MODERN OPERATING SYSTEMS
ANDREW S. TANENBAUM

Chapter 1
Introduction

1

Outline

- What is an operating system
- The history of Operating System
- Hardware of operating system
- Important concepts of Operating System

2

What Is An Operating System

A modern computer consists of:

- One or more processors
- Main memory
- Disks
- Printers
- Various input/output devices

Managing all these components requires a layer of software – the **operating system**

3

What is an Operating System?

- A modern computer is very complex.
  - Networking
  - Disks
  - Video/audio card
  - ….
- It is impossible for every application programmer to understand every detail
- A layer of computer software is introduced to provide a better, simpler, cleaner model of the resources and manage them

4

What Is An Operating System

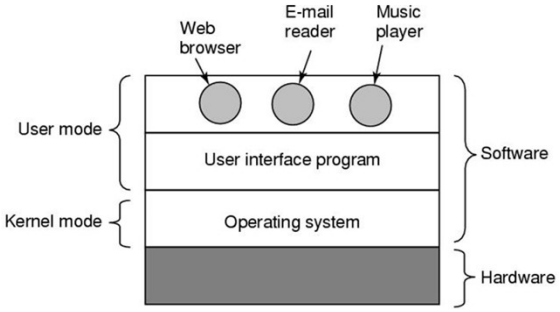


Figure 1-1. Where the operating system fits in.

5

What is an Operating System?

- Users use various OS
  - Windows, Linux, Mac OS etc.
- User interacts with shell or GUI
  - part of OS?
  - they use OS to get their work done
- Is device driver part of OS?

6

What is an Operating System?

- On top of hardware is OS
- Most computers have two modes of operation:
  - Kernel mode and user mode
  - OS runs in **kernel mode,** which has complete access to all hardware and can execute any instruction
  - Rest of software runs in user mode, which has limited capability
  - Shell or GUI is the lowest level of user mode software

7

What is an operating system?

- Two functions:
  - From top to down: provide application programmers a clean abstract set of resources instead of hardware ones
  - From down to top: Manage these hardware resources

8

The Operating System as an Extended Machine

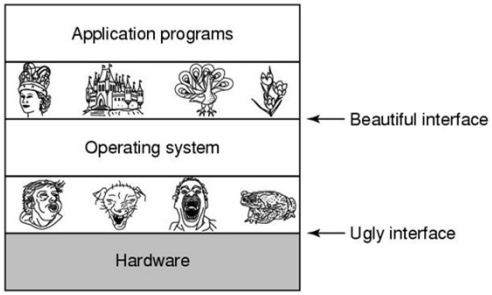


Figure 1-2. Operating systems turn ugly (xấu xí) hardware into beautiful abstractions.

9

As an extended machine

- Abstraction:
  - CPU—process
  - Storage –- files
  - Memory– address space
- 4 types of people:
  - Industrial engineer: design hardware
  - Kernel designer
  - Application programmer: OS's user
  - End users

10

The Operating System as a Resource Manager

- Allow multiple programs to run at the same time
- Manage and protect memory, I/O devices, and other resources
- Includes multiplexing (sharing) resources in two different ways:
  - In time
  - In space

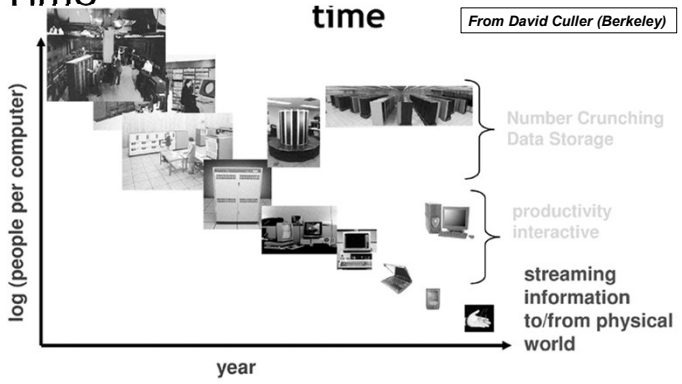
11

As a resource manager

- Modern OS runs multiple programs of multiple users at the same time
  - Imagine (tưởng tượng) what would happen if several programs want to print at the same time?
  - How to account the resource usage of each process?
  - Resources can be multiplexed:
    - How to ensure fairness and efficiency?

