Operating Systems
CS 571

Prof. Sanjeev Setia

Fall 2002

Overview

- Prerequisites
  - Computer Architecture (CS 365)
  - Data structures and programming (CS 310)
    - (C++/C/Java progamming)
- Textbooks
  - Modern Operating Systems (Tannenbaum)
  - Distributed Systems – Concepts and Design (Coulouris, Dollimore, Kindberg)

Overview cont’d

- Grading
  - One midterm exam (25%), Final Exam (25%)
  - Assignments (50%)
- Four Programming Assignments
  - Concurrent Programming (2 weeks)
  - Interprocess Communication
    - Socket programming (2 weeks)
    - RMI/RPC (2 weeks)
  - Distributed File System (5 weeks)
    - First three to be done individually, fourth may be done in groups of two
    - Late submission penalty – 10% per day
  - Plagiarism => Severe penalties (fail the class, expelled from program)

Logistics

- Class Web Page
  - http://www.cs.gmu.edu/~setia/cs571/
  - Slides, Handouts, Old Exams, Useful Links
- Slides
  - More than 90% of slides taken from slides made available by authors of textbooks
  - http://www.cs.vu.nl/~ast/books/mos2/
  - http://www.cdk3.net/

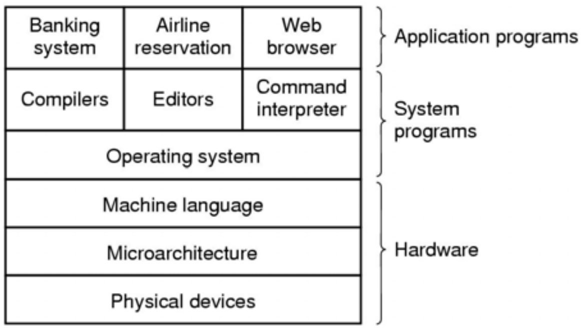
Logistics

- Office Hrs
  - Tuesday, 2 – 4 pm, Room 347, S&T II
  - setia@cs.gmu.edu
- TA
  - Arshad Ahmed (aahmed8@gmu.edu)
  - Office Hrs: Mon 2:30-4:30 pm, Tue 7:30-9:00 pm
  - Office, S&T II Room 435
- Computer Accounts (IT&E or your own machine at home/work)

Chapter 1

Introduction

Introduction



- A computer system consists of
  - hardware
  - system programs
  - application programs

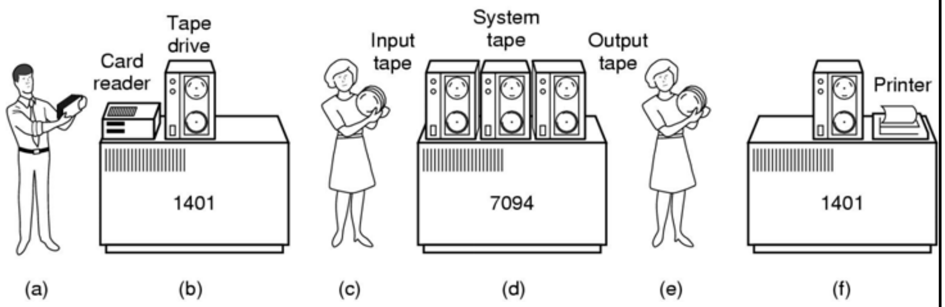
7

What is an Operating System

- It is an extended machine
  - Hides the messy details of actions which must be performed to use the hardware
  - Presents user with a virtual machine, easier to use
- It is a resource manager
  - Each program gets time with the resource
  - Each program gets space on the resource

8

History of Operating Systems (1)



Early batch system
- bring cards to 1401
- read cards to tape
- put tape on 7094 which does computing
- put tape on 1401 which prints output

