Operating Systems Design

Dr. Jerry Shiao, Silicon Valley University

Course Description

- Class Evaluation:
  - 25 %: Midterm. First half material.
  - 25 %: Final. Second half material.
  - 20 %: Homework
  - 10 %: Attendance / Class Participation
- Case Study: Linux Operating System
  - 20 %: Case Study Assignments.
  - Raspberry Pi SOC ARMV6 Core. 512 MByte RAM, 4/8/16 GigByte SD Card, HDMI, GPU, network and USB ports.
    - Raspbian Linux, Debian distribution and optimized for Raspberry Pi.
- References:
  - Operating Systems Concepts 8th Edition, Silberschatz, Galvin, Gagne, 2009 John Wiley & Sons, Inc.
  - Operating Systems Internals and Design Principles 6th Edition, Stallings, 2009 Pearson Education, Inc.
  - http://class.svuca.edu/~sau/class/CS500/
    - Lecture PowerPoint Slides
    - Homework Assignments
    - Case Study Assignments
- Email: sau@svuca.edu

Course Description: Raspberry Pi

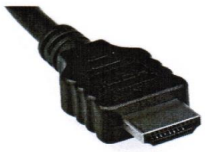
http://www.adafruit.com/raspberrypi

1) Hardware and Software Accessories for Raspberry PI.
2) Raspberry PI Model B 512 Mbytes
3) Raspberry PI Model B+ 512 Mbytes

To set up your Raspberry Pi you will need:

	Item	Minimum recommended specification & notes
1	SD card	• Minimum size 4Gb; class 4 (the *class* indicates how fast the card is). • We recommend using branded SD cards as they are more reliable.
2a	HDMI to HDMI / DVI lead	• HDMI to HDMI lead (for HD TVs and monitors with HDMI input). **OR** HDMI to DVI lead (for monitors with DVI input). • Leads and adapters are available for few pounds -- there is no need to buy expensive ones!
2b	RCA video lead	• A standard RCA composite video lead to connect to your analogue display if you are not using the HDMI output.
3	Keyboard and mouse	• Any standard USB keyboard and mouse should work. • Keyboards or mice that take a lot of power from the USB ports, however, may need a powered USB hub. This may include some wireless devices.
4	Ethernet (network) cable [optional]	• Networking is optional, although it makes updating and getting new software for your Raspberry Pi much easier.
5	Power adapter	• A good quality, micro USB power supply that can provide at least **700mA at 5V** is essential. • Many mobile phone chargers are suitable—check the label on the plug. • If your supply provides less than 5V then your Raspberry Pi may not work at all, or it may behave erratically. Be wary of very cheap chargers: some are not what they claim to be. • It does not matter if your supply is rated at *more* than 700mA.
6	Audio lead [optional]	• If you are using HDMI then you will get digital audio via this. • If you are using the analogue RCA connection, stereo audio is available from the 3.5mm jack next to the RCA connector.

Know your leads:

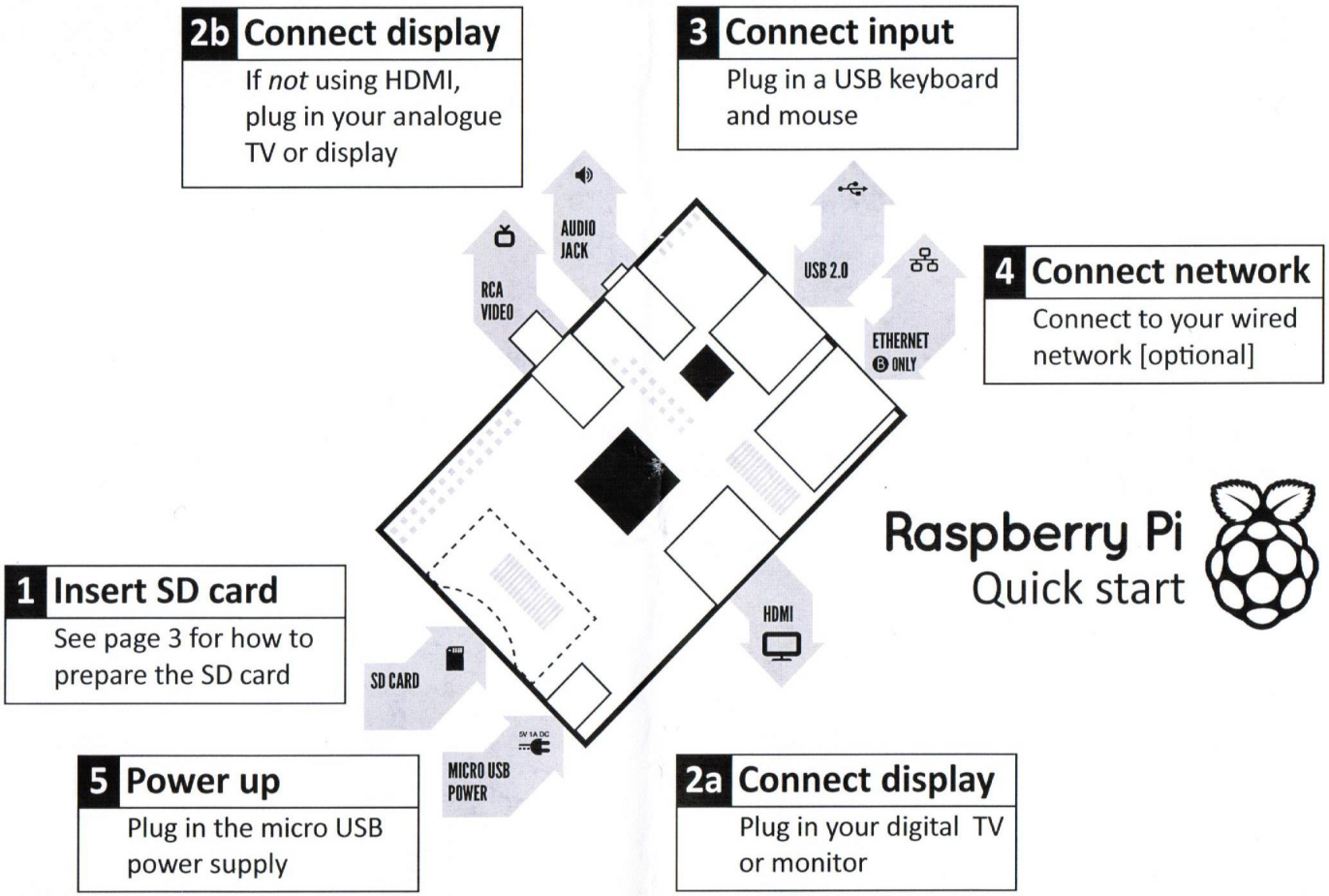


HDMI connector

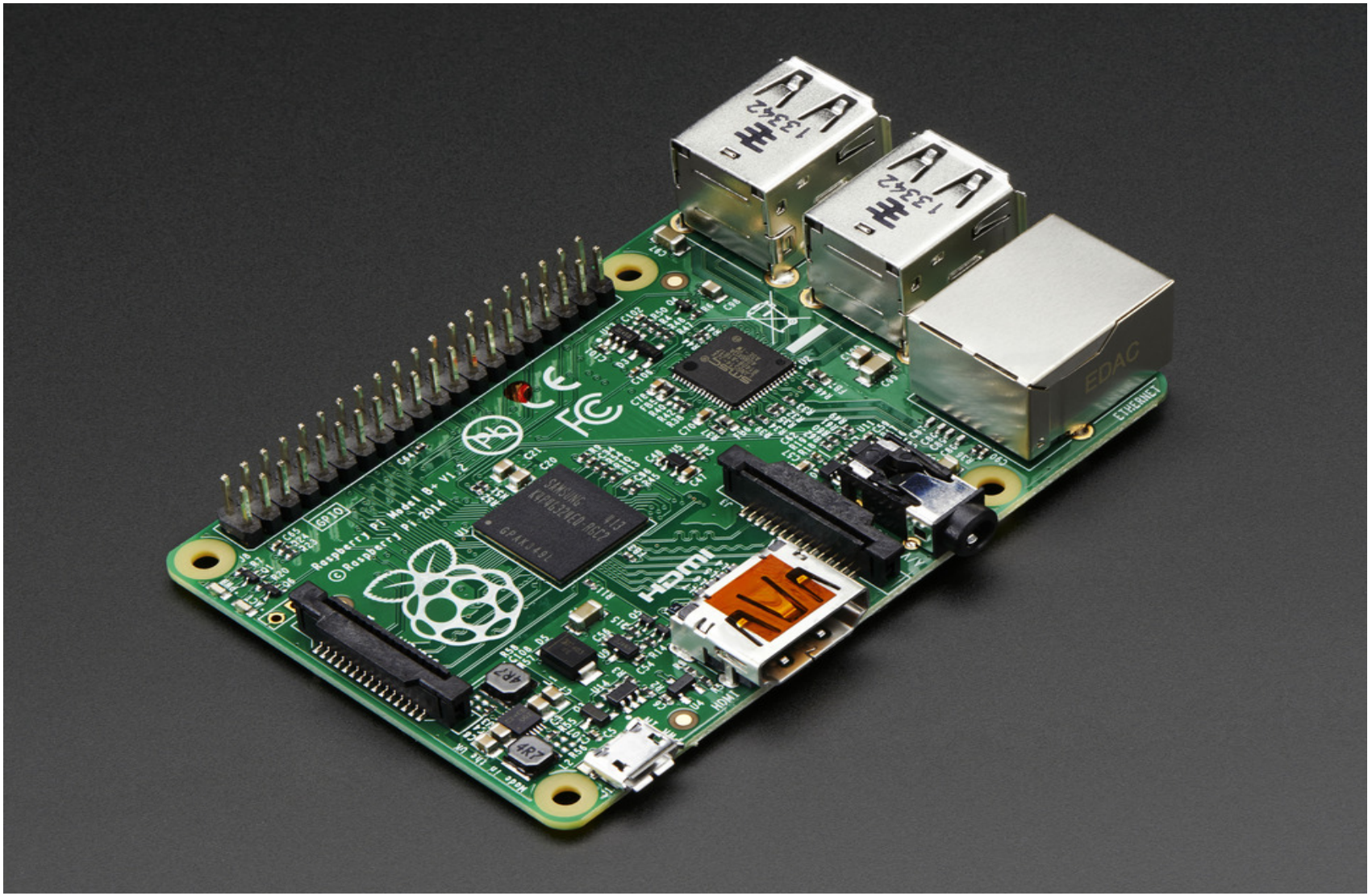
HDMI to DVI lead

RCA composite video connector

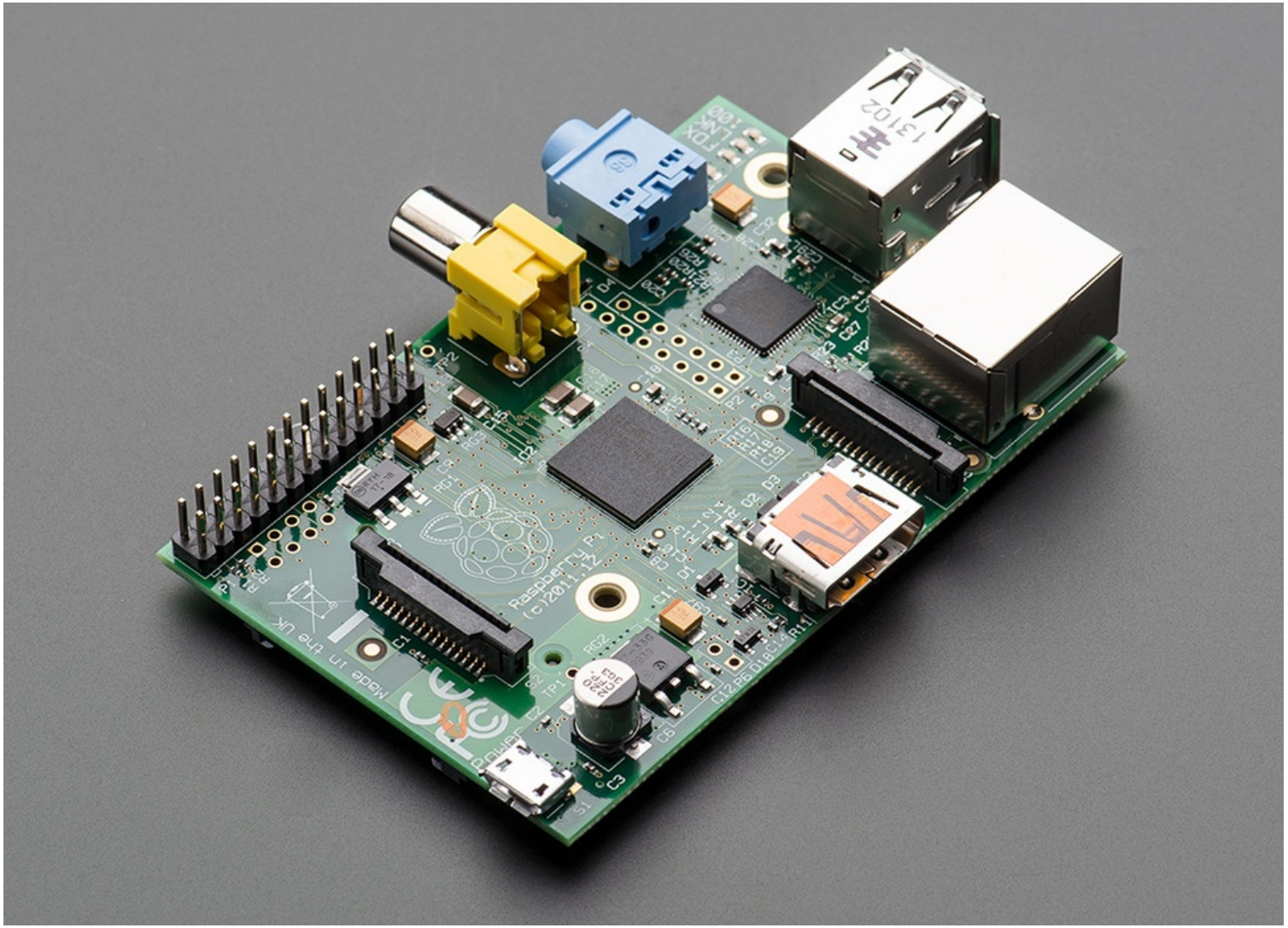
Course Description: Raspberry Pi



Course Description: Raspberry Pi Model B+



Course Description: Raspberry Pi Model B

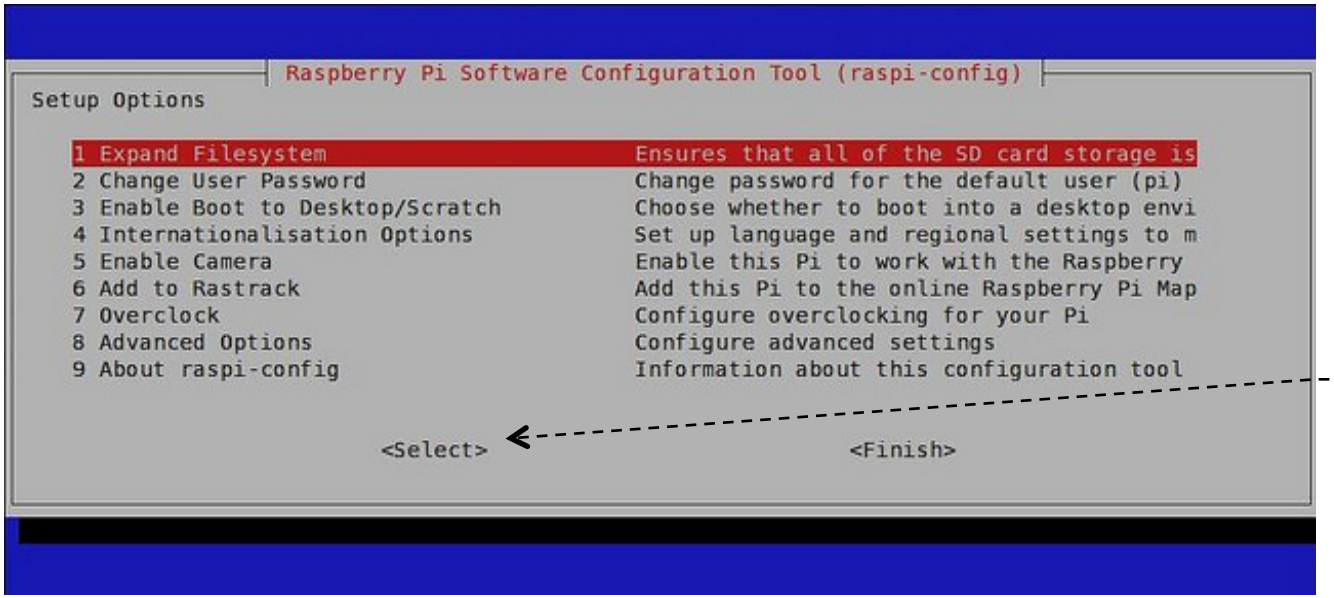


Course Description: Raspberry Pi MicroSD Card/Adapter

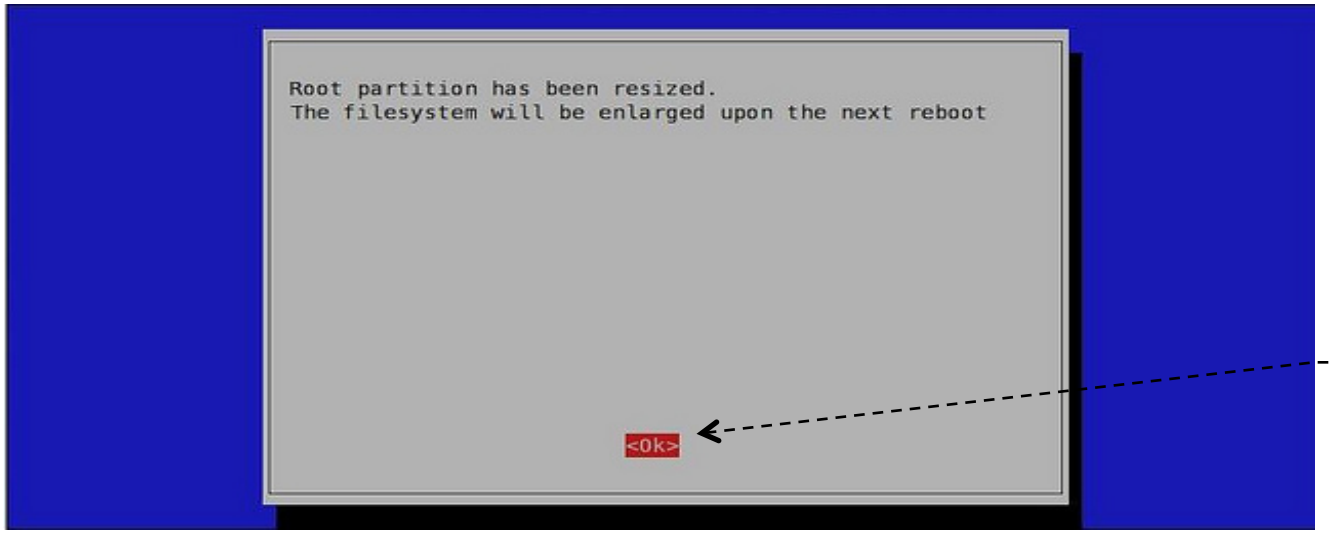
Course Description: Raspberry Pi

- http://www.elinux.org/RPi_SD_cards

Compatible SD cards for Raspberry Pi.



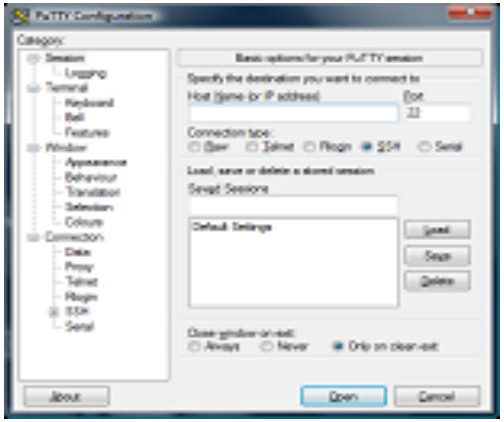
Select “Expand Filesystem”



Select “Ok”

Course Description

- http://www.putty.org



Download PuTTY

PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers.

You can download PuTTY here.

- **Binaries**

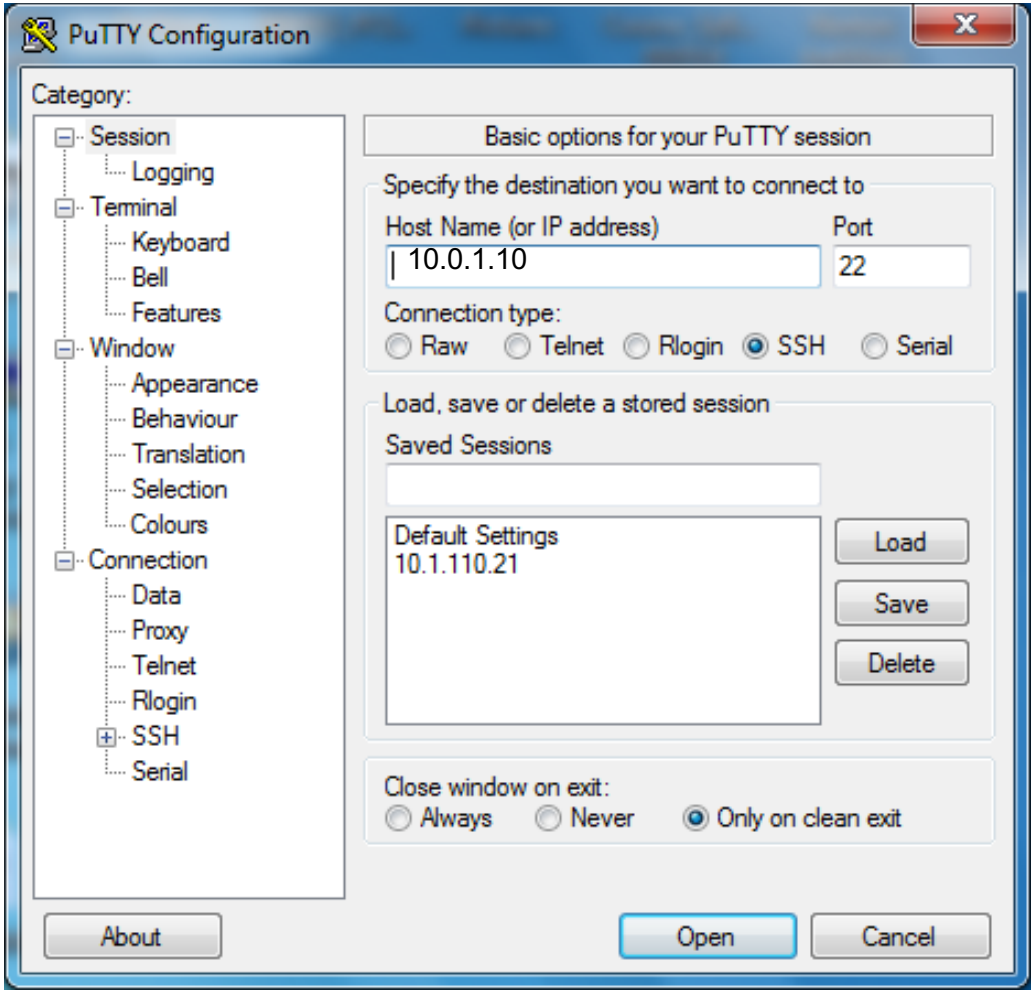
The latest release version (beta 0.63). This will generally be a version I think is reasonably likely to work well. If you have a problem with the release version, it might be worth trying out the latest development snapshot (below) to see if I've already fixed the bug, before reporting it to me.

For Windows on Intel x86

PuTTY: putty.exe (or by FTP) (RSA sig) (DSA sig)

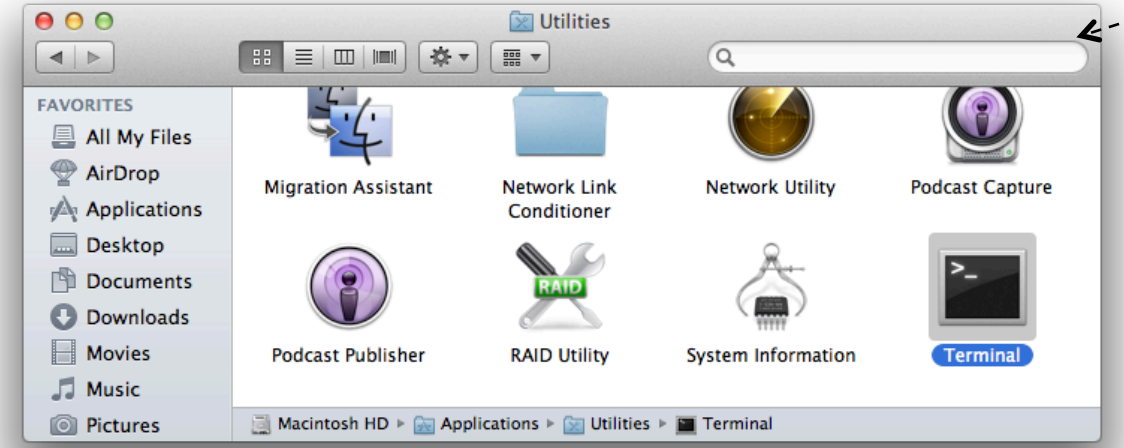
. . .

Course Description



Course Description

- MAC OS X:
  In the “Applications/Utilities” folder, open “Terminal”.

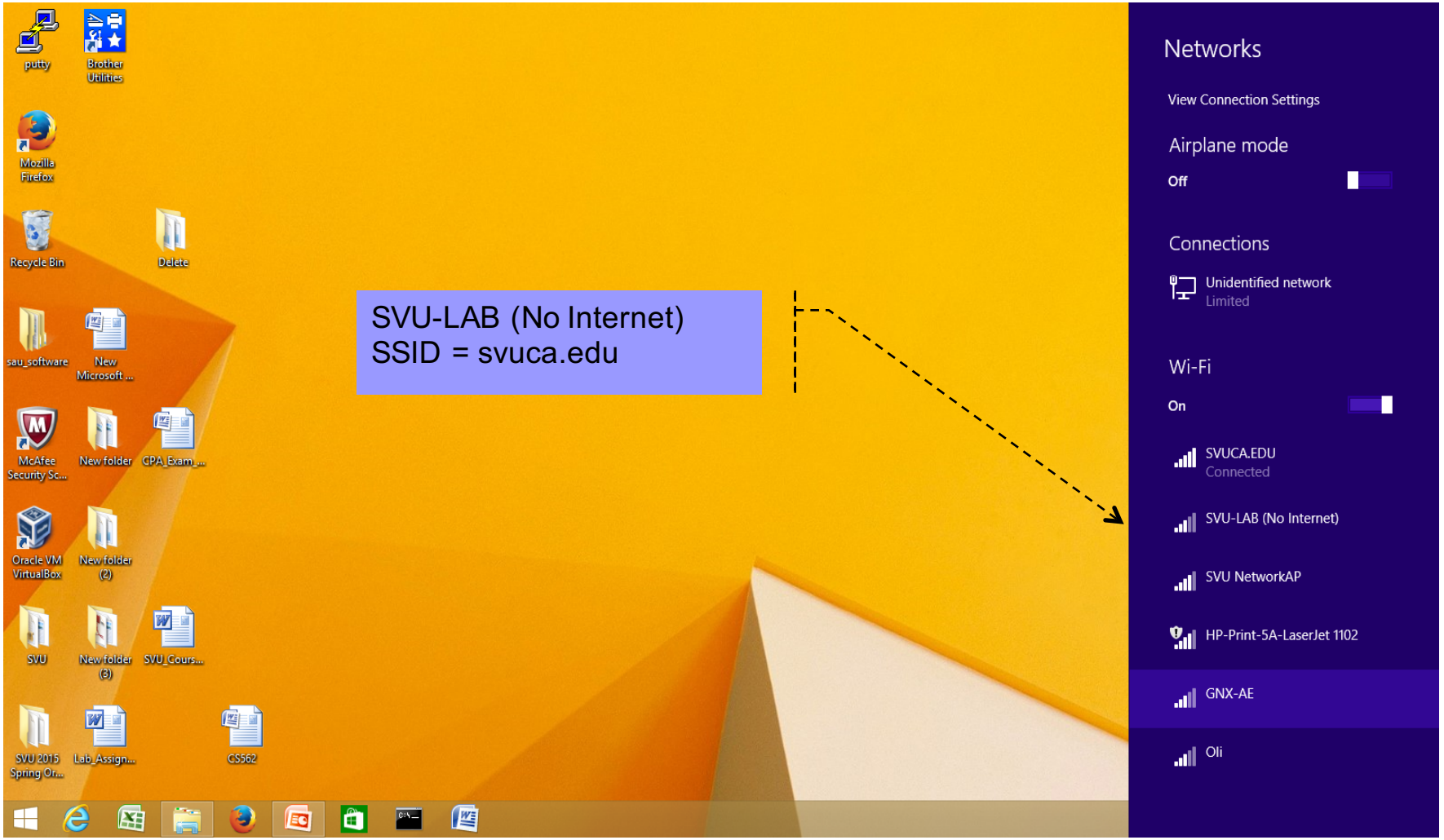


In the upper-right corner of the MAC main window, there is also a “Search” area. Type in “Terminal”.

