CSS222 - Summary

مقدمة الملف

أهلًا فيكم !

أتمنى أن يكون هذا الملف مساعدًا لك في فهم الدروس بسهولة ويسر.

أسأل الله أن يوفقك في كل خطوة، وييسر لك الصعوبات، ويجعل العلم نورًا يهديك في حياتك

@Nubgha

CSS222 - Summary

Operating System Services

Operating systems provide an environment for executing programs and offer services to both users and programs

.1.Services Helpful to the User

User Interface (UI):
Most operating systems provide a user interface that allows users to interact with the system. It may be a Command-Line Interface (CLI), Graphical User Interface (GUI), or Batch Processing, which runs a series of programs automatically without human interaction.

Program Execution:
The operating system loads programs into memory, runs them, and terminates their execution either normally or due to an error.

I/O Operations:
Running programs may require input/output operations involving files or I/O devices.

File-System Manipulation:
Programs can create, delete, read, write, and search files and directories, as well as manage file information and permissions.

Communications:
Processes can exchange information within the same computer or across networks using shared memory or message passing.

Error Detection:
The operating system detects errors that may occur in the CPU, memory, I/O devices, or user programs and takes appropriate actions to maintain correct system operation.

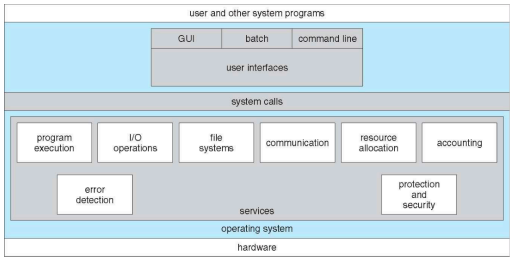
2.Services for Efficient System Operation

Resource Allocation:
The operating system allocates resources such as CPU time, memory, file storage, and I/O devices among multiple users or programs running at the same time.

Accounting:
The system keeps track of how much and what types of computer resources are used by each user.

Protection and Security:
Protection ensures controlled access to system resources to prevent interference between processes. Security protects the system from unauthorized access through user authentication and safeguards against invalid access attempts.

A View of Operating System Services



خدمات نظام التشغيل

يوفّر نظام التشغيل بيئة لتشغيل البرامج ويقدّم خدمات لكلٍ من المستخدمين والبرامج.

1. الخدمات المفيدة للمستخدم

واجهة المستخدم:
توفر معظم أنظمة التشغيل واجهة تتيح للمستخدم التفاعل مع النظام. قد تكون هذه الواجهة **واجهة سطر الأوامر** أو **واجهة رسومية** أو **المعالجة الدفعية** التي تقوم بتشغيل مجموعة من البرامج تلقائيًا دون تدخل بشري.

تنفيذ البرامج:
يقوم نظام التشغيل بتحميل البرامج إلى الذاكرة وتشغيلها ثم إنهاء تنفيذها، سواء كان الإنهاء طبيعيًا أو بسبب حدوث خطأ.

عمليات الإدخال والإخراج:
قد تحتاج البرامج أثناء تشغيلها إلى عمليات إدخال وإخراج، والتي قد تتضمن التعامل مع الملفات أو أجهزة الإدخال والإخراج.

التعامل مع نظام الملفات:
تستطيع البرامج إنشاء الملفات والمجلدات وحذفها وقراءتها وكتابتها والبحث عنها، بالإضافة إلى عرض معلومات الملفات وإدارة الصلاحيات.

الاتصالات:
يمكن للعمليات تبادل المعلومات داخل الحاسوب نفسه أو بين حواسيب مختلفة عبر الشبكة، ويتم ذلك باستخدام الذاكرة المشتركة أو تمرير الرسائل.

اكتشاف الأخطاء:
يقوم نظام التشغيل باكتشاف الأخطاء التي قد تحدث في المعالج أو الذاكرة أو أجهزة الإدخال والإخراج أو في برامج المستخدم، ويتخذ الإجراءات المناسبة لضمان عمل النظام بشكل صحيح.

2. خدمات لضمان كفاءة تشغيل النظام

تخصيص الموارد:
يقوم نظام التشغيل بتوزيع موارد النظام مثل وقت المعالج والذاكرة والتخزين وأجهزة الإدخال والإخراج بين عدة مستخدمين أو برامج تعمل في الوقت نفسه.

