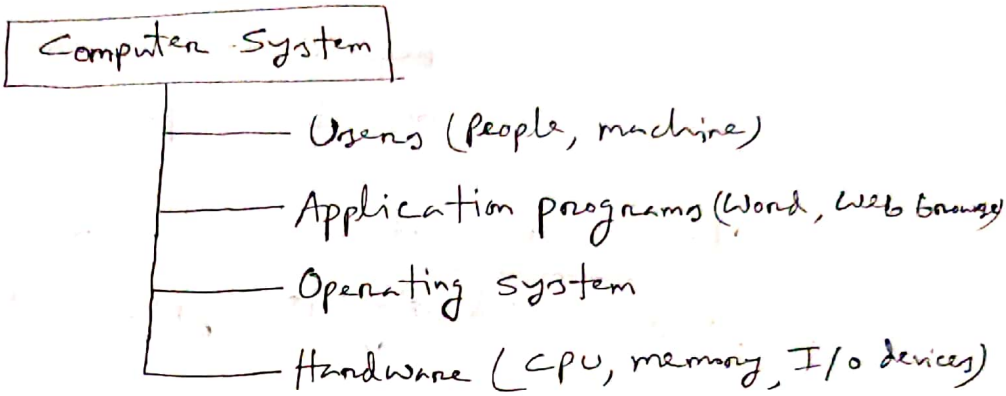
Mid (Lecture 1)

[Operating System concepts]

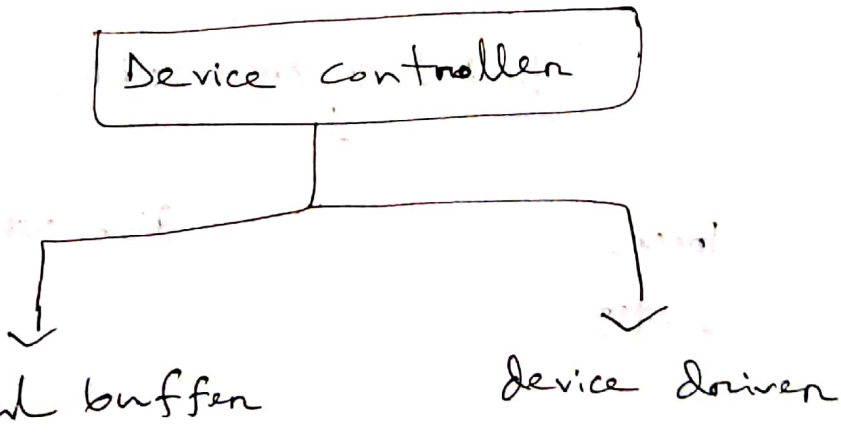
Computer System



* Operating System is a resource allocator and control program making efficient use of hardware and managing execution of user programs.

* Kernal: The one program running at all times on the computer is called Kernal.

*



* An operating system is interrupt driven.

I/O Structure :

① System call: Request to the OS to allow user for I/O completion.

② Device status table: It contains entry for each I/O device indicating its type, address and state.

Storage Structure :

① Main memory → large storage media that the CPU can access.
- Random access
- Typically volatile
- Typically Random access memory

② Secondary storage → non-volatile

③ Hard disk drives (HDD)
- tracks
  - sectors
- disk controller

④ Non volatile memory (NVM) → non-volatile → faster than hard disk

Storage Hierarchy

* Speed * Cost * Volatility

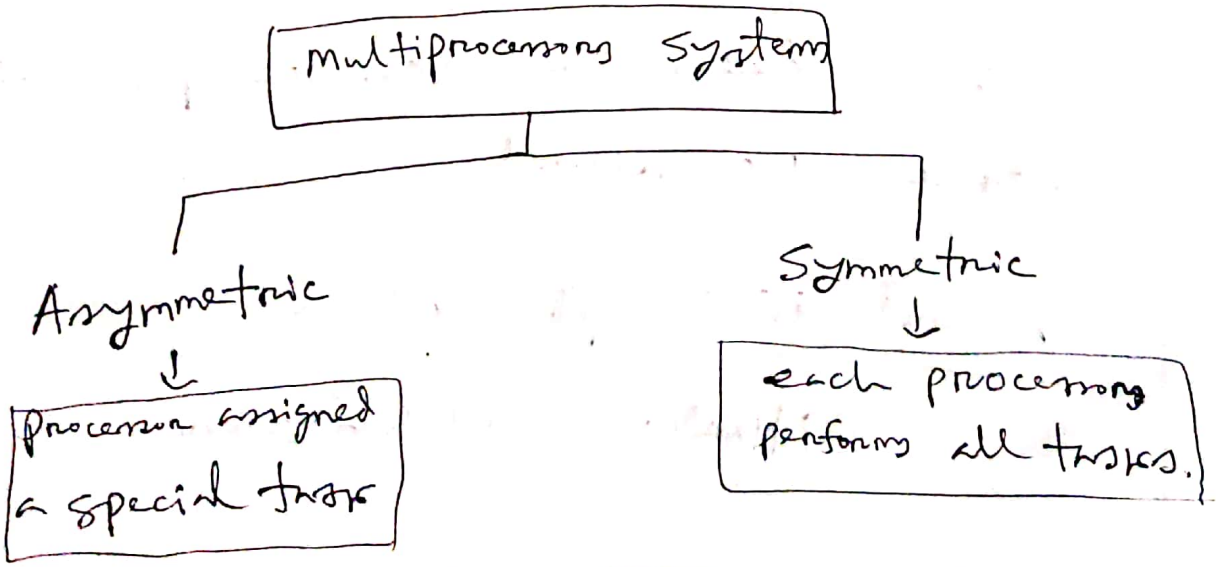
* **Caching:** Copying info. into faster storage system. main memory can be viewed as a cache for secondary storage.