- **ssh <User Name>@<Server IP Address>**
  ssh cs500a1@10.0.1.10

Course Description

- WIFI Access Point changed to SVU-LAN (No Internet).



Course Description

- Week 1 – 4 (Review)
  - **Operating System: Software used for controlling/allocating resources (CPU, Memory, File-Storage, I/O Devices).**
    - Operating System Structure: Layered, Modular, MicroKernel.
    - Operating System Services.
    - User Operating System Interface (System Calls).
  - Process Management
    - Concept of Process: Unit of Work in a System.
    - Concurrent Kernel/User Processes and Inter-Process Communication.
    - Process Scheduling Policy: Set of Rules Determining When and How a Process is to Run.
    - Process Synchronization (Semaphores and Mutexes).
  - Memory Management
    - Management of Main Memory During Process Execution.
    - Memory Management Schemes / Approaches.
  - Storage Management
    - File Systems (Virtual File Systems).
    - System I/O Control and Designs.
    - Performance Issues with I/O Devices.

Course Description

- Week 5
  - I/O Systems
- Week 6
  - Protection
- Week 7
  - Security
- Week 8
  - Midterm
- Week 9
  - Distributed File Systems
- Week 10
  - Distributed Coordination
- Week 11
  - Real-Time Systems
- Week 12
  - Multimedia Systems

Course Description

- Week 13 Case Study
  - The Linux Systems
- Week 14 Case Study
  - Windows XP
- Week 15
  - Final

Operating System Overview

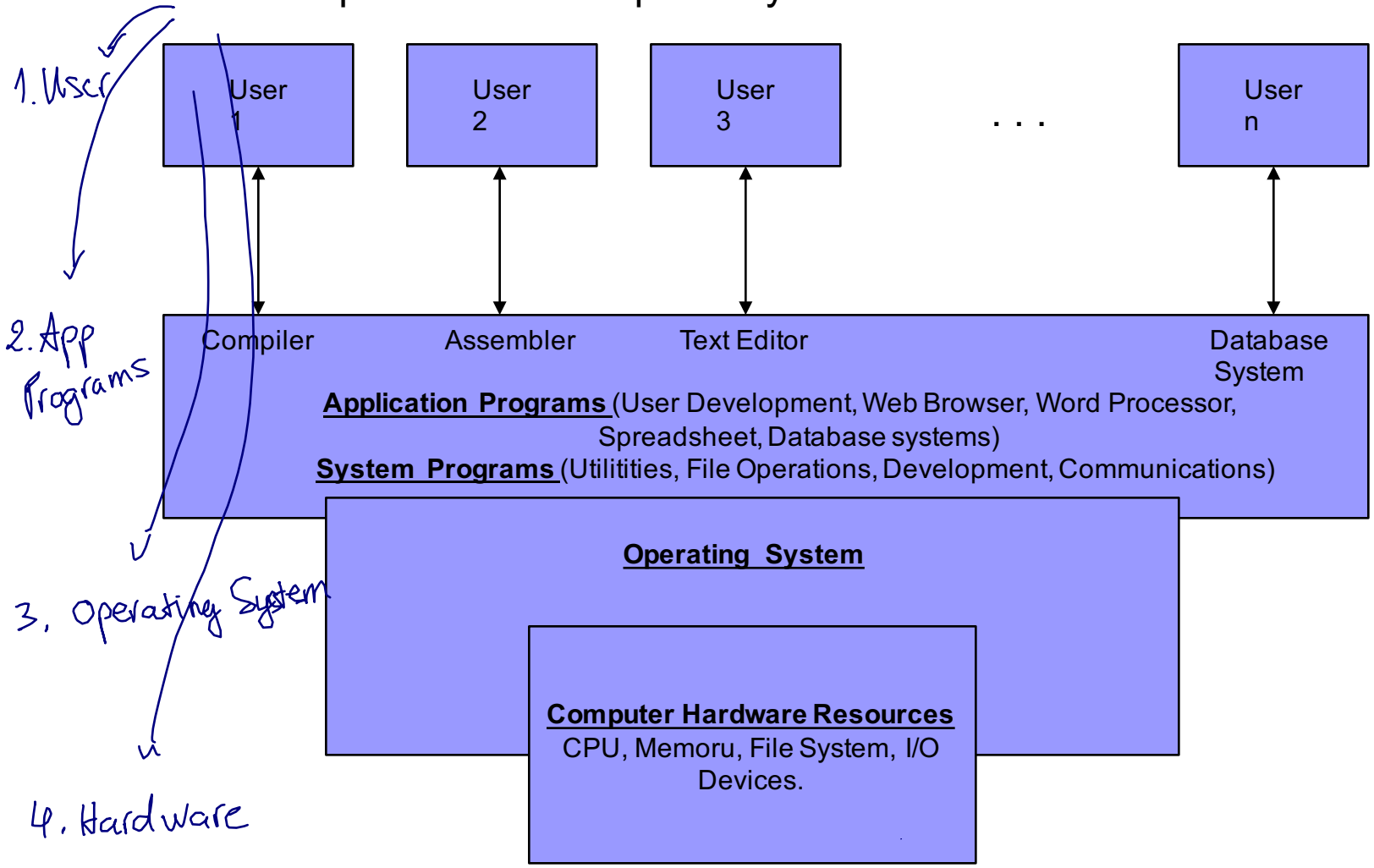
- Operating System Overview Part 1
  - Computer System
  - Operating System
  - Processes
  - Threads
- Operating System Overview Part 2
  - CPU Scheduling
  - Process Synchronization
  - Deadlocks
  - Main Memory
  - Virtual Memory
  - File-System Interface
  - File-System Implementation
  - Mass-Storage Structure

Operating System Overview

- Computer System.
  - Goal of Computer System is to execute user programs and solve user programs easier.
  - Computer System
    - Computer hardware (CPU, Memory, Peripheral I/O Devices).
    - Computer hardware resource need to be controlled and allocated.
    - **Software used for controlling/allocating resources (CPU time, Memory Space, File-Storage Space, I/O Devices) is the Operating System**.
  - Operating System Consists of Kernel and . . .
    - System Programs
      - Utilities providing environment for user application development and execution.
      - System Calls (i.e. File Operations).
      - System programs to manipulate files and directories (i.e. mkdir, cp, rm, ls).
      - System programs for development (i.e. compilers, assemblers, loaders).
      - Communications between processes or other computer systems (i.e. sockets, network protocols).
    - Application Programs
      - Utilities not associated with operation of the Operating System, but used for problem solving.
      - Web Browsers, Word Processor, Spreadsheets, Database Systems, Games.

Operating System Overview

- Four Components Of Computer System



Operating System Overview

- Computer System Architecture
- Single Processor Systems
- One General-Purpose CPU
  - One main CPU executing general-purpose instruction set.
  - System can have multiple special-purpose processors embedded in hardware (i.e. keyboard) or in controllers.
    - Special-purpose processors has limited instruction set.
      - Disk controller has disk queue and scheduling algorithmn.
      - Keyboard processor converts keystrokes into codes.
    - Controlled by main CPU.
- Multiprocessor Systems
  - Less cost than multiple single-processor systems.
    - Share peripherals, mass storage, memory, power supplies.
  - Increased throughput.
    - Incur overhead keep processing synchronized.
    - Incur contention shared resources.
  - Increased Reliability.
    - Built-in redundancy in core.
    - Graceful Degradation: Failure of one processor will not crash Computer System.

Asymmetric
Symmetric : Unix, Windows ...

Operating System Overview

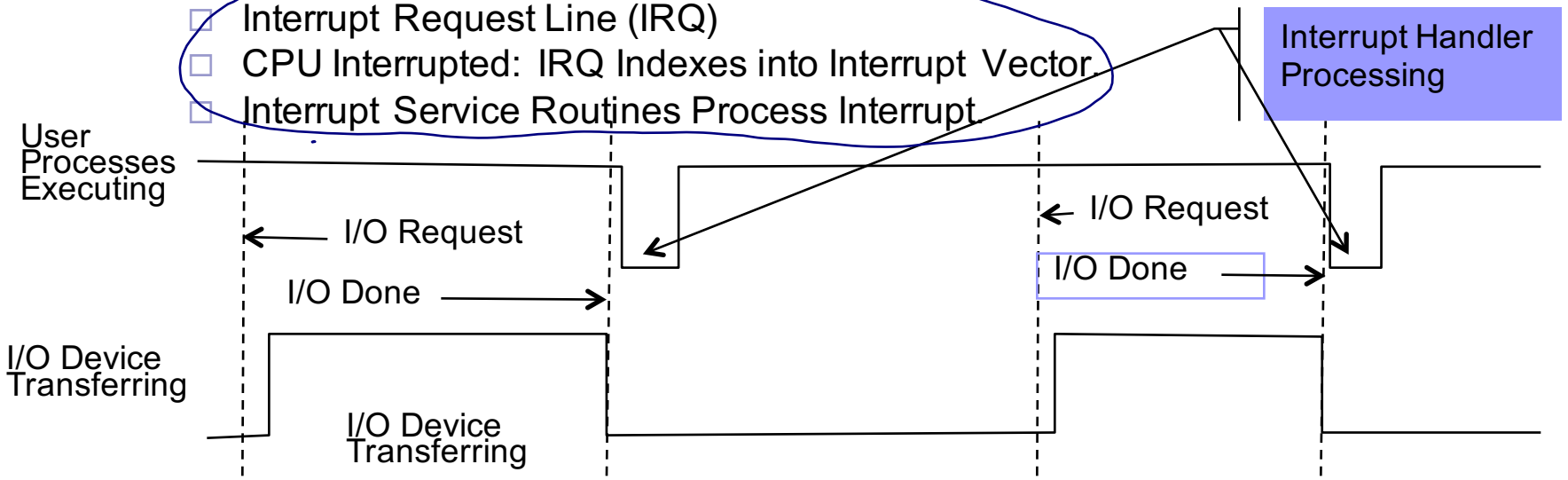
- Computer System Architecture
- Multiprocessor Systems
  - Asymmetric Multiprocessing
    - Master-Slave Processor Relationship: Each processor assigned task.
    - Master Processor schedules and allocates work to the Slave Processor.
    - CPU can have specific actions: One CPU executes Operating System code, another CPU executes only I/O operations.
  - Symmetric Multiprocessing (SMP)
    - Peer-Peer Processor Relationship.
    - Each processor performs all tasks within the Operating System
    - Each processor can run separate process.
    - Sun Solaris, Windows, Windows XP, MAC OS X, Linux.
  - Cache Memory speed-up Main Memory access.
- Multi-core System
  - Single chip contains multiple computing cores.
  - Less cost than multiple single-processor systems.
    - Share peripherals, mass storage, memory, power supplies.
  - On-Chip communication faster.
  - Less power
  - Server Systems (Web Servers).

Operating System Overview

- Computer System Architecture
- Clustered Systems
  - Individual Computer Systems share storage and linked through LAN.
  - Cluster Computer Systems (Nodes) running layer of Cluster Software linking nodes together.
  - High-Availability Service: Failure on one system causes service to restart on the redundant system.
  - Asymmetric Clustering
    - One system active, one system in Hot-Standby System.
  - Symmetric Clustering
    - Both systems active executing applications, but monitoring each other.
  - Parallel Cluster over WAN.
    - Multiple hosts accessing same data on shared storage.
    - Require special applications: Oracle Real Application Cluster database.
      - Access Control using Distributed Lock Manager (DLM).
  - Beowulf Clusters
    - High Performance Computation sharing computing power of multiple computers.
    - Personal Computers linked through software libraries.
    - Open Source Linux Systems.

Operating System Overview

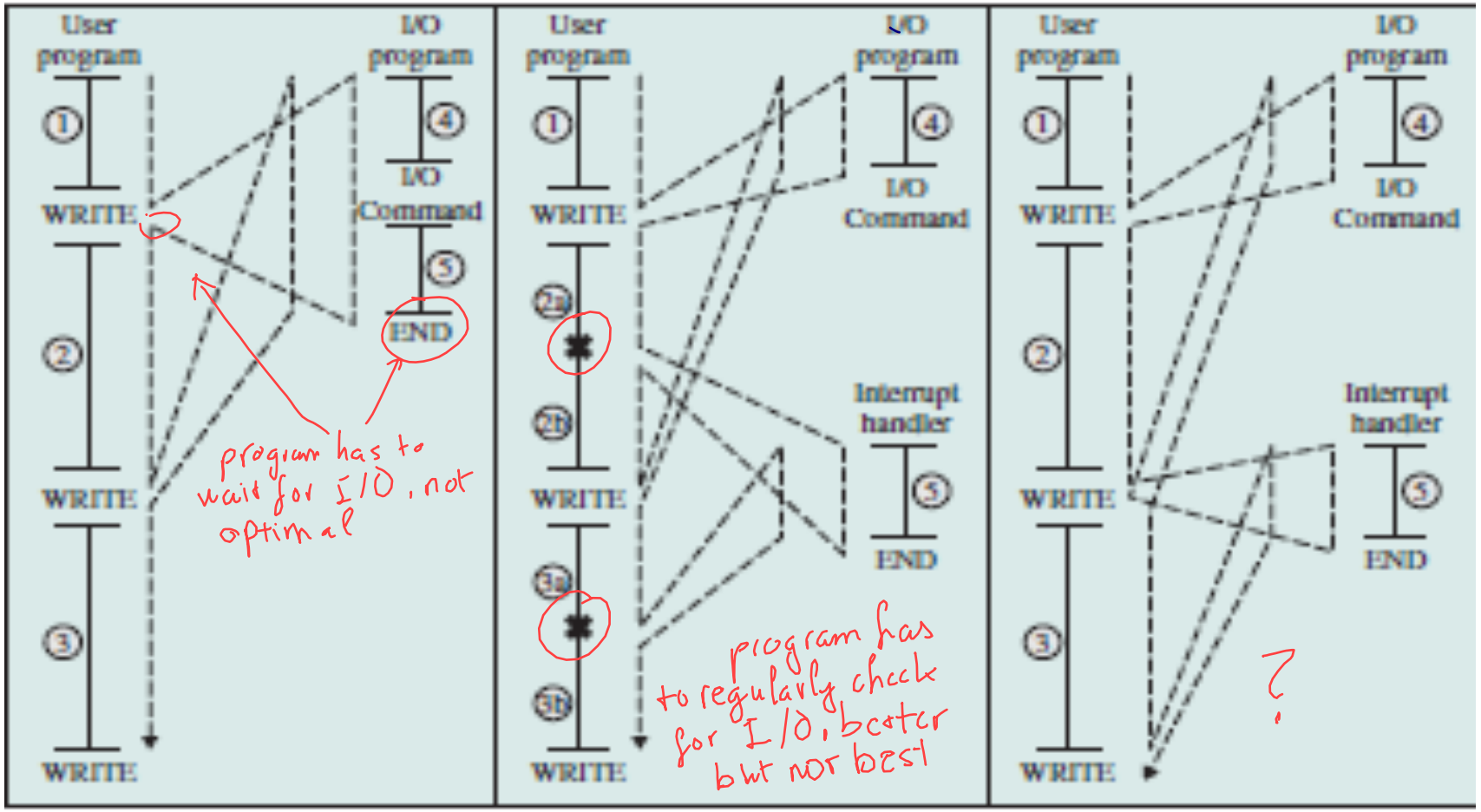
- Computer System Organization
- Computer System Operation Steps
- Firmware (Bootstrap Program)
  - Computer Initialization – Bootstrap Program on ROM or Flash
  - Diagnostic Tests of Peripherals, Hardware, and Memory Devices.
  - Initializes CPU Registers, Device Controllers, Memory Contents.
  - Loads First Process (i.e. init process) into Memory and executes the process.
- Operating System Runs (Waits for Events)
  - Schedules Processes
- Events
  - Interrupt Request Line (IRQ)
  - CPU Interrupted: IRQ Indexes into Interrupt Vector.
  - Interrupt Service Routines Process Interrupt.



Operating System Overview

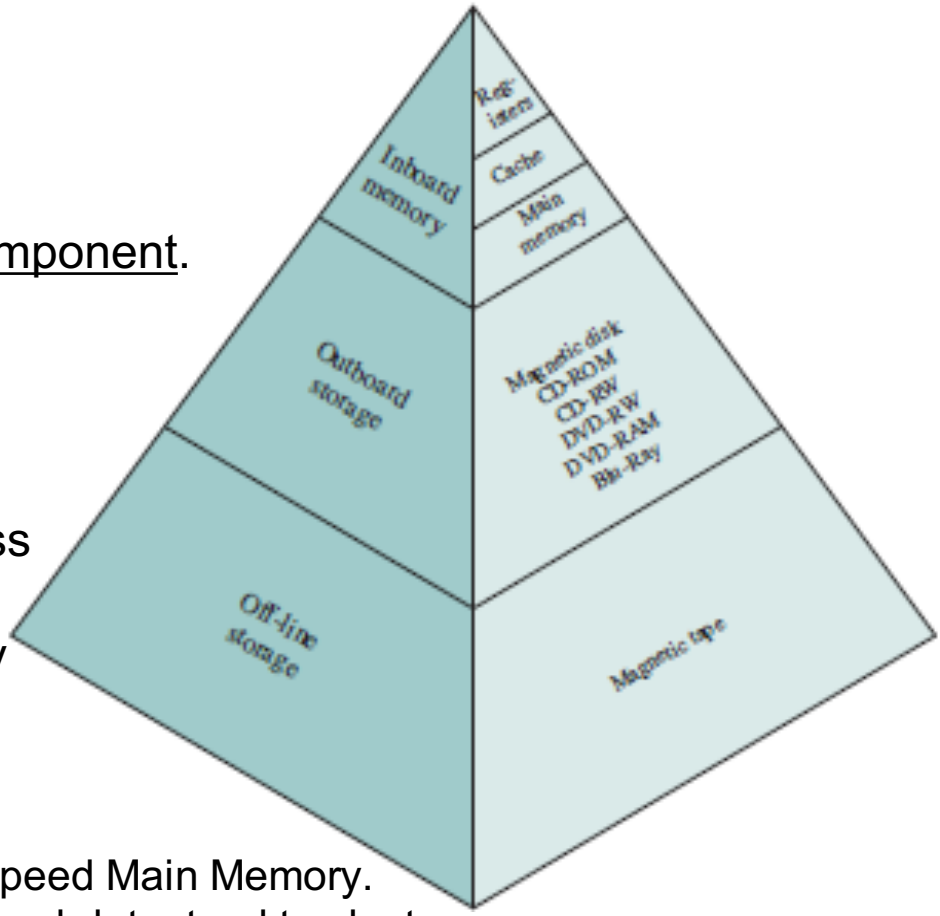
- Computer System Organization
- Program Flow With Interrupts

Solid Line: Segments of code in a program.
Dashed Line: Path of execution by the processor.



Operating System Overview

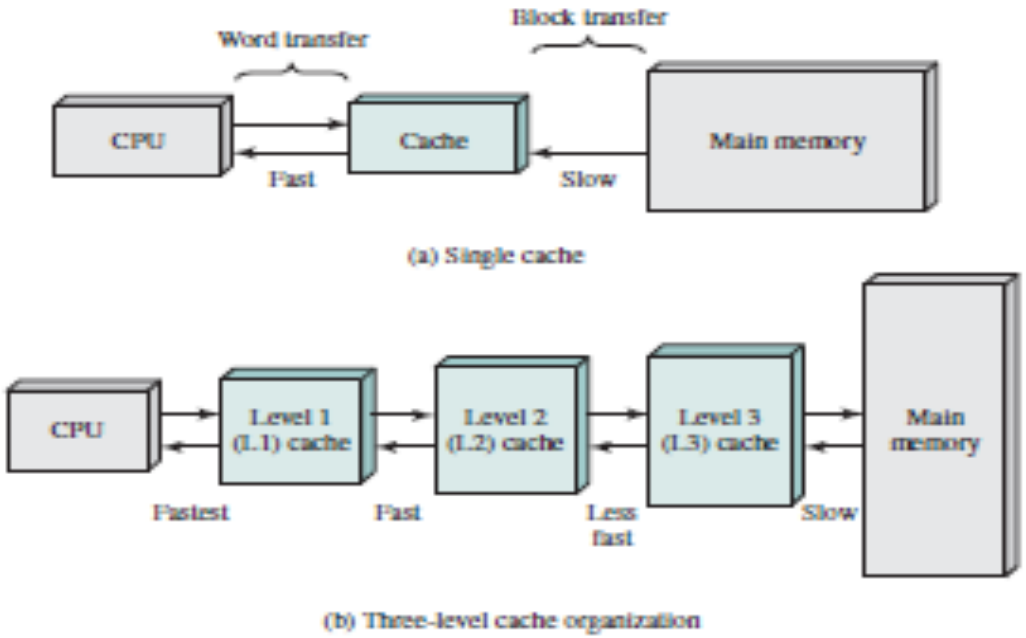
- Computer System Organization
- Memory Hierarchy
  - Do not rely on single Memory Component.
  - Going down the Hierarchy:
    a) Decreasing cost per bit.
    b) Increasing Capacity.
    c) Increasing Access Time.
    d) Decreasing frequency of access to the memory by processor.
  - Strategy is to decrease frequency of access at lower levels.
  - Locality of Reference:
    - Two levels of access,
      High Speed Cache and Slower Speed Main Memory.
    - Memory references, instructions and data, tend to cluster.
      Programs have iterative loops and subroutines.
      Tables and arrays involve access to clustered set of data bytes.
    - Short period of time, processor using fixed clusters of memory references.
    - Cache stages movement of data between main memory and registers to improve performance.



Operating System Overview

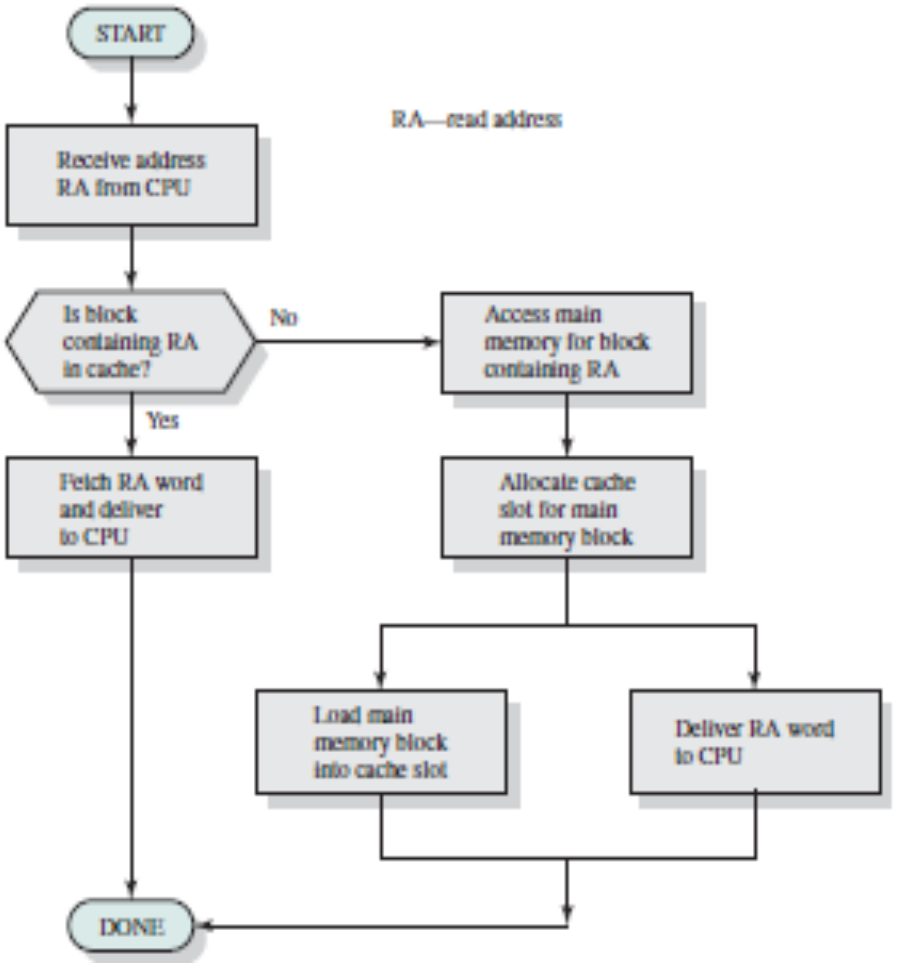
- Computer System Organization
- Cache Memory
- Instruction Cycle:
  - Processor access memory to fetch instruction, and one or more times to fetch operands and/or store restults.
  - Rate processor can execute instruction limited by memory cycle time.
  - Process speed has been increasing more rapidly than memory access speed.
  - Main memory should be built with same technology as processor registers, giving memory cycle times comparable to processor cycle times.
- Cache provides a small, fast memory between processor and Main Memory.

Block of data is fetched into the cache to satisfy a single memory reference. Likely many of the future memory references will be to other bytes in the block.



