة أكبر. تتواصل المعالجات مع بعضها البعض من خلال خطوط اتصال مختلفة (مثل الموصلات عالية السرعة أو خطوط الهاتف). ويشار إلى هذه على أنها أنظمة موزعة.

قد تختلف المعالجات في النظام الموزع من حيث الحجم والوظيفة. يشار إلى هذه المعالجات باسم المواقع والعقد وأجهزة الكمبيوتر وما إلى ذلك .

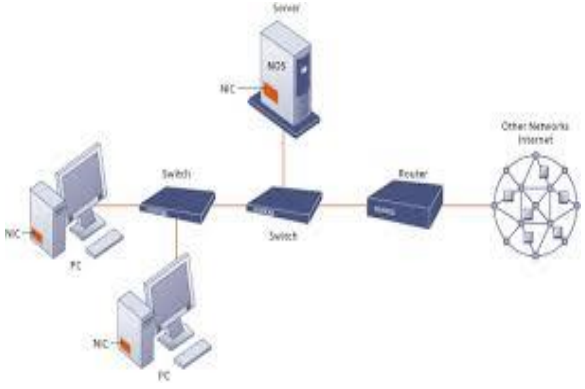


فيما يلي مزايا الأنظمة الموزعة:

- مع إمكانية مشاركة الموارد ، قد يتمكن المستخدم في موقع ما من استخدام الموارد المتاحة في موقع آخر.
- تسريع تبادل البيانات مع بعضها البعض
- إذا فشل أحد المواقع في نظام موزع ، فمن المحتمل أن تستمر المواقع المتبقية في العمل .
- خدمة أفضل للعملاء.
- تقليل الحمل على الكمبيوتر المضيف.
- الحد من التأخير في معالجة البيانات .

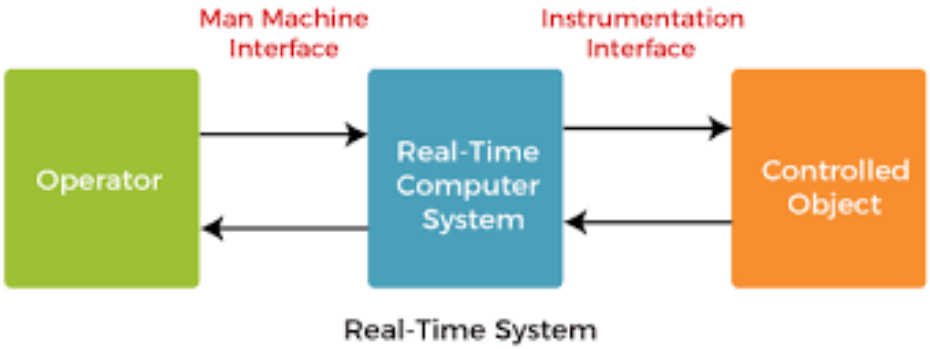
4. نظام تشغيل الشبكة Network operating System

يعمل نظام تشغيل الشبكة على خادم ويوفر للخادم القدرة على إدارة البيانات والمستخدمين والمجموعات والأمن والتطبيقات ووظائف الشبكات الأخرى. الغرض الأساسي من نظام تشغيل الشبكة هو السماح بالوصول المشترك للملفات والطابعة بين أجهزة كمبيوتر متعددة في شبكة ، وعادةً ما تكون شبكة محلية (LAN) أو شبكة خاصة أو شبكات أخرى. من أمثلة أنظمة تشغيل الشبكة Microsoft Windows Server , UNIX , Linux , Mac OS X



5. نظام التشغيل الوقت الحقيقي Real Time operating System

يُعرّف نظام الوقت الحقيقي بأنه نظام معالجة بيانات يكون فيه الفاصل الزمني المطلوب لمعالجة المدخلات والاستجابة لها صغيرًا جدًا لدرجة أنه يتحكم في البيئة. تكون المعالجة في الوقت الفعلي دائمًا عبر الإنترنت ويُطلق على الوقت الذي يستغرقه النظام للاستجابة لإدخال وعرض المعلومات المحدثة المطلوبة وقت الاستجابة لذا في هذه الطريقة يكون وقت الاستجابة صغير جدا. تُستخدم أنظمة الوقت الفعلي عندما تكون هناك متطلبات زمنية مهمة جدا لتشغيل المعالج أو تدفق البيانات ويمكن استخدام أنظمة الوقت الفعلي كجهاز تحكم في تطبيق مخصص. يحتوي نظام التشغيل في الوقت الفعلي على قيود زمنية محددة جيدًا وثابتة وإلا سيفشل النظام. على سبيل المثال ، التجارب العلمية وأنظمة التصوير الطبي وأنظمة التحكم الصناعية وأنظمة الأسلحة والروبوتات وأجهزة التحكم في الأجهزة المنزلية ونظام التحكم في الحركة الجوية وما إلى ذلك .



Real-Time System

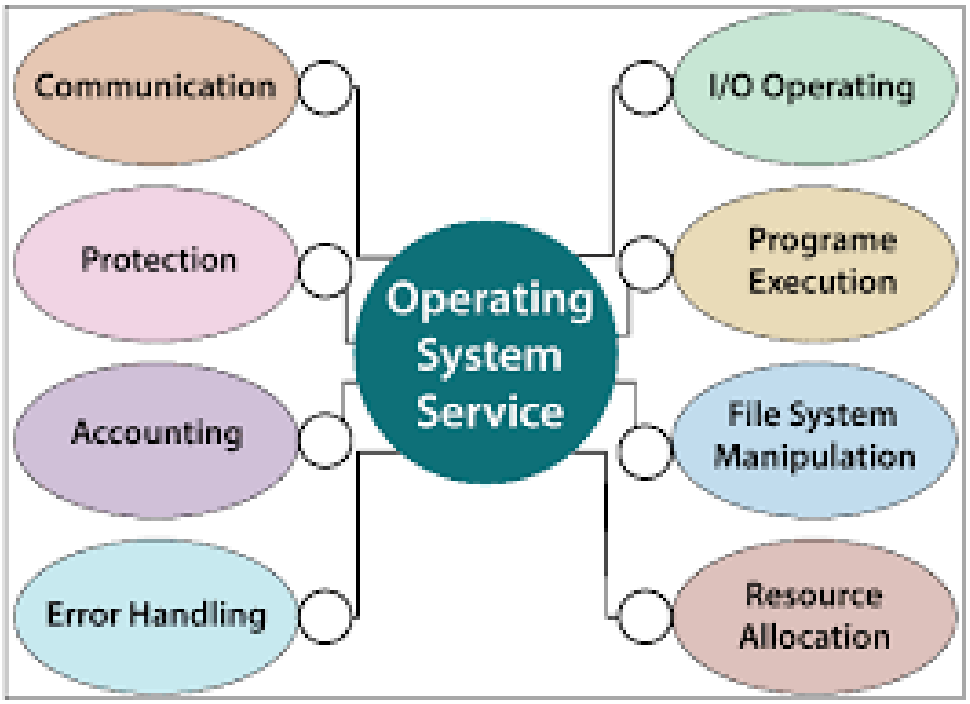
Operating Systems

Lecture # 4

Department of Computer

4th Class

Operating system services



By

Dr. Ahmed Khudhair Abbas

Collage of education for pure science - Computer department

وفر نظام التشغيل بيئة عمل لتنفيذ البرامج، فهو يقدم خدمات للبرامج ولمستخدِمي تلك البرامج. وتختلف هذه الخدمات من نظام تشغيل إلى آخر، بيد أنه يمكن تمييز بعض التصنيفات العامة لها، تجعل هذه الخدمات المبرمجين أكثر مقدرة على البرمجة وتسهل عليهم عملهم.

- تنفيذ البرنامج Program execution
- عمليات الادخال والاخراج Input/Output Operation
- التعامل مع نظام الملفات File System manipulation
- الاتصالات Communication
- اكتشاف الخطأ Error Detection
- تخصيص الموارد Resource Allocation
- الحماية Protection

تنفيذ البرنامج Program execution

يتعامل نظام التشغيل مع العديد من الأنشطة من برامج المستخدم إلى برامج النظام مثل التخزين المؤقت للطابعة وخوادم الأسماء وخادم الملفات وما إلى ذلك ويتم تنفيذ كل من هذه الأنشطة كعملية Process. وتتضمن العملية سياق التنفيذ الكامل (التعليمات البرمجية للتنفيذ ، والبيانات التي يجب معالجتها ، والسجلات ، وموارد نظام التشغيل قيد الاستخدام) .

فيما يلي الأنشطة الرئيسية لنظام التشغيل فيما يتعلق بإدارة البرنامج:

- تحميل البرنامج في الذاكرة.
- ينفذ البرنامج .
- يوفر آلية لمزامنة العملية.
- يوفر آلية لعملية الاتصال.
- يوفر آلية للتعامل مع المشاكل التي تحدث

عمليات الادخال والاخراج Input/Output Operation

يتكون نظام الادخال / الاخراج الفرعي من أجهزة الادخال / الاخراج وبرامج التشغيل المقابلة لها ويدير نظام التشغيل الاتصال بين المستخدم وبرامج تشغيل الجهاز .

فيما يلي الأنشطة الرئيسية لنظام التشغيل فيما يتعلق بعملية الادخال / الاخراج :

- تعني عملية الادخال / الاخراج قراءة أو كتابة عملية باستخدام أي ملف أو أي جهاز إدخال / إخراج محدد .
- قد يتطلب البرنامج أي جهاز ادخال / اخراج أثناء التشغيل .
- يوفر نظام التشغيل الوصول إلى جهاز الادخال / الاخراج المطلوب عند الحاجة .

نظام الملفات File System manipulation

يمثل الملف مجموعة من المعلومات ذات الصلة ويمكن للكمبيوتر تخزين الملفات على القرص (التخزين الثانوي)، لغرض التخزين على المدى الطويل. أمثلة قليلة على وسائط التخزين هي الشريط المغناطيسي والأقراص المغناطيسية ومحركات الأقراص الضوئية مثل الأقراص المضغوطة وأقراص .DVD كل من هذه الوسائط لها خصائصها الخاصة مثل السرعة والسعة ومعدل نقل البيانات وطرق الوصول إلى البيانات .

عادة ما يتم تنظيم نظام الملفات في أدلة لسهولة التصفح والاستخدام. قد تحتوي هذه الأدلة على ملفات وتوجيهات أخرى .

فيما يلي الأنشطة الرئيسية لنظام التشغيل فيما يتعلق بإدارة الملفات .

- يحتاج البرنامج إلى قراءة ملف أو كتابة ملف.
- يمنح نظام التشغيل الإذن للبرنامج للعمل في الملف.
- الإذن يختلف من قراءة فقط، للقراءة والكتابة، مرفوض وما الى ذلك.
- يوفر نظام التشغيل واجهة للمستخدم لإنشاء / حذف الملفات .
- يوفر نظام التشغيل واجهة للمستخدم لإنشاء / حذف الدلائل .
- يوفر نظام التشغيل واجهة لإنشاء نسخة احتياطية لنظام الملفات .

الاتصالات Communication

في حالة الأنظمة الموزعة التي هي عبارة عن مجموعة من المعالجات التي لا تشترك في الذاكرة أو الأجهزة الطرفية أو الساعة، يقوم نظام التشغيل بإدارة الاتصالات بين العمليات. عمليات متعددة مع بعضها البعض من خلال خطوط الاتصال في الشبكة. يتعامل نظام التشغيل مع استراتيجيات التوجيه والاتصال، ومشكلات الخلاف والأمن .

فيما يلي الأنشطة الرئيسية لنظام التشغيل فيما يتعلق بالاتصالات :

- غالبًا ما تتطلب عمليتان نقل البيانات بينهما .
- يمكن أن تكون العمليتان على جهاز كمبيوتر واحد أو على جهاز كمبيوتر مختلف ولكنهما متصلتان عبر شبكة الكمبيوتر.
- يمكن تنفيذ الاتصال بطريقتين إما عن طريق الذاكرة المشتركة أو عن طريق تمرير الرسائل .

اكتشاف الخطأ Error Detection

يمكن أن يحدث الخطأ في أي وقت وفي أي مكان. قد يحدث خطأ في وحدة المعالجة المركزية أو في أجهزة الادخال / الاخراج أو في أجهزة الذاكرة.

فيما يلي الأنشطة الرئيسية لنظام التشغيل فيما يتعلق بمعالجة الأخطاء:

- يبقى نظام التشغيل على علم بالأخطاء المحتملة باستمرار .
- يتخذ نظام التشغيل الإجراء المناسب لضمان الحوسبة الصحيحة والمتسقة.

تخصيص الموارد Resource Allocation

في حالة وجود بيئة متعددة المستخدمين أو متعددة المهام فان الموارد مثل الذاكرة الرئيسية ودورات وحدة المعالجة المركزية والملفات يتم تخصيص التخزين لكل مستخدم أو وظيفة.

فيما يلي الأنشطة الرئيسية لنظام التشغيل فيما يتعلق بإدارة الموارد:

- يدير نظام التشغيل جميع أنواع الموارد باستخدام أدوات الجدولة .
- يتم استخدام خوارزميات جدولة وحدة المعالجة المركزية لتحسين استخدام وحدة المعالجة المركزية .

الحماية Protection

بالنظر إلى أن أنظمة الكمبيوتر التي لديها عدة مستخدمين هي التنفيذ المتزامن لعمليات متعددة، فيجب حماية العمليات المختلفة من أنشطة بعضها البعض. تشير الحماية إلى آلية أو طريقة للتحكم في وصول البرامج أو العمليات أو المستخدمين إلى الموارد التي تحددها أنظمة الكمبيوتر .

فيما يلي الأنشطة الرئيسية لنظام التشغيل فيما يتعلق بالحماية :

- يضمن نظام التشغيل التحكم في الوصول إلى موارد النظام بالكامل .
- يضمن نظام التشغيل حماية أجهزة الادخال / الاخراج الخارجية من محاولات الوصول غير الصالحة.
- يوفر نظام التشغيل ميزة المصادقة لكل مستخدم عن طريق كلمة مرور.

Operating Systems

Lecture # 5

Department of Computer

4th Class

System Calls and Memory protection

By

Dr. Ahmed Khudhair Abbas

Collage of Education For Pure Science – Computer Department

استدعاءات النظام في نظام التشغيل System Calls in Operating System

استدعاء النظام هو وسيلة برامج المستخدم للتفاعل مع نظام التشغيل. يطلب البرنامج العديد من الخدمات ويستجيب نظام التشغيل من خلال استدعاء سلسلة من استدعاءات النظام لتلبية الطلب ويمكن كتابة استدعاء النظام بلغة التجميع أو لغة عالية المستوى مثل C أو .Pascal وان استدعاءات النظام هي وظائف محددة مسبقًا قد يستدعيها نظام التشغيل مباشرةً إذا تم استخدام لغة عالية المستوى.

ما هو استدعاء النظام What is a System Call

استدعاء النظام هو طريقة لبرنامج الكمبيوتر لطلب خدمة من نواة نظام التشغيل Kernel بمعنى ان استدعاء النظام هو طلب من برنامج الكمبيوتر إلى نواة نظام التشغيل.

تعمل واجهة برمجة التطبيقات Application Program Interface (API) على توصيل وظائف نظام التشغيل ببرامج المستخدم ويعمل كحلقة وصل بين نظام التشغيل والعملية Process مما يسمح للبرامج على مستوى المستخدم بطلب خدمات نظام التشغيل ولا يمكن الوصول إلى نظام kernel إلا باستخدام استدعاءات النظام واستدعاءات النظام مطلوبة لأية برامج تستخدم الموارد.

كيف يتم إجراء استدعاءات النظام How are system calls made

فيما يلي بعض الأمثلة على كيفية اختلاف استدعاء النظام من وظيفة المستخدم system call varies from a user function

1. قد تقوم وظيفة استدعاء النظام بإنشاء واستخدام عمليات kernel لتنفيذ المعالجة غير المتزامنة.
2. استدعاء النظام له سلطة أكبر من برنامج ويتم تنفيذ استدعاء نظام بامتياز من خلال وضع ال kernel في مجال الحماية
3. لا يُسمح لاستدعاءات النظام باستخدام المكتبات المشتركة أو أي برامج غير موجودة في مجال حماية.kernel
4. يتم تخزين برامج وبيانات استدعاءات النظام في ذاكرة kernel

لماذا تحتاج إلى استدعاءات النظام في نظام التشغيل ?Why do you need system calls in Operating System

1. يجب أن يطلب عندما يريد نظام الملفات إنشاء أو حذف ملف.
2. تتطلب اتصالات الشبكة استدعاءات النظام لإرسال واستقبال حزم البيانات.
3. قراءة ملف أو كتابته بحاجة إلى استدعاءات النظام.
4. الوصول إلى الأجهزة ، بما في ذلك الطابعة والماسح الضوئي بحاجة إلى استدعاءات نظام.
5. يتم استخدام استدعاءات النظام لإنشاء وإدارة عمليات جديدة.

كيف تعمل استدعاءات النظام How System Calls Work

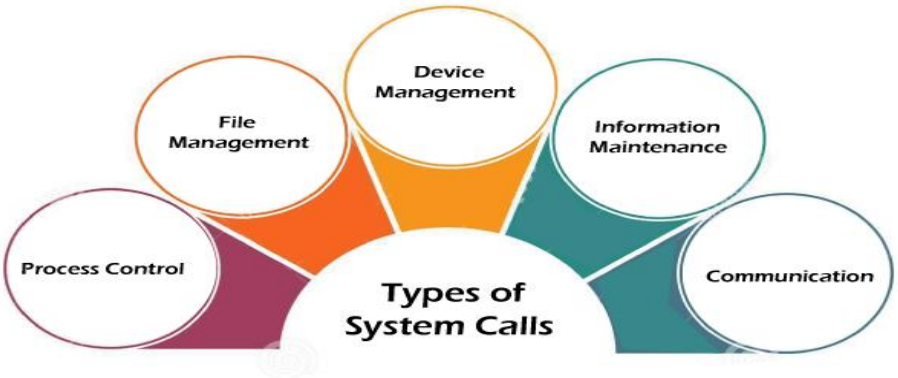
تعمل التطبيقات في منطقة من الذاكرة تُعرف باسم مساحة المستخدم User Space يتصل استدعاء النظام بنواة نظام التشغيل والتي يتم تنفيذها في مساحة kernel عندما يقوم أحد التطبيقات بإنشاء استدعاء نظام فيجب أولاً الحصول على إذن من ال kernel ويحقق ذلك باستخدام طلب مقاطعة والذي يوقف العملية الحالية مؤقتًا وينقل التحكم إلى ال.kernel

قد يستغرق استدعاء النظام البسيط بضع من النانو ثانية لتقديم النتيجة مثل استرداد تاريخ النظام ووقته وقد يستغرق استدعاء النظام الأكثر تعقيدًا مثل الاتصال بجهاز الشبكة بضع ثوانٍ وتطلق معظم أنظمة التشغيل مؤشر ترابط kernel مميز لكل استدعاء للنظام لتجنب الاختناقات. أنظمة التشغيل الحديثة تستطيع التعامل مع مكالمات النظام المختلفة في نفس الوقت.

