



1. Basic Concepts

Giuseppe Anastasi
g.anastasi@iet.unipi.it
Pervasive Computing & Networking Lab. (PerLab)
Dept. of Information Engineering, University of Pisa



Based on original slides by Silberschatz, Galvin and Gagne

Overview

- Preliminary Concepts
- Services
- System Calls
- System Programs
- Internal Structure
- System Boot

Basic Concepts 2 Operating Systems

Overview

- **Preliminary Concepts**
- Services
- System Calls
- System Programs
- Internal Structure
- System Boot

Basic Concepts 3 Operating Systems



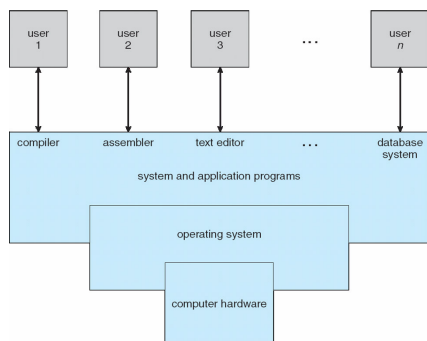
What is an Operating System?



- A program that acts as an intermediary between a user of a computer and the computer hardware.
- Operating system goals:
 - Provide an environment for executing user programs and making solving user problems easier.
 - Make the computer system convenient to use.
 - Use the computer hardware in an efficient manner.

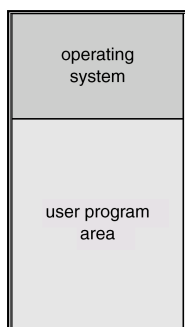


Abstract View

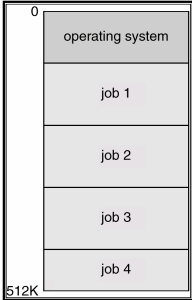




Mono-programmed Systems



Multi-programmed Systems



The diagram shows a vertical stack of memory blocks. At the top is the 'operating system' block. Below it are four blocks labeled 'job 1', 'job 2', 'job 3', and 'job 4'. The address '0' is marked at the top of the stack, and '512K' is marked at the bottom. The PerLab logo is in the top right corner.

Basic Concepts 7 Operating Systems

Features for Multiprogramming



- **Memory management**
 - the system must allocate the memory to several jobs.
- **CPU scheduling**
 - the system must choose among several jobs ready to run.
- **Device management**
 - Allocation of devices to concurrent processes

Basic Concepts 8 Operating Systems

System Classification (1)



- **Batch systems**
 - No interaction with the user
- **Interactive systems**
 - The user interacts with the systems during process execution
 - Response times should be short
- **Real time systems**
 - Soft real-time systems
 - Hard real-time systems
- **General-purpose systems**
 - The system has to manage a mix of batch, interactive and soft real-time processes

Basic Concepts 9 Operating Systems

 **System Classification (2)** 

- Mainframe systems
- Desktop systems
- Server systems
- Parallel systems
- Distributed systems
- Cluster systems
- Embedded systems
- Hand-held systems



Basic Concepts 10 Operating Systems

 **Goals of an Operating System** 

- Make the computer easy to use (e.g. PC)
- Optimize the computational resources (e.g. mainframe)
- Optimize shared resources (e.g., distributed systems)
- Make the computer easy to use and optimize energetic resources (e.g., handheld computers)
- ...



Convenience vs. efficiency

Basic Concepts 11 Operating Systems

 **Overview** 



- Preliminary Concepts
- **Services**
- System Calls
- System Programs
- Internal Structure
- System Boot

Basic Concepts 12 Operating Systems

 **Operating System Services** 



- **User Interface**
 - ▶ Command-Line (CLI)
 - ▶ Graphics User Interface (GUI)
 - ▶ Batch
- **Program execution**
 - ▶ system capability to load a program into memory and to run it.
- **I/O operations**
 - ▶ since user programs cannot execute I/O operations directly, the operating system must provide some means to perform I/O.
- **File-system manipulation**
 - ▶ program capability to read, write, create, and delete files.

