

1. مقدمة عن نظم التشغيل (Introduction & What Operating Systems Do) تعريف نظام التشغيل (OS): هو برنامج يُعتبر بمثابة الوسيط (Interface) بين عتاد الحاسوب (Hardware) والمستخدمين، ويهدف إلى ضمان تنفيذ البرامج بكفاءة وسهولة. إدارة موارد الحاسوب (مثل: المعالج، الذاكرة، وحدات التخزين، وأجهزة الإدخال والإخراج). العمل كـ "موزع للموارد" (Resource Allocator) ومتحكم في البرامج لمنع الأخطاء وسوء استخدام الجهاز. تلبية احتياجات المستخدمين (السهولة وسرعة الأداء في الأنظمة الشخصية، مقابل كفاءة واستغلال الموارد في الأنظمة المشتركة والخوادم).

2. مكونات نظام الحاسوب (Computer System Components) ينقسم نظام الحاسوب إلى 4 مكونات رئيسية: 1. العتاد (Hardware): الموارد الأساسية للحاسوب (المعالج CPU، الذاكرة، وأجهزة الإدخال والإخراج). 2. نظام التشغيل (Operating System): يتحكم وينسق استخدام العتاد بين التطبيقات والمستخدمين المختلفين. 3. برامج التطبيقات (Application Programs): تحدد كيفية استخدام موارد النظام لحل مشكلات المستخدم (مثل: متصفحات الويب، معالجات النصوص، الألعاب). 4. المستخدمون (Users): الأشخاص، الآلات، أو الأنظمة الأخرى التي تستخدم الحاسوب. 3. تنظيم نظام الحاسوب والعمليات (Computer System Organization & Operation) آلية العمل: يتكون النظام من معالج (أو عدة معالجات) ومتحكمات أجهزة تتصل فيما بينها عبر ناقل مشترك (Common Bus) للوصول إلى الذاكرة المشتركة. المقاطعات (Interrupts): تعتمد نظم التشغيل بشكل أساسي على المقاطعات؛ حيث يقوم جهاز الإدخال/الإخراج بإرسال إشارة مقاطعة للمعالج عند إتمام مهمته لتغيير مسار التنفيذ مؤقتاً عبر جدول ناقل المقاطعات (Interrupt Vector). الوضع المزدوج (Dual-Mode Operation): لحماية النظام، وضع المستخدم (User Mode): لتنفيذ تطبيقات البرامج العادية. وضع النواة (Kernel Mode): للعمليات الحساسة والمصرح لها بصلاحيات كاملة، ويتم التبديل بينهما عبر بت نمط (Mode Bit). 4. إدارة العمليات والذاكرة والتخزين (Process, Memory Management): العملية هي برنامج قيد التنفيذ (Program in execution) وتعتبر وحدة العمل الأساسية في النظام. يتولى نظام التشغيل مسؤوليات مثل إنشاء/حذف العمليات، إيقافها واستئنافها، وتوفير آليات المزامنة والاتصال ومعالجة التجاوزات (Deadlocks). إدارة الذاكرة (Memory Management): تتطلب تنفيذ البرامج تحميل تعليماتها وبياناتها (أو جزء منها) إلى الذاكرة الرئيسية. يتابع نظام التشغيل الأجزاء المستعملة من الذاكرة ومن يستخدمها، ويخصص أو يحرر مساحات الذاكرة حسب الحاجة. إدارة التخزين (Storage Management): يوفر نظام التشغيل نظرة منطقية موحدة لتخزين المعلومات عبر تجريد الخصائص الفيزيائية لوحدة التخزين إلى "ملفات" (Files) وإدارتها في مجلدات (Directories).

5. المعماريات والأنظمة مفتوحة المصدر (System Architectures & Open-Source OS) أنظمة المعالجات المتعددة (Multiprocessors): تزيد من إنتاجية النظام وموثوقيته (Fault Tolerance)، وتنقسم إلى معالجة متماثلة (Symmetric Multiprocessing) وغير متماثلة (Asymmetric Multiprocessing). نظم التشغيل مفتوحة المصدر (Open-Source Operating Systems): تتيح الشيفرة البرمجية للمستخدمين ليتمكنوا من دراستها، تعديلها، وتحسينها (مثل أنظمة Linux و BSD UNIX).

1. خدمات نظم التشغيل (Operating System Services) يقدم نظام التشغيل بيئة لتنفيذ البرامج وخدمات للمستخدمين، واجهة المستخدم (User Interface - UI): توفر وسيلة للتفاعل مع المستخدم، وتكون إما عبر سطر الأوامر (CLI)، أو الواجهة الرسومية (GUI)، أو الأوامر النصية (Batch). تنفيذ البرامج (Program Execution): قدرة النظام على تحميل البرنامج إلى الذاكرة وتنفيذه وإنهائه (سواء بشكل طبيعي أو في حال حدوث خطأ). عمليات الإدخال والإخراج (I/O Operations): التعامل مع طلبات البرامج المتعلقة بملفات أو أجهزة الإدخال والإخراج. معالجة وملفات النظام (File-system Manipulation): قراءة، كتابة، إنشاء، حذف، والبحث في الملفات والمجلدات مع إدارة الصلاحيات. الاتصالات (Communications): تبادل المعلومات بين العمليات سواء على نفس الحاسوب أو عبر الشبكة (عن طريق الذاكرة المشتركة Shared Memory أو تمرير الرسائل Message Passing). اكتشاف الأخطاء (Error Detection): مراقبة الأخطاء في المعالج، الذاكرة، أو الأجهزة واتخاذ الإجراء المناسب لضمان استقرار العمل. تخصيص الموارد (Resource Allocation): توزيع الموارد (مثل دورات المعالج، الذاكرة، والتخزين) عند تشغيل عدة مستخدمين أو مهام بالتوازي. المحاسبة (Accounting): تتبع استخدام المستخدمين للموارد وكميتها. الحماية والأمان (Protection and Security): التحكم في الوصول إلى موارد النظام، وتوثيق هوية المستخدمين وحماية الأجهزة الخارجية من محاولات الوصول غير المرغوب فيها. 2. استدعاءات النظام (System Calls) التعريف: هي واجهة البرمجة (Programming Interface) للخدمات التي يقدمها نظام التشغيل، وتُكتب عادة

بلغات عالية المستوى مثل C أو ++C. آلية العمل: يتم الوصول إليها غالباً عبر واجهة برمجة التطبيقات (API) مثل Win32 لأنظمة ويندوز أو POSIX لأنظمة يونكس ولينكس. يتم تخصيص رقم لكل استدعاء نظام، ويحتفظ الجدول (Table) بهذه الأرقام لتنفيذ الطلب داخل النواة (Kernel) مع إخفاء تفاصيل التنفيذ المعقدة عن المبرمج. أنواع استدعاءات النظام الرئيسية: 1. التحكم بالعمليات (Process Control): مثل إنشاء/إنهاء العمليات، تحميل وتلييس البرامج، وإدارة الأخطاء. 2. إدارة الملفات (File Management): إنشاء، حذف، فتح، قراءة، وكتابة الملفات. 3. إدارة الأجهزة (Device Management): طلب أو تحرير الأجهزة، وقراءة/كتابة البيانات عليها. 4. صيانة المعلومات (Information Maintenance): جلب أو ضبط الوقت، التاريخ، والبيانات والخصائص. 5. الاتصالات (Communications): إنشاء وحذف روابط الاتصال وإرسال واستقبال الرسائل. 6. الحماية (Protection): التحكم في الوصول للموارد والصلاحيات. 3. برامج النظام (System Programs) توفر بيئة ملائمة لتطوير وتفقد وتنفيذ البرامج، وغالباً ما تكون هي النظرة الشائعة للمستخدم عن نظام التشغيل بدلاً من استدعاءات النظام المباشرة. إدارة الملفات (File Management): إنشاء، نسخ، تعديل، وعرض الملفات والمجلدات. معلومات الحالة (Status Information): عرض التاريخ، الوقت، الذاكرة المتاحة، أو استخدام الـ Registry لتخزين الإعدادات. تعديل الملفات (File Modification): محرر النصوص والأوامر الخاصة بالبحث والتعديل. دعم لغات البرمجة: المترجمات (Compilers)، المجمعات (Assemblers)، والمصححات (Debuggers). تحميل وتنفيذ البرامج: برامج التحميل (Loaders) ومحرري الروابط (Linkers). الخدمات الخلفية (Background Services / Daemons): برامج تعمل في الخلفية منذ الإقلاع لتقديم خدمات مثل جدولة العمليات وفحص الأقراص. 4. تصميم وتنفيذ نظم التشغيل (OS Design and Implementation) الأهداف: تحديد أهداف النظام (أهداف المستخدم لتكون سهلة وسريعة، وأهداف النظام لتكون سهلة الصيانة ومرنة). فصل السياسة عن الآلية (Policy vs). السياسة: تحديد "ماذا سيفعل؟" (What will be done?). الآلية: تحديد "كيف سيتم فعله؟" (How to do it?). (فصلهم يمنح مرونة عالية للنظام). لغات التنفيذ: تبدأ الأنظمة عادة بلغة التجميع (Assembly) ثم لغات مثل C و ++C، وغالباً ما تستخدم مزيجاً من اللغات لتحقيق السرعة والكفاءة وسهولة النقل للأجهزة المختلفة. 1. مفهوم العمليات (Process Concept) التعريف: العملية هي برنامج قيد التنفيذ (Program in Execution)، وتعتبر أساس كل عمليات الحاسوب في نظام التشغيل. البرنامج (Program): كيان خامل (Passive entity) مخزن على القرص الصلب (ملف تنفيذي). العملية (Process): كيان نشط (Active entity) يتم إنشاؤه عند تحميل البرنامج إلى الذاكرة وتنفيذه. النص (Text section): كود البرنامج. العداد والمسجلات (Program Counter & CPU Registers): لتتبع النشاط الحالي. المكس (Stack): يحتوي على البيانات المؤقتة (مثل معلمات الدوال، عناوين العودة، والمتغيرات المحلية). قسم البيانات (Data section): يحتوي على المتغيرات العامة (Global variables). الكومة (Heap): الذاكرة المخصصة ديناميكياً أثناء وقت التشغيل (2). Run time. حالات العملية (Process States) جديدة (New): العملية قيد الإنشاء. قيد التنفيذ (Running): يتم تنفيذ تعليمات العملية بواسطة المعالج. منتظرة (Waiting): تنتظر العملية حدوث حدث معين (مثل اكتمال إدخال/إخراج). جاهزة (Ready): تنتظر العملية لتُعيّن وتُخصص لمعالج. منتهية (Terminated): انتهت العملية من التنفيذ. 3. كتلة تحكم العملية (Process Control Block - PCB) هو هيكل بيانات في نظام التشغيل يمثل كل عملية، حالة العملية (Process state). رقم العملية (/ Process number). PID). عداد البرنامج (Program counter). مسجلات المعالج (CPU registers) ومعلومات الجدولة وإدارة الذاكرة. 4. جدولة العمليات (Process Scheduling) الهدف: تعظيم استفادة المعالج (CPU utilization) والتبديل السريع بين العمليات لتقاسم الوقت (Time sharing). طوابير الجدولة (Scheduling Queues): طابور الوظائف (Job queue): يحتوي على جميع العمليات في النظام. طابور الجاهزية (Ready queue): العمليات المتواجدة في الذاكرة الرئيسية وجاهزة للتنفيذ. طوابير الأجهزة (Device queues): العمليات المنتظرة لأجهزة الإدخال والإخراج. المجدول قصير الأجل (/ Short-term scheduler). المجدول طويل الأجل (Long-term scheduler / Job scheduler): يختار العمليات الجاهزة لتنفيذها بالمعالج (يُستدعى بشكل متكرر وسريع جداً بالمللي ثانية). المجدول قصير الأجل (Short-term scheduler / Job scheduler): يختار العمليات الجاهزة لتنفيذها بالمعالج (يُستدعى ببطء أكثر، بالثواني أو الدقائق). I/O-bound process: تقضي وقتاً في عمليات الإدخال والإخراج أكثر من الحسابات ولها نوبات قصيرة بالمعالج. CPU-bound process: تقضي وقتاً طويلاً في العمليات الحسابية والمعالجة. 5. إنشاء العمليات والعمليات المتزامنة (Process Creation & Concurrency) الإنشاء: تقوم العملية الرئيسية (Parent process) بإنشاء عمليات فرعية (Child

(processes)، لتشكل شجرة من العمليات يتم تعريفها عبر معرف العملية (PID). خيارات مشاركة الموارد: تتشارك العمليات الأب والأبناء إما في جميع الموارد، أو جزء منها، أو لا تتشارك شيئاً. خيارات التنفيذ: قد تنفذ العمليات بالتزامن (Concurrently)، أو ينتظر الأب حتى تنتهي العمليات الفرعية. 6. الاتصال بين العمليات (Interprocess Communication – IPC) العمليات قد تكون مستقلة أو متعاونة، ويتم عملية الاتصال والتبادل بينها إما عن طريق الذاكرة المشتركة (Shared memory) أو تمرير الرسائل (1). Message passing. مفهوم الخيوط (Thread Concept) التعريف: الخيط (Thread) هو الوحدة الأساسية لاستخدام المعالج (CPU utilization)، ويتكون من عداد برنامج، ومجموعة مسجلات، ومكدس (Stack). العلاقة بالعملية: الخيط يقع داخل العملية، وتدعم المعالجات الحديثة تعدد الخيوط (Multithreading) داخل العملية الواحدة لتنفيذ مهام متعددة في نفس الوقت (مثل تحديث الشاشة، جلب البيانات، والتدقيق الإملائي في نفس التطبيق). 2. فوائد تعدد الخيوط (Benefits of Multithreading) الاستجابة (Responsiveness): السماح بتمرير التنفيذ حتى لو كان جزء من العملية محظوراً أو مشغولاً. مشاركة الموارد (Resource Sharing): تتشارك الخيوط موارد العملية التابعة لها (الذاكرة والملفات) بسهولة أكبر مقارنة بمشاركة الذاكرة أو تمرير الرسائل. الاقتصاد (Economy): إنشاء الخيوط وتبديل السياق (Context switching) أسرع وأقل تكلفة من إنشاء العمليات. قابلية التوسع (Scalability): استغلال بني المعالجات متعددة النوى (Multiprocessor architectures) بشكل أفضل. 3. البرمجة متعددة النوى (Multicore Programming) تفرض الأنظمة متعددة النوى تحديات في البرمجة مثل تقسيم الأنشطة، التوازن، تداخل البيانات (Data dependency)، واختبار وتصحيح الأخطاء. أنواع التوازي (Types of Parallelism): توازي البيانات (Data parallelism): توزع أجزاء من نفس البيانات على عدة أنوية مع تنفيذ نفس العملية عليها. توازي المهام (Task parallelism): توزع الخيوط عبر النوى، بحيث يقوم كل خيط بتنفيذ عملية فريدة. 4. خيوط المستخدم وخيوط النواة (User Threads and Kernel Threads) خيوط المستخدم (User Threads) تتم إدارتها بواسطة مكتبات خيوط على مستوى المستخدم (User-level threads library)، وتتميز بسرعة الإنشاء ولكن قد تؤدي حجب خيط واحد إلى حجب العملية كلها. خيوط النواة (Kernel Threads): تدعمها وتديرها مباشرة نواة نظام التشغيل (مثل أنظمة ويندوز، لينكس، وماك). 5. نماذج تعدد الخيوط (Multithreading Models) نموذج (Many-to-One): ربط العديد من خيوط المستخدم بخيط نواة واحد (خيط واحد محجوب يحجب الجميع). نموذج (One-to-One): ربط كل خيط مستخدم بخيط نواة مقابل، مما يوفر توازياً أكبر ولكن بحد أقصى لعدد الخيوط لتجنب عبء النظام (مثل ويندوز وليناكس). نموذج (Many-to-Many): السماح بتعدد خيوط المستخدم بالارتباط بعدد مناسب أو أقل من خيوط النواة. 6. مكتبات الخيوط وقضايا متعلقة بها (Thread Libraries & Issues) مكتبات الخيوط: توفر للمبرمج واجهة برمجة تطبيقات (API) لإنشاء وإدارة الخيوط، وأبرزها معيار Pthreads (المستخدم في أنظمة UNIX)، خيوط ويندوز (Windows threads)، وخيوط جافا (Java threads). أبرز القضايا والمشكلات (Threading Issues): دالات الأوامر `fork()` و `exec()` عند استخدام الخيوط. معالجة الإشارات (Signal Handling) المتزامنة وغير المتزامنة. إلغاء الخيط المستهدف (Thread cancellation) سواء بشكل فوري أو مؤجل. التخزين الخاص بالخيط (1). Thread-local storage – TLS. خيوط ويندوز (Windows Threads): تُنفذ واجهة برمجة التطبيقات الأساسية (Windows API) وتعتمد نموذج التعيين (One-to-One) على مستوى النواة. يحتوي كل خيط على معرف خاص (Thread ID)، مجموعة مسجلات، مكدسات منفصلة لبيئة المستخدم والنواة، ومنطقة تخزين بيانات خاصة (TLS). تستخدم هياكل بيانات رئيسية في النواة مثل: `ETHREAD` (للتنفيذ)، `KTHREAD` (لنواة النظام)، و `TEB` (لبيئة الخيط). خيوط لينكس (Linux Threads): لا تفرق لينكس بشكل كبير بين العمليات والخيوط؛ بل تشير إليها عادة باسم Tasks (مهام). يتم إنشاء الخيوط أو المهام باستخدام استدعاء النظام `clone()`، والذي يسمح بتحديد مشاركة مساحة الذاكرة والبيانات بين العملية الأب والمهام الفرعية عبر مؤشرات (مثل `struct task_struct` 1). خيوط ويندوز (Windows Threads): تُنفذ واجهة برمجة التطبيقات الأساسية (Windows API) وتعتمد نموذج التعيين (One-to-One) على مستوى النواة. يحتوي كل خيط على معرف خاص (Thread ID)، مجموعة مسجلات، مكدسات منفصلة لبيئة المستخدم والنواة، ومنطقة تخزين بيانات خاصة (TLS). تستخدم هياكل بيانات رئيسية في النواة مثل: `ETHREAD` (للتنفيذ)، `KTHREAD` (لنواة النظام)، و `TEB` (لبيئة الخيط). خيوط لينكس (Linux Threads): لا تفرق لينكس بشكل كبير بين العمليات والخيوط؛ بل تشير إليها عادة باسم Tasks (مهام). يتم إنشاء الخيوط أو المهام باستخدام استدعاء النظام `clone()`، والذي يسمح بتحديد

مستوى مشاركة مساحة الذاكرة والبيانات بين العملية الأب والمهام الفرعية عبر مؤشرات (مثل struct task_struct).