أنواع استدعاءات النظام Types of System Calls

هناك خمسة أنواع شائعة من استدعاءات النظام:

1. **التحكم في العمليات Process Control**: التحكم في العملية هو استدعاء النظام المستخدم لتوجيه العمليات. تتضمن بعض أمثلة التحكم في العملية الإنشاء ، التحميل ، الإحباط ، الإنهاء ، التنفيذ ، المعالجة ، إنهاء العملية ، إلخ.
2. **إدارة الملفات File Management :** إدارة الملفات هي استدعاء نظام يتم استخدامه للتعامل مع الملفات. تتضمن بعض أمثلة إدارة الملفات إنشاء الملفات وحذفها وفتحها وإغلاقها وقراءتها وكتابتها وما إلى ذلك.
3. **إدارة الجهاز Device Management :** إدارة الجهاز هي استدعاء نظام يُستخدم للتعامل مع الأجهزة وتتضمن بعض أمثلة إدارة الجهاز القراءة ، والجهاز ، والكتابة ، والحصول على سمات الجهاز ، وإصدار الجهاز ، وما إلى ذلك.
4. **صيانة المعلومات Information Maintenance :** صيانة المعلومات هي استدعاء نظام يتم استخدامه للحفاظ على المعلومات. هناك بعض الأمثلة على صيانة المعلومات ، بما في ذلك الحصول على بيانات النظام
5. **الاتصالات Communication :** الاتصال هو استدعاء نظام يتم استخدامه للاتصال. هناك بعض الأمثلة على الاتصال مثل إنشاء الاتصال وحذفه وإرسال الرسائل واستلامها وما إلى ذلك.



Examples of Windows and Unix system calls

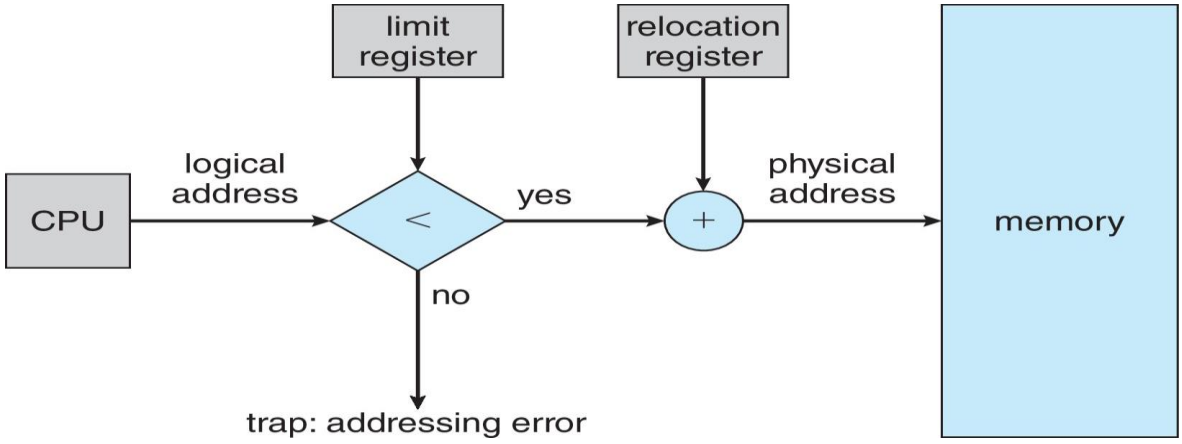
There are various examples of Windows and Unix system calls. These are as listed below in the table:

Process	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	Fork() Exit() Wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	Open() Read() Write() Close()
Device Management	SetConsoleMode() ReadConsole() WriteConsole()	Ioctl() Read() Write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	Getpid() Alarm() Sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	Pipe() Shmget() Mmap()

Memory Protection in Operating Systems حماية الذاكرة في أنظمة التشغيل

أن أنظمة التشغيل المختلفة تستخدم أشكالًا مختلفة من حماية الذاكرة وفي حماية الذاكرة يتعين علينا حماية نظام التشغيل من عمليات المستخدم والتي يمكن القيام بها باستخدام relocation register بسجل limit register هنا يحتوي سجل relocation register على قيمة أصغر عنوان فعلي physical address بينما يحتوي limit register على نطاق العناوين المنطقية logical addresses هذان السجلان لهما بعض الشروط مثل كل عنوان منطقي يجب أن يكون أقل من limit register. وتُستخدم وحدة

إدارة الذاكرة memory management unit لترجمة العنوان المنطقي logical address بالقيمة الموجودة في relocation register ديناميكيًا وبعد ذلك يتم إرسال العنوان المترجم (أو المعين) إلى الذاكرة.



Support for relocation and limit registers in process

في الرسم البياني أعلاه عندما يحدد المجدول scheduler عملية لعملية التنفيذ execution process ، يكون المرسل dispatcher من ناحية أخرى مسؤولاً عن تحميل سجلات relocation and limit registers بالقيم الصحيحة كجزء من تبديل السياق context switch مثل كل عنوان تم إنشاؤه بواسطة وحدة المعالجة المركزية CPU يتم فحصه مقابل هذين المسجلين ، وقد نحمي نظام التشغيل والبرامج وبيانات المستخدمين من التغيير من خلال هذه العملية قيد التشغيل.

الحاجة إلى حماية الذاكرة Need of Memory protection

تمنع حماية الذاكرة العملية من الوصول إلى الذاكرة غير المخصصة unallocated memory في نظام التشغيل لأنها تمنع البرنامج من السيطرة على مقدار زائد من الذاكرة وقد يتسبب في تلف يؤثر على البرامج الأخرى المستخدمة حاليًا أو قد يؤدي إلى فقدان البيانات المحفوظة. تساعد موارد حماية الذاكرة هذه أيضًا في اكتشاف التطبيقات الضارة malicious or harmful applications والتي قد تتلف بعد عمليات نظام التشغيل.

طرق حماية الذاكرة Methods of memory protection

هناك طرق مختلفة للحماية من الوصول إلى الذاكرة التي لم يتم تخصيصها وبعض الطرق شائعة الاستخدام مذكورة أدناه:

- **حماية الذاكرة باستخدام المفاتيح Memory Protection using Keys :**

يمكن العثور على مفهوم استخدام حماية الذاكرة مع المفاتيح في معظم أجهزة الكمبيوتر الحديثة بغرض تنظيم الذاكرة المقسمة إلى صفحات paged memory وللتوزيع الديناميكي بين برامج التشغيل المتوازية تعتمد المفاتيح على استخدام رموز خاصة حيث يمكننا التحقق من التوافق بين استخدام مصفوفات خلايا الذاكرة وعدد البرامج قيد التشغيل وتمنح هذه الطريقة الأساسية المستخدمين عملية لفرض الحماية على أساس الصفحة page-based protections دون أي تعديل في جداول الصفحات.

- **حماية الذاكرة باستخدام الحلقات Memory Protection using Rings :**

في علم الكمبيوتر تسمى المجالات domains المتعلقة بالحماية المطلوبة حلقات الحماية Protection Rings تساعد هذه الطريقة في تحسين التسامح مع الخطأ fault tolerance وتوفر الأمان. يتم ترتيب هذه الحلقات في تسلسل هرمي من الأكثر امتيازًا إلى الأقل امتيازًا most privileged to least privileged في نظام تشغيل المشاركة أحادي المستوى single-level sharing OS يحتوي كل جزء على حلقة حماية لعملية القراءة والكتابة وتنفيذ العمليات فإذا كان هناك استخدام لرقم حلقة أعلى من خلال العملية ، فإن رقم الحلقة الخاص بالمقطع يؤدي إلى حدوث خطأ لكن لدينا بعض الطرق لاستدعاء الإجراءات بأمان والتي يمكن تشغيلها برقم رنين أقل ثم العودة إلى رقم الحلقة الأعلى.

- **العنونة القائمة على القدرة Capability-based addressing :**

هي طريقة لحماية الذاكرة لا يمكن رؤيتها في أجهزة الكمبيوتر التجارية الحديثة. هنا يتم استعادة المؤشرات pointers (التي تتكون من عنوان ذاكرة) بواسطة كائنات القدرات capabilities objects التي لا يمكن إنشاؤها إلا من خلال التعليمات المحمية ويمكن تنفيذها فقط من خلال نواة Kernel أو من خلال عملية أخرى مصرح لها بالتنفيذ ، وبالتالي فهي تمنح ميزة للتحكم في العمليات غير المصرح بها في إنشاء مساحات عناوين منفصلة إضافية في الذاكرة.

- **حماية الذاكرة باستخدام الأقنعة Memory Protection using masks**

تستخدم الأقنعة في حماية الذاكرة أثناء تنظيم الصفحات organization of paging في هذه الطريقة ، قبل التنفيذ تتم الإشارة إلى أرقام الصفحات لكل برنامج ويتم حجزها لوضع توجيهاتها. هنا يتم منح الصفحات المخصصة للبرنامج الآن التحكم في نظام التشغيل في شكل رمز قناع (رمز ثنائي بت) والذي يتم تشكيله لكل برنامج عمل يتم تحديده بواسطة عدد البت

- **حماية الذاكرة باستخدام التجزئة Memory Protection using Segmentation**

هي طريقة لتقسيم ذاكرة النظام إلى أجزاء مختلفة. تُستخدم هياكل البيانات الخاصة بمعمارية x86 لنظام التشغيل مثل جدول واصف محلي local descriptor وجدول واصف عالمي global descriptor في حماية الذاكرة.

- **حماية الذاكرة باستخدام التجزئة المحاكاة Memory Protection using Simulated segmentation**

باستخدام هذه التقنية ، يمكننا مراقبة البرنامج لتفسير تعليمات كود الآلة الخاصة بهياكل النظام ويمكن للمحاكاة المساعدة في حماية الذاكرة باستخدام التجزئة باستخدام المخطط والتحقق من صحة العنوان الهدف لكل تعليمات في الوقت الفعلي.

حماية الأجهزة ونوع حماية الأجهزة Hardware Protection and Type of Hardware Protection

أن نظام الكمبيوتر يحتوي على الأجهزة مثل المعالج والشاشة وذاكرة الوصول العشوائي وغيرها الكثير ، وشيء واحد يضمنه نظام التشغيل أن هذه الأجهزة لا يمكن للمستخدم الوصول إليها مباشرة.

بشكل أساسي ، يتم تقسيم حماية الأجهزة إلى 3 فئات: حماية وحدة المعالجة المركزية وحماية الذاكرة وحماية الإدخال / الإخراج. على النحو التالي.

1. حماية وحدة المعالجة المركزية CPU Protection

يشار إلى حماية وحدة المعالجة المركزية لأننا لا نستطيع إعطاء وحدة المعالجة المركزية لعملية ما إلى الأبد ، يجب أن يكون ذلك لبعض الوقت المحدود وإلا فلن تحصل العمليات الأخرى على فرصة للتنفيذ. لذلك يتم استخدام عداد الوقت timer للخروج من هذا الموقف والذي يمنح بشكل أساسي قدرًا معينًا من الوقت للعملية وبعد تنفيذ الموقت ، سيتم إرسال إشارة إلى العملية لمغادرة وحدة المعالجة المركزية. ومن ثم لن تحتفظ العملية بوحدة المعالجة المركزية لمزيد من الوقت.

2. .2حماية الذاكرة Memory Protection

في حماية الذاكرة ، نتحدث عن هذا الموقف عندما تكون هناك عمليتان أو أكثر في الذاكرة وقد تصل إحدى العمليات إلى ذاكرة العملية الأخرى. ولمنع حدوث هذا الموقف ، نستخدم سجلين على النحو التالي:

1. **السجل القاعدة Base register**
2. **تسجيل الحد Limit register**

يقوم مسجل Base register بتخزين عنوان بدء البرنامج starting address وتخزين السجل الحد limit register لحجم العملية لذلك عندما تريد إحدى العمليات الوصول إلى الذاكرة ، يتم التحقق من إمكانية الوصول إلى الذاكرة أو عدم قدرتها على الوصول إليها.

حماية الإدخال / الإخراج I/O Protection

عندما نضمن حماية الإدخال / الإخراج I/O protection فلن تحدث بعض الحالات مطلقًا في النظام مثل:

1. إنهاء الإدخال / الإخراج لعملية أخرى Termination I/O of other process
2. عرض الإدخال / الإخراج لعملية أخرى View I/O of other process
3. إعطاء الأولوية لعملية إدخال / إخراج معينة Giving priority to a particular process I/O

حماية النظام في نظام التشغيل System Protection in Operating System

تشير الحماية إلى آلية تتحكم في وصول البرامج أو العمليات أو المستخدمين إلى الموارد المحددة بواسطة نظام الكمبيوتر. يمكننا أن نأخذ الحماية كمساعد لنظام تشغيل متعدد البرمجة multi programming بحيث يمكن للعديد من المستخدمين مشاركة مساحة اسم منطقي مشتركة بأمان مثل الدليل أو الملفات.

الحاجة للحماية Need of Protection

- لمنع وصول المستخدمين غير المصرح لهم

- للتأكد من أن كل برامج أو عمليات نشطة في النظام تستخدم الموارد فقط كما هو مذكور ،
- لتحسين الموثوقية من خلال الكشف عن الأخطاء الكامنة.

الحماية في نظام الملفات Protection in File System

في أنظمة الكمبيوتر، يتم تخزين الكثير من معلومات المستخدم والهدف من نظام التشغيل هو الحفاظ على أمان بيانات المستخدم من الوصول غير الصحيح إلى النظام ويمكن توفير الحماية بعدة طرق

أنواع الوصول Types of Access

تحتاج الملفات التي لها وصول مباشر لأي مستخدم إلى الحماية ولا تتطلب الملفات التي لا يمكن الوصول إليها من قبل المستخدمين الآخرين أي نوع من الحماية. توفر آلية الحماية تسهيل الوصول المتحكم فيه فقط من خلال الحد من أنواع الوصول إلى الملف. يمكن منح حق الوصول أو عدم منحه لأي مستخدم بناءً على عدة عوامل أحدها هو نوع الوصول المطلوب. يمكن التحكم في عدة أنواع مختلفة من العمليات:

- **Read –** Reading from a file.
- **Write –** Writing or rewriting the file.
- **Execute –** Loading the file and after loading the execution process starts.
- **Append –** Writing the new information to the already existing file, editing must be end at the end of the existing file.
- **Delete –** Deleting the file which is of no use and using its space for another data.
- **List –** List the name and attributes of the file.

عمليات مثل إعادة التسمية وتحرير الملف الحالي والنسخ يمكن أيضًا التحكم فيها. هناك العديد من آليات الحماية. كل آلية منها لها مزايا وعيوب مختلفة ويجب أن تكون مناسبة للتطبيق المقصود.

Operating Systems

Lecture # 6

Department of Computer

4th Class

Operating System Properties

By

Dr. Ahmed Khudhair Abbas

Collage of Education For Puer Science

نظام التشغيل هو واجهة تجعل الأمور أكثر بساطة للمستخدمين ويوفر الخدمات لكل من المستخدم والنظام لإجراء التفاعل بينهما. كما أن له خصائص مختلفة وتعد خصائص نظام التشغيل هذه من أكثر المهام التي يقوم بها نظام التشغيل اهمية كونها ميزة تجعل النظام سهل الاستخدام.

هناك خصائص مختلفة لنظام التشغيل وبعض من هذه الخصائص كما يلي:

- المعالجة بالدفعات Batch Processing
- تعدد البرمجة Multiprogramming
- التفاعل Interactivity
- أنظمة الوقت الحقيقي Real-Time System
- البيئة الموزعة Distributive Environment
- تعدد المهام Multitasking
- التخزين المؤقت Spooling

تعدد المهام Multitasking

يشير مصطلح "تعدد المهام" الى تنفيذ مهام متعددة بواسطة وحدة المعالجة المركزية في وقت واحد عن طريق التبديل بينها. تحدث المحولات بشكل متكرر لدرجة أن المستخدمين قد يتفاعلون مع كل برنامج أثناء تشغيله.

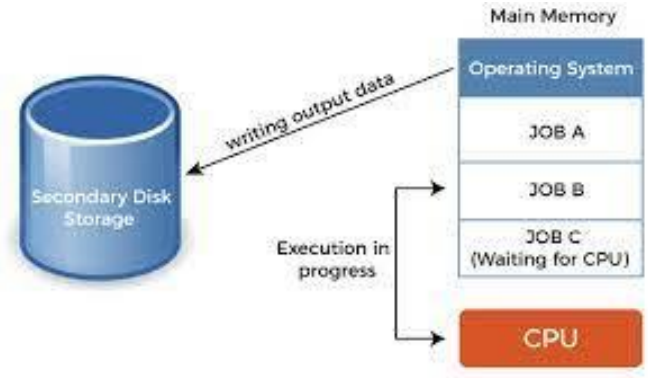
يقوم نظام التشغيل بالأنشطة التالية المتعلقة بتعدد المهام:

- يعطي المستخدم التعليمات لنظام التشغيل أو البرنامج مباشرة، ويتلقى استجابة فورية.
- يتعامل نظام التشغيل مع المهام المتعددة multitasking بالطريقة التي يمكنه من خلالها التعامل مع عمليات متعددة multiple operations / تنفيذ برامج متعددة multiple programs في نفس الوقت.
- تُعرف أنظمة التشغيل متعددة المهام أيضًا باسم أنظمة مشاركة الوقت Time-sharing systems
- تم تطوير أنظمة التشغيل هذه لتوفير الاستخدام التفاعلي لنظام الكمبيوتر بتكلفة معقولة.
- يستخدم نظام التشغيل الوقت المشترك كمفهوم لجدولة وحدة المعالجة المركزية والبرمجة المتعددة CPU scheduling and multiprogramming لتزويد كل مستخدم بجزء صغير من وقت وحدة المعالجة المركزية.
- يشار إلى البرنامج Code الذي يتم تحميله في الذاكرة ويتم تنفيذه على أنه عملية Process
- حتى تكتمل. خلال هذا الوقت يمكن استخدام وحدة المعالجة المركزية من خلال عملية أخرى.
- يسمح نظام التشغيل للمستخدمين بمشاركة الكمبيوتر في وقت واحد. نظرًا لأن كل إجراء أو أمر في نظام مشترك زمنيًا يميل إلى أن يكون قصيرًا، فلا يلزم سوى القليل من وقت وحدة المعالجة المركزية لكل مستخدم.
- نظرًا لأن النظام يقوم بتبديل وحدة المعالجة المركزية بسرعة من مستخدم / برنامج إلى آخر يتم إعطاء كل مستخدم انطباعًا بأن لديه / لديها وحدة المعالجة المركزية الخاصة به / بها بينما في الواقع تتم مشاركة وحدة المعالجة المركزية الواحدة بين العديد من المستخدمين.

تعدد البرمجة Multiprogramming

عند وجود برنامجين أو أكثر في الذاكرة في نفس الوقت تكون مشاركة المعالج وهذا يشير إلى البرمجة المتعددة وتفترض البرمجة المتعددة وجود معالج واحد مشترك وتزيد البرمجة المتعددة من استخدام وحدة المعالجة المركزية عن طريق تنظيم الوظائف بحيث يكون لدى وحدة المعالجة المركزية دائمًا واحدًا لتنفيذه.

بمعنى اخر يمكن تحميل برامج متعددة في الذاكرة الرئيسية لتنفيذها في نظام متعدد البرمجة وقد يستخدم برنامج أو عملية واحدة فقط وحدة المعالجة المركزية لتنفيذ التعليمات في وقت واحد، بينما يجب على الآخرين انتظار وقتهم. الهدف الرئيسي من استخدام نظام متعدد البرامج هو التغلب على نقص استخدام وحدة المعالجة المركزية والذاكرة الأساسية وإدارة موارد النظام بالكامل. المكونات الرئيسية لنظام متعدد البرمجة هي نظام التحكم في الإدخال / الإخراج ومعالج الأوامر والمنطقة المؤقتة ونظام الملفات.



يقوم نظام التشغيل بالأنشطة التالية المتعلقة بالبرمجة المتعددة:

- يحتفظ نظام التشغيل بالعديد من المهام في الذاكرة في وقت واحد.
- يختار نظام التشغيل إحدى المهام ويبدأ في تنفيذها في الذاكرة.
- يراقب نظام التشغيل متعدد البرامج حالة جميع البرامج النشطة وموارد النظام باستخدام برامج إدارة الذاكرة لضمان عدم خمول وحدة المعالجة المركزية أبدًا ما لم تكن هناك وظائف.

