OS -Last min notes

Owner	N nithu
Tags	

Unit 1 - The basics

OS - intermediary pgm bw user and hw

Goals: Execute pgms, solving easier

Convenience of sys

Hw is used efficiently

Functions: Resource allocator

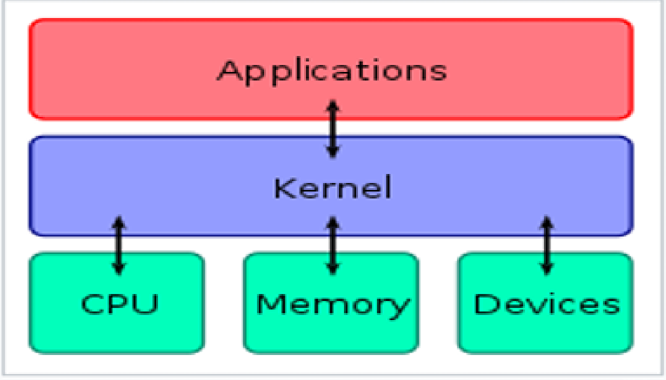
Control pgm

Kernel - always running on the sys , Everything else is either sys.pgm or app.pgm

Bootstrap - loaded at powerup or reboot(aka firmware)

Initializer of the system

Loads kernel and starts execution



A kernel connects the application software to the hardware of a computer.

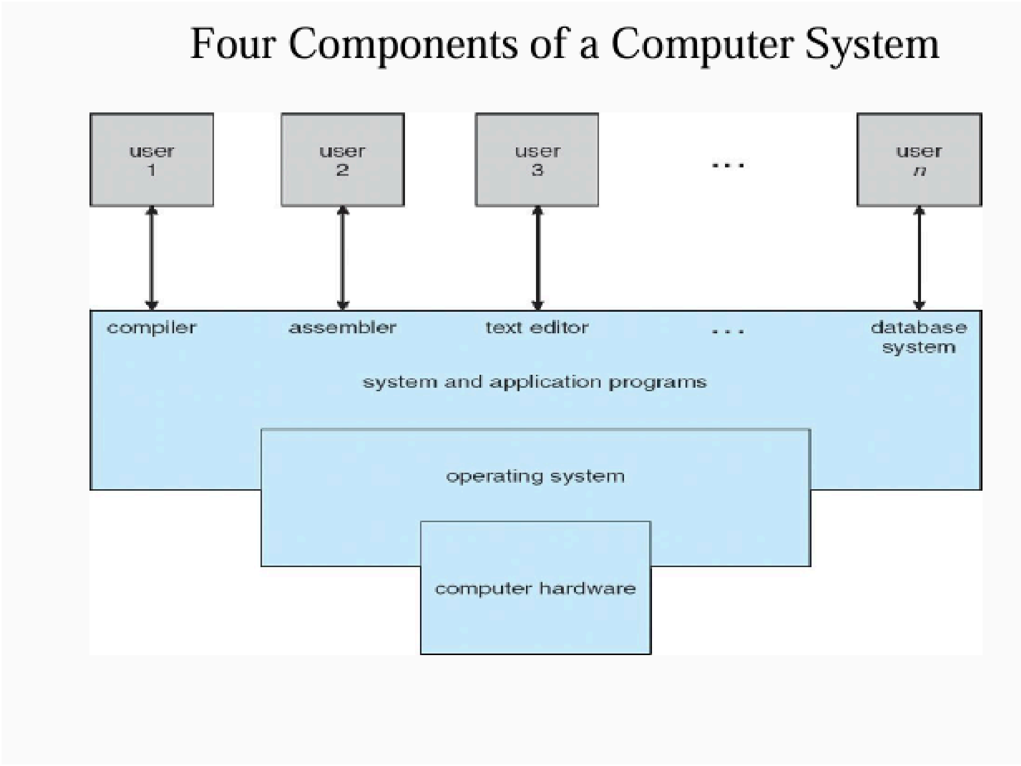
Components of a com.sys:

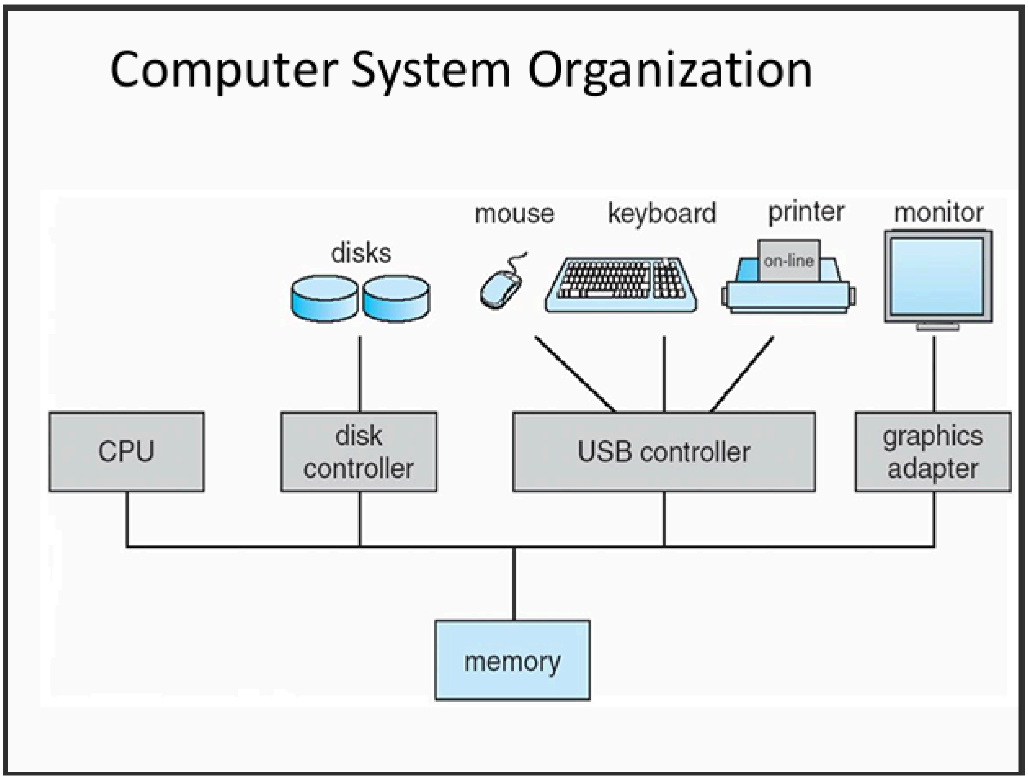
Hw- basic com.resources

OS - hw and the user

App.pgms - ways to solve problems with resources

Users





Concurrent execution of I/O and CPU

Each device controller - particular device

Local buffer

CPU ⟶ main mem. to local buffers

I/O ⟶ local buffers to controller

Task done⟶ interrupt

Interrupt ⟶ ISR thro interrupt vector

Exception ⟶ software-generated interrupt

OS⟶ interrupt-driven

OS⟶ saves the state

Types of OS:

Batch ⟶ users do not interact

punch cards ⟶ computer operator

jobs are batched

operator ⟶ sorts and feeds as batches

Eg: Payroll,Banking

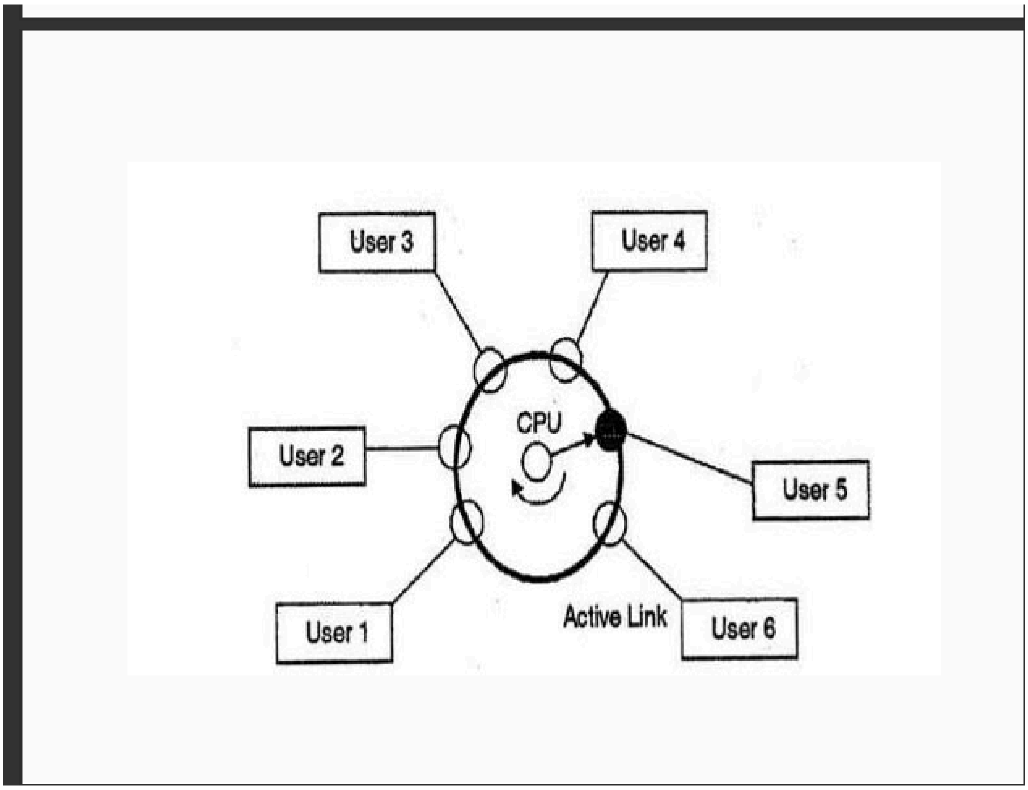
Time sharing

Many ppl, comp.sys ⟶ same time

Time shared simultaneously with multiple users ⟶ time-sharing

Eg.Multics , Unix

Runs on a schedule(divided among users, full access during that time, time slice)