* **Device driver:** for each device controller to manage I/O. Provides Uniform interface between controller & kernel.

Computer System Architecture:

① Single processors systems.

② multiprocessors systems / Parallel systems / tightly-coupled systems



Clustered System: Like multiprocessor system.

* Sharing storage via a storage-area-network (SAN)

* two types: (i) Asymmetric clustering,
(ii) Symmetric clustering.

* High performance computing (HPC)
* Distributed lock manager (DLM).

Operating System Operations:

* Bootstrap prog: Simple code to initialize the system.
✓ Load the kernel.

* System daemons → Service provided outside the kernel.

* Kernel interrupt driven:

(i) Hardware (ii) Software

Multiprogramming & Multitasking (★★★)

* (Code & Data) → CPU always has one to execute.
* A subset of total jobs in system is kept in memory.
* one job selected & run via job scheduling.
* When it has to wait (i/o device), OS switches to another job.

Multiprogramming

multitasking or timesharing

It is a logical extension in which cpu switches jobs so frequently.

* Response time should be <1 sec.
* Each user ~~bat~~ has at least one program to execute → [process]
* Several jobs ready to run at the same time ⇒ CPU Scheduling
* switches another job (OS).

Dual mode
- user mode
- kernel "

* Virtual machine manager (VMM) mode for guest VMs.

Process management

* A process is a program in execution.
* Program is a passive entity.
* Process is an active entity.

What is file?

=) Abstracts physical properties to logical storage unit is called file.

Narayan

02/11/2020

Operating System Structures

#OS Systems services :

* Some functions that are helpful to the user.

① UI (user interface) —> Command line (CLI)
Graphics user interface (GUI)
touch-screen, Batch.

② Program execution ③ I/O operations

④ file system manipulation ⑤ Communications

⑥ Error detection

* Some functions that are sharing resources to the user.

① Resource allocations (Cpu cycles, file storage, I/O devices, memory)

② logging ③ Protection & Security.

System calls

* Programming interface to the services provided by the OS.

* Application programming interface (API)

⊛ Three most common API:

① Win 32 API for windows,

② POSIX API for Unix, Linux, mac, os x.

③ Java API for JVM.

⊛ System calls sequence to copy the contents of one file to another file.

System calls parameter passing:

* Three general methods used to pass parameters to the OS.

① Simplest ② Block / Table ③ Stack

* Types of system calls:

① Process control type (load, execute) ② file management (creat, write)

③ Device management (request, release device) ④ Communications (send, receive msg)

⑤ Information maintainance (get/set time; get/set data)

⑥ Protection type. (get & set permissions; allow user)

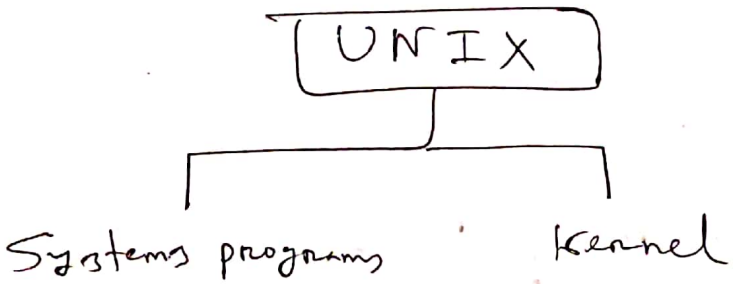
① + ② + ③ → {Create, delete, terminate}

④ → {open, close, read, write}

* Example:

	Windows	Unix / Linux
Process control	CreateProcess()	fork()
	ExitProcess()	exit()

Unix



MicroKernels

* Example: Mach → Mac OS X kernel partly based on Mach

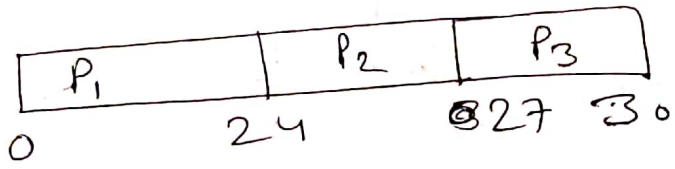
* message passing communication.

Process Scheduling

FCFS:

Process	Burst time
P_1	24
P_2	3
P_3	3

* Gantt chart:



Avg waiting time: (S.T − A.T)/total process

$(0-0)+(24-0)+(27-0)/3$

$= 27/3$

Avg turnaround time: Burst time + W.T

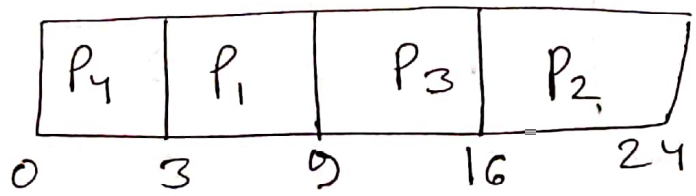
$(24+0)+(3+24)+(3+27)/3$

$= 81/3 = 27$

SJF : Shortest job first :

Process	Burst time
P_1	6
P_2	8
P_3	7
P_4	3

=> Gantt chart :



Now,

Avg waiting time = (T.T − A.T) / total process

= (3 − 0) + (16 − 0) + (9 − 0) + (0 − 0) / 4

= (3 + 16 + 9 + 0) / 4

= 7.