Operating System Overview

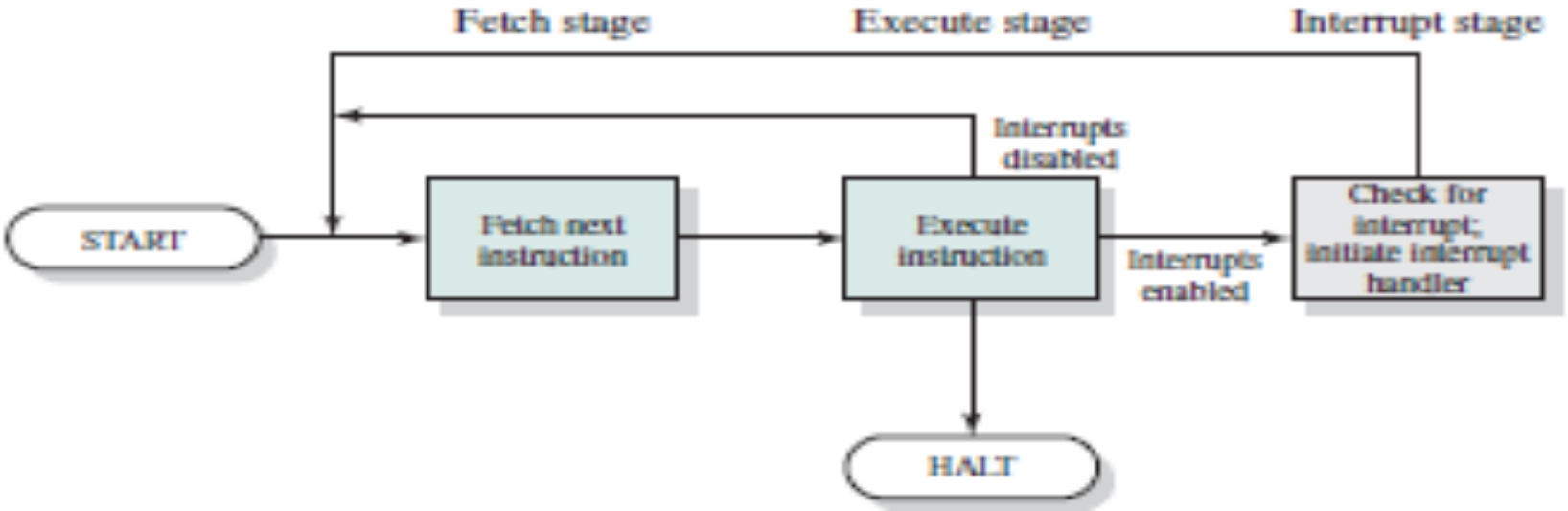
- Computer System Organization
- Cache Design
  - Cache Size: Reasonably small caches can have impact.
  - Block Size: As block size increase hit ratio will increase. If block size gets too large, hit ratio decrease because block is larger than the principle of locality.
  - Mapping Function: Replacement Algorithm determines which block will be replaced (Least Recently Used replaces the block in the cache the longest).
  - Write Policy: Contents of the cache are altered, when does the memory write take place. Writing can occur every time the block is updated, or only when the block is replaced.



Operating System Overview

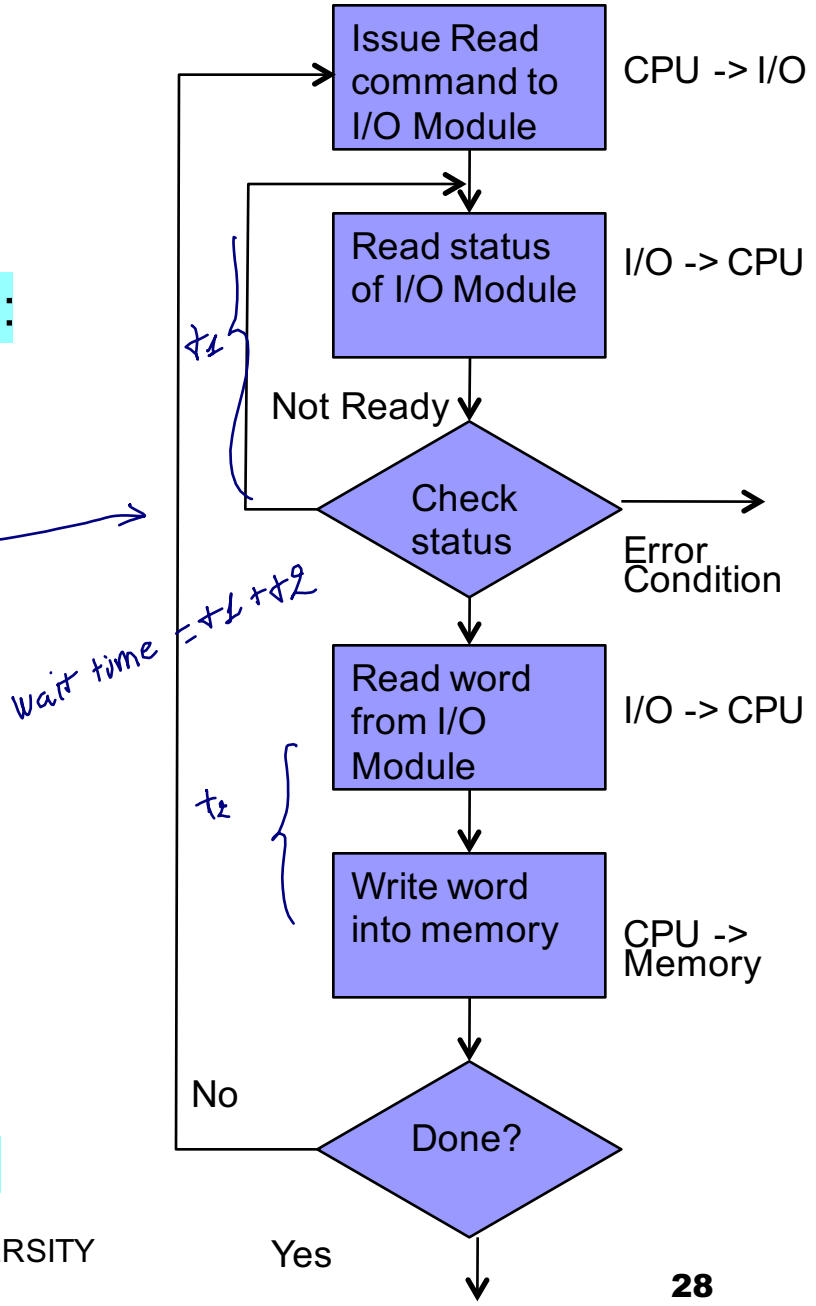
2 types of interruption
- Hardware can trigger interruption via system bus to CPU
- Software triggers interruptions to CPU via system calls

- Computer System Organization
- Instruction-Execution Cycle Interrupt Stage
  - Process interprets instruction and performs required action.
  - Processor checks to see if interrupt signal has been raised.
  - Interrupt Pending: Suspends execution of current program and executes interrupt-handler routine.
  - Interrupt-handler performs action requested by event.
  - Interrupt-handler completes, process resume execution at the point of interruption.



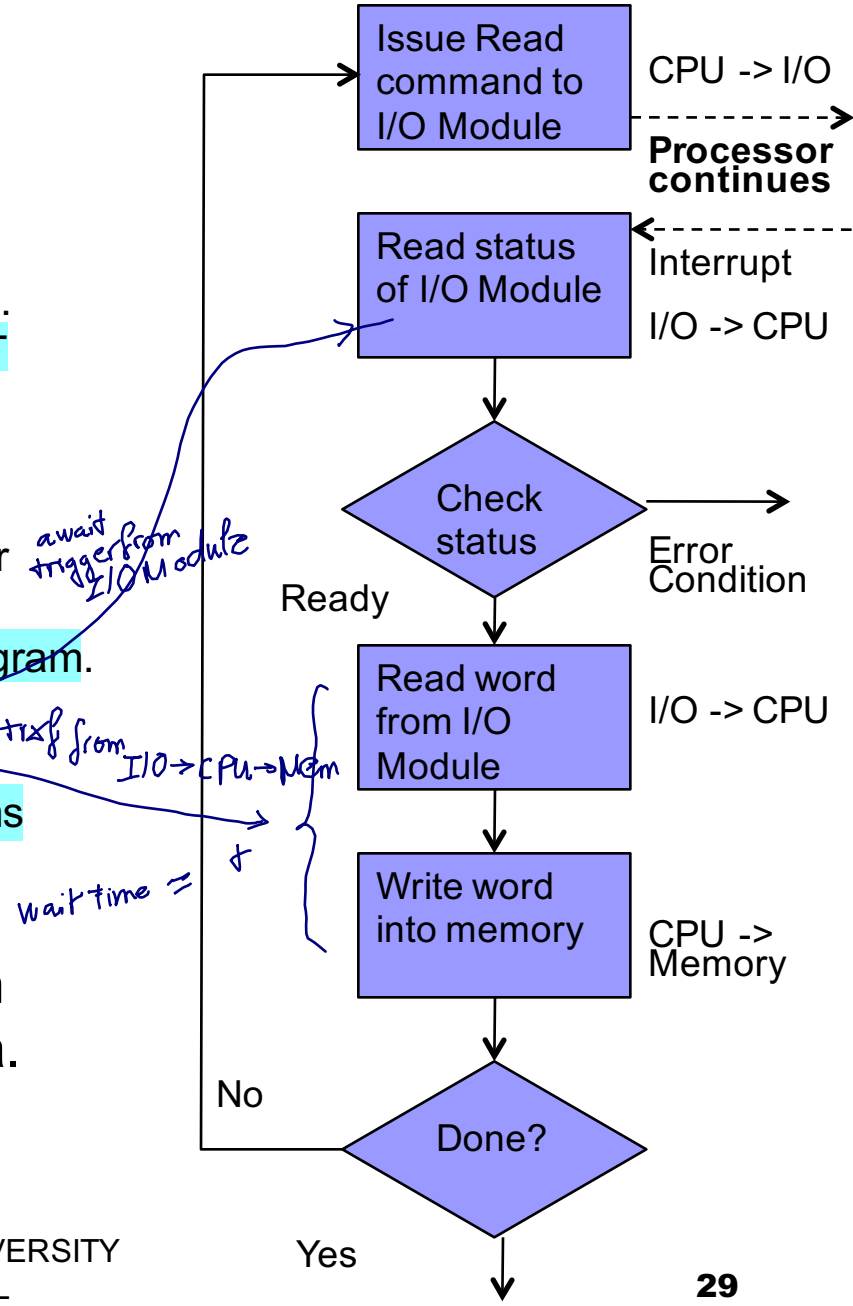
Operating System Overview

- Computer System Organization
- I/O Communication
- Process executing I/O instruction uses:
  - Programmed I/O
  - Interrupt-Driven I/O
  - Direct Memory Access (DMA)
- Programmed I/O
  - Processor issues command to I/O Module.
  - I/O Module completes the request and sets I/O Status Register.
  - Processor must periodically check I/O Module for completion.
  - I/O Module Instruction Set:
    - Control: Activate and commands to I/O Module.
    - Status: Test I/O Module status.
    - Transfer: Read/Write data between processor register and I/O Module.
  - Disadvantage: Time consuming and keeps processor busy.

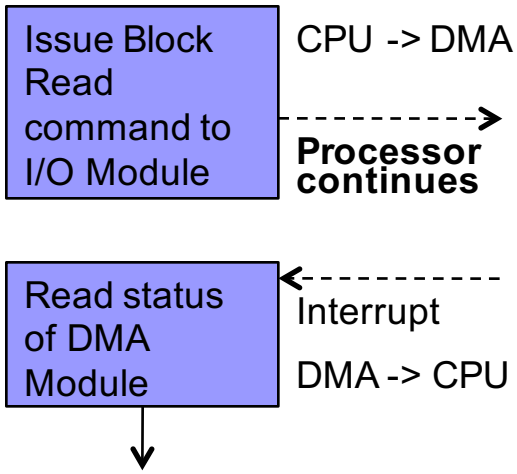


Operating System Overview

- Computer System Organization
- Interrupt-Driven I/O
  - Processor issues command to I/O Module.
  - Processor continue other work: Does NOT check I/O Module repeatedly.
  - I/O Module will interrupt the processor to request service.
  - At end of each instruction cycle, processor checks for interrupts.
  - Processor saves context of executing program.
  - Processor executes Interrupt Handler.
  - Processor executes the data transfer.
  - Processor restores context of the programs that issued I/O command.
- Eliminates needless waiting.
- Consumes processor time: Data from memory to I/O Module and vice versa.



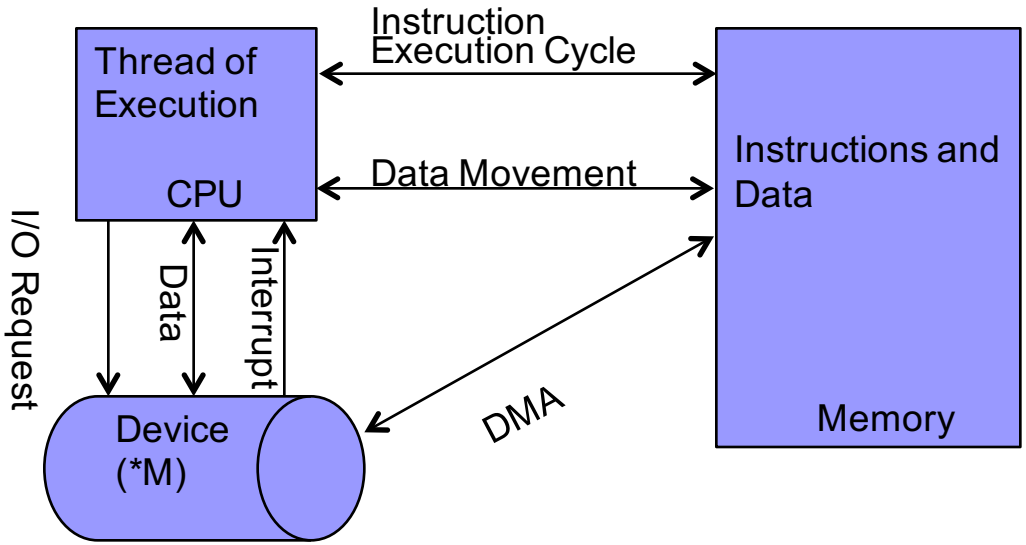
Operating System Overview



- Computer System Organization
- Direct Memory Access (DMA) I/O
  - Processor issues command to I/O Module.
  - DMA function in I/O Module handles data transfer.
  - Processor sends DMA Module:
    - Read or Write Request.
    - Address of I/O Device.
    - Memory address to Read from or Write to.
    - Number of words to Read or Write.
- DMA Module takes control of Bus to transfer data to and from memory.
- DMA Transfer Modes:
  - Cycle Steal: Processor must wait one Bus cycle when Bus conflict with DMA Module.
  - Burst Mode: Entire block of data transferred in one contiguous sequence. Processor must wait for DMA to complete.
  - Transparent Mode: DMA controller only transfers data when CPU is not using the System Bus. Best system performance, but most complicated.

Operating System Overview

- Computer System Organization
- Storage Structure
- I/O Structure
  - Large portion of Operating System responsible for managing I/O.
  - Device Driver connected to Device Controller.
  - Device Controller supports multiple devices (i.e. SCSI Controller).
  - Direct Memory Access (DMA).
    - Transfers block of data directly to or from its own buffer storage to memory.
    - Does not involve CPU during transfer,



Computer System with DMA

? (review textbk)

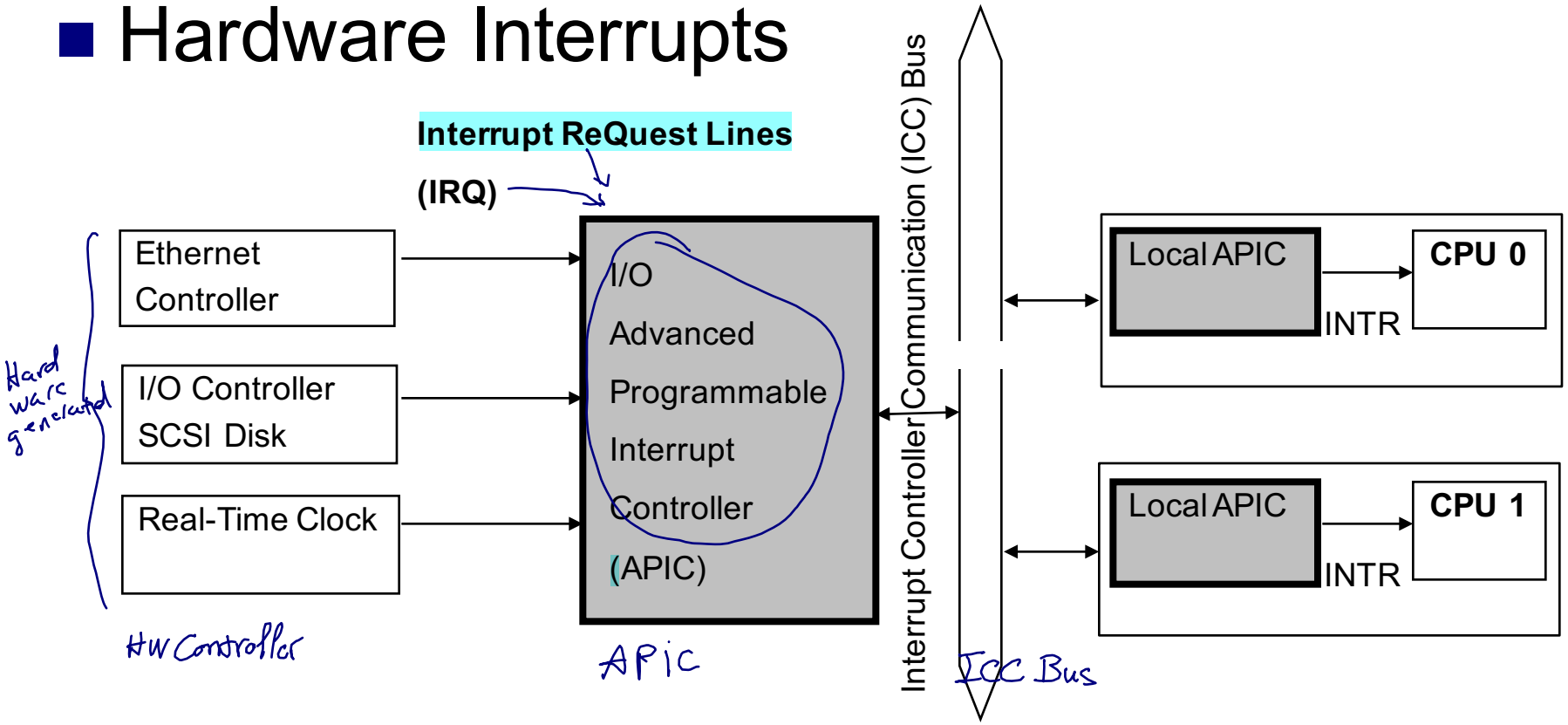
Operating System Overview

- Operating System Operations
- Interrupt Driven
- Interrupt Event Alters the Sequence of Instructions Executed by a Processor
  - Synchronous interrupts (Trap or Exceptions) produced by CPU.
    - Software Interrupts.
    - Programming errors, divide by zero, illegal memory reference.
    - Conditions handled by kernel, page fault.
    - System calls.
  - Asynchronous interrupts generated by hardware devices.
    - Hardware Interrupts.
    - Keyboard keystrokes.
    - Data received.
    - Timer: Ensure Operating System maintains control over the CPU. Event used to schedule programs for execution.
  - Interrupt and exception handlers are not processes, they execute in the system control path.

Software Interrupt

Hardware Interrupt

Hardware Interrupts



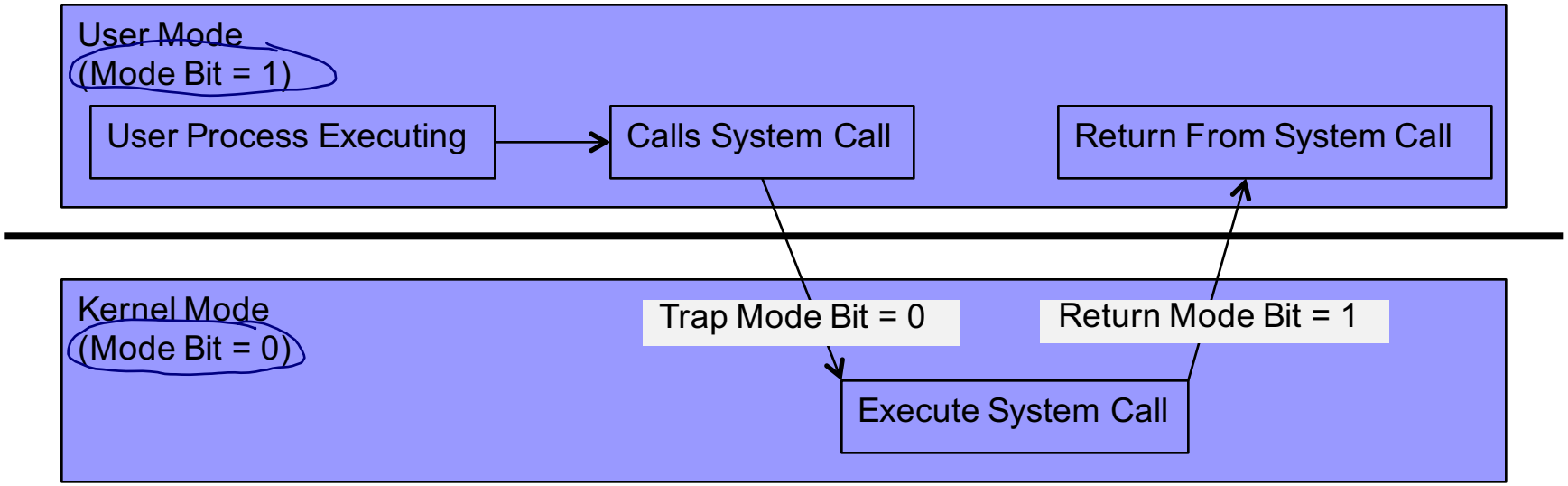
I/O devices: Unique or shared Interrupt ReQuest Lines (IRQs).

I/O APIC: 24 IRQ lines. Send/Receive APIC Messages over ICC BUS to Local APICs.
IRQ signals delived to available CPUs (Static or Dynamic Distribution)

Local APIC: Mapped to interrupt vectors.
Signals processor with INTR (issues an interrupt).

Operating System Overview

- Operating System Operations
- Dual-Mode Operation: Distinguish between User and Kernel Mode.
- User Mode: Computer system executing on behalf of an user application.
- Kernel (System) Mode: Operating System execution.
  - Trap or interrupt, switch to Kernel Mode.
  - System call is a trap (software interrupt) caused by user program requesting Operating System to perform task reserved for the operating system.
    - I/O Control, Timer Management, Interrupt Management.
  - Privileged Instructions.



Operating System Overview

- Operating System Process Management
- CPU executes Instructions of a Process (User or System)
- **What is a Process? - A Unit of Work in a Computer System -**
  - Computer System consists of Operating System processes (executes system code) and user processes (executes user code).
  - Process needs resources – CPU, memory, files, I/O devices.
  - Process might need initialization data (input).
  - Process termination, Operating System reclaims resources.
- Multi-threaded: Parent process creates multiple process threads
  - Shares memory address space and resources (i.e. open files).
- Multi-process: Parent process creates multiple child processes
  - Separate memory address space and resources.
- Operating System Provides:
  - Scheduling processes and threads on the CPUs.
  - Creating and deleting both User and System processes.
  - Suspending and resuming processes.
  - Providing mechanisms for process synchronization.
  - Providing mechanisms for process communication.

Operating System Overview

- Operating System Memory Management
- How does CPU executes User Process?
- User Process User Programs loaded to Main Memory before execution.
  - User Process has own address space (User Space), separate from Operating System space (Kernel Space).
- Main Memory central to operation of Computer System
  - CPU executes instructions from Main Memory
  - Data transferred to Main Memory for processing
- Multiple user programs kept in Main Memory
  - User programs in its own address space.
  - Operating System Scheduler selects the user program to execute.
- Memory Management Required to:
  - Keep track of which parts of memory are currently being used and by which user program.
  - Decide processes (or parts thereof) and data to move into and out of memory.
  - Allocating and deallocating memory space as needed.

Operating System Overview

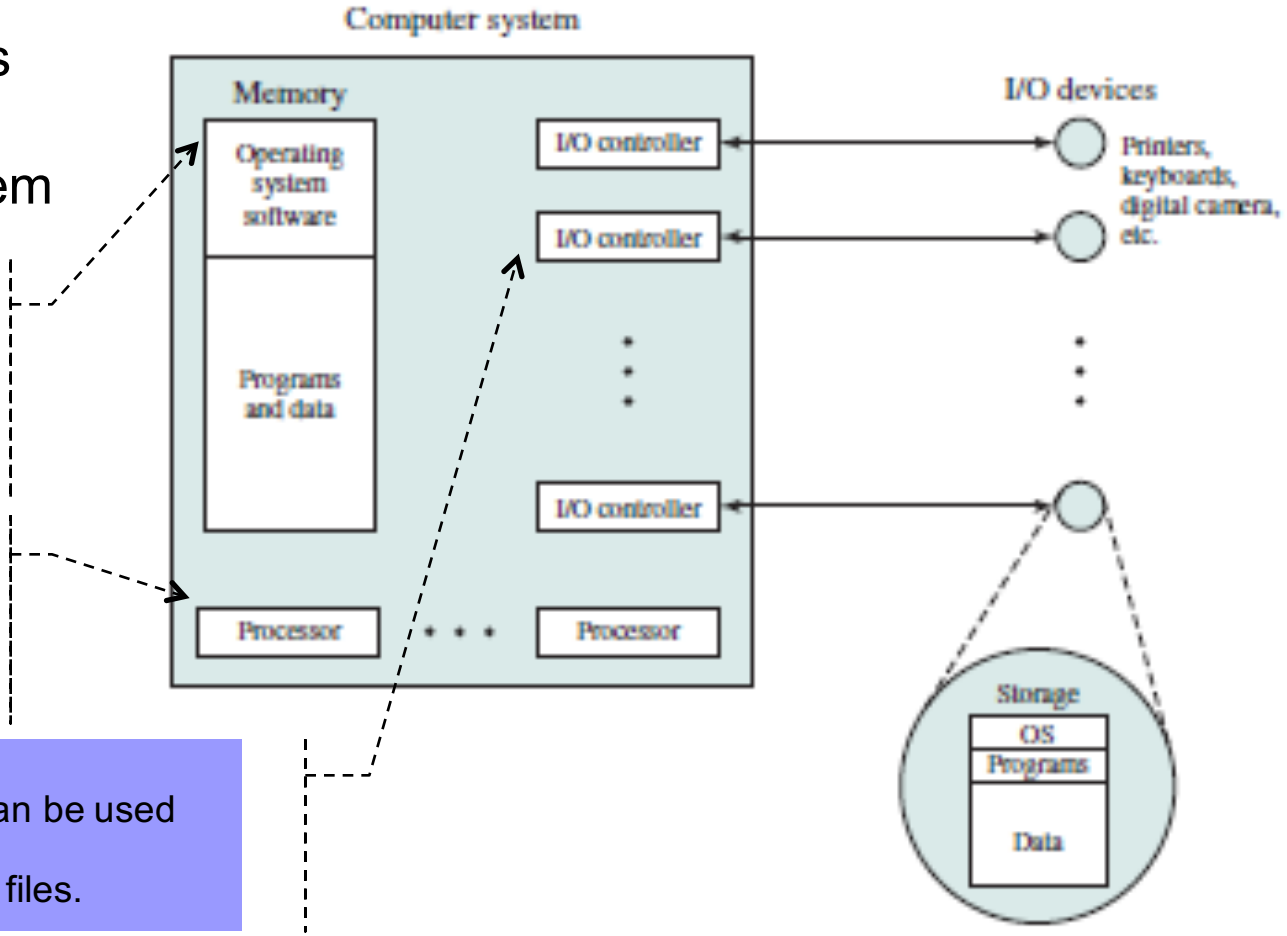
- Operating System Storage Management
- User Programs stored in File System before loaded into Main Memory.
- File System Management:
  - File: Abstraction of the physical properties of a storage device (i.e. disk).
  - Multiple access controlled by access permissions (i.e. read, write).
  - Operating System Provides File Management:
    - Creating and Deleting files.
    - Creating and Deleting Directories to organize files.
    - Supporting primitives for manipulating files and directories.
    - Mapping files onto secondary storage.
    - Backing up files on stable (nonvolatile) storage media.
- Mass Storage Management
  - Main Memory programs and data backed up to secondary storage.
  - Most programs stored on mass storage device until loaded into Main Memory.
  - Operating System Provides Mass Storage Management:
    - Free-space Management.
    - Storage Allocation.
    - Disk Scheduling.