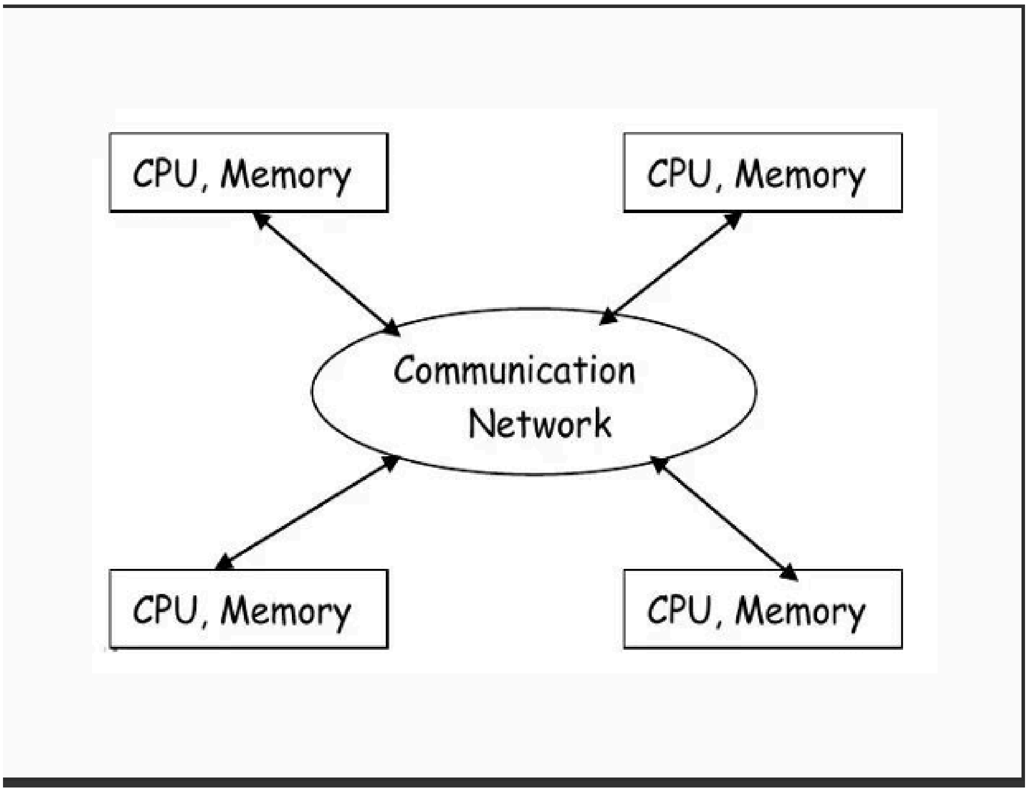
Distributed

Multiple CPUs and multiple real-time apps

communication lines

Loosely coupled system ⟶each processor with its local memory

Eg.Locus

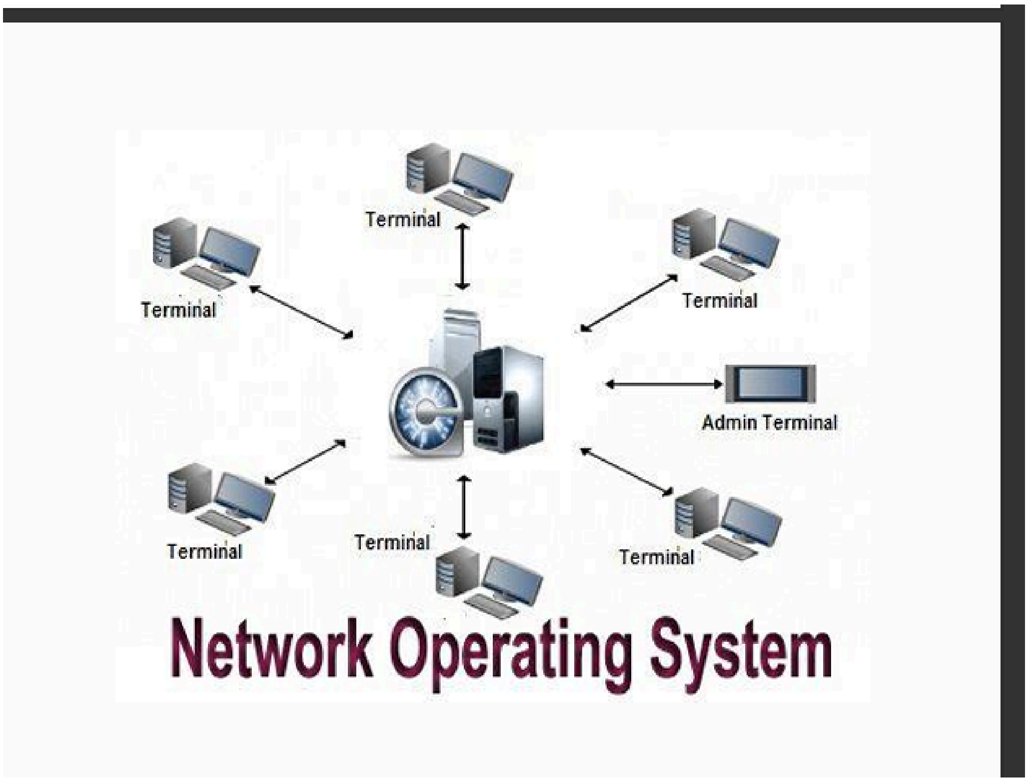


Network

Connection through LANs or Inter-network

Runs on a server

Shared file and printer access over multiple computers



Storage Structure:

Main mem → RAM(Random access and volatile)

