

مقدمة إلى نظم التشغيل

Introduction to Operating System



د. رافة معين خازم

Damascus University

مقدمة



نظام التشغيل هو برنامج

■ يدير عتاديات الحاسب

■ يؤمن الأساس لعمل برامج التطبيقات

■ يتصرف كوسيط بين مستخدمي الحاسب وعتادياته

دور نظام التشغيل

Role of Operating System



من وجهة نظر المستخدم:

■ يصمم نظام تشغيل الحواسيب الشخصية PC لتسهيل استخدام الحاسب مع بعض الاهتمام بالأداء بدون أي انتباه بمدى الاستفادة من موارد الحاسب Resource Utilization (كيفية تشارك الموارد البرمجية و العتادية)

■ يصمم نظام تشغيل المنصات Mainframe للوصول إلى الحد الأعظمي من مدى الاستفادة من عتاديات الحاسب

دور نظام التشغيل



- يصمم نظام تشغيل محطات العمل Workstation للموازنة بين الاستخدام الشخصي للجهاز و مدى الاستفادة من الموارد
- يصمم نظام تشغيل الحواسيب المحمولة Handheld computers لدعم الاستخدام الشخصي و أداء الجهاز بالنسبة لعمر البطارية
- يصمم نظام تشغيل الحواسيب المضمنة Embedded computers للعمل بدون أي تدخل من المستخدم

دور نظام التشغيل



من وجهة نظر الحاسب:

■ نظام التشغيل هو محصص للموارد Resource Allocator ⇔ حل أي مشكلة يتطلب استخدام كثير من موارد الحاسب (وحدة المعالجة، الذاكرة، الملفات، أجهزة الدخل و الخرج...) الأمر الذي يجعل نظام التشغيل يتصرف كمدير لهذه الموارد فيقرر كيفية تحصيل هذه الموارد بين البرامج و المستخدمين بحيث يشغل الحاسب بشكل فعال و عادل.

دور نظام التشغيل



يعد تخصيص الموارد هاما بشكل خاص عندما يكون هناك عدد من المستخدمين يقوموا بالنفاز إلى نفس الموارد (mainframe, workstation)

■ نظام التشغيل هو برنامج تحكم Control Program ⇔ يقوم نظام التشغيل بإدارة تنفيذ برامج المستخدم لمنع الأخطاء و الإستخدام غير الصحيح للحاسب و بشكل خاص يعنى بتشغيل أجهزة الدخل و الخرج و التحكم بها

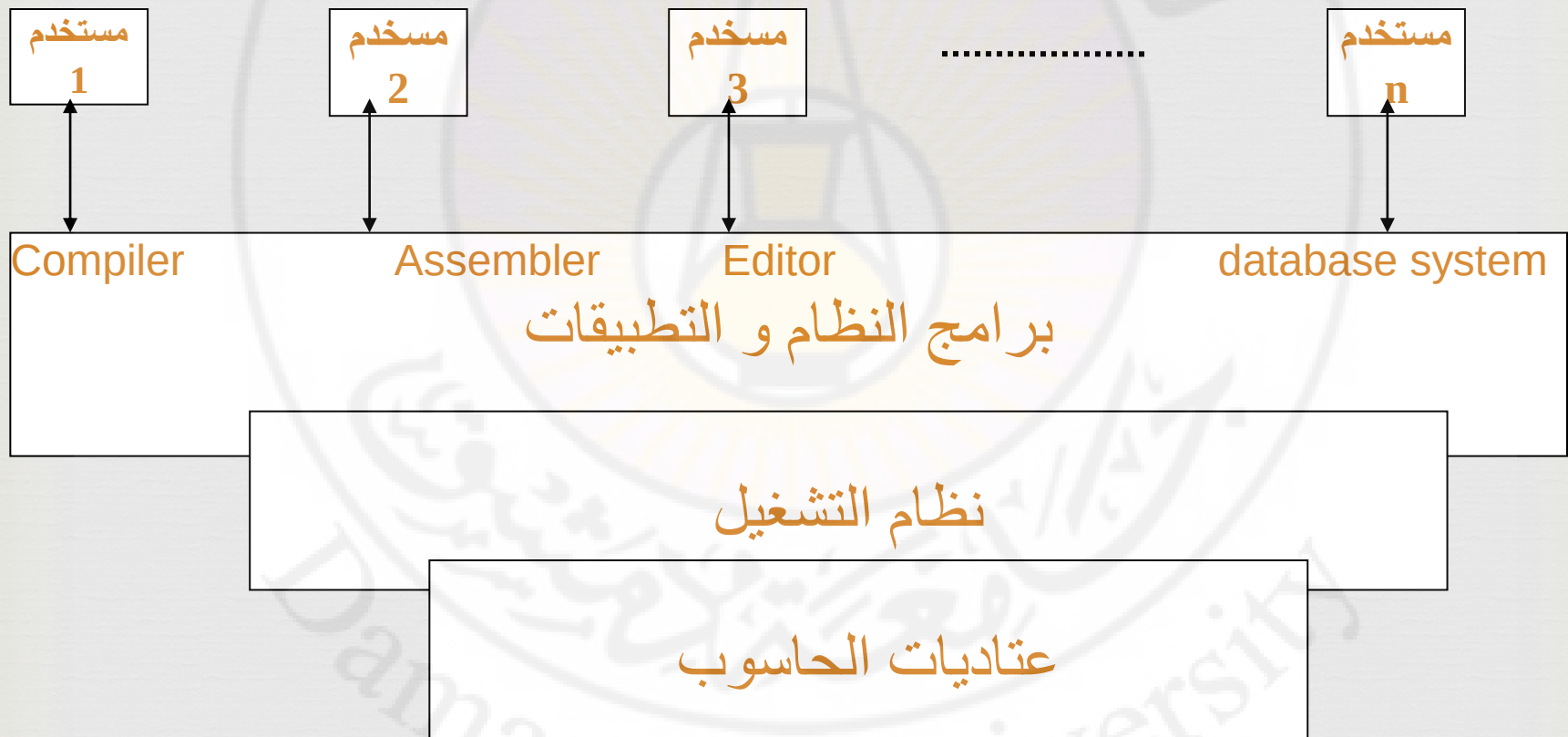
خلاصة



بعض أنظمة التشغيل تصمم لتلائم المستخدم Convenient و بعضها تصمم لتزويد من فعالية الحاسب Efficient و بعضها يحقق الأمرين معا

مكونات نظام الحاسوب

Components of Computer System



مكونات نظام الحاسوب



- تمثل عتاديات الحاسوب الموارد الرئيسية التي يتم من خلالها استثمار الحاسوب
- تحدد برامج النظام و التطبيقات الطرق التي تستخدم فيها الموارد لحل مشاكل المستخدم
- يقوم نظام التشغيل بالتحكم و التنسيق بين العتاديات و برامج التطبيقات المختلفة الخاصة بعدد من المستخدمين

نظام التشغيل كنواة

Kernel



يطلق على نظام التشغيل اسم نواة kernel و يعرف بأنه البرنامج الذي يكون بحالة تنفيذ مستمرة و الذي يشرف على تنفيذ البرامج الأخرى

مبادئ رئيسية

Basics



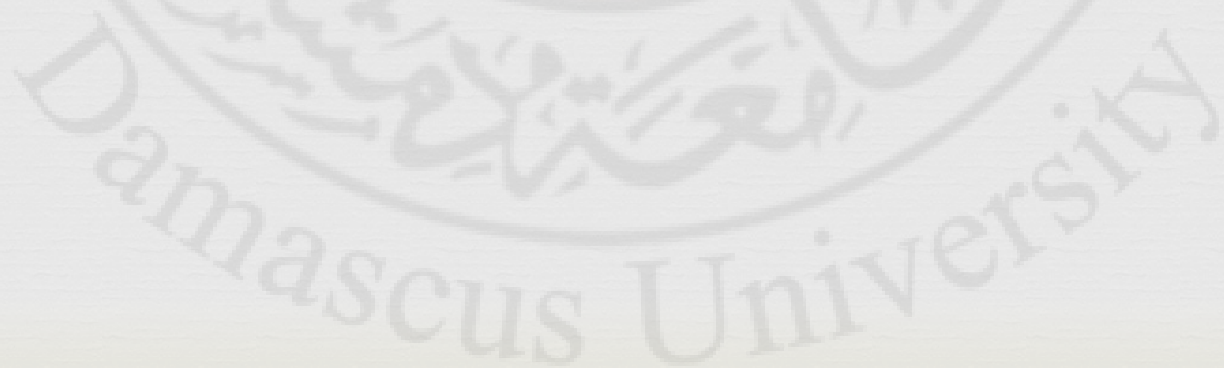
- لا يمكن لبرنامج أن يبدأ تنفيذه ما لم يقوم نظام التشغيل بتحميله في الذاكرة.
- يبقى البرنامج في الذاكرة حتى ينهي تنفيذه تماما (هناك حالات خاصة لهذه القاعدة)
- توصف وحدة المعالجة بحالتين: مشغولة busy (هناك برنامج ينفذ عليها) أو غير مشغولة idle

تطور نظام التشغيل

Operating System Development



- نظام البرنامج الوحيد Single-Program/ Uniprogrammed System
- نظام البرامج المتعددة Multiprogrammed System
- نظام المشاركة بزمان المعالج Time-Sharing System



النظام وحيد البرنامج

Uniprogrammed System



- يحمل نظام التشغيل برنامج واحد في الذاكرة و لا يحمل برنامج آخر حتى انتهاء الأول من التنفيذ
- عند ترك البرنامج قيد التنفيذ المعالج للقيام بعمليات الدخل و الخرج يبقي المعالج في حالة لا عمل $\Rightarrow$ لا يمكن ابقاء وحدة المعالجة وأجهزة الدخل/الخرج في حالة انشغال طوال الوقت.

النظام متعدد البرامج

Multiprogrammed System



■ يحمل نظام التشغيل عدد من البرامج في ذاكرة الحاسب بنفس اللحظة $\leq$

١. زيادة معدل الإستفادة من موارد الحاسب :المعالج، الذاكرة، اجهزة الدخل/الخرج

٢. زيادة التدفق

النظام متعدد البرامج



معدل الإستفادة من وحدة المعالجة CPU Utilization هو النسبة المئوية للزمن الذي تكون فيه وحدة المعالجة مشغولة بالنسبة لزمن تشغيل الجهاز

المردود Throughput هو عدد البرامج التي أنهت تنفيذها خلال واحدة الزمن

نظام المشاركة بزمان المعالج

Time-Sharing System



- امتداد منطقي لتعدد البرامج
- تقوم وحدة المعالجة بتنفيذ عدد من البرامج عن طريق التبدل فيما بينها
- يتم التبدل بحيث يسمح لكل مستخدم بالتفاعل مع برنامجه خلال تنفيذه

نظام المشاركة بزمان المعالج



- يقسم زمن المعالج إلى شرائح زمنية تتناوب عليها البرامج
==< ظهور الأنظمة التفاعلية Interactive System التي تؤمن
الاتصال بين المستخدم و الحاسب ==< ظهور مفهوم زمن
الاستجابة Response Time (الزمن الفاصل بين لحظة إعطاء
الأمر و لحظة اعطاء أول استجابة لهذا الأمر).
- يسمح لعدد من المستخدمين بالتشارك بالحاسب بأن واحد.

نظام المشاركة بزمان المعالج



■ إضافة وظائف رئيسية لنظام التشغيل

١. جدولة الأعمال Job Scheduling

٢. جدولة وحدة المعالجة المركزية CPU Scheduling

٣. نظام الملفات File System

٤. ادارة الذاكرة Memory Management

٥. حماية الموارد Resource Protection

٦. تزامن و اتصال المهام Process Synch & Communication

الحواسيب الشخصية

Personal Computers



- تكلفة منخفضة
- أجهزة الدخل: لوحة المفاتيح و الفأرة
- أجهزة الخرج: شاشة العرض و الطابعة الصغيرة
- في البداية لم يكن محميا
- يصمم نظام التشغيل ليكون سهل الاستخدام
- شركات التصنيع :

Microsoft → Windows

Apple → Apple Macintosh

معالج الاغراض العامة و معالج الاغراض الخاصة

General-purpose Processor Vs. Special-purpose Processor



- معالجات الأغراض العامة General-purpose Processor CPU : تنفذ تعليمات ذات أغراض عامة بما فيها تعليمات برامج المستخدمين.
- معالجات الأغراض الخاصة Special-purpose Processor : تنفذ مجموعة من التعليمات الخاصة لا تشمل تعليمات المستخدمين و تدار من قبل نظام التشغيل و يكون لها أكثر من شكل :

معالج الاغراض العامة و معالج الاغراض الخاصة



١. معالجات خاصة بالتجهيزات Device-specific Processor
: متحكمات Controller القرص و لوحة المفاتيح و الشاشة

٢. معالجات أغراض عامة General-purpose Processor :
معالجات الدخل/الخرج

النظام وحيد المعالج و النظام متعدد المعالجات

Uniprocessor Vs. Multiprocessor System



يمكن تصنيف نظام الحاسب بناء على عدد معالجات الأغراض العامة General-purpose Processor CPU إلى نوعين:

١. نظم وحيدة المعالج Single/Uni-processor Systems و تحتوي على CPU واحدة
٢. نظم متعددة المعالجات Multiprocessor Systems و تحتوي على أكثر من CPU

النظم متعددة المعالجات

Multiprocessor Systems



- النظم التفرعية Parallel Systems
- النظم الموزعة Distributed Systems

النظام التفرعي

Parallel System

نظام يحتوي معالجين أو أكثر مرتبطة ببعضها بشكل وثيق Tightly coupled systems $\Leftrightarrow$ تتشارك المعالجات بالذاكرة و الطرفيات و المسرى و في بعض الأحيان بالساعة.

فوائد النظام التفرعي

- زيادة المردود
- توفير في تكلفة العتاديات
- زيادة الوثوقية

النظام التفرعي



أنواع النظام التفرعي

■ نظام متعدد المعالجة غير متناظر Asymmetric Multiprocessing

■ نظام متعدد المعالجة متناظر Symmetric Multiprocessing

يمكن الحصول على المعالجة المتناظرة و غير المتناظرة إما من خلال العتاديات أو من خلال البرمجيات، و يعد SunOS Ver 4 مثالا عن المعالجة غير المتناظرة و SunOS Ver5 مثالا عن المعالجة المتناظرة و لنفس العتاديات.

النظام الموزع

Distributed System



مجموعة من الحواسيب المنفصلة و غير المتجانسة التي يربط بينها شبكة مما يمكن المستخدمين من النفاذ إلى الموارد المتنوعة فيه.

فوائد النظام الموزع

- تسريع المعالجة
- وفرة في المعطيات
- زيادة في القدرة الوظيفية
- زيادة الوثوقية

نظام الزمن الحقيقي

Real-Time System

- نظام التشغيل الذي يستخدم في تشغيل الأنظمة الحاسوبية المضمنة و هو يستخدم عندما تفرض قيودا زمنية صارمة على عمل المعالج و تدفق المعطيات == < غالبا ما يستخدم كتجهيزة تحكم في تطبيقات مخصصة.
- يكون لهذا النظام قيودا زمنية ثابتة و محددة جيدا ⇔ المعالجة يجب أن تتم ضمن هذه الحدود و إلا يفشل النظام.

نظام الزمن الحقيقي

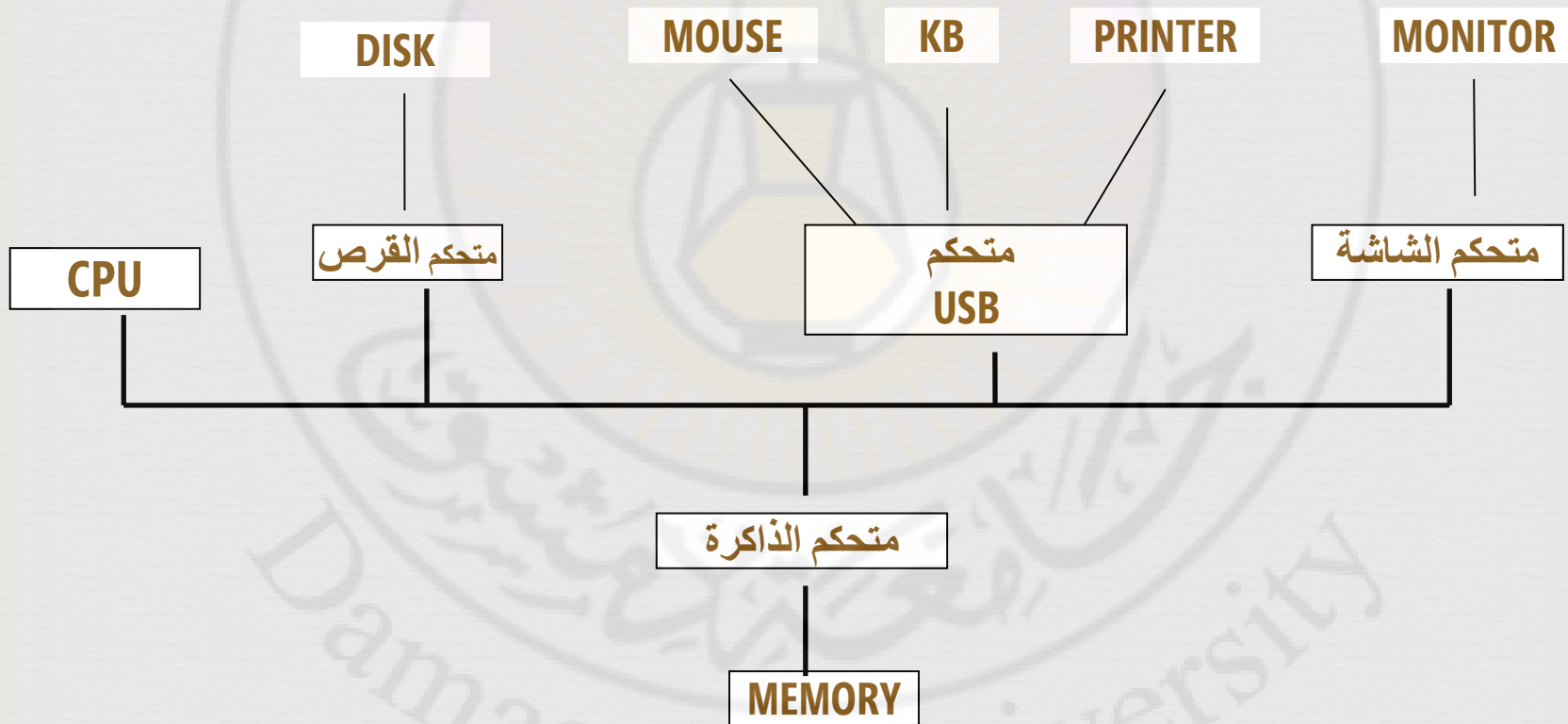


أنواع نظام الزمن الحقيقي

- نظام الزمن الحقيقي المرن (Soft Real-Time System)
- نظام الزمن الحقيقي الصارم (Hard Real-Time System)

بنية النظام الحاسوبي الحديث

Structure of Modern Computer System



عمل نظام الحاسوب

Operation of Computer System



تشغيل الجهاز
Power-On



تشغيل برنامج
Bootstrap



تحميل و تشغيل
نظام التشغيل



حدوث مقاطعة
برمجية أو عتادية

آلية المقاطعة

Interrupt Mechanism

حدوث مقاطعة

إجرائية عامة لتفحص المقاطعة و تحديد مكان برنامج خدمتها

استدعاء برنامج خدمة المقاطعة ISR

تنفيذ ISR ثم العودة الى التعليمات التي تمت مقاطعتها

بنية التخزين

Storage Structure

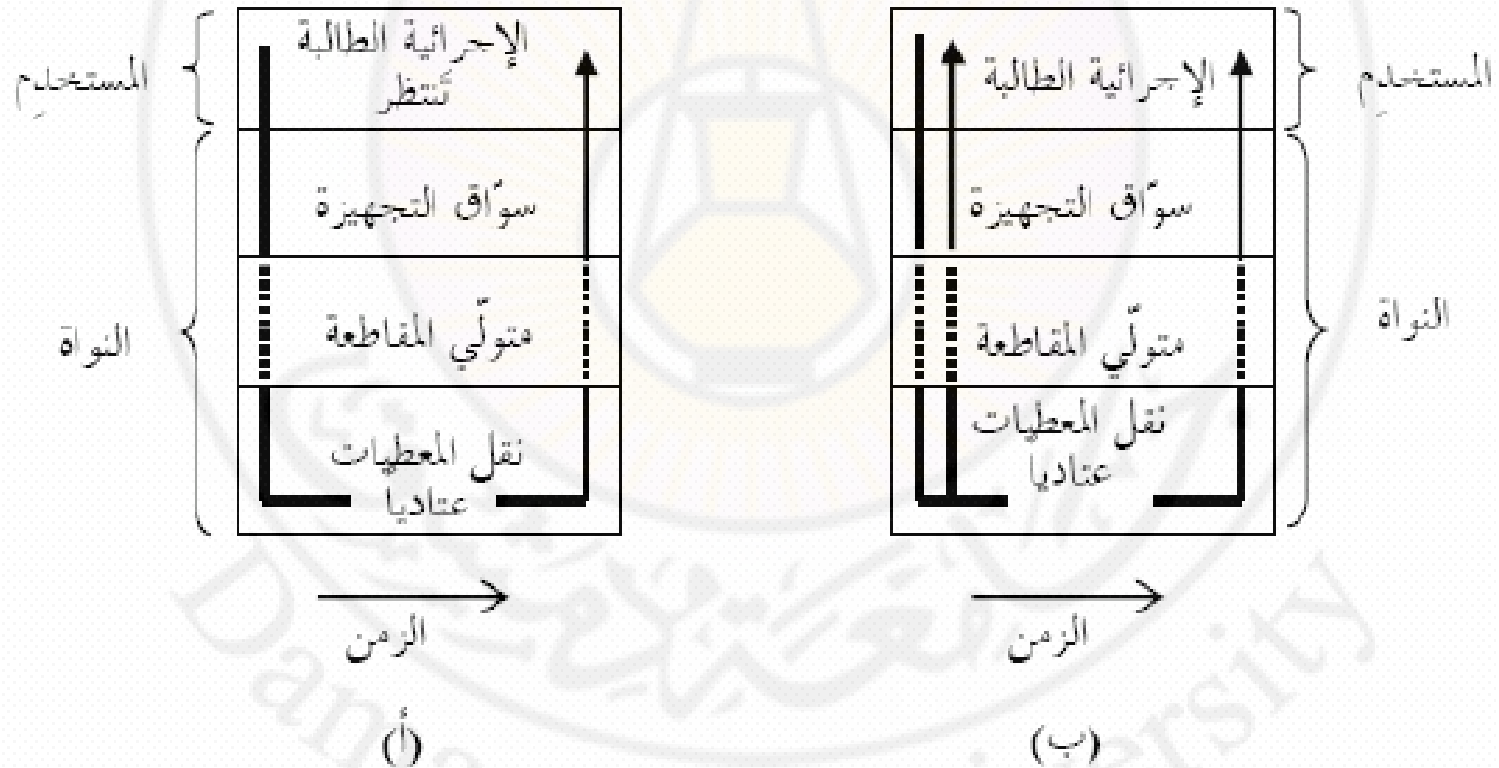


بنية الدخل/الخرج

I/O Structure

- متزامن Synchronous: تنتظر الإجراءات انتهاء عملية الدخل / الخرج لتتابع تنفيذها
- غير متزامن Asynchronous: بعد أن تطلب الإجراءات عملية الدخل/الخرج تتابع تنفيذها دون انتظار انتهاء عملية الدخل/الخرج

بنية الدخل/الخرج



التخبيئة

Caching

- تعد التخبيئة Caching من أهم مبادئ عمل نظام الحاسوب حيث يتم حفظ المعطيات في نظام تخزين ليتم نقلها إلى نظام تخزين أسرع (الذاكرة الخبيئة) عند استخدامها.
- حركة المعلومات بين مستويات هرمية التخزين إما أن تكون صريحة بإشراف نظام التشغيل (من القرص إلى الذاكرة) أو ضمنية بإشراف العتاديات (من الذاكرة الخبيئة إلى المعالج)

اتساق الذاكرة الخبيئة

Cache Memory Coherency

تؤدي التخبئة إلى ظهور نفس المعطيات في مستويات مختلفة من هرم التخزين. مثلاً عند تنفيذ عملية زيادة عدد صحيح A في ملف B يتم ما يلي

- ١- نسخ كتلة القرص التي تحوي A إلى الذاكرة الرئيسية
- ٢- نسخ A إلى الذاكرة الخبيئة
- ٣- نسخ A إلى سجل CPU
- ٤- زيادة A

للمحافظة على تجانس قيمة A بين مستويات هرمية التخزين يتم إعادة نسخ قيمتها بعد التعديل من CPU إلى القرص

حماية عتاديات نظام الحاسوب

Hardware Protection



- نظام التشغيل هو نظام مساق بالمقاطعة (Interrupt Driven)
- التشغيل ثنائي النمط (Dual-Mode):
 ١. نمط المستخدم (User-Mode)
 ٢. نمط النظام (System-Mode)
- بت النمط (mode bit) : 0 نمط النظام ، 1 نمط المستخدم
- التعليمات ذات الامتياز (Privileged Instruction)

حماية الدخل/الخرج

I/O Protection



تحدد جميع تعليمات الدخل و الخرج لتكون تعليمات ذات امتياز <=> لا يستطيع المستخدم تنفيذها و إنما يطلب ذلك من نظام التشغيل عبر استخدام استدعاءات النظام الخاصة بذلك.

حماية الذاكرة

Memory Protection



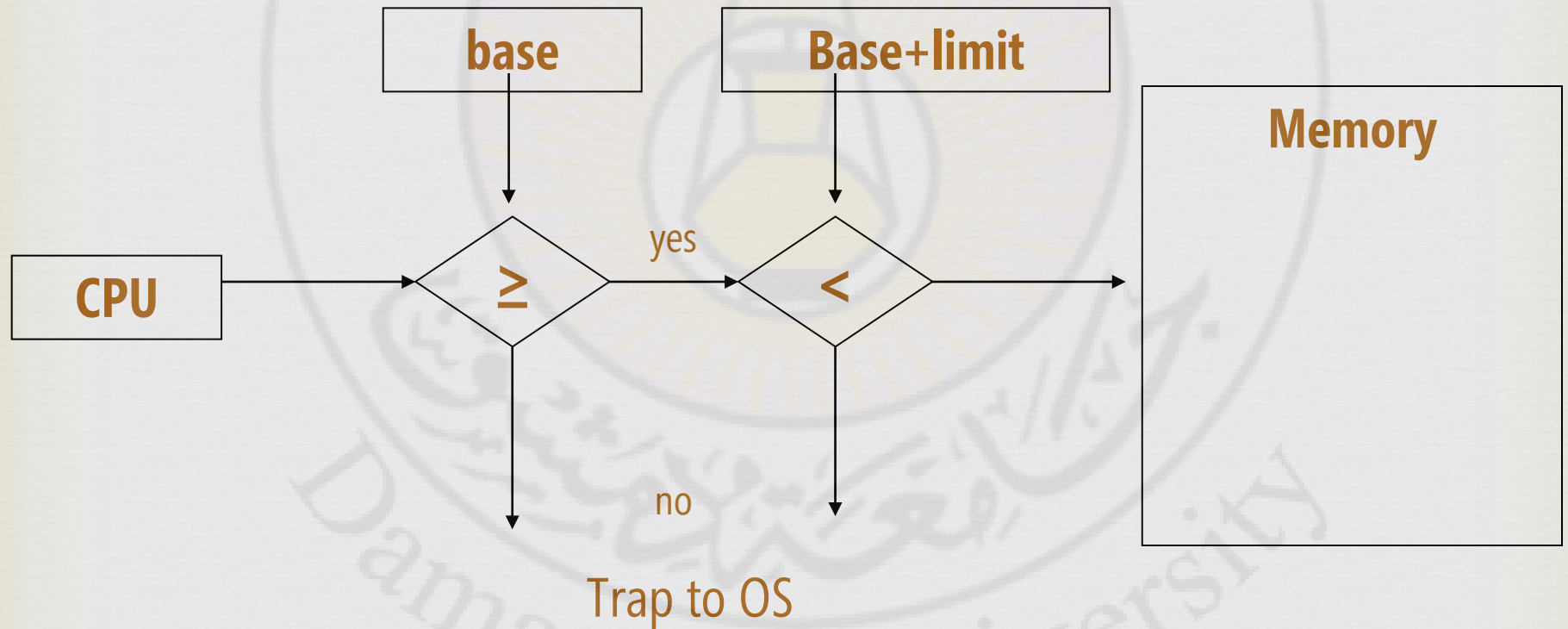
منع المهمة قيد التنفيذ من النفاذ إلى ذاكرة مهمة أخرى أو ذاكرة نظام التشغيل <== تحديد مجال العناوين التي يمكن للمهمة النفاذ إليها خلال تنفيذها <== إضافة السجلين إلى عتاديات الحاسب

١. سجل القاعدة Base Register: يحمل بعنوان أول موقع تحتله المهمة في الذاكرة

٢. سجل الحد Limit Register: يحمل بعدد مواقع الذاكرة التي تحتلها المهمة.

يقوم نظام التشغيل بتحميل سجلي القاعدة و الحد بقيمها ⇔ تعليمات ذات امتياز

حماية الذاكرة



حماية وحدة المعالجة المركزية

CPU-Protection



منع المهمة خلال تنفيذها من الالتصاق على وحدة المعالجة المركزية وعدم تمكين مهام أخرى من الحصول عليها <=> عدم إعادة التحكم لنظام التشغيل لبدء تنفيذ مهام.

■ لتحقيق عملية الحماية للمعالج يتم ربطه بمؤقت زمني Timer ويضبط من قبل نظام التشغيل بحيث يقوم بمقاطعة المهمة التي تنفذ على المعالج بعد فترة زمنية.

■ قد تكون الفترة الزمنية ثابتة أو متغيرة

■ يتكون المؤقت المتغير من ساعة و عداد

يقوم نظام التشغيل بتحميل المؤقت بقيمته ⇔ تعليمة ذات امتياز

خدمات نظام التشغيل Operating System Services



د. رافة معين خازم

Damascus University

خدمات نظام التشغيل



يؤمن نظام التشغيل البيئة المناسبة لتنفيذ البرامج و ذلك بتقديم الخدمات التي تساعد برامج المستخدمين و الحاسب <== يمكن تصنيف هذه الخدمات :

- ١ - خدمات تساعد المستخدم
- ٢ - خدمات تضمن تشغيل الحاسب بشكل فعال

خدمات المستخدم

User-Oriented Services



١. تأمين واجهة اتصال للمستخدم مع نظام التشغيل

- command-Line Interface

- Batch Interface

- Graphical-User Interface

٢. تنفيذ البرامج $\Leftrightarrow$ نظام التشغيل يحمل البرنامج و ينفذه

خدمات المستخدم



٣. عمليات الدخل و الخرج $\Leftrightarrow$ لا يستطيع المستخدم التحكم بأجهزة الدخل و الخرج مباشرة لذلك عندما يحتاج إلى عملية دخل/خرج فإنه يطلب ذلك من النظام

٤. معالجة نظام الملفات $\Leftrightarrow$ قد تحتاج البرامج التعامل مع الملفات (قراءة، كتابة، إنشاء، حذف، إغلاق) و تكون مهمة نظام التشغيل التعامل مع نظام الملفات

خدمات المستخدم



٥. الإتصال $\Leftrightarrow$ يؤمن نظام التشغيل طرقا لاتصال البرامج مع بعضها البعض و لتبادل المعلومات. هناك طريقتين للاتصال: الذاكرة المشتركة Shared Memory و تمرير الرسائل Message Passing . قد تكون الإجراءات المتصلة مع بعضها البعض في نفس الجهاز أو في أجهزة مختلفة.

خدمات المستخدم



٦. اكتشاف الأخطاء ⇔ يجب أن يكون نظام التشغيل مدركا للأخطاء التي يمكن أن تنجم عن العتاديات (معالج، ذاكرة، أجهزة دخل/خرج و برنامج المستخدم) و من أجل كل نوع من الأخطاء يتخذ نظام التشغيل الفعل المناسب

خدمات الحاسب

System-Oriented Services

١. تخصيص الموارد $\Leftrightarrow$ عندما يكون هناك أكثر من مستخدم يستخدم الجهاز، تكون مهمة نظام التشغيل تخصيص الموارد بشكل عادل بين المستخدمين
٢. المحاسبة: الإحصاء $\Leftrightarrow$ أن يكون النظام قادرا على تعقب المستخدمين الذين يستخدمون موارد النظام و على تحديد كمية الموارد و أنواعها.

خدمات النظام



٣. الحماية والسرية ⇔ أن يكون النظام قادرا على التحكم بموارد النظام <==> منع تداخل الإجراءات مع بعضها و مع النظام. و يجب أن يؤمن النظام السبل لحماية المعلومات من النفاذ غير المرغوب فيه .

استدعاءات النظام

System Calls

و هي الأدوات التي تؤمن الاتصال بين المهمة ونظام التشغيل،
فمن خلالها تقوم المهمة بطلب خدمة من نظام التشغيل.

أنواع استدعاءات النظام

- ١ - التحكم بالمهمة: end, abort, load, execute
- ٢ - إدارة الملفات: create file, open, close, read, write

استدعاءات النظام



- ٣- إدارة التجهيزات: release& request device, read, write
- ٤- صيانة المعلومات: get time, get date, set time, set date
- ٥- الإتصالات: create and delete connection, send and receive message

استدعاءات النظام



إن تنفيذ استدعاءات النظام يتطلب اسم الاستدعاء و معلومات تختلف من استدعاءة لأخرى و من نظام تشغيل لآخر. يتم تمرير المعلومات إلى الإستدعاءات

- عبر السجلات
- عبر كتلة من الذاكرة
- عبر المكس

برامج النظام

System Programs

تؤمن هذه البرامج البيئة الملائمة لتطوير و تنفيذ البرامج ويمثل بعضها واجهة اتصال المستخدم مع استدعاءات النظام

١ - إدارة الملفات: print create, delete, copy, rename,

٢ - معلومات الحالة: date, time, amount of memory, number of users

٣ - تعديل الملفات: editors

برامج النظام



٤- دعم لغات البرمجة: compilers, assemblers, debuggers

٥- تحميل و تنفيذ البرامج: loaders, linkers

٦- الإتصالات: إنشاء وصلات افتراضية بين المستخدمين و المهمات و الأجهزة. تسمح للمستخدمين بإرسال رسائل عبر الشاشات، البحث عبر صفحات الويب، إرسال رسائل إلكترونية، نقل الملفات بين الأجهزة

المحاضرة الأولى

نُظْمُ التَّشْغِيلِ

القِسْمُ الْعَمَلِيُّ

LinuxTM



كتابة وتنسيق:
هاشيم حسن ونوس

مُفسّراتُ الأوامر:

أنواعُها:

1.SH.

2.KSH.

3.Bash، (وهو الافتراضي).

ولِلاستفادَةِ مِنْها، نَسْتَعْمَلُ ما يُسمّى بالـ **Terminal**، وهو شبيهٌ بالـ **CMD** الخاصِّ بِنظامِ تَشغيلِ **Windows**.

في كُلِّ مرّةٍ نودُ كتابةَ أمرٍ في الـ **Terminal**، سَنَجِدُ هذا الكلامَ مكتوباً دوماً:

```
[user@localhost ~]$
```

حيثُ أَنَّهُ:

- 1.user: اسمُ المُستخدمِ الَّذي تمَّ تسجيلُ الدخولِ مِنْهُ.
- 2.~: دليلٌ على أَنَّ اسمَ المجلدِ المفتوحِ حالياً هو الـ **home**، ويُمكنُ أَنْ تُستبدَلَ بِاسمِ آخَرَ لتدلَّ على مجلّدٍ آخَرَ مفتوحٍ غيرُهُ.
- 3.\$: تدلُّ على أَنَّ المُستخدمَ الحاليَّ مُستخدمٌ عاديٌّ وليسَ **root** (مسؤولاً عَنِ النظامِ)، وإنَّ كانَ كذلكَ، فَتُستبدَلُ بِـ **#**.

الأوامر الرئيسية:

1.date: تعرض تاريخ وساعة النظام الحاليين.

```
[user@localhost ~]# date
```

2.cal: تعرض التقويم للشهر الحالي في تاريخ النظام، (و **cal** هي اختصار لـ **calendar**).

```
[user@localhost ~]# cal
```

ويمكن تحديد الشهر المطلوب عرضه، (مثلاً **3** لعرض الشهر الثالث، ويكون من السنة الحالية لتاريخ النظام).

```
[user@localhost ~]# cal 3
```

كما يمكن تحديد السنة المطلوبة عرضها أيضاً، (مثلاً سنة **2003**).

```
[user@localhost ~]# cal 3 2003
```

3.uptime: تعرض المدة التي مضت منذ بداية إقلاع النظام، (و **uptime** هي اختصار لـ **up to time**).

```
[user@localhost ~]# uptime
```

4.who: تعرضُ المُستخدمينَ المُسجّلينَ على النظامِ.

```
[user@localhost ~]# who
```

5.whoami: تعرضُ اسمَ المُستخدمِ الحاليّ، (و **whoami** هي اختصارٌ لـ **who am I**).

```
[user@localhost ~]# whoami
```

6.man: تقومُ بِشرحِ تعلِيمَةٍ ما، حيثُ يُكتَبُ بَعْدَها اسمُ التعلِيمَةِ المطلوبة، (مثلاً هُنَا طَلَبْنَا مِنْهَا أَنْ تشرحَ التعلِيمَةَ **cal**، و **man** هي اختصارٌ لـ **manual**).

```
[user@localhost ~]# man cal
```

7.info: تعرضُ معلوماتٍ كاملةٍ عَنَ تعلِيمَةٍ ما، حيثُ يُكتَبُ بَعْدَها اسمُ التعلِيمَةِ المطلوبة، (مثلاً هُنَا طَلَبْنَا مِنْهَا أَنْ تشرحَ التعلِيمَةَ **date**، و **info** هي اختصارٌ لـ **information**).

```
[user@localhost ~]# info date
```


clear.8: تقوم بالانتقال لصفحة جديدة من ال **Terminal**، (ولا يتم مسح التعليمات القديمة، إذ أنها لا تزال موجودة لو انتقلنا لأعلى الصفحة).

```
[user@localhost ~]# clear
```

echo.9: تُشبه تعليمة **print** الخاصة بلغات البرمجة، حيث تقوم بطباعة عبارة على ال **Terminal**.

```
[user@localhost ~]# echo Hello, World!
```

uname.10: تعرض معلومات عن حالة النظام.

```
[user@localhost ~]# uname
```

ويمكن عرض معلومات مفصلة بشكل أكبر عن طريق كتابة **-a** بعد الأمر.

```
[user@localhost ~]# uname -a
```

free.11: تعرض معلومات عن حالة الذاكرة.

```
[user@localhost ~]# free
```

df.12: تعرضُ معلوماتٍ عَنْ حالةِ الأقراصِ، (و **df** هي اختصارٌ لـ **disk free**).

```
[user@localhost ~]# df
```

sleep.13: تقومُ بإيقافِ ال **Terminal** لعددٍ مُحدّدٍ مِنَ الثواني ،
نقومُ بتمريرها بعدَ الأمرِ، (مثلاً نكتبُ 5 ثوانٍ).

```
[user@localhost ~]# sleep 5
```

ولإلغاءِ عمليةِ الإيقافِ، نضغطُ **CTRL + Z**، (والتي يمكنُ أَنْ تُستعملَ
مَعَ أوامرٍ أُخرى أيضاً).

أوامرُ الأدلة:

pwd.1: تعرضُ المسارَ الحاليّ، (و **pwd** هي اختصارٌ لـ **print**
working directory).

```
[user@localhost ~]# pwd
```

cd.2: تقومُ بتغييرِ المسارِ الحاليّ لِمسارٍ آخرٍ، وعِنْدَ كتابتها بدونِ
مسارٍ، تقومُ بتغييره إلى ال **home**، (و **cd** هي اختصارٌ لـ
create directory).

```
[user@localhost ~]# cd
```

ويمكن أيضاً كتابة المدة لتحقيق نفس النتيجة.

```
[user@localhost ~]# cd ~
```

نكتب مساراً معيناً نريد الدخول إليه، (مثلاً للدخول إلى مجلد **hello** الذي في سطح المكتب نكتب هكذا، مع العلم أن مجلد سطح المكتب **Desktop** يقع في مجلد باسم صاحب النظام، والذي بدوره يقع في مجلد باسم **home**).

(أيضاً يجب الانتباه لحساسية الأحرف عند كتابة أسماء المجلدات والملفات).

```
[user@localhost ~]# cd  
home/user/Desktop/hello
```

نلاحظ تغير اسم المجلد الحالي إلى **hello**.

```
[user@localhost hello]#
```

لِلرّجوع مرّةً لِلخلفِ إلى المجلد السابق نكتب الآتي.


```
[user@localhost hello]# cd ..
```

وطالما أننا في مجلد ال **Desktop**، يمكننا مباشرة كتابة اسم مجلد **hello** بدون مساره، (وذلك لأنه يقع في مجلد **Desktop**).

```
[user@localhost Desktop]# cd hello
```

ويمكننا أيضاً الرجوع مرتين للخلف.

```
[user@localhost hello]# cd ../../
```

أنواع المسارات:

3. المطلق: نكتب المسار كاملاً ابتداءً من عند الدليل الجذر **home**.

```
[user@localhost ~]# cd  
home/user/Desktop/hello
```

4. النسبي: نكتب المسار ابتداءً من الدليل الحالي.

```
[user@localhost Desktop]# cd hello
```

تكملة أوامر الأدلة:

ls.3: تستعرض محتويات الدليل الحالي، حيث تقوم بعرض كل الملفات والمجلدات والروابط التي يحتويها.

```
[user@localhost Desktop]# ls
```

ويكون الخرج أسماء المحتويات كآتي، (ويكون الملف مكتوباً بلون أسود، والمجلد بأزرق، والرابط بأخضر).

```
hello text shortcut
```

كما يمكن أن نُحدّد مسار مجلد مُعيّن بغض النظر عن الدليل الحالي.

```
[user@localhost hello]# ls  
home/user/Desktop
```

ويمكن أيضاً استعراض تلك المحتويات بشكلٍ مُفصّلٍ أكثر عن طريق كتابة **l-** بعد الأمر.

```
[user@localhost hello]# ls -l  
home/user/Desktop
```

عند تنفيذ الأمر السابق، نلاحظ ظهور الآتي.

```
total 3
drwxr-x--- user Mar 15 15:33 hello
-rwxrwxr-- user Mar 15 18:22 text
lr-xr-xr-- user Mar 15 11:45 shortcut
```

أول جملة **total 3** تشير إلى أنه هنالك ثلاث محتويات داخل المجلد.

أول حرف يشير إلى نوع الملف، إذا كان **d** فهو يعني مجلداً، وإذا كان **l** فهو يعني رابطاً، وإن كان **-** فهو يعني ملفاً عادياً.

ما بعد أول حرف هو **9** حروف نقوم بتقسيمها إلى **3** أقسام، القسم الأول يشير إلى صلاحيات مالك النظام **root**، والثاني يشير إلى صلاحيات المجموعة **group**، والأخير يشير إلى صلاحيات الأشخاص الآخرين **others**.

والصلاحيات تكتب هكذا **rw**، حيث أن **r** اختصاراً لـ **read** وتعني أن الملف قابل للقراءة، و **w** اختصاراً لـ **write** وتعني أن الملف قابل للكتابة، و **x** اختصاراً لـ **execute** وتعني أن الملف قابل للتنفيذ، وإن تم استبدال أي من هذه الحروف بـ **-** فهذا يعني أن ليس لها صلاحية.

كلمة **user** تشير إلى صاحب الملف، وهو في هذه الحالة هو صاحب النظام.

وأخيراً ما قبل اسم الملف هو تاريخ إنشائه.

4.tree: تعرض محتويات المجلد الحالي بشكل بُنية شجرية.

```
[user@localhost Desktop]# tree
```

كما يمكن تحديد دليل أي مجلد آخر.

```
[user@localhost ~]# tree  
home/user/Desktop
```

5.mkdir: تقوم بإنشاء مجلد في المجلد الحالي، (و **mkdir** هي اختصار لـ **make directory**).

```
[user@localhost Desktop]# mkdir  
welcome1
```

كما يمكن إنشاء مجلد في دليل أي مجلد آخر.

```
[user@localhost hello]# mkdir  
home/user/Desktop/welcome2
```

وأخيراً، يمكنُ إنشاءُ مُجلّدٍ يَحتوي عدّةَ مُجلّداتٍ داخِله، وذلكَ عَن طريقِ كتابةٍ **-p** بعدَ الأمرِ.

```
[user@localhost Desktop]# mkdir -p  
Hashem/is/Awesome
```

cat.6: تقومُ بإنشاءِ ملفٍ نصّيٍّ والكتابةُ داخِله، ولِلخروجِ مِنْهُ يجبُ أنْ نضغطَ **CTRL + C**، (و **cat** هي اختصارٌ لـ **concatenate**).

```
[user@localhost hello]# cat > text1  
Welcome to Red Hat with Hashem.
```

ويمكنُ أيضاً استبدالُ مُحتوى ملفٍ نصّيٍّ موجودٍ مُسبقاً، (وهذا الأمرُ سيفتحُ الملفَّ ويطلبُ منك الكتابةَ داخِله).

```
[user@localhost hello]# cat > text1  
Hashem replaced this text.
```

ويمكنُ أيضاً استبدالُ مُحتوى ملفٍ نصّيٍّ، بِمحتوى ملفٍ نصّيٍّ آخرٍ.

```
[user@localhost hello]# cat text2 > text1
```

ويمكن أيضاً إضافة نصٍّ لِمُحتوى ملفٍّ نصِّيٍّ، بدون مسح المُحتوى السابق.

```
[user@localhost hello]# cat >> text1  
Hashem added new text.
```

ويمكن أيضاً إضافة مُحتوى ملفٍّ نصِّيٍّ، إلى مُحتوى ملفٍّ نصِّيٍّ آخرٍ، بدون مسح المُحتوى السابق.

```
[user@localhost hello]# cat text2 >> text1
```

حيثُ أنَّ رمزَ > يشيرُ إلى إنشاءِ الملفِّ أو استبدالِ مُحتواه، ورمزُ >> يشيرُ إلى الإضافةِ إليه بدون مسحِ المُحتوى السابق.

المحاضرة الثانية

نُظْمُ التَّشْغِيلِ القِسْمُ الْعَمَلِيُّ

LinuxTM



كتابة وتنسيق:
هاشيم حسن ونوس

تكملة أوامر الأدلة:

mv.7: تقومُ بنقلِ ملفٍّ من مسارٍ إلى آخرٍ، مع القدرة على تغيير اسمه بعدَ عمليّة النقلِ، (و **mv** هي اختصارٌ لـ **move**).

```
[user@localhost ~]$ mv folder1/file  
folder2
```

ولتغيير اسم الملف بعد نقله نقوم باستبدال الاسم الأخير في المسار الثاني.

```
[user@localhost ~]$ mv folder1/file1  
folder2/file2
```

كما يمكن نقل مُجلّدٍ من مسارٍ إلى آخرٍ، وذلك عن طريق كتابة **-r** بعد الأمر.

```
[user@localhost ~]$ mv  
folder1/subfolder1 folder2 -r
```

كما يمكن عرض رسالة توضيحية عن عمليّة النقلِ، وذلك عن طريق كتابة **-v** بعد الأمر.

```
[user@localhost ~]$ mv folder1/file  
folder2 -v
```

وعند نقل ملف أو مُجلّد إلى مسارٍ يحتوي على نفس اسم الملف أو المُجلّد المنقول، عندها يتم استبداله بما هو منقول، ويُحى الملف أو المُجلّد في المسار الثاني.

8.cp: تقوم بنسخ ملف من مسار إلى آخر، مع القدرة على تغيير اسمه بعد عملية النسخ، (و **cp** هي اختصار لـ **copy**).

```
[user@localhost ~]$ cp folder1/file  
folder2
```

ولتغيير اسم الملف بعد نسخه نقوم باستبدال الاسم الأخير في المسار الثاني.

```
[user@localhost ~]$ cp folder1/file1  
folder2/file2
```

كما يمكن نسخ مُجلّد من مسار إلى آخر، وذلك عن طريق كتابة **-r** بعد الأمر.

```
[user@localhost ~]$ cp  
folder1/subfolder1 folder2 -r
```

كما يمكن عرض رسالة توضيحية عن عملية النسخ، وذلك عن طريق كتابة **-v** بعد الأمر.

```
[user@localhost ~]$ cp folder1/file  
folder2 -v
```

وعند نسخ ملف أو مُجلّد إلى مسارٍ يحتوي على نفس اسم الملف أو المُجلّد المُنسوخ، عندها يتم استبداله بما هو مَنسوخ، ويُحى الملف أو المُجلّد في المسار الثاني.

rm.9: تقوم بحذف ملف، (و **rm** هي اختصار لـ **remove**).

```
[user@localhost ~]$ rm file
```

كما يمكن حذف مُجلّد، وذلك عن طريق كتابة **-r** بعد الأمر.

```
[user@localhost ~]$ rm folder -r
```

كما يمكن إظهار رسالة تأكيد قبل عملية الحذف، وذلك عن طريق كتابة **-i** بعد الأمر، وحينها لا تتم عملية الحذف إلا عندما يكتب المُستخدم حرف **y**، (و **y** هي اختصار لـ **yes**).

```
[user@localhost ~]$ rm file -i
```

ولحذف جميع محتويات مُجلّد ما، دون حذف المُجلّد نفسه، يمكن كتابة رمز ***** بعد اسم المُجلّد.


```
[user@localhost ~]$ rm folder/
```

10.chmod: تقوم بتغيير صلاحيات ملف أو مجلد ما، (و **chmod** هي اختصار لـ **change mode**).

لدينا ثلاث أنواع من صلاحيات:

- I. user:** المُستخدم، ورمزه **u**.
- II. group:** المجموعة، ورمزها **g**.
- III. others:** الآخرون، ورمزهم **o**.

وأيضاً ثلاث أنواع من الصلاحيات:

- I. read:** قراءة، ورمزها **r**.
- II. write:** كتابة، ورمزها **w**.
- III. execute:** تنفيذ، ورمزه **e**.

وأيضاً ثلاث طرق لتعديل هذه الصلاحيات:

- I.** إضافة صلاحيات، وذلك عن طريق الرمز **+**.
- II.** إزالة صلاحيات، وذلك عن طريق الرمز **-**.
- III.** إسناد صلاحيات، وذلك عن طريق الرمز **=**.

في هذا المثال قمنا بإضافة صلاحيات كتابة الملف للمستخدم.

```
[user@localhost ~]$ chmod u+w file
```

وفي هذا المثال قُمنا بإزالة صلاحية قراءة وتنفيذ الملف من المجموعات.

```
[user@localhost ~]$ chmod g-rx file
```

وفي هذا المثال قُمنا بإسناد صلاحية كتابة الملف للآخرين والمجموعات.

```
[user@localhost ~]$ chmod go=w file
```

وإن لم نكتب أي من صاجي الصلاحيات، حينها يتم مُعاملة الأمر كما لو أنك كتبتهم جميعاً.

```
[user@localhost ~]$ chmod +wx file
```

وإن لم نكتب أي من الصلاحيات نفسها، حينها يتم مُعاملة الأمر كما لو أنك كتبتهم جميعاً أيضاً.

```
[user@localhost ~]$ chmod uo- file
```

أي أن الأمر الآتي يقوم بإسناد جميع الصلاحيات لجميع صاجي الصلاحيات.

```
[user@localhost ~]$ chmod = file
```

التعاملُ مَعَ الصلاحيَّاتِ بِالنظامِ الثُّنائيِّ:

هُنَاكَ طَرِيقَةٌ أُخْرَى لِلتَّعْدِيلِ عَلَى الصَّلاحيَّاتِ، وَهِيَ أَنْ نَعْتَبِرَ أَنَّ لِكُلِّ صَاحِبِ صَلاحيَّاتٍ 3 خاناتٍ ثُنائيَّةٍ، وَكُلُّ خَانَةٍ تُمَثِّلُ الصَّلاحيَّاتِ الثَّلَاثَ.

	user			group			others		
	r	w	x	r	w	x	r	w	x
-----	0	0	0	0	0	0	0	0	0
-----x	0	0	0	0	0	0	0	0	1
-----w-	0	0	0	0	0	0	0	1	0
-----wx	0	0	0	0	0	0	0	1	1
-----r--	0	0	0	0	0	0	1	0	0
...									
-wx-w-rw-	0	1	1	0	1	0	1	1	0
-wx-w-rwx	0	1	1	0	1	0	1	1	1
...									
rw-rw-rwx	1	1	1	1	1	1	1	1	1

وَلَكِنَّ هَذِهِ الطَّرِيقَةَ لَا يُمْكِنُ اسْتِعْمَالُهَا بِشَكْلِ مُبَاشِرٍ فِي الـ **Terminal**، وَلَا بُدَّ مِنْ اسْتِعْمَالِ طَرِيقَةِ النِّسْبَةِ الثُّمَانِيَّةِ، وَالَّتِي كَانَ يَجِبُ أَنْ نَشْرَحَ طَرِيقَةَ النِّسْبَةِ الثُّمَانِيَّةِ قَبْلَهَا.

التعاملُ مَعَ الصلاحيّاتِ بِالنظامِ الثُمانيّ:

نقومُ أولاً بالتعبيرِ عَنِ الصلاحيّاتِ بِطريقةِ النظامِ الثُمانيّ – كما ذُكِرَ سابقاً – ، ثُمَّ نقومُ بتحويلِ العددِ الثُمانيّ الناتجِ إلى نَظيرِهِ في النظامِ الثُمانيّ، حيثُ أنُ كُلَّ 3 خاناتٍ في النظامِ الثُمانيّ، يتمُّ التعبيرُ عَنْها بِخانةٍ واحدةٍ في النظامِ الثُمانيّ، ممّا يجعلُ مِنْهُ الطريقةُ المثاليّةُ للتعبيرِ عَنِ الصلاحيّاتِ.

مثلاً لنَقلُ أنّنا نودُّ وضعَ الصلاحيّاتِ الآتيةِ على ملفٍّ ما:

I. جميعُ الصلاحيّاتِ لِلْمُستخدِمِ.

II. الكتابةُ فقط لِلْمَجموعاتِ.

III. القراءةُ والتنفيذُ للآخرين.

نعبّرُ عَنِ ذلكِ بِالنظامِ الثُمانيّ كالآتي:

	user			group			others		
	r	w	x	r	w	x	r	w	x
rwX-w-r-x	1	1	1	0	1	0	1	0	1

ثُمَّ نحوّلُ كُلَّ 3 خاناتٍ على حدىّ إلى النظامِ الثُمانيّ، ممّا يُعطينا العددَ 725، وحينها نكتبُ الأمرَ كما يلي.

```
[user@localhost ~]$ chmod 725 file
```


أوامر النظام:

لا يمكن التعامل مع هذه الأوامر إلا عندما نسجل الدخول إلى حساب ال **root** بشكل مباشر.

1.su: تقوم بتغيير المُستخدم الحالي إلى مُستخدمٍ آخر، وعند كتابتها يتم طلب كلمة سرّ المُستخدم المطلوب، (و **su** هي اختصار لـ **select user**).

```
[user@localhost ~]$ su hashem
```

إن لم نُحدّد اسم المُستخدم المطلوب، حينها يتم الانتقال إلى حساب ال **root**، ويطلب كلمة سرّه طبعاً، (وبالرغم من هذا، فإننا لا نزال غير قادرين على استعمال أوامر النظام، لأنه يجب الدخول إلى ال **root** بشكل مباشر، وليس عن طريق ال **Terminal**).

```
[user@localhost ~]$ su
```

وعند الانتقال من حساب ال **root** لحساب مُستخدمٍ آخر، فإن كلمة سرّه غير مطلوبة، (والعكس غير صحيح طبعاً).

2.passwd: تقوم بتعديل كلمة سرّ المُستخدم الحالي، حيث أنها ستطلب كلمة السرّ الحالية، ثم كلمة السرّ الجديدة، يليها تأكيد كلمة السرّ الجديدة، (و **passwd** هي اختصار لـ **password**).

```
[user@localhost ~]$ passwd
```

كما يمكن تغيير كلمة سرِّ مُستخدمٍ آخرٍ ما، وذلك فقط عندما نكون في حساب الـ **root**، وحينها لن نطلب كلمة السرِّ الحالية الخاصة به.

```
[root@localhost ~]# passwd hashem
```

3.groupadd: تقوم بإضافة مجموعة جديدة باسم مُحدّد.

```
[root@localhost ~]# groupadd group1
```

كما يمكن وضع رقم تعريفٍ **ID** للمجموعة، ويجب أن يكون عدداً غير مُكرّر من قبل، وذلك عن طريق كتابة **-g** بعد الأمر.

```
[root@localhost ~]# groupadd group1 -g 999999
```

4.groupmod: تقوم بالتعديل على مجموعة ما، فمثلاً لتعديل اسم مجموعة، نكتب **-n** بعد الأمر، ثمَّ اسم المجموعة الجديد، يليه اسمها القديم، (و **groupmod** هي اختصارٌ لـ **group modify**).

```
[root@localhost ~]# groupmod -n group2 group1
```

5.groupdel: تقوم بحذف مجموعة ما، عَنْ طريق تمرير اسمها،
(و **groupdel** هي اختصار لـ **group delete**).

```
[root@localhost ~]# groupdel group2
```

6.useradd: تقوم بإضافة مُستخدمٍ جديدٍ باسمٍ مُحدّدٍ، وبلا كلمة سرّ، (يمكنُ تعديلُ هذا لاحقاً باستعمالِ أمرٍ **passwd**).

```
[root@localhost ~]# useradd user1
```

كما يمكنُ وضعُ رقمٍ تعريفٍ **ID** لِلْمُستخدمِ، وَيَجِبُ أَنْ يَكُونَ عددًا غير مُكرّرٍ مِنْ قَبْلٍ، وَذَلِكَ عَنْ طريقِ كتابةِ **-u** بعدَ الأمرِ.

```
[root@localhost ~]# useradd user1 -u  
999999
```

كما يمكنُ تحديدُ مَسَارِ مُجلّدِ الـ **home** الخاصِّ بِالْمُستخدمِ، وَذَلِكَ عَنْ طريقِ كتابةِ **-d** بعدَ الأمرِ، ثُمَّ المَسَارِ المطلوبِ.

```
[root@localhost ~]# useradd user1 -d  
/home/user1
```

7.usermod: تقومُ بالتعديلِ على مُستخدمٍ ما، فمثلاً لتعديلِ نوعِ مُفسّرِ الأوامرِ الخاصِّ بهِ، نكتبُ **-s** بعدَ الأمرِ، ثُمَّ مَسَارَ مُفسّرِ الأوامرِ الجديدِ، وفي هَذَا المِثَالِ عدّلناه إلى النوعِ **SH**، (و **usermod** هي اختصارٌ لـ **user modify**).

```
[root@localhost ~]# usermod user1 -s /bin/sh
```

كما يمكنُ إضافةَ المُستخدمِ إلى مَجْموعَةٍ ما، وذلكَ عَنْ طريقِ كتابةِ **-G** بعدَ الأمرِ، ثُمَّ اسمِ المَجْموعَةِ.

```
[root@localhost ~]# usermod user1 -G group1
```

8.userdel: تقومُ بِحذفِ مُستخدمٍ ما، عَنْ طريقِ تَمَرِيرِ اسمِهِ، (و **userdel** هي اختصارٌ لـ **user delete**).

```
[root@localhost ~]# userdel user1
```

9.groups: تقومُ بِإظهارِ المَجْموعاتِ الَّتِي يَنْتَمِي إليها مُستخدمٌ ما.

```
[root@localhost ~]# groups hashem
```


المحاضرة الثالثة

نُظْمُ التَّشْغِيلِ القِسْمُ الْعَمَلِيُّ

LinuxTM



كتابة وتنسيق:
هاشيم حسن ونوس

حالات الإجراءات - المهام:

1. **new**: الإجراءية تكون في طور التكوين، مع عدم حجز المعالج والذاكرة.

2. **ready**: عندما تستحوذ الإجراءية على الموارد، وذلك قبل تنفيذها.

3. **running**: عندما تكون الإجراءية قيد التنفيذ، وفي عدة حالات تمر إلى حالة **waiting**، ثم تعود إلى حالة **ready**، وهم:

- a. انتهاء وقت تنفيذ الإجراءية.
- b. إجراءية أخرى تملك أولوية أعلى.

4. **waiting**: عندما تحدث مقاطعة أو انتظار موارد معينة، أو إجراءية أخرى.

5. **terminated**: عندما ينتهي تنفيذ الإجراءية، أو عندما تلغى.

يوضح المخطط الآتي حالات الإجراءات - المهام، وكيفية التنقل بينها:



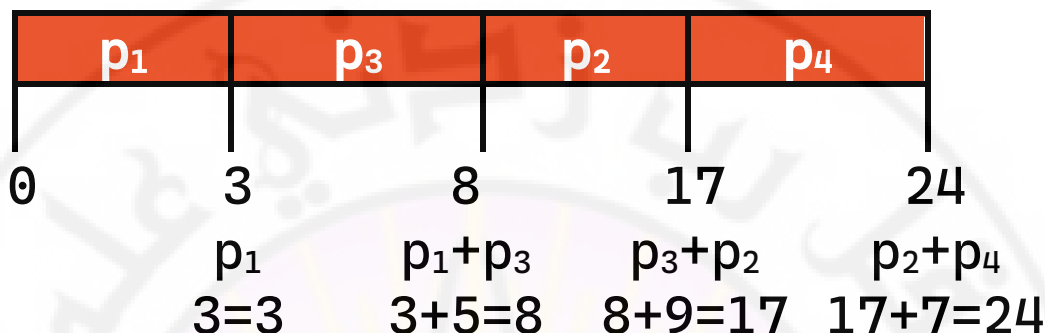
أولاً: خوارزمية القادم أولاً يُخدم أولاً – First Come First Serves :(FCFS)

لِكُلِّ مُهِمَّةٍ – **Process**، هُنَالِكَ زَمَنُ تَنْفِيذٍ – **Burst Time**، يُقَاسُ
بِالْمِئَلِي ثَانِيَةٍ – **Milliseconds**.

مَثَلًا: تَأْتِي أَرْبَعُ مَهَامٍ بِالترتيبِ الآتِي: p_1 , p_3 , p_2 , p_4 ، وَالْمَطْلُوبُ
حِسَابُ مَخْطَاطِ غَانْت – **Gantt Chart**، وَالْمُتَوَسَّطِ الْحِسَابِيِّ لِزَمَنِ
الانتظار – **Average Waiting Time**:

Process (p)	Burst Time (BS)
p_1	3
p_2	9
p_3	5
p_4	7

الحل:



لِحسابِ الْمُتَوَسِّطِ الْحِسَابِيِّ لِزَمَنِ الْإِنْتِظَارِ، نَقُومُ بِحِسَابِهِ لِكُلِّ مَهْمَةٍ عَلَى حِدَةٍ، وَزَمَنِ الْإِنْتِظَارِ - **Waiting Time**، فِي خَوَازِمِيَّةِ **FCFS** هُوَ نَفْسُهُ زَمَنِ الْبِدَايَةِ - **Starting Time**:

$$WT (Waiting Time) = ST (Starting Time)$$

$$p_1: 0ms$$

$$p_2: 8ms$$

$$p_3: 3ms$$

$$p_4: 17ms$$

$$AWT (Average Waiting Time) =$$

$$= \frac{WT_1 + WT_2 + \dots + WT_n}{n} = \frac{0 + 8 + 3 + 17}{4} = 7ms$$

ثانياً – خوارزمية الجدولة الدائرية – Round Robin (RR):

في هذه الخوارزمية يُوجد ما يُعرف بِطول الشريحة الزمنية – **Quantum**، والتي تُقاسُ أيضاً بِالْمِلي ثانية – **Milliseconds**، وهي مقدارُ الزمنِ الأعظمي الذي يمكنُ لِمَهْمَةٍ ما أَنْ تَحْجِزَهُ في المَرَّةِ الواحدة، ثُمَّ الانتقالَ لِلْمَهْمَةِ الَّتِي تَلِيها، وتكرارُ هذه العملية إلى أَنْ يَتِمَّ الانتهاءُ مِنْ كُلِّ المَهَامِّ.

مثلاً: تأتي خَمْسُ مَهَامِّ، وَلَدِينا طولُ الشريحة الزمنية – **Quantum** هو **5ms**، والمَطْلُوبُ حِسابُ مَخَطِّ غانت – **Gantt Chart**، والمتوسِّطِ الحِسابيِّ لِزَمَنِ الانتظارِ – **Average Waiting Time**:

Process (p)	Burst Time (BS)
p ₁	12
p ₂	8
p ₃	4
p ₄	10
p ₅	5

الحل:

يتم تنفيذ المهام بالترتيب ذاته، (لأن الترتيب لم يُعطى)، وفي أول دورة، يتم تنفيذ كل مهمة لمدة 5ms فقط، ويصبح المخطط كالآتي:

Process (p)	Burst Time (BS)
p ₁	12 -> 7
p ₂	8 -> 3
p ₃	4 -> _
p ₄	10 -> 5
p ₅	5 -> _

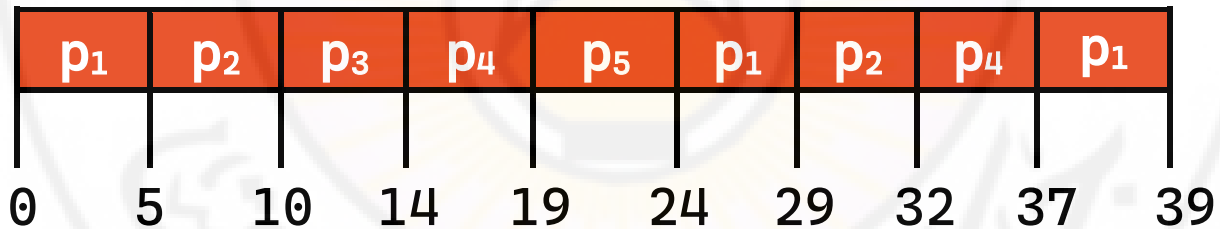
والدورة التالية كالآتي:

Process (p)	Burst Time (BS)
p ₁	12 -> 7 -> 2
p ₂	8 -> 3 -> _
p ₃	4 -> _
p ₄	10 -> 5 -> _
p ₅	5 -> _

وَيَتِمُّ الْقِيَامُ بِدَوْرَةٍ أُخِيرَ لِأَجْلِ p_1 :

Process (p)	Burst Time (BS)
p_1	12 -> 7 -> 2 -> _
p_2	8 -> 3 -> _
p_3	4 -> _
p_4	10 -> 5 -> _
p_5	5 -> _

وَيَكُونُ الْمُخَطَّطُ كَالآتِي:



يُعْطَى زَمَنُ الْإِنْتِظَارِ - **Waiting Time**، فِي خَوَازِمِيَّةِ الْجَدْوَلِ
الدَّائِرِيَّةِ - **RR**، عَلَى أَنَّهُ زَمَنُ الْبَدَايَةِ الْأَوَّلِ - **Starting Time₁**،
مَجْمُوعٌ مَعَ فَرْقِ زَمَنِ الْبَدَايَةِ الَّذِي يَلِيهِ - **Starting Time_n**، وَزَمَنُ
نِهَائِيَّتِهِ - **Finish Time_{n-1}**:

WT (Waiting Time) =

= ST₁(Starting Time₁) +

+ (ST₂(Starting Time₂) - FT₁(Finishing Time₁)) +

+ (ST₂(Starting Time₃) - FT₁(Finishing Time₂)) +

+ ... +

+ (ST_n(Starting Time_n) - FT_{n-1}(Finishing Time_{n-1}))

p₁: 0 + (24 - 5) + (37 - 29) = 27ms

p₂: 5 + (29 - 10) = 14ms

p₃: 10ms

p₄: 14 + (32 - 19) = 27ms

p₅: 19ms

AWT (Average Waiting Time) =

= $\frac{27 + 24 + 10 + 27 + 19}{5} = 21.4ms$

ثالثاً – خوارزمية الأقصر تنفيذاً أولاً – :(SJF) Shortest Job First

في هذه الخوارزمية يتم تنفيذ المهمة ذي زمن التنفيذ – **Burst Time** الأقل، بغض النظر عما أتى أولاً، (زمن الوصول – **Arrival Time**).

ملاحظة: يتم تنفيذ أول مهمة وصلت دوماً، حتى لو كانت ذا زمن التنفيذ الأكثر.

مثلاً: تأتي أربع مهام كما هو موضح في الجدول أدناه، والمطلوب حساب مخطط غانت – **Gantt Chart**، والمتوسط الحسابي لزمن الانتظار – **Average Waiting Time**، والمتوسط الحسابي لزمن التنفيذ الكلي – **Turnaround Time**:

Process (p)	Arrival Time (AT)	Burst Time (BT)
p ₁	0	7
p ₂	2	4
p ₃	4	1
p ₄	5	4

الحل:

إنَّ المَهْمَّةَ ذا زَمَنِ التنفيذِ الأقصرِ هي p_3 ، ولهذا يَتِمُّ تنفيذهُ المَهْمَّةِ p_1 أولاً، ثُمَّ المَهْمَّةِ p_3 ، وهكذا:

p_1	p_3	p_2	p_4
0	7	8	12
	p_1	p_1+p_3	p_3+p_2
	$7=7$	$7+1=8$	$8+4=12$
			p_2+p_4
			$12+4=16$

يُعطى زَمَنُ الانتظارِ - **Waiting Time**، في خوارزميةِ الأقصرِ تنفيذاً أولاً - **SJF**، على أَنَّهُ زَمَنُ البداية - **Starting Time**، ناقصَ زَمَنِ الوصولِ - **Arrival Time**:

$$WT (Waiting Time) =$$

$$= ST (Starting Time) - AT (Arrival Time)$$

$$p_1: 0 - 0 = 0ms$$

$$p_2: 8 - 2 = 6ms$$

$$p_3: 7 - 4 = 3ms$$

$$p_4: 12 - 5 = 7ms$$

AWT (Average Waiting Time) =

$$= \frac{0 + 6 + 3 + 7}{4} = 4ms$$

يُعطى زَمَنُ التنفيذِ الكُلِّيّ – **Turnaround Time**، في هذه الخوارزمية على أَنَّهُ زَمَنُ الانتهاءِ – **Finish Time**، ناقِصَ زَمَنِ الوصولِ – **Arrival Time**:

TT (Turnaround Time) =

$$= FT (Finish Time) - AT (Arrival Time)$$

$$p_1: 7 - 0 = 7ms$$

$$p_2: 12 - 2 = 10ms$$

$$p_3: 8 - 4 = 4ms$$

$$p_4: 16 - 5 = 11ms$$

ATT (Average Turnaround Time) =

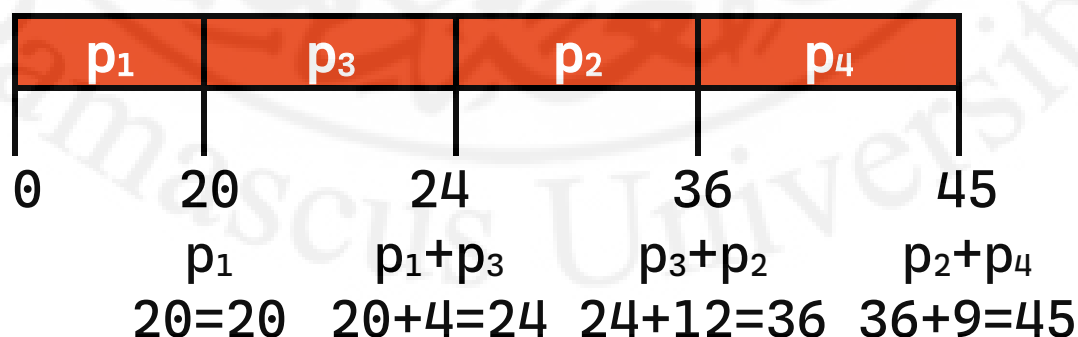
$$= \frac{TT_1 + TT_2 + \dots + TT_n}{n} = \frac{7 + 10 + 4 + 11}{4} = 8ms$$

لِنَفْرِضَ مِثَالاً آخِراً: تَأْتِي أَرْبَعُ مَهَامٍّ كَمَا هُوَ مُوَضَّحٌ فِي الْجَدُولِ أَدْنَاهُ،
وَالْمَطْلُوبُ حِسَابُ مُخَطَّطِ غَانْت - **Gantt Chart**، وَالْمُتَوَسِّطِ
الْحِسَابِيِّ لِزْمَنِ الْإِنْتَظَارِ - **Average Waiting Time**،
وَالْمُتَوَسِّطِ الْحِسَابِيِّ لِزْمَنِ التَّنْفِيدِ الْكُلِّيِّ - **Turnaround Time**:

Process (p)	Arrival Time (AT)	Burst Time (BT)
p ₁	0	20
p ₂	3	12
p ₃	2	4
p ₄	5	9

الحلُّ:

إِنَّ الْمَهْمَةَ ذَا زَمَنِ التَّنْفِيدِ الْأَقْصَرَ هِيَ **p₃**، وَلِهَذَا يَتِمُّ تَنْفِيدُ الْمَهْمَةِ **p₁** أَوَّلًا،
ثُمَّ الْمَهْمَةُ **p₃**، وَهَكَذَا:



المُتوسَّط الحِسابيُّ لِزَمَنِ الانتظارِ – **Average Waiting Time**:

WT (Waiting Time) =

= ST (Starting Time) – AT (Arrival Time)

$p_1: 0 - 0 = 0ms$

$p_2: 20 - 2 = 18ms$

$p_3: 24 - 3 = 21ms$

$p_4: 36 - 5 = 31ms$

AWT (Average Waiting Time) =

$$= \frac{0 + 18 + 21 + 31}{4} = 17.5ms$$

المُتوسَّط الحِسابيُّ لِزَمَنِ التنفيذِ الكُلِّيِّ – **Average**
:Turnaround Time

TT (Turnaround Time) =

= FT (Finish Time) – AT (Arrival Time)

$$p_1: 20 - 0 = 20ms$$

$$p_2: 36 - 12 = 24ms$$

$$p_3: 24 - 4 = 20ms$$

$$p_4: 45 - 9 = 36ms$$

ATT (Average Turnaround Time) =

$$= \frac{20 + 24 + 20 + 36}{4} = 25ms$$