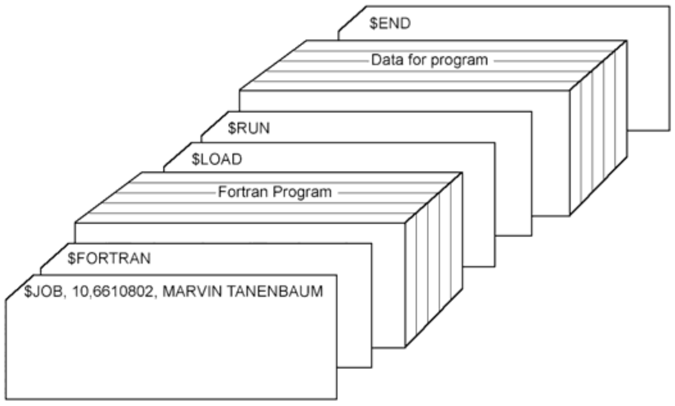
9

History of Operating Systems (2)

- First generation 1945 - 1955
  - vacuum tubes, plug boards
- Second generation 1955 - 1965
  - transistors, batch systems
- Third generation 1965 – 1980
  - ICs and multiprogramming
- Fourth generation 1980 – present
  - personal computers
- Fifth generation?? 2000 – present
  - Handheld and embedded devices

10

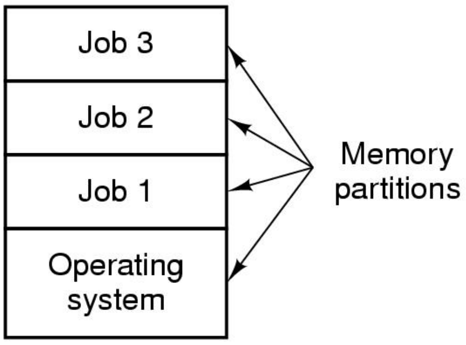
History of Operating Systems (3)



- Structure of a typical FMS job – 2nd generation

11

History of Operating Systems (4)



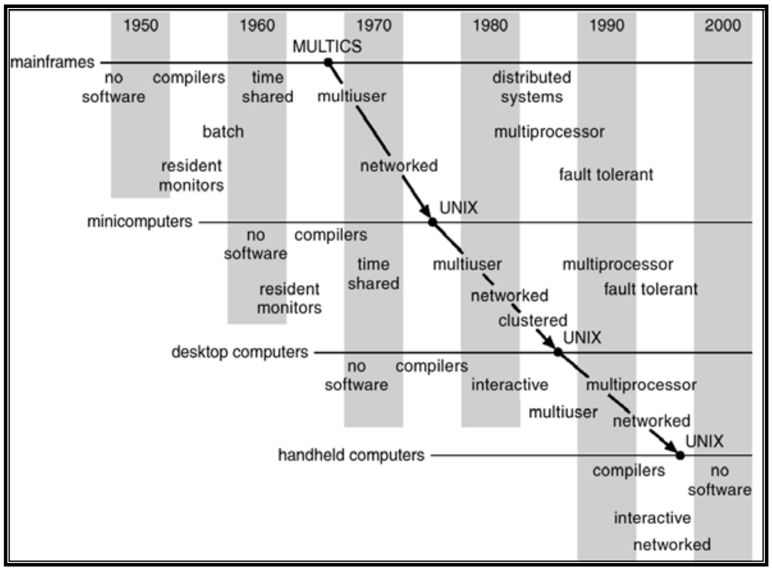
- Multiprogramming system
  - three jobs in memory – 3rd generation

12

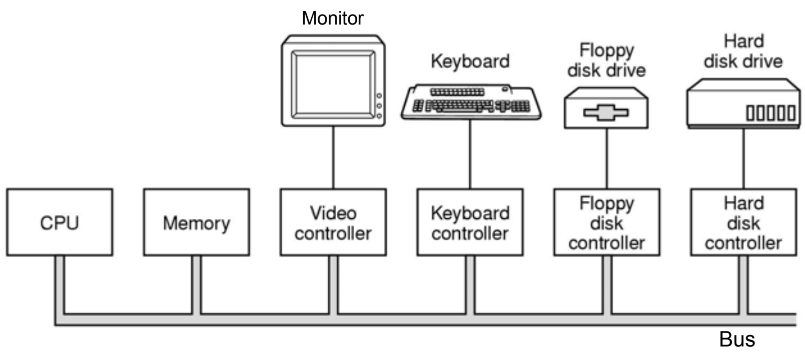
The Operating System Zoo

- Mainframe operating systems
- Server operating systems
- Multiprocessor operating systems
- Personal computer operating systems
- Real-time operating systems
- Embedded operating systems
- Smart card operating systems