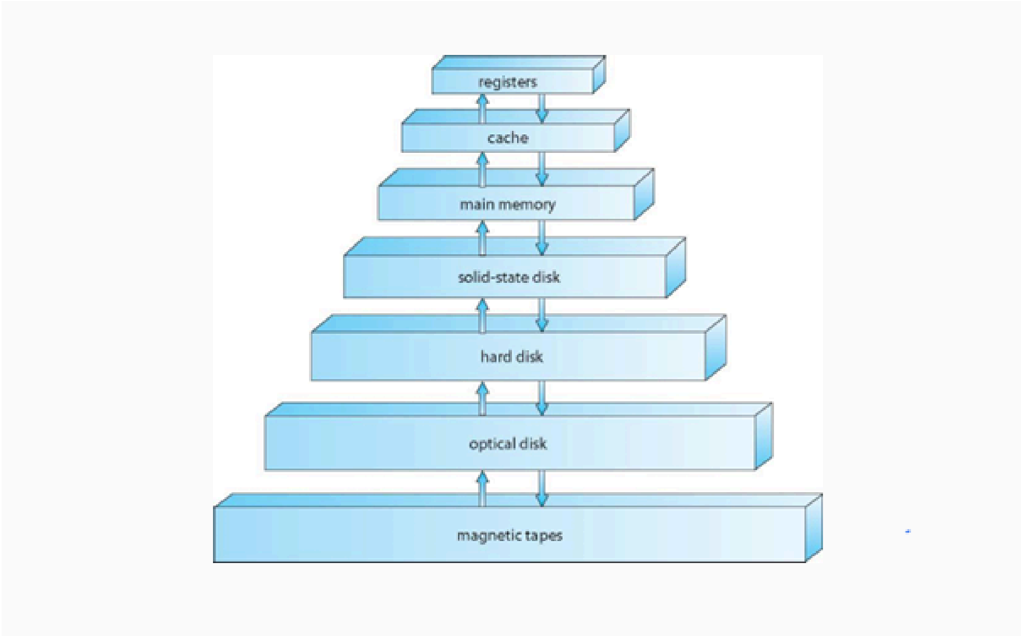
Sec . Storage → Non-volatile

Hard disks → tracks and sectors. Disk controller determines logical interaction.

Solid-state disks → non-volatile

Caching → copying info. to a faster storage →main mem is a cache for sec. storage

Device driver → manages I/O



Slower storage → Faster storage

Faster storage(Cache) is checked for info. if present, it is used from there

If not, copied to cache and used there

Device controller → data is sent directly to main mem. (no CPU involved)

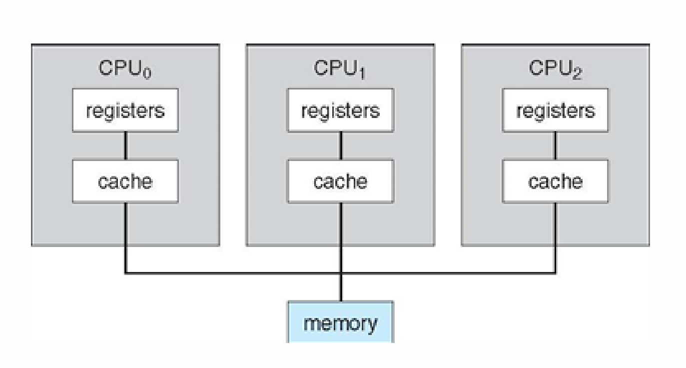
Computer - Sys Arch

Single processor

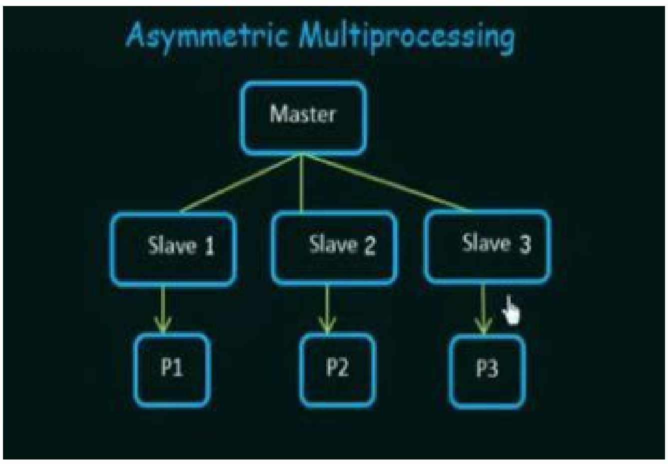
Multi-processor(Parallel systems, tight-coupled systems)

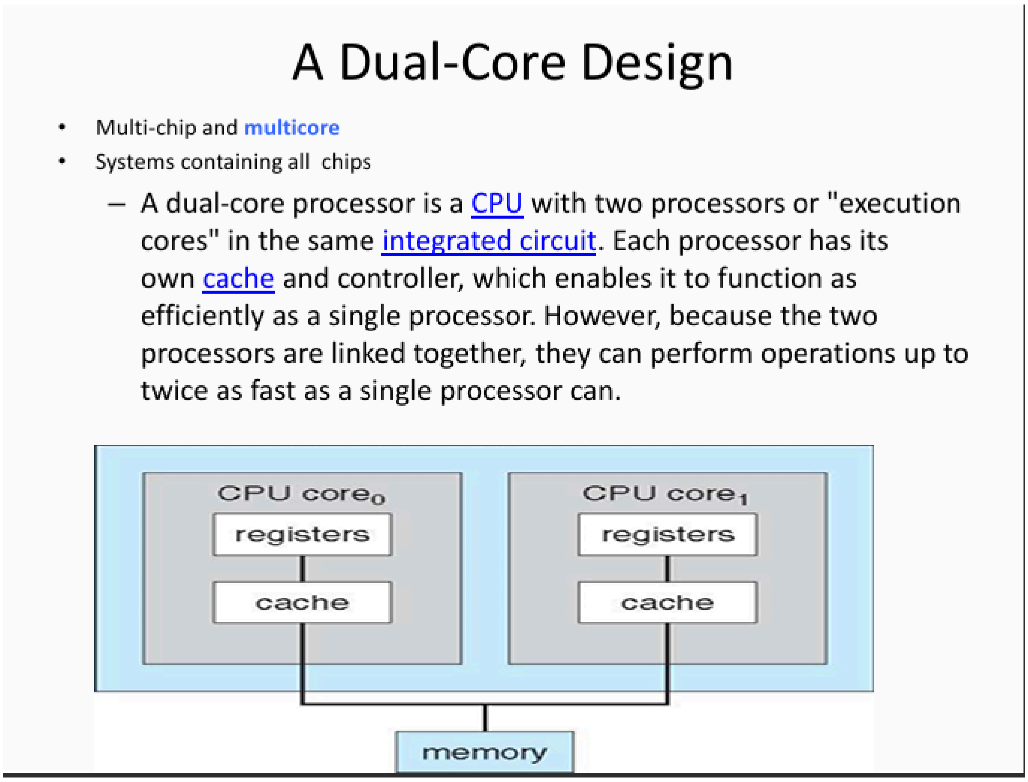
BASIS FOR COMPARISON	LOOSELY COUPLED MULTIPROCESSOR SYSTEM	TIGHTLY COUPLED MULTIPROCESSOR SYSTEM
Basic	Each processor has its own memory module.	Processors have shared memory modules.
Efficient	Efficient when tasks running on different processors, has minimal interaction.	Efficient for high-speed or real-time processing.
Memory conflict	It generally, do not encounter memory conflict.	It experiences more memory conflicts.
Interconnections	Message transfer system (MTS).	Interconnection networks PMIN, IOPIN, ISIN.
Data rate	Low.	High.
Expensive	Less expensive.	More expensive.