المحاسبة:
يقوم النظام بتتبع مقدار وأنواع الموارد الحاسوبية التي يستخدمها كل مستخدم.

الحماية والأمان:
تعني الحماية التحكم في الوصول إلى موارد النظام لمنع تعارض العمليات مع بعضها. أما الأمان فيهدف إلى حماية النظام من الوصول غير المصرح به من خلال التحقق من هوية المستخدمين ومنع محاولات الوصول غير الصحيحة.

CSS222 - Summary

User Operating System Interface – Command Line Interface

Command Line Interface:

The Command Line Interface allows users to enter commands directly. It may be implemented in the kernel or by a system program. Sometimes there are multiple versions called shells. Its main function is to fetch a command from the user and execute it. Some commands are built-in, while others are simply program names, allowing new features to be added without modifying the shell.

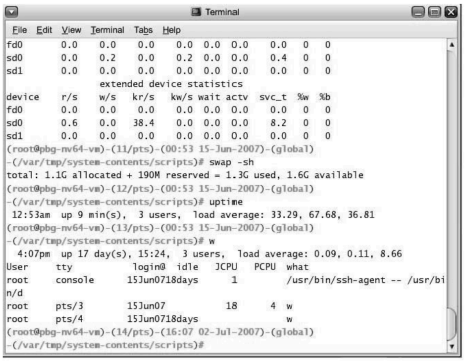
Graphical User Interface:

This interface usually uses a mouse, keyboard, and monitor. Icons represent files, programs, and actions. Different mouse clicks on interface objects can perform actions such as displaying information, showing options, executing functions, or opening a directory (folder). This interface was invented at Xerox PARC.

Using Both Interfaces:

Many systems now include both CLI and GUI. For example, Microsoft Windows mainly uses a GUI but also provides a command shell. Apple Mac OS X has the Aqua GUI with a UNIX kernel and available shells. Solaris mainly uses CLI with optional GUI interfaces.

Bourne Shell Command Interpreter



واجهة المستخدم في نظام التشغيل – واجهة سطر الأوامر

واجهة سطر الأوامر:

تسمح واجهة سطر الأوامر للمستخدم بإدخال الأوامر مباشرة. قد يتم تنفيذها أحيانًا داخل النواة وأحيانًا بواسطة برنامج نظام. وقد توجد عدة أنواع منها تُعرف بالقواقع. وظيفتها الأساسية هي جلب الأمر من المستخدم وتنفيذه. بعض الأوامر تكون مدمجة داخل الواجهة، بينما يكون بعضها الآخر مجرد أسماء لبرامج، مما يسمح بإضافة ميزات جديدة دون الحاجة إلى تعديل القوقعة

.

الواجهة الرسومية للمستخدم:

تعتمد عادةً على الفأرة ولوحة المفاتيح والشاشة. تمثل الأيقونات الملفات والبرامج والإجراءات المختلفة. ويمكن لأزرار الفأرة المختلفة عند النقر على العناصر في الواجهة تنفيذ عدة إجراءات مثل عرض المعلومات أو إظهار الخيارات أو تنفيذ وظيفة أو فتح مجلد. تم اختراع هذا النوع من الواجهات في شركة زيروكس

.

استخدام الواجهتين معًا:

تدعم العديد من الأنظمة حاليًا كلاً من واجهة سطر الأوامر والواجهة الرسومية.
فنظام مايكروسوفت ويندوز يعتمد أساسًا على الواجهة الرسومية مع وجود واجهة سطر أوامر.
أما نظام ماك فيوفر واجهة رسومية مع نواة يونكس وإمكانية استخدام القواقع.
كما أن نظام سولاريس يعتمد أساسًا على واجهة سطر الأوامر مع إمكانية إضافة واجهة رسومية.

CSS222 - Summary

The Mac OS X GUI



System Calls

System calls provide the programming interface to the services offered by the operating system. They are typically written in high-level languages such as C or C++. Programs usually access these services through an **Application Programming Interface (API)** rather than calling system calls directly.

