Operating System

বঙ্গবন্ধু শেখ মুজিবুর রহমান বিজ্ঞান ও প্রযুক্তি বিশ্ববিদ্যালয়, গোপালগঞ্জ

নম্বর - 42109 অতিরিক্ত উত্তরপত্র

স্টুডেন্ট নম্বর ঃ .....

কোর্স নম্বর ঃ .....

কোর্স শিরোনাম ঃ .....

সেকশন ঃ এ / বি সীল ..... ইনভিজিলেটরের স্বাক্ষর

Class 1

What happens when Pc is Powered up :-

1) sends Power to Power supply unit. Send power to the motherboard and other components
2) Pc performs a post function. It checks for hardware failure.
3) Pc shows details about the boot process. on the the monitor
4) BIOS confirms that. there is a bootstarp loader. it loads the boot loader into RAM.
5) BIOS handover it's work to boot loader which in turns loads OS into memory.
6) Boot loader handover it's responsibility to OS. The OS is ready for interaction.

OS :-

Operating system is program that control the execution of all kinds of application Programs and acts as an interface between user and computer hardware.

Scanned by CamScanner

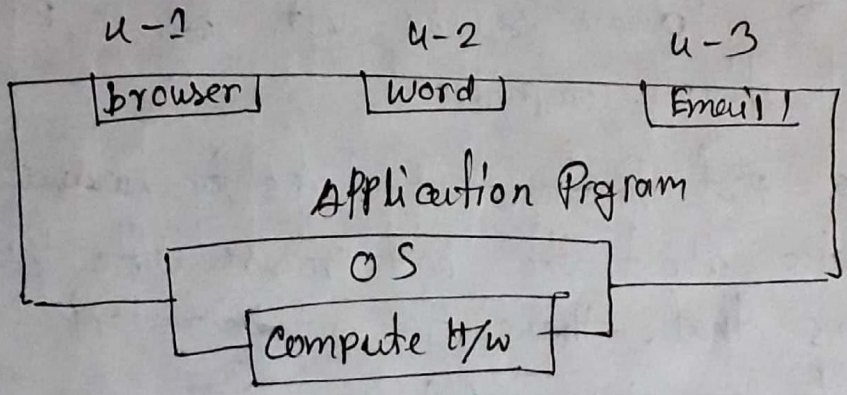
OS objectives :-

- Execute user program and make solving easier
- Make computer system convenient to use.
- use inefficient manner.

Computer System :-

It has 4 component.

- OS
- Application Program
- Computer hard ware
- User



Function of OS :-

① Processor mangment - Putting task into order and paring them before they go to cpu

② Memory mangment, and CO-ordinats data to/from memory and determine the necessity of ~~vir~~ virtual memory

③ Divice manage-ment, Provide interface for connected device

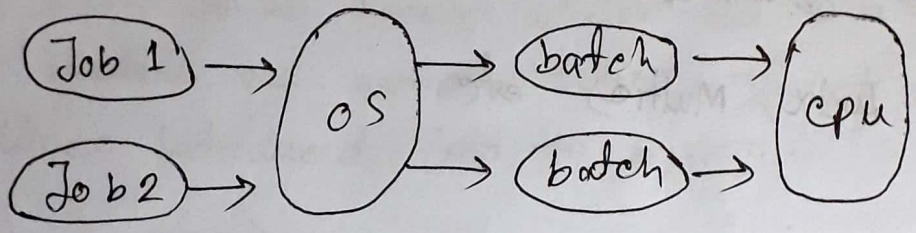
Scanned by CamScanner

④ Resource Allocation — It handles memory ~~allocatio~~ and cpu time, used by various applications or peripheral devices

⑤ security. It also provides security.

Types of OS :-

1. Batch OS:



User doesn't interact with computer directly. which takes similar jobs having same requirments and group them into batches. It is the responsibility of operator of sort the job with similar job

Disadvantage

(i) Through put is less
(ii) Not user interactive
(iii) Job will go in starvation.

Scanned by CamScanner

3. Multitasking

Multiple job are executed by the cpu simultaneously switching between them. Switching occurs so frequently that users can interact with each Program. It also called shold system

Advantage – Increased throughput

Disadvantage – Data Communication Problem

Example – Unix, Multics etc

4. Multi Processor

Several processors that share a single memory. All the processor are in single OS.

Advantage.

Performance high
Reliability increased

Disadvantage

Dependency

Example

Vector Array, Mordern Processor.

বঙ্গবন্ধু শেখ মুজিবুর রহমান বিজ্ঞান ও প্রযুক্তি বিশ্ববিদ্যালয়, গোপালগঞ্জ

নম্বর - 42110 অতিরিক্ত উত্তরপত্র

স্টুডেন্ট নম্বর ঃ .....

কোর্স নম্বর ঃ .....

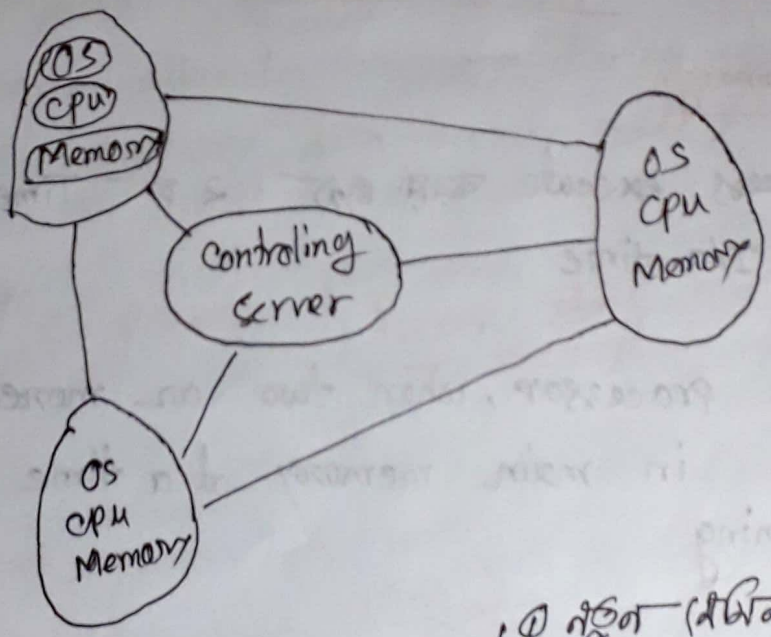
কোর্স শিরোনাম ঃ .....

সেকশন ঃ এ / বি

সীল

ইনভিজিলেটরের স্বাক্ষর

5. Distributed OS:- A set of internally connected ~~separ~~ seperated machines, users can not specify wheter the ~~seve~~ get several from one machine on another machine. If work load increase new machines introduced into the system.