Symmetric → each processor performs all



Asymmetric → each processor to a specific task





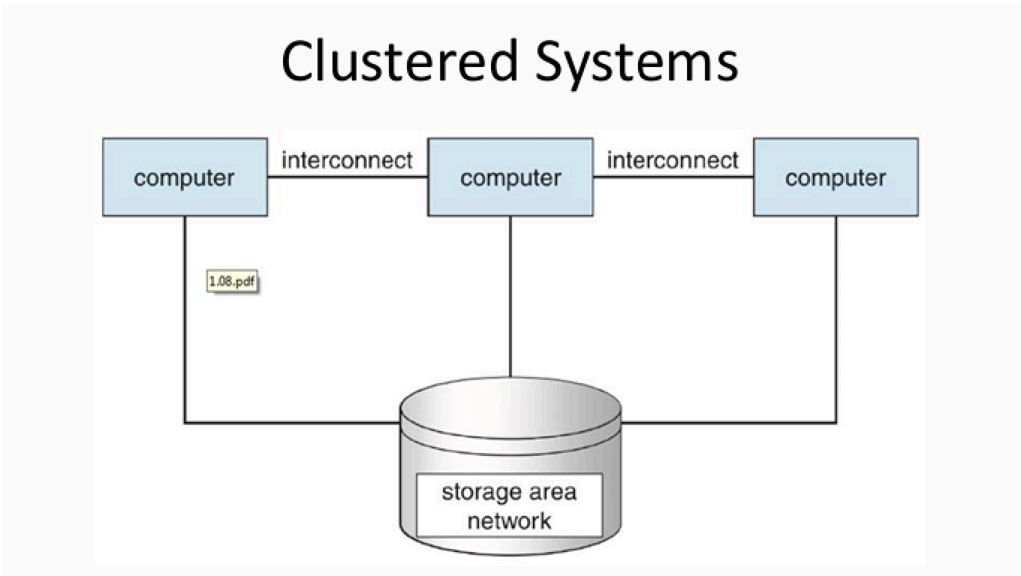
Clustered

→ Storage area network

Symmetric clustering

Asymmetric clustering

HPC



Services:

UI - CLI and GUI

CLI - direct cmd entry

kernel, system programs

shells

fetches cmd and executes

GUI - monitor,mouse,keyboard

icon

folder

Touch screen interface

Pgm execution -to load, run pgm successfully, and end

I/O Operations - I/O involving file or I/O device

File Sys.Manipulation — create directories, folders,files,permissions

Communications - interchange info

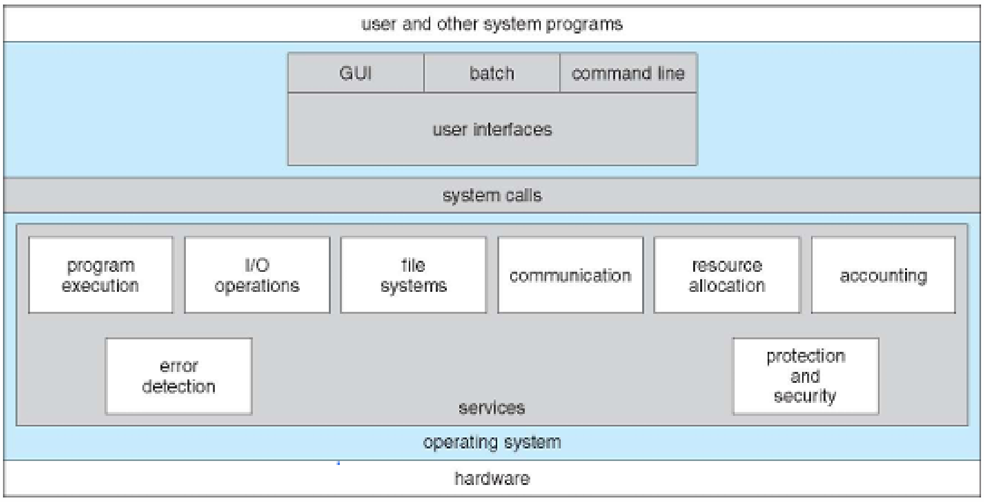
Error Detection - CPU, hw,I/O

Debugging

Resource allocation

Accounting - how much and what kinds of com.resources

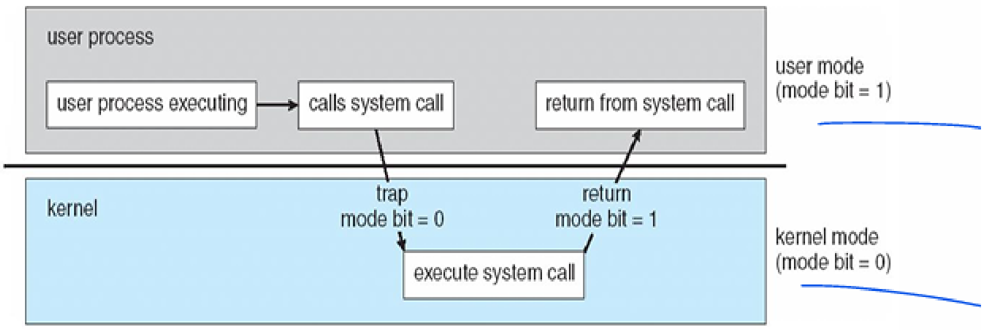
Protection and security



Operations: Dual Mode(User and Kernel)

Mode bit - 0 for kernel and 1 for user) to distinguish which mode code is executed)

privileged → kernel code



Transition process:

Set timer

Counter is decremented

Set the counter

When counter is 0, give interrupt

Set up before to regain control or terminate pgm.