12

People-to-Computer Ratio Over Time



13

Early Systems - Bare Machine (1950s)

Hardware – ***expensive*** ; **Human –** ***cheap***

- Structure
  - Large machines run from console
  - Single user system
    - Programmer/User as operator
  - Paper tape or punched cards
- Early software
  - Assemblers, compilers, linkers, loaders, device drivers, libraries of common subroutines.
- Secure execution
- Inefficient use of expensive resources
  - Low CPU utilization, high setup time.

14

Simple Batch Systems (1960's)

- Reduce setup time by batching jobs with similar requirements.
- Add a card reader, Hire an operator
  - User is NOT the operator
  - Automatic job sequencing
    - Forms a rudimentary OS.
  - Resident Monitor
    - Holds initial control, control transfers to job and then back to monitor.
  - Problem
    - Need to distinguish job from job and data from program.

15

Supervisor/Operator Control

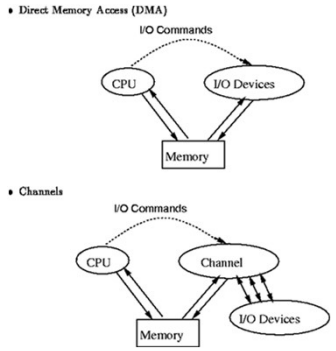
- Secure monitor that controls job processing
  - Special cards indicate what to do.
  - User program prevented from performing I/O
- Separate user from comput
  - User submits card deck
  - cards put on tape
  - tape processed by operator
  - output written to tape
  - tape printed on printer
- Problems
  - Long turnaround time - up to 2 DAYS!!!
  - Low CPU utilization
    - I/O and CPU could not overlap; slow mechanical devices.

16

Batch Systems - Issues

- Solutions to speed up I/O:
- Offline Processing
  - load jobs into memory from tapes, card reading and line printing are done offline.
- Spooling
  - Use disk (random access device) as large storage for reading as many input files as possible and storing output files until output devices are ready to accept them.
  - Allows overlap - I/O of one job with computation of another.
  - Introduces notion of a job pool that allows OS choose next job to run so as to increase CPU utilization.

Speeding up I/O



Batch Systems - I/O completion

- How do we know that I/O is complete?
  - Polling:
    - Device sets a flag when it is busy.
    - Program tests the flag in a loop waiting for completion of I/O.
  - Interrupts:
    - On completion of I/O, device forces CPU to jump to a specific instruction address that contains the interrupt service routine.
    - After the interrupt has been processed, CPU returns to code it was executing prior to servicing the interrupt.

Multiprogramming

- Use interrupts to run multiple programs simultaneously
  - When a program performs I/O, instead of polling, execute another program till interrupt is received.
- Requires secure memory, I/O for each program.
- Requires intervention if program loops indefinitely.
- Requires CPU scheduling to choose the next job to run.

Timesharing

Hardware – *getting cheaper*; Human – *getting expensive*

- Programs queued for execution in FIFO order.
- Like multiprogramming, but timer device interrupts after a quantum (timeslice).
  - Interrupted program is returned to end of FIFO
  - Next program is taken from head of FIFO
- Control card interpreter replaced by command language interpreter.

Timesharing (cont.)

- Interactive (action/response)
  - when OS finishes execution of one command, it seeks the next control statement from user.
- File systems
  - online filesystem is required for users to access data and code.
- Virtual memory
  - Job is swapped in and out of memory to disk.

Personal Computing Systems

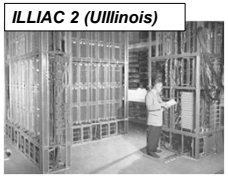
Hardware – *cheap* ; Human – *expensive*

- Single user systems, portable.
- I/O devices - keyboards, mice, display screens, small printers.
- Laptops and palmtops, Smart cards, Wireless devices.
- Single user systems may not need advanced CPU utilization or protection features.
- Advantages:
  - user convenience, responsiveness, ubiquitous

Parallel Systems

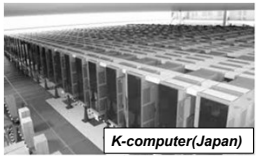
- Multiprocessor systems with more than one CPU in close communication.
- Improved Throughput, economical, increased reliability.
- Kinds:
  - Vector and pipelined
  - Symmetric and asymmetric multiprocessing
  - Distributed memory vs. shared memory
- Programming models:
  - Tightly coupled vs. loosely coupled ,message-based vs. shared variable

Parallel Computing Systems



ILLIAC 2 (Ulllinois)

Climate modeling, earthquake simulations, genome analysis, protein folding, nuclear fusion research, …..