المزايا Advantages

- استخدام عالي وفعال لوحدة المعالجة المركزية.
- يشعر المستخدم أنه يتم تخصيص العديد من البرامج لوحدة المعالجة المركزية تقريبًا في وقت واحد.

العيوب Dis-advantages

- جدولة وحدة المعالجة المركزية مطلوبة.
- لاستيعاب العديد من المهام في الذاكرة يلزم إدارة الذاكرة.

أنظمة الوقت الحقيقي Real-Time System

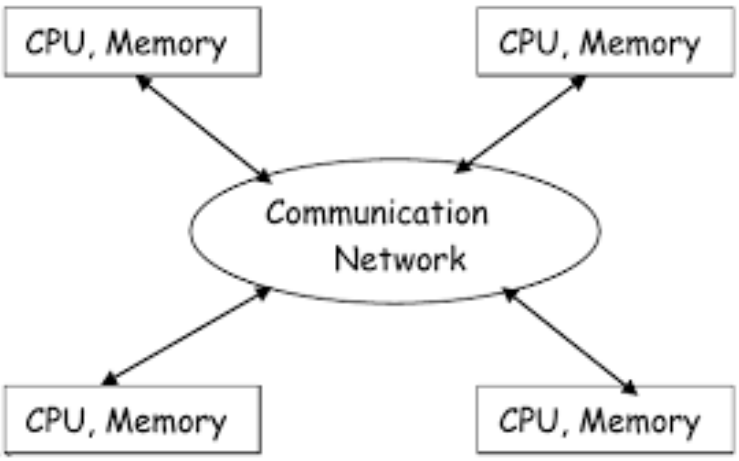
تمثل أنظمة الوقت الفعلي عادةً أنظمة مدمجة مخصصة. في مثل هذه الأنظمة تقرأ أنظمة التشغيل عادةً من بيانات المستشعر وتتفاعل معها و يجب أن يضمن نظام التشغيل الاستجابة للأحداث خلال فترات زمنية محددة لضمان الأداء الصحيح.

البيئة الموزعة Distributive Environment

تشير البيئة الموزعة إلى العديد من وحدات المعالجة المركزية المستقلة أو المعالجات في نظام الكمبيوتر.

يقوم نظام التشغيل بالأنشطة التالية المتعلقة بالبيئة الموزعة :

- نظام التشغيل OS يوزع بين عدة معالجات.
- لا تشترك المعالجات في الذاكرة.
- بدلاً من ذلك يمتلك كل معالج ذاكرة محلية خاصة به.
- نظام التشغيل يدير الاتصالات بين المعالجات. يتواصلون مع بعضهم البعض من خلال خطوط الاتصال المختلفة.



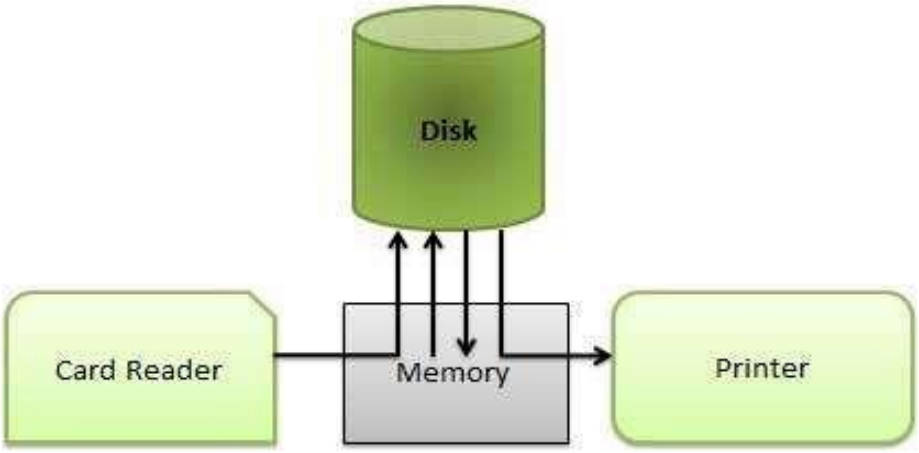
التخزين المؤقت Spooling

Spooling (و معناه التخزين المؤقت) هو اختصار للعمليات الطرفية المتزامنة. يشير التخزين المؤقت إلى وضع بيانات وظائف الادخال / الاخراج المختلفة في مخزن مؤقت. هذا المخزن المؤقت هو منطقة خاصة في الذاكرة أو القرص الصلب يمكن الوصول إليها من قبل أجهزة الادخال والإخراج Input/Output

يقوم نظام التشغيل بالأنشطة التالية المتعلقة بالبيئة الموزعة:

- يتعامل نظام التشغيل مع التخزين المؤقت لبيانات جهاز الإدخال / الإخراج لأن الأجهزة لها معدلات وصول مختلفة إلى البيانات.

- يحافظ نظام التشغيل على المخزن المؤقت للتخزين المؤقت الذي يوفر محطة انتظار حيث يمكن للبيانات أن تستريح أثناء اللحاق بالجهاز الأبطأ.
- يحافظ نظام التشغيل على الحساب المتوازي بسبب عملية التخزين المؤقت حيث يمكن لجهاز الكمبيوتر تنفيذ عمليات الاخال / الاخراج بطريقة متوازية. يصبح من الممكن أن يقوم الكمبيوتر بقراءة البيانات من شريط وكتابة البيانات على القرص والكتابة إلى طابعة شريط أثناء قيامه بمهمة الحوسبة.



Operating Systems

Lecture # 7

Department of Computer

4th Class

Operating System Processes

By

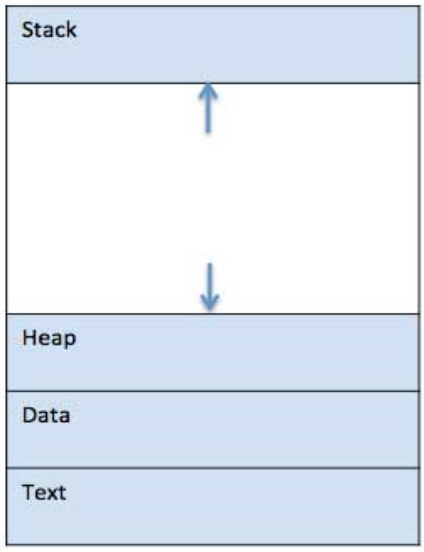
Dr. Ahmed Khudhair Abbas

Collage of Education For Pure Science – Computer Department

العملية Process

هي في الأساس برنامج قيد التنفيذ. وان تنفيذ العملية يجب أن يتقدم بطريقة متسلسلة. ويتم تعريف العملية على أنها كيان يمثل وحدة العمل الأساسية التي سيتم تنفيذها في النظام، وبعبارات بسيطة، نكتب برامج الكمبيوتر الخاصة بنا في ملف نصي وعندما نقوم بتنفيذ هذا البرنامج، تصبح عملية تؤدي جميع المهام المذكورة في البرنامج وعندما يتم تحميل أي برنامج في الذاكرة فانه يصبح عملية ويمكن تقسيمه إلى أربعة أقسام

- مكدس Stack
- كومة Heap
- نص Text
- بيانات Data



S.N.	Component & Description
1	**Stack:** The process Stack contains the temporary data such as method/function parameters, return address and local variables. يتحتوي ال Stack على البيانات المؤقتة الدوال والمتغيرات المحلية
2	**Heap:** This is dynamically allocated memory to a process during its run time. هي ذاكرة مخصصة ديناميكيًا لعملية ما أثناء وقت تشغيلها
3	**Text:** This includes the current activity represented by the value of Program Counter and the contents of the processor's registers. عداد البرامج ومحتويات سجلات المعالج
4	**Data:** This section contains the global and static variables. يحتوي هذا القسم على المتغيرات العامة والثابتة

البرنامج Program

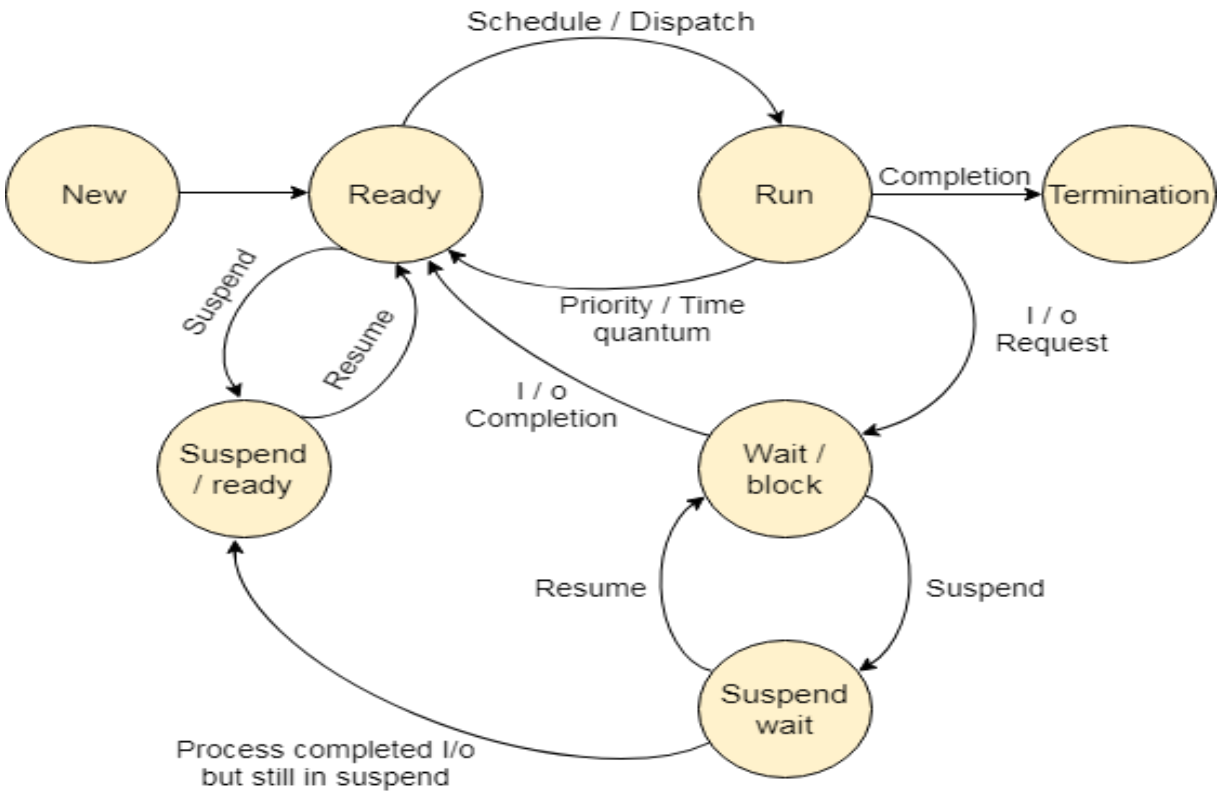
البرنامج عبارة عن جزء من التعليمات البرمجية قد يكون سطرًا واحدًا أو ملايين الأسطر. عادة ما يتم كتابة برنامج الكمبيوتر بواسطة مبرمج كمبيوتر بلغة برمجة. على سبيل المثال، هنا برنامج بسيط مكتوب بلغة البرمجة C

```
#include <stdio.h>
int main()  {
  printf("Hello, World! \n");
  return 0; }
```

يُعرف جزء من برنامج الكمبيوتر الذي يؤدي مهمة محددة باسم الخوارزمية. يشار إلى مجموعة من برامج الكمبيوتر والمكتبات والبيانات ذات الصلة بالبرمجيات.

دورة حياة العملية Process Life Cycle

عندما يتم تنفيذ العملية، فإنها تمر عبر حالات مختلفة. قد تختلف هذه المراحل في أنظمة التشغيل المختلفة، كما أن أسماء هذه الحالات ليست موحدة وبشكل عام، يمكن أن تشتمل العملية على إحدى الحالات التالية في كل مرة.



1 .جديد New

البرنامج الذي سوف يحمله نظام التشغيل في الذاكرة الرئيسية يسمى عملية جديدة.

2 .جاهز Ready

عندما يتم إنشاء عملية، فإنها تدخل مباشرة في حالة الاستعداد، حيث تنتظر تعيين وحدة المعالجة المركزية ويختار نظام التشغيل العمليات الجديدة من الذاكرة الثانوية ويضعها جميعًا في الذاكرة الرئيسية.

تسمى العمليات الجاهزة للتنفيذ والموجودة في الذاكرة الرئيسية عمليات الحالة الجاهزة. يمكن أن يكون هناك العديد من العمليات الموجودة في حالة الاستعداد.

3 .التنفيذ Running

سيتم اختيار إحدى العمليات من حالة الاستعداد بواسطة نظام التشغيل اعتمادًا على خوارزمية الجدولة. ومن ثم، إذا كان لدينا وحدة معالجة مركزية واحدة فقط في نظامنا، فسيكون عدد العمليات الجارية لوقت معين دائمًا واحدًا. إذا كان لدينا معالجات n في النظام، فيمكننا تشغيل عمليات n في وقت واحد.

4 .حظر أو الانتظار Block or wait

من حالة التشغيل، يمكن لعملية الانتقال إلى حالة الحظر أو الانتظار اعتمادًا على خوارزمية الجدولة أو السلوك الجوهري للعملية. وعندما تنتظر إحدى العمليات تعيين مورد معين أو إدخال من المستخدم، يقوم نظام التشغيل بنقل هذه العملية إلى حالة الحظر أو الانتظار وتعيين وحدة المعالجة المركزية إلى العمليات الأخرى.

5 .الإكمال أو الإنهاء Completion or termination

عندما تنتهي العملية من تنفيذها، فإنها تأتي في حالة الإنهاء. سيتم أيضًا حذف كل سياق العملية (كتلة التحكم في العملية) وسيتم إنهاء العملية بواسطة نظام التشغيل.

6 .التعليق Suspend ready

يتم استدعاء العملية في حالة الاستعداد، والتي يتم نقلها إلى الذاكرة الثانوية من الذاكرة الرئيسية بسبب نقص الموارد (الذاكرة الأساسية بشكل أساسي) في حالة الاستعداد المرحلي.

إذا كانت الذاكرة الرئيسية ممتلئة وتأتي عملية ذات أولوية أعلى للتنفيذ، فيجب على نظام التشغيل توفير مساحة للعملية في الذاكرة الرئيسية عن طريق التخلص من العملية ذات الأولوية المنخفضة في الذاكرة الثانوية. تظل العمليات الجاهزة للتعليق في الذاكرة الثانوية حتى تتوفر الذاكرة الرئيسية.

7 .تعليق الانتظار Suspend wait

بدلاً من إزالة العملية من قائمة الانتظار الجاهزة، من الأفضل إزالة العملية المحظورة التي تنتظر بعض الموارد في الذاكرة الرئيسية. نظرًا لأنه ينتظر بالفعل توفر بعض الموارد، فمن الأفضل أن ينتظر في الذاكرة الثانوية ويفسح المجال لعملية الأولوية الأعلى. تكمل هذه العمليات تنفيذها بمجرد توفر الذاكرة الرئيسية وينتهي انتظارها.

العمليات في العملية Operations on the Process

1 .التكوين Creation

بمجرد إنشاء العملية، ستكون جاهزة وستدخل في قائمة الانتظار الجاهزة (الذاكرة الرئيسية) وستكون جاهزة للتنفيذ.

2 .الجدولة Scheduling

من بين العديد من العمليات الموجودة في قائمة الانتظار الجاهزة، يختار نظام التشغيل عملية واحدة ويبدأ في تنفيذها. يُعرف اختيار العملية التي سيتم تنفيذها بعد ذلك باسم الجدولة.

3 .التنفيذ Execution

بمجرد جدولة العملية للتنفيذ، يبدأ المعالج في تنفيذها. قد تصل العملية إلى حالة الحظر أو الانتظار أثناء التنفيذ، وفي هذه الحالة يبدأ المعالج في تنفيذ العمليات الأخرى.

4 .الحذف / القتل Deletion/killing

بمجرد انتهاء الغرض من العملية، سيقتل نظام التشغيل العملية. سيتم حذف سياق العملية (PCB) ويتم إنهاء العملية بواسطة نظام التشغيل.

كتلة التحكم في العمليات Process Control Block (PCB)

كتلة التحكم في العمليات هي بنية بيانات يحتفظ بها نظام التشغيل لكل عملية. يتم تحديد كتلة التحكم في العملية من خلال معرّف العملية .(PID) وتحتفظ كتلة التحكم في العملية بجميع المعلومات اللازمة لتتبع العملية كما هو موضح في الجدول:

S.N.	Information & Description
1	**Process State**: The current state of the process i.e., whether it is ready, running, waiting, or whatever.
2	**Process privileges**: This is required to allow/disallow access to system resources.
3	**Process ID (PID)**: Unique identification for each of the process in the operating system.
4	**Pointer:** A pointer to parent process.

5	**Program Counter**: Program Counter is a pointer to the address of the next instruction to be executed for this process.
6	**CPU registers**: Various CPU registers where process need to be stored for execution for running state.
7	**CPU Scheduling Information**: Process priority and other scheduling information which is required to schedule the process.
8	**Memory management information:** This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.
9	**Accounting information:** This includes the amount of CPU used for process execution, time limits, execution ID etc.
10	**IO status information:** This includes a list of I/O devices allocated to the process.

تعتمد بنية PCB بشكل كامل على نظام التشغيل وقد تحتوي على معلومات مختلفة في أنظمة تشغيل مختلفة. ويتم الاحتفاظ بـ PCB للعملية على طوال عمرها، ويتم حذفها بمجرد انتهاء العملية. هنا رسم تخطيطي مبسط كتلة التحكم في العمليات

Process Control Block (PCB)

انشاء العمليات Process Creations

نظام العمليات يسمح بإنشاء العديد من العمليات الجديدة بواسطة استدعاء النظام (Call System) طيلة فترة التطبيق لهذه العملية وأن إنشاء العملية يطلق عليها Process Parent وهذا ما يقصد به الأب للعملية والعملية الجديدة يطلق عليه Process Child وهو الابن الذي تم انشائه من قبل العملية الأب.

وكل تلك العلميات الجديدة (الأبناء) قد تستطيع إنشاء عمليات جديدة أخرى وبالإمكان تجميعها بالشكل الشجري للعمليات، قد تكون هناك أشياء مشتركة بين الأب والابن حيث أن الابن قد يكون نسخه طبق الأصل عن الأب أو يشتركان في بعض الموارد أو أن لا يكون بينهما أي موارد مشتركة وفي وقت التنفيذ إما أن ينفذا في وقت متزامن أو أن ينتظر الأب حتى تنتهي عمليات التطبيق الخاصة بالأبناء ويوجد أيضا احتمالات لمكان وجود العملية الجديدة (الابن).

الابن عملية مزدوجة من العملية الأم والابن له برنامجه الخاص ومكان جديد يوجد به والكثير من أنظمة التشغيل بما في ذلك اللينكس والويندوز تقوم بتعريف العمليات تبعا لمعرف العملية Identifier Process الخاص الذي يظهر عادة كرقم صحيح.