User mode → OS runs a user app. like a text editor

(transition requests the help of OS or an interrupt)

→ 1 to 0

Kernel Mode → Boots after OS is loaded.

→ 0 to 1

Process mgmt.

Process → active entity

Program → passive entity

Resources required: CPU, mem, I/O, files

Single-threaded → one PC

Multi-threaded → one PC/thread

Process Management Activities:

Create/Delete

Suspend/Resume

Process sync.

Process.com

Deadlock handling

Memory mgmt. → CPU utilization , computer response

→ Track of parts of memory being used.

→ Decide which processes to move into and out

→ Allocation and deallocation of mem.space

Storage mgmt. → storage units (file) associated with a device(tape drive, disks)

File Sys.mgmt

System calls —> between processes and OS

User mode needs a resource → system call → kernel mode provides the resource

API(Application Program Interface) → service provider of the OS to user pgms.

Eg: POSIX API,Win32 API,Java API

```
#include <unistd.h>

ssize_t     read(int fd, void *buf, size_t count)
```

return value — function name — parameters

A program that uses the read() function must include the unistd.h header file, as this file defines the ssize_t and size_t data types (among other things). The parameters passed to read() are as follows:

- int fd—the file descriptor to be read
- void *buf—a buffer where the data will be read into
- size_t count—the maximum number of bytes to be read into the buffer

Table of system-call interface.

Types of Sys.Calls:

Process Control - fork(), exec(), exit(), wait()

File mgmt. - read(), write(), open()

Device mgmt.- read(), write(), ioctl()

Info.mgmt. - alarm(), sleep(), getpid()

Protection - umask(), chown()

Communications -pipe(), shmget(), mmap()

System programs → File mgmt, Status info, File modification , Programming lang.support, Program loading, execution , Communications

Eg: windows 10, Mac, Linux,Unix

OS Structure

→ Simple

▼ More complex

→ Layered

→ Microkernel -Mach

Need to look at Virtualization and Structure in depth!

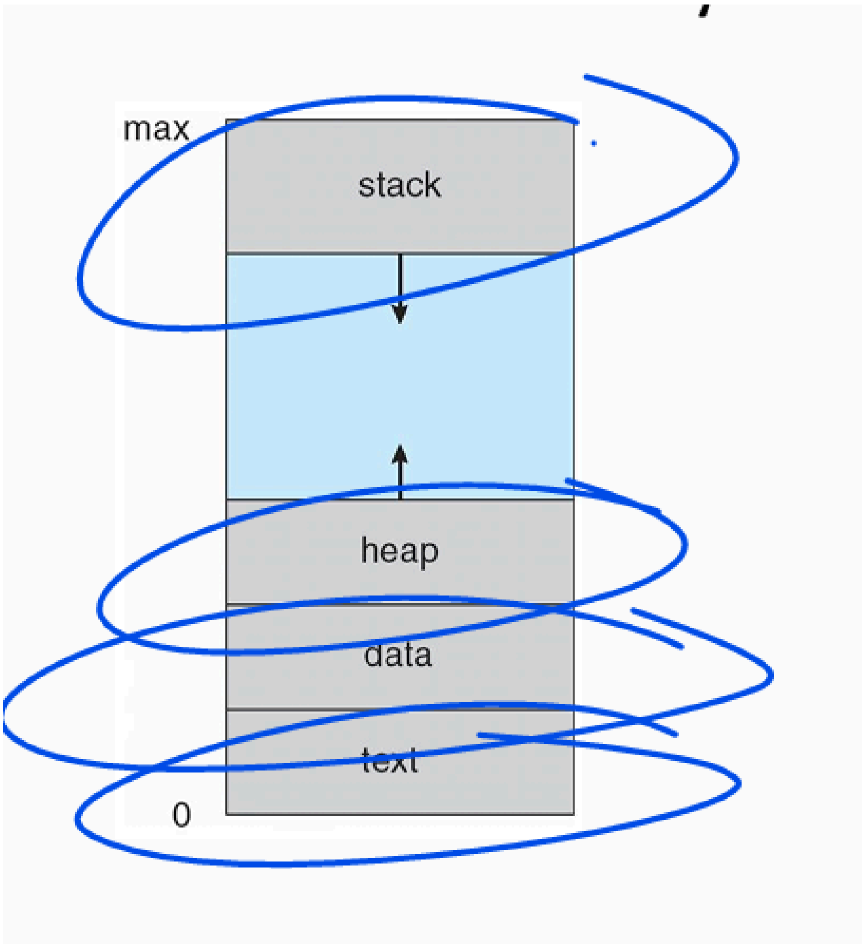
Unit 2 - Processes

Program → passive,on-disk

Process → active,on mem

A program which is on execution.