K-computer(Japan)

Connection Machine (MIT)

Tianhe-1(China)

IBM Blue Gene

25

Distributed Systems

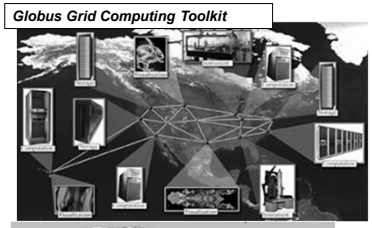
Hardware – ***very cheap*** ; **Human – *very expensive***

- Distribute computation among many processors.
- Loosely coupled -
  - no shared memory, various communication lines
- client/server architectures
- Advantages:
  - resource sharing
  - computation speed-up
  - reliability
  - communication - e.g. email
- Applications - digital libraries, digital multimedia

26

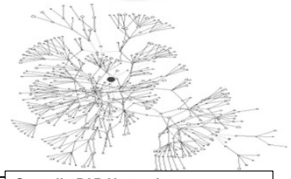
Distributed Computing Systems

Globus Grid Computing Toolkit



Cloud Computing Offerings

PlanetLab

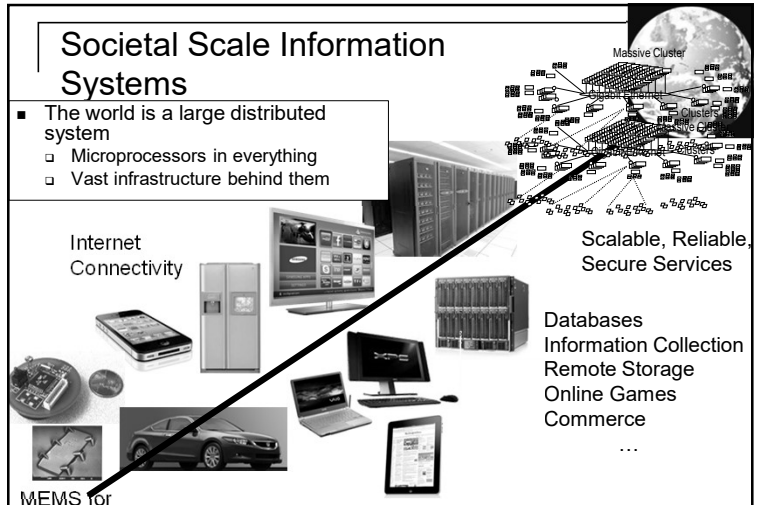


Gnutella P2P Network

27

Societal Scale Information Systems

- The world is a large distributed system
  - Microprocessors in everything
  - Vast infrastructure behind them



28

Real-time systems

- Correct system function depends on timeliness
- Feedback/control loops
- Sensors and actuators
- Hard real-time systems -
  - Failure if response time too long.
  - Secondary storage is limited
- Soft real-time systems -
  - Less accurate if response time is too long.
  - Useful in applications such as multimedia, virtual reality.

29

Computer Hardware Review

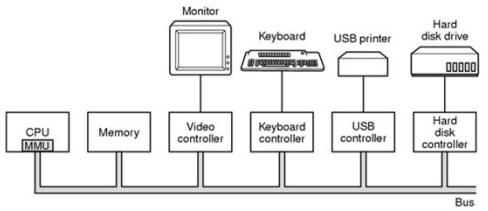


Figure 1-6. Some of the components of a simple personal computer.