Operating System Overview

- Operating System Protection and Security
- Regulate Multiple Users and Concurrent Processes execution.
- Protection: Mechanism for controlling access of processes or users to Operating System resources.
  - Operating System Physical User Protection
    - Providing private address space for User Processes.
    - Separation of User Space and Kernel Space.
  - Mechanism for controlling access to Computer System Resources.
    - Process must have proper authorization from Operating System.
    - Files (File Management), Memory Segments, CPU.
- Security: Defend system from external and internal attacks.
  - Stolen username and passwords.
  - Viruses, Worms, Denial-of-Service.
- Protection and Security require means to distinguish among users.
  - User Identifiers (User Names + User Identifier) unique for each user.
    - Identify Individual User access.
  - Group Identifiers (User Names + Identifier + Group Identifier) unique for each user in a group.
    - Identify sets of user in a group access.

Operating System Structures

- Operating System as Resource Manager
- Operating System as a control mechanism:
  - Operating System functions as computer software: Program or suite of programs executed by processor.
  - Operating System must relinquish control to allow other (User) programs to execute.
- Main Resources Managed by Operating System



Memory:

Kernel functions resides in memory.

User programs and data occupy the rest.

Processor(s):

OS determines how much time devoted to each user program.

I/O Devices:

OS decides when I/O device can be used by program in execution.

OS controls access and use of files.

Operating System Structures

- Development of Operating System (Cont)
- Five Major Theoretical Advances in Operating System Development to meet objectives of convenience, efficency, and ability to evolve:
  - Processes
  - Memory Management
  - Information Protection and Security
  - Scheduling and Resource Management
  - System Structure
- Design Issue: Processes
  - Fundamental to the structure of Operating System. Process is:
    - Program in execution or instance of a program running on a computer.
    - Entity that can be assigned to and executed on a processor.
    - Unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources.
  - **Three major lines of Computer System development based on the concept of the process.**
    - Multiprogramming Batch Operations.
    - Time Sharing.
    - Real-Time Transaction Systems.
    - Contributing to timing and synchronication problems that must be resolved for the process.

Operating System Structures

- Design Issue: Processes (Cont)
  - Multiprogramming Batch Operations:
    - Designed to keep the processor and I/O devices (including storage devices) simultaneously busy to achieve maximum efficiency.
    - Interrupt-Driven I/O or DMA allow processor to issue I/O command for one job and schedule another job for execution.
  - Time-Sharing:
    - Designed to be responsive to needs of individual user and supporting many users simultaneously. Users engage in multiple development activities.
  - Real-Time Transaction Processing:
    - Designed to be responsive to one or few applications (queries or updates against database in Airline Reservation System).
  - **System software to coordinate the different activities difficult**:
    - Improper Synchronization: Lost signal when waiting for event (I/O Complete).
    - Failed Mutual Exclusion: Two users fail to lock-out when using shared resource (Writing to File).
    - Non-determinate Program Operation: Programs writing shared memory areas, altering program execution.
    - Deadlocks: Two programs waiting for each other to release desired resource.

What is required to design a good System

Common Problems

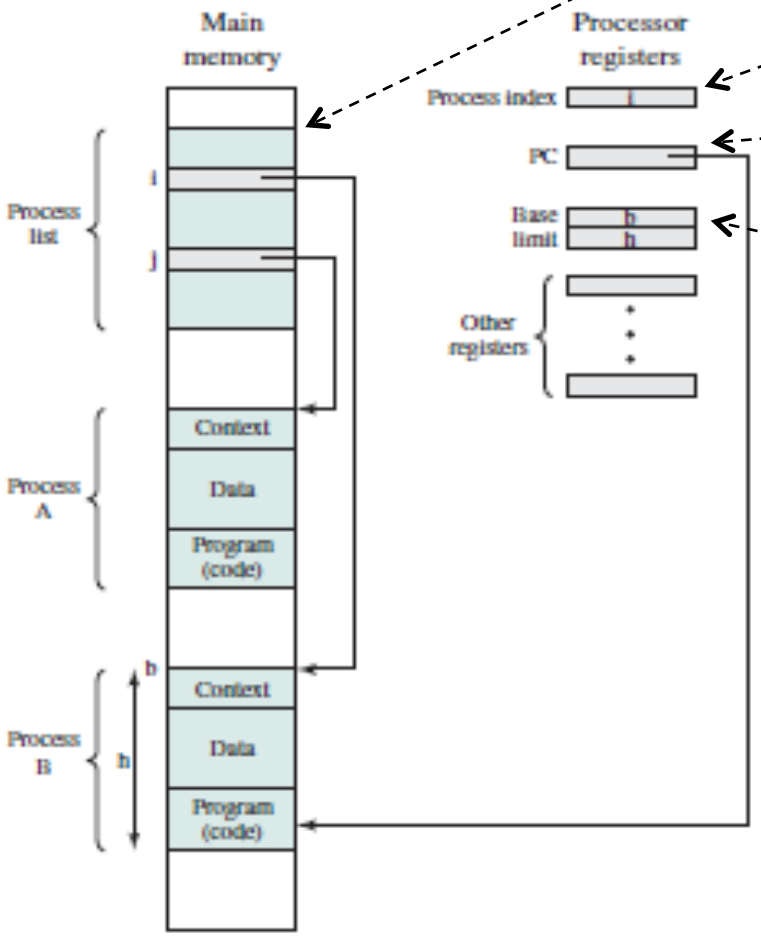
modifies shared mem space

Operating System Structures

- Design Issue: Processes (Cont)
  - Process: Foundation to systematically monitor and control programs executing on the Processor.
    - An Executable Program.
    - Associated data (variables, work space, buffers, etc).
    - Execution context of the program.
  - Execution Context:
    - Internal data used by Operating System to supervise and control the process.
    - Contents of the processor registers (Program Counter, Data Registers).
    - State of the process
      - Running or ready to run (Runqueue).
      - Uninterruptible sleep (usually I/O).
      - Interruptible sleep (waiting for event).
      - Zombie (Terminated but parent has not received status).
      - Stopped (Job control signal or being traced).
    - Priority of the process.
  - **Process is a Data Structure**.
    - Executing or awaiting execution.

Operating System Structures

- Design Issue: Processes (Cont)



Process List: One entry for each process. Pointer to the block of memory containing the process.

Process Index: Index into the Process List.

Program Counter: Next instruction in the process to execute.

Base and Limit Registers: Region in memory occupied by the process.

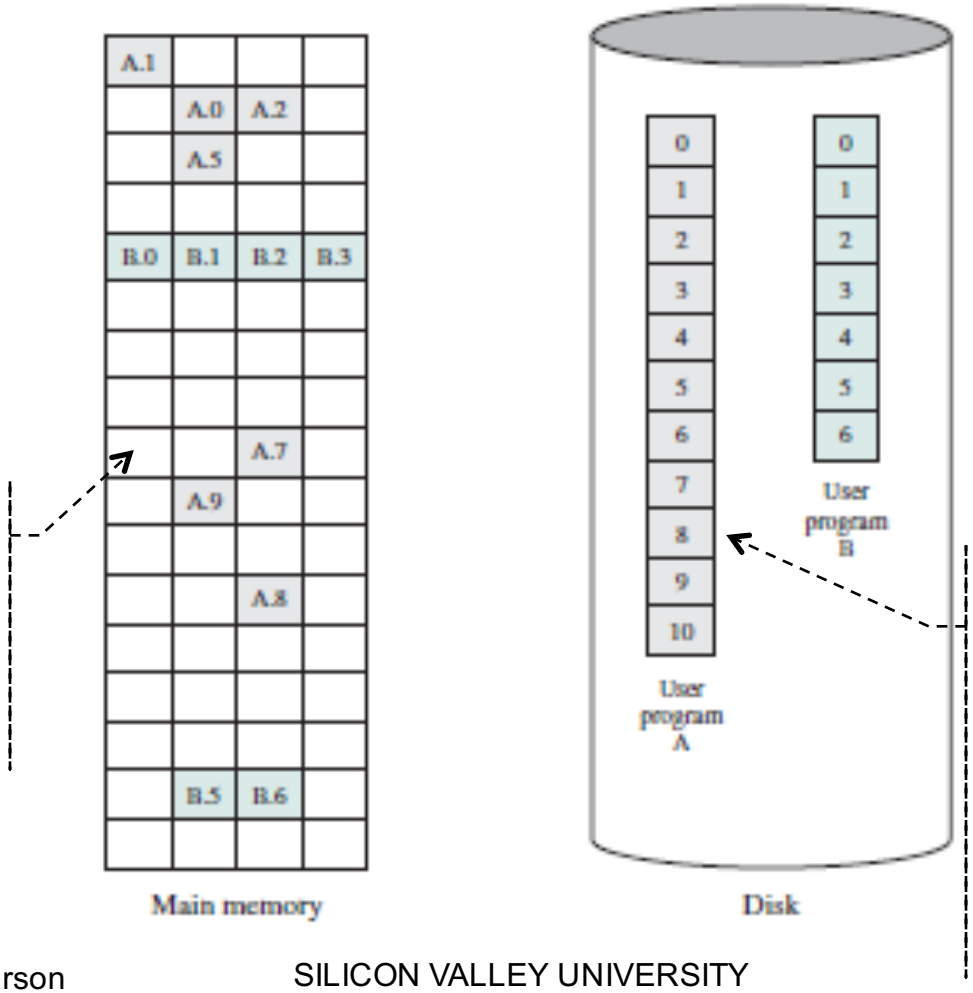
Operating System Structures

- Development of Operating System
- Design Issue: Memory Management
  - Operating System five principle Storage Management requirements:
    - Process Isolation: Prevent independent process from interfering with other's memory.
    - Automatic Allocation and Management: Programs dynamically allocated across memory hierarchy (Virtual Memory).
    - Support of Modular Programming.
    - Protection and Access Control: Support sharing of memory, when required by processes.
    - Long-Term Storage: Support storing of information persistently.
  - Requirements meet by Virtual Memory and File System Facilities.
  - Virtual Memory (MMU Memory Management Unit):
    - Facility that allows programs to address memory from logical point of view, without regard to the amount of main memory physically available.
    - Necessary to keep multiple user jobs in Main Memory concurrently.
    - Paging System divided process into fixed-sized blocks that can be located anywhere in Main Memory and on secondary storage.
    - Virtual Address: Page number and Offset within page.
  - File System:
    - Stores information in named objects, Files.
    - Files used by Operating System as unit of Access Control and Protection.

Operating System Structures

- Design Issue: Memory Management (Cont)

Main memory consists of a number of fixed-length frames, each equal to the size of a page.



Secondary Memory (disk) can hold many fixed-length pages. A user program consists of some number of pages. **Pages for all programs plus the Operating System are on disk**, as files.

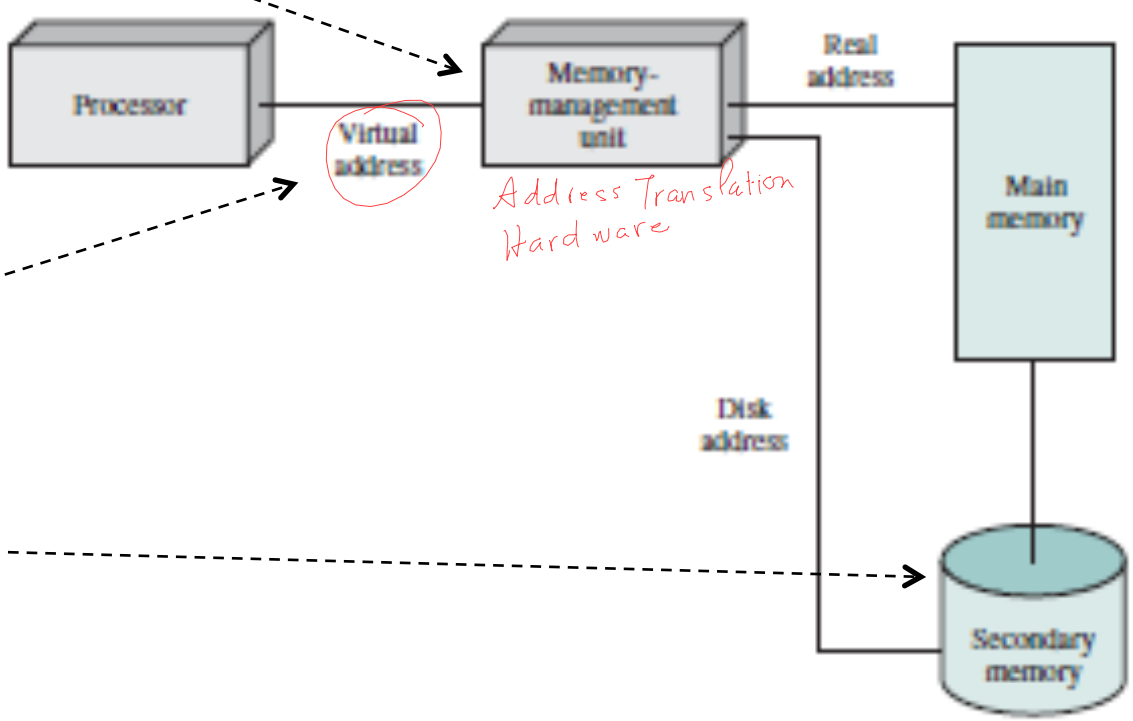
Operating System Structures

- Design Issue: Memory Management (Cont)

Address Translation Hardware (Memory Management Unit) is interposed between the processor and memory.

Virtual Address maps programs reference locations into real Main Memory addresses.

Virtual Address not in memory, portion of contents of memory is swapped out to auxiliary memory and desired block of data swapped in.



Operating System Structures

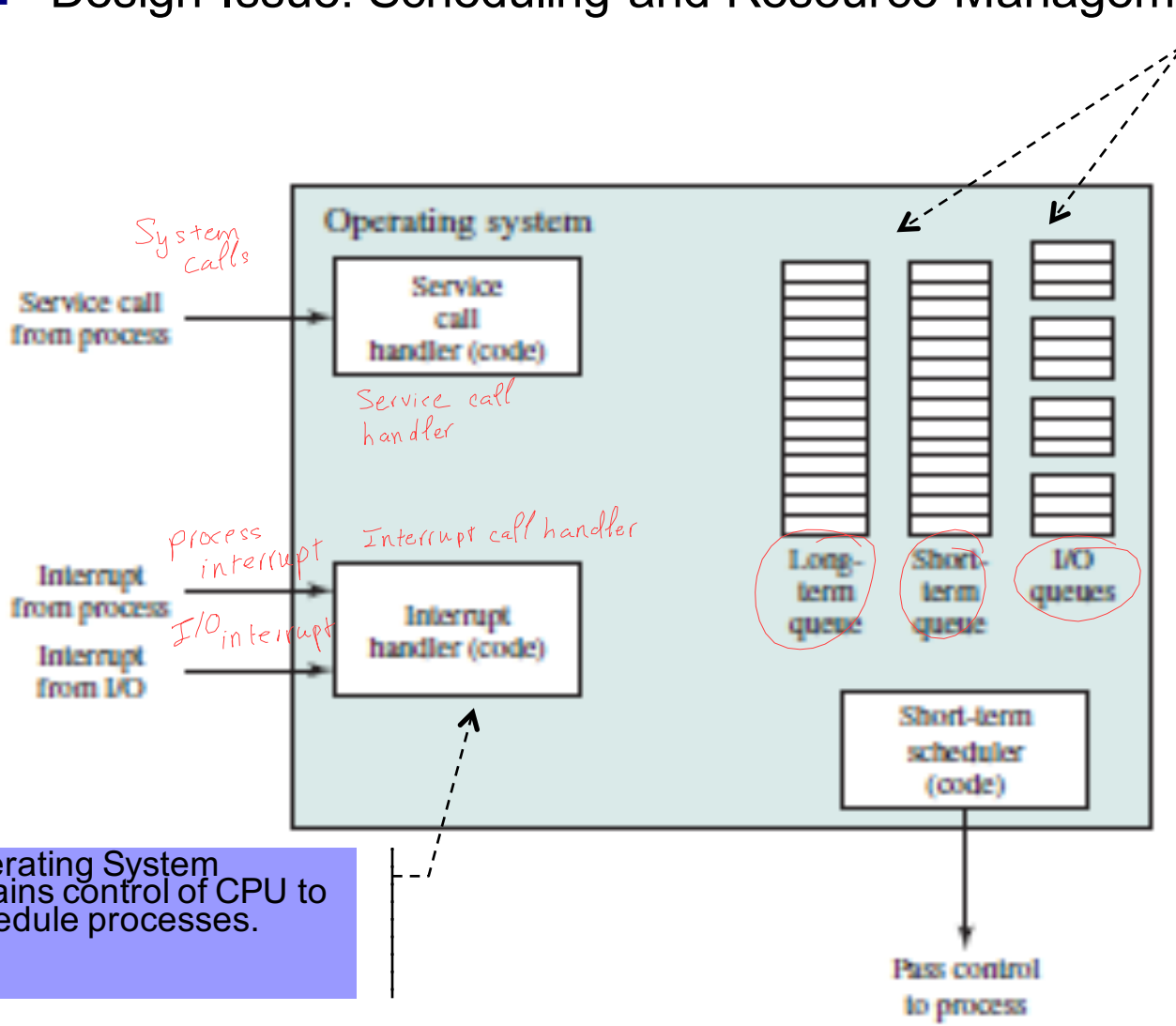
- Development of Operating System
- Design Issue: Information Protection and Security
- Growth in Time-Sharing Systems has increased concerns in controlling access to Computer System and information stored on the Computer System.
- Operating System has built-in protection and security mechanism.
- Four Protection and Security catagories:
  - Availability: Protecting the system against interruption.
    - Denial-of-Service attacks.
    - Viruses, Worms
  - Confidentiality: Protect data from unauthorized access.
    - User Authentication.
    - Access Control for File System.
  - Data Integrity: Protect data from unauthorized modification.
    - Symmetric and Asymmetric Encryption algorithms.
  - Authenticity: Verifying the identity of users and validity of messages or data.
    - Symmetric and Asymmetric Encryption algorithms.
    - Message Digest.

Operating System Structures

- Development of Operating System
- Design Issue: Scheduling and Resource Management
- Key responsibility of Operating System:
  - Manage available resources (Main Memory, I/O Devices, Processor(s)).
  - Schedule available resources use by processes.
- Resource Allocation and Scheduling Policy Factors:
  - Fairness: Processes (in same class or category) have fair access to resources.
  - Differential Responsiveness: Different service requirements require different allocation and scheduling decisions.
  - Efficiency: Maximize throughput, minimize response time, and maximize users.
- Major elements in scheduling of processes and allocation of resources:
  - Short-term Queues, Long-term Queues, and I/O Queues.
  - Queue strategies: Round-Robin, Priority Levels.
  - Interrupt Handler: Operating System receives control of the processor.

Operating System Structures

- Design Issue: Scheduling and Resource Management (Cont)



Operating System Structures

- Development of Operating System
- Design Issue: Operating System's System Structure
- Complexity requires focus on the Software Structure.
  - MAC OS X has over 85 million lines of code.
  - Windows 7 has over 40 million lines of code.
  - Linux 3.1 over 15 million lines of code (Debian Distribution over 67 million).
- Full featured Operating System problems:
  - Large code base more likely to have latent bugs that are not caught during QA testing and appear in the field. **Latent bugs must be fixed and reworked.**
  - Performance degrades after bug fixes or upgrades.
  - Complexity makes it vulnerable to viruses, worms, and unauthorized access.
- How can Operating System's System Structure be organized to limit effort in diagnosing and fixing errors?
  - Modularity: Modules with well-defined interfaces facilitates system evolution without impacting other modules.
  - Hierarchical layers and information abstraction.
    - Operating System is a series of levels.
    - Each lower level performs a related subset (more primitive) of the functions required of the Operating System.
    - Changes in one level does not require changes in another level: Decomposed one large problem into a number of more manageable subproblems.

Operating System Structures

- Design Issue: Operating System's System Structure (Cont)
- Modularity: Changes to inner working of the system more controlled.
- Layered Approach
  - Operating System divided into number of layers.
  - Bottom layer (Layer 0) is the hardware.
  - Top layer (Layer N) is the user interface.
  - Layer Functionality: Abstract object made of data and operations that can manipulate those data.
    - The Operating System layer consists of data structures and a set of routines that can be invoked by higher-level layer.
    - The Operating System layer can only invoke operations on lower-level layer.

this means a layer can invoke operations on layer below it, but not above

  - Advantages:
    - Simplifies implementation of the Operating System. Layer can only use functions and services of lower-level layer.
    - Simplifies debugging, and system verification. Once one layer is debugged, its operations is assumed correct and next layer can be debugged.
  - Disadvantages:
    - Defining the Operating System layers. Execution flow has to flow downward.
    - Overhead of calling the functions of each layer as the execution flow flows downward.

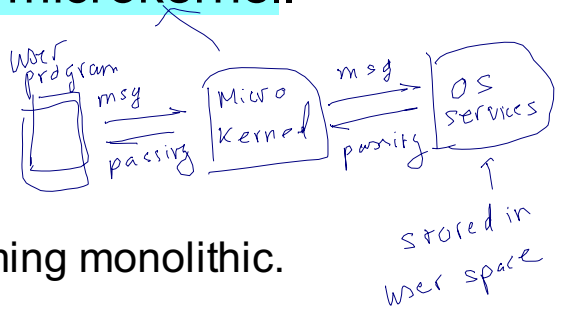
Operating System Structures

- Design Issue: Operating System's System Structure (Cont)
  - Lower layers have shorter time scale.
  - Layer 13 at user level communicates with user at slow pace (in seconds).
  - Layer 1 at hardware level communicates in micro or nanoseconds.
- Levels 13 – 8: Operating System deals with external objects
- Levels 5 – 7: Operating System and Multiprogramming support.
- Levels 4 – 1: Processor Hardware.

Level	Name	Objects
13-8	Shell, User processes, Directories, Devices, File System, Communications	User Programming Environment: Interface to Operating System for user. Process Control (Virtual Address space, parameters, process management). Mapping file name to inode to locate objects. Printers, Displays, Keyboards.
7-5	Virtual Memory, Local Secondary Store, Processes	Segments/Pages: Virtual Memory blocks moved between Main Memory and Disk. Processes, Semaphores, Ready List.
4-1	Interrupts, Procedure Call Stack, Instruction Set, Electronics	Interrupt Handler, Context Switch, Procedure Call Stack, Registers,

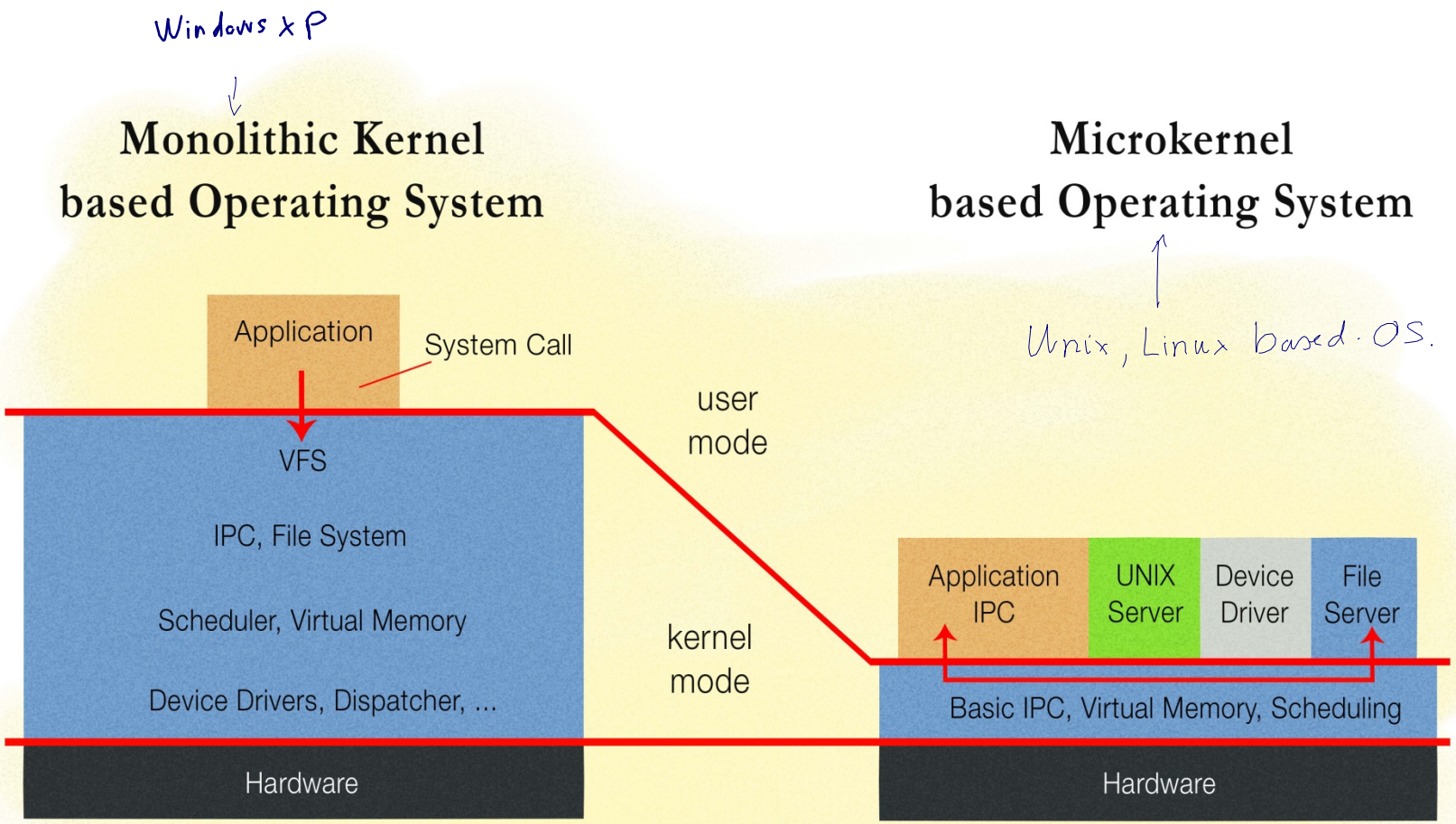
Operating System Structures

- Evolution of Operating System Structure and Capabilities (Cont)
- Microkernel
- Condensed kernel, all nonessential components of the kernel moved to user space.
  - Many Operating System services now in user space.
- Kernel supports process and memory management.
- User programs communicates with Operating System services (in user space) using message passing through the microkernel.
- Operating Systems:
  - Carnegie Mellon MACH Kernel: DEC UNIX, MAC OS X.
  - QNX: Real-time Operating System.
  - Windows NT partially microkernel.
  - Windows XP initial design was microkernel before becoming monolithic.
- Benefits:
  - Operating System extension by adding new services in user space.
  - Porting microkernel to another platform simplier.
  - More security and reliability because Operating System services are in user space (i.e. service failure in user space will not crash microkernel).
- Problems:
  - Performance degradation caused by message passing.