Contents of a process: Text section, PC, stack,heap,data,processor registers



States of process execution:

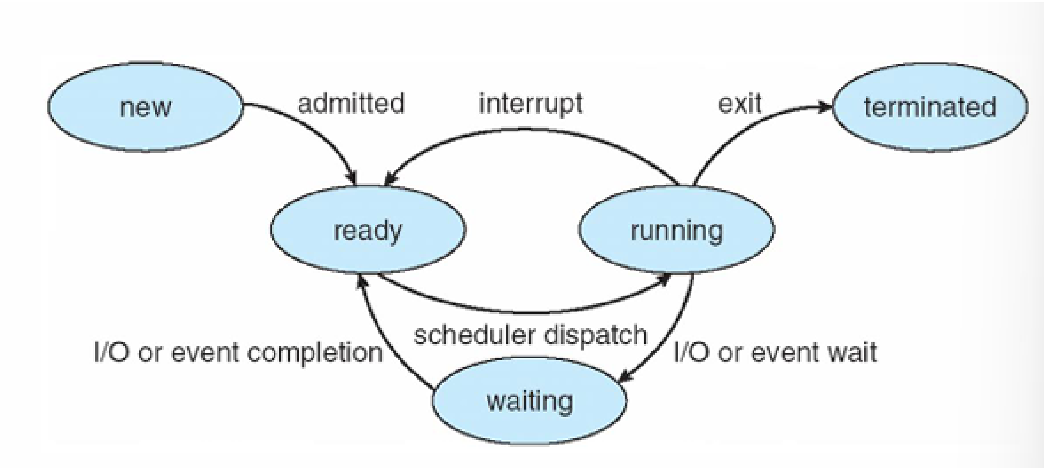
new() - the process is created

running() - execution is proceeding

waiting() - waiting for some event to occur

terminated() - finishing

ready() - waiting to be assigned to a pro.



Process Control Block(PCB)

State

Counter

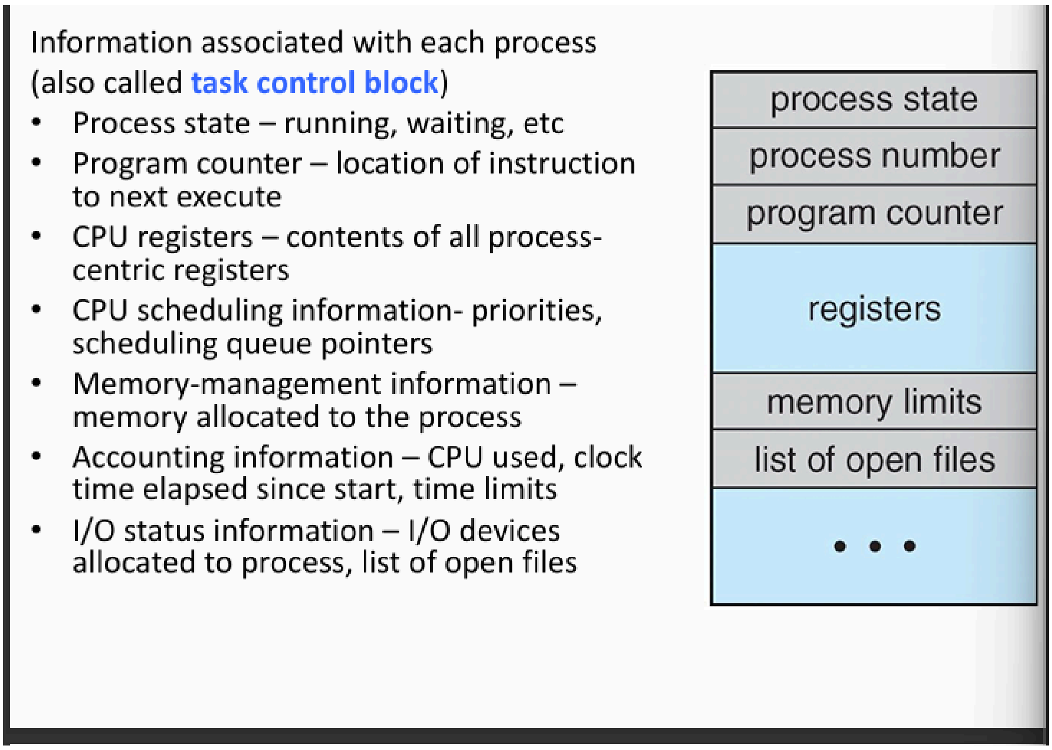
CPU registers

Scheduling info.

Mem.mgmt info

Acct.info

IO status info.



Threads: Light weight process → perform only one task at a time

Process scheduler → some process should run at all times

max.CPU utilization

selects among whatever is available for execution

- Job queue – set of all processes in the system
- Ready queue – set of all processes residing in main memory, ready and waiting to execute
- Device queues – set of processes waiting for an I/O device

Process Scheduling Queue

Ready queue————————CPU

I/O Device←———–-I/O request———I/O queue

Time slice expired

Child executes——Fork a child

Interrupt occurs←————Interrupt is requested

Schedulers→ Short-term(CPU scheduler) → invoked freq. , process to be executed next and CPU

Long-term scheduler(Job scheduler) → infrequently, select processes for ready

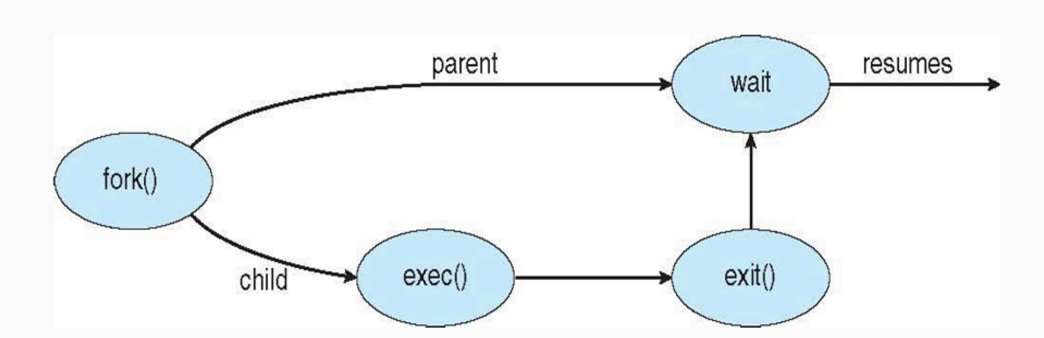
Processes: I/O bound process(more of I/O based)and CPU-bound process(computations-based)

Context switching → save the state and load the saved state using context switch(overhead in PCB)

Process creation → parent→ child → tree(Differ by pid)

Fork() — creating

exec() — to replace the process memory with a new pgm.