انهاء العمليات Process Termination

أسباب انهاء العمليات :

- **الخروج الطبيعي Exist Normal :** وتكون العملية في هذه الح الة قد أنهت عملها وتم إنهاءها
- **الخروج بسبب خطأ Exist Error :** في هذه الحالة تكتشف العملية خطأ فادح Fatal Error مثال على ذلك محاولة تأليف compile لبرنامج غير موجود.
- **خطأ فادح Error Fatal :** وهنا يكون إنهاءها ناتج عن خطأ قامت به العملية، :مثل تنفيذها لأمر غير مسموح به كالقسمة على صفر أو الإحالة إلى مكان غير موجود في الذاكرة
- **قتلها kill** بواسطة عملية أخرى

الاتصال بين العمليات Inter-Process Communication

هناك أنواع للعمليات في نظام التشغيل اثناء تنفيذ العملية Process

1. **عملية مستقلة Independent**

وهي العملية المستقلة التي لا تتأثر أو تأثر في تنفيذ عملية أخرى في النظام وإنما تعمل مستقلة بذاتها

2. **عملية متعاونة Cooperating**

وهي العملية المتعاونة والتي يمكن أن تأثر أو تتأثر بتنفيذ عملية أخرى في النظام

Operating Systems

Lecture # 8

Department of Computer

4th Class

Process Schedulers

By

Dr. Ahmed Khudhair Abbas

Collage of Education for Pure Science – Computer Department

جدولة العملية Process scheduling :هو الوظيفة التي يقوم بها مدير العملية Process Manager والذي يتولى إزالة العملية الجارية من وحدة المعالجة المركزية واختيار عملية أخرى على أساس إستراتيجية معينة.

وتعد جدولة العمليات جزءًا أساسيًا من أنظمة تشغيل المعتمدة على البرمجة المتعددة Multiprogramming operating systems وتسمح أنظمة التشغيل هذه بتحميل أكثر من عملية واحدة قابلة للتنفيذ في الذاكرة بنفس الوقت.

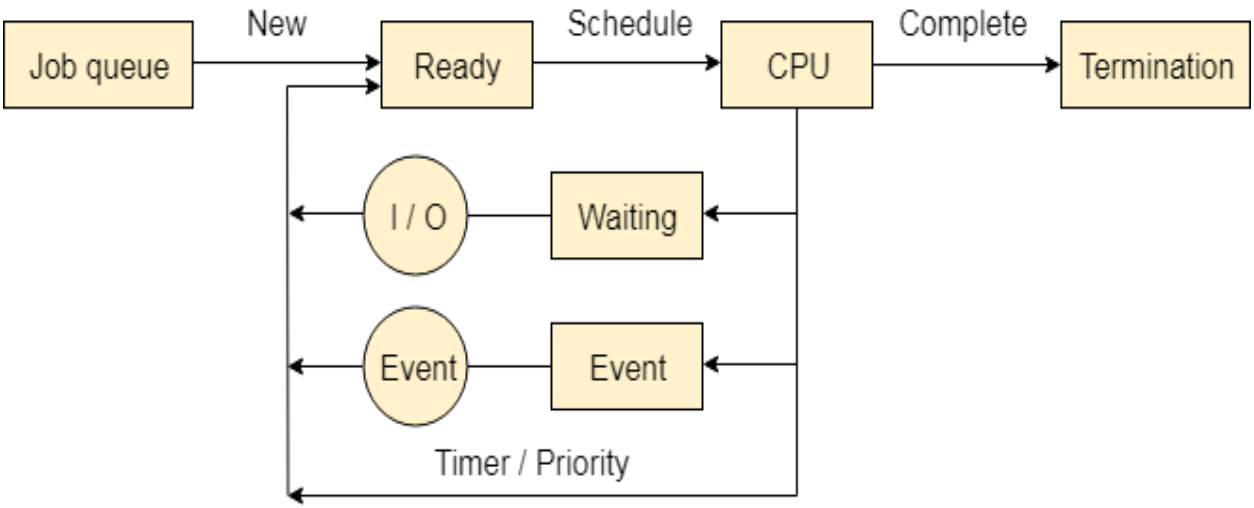
فئات الجدولة Categories of Scheduling

هناك فئتان من الجدولة:

1. غير استباقي **Non-preemptive**: هنا لا يمكن أخذ المورد resource من العملية حتى يكتمل تنفيذ العملية الحالية ويحدث تبديل الموارد عندما تنتهي العملية الجارية او تنتقل إلى حالة الانتظار Waiting
2. استباقية **Preemptive:** هنا يخصص نظام التشغيل الموارد لعملية لفترة محددة من الوقت. أثناء تخصيص الموارد وتنتقل العملية من حالة التشغيل Running إلى حالة الاستعداد Ready أو من حالة الانتظارWaiting إلى حالة الاستعداد Ready ويحدث هذا التبديل لأن وحدة المعالجة المركزية قد تعطي الأولوية للعمليات الأخرى وتستبدل العملية بأولوية أعلى بالعملية الجارية.

قوائم انتظار المعالجة Process Queues

يسيطر نظام التشغيل على أنواعًا مختلفة من قوائم الانتظار لكل حالة من حالات العملية ويتم أيضًا تخزين PCB المتعلق بالعملية في قائمة الانتظار من نفس الحالة اما إذا تم نقل العملية من حالة إلى حالة أخرى، فسيتم إلغاء ربط PCB الخاص بها من قائمة الانتظار المقابلة وإضافتها إلى قائمة انتظار الحالة الأخرى .



وادناه قوائم الانتظار التي يحتفظ بها نظام التشغيل.

1. **طابور المهام Job Queue**

في البداية، يتم تخزين جميع العمليات في قائمة انتظار المهام ويتم الاحتفاظ بها في الذاكرة الثانوية ثم يقوم برنامج **الجدولة طويلة المدى** (برنامج جدولة المهام) باختيار بعض المهام ووضعها في الذاكرة الأساسية.

2. **طابور الجاهزية Ready Queue**

يتم الاحتفاظ بقائمة الانتظار الجاهزة في الذاكرة الأساسية ويقوم برنامج **الجدولة قصير المدى** باختيار المهمة من قائمة الانتظار الجاهزة وإرسالها إلى وحدة المعالجة المركزية للتنفيذ.

3. **طابور الانتظار Waiting Queue**

عندما تحتاج العملية إلى بعض عمليات الإدخال / الإخراج لإكمال تنفيذها، يقوم نظام التشغيل بتغيير حالة العملية من التشغيل إلى الانتظار ثم يتم تخزين ال PCB المرتبط بالعملية في قائمة الانتظار التي سيستخدمها المعالج عندما تنتهي العملية من الإدخال / الإخراج.

المجدولات Schedulers

المجدولات هي برنامج خاص يتعامل مع جدولة العمليات بطرق مختلفة مهمته الرئيسية هي تحديد العملية التي سيتم تشغيلها. وتتكون الجدولة من ثلاثة أنواع

1. جدولة طويلة المدى Long-Term Scheduler
2. جدولة قصيرة المدى Short-Term Scheduler
3. جدولة متوسطة المدى Medium-Term Scheduler

جدولة طويلة الامد Long-Term Scheduler

ويسمى أيضًا بجدولة المهام ويقوم المجدول طويل الأجل بتحديد البرامج التي يتم قبولها في النظام للمعالجة وكذلك يقوم بتحديد العمليات من قائمة الانتظار وتحميلها في الذاكرة للتنفيذ ثم يتم تحميل العملية في الذاكرة لجدولة وحدة المعالجة المركزيةCPU.

جدولة قصيرة المدى Short-Term Scheduler

ويسمى أيضًا باسم جدولة وحدة المعالجة المركزية CPU scheduler هدفها الرئيسي هو زيادة أداء النظام ويتم في هذه الجدولة تغيير من حالة الاستعداد ready state إلى حالة تشغيل العملية running state ويتم تحديد عملية من بين العمليات الجاهزة

للتنفيذ ويخصص وحدة المعالجة المركزية لإحدى هذه العمليات يقوم المجدول على المدى القصير باتخاذ قرار بشأن العملية التي سيتم تنفيذها بعد ذلك. يعتبر المجدول على المدى القصير أسرع من المجدول على المدى الطويل.

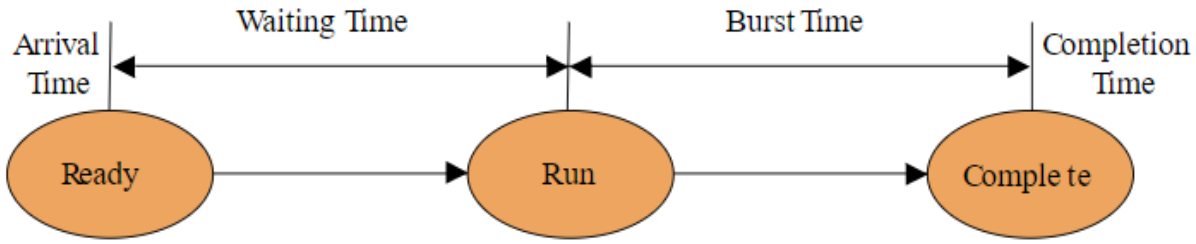
جدولة متوسطة المدى Medium-Term Scheduler

الجدولة متوسطة المدى هي جزء من المبادلة swapping يقوم بإزالة العمليات من الذاكرة. قد يتم تعليق العملية الجارية suspended ولا يمكن للعمليات المعلقة إحراز أي تقدم نحو الانتهاء في هذه الحالة، لإزالة العملية من الذاكرة وإفساح المجال للعمليات الأخرى، يتم نقل العملية المعلقة إلى التخزين الثانوي secondary storage تسمى هذه العملية بالمبادلة swapping ويقال إن العملية يتم تبديلها أو طرحها swapped out or rolled out وقد يكون التبادل ضروريًا لتحسين مزيج العملية.

مقارنة بين المجدولات Comparison among Scheduler

S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short- and long-term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time-sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

أوقات العملية Times related to the Process



$$CT - AT = WT + BT$$

$$TAT = CT - AT$$

$$\text{Waiting Time} = TAT - BT$$

TAT ⟶ Turn around time

BT ⟶ Burst time

AT ⟶ Arrival time

1 .وقت الوصول Arrival Time

يُطلق على الوقت الذي تدخل فيه العملية إلى قائمة الانتظار الجاهزة ready queue بوقت الوصول.

2 .وقت الاندفاع Burst Time

يُطلق على إجمالي الوقت الذي تتطلبه وحدة المعالجة المركزية لتنفيذ العملية بأكملها اسم .Burst Time هذا لا يشمل وقت الانتظار waiting time

3 .وقت الإنجاز Completion Time

يسمى الوقت الذي تدخل فيه العملية في حالة الإكمال completion state أو الوقت الذي تكمل فيه العملية تنفيذها بوقت الانتهاء.

4 .الوقت المستغرق Turnaround time

يُطلق على إجمالي الوقت الذي تقضيه العملية من وصولها إلى اكتمالها بوقت التحول.

5 .وقت الانتظار Waiting Time

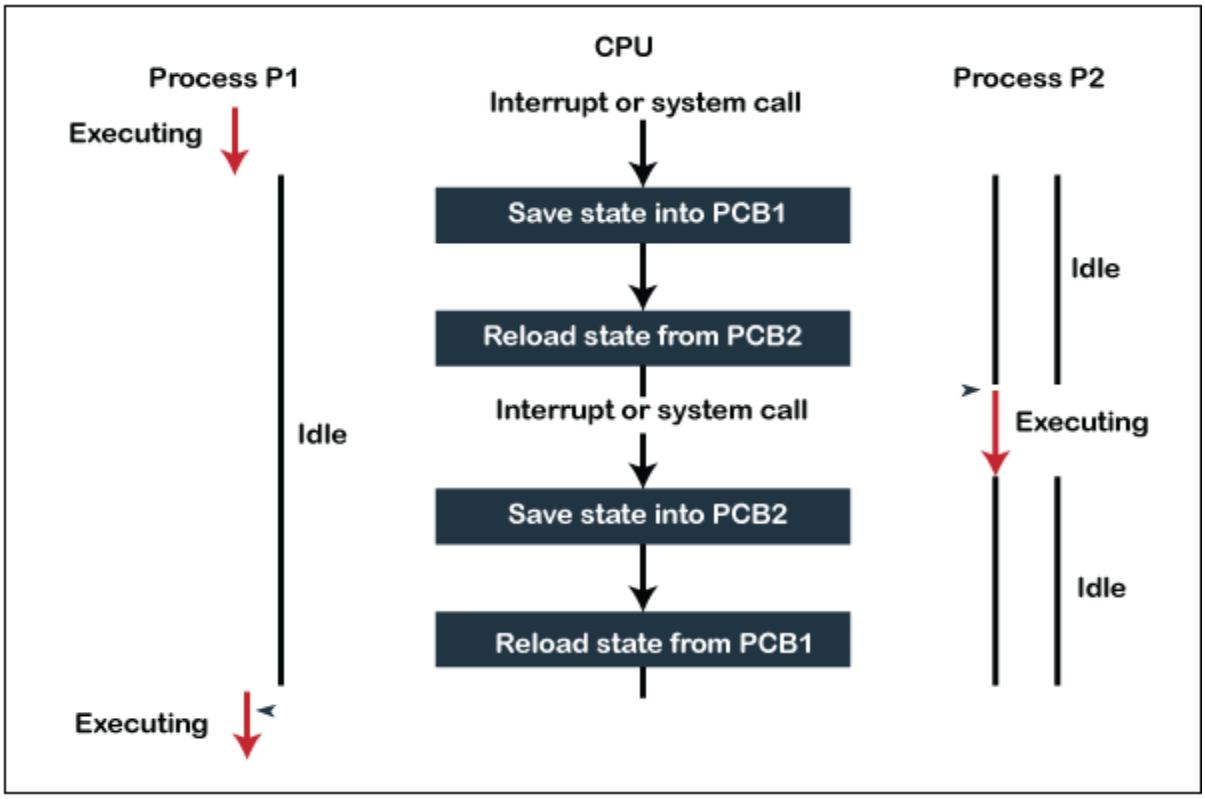
يُطلق على إجمالي الوقت الذي تنتظره العملية حتى يتم تعيين وحدة المعالجة المركزية (CPU) لها بوقت الانتظار.

6 .زمن الاستجابة Response Time

يسمى الفرق بين وقت الوصول arrival time والوقت الذي تحصل فيه العملية على وحدة المعالجة المركزية لأول مرة بوقت الاستجابة.

تبديل السياق Context Switch

تبديل السياق هو آلية لتخزين واستعادة حالة أو سياق وحدة المعالجة المركزية في كتلة التحكم في العملية PCB بحيث يمكن استئناف تنفيذ العملية من نفس النقطة في وقت لاحق. باستخدام هذه التقنية، يتيح مبدل السياق لعمليات متعددة في مشاركة وحدة معالجة مركزية واحدة ويعد تبديل السياق جزءًا أساسيًا من ميزات نظام التشغيل متعدد المهام multitasking operating system. وعندما يقوم المجدول بتحويل وحدة المعالجة المركزية من تنفيذ عملية إلى أخرى، يقوم مبدل السياق بحفظ محتوى جميع سجلات المعالج processor registers للعملية التي تتم إزالتها من وحدة المعالجة المركزية في وصف العملية الخاص بها ويتم تمثيل سياق العملية في كتلة التحكم في العملية PCB.



الشكل أعلاه يمثل تنفيذ عمليتين (Process 1 and Process 2) داخل وحدة المعالجة المركزية CPU والية التبديل بينهما وكذلك يبين الشكل عملية حفظ وتحميل ال PCB الخاصة بكل عملية

معايير الجدولة Scheduling Criteria

عملية الجدولة من أهم الخصائص في تشغيل العمليات حيث ينظم دخول العمليات المراد تنفيذها إلى وحده المعالجة المركزية (CPU) وتعتمد هذه العملية على العديد من المعايير التي تحدد من هي العملية التي يجب تنفيذها و من أهمها:

1. **استغلال وحده المعالجة المركزية CPU utilization:**
استغلال كل وقت وحده المعالجة المركزية (CPU) في تنفيذ العمليات. اي ان تكون وحده المعالجة المركزية (CPU) مشغولة بقدر الامكان ليتم استغلالها الاستغلال الأمثل

2. **كمية البيانات المتدفقة في الثانية الواحدة Throughput:**
تبيّن معدل العمليات التي يمكن انجازها في وقت معين . **مثال توضيحي /** استطيع انجاز بمعدل 4 عمليات في 3 ثواني.

3. **الوقت المستغرق Turnaround time (TAT) :**
الوقت اللازم لتنفيذ عملية ما (الوقت المستغرق من بداية تنفيذ العملية إلى نهايتها) .
مثال توضيحي/ابتدأ العملية في الثانية الثالثة وانتهت في الثانية السابعة الوقت اللازم لتنفيذ هذه العملية هو 4 ثواني

4. **وقت الانتظار Waiting time :**
هو الوقت الذي تستغرقه العملية في الانتظار داخل مصفوفة الانتظار(ready queue) قبل دخولها إلى وحده المعالجة المركزية(CPU)

5. **زمن الاستجابة Response time :** هو الوقت الذي يحتاجه البرنامج لبدأ فعليا. **مثال توضيحي /**منذ الضغط على البرنامج ضغطاً مزدوج إلى أن يعمل البرنامج فعليا.

هذه المفاهيم تمكنا من اختيار أفضل طريقة للجدولة ولكن من الصعب تحقيقها جميعا

حتى تصل وحده المعالجة المركزية إلى الكمال لابد من توفر التالي:

- ***Minimize latency:*** وهو تقليل تأخر زمن الاستجابة واكتمال تنفيذ البرنامج.
- **Maximize throughput:** وهو زيادة الوقت الذي يتم تنفيذ العمليات خلاله
- **Maximize utilization:** هو زيادة الاستفادة من وحدة المعالجة المركزية وإبقائها مشغولة طوال الوقت .
- **Fairness:** وهو الإنصاف بين البرامج حيث يسمح لجميع البرامج من استخدام وحدة المعالجة المركزية ولا تكون الصلاحيات جميعها لبرنامج واحد.

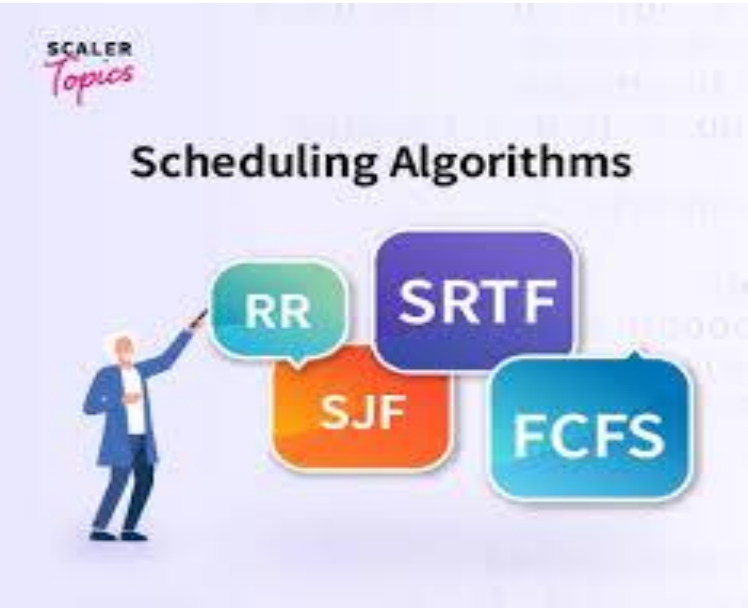
Operating Systems

Lecture # 9

Department of Computer

4th Class

Operating System Scheduling Algorithms



By

Dr. Ahmed Khudhair Abbas

Collage of Education for Pure Science – Computer Department

جدولة وحدة المعالجة المركزية CPU Scheduling

في أنظمة Uni-Programming مثل MS DOS ، عندما تنتظر إحدى العمليات إجراء أي عملية إدخال / إخراج ، تظل وحدة المعالجة المركزية معطلة وهذا عبء لأنه يضيع الوقت ويسبب مشكلة الجوع Starvation ومع ذلك في أنظمة البرمجة المتعددة لا تبقى وحدة المعالجة المركزية متوقفة أثناء وقت انتظار العملية وتبدأ في تنفيذ العمليات الأخرى ويجب أن يحدد نظام التشغيل العملية التي سيتم منحها وحدة المعالجة المركزية CPU .