The most common APIs are:

- **Win32 API** for Windows
- **POSIX API** for POSIX-based systems such as UNIX, Linux, and macOS
- **Java API** for the Java Virtual Machine (JVM)

Why APIs are used instead of direct system calls:

1. System calls differ between platforms, while APIs provide a stable interface that makes software easier to move across systems.
2. When operating systems update system calls, APIs are usually updated to support these improvements.
3. APIs often provide more useful functionality than direct system calls.
4. APIs can support multiple operating system versions and choose the appropriate one during execution.

استدعاءات النظام

تمثل استدعاءات النظام واجهة البرمجة للخدمات التي يوفرها نظام التشغيل. غالبًا تُكتب باستخدام لغات برمجة عالية المستوى مثل **سي أو سي بلس بلس**. عادةً تصل البرامج إلى هذه الخدمات من خلال **واجهة برمجة التطبيقات** بدلاً من استخدام استدعاءات النظام مباشرة.

أشهر واجهات برمجة التطبيقات هي:

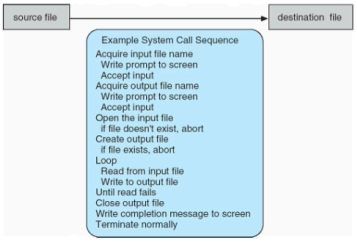
- **واجهة ويندوز 32** لنظام ويندوز
- **واجهة بوزيكس** للأنظمة المعتمدة على بوزيكس مثل يونكس ولينكس وماك
- **واجهة جافا** الخاصة بآلة جافا الافتراضية

أسباب استخدام واجهات برمجة التطبيقات بدل استدعاءات النظام مباشرة:

1. استدعاءات النظام تختلف بين الأنظمة، بينما توفر واجهات البرمجة واجهة ثابتة تسهل نقل البرامج بين الأنظمة.
2. عند تحديث استدعاءات النظام، يتم عادة تحديث واجهة البرمجة لدعم هذه التحسينات.
3. غالبًا توفر واجهات البرمجة وظائف أكثر فائدة من استدعاء النظام مباشرة.
4. يمكن لواجهة البرمجة دعم عدة إصدارات من نظام التشغيل واختيار الإصدار المناسب أثناء التنفيذ.

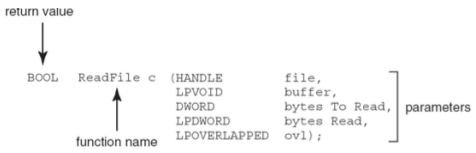
CSS222 - Summary

Example of System Calls



Example of Standard API

- Consider the ReadFile() function in the
- Win32 API—a function for reading from a file



ReadFile() Parameters

HANDLE file: The file to read.
LPVOID buffer: Memory buffer where data will be stored.
DWORD bytesToRead: Number of bytes to read into the buffer.
LPDWORD bytesRead: Number of bytes actually read.
LPOVERLAPPED ovl: Indicates whether overlapped (asynchronous) I/O is used.

System Call Implementation

Each system call is usually associated with a unique number, maintained in a system-call table.
The system-call interface invokes the corresponding system call in the OS kernel and returns its status and any results.
The programmer does not need to know the internal implementation, only the API and expected OS behavior.
Most OS interface details are hidden by the API, managed via runtime support libraries provided with the compiler.

معاملات دالة ReadFile()

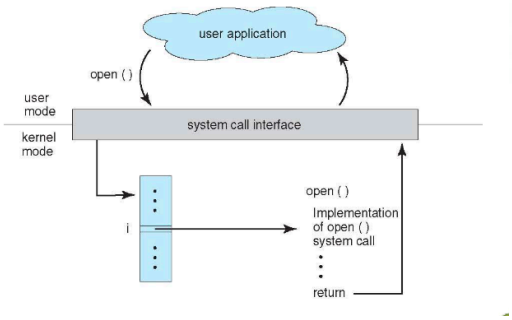
- **HANDLE file:** الملف الذي سيتم قراءته.
- **LPVOID buffer:** المخزن المؤقت الذي سيتم تخزين البيانات فيه.
- **DWORD bytesToRead:** عدد البايتات المطلوب قراءتها في المخزن المؤقت.
- **LPDWORD bytesRead:** عدد البايتات المقروءة فعليًا.
- **LPOVERLAPPED ovl:** يشير إلى ما إذا كان يتم استخدام إدخال/إخراج متداخل (غير متزامن).