30

Hardware: processor

- Brain of computer
  - Fetches instruction from memory and execute
  - Cycle of CPU:
    - fetch, decode, execute
  - CPU has registers to store variable and temporary result: load from memory to register; store from register to memory
  - Program counter: next instruction to fetch
  - Stack pointer: the top of the current stack
  - PSW: program status word, priority, mode…

31

A reference

- Adam Smith: Wealth of Nation
- Of the Division of labor
- One man draws out the wire, another straights it, a third cuts it, a fourth points it, a fifth grinds it at the top…
- One person could make up to forth-eight thousand pins in a day

32

CPU Pipelining

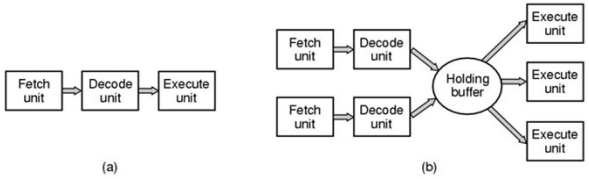


Figure 1-7. (a) A three-stage pipeline. (b) A superscalar CPU.

33

CPU Pipeline

- Accelerate (đẩy nhanh) the execution
- Cause headaches (đau đầu) to OS/compiler writers
- For superscalar: instructions are often executed out of order

34

Multithreaded and Multicore Chips

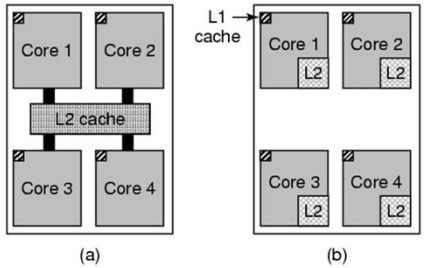


Figure 1-8. (a) A quad-core chip with a shared L2 cache.
(b) A quad-core chip with separate L2 caches.

35

Memory (1)

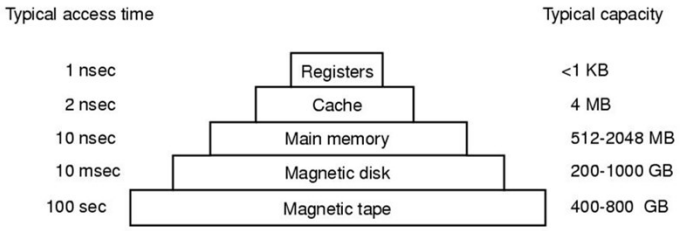


Figure 1-9. A typical memory hierarchy.
The numbers are very rough approximations.

36

memory

- Memory is where computer fetch and store data, ideally, it should be both chip/large
- The best people can do, is to construct a memory hierarchy
- Cache lines:
  - Memory divided into cache lines; the mostly used ones are stored in caches
  - Cache hit, cache miss
  - Cache: whenever there is disparity (chênh lệch) in usage or speed; used to improve performance

37

Memory (2)

Questions when dealing with cache:

- When to put a new item into the cache.
- Which cache line to put the new item in.
- Which item to remove from the cache when a slot is needed.
- Where to put a newly evicted item in the larger memory.

38

Disks

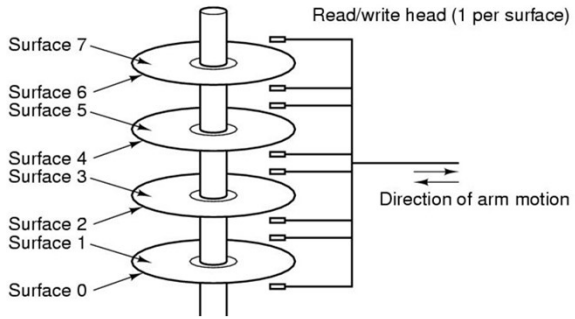


Figure 1-10. Structure of a disk drive.

39

Disks

- Cheap and large: two orders better than RAM
- Slow: three orders worse than RAM
- Mechanical movement to fetch data
- One or more platter—rotate– rpm
- Information is stored on concentric (đồng tâm) circles
- Arm, track, cylinder, sector
- Disk helps to implement Virtual Memory
  - When no enough memory is available, disks are used as the storage, and memory as cache

40

I/O Devices

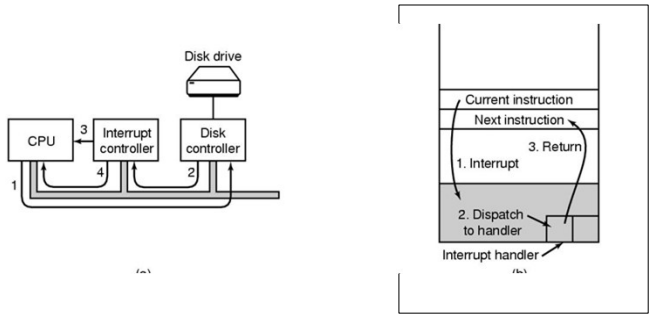


Figure 1-11. (a) The steps in starting an I/O device and getting an interrupt.