① নতুন মেশিন যুক্ত করা

Advantage: Reliability high
Scalibility ① high

Disadvantage: Expensive
centeralized server Problem

Example : locus.

Scanned by CamScanner

6. Real time OS:

Real time operating system is a data processing system the time interal it takes to respond to it's input is so small that it control the environment.

Example: Industrial control system, Medical imageing system, Scientific system

Disadvantage: Expensive.

Process Management

2. Multi programming

একটা Process execute করার জন্য ২ টা Time প্রয়োজন

① CPU ② I/O time

Sharing the processor, when two or more programs are reside in main memory at a time it called multiprogramming

Advantage: CPU utilization is maximized

Disadvantage: CPU scheduling is required

Example: Computer running excel
printing doc etc at a time.

Process
Process

Text
the
Data
Heap
Stack
Para

Attr

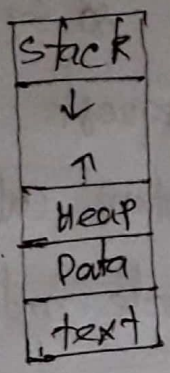
Proc
Prog
Proc
Pri
Cre
Li
}

Scanned by CamScanner

Process Management

Process: When a Program is in a execution.

Process structure:



Text Represent the current activity by the value of the instruction pointer

Data Contains static & gobal variable

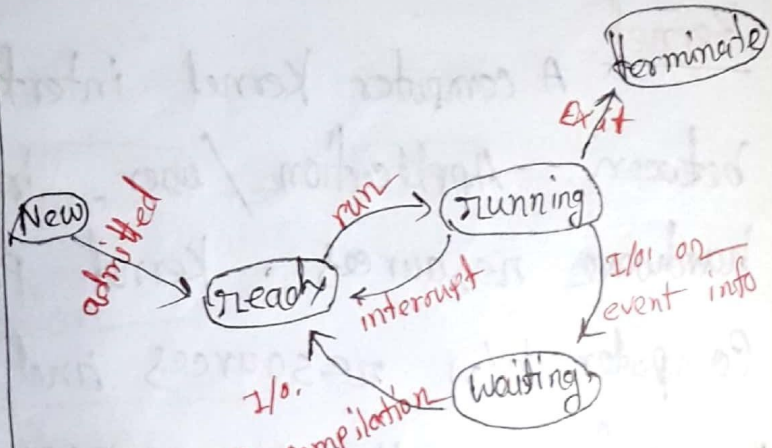
Heap Dinamically allocates memory during its runtime

stack Contains temporary data, return address, function Parameter etc.

Attributes of a Process

Process id,
Program counter,
Process state,
Priority,
General purpose registor,
List of open file
List of open device

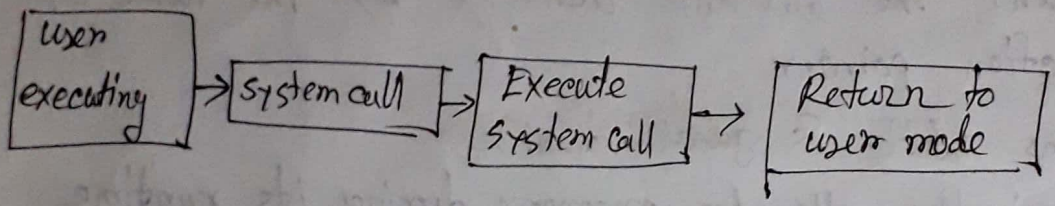
Process state



Scanned by CamScanner

Computer System operating mode / Dual mode Operation

user mode :- While writing a text document or using any application program then system is in user mode. when user application request for a service or interrupt occurs or system call. Then OS switch from user mode to kernel mode. and mode bit is 0.



Kernel Mode :- When system boots then h/w in kernel mode and it starts user application in user mode.

Kernel A computer kernel interface and providing services between Application / user interface and cpu, memory hardware resources. Kernel provide and manage Computer h/w resources and allowing Programs to run and use these resources.

System call is a Programatic way in which a computer Program request for a service from the kernel of an OS.

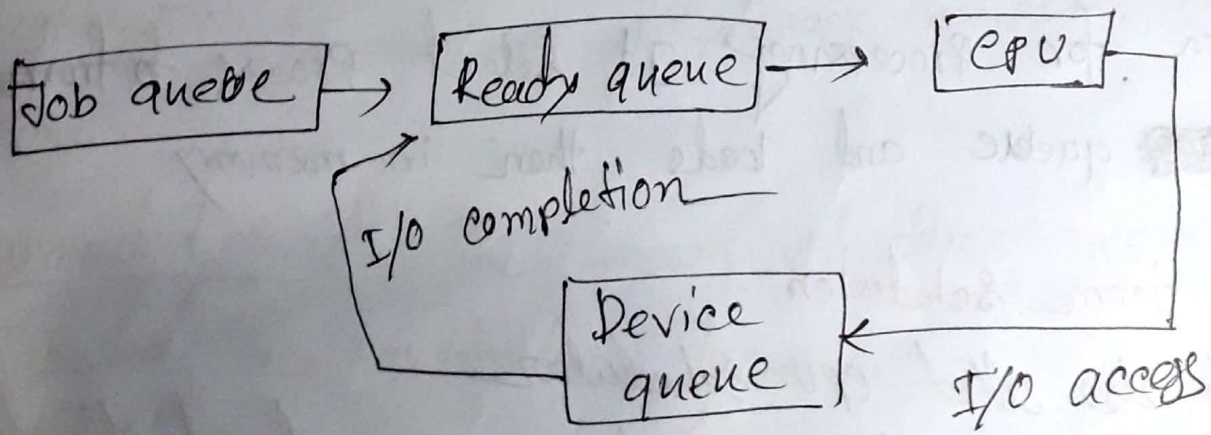
Scanned by CamScanner

Process Scheduling

Process scheduling is an activity of process manager that handls the removal of running process from cpu and selection of another process based in a particular strategy.

- Non Premtive
- Premtive

Process Scheduling queue



Scanned by CamScanner

Job queue :- keeps all process in the system

Ready queue :- keeps all the process in main memory ready and waiting to execute

Device queue :- Process which are blocked unavailability of I/O device.

Schedulers special system software which handles the Process scheduling in various way. Main task is to select the job that is submitted to system and to decide which process to run

1. long term scheduler
2. short " "
3. Mid " "

Long term scheduler It is also called Job schedulers. determines which Program are admitted ~~to~~ into the system for Processing. It select Process ~~fo~~ from Job ~~queue~~ queue and loads then in memory

Short term scheduler

→ It is also called CPU scheduler.

→ Select process from Ready queue at and allocate CPU to one of them

→ It is also takes decision which process to executed next.
→ It is also known as dispatcher.

(iii) Mid-term Scheduler

A running process may be suspended if it makes an I/O request. All of this process are handled mid term Scheduler

CPU Scheduling

CPU Scheduling means one process to use the cpu while execution of another is on hold.

Criteria

1) CPU utilization : To make the best use of cpu and not waste any CPU cycle

2) Throughput : The numbers of process completed per unit time

3) Turnaround Time :- The amount of time required to executed a particular time.

Scanned by CamScanner

waiting time Total time the process has to wait before it execution begins. শুরু করতে কত সময় বসে থাকতে হয়।

~~Response time~~

Brust time The amount of cpu time that is required by a process to finish its task. কত সময় CPU তে থাকতে হয়

completion time The time at which a process finishes its task. কখন শেষ হয়

Completion time − Arrival time = Brust time + waiting time
(Trunaround time)

বঙ্গবন্ধু শেখ মুজিবুর রহমান বিজ্ঞান ও প্রযুক্তি বিশ্ববিদ্যালয়, গোপালগঞ্জ

নম্বর - 42112 অতিরিক্ত উত্তরপত্র

স্টুডেন্ট নম্বর : .....

কোর্স নম্বর : .....

কোর্স শিরোনাম : .....

সেকশন : এ / বি

সীল

ইনভিজিলেটরের স্বাক্ষর

Algorithm's

① FCFS (Queue Data structure)
- Non- Pre-emptive
- Batch operating System.

* যে প্রথমে আসবে সে প্রথমে কাজ করবে।

Q. Provide

P-id	A-T	B-T	C-T	T-T	W-T
P_1	0	2	2	2	0
P_2	1	3	5	4	1
P_3	2	5	10	8	3
P_4	3	4	14	11	7
P_5	4	6	20	16	10

Gantt chart

$0 \xrightarrow{P_1} 2 \xrightarrow{P_2} 5 \xrightarrow{P_3} 10 \xrightarrow{P_4} 14 \xrightarrow{P_5} 20$

Convoy effect :- convoy effect is a situation when one Process needs to use a resources for a short span of time but this process is blocked by another process who is holding that resources for a long period of time.

Scanned by CamScanner

Example [FCFS, Non-Pre-emtive]

P-id	A-T	B-T	C-T	T-T	W-T
P_1	0	2	2	2	0
P_2	3	1	4	1	0
P_3	5	6	11	6	0

$0 \xrightarrow{P_1} 2$ (1 idel) $3 \xrightarrow{P_2} 4$ (1 idel) $5 \xrightarrow{P_3} 11$

2. Shortest Job First

→ Depend on shortest brust time

⇒ Pre-emtive (Multitasking)

⇒ Non-pre-emtive (Batch)

[যার Brust time সব কম সে সবার আগে]

Example (Non-pre-emtive) SJF

P-id	A.T	B.T	C.T	T.T	W.T
P_1	1	7	8	7	0
P_2	2	5	16	14	9
P_3	3	1	9	6	5
P_4	4	2	11	7	5
P_5	5	8	24	19	11

$1 \xrightarrow{P_1} 8 \xrightarrow{P_3} 9 \xrightarrow{P_4} 11 \xrightarrow{P_2} 16 \xrightarrow{P_5} 24$

Scanned by CamScanner

Example [SJF / Shortest Remaing Time First (SRTF)]

Pre-emtive

P.id	A.T	B.T	C.T	T.T	W.T
P_1	0	6	7	7	1
P_2	2	1	3	1	0
P_3	4	4	14	10	6
P_4	5	3	10	5	2

$0\ \overset{P_1}{—}\ 2\ \overset{P_2}{—}\ 3\ \overset{P_1}{—}\ 7\ \overset{P_4}{—}\ 10\ \overset{P_3}{—}\ 14$

Example | SJF | Pre-emtive |

P-id	A.T	B.T	C.T	T.T	W.T
P_1	2	6	15	13	7
P_2	5	2	7	2	0
P_3	1	8	23	22	14
P_4	0	3	3	3	0
P_5	4	4	10	6	2

$0\ \overset{P_4}{—}\ 3\ \overset{P_1}{—}\ 4\ \overset{P_5}{—}\ 5\ \overset{P_2}{—}\ 7\ \overset{P_5}{—}\ 10\ \overset{P_1}{—}\ 15\ \overset{P_3}{—}\ 23$

Scanned by CamScanner

3. Priority Scheduling

i) Priority is assigned for each process

ii) Priority with lower priority number is considered at highest priority process.

iii) Same priority process will be ~~considered~~ executed based on process id.

Example (Non Pre-emptive)

P-id	B-T	Priority	C.T	T.T	W.T
P_1	10	3	16	16	6
P_2	1	1	1	1	0
P_3	2	4	18	18	16
P_4	1	5	~~19~~ 19	~~19~~ 19	~~18~~ 18
P_5	5	2	6	6	1

0 — P_2 — 1 — P_5 — 6 — P_1 — 16 — P_3 — 18 — P_4 — ~~19~~ 19

Note: A-T যদি দেওয়া না থাকলে A-T = 0 for all process

স্টুডেন্ট নম্বর
:কোর্স নম্বর
:কোর্স শিরোনাম
:সকশন

Examp

P-id

P_1
P_2
P_3
P_4

0 — P_1

4. R

1) A
qu

ii)

Scanned by CamScanner

Example [Pre-emptive]

P-Id	A-T	B-T	Priority	CT	T-T	W-T
P_1	0	5	2	13	13	8
P_2	4	8	1	12	8	0
P_3	6	2	4	21	15	13
P_4	8	6	3	19	4	5

$0 \xrightarrow{P_1} 4 \xrightarrow{P_2} 12 \xrightarrow{P_1} 13 \xrightarrow{P_4} 19 \xrightarrow{P_3} 21$

4. Round - Robin Scheduling

i) A fixed time is allocated for each process is called quantum time.

ii) Pre-emptive Algo —

Scanned by CamScanner

Example T.q = 5

P-id	B.T	C.T	T.T	W.T
P_1	21	32	32	4
P_2	3	8	8	5
P_3	6	21	21	15
P_4	2	15	15	13

0 —P_1— 5 —P_2— 8 —P_3— 13 —P_4— 15 —P_1— 20 —P_3— 21 —P_1— 32

Example T.q = 3

P-id	A.T	B.T	C.T.	T.T	W.T
P_1	0	9	12	12 12	8 8
P_2	2	5	14	12 12	7
P_3	4	3	9	5 5	2 2
P_4	5	2	11	6	4 4

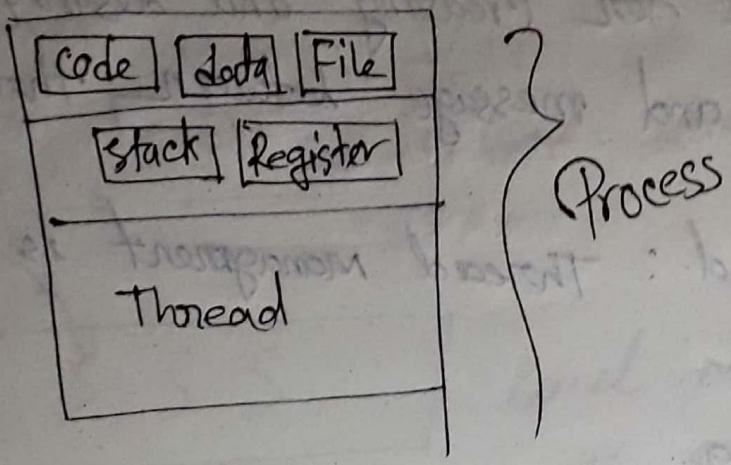
0 —P_1— 3 —P_2— 6 —P_3— 9 —P_4— 11 —P_1— 12 —P_2— 14

Scanned by CamScanner

Process Synchronization :-

Thread :

Thread is an execution unit which consist of its own Program Counter, stack, register, etc.



Difference between thread and Process

Process	Thread
① Process is heavy weight	① Thread is light weight
② Process switching needs interaction of OS	② Thread need does not need interact with OS.
③ Each power allocate different memory location	③ Each thread under same process belongs to the same memory location
④ If one process is blocked no other process can be executed until it is unblocked.	④ If one thread is blocked another thread can be run.

Scanned by CamScanner

ample T.9=5

Types of thread

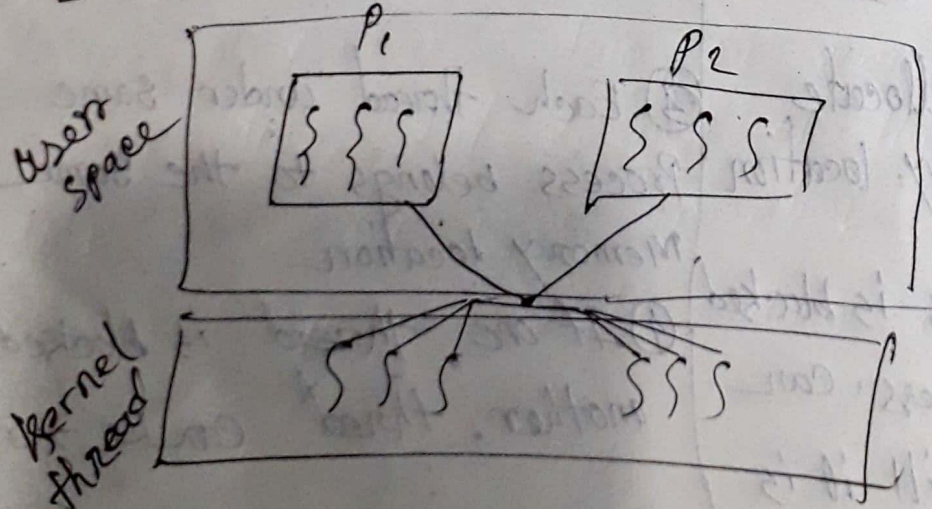
① User thread: Application Program — user create and use in their Programs. Thread library contains code for creating and destroying threads, passing data and message between threads.

② Kernel thread: Thread management is done by kernel.

③ Multithreading

Multithreading is the ability of executing multiple threads at a same time.

- Many to many: multiple number of threads user level threads to smaller or equal kernel ~~node~~ thread.



Process

(11) Ma

acces

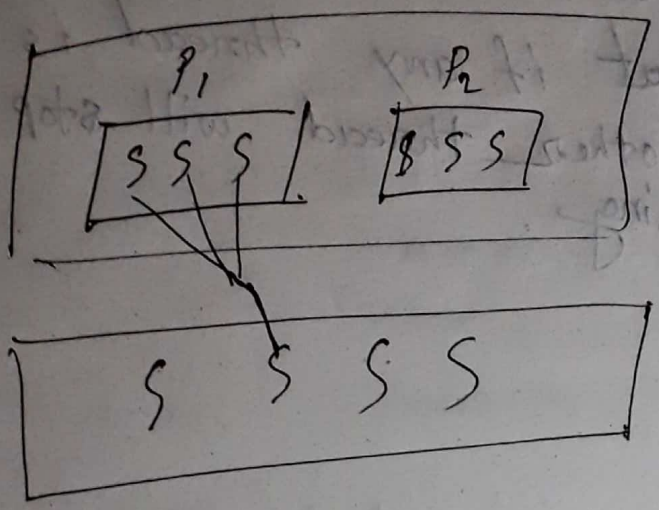
three

(111) O

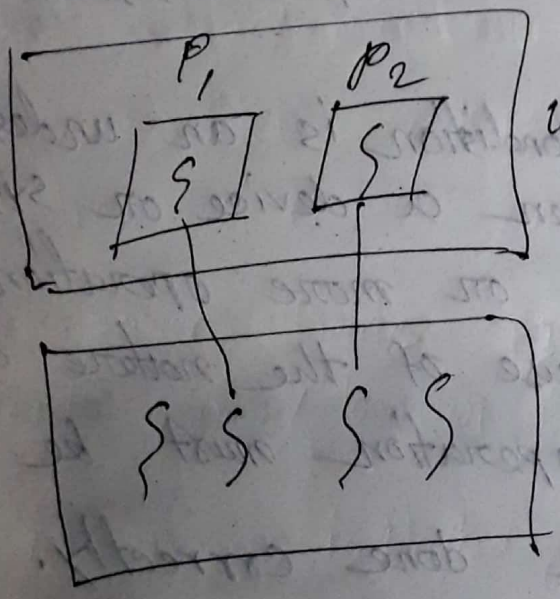
to

Scanned by CamScanner

(ii) Many to one :- In this only one threads can access the kernel at a time, So multiple threads are unable to run parallel.



(iii) one to one :- it supports multiple threads to execute in parallel.



Scanned by CamScanner

xample T.q=5

Benefits of multithreading

- Resource sharing (memory, data, code, files etc.)
- Responsiveness ⇒ it provides continuous response that if any thread is block no other thread will stop working.
- Economy.

Process Synchronization

It means sharing system resources by process in such a way that con-current execution of shared data doesnot create any inconsistent result

Race-Condition:- Race condition is an undesirable situation that occurs when a device or system attempts to perform two or more operation at the same time, but because of the nature of the system or device, the operation must be done in proper sequence to be done correctly.

Scanned by CamScanner

Process Synchronization :-

① Co-operative ——— Share : variable, Memory, Code, Resources (CPU, Printer, Scanner)

② Independent

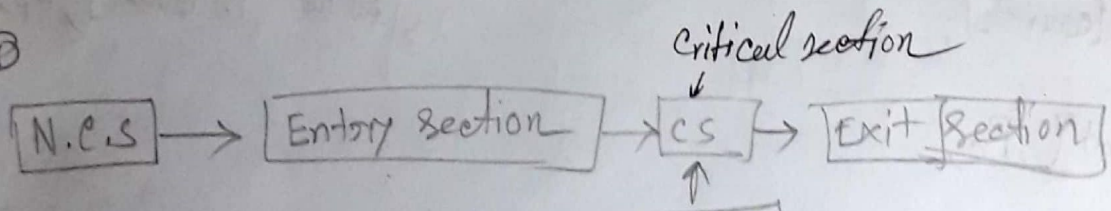
Example

```
int shared = 5;

P1                          P2
int x = shared;             int y = shared;
x++;                        y--;
sleep (1);                  sleep (1);
shared = x;                 shared = y;
```

* Context switching
* Race Condition

* **Critical Section** — it is Part of the Program where Shared resources are accessed by various Process.

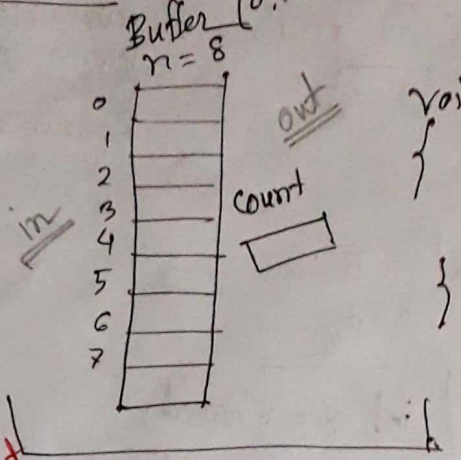


4 Condition for active synchronization

1) Mutual exclusion } Primary
2) Progress
3) Bounded waiting
4) No assumption related to H/W speed. } Secondary

Scanned by CamScanner

Producer, Consumer Problem




```
void Consumer (void)
{
  int itemc;
  while (true)
  {
    while (count == 0);
      itemC = Buffer (out);
      out = (out + 1) mod n;
      Count = Count - 1;
    Process. item (itemc);
  }
}
```

```
void Producer ( )
{
  int itemp;
  while (true)
  {
    Producer. item (itemp);
    while (count == n);
      Buffer [in] = itemp
      in = (in + 1) mod n;
    Count = Count + 1;
  }
}
```

```
int Count = 0;
```

```
X1 | Load Rc m[count];
   | DEC Rc;
   | Store m[count], Rc;
```

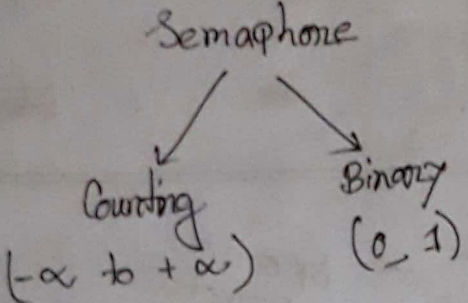
```
I1 | Load Rp, m[count];
I2 | INCR Rp;
I3 | Store m[count], Rp;
```

Worst case (Problem face)

Producer $I_1 I_2$ Consumer $I_1 I_2$ Producer I_3 Consumer I_3

Scanned by CamScanner

Semaphore is an integer variable which is used in mutual exclusive manner by various Concurrent Co-operative Process in order to achive Synchronigation




```
Down (Semaphore S) {
S.value = S.value - 1;
  if (S.value < 0)
   {
   Put Process (PCB) in
   Suspended list,
   sleep ();
   }
  else
   return;
}
```

Process control block

```
up (Semaphore S)
{
  S.value = S.value + 1;
if (S.value ≤ 0)
  { select a process from suspended
   list,
   wake up ();
  }
}
```

P_1, P_2, P_3

Entry section — P(), Down, wait

critical section — V(), up, signal/post/release

Exit

1. $S = 10$, $P = 6$, $V = 4$. $S_f = ?$

Down, up

$S_f = 10 - 6 + 4$

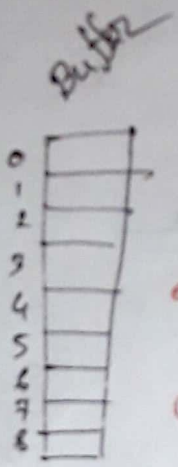
$= 8$

Solution of Producer & Consumer Problem using Semaphore

Counting Semaphore → full = 0 (No. of filled slots).

→ empty = N (No. of empty slots).

Binary Semaphore S = 1

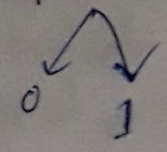



```
Producer item (item p)
{
entry | down (empty);
      | down (S);
cs    | Buffer[IN] = item p;
      | IN = (IN+1) mod n;
exit  | up (S);
      | up (full);
}
```

```
Consumer
entry | down (full);
      | down (S)
cs    | item c = Buffer[out];
      | out = (out+1) mod n;
exit  | up (S);
      | down (empty);
```

Scanned by CamScanner

Binary Semaphore



starting
* S.value must (1)

```
Down (Semaphore S)
{ if (value == 1)
    s.value = 0;
  else {
  Block this process &
  Place in suspended list
  Sleep ();
  }
}
```

```
up (Semaphore S)
 { if (suspended list is
     empty)
       s.value = 1;
  else {
    Select a process
    from suspended list
    and wake up ()
    }
}
```

Reader - writer Problem

*(Same data)

1. R - W → Problem
2. W - R → ,,
3. W - W → Problem
4. R - R → No Problem

```
void writer ()
{ while (1)
  {
  ~~data~~ (d
  down (db)
  DB
  up (db)
  }
}
```

```
int rc = 0;          ← Read count
Semaphore mutex = 1,
semaphore db = 1,
Void Reader (vo )
 { while (1)
   {
   down (mutex);
    rc = rc + 1;
   if (rc == 1) then down (db);
   up (mutex):
```

Entry

CS — DB

```
down (mutex)
  rc = rc - 1;
if (rc == 0) then up (db)
  up (mutex)
  Process data. }}
```

Exit

স্টুডেন্ট নম্বর
কার্স নম্বর
কার্স শিরোনাম
সকশন

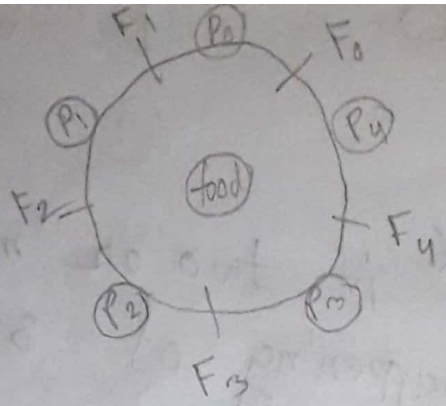
Dining

```
Void
{ wh
  { fl
wait (fo
wait (fo
Signal (P
Signal (P
  }
}
```

Solve

Scanned by CamScanner

Dining philosophers problem




```
void philosopher()
{ while(1)
  { throught()
wait (take_fork(i))
wait (take_fork((i+1)%N))
    EAT()
Signal (put_fork(i))
Signal (put_fork((i+1)%N))
  }
}
```

S_0, S_1, S_2, S_3, S_4

Entry

CS:

Exit

** যেই সাথে দুটা Process CS এ প্রবেশ করতে পারবে

⊛ Deadlock

f_1 যখন philosophers left_fork নেবার পর preempt pre-empt হলে Deadlock হয়ে যাবে। কারণ সকল Semaphore এর value 0 হয়ে যায় তখন কেউ খাবার গ্রহণ করতে পারবে না,

Solve: যখন একটু অপেক্ষা করবে → right_fork নিবে

Scanned by CamScanner

Q. Practice

What is the maximum number of Process that may present in cs at any part of time ?

Ans:-

for Pi (i=1 to 9)	for P10	P10
P ( )	V ( )	V ( )
cs	cs	cs
V ( )	V ( )	P ( )

of course 3

Ans: 10

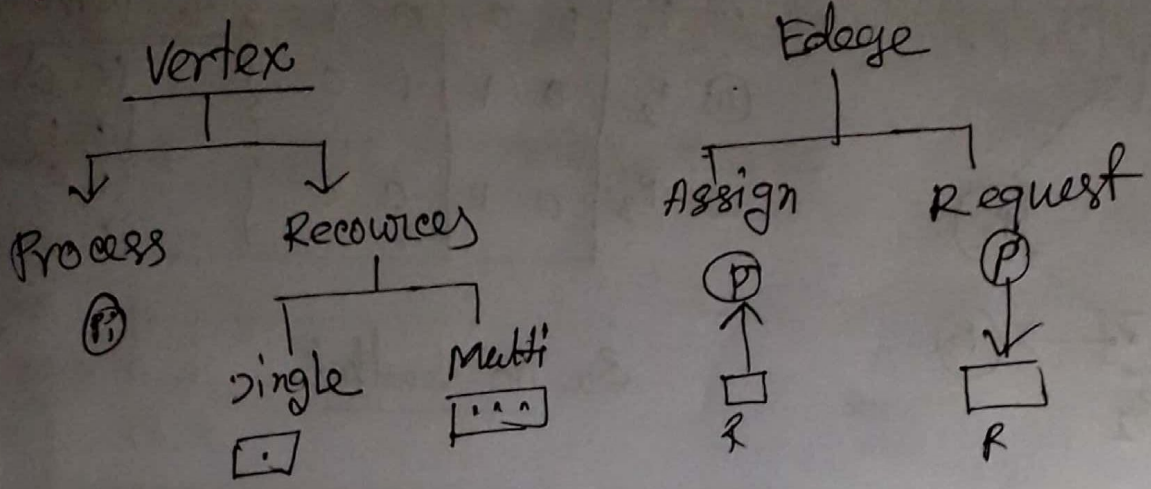
Deadlock if two or more process are waitting on happening of some event, which never happens, then we say these process can involed in deadlock then the state is known as deadlock.

4 - Condition

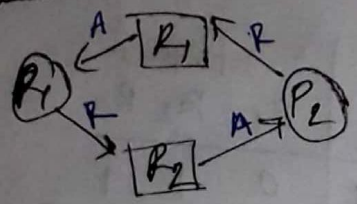
1. mutual - exclusion
2. No Premption
3. Hold & wait
4. Circular wait.

Scanned by CamScanner

Resource Allocation Graph (RAG)



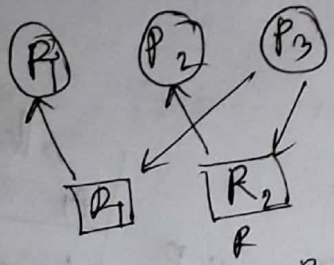
Single instance



	A		R	
	R_1	R_2	R_1	R_2
P_1	1	0	0	1
P_2	0	1	1	0

Availability = (0,0)

Deadlock



	A			R		
	R_1	R_2	R_3	R_1	R_2	R_3
P_1	1	0		0	0	
P_2	0	1		0	0	
P_3	0	0		1	1	

Availability (0,0)

Not deadlock

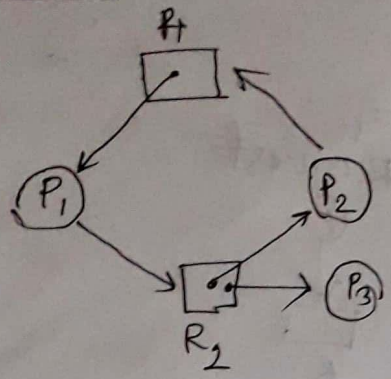
Waiting
- finite (starvation)
- Infinite (deadlock)

SI && CW ⇒ Deadlock (true)

MI && CW ⇒ Deadlock (false)

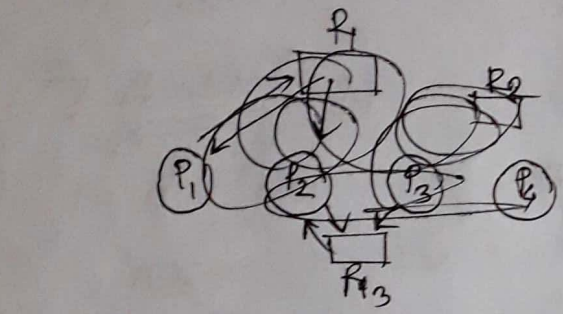
Scanned by CamScanner

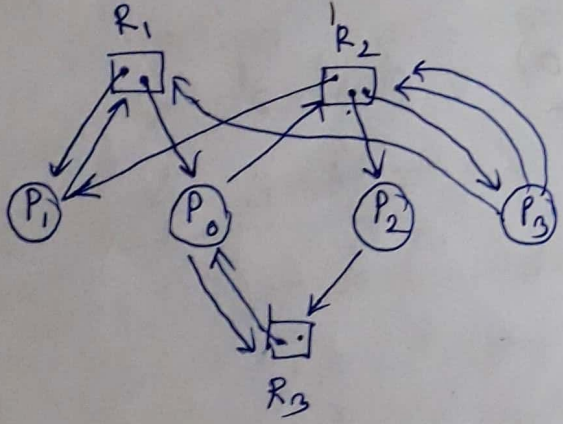
Multi-instance



		Allocate		Request		Current Availability
		R1	R2	R1	R2	
(1)	P1	1	0	0	1	(0,0)
(2)	P2	0	1	1	0	(0,1)
(3)	P3	0	1	0	0	(1,0)

So, No deadlock





		Allocation			Request		
		R1	R2	R3	R1	R2	R3
(2)	P0	1	0	1	0	1	1
(3)	P1	1	1	0	1	0	0
(1)	P2	0	1	0	0	0	1
(4)	P3	0	1	0	1	2	0

Current Availability

R1	R2	R3
(0,	0,	1)
0	1	0
1	4	2
2	2	2
2	3	2

No deadlock

Scanned by CamScanner

Deadlock Handling

four method

- Deadlock ignorance (ostrich method) windows, Linux
- " Prention
- " Avoidance (Banker's Algo)
- " detection & Recovery.

Banker Algo

Process	Allocation E F G2	Max E F G2	Available E F G2	Remain E F G2
P_0	1 0 1	4 3 1	3 3 0	3 3 0
P_1	1 1 2	2 1 4	4 3 1	1 0 2
P_2	1 0 3	1 3 3	5 3 4	0 3 0
P_3	2 0 0	5 4 1	6 4 6	3 4 1
			8 4 6	

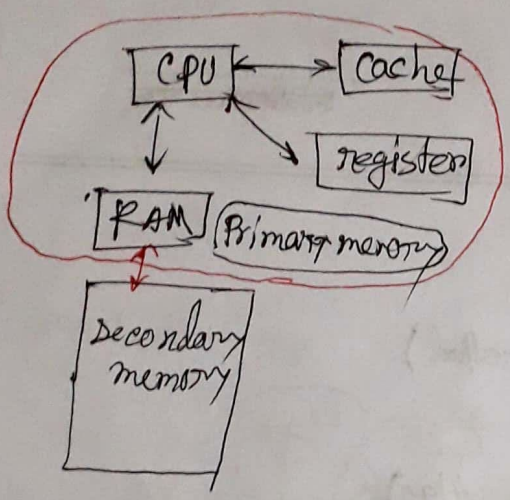
E = 8 +
F = 4
G2 = 6

P_0
P_2
P_1
P_3

→ Safe Sequence
→ state " $P_0 \rightarrow P_2 \rightarrow P_1 \rightarrow$

Scanned by CamScanner

Memory Management → method of managing
Primary memory
Goal: Efficient utilization of memory



→ Degree of multiprogramming
Ready state ত Process যত বেশি

P_n ← No. of Process

ধরি k^{P_n} is I/O operation

CPU utilization $(1 - k^{P_n})$

→ RAM size বাড়লে Degree বাড়ে

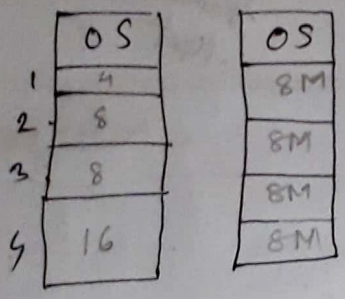
Memory Management techniques

- Contiguous
  - Fixed (static)
  - variable (dynamic)
- Non Contiguous
  - Paging
  - Multilevel paging
  - Inverted "
  - Segmentation
  - Segmented paging.

Scanned by CamScanner

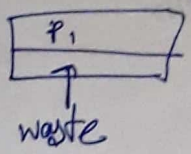
Fixed Partitioning (Static Partition)

- No. of partition are fixed
- size of each partition may or may not same.
- Contiguous allocation so & paging not allowed.



i) Internal fragmentation

$P_1 = 2MB$



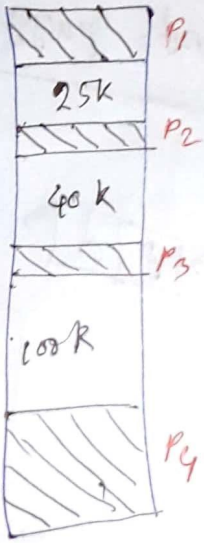
ii) limit in process size.

iii) limitation on degree of multiprogramming

First-Fit: Allocate the first hole that is big enough

Next-fit: Same as first-fit but start search always from last allocate full.

Best-fit: Allocate the smallest hole that is by Prath



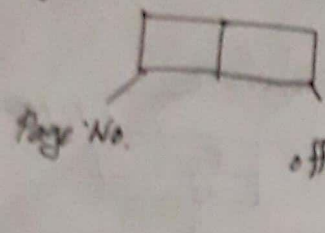
worst-fit: Allocate the largest hole

Scanned by CamScanner

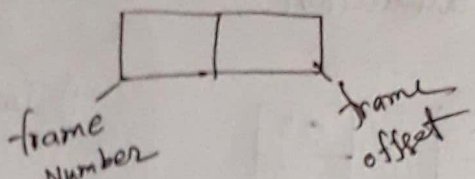
Paging

frame size == Page size

logical address



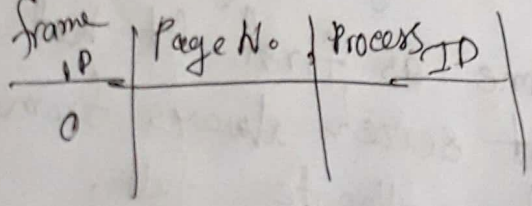
Physical Address

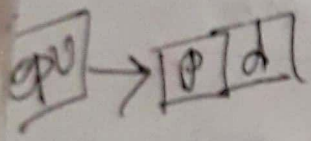


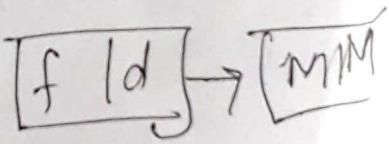
Non-Contiguous Memory Allocation

* Each process has its own page table.
* Page table will be in Main Memory.

Inverd paging







* Linear search
* Searching time program

Scanned by CamScanner

Virtual memory

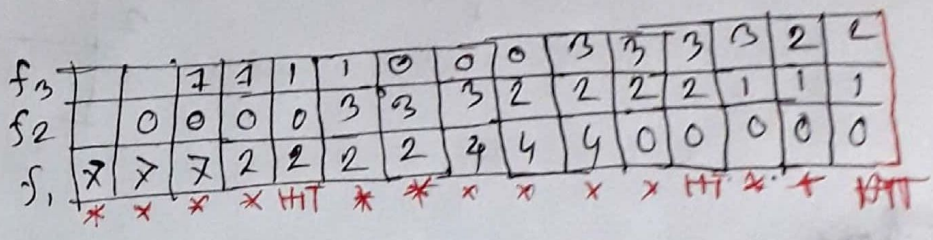
Page replacement algo
- FIFO
- optimal page
- Least Recently

Page fault

CPU যখন Process কে খুঁজে
পেল কিন্তু RAM তে নেই
তখন Page fault বলে।

* FIFO (Page replace)

string : 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 1, 2, 0



~~Hit~~ = 3

Faults = 12

Hit Ratio = $3/15 \times 100$

Mis Ratio = $\frac{12}{15} \times 100$

Belady's Anomaly : frame number বাড়ানোর ফলে
Hits কমে যায়,

String : 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

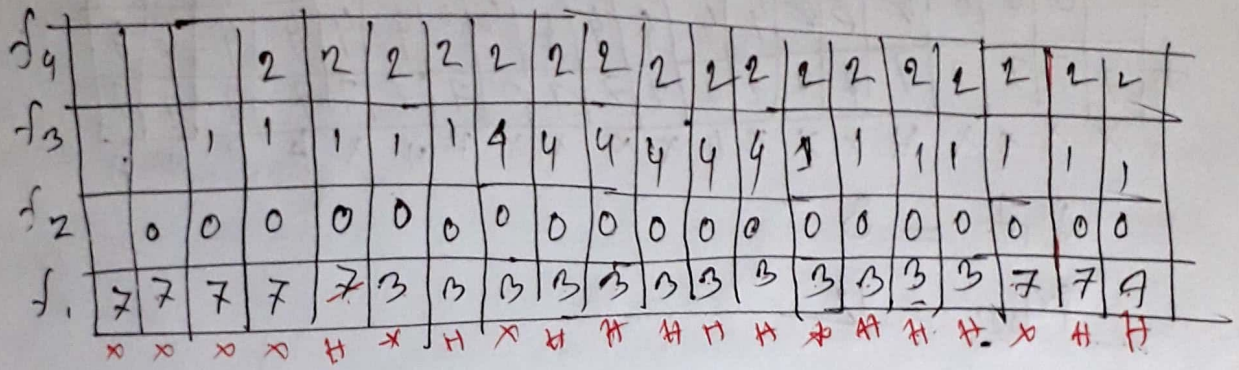
Scanned by CamScanner

Optimal Page replacement

* Replace the page which is not used in longest Demand of time in future.

Example :-

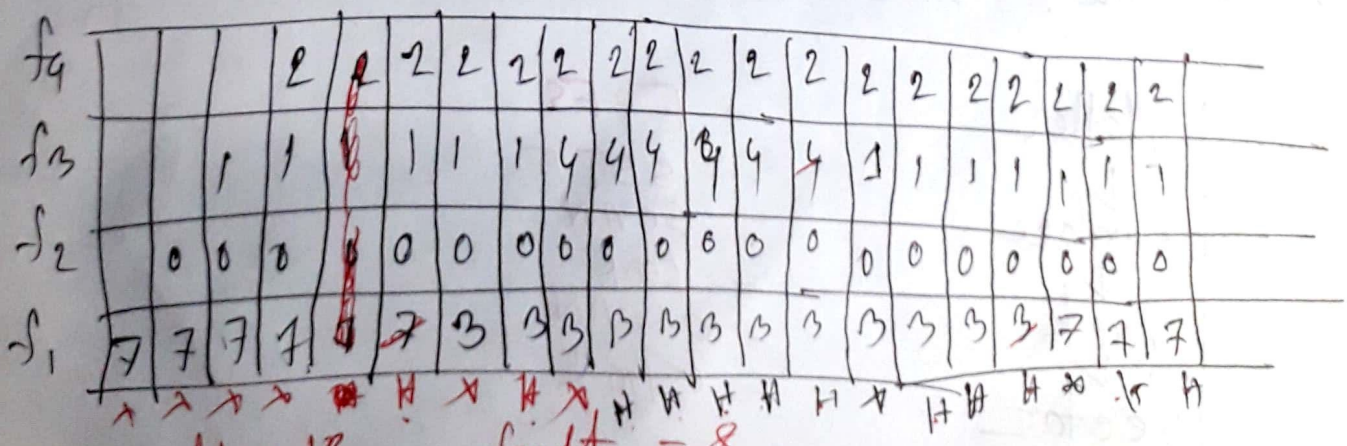
string :- 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



H = 12
fault = 8

Least Recently used (Replace the least recently used page in past)

Example



H = 12, fault = 8.

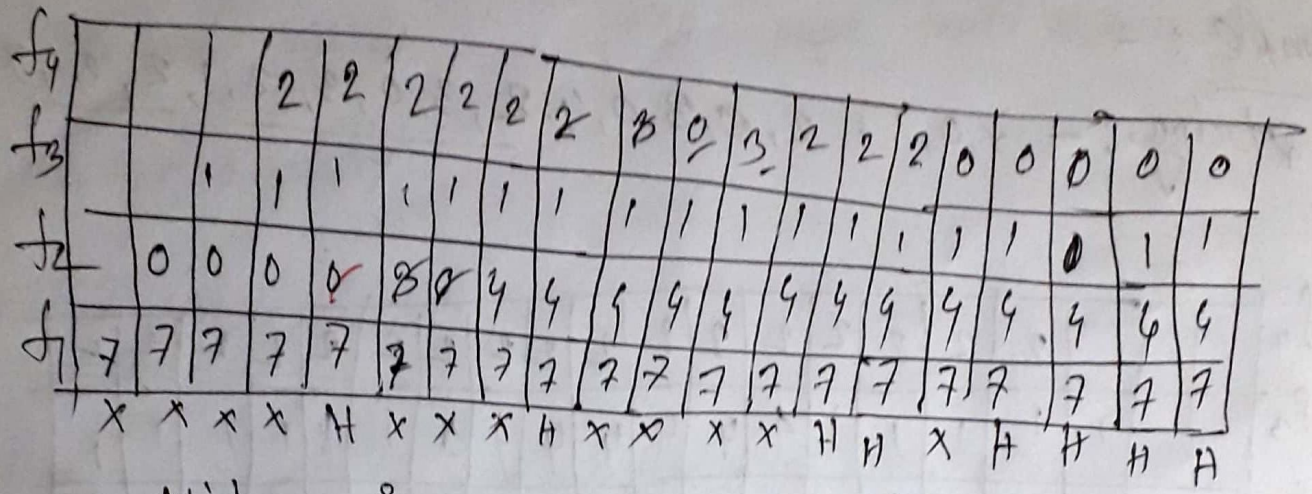
Scanned by CamScanner

Most Recently used (Replace the most recently used Page in past)

Example

string: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

8/12



Hit = 8

Miss = 12

~~Disk Management~~

Disk scheduling Algo

Scanned by CamScanner