تنفيذ استدعاءات النظام

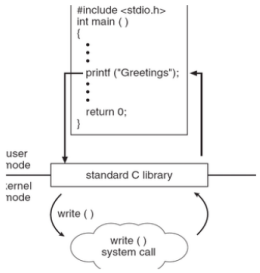
- يرتبط كل استدعاء نظام عادةً برقم فريد محفوظ في جدول استدعاءات النظام.
- تقوم واجهة استدعاءات النظام باستدعاء الاستدعاء المناسب داخل نواة نظام التشغيل وإرجاع حالته وأي قيم نتيجة.
- لا يحتاج المبرمج لمعرفة تفاصيل التنفيذ الداخلية، فقط الالتزام بالواجهة وفهم سلوك النظام المتوقع.
- معظم تفاصيل واجهة النظام مخفية بواسطة واجهة البرمجة، وتدار من خلال مكتبات دعم وقت التشغيل المدمجة مع المترجم.

CSS222 - Summary

API – System Call – OS Relationship



Standard C Library Example



System Call Parameter Passing

System calls often require more information than just the call's identity. The type and amount of information depend on the operating system and the specific call.

Three common methods to pass parameters to the OS:

1. **Registers:** Parameters are passed directly in CPU registers.
    - Limitation: sometimes there are more parameters than available registers.
2. **Memory Block/Table:** Parameters are stored in a memory block, and the block's address is passed in a register.
    - Used by systems like Linux and Solaris.
3. **Stack:** Parameters are pushed onto the program stack and popped by the OS.

- Using memory blocks or the stack allows passing any number or length of parameters without limitation.

تمرير معاملات استدعاء النظام

غالبًا ما تحتاج استدعاءات النظام إلى معلومات أكثر من مجرد تحديد هوية الاستدعاء. يختلف نوع وكمية المعلومات بحسب نظام التشغيل والاستدعاء المحدد.

ثلاث طرق شائعة لتمرير المعاملات إلى نظام التشغيل:

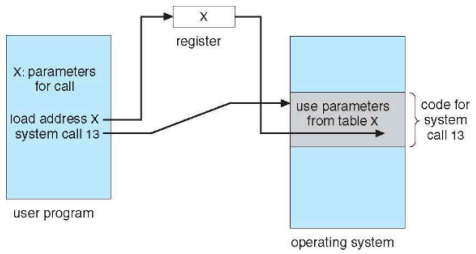
السجلات *(Registers)*: يتم تمرير المعاملات مباشرة عبر سجلات المعالج.
القيد: أحيانًا يكون عدد المعاملات أكبر من عدد السجلات المتاحة.

كتلة/جدول في الذاكرة: تُخزن المعاملات في كتلة بالذاكرة، ويتم تمرير عنوان الكتلة في سجل.
تُستخدم هذه الطريقة في أنظمة مثل لينكس وسولاريس.

المكدس *(Stack)*: يقوم البرنامج بدفع المعاملات إلى المكدس ويقوم نظام التشغيل بسحبها عند التنفيذ.
طرق الكتلة والمكدس لا تحد من عدد أو طول المعاملات الممررة.

CSS222 - Summary

Parameter Passing via Table



Types of System Calls
Process control
end, abort
load, execute
create process, terminate process
get process attributes, set process attributes
wait for time
wait event, signal event
allocate and free memory
File management
create file, delete file
open, close file
read, write, reposition
get and set file attributes
Device management
request device, release device
read, write, reposition
get device attributes, set device attributes
logically attach or detach devices
Information maintenance
get time or date, set time or date
get system data, set system data
get and set process, file, or device attributes
Communications
create, delete communication connection
send, receive messages
transfer status information
attach and detach remote devices