41

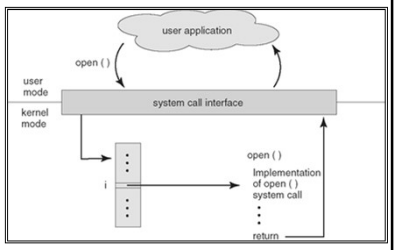
I/O devices

- Two parts: a controller and a device
- Controller: to provide a simpler interface to OS
- Device driver: talks to controller, give commands and accepts response
- Busy waiting/interrupt/DMA

42

System Calls

- Interface between running program and the OS.
  - Assembly language instructions (macros and subroutines)
  - Some higher level languages allow system calls to be made directly (e.g. C)
- Passing parameters between a running program and OS via registers, memory tables or stack.
- Unix has about 32 system calls
  - read(), write(), open(), close(), fork(), exec(), ioctl(),…..



43

System Calls

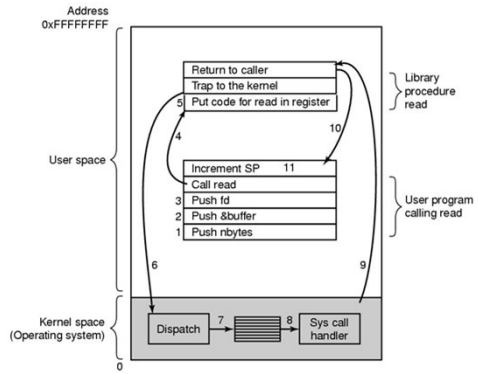


Figure 1-17. The 11 steps in making the system call read(fd, buffer, nbytes).

44

System Calls for Process Management

Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

Figure 1-18. Some of the major POSIX system calls.

45

System Calls for File Management (1)

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing, or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Figure 1-18. Some of the major POSIX system calls.

46

System Calls for File Management (2)

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Figure 1-18. Some of the major POSIX system calls.

47

Miscellaneous System Calls

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

Figure 1-18. Some of the major POSIX system calls.

48

A Simple Shell

```
#define TRUE 1

while (TRUE) {                                  /* repeat forever */
      type_prompt( );                           /* display prompt on the screen */
      read_command(command, parameters);        /* read input from terminal */

      if (fork( ) != 0) {                       /* fork off child process */
          /* Parent code. */
          waitpid(-1, &status, 0);              /* wait for child to exit */
      } else {
          /* Child code. */
          execve(command, parameters, 0);       /* execute command */
      }
}
```

Figure 1-19. A stripped-down shell.

49

Memory Layout

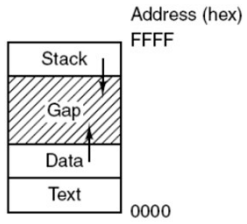


Figure 1-20. Processes have three segments: text, data, and stack.

50

Linking

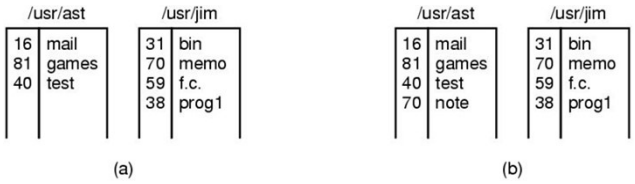


Figure 1-21. (a) Two directories before linking */usr/jim/memo* to ast's directory. (b) The same directories after linking.

51

Mounting

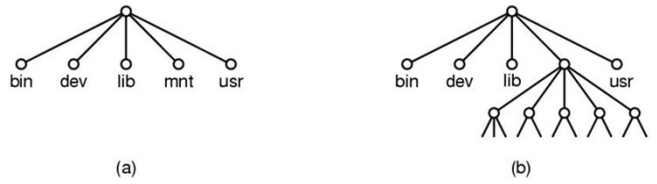


Figure 1-22. (a) File system before the mount. (b) File system after the mount.