Avg turnaround time: Burst time + W.T

$(3+6)+(16+8)+(9+7)+(0+13)/4$

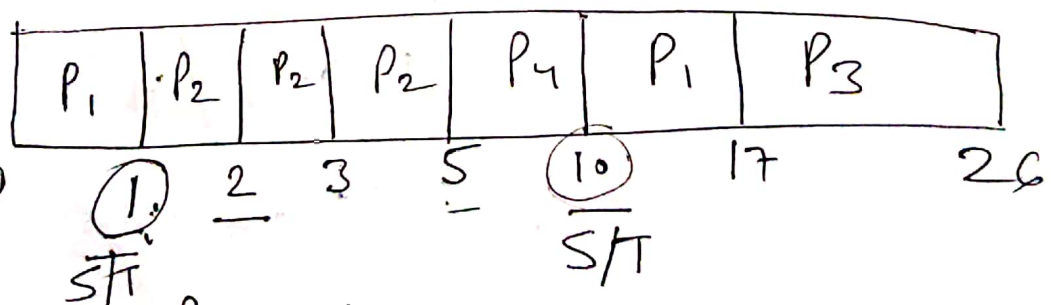
$= 52/4 = 13.$

SJF with remaining time:

Process	A. Time	Burst time	Remaining
P_1	0	8	7/0
P_2	1	4	3/2/0
P_3	2	9	9/0
P_4	3	5	5/0

~~P1 P2 P3 P4~~

=> Grantt chart:



W.T: $P_2 = 1 - 1 = 0$

$P_1 = 10 - 1 = 9$

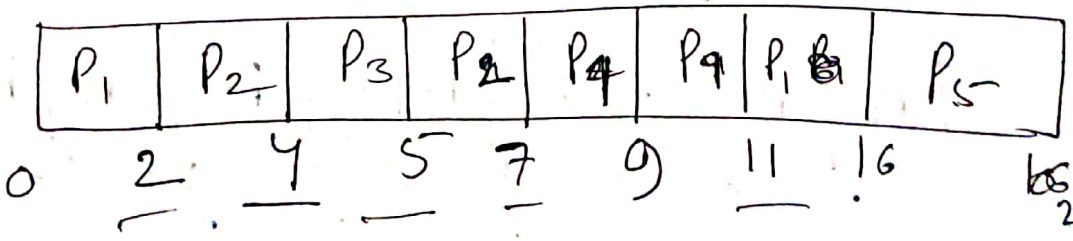
(*)

Available Process in the CPU:

~~P_1 P_2 P_5 P_3 P_4~~

Process	A.T	B.T	Remaining
P_1	0	7	5/
P_2	2	4	2/0
P_3	4	1	0 ~~0~~
P_4	5	4	2/0
P_5	3	5	

=> Gantt chart:



Waiting time: $P_1 = 11-2 = 9$

$P_2 = 5-2 = 3$

$P_3 = 4-4 = 0$

~~$P_4 = 9-5 = 4$~~

$P_4 = 7-5 = 2$

$P_5 = 16-3 = 13$

(*)

Available:

~~P_2~~ ~~P_3~~ ~~P_4~~ ~~P_1~~ ~~P_5~~

Process	A.T	B.T	Remaining time
P_2	0	12	9
P_3	3	8	6/5/3/0
P_4	5	4	0
P_1	10	10	
P_5	12	6	0

=> Gantt chart:

P_2	P_3	P_4	P_3	P_3	P_3	P_5	P_2	P_1

0 3 5 9 10 12 15 21 30 40

P_2 P_2

Avg. W.T: $(30-10)+(21-3)+(9-5)+(5-0)+(15-12)/5 = 45/5 = 9$

Priority Scheduling with round robin

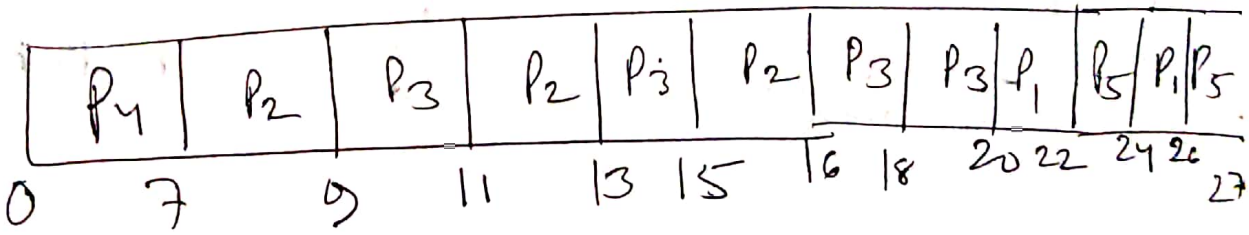
Process	B.T	Priority
P_1	4	3
P_2	5/3/1/0	2
P_3	8/6/4/0	2
P_4	7	1
P_5	3	3

Q = 2

=> Grantt chart:

Available

~~P_4~~ ~~P_2~~ ~~P_3~~ P_1 P_5



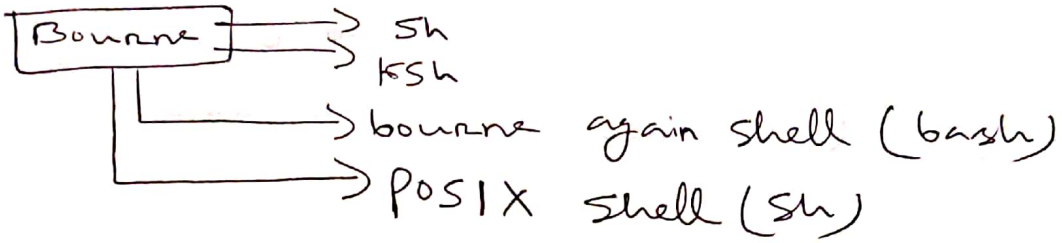
LAB (mid)

Tubayen

Lecture 1

Shell: A shell is a program that provides an interface between a user and an OS(K). (Kernel)

* two types: ① Bourne shell ($)
② C shell (%)



# Command	details
① uname -r	What kernal version
② uname -m	What architecture running
③ pwd (Print working directory)	What your current directory
④ ls (list)	used to print contents of ②
⑤ mkdir file_name	create a directory (new).
⑥ cd /folder_name	change the directory.
⑦ ls -a (all)	all files (even the hidden ones)
⑧ $ cd..	To go parent directories.
~~⑨ >> touch.txt~~	~~create empty file~~
⑨ >>touch test.txt	create an empty file.