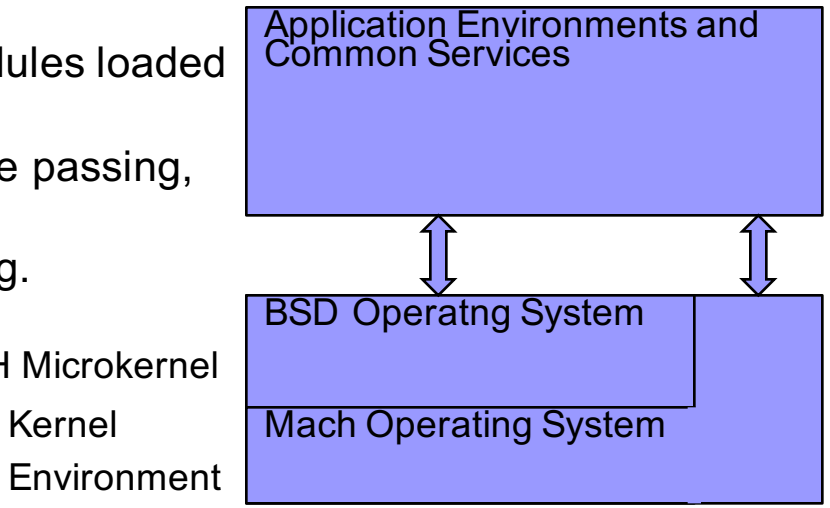
Operating System Structures

- Evolution of Operating System Structure and Capabilities (Cont)



Operating System Structures

- Evolution of Operating System Structure and Capabilities (Cont)
- Modular Kernel
  - Core Component: Links in additional modules during boot time or run time.
  - Dynamic Loadable Modules for:
    - Device and Bus Drivers for specific hardware.
    - Drivers for different file systems.
  - Similar to Layered System
    - Kernel sections has protected interfaces.
    - Modules not restricted to calling only layer below interfaces.
  - Similar to Microkernel
    - Core function with other modules loaded dynamically as needed.
    - Microkernel requires message passing, loadable modules does not need to use message passing.
    - Apple Mac OS X hybrid
      - Layered System with MACH Microkernel as one layer.

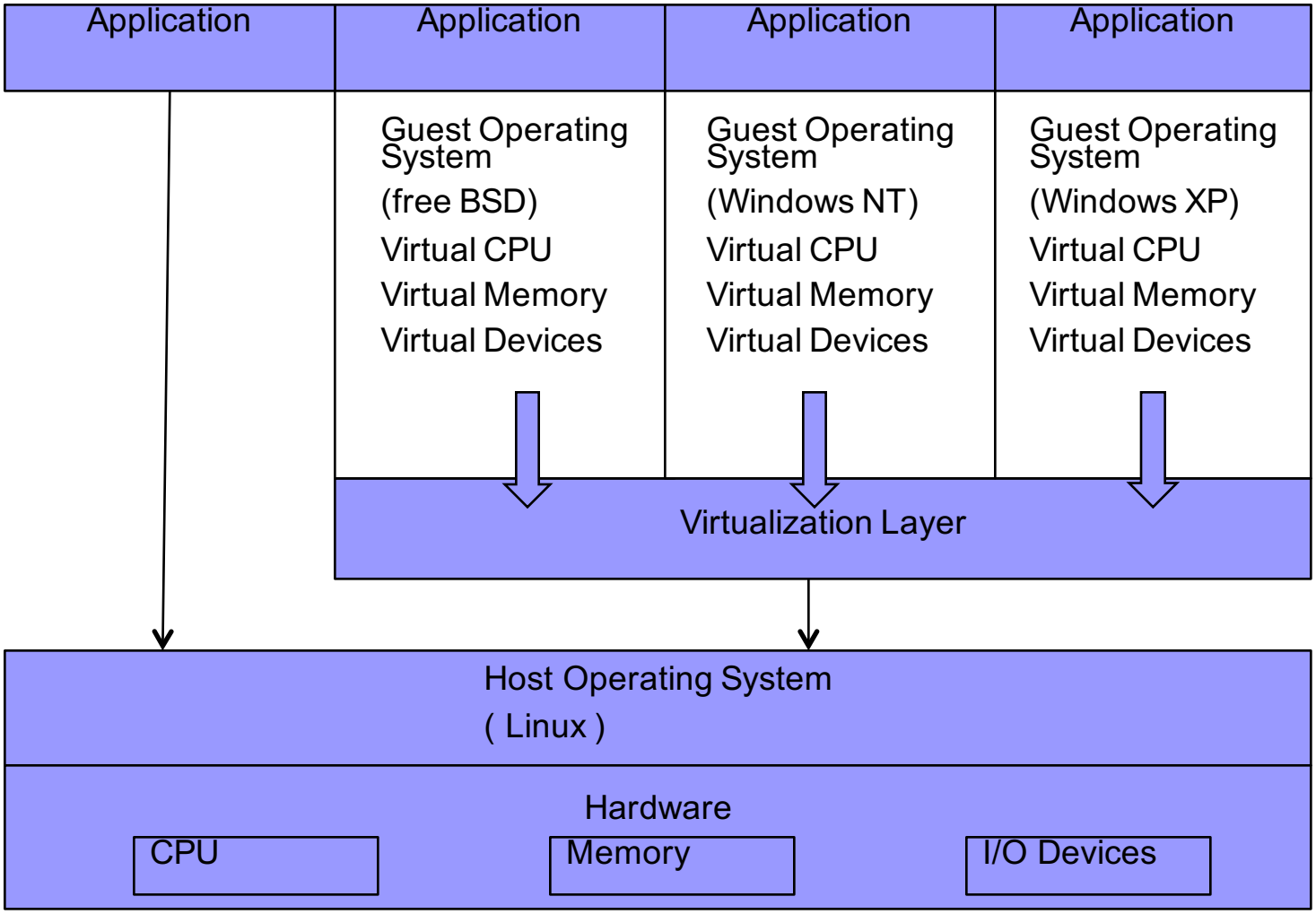


Operating System Structures

- Evolution of Operating System Structure and Capabilities (Cont)
- Virtual Machines – Layered Approach
  - Hardware abstraction of the CPU, memory, disk into separate execution environment.
- Each separate execution environment appear as one computer system.
- Benefit
  - Run several execution environment (Virtual Machines), but share the same hardware.
    - Resource optimization. Light resource systems can be combined with other light resource systems to share resources on one computer system.
    - Data Center System consolidation.
    - Do not need a computer system for each environment.
  - Host computer system is protected from the Virtual Machines.
  - Virtual Machines are protected from other Virtual Machines.
  - Ideal Operating system research and development.
    - Crash of Virtual Machine will not crash the host Operating System or other Virtual Machines.
    - System development occur along with normal host system operation, dedicated downtime for system development not needed.

Operating System Structures

- Evolution of Operating System Structure and Capabilities (Cont)
- VMware



Operating System Structures

- Evolution of Operating System Structure and Capabilities (Cont)
- Virtual Machines
- Implementation Concepts
  - Virtual User Mode and Virtual Kernel Mode switching handled by Virtual Machine Monitor (VMM).
  - Virtualization require hardware support in CPU.
    - Intel Vitualization Technology (VT) has extra instruction set (Virtual Machine Extensions).
    - Extended Page Tables (EPT) allows each Virtual Machine to have its own Page Table.
    - AMD Virtualization Technology.
- VMWare Workstation commercial application that abstracts Intel X86 and compatible hardware into isolated Virtual Machine.
  - Runs as an application under host operating systems of Windows or Linux.
  - Virtualization Layer: Abstracts physical hardware into Virtual Machines running as guest operating systems (virtual CPU, memory, disk drives, network interfaces),

Operating System Structures

- Operating System Environment
- For execution of programs.
  - Services for Users.
    - User Interface (CLI, GUI).
    - Program Execution (Load, execute program).
    - I/O Operations
    - File System Manipulations (Create/Delete/Open/Read/Write).
    - Communications (Shared memory or message passing).
    - Error Detection (Correct and consistent computing).
  - Services for to ensure effect operation of Operating System.
    - Resource Allocation (Manage multiple users, main memory).
    - Accounting (Accumulate resource usage).
    - Protection and Security (Authentication to use resources).
  - Interface made available to users and programmers.
    - System Library.

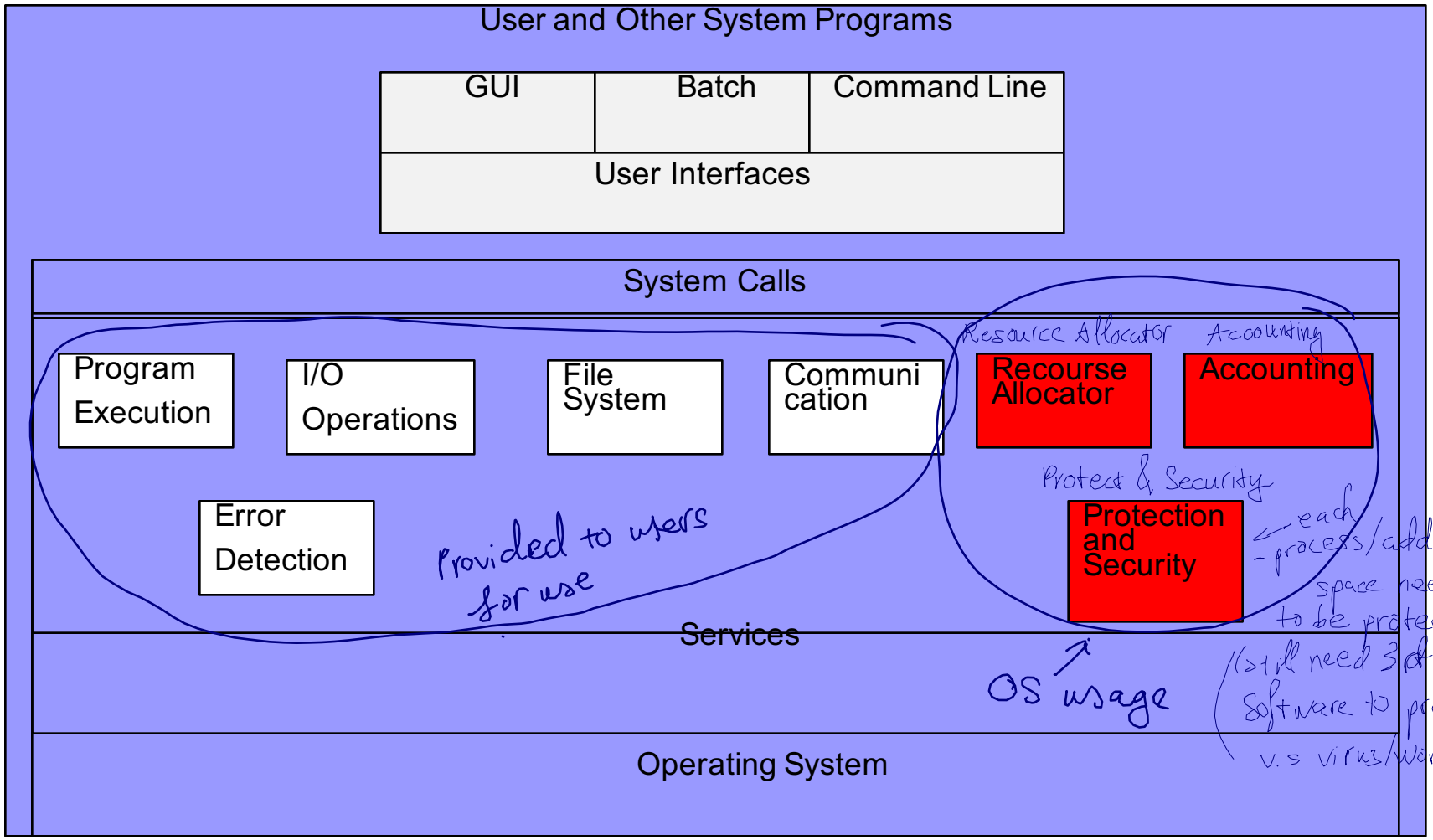
Operating System Structures

May 19th 2015
2nd class

- Operating System Services
- Services: Helpful For User:
  - User Interface – Command Line Interface, Batch Job Interface (files), Graphical User Interface (GUI).
  - Program Execution – Load program into memory, execute the program, and terminate the program normally or abnormally.
  - I/O Operations – User cannot control I/O directly, Operating System has to provide an interface through System Calls.
  - File System Manipulations – Create/Delete/Open/Read/Write files and directories.
  - Communications – Process communication via shared memory or though message passing.
  - Error Detection – Operating System must ensure correct and consistent computing,
- Services: Ensuring Effective Operation of the Operating System
  - Resource Allocation – Manages CPU cycles, main memory, file storage, for multiple users or jobs concurrently executing.
  - Accounting – Accumulating resource usage for accounting and system configuration.
  - Protection and Security – Protection ensures all access to system resources are controlled. Security requires user authentication to gain access to system resources.

Operating System Structures

- Operating System Services

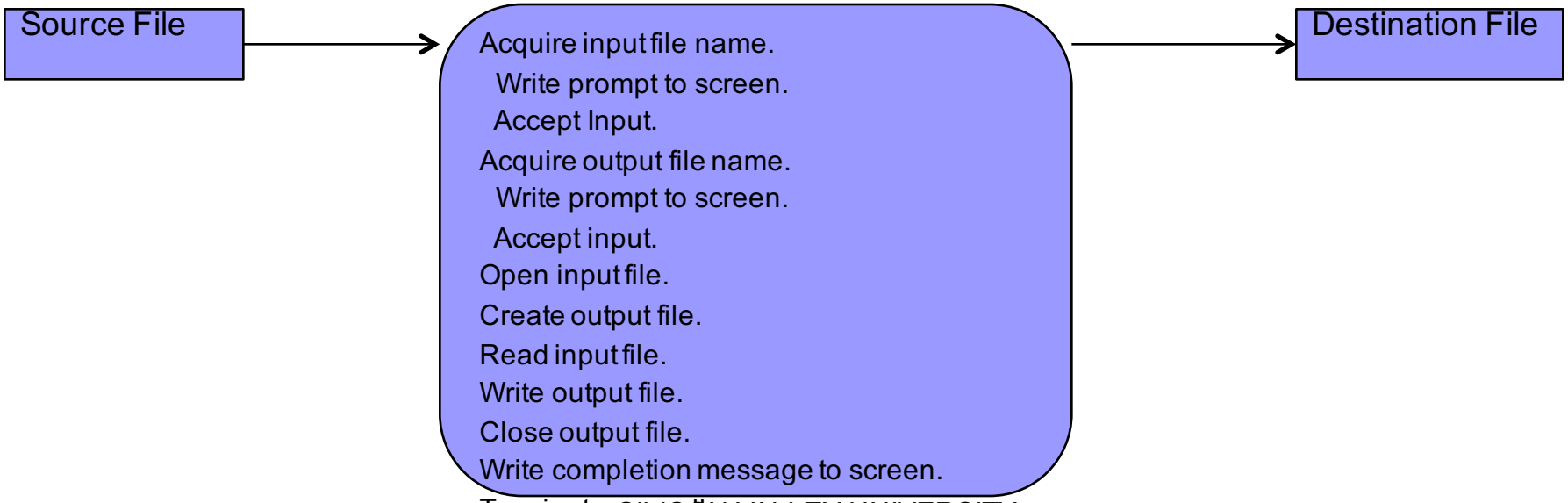


Operating System Structures

- Operating System Interface: System Calls
  - **Interface to the services** made available by an Operating System.
- Application Programming Interface (API)
  - **API (specifies a set of functions) used by application programs to make System Calls**: User does not know how system call implemented.
  - APIs available to application programmers:
    - Win32 API for Windows system.
    - POSIX API for POSIX-based systems (UNIX, Linux, MAC OS X).
    - Java API for programs running on the Java Virtual Machine.

Example

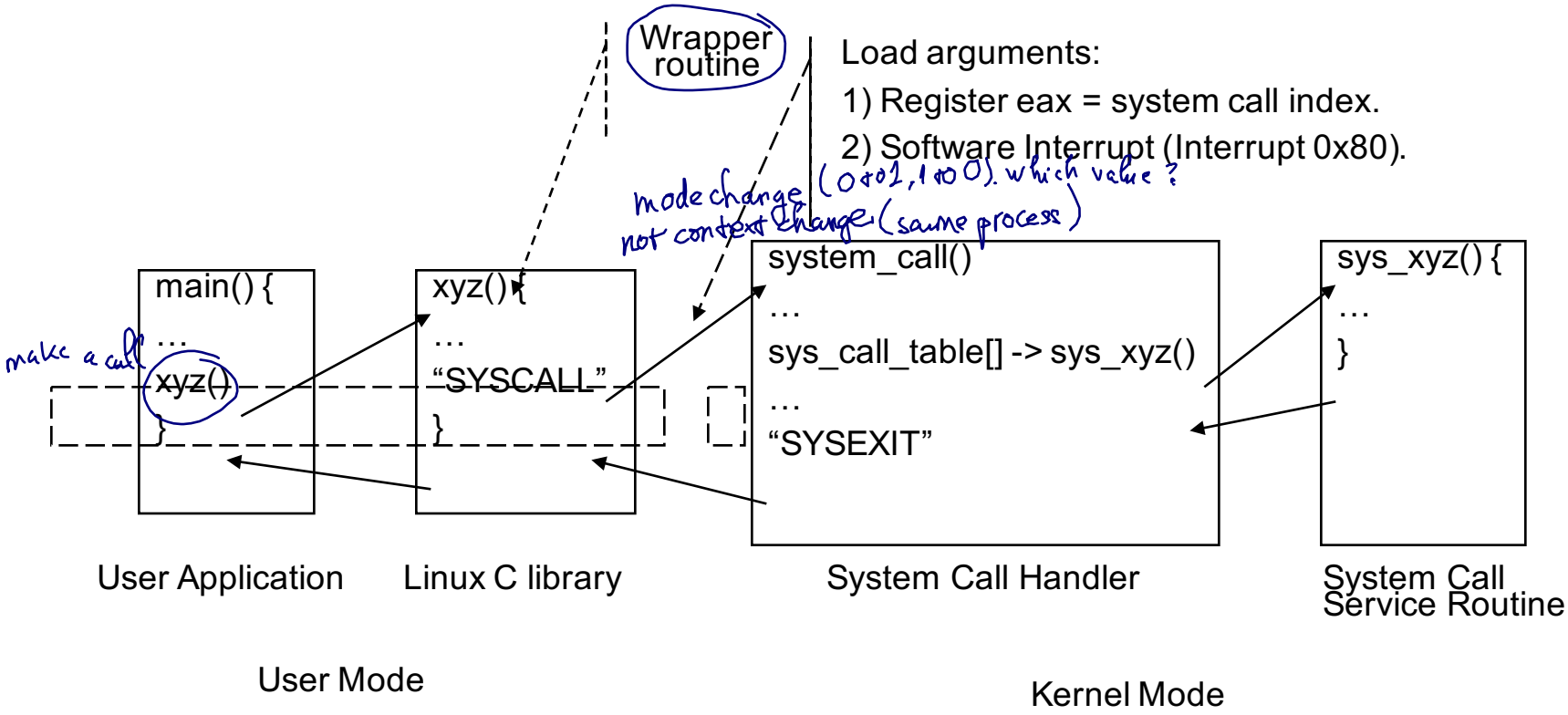
- How System Calls are used: Copy "source" to "destination" file.



Operating System Structures

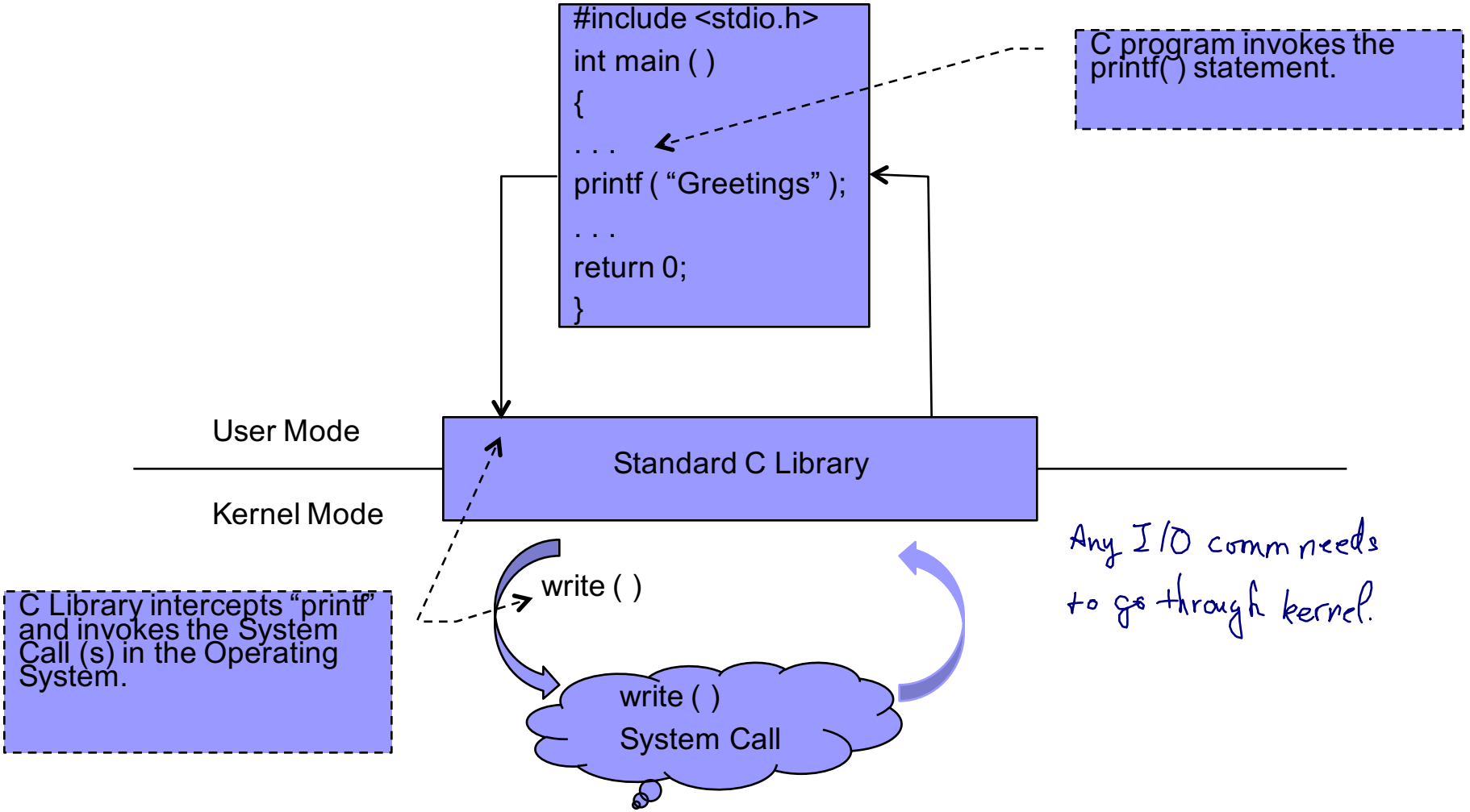
Operating System Interface: System Calls

- System Call – Explicit request to the kernel made through a software interrupt.
  Service provided in the kernel, cross the user-space / kernel boundary.
- Linux C library – Provides the wrapper routines to issue a system call.



Operating System Structures

- Operating System Interface: System Calls
- C Library using System Calls



Operating System Structures

- Windows and UNIX System Calls

Types of System Calls	Windows	UNIX
Process Control	CreateProcess()	fork()
	ExitProcess()	exit()
	WaitForSingleObject()	wait()
File Manipulation	CreateFile()	open()
	ReadFile()	read()
	WriteFile()	write()
	CloseHandle()	close()
Device Manipulation	SetConsoleMode()	ioctl()
	ReadConsole()	read()
	WriteConsole()	write()

backdoor to OS. pass special params

Operating System Structures

- Windows and UNIX System Calls

Types of System Calls	Windows	UNIX
Information Maintenance	GetCurrentProcessID()	getpid()
	SetTimer()	alarm()
	Sleep()	sleep()
Communication	CreatePipe()	pipe()
	CreateFileMapping()	shmget()
	MapViewOfFile()	mmap()
Protection	SetFileSecurity()	chmod()
	InitializeSecurityDescriptor()	umask()
	SetSecurityDescriptorGroup()	chown()

Operating System Structures

- System Programs
- System utilities provide environment to users for development and execution of application programs.
  - Some System Utilities are user interface to System Calls.
- Categories of System Utilities:
  - File Management: Programs to create, delete, copy, rename, print, dump, list and manipulate files and directories (cp, rm, mv, mkdir).
  - Status Information: Programs to return date, time, amount of available memory or disk space, performance, logging information (date, df, iostat)
  - File Modification: Text editors and programs for searching and modify text in the file.
  - Programming-Language support: Programs for development, i.e. compilers , assemblers, debuggers, interpreters for programming languages.
  - Program Loading and Execution: Programs to load user applications for execution, i.e. loaders.
  - Communications: Programs to create a virtual connection between processes and between computer systems, i.e. Web Browser, email.
- Application Programs for solving common problems:
  - Web Browsers, Word Processors, Spreadsheets, DataBase Systems, Compilers, Games (vi, emacs, gcc).
- **User impression and views of the Operating System usually seen through the System Utilities and Application Programs.**

Processes

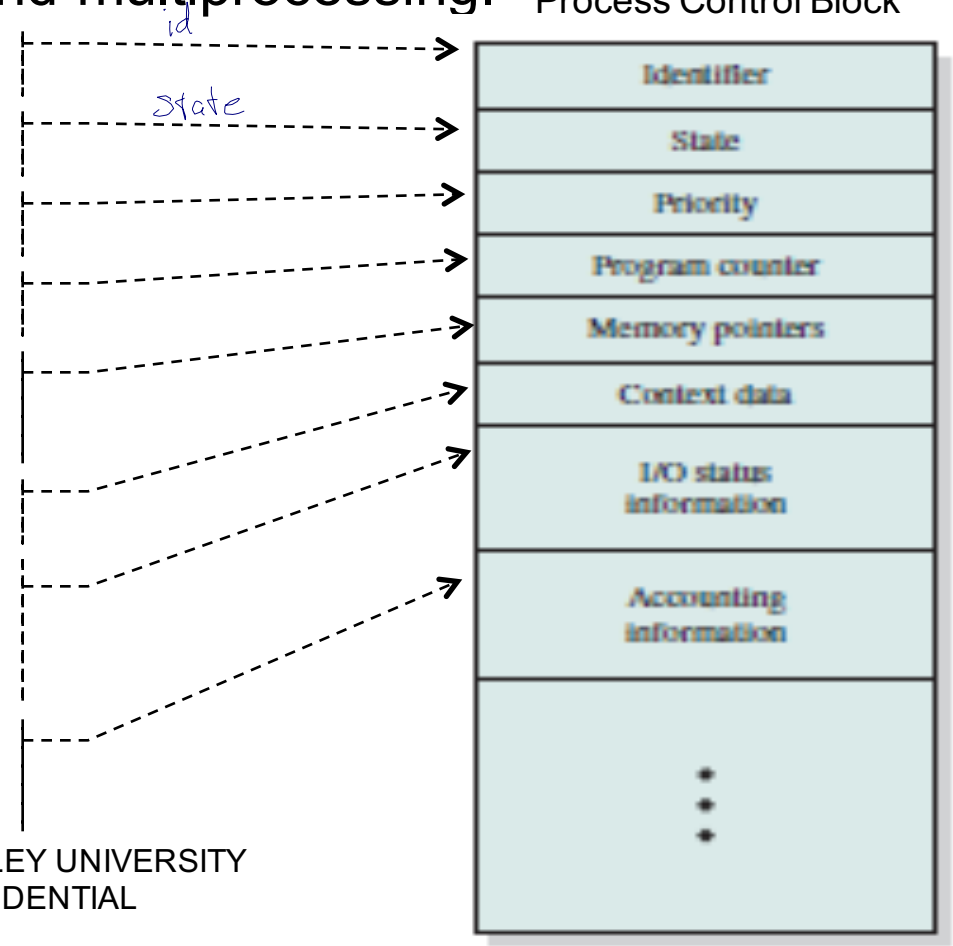
Process Control Block = Data Structure of the Process

- Fundamental Task of Operating System:
  - Process Management.
- Operating System must:
  - Allocate Resources to Processes.
  - Support InterProcess Communication and User Creation of Processes.
  - Protect Resources of Process from other Processes.
  - Interleave execution of Multiple Processes to maximize CPU.
- Operating System maintains a Data Structure for each Process. (Process Control Block : contains context of process)
  - Enables Operating System to exert Process Control.
- Difficulty of Process Management: Concept of Thread.
  - Thread Property allows concurrent execution streams running within a Process.

