الكلية التطبيقية

نـظـم التشـــغـيـل

سنة ثانية

مدرس المقرر

د.م. عبدالرزاق المحمد المحمود

2026 - 2025

(1)

مفاهيم أساسية Fundamentals

نظام التشغيل Operating System هو برمجية أساسية تعمل كوسيط بين المستخدم والأجهزة المادية للحاسوب، تدير جميع موارد الحاسوب مثل: المعالج CPU ، الذاكرة RAM ، أجهزة التخزين (الأقراص الصلبة أو SSD) ، وأجهزة الإدخال/الإخراج (الشاشة، لوحة المفاتيح، والماوس).

كما يوفر نظام التشغيل واجهة سهلة الاستخدام، سواء كانت نصية مثل موجه الأوامر أو رسومية مثل واجهة Windows، مما يتيح للمستخدمين تشغيل التطبيقات وإدارة الملفات بسهولة.

بدون نظام تشغيل، يحتاج الحاسوب لكتابة تعليمات معقدة بلغة الآلة لكل مهمة، وهذا أمر غير عملي. على سبيل المثال: عندما تفتح متصفح الإنترنت، يقوم نظام التشغيل بتخصيص الذاكرة، إدارة الاتصال بالشبكة، وعرض الواجهة الرسومية، دون أن تحتاج إلى التدخل في كل هذه التفاصيل.

تاريخ أنظمة التشغيل

بدأت أنظمة التشغيل في الظهور مع تطور الحواسيب في منتصف القرن العشرين.

1. **الخمسينيات – البدايات** : الحواسيب مثل IBM 701 لم تكن تحتوي على أنظمة تشغيل بالمعنى الحقيقي، وإنما كان المبرمجون يكتبون التعليمات مباشرة بلغة الآلة، وكانت المهام تُنفذ يدويا عبر بطاقات مثقبة أو أشرطة مغناطيسية، كانت هذه العملية بطيئة وتتطلب خبرة تقنية عالية.
2. **الستينيات – الأنظمة الدفعية** : ظهرت أنظمة تشغيل بدائية مثل OS/360 من IBM ركزت هذه الأنظمة على معالجة المهام بشكل دفعي Batch Processing ، حيث يتم تجميع المهام وتنفيذها دفعة واحدة، كما بدأت بدعم المهام المتعددة Multiprogramming، مما سمح بتشغيل عدة برامج في وقت واحد عن طريق تقسيم وقت المعالج.
3. **السبعينيات – ثورة UNIX** حيث طوّرت مختبرات بيل نظام UNIX ، وهو نظام تشغيل مرن ومحمول تم كتابته بلغة C ، وقد أثر ظهور UNIX بشكل كبير على أنظمة التشغيل الحديثة بفضل تصميمه المعياري ودعمه للشبكات.

4. **الثمانينيات – الحواسيب الشخصية** : مع انتشار الحواسيب الشخصية مثل IBM PC، ظهرت أنظمة تشغيل مثل MS-DOS من مايكروسوفت، والتي اعتمدت على واجهة نصية، وفي نفس الوقت قدمت أبل نظام Mac OS مع واجهة رسومية مبتكرة، مما جعل الحواسيب أكثر سهولة للمستخدمين العاديين.
5. **التسعينيات وما بعدها** : أصبحت الواجهات الرسومية معياراً أساسياً مع إصدارات مثل Windows و Linux الذي تم تطويره كبديل مفتوح المصدر عوضاً عن UNIX .
6. **في القرن الحادي والعشرين**، ظهرت أنظمة تشغيل للأجهزة المحمولة مثل Android و iOS، بالإضافة إلى أنظمة الحوسبة السحابية.

هدف أنظمة التشغيل

تسعى أنظمة التشغيل إلى تحقيق مجموعة من الأهداف لضمان الأداء الفعال وسهولة الاستخدام :

1. **إدارة الموارد** : يقوم نظام التشغيل بتخصيص الموارد بين العمليات المختلفة مثل (وقت المعالج، مساحة الذاكرة، وسعة التخزين)، على سبيل المثال، عند تشغيل عدة تطبيقات، يحدد النظام أولويات التنفيذ لتجنب التعارض.
2. **توفير واجهة استخدام مريحة** : يتيح نظام التشغيل التفاعل مع الحاسوب عبر واجهات مثل موجه الأوامر Command Line Interface أو الواجهات الرسومية Graphical User Interface على سبيل المثال، واجهة Windows توفر أيقونات ونوافذ سهلة الاستخدام.
3. **تشغيل التطبيقات بكفاءة** : يوفر النظام بيئة تنفيذ مستقرة للتطبيقات، مع إدارة الموارد الحاسوبية (الذاكرة ، والأجهزة الطرفية).
4. **الأمان والحماية** : يحمي النظام البيانات والموارد من الوصول غير المصرح به باستخدام آليات مثل كلمات المرور، التشفير، الأذونات.
5. **تحسين الأداء** : يعمل النظام على تقليل وقت الانتظار وزيادة سرعة الاستجابة من خلال تقنيات مثل جدولة العمليات Scheduling وإدارة الذاكرة الافتراضية.
6. **التوافقية** : يضمن تشغيل التطبيقات عبر منصات مختلفة، مثال ذلك يضمن Android تشغيل التطبيقات على أجهزة بمواصفات متنوعة.

