Sir Ali Hassan
Sial
SF-227

26-10-23

Operating Systems

Books:

→ Operating System Concepts, 8th & 9th editions
Silberschatz et al. Wiley

→ * Operating Systems: Internal & Design Principles (8th Ed.).
William Stallings

Diagrams

02-11-23

Introduction To Operating Systems:

Computer System: An integrated form of different components working together, each serving its own purpose.

Computer System Structure / Architecture

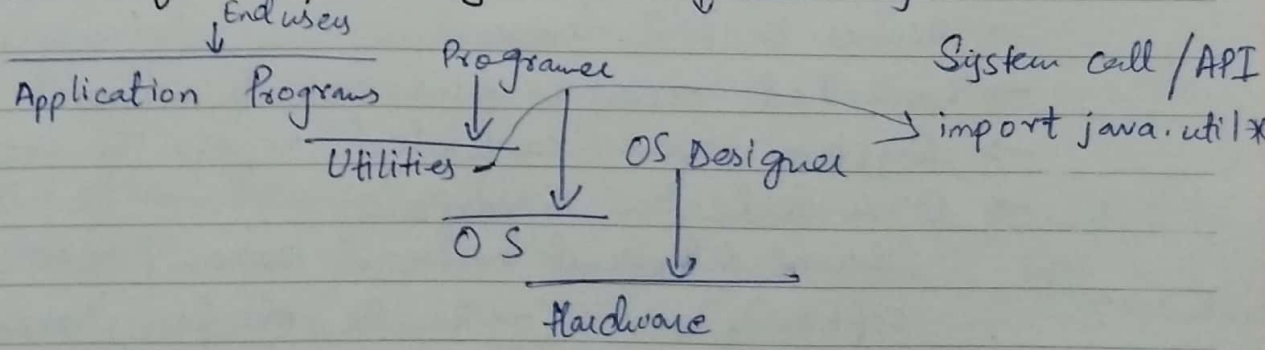
① Hardware: provide basic computing resources

② Operating System: a system software, bridge between hardware & application programs, an interface or a middleware

③ Application programs: define the ways in which system resources are used to solve computing problems

④ Users: end-users, people, machines.

Layers of Computer System ↓ → used by.

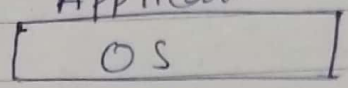


Operating System (OS): OS is a system software

An intermediary platform b/w user & computer hardware.

Goals:

1. Exploits hardware resources.
2. Provides a set of services
3. manages secondary memory & I/O devices



. Kernel is a nucleus or brain of an OS

OS as a resource manager:

- Allocating resources across space & time.
  - time sharing a resource (scheduling) time-space tradeoff
  - space sharing a resource (allocation)
  - CPU, memory, disks, network, ...
- Efficient use of limited resources.
  - improving utilization, minimizing overheads.
  - improving throughput (
- OS is a control program.
  controls execution of programs, improving errors).

Services provided by the OS:

→ Abstract Machine :- hides complex details
→ Resource Manager:- controls shared resources.
→ Program development:
→ Program execution :- a process
→ Access to I/O devices.
→ Controlled access to files: restrict access
→ System access: restricted access to users.
→ Error detection & response:
  internal & external hardware errors. (memory, device failures)
  software errors (arithmetic overflow, access forbidden to memory location).
→ Accounting.

inter-process communication

Basic Elements concerned with OS

volatile memory

① Processor ② Main Memory (referred to primary mem (RAM))

③ Kernel: ③ I/O Modules => secondary memory devices.

micro monolithic ④ System bus => (communication among processors, memory & I/O modules)

AX — AL (lower bits) } divided into segments
AX — AH (higher bits) }

User-visible register:

MOV AX, 01H
↓ opcode | operands
General-purpose register.

programmer can minimize main-mem references by optimizing register use.

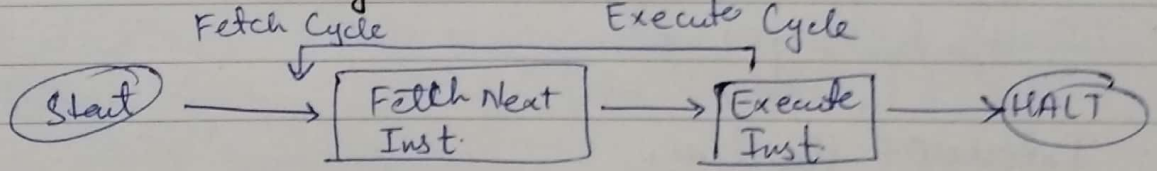
Type ① Data Register
② Addres Registers.
↳ Index | ↳ Segment Pointer
↳ Stack Pointer.

Control & Status Register:

They're used by processor to control operating & executing of the processor.

① Program Counter (PC): address of next instruction.
② Instruction Register (IR):
③ Program Status Word (PSW): determines piviledged instructions status changes

Instruction Cycle:



Primitive: program: continuous execution
Non-Primitive program: stops.

Instruction Register. Fetched instruction is placed into IR where it is decoded to find instructions type & operands. Then decoded instruction is executed.

16-NOV-23

INTERRUPTS IN OS

An activity or process that stops the continuous progress of a program.

INT 21H. → to take input & generate output.

Types of Interrupts.

① Hardware Interrupts: a signal to CPU through an external device or hardware.

② Software Interrupts: Exceptions.

Types of Hardware Interrupts.

① Non-Maskable Interrupts (Nested): handles priority-wise

② ~~Non~~-maskable Interrupts: first come first served. (sequential).

Classes of Interrupts.

- Program.
  - → arithmetic overflow. → execute illegal instruction } software interrupts
  - → division by 0 → reference outside memory }
- Timer. — Primitive (specified time slice).
  — non-primitive (without any delay).
- I/O ⇒ input/output modules interrupts.
- Hardware failure

Interrupt Handler:

↳ determines the nature of interrupt.
↳ how to handle it in terms of scheduling it.

Direct Memory Access (DMA)

Allowing devices to write directly (via bus) into main memory.

Multiprogramming:

One processor can do multiple tasks (program) at a time is multiprog-ramming

Disk Cache:

Three types of cache

- L_1 → within CPU.
- L_2 → beneath CPU.
- L_3 → outside CPU.

cache is a temporary memory

Virtual memory is a temporary memory

Disk cache is a portion of main memory used as a buffer to temporarily to hold data for the disk.
Data is divided into clusters (segments) to get stored in disk cache
Data retrieval is slow in in disk cache as compared to other caches.

mapping functions.

Cache / Main-memory structure.

Cache Design:

Cache size: small size cache, good peeformance.
Block size: unit of data exchanged b/w cache & main-memory

Mapping function: maps cache location with blocks.
Replacement algorithms: which page/block to replace.
LRU ⇒ Least Recently Used.
Write policy :- memory write operation.

Programmed I/O

Interrupt-driven I/O.
processor is interrupted when I/O module is ready to exchange data.

UNIX OVERVIEW
Linux's predecessor.
Unix uses single processor.

Important Topics of the lecture.
30 NOV / 7 dec
Quiz.

① Interrupts
↳ Definition
↳ Types (SW, HW).
↳ Classes (SW, HW).
Maskable, Non-Maskable
Interrupt Handler, Interrupt Cycle.
Program Flow (Nested, Sequential)
DMA → Cache Memory (Cache Design).
M/P, Program I/O, Interrupt-driven I/O.
First lecture.
Computer System, CS structure, OS Def., services.
Elements of OS.

Lecture # 03. 23-11-23

EVOLUTION OF OPERATING SYSTEMS

Types of OS

1. Early Systems (1950) => ENIAC (Mechanical Computer).
2. Simple Batch Systems (1960) => used to hold multiple batch jobs.
   - One by one job done.
   - operator driven shops. resident monitor.
3. Multiprogrammed Batch Systems (1970) => queues based. job done in batches.
4. Time-sharing & Real time Systems (1970): RTOS

RTOS
- Embedded Computer
  - ARM
  - Android OS
  - MOS RTOS
  - Mobile OS
- Arduino Mega
  - ↓ Java / C++
  - ↓ Linux / OS ↗ RTOS/MOS (Mobile-based OS)

Arduino Uno, Mega, Nano

5. Personal / Desktop Computers (1980) => DOS Disk Operating System.

6. Multiprocessor Systems (1980) => Requires General purpose OS.
   - → 4 Bit → Intel 4004 (DOS).
   - → 8 Bit — Intel 8085
   - → 16 Bit — 8086.
   - → 32 Bit — Pentium II
   - → 64 Bit — P4, core 2.