في أنظمة البرمجة المتعددة **systems** Multiprogramming يقوم نظام التشغيل بجدولة العمليات على وحدة المعالجة المركزية لتحقيق أقصى استفادة منها ويسمى هذا الإجراء جدولة وحدة المعالجة المركزية ويستخدم نظام التشغيل خوارزميات جدولة مختلفة لجدولة العمليات.

خوارزميات الجدولة Scheduling Algorithms

هناك العديد من الخوارزميات التي يستخدمها نظام التشغيل لجدولة العمليات على المعالج بطريقة فعالة والغرض من خوارزمية الجدولة هو:

1 .أقصى استخدام لوحدة المعالجة المركزية Maximum CPU utilization
2 .التخصيص العادل لوحدة المعالجة المركزية Fair allocation of CPU
3 .الحد الأقصى للإنتاجية Maximum throughput
4 .الحد الأدنى من الوقت المستغرق Minimum turnaround time
5 .الحد الأدنى لوقت الانتظار Minimum waiting time
6 .الحد الأدنى من وقت الاستجابة Minimum response time

يقوم برنامج جدولة العمليات بجدولة عمليات مختلفة ليتم تخصيصها لوحدة المعالجة المركزية بناءً على خوارزميات جدولة معينة. هناك ست خوارزميات شائعة لجدولة العمليات وهي:

1. جدولة من يأتي أولاً يخدم أولاً First-Come, First-Served (FCFS) Scheduling
2. جدولة أقصر مهمة تالية Shortest-Job-Next (SJN) Scheduling
3. جدولة الأولوية Priority Scheduling
4. أقصر وقت متبقي Shortest Remaining Time
5. جدولة راوند روبن Round Robin (RR) Scheduling
6. جدولة قوائم الانتظار متعددة المستويات Multiple-Level Queues Scheduling

هذه الخوارزميات إما غير استباقية أو استباقية **non-preemptive or preemptive** وتم تصميم الخوارزميات غير الاستباقية **non-preemptive** بحيث بمجرد دخول العملية إلى حالة التشغيل لا يمكن ايقافها حتى تكمل الوقت المخصص لها

وفي حين أن الجدولة الاستباقية **preemptive** تعتمد على الأولوية حيث قد يستبق المجدول عملية تشغيل ذات أولوية منخفضة في أي وقت عندما تكون الأولوية عالية تدخل العملية في حالة جاهزة.

1. جدولة من يأتي أولاً يخدم أولاً First-Come, First-Served (FCFS) Scheduling

تقوم خوارزمية جدولة من يأتي أولاً يخدم أولاً (FCFS) بجدولة الوظائف وفقًا لوقت وصولها وستحصل المهمة التي تأتي أولاً في قائمة الانتظار الجاهزة على وحدة المعالجة المركزية أولاً وكلما قل وقت وصول الوظيفة ، زادت سرعة حصول الوظيفة على وحدة المعالجة المركزية وقد تتسبب جدولة FCFS في حدوث مشكلة الجوع Starvation إذا كان وقت الاندفاع للعملية الأولى هو الأطول بين جميع الوظائف ومن اهم مزايا هذه الطريقة البساطة وسهولة التطبيق ومن يأتي أولا يخدم أولا اما من اهم العيوب لهذه الخوارزمية فهي:

1 .طريقة الجدولة ليست وقائية **non-preemptive** وسوف تستمر العملية حتى الاكتمال.

2 .بسبب الطبيعة غير الوقائية للخوارزمية قد تحدث مشكلة الجوع Starvation

3 .على الرغم من سهولة تنفيذها إلا أنها ضعيفة في الأداء نظرًا لأن متوسط وقت الانتظار أعلى مقارنة بخوارزميات الجدولة الأخرى.

مثال Example

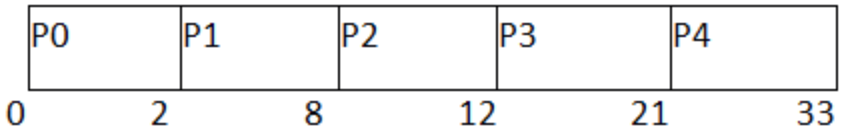
في الجدول التالي، توجد 5 عمليات بمعرف العملية P0 و P1 و P2 و P3 و .P4 تصل P0 في الوقت 0 ، P1 في الوقت 1 ، P2 في الوقت 2 ، P3 تصل في الوقت 3 وتصل العملية P4 في الوقت 4 في قائمة الانتظار الجاهزة ويتم حساب وقت الاستجابة ووقت الانتظار باستخدام الصيغة التالية.

1. Turn Around Time = Completion Time - Arrival Time
2. Waiting Time = Turnaround time - Burst Time

يتم تحديد متوسط وقت الانتظار من خلال جمع وقت الانتظار المعني لجميع العمليات وقسمة المجموع على إجمالي عدد العمليات.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P0	0	2	2	2	0
P1	1	6	8	7	1
P2	2	4	12	10	6
P3	3	9	21	18	9
P4	4	12	33	29	17

Avg Waiting Time=31/5

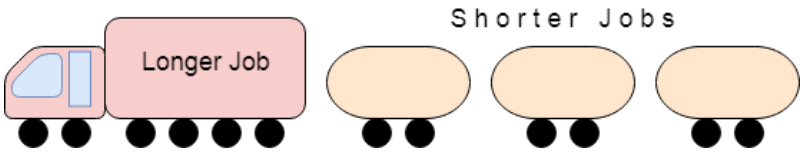


(Gantt chart)

تأثير المواكبة او المجاعة Convoy effect or Starvation **في FCFS**

قد يعاني FCFS من تأثير المواكبة إذا كان وقت تنفيذ المهمة الأولى هو الأعلى بين الجميع. كما هو الحال في الحياة الواقعية إذا كانت القافلة تمر عبر الطريق فقد يتم حظر الأشخاص الآخرين حتى تمر بشكل كامل ويمكن محاكاة ذلك في نظام التشغيل أيضًا. إذا حصلت وحدة المعالجة المركزية على عمليات وقت الوصول الأعلى في الطرف الأمامي لقائمة الانتظار الجاهزة، فقد يتم حظر عمليات وقت الوصول المنخفض مما يعني أنها قد لا تحصل على وحدة المعالجة المركزية مطلقًا إذا كانت المهمة في التنفيذ تستغرق وقتًا طويلاً للغاية. وهذا ما يسمى تأثير المواكبة أو الجوع Convoy effect or Starvation.

The Convoy Effect, Visualized Starvation



Example

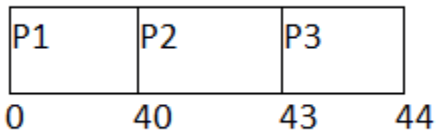
In the Example, We have 3 processes named as **P1, P2 and P3**. The Burt Time of process P1 is highest. The Turnaround time and the waiting time in the following table, are calculated by the formula:

Turn Around Time = Completion Time - Arrival Time

Waiting Time = Turn Around Time - Burst Time

في السيناريو الأول، تصل العملية P1 إلى المرتبة الأولى في قائمة الانتظار بالرغم من ذلك فان وقت وصول العملية هو الأعلى بين الجميع. نظرًا لأن خوارزمية الجدولة التي نتبعها هي FCFS ، فإن وحدة المعالجة المركزية ستنفذ العملية P1 أولاً. وفي هذا الجدول سيكون متوسط وقت انتظار النظام مرتفعًا جدًا ويتعين على العمليات الأخرى P2 و P3 انتظار دورها لمدة 40 وحدة زمنية على الرغم من أن وقت وصولها منخفض للغاية وهذا الجدول يعاني من الجوع starvation

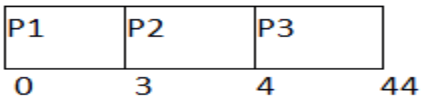
Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P1	**0**	**40**	**40**	**40**	**0**
P2	**1**	**3**	**43**	**42**	**39**
P3	**1**	**1**	**44**	**43**	**42**



Avg waiting Time = 81/3

في السيناريو الثاني إذا كانت العملية P1 قد وصلت إلى آخر قائمة الانتظار والعمليات الأخرى P2 و P3 في وقت سابق ، فلن تكون مشكلة الجوع موجودة. ويوضح المثال التالي الانحراف في أوقات الانتظار لكلا السيناريوهات على الرغم من أن طول الجدول الزمني هو نفسه 44 وحدة ، إلا أن وقت الانتظار سيكون أقل في هذا الجدول.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	**1**	**40**	**44**	**43**	**3**
2	**0**	**3**	**3**	**3**	**0**
3	**0**	**1**	**4**	**4**	**3**



Avg Waiting Time = 6/3

2. جدولة أقصر مهمة أولاً Shortest Job First (SJF) Scheduling

تقوم خوارزمية جدولة SJF بجدولة العمليات وفقًا لوقت وصولها. في جدولة SJF ، ستتم جدولة العملية ذات أقل وقت تنفيذ من بين قائمة العمليات المتاحة في قائمة الانتظار الجاهزة ومن اهم مزايا هذه الطريقة هو تحقيقها الحد الأقصى من الإنتاجية والحد الأدنى لمتوسط الانتظار ووقت الاستجابة . اما عيوبها فقد تعاني من مشكلة الجوع Starvation .

Example

In the following example, there are five jobs named as P1, P2, P3, P4 and P5. Their arrival time and burst time are given in the table below.

PID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
p1	1	7	8	7	0
p2	3	3	13	10	7
p3	6	2	10	4	2
p4	7	10	31	24	14
p5	9	8	21	12	4

	P1	P3	P2	P5	P4	
0	1	8	10	13	21	31

Avg Waiting Time = 27/5

كما نلاحظ لم تصل أي عملية في الوقت 0 وبالتالي ستكون هناك فتحة فارغة في مخطط جانت من الوقت 0 إلى 1 (الوقت الذي تصل فيه العملية الأولى) ووفقًا للخوارزمية، يقوم نظام التشغيل بجدولة العملية التي لها أقل وقت تنفيذ بين العمليات المتاحة في قائمة الانتظار الجاهزة. حتى الآن، لدينا عملية واحدة فقط في قائمة الانتظار الجاهزة ومن ثم سيقوم المجدول بجدولة ذلك إلى المعالج بغض النظر عن وقت تنفيذه وسيتم تنفيذ هذا حتى 8 وحدات زمنية حتى ذلك الحين، وصلنا ثلاث عمليات أخرى إلى قائمة الانتظار الجاهزة، ومن ثم سيختار المجدول العملية بأقل وقت للتنفيذ.

من بين العمليات الواردة في الجدول، سيتم تنفيذ P3 بعد ذلك نظرًا لأنه يحتوي على أقل وقت تنفيذ بين جميع العمليات المتاحة وهذه هي الطريقة التي سيستمر بها الإجراء في خوارزمية جدولة أقصر مهمة أولاً.(SJF)

خوارزمية جدولة أقصر وقت متبقي أولاً Shortest Remaining Time First (SRTF)

هذه الخوارزمية هي النسخة الاستباقية **preemptive version** of **SJF scheduling** لجدولة .SJF في SRTF ويمكن إيقاف تنفيذ العملية بعد فترة زمنية معينة. عند وصول كل عملية يقوم برنامج الجدولة قصير المدى بجدولة العملية بأقل وقت اندفاع متبقي بين قائمة العمليات المتاحة والعملية الجارية وبمجرد توفر جميع العمليات في قائمة الانتظار الجاهزة ، لن يتم إجراء أي إجراءات استباقية وستعمل الخوارزمية كجدولة

.SJF يتم حفظ سياق العملية في كتلة التحكم في العملية عند إزالة العملية من التنفيذ ويتم جدولة العملية التالية. يتم الوصول إلى هذا PCB عند التنفيذ التالي لهذه العملية.

In this Example, there are five jobs P1, P2, P3, P4, P5 and P6. Their arrival time and burst time are given below in the table.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time	Response Time
p1	0	8	20	20	12	0
p2	1	4	10	9	5	1
p3	2	2	4	2	0	2
p4	3	1	5	2	1	4
p5	4	3	13	9	6	10
p6	5	2	7	2	0	5

P1	P2	P3	P4	P5	P6	P7	P8	P9
0	1	2	3	4	5	6	7	8

Avg Waiting Time = 24/6

The Gantt chart is prepared according to the arrival and burst time given in the table.

1. نظرًا لأنه في الوقت 0 ، فإن العملية الوحيدة المتاحة هي P1 مع وقت اندفاع وحدة المعالجة المركزية 8. هذه هي العملية الوحيدة المتاحة في القائمة وبالتالي تمت جدولتها.
2. تصل العملية التالية إلى الوحدة الزمنية 1. نظرًا لأن الخوارزمية التي نستخدمها هي SRTF وهي خوارزمية استباقية، يتم إيقاف التنفيذ الحالي ويتحقق المجدول من العملية بأقل وقت للتنفيذ.
3. حتى الآن ، هناك عمليتان متاحتان في قائمة الانتظار الجاهزة. نفذ نظام التشغيل P1 لوحدة زمنية واحدة حتى الآن ؛ وقت التنفيذ المتبقي لـ P1 هو 7 وحدات. وقت تنفيذ العملية P2 هو 4 وحدات. ومن ثم تمت جدولة العملية P2 على وحدة المعالجة المركزية وفقًا للخوارزمية.
4. تصل العملية التالية P3 إلى الوحدة الزمنية 2. في هذا الوقت ، يتم إيقاف تنفيذ العملية P3 ويتم البحث في العملية ذات أقل وقت للتنفيذ المتبقي. نظرًا لأن العملية P3 تحتوي على وحدتين من الوقت ، فسيتم منحها الأولوية على الآخرين.
5. تصل العملية التالية P4 إلى الوحدة الزمنية 3. عند هذا الوصول ، سيوقف المجدول تنفيذ P4 ويتحقق من العملية التي لديها أقل وقت للتنفيذ بين العمليات المتاحة (P1) و P2 و P3 و P1. (P4 و P2 لهما الوقت المتبقي 7 وحدات و 3 وحدات على التوالي. P3و P4 لهما وقت متبقي 1 وحدة لكل منهما. نظرًا لأن كلاهما متساويان ، فسيتم إجراء الجدولة وفقًا لوقت وصولهما. يصل P3 في وقت أبكر من P4 وبالتالي سيتم جدولته مرة أخرى.
6. تصل العملية التالية P5 إلى الوحدة الزمنية 4. حتى هذا الوقت ، أكملت العملية P3 تنفيذها ولم تعد موجودة في القائمة. سيقارن المجدول وقت الاندفاع المتبقي لجميع العمليات المتاحة. بما أن زمن التنفيذ للعملية P4 هو 1 وهو الأقل من بين كل ذلك، فسيتم جدولته.
7. تصل العملية التالية P6 إلى الوحدة الزمنية 5 ، حتى هذا الوقت ، أكملت العملية P4 تنفيذها. لدينا 4 عمليات متاحة حتى الآن، وهي (7) P1 و (3) P2 و (3) P5 و .(2) P6 وقت التنفيذ لـ P6 هو الأقل بين جميع الأوقات ، ومن ثم تمت جدولة P6.، أصبحت جميع العمليات متاحة ، وبالتالي ستعمل الخوارزمية الآن مثل .SJF سيتم تنفيذ P6 حتى اكتماله ثم سيتم جدولة العملية مع أقل وقت متبقي.

بمجرد وصول جميع العمليات، لا يتم إجراء أي إجراءات وقائية وستعمل الخوارزمية مثل.SJF

Example: Given the arrival time and burst time of 3 jobs in the table below. Calculate the Average waiting time of the system.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P1	0	9	13	13	4
P2	1	4	5	4	0
P3	2	9	22	20	11

توجد ثلاث وظائف P1 و P2 و .P3 يصل P1 في الوحدة الزمنية 0 ؛ سيتم جدولته أولاً في الوقت حتى وصول العملية التالية. يصل P2 في وحدة زمنية واحدة. وقت تنفيذها هو 4 وحدات وهو الأقل بين الوظائف في قائمة الانتظار. ومن ثم سيتم جدولتها بعد ذلك وفي الوقت 2 ، سيصل P3 مع وقت تنفيذ 9. حيث أن وقت التنفيذ المتبقي لـ P2 هو 3 وحدات وهي الأقل من بين الوظائف المتاحة. ومن ثم سيستمر المعالج في تنفيذه حتى اكتماله. نظرًا لأن جميع الوظائف قد وصلت ، فلن يتم إجراء أي إجراءات استباقية الآن وسيتم تنفيذ جميع الوظائف حتى الانتهاء وفقًا لـ .SJF

P1	P2	P3	P4	P5	
0	1	2	5	13	22

Avg Waiting Time = (4+0+11)/3 = 5 units

خوارزمية جدولة روبن Round Robin Scheduling Algorithm

تعد خوارزمية جدولة Round Robin واحدة من أكثر خوارزمية الجدولة شيوعًا والتي يمكن تنفيذها بالفعل في معظم أنظمة التشغيل. هذه هي النسخة الاستباقية Preemptive version من جدولة من يأتي أولاً يُخدم أولاً. تركز الخوارزمية على مشاركة الوقت. في هذه الخوارزمية، يتم تنفيذ كل عملية بطريقة دورية. يتم تحديد شريحة زمنية معينة في النظام تسمى كمية الوقت Quantum يتم تخصيص وحدة المعالجة المركزية لكل عملية موجودة في قائمة الانتظار الجاهزة لهذا الوقت الكمي، إذا اكتمل تنفيذ العملية خلال ذلك الوقت، فستنتهي العملية وإلا ستعود العملية إلى قائمة الانتظار الجاهزة Ready queue وتنتظر حتى يكتمل الدور التالي للتنفيذ.

مزايا Advantages

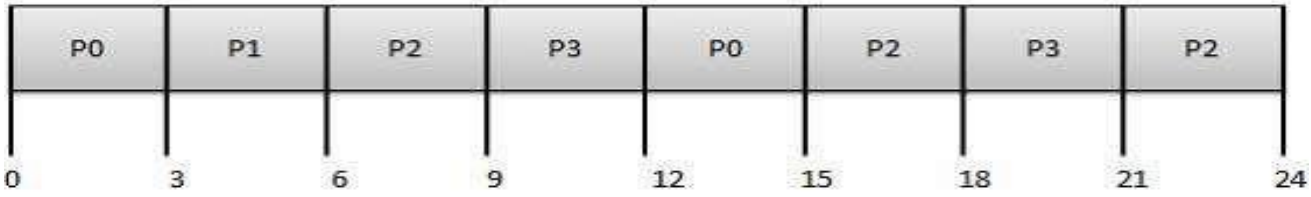
- يمكن أن يكون قابلاً للتنفيذ فعليًا في النظام لأنه لا يعتمد على وقت التنفيذ
- لا يعاني من مشكلة الجوع starvation
- تحصل جميع الوظائف على تخصيص ثابت لوحدة المعالجة المركزية.

سلبيات Disadvantages

- كلما زاد مقدار الوقت ، زاد وقت الاستجابة في النظام.
- كلما انخفض مقدار الوقت ، زادت كلفة التبديل في النظام.
- تحديد الوقت المثالي time quantum هو في الحقيقة مهمة صعبة للغاية في النظام.