Lecture 2

Command	details
① cat > f1.txt	——> write anything in f1 file.
② cat f1	——> To show the content of a file f1.
③ cat file1 >> f1	——> append file1 & f1.
④ cat file1 f2 > f3	——> concatenate three files.
⑤ cp source destination	——> copy files
⑥ mv oldname new-name	——> rename files
⑦ rm filename	——> remove files
⑧ rm *	——> delete all files
⑨ rm -r file	——> remove everything that folder.
⑩ uname -n	——> machine name in network
⑪ $ cal -3	——> To display the previous, current & next month
⑫ $ ncal -S	——> current month starting from sunday
⑬ $ date +%.h%.m	——> show month name and month (Jan;01)
⑭ $ date +%.H	——> display the time in hours
⑮ $ date +%.r	——> show time with am or pm.

(16) $ history ——> all the commands show that you have used in the past for the current terminal.

(17) clear ——> clear all ; starting from the first in terminal.

(18) $ bc ——> To calculate the values

Example: 2 + 3

5

(19) $ man cd ——> Show the manual page of cd command.

(20) $ wc f1 ——> To show number of words, lines and bytes

(21) $ wc -c file ——> Show number of characters in a file.

(22) $ nl -i5 filename ——> increment the line number by 5

(23) $ sort -r filename ——> Sort the content of file by reverse order.

Command — details

(24) $ Sort -u filename ⟶ Sort & remove duplicate

(25) $ head filename ⟶ display first 10 lines

(26) $ head -5 f1 f2 ⟶ print 5 lines from f1 and f2

(27) $ tail filename ⟶ display last 10 lines

(28) $ cat > cutfile.txt ⟶ cut a file

(29) cut -d",", -f 1,3 cutfile

Example:

Jubayer, 2, 3
Ahamed, 6, 9
Roots, 7, 10

⟶

Jubayer, 3
Ahamed, 9
Roots, 10

(30) # grep command

* grep -i "test" f4
(where test word finds, then this clines ~~are showed~~ will show.)

* grep -c "test" f4
(How many times test word finds?)

* grep -o "test" f4
(only the word 'test' will show)

* grep -v "test" f4
(remove the lines where test word finds and show the lines without test word)

* grep -n "test" f4
(Show the test lines with the serial number)

file permissions in Linux/Unix

* Linux is a clone of UNIX.

* Linux divides authorization by 2 types:

(I) Ownership
- User → owner of the file
- Group → A user group
- Other → Any other who has access to the file

(II) Permission
- read (r)
- write (w)
- X

Theory

Processes

16/11/2020

Process is a program in execution.

Process is an active entity.

Program becomes process when executable file loaded into memory.

Process structure

* It includes current activity : ① Program counter
② Processor's registers

* It also includes the process stack, which contains the temporary data (Parameters, return address, local variables)

* It also includes data section, which contains global variables.

* It also includes a heap.

Process state

(I) **new** : Process is being created.

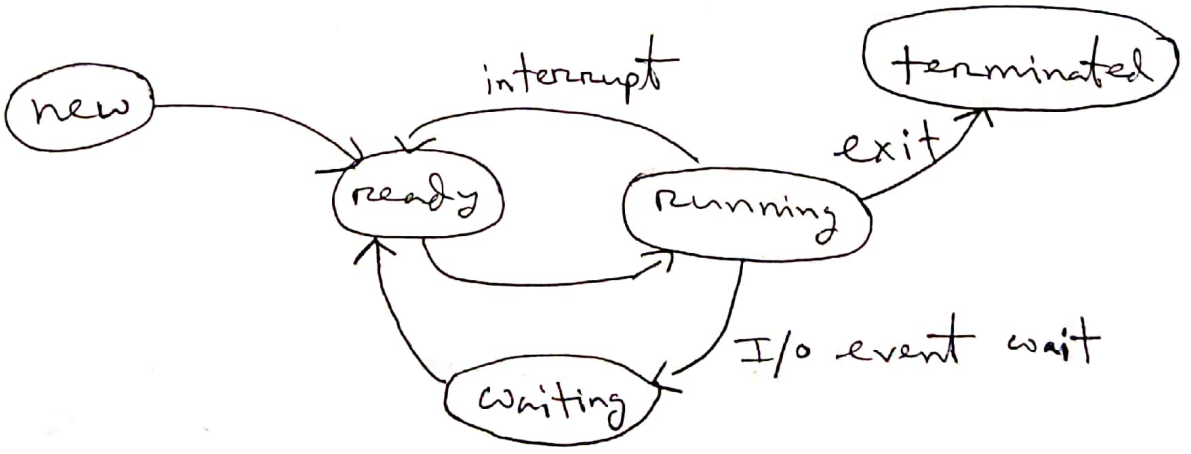
(II) **ready** : The process is waiting to be assigned to a processor.

(III) **waiting** : The process is waiting for some event to occur.

(IV) **running** : Instructions are being executed.

(V) **terminated** : Process has finished execution.

* **Diagram** :



PCB ——> Process control block

* Information associated with each process.
* Also called task control block.
* Process state: running, waiting etc

Scheduling Queues

① Job Queue: Set of all processes in the system

② Ready Queue: Set of all processes residing in main memory, ready & waiting to execute.

③ Device Queue: Set of processes waiting for an I/O device.

Schedulers

① Short term scheduler ——> CPU scheduler
└——> miliseconds (fast)

② Long term scheduler ——> Job scheduler
——> Second, min (late)
——> degree of multipro

* Processes can be described as either:

① I/O bound process: spends more time doing I/O than computations.