7. Networked / Distributed Systems: (1980).

Cloud Computing
- SaaS ↓ GitHub
- PaaS ↓ LMS / Onedrive
- IaaS ↓

④ Web-based Systems (1990).

① Early Systems:
Single user system.. ENIAC
User is operator. (Open shop).

② Simple Batch Systems
use of high-level-languages used to execute tasks.
Serially jobs done in batches.
An operator was hired to perform tasks.
Resident monitor is in main memory & available for execution (jobs sequencer/scheduler).

Operator driven Shop
Operator is OS and will do jobs.

Operation of SBS
user submits a job written on cards / tape.
Then computer operator will place a batch of jobs.
Resident monitor (a kernel).

Idea of OS
Reduce setup time by batching similar jobs.
Alternate execution b/w user & monitor program.
Use Automatic Job Sequencing.

Control Cards (1)
Probans: How to identify job's nature.?
How to distinguish b/w a job from job.
& data from program.
Solution: JCL & control cards.
Control cards used to instruct resident monitor which job to execute/run.

Job Control Language (JCL).

instructs resident monitor what compiler to use & what data to use.

Effects of JCL

JCL involves kernel routines, incorporate any calls of.

JCL works as interprete.

Resident Monitor.

Desired Hardware Features.

Memory protection: memory can't be altered by a user program.

Privileged Instructions: for dedicated users.

↳ can be executed only by resident monitors.

Interrupts:

Timer Interrupts.

Offline Operations.

Computation speed can be increased.

Spooling

Simultaneous peripheral operations On Line.

Job pool, a data structure.

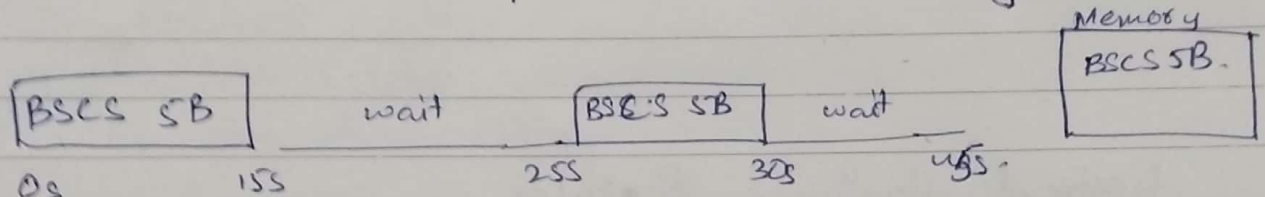
↳ instructs OS kernel which job to execute in order to increase CPU utilization.

Uniprogramming:

more latency rate.

Time consumption is complex.

Execution of a specific program using single processor



BSCS SB => 20s → Execution.

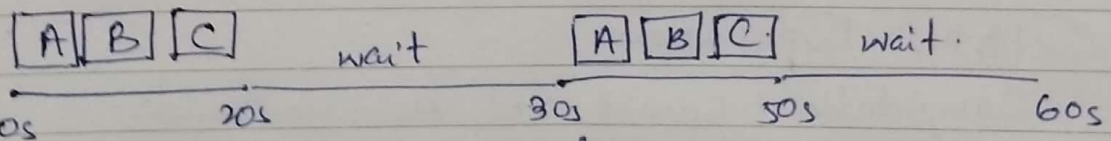
Wait => 25s → Wait.

Wait { Interrupt State. / Suspended State

(3) Batch Multiprograming.

Memory Layout for Batch Multiprogramming.

Several jobs are kept in main-memory at one multiplexed Parallel execution



A, B, C = 40s → Execution.

wait(A, B, C) = 20s → Wait.

Why Multiprogramming:.

- To minimize waiting time. , latency decreased.
- Multiple jobs are being organized together or being executed parallely.
- Job Sequencer / Job Scheduler / Resident Monitor.

Requirements for Multiprogramming:

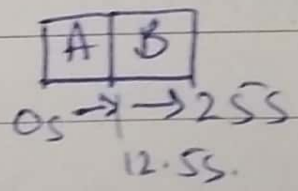
- Hardware support: I/O interrupts & DMA controllers.
  Timer Interrupts.
  Memory management. ready-to-run job in memory.
  Memory protection..
  Software support from the OS.
  for scheduling and resources contemption (overcoming from over lapping)

More usage of processor.
Throughput rate is increased. (jobs executed per unit time).

30-11-23

Characteristics of Time Sharing :

Execution time is distributed among multiple processors.



simultaneous execution

Batch Multiprogramming v/s Time Sharing.

	Batch MP	Time Sharing.
Principal Objective	Max. Processor use	Max. responsive time
Source of directive to OS	JCL commands Provided with job	Commands entered at terminal.

Characteristics of Modern OS.

Task

Monolithic kernel.

→ Microkernel architecture.

- interprocess communication (IPC) → how to communicate b/w different processes.
- Schedule → Dispatcher (a program for scheduling)
- Multithreading: dispatchable.
  A thread is an ~~executable~~ unit of work.
  executes sequentially & is interruptable.
  Process is a collection of one or more threads.
  & they execute sequentially.
  Single-threaded approach.
- Symmetric multiprocessing: multiple processors.
  every processor share same main memory & I/O fac-
- Distributed operating systems.
  everything is distributed among different spaces.
  different segments.
- Object-oriented design.
  added modularity.

LECTURE #4.

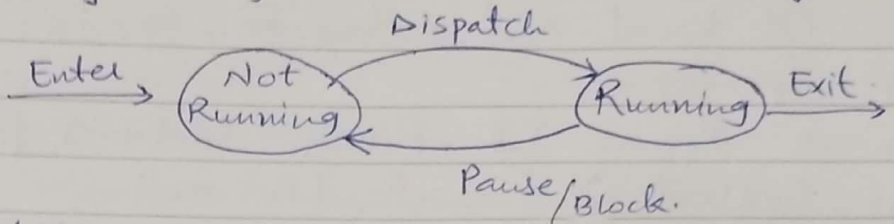
OPERATING SYSTEM PROCESSES / PROCESS MANAGEMENT

Process: A program in execution.
Multiple threads make a process.
An instance of a ~~programming~~ running on a computer.

Traces of Process. List of the process.
Dispatcher is a small program to switch or schedule process
OS governs which instructions are in processor.
↳ Nature of a process.

Two - State Process Model.

Process may be in one of two states.
- Running (waiting for I/O) - Not Running. (Ready to execute).



Dispatcher can't schedule a process that is in ready-queue for long, it will mark it block.

Process Creation.

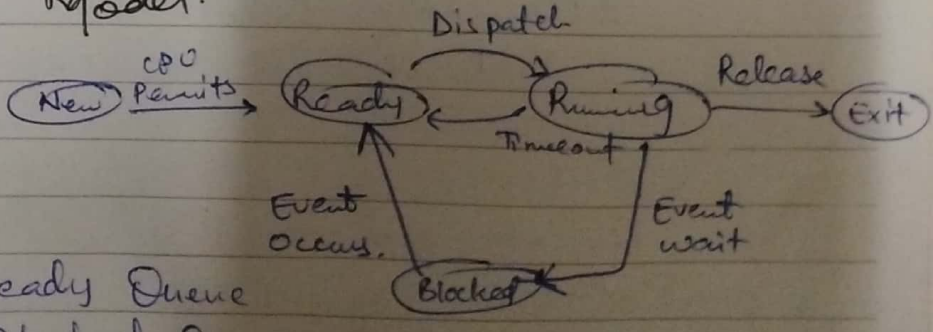
- Submission of a batch job.
- user logs on.

Process Termination.

- Normally completed.
- Time limit exceed.
- Memory not available for a program
- Bounds violation
- I/O failure.
- Invalid Instruction
- Data misuse.
- Parent termination, child.
- Protection error. write to read only file.
- Arithmetic error.
- Time overrun: process waited for long time.
- Privileged Instruction e.g. root user
- OS intervention => deadlock.
- Parent request.

A five - state Model.

- Running
- Ready
- Blocked
- New
- Exit



Ready Queue
Blocked Queue.
~~Event Queues~~

Process Control Block. PCB

A data structure.

Each & every process is represented by PCB.

It is in OS kernel and information needed to manage processes.

kernel level — Determines Process Id, Process State Information

user level — Process Control Information (Nature).

Contains a user stack, private user space, shared address space.

Pointers: contains address of another process (instruction Pointer.) present in ready queue.

Process State: Information regarding state of process.
① New ② Ready ③ Running ④ Wait ⑤ Halted

Program Counter: address of next instruction.

CPU registers: variety of registers.

CPU scheduling Information:
Process priority, Pointer to scheduling queues, Events.

Memory management information:
- value of base & limit registers.
- page tables.

Accounting Information: Statistical measures of memory & time. & processor used amount.

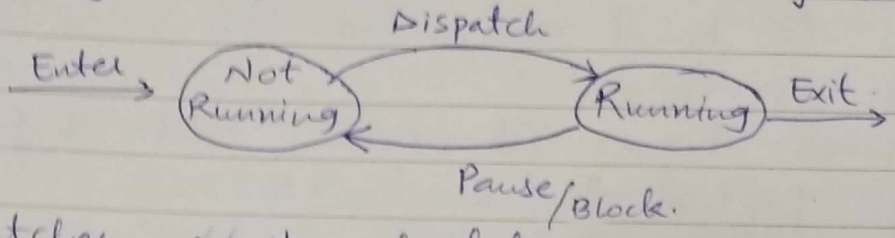
I/O status Information.
List of I/O devices used.

It has critical & confidential information. It must be at protected place.

placed at the top of the corner.

Two - State Process Model.

Process may be in one of two states.
- Running (waiting for I/O) — Not Running. (Ready to execute).



Dispatcher can't schedule a process that is in ready-queue for long, it will mark it block.

Process Creation.

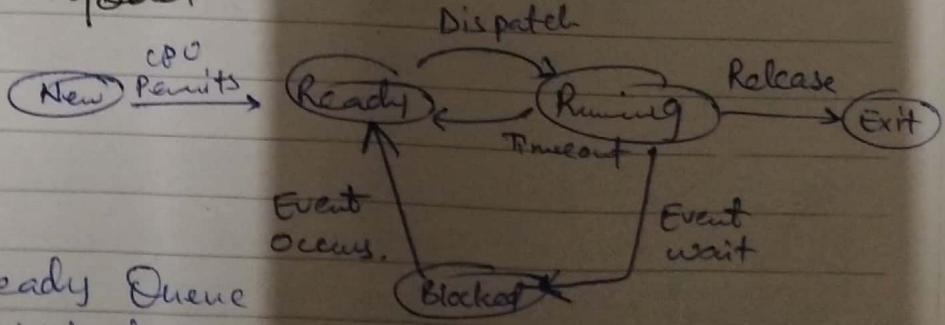
- Submission of a batch job.
- user logs on.

Process Termination.

- Normally completed.
- Time limit exceed.
- Memory not available for a program.
- Bounds violation
- I/O failure.
- Invalid Instruction
- Data misuse.
- Parent termination, child.
- Protection error. write to read only file.
- Arithmetic error.
- Time overrun: process waited for long time.
- Privileged Instruction e.g. root user.
- OS intervention => deadlock.
- Parent request.

A five - state Model.

- Running
- Ready
- Blocked
- New
- Exit



Ready Queue
n Blocked Queue.
~~Event Queues~~

Change of Process State.

- Save context of current state of process.
- update the process state. currently running.
- Move process control block to appropriate queue.
- select another process.
- Update process control block.
- Update memory management DS.
- Restore context..

Kernel.

Nucleus of an OS. central part of an OS. manages the task of system software.

Preemptive Scheduling:.

Eg. Round Robin Algorithm.

Time slice is given & priority-wise execution. switching b/w processes.

Non-preemptive Scheduling:

A running task will be executed & can't be interrupted until its normal completion.

Eg. First in First out.

3rd Chapter Important

Types of OS.

Simple batch system

- JCL
- RM.

Multiprogramming

Uniprogramming.

Characteristics of OS

Spooling.

4th Chapter Important.

Process.

Five-State Process Model

Two-State Process Model

CPU-scheduling.

PCB

Is organized structure

Lecture # 5.

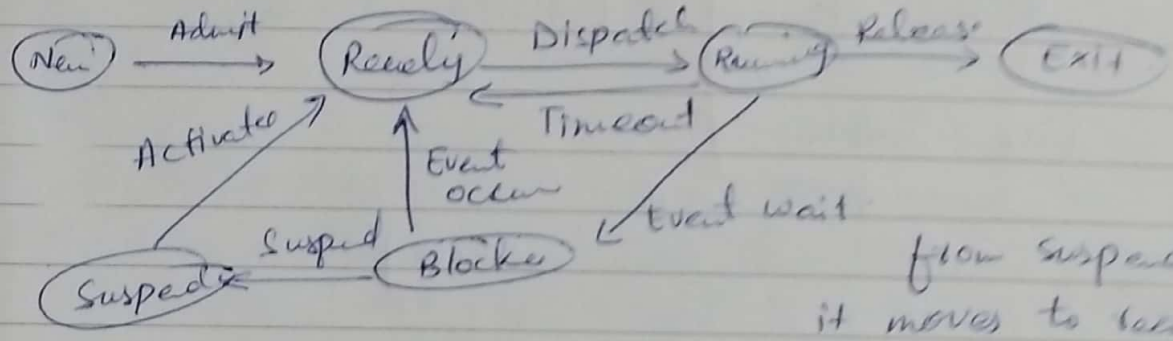
7-12-23

Suspended Processes in OS

Suspended Process:

→ Process are paused due to a malfuloning.
→ They're moved to disk.
→ Blocked state becomes suspended state when swapped to disk.
→ Two new states.
  • Blocked, suspended • Ready, suspended
→ when process is in blocked state & required events occuring then it'll be moved to suspended state.

① One-Suspended State (6 States Process model)



from suspended it moves to ready as a new process.

② Two-Suspended State (7 states).

Ready/Suspend. Process execute without doing any useful activity.

Reasons for Process Suspension:

① Swapping: swap any two processes, changing user-address space.
② Other OS reasons:
③ Interactive user request: user want to ~~sus~~ suspend a process.
④ Timing:
⑧ Parent process request: parent want to end.

Processes will be managed. How?

Operating System Control Structures.

A data structure like table will store state of process.

Memory Tables:

Processes are stored in memory (secondary / primary).
Protection of memory (as it is shared).
Information needed for virtual memory -

I/O Tables:

- I/O device is available or assigned.
- Status of operation.

File Table:

- where files are.
- location in main memory.

Process Table:

process id, status -

Process location:

PCB manages.
collection of program, data, stack & attributes.
↳ process image.

LECTURE #5

7-12-23

THREADS AND THEIR MANAGEMENT

Threads:

- Threads are light-weight process.
- A sequence of instuctions, not a program. it can't run on its own.
- A process is divided into threads (smaller tasks)
- CPU Utilizes.
- executes parallely, concurrently.

Types of Threading:

① Single Threading:

Single Threaded Approach:: Single execution path per process, where concept of thread is not there.

② Multithreading::

- threads executing concurrently within a single
- A process / task is a unit of resource Process ownership:

Examples::

1 → Web Browsers:: One thread will retrieve data from one resource (server). Google etc.

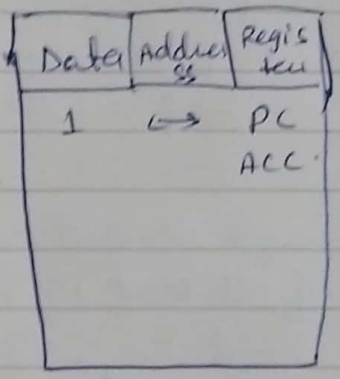
2 → Word Processors:: one thread will get text based.. another one will get design related attributes.. MS Office. etc.

Benefits:

- Process response much better.
- fast processing.

- communication b/w threads: (efficient) because they share address space
- multiprocessor system can get advantage of multithreading.

Example:
MS DOS in UNIX used single-threaded approach.



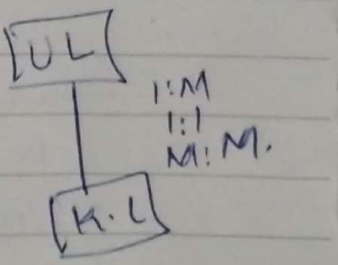
Thread Control Block is in PCB.

Implementation of Threads.

① Kernel Level Threads:
Implemented by kernel.

② User-Level Threads:
Managed in user space.
For every thread a private thread table.
It consists of PC, Stack Pointer, Registers.

③ Combined Approaches:
Solaris used hybrid approach.
one to many / many to many / one to one mapping for user level & kernel level interprocess communication.



- If a process is suspended, every thread in the process would be suspended
- Same for termination.

Threads State:

1. Spawn: One thread can create another thread.
2. Block:
3. Unblock:
4. Finish: Deallocate register context & stacks.

Remote Procedure Call Using Threads RPC

Remote Procedure Call is a software/distributed communication protocol in which one process (client) requests another process (server) to communicate, synchronize, & ~~provide to and forth~~ disseminate* information to and forth from source to destination.

*transfer.

FINALS

21-12-23.

Multithreading Models & Thread States in OS:

Multithreaded Models:

One to One Model:

each of the user level thread ~~is~~ is mapped to a kernel level thread.

using system blocking call, other threads will be blocked.

Many to One Model:

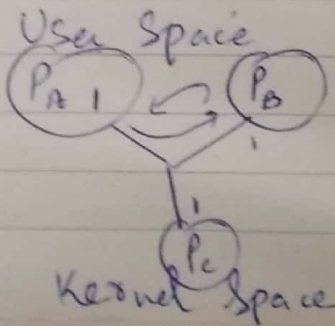
- multiple user-level threads are mapped to one kernel level thread.
- if ~~one~~ kernel level thread calls system blocked call, every user level thread will be blocked.
- even if multiple user level thread are mapped to a kernel level thread, each thread will be execute one by one.

Many to Many Model:

- multiple user level threads are mapped to multiple kernel level threads. user equal or lesser to kernel.
- multiple user threads in their corresponding kernel threads can run in parallel.

Relationship B/w Threads & Processes.

Threads : Process	Description	Example:
1 : 1	each thread belongs to a unique process	Traditional UNIX.
M : 1	Multiple threads belongs to one process with its own address space & resources	Windows, Solaris.
1 : M	A thread may migrate from one process environment to another using Interprocess communication	Ra (Clouds), Emerald.



Threads : Process	Description	Examples
M : M	Both 1:M and M:M	PRIX.

Categories of Computer System (Parallel Processing).

① Single Instruction Single Data (SISD)

Single instruction working on single data.

int a = 20

NOP
↳ no operands.

② Single Instruction Multiple Data (SIMD).

Single instruction working on multiple data.

a + b

a - b

③ Multiple Instruction Single Data (MISD)

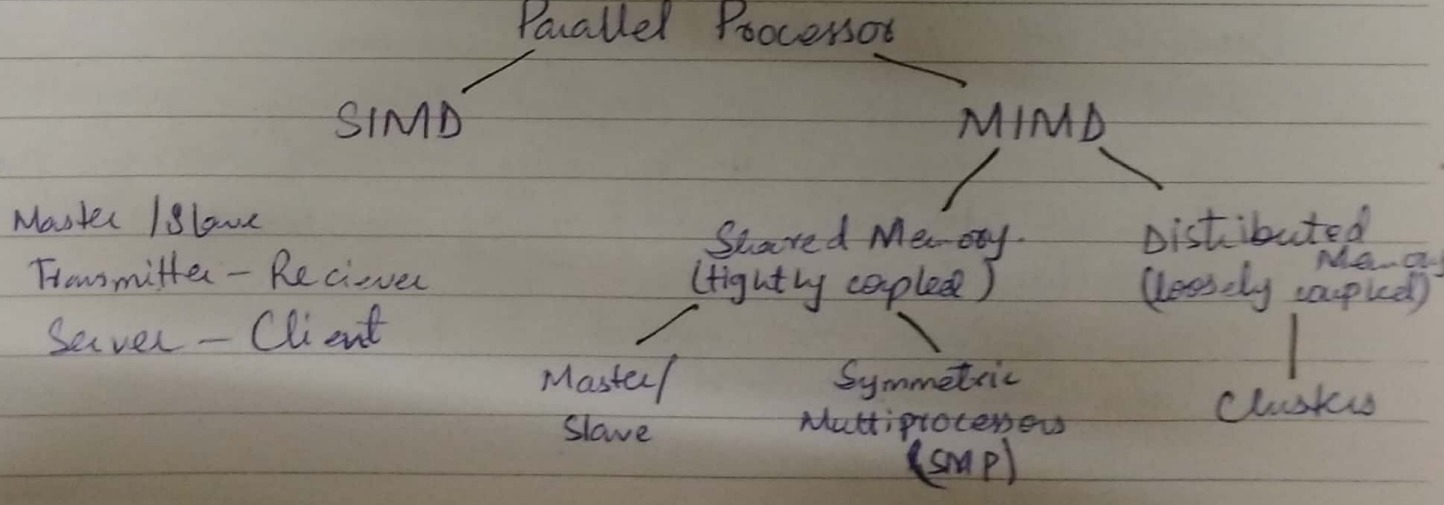
++a **x

↳ single data.

④ Multiple Instruction Multiple Data (MIMD).

a * b - c + d

simultaneous execution of multiple instructions on different data, multiple processors are working.



Symmetric Multiprocessing

- Kernel can execute on any processor.
- each processor can self-schedule processes / threads

Multiprocessor Operating System Design Considerations:

→ Simultaneous / concurrent execution of processes or threads.
→ Scheduling of processes or threads.
→ Synchronization. (IPC, multithreading) etc.
→ Memory management (Segmentation, mapping blw pages).
→ Reliability & Fault Tolerance (Effective error handling in OS).

Microkernels:

- performs essential functions of OS.
- Small OS core.
  → device drivers: use as an interface blw external devices.
  → file systems (permissions of files).
  → virtual memory managed.
  → windowing system
  → Security services.

Benefits of Microkernel Organization:

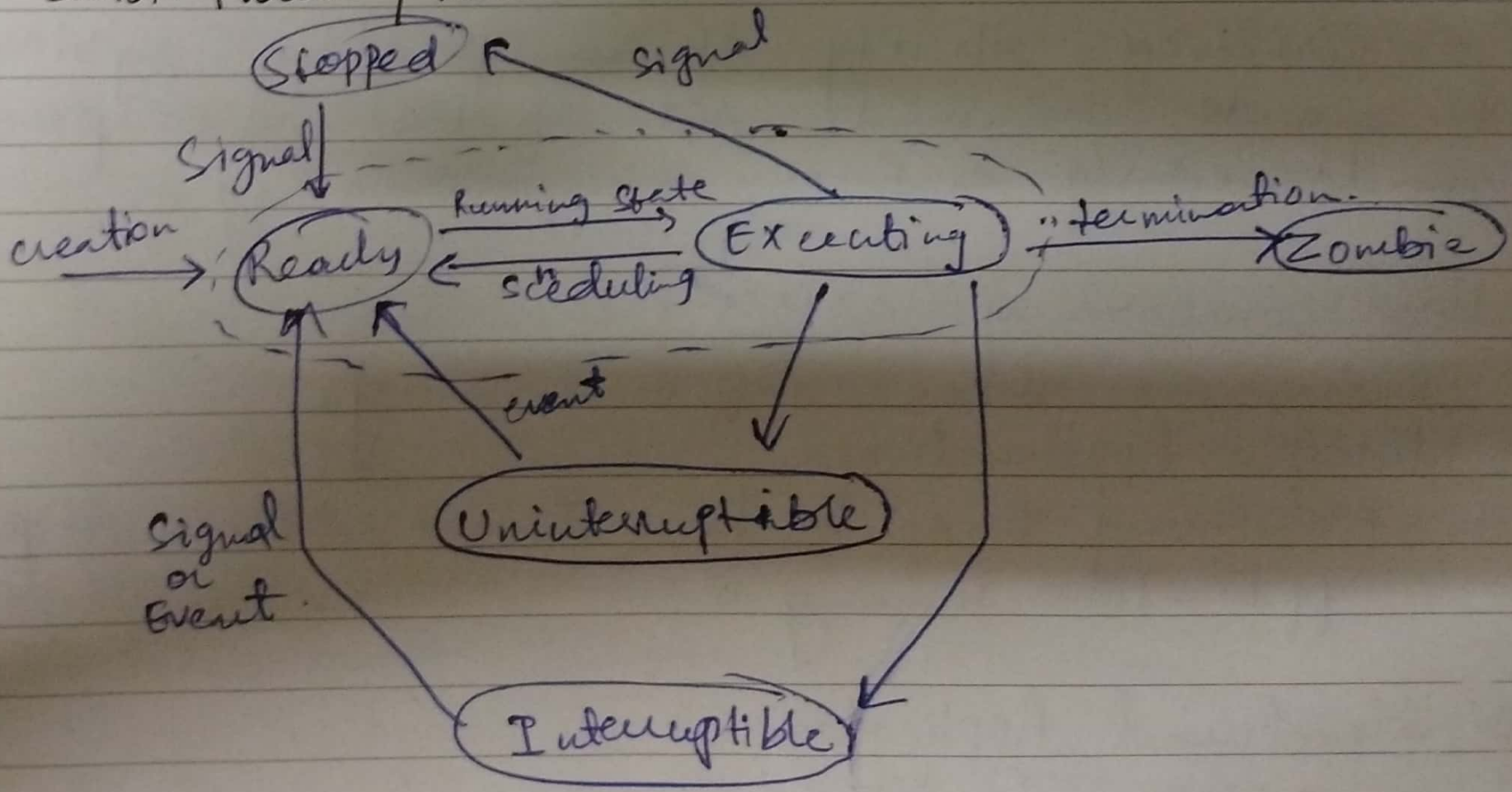
- Uniform interface on request made by a process
  uniform response, in a way, means of message passing
- Extensibility: allows addition of new services
- Flexibility:
  new features are added.
  old are subtracted.

- Portability: It embraces the change.
- Reliability:
  Modular design.
  Testing is done on microkernel.
- Distributed System Support:
  Messages are sent without knowing about target machine
- Object-oriented OS:
  Component or modules interfacing with each other.

Microkernel Design:

- Low-level memory management.
  -> low-level => assembley level.
  -> any logical (virtual page) memory is mapped to a physical one (physical page frame)
- Inter-process communication.
- I/O or interrupt management.

Linux Process / Thread Model.



CONCURRENCY
MUTUAL EXCLUSION AND SYNCHRONIZATION:

Multiple Processes.

OS is concerned with the management of processes on threads.

→ Multiprogramming
→ Multiprocessing
→ Distributed Processing.

↓

When multiple processes or threads are being executed on distinct systems without knowing the essential details of processing architectures or targeted machines.

For Example: Synchronization of multiple processes with unique address space, supportability and memory management.

Concurrency: Ability of an operating system to execute concurrent set of tasks or processes simultaneously is known as concurrency.

How concurrency arises??

There are three different ways.

1. Multiple Applications:
   processing time to be shared b/w multiple applications running.

2. Structured Applications:
   → using OOP methodology with structured way.

3. Operating System Structure:
OS is also implemented as a set of processes / threads

Concurrency & Shared Data:

- Concurrent processes may share data to support communication or can exchange information.
- Threads in a process can share global address space (globally defined attributes).
- Concurrent sharing may cause problems.
  e.g. lost updates.

Concurrency Key terms

① Atomic Operation: can't be perceived, Isolated process

② Critical Section: can't be treated by another.
A process using one resource, another process also need it, first process is in critical section, second process can't access it.

③ Deadlock: process can't change their states.
each process is waiting for another one to do something.

④ Livelock: opposite of deadlock, continuously changing their states without doing any useful activity

⑤ Mutual Exclusion: one process is in critical section by taking resources, no other process can do so
a process in critical section is said to be mutually exclusive.

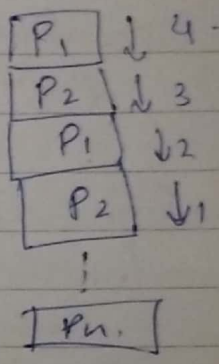
⑥ Race Condition: a situation in which multiple threads / process read / write shared data item & final result depends upon relative time

⑦ Starvation: The process are not running but they can be executed, but they are doubtful so dispatcher is not switching them.

28-12-23

Principles of Concurrency:

① Interleaving Execution of Processes.
execution time is increased. (disadvantage).



② Overlapping Execution of Processes.

Difficulties of Concurrency:

→ Sharing of global resources.
→ difficult to allocate resources optimally.
→ can't detect programmatical error.

Operating System Concerns:

The OS must
→ keep track of various process.
→ allocate & deallocate resources for active process.
→ protect data & physical resources for a process from another process's interferences.
→ process & outputs are independent of processing speed.

Process Interaction:

1. Process unaware of each other.
2. Process indirectly aware of each other (e.g. shared object).
3. Process are directly aware of each other. (have communication primitives available to them).

Timing is affected in each case

The Critical Section Problem.

waiting time for other processes can increase

Mutual Exclusion & Data Coherence:

Mutual Exclusion.

Critical Section.

Data coherence.

Deadlock & Starvation.

Deadlock

Starvation.

Mutual Exclusion.

Requirements for Mutual Exclusion:

→ Mutual exclusion must be enforced.
→ Other process do not interfere.
→ No deadlock or starvation.
→ Progress: a process must not be denied access to critical section.

Mutual Exclusion Hardware Support

Interrupt disabling:

- In uniprocessor systems.
- disabling interrupts ensures mutual exclusion.

Disadvantages..

Special Machine Instructions..
Compare & Swap Instruction.

Special Machine Instruction & Advantages::

- Share same main memory.
-
- Simple & easy.

Disadvantages::

- Busy-waiting, consume processor time.
- Starvation.
- Deadlock.

Common Concurrency Mechanism:

① Semaphores:

A variable which stores integer variable.
used for message passing.
used for signaling.
waiting decrements its value.
Signaling increments it.
may be initialized with a non-negative number

② Binary Semephores: store 1 or 0.

(13) Mutex: Strong/Weak Semaphore.

Strong Semaphore:
Process waiting for long is executed. (FIFO)

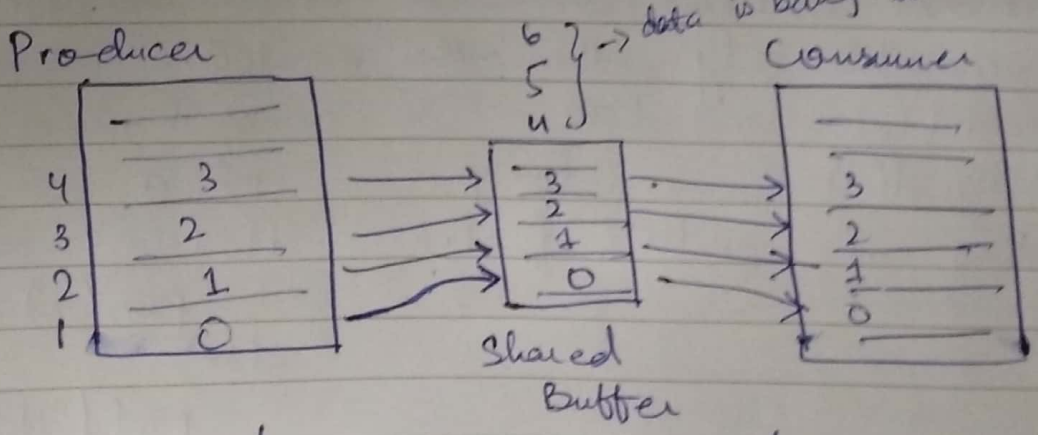
Weak Semaphore:
unordered, random ordered process in queue are removed

Shared Data Protected by a Semaphore:

Producer/Consumer Problem: /Buffer Bounded Problem.

General Situation:
→ one or more producers generates one or more data in placing it in a buffer.
→ single consumer can take out items from buffer.
→ producer/consumer may access buffer at any time.

The Problem:
→ if buffer is full, producer can't added data.
→ consumer can't get data from an empty buffer.



4-1-24

Producer/Consumer problem (Program it) and explain it.

Implementation of Semaphores:

- SemWait & semSignal are operations, implemented as atomic primitives (indivisible).
- can be implemented in hardware / firmware.
- Dekker's & Peterson's algorithm
- hardware-supported schemes for mutual exclusion

Monitors

- a concept of c process synchronization
- a programming language construct.
- can be implemented using programming languages e.g. Java.

Monitor Characteristics.

- Local variable can only be accessed using monitor's procedure
- A process enters by invoking monitor's procedure.
- only one process is handled at a time.

Synchronization

- conditional variables, contained in the monitor, they are accessible only within the monitor.
  - → cwait(p) wait karo.
  - → cSignal(1) resume activity.
  - they are monitor's function.

```
read name.
int a
if (a == name)
    prin(a)
else
    print("Not allowed")
```

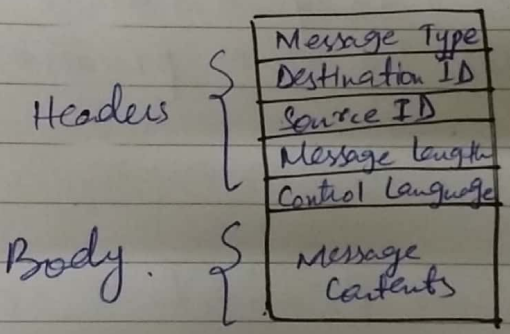
Message Passing: communication among two or more processes.

- Process interaction requires
  - ↳ Synchronization: enforce mutual exclusion
  - ↳ communication: gives primitives of send & recieve to exchange information.
- message passing does both functions.
- two primitives:

  send (destination, message).

  receive (source, message)

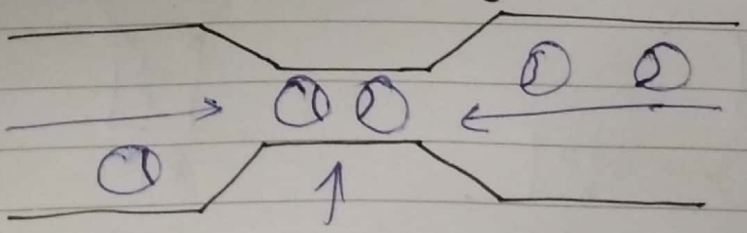
General Message Format:



The Deadlock Problem

- set of process, each using a resource & waiting for another resource that is being used by another process.

Bridge Crossing Problem



Starvation is possible.

deadlock.

System Model

- Resources type : $R_1, R_2, \ldots, R_n$.
  CPU cycles, memory, I/O devices.
- each resource has one or more instances.
- resource follows.
  request, use, release.

Deadlock Characterization:

Four conditions.

① Mutual Exclusion: one process at a time can use a resource.

② Hold & Wait: a process holding a resource & waiting for other resources, used by other processes.

③ No preemption:

④ Circular Wait:

$P_0 \xrightarrow{wait} P_1 \xrightarrow{wait} P_2 \xrightarrow{wait} P_3 \xrightarrow{wait} P_0$

or $P_0 \xrightarrow{wait} P_1 \xrightarrow{wait} P_0$

Resource Allocation Graph

directed

P = Processes R = Resources.

request edge : $P \rightarrow R$

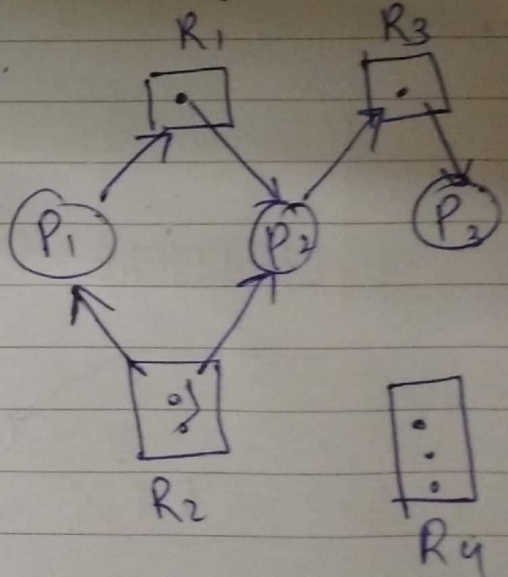
assignment edge : $R \rightarrow P$

Process

Resource 4 instances

P requests R

R assigns P



Relationship of cycles to deadlocks:

- no cycles => no deadlocks.
- contains a cycle
-

Methods for Handling Deadlocks:

- Prevention: avoid scenarios which leads to deadlock.
- Avoidance, ensure, system doesn't enter an unsafe state. (↓ possibility of deadlocks.)
- Detection: allow deadlock, apply algorithms. apply recovery mechanisms.
- Do nothing: ignore the problem, user or system responds to the problem.

Deadlock Prevention:

- Mutual Exclusion.
- Hold in Wait
- No preemption
- Circular wait.

Deadlock Avoidance:

- available amount of resources, allocated resources
- deadlock avoidance algorithm.
- resource allocation state -> no. of available & allocate resources.

Resource Request Algorithm → Banker's Safety Algorithm → Algorithm

11-1-24.

THE DEADLOCK PROBLEM

Safe State :. no deadlocks.
Unsafe State :. Possibility of deadlock.
Avoidance : ensure that the system will never enter an unsafe state.

Deadlock Detection:

→ an algorithm to examine / detect the deadlock.
→ an algorithm to recover from the deadlock.

Recovery from deadlock:

two approaches
↳ Process termination
↳ resource preemption.

① Process Termination:

→ Abort all deadlocked processes. Terminate the process
→ Abort one process at a time until the deadlock...

② Resource Preemption:

→ move resources from one process which is making the deadlock, to another process.
→ three issues
- Selecting a victim, which resources & processes are to be preempted?
- Rollback: preempting a resource, process will do what, it'll go in ready state
- Starvation: Guaranteeing, resources will not be preempted always.

Banker's Algorithm:

- Deadlock avoidance algorithm, ~~or~~
- There are two algorithmic approaches.
  ① Safety algorithm : check for safe state
  ② Resource request ~~allocation~~ algorithm: checks if the resource can be allocated or not.

Data Structures used to ~~implement~~ it:.

Matrix.

→ available
→ max : maximum resources.
→ allocation.
→ need : need matrix Max - Allocation.

1. SAFETY ALGORITHM:

Process	Allocation A B C	Max A B C	Available (work) A B C	Need Max - Allocation A B C
P_0	0 1 0	7 5 3	~~3 3 2~~	7 4 3
P_1	2 0 0	3 2 2	~~5 3 2~~ (P_1)	1 2 2
P_2	3 0 2	9 0 2	~~7 4 3~~ (P_3)	6 0 0
P_3	2 1 1	2 2 2	~~7 4 5~~ (P_4)	0 1 1
P_4	0 0 2	4 3 3	~~7 5 5~~ (P_0) 1057 (P_2)	4 3 1

Compare Need with Available (work)

	Need	≤	Available	
P_0	743	≤	3 3 2	× Unsafe
P_1	122	≤	3 3 2	✓ => available = available + allocation

= 332 + 200
= 532.

Safe
P_1, P_3, P_4, P_0, P_2

$P_2 = 600 \leq 532$ ✗

$P_3 = 011 \leq 532$ ✓ => available = 532 + 211 = 743

$P_4 = 431 \leq 743$ ✓ => available = 743 + 002 = 745

$P_0 = 743 \leq 745$ ✓ => available = 745 + 010 = 755

$P_2 = 600 \leq 755$ ✓ => available = 755 + 302 = 1057

P_1, P_3, P_4, P_0, P_2 -> Safe states; Process should run in this sequence.

Assignment 2 for ~~Ba~~ Safety algorithm.

Resource request Algorithm:

Steps:

1. Request $\leq$ need => goto step 2.
2. Request $\leq$ available => goto step 3.
3. i) Available = available - request.
   ii) Allocation = Allocation + request.
   iii) Need = need - request.
4. Check if system is safe or not using safety algorithm.

Process	Allocation A B C	Max A B C	Available	Need = Max - Allocation A B C.
P_0	0 1 0	7 5 3	~~3 3 2~~ ~~2 3 0~~ P_1	7 4 3
P_1	~~2 3 0 2~~	3 2 2	~~5 3 2~~ P_1 ~~7 4 3~~ P_3	~~1 2 2~~ 0 2 0
P_2	3 0 2	9 0 2	~~7 4 5~~ P_4	6 0 0
P_3	2 1 1	2 2 2	~~7 5 5~~ P_0	0 1 1
P_4	0 0 2	4 3 3	1057 P_2	4 3 1

$P_1 \rightarrow R(102)$.

1. Request $\leq$ need.
   $102 \leq 122$ ✓
2. Request $\leq$ available.
   $102 \leq 332$. ✓
3. (i) Available = Available − request = 332 − 102 = 230
   (ii) Allocation = Allocation + request = 200 + 102 = 302.
   (iii) Need = Need − request = 122 − 102 = 020.

4. Safety Algorithm:
   P_0 => Need $\leq$ Available.
   P_0 => 743 $\leq$ 230 X available = available + allocation

   P_1 => 020 $\leq$ 230 ✓ available = 230 + 302
   = 532

   P_2 => 600 $\leq$ 532 X

   P_3 => 011 $\leq$ 532 ✓ available = 532 + 211
   = 743

   P_4 => 431 $\leq$ 743 ✓ available = 743 + 002
   = 745

   P_0 => ~~010~~ 743 $\leq$ 745 ✓ P_1 . P_3 . P_4 . P_0 . P_2
   available = 745 + 010 = 755

   P_2 => 600 $\leq$ 755 ✓ available = 755 + 302
   1057

P_1 P_3 P_4 P_0 P_2.

Multithreading models.
microkernels. architecture.
concurrency techniques
key terms.
Semaphore - - - Monitor.
Deadlocks.
Resource allocation graph.

MEMORY MANAGEMENT (PAGING AND SEGMENTATION

18-1-24

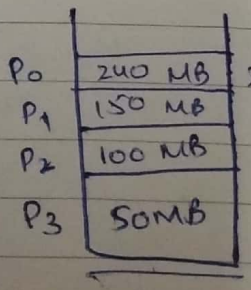
Memory: Internal storage area of the computer.
Primary memory: RAM, ROM, cache.
Secondary memory: Hard disk.

Data :- raw data, unorganized, unprocessed data.
Information: processed data, meaningful.

Types of Memory :-
① Physical Memory: tangible, RAM, Hard disk.
② Logical Memory: CPU memory, cache, embedded.
③ Virtual Memory: imaginary memory, an extension of logical memory, take a portion of memory.

If the size of a program > available memory size then we use virtual memory

Available Memory:
Amount of RAM not in use



256MB - 16MB
= 106MB
= 156MB
= 206MB
AV MEM = 484 MB

Memory Management

- A functionality of OS to handle & manage ~~OS~~ Primary memory.
- Keeps track of memory location that either it allocated to some process or it is free.
- MMU - Memory Management Unit does this functionalities.

→ Two broad tasks

1. Each process must have enough memory to execute.
2.

Memory Management Unit (MMU):

MMU → how OS handles RAM.

→ Keep track of what parts of memory are in use?
→ allocate memory to processes when needed.
→ Deallocate when done.
→ Swapping ~~or~~ & paging b/w main memory & disk.

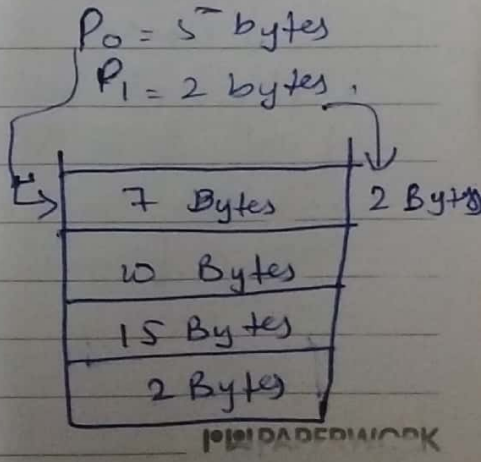
Static & Dynamic ~~&~~ Memory:

Static RAM SRAM: faster & less volatile.
requires more power.

Dynamic: memory will be constantly reenergized.
RAM → DRAM.

How RAM works?

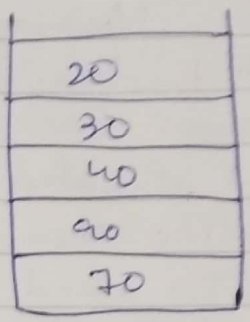
3 ways to put data in RAM:



① First Fit: Scans the memory from begining & choose available block.
- faster in allocation, but wastes memory

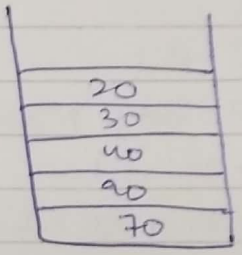
Worst Fit:

Search the entire list of blocks
Search for largest memory block.
It less efficient.
Time & memory waste.



Best Fit:

- Chooses the block that is closest to the required.
- Traverse all block & identifies the closest block.
- Takes more time but no memory wastage (time-space tradeoff).



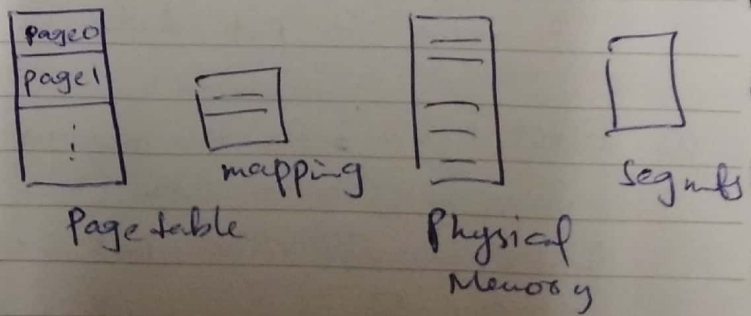
Paging:

25-01-24

- A memory management technique.
- A logical concept.
- Divides memory into fixed sized of pages.
- Faster access of data.
- Memory → Pages —(Mapping function)→ Segments (frames)
  - Pages → page table
  - Segments → frame list
- Converts into segments → a physical concept.

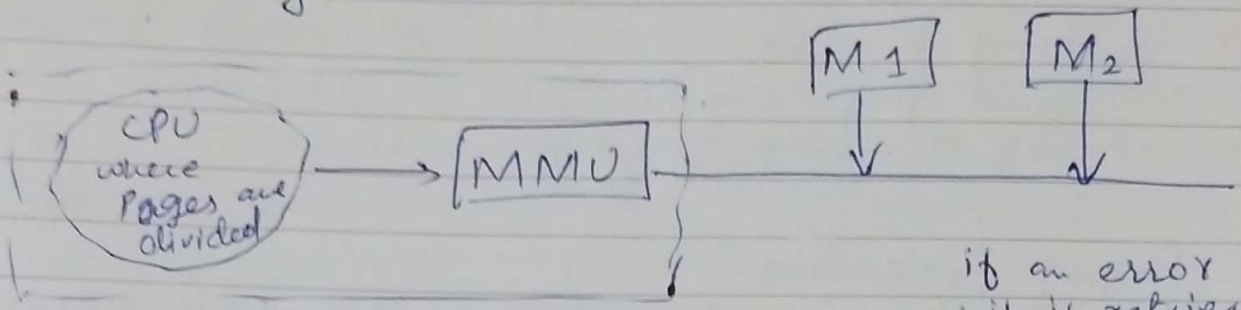
Frames

- physical concept.
- present in RAM
- pages can't be seen but frames can be.



Mapping: A translation mechanism.

- Translate/ convert pages (virtual address) into frames (physical address)
- Performed by M.M.U.

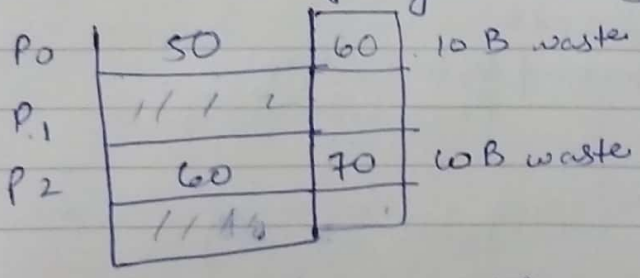


Logical address & Physical address.

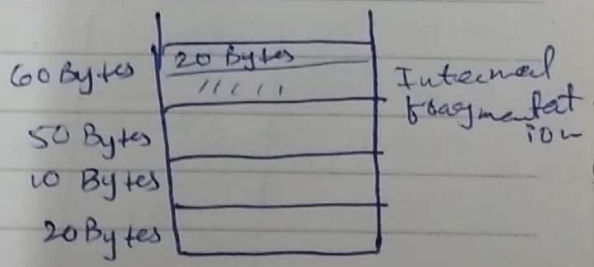
Page Translation: Examples: architectural diagram. (line wali)
block diagram.

Paging Protection:
→ add valid / invalid bit.

Fragmentation: It is basically an unwanted problem in which memory spaces are allocated with ~~significa~~ a certain memory size and the remaining amount of space is ~~unuti~~ not utilized or wasted, this situation is known as fragmentation.



External fragmentation.



Internal Fragmentation.

Paging

Advantages:

① No external fragmentation.
② Simple algorithms.
③ Swapping is easy.
(fixed page size).
equal sized pages in page frames.

Disadvantages

① Internal fragmentation
② Page tables may consume memory.

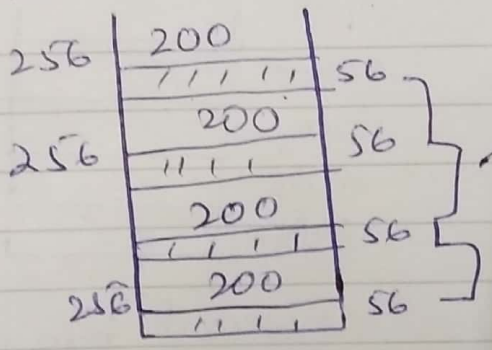
25-01-24

Paging Protection:

- achieved by insertion of an additional bit, VALID / INVALID
- 14 bit address space = total 2^{14} = 16383.
- address limit is 10447 68
- If 5 process (P0 – P4) are defined within this address, they're considered as VALID BIT
-

Fragmentation: Ex

Process 1.