Process	Arrival Time	Execute Time
P0	0	5
P1	1	3
P2	2	8
P3	3	6

Quantum = 3



Wait time of each process is following

Process	Wait Time : Service Time - Arrival Time
P0	(0-0) + (12-3) = 9
P1	(3-1) = 2
P2	6-2) + (15-9) = 10
P3	(9-3) + (18-12) = 12

Average Wait Time: (9+2+10+12) / 4 = 8.25

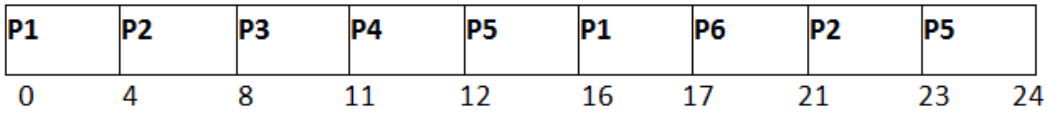
RR Scheduling Example

In the following example, there are six processes named as P1, P2, P3, P4, P5 and P6. Their arrival time and burst time are given below in the table. The time quantum of the system is 4 units.

Process ID	Arrival Time	Burst Time
P1	0	5
P2	1	6
P3	2	3
P4	3	1
P5	4	5
P6	6	4

According to the algorithm, we have to maintain the ready queue and the Gantt chart. The structure of both the data structures will be changed after every scheduling.

GANTT chart



The completion time, Turnaround time and waiting time will be calculated as shown in the table below.

As, we know,

1. Turn Around Time = Completion Time - Arrival Time
2. Waiting Time = Turn Around Time - Burst Time

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P1	0	5	17	17	12
P2	1	6	23	22	16
P3	2	3	11	9	6
P4	3	1	12	9	8
P5	4	5	24	20	15
P6	6	4	21	15	11

Avg Waiting Time = (12+16+6+8+15+11)/6 = 76/6 units

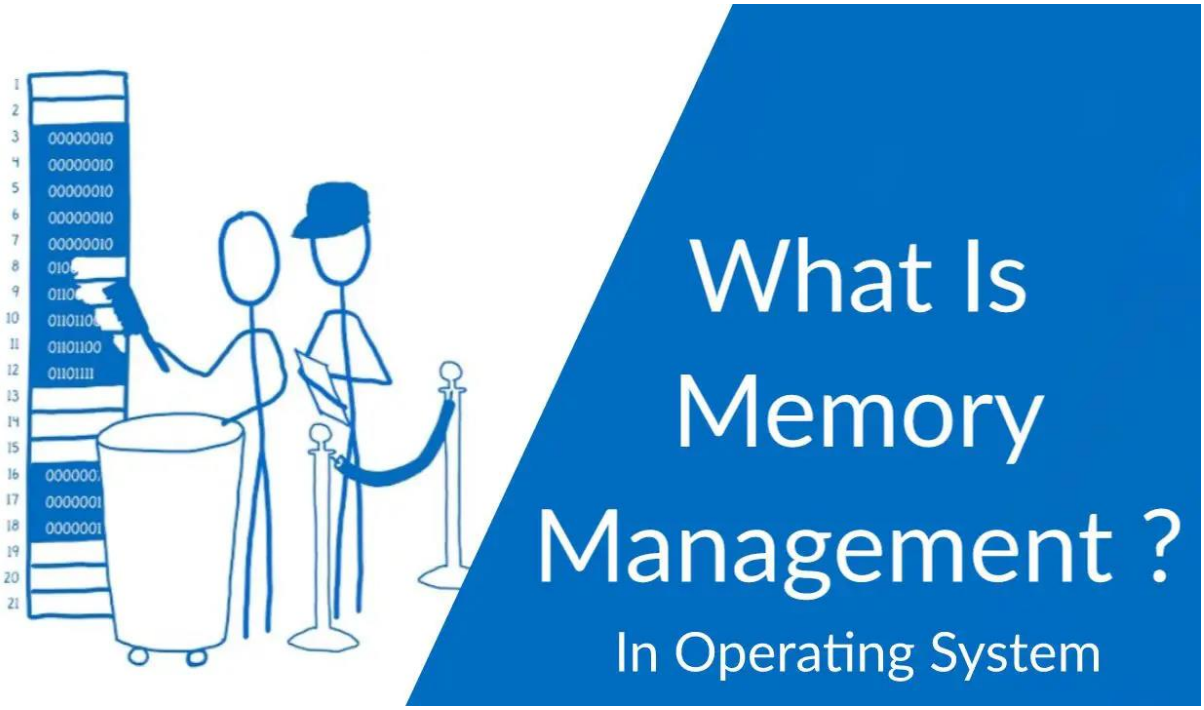
Operating Systems

Lecture # 10

Department of Computer

4th Class

Memory management in Operating System



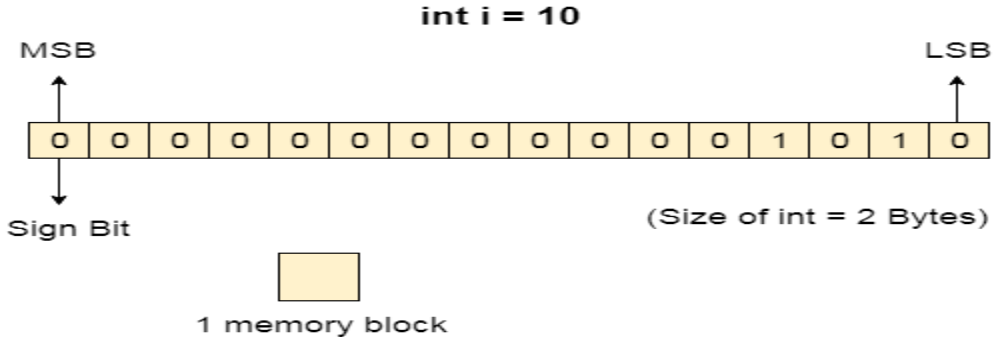
ما هي الذاكرة Memory ؟

يمكن تعريف ذاكرة الكمبيوتر على أنها مجموعة من البيانات الممثلة في تنسيق ثنائي ، ويمكن تصنيف الذاكرة إلى فئات مختلفة ويسمى جهاز الكمبيوتر القادر على تخزين أي معلومات أو بيانات بشكل مؤقت أو دائم، بجهاز التخزين.

كيف يتم تخزين البيانات في نظام الكمبيوتر؟ How Data is being stored in a computer system

لفهم إدارة الذاكرة، يتعين علينا توضيح كل شيء حول كيفية تخزين البيانات في نظام الكمبيوتر.

يفهم الجهاز اللغة الثنائية فقط التي تتكون من 0 أو 1 ويحول الكمبيوتر كل البيانات إلى لغة ثنائية أولاً ثم يخزنها في الذاكرة. هذا يعني أنه إذا كان لدينا سطر برنامج مكتوب بلغة برمجية مثل (int α = 10) ، فسوف يقوم الكمبيوتر بتحويله إلى اللغة الثنائية Binary language ثم يخزنه في كتل الذاكرة ويظهر كما في الشكل ادناه



الحاجة إلى البرمجة المتعددة Need for Multi programming

يمكن لوحدة المعالجة المركزية الوصول مباشرة إلى الذاكرة الرئيسية والسجلات وذاكرة التخزين المؤقت للنظام ويتم تنفيذ البرنامج دائمًا في الذاكرة الرئيسية كما ويؤثر حجم الذاكرة الرئيسية على درجة البرمجة المتعددة لمعظم ما هو موجود فإذا كان حجم الذاكرة الرئيسية أكبر ، فيمكن لوحدة المعالجة المركزية تحميل المزيد من العمليات في الذاكرة الرئيسية في نفس الوقت ، وبالتالي ستزيد من درجة البرمجة المتعددة وكذلك استخدام وحدة المعالجة المركزية.

إدارة الذاكرة في نظام التشغيل (OS) Memory Management in Operating System (OS)

الذاكرة هي الجزء المهم من الكمبيوتر المستخدم لتخزين البيانات وتعد إدارتها أمرًا بالغ الأهمية لأن حجم الذاكرة الرئيسية المتوفرة في نظام الكمبيوتر محدود للغاية وتتنافس العديد من العمليات على ذلك علاوة على ذلك ، لزيادة الأداء ، يتم تنفيذ العديد من العمليات في وقت واحد. لهذا ، يجب أن نحتفظ بالعديد من العمليات في الذاكرة الرئيسية وذلك من الأهمية لا إدارتها بفعالية.

دور إدارة الذاكرة Role of Memory management

فيما يلي الأدوار المهمة لإدارة الذاكرة في نظام الكمبيوتر:

- مدير الذاكرة يستخدم لتتبع حالة مواقع الذاكرة سواء كانت خالية من البيانات أو تحتوي على البيانات.
- يسمح مدير الذاكرة لأجهزة الكمبيوتر التي تحتوي على كمية صغيرة من الذاكرة الرئيسية بتنفيذ برامج أكبر من حجم أو مقدار الذاكرة المتوفرة ويقوم بذلك عن طريق نقل المعلومات ذهابًا وإيابًا بين الذاكرة الأولية والذاكرة الثانوية باستخدام مفهوم المبادلة Swapping .

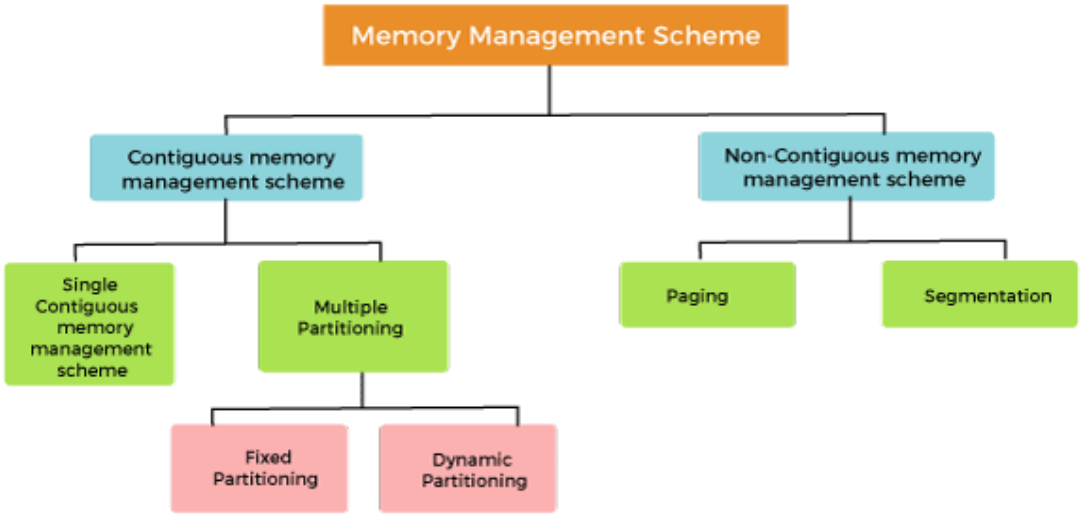
- مدير الذاكرة مسؤول عن حماية الذاكرة المخصصة لكل عملية من التلف بسبب عملية أخرى .
- يجب على مدير الذاكرة تمكين مشاركة مساحة الذاكرة بين العمليات وبالتالي ، يمكن أن يتواجد برنامجان في نفس موقع الذاكرة ولكن في أوقات مختلفة.

توفر إدارة الذاكرة الحماية باستخدام سجلين ، سجل أساسي **base register** وسجل حد **limit register** . يحتفظ السجل الأساسي بأصغر عنوان للذاكرة الفعلية القانونية physical memory ويحدد سجل الحد حجم النطاق.

تقنيات إدارة الذاكرة Memory Management Techniques

يمكن تصنيف تقنيات إدارة الذاكرة إلى الفئات الرئيسية التالية:

- إدارة الذاكرة المجاورة Contiguous memory management schemes
- إدارة الذاكرة غير المتجاورة Non-Contiguous memory management schemes



Classification of memory management schemes

:

مخططات إدارة الذاكرة المجاورة Contiguous memory management schemes

في مخطط إدارة الذاكرة المتجاورة، يحتل كل برنامج كتلة واحدة متجاورة من مواقع التخزين، أي مجموعة من مواقع الذاكرة بعناوين متتالية.

مخططات إدارة الذاكرة المتجاورة الفردية Single contiguous memory management schemes

يعد نظام إدارة الذاكرة المتجاورة المفردة أبسط مخطط لإدارة الذاكرة يستخدم في الجيل الأول من أنظمة الكمبيوتر. في هذا المخطط، يتم تقسيم الذاكرة الرئيسية إلى منطقتين أو أقسام متجاورة. نظام التشغيل في قسم وعملية المستخدم في القسم الاخر.

مزايا أنظمة إدارة الذاكرة المتجاورة Advantages of Single contiguous memory management schemes

1. سهل التنفيذ.
2. سهل الإدارة والتصميم.
3. في نظام إدارة ذاكرة واحد متجاور ، بمجرد تحميل العملية ، يتم منحها وقت المعالج الكامل

عيوب أنظمة إدارة الذاكرة المتجاورة الفردية Disadvantages of Single contiguous memory management schemes

1. إهدار مساحة الذاكرة بسبب الذاكرة غير المستخدمة حيث من غير المحتمل أن تستخدم العملية كل مساحة الذاكرة المتاحة.
2. تظل وحدة المعالجة المركزية خاملة.
3. لا يمكن تنفيذه إذا كان البرنامج كبيرًا جدًا بحيث لا يتناسب مع مساحة الذاكرة الرئيسية المتاحة بالكامل.
4. لا يدعم البرمجة المتعددة، أي أنه لا يمكنه التعامل مع برامج متعددة في نفس الوقت.

التقسيم المتعدد Multiple Partitioning

يعتبر نظام إدارة الذاكرة المتجاورة غير فعال لأنه يحد من أجهزة الكمبيوتر لتنفيذ برنامج واحد فقط في كل مرة مما يؤدي إلى إهدار مساحة الذاكرة ووقت وحدة المعالجة المركزية ويمكن التغلب على مشكلة الاستخدام غير الفعال لوحدة المعالجة المركزية باستخدام البرمجة المتعددة التي تسمح بتشغيل أكثر من برنامج في وقت واحد للتبديل بين عمليتين وتحتاج أنظمة التشغيل إلى تحميل كلتا العمليتين في الذاكرة الرئيسية ويحتاج نظام التشغيل إلى تقسيم الذاكرة الرئيسية المتاحة إلى أجزاء متعددة لتحميل عمليات متعددة في الذاكرة الرئيسية وبالتالي يمكن أن توجد عمليات متعددة في الذاكرة الرئيسية في وقت واحد.

يمكن أن تكون أنظمة التقسيم المتعددة من نوعين:

1. التقسيم الثابت Fixed Partitioning
2. التقسيم الديناميكي Dynamic Partitioning

التقسيم الثابت Fixed Partitioning

أقدم وأبسط التقنيات التي يمكن استخدامها لتحميل أكثر من عملية واحدة في الذاكرة الرئيسية هي التقسيم الثابت أو تخصيص الذاكرة المتجاورة. في هذه التقنية ، يتم تقسيم الذاكرة الرئيسية إلى أقسام ذات أحجام متساوية أو مختلفة ويوجد نظام التشغيل دائمًا في القسم الأول بينما يمكن استخدام الأقسام الأخرى لتخزين عمليات المستخدم ويتم تخصيص الذاكرة للعمليات بطريقة متجاورة.

في التقسيم الثابت:

1. لا يمكن أن تتداخل الأقسام.
2. يجب أن تكون العملية موجودة بشكل متواصل في قسم للتنفيذ.

هناك عيوب مختلفة لاستخدام هذه التقنية:

1. التجزئة الداخلية Internal Fragmentation

إذا كان حجم العملية أقل ، فسيتم إهدار حجم القسم ويظل غير مستخدم وهذا هو إهدار للذاكرة ويسمى التجزئة الداخلية كما هو موضح في الصورة أدناه ، يتم استخدام قسم 4 ميغا بايت لتحميل عملية 3 ميغا بايت فقط ويتم إهدار 1 ميغا بايت المتبقية.

2. التجزئة الخارجية External Fragmentation

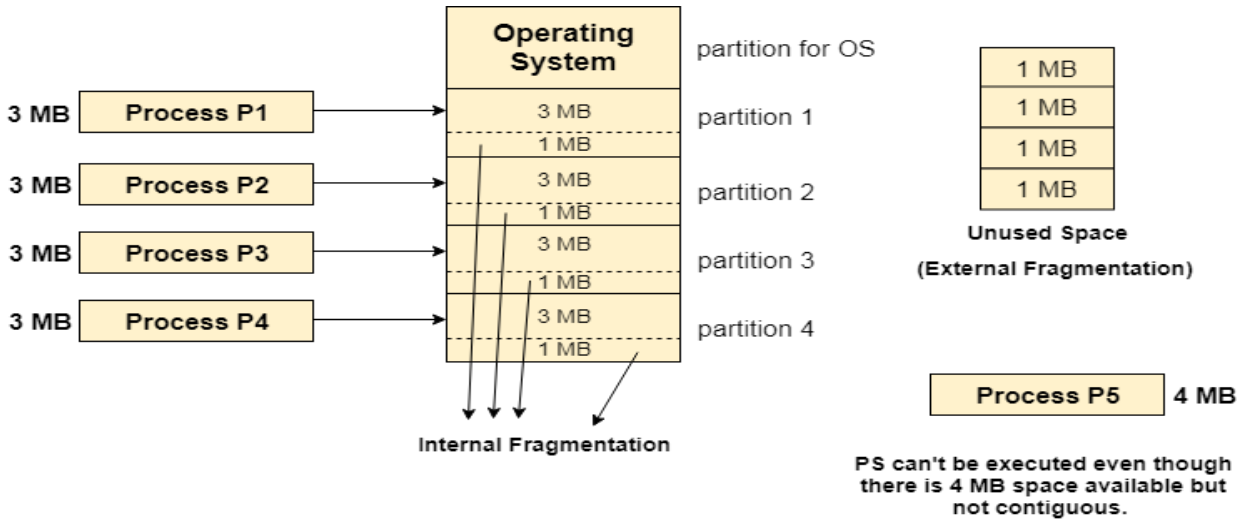
لا يمكن استخدام المساحة الإجمالية غير المستخدمة لأقسام مختلفة لتحميل العمليات على الرغم من وجود مساحة متاحة ولكن ليس في الشكل المتجاور كما هو موضح في الصورة أدناه، لا يمكن استخدام المساحة المتبقية 1 ميجا بايت لكل قسم كوحدة لتخزين عملية 4 ميجا بايت. على الرغم من توفر المساحة الكافية لتحميل العملية، لن يتم تحميل العملية.

3. تحديد حجم العملية Limitation on the size of the process

إذا كان حجم العملية أكبر من حجم القسم الأقصى، فلا يمكن تحميل هذه العملية في الذاكرة. لذلك، يمكن فرض قيود على حجم العملية بحيث لا يمكن أن تكون أكبر من حجم القسم الأكبر.

4. درجة تعدد البرمجة أقل Degree of multiprogramming is less

حسب درجة البرمجة المتعددة ، فإننا نعني ببساطة الحد الأقصى لعدد العمليات التي يمكن تحميلها في الذاكرة في نفس الوقت. في التقسيم الثابت ، تكون درجة البرمجة المتعددة ثابتة وأقل جدًا نظرًا لحقيقة أن حجم القسم لا يمكن تغييره وفقًا لحجم العمليات



Fixed Partitioning

(Contiguous memory allocation)

مزايا أنظمة إدارة ذاكرة التقسيم الثابت Advantages of Fixed Partitioning memory management schemes

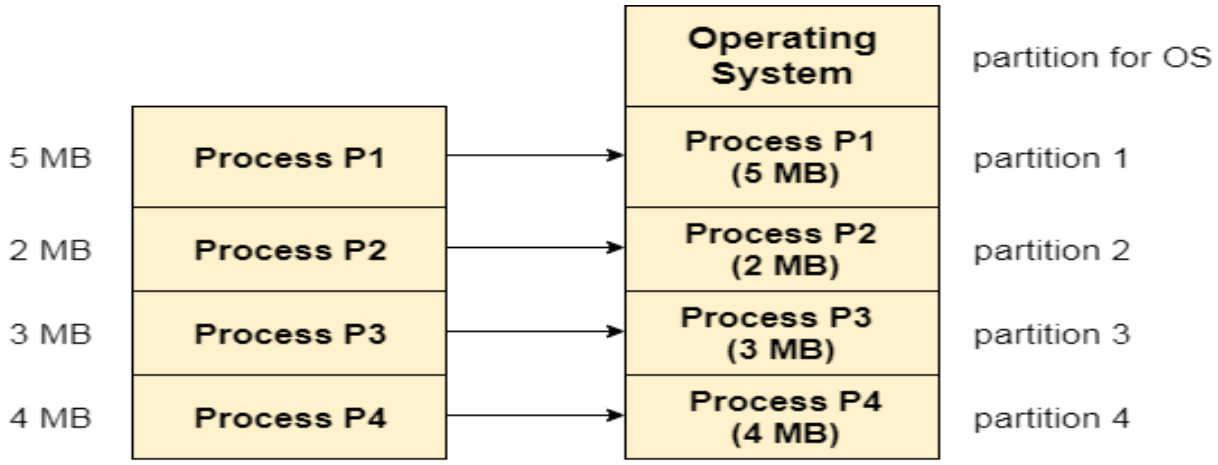
2. سهل التنفيذ.
3. سهل الإدارة والتصميم.

عيوب أنظمة إدارة ذاكرة التقسيم الثابت Disadvantages of Fixed Partitioning memory management schemes

1. يعاني هذا المخطط من تجزئة داخلية.
2. يتم تحديد عدد الأقسام في وقت إنشاء النظام.

التقسيم الديناميكي Dynamic Partitioning

يحاول التقسيم الديناميكي التغلب على المشاكل الناتجة عن التقسيم الثابت وفي هذه التقنية ، لا يتم التصريح عن حجم القسم في البداية وتم الإعلان عنه في وقت تحميل العملية ويكون القسم الأول محجوز لنظام التشغيل اما المساحة المتبقية مقسمة إلى أجزاء وسوف يكون حجم كل قسم مساويًا لحجم العملية يختلف حجم القسم حسب حاجة العملية بحيث يمكن تجنب التجزئة الداخلية.



Dynamic Partitioning

(Process Size = Partition Size)

مزايا التقسيم الديناميكي على التقسيم الثابت Advantages of Dynamic Partitioning over fixed partitioning

1. لا تجزئة داخلية No Internal Fragmentation

بالنظر إلى حقيقة أن الأقسام في التقسيم الديناميكي يتم إنشاؤها وفقًا لاحتياجات العملية ، فمن الواضح أنه لن يكون هناك أي تجزئة داخلية لأنه لن يكون هناك أي مساحة متبقية غير مستخدمة في القسم.

2. لا قيود على حجم العملية No Limitation on the size of the process

في التقسيم الثابت، لا يمكن تنفيذ العملية ذات الحجم الأكبر من حجم القسم الأكبر بسبب عدم وجود ذاكرة متجاورة كافية هنا ، في التقسيم الديناميكي ، لا يمكن تقييد حجم العملية حيث يتم تحديد حجم القسم وفقًا لحجم العملية.

3. درجة البرمجة المتعددة ديناميكية Degree of multiprogramming is dynamic

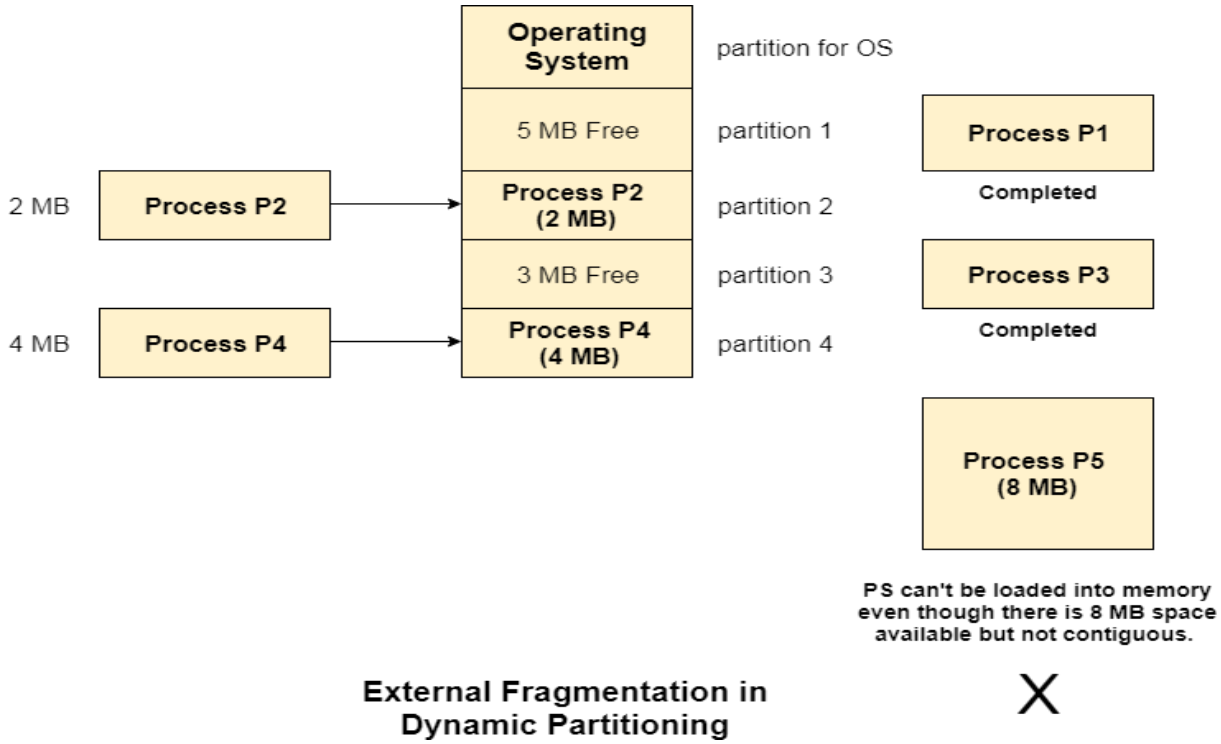
نظرًا لعدم وجود تجزئة داخلية ، لن يكون هناك أي مساحة غير مستخدمة في القسم ، وبالتالي يمكن تحميل المزيد من العمليات في الذاكرة في نفس الوقت.

مساوئ التقسيم الديناميكي Disadvantages of dynamic partitioning

التجزئة الخارجية External Fragmentation

لا يعني عدم وجود التجزئة الداخلية أنه لن يكون هناك تجزئة خارجية. لنفكر في ثلاث عمليات (1 MB) P1 و (3 MB) P2 و (1 MB) P3 يتم تحميلها في الأقسام المعنية من الذاكرة الرئيسية.

بعد مرور بعض الوقت ، تم الانتهاء من P1 و P3 وتم تحرير المساحة المخصصة لهما. يوجد الآن قسمان غير مستخدمين (1 ميجابايت و 1 ميجابايت) متاحان في الذاكرة الرئيسية ولكن لا يمكن استخدامهما لتحميل عملية 2 ميجابايت في الذاكرة نظرًا لعدم وجودهما في مكان قريب.



External Fragmentation in Dynamic Partitioning

مخططات إدارة الذاكرة غير المتجاورة Non-Contiguous memory management schemes

في نظام إدارة الذاكرة غير المتجاورة ، ينقسم البرنامج إلى كتل مختلفة ويتم تحميله في أجزاء مختلفة من الذاكرة لا يلزم بالضرورة أن تكون متجاورة مع بعضها البعض ويمكن تصنيف هذا المخطط اعتمادًا على حجم الكتل وما إذا كانت الكتل موجودة في الذاكرة الرئيسية أم لا.

الصفحات Paging

نظام الصفحات او الترحيل هو أسلوب يلغي متطلبات التخصيص المتجاور للذاكرة الرئيسية ويتم تقسيم الذاكرة الرئيسية إلى كتل ذات حجم ثابت من الذاكرة الفعلية تسمى الإطارات Frames

مزايا نظام الصفحات Advantages

1. تقلل من التجزئة الخارجية.
2. سهل التنفيذ.
3. يزيد من كفاءة الذاكرة.
4. نظرًا لحجم الإطارات المتساوي ، يصبح التبديل أمرًا سهلاً للغاية.
5. يتم استخدامه للوصول السريع إلى البيانات.

التقسيم (التجزئة) Segmentation

التجزئة هي تقنية تلغي متطلبات التخصيص المتجاور للذاكرة الرئيسية ويتم تقسيم الذاكرة الرئيسية إلى كتل متغيرة الحجم من الذاكرة الفعلية تسمى المقاطع ويعتمد على الطريقة التي يتبعها المبرمج لهيكلة برامجه ومع تخصيص الذاكرة المجزأة ، يتم تقسيم كل مهمة إلى عدة أقسام بأحجام مختلفة.

خوارزميات التقسيم Partitioning Algorithms

هناك العديد من الخوارزميات التي يتم تنفيذها بواسطة نظام التشغيل من أجل اكتشاف الثغرات الموجودة في القائمة المرتبطة وتخصيصها للعمليات وفيما يلي شرح حول كل خوارزمية.

1. First Fit Algorithm

تقوم خوارزمية First Fit بمسح القائمة المرتبطة وعندما تجد أول مساحة كبيرة بما يكفي لتخزين عملية ما ، فإنها تتوقف عن المسح وتحميل العملية في تلك المساحة وتحتفظ خوارزمية First Fit بالقائمة المرتبطة وفقًا للترتيب المتزايد لمؤشر البداية وهذا هو أبسط تطبيق بين جميع الخوارزميات وينتج فراغات أكبر مقارنة بالخوارزميات الأخرى.

2 . **Best Fit Algorithm**

تحاول خوارزمية Best Fit اكتشاف أصغر فراغ ممكن في القائمة يمكنه استيعاب متطلبات الحجم للعملية.

استخدام Best Fit له بعض العيوب.

1. انه أبطأ لأنه يمسح القائمة بأكملها في كل مرة ويحاول اكتشاف أصغر فراغ يمكنه تلبية متطلبات العملية.
2. نظرًا لحقيقة أن الفرق بين الحجم الكلي وحجم العملية صغير جدًا ، فإن الفراغات الناتجة ستكون صغيرة بحيث لا يمكن استخدامها لتحميل أي عملية وبالتالي تظل غير مجدية. على الرغم من حقيقة أن هذه الخوارزمية هي الأنسب ، إلا أنها ليست أفضل خوارزمية بين الجميع.

3. **Worst Fit Algorithm**

تقوم الخوارزمية الأسوأ بمسح القائمة بأكملها في كل مرة وتحاول اكتشاف أكبر فراغ في القائمة والتي يمكن أن تفي بمتطلبات العملية. على الرغم من حقيقة أن هذه الخوارزمية تنتج فراغات أكبر لتحميل العمليات الأخرى ، إلا أن هذا ليس النهج الأفضل نظرًا لكونه أبطأ لأنه يبحث في القائمة بأكملها في كل مرة مرارًا وتكرارًا.

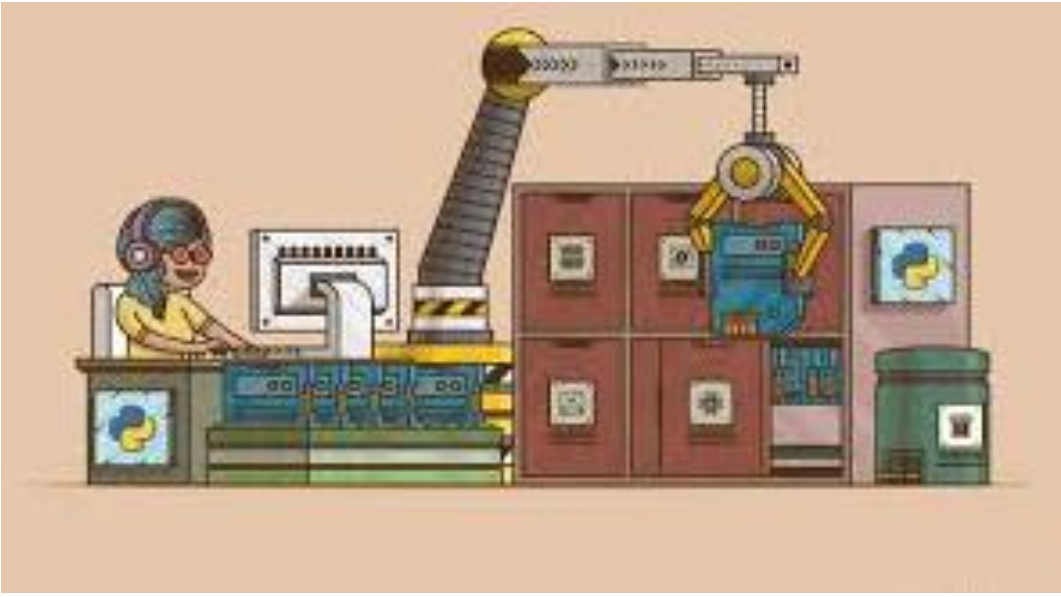
Operating Systems

Lecture # 11

Department of Computer

4th Class

Memory management allocation algorithms



هناك العديد من الخوارزميات التي يتم تنفيذها بواسطة نظام التشغيل من أجل اكتشاف الثغرات الموجودة في القائمة المرتبطة وتخصيصها للعمليات. فيما يلي شرح حول كل من الخوارزمية

These partitions may be allocated by 4 ways:

1. First-Fit Memory Allocation

2. Best-Fit Memory Allocation

3. Worst-Fit Memory Allocation

4. Next-Fit Memory Allocation

Best-Fit Memory Allocation:

This method keeps the free/busy list in order by size – smallest to largest. In this method, the operating system first searches the whole of the memory according to the size of the given job and allocates it to the closest-fitting free partition in the memory, making it able to use memory efficiently. Here the jobs are in the order from smallest job to largest job.

As illustrated in above figure, the operating system first search throughout the memory and allocates the job to the minimum possible memory partition, making the memory allocation efficient.

Advantages of Best-Fit Allocation :

Memory Efficient. The operating system allocates the job minimum possible space in the memory, making memory management very efficient.

To save memory from getting wasted, it is the best method.

- Improved memory utilization
- Reduced memory fragmentation
- Minimizes external fragmentation

Disadvantages of Best-Fit Allocation :

It is a Slow Process. Checking the whole memory for each job makes the working of the operating system very slow. It takes a lot of time to complete the work.

- Increased computational overhead
- May lead to increased internal fragmentation
- Can result in slow memory allocation times.

Best-fit allocation is a memory allocation algorithm used in operating systems to allocate memory to processes. In this algorithm, the operating system searches for the smallest free block of memory that is big enough to accommodate the process being allocated memory.

The operating system maintains a list of all free memory blocks available in the system.

When a process requests memory, the operating system searches the list for the smallest free block of memory that is large enough to accommodate the process.

If a suitable block is found, the process is allocated memory from that block.

If no suitable block is found, the operating system can either wait until a suitable block becomes available or request additional memory from the system.

The best-fit allocation algorithm has the advantage of minimizing external fragmentation, as it searches for the smallest free block of memory that can accommodate a process. However, it can also lead to more internal fragmentation, as processes may not use the entire memory block allocated to them.

Overall

the best-fit allocation algorithm can be an effective way to allocate memory in an operating system, but it is important to balance the advantages and disadvantages of this approach with other allocation algorithms such as first-fit, next-fit, and worst-fit.

Worst-Fit Memory Allocation :

In this allocation technique, the process traverses the whole memory and always search for the largest hole/partition, and then the process is placed in that hole/partition. It is a slow process because it has to traverse the entire memory to search the largest hole.

Advantages of Worst-Fit Allocation :

Since this process chooses the largest hole/partition, therefore there will be large internal fragmentation. Now, this internal fragmentation will be quite big so that other small processes can also be placed in that leftover partition.

Disadvantages of Worst-Fit Allocation :

It is a slow process because it traverses all the partitions in the memory and then selects the largest partition among all the partitions, which is a time-consuming process.

First-Fit Allocation is a memory allocation

First-Fit Allocation is a memory allocation technique used in operating systems to allocate memory to a process. In First-Fit, the operating system searches through the list of free blocks of memory, starting from the beginning of the list, until it finds a block that is large enough to accommodate the memory request from the process. Once a suitable block is found, the operating system splits the block into two parts: the portion that will be allocated to the process, and the remaining free block.

Advantages of First-Fit Allocation

Advantages of First-Fit Allocation include its simplicity and efficiency, as the search for a suitable block of memory can be performed quickly and easily. Additionally, First-Fit can also help to minimize memory fragmentation, as it tends to allocate memory in larger blocks.

Disadvantages of First-Fit Allocation

Disadvantages of First-Fit Allocation include poor performance in situations where the memory is highly fragmented, as the search for a suitable block of memory can become time-consuming and inefficient. Additionally, First-Fit can also lead to poor memory utilization, as it may allocate larger blocks of memory than are actually needed by a process.

Overall

First-Fit Allocation is a widely used memory allocation technique in operating systems, but its effectiveness may vary depending on the specifics of the system and the workload being executed.

As illustrated above, the system assigns J1 the nearest partition in the memory. As a result, there is no partition with sufficient space is available for J3 and it is placed in the waiting list.The processor ignores if the size of partition allocated to the job is very large as compared to the size of job or not. It just allocates the memory. As a result, a lot of memory is wasted and many jobs may not get space in the memory, and would have to wait for another job to complete.

Comparison of Partition Allocation Methods:

Sl.No.	Partition Allocation Method	Advantages	Disadvantages
1.	Fixed Partition	Simple, easy to use, no complex algorithms needed	Memory waste, inefficient use of memory resources
2.	Dynamic Partition	Flexible, more efficient, partitions allocated as required	Requires complex algorithms for memory allocation
3.	Best-fit Allocation	Minimizes memory waste, allocates smallest suitable partition	More computational overhead to find smallest split
4.	Worst-fit Allocation	Ensures larger processes have sufficient memory	May result in substantial memory waste
5.	First-fit Allocation	Quick, efficient, less computational work	Risk of memory fragmentation

Exercise: Consider the requests from processes in given order 300K, 25K, 125K, and 50K. Let there be two blocks of memory available of size 150K followed by a block size 350K.

Which of the following partition allocation schemes can satisfy the above requests?
A) Best fit but not first fit.
B) First fit but not best fit.
C) Both First fit & Best fit.
D) neither first fit nor best fit.

Solution: Let us try all options.
Best Fit:
300K is allocated from a block of size 350K. 50 is left in the block.
25K is allocated from the remaining 50K block. 25K is left in the block.
125K is allocated from 150 K block. 25K is left in this block also.
50K can't be allocated even if there is 25K + 25K space available.

First Fit:
300K request is allocated from 350K block, 50K is left out.
25K is allocated from the 150K block, 125K is left out.
Then 125K and 50K are allocated to the remaining left out partitions.
So, the first fit can handle requests.

So option B is the correct choice.