② CPU bound process: spends more time doing computations.

Context Switch

* When CPU switches to another process, the system must ~~have~~ save the state of the old process and load the saved state for the new process by using a context ~~st~~ switch.

* Context of a process represented in the PCB.

Operations at process

* Process creation :

- Parent process creates child process.
- Process is identified and managed via a process identifier (pid).
- Resource sharing among parents and children options.
- Execution options
  → Parent waits until children terminate.
- Address space
  → A child is a duplicate of the parent address space.

- UNIX examples

→ fork ( ) : creates a process (new).

(system call)

Process termination

- When process terminated, use exit() System call and delete it.
- A parent may terminate the child's execution using the abort() System call.
- Cascading termination : All children, grand children etc. are terminated.

IPC ⟶ Interposes communication

* Two types : (i) Shared memory,
(ii) Message passing.

Tubagus

Final

06/12/2020

CPU Scheduling

#Multilevel Queue

Priority = 0 | T_1 | T_2 | T_3 | T_4 |

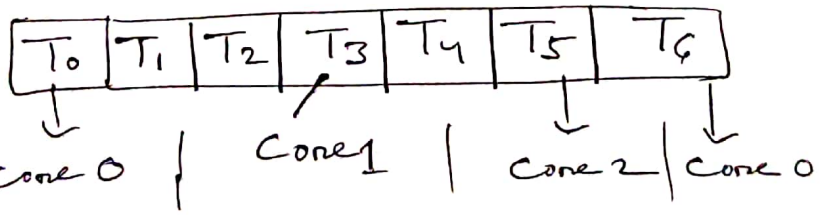
Priority = 1 | T_1 | T_2 | T_3 | T_4 |

Priority = 2 | T_1 | T_2 | T_3 | T_4 | T_5 |

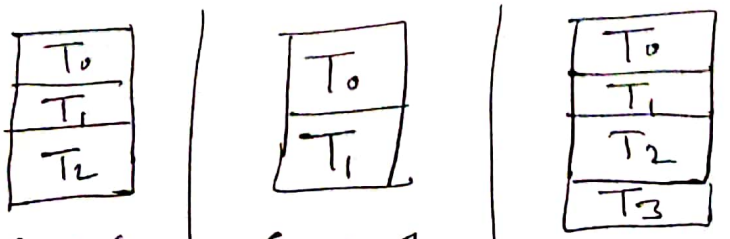
⊛ Schedule the process in the highest-priority queue.

#Multiple processor scheduling

* Symetric (Smp) multiple processing is where each processor is self scheduling.
* All threads may be in a common ready queue.



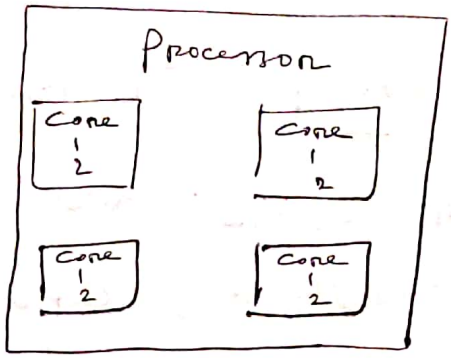
*



[each processor have it's own queue of threads.]

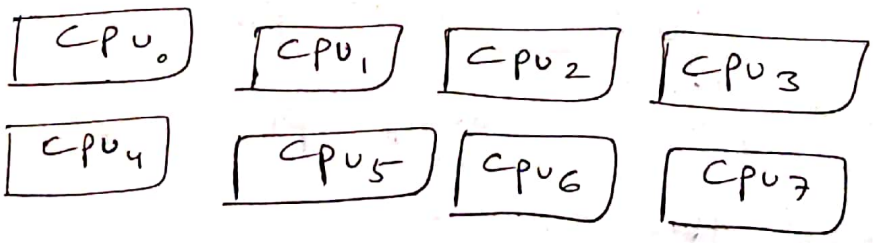
Multithreaded multicore System :

Chip multithreading (CMT) assigns each core multiple hardware threads.



{1, 2} → hardware thread

OS view: 4 X 2 = 8 cpu



SMP:

① Load balancing: প্রত্যেকটি Processor এর মধ্যে load distribution এর মধ্যেকার balance করে।

② Push migration: Periodic task checks load on each processor; কোথাও overload পেলে সেই CPU থেকে অন্য CPU তে নিয়ে যায়। Overloaded CPU থেকে অন্য CPU তে pushes task পাঠানো হয় কি না খুঁজে দেখে।

③ Pull migration: Idle processors pull waiting task from busy processors.

Real time CPU Scheduling

① Soft real time systems: Critical tasks have the highest priority, but no guarantee as to when tasks will be scheduled.

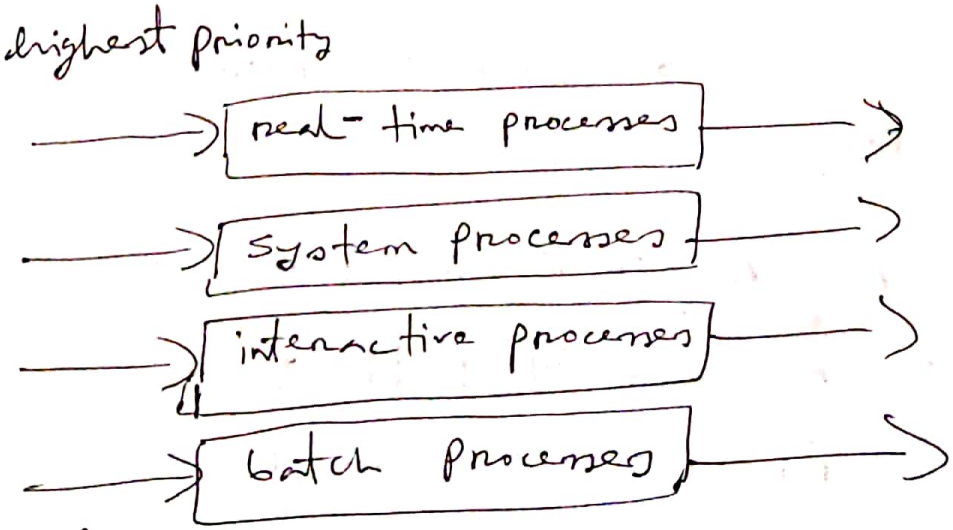
② Hard real time System: task must be serviced by its deadline.