التحكم في العمليات
إنهاء، إلغاء
تحميل، تنفيذ
إنشاء عملية، إنهاء عملية
الحصول على خصائص العملية، ضبط خصائص العملية
الانتظار لمدة
انتظار حدث، إرسال إشارة للحدث
تخصيص وتحرير الذاكرة
إدارة الملفات
إنشاء ملف، حذف ملف
فتح، إغلاق ملف
قراءة، كتابة، إعادة تحديد موضع
الحصول على خصائص الملف، ضبط خصائص الملف
إدارة الأجهزة
طلب جهاز، تحرير جهاز
قراءة، كتابة، إعادة تحديد موضع
الحصول على خصائص الجهاز، ضبط خصائص الجهاز
توصيل أو فصل الأجهزة منطقيًا
صيانة المعلومات
الحصول على الوقت أو التاريخ، ضبط الوقت أو التاريخ
الحصول على بيانات النظام، ضبط بيانات النظام
الحصول على وضبط خصائص العمليات أو الملفات أو الأجهزة
الاتصالات
إنشاء، حذف اتصال اتصالات
إرسال، استقبال الرسائل
نقل معلومات الحالة
توصيل وفصل الأجهزة عن بعد

CSS222 - Summary

Examples of Windows and Unix System Calls

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

System Programs

System programs provide a convenient environment for program development and execution. They include:

- **File Manipulation:** create, delete, copy, rename, print, dump, list files and directories.
- **Status Information:** show system info (date, time, memory, disk space, users) and performance/debugging data; may use a system registry.
- **File Modification:** text editors, search and transform file contents.
- **Programming Language Support:** compilers, assemblers, debuggers, interpreters.
- **Program Loading and Execution:** loaders, linkage editors, overlay loaders, debugging systems.
- **Communications:** create connections between processes/ users/systems, send messages, browse web, email, remote login, file transfer.
- **Application Programs:** interface for user-level applications.

Most users interact with the operating system through **system programs**, not directly through system calls.

برامج النظام

توفر **برامج النظام** بيئة مريحة لتطوير وتشغيل البرامج، وتشمل:

- **إدارة الملفات:** إنشاء، حذف، نسخ، إعادة تسمية، طباعة، عرض، والتعامل مع الملفات والمجلدات.
- **معلومات النظام:** عرض معلومات النظام (الوقت، التاريخ، الذاكرة، مساحة القرص، عدد المستخدمين) وأداء النظام والتصحيح؛ قد يستخدم بعض الأنظمة سجلاً لتخزين الإعدادات.
- **تعديل الملفات:** محررات نصية، أوامر للبحث أو تعديل محتوى الملفات.
- **دعم لغات البرمجة:** مترجمات، مجمّعات، مصححات، ومفسرات.
- **تحميل وتنفيذ البرامج:** محملات مطلقة أو قابلة لإعادة التوطين، محررات الربط، محملات متراكبة، وأنظمة تصحيح للغات عالية أو لغة الآلة.
- **الاتصالات:** إنشاء اتصالات بين العمليات أو المستخدمين أو الأنظمة، إرسال رسائل، تصفح الإنترنت، البريد الإلكتروني، تسجيل دخول عن بعد، ونقل الملفات.
- **برامج التطبيقات:** واجهة للتطبيقات على مستوى المستخدم.

يتفاعل معظم المستخدمين مع نظام التشغيل عبر **برامج النظام** وليس من خلال استدعاءات النظام مباشرة.

CSS222 - Summary

Operating System Design and Implementation

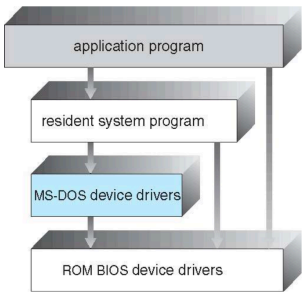
Designing and implementing an OS is not fully "solvable," but certain approaches are effective.

- **Internal structure** varies widely between systems.
- **Start by defining goals and specifications**, influenced by hardware and system type.
- **Goals:**
  - **User goals:** convenient, easy to learn, reliable, safe, fast.
  - **System goals:** easy to design, implement, maintain; flexible, reliable, error-free, efficient.
- **Key principle:** separate **policy** (what to do) from **mechanism** (how to do it) to allow flexibility in changing policies later.

Simple Structure Example:

- **MS-DOS:** designed to provide maximum functionality in minimal space.
  - Not modular; interfaces and levels of functionality are not well separated.
  -

MS-DOS Layer Structure



Layered Approach –

The OS is divided into multiple layers, each built on top of the lower ones.