Migration of Operating-System Concepts and Features



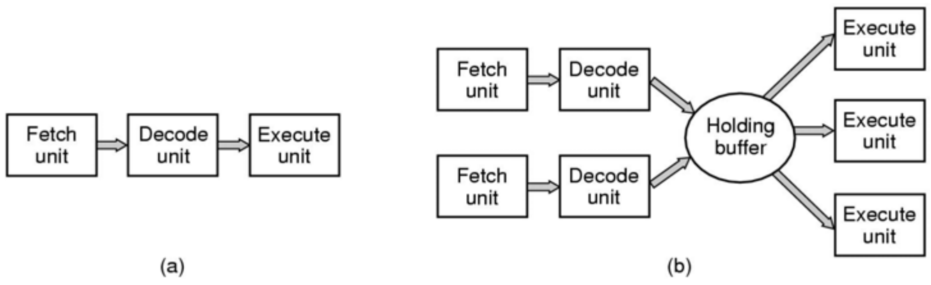
Computer Hardware Review (1)



- Components of a simple personal computer

15

Computer Hardware Review – CPU architecture



(a) A three-stage pipeline
(b) A superscalar CPU

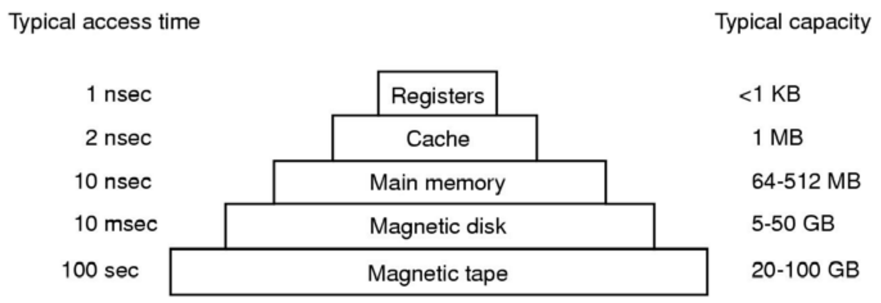
16

Computer Hardware Review – CPU architecture

- CPU has several general-purpose and special-purpose registers
- Special purpose registers
  - PC (program counter)
  - SP (stack pointer)
  - PSW (processor status word)
    - Reflects "status" of CPU, i.e., user/kernel mode, interrupt mask, etc.

17

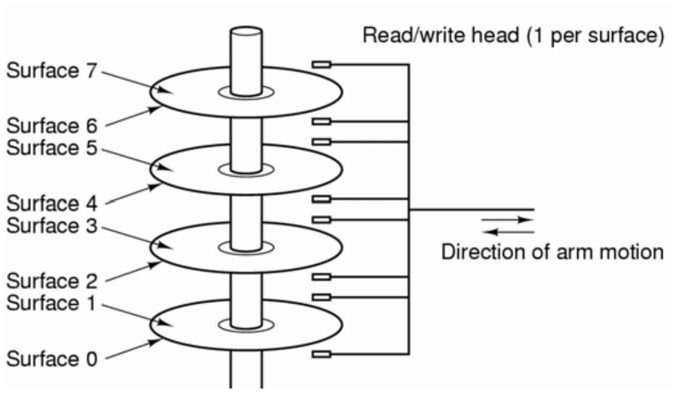
Computer Hardware Review – Storage Hierarchy