Processes

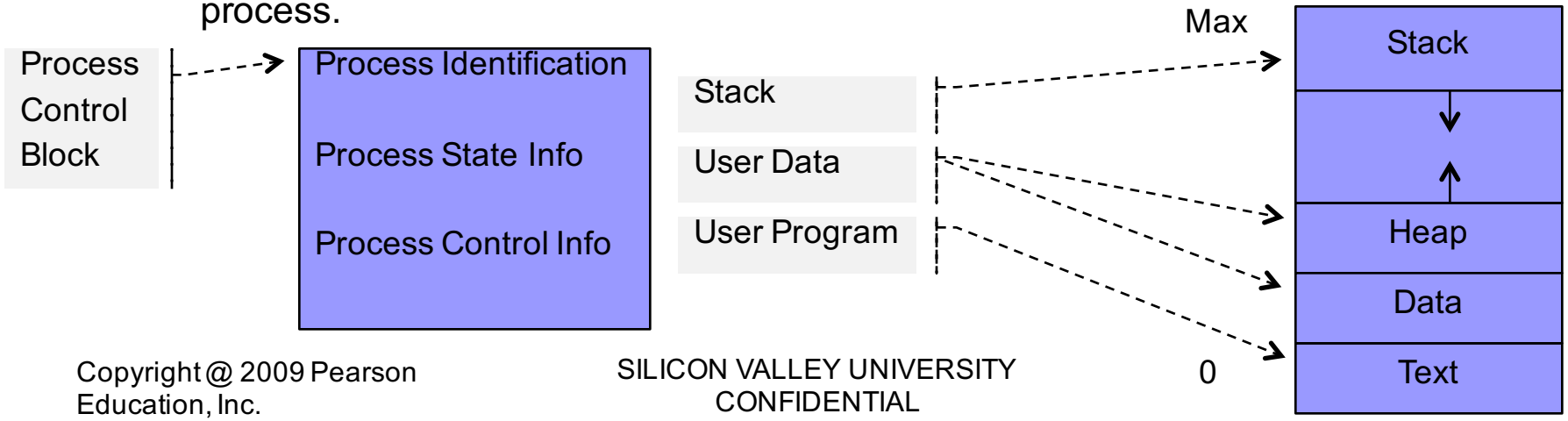
- Process Control Block (PCB)
  - Process consists of Program Code, associated Data, and PCB.
  - Executing Program is characterized by PCB Data Structure.
- PCB allows Operating System to interrupt a running process, support multiple processes, and multiprocessing.

Unique Identifier associated with process.
Running, Ready, Blocked/Waiting, New, Exit
Priority level relative to other processes.
Address of next instruction in program.
Pointer to program code, process data, shared memory blocks.
Data in Registers in the processor.
Outstanding I/O Requests, I/O Devices assigned to process, List of files used by process.
Processor time and clock time used, time limits, account numbers, statistics.



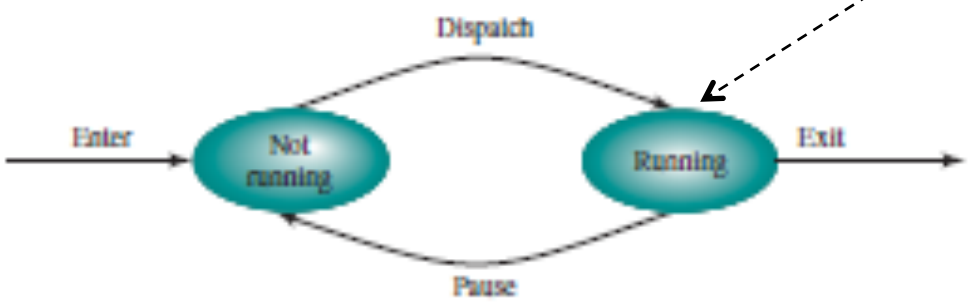
Processes

- Process Description
- Process Control Structure
  - Process Control Block (PCB) contains all of the information about a process needed by the Operating System.
  - PCB is read and/or modified by almost every module in the Operating System (scheduling, resource allocation, interrupt processing, performance monitoring and analysis).
- Process Image: Physical representation of a Process.
  - User Data: Modifiable part of User Space (Program Data, User Stack Area, and programs).
  - User Program: Program to be executed.
  - Stack: LIFO structure used to store parameters and calling addresses for procedures and System Calls.
  - Process Control Block: Data block used by Operating System to control the process.

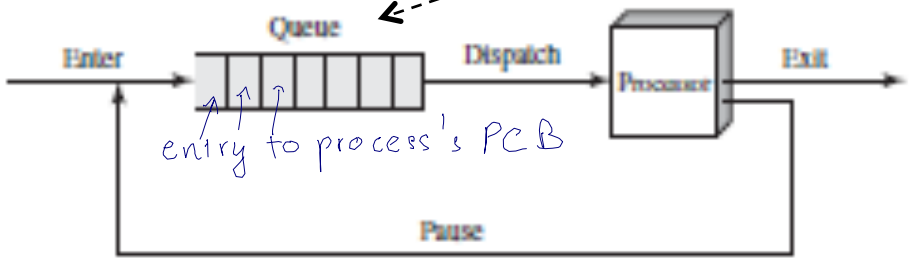


Processes

- Process Concept
- Operating System determines interleaving pattern for execution and allocating resources to process.
- Two-State Process Model



(a) State transition diagram



(b) Queueing diagram

Simple Process State, either Running or Not Running.

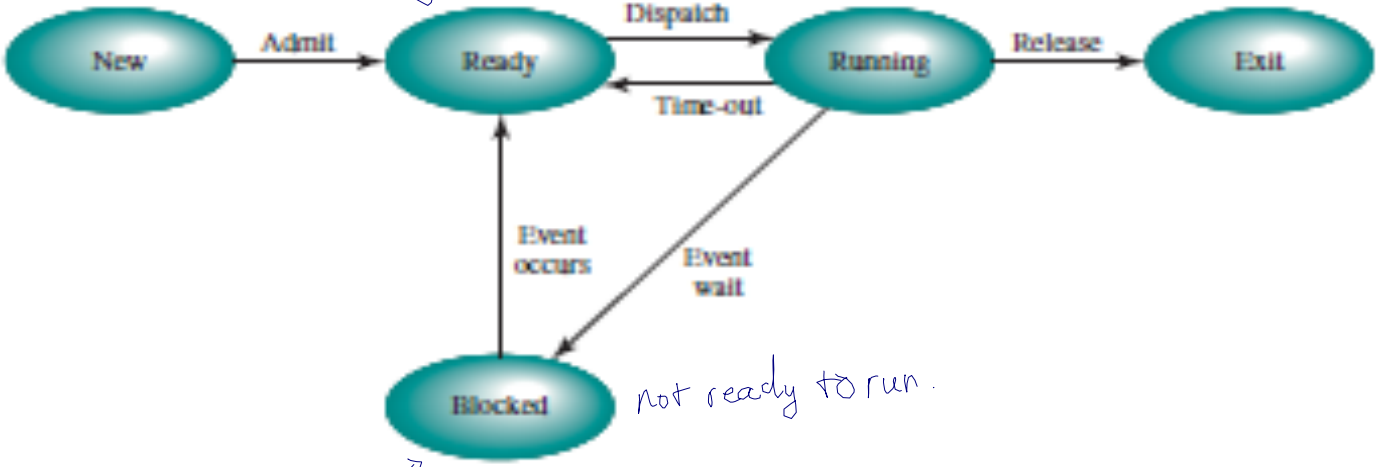
Operating System keeps track of PCB in a queue.

Queue entry is a pointer to the PCB.

Inefficent, need to search every entry in Queue to find non-blocked process to run.

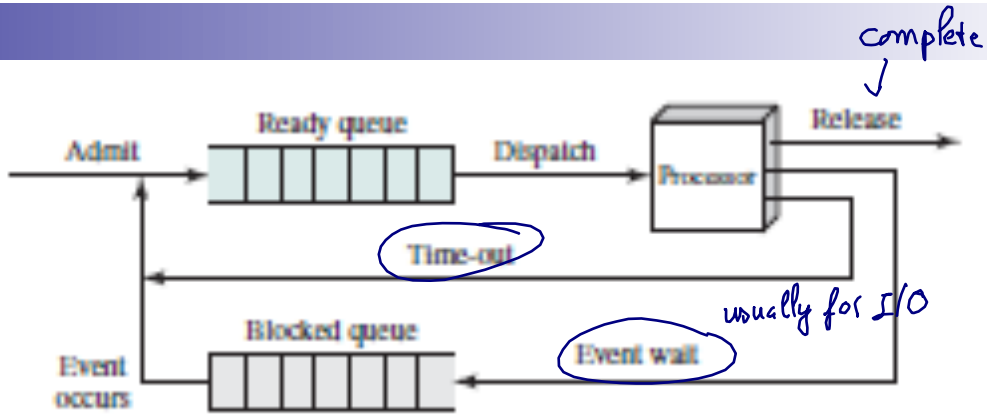
Processes

- Process Concept
- Five-State Model
  - Ready State needs additional state for process waiting (blocked) for event (I/O).
    - Queue for Blocked Processes waiting for an Event
    - Require: Queue for each Event.
  - Ready State processes are dispatched based on Priority Scheme.
    - Require: Queue for each Priority Level.



Processes

- Process Concept
- Queuing Model

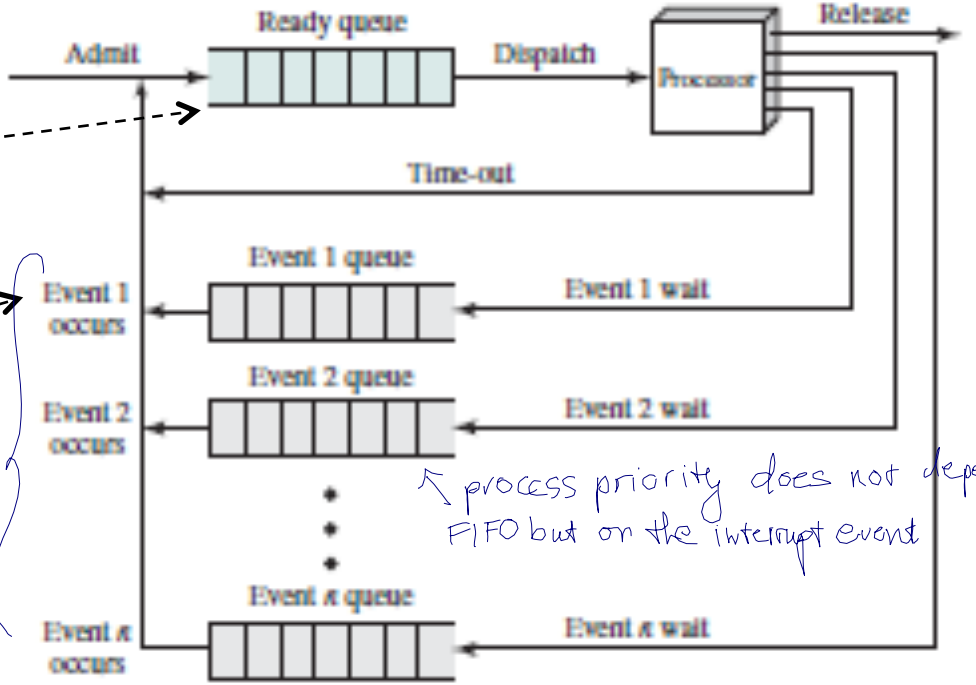


(a) Single blocked queue

Ready Queue could be expanded to queue per Priority.

Event Queue: Queue per Event.

Multiple Blocked queue



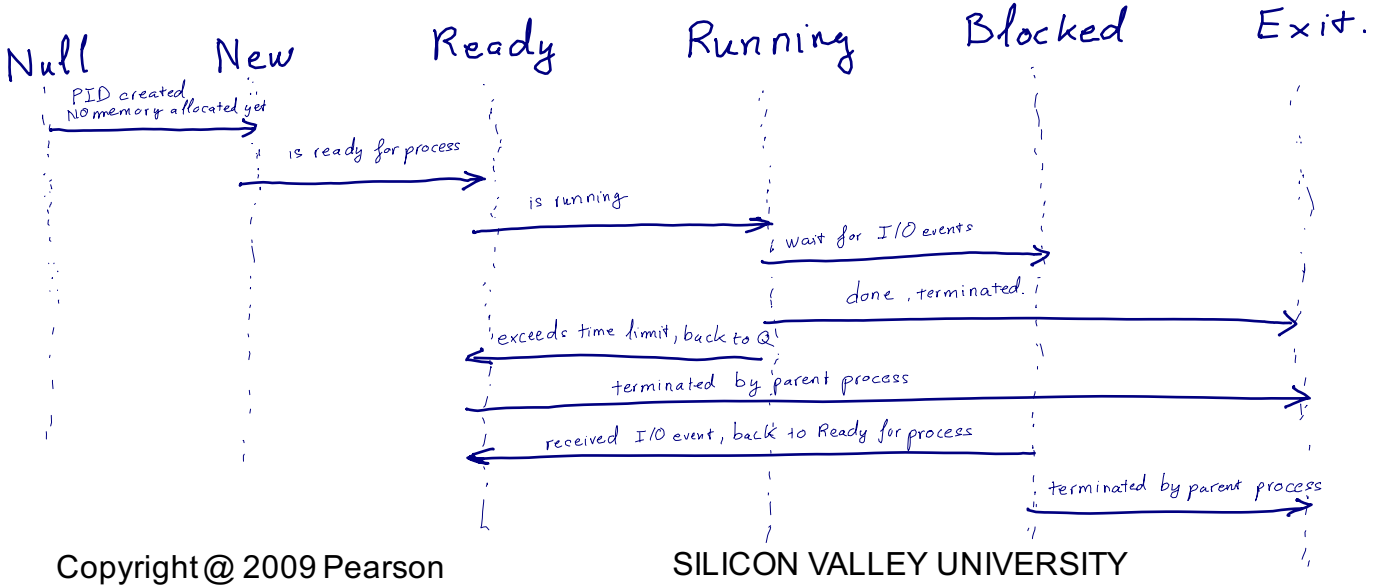
(b) Multiple blocked queues

Processes

- Process Concept
- Process State Transition
  - Null → New: process does not have enuf info to run 1st, no address space.
    - New User logs in or new Batch Job is submitted. PCB with Process ID is created, but OS has not brought the job into memory. Example: OS has limit of number of processes (prevent performance degradation) or memory not available.
  - New → Ready: ready to run
    - OS ready schedule the new process.
  - Ready → Running: running.
    - OS Scheduler selects one of the processes in the Ready state to run.
  - Running → Exit: done, terminated.
    - Process notifies OS to exits or abort. Process tables and other information temporarily preserved until OS utilities extract the needed information.
  - Running → Ready: not done, back to Q limit time
    - Process has reached its quantum and OS uses scheduling algorithm to find next process in the Ready state. OS will preempt running Process, when a higher priority process is ready to run. Process can also release processor.
  - Running → Blocked: I/O wait.
    - Process issues a request and must wait for the event. Process makes a System Call (I/O Operation, blocking for a semaphore).

Processes

- Process Concept
- Process State Transition
  - Blocked → Ready: I/O ready, back to Ready to run.
    - Event for which a process is waiting for has occurred.
  - Ready → Exit: being terminated by parent process
    - Parent process terminates a child process. Parent process terminates and all child processes also terminates.
  - Blocked → Exit: being terminated by parent process
    - Parent process terminates a child process. Parent process terminates and all child processes also terminates.



Processes

- Process Concept
- Swapping
  - Larger memory means larger processes, not more processes in memory.
  - Moving part or all of a process from Main Memory to Disk.
- Characteristics of a Swapped or Suspended State for a Process:
  - Process not immediately available for execution.
  - Process may or may not be waiting on an event. Blocking condition is independent of the suspend condition: just that the process will not be executing in the near future.
  - Process placed in Suspend State by an agent: itself, parent process, or the Operating System to prevent its execution.
    - Agent must explicitly remove the process from the Suspend State.

interchangeable

Suspend State is different from Blocked State, i.e Suspend State may or may not be caused by event waiting, suspend state process is not in memory, but in swapped disk.

Processes

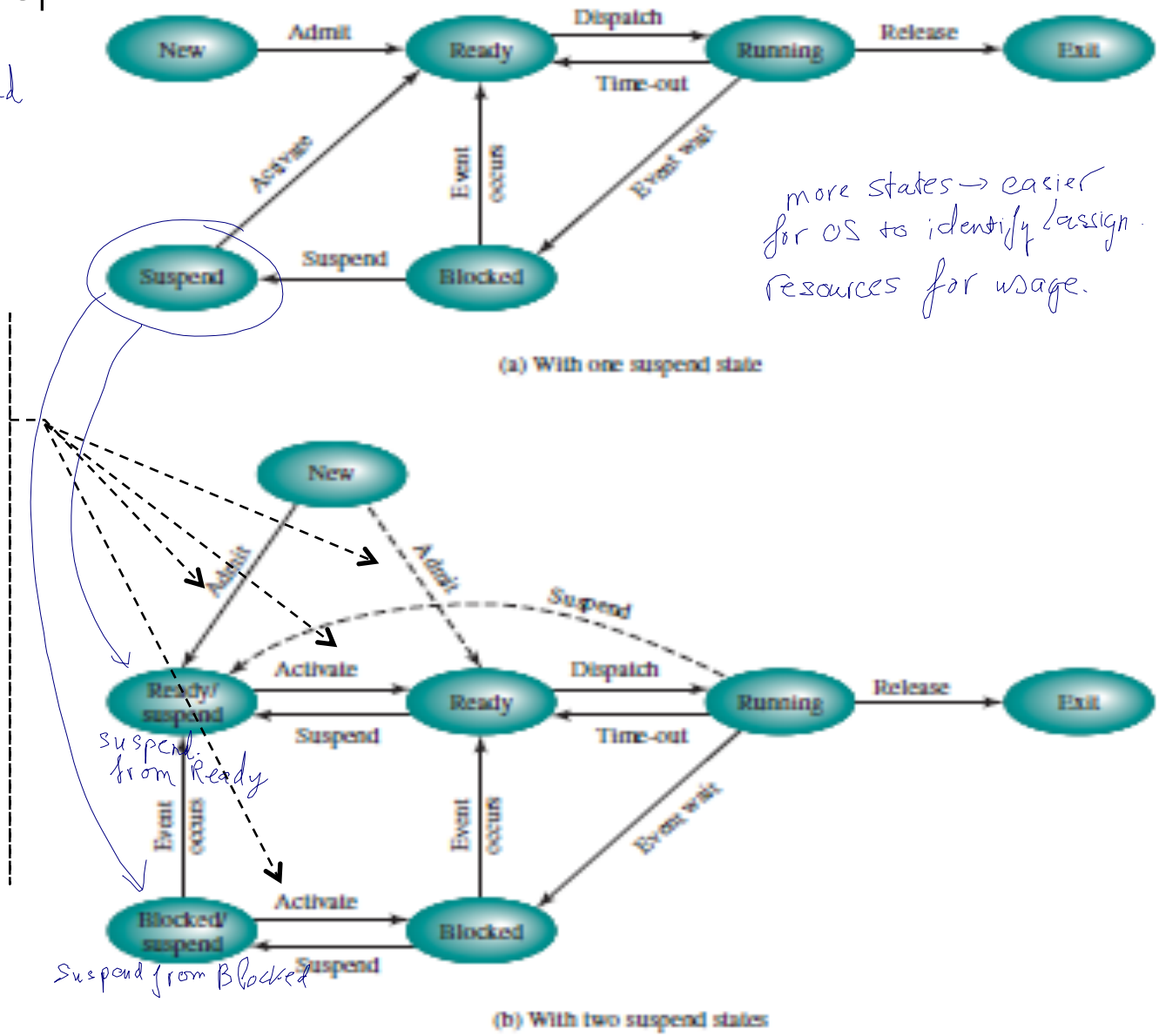
- Process Concept
- Swapping

Choices for selecting process to bring back to Main Memory:

1) Newly created process directly admitted to Main Memory or transitioned to Ready State.

2) Previously suspended process that has transitioned to Ready State.

3) Previously suspended process still in the Blocked State.



Processes

- Process Concept
- Process State Transition
  - Blocked → Blocked/Suspend: move process to swap as system needs more RAM for others.
    - Current process or Ready or Ready/Suspend process needs Main Memory, newly created process needs Main Memory.
  - Blocked/Suspend → Ready/Suspend: (when no longer wait for event, but still no free memory yet)
    - Event waited on by the Blocked process occurs.
  - Ready/Suspend → Ready: enough mem, back to the Q now
    - OS Schedular already scheduled other Ready processes or process has higher priority than other Ready processes.
  - Ready → Ready/Suspend:
    - OS needs to free up Main Memory for current process or next scheduled process.
  - New → Ready/Suspend: not enuf mem to exec yet.
    - New process cannot be added into Main Memory.
  - New → Ready:
    - New process can be added into Main Memory.

Processes

- Process Concept
- Process State Transition
  - Blocked/Suspend → Blocked: (still wait for event, but has more mem now)
    - Preemptive move when Main memory is available, and Blocked/Suspend State process has high priority.
  - Running → Ready/Suspend: when another process in Ready/Suspend or Blocked/Suspend has higher priority
    - OS preempting current process when higher-priority Ready/Suspend or Blocked/Suspend process has become unblocked and ready for execution.
  - Any State → Exit
    - Process completes execution or fatal fault condition occurs. Process can be terminated by process that created it or the parent process itself is terminated.

Processes

- Process Concept
- Process Switching
- Types of Interrupts:
  - Clock Interrupt: Operating System determines currently running process has exceeded its allowable running time (Quantum). Currently running process
    - Process State switched from Running to Ready or Ready/Suspended and another process scheduled.
  - I/O Interrupt: Operating System determines I/O action completed.
    - All blocked processes waiting for this I/O event has their Process State from Blocked or Blocked/Suspend to Ready or Ready/Suspend.
  - Memory Fault: Process Virtual Memory reference for addresses not in Main Memory.
    - Process switched to Blocked and OS switch to another process.
- Traps
  - Fatal error or exception condition.
    - Process State changed to Exit and OS switch to another process.
  - System Call.
    - Operating System service can place the Process in a Blocked state (I/O Request).

- mode switch : User mode → kernel mode.
- context switch.

- 3 types of Interrupts : Clock, I/O, Mem fault
- 2 types of Traps : fatal err/exception, system call.

Processes

- Process Concept
- Process State Change
- Operating System Environment Changes
  - Processor context (Program Counter and other Process State Information) saved in PCB. Update other PCB Process Control Information fields.
  - Process State changed to other state (Ready, Blocked, …).
  - PCB moved to appropriate queue (Ready, Blocked on Event, …).
  - Select another process for execution (Highest priority).
  - Selected process Process State changed to Running.
  - Update Process Memory Management data structures (Page Table).
  - Restore context of selected process. Program Counter restored to the instruction to be executed before the process was switched out.

Context Switch

copy back to CPU

Processes

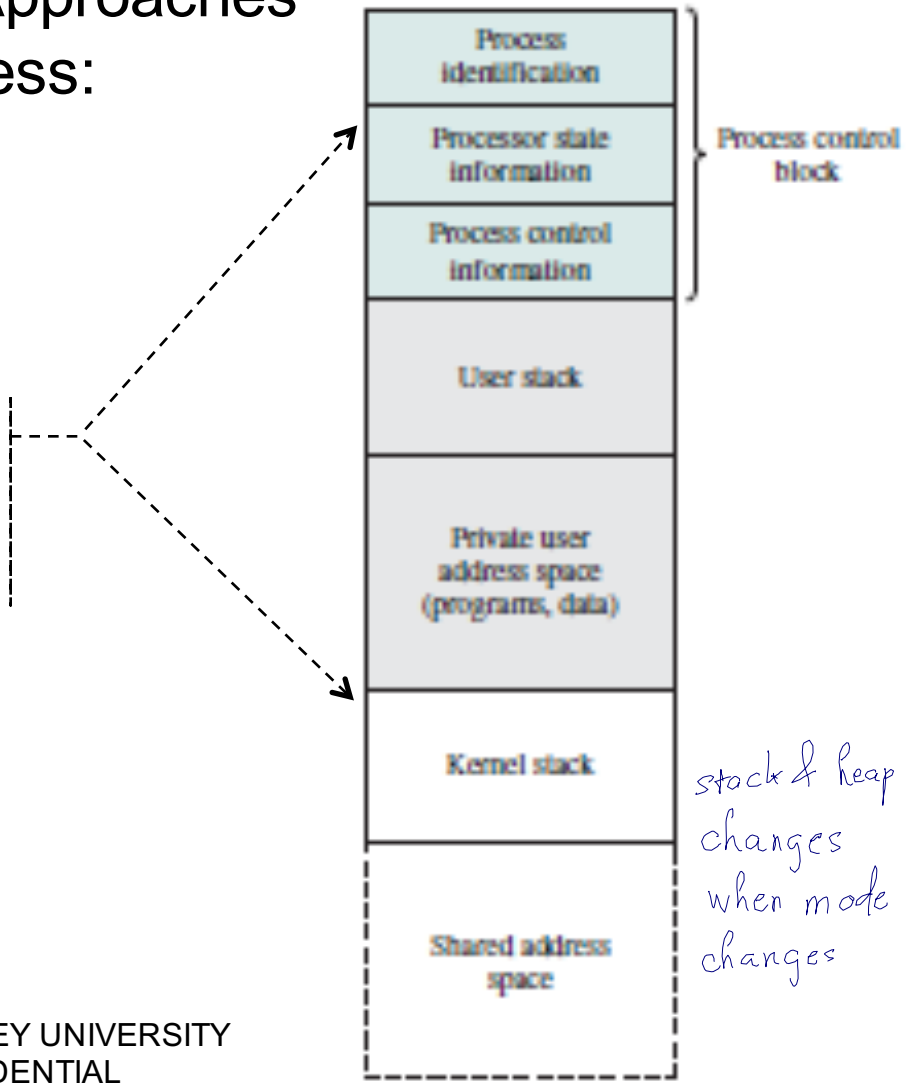
Mode Switching - user/kernel mode switching <> Process State Change. - Process running state

- Process Concept
- Mode Switching
  - Different than Process State Change (Context Switch).
  - Switch from User Mode to Kernel Mode to execute Kernel code.
- Interrupt Detection:
  - Instruction cycle, processor checks for presence of interrupt signal after executing the instruction.
  - If an interrupt is pending:
    - Sets Program Counter to Interrupt Handler program.
    - Switches from User Mode to Kernel Mode so that privileged instructions can be executed.
    - Context of the interrupted user process (the processor state information: Program Counter, registers, stack information) is saved in the Process Control Block of the interrupted program.
  - After interrupt completes:
    - If another process is scheduled (interrupt is the Real-Time Clock and the current process running time has expired), then a Process State change occurs.
    - If the current running process can resume execution, then the processor state information is restored from the PCB and the process resumes running.

Processes

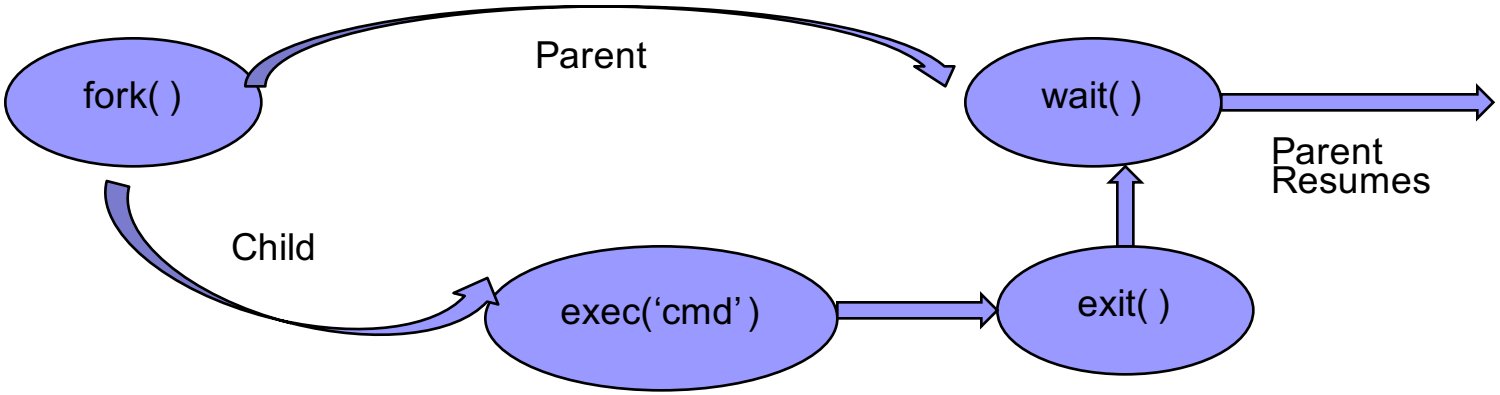
- Process Concept
- Operating System Design Approaches
- Execution within User Process: Process Image

When interrupt, trap, or System Call occurs, the processor is placed in Kernel Mode and the user mode context is saved and a mode switch takes place to an OS routine.



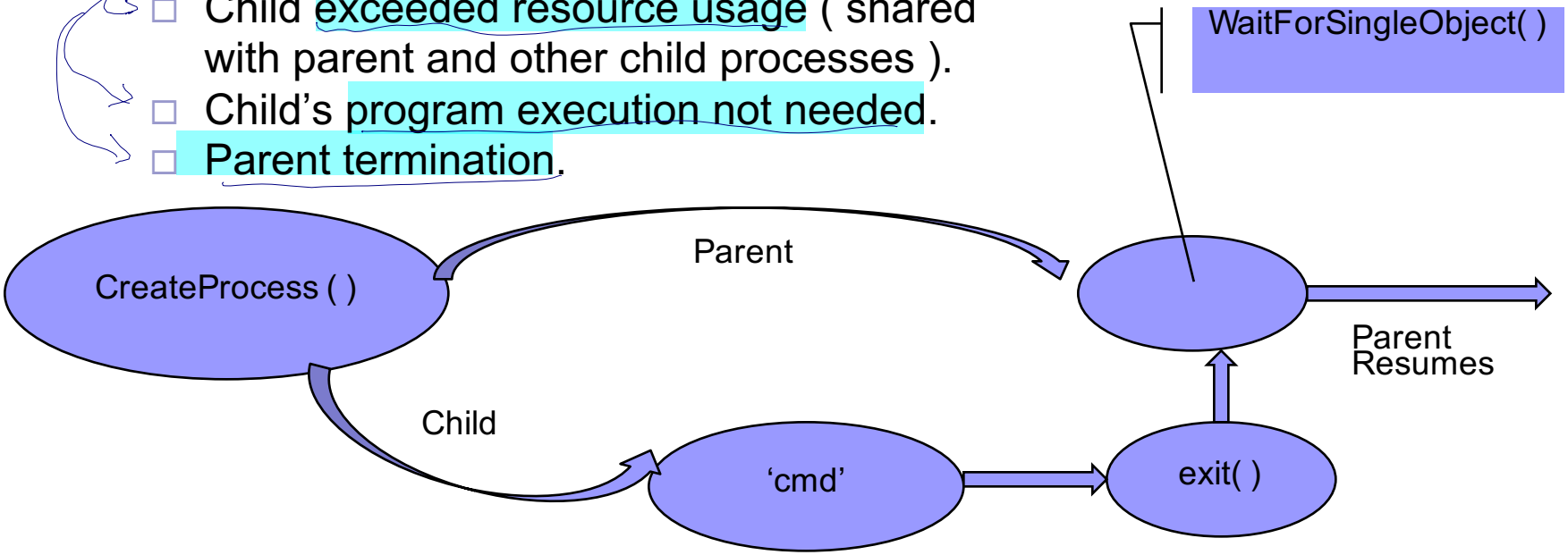
Processes

- Operations on Processes
- Process Creation
  - Process (parent process) creates new processes (child process).
  - Process Identifier (PID) uniquely identifies a process.
  - UNIX: fork ( ) System Call.
    - Created process used to execute another program in parallel (parent process continue to execute) or parent program will wait for child process to complete.
    - exec ( ) system call loads binary image into memory to execute, but loaded over the process that called exec ( ).



Processes

- Operations on Processes
- Process Creation
  - Windows: CreateProcess ( ) System Call.
    - Similar to UNIX fork( ), but CreateProcess( ) is passed, as a parameter, the program to load and execute.
- Process Termination
  - Child exceeded resource usage (shared with parent and other child processes).
  - Child's program execution not needed.
  - Parent termination.

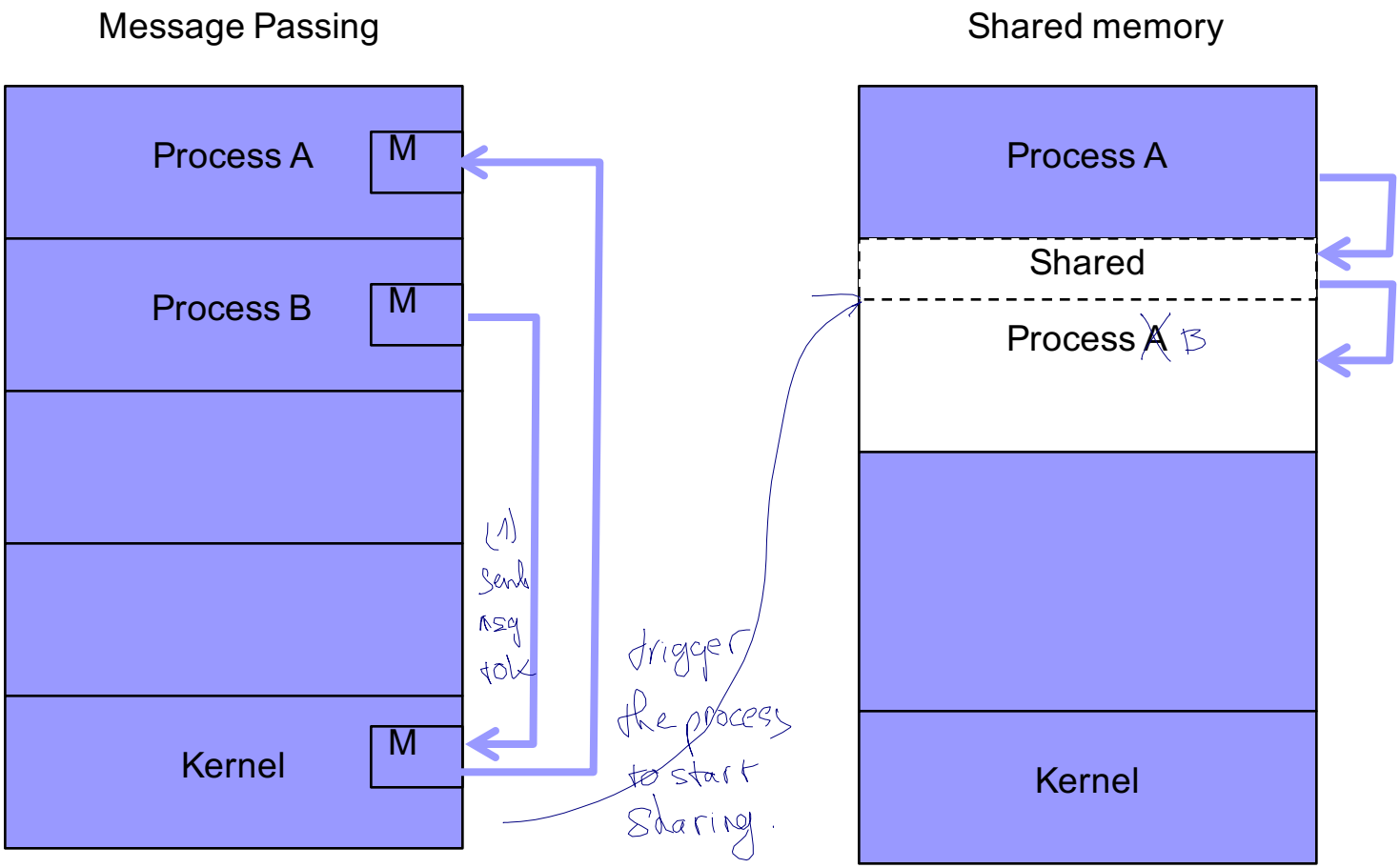


Processes

- Interprocess Communication → via message passing / via shared memory.
- Process is independent if it cannot affect or be affected by other processes in the system.
- Process is cooperating if it can affect or be affected by other processes in the system.
  - Concurrent access to shared data.
  - Computational speedup with subtasks.
  - Modularity in system design with separate processes or threads.
  - Convenience with subtasks.
- Cooperating Process and InterProcess Communcation (IPC)
  - Shared-Memory Systems
    - Region of memory shared between processes to exchange messages and data.
  - Message Passing Systems
    - Messages exchanged through System Calls to the Kernel.

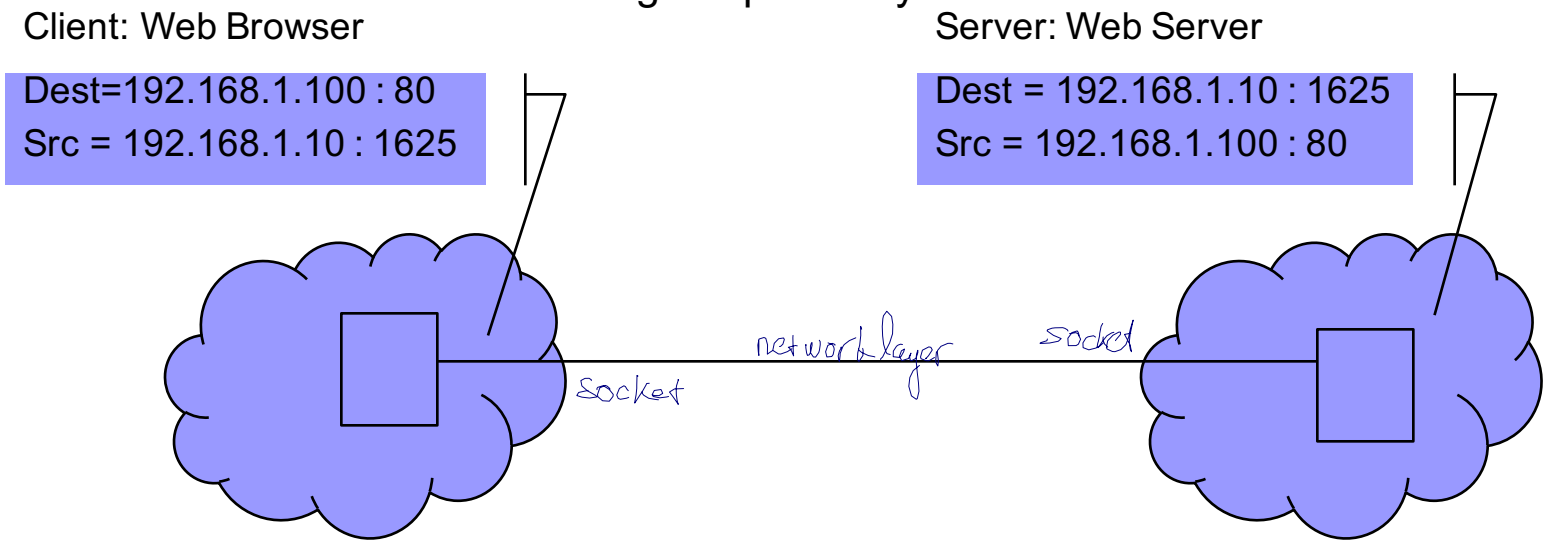
Processes

- Interprocess Communication



Processes

- Client-Server System Communications
- Sockets
  - Endpoints for communication.
  - Network IP address: port uniquely identifies connection
    - Port identifies the services to handle the request.
    - Port 23 = Telnet, Port 21 = FTP, Port 80 = HTTP.
    - Ports below 1024 are well-known for standard services.
    - Structure of message imposed by the client and server.



Processes

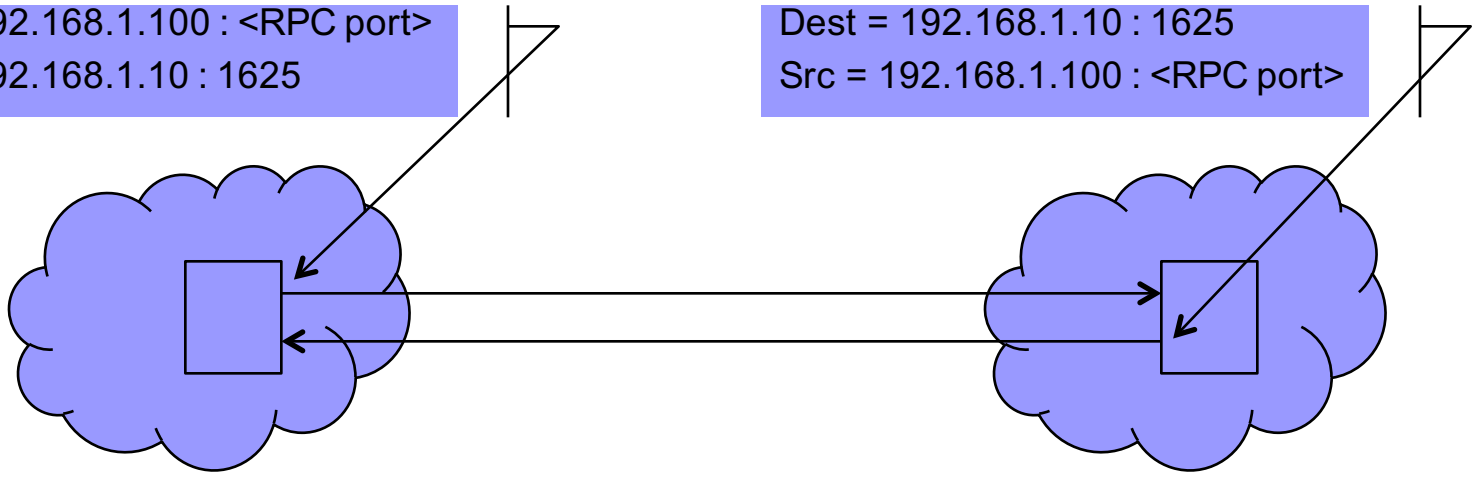
- Client-Server System Communications
- Remote Procedure Calls (RPC)
  - Abstract procedure-call mechanism to use between computer systems over a network connection.
  - Network IP address: RPC port uniquely identifies RPC program
  - RPC Port:
    - Statically defined RPC port number.
    - A daemon (program) at fixed port number that returns the RPC port number.

Client: Send command in RPC message.

Dest=192.168.1.100 : <RPC port>
Src = 192.168.1.10 : 1625

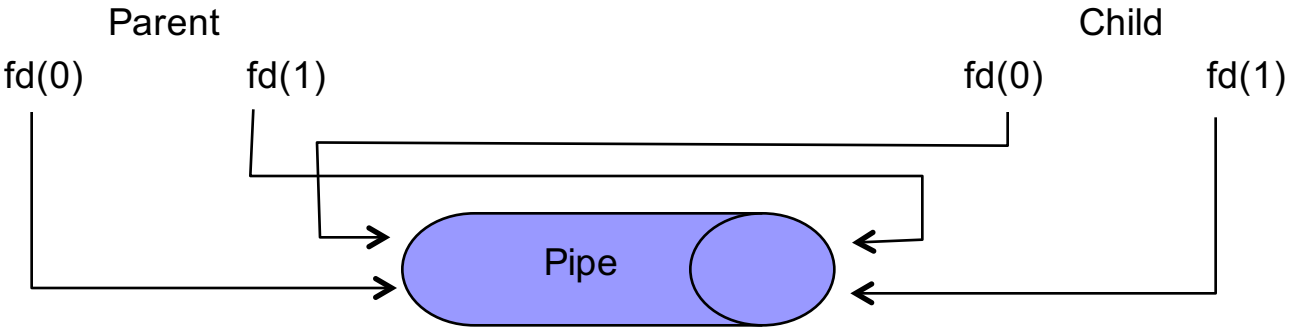
Server: Process command in RPC message.

Dest = 192.168.1.10 : 1625
Src = 192.168.1.100 : <RPC port>



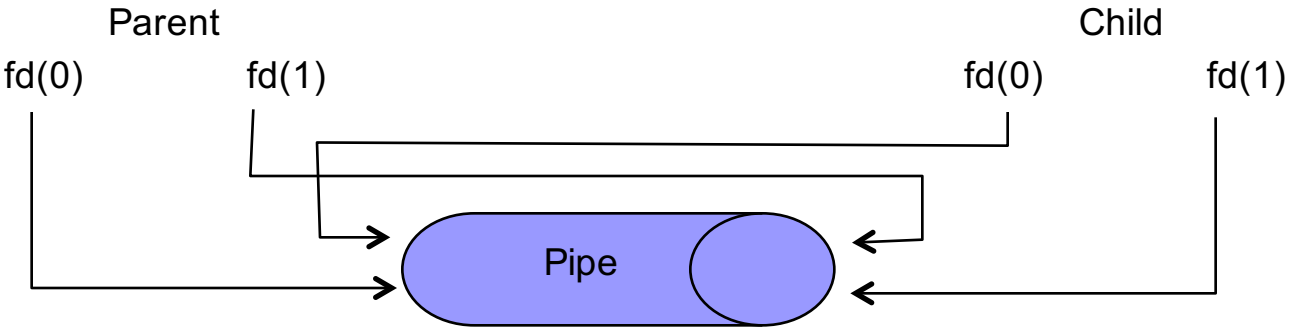
Processes

- Client-Server System Communications
- Pipes
- Conduit allowing two processes to communicate
- Ordindary Pipes → pipe goes away when process end.
  - Unidirectional, one-way communication.
  - Two-way communication require two pipes.
  - Parent-Child process relationship.
  - UNIX treats pipes as special files, requiring file descriptors.
    - pipe (int fd [ ]) command creates two file descriptors.
    - fd [0] is the read-end of the pipe.
    - fd [1] is the write-end of the pipe.
    - Use standard read ( ) and write ( ) system calls.



Processes

- Client-Server System Communications
- Pipes
- Conduit allowing two processes to communicate
- Ordindary Pipes
  - Unidirectional, one-way communication.
  - Two-way communication require two pipes.
  - Parent-Child process relationship.
  - UNIX treats pipes as special files, requiring file descriptors.
    - pipe (int fd [ ]) command creates two file descriptors.
    - fd [0] is the read-end of the pipe.
    - fd p 1] is the write-end of the pipe.
    - Use standard read ( ) and write ( ) system calls.

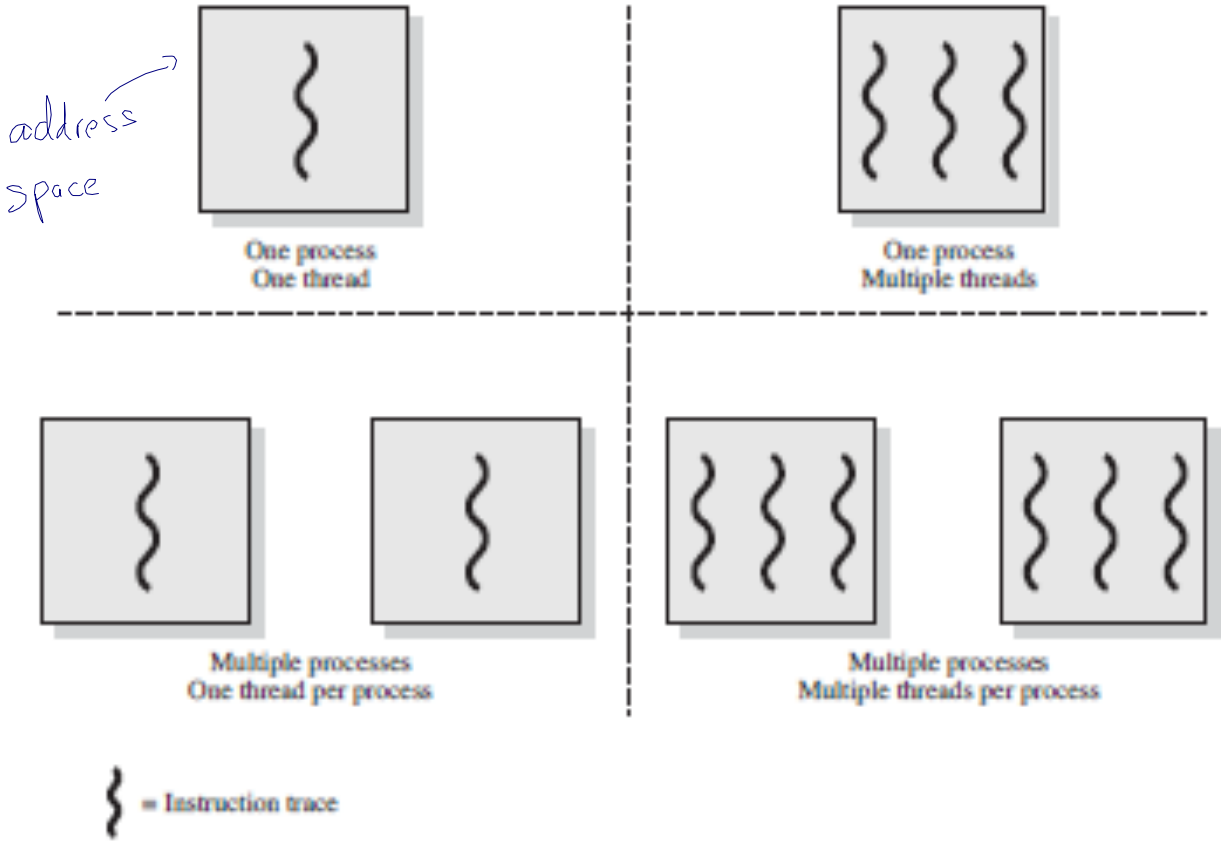


Processes

- Client-Server System Communications
- Pipes
- Named Pipes
  - Bidirectional, but only half-duplex is permitted.
  - No Parent-Child process relationship.
  - Named pipes continue to exist after processes exits.
  - UNIX treats pipes as special FIFO files, requiring file descriptors.
    - mkfifo (char *pathname) command creates file.
    - Use standard open ( ) to get file descriptor and then use read ( ) and write ( ) system calls.
    - Both sides of fifo file must be opened, else one side blocks on open until another process opens the fifo file.
    - Processes must reside on same computer system.
  - Windows bidirectional, full-duplex permitted.
    - Processes can reside on the same or different computer systems.
    - CreateNamedPipe ( ) creates the named pipe.
    - ConnectNamedPipe ( ) connects to named pipe.
    - ReadFile() and WriteFile() used to manipulate the named pipe.

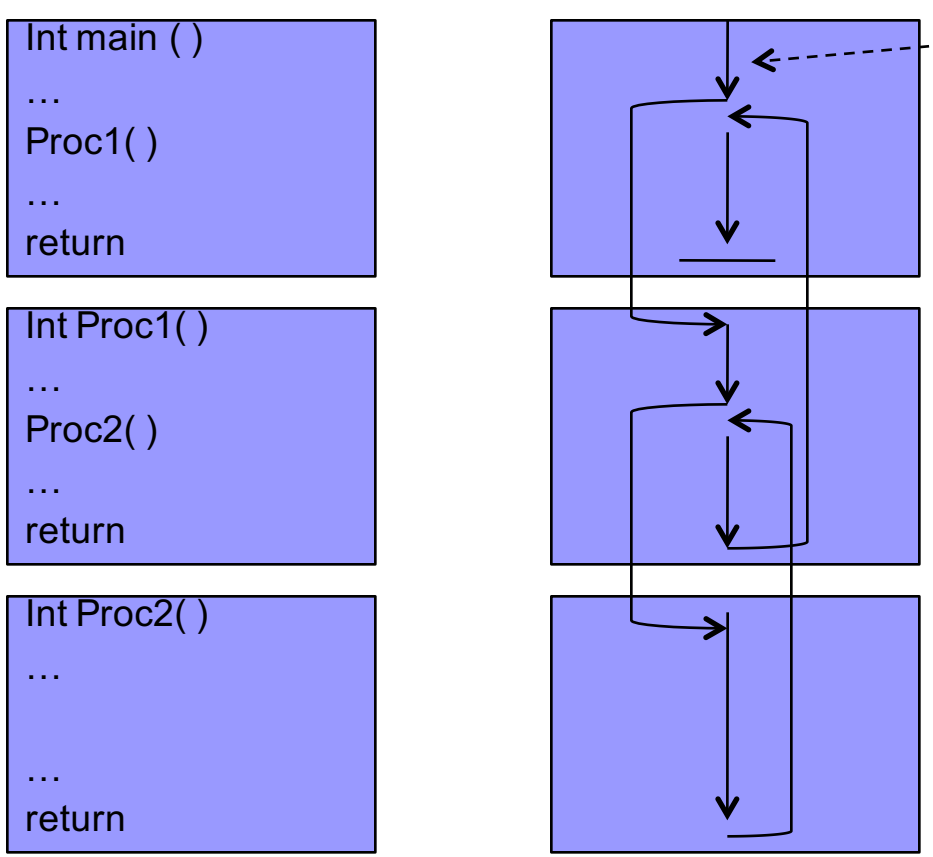
Threads

- Thread: Path of execution within a Process.
- Multithreading: Ability of OS to support multiple, concurrent paths of execution within a single process.



Threads

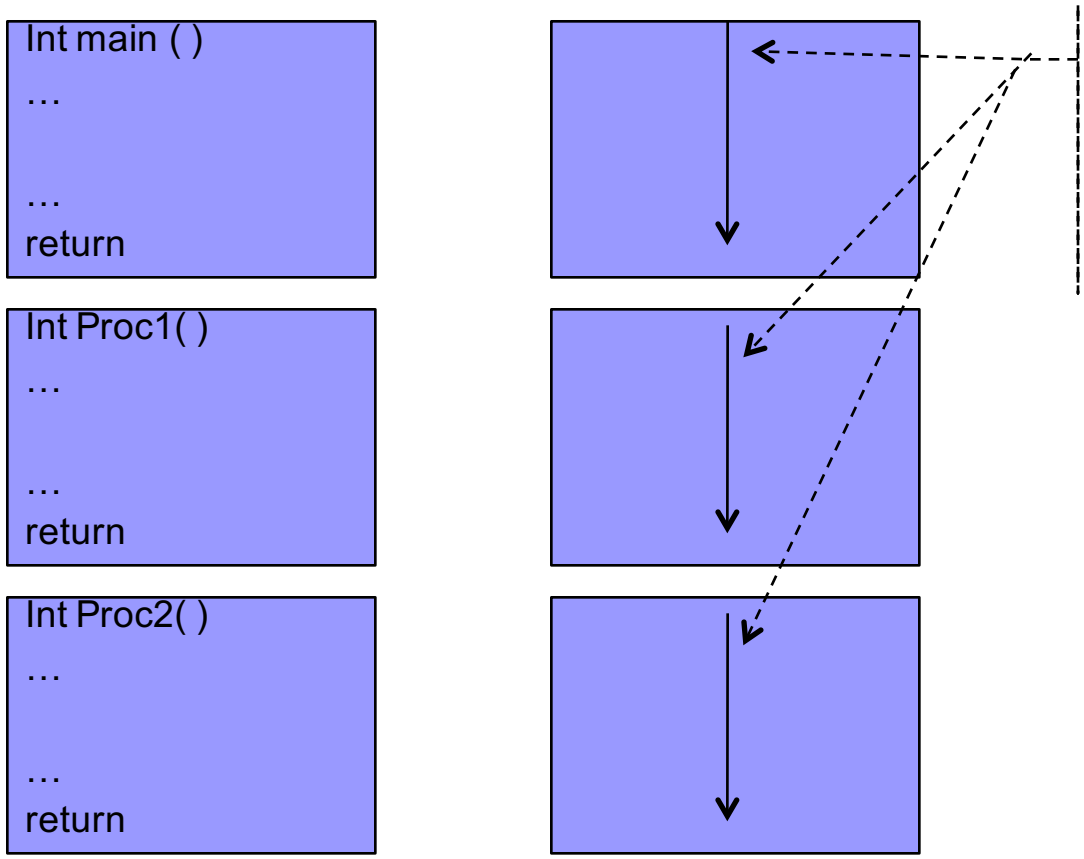
- Execution Thread
  - Single thread of execution per process.



Execution of a process follows an execution path through one or more programs.

Threads

- Execution Thread
  - Multithreading is the ability of an OS to support multiple, concurrent paths of execution within a single process.

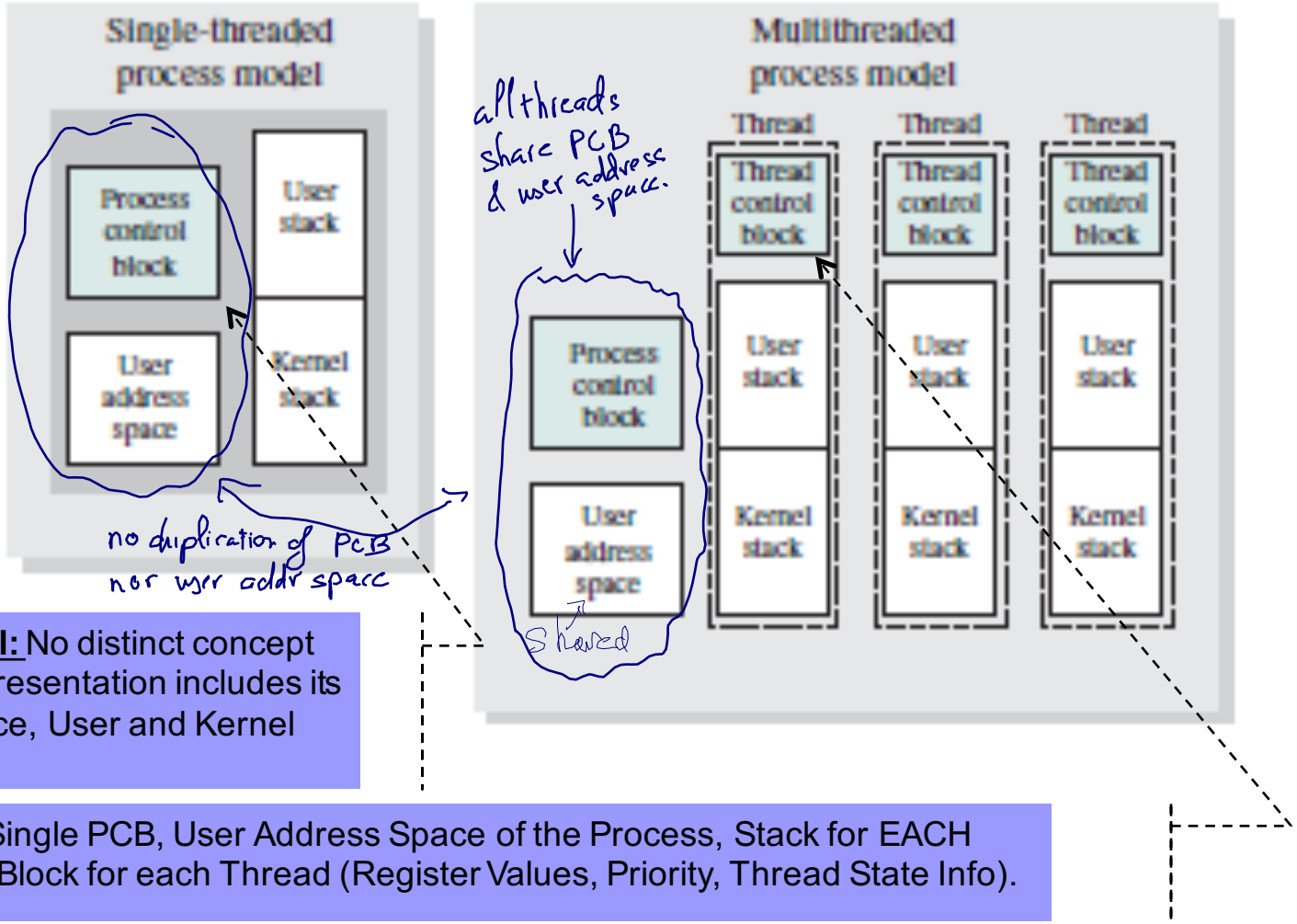


Execution of a thread follows an execution path, but over separate processors.

Concurrent execution of different threads of a process on separate processor.

Threads

- Process Manager view of Threads and Processes.

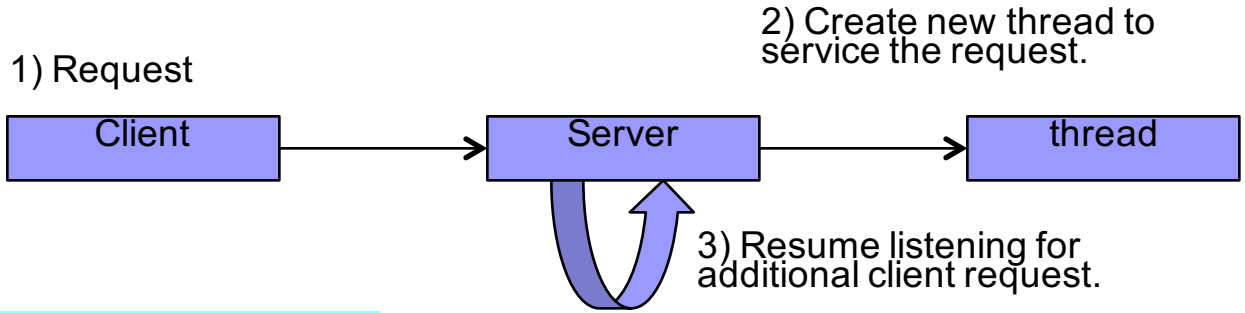


Single Threaded Model: No distinct concept of a thread. Process representation includes its PCB, User Address Space, User and Kernel Stacks.

Multithreaded Model: Single PCB, User Address Space of the Process, Stack for EACH Thread, Thread Control Block for each Thread (Register Values, Priority, Thread State Info).

Threads

- Multithread Process
- Benefits
  - Responsiveness
    - Allow a program to continue, even if it is blocked or executing lengthy operation.
    - Web Server can create separate thread listening for connections. When new connection is made, Web Server will create a new thread to service the new connection.



  - Resource Sharing
    - Threads share the memory and resources of the parent process.
    - Application can have several different threads of activity within the same address space.
    - Word processor can have thread displaying graphics, thread responding to user keystrokes, thread for performing spelling and grammer checking in the background.

Threads

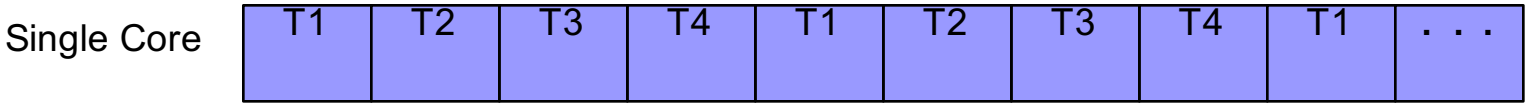
- Multithread Processing
- Benefits
  - Economy
    - Threads share the resources of the parent process.
    - More economical to create and context-switch threads.
    - Process creation and process context-switch slower than thread creation.
  - Scalability
    - Threads may run in parallel in multiprocessor architecture.
    - Multithreading on a multi-CPU machine increases parallelism.

Threads

- Multithread Processing
- Multicore Programming
  - On system with multicore, concurrency means that the threads can run in parallel, as the system assigns a separate thread to each core.
  - Areas present challenges in programming multicore systems:
    - Dividing Activities: Application must find areas that can divided into separate concurrent tasks.
    - Balance: Concurrent tasks must perform equal work of equal value.
    - Data Splitting: Data accessed and manipulated by concurrent tasks must be divided to run on separate cores.
    - Data Dependency: Concurrent tasks must be synchronized to accommodate data dependency.
    - Testing and Debugging: Testing and debugging concurrent tasks are inherently more difficult than testing and debugging single-threaded tasks.

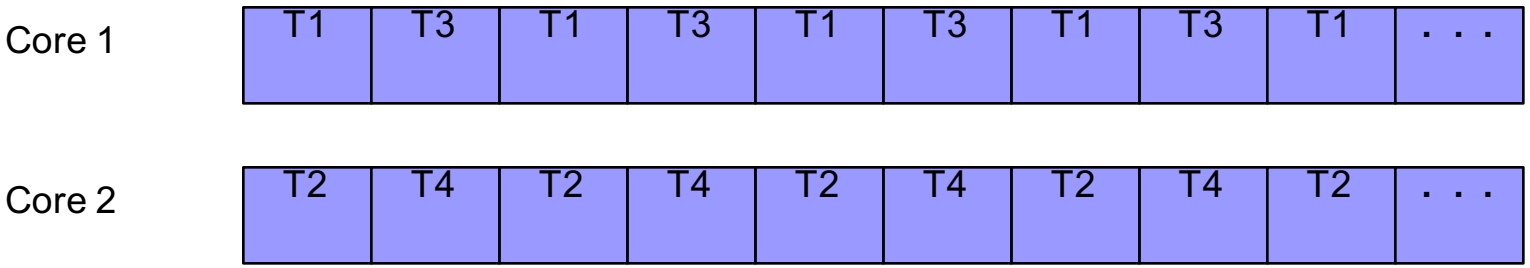
Threads

- Multithread Processing
- Multicore Programming



Single Core

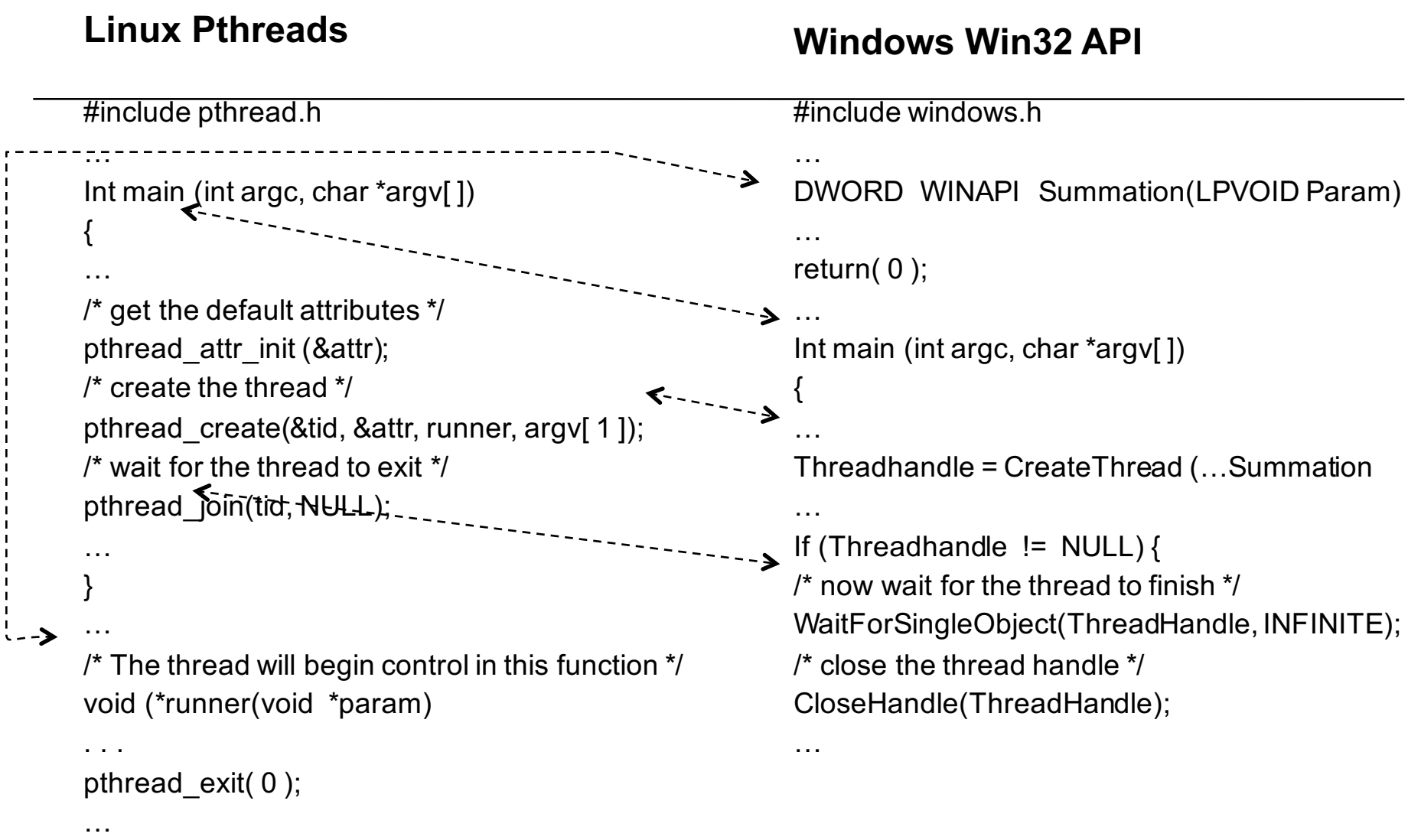
Concurrency: Execution of the threads interleaved over time.



Multiple Core

Concurrency: Threads run in parallel, Operating System assigns separate thread for each thread.

Threads



Threads

- Linux Operating System Threads
  - Linux Threading: clone()
    - fork(): process

params used in process creation

      - CLONE_CHILD_CLEARTID: erase child thread ID when child exits.
      - CLONE_CHILD_SETTID: store child thread ID.
      - SIGCHLD: Termination signal sent to parent when child dies.
    - pthread_create(): thread

params used in thread creation

      - CLONE_FILES: share the same file descriptor table.
      - CLONE_VM: share the same memory space.
      - CLONE_FS: share file system information
      - CLONE_SIGHAND: share same table of signal handlers.
      - CLONE_THREAD: share the same thread group as the calling process.
      - CLONE_SYSVSEM: share a single list of System V semaphores.
      - CLOSE_SETTLS: Set Thread Local Storage descriptor.
      - CLONE_PARENT_SETTID: store child thread ID.
      - CLONE_CHILD_CLEARTID: erase child thread ID when child exits.

Threads

- Linux Process Vs Threads

```
int main()
{
…
    pid_t pID = fork();
    If (pID == 0) {
        sIdent = “Child Process: “;
…
    } else if (pID < 0) {
        /* Failed to fork. */
        exit(1);
    } else {
        sident = “Parent Process:”;
    }
…
}
```

fork:

Spawn a new child process which is an identical process to the parent, except that it has a new system process ID. The process is copied in memory from the parent.

- Copy-on-write memory pages created.
- Duplicate parent’s page tables.
- Create unique task_struct for the child.
- Return value = 0 indicates child’s process.
  child PID in the parent’s process.
  -1 error, no child process created.

>0 indicates parent process

Threads

- Linux Process Vs Threads

```
int func1()
{
/* Thread function. */
…
}

int main()
{
…
   iret1 = pthread_create(
      &tid1, NULL, func1,
      (void *)message1);
…
   pthread_join( tid1, NULL);
…
   return 0;
}
```

pthread_create:

Creates a new thread with attributes in the call or NULL for default attributes.

- tid1 recieves the thread ID.
- NULL uses the default thread attributes.
- func1 is the pointer to the function to be threaded.
- message1 is the pointer to the argument.

pthread_join:

Waits for termination of another thread.

- tid1 is the thread the main process is waiting to terminate.

NULL specifies return value not needed.

Operating System Overview

- Operating System Overview Part 1
  - Computer System
  - Operating System
  - Processes
  - Threads
- Operating System Overview Part 2
  - CPU Scheduling
  - Process Synchronization
  - Deadlocks
  - Main Memory
  - Virtual Memory
  - File-System Interface
  - File-System Implementation
  - Mass-Storage Structure

Operating System Overview

done for May 19th

- Extra Slides

Operating System Overview

- Operating System Structure
- Operating System Provides Multi-Programming Environment
- Increases CPU Utilization (CPU always busy): Single program cannot keep CPU or I/O Devices busy
- System Resources (i.e. CPU, Memory, and Peripheral Devices) are Utilized Effectively
- Time-Sharing Systems
  - CPU switches between multiple users.
  - User interacts with Operating System with keyboard, mouse, monitor.
  - Operating System switches between users rapidly, giving impression of sysem dedicated to one user.
  - Scheduler decides which processor to run.
  - Scheduler Policy : Set of rules determing when and how a process is to run.
  - Scheduler Time Sharing principle based on timer interrupts.
  - Scheduler selects process based on priority.
    - Process priority based on how often the CPU is being used .
    - Interactive Process
    - Batch Process

Operating System Overview

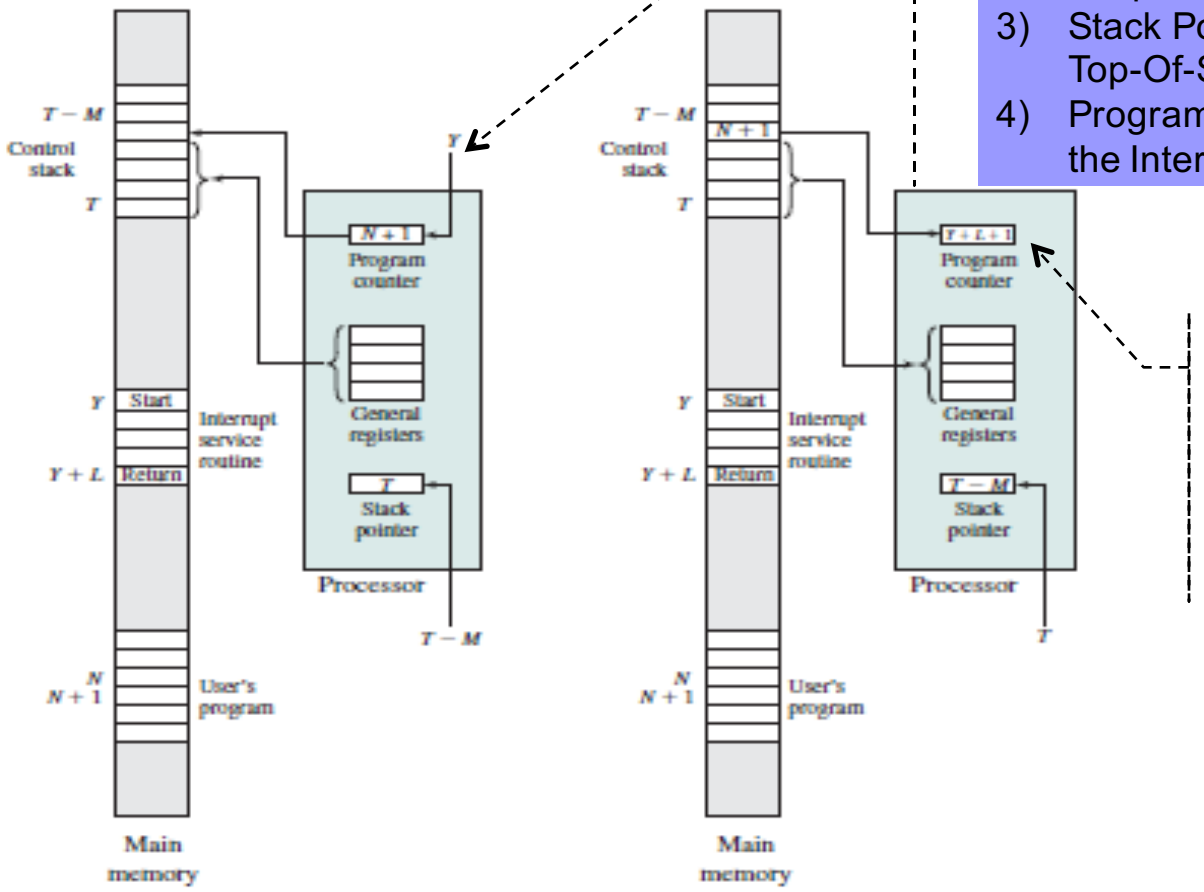
- Computer System Networks
- Operating System provides Users Access to Distributed Systems
  - Distributed Systems
    - Collection of physically separate Computer Systems connected by Computer Network.
    - Provides access to shared resources at different Computer Systems.
      - Increases computation speed, functionality, data availability, and reliability.
  - Abstracted through software (i.e. Cluster Layer supporting Distributed Database).
  - Access through Network Utilities (i.e. FTP, ssh).
- Category of Computer Networks:
  - LANs: Local Area Network connects hosts room, building, or campus (up to 1 kilometers).
  - MAN: Metropolitan Area Network connects hosts within city or between cities (1 to 20 kilometers).
  - WAN: Wide Area Network connects hosts within state or country (10+ kilometers to 1,000+ kilometers).
  - Internetworking: Network of networks through routers or gateways.

Operating System Overview

- Special-Purpose Computer Systems
- Computer Systems with limited computation domain: Not General-Purpose Computer Systems
- Real-Time Embedded Systems
  - Specialized embedded Computer Systems.
  - Monitoring (sensor data)and managing hardware devices.
  - Must process in fixed time constraints or the system will fail.
  - Automobile engine and controls (ABS Brakes, fuel injection), airplane controls, robotics, industrial control systems, home appliances, and military weapons.
- Multimedia Systems
  - Multimedia data systems (audio files, DVD movies, video, video conferencing, live webcast).
  - Requirement on delivery, streaming video at 30 frames per second.
- Handheld Computer Systems
  - Memory Management issues, low power or slow processors, small “consoles”.
  - Uses embedded Computer Systems where the Operating System and application have to work with slow processors.
  - I/O limitations due to small keyboards and small “console” window.

Operating System Overview

- Interrupt Handling



(a) Interrupt occurs after instruction at location N

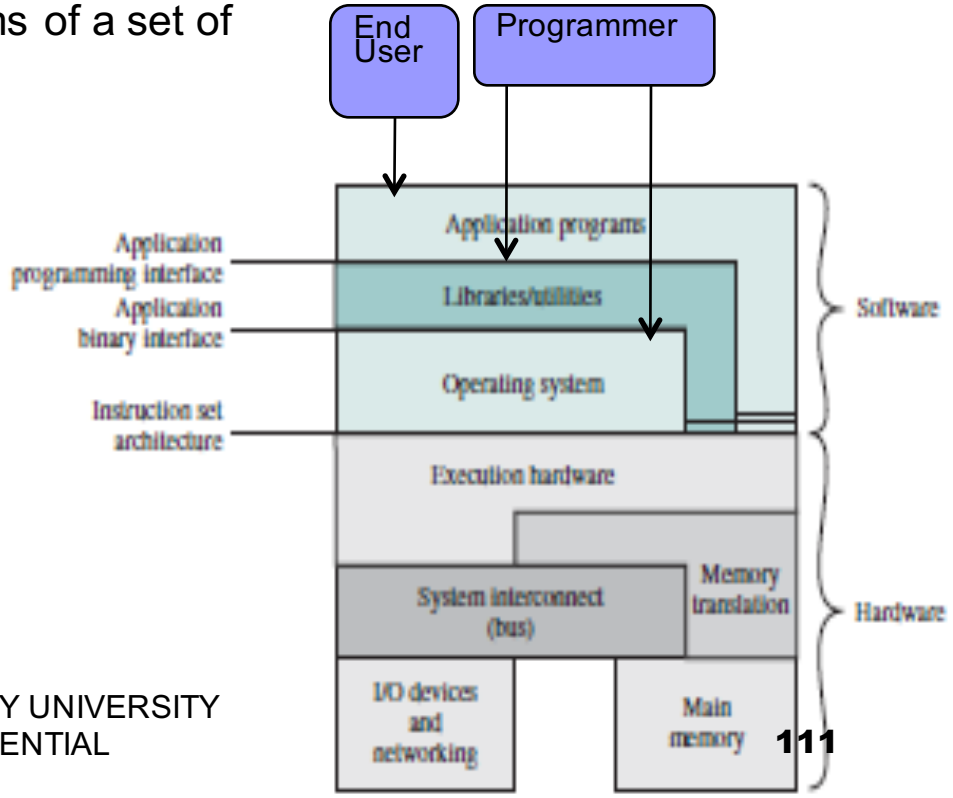
(b) Return from interrupt

1) User program interrupted after instruction at location N.
2) Contents of all registers plus address of next instruction (N+1), total M words are pushed onto Kernel Stack.
3) Stack Pointer updated to point to new Top-Of-Stack.
4) Program Counter updated to point to the Interrupt Service Routine.

1) Interrupt processing is complete, saved registers retrieved from Kernel Stack.
2) Program Counter restored from Kernel Stack.

Operating System Structures

- Operating System Objectives
  - Operating System is a program that controls the execution of application programs and acts as an interface between applications and computer hardware.
  - **Convenience**: Makes a computer more convenient to use.
  - **Efficiency**: Allows the Computer System resources to be used in an efficient manner.
  - **Ability to evolve**: Should be constructed to permit the effective development, testing and introduction of new system functions without interfering with service.
- Operating System as a User/Computer Interface
  - User views Computer System in terms of a set of applications.
  - Programmer make use of system programs as an interface to the Computer System.
  - The Operating System masks the details of the Computer System hardware from the programmer and provides the programmer with a convenient interface for using the Computer System.



Operating System Structures

- Development of Operating System
- Define Goals and Specifications - Type of Operating System:
  - Batch, Time-Sharing, Single User, Multi-User, Distributed, Real-Time, General Purpose.
  - Different requirements create large variety of solutions:
    - VxWorks: Real-Time Embedded System.
    - MVS: Multi-User for IBM Mainframes.
    - MS-DOS: Single User System.
    - UNIX / Linux: Time-Sharing
- Software Engineering Prinicples
  - Separation of Policy from Mechanism.
  - Mechanism determines how to do something.
    - Watch-Dog Time Out prevents system hanging.
    - Clock Interrupt occurs every system tick.
  - Policies determine what will be done.
    - How long the timer is set before Watch-Dog Time Out.
    - Clock Interrupt used to schedule processes based on priority policy.
- Implementation
  - Mostly written in C Programming Language.
    - Code written faster, easier to understand, easier to port to other hardware solution.

Operating System Structures

- Development of Operating System
- Evolution of Operating System Structure and Capabilities
- Design Elements creating change in nature of Operating System:
  - Hardware Arena: Multi-processor Systems, Increased Processor Speeds, High-Speed Networking, Increased size of Memory Storage Devices.
  - Application Arena: Multimedia Applications, Internet and Web Access, Client/Server Computing.
  - Security: Increased threats and attacks with viruses, worms, hacking
- New ways of Organizing the Operating System:
  - MicroKernel Architecture: Assigns few essential functions to Kernel (Address Spaces, IPC, Scheduling). Other OS Services are User mode processes, decoupled from Kernel mode.
  - Multithreading: Process becomes collection of threads, for concurrent operation.
  - Symmetric Multiprocessing: Operating System of an SMP schedules processes or threads across all of the processors.
  - Distributed Operating Systems: Single system for a cluster of separate Computer Systems (Illusion of unified memory and file system).
  - Object-Oriented Design: Modularization to the Kernel, allows customization without disrupting system integrity.