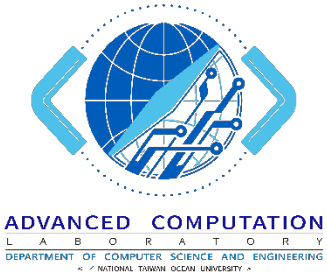


Introduction to Computer Science

William Hsu
Advanced Computation Laboratory
Department of Computer Science and Engineering
Department of Environmental Biology and Fisheries Science
National Taiwan Ocean University

Chapter 3: Operating Systems

*An **operating system** is the **most important software** that runs on a computer. It manages the computer's **memory**, **processes**, and all of its **software** and **hardware**. It also allows you to **communicate** with the computer without knowing how to speak the computer's "language." **Without an operating system, a computer is useless.***

- 3.1 The History of Operating Systems
- 3.2 Operating System Architecture
- 3.3 Coordinating the Machine's Activities
- 3.4 Handling Competition Among Processes
- 3.5 Security

Examples of Operating Systems

- Windows
- UNIX
- Mac OS
- Solaris (Sun/Oracle machines)
- Linux

Smartphone Operating Systems

- Apple iOS
- Windows Phone
- BlackBerry OS
- Nokia Symbian OS
- Google Android

Functions of Operating Systems

- Oversee operation of computer.
- Store and retrieve files.
- Schedule programs for execution.
- Coordinate the execution of programs.

History of Operating Systems

- Each program is called a “job”
- Early computers required significant setup time
- Each “job” required its own setup
- Operating Systems began as systems for simplifying setup and transitions between jobs

History of Operating Systems

- Batch processing (job queue)
- Interactive processing (real time)
- Time-sharing (one machine, many users)
- Multitasking (one user, many tasks)
- Multiprocessor machines (load balancing)
- Embedded Systems (specific devices)

Functions of O.S

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Memory Management

- Memory management refers to management of Primary Memory or Main Memory.
  - Main memory is a large array of words or bytes where each word or byte has its own address.
  - Main memory provides a fast storage that can be accessed directly by the CPU. So for a program to be executed, it must be in the main memory.
  - Activities of O.S for memory management.
  - Keeps tracks of primary memory i.e. what part of it are in use by whom, what part are not in use.
  - In multiprogramming, OS decides which process will get memory when and how much.
  - Allocates the memory when the process requests it to do so.
  - De-allocates the memory when the process no longer needs it or has been terminated.

Processor management

- In multiprogramming environment, OS decides which process gets the processor when and how much time. This function is called process scheduling.
- Activities O. S for processor management.
  - Keeps tracks of processor and status of process. Program responsible for this task is known as traffic controller.
  - Allocates the processor(CPU) to a process.
  - De-allocates processor when processor is no longer required.

Device management

- OS manages device communication via their respective drivers. Operating System does the following activities for device management.
  - Keeps tracks of all devices. Program responsible for this task is known as the I/O controller.
  - Decides which process gets the device when and for how much time.
  - Allocates the device in the efficient way.
  - De-allocates devices.

File management

- A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.
- Activities of the OS in file management.
  - Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.
  - Decides who gets the resources.
  - Allocates the resources.
  - De-allocates the resources.

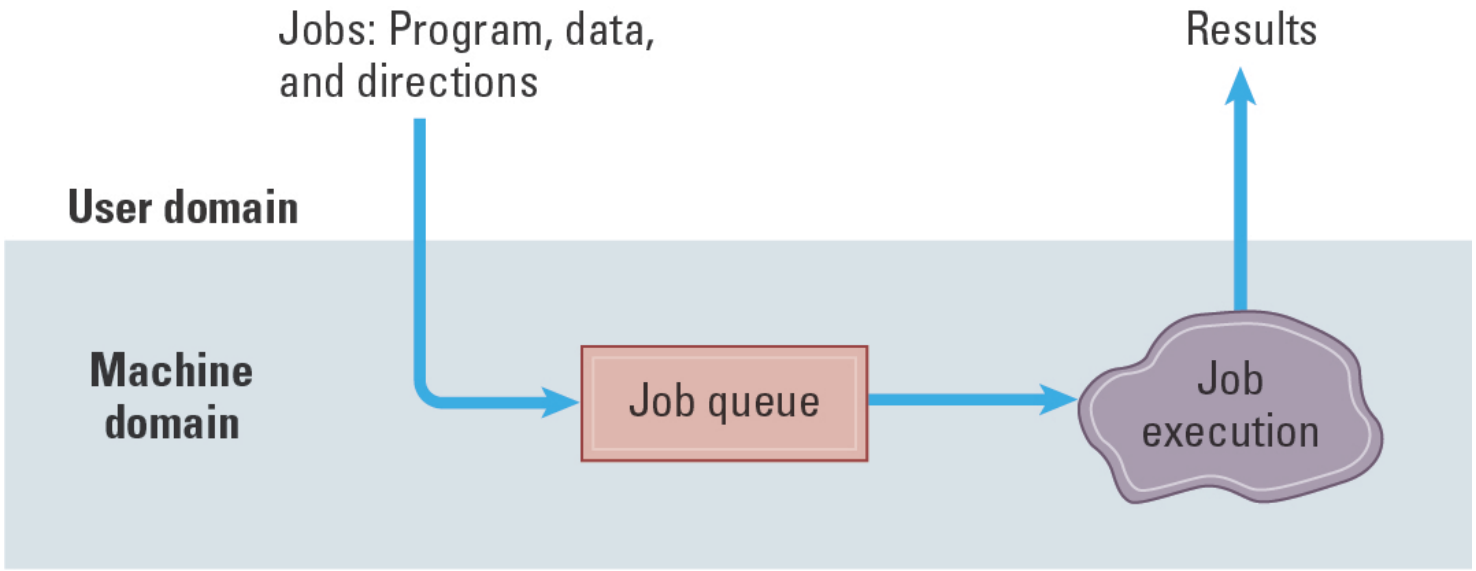
Other activities

- Following are some of the important activities that Operating System does.
- Security -- By means of password and similar other techniques, preventing unauthorized access to programs and data.
- Control over system performance -- Recording delays between request for a service and response from the system.
- Job accounting -- Keeping track of time and resources used by various jobs and users.
- Error detecting aids -- Production of dumps, traces, error messages and other debugging and error detecting aids.
- Coordination between other softwares and users -- Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

Batch Processing

- The users of batch operating system do not interact with the computer directly.
  - Each user prepares his job on an off-line device like punch cards and submits it to the computer operator.
  - To speed up processing, jobs with similar needs are batched together and run as a group.
    - Programmers leave their programs with the operator.
    - The operator then sorts programs into batches with similar requirements.
- The problems with Batch Systems are following.
  - Lack of interaction between the user and job.
  - CPU is often idle, because the speeds of the mechanical I/O devices is slower than CPU.
  - Difficult to provide the desired priority.

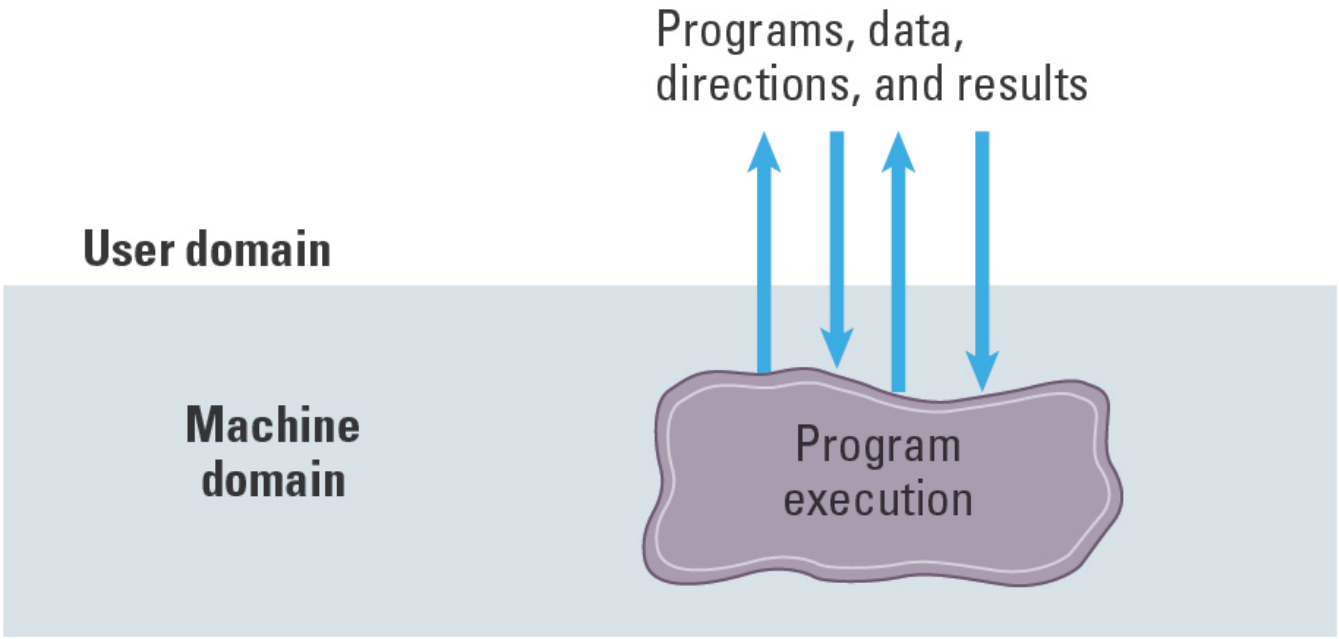
Batch Processing



Job Queue: First in First out (FIFO)

Priority Queues?

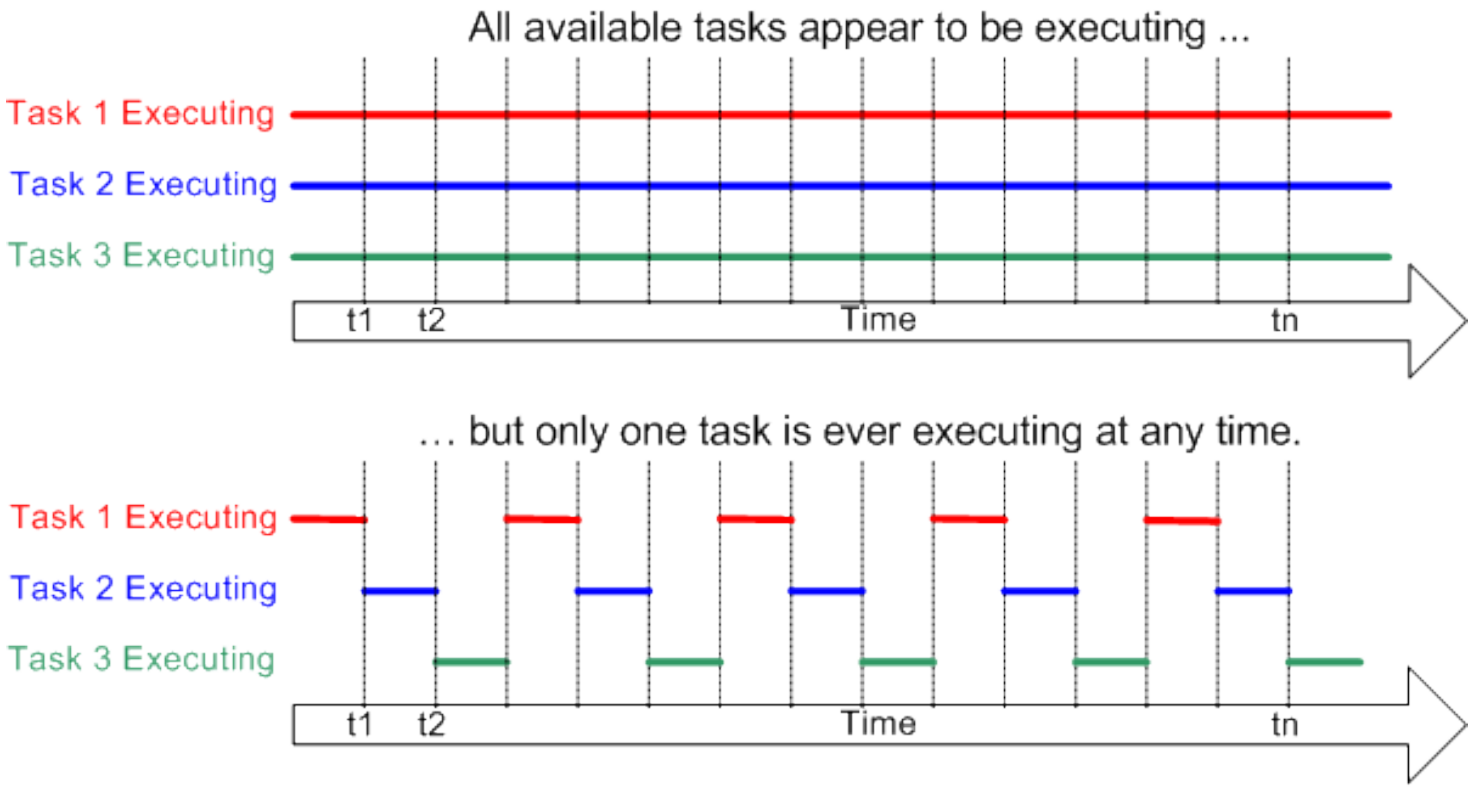
Interactive Processing



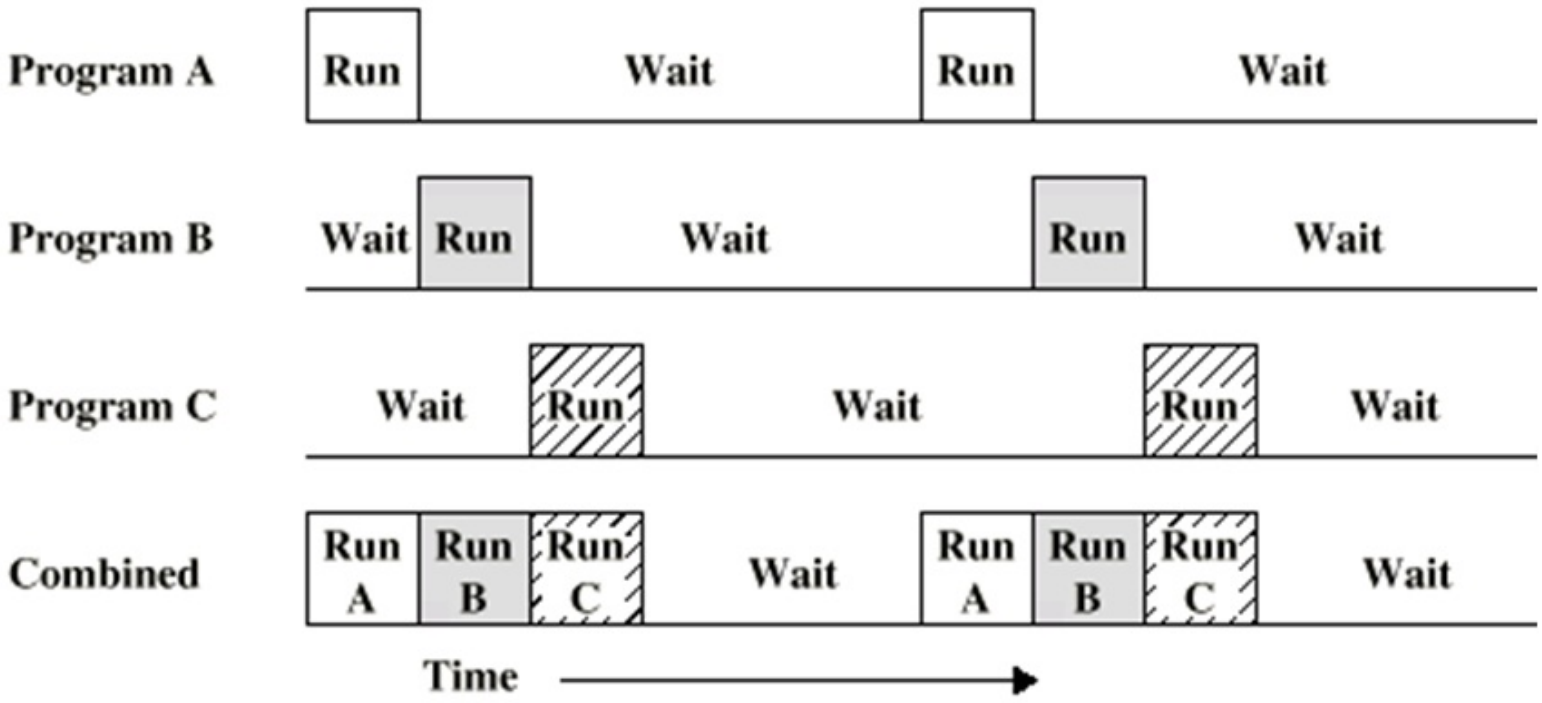
Multitasking

- Computers in the 1960s and 1970s were expensive, so each machine had to serve more than one user.
  - **Multiprogramming** in which time is divided into intervals and then the execution of each job is restricted to only one interval at a time. (**Timesharing**)
  - **Multitasking** refers to one user executing numerous tasks simultaneously.
- Advanced operating systems have
  - **Load balancing:** dynamically allocating tasks to the various processors so that all processors are used efficiently.
  - **Scaling:** breaking tasks into a number of subtasks compatible with the number of processors available.

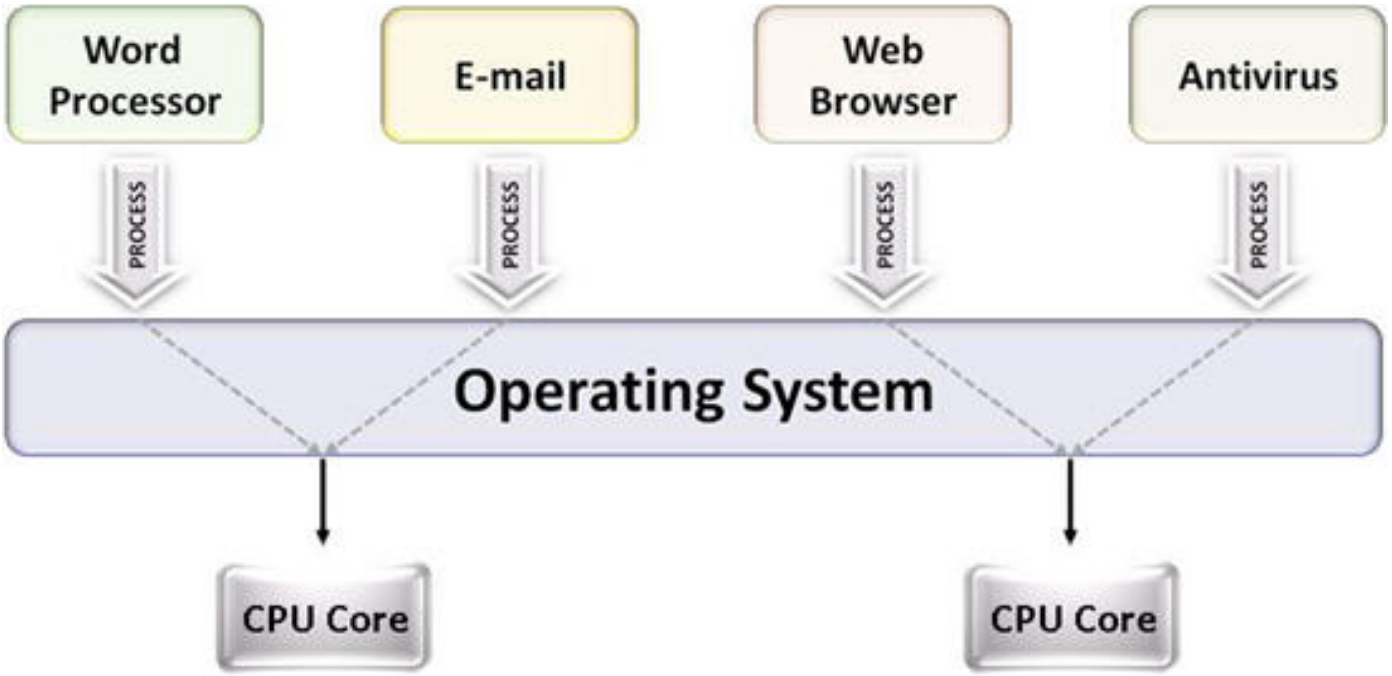
Multiprogramming



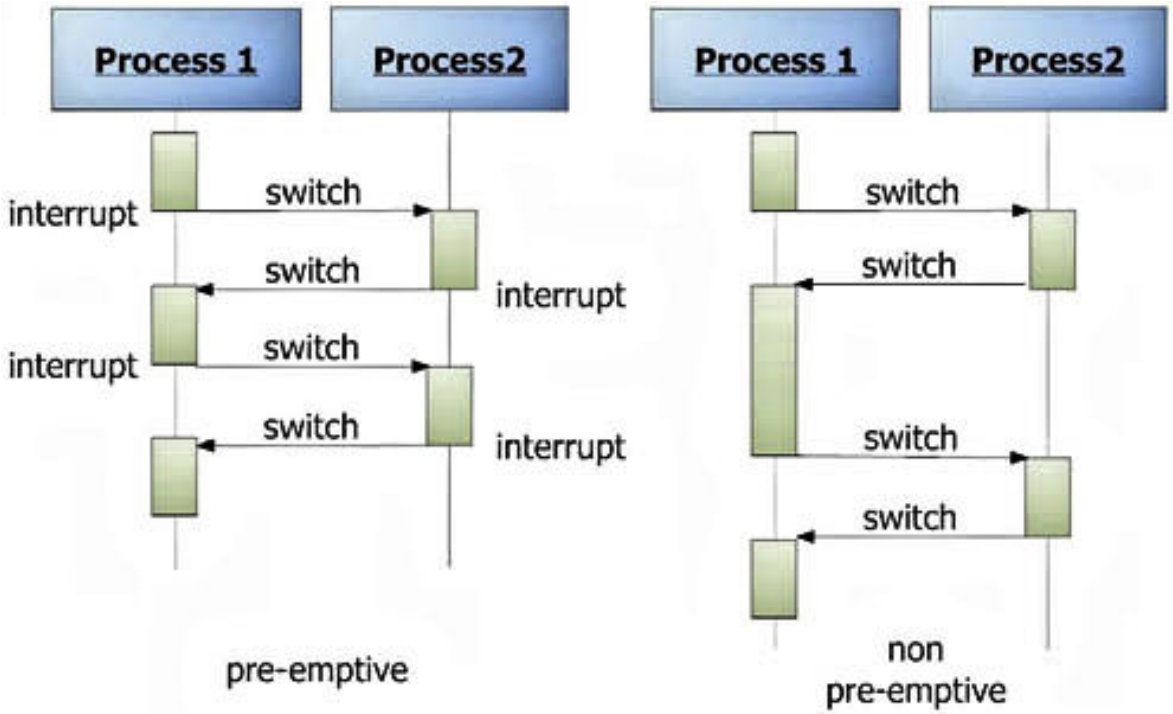
Multiprogramming



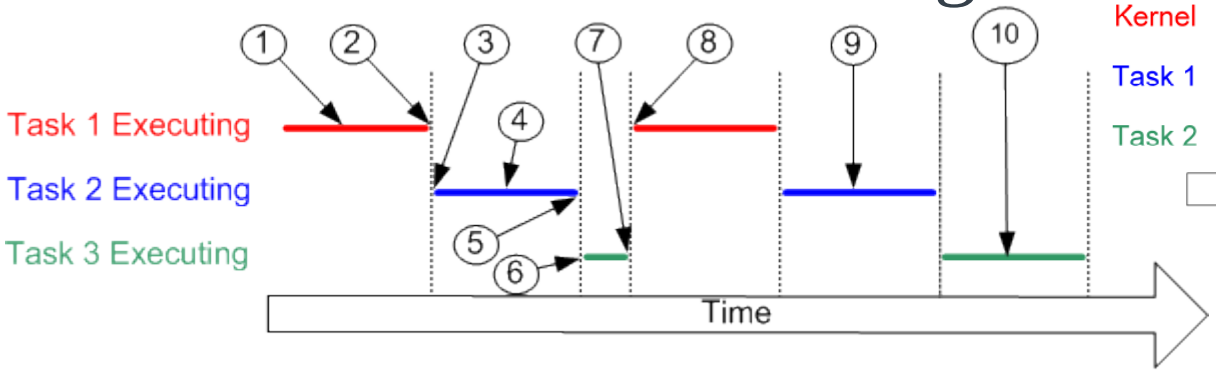
Multitasking

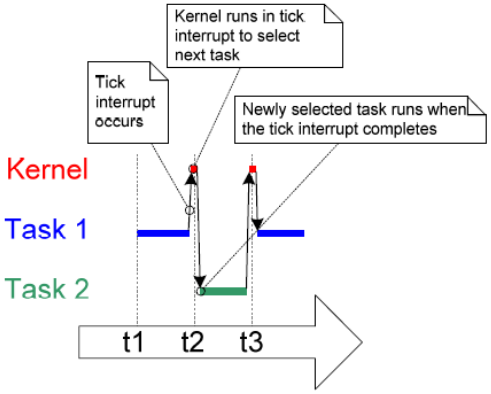


Preemptive vs Non-preemptive



Events of context switching





1. Task 1 is executing.
2. The kernel suspends (swaps out) task 1. (With some required time overhead)
3. Resumes task 2.
4. While task 2 is executing, it locks a processor peripheral for its own exclusive access.
5. The kernel suspends task 2.
6. Resumes task 3.
7. Task 3 tries to access the same processor peripheral, finding it locked task 3 cannot continue so suspends itself at.
8. The kernel resumes task 1.
9. …
10. The next time task 2 is executing (9) it finishes with the processor peripheral and unlocks it.
11. The next time task 3 is executing (10) it finds it can now access the processor peripheral and this time executes until suspended by the kernel.

Example: OMP

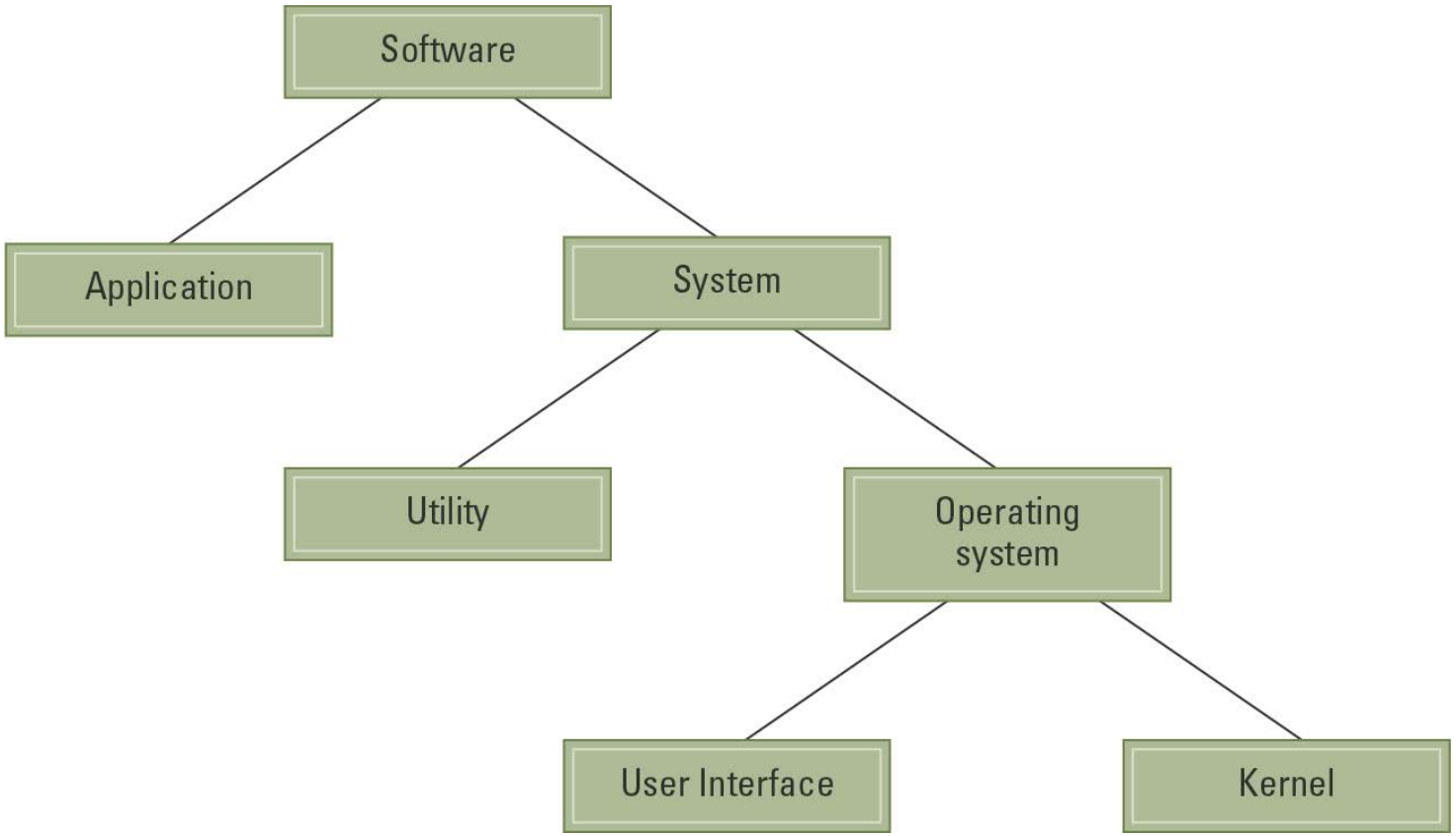
```
#include <stdio.h>
#include <omp.h>

int main( void )
{
  int accumulator = 0;
  int i;
  #pragma omp parallel for
  for( i = 0; i < 100000; i++ )
  {
    #pragma omp atomic
    accumulator++;
    printf( "%d  ", i );
  }
  printf( "\nTotal numbers = %ld\n", accumulator );
}
```

Types of Software

- Application software
  - Performs specific tasks for users (productivity, games, software development)
- System software
  - Provides infrastructure for application software
  - Consists of operating system and utility software

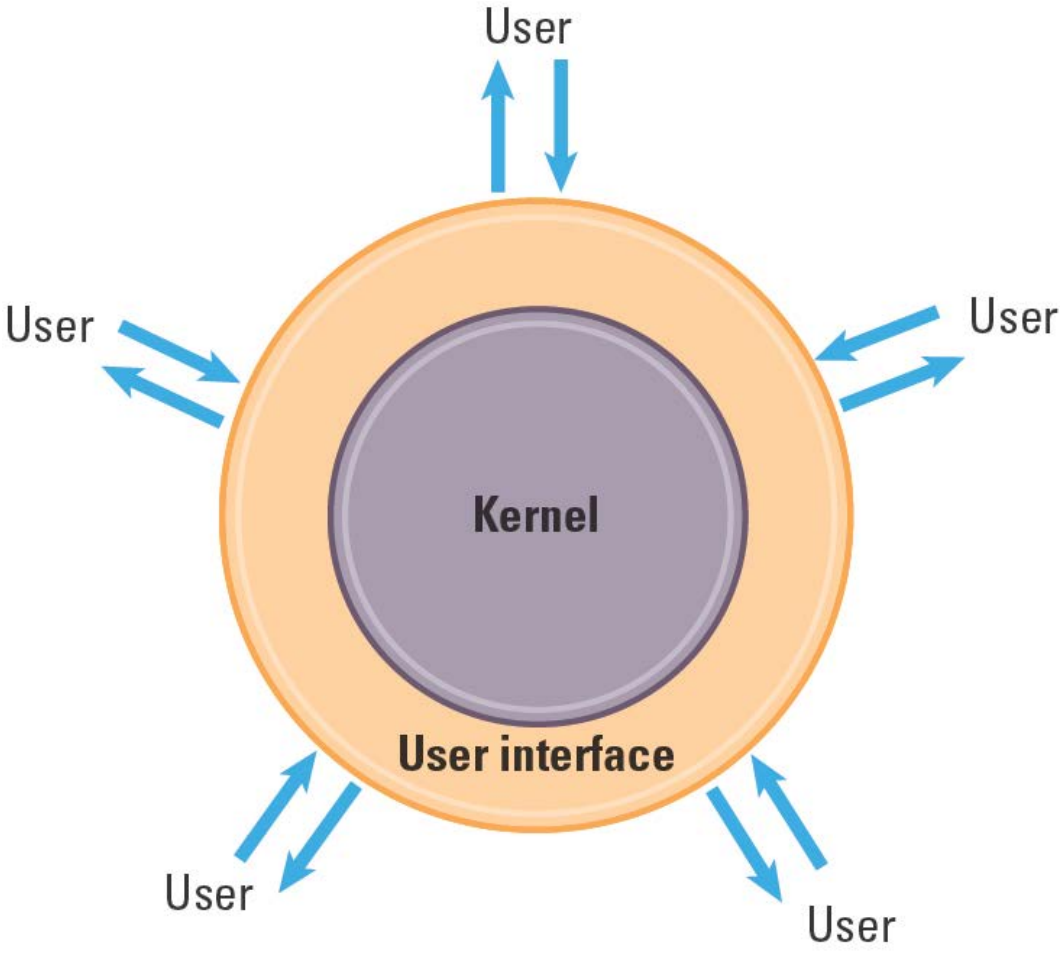
Software Classification



Operating System Components

- **User Interface**: Communicates with users
  - Text based (Shell)
  - Graphical user interface (GUI)
- **Kernel**: Performs basic required functions
  - File manager
  - Device drivers
  - Memory manager
  - Scheduler and dispatcher

The UI Acts as an Intermediary Between Users and the OS Kernel



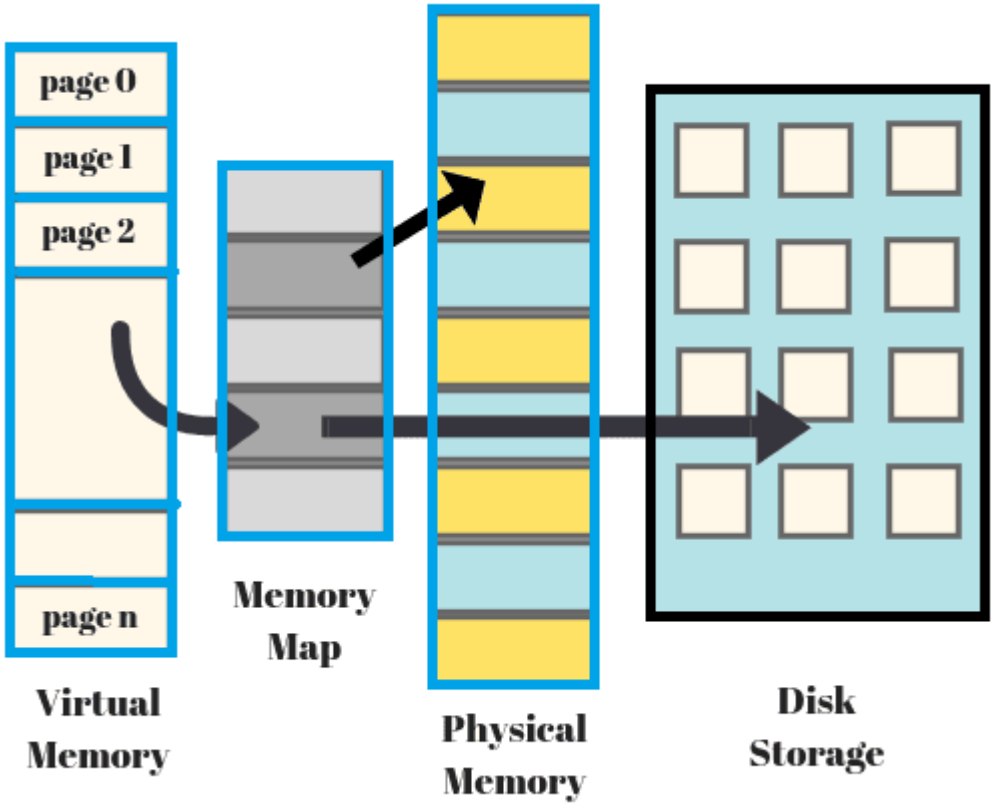
File Manager

- **Directory** (or **Folder**): A user-created bundle of files and other directories (subdirectories)
- **Directory Path:** A sequence of directories within directories

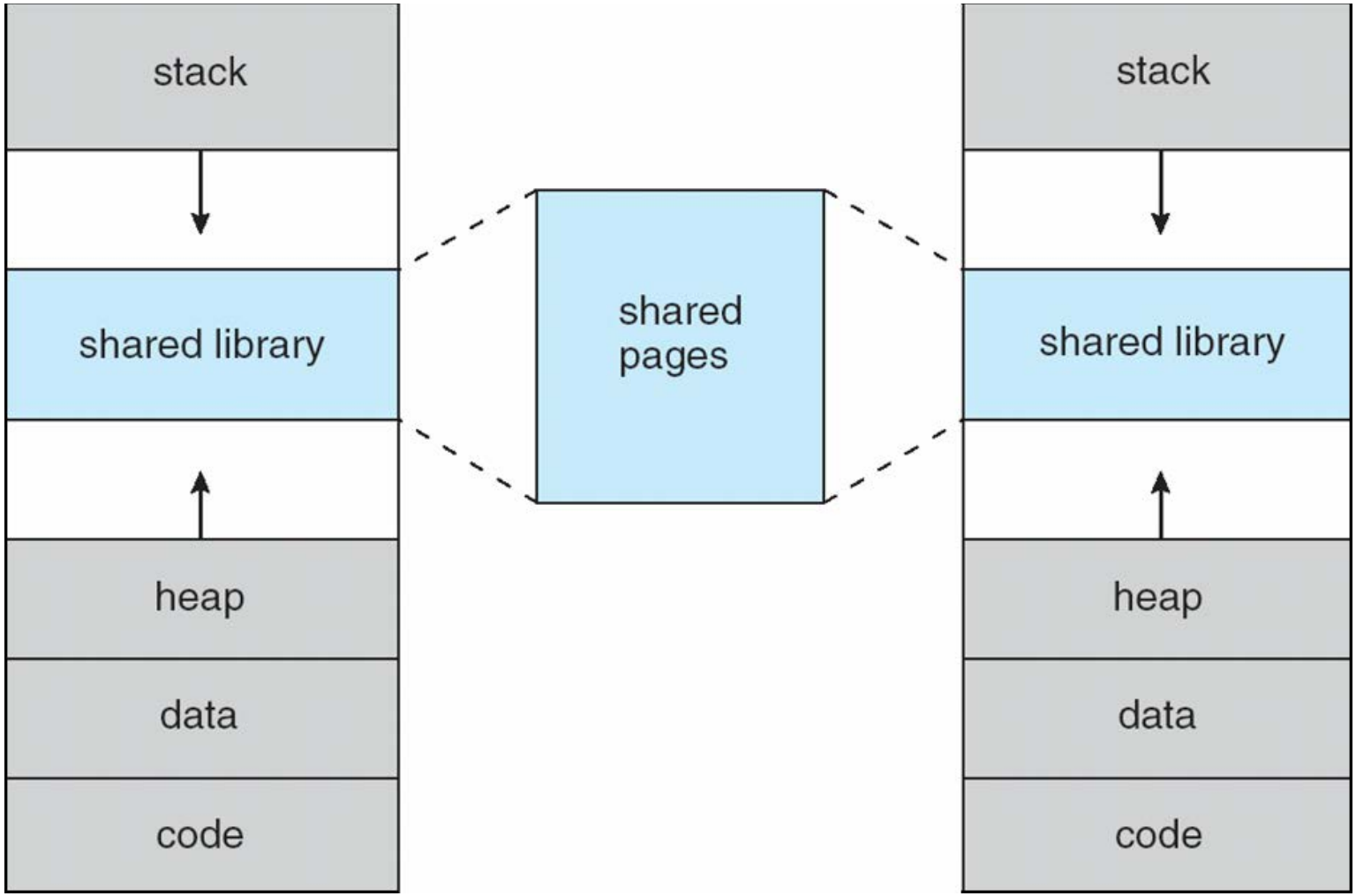
Memory Manager

- Allocates space in main memory.
- May create the illusion that the machine has more memory than it actually does (**virtual memory**) by playing a “shell game” in which blocks of data (**pages**) are shifted back and forth between main memory and mass storage.

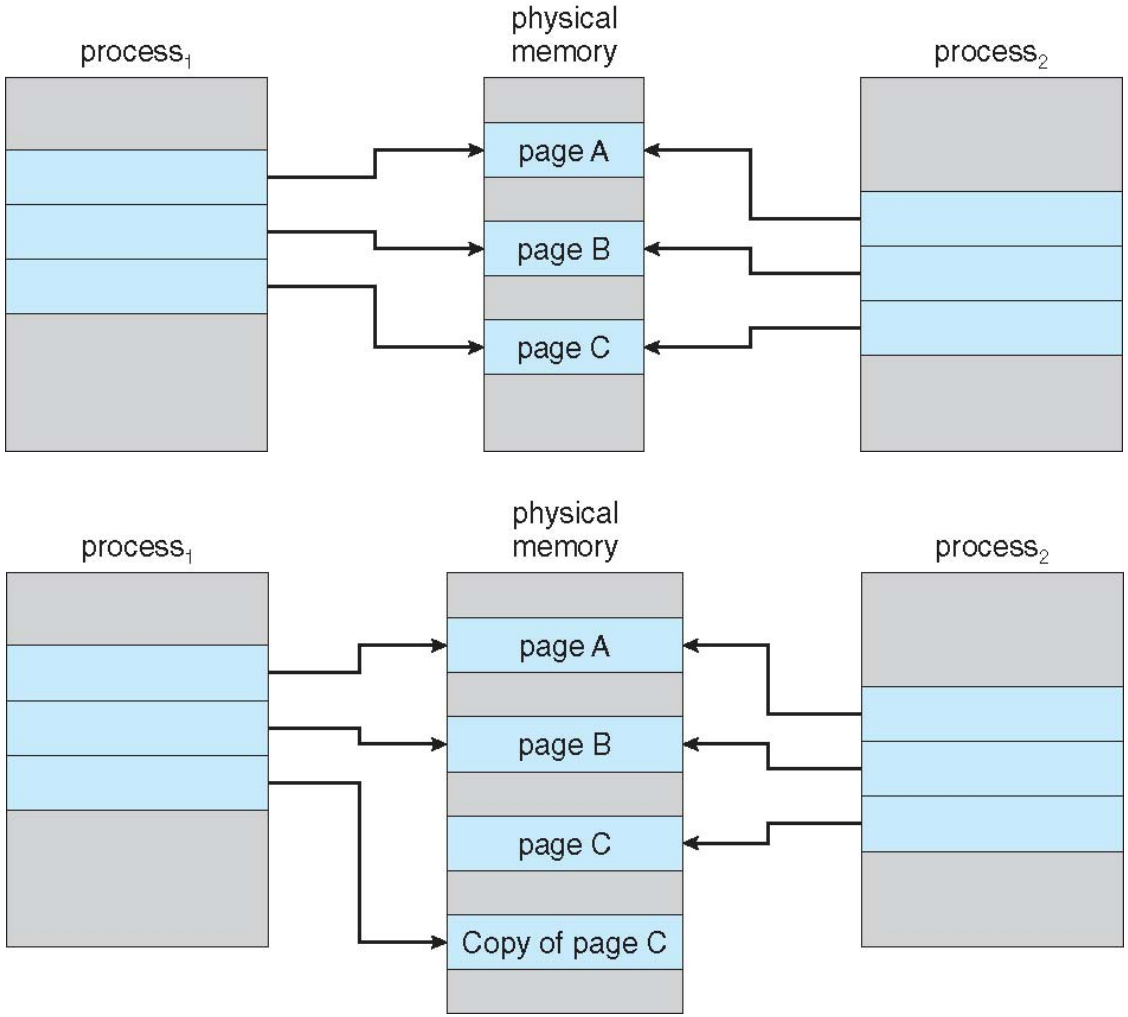
Virtual memory systems



Shared library using virtual memory

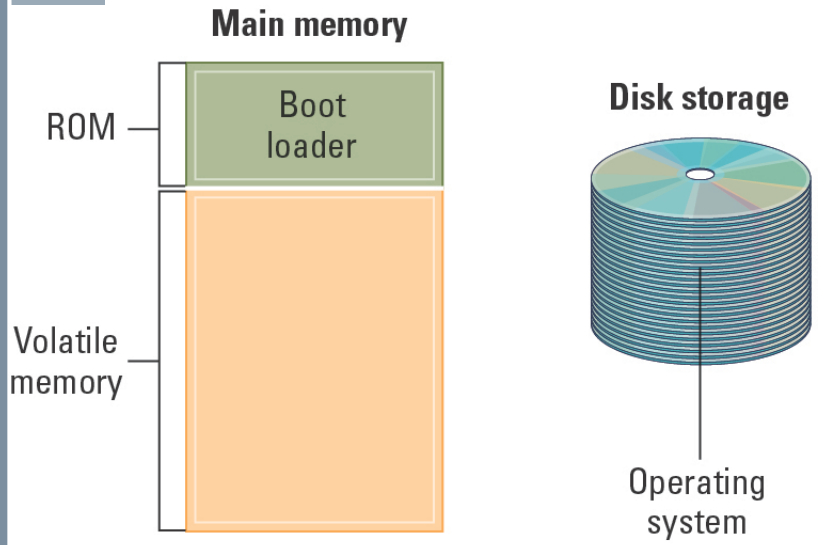


Shared page conflicts

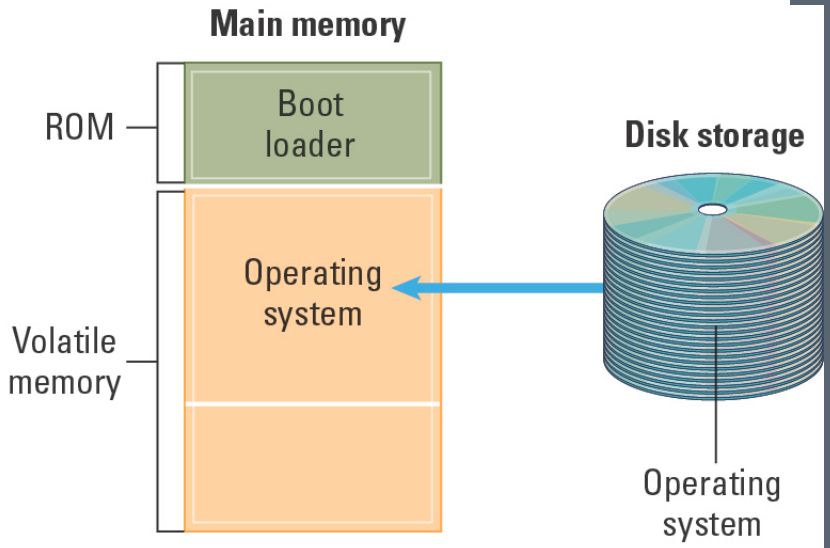


Getting it Started (Bootstrapping)

- **Boot loader:** Program in ROM (example of **firmware**)
  - Run by the CPU when power is turned on.
  - Transfers operating system from mass storage to main memory.
  - Executes jump to operating system.



Step 1: Machine starts by executing the boot loader program already in memory. Operating system is stored in mass storage.



Step 2: Boot loader program directs the transfer of the operating system into main memory and then transfers control to it.

Coordinating the Machine's Activities

- An operating system coordinates the execution of application software, utility software, and units within the operating system itself.

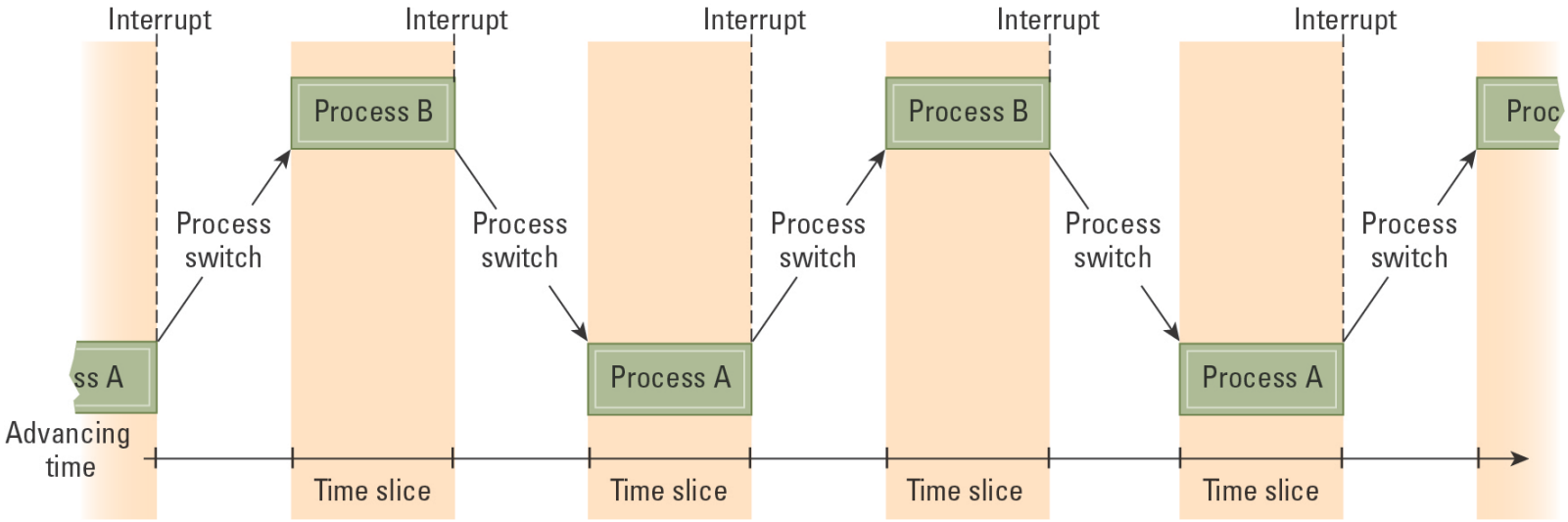
Processes

- **Process:** The activity of executing a program.
- **Process State:** Current status of the activity.
  - Program counter
  - General purpose registers
  - Related portion of main memory

Process Administration

- **Scheduler:** Adds new processes to the process table and removes completed processes from the process table.
- **Dispatcher:** Controls the allocation of time slices to the processes in the process table.
  - The end of a time slice is signaled by an interrupt.
  - Context switching happens then.

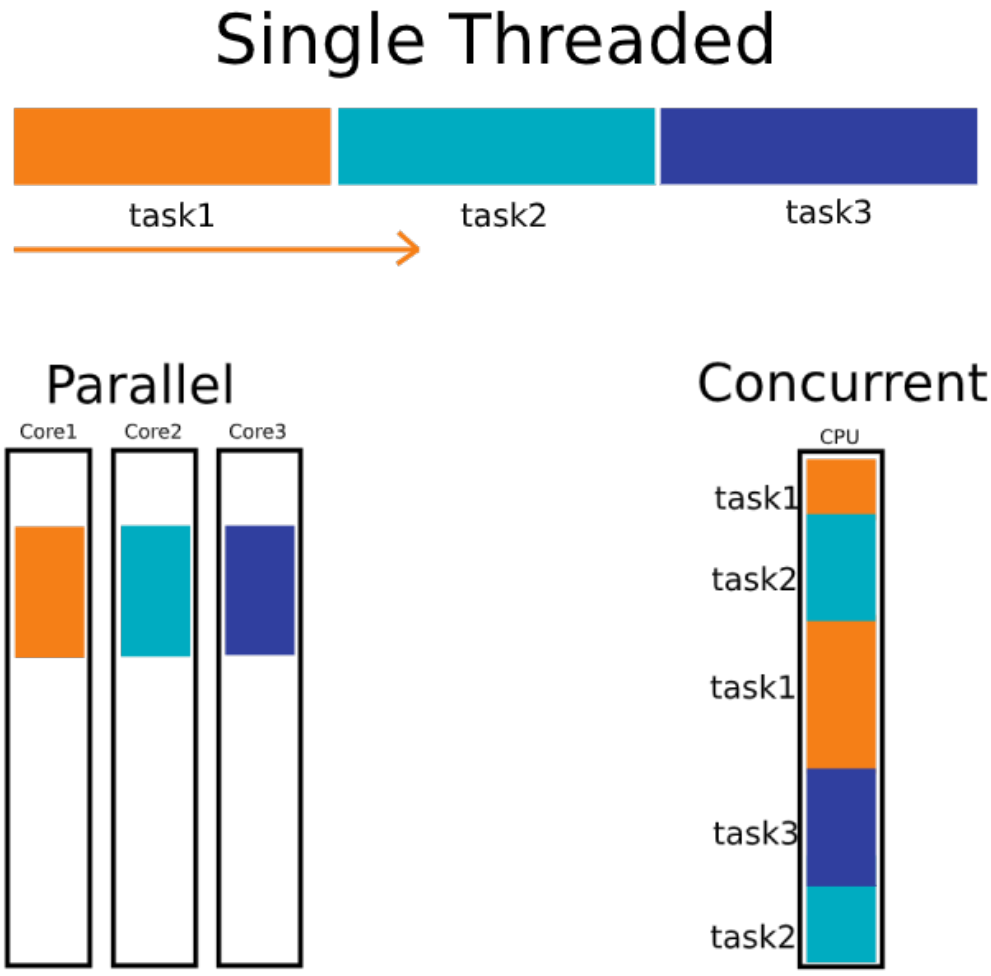
Time-sharing Between Process A and Process B



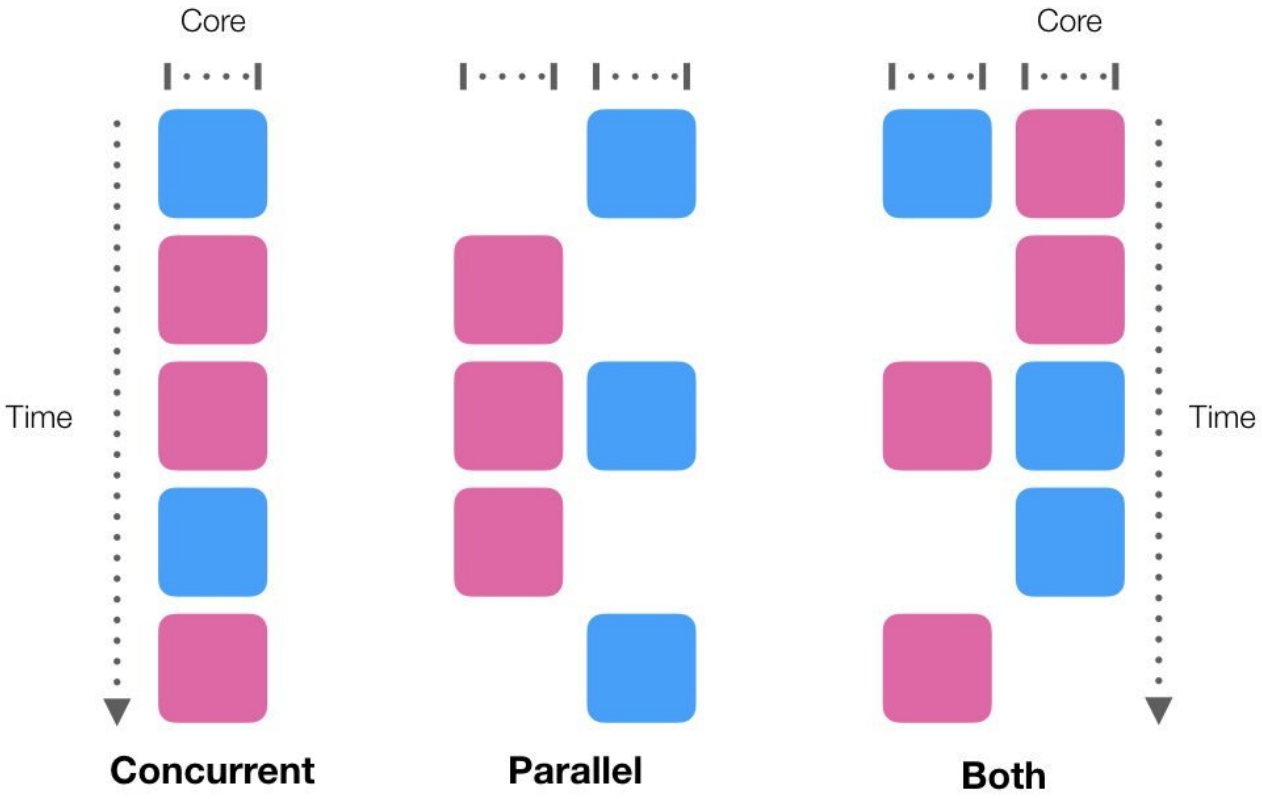
Handling Competition for Resources

- **Semaphore:** A “control flag”.
- **Critical Region:** A group of instructions that should be executed by only one process at a time.
- **Mutual exclusion:** Requirement for proper implementation of a critical region.

Parallelism vs Concurrency



Parallelism vs Concurrency



Deadlock

- Processes block each other from continuing because each is waiting for a resource that is allocated to another.
- Conditions required for deadlock:
  1. Competition for non-sharable resources.
  2. Resources requested on a partial basis.
  3. An allocated resource can not be forcibly retrieved.

Deadlock

- Resource sharing
  - Memory management and processor sharing
- Many programs competing for limited resources
- Lack of **process synchronization** consequences
  - **Deadlock:** "deadly embrace"
    - Two or more jobs placed in HOLD state
    - Jobs waiting for unavailable vital resource
    - System comes to standstill
    - Resolved via external intervention
  - **Starvation**
    - Infinite postponement of job

Deadlock

- More serious than starvation
- Affects entire system
  - Affects more than one job
    - Not just a few programs
  - All system resources become unavailable
- Example: traffic jam
- More prevalent in interactive systems
- Real-time systems
  - Deadlocks quickly become critical situations
- No simple and immediate solution

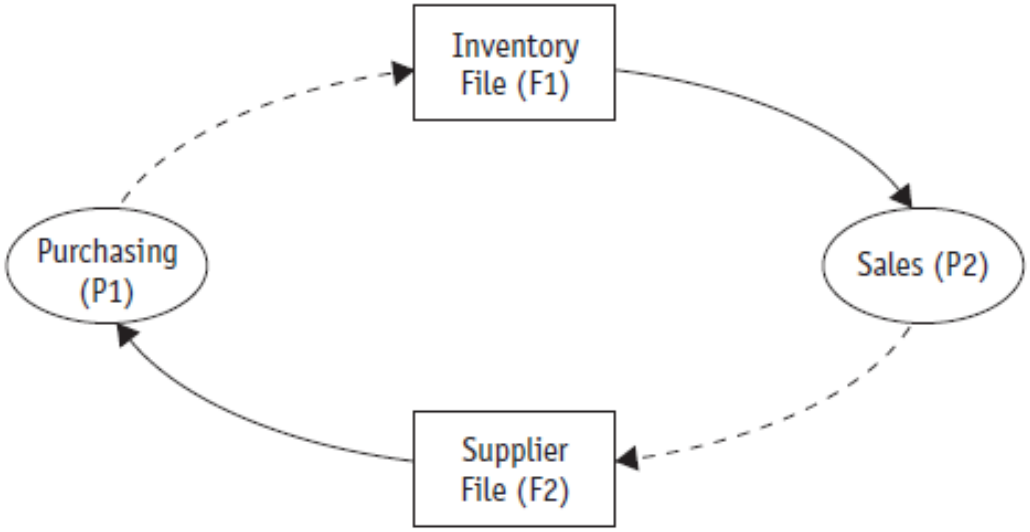
Seven Cases of Deadlock

- Nonsharable/nonpreemptable resources
  - Allocated to jobs requiring same type of resources
- Resource types locked by competing jobs
  - File requests
  - Databases
  - Dedicated device allocation
  - Multiple device allocation
  - Spooling
  - Network
  - Disk sharing

Case 1: Deadlocks on File Requests

- Jobs request and hold files for execution duration
- Example
  - Two programs (P1, P2) and two files (F1, F2)
  - Deadlock sequence
    - P1 has access to F1 and also requires F2
    - P2 has access to F2 and also requires F1
  - Deadlock remains
    - Until one program withdrawn *or*
    - Until one program forcibly removed and file released
  - Other programs requiring F1 or F2
    - Put on hold for duration of situation

Case 1: Deadlocks on File Requests



(figure 5.2)

Case 1. These two processes, shown as circles, are each waiting for a resource, shown as rectangles, that has already been allocated to the other process, thus creating a deadlock.

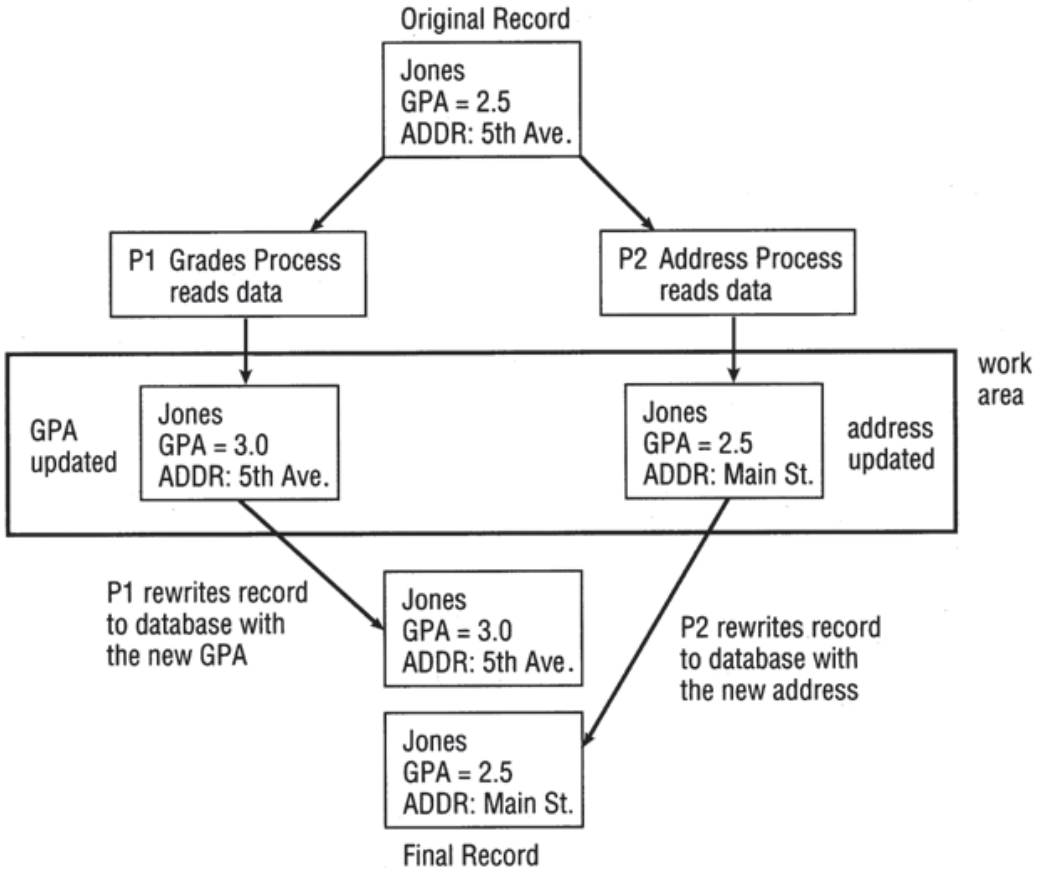
Case 2: Deadlocks in Databases

- Two processes access and lock database records
- **Locking**
  - Technique
    - One user locks out all other users
    - Users working with database
  - Three locking levels
    - Entire database for duration of request
    - Subsection of database
    - Individual record until request completed

Case 2: Deadlocks in Databases

- Example: two processes (P1 and P2)
  - Each needs to update two records (R1 and R2)
  - Deadlock sequence
    - P1 accesses R1 and locks it
    - P2 accesses R2 and locks it
    - P1 requests R2 but locked by P2
    - P2 requests R1 but locked by P1
- **Race between processes**
  - Results when locking not used
  - Causes incorrect final version of data
  - Depends on process execution order

Case 2: Deadlocks in Databases



(figure 5.3)

Case 2. P1 finishes first and wins the race but its version of the record will soon be overwritten by P2. Regardless of which process wins the race, the final version of the data will be incorrect.

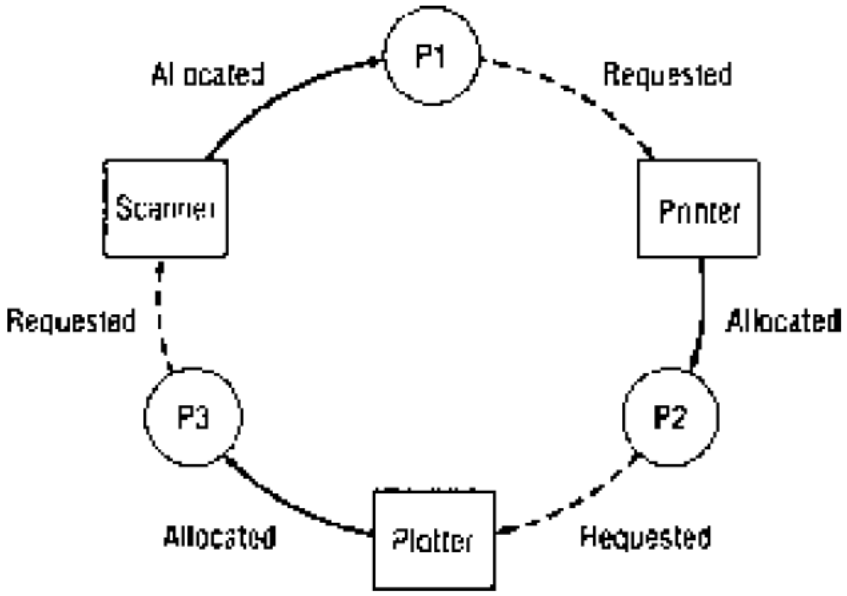
Case 3: Deadlocks in Dedicated Device Allocation

- Limited number of dedicated devices
- Example
  - Two programs (P1, P2)
    - Need two tape drives each
    - Only two tape drives in system
  - Deadlock sequence
    - P1 requests tape drive 1 and gets it
    - P2 requests tape drive 2 and gets it
    - P1 requests tape drive 2 but blocked
    - P2 requests tape drive 1 but blocked

Case 4: Deadlocks in Multiple Device Allocation

- Several processes request and hold dedicated devices
- Example
  - Three programs (P1, P2, P3)
  - Three dedicated devices (tape drive, printer, plotter)
  - Deadlock sequence
    - P1 requests and gets tape drive
    - P2 requests and gets printer
    - P3 requests and gets the plotter
    - P1 requests printer but blocked
    - P2 requests plotter but blocked
    - P3 requests tape drive but blocked

Case 4: Deadlocks in Multiple Device Allocation



(figure 5.4)

Case 4. Three processes, shown as circles, are each waiting for a device that has already been allocated to another process, thus creating a deadlock.

Case 5: Deadlocks in Spooling

- **Virtual device**
  - Dedicated device made sharable
  - Example
    - Printer: high-speed disk device between printer and CPU
- **Spooling**
  - Process
    - Disk accepts output from several users
    - Acts as temporary storage for output
    - Output resides in disk until printer accepts job data

Case 5: Deadlocks in Spooling

- Deadlock sequence
  - Printer needs all job output before printing begins
    - Spooling system fills disk space area
    - No one job has entire print output in spool area
    - Results in partially completed output for all jobs
    - Results in deadlock

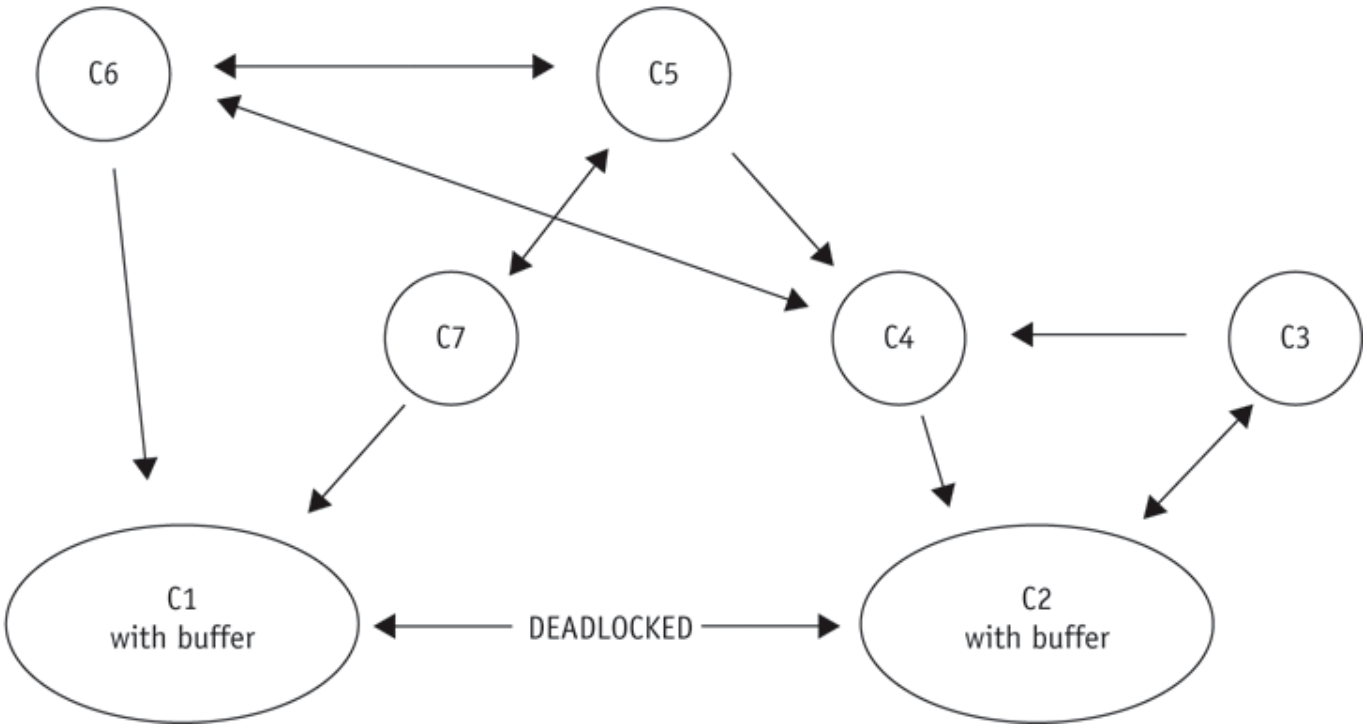
Case 6: Deadlocks in a Network

- No network protocols controlling network message flow
- Example
  - Seven computers on network
    - Each on different nodes
  - Direction of arrows
    - Indicates message flow
  - Deadlock sequence
    - All available buffer space fills

Case 6: Deadlocks in a Network (cont'd.)

(figure 5.5)

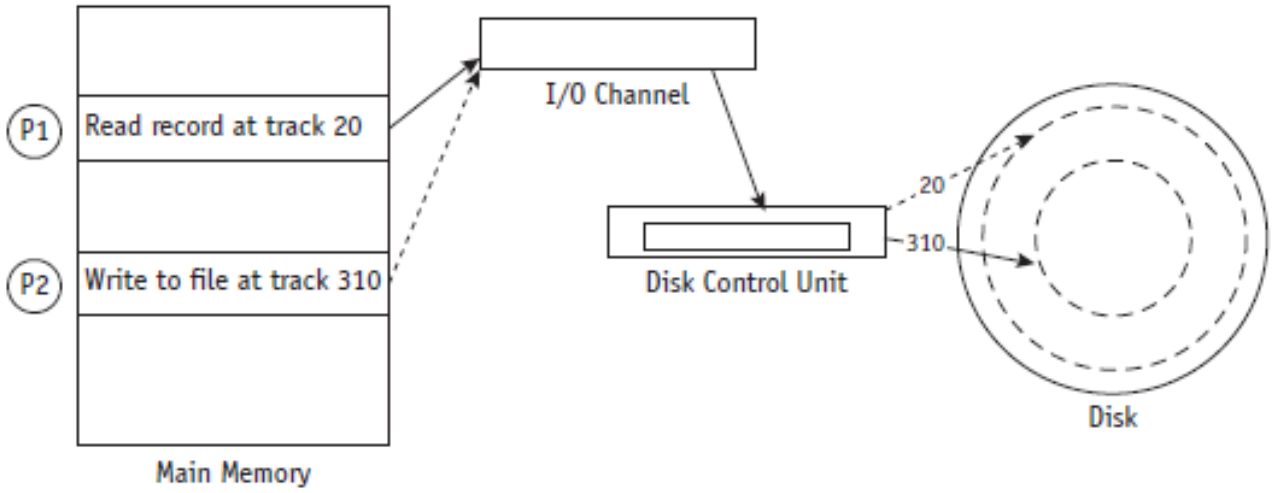
Case 6, deadlocked network flow. Each circle represents a node and each line represents a communication path. The arrows indicate the direction of data flow.



Case 7: Deadlocks in Disk Sharing

- Competing processes send conflicting commands
  - Scenario: disk access
- Example
  - Two processes
  - Each process waiting for I/O request
    - One at cylinder 20 and one at cylinder 310
  - Deadlock sequence
    - Neither I/O request satisfied
    - Device puts request on hold while attempting to fulfill other request for each request
  - **Livelock** results

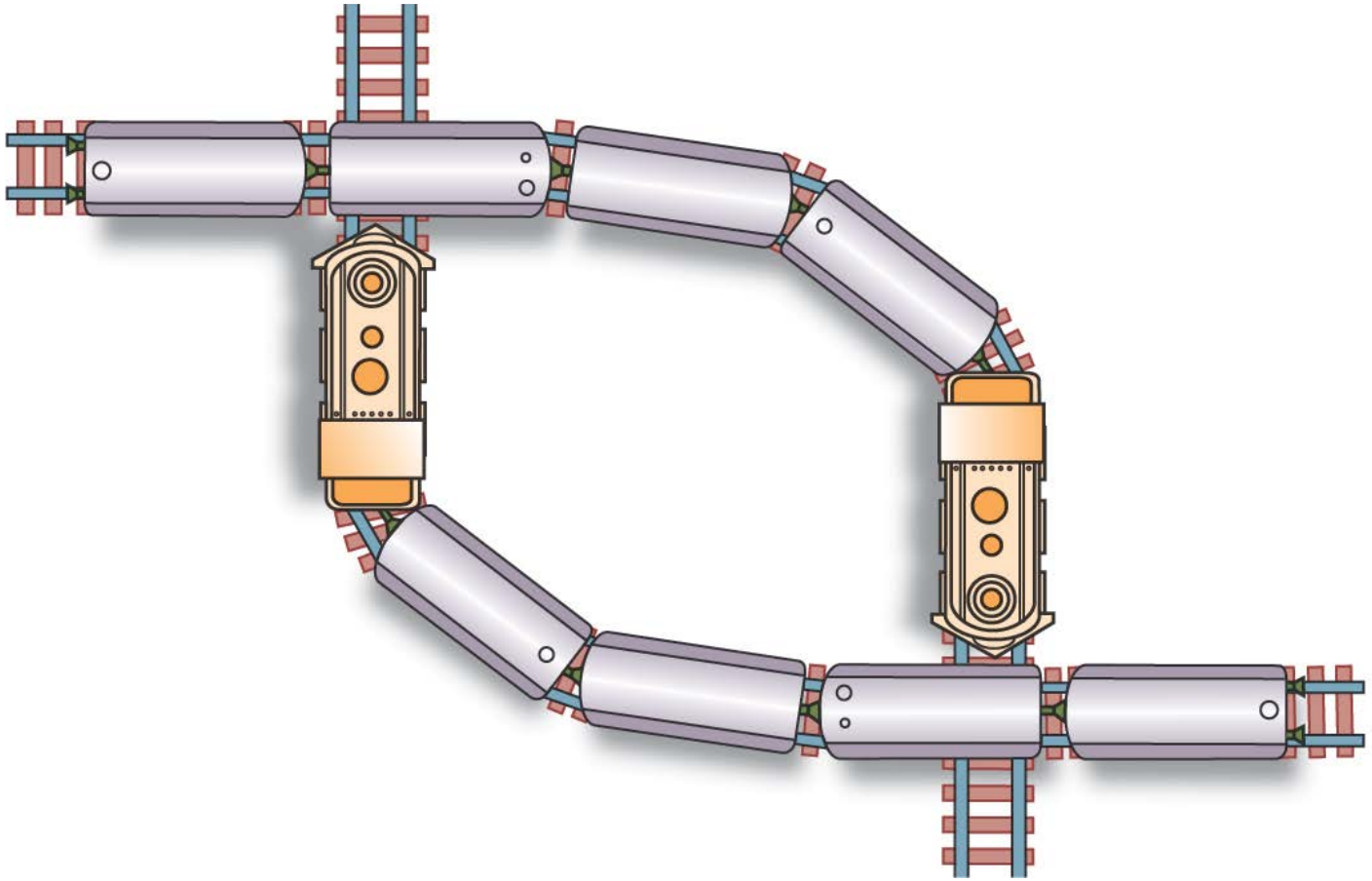
Case 7: Deadlocks in Disk Sharing (cont'd.)



(figure 5.6)

Case 7. Two processes are each waiting for an I/O request to be filled: one at track 20 and one at track 310. But by the time the read/write arm reaches one track, a competing command for the other track has been issued, so neither command is satisfied and livelock occurs.

A Deadlock Resulting from Competition for Nonshareable Railroad Intersections



Conditions for Deadlock

- Four conditions simultaneously occurring prior to deadlock or livelock
  - Mutual exclusion
  - Resource holding
  - No preemption
  - Circular wait
- All needed by operating system
  - Must recognize simultaneous occurrence of four conditions
- Resolving deadlock
  - Removal of one condition

Avoiding Deadlocks

- **Mutual Exclusion:** Resources shared such as read-only files do not lead to deadlocks but resources, such as printers and tape drives, requires exclusive access by a single process.
- **Hold and Wait:** In this condition processes must be prevented from holding one or more resources while simultaneously waiting for one or more others.
- **No Preemption:** Preemption of process resource allocations can avoid the condition of deadlocks, where ever possible.
- **Circular Wait:** Circular wait can be avoided if we number all resources, and require that processes request resources only in strictly increasing(or decreasing) order.

Handling Deadlocks

- **Preemption**: We can take a resource from one process and give it to other. This will resolve the deadlock situation, but sometimes it does causes problems.
- **Rollback**: In situations where deadlock is a real possibility, the system can periodically make a record of the state of each process and when deadlock occurs, roll everything back to the last checkpoint, and restart, but allocating resources differently so that deadlock does not occur.
- Kill one or more processes: This is the simplest way, but it works.

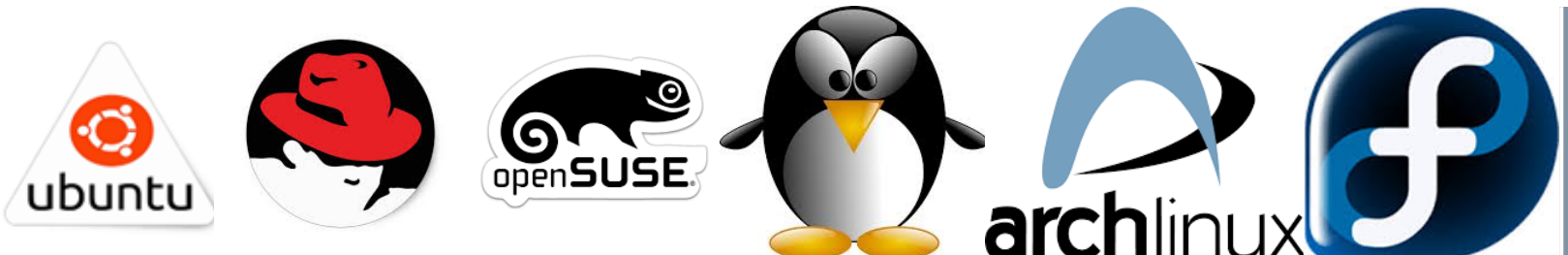
Starvation

- Job execution prevented
  - Waiting for resources that never become available
  - Results from conservative resource allocation
- Example
  - “The dining philosophers” by Dijkstra
- Starvation avoidance
  - Implement algorithm tracking how long each job waiting for resources (aging)
  - Block new jobs until starving jobs satisfied

Security

- Attacks from outside:
  - Problems
    - Insecure passwords
    - Sniffing software
  - Counter measures
    - Auditing software
- Attacks from within
  - Problem: Unruly processes.
  - Counter measures: Control process activities via privileged modes and privileged instructions.

Introduction to Linux



Overview

- What is Unix/Linux?
- History of Linux.
- Features Supported Under Linux.
- The future of Linux.

Before Linux

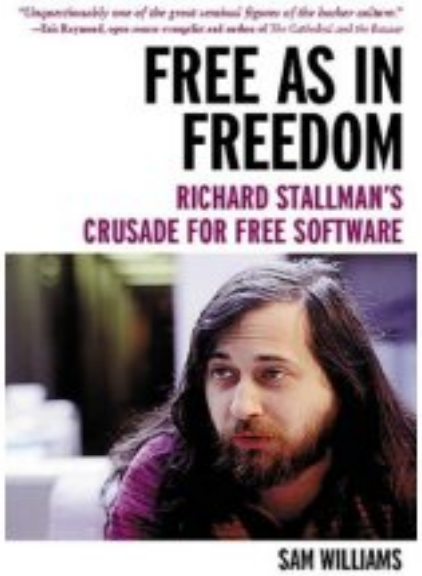
- In 80's, Microsoft's DOS was the dominated OS for PC.
  - Apple MAC was better, but expensive. (Some people's pov)
- UNIX was much better, but much, much more expensive. Only for minicomputer for commercial applications.
- People was looking for a UNIX based system, which is cheaper and can run on PC.
- Both DOS, MAC and UNIX were proprietary, i.e., the source code of their kernel is protected.
- No modification is possible without paying high license fees.

Unix Family

- ***Linux***
- System **V** Release 4 (**SVR**4), developed by **AT&T** (now owned by the **SCO** Group);
- the 4.4 **BSD** release from the University of California at Berkeley (4.4**BSD**);
- Digital **Unix** from Digital Equipment Corporation (now Hewlett-Packard);
- **AIX** from **IBM**;
- **HP-UX** from Hewlett-Packard;
- **Solaris** from Sun Microsystems;
- **Mac OS X** from Apple Computer, Inc.

GNU Project

- Established in 1984 by Richard Stallman, who believes that software should be free from restrictions against copying or modification in order to make better and efficient computer programs.



GNU is a recursive acronym for "GNU's Not Unix"

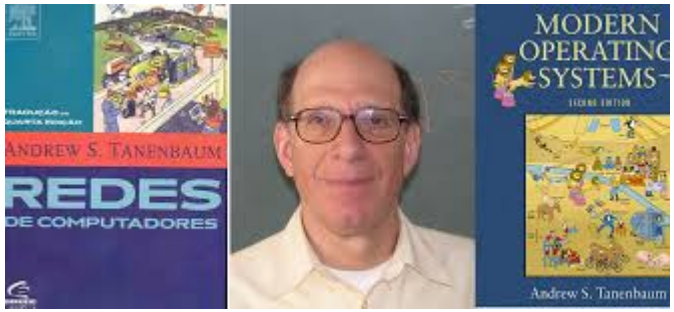
Aim at developing a complete Unix-like operating system which is free for copying and modification

Companies make their money by maintaining and distributing the software, e.g. optimally packaging the software with different tools (Redhat, Slackware, Mandrake, SuSE, etc)

Stallman built the first free GNU C Compiler in 1991. But still, an OS was yet to be developed.

Beginning of Linux

- A famous professor Andrew Tanenbaum developed Minix, a simplified version of UNIX that runs on PC.
  - Minix is for class teaching only.
  - No intention for commercial use.
- In Sept 1991, Linus Torvalds, a second year student of Computer Science at the University of Helsinki, developed the preliminary kernel of Linux, known as Linux version 0.0.1



PK!

- Message from Professor Andrew Tanenbaum:

  " I still maintain the point that designing a monolithic kernel in 1991 is a fundamental error. Be thankful you are not my student. You would not get a high grade for such a design :-)“

- Andrew Tanenbaum to Linus Torvalds.

- Soon more than a hundred people joined the Linux camp.
  - Then thousands.
  - Then hundreds of thousands.
- It was licensed under GNU General Public License, thus ensuring that the source codes will be free for all to copy, study and to change.

Linux Today

- Linux has been used for many computing platforms
  - PC, Supercomputer, Switches, Routers, Handheld devices, Micro devices, Name card computers…
- Not only character user interface but graphical user interface is available.
- Commercial vendors moved in Linux itself to provide freely distributed code.
  - They make their money by compiling up various software and gathering them in a distributable format.

Linux has Many Distributions

GNU (Linux) Operating System

Linux Kernel

+

system programs (e.g. compilers, loaders, linkers, and shells)

+

system utilities (commands)

+

libraries

+

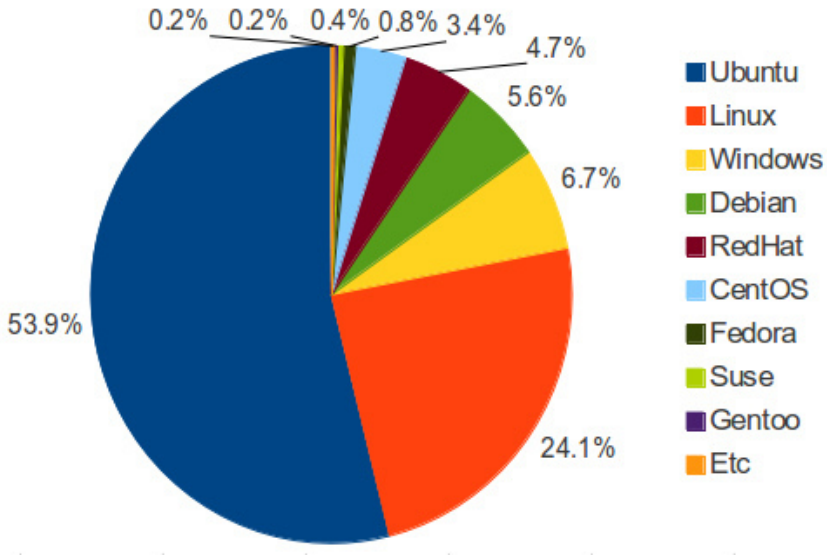
graphical desktops (e.g. **X** windows).

Operating System Objectives

- Interact with the hardware components, servicing all low-level programmable elements included in the hardware platform.
  - In a modern OS like Linux, the above functionality is provided by the Linux kernel.
  - A user program can not directly operate on a hardware.
- Provide an execution environment to the applications that run on the computer system (the so-called user programs).

Growing and Growing

- In order to encourage wide dissemination of his OS, Linus made the source code open to public.
- At the end of 1992 there were about a hundred Linux developers. Next year there were 1000.
  - The numbers multiplied every year.



The estimation of linux users

At this moment, there are

597,509

users and

164,472

machines registered.
My guess at the number of Linux users:

92,950,269

World population: 7,522,443,132
Internet users: 3,645,108,605

Linux – Free Software

- Free software, as defined by the FSF (Free Software Foundation), is a "matter of liberty, not price."
- To qualify as free software by FSF standards, you must be able to:
  - Run the program for any purpose you want to, rather than be restricted in what you can use it for.
  - View the program's source code.
  - Study the program's source code and modify it if you need to.
  - Share the program with others.
  - Improve the program and release those improvements so that others can use them.

Hardware Dependency (1)

- Linux supports a broad range of platforms and hardware.
  - alpha
    - Hewlett-Packard's Alpha workstations
  - arm
    - ARM processor-based computers and embedded devices
  - cris
    - "Code Reduced Instruction Set" CPUs used by Axis in its thin-servers, such as web cameras or development boards

Hardware Dependency (2)

- i386
  - IBM-compatible personal computers based on 80 x 86 microprocessors
- ia64
  - Workstations based on Intel 64-bit Itanium microprocessor
- m68k
  - Personal computers based on Motorola MC680 x 0 microprocessors
- mips
  - Workstations based on MIPS microprocessors
- mips64
  - Workstations based on 64-bit MIPS microprocessors

Hardware Dependency (3)

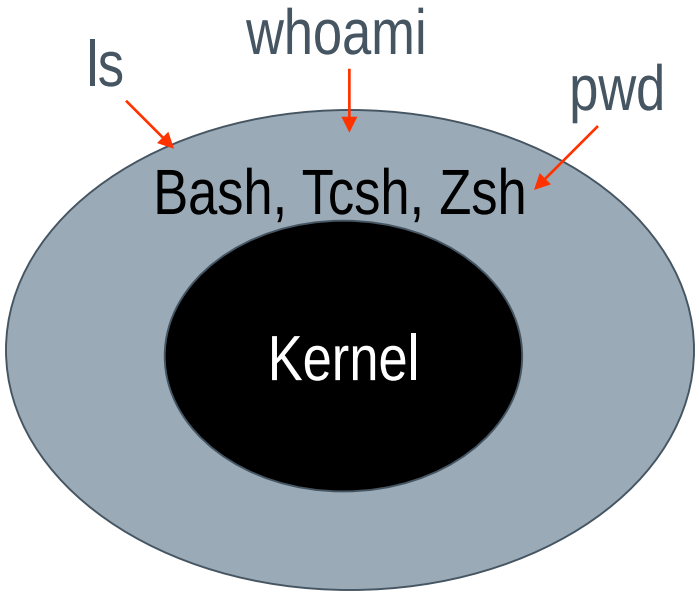
- parisc
  - Workstations based on Hewlett Packard HP 9000 PA-RISC microprocessors
- ppc
  - Workstations based on Motorola-IBM PowerPC microprocessors
- s390
  - 32-bit IBM ESA/390 and zSeries mainframes
- s390 x
  - IBM 64-bit zSeries servers
- sh
  - SuperH embedded computers developed jointly by Hitachi and STMicroelectronics
- sparc
  - Workstations based on Sun Microsystems SPARC microprocessors
- sparc64
  - Workstations based on Sun Microsystems 64-bit Ultra SPARC microprocessors

Licenses that Govern Linux and Free and Open-source Software

- There is a difference between free software and open-source software.
  - Both use the same license—the GPL.
  - Free software helps users focus on freedom and ethics.
  - Open-source software is related to the efficiencies gained by the free-software approach to development.

Linux Shells

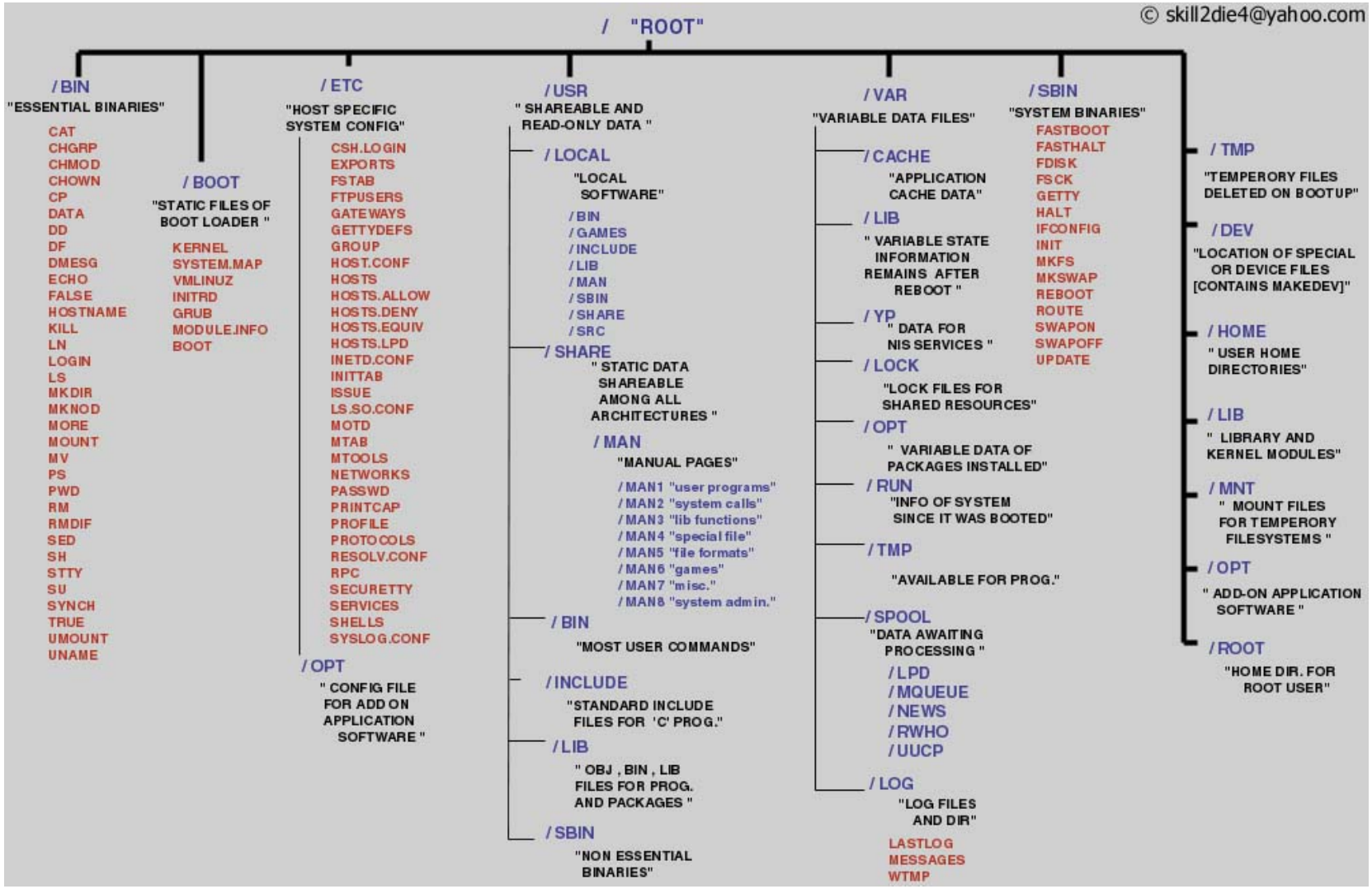
- Shell interprets the command and request service from kernel.
- Similar to DOS but DOS has only one set of interface while Linux can select different shell.
  - Bourne Again shell (Bash), TC shell (Tcsh), Z shell (Zsh)
- Different shell has similar but different functionality.
- Bash is the default for Linux.
- Graphical user interface of Linux is in fact an application program work on the shell.



Directory Tree

- When you log on the Linux OS using your username you are automatically located in your home directory.
- The *Nix (Unix or Linux) file system is a hierarchical directory structure
- The structure resembles an upside down tree.
- Directories are collections of files and other directories. The structure is recursive with many levels.
- Every directory has a parent except for the root directory.
- Many directories have children directories.
- Unlike Windows, with multiple drives and multiple file systems, a *Nix system only has ONE file system.
- The Linux Standard Base (LSB) specifies the structure of a Linux file system.

Directory Tree



Some Important Subdirectories

- **/bin**: Important Linux commands available to the average user.
- **/boot**: The files necessary for the system to boot. Not all Linux distributions use this one. Fedora does.
- **/dev**: All device drivers. Device drivers are the files that your Linux system uses to talk to your hardware. For example, there's a file in the /dev directory for your particular make and model of monitor, and all of your Linux computer's communications with the monitor go through that file.
- **/etc**: System configuration files.
- **/home**: Every user except root gets her own folder in here, named for her login account. So, the user who logs in with linda has the directory /home/linda, where all of her personal files are kept.
- **/lib**: System libraries. Libraries are just bunches of programming code that the programs on your system use to get things done.

Some Important Subdirectories

- **/mnt**: Mount points. When you temporarily load the contents of a CD-ROM or USB drive, you typically use a special name under /mnt. For example, many distributions (including Fedora) come, by default, with the directory /mnt/cdrom, which is where your CD-ROM drive's contents are made accessible.
- **/root**: The root user's home directory.
- **/sbin**: Essential commands that are only for the system administrator.
- **/tmp**: Temporary files and storage space. Don't put anything in here that you want to keep. Most Linux distributions (including Fedora) are set up to delete any file that's been in this directory longer than three days.
- **/usr**: Programs and data that can be shared across many systems and don't need to be changed.
- **/var**: Data that changes constantly (log files that contain information about what's happening on your system, data on its way to the printer, and so on).

Home Directory

- This is where you put your own stuff.
  - Linux is a multiuser, multiprocessing machine.
  - Resource is shared among all.

More Linux

- Labs will be conducted after midterm.