Process Termination:

exit() ⟶ returns status data from child to parent and deallocation

abort()

Zombie process ⟶ not parent wait and did not invoke wait()
Orphan ⟶ parent terminated without invoking wait.

IPC - Inter-process communication

Independent or Cooperating

Why cooperating? Either for info.sharing, computation speedup, modularity,convenience

Two models of IPC:

Shared memory

Message passing: no sharing of same address space.

Send:

Receive:

Message can be fixed or variable

Communication link is a must.

⟶ Direct or indirect(naming)

Direct - explicit naming

one link only

between two processes

Indirect ⟶ uses a mailbox-kind of system(shared)

More than two are allowed

shared mailbox is the key

→ Asynchronous or synchronous(synchronization)

Synchronous/blocking

Blocking send→ process is blocked until msg is received by the receiving process

Blocking receive → the receiver blocks until a msg is available

Asynchronous/nonblocking

Nonblocking send → sends the msg and resumes

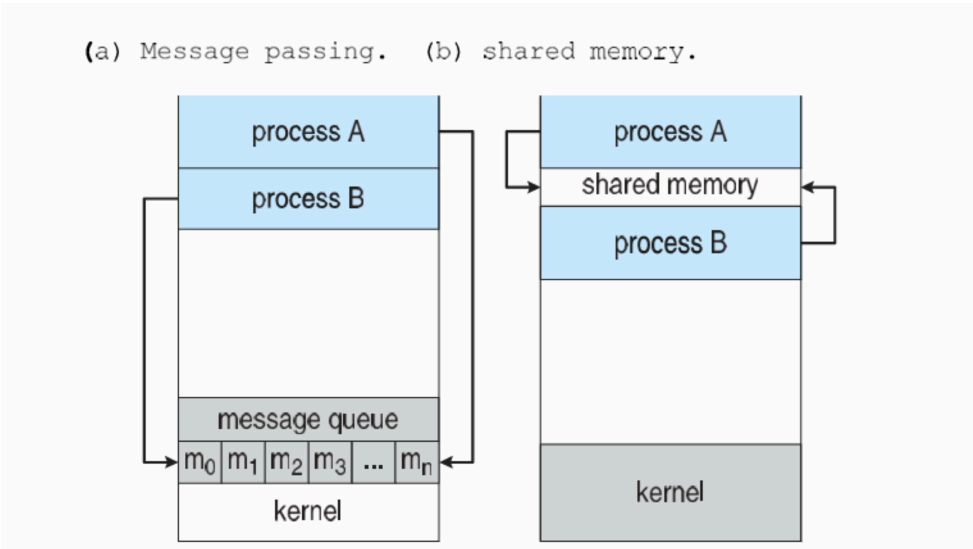
Nonblocking receive → retrieves a valid msg or null.

→ Automatic or explicit(buffering)

Zero cap

Bounded cap

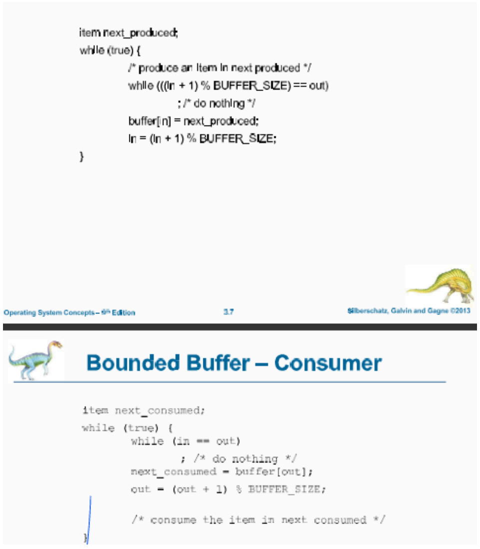
Unbounded cap



Producer-Consumer Prob:

Unbounded buffer : size has no limit

Bounded buffer : size has a limit



CPU Scheduling

Scheduling queue:

Running to waiting and terminating → non-preemptive

Running to ready and Waiting to ready → pre-emptive

Criteria:

Keep CPU busy

Throughput

TAT
WT
RT

FCFS → Convoy Effect: Processes who need to use a resource for a short time is blocked by bigger processes.

> 💼 Look at threads!

Unit-3 Process Synchronization - Deadlocks

Process Synchronization — coordinating processes such that no 2 process, same data and resources at same time.

Leads to inconsistency

Race Condition: more than two cooperating processes, accessing same resources and data concurrently, access.

CRITICAL SECTION : code segment for shared variables.

Race condition inside critical section.

Entry section - permissions

Critical section - sharing

Exit section - critical section completes

Remainder section - rem.part

Three conditions:
Mut.exclusion - one process happening, no other process can interfere.

Progress: no progress in execution → waiting u have some→ one of these

Bounded wait: Process requests, certain set criteria or bound, before request is granted

Semaphore → wait() and signal()