أنواع أنظمة التشغيل

تختلف أنظمة التشغيل بناء على الغرض والتطبيق :

1. **أنظمة التشغيل أحادية المهام : Single-Tasking** تدعم تنفيذ برنامج واحد فقط في كل مرة، مثال : MS-DOS، حيث يمكن للمستخدم تشغل برنامج واحد مثل (معالج نصوص مثلاً) ثم يغلقه لتشغيل برنامج آخر، هذه الأنظمة بسيطة لكنها محدودة.
2. **أنظمة التشغيل متعددة المهام : Multi-Tasking** تتيح تشغيل عدة برامج في وقت واحد مثل Windows، Linux، mac OS باستخدام تقنيات مثل الجدولة الزمنية Time-Sharing. على سبيل المثال، يمكن تصفح الإنترنت، تشغيل الموسيقى، وتحرير مستند (كل هذه المهام يتم تنفيذها في نفس الوقت).
3. **أنظمة التشغيل الموزعة : Distributed OS** تدير مجموعة من الحواسيب كأنها كيان واحد، مما يتيح مشاركة الموارد عبر الشبكة، تُستخدم في الحوسبة السحابية مثل Apache Hadoop
4. **أنظمة التشغيل المدمجة : Embedded OS** وهي أنظمة مصممة للأجهزة ذات الموارد المحدودة مثل الهواتف الذكية، الأجهزة المنزلية (الثلاجات الذكية)، أو السيارات، مثال ذلك Android، iOS
5. **أنظمة التشغيل في الوقت الحقيقي : Real-Time OS** تُستخدم في التطبيقات التي تتطلب استجابة فورية، مثل أنظمة التحكم في الطائرات، الأجهزة الطبية (أجهزة مراقبة القلب)، أو الروبوتات وتنقسم إلى :
    - **الوقت الحقيقي الصلب : Hard Real-Time** لا يسمح بأي تأخير / مثل أنظمة الطيران
    - **الوقت الحقيقي الناعم : Soft Real-Time** يسمح بتأخيرات طفيفة / مثل بث الفيديو

معايير أنظمة التشغيل الموحدة

يهدف توحيد المعايير إلى ضمان التوافق بين الأنظمة المختلفة، وتسهيل تطوير البرمجيات وتشغيلها عبر منصات متعددة بدلاً من تطوير نسخة مختلفة لكل نظام تشغيل، مما يقلل الجهد والتكلفة.

أهم المعايير التي تم تطويرها :

1. **واجهات أنظمة التشغيل المحمولة Portable Operating System Interface (POSIX)** :

معيار طورته IEEE لضمان التوافق بين أنظمة التشغيل الشبيهة بـ UNIX (Linux وmac OS)

يحدد معايير موحدة لـ: موجه الأوامر Shell ، واجهات برمجة التطبيقات APIs، إدارة الملفات، العمليات، والمعالجة المتوازية Threads.

2. **واجهات برمجة التطبيقات APIs** :

وهي واجهات معيارية تتيح للمطورين كتابة برامج تعمل على أنظمة تشغيل مختلفة ،،، مثل OpenGL للرسوميات ثلاثية الأبعاد، OpenAL للصوت ، وDirectX من مايكروسوفت.

3. **معايير الملفات** :

تحدد كيفية تخزين البيانات وتنظيمها، مثل FAT32 ، NTFS، و ext4

على سبيل المثال: يدعم FAT32 التوافق بين Windows وأجهزة USB ، بينما يوفر NTFS ميزات أمان متقدمة.

4. **معايير الشبكات** :

مجموعة من البروتوكولات مثل TCP/IP وHTTP لضمان التواصل بين الأنظمة المختلفة عبر الشبكات، كتصفح الإنترنت أو مشاركة الملفات.

5. **معايير الأجهزة** :

تتيح لنظام التشغيل التعرف على الأجهزة الطرفية وإدارتها بسهولة، مثل USB وPCI Express.

طبقات نظام التشغيل

يتكون نظام التشغيل من عدة طبقات مترابطة تعمل معاً لتوفير الخدمات، الطبقات الرئيسية هي:

1. **النواة : Kernel**

هي الجزء الأساسي من نظام التشغيل، وتتفاعل مباشرة مع العتاد، تدير النواة (المعالج، الذاكرة، وأجهزة الإدخال/الإخراج)، هناك أنواع مختلفة للنواة : (أحادية، دقيقة، هجينة).

2. **استدعاء النظام : System Calls**

واجهة برمجية تتيح للتطبيقات طلب خدمات من النواة، مثل (فتح ملف أو إنشاء عملية جديدة)، على سبيل المثال، عندما يحفظ برنامج مستنداً، يستخدم استدعاء نظام System Call للوصول إلى القرص الصلب.