- Typical storage hierarchy
  - numbers shown are rough approximations

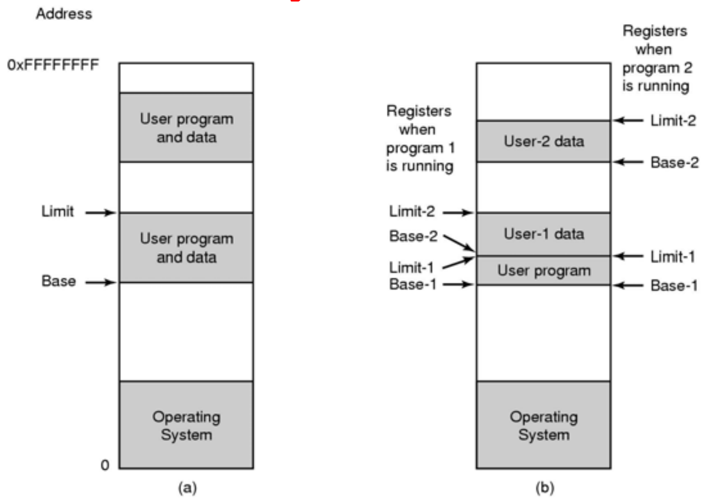
18

Computer Hardware Review - Disk Drives



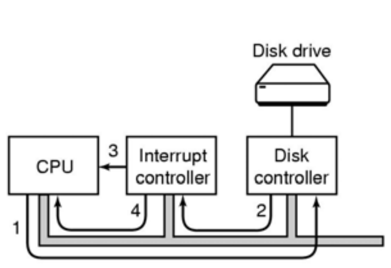
Structure of a disk drive

Computer Hardware Review - Memory Protection

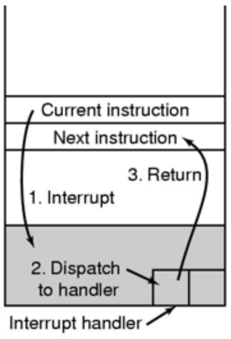


One base-limit pair and two base-limit pairs

Computer Hardware Review – Interrupts



(a)



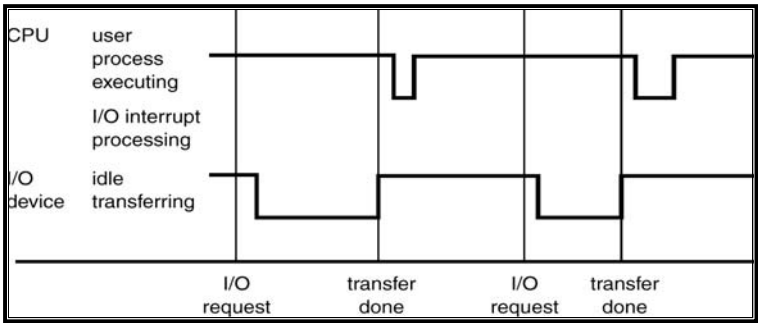
(b)

(a) Steps in starting an I/O device and getting interrupt
(b) How the CPU is interrupted

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter.
- Determines which type of interrupt has occurred:
  - Usually a *vectored* interrupt system
- Separate segments of code (interrupt handlers) determine what action should be taken for each type of interrupt

Interrupt Time Line For a Single Process Doing Output



23

Computer Hardware Review - Direct Memory Access (DMA)

- Used for high-speed I/O devices able to transmit information at close to memory speeds.
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention.
- Only one interrupt is generated per block, rather than the one interrupt per byte.

24

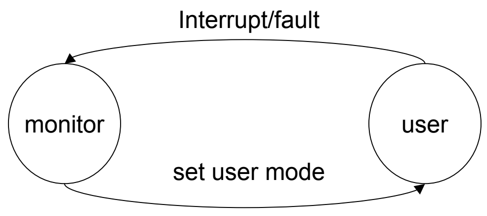
Computer Hardware Review - Dual-Mode Operation

- Sharing system resources requires operating system to ensure that an incorrect program cannot cause other programs to execute incorrectly.
- Provide hardware support to differentiate between at least two modes of operations.
  1. *User mode* – execution done on behalf of a user.
  2. *Monitor mode* (also *kernel mode* or *system mode*) – execution done on behalf of operating system.

25

Dual-Mode Operation (Cont.)

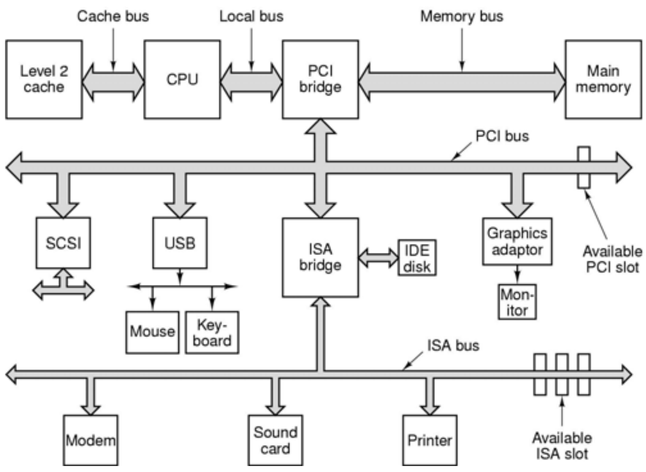
- *Mode bit* added to computer hardware to indicate the current mode: monitor (0) or user (1).
- When an interrupt or fault occurs hardware switches to monitor mode.



Privileged instructions can be issued only in monitor mode.

26

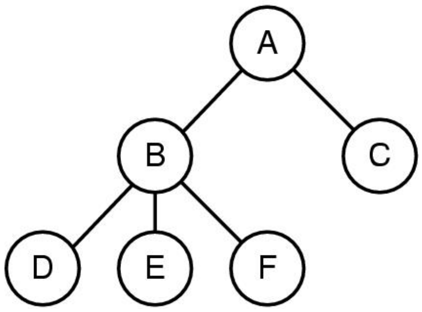
Computer Hardware Review



Structure of a large Pentium system

27

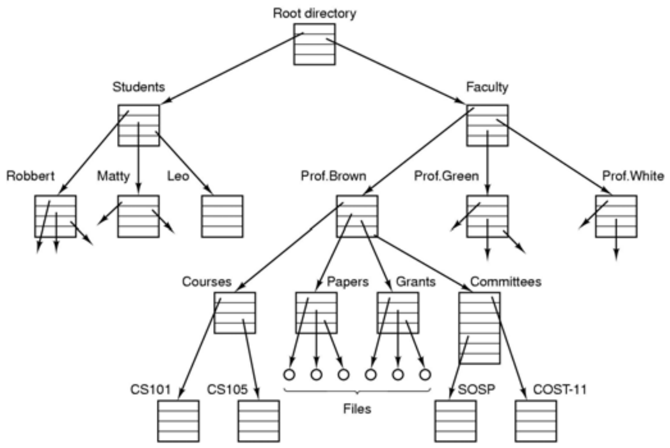
Operating System Concepts (1)



- A process tree
  - A created two child processes, B and C
  - B created three child processes, D, E, and F

28

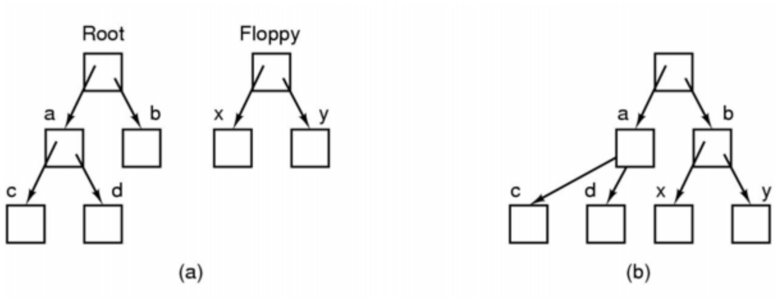
Operating System Concepts (2)