224 MB
Free / Internally fragmented Space.

contiguous memory.

It's question can be like this

Q. Consider that 1024 MB memory with 256 MB of 4 allocated memory & processes. Assume that each memory process has allocated with 200 MB space What will be the free / fragmented space?

SEGMENTATION

- Memory management way.
- memory protection can be achieved
- due to internal fragmentation, user's view of memory is lost.
- combination of segments (for user).
- each segment has a specific length & some permissions
- When a process tries to access memory, it checks whether it has required permissions, & the length specified, is in range (processe's required).

Segmentation is a memory management technique in which memory is divided into multiple segments or blocks, which forms a frame list.

Example.

① Virtual addresses (Pages) are translated via a mapping function, the required page table is converted into a fixed size or variable size segment (frame).

Process 1

P_0	40	40
P_1	50	50
P_2	10	10
P_3	20	20

All blocks are fully utilized contiguous

⇒

P_1	50
P_3	20

40+10 = 50MB memory is now externally fragmented.

there are free memory blw blocks, non-contiguous.

① Consider a memory size 1024MB, assume that each memory block or segment has been assigned with 256MB of each memory block or space. Suppose that Po has occupied

Q① Consider a memory size 1024 MB assume that each memory block or segment has been assigned with 256 MB of each memory block or space. Suppose that P_0, P_1, P_2, P_3 has occupied 240 MB are assigned in each memory segment - What will be the internally fragmented space?

② If P_0 and P_2 has completed their execution then compute the externally fragmented space:

③ After computing the external fragmented space, Will P_1 & P_3 will be executed or not?
non-contiguous memory (can't be executed).

Segmentation hardware.
limit → length of segmented.
base → starting index address.

offset value is 12 (by default).

Segment Protection:
① advantage, protection is associated with each segment; also the specified length
② Sharing can be achieved. multiple processes can use one segment (multiple referencing).
③ Advantages
- No internal fragmentation
- Segment tables take less memory space.

Disadvantages:
- Memory management is costly
- When process are loaded external fragmentation occurs.

In combined paging | segmentation,

CPU Scheduling Algorithms

1) FCFS - First Come First Serve. } Non-preemptive
2) SJF - Shortest Job First
3) SRTJF - Shortest Remaining Job First } Preemptive.
4) RR - Round Robin.
5) Priority based.

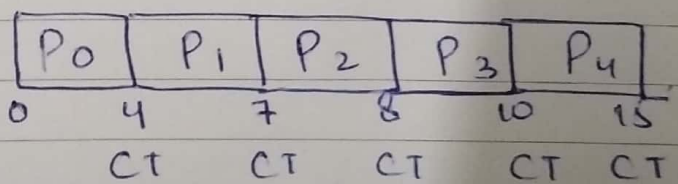
1. FCFS

PID/Process	AT (Arrival Time)	BT (Burst Time)	WT (Waiting Time)	TAT (Turn Around Time)
P_0	0	4	4-4=0	4-0=4
P_1	1	3	6-3=3	7-1=6
P_2	2	1	6-1=5	8-2=6
P_3	3	2	7-2=5	10-3=7
P_4	4	5	11-5=6	15-4=11

Formula for WT (Waiting Time) = TAT - BT
Formula for TAT (Turn Around Time) = CT - AT

Gantt Chart.

CT (Completion Time).



$$\text{Average Time} = \frac{\Sigma WT}{\Sigma P} = \frac{0+3+5+5+6}{5} = \frac{19}{5} \approx 3.$$

Average Turn Around Time = $\frac{\text{Total of TAT}}{\text{No. of Processes}}$

$$ATAT = \frac{4+6+6+7+11}{5} = \frac{40}{5} = 8$$

$$\text{Throughput} = \frac{\text{No. of Processes}}{\text{Max(CT)} - \text{Min(AT)}} = \frac{5}{15-0} = \frac{5}{15} = \boxed{\frac{1}{3}}$$

Nonpreemptive

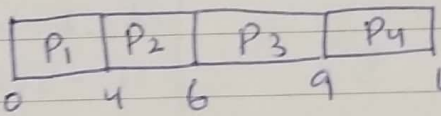
1-04-24

1) FCFS

PID	AT	BT	TAT	WT	CT
P_1	0	4			
P_2	0	2			
P_3	0	3			
P_4	0	1			

if AT is not given, consider 0 for every process.

Gantt Chart:



TAT

WT

Total TAT

Total CT

Average TAT

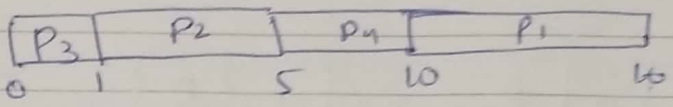
Average WT

Throughput.

2. Shortest Job First - SJF

Non - Preemptive.

Process	AT	BT	WT	TAT	CT
P_1	0.0	6	10	16	16
P_2	0.0	4	1	5	5
P_3	0.0	1	0	1	1
P_4	0.0	5	5	10	10



AWT = $\frac{10+1+0+5}{4} = \frac{16}{4} = 4$

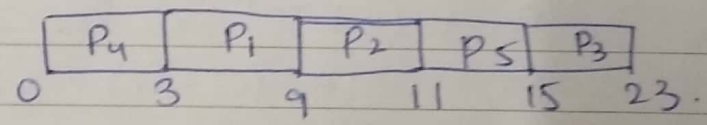
ATAT = $\frac{16+5+1+10}{4} = \frac{32}{4} = 8$

Throughput = $\frac{4}{16-0} = \frac{1}{4}$

Example.

PID	AT	BT	CT	TAT	WT
P_1	2	6	9	7	1
P_2	5	2	11	6	4
P_3	1	8	23	22	14
P_4	0	3	3	3	0
P_5	4	4	15	11	7

Gantt Chart



First check, whose arrival time is 0, it will be first. Then, check ~~when~~ within the completion time of first process, which processes, arrived? if they are more than one, then take the less burst time process ...

AWT = 1+4+14+0+7/5 = 26/5

ATAT = 7+6+22+3+11/5 = 49/5.

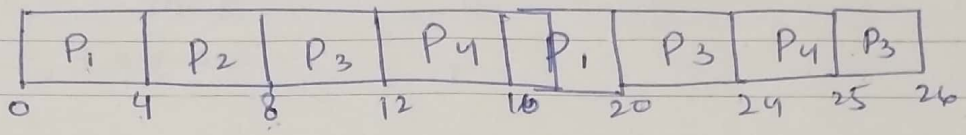
Throughput = 5/23-0 = 5/23.

Preemptive

3. Round Robin (RR). QT - Quantum Time = 4.

PID	AT	BT	CT	TAT	WT
P_1	0	~~8~~ ~~4~~ 0	20	20	12
P_2	1	~~4~~ 0	8	7	3
P_3	2	~~9~~ ~~8~~ ~~4~~ 0	26	24	15
P_4	3	~~5~~ ~~1~~ 0	25	22	17

Gantt Chart:



After Quantum time, next process.
if remaining time / new burst time < Quantum time then run according to ~~less~~ burst time.

AWT = 12 + 3 + 15 + 17 / 4 =

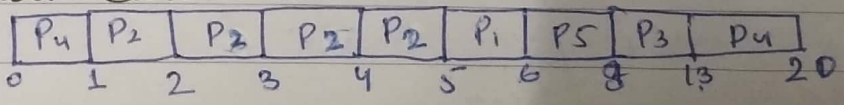
ATAT = 20 + 7 + 24 + 22 / 4 =

Throughput = 4/26 − 0 = 4/26 = 2/13

4. Shortest Remaining Job First (SRJF) or Preemptive SJF

PID	AT	BT	CT	TAT (CT-AT)	WT (TAT-BT)
P_1	4	1	6	2	1
P_2	1	~~4~~ ~~3~~ ~~2~~ 0	5	4	0
P_3	2	5	13	11	6
P_4	0	~~8~~ 7	20	20	12
P_5	3	2	8	5	3

Gantt Chart



AWT = $\frac{1+0+6+12+3}{5} = \frac{22}{5}$

ATAT = $\frac{2+4+11+20+5}{5} = \frac{43}{5}$

Throughput = $\frac{5}{20-0} = \frac{1}{4}$

Steps for SRJF (Criteria).

① Increase 1 unit of completion time & Subtract 1 unit of burst time.

② If the burst time is same then check the shortest AT between two or multiple processes.

③ If all the processes arrive ~~after~~ by increasing 1 unit of CT then implement SJF.