Program for First Fit algorithm in Memory Management

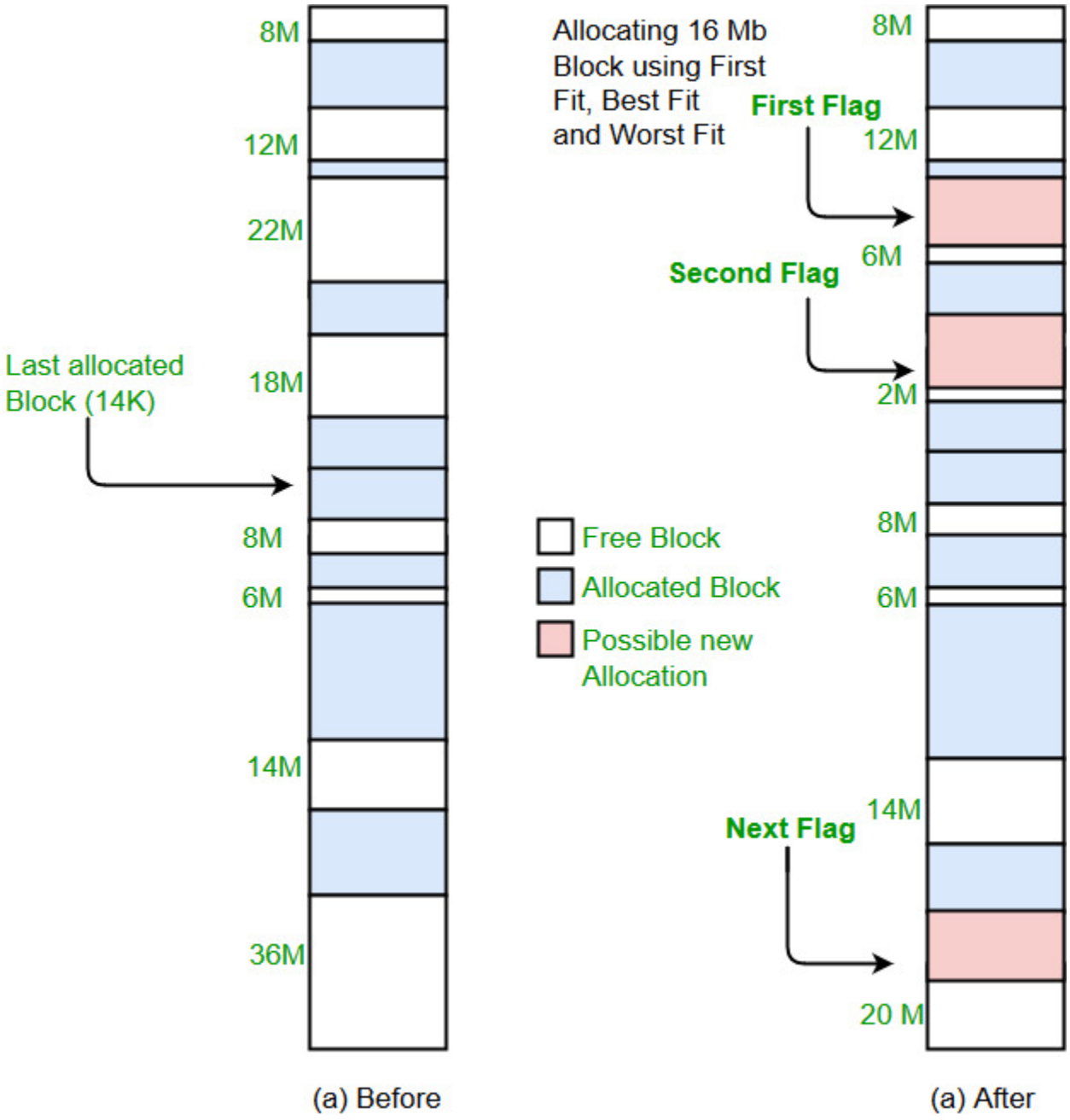
In the first fit, the partition is allocated which is first sufficient from the top of Main Memory.
Example :

Input : blockSize[] = {100, 500, 200, 300, 600};
processSize[] = {212, 417, 112, 426};
Output:

Process No.	Process Size	Block no.
1	212	2
2	417	5
3	112	2
4	426	Not Allocated

- Its advantage is that it is the fastest search as it searches only the first block i.e. enough to assign a process.
- It may have problems of not allowing processes to take space even if it was possible to allocate. Consider the above example, process number 4 (of size 426) does not get memory. However it was possible to allocate memory if we had allocated using best fit allocation [block number 4 (of size 300) to process 1, block number 2 to process 2, block number 3 to process 3 and block number 5 to process 4].

Implementation:
1- Input memory blocks with size and processes with size.
2- Initialize all memory blocks as free.
3- Start by picking each process and check if it can
be assigned to current block.
4- If size-of-process <= size-of-block if yes then
assign and check for next process.
5- If not then keep checking the further blocks.


8M
12M
22M
18M
8M
6M
14M
36M
Last allocated
Block (14K)
(a) Before
Allocating 16 Mb
Block using First
Fit, Best Fit
and Worst Fit
First Flag
Second Flag
Next Flag
Free Block
Allocated Block
Possible new
Allocation
8M
12M
6M
2M
8M
6M
14M
20 M
(a) After

```
// C++ implementation of First - Fit algorithm
#include<bits/stdc++.h>
using namespace std;
// Function to allocate memory to
// blocks as per First fit algorithm
void firstFit(int blockSize[ ], int m,
                        int processSize[ ], int n)
{
        // Stores block id of the
        // block allocated to a process
        int allocation[n];
        // Initially no block is assigned to any process
        memset(allocation, -1, sizeof(allocation));
        // pick each process and find suitable blocks
        // according to its size ad assign to it
        for (int i = 0; i < n; i++)
        {
                for (int j = 0; j < m; j++)
                {
                        if (blockSize[j] >= processSize[i])
                        {
                                // allocate block j to p[i] process
                                allocation[i] = j;

                                // Reduce available memory in this block.
                                blockSize[j] -= processSize[i];

                                break;
                        }
                }
```

```
        }
        cout << "\nProcess No.\tProcess Size\tBlock no.\n";
        for (int i = 0; i < n; i++)
        {
                cout << " " << i+1 << "\t\t"
                        << processSize[i] << "\t\t";
                if (allocation[i] != -1)
                        cout << allocation[i] + 1;
                else
                        cout << "Not Allocated";
                cout << endl;
        }
}
```

// Driver code

```
int main()
{
        int blockSize[] = {100, 500, 200, 300, 600};
        int processSize[] = {212, 417, 112, 426};
        int m = sizeof(blockSize) / sizeof(blockSize[0]);
        int n = sizeof(processSize) / sizeof(processSize[0]);
        firstFit(blockSize, m, processSize, n);
        return 0 ;
}
```

Output :

Process No.	Process Size	Block no.
1	212	2
2	417	5
3	112	2
4	426	Not Allocated

Time complexity of First Fit algorithm is **O(n*m)**, where n is the number of processes and m is the number of memory blocks. The outer for loop runs for n times and the inner for loop runs for m times.
Auxiliary Space of First Fit algorithm is **O(n)**, where n is the number of processes. The allocation array is used to store the block number allocated to the process, which takes a space of O(n).

Program for Best Fit algorithm in Memory Management

Best fit allocates the process to a partition which is the smallest sufficient partition among the free available partitions.

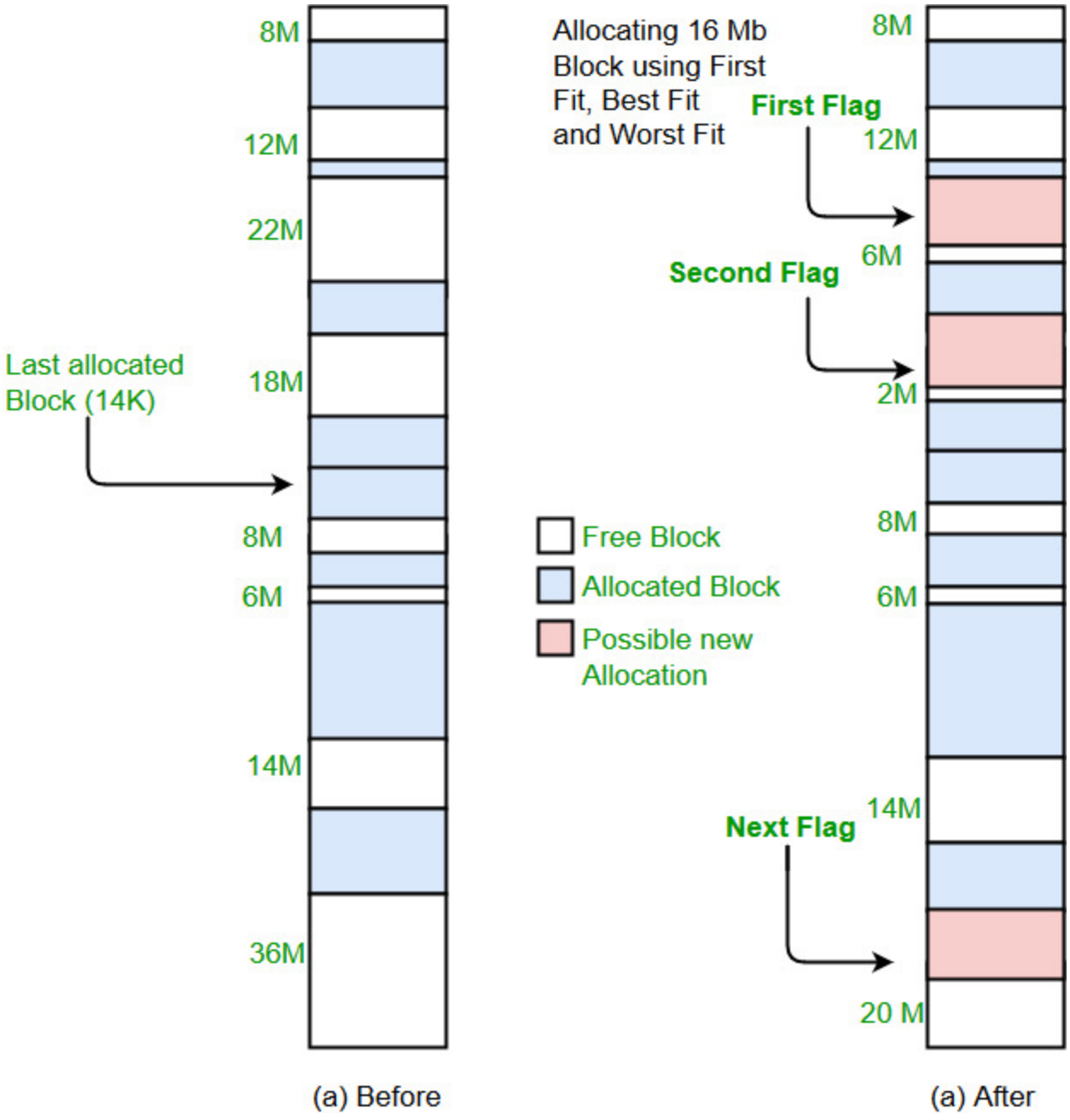
Example:

Input : blockSize[] = {100, 500, 200, 300, 600};

processSize[] = {212, 417, 112, 426};

Output:

Process No.	Process Size	Block no.
1	212	4
2	417	2
3	112	3
4	426	5



Implementation:

1- Input memory blocks and processes with sizes.

2- Initialize all memory blocks as free.

3- Start by picking each process and find the minimum block size that can be assigned to current process i.e., find min(bockSize[1],

blockSize[2],.....blockSize[n]) >

processSize[current], if found then assign

it to the current process.

5- If not then leave that process and keep checking

the further processes.

// C++ implementation of Best - Fit algorithm

```
#include<iostream>
using namespace std;
 // Method to allocate memory to blocks as per Best fit algorithm
void bestFit(int blockSize[], int m, int processSize[], int n)
{
    // Stores block id of the block allocated to a process
    int allocation[n];
     // Initially no block is assigned to any process
    for (int i = 0; i < n; i++)
        allocation[i] = -1;
     // pick each process and find suitable blocks
    // according to its size ad assign to it
    for (int i = 0; i < n; i++)
    {
        // Find the best fit block for current process
        int bestIdx = -1;
        for (int j = 0; j < m; j++)
        {
            if (blockSize[j] >= processSize[i])
            {
                if (bestIdx == -1)
                    bestIdx = j;
```

```
            else if (blockSize[bestIdx] > blockSize[j])
                bestIdx = j;
        }
      }
       // If we could find a block for current process
      if (bestIdx != -1)
      {
        // allocate block j to p[i] process
        allocation[i] = bestIdx;

        // Reduce available memory in this block.
        blockSize[bestIdx] -= processSize[i];
      }
   }
    cout << "\nProcess No.\tProcess Size\tBlock no.\n";
   for (int i = 0; i < n; i++)
   {
      cout << " " << i+1 << "\t\t" << processSize[i] << "\t\t";
      if (allocation[i] != -1)
        cout << allocation[i] + 1;
      else
        cout << "Not Allocated";
      cout << endl;
   }
}

// Driver Method
int main()
```

```
{
   int blockSize[] = {100, 500, 200, 300, 600};
   int processSize[] = {212, 417, 112, 426};
   int m = sizeof(blockSize) / sizeof(blockSize[0]);
   int n = sizeof(processSize) / sizeof(processSize[0]);


   bestFit(blockSize, m, processSize, n);
   return 0 ;
}
```

Output:

Process No.	Process Size	Block no.
1	212	4
2	417	2
3	112	3
4	426	5

The **time complexity** of Best-Fit algorithm is **O(n2)** as it requires two loops to process the memory blocks and processes. The outer loop is used to iterate through the processes and the inner loop is used to iterate through the blocks.
The **space complexity** of Best-Fit algorithm is **O(n)** as it requires an array of size n to store the block allocation for each process.

Program for Worst Fit algorithm in Memory Management

Worst Fit allocates a process to the partition which is largest sufficient among the freely available partitions available in the main memory. If a large process comes at a later stage, then memory will not have space to accommodate it.

Example:

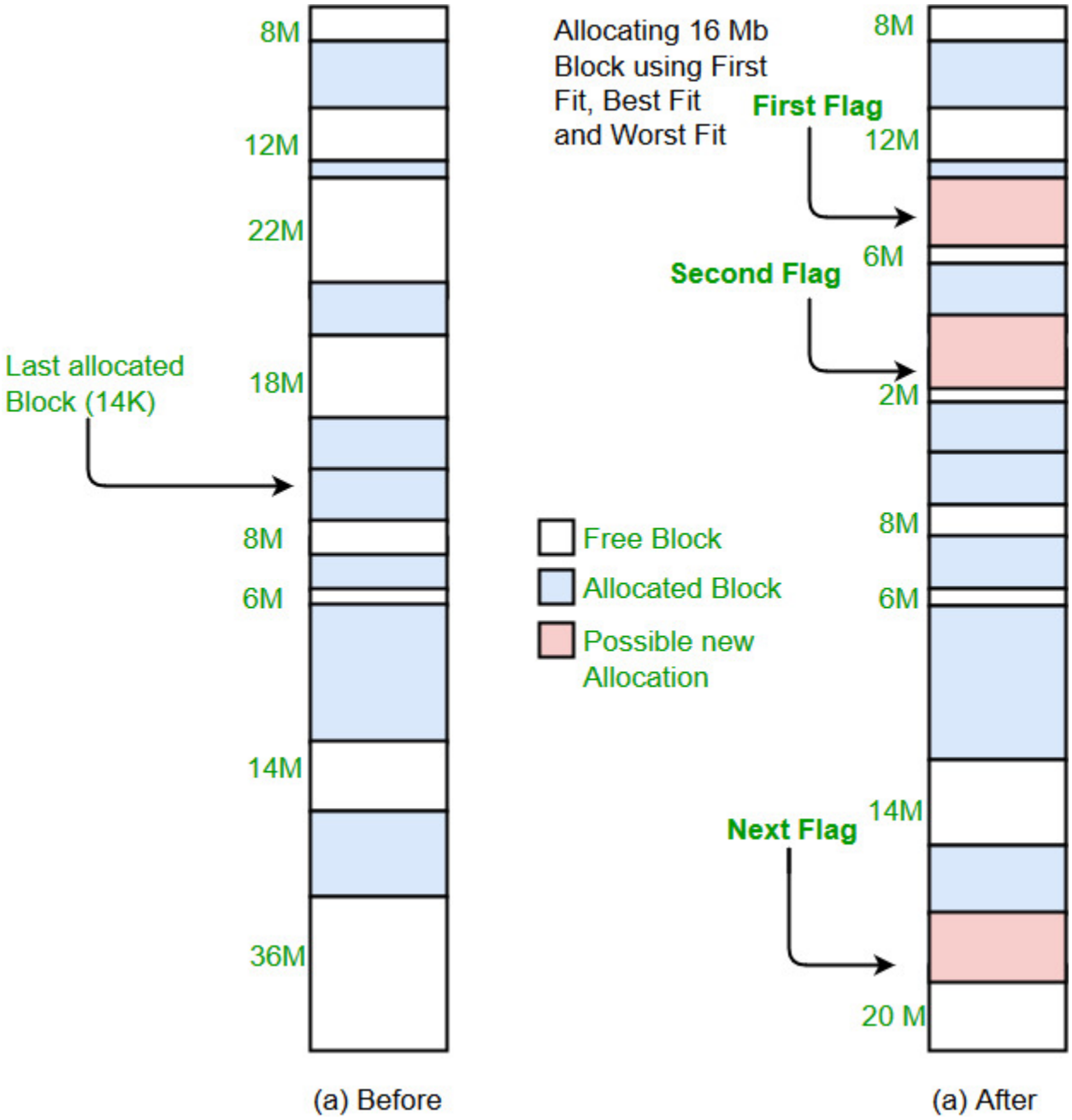
Input : blockSize[] = {100, 500, 200, 300, 600};

processSize[] = {212, 417, 112, 426};

Output:

Process No. Process Size Block no.

1	212	5
2	417	2
3	112	5
4	426	Not Allocated



Implementation:

1- Input memory blocks and processes with sizes.

2- Initialize all memory blocks as free.

3- Start by picking each process and find the
maximum block size that can be assigned to
current process i.e., find max(bockSize[1],
blockSize[2],.....blockSize[n]) >
processSize[current], if found then assign
it to the current process.

5- If not then leave that process and keep checking
the further processes.

// C++ implementation of worst - Fit algorithm

```
#include<bits/stdc++.h>
using namespace std;
  // Function to allocate memory to blocks as per worst fit
// algorithm
void worstFit(int blockSize[], int m, int processSize[],  int n)
{
    // Stores block id of the block allocated to a
    // process
    int allocation[n];
      // Initially no block is assigned to any process
    memset(allocation, -1, sizeof(allocation));
      // pick each process and find suitable blocks
    // according to its size ad assign to it
    for (int i=0; i<n; i++)
    {
        // Find the best fit block for current process
```

```
    int wstIdx = -1;
    for (int j=0; j<m; j++)
    {
       if (blockSize[j] >= processSize[i])
       {
          if (wstIdx == -1)
             wstIdx = j;
          else if (blockSize[wstIdx] < blockSize[j])
             wstIdx = j;
       }
    }
     // If we could find a block for current process
    if (wstIdx != -1)
    {
       // allocate block j to p[i] process
       allocation[i] = wstIdx;

       // Reduce available memory in this block.
       blockSize[wstIdx] -= processSize[i];
    }
 }
  cout << "\nProcess No.\tProcess Size\tBlock no.\n";
 for (int i = 0; i < n; i++)
 {
    cout << "   " << i+1 << "\t\t" << processSize[i] << "\t\t";
    if (allocation[i] != -1)
       cout << allocation[i] + 1;
    else
```

```
            cout << "Not Allocated";
        cout << endl;
    }
}


// Driver code
int main()
{
    int blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    int m = sizeof(blockSize)/sizeof(blockSize[0]);
    int n = sizeof(processSize)/sizeof(processSize[0]);
      worstFit(blockSize, m, processSize, n);
     return 0 ;
}
```

Output

Process No.	Process Size	Block no.
1	212	5
2	417	2
3	112	5
4	426	Not Allocated

Time Complexity: O(N*M) where N is processSize length and M is blockSize length.
Auxiliary Space: O(N)