

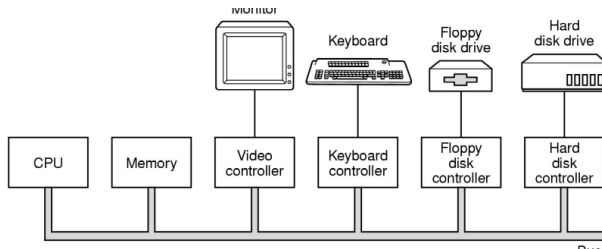
# Lecture 2

CprE 308

January 15, 2013

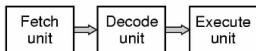
# Computer Hardware Review

# Components of a single personal computer

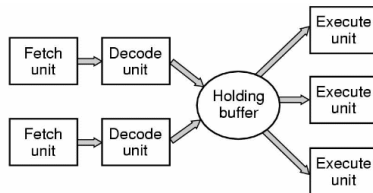


# Sample CPU Architecture

- 1 A three-stage pipeline
- 2 A superscalar CPU

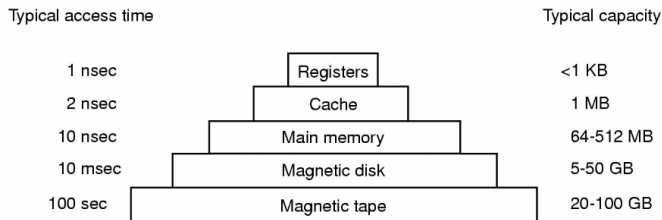


(a)



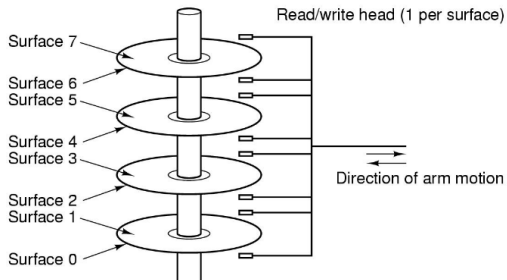
(b)

# Typical memory hierarchy

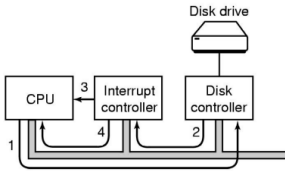


- numbers shown are rough approximations

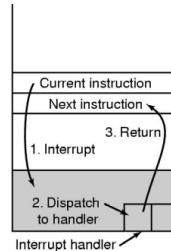
# Structure of a disk drive



# Interrupts



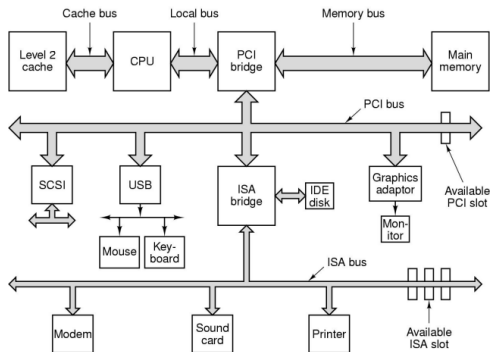
(a)



(b)

- 1 Steps in starting an I/O device and getting an interrupt
- 2 How the CPU is interrupted

# Structure of a large Pentium system





# Programs & Processes

# Program vs. Process

A process is a program in execution

- Program is a piece of code
- Process has “state” (what could this state be?)