File system for a university department

29

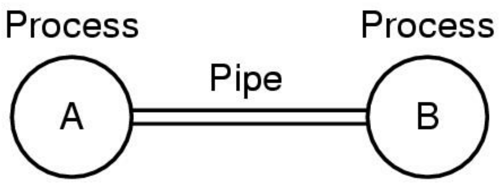
Operating System Concepts (3)



- Before mounting,
  - files on floppy are inaccessible
- After mounting floppy on b,
  - files on floppy are part of file hierarchy

30

Operating System Concepts (4)



Two processes connected by a pipe

31

Operating System Functions

- Process management
- Inter-process communication
- Memory management
- File management
- I/O and Storage system management
  - Power management in newer systems
- Networking
- Security & Protection
- User Interface
  - Can be part of the OS (e.g. Windows) or outside the OS (Unix)

32

Operating System Interface

- System commands
  - ls, ps, cat (UNIX), dir, type (Windows)
- System calls
  - API for invoking operating system services
  - read(), write(), fork()

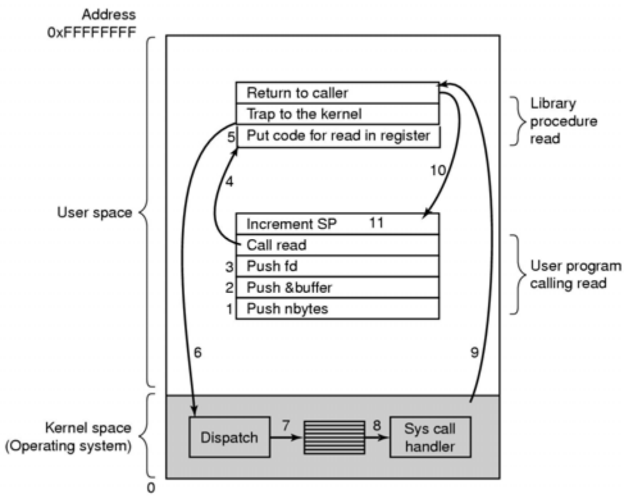
33

Command-Interpreter System

- Many commands are given to the operating system by control statements which deal with:
  - process creation and management
  - I/O handling
  - secondary-storage management
  - main-memory management
  - file-system access
  - protection
  - networking

34

Steps in Making a System Call



There are 11 steps in making the system call read (fd, buffer, nbytes)

Some 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

Some System Calls For File Management

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

37

Some System Calls For Directory Management

Directory and file system management

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

38

Some System Calls For Miscellaneous Tasks

Miscellaneous

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

39

System Calls (1)

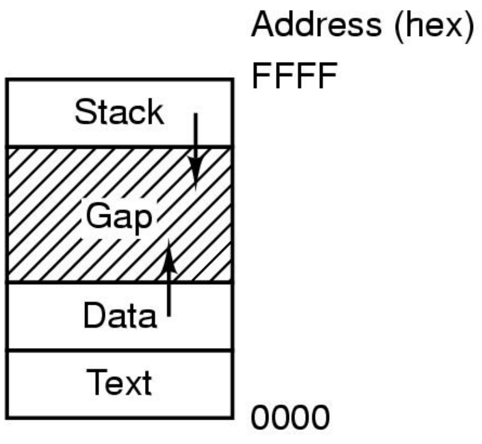
- A stripped down shell:

```
while (TRUE) {                                      /* repeat forever */
    type_prompt( );                                 /* display prompt */
    read_command (command, parameters)              /* 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 */
 }
}
```

40

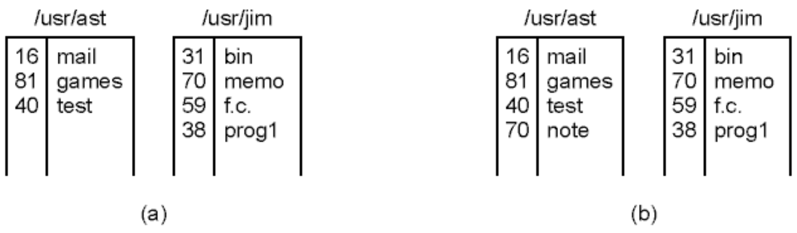
System Calls (2)



- Processes have three segments: text, data, stack

41

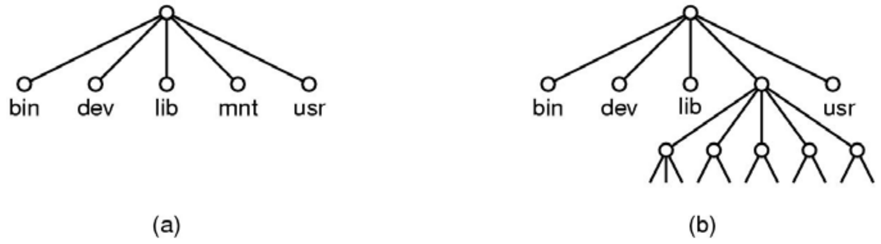
System Calls (3)



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

42

System Calls (4)

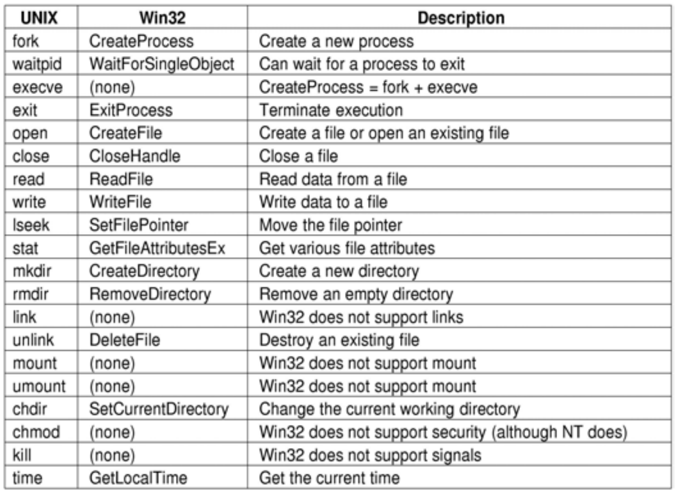


(a) File system before the mount

(b) File system after the mount

43

System Calls (5)

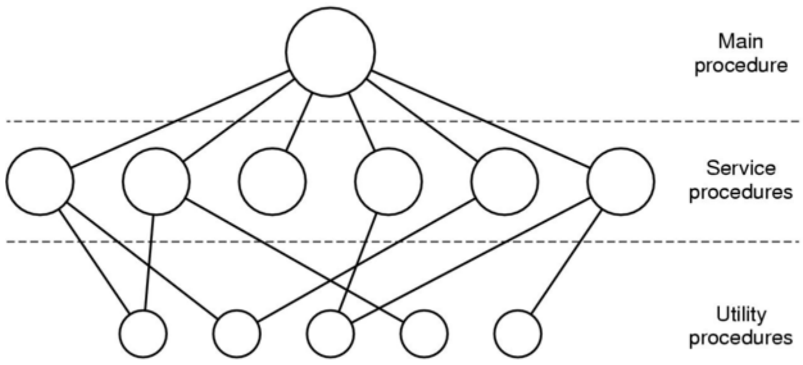


UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	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

Some Win32 API calls

44

Operating System Structure (1)



Simple structuring model for a monolithic system

45

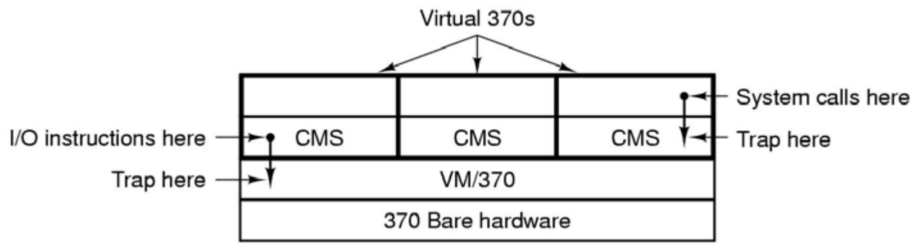
Operating System Structure (2)

Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

Structure of the THE operating system

46

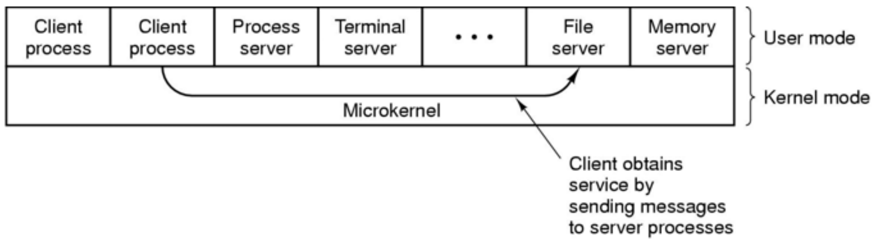
Operating System Structure (3)



Structure of VM/370 with CMS

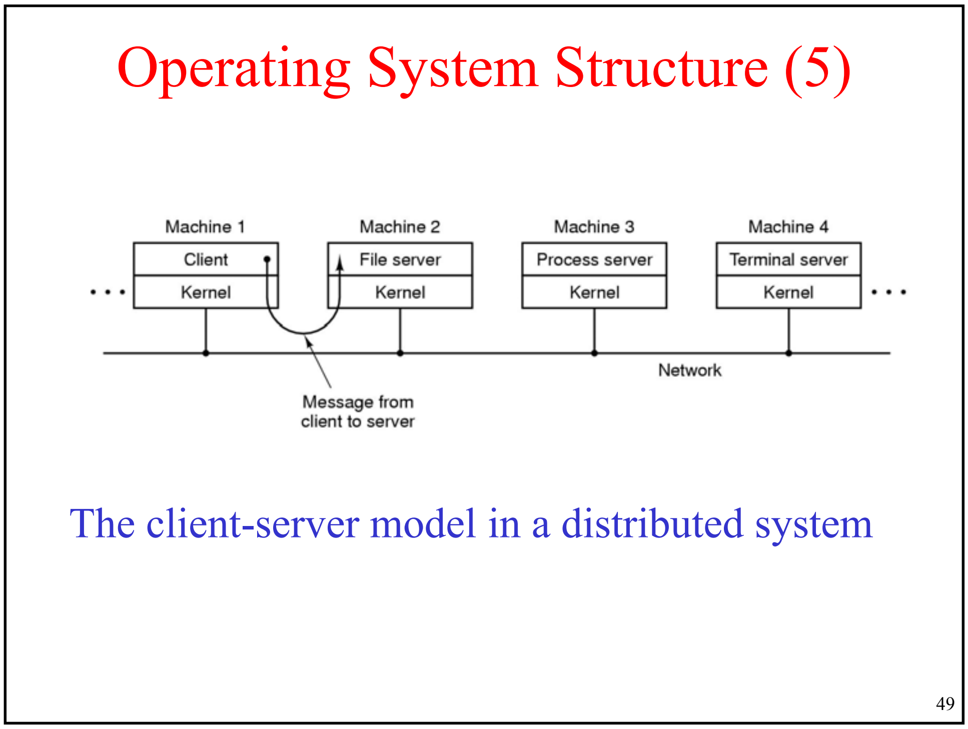
47

Operating System Structure (4)

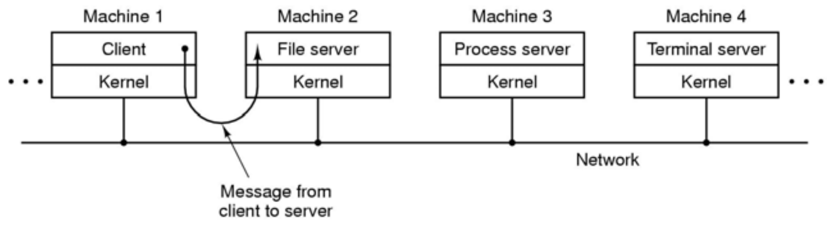


The client-server model

48



Operating System Structure (5)



The client-server model in a distributed system