Basic Concepts 13 Operating Systems

 **Operating System Services** 

- **Communications**
 - exchange of information between processes on the same computer or on different systems tied together by a network.
 - Implemented via *shared memory* or *message passing*.
- **Error detection**
 - ensure correct computing by detecting errors in the CPU and memory hardware, in I/O devices, or in user programs.

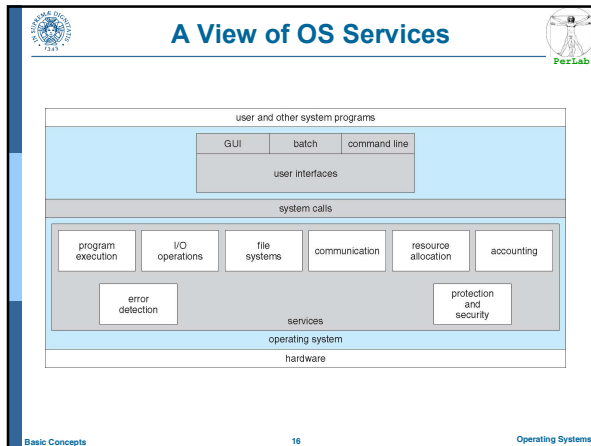
Basic Concepts 14 Operating Systems

 **Operating System Services** 

Additional functions exist not for helping the user, but rather for ensuring efficient system operations.



- **Resource allocation**
 - Many types of resources - Some (such as CPU cycles, main memory, and file storage) may have special allocation code, others (such as I/O devices) may have general request and release code
- **Accounting**
 - To keep track of which users use how much and what kinds of computer resources
- **Protection and security**
 - **Protection** involves ensuring that all access to system resources is controlled
 - **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts

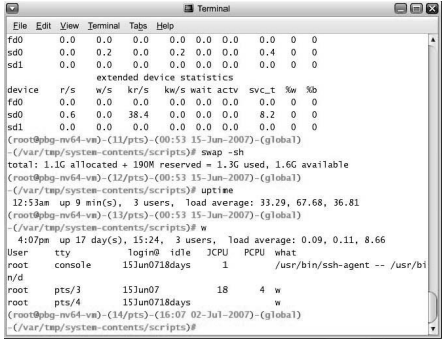
Basic Concepts 15 Operating Systems



- ## User Interface
- **Command Line Interface (CLI) – Command Interpreter**
 - Sometimes implemented in kernel, sometimes by systems program
 - ▶ Sometimes multiple flavors implemented – shells
 - Primarily fetches a command from user and executes it
 - ▶ Sometimes commands built-in, sometimes just names of programs
 - If the latter, adding new features doesn't require shell modification

- ## Graphic User Interface - GUI
- **User-friendly desktop metaphor interface**
 - Usually mouse, keyboard, and monitor
 - Icons represent files, programs, actions, etc
 - Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a folder))
 - Invented at Xerox PARC
 - **Many systems now include both CLI and GUI interfaces**
 - Microsoft Windows is GUI with CLI "command" shell
 - Apple Mac OS X as "Aqua" GUI interface with UNIX kernel underneath and shells available
 - Solaris is CLI with optional GUI interfaces (Java Desktop, KDE)