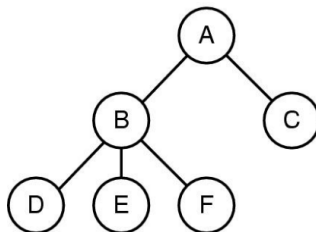
There could be multiple processes all simultaneously executing the same program

- Coordinate accesses and sharing of resources
- Sharing in time - CPU cycles
- Sharing in space - Memory

# Processes

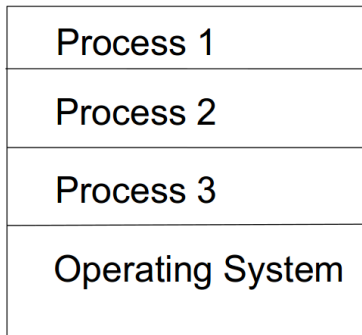
- Process = program in execution
  - Address space: Program (text), data, stack
  - Registers: Program counter, stack pointer, etc.
- Process can be created, suspended, restarted, killed (!)
- Process scheduler decides which process to run next among all the current processes

# Process Creation



- In UNIX, there is a way for one process to “spawn” more processes
- A process tree
  - Process A created two child processes, B and C
  - Process B created three child processes, D, E, and F

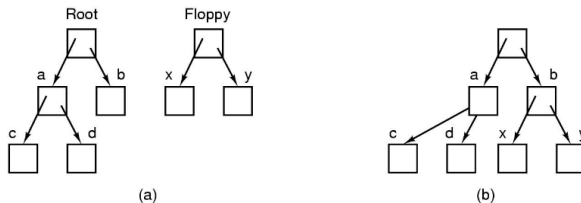
# Memory Sharing



Physical memory

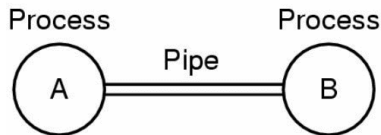
# Files

# Mounting Files (UNIX)



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

# Inter process communication (UNIX)

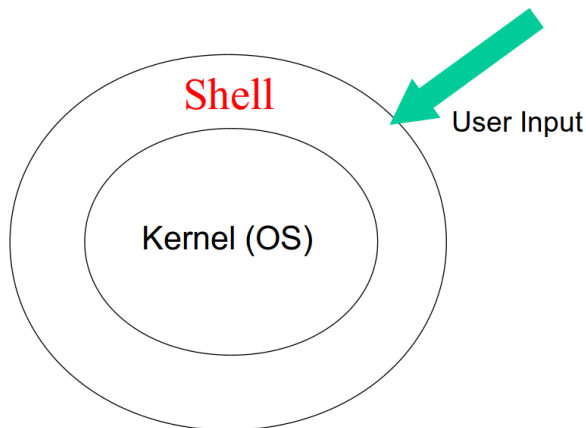


- A *pipe* is like a pseudo file
- Processes set up a pipe in advance
- Processes read from or write to a pipe



# System Calls

# Structure of UNIX



# System Calls

- Interface between the user and the operating system (kernel)
- Handle processes, files, directories, time, input/output
- Switch processor from user to kernel mode
  - In *User mode*, some instructions are forbidden
  - In *Kernel mode*, all instructions are allowed

# Example

- Read from file  
`n = read(fd, buffer, nbytes);`
- Change directory  
`s = chdir(dirname);`
- Get time  
`s = time(&seconds);`

# System Calls for File Management

**File management**

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

# System Calls for Directory Management

**Directory and file system management**

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

# System Calls for Miscellaneous Tasks

## Process management

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

## File management

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

## Directory and file system management

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

## Miscellaneous

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

# Quick Questions

Do these belong in the OS?

- Text editor



# Quick Questions

Do these belong in the OS?

- Text editor
- Compiler

# Quick Questions

Do these belong in the OS?

- Text editor
- Compiler
- Web browser

# Quick Questions

Do these belong in the OS?

- Text editor
- Compiler
- Web browser
- Shell

# Quick Questions

Do these belong in the OS?

- Text editor
- Compiler
- Web browser
- Shell
- Email client

# Quick Questions

Do these belong in the OS?

- Text editor
- Compiler
- Web browser
- Shell
- Email client
- Program which copies files

# Quick Questions

Do these belong in the OS?

- Text editor
- Compiler
- Web browser
- Shell
- Email client
- Program which copies files
- Device driver (program which controls a hardware device)