3. **إدارة العمليات : Process Management**

العملية هي البرنامج الذي يتم تنفيذه، وإدارة العمليات هي التي تتحكم في (إنشاء العمليات، جدولتها، وإنهائها) ، يستخدم النظام خوارزميات جدولة مثل Round-Robin لتوزيع وقت المعالج بين العمليات.

4. **إدارة الذاكرة : Memory Management**

مهمتها تخصيص الذاكرة للعمليات وإدارة الذاكرة الافتراضية Virtual Memory التي تتيح للبرامج استخدام مساحة ذاكرة أكبر من المتوفرة فعلياً عن طريق استخدام القرص الصلب كذاكرة إضافية.

5. **إدارة أنظمة الملفات : File System Management**

مهمتها تنظم تخزين الملفات وإدارتها على أجهزة التخزين، على سبيل المثال، يستخدم Windows نظام ملفات NTFS ، بينما يستخدم Linux نظام ext4 .

6. **إدارة الأجهزة : Device Management**

تتحكم في الأجهزة الطرفية مثل (الطابعات والماسحات الضوئية) عبر برامج التعريف Drivers.

7. **واجهة المستخدم : User Interface**

توفر هذه الطبقة وسيلة التفاعل ما بين المستخدم والنظام، ممكن أن تكون الواجهات إما نصية مثل Bash في Linux أو رسومية مثل mac OS أو Windows.

أنواع بنية النواة في أنظمة التشغيل

تختلف بنية أنظمة التشغيل بناءً على طريقة تنظيم مكونات النواة، فيما يلي الأنواع الرئيسية:

1. النواة الأحادية (Monolithic Kernel)

في هذا النوع، تعمل جميع خدمات النظام (إدارة الملفات، العمليات، الذاكرة، والأجهزة) ضمن مساحة نواة واحدة.

المزايا: سرعة عالية بسبب التكامل الوثيق والتواصل المباشر بين المكونات.

العيوب: تعقيد في الصيانة، حيث يمكن أن يؤدي خطأ في جزء واحد إلى انهيار النظام بأكمله.

أمثلة: Linux، UNIX، FreeBSD.

ملاحظة: معظم الأنظمة الأحادية الحديثة (مثل Linux) تدعم الوحدات النمطية (Loadable Kernel Modules)، مما يسمح بتحميل بعض المكونات ديناميكياً دون إعادة تشغيل النظام، مما يمنحها مرونة إضافية.

2. النواة الدقيقة (Microkernel)

في هذا النوع، تُختزل النواة إلى الحد الأدنى من الخدمات الأساسية (التواصل بين العمليات، جدولة العمليات، إدارة الذاكرة الأساسية).

بينما تعمل باقي الخدمات (نظام الملفات، إدارة الأجهزة، بروتوكولات الشبكة) كعمليات منفصلة في مساحة المستخدم.

المزايا: مرونة عالية، سهولة التعديل والصيانة، استقرار أكبر (فشل خدمة لا يؤدي إلى انهيار النظام).

العيوب: أداء أبطأ نسبياً بسبب الحاجة إلى التنسيق والاتصال بين العمليات المختلفة.

أمثلة: QNX، Minix، Mach.

3. النواة الهجينة (Hybrid Kernel)

تجمع بين خصائص النواة الأحادية والنواة الدقيقة، تعمل بعض الخدمات الحيوية داخل النواة لتحقيق الأداء العالي، بينما تعمل خدمات أخرى كعمليات منفصلة في مساحة المستخدم.

المزايا: توازن بين الأداء والمرونة.

العيوب: تعقيد في التصميم مقارنة بالأنواع الأخرى.

أمثلة: Windows NT (والأنظمة المبنية عليه مثل Windows 10/11)، macOS، Solaris.

4. البنية الطبقية (Layered Architecture)

وهي ليست نوعاً مستقلاً من أنواع النواة وإنما نمطاً معمارياً يمكن تطبيقه على أنواع النواة المختلفة.

في هذه البنية يتم تنظيم النظام في طبقات متتالية، حيث تتفاعل كل طبقة مع الطبقة التي تليها فقط عبر واجهات محددة.

المزايا: سهولة التصميم، العزل بين المكونات، تبسيط عملية التطوير والصيانة.

العيوب: انخفاض الأداء نسبياً بسبب مرور الطلبات عبر طبقات متعددة.

أمثلة: بعض الأنظمة المدمجة (Embedded Systems).

5. البنية القائمة على المحاكاة الافتراضية (Virtualization-based Architecture)

هذا النموذج لا يُصنف كنوع من أنواع بنية النواة، وإنما هو طبقة إضافية تعمل فوق أنظمة التشغيل، تسمح بتشغيل عدة أنظمة تشغيل (ضيوف) بشكل متزامن ومعزول عن بعضها البعض، ويُستخدم على نطاق واسع في الحوسبة السحابية ومراكز البيانات.

المزايا: عزل عالي بين الأنظمة، استغلال أمثل للموارد، سهولة الاختبار والترحيل.

العيوب: حمل إضافي (Overhead) على الأداء مقارنة بالتشغيل المباشر.

أمثلة: VMware ESXi، Microsoft Hyper-V.