- **Layer 0:** hardware; **Layer N:** user interface.
- **Modularity:** each layer uses only the functions and services of lower layers, ensuring independence.
-

تصميم وتنفيذ نظم التشغيل

تصميم وتنفيذ نظام التشغيل ليس مسألة قابلة للحل بالكامل، لكن بعض الطرق أثبتت نجاحها.

- **الهيكل الداخلي** يختلف بشكل كبير بين الأنظمة.
- **يبدأ بتحديد الأهداف والمواصفات**، متأثرًا بنوع العتاد والنظام.
- **الأهداف:**
  - **أهداف المستخدم:** أن يكون النظام سهل الاستخدام، سهل التعلم، موثوق، آمن، وسريع.
  - **أهداف النظام:** أن يكون سهل التصميم والتنفيذ والصيانة، ومرن، موثوق، خالٍ من الأخطاء، وفعال.
- **المبدأ المهم:** فصل **السياسة** (ما الذي سيتم فعله) عن **الآلية** (كيف يتم فعله)، مما يوفر مرونة لتغيير السياسات لاحقًا.

مثال على الهيكل البسيط:

- **MS-DOS:** صُمم لتوفير أكبر قدر من الوظائف بأقل مساحة.
  - غير مقسم إلى وحدات؛ الواجهات ومستويات الوظائف غير مفصولة بشكل جيد.

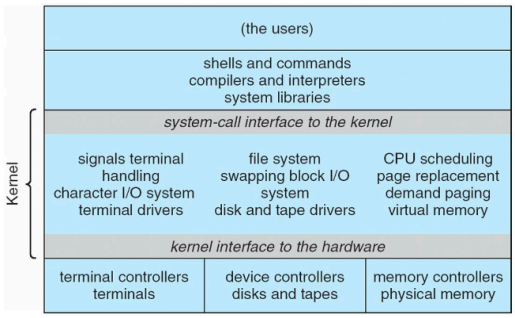
النهج الطبقي

يُقسّم نظام التشغيل إلى عدة طبقات، كل طبقة مبنية فوق الطبقة الأدنى.

- **الطبقة 0:** الأجهزة؛ **الطبقة N:** واجهة المستخدم.
- **الاستقلالية:** كل طبقة تستخدم فقط وظائف وخدمات الطبقات الأدنى، مما يضمن فصلها واستقلالها.

CSS222 - Summary

Traditional UNIX System Structure



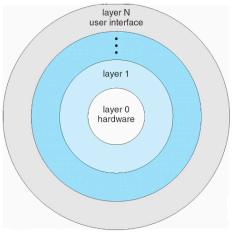
UNIX – limited by hardware functionality, the original UNIX operating
system had limited structuring. The UNIX OS consists of two separable parts

- Systems programs
- The kernel
- Consists of everything below the system-call interface and above the physical hardware
- Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level

UNIX – نظراً لقيود وظائف الأجهزة، كان نظام التشغيل *UNIX* الأصلي يتميز بهيكل محدود. يتكون نظام التشغيل *UNIX* من جزأين منفصلين

- برامج النظام
- النواة
- تتألف من كل ما يقع أسفل واجهة استدعاء النظام وفوق الأجهزة المادية
- توفر نظام الملفات، وجدولة وحدة المعالجة المركزية، وإدارة الذاكرة، ووظائف نظام التشغيل الأخرى؛ وهو عدد كبير من الوظائف بالنسبة لمستوى واحد

Layered Operating System



Microkernel System Structure

Moves as much from the kernel into "user" space

- *Communication takes place between user modules using message passing*
- *Benefits:*
- *Easier to extend a microkernel*
- *Easier to port the operating system to new architectures*
- *More reliable (less code is running in kernel mode)*
- *More secure*
- *Detriments (drawbacks):*
- *Performance overhead of user space to kernel space communication*

Microkernel System Structure

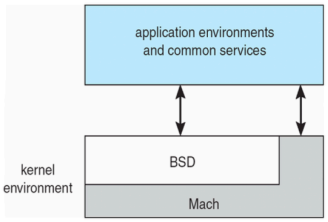
نقل أكبر قدر ممكن من النواة إلى مساحة "المستخدم"

- يتم الاتصال بين وحدات المستخدم عن طريق تمرير الرسائل

- المزايا:
- سهولة أكبر في توسيع النواة الصغيرة
- سهولة أكبر في نقل نظام التشغيل إلى بنى جديدة
- موثوقية أعلى (يتم تشغيل كمية أقل من التعليمات البرمجية في وضع النواة)
- أكثر أمانًا
- العيوب (السلبيات):
- عبء الأداء الناتج عن الاتصال بين مساحة المستخدم ومساحة النواة

CSS222 - Summary

Mac OS X Structure



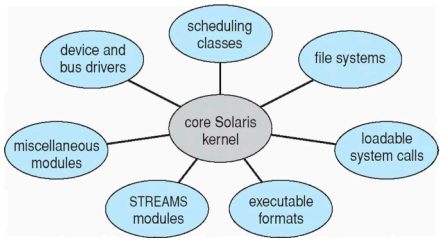
Modules

- Modern OS kernels use **modules**.
- Follows an **object-oriented approach**: each core component is separate.
- Components communicate via **known interfaces**.
- Modules can be **loaded as needed** within the kernel.
- Similar to layered structure but more **flexible**.

الوحدات في أنظمة التشغيل

- في النواة **(Modules)** تستخدم معظم أنظمة التشغيل الحديثة **وحدات**.
- تعتمد على **البرمجة الكائنية التوجه**: كل مكوّن أساسي مستقل.
- تتواصل المكونات عبر **واجهات معرفة مسبقًا**.
- يمكن **تحميل الوحدات حسب الحاجة** داخل النواة.
- مشابهة للهيكل الطبقي ولكن أكثر **مرونة**.

Solaris Modular Approach



Virtual Machines

- A **virtual machine** extends the layered approach fully.
- Treats **hardware and OS kernel** as hardware.
- Provides an interface **identical to the underlying hardware**.
- The host OS creates the illusion that each process has its **own processor and virtual memory**.
- Each guest OS or process gets a **virtual copy** of the underlying computer.

الآلات الافتراضية

- **الآلة الافتراضية** تأخذ النهج الطبقي إلى نهايته المنطقية.
- تعتبر **العتاد ونواة النظام** كما لو كانت كلها عتادًا.
- توفر واجهة **مطابقة للعتاد الأساسي**.
- يقوم نظام التشغيل المضيف بخلق وهم أن لكل عملية **معالجها وذاكرتها الافتراضية الخاصة**.
- يحصل كل ضيف على **نسخة افتراضية** من الكمبيوتر الأساسي.

CSS222 - Summary

Virtual Machines – History and Benefit

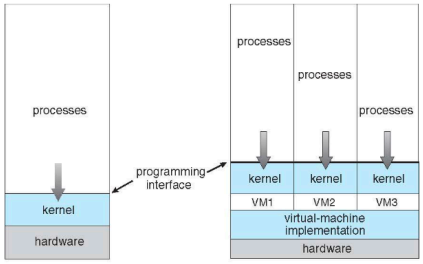
- First commercial use: **IBM mainframes, 1972**.
- Multiple execution environments (different OSs) can **share the same hardware**.
- **Isolation:** virtual machines are protected from each other.
- Controlled **file sharing** is possible.
- Can **communicate** with each other and other physical systems via networking.
- Useful for **development and testing**.
- **Consolidation:** combine many low-resource systems into fewer, busier machines.
- **Open Virtual Machine Format (OVMF):** standard format allowing a VM to run on multiple host platforms.

تاريخ وفوائد الآلات الافتراضية

الاستخدام التجاري الأول: **الحواسيب العملاقة لشركة IBM، 1972**.

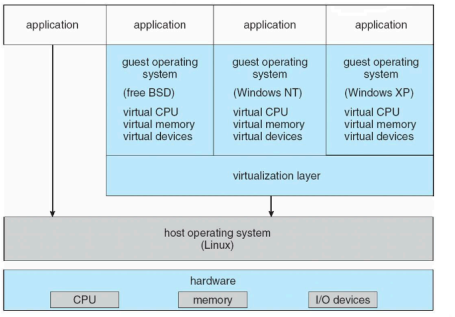
- يمكن لعدة بيئات تنفيذية (أنظمة تشغيل مختلفة) **مشاركة نفس العتاد**.
- **العزل:** تحمي الآلات الافتراضية بعضها من بعض.
- يمكن السماح بمشاركة **الملفات** بشكل محدود ومراقب.
- يمكنها **التواصل** مع بعضها البعض ومع أنظمة فيزيائية أخرى عبر الشبكات.
- مفيدة لأغراض **التطوير والاختبار**.
- **التوحيد (Consolidation):** دمج العديد من الأنظمة منخفضة الاستخدام في عدد أقل من الأجهزة النشطة.
- **تنسيق الآلة الافتراضية المفتوح (OVMF):** تنسيق قياسي يسمح بتشغيل الآلة الافتراضية على عدة منصات مضيفة.

Virtual Machines

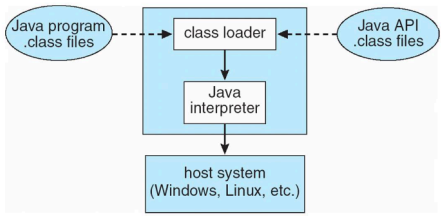


(a) Nonvirtual machine (b) virtual machine

VMware Architecture



The Java Virtual Machine



CSS222 - Summary

Operating-System Debugging and Booting

Debugging:

- Finding and fixing errors (bugs) in the system.
- **Log files:** OS generates logs with error information.
- **Core dump:** captures memory of a failed application.
- **Crash dump:** captures kernel memory on OS failure.
- **Performance tuning:** optimizes system performance beyond fixing crashes.
- **Kernighan's Law:** "Debugging is twice as hard as writing the code."
- Tools like **DTrace** (Solaris, FreeBSD, Mac OS X) allow live monitoring, capturing state data when code executes.

Operating System Generation:

- OS designed for a class of machines; must be configured for each specific computer.
- **SYSGEN:** program to obtain hardware configuration info.
- **Booting:** starting a computer by loading the kernel.
- **Bootstrap program:** stored in ROM, locates and loads the kernel into memory, starts execution.

System Boot:

- OS must be available to hardware to start.
- **Bootstrap loader:** small code to locate kernel, load it, and start it.
- Sometimes a **two-step process:** boot block loads bootstrap loader.
- Execution starts at a **fixed memory location** using firmware to hold initial boot code.
-

تصحيح وإقلاع نظام التشغيل

تصحيح الأخطاء (Debugging):

- إيجاد وإصلاح الأخطاء (Bugs) في النظام.
- **ملفات السجل:** تولدها أنظمة التشغيل لتحتوي على معلومات الأخطاء.
- **Core dump:** يحفظ ذاكرة التطبيق عند فشله.
- **Crash dump:** يحفظ ذاكرة النواة عند فشل النظام.
- **تحسين الأداء:** تحسين أداء النظام يتجاوز مجرد إصلاح الأعطال.
- **قانون Kernighan:** "تصحيح الأخطاء أصعب مرتين من كتابة الكود."
- أدوات مثل **DTrace** (Solaris، FreeBSD، Mac OS X) تسمح بالمراقبة الحية، تلتقط حالة النظام عند تنفيذ الكود.

توليد نظام التشغيل:

- يصمم النظام للعمل على فئة من الحواسيب؛ يجب تهيئته لكل جهاز محدد.
- **SYSGEN:** برنامج للحصول على معلومات تكوين العتاد.
- **الإقلاع (Booting):** بدء تشغيل الكمبيوتر عن طريق تحميل النواة.
- **برنامج Bootstrap:** مخزن في ROM، يحدد موقع النواة، يحملها في الذاكرة، ويبدأ تنفيذها.

إقلاع النظام:

- يجب أن يكون النظام متاحًا للعتاد لبدء التشغيل.
- **Bootstrap loader:** كود صغير لتحديد موقع النواة، تحميلها، وتشغيلها.
- أحيانًا عملية **خطوتين:** كتلة الإقلاع (boot block) تقوم بتحميل الـ Bootstrap loader.
- يبدأ التنفيذ في **موقع ذاكرة ثابت** باستخدام البرنامج الثابت (Firmware) لحفظ كود الإقلاع الأولي.