* Event latency: When an event occurs to when it is serviced এর মধ্যকার সময়।

* Interrupt latency: একটি interrupt আসার পর চলমান Process থামিয়ে নতুন কাজ start করে। আসলে যেই start এর মধ্যকার সময়টাই Interrupt latency.

* Dispatch latency: Current process off করে অন্যটার switch করার জন্য যে সময়টা লাগে।

Prioritization based upon process type:



2] # Banker's Algorithm

(11) AL = Allocation

A = Available

A = 2 1 3 3

P_2 = 2 1 2 2

A = 0 0 1 1

1) need : Max − Allocation

P_0	3	2	2	1
P_1	3	6	4	4
P_2	2	1	2	2
P_3	1	1	1	4

No need can be satisfied. Return all resources by P_2

A = A + AL = 2 1 3 3 + 0 0 3 4

So, A = 2 1 6 7

P_3 = 1 1 1 4

A = 1 0 5 3

No need can be satisfied. Return all resources by P_3.

A = A + AL = 2167 + 2352

So, A = 4 4 11 9

P_0 = 3 2 2 1

A = 1 2 9 8

No need can be satisfied. Return all the resources by P_0.

$$A = A + AL = 4\ 4\ 11\ 9 + 4\ 0\ 1\ 3$$
$$= 8\ 4\ 12\ 12$$

The need of P_1 is not possible to match.

So,

The system is not in safe state. There might be deadlock.

Security

#Categories:

(I) Breach of confidentiality
→ Unauthorized reading of data.

(II) Breach of integrity
→ Unauthorized write of data.

(III) Breach of availability
→ Destroy the data.

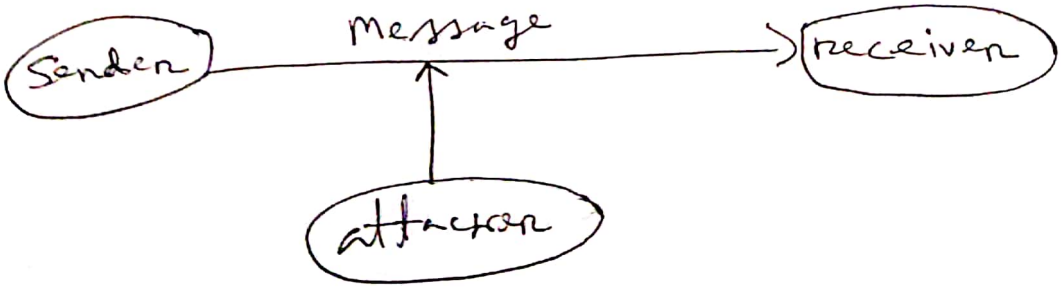
→ CIA model

(IV) Theft of service → (use of resources)

(V) Denial of service (DOS)

#Method:

* man in the middle attack (Mim attack)



attacker: read the message and change it.

Threats:

⊛ Trojan Horse: one type of attack.

→ code এর মধ্যে attack (code read, write)
[change]

⊛ Malware: Software এর মাধ্যমে attack.
One of the powerful attack.

⊛ Trojan horese: Two types.

① Spyware: Webpage এ যে ad গুলো আসে তার মধ্যে কিছু virus যুক্ত ad.

② Ransomware: attacker system এর data কে lock করে then কিছু চাইবে। (মুক্তিপণ)

⊛ Virus dropper: Inserts a virus onto the system.

① file ② boot
③ macro ④ Source code

⊛ Denial of service (DOS):

→ Server এ request করা (limited)।

→ Dos attacker এরকমের অনেক request পাঠাতে থাকে server এ বারবার। Server কে hang করায়।

→ Valid user করা চেষ্টা করবে 'Server not found' (error message)

⊛ Distributed DOS: Dos এ একটা attacker থাকে। কিন্তু DDOS এ একের অধিক attacker একসাথে attack করে।

→ More powerful than DOS.

Synchronization Tools

Memory Barriers :

① Strongly ordered : One processor's memory changes is immediately visible to all. others.

② Weakly ordered : Not be immediately visible to all other processors.

Hardware Instructions

* test_and_set :

```
boolean test_and_set (boolean *target)
{
    boolean rv = *target;
    *target = true;
    return rv;
}
```

* Compare_and_swap :

```
int compare_and_swap (int *value, int expected,
                      int new_value)
{
    int temp = *value;
    if (*value == expected)
        *value = new_value;
    return temp;
}
```

Atomic variables : Uninterruptible.

* Use increment ( ).

mutex lock

* Use acquire ( ) and release ( ).

```
*  while (true) {
       acquire lock
       critical section
       release lock
       remainder section
   }
```

Semaphore Usage

① Counting semaphore: greater than 1 (value).

② Binary semaphore: 0 → 1 range value. [Same as a mutex lock]

* Use wait() and signal().

```
wait (S) {
        while (S<=0)
            S--;
    }
signal (S)
{
   S++;
}
```

P1: S1;
    signal (Synch);

P2: wait (Synch);
    S2;

//Synch initialized to 0.