52

Windows Win32 API

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time

Figure 1-23. The Win32 API calls that roughly correspond to the UNIX calls of Fig. 1-18.

53

Operating System Services

- Services that provide user-interfaces to OS
  - Program execution - load program into memory and run it
  - I/O Operations - since users cannot execute I/O operations directly
  - File System Manipulation - read, write, create, delete files
  - Communications - interprocess and intersystem
  - Error Detection - in hardware, I/O devices, user programs
- Services for providing efficient system operation
  - Resource Allocation - for simultaneously executing jobs
  - Accounting - for account billing and usage statistics
  - Protection - ensure access to system resources is controlled

54

System Programs

- Convenient environment for program development and execution. User view of OS is defined by system programs, not system calls.
  - Command Interpreter (sh, csh, ksh) - parses/executes other system programs
  - File manipulation - copy (cp), print (lpr), compare(cmp, diff)
  - File modification - editing (ed, vi, emacs)
  - Application programs - send mail (mail), read news (rn)
  - Programming language support (cc)
  - Status information, communication
  - etc….

55

Command Interpreter System

- Commands that are given to the operating system via command statements that execute
  - Process creation and deletion, I/O handling, Secondary Storage Management, Main Memory Management, File System Access, Protection, Networking.
- Obtains the next command and executes it.
- Programs that read and interpret control statements also called -
  - Control card interpreter, command-line interpreter, shell (in UNIX)

56

System Design and Implementation

- Establish design goals
  - User Goals
  - System Goals
- Software Engineering -
  - Separate mechanism from policy. Policies determine what needs to be done, mechanisms determine how they are done.
- Choose a high-level implementation language
  - faster implementation, more compact, easier to debug

57

System Generation

- OS written for a class of machines, must be configured for each specific site.
  - SYSGEN program obtains info about specific hardware configuration and creates version of OS for hardware
- Booting
- Bootstrap program - loader program loads kernel, kernel loads rest of OS.
  - Bootstrap program stored in ROM

58

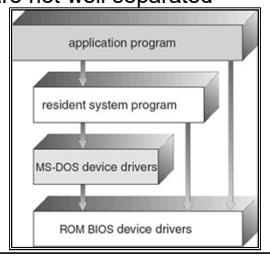
Operating Systems: How are they organized?

- Simple
  - Only one or two levels of code
- Layered
  - Lower levels independent of upper levels
- Microkernel
  - OS built from many user-level processes
- Modular
  - Core kernel with Dynamically loadable modules

59

OS Structure - Simple Approach

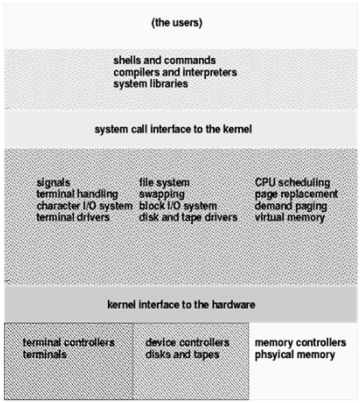
- MS-DOS - provides a lot of functionality in little space.
  - Not divided into modules, Interfaces and levels of functionality are not well separated



60

UNIX System Structure

- UNIX - limited structuring, has 2 separable parts
  - Systems programs
  - Kernel
    - everything below system call interface and above physical hardware.
    - Filesystem, CPU scheduling, memory management



61

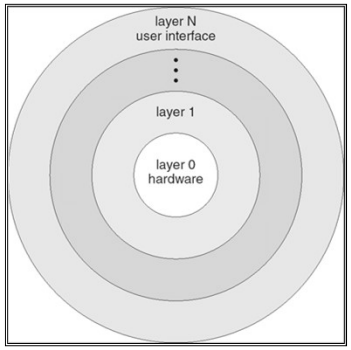
Layered OS Structure

User Programs
Interface Primitives
Device Drivers and Schedulers
Virtual Memory
I/O
CPU Scheduling
Hardware

- OS divided into number of layers - bottom layer is hardware, highest layer is the user interface.
- Each layer uses functions and services of only lower-level layers.
- THE Operating System Kernel has successive layers of abstraction.

62

Layered Operating System



63

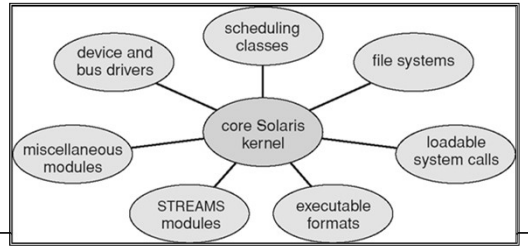
Microkernel Structure

- Moves as much from the kernel into "*user*" space
  - Small core OS running at kernel level
  - OS Services built from many independent user-level processes
- Communication between modules with message passing
- Benefits:
  - Easier to extend a microkernel
  - Easier to port OS to new architectures
  - More reliable (less code is running in kernel mode)
  - Fault Isolation (parts of kernel protected from other parts)
  - More secure
- Detriments:
  - Performance overhead severe for naïve implementation

64

Modules-based Structure

- Most modern operating systems implement 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 flexible



65

OS Task: Process Management

- Process - fundamental concept in OS
  - Process is a program in execution.
  - Process needs resources - CPU time, memory, files/data and I/O devices.
- OS is responsible for the following process management activities.
  - Process creation and deletion
  - Process suspension and resumption
  - Process synchronization and interprocess communication
  - Process interactions - deadlock detection, avoidance and correction

66

OS Task: Memory Management

- Main Memory is an array of addressable words or bytes that is quickly accessible.
- Main Memory is volatile.
- OS is responsible for:
  - Allocate and deallocate memory to processes.
  - Managing multiple processes within memory - keep track of which parts of memory are used by which processes. Manage the sharing of memory between processes.
  - Determining which processes to load when memory becomes available.

67

OS Task: Secondary Storage and I/O Management

- Since primary storage is expensive and volatile, secondary storage is required for backup.
- Disk is the primary form of secondary storage.
  - OS performs storage allocation, free-space management and disk scheduling.
- I/O system in the OS consists of
  - Buffer caching and management
  - Device driver interface that abstracts device details
  - Drivers for specific hardware devices

68

OS Task: File System Management

- File is a collection of related information defined by creator - represents programs and data.
- OS is responsible for
  - File creation and deletion
  - Directory creation and deletion
  - Supporting primitives for file/directory manipulation.
  - Mapping files to disks (secondary storage).
  - Backup files on archival media (tapes).

69

OS Task: Protection and Security

- Protection mechanisms control access of programs and processes to user and system resources.
  - Protect user from himself, user from other users, system from users.
- Protection mechanisms must:
  - Distinguish between authorized and unauthorized use.
  - Specify access controls to be imposed on use.
  - Provide mechanisms for enforcement of access control.
  - Security mechanisms provide trust in system and privacy
    - authentication, certification, encryption etc.

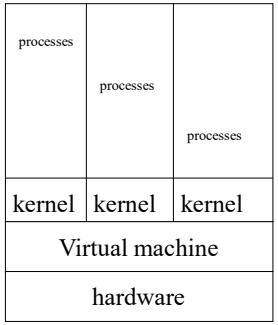
70

OS Task: Networking

- Connecting processors in a distributed system
- Distributed System is a collection of processors that do not share memory or a clock.
- Processors are connected via a communication network.
- Advantages:
  - Allows users and system to exchange information
  - provide computational speedup
  - increased reliability and availability of information

71

Virtual Machines



- Logically treats hardware and OS kernel as hardware
- Provides interface identical to underlying bare hardware.
- Creates illusion of multiple processes - each with its own processor and virtual memory

72

Summary

- Operating system is a software
  - Is a complex software
  - Runs in kernel mode
  - Manages hardware
  - Provide a friendly interface for application programmer

73

Summary

- What is an operating system?
- Early Operating Systems
- Simple Batch Systems
- Multiprogrammed Batch Systems
- Time-sharing Systems
- Personal Computer Systems
- Parallel and Distributed Systems
- Real-time Systems

74

74

Summary

- Operating System Concepts
- Operating System Services, System Programs and System calls
- Operating System Design and Implementation
- Structuring Operating Systems

75