 **Bourne Shell Command Interpreter** 

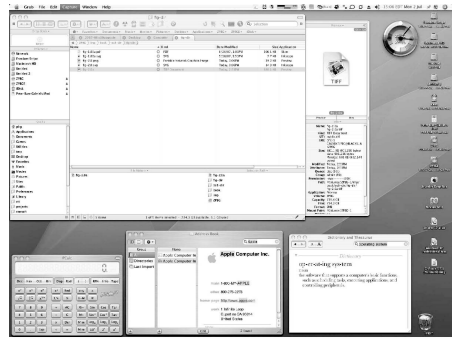


```



0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0
sdd 0.0 0.2 0.0 0.2 0.0 0.0 0.4 0 0
sdl 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
extended device statistics
device r/s w/s kr/s kw/s wait actv svc_t %w %b
fd0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
sdd 0.6 0.0 38.4 0.0 0.0 0.0 8.2 0 0
sdl 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
(root@qbg-nv64-va)-(11/pts)-(00:53 15-Jun-2007)-(global)
-/var/tmp/system-contents/scripts)# swap -sh
total: 1.1G allocated + 190M reserved = 1.3G used, 1.6G available
(root@qbg-nv64-va)-(12/pts)-(00:53 15-Jun-2007)-(global)
-/var/tmp/system-contents/scripts)# uptime
12:53am up 9 min(s), 3 users, load average: 33.29, 67.68, 36.81
(root@qbg-nv64-va)-(13/pts)-(00:53 15-Jun-2007)-(global)
-/var/tmp/system-contents/scripts)# w
4:07pm up 17 day(s), 15:24, 3 users, load average: 0.09, 0.11, 8.66
User tty login# idle JCPU PCPU what
root console 15Jun07 8days 1 /usr/bin/ssh-agent -- /usr/bi
n/d
root pts/3 15Jun07 18 4 w
root pts/4 15Jun07 8days w
(root@qbg-nv64-va)-(14/pts)-(16:07 02-Jul-2007)-(global)
-/var/tmp/system-contents/scripts)#
  
```

Basic Concepts 19 Operating Systems

 **The Mac OS X GUI** 



Basic Concepts 20 Operating Systems

 **Overview** 

- Preliminary Concepts
- Services
- **System Calls**
- System Programs
- Internal Structure
- System Boot

Basic Concepts 21 Operating Systems

System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level **Application Program Interface (API)** rather than direct system call use
 - Win32 API for Windows
 - POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X)
 - Java API for the Java virtual machine (JVM)
- Why use APIs rather than system calls?
 - **Portability**
 - **Usability** (API functions are typically easier to use than system calls)

Basic Concepts 22 Operating Systems

Example of System Calls

- System call sequence to copy the contents of one file to another file

source file

→

destination file

Example System Call Sequence

Acquire input file name
Write prompt to screen
Accept input
Acquire output file name
Write prompt to screen
Accept input
Open the input file
if file doesn't exist, abort
Create output file
if file exists, abort
Loop:
 Read from input file
 Write to output file
Until read fails,
Close output file
Write completion message to screen
Terminate normally

Basic Concepts 23 Operating Systems

Example of Standard API

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

return value

↓

BOOL

ReadFile

↑

function name

(HANDLE file, LPVOID buffer, DWORD bytesToRead, LPDWORD bytesRead, LPOVERLAPPED overl);

}

parameters

- Description of the parameters passed to ReadFile()
 - HANDLE file—the file to be read
 - LPVOID buffer—a buffer where the data will be read into and written from
 - DWORD bytesToRead—the number of bytes to be read into the buffer
 - LPDWORD bytesRead—the number of bytes read during the last read
 - LPOVERLAPPED overl—indicates if overlapped I/O is being used

Basic Concepts 24 Operating Systems

System Call Implementation

- Typically, a number associated with each system call
 - The compiler maintains a table of system calls
- The system call interface invokes intended system call in OS kernel and returns status of the system call and any return values
- The caller needs know nothing about how the system call is implemented
 - Just needs to obey API and understand what OS will do as a result call
 - Most details of OS interface are hidden from programmer by API
 - ▶ Managed by run-time support library (set of functions built into libraries included with compiler)

Basic Concepts 25 Operating Systems

API–System Call–OS Relationship

The diagram illustrates the relationship between a user application, a system call interface, and the OS kernel. A cloud labeled 'user application' is shown in 'user mode'. An arrow labeled 'open ()' points from the application to a grey box labeled 'system call interface'. This interface sits on the boundary between 'user mode' and 'kernel mode'. Below the interface, a vertical bar represents a table with an entry 'i'. An arrow points from this entry to the 'Implementation of open () system call' in the kernel. A 'return' arrow points back from the kernel implementation to the system call interface, which then returns to the user application.

Basic Concepts 26 Operating Systems

Transition from User to Kernel Mode

The diagram shows the transition between user and kernel modes. A grey box labeled 'user process' is divided into 'user process executing' and 'return from system call'. An arrow labeled 'calls system call' points from the executing part to a blue box labeled 'kernel'. Inside the kernel box, an arrow labeled 'execute system call' points from the 'trap mode bit = 0' state to the 'return mode bit = 1' state. An arrow labeled 'return from system call' points from the kernel back to the 'return from system call' part of the user process. The user mode is labeled 'mode bit = 1' and the kernel mode is labeled 'mode bit = 0'.

Basic Concepts 27 Operating Systems

Standard C Library Example

- C program invoking `printf()` library call, which calls `write()` system call

```

#include <stdio.h>
int main ()
{
    .
    .
    .
    printf ("Greetings");
    .
    .
    .
    return 0;
}
    
```

user node
kernel node

standard C library

write ()

write () system call

Basic Concepts 28 Operating Systems

System Call Parameter Passing

- Often, more information is required than simply identity of desired system call
 - Exact type and amount of information vary according to OS and call
- Three general methods used to pass parameters to the OS
 - Simplest: pass the parameters in *registers*
 - In some cases, may be more parameters than registers
 - Parameters stored in a *block*, or table, in memory, and address of block passed as a parameter in a register
 - This approach taken by Linux and Solaris
 - Parameters placed, or *pushed*, onto the *stack* by the program and *popped off* the stack by the operating system
 - Block and stack methods do not limit the number or length of parameters being passed

Basic Concepts 29 Operating Systems

Parameter Passing via Table

user program

operating system

X: parameters for call
load address X
system call 13

X
register

use parameters from table X

code for system call 13

Basic Concepts 30 Operating Systems

Types of System Calls

- Process control
- File management
- Device management
- Information maintenance
- Communications

Basic Concepts 31 Operating Systems

Process control



- create_process()
- end(), abort()
- load()
- execute()
- get_process_attribute(), set_process_attribute()
- wait(time), wait(event)
- signal(event)

Basic Concepts 32 Operating Systems

MS-DOS execution



(a) At system startup (b) running a program

Basic Concepts 33 Operating Systems

 **FreeBSD Running Multiple Programs** 



process D
free memory
process C
interpreter
process B
kernel

Basic Concepts 34 Operating Systems

 **File Manipulation** 



- create_file(), delete_process()
- open(), close()
- read(), write(),
- get_file_attributes(), set_file_attributes()

Basic Concepts 35 Operating Systems

 **Device Management** 



- request_device(), release_device()
- read(), write()
- get_device_attribute(), set_device_attribute()
- logical_attach_device(), logical_detach_device()

Basic Concepts 36 Operating Systems

 **System Calls (5)** 



- `get_time()`, `get_date()`
- `set_time()`, `set_date()`
- `get_process_attribute()`, `set_process_attribute()`
- `get_file_attribute()`, `set_file_attribute()`
- `get_device_attribute()`, `set_device_attribute()`

Basic Concepts 37 Operating Systems

 **Communications** 


- `create_connection()`,
- `delete_connection()`
- `send(msg)`, `receive(msg)`

Basic Concepts 38 Operating Systems


 **Examples of Windows/Unix System Calls** 

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

Basic Concepts 39 Operating Systems




Overview




- Preliminary Concepts
- Services
- System Calls
- **System Programs**
- Internal Structure
- System Boot

Basic Concepts 40 Operating Systems




System Programs




- System programs provide a convenient environment for program development and execution. They can be divided into:
 - File manipulation
 - Status information
 - File modification
 - Programming language support
 - Program loading and execution
 - Communications
 - Application programs
- Most users' view of the operation system is defined by system programs, not the actual system calls

Basic Concepts 41 Operating Systems





System Programs





- Provide a convenient environment for program development and execution
 - Some of them are simply user interfaces to system calls; others are considerably more complex
- File management
 - Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories
- Status information
 - Some ask the system for info - date, time, amount of available memory, disk space, number of users
 - Others provide detailed performance, logging, and debugging information
 - Typically, these programs format and print the output to the terminal or other output devices

Basic Concepts 42 Operating Systems

 **System Programs (cont'd)** 



- **File modification**
 - Text editors to create and modify files
 - Special commands to search contents of files or perform transformations of the text
- **Programming-language support**
 - Compilers, assemblers, debuggers and interpreters sometimes provided
- **Program loading and execution**
 - Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language
- **Communications**
 - Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

Basic Concepts 43 Operating Systems

 **Overview** 



- Preliminary Concepts
- Services
- System Calls
- System Programs
- **Internal Structure**
- System Boot

Basic Concepts 44 Operating Systems

 **Common System Components** 



- User Interface
- Process Manager
- Memory Manager
- File Manager
- I/O System Manager
- Secondary Memory Manager
- Networking
- Protection System

Basic Concepts 45 Operating Systems

 **Process Manager** 



- The process manager is responsible for the following activities
 - Process creation and deletion.
 - process suspension and resumption.
 - Provision of mechanisms for:
 - ▶ process synchronization
 - ▶ process communication

Basic Concepts 46 Operating Systems

 **Memory Manager** 



- Memory is a large array of words or bytes, each with its own address.
- It is a repository of quickly accessible data shared by the CPU and I/O devices.
- Main memory is a volatile storage device.
- The memory manager is responsible for the following activities
 - Keep track of which parts of memory are currently being used and by whom.
 - Allocate and deallocate memory space as needed.

Basic Concepts 47 Operating Systems

 **File Manager** 



- A file is a collection of related information defined by its creator.
- The file manager is responsible for the following activities:
 - File creation and deletion.
 - Directory creation and deletion.
 - Support of primitives for manipulating files and directories.
 - Mapping files onto secondary storage.
 - File backup on stable (nonvolatile) storage media.

Basic Concepts 48 Operating Systems

 **I/O System Manager** 



- The I/O system consists of:
 - A general device-driver interface
 - Drivers for specific hardware devices
 - A buffer-caching system

Basic Concepts 49 Operating Systems

 **Secondary-Storage Manager** 

- Main memory (*primary storage*) is volatile and too small to accommodate all data and programs permanently
- The computer system must provide *secondary storage* as a permanent storage system.
 - Typically Disks
- The operating system is responsible for the following activities in connection with disk management:
 - Free space management
 - Storage allocation
 - Disk scheduling

Basic Concepts 50 Operating Systems

 **Networking (Distributed Systems)** 

- A *distributed system* is a collection processors that do not share memory or a clock. Each processor has its own local memory.
- The processors in the system are connected through a communication network.
- Communication takes place using a *protocol*.
- A distributed system provides user access to various system resources.
- Access to a shared resource allows:
 - Computation speed-up
 - Increased data availability
 - Enhanced reliability

Basic Concepts 51 Operating Systems

Protection System

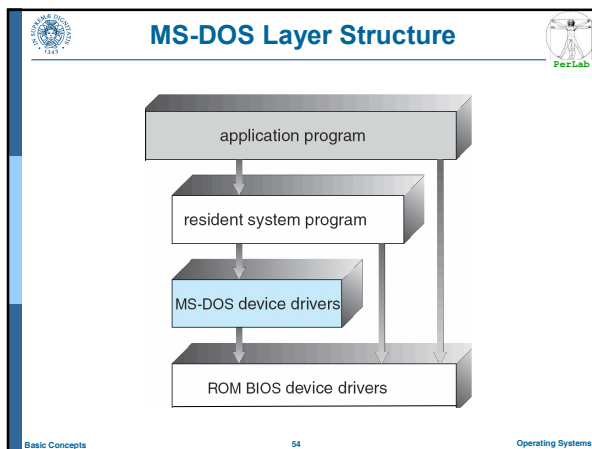
- *Protection* refers to a mechanism for controlling access by programs, processes, or users to both system and user resources.
- The protection mechanism must:
 - distinguish between authorized and unauthorized usage.
 - specify the controls to be imposed.
 - provide a means of enforcement.

Basic Concepts 52 Operating Systems

Simple Structure

- MS-DOS – written to provide the most functionality in the least space
 - Not divided into modules
 - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated

Basic Concepts 53 Operating Systems



Layered Approach

- The operating system is divided into a number of layers (levels), each built on top of lower layers
 - The bottom layer (layer 0), is the hardware
 - the highest (layer N) is the user interface.
- Each layer uses functions (operations) and services of only lower-level layers

Basic Concepts 55 Operating Systems

Traditional UNIX System Structure

(the users)			
shells and commands compilers and interpreters system libraries			
<i>system-call interface to the kernel</i>			
Kernel	signals terminal handling	file system swapping block I/O system	CPU scheduling page replacement demand paging virtual memory
	character I/O system terminal drivers	disk and tape drivers	
<i>kernel interface to the hardware</i>			
terminal controllers terminals	device controllers disks and tapes	memory controllers physical memory	

Basic Concepts 56 Operating Systems

UNIX

- 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

Basic Concepts 57 Operating Systems

Layered Operating System

The diagram illustrates a layered operating system architecture. It consists of three concentric circles. The innermost circle is labeled "layer 0 hardware". The middle circle is labeled "layer 1". The outermost circle is labeled "layer N user interface". Three vertical dots are placed between "layer 1" and "layer N user interface", indicating intermediate layers. The entire structure is set against a light gray background.

Basic Concepts 58 Operating Systems

Microkernel System Structure

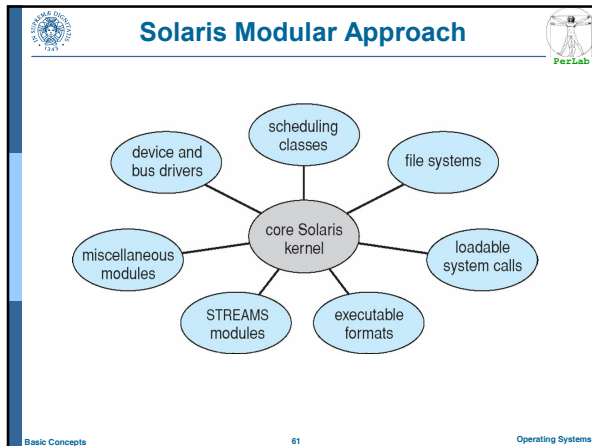
- Moves as much from the kernel into “user” space
- Communication takes place between user modules using message passing
- Benefits:
 - Easier to extend a microkernel
 - Easier to port the operating system to new architectures
 - More reliable (less code is running in kernel mode)
 - More secure
- Detriments: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

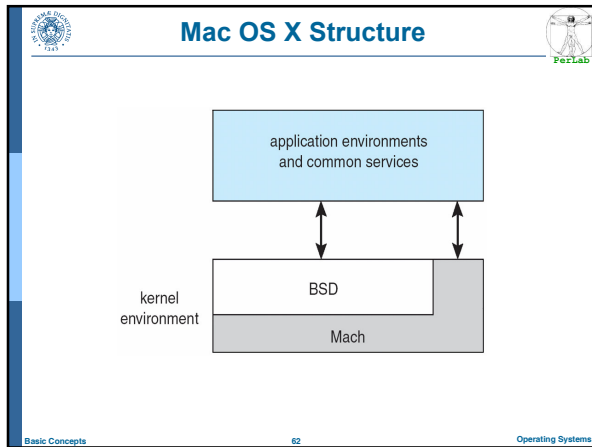
Basic Concepts 59 Operating Systems

Modules

- Most modern operating systems implement kernel modules
 - Uses object-oriented approach
 - Each core component is separate
 - Each talks to the others over known interfaces
 - Each is loadable as needed within the kernel
- Overall, similar to layers but with more flexibility

Basic Concepts 60 Operating Systems





- ### Mac OS X Structure (cont'd)
- Mach Micro-kernel is responsible for
 - Memory management
 - Remote procedure calls (RPC)
 - Inter-process communication (IPC)
 - CPU scheduling
 - BSD kernel provides
 - CLI User Interface
 - File manipulation services
 - Networking services
 - POSIX API (including Pthreads)
- Basic Concepts 63 Operating Systems

Overview

- Preliminary Concepts
- Services
- System Calls
- System Programs
- Internal Structure
- **System Boot**

Basic Concepts 64 Operating Systems

System Boot

- Operating system must be made available to hardware so hardware can start it
- **Bootstrap loader**
 - locates the kernel, loads it into memory, and starts it
 - When power initialized on system, execution starts at a fixed memory location
 - firmware used to store the initial boot code (ROM does not need to be initialized and is immune to viruses)
 - Small computers store the entire operating system in ROM (EPROM)
- Two-Step bootstrap
 - **boot block** at fixed location loads bootstrap loader

Basic Concepts 65 Operating Systems

Disk Organization

- Il disco può essere suddiviso in partizioni ognuna contenente un proprio file system
- Il partizionamento del disco avviene mediante la formattazione di alto livello

Disco

Tabella delle Partizioni

Partizioni

MBR

Blocco di avvio Super blocco Gestione blocchi liberi I-node Directory Radice File e Directory

Basic Concepts 66 Operating Systems

Disk Organization

- **MBR (Master Boot Record)**
 - Contiene programma di avvio
 - La fine del MBR contiene la tabella delle partizioni
- **Tabella delle partizioni**
 - Contiene punto di inizio e fine di ogni partizione
 - Una sola partizione è marcata come attiva
 - E' la partizione da cui verrà caricato il SO

The diagram shows a horizontal line representing a disk. On the left, a small box is labeled 'MBR'. To its right is a larger box labeled 'Tabella delle Partizioni'. Further right, several rectangular boxes represent 'Partizioni'. Arrows indicate the flow from the MBR to the Partition Table, and from the Partition Table to the individual partitions.

Basic Concepts 67 Operating Systems

Disk Organization

- **Blocco di avvio**
 - Contiene semplice eseguito in fase di bootstrap e serve a caricare il kernel
 - Ogni partizione contiene il Blocco di Avvio anche se non contiene il SO (potrebbe contenerne uno)
- **Superblocco**
 - Contiene informazioni sul file system
 - ▶ Numero magico che identifica il FS
 - ▶ Numero di blocchi del FS
 - ▶ ...
- **Gestione per lo spazio libero**
 - Strutture dati per la gestione dei blocchi liberi
- **I-node**
 - Nei SO che utilizzano gli i-node questi sono raggruppati in un'parte del disco
- **Directory radice**
- **File e directory**

Basic Concepts 68 Operating Systems

Two-Step Bootstrap (Windows 2000)

- **Esecuzione del programma di avvio in ROM**
 - ▶ Diagnosi
 - ▶ Caricamento del MBR
- **Esecuzione del codice di avvio contenuto nel MBR**
 - ▶ Localizza la partizione attiva dalla tabella delle partizioni
 - ▶ Legge il primo blocco (blocco di avvio) e lo esegue
- **Esecuzione del codice nel Blocco di Avvio**
 - ▶ Localizza il kernel nella partizione attiva
 - ▶ Carica in memoria il kernel
 - ▶ Cede il controllo al kernel

Basic Concepts 69 Operating Systems



Questions?