Protection

* One protection model; computer consists of a collection of objects, hardware / Software.

* Each object has unique name, set of operations.

* Protection means permission.

* Objects are 2 types:

① Hardware objects: devices (printer)

② Software objects: such as files, code.

Domain structure

Access-right = <object name; rights-set>

D_1

<O_3, {read, write}>

<O_1, {execute}>

* Domain = set of access rights.

Access Matrix

View protection as a matrix.

Rows represent domains.

Columns " objects.

object / domain	f_1	f_2	f_3	printer
D_1	read		read	
D_2				print
D_3		read	~~execute~~ execute	

Deadlocks

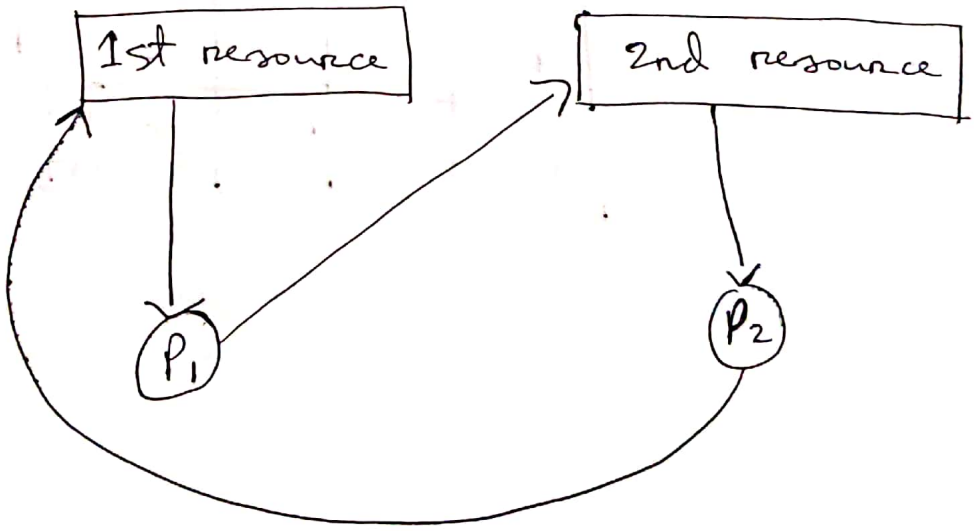
Resource type → $R_1, R_2, R_3 \ldots R_n$.

Process → $P_1, P_2, P_3 \ldots P_n$.

Process utilizes a resource as follows:

1) request (ii) use (iii) release

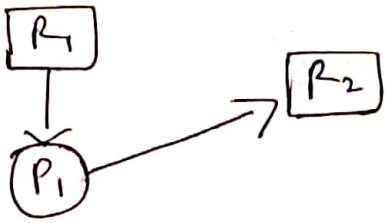
Deadlock with a resource allocation graph:



Deadlock characterization :

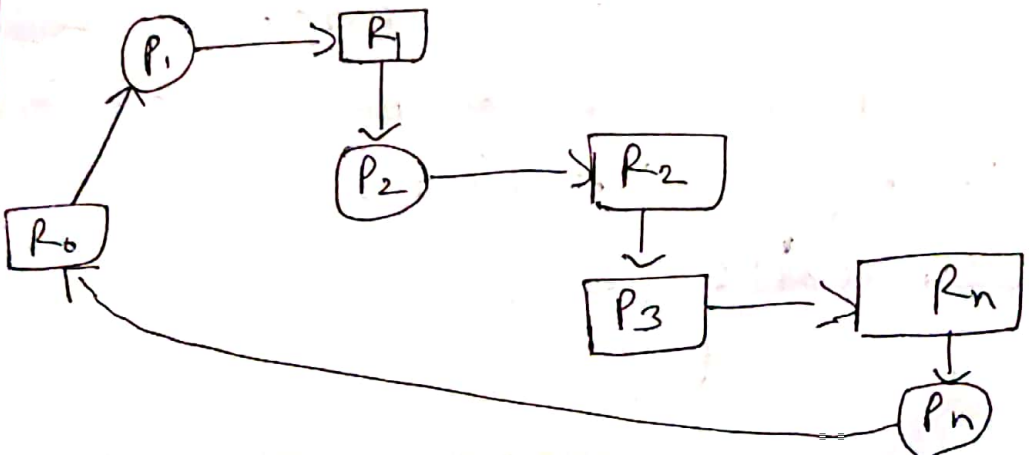
① Mutual exclusion : one process can use one resource at a time.

② Hold & wait : one process is holding one resource & is going to request one resource also.



③ No preemption : When process has completed it's task, then a resource can be released.

④ Circular wait :



Methods for handling deadlocks :

Deadlock prevention —> allow the system to enter a deadlock state and then recover.

Deadlock avoidance

└—> Ignore the problem and never enter unsafe state.

1) Deadlock prevention :

i) Mutual exclusion : not required for sharable resources ; must hold for non-sharable resource

ii) Hold & wait : Whenever a process is going to request a resource, it does not hold any resource in that time.

iii) No preemption : Process resource ~~ক~~ request ~~করবে~~ ~~তখনই~~ allocated resource will be released. Process will be restarted only when it regains its old resources.

iv) Circular wait : Two processes wait for each other to retreat.

2) Deadlock avoidance :

Safe state : Process requests a resource —>

i) resource is free.

ii) resource is busy, but must be released (free).

* P_i needs are not immediately available, then P_i can wait until all P_j have finished.
* P_j is finished then P_i use that resource.

Basic facts :

i) If a system is in safe state;
ii) If a graph contains no cycle;
} -> no deadlocks

i) System is in unsafe state.
ii) Graph contains cycle.
} -> possibility of deadlock

Avoidance algorithms

i) Single instance of a resource type
→ Use a resource allocation graph.

ii) Multiple instances of a resource type
→ Use the banker's algorithm.

Banker's algorithm

n = number of processes

m = number of resources

Available: available instances of a resource.

Max: Process Pi may request at most k instances of resource type Rj.

Allocation: Process Pi is holding k instances of resource Rj.

Need: Max − Allocation

10 − 7

= 3 = need [Process's need]

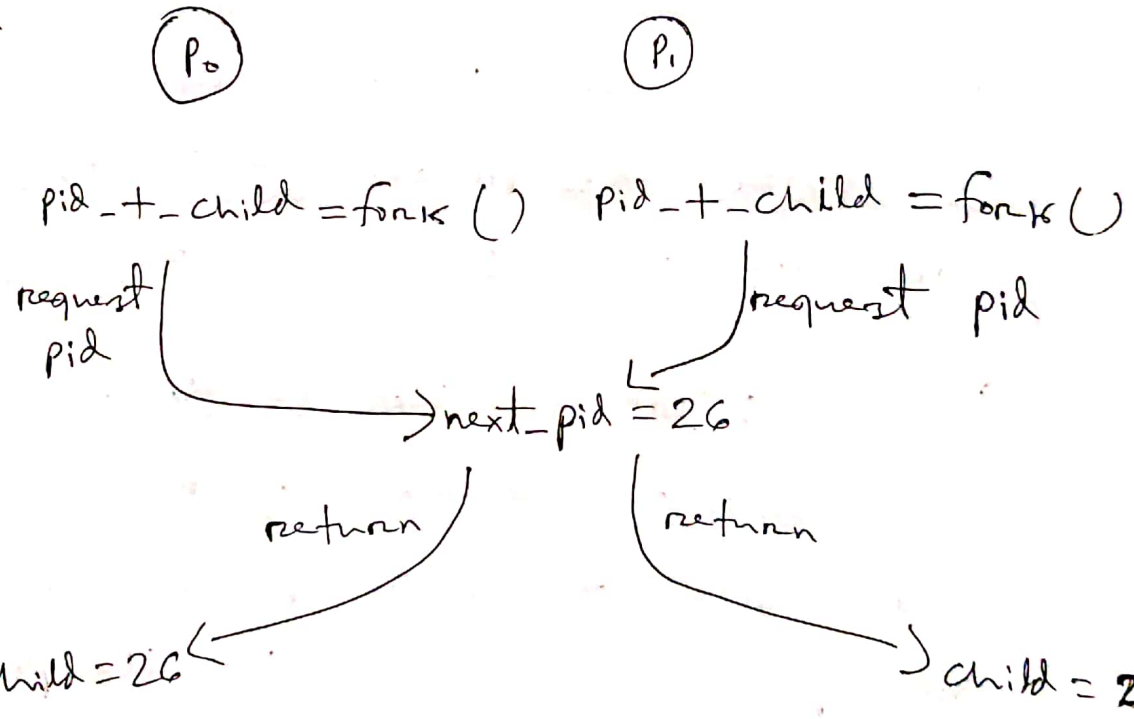
Synchronization tools

Race Condition

=> When device on a system attempts to perform two or more operations at the same time.

=> It occurs two or more threads can access shared data and they try to change it at the same time.

*



*Unless there is mutual exclusion, the same pid could be assigned to two different processes.

Critical section Problem

* Critical section problem is to design protocol to solve the process's problem.

* Process must ask permission to enter critical section in entry section, may follow critical section to exit section, then remainder section.

Solution to critical section problem :

(i) Mutual exclusion : Process critical section এ থাকলে, অন্য Process ঢুকতে পারবে না critical section এ। (critical sec → one process at a time)

(ii) Progress : যদি critical section টি free থাকে, at the same time কিছু Process entry করতে চাচ্ছে তখন OS must be select one process and permit the process to enter.

(iii) Bounded waiting : Process permission চাওয়ার পর OS check করে দেখার জন্য কে অপেক্ষা নেয়।

Peterson's Solution

* Modern architectures এ কাজ করার guaranty নাই।
* Two process solution
  -> int turn ;
  -> boolean flag [ ] ;

- The variable turn indicates whose turn it is.
- flag array is used to indicate if a process is ready or not to enter the critical section.

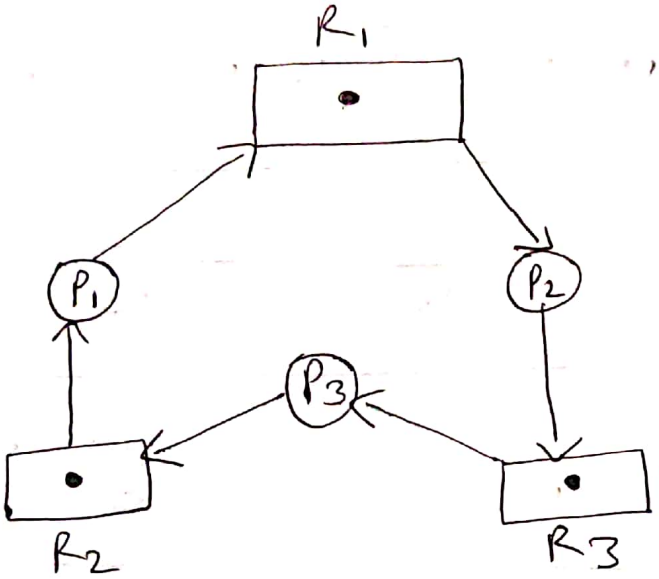
* For single threaded this is ok.
* For multithreaded the reordering may produce inconsistent or unexpected results.

Name: Jubayer Ahamed

ID: 18-36325-1

Ans. to the Ques. no. 1

① Resource allocation graph with deadlock:



Without deadlock graph / No deadlock: