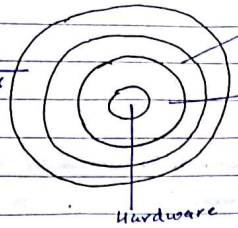
24/1/17

Structure of operating system



like in MS word we have copy, mv, cv, etc commands.

Shell contains batch file which contains a no. of DOS commands.

↓ converts DOS commands into system calls.

1. It is an interface between the user and the hardware.
2. It manages the resources of the computer in an optimum manner.

In GUI when we click on any particular icon an event is generated and a program is written in it beforehand which allows the operation to be performed.

1st user interacts with the shell which has specific commands then Kernel converts commands into set of system calls. Microoperation are the system calls. Then the system calls are fired to the hardware.

* Shell contains one or more commands.
* Kernel contains a set of system calls for a single command and hardware understands these system calls as it cannot understand user specific commands. These commands are understood by the user only.

→ Prepare for next turn :- Difference b/w. Job, Task, Process and Thread

Program :- static

Process :- dynamic state of a program.

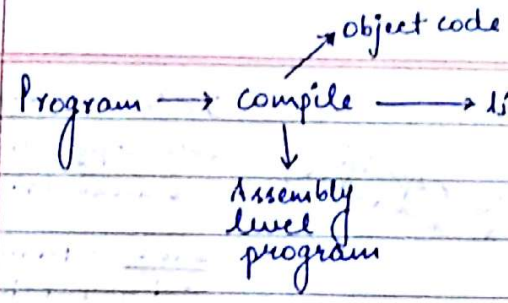
when a program is loaded into memory it is known as process..

or

when the program is in execution phase it is known as process.

www.imrankhalid.in

Scanned by CamScanner



Assembler

Assembler is a computer program which is used to translate program written in assembly language into machine language the translated program is known as object program. Assembler checks each instructions for its correctness and generates diagnostic messages if there are mistakes in the application.. An assembler enables software to access and manage the computer hardware.

It translates mnemonic operation codes to their machine language equivalents, it also assigns the machine addresses.

The output of the assembler program is called the object code

An object code is usually a machine code which can be understood directly by a specific type of a CPU.

Compiler

A compiler is a program that translates a high level language program to executable machine language

Job of a Compiler

1. To translate high level language into machine code
2. To assign memory to the variables in the program.
3. To include linkage for sub-routines and functions.
4. To generate error messages if there are errors in programs.

Interpreter

Interpreter is a computer program that executes instructions written in a programming language. In a compiler the program is translated completely and executable is generated. whereas in an

www.imamraid.in

Scanned by CamScanner

classmate

Interpreter each instruction is translated and executed and then the next instruction is translated and executed. In this case object code is not formed.

Linking and Object Program

1. Combines more than one assembled module into one executable program
2. generates an exe module and initialises it with special instructions to facilitate its loading for execution.

Loading

1. Once a program is ready it can be run from the command window
2. The loader component of the operating system takes the executable image from disk and transfers it to memory for execution.

Steps for executing a program

Source program is the assembly language program
The object program this is the output of the assembler.

1. The first step is the translation of the source code into the object code and generating an intermediate obj file, this is done by the assembler.
2. The second step is the linking step which envolves of converting the obj module to an exe machine code module. The linkers task is to combine seperate assembled programs into one executable module
3. The last step is to load the program for execution. It is the task of the loader to load the program from the hard disk to the memory.

www.imrankhalid.in

Scanned by CamScanner

Unit - 2

Process

A process is defined as an entity which represents the basic unit of work to be implemented in the system. A process is an execution of a program. A process is complete with two things :- data and resources.

Code It is the object program to be executed.

Data This is the data used for executing the program.

C = (a) + (b)

Data of the program.

Resources While executing the program, the program may require some resources like printer, RAM or Hard Disk.

Process :

1. Code
2. Data
3. Files

(Thread has these 3 things common in it.)

4. Register
5. Stack
6. Program Counter

Thread

Thread is the execution of the smallest unit of processing that can be scheduled by an operating system. It results in the breaking of a computer program into two or more running tasks.

Multiple threads share the resources whereas different processes do not share the resources.

Threads are known as a light weight process (LWP). Each thread needs a different stack. This is because each threads will have different procedures and a different execution history.

www.imrankhalid.in

Scanned by CamScanner

A thread consists of a program counter, a register ~~stack~~ set and a stack. Threads share with other threads code, data and files.

Jobs

The names job and task mean the same. In a batch system we refer to job and in a time share system we refer it to as task.
Even on a single user system such as Microsoft Windows a user may be able to run several programs at one time a word processor, a web browser and e-mail therefore these programs are also known as task and collectively all the task are known as processes.
The terms job and process also mean the same.

Process

A process is a program in execution. A process is not only a program code which is also known as the text. It also includes the activity represented by the program counter and the content of the process register.
A process includes -

1. Stack : Contains temporary data such as function variables/parameters, return addresses and local variables.
2. Data section : This contains the global variables.
3. Heap : This is a memory that is dynamically allocated during process runtime.
4. Code : It is the code of the program.

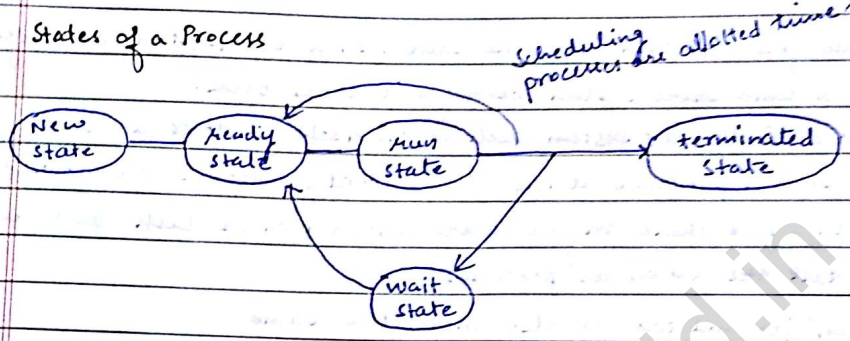
www.imrankhalid.in

Scanned by CamScanner

Process are of two types

1. CPU Bound
2. I/O Bound

States of a Process



As a process executes it changes state. The state of a process is defined by the activity of that process, each process has the following states.

1. New State : The process is being created.
2. Ready state : The process is waiting to be assigned to a processor.
3. Run State : Instructions are being executed.
4. Waiting State : The process is waiting for some event to occur.
5. Terminated State : The process has finished execution.

Only one process is running on any processor at one time :.

www.imrankhalid.in

Scanned by CamScanner

Process Control Block

Each process is represented in OS by a process control block also called a task control block.

It contains information associated with a specific process. The information stored in the PCB

1. Process State
2. Program Counter
3. CPU register : They include accumulators, index registers, stack pointers and general purpose registers.
4. CPU scheduling information : This information includes process priority and pointers to scheduling queues.
5. Memory management information: This information includes the value of the base and limit registers the page table and the segment table.
6. Accounting information : This information includes the amount of CPU and realtime used.
7. I/O status information : This includes the list of I/O devices allocated to the process.

Scheduler

The task of selecting a process is carried out by the scheduler. There are 3 different types of scheduler.

(1) Long term scheduler

Often more processes are submitted than can be executed immediately. These processes are spooled to mass storage devices where they are kept for later execution. The long term scheduler or job scheduler selects processes from this spool and tasks them into memory for execution.

(2) Short term scheduler or CPU scheduler

Selects from among the processes that are ready to execute and allocates to CPU one of them.

www.imrankhalid.in

Scanned by CamScanner

(3) Medium term schedular

Sometimes it is very important or advantageous to remove processes from memory and later on these processes can be re-introduced into memory and its execution is started from where it was left. This is also known as swapping.

The primary distinction in the schedulars lies in the frequency of execution must select a new process for the CPU very frequently often the short term schedular executes atleast once in every 100 milliseconds because of the short ~~term~~ time between executions. the short term schedular must also be very fast. The long term schedular executes less frequently, if the degree of the multiprogramming is stable then the average rate of process creation must be equal to the average ~~process~~ departure rate of process leaving the system or RAM. Therefore a long term schedular is invoke. when a process leaves the system.

Preemptive and Non-Preemptive Scheduling

Under non-primitive scheduling once the CPU has been allocated to a process the process keeps the CPU until it releases the CPU either by terminating or by switching to waiting state. A process is removed from the run state to the ready queue then we say that the process has been preempted or forced to relinquish its control over the CPU.

www.imrankhalid.in

Scanned by CamScanner

Scheduling Criterion

Different CPU scheduling algorithms have different properties and the choice of a particular algorithm may favour 1 class of process over another.

Many criteria have been suggested for comparing CPU scheduling algorithms. The criteria include the following.

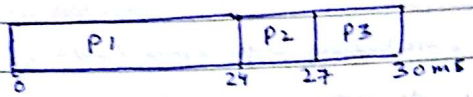
1. CPU utilisation → We want to keep the CPU as busy as possible. CPU utilisation is 40% in a lightly loaded system whereas in a heavily used system it is 90%.
2. Through put → It is the no. of processes that are completed in a given time. For a long process it may be 1 process per hour and for short transactions it can be 10 processes per second.
3. Waiting Time → The CPU scheduling algorithm does not effect the amount of time a process takes to execute or perform an I/O task. It effects only the amount of time that a process spends waiting in a ready queue. Waiting time is sum of the periods spend waiting in the ready queue.
4. Turn around time → The interval from the time of submission of a process to the time of completion is the turn around time. Turn around time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU and doing I/O.
5. Response time → Often the turn around time may not be the best criteria specially for interactive system. Therefore we calculate the response time which is time from submission of request until the first response is produced. It is actually the time the process takes to start responding.

www.imrankhalid.in

Scanned by CamScanner

FCFS ⟶ Non-preemptive

P1 → 24 ms
P2 → 3 ms
P3 → 3 ms



Gantt Chart

Process	waiting time (w.t.)
P1	0 ms
P2	24 ms
P3	27 ms

Avg wt. = $\frac{0+24+27}{3}$ = 17 ms

Process	TAT (Burst time + waiting time)
P1	0+24 = 24 ms
P2	24+3 = 27 ms
P3	27+3 = 30 ms

FCFS scheduling

This scheduling algorithm is the simplest of all the scheduling algorithm and with this scheme the process that requests the CPU first is allocated the CPU first. The implementation of FCFS is done using a queue.

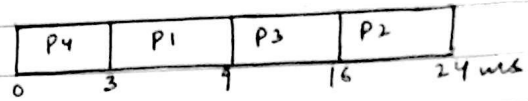
The average waiting time under the FCFS is generally quite long. The FCFS scheduling algorithm is non-preemptive. Once the CPU has been allocated to the process. The process keeps the CPU until it releases the CPU either by turning or requesting I/O.

www.imrankhalid.in

Scanned by CamScanner

SJF

P1 6 ms
P2 8 ms
P3 7 ms
P4 3 ms



Process	wt
P4	0
P1	3
P3	9
P2	16

TAT

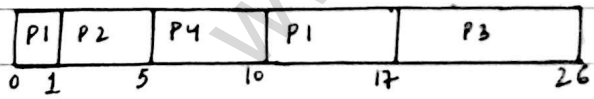
0+3 = 3
3+6 = 9
9+7 = 16
16+8 = 24
52

Avg. wt. = $\frac{0+3+9+16}{4} = \frac{28}{4}$ = 7ms

Avg. TAT = $\frac{52}{4}$ = 13 ms

Q- SJF.

	Arrival time (ms)	Burst time (ms)
P1	0	8
P2	1	4
P3	2	9
P4	3	5



Process	wt
P1	0+9 ms
P2	0 ms
P3	2 ms
P4	15 ms

Avg Wt = $\frac{26}{4}$ = 6.5 ms

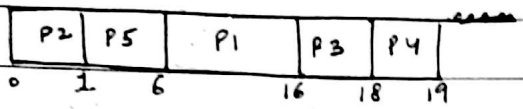
www.imrankhalid.in

Scanned by CamScanner

Priority Scheduling (with non-preemptive)

Process	Burst time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

G Charth



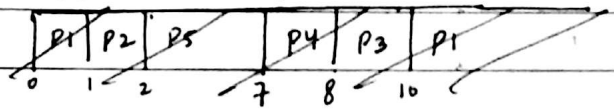
Process	wt
P2	0
P5	1
P1	6
P3	16
P4	18
Av. wt.	41/5 = 8.2 ms

TAT

0+1 = 1
1+5 = 6
6+10 = 16
16+2 = 18
18+1 = 19

60/5 = 12 ms (Av. TAT)

if premption is allowed. In question since arrival is not given we assume that all the processes were in the ready queue when the scheduling started.



Process	Burst time	Arrival time	Priority
P1	10	0 ms	3
P2	1	1 ms	1
P3	2	3 ms	4
P4	1	2 ms	5
P5	5	4 ms	2

www.imrankhalid.in

Scanned by CamScanner

classmate
Date
Page



$$0+1+1+4+7+16+18 = 47$$

$$\frac{47}{5} = 9.4 \text{ ms}$$

X Avg wt = $\frac{0+1+2+4+9+16+18}{5} = \frac{50}{5} = 10$ ms

✓ Avg wt = $\frac{0+0+1+0+6+13+16}{5} = \frac{36}{5} = 7.2$ ms

TAT = $\frac{18+1+15+17+5}{5} = \frac{56}{5} = 11.2$ ms

Starvation

A process that is ready to run can wait indefinitely in the ready queue when priority scheduling is running this is known as starvation because the process is unable to execute and is set to be starved of the CPU.

Aging

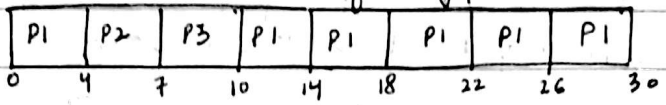
A solution to the problem of indefinite blockage of low priority processes is aging. Aging is a technique when the priority of a process is gradually increased

Round Robin scheduling (Like FCFS)

Arrival time for all processes are same. since arrival time for any process is not mentioned.

P1	24
P2	3
P3	3

4ms x quantum



Wt. time

P1 = 10 − 4 = 6 ms

P2 = 4 ms

P3 = 7 ms

$\frac{17}{3} = 5.66$ ms

www.imankhalid.in

Scanned by CamScanner

21/2/17

Memory Management

Address Binding

Address

- logical address → this address is in a sequential or serial manner. eg. S.No
- Physical address → It is the address at which the program is stored in the memory.

Address space → set of address assigned to the program.

Physical address = logical address + base address of program

for ex: = 0 + 100 = 100 → P.A.

Next instruction = 1 + 100 = 101 → P.A.

A program resides on disk as a executable file (data is in binary form). To be executed the program must be brought into memory and placed within a process. A process may move from disk to memory and vice versa during its life span. Most systems allows a user process to reside in any part of the physical memory. Although the address space of the computer starts at address zero, but the first address of the user process may not be address zero.

Addresses in the source program are generally symbolic. A compiler binds these symbolic addresses to relocatable addresses.

Binding is a mapping from address space to another.

① Compile Time

If it is decided at compile time that where a process will reside in memory then this is known as compile time binding. If the starting location of a program changes then the program has to be compiled again.

www.imrankhalid.in

Scanned by CamScanner

② Load Time

If it is not known at compile time where the process will reside in memory then the compiler generates relocatable code. In this case final binding is delayed until load time.

③ Execution Time

If the process can be moved during its execution from one memory segment to another then binding must be delayed until run time.

* All functions are generally known as routine.

Stub → It checks whether that sub-routine is present in the RAM or not. If it is not present then it generates an error. If the sub-routine is present in RAM then it links the sub-routine objcode with the program obj-code.

Dynamic Linking

At execution time, the sub-routines need not be built into the EXE.

compile + load + link + run

Static Linking

compile + link + Load + Run

↓

obj file

obj1 obj2
obj3 obj4
... obj n

↓

exe file

Swapping

RAM ⟶ Hard Disk

Hard Disk ⟶ RAM

www.imrankhalid.in

Scanned by CamScanner

Contageous Memory Allocation

1. First Fit
2. Best Fit
3. Worst Fit

Fragmentation

Both the first fit and best fit strategies for memory allocation suffer from external fragmentation as processes are loaded and removed from memory the free memory space is broken into little pieces external fragmentation exist when there is enough total memory space to satisfy a request but the available spaces are not contageous. Storage is fragmented into a large no. of small holes.

Hard disk	RAM
Page	Block

Size of 1 page = Size of 1 block.

Program → set of instructions

if there are 100 instructions

and 1 page has 5 lines only.

1 page = 5 instructions

20 pages = 100 instructions.

if there are 96 pages then also 20 pages are used.

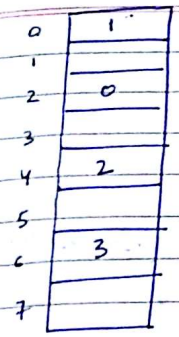
So, the 20th page 4 addresses are left unused.

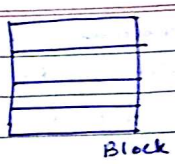
↓

internal fragmentation

www.imrankhan.in

Scanned by CamScanner





No. of pages	offset

If there are 20 instructions and 1 page accupies only 2 addresses then there are 10 pages so.

Page No. Line No.

0 0 0 0	0

4 bits = 10 page → 1 bit for 2 lines (0 or 1)

↓

0 ↔ 9

60 programs and if 1 address = 1 instruction.

4 address in 1 page in total 15 pages. $\left(\frac{60}{4} = 15 \text{ pages}\right)$.

0	0	0	0	0	0

4 bits for 15 pages 2 bits for 4 lines

0	0	0	0	0	0

www.imrankhalid.in

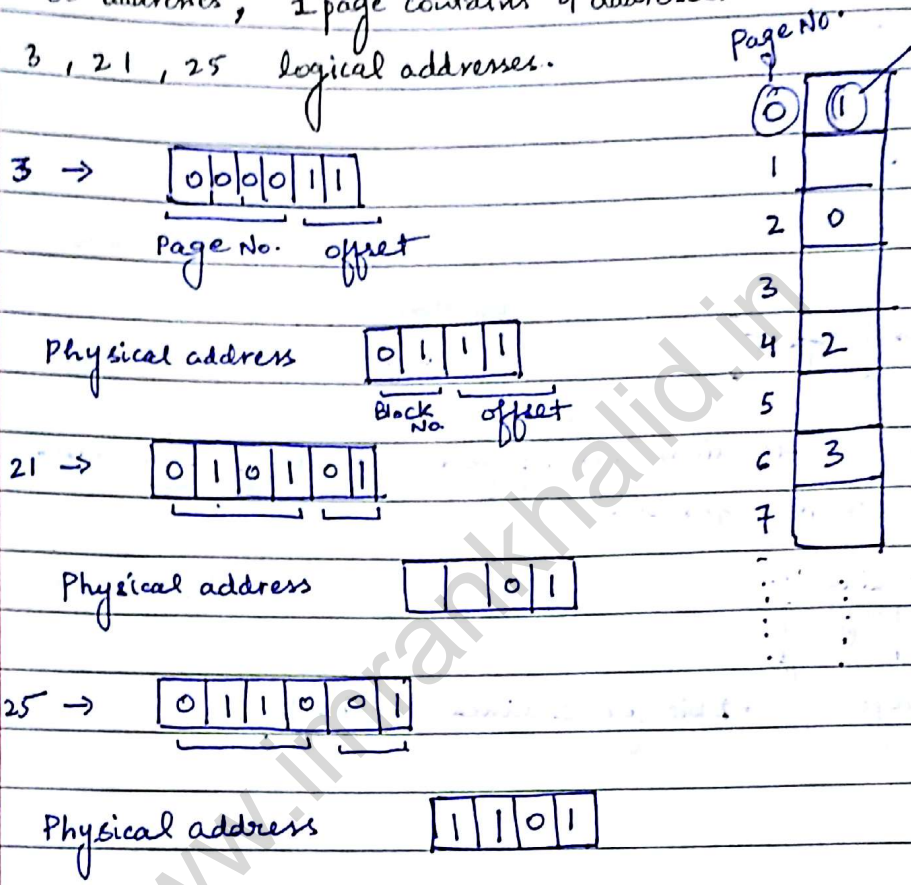
Scanned by CamScanner

If memory have 4 blocks then it will take 2 bits and offset will require 2 bits

so, physical address will be = 2+2 = 4 bits.

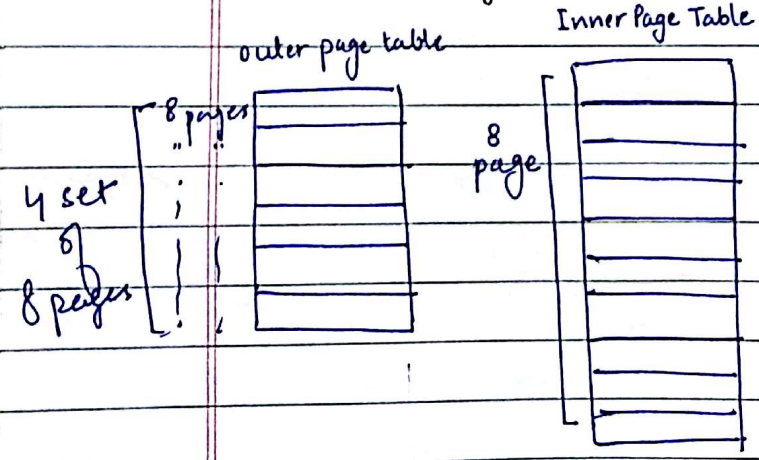
Q. 60 addresses, 1 page contains 4 addresses.

3, 21, 25 logical addresses.



Different Types of Page Table (are made so that it takes less space in RAM)

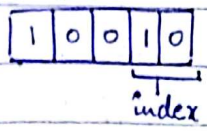
① Hierarchial Page Table

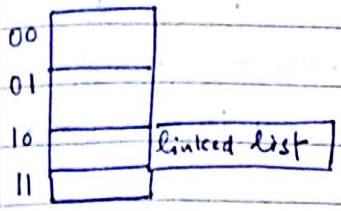


www.imrankhalid.in

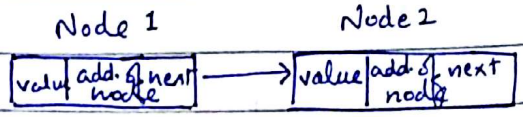
Scanned by CamScanner

(2) Hash Table





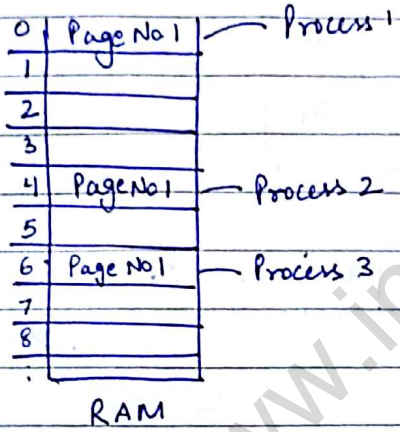
This contains nodes



like this there are 8 nodes

if a node does not match then the next node is referred.

(3) Inverted Page Table (Frame Table)



1 Block = 1 Frame

Segmentation

It is of unequal size and have many programs

Declaration Part

There are different segments containing different address spaces. like

Segment 1 — I/O based

Segment 2 — C.P.U based.

www.imrankhalid.in

Scanned by CamScanner

Segment Table

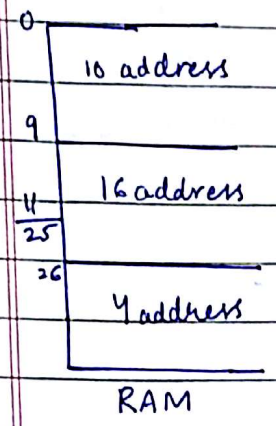
Segment	Base	limit

describes the maximum no. of segments that can be present.

10 11	
Segment No.	offset

4 bits for this particular case

As RAM below contains maximum 16 addresses.



Segment Table

0 →	0	9
1 →	10	25
2 →	26	29

2	8	→

26+8 = 34

34 > 29

26
+11
37

If offset < limit then it is a valid

If offset > limit then it is not valid

Sharing of Pages

Page 1

2
3
4

Page 2

2
0
1

Page 3

2
5
0

www.imranhalid.in

Scanned by CamScanner

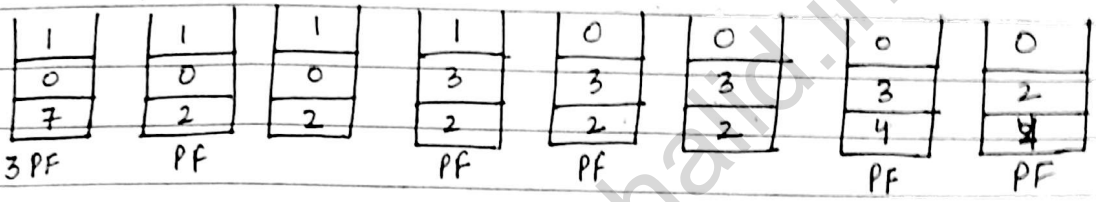
Virtual Memory

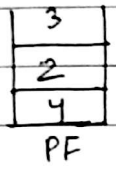
Replacement Technique

1. FIFO (First IN First OUT)
2. LRU (Least Recently Used)
3. optimal

Series 7, 0, 1, 2, 0, 3, 0, 0, 4, 2, 3

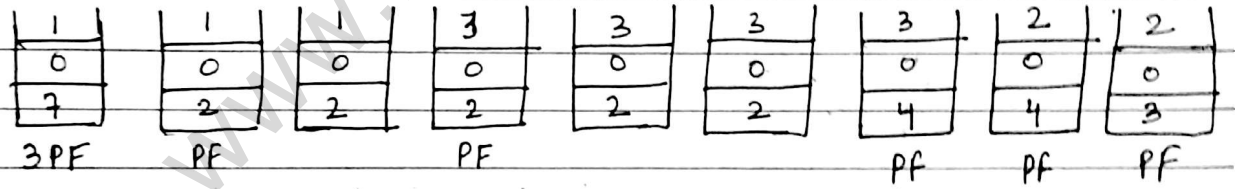
(1)





9 Page faults in FIFO

(2)



8 page faults in LRU.

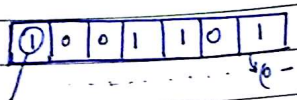
www.imrankhalid.in

Scanned by CamScanner

LRU

Reference Bits

1 → It has been referred.
0 → Not referred.



The no. that have 1 as its MSB that means its the recently used.

The one whose decimal count is the least of all is the least recently used.

Second Chance Algorithm (also done through reference bits)

1 0 0 1 0 0 1

Modified Second Chance Algorithm

referred → write

0 0
0 1 → ex: c = a+b here we only store the result.
1 0
1 1

Balady's Anomoly

In some cases of FIFO if we increase the memory then instead of decreasing the page fault it increases.

www.imrankhalid.in

Scanned by CamScanner

classmate
Date
Page 27

Unit-2

Synchronisation

Prog 1	Prog 2	Prog 3
c = a + b	c = a - b	a = a + 1
Critical section printf (c)		

Once a program is there in critical section till the time it is not executed ~~it with~~ next program will not execute.

To guard against a race condition we need to ensure that only one process at a time can be manipulating the shared variables, to make such a guarantee synchronisation of the processes is required.

The Critical Section Problem

There might exist 'n' no. of processes in a system. Each process has a segment of code called the critical section in which the process may be changing common variables. When one process is executing in the critical section no other process is allowed in the critical section therefore, we say the execution of critical section by the processes is mutual exclusive. The critical section problem is designed so that the processes co-operate with each other. Each process must request permission to enter its critical section. The section of code implementing this ~~set~~ request is the entry section then you have the critical section followed by the exit section, the remaining part is the remainder section.

www.imrankhalid.in

Scanned by CamScanner

```
do
{ Entry Section
  Critical Section
  Exit Section
  Remainder Section
}
while ( )
```

There are 3 conditions that can solve the problem for critical section.

① Mutual Exclusion (At one time only 1 process will enter into critical section)
② Progress (If a process has crossed all the first 3 section and is there in the remainder section then it cannot enter again into the critical section as other processes are already there into the entry section)
③ Bound waiting

A solution to the critical section problem must satisfy the following 3 requirements.

① **Mutual Exclusion**

If process pi is executing in its critical section then no other process can be executing in their critical section.

② **Progress**

If no process is executing in its critical section then only those processes that are not executing in their remainder section can participate in the decision to enter the critical section

③ **Bound Waiting**

There exists a bound on the no. of times that other processes are allowed to enter their critical section, after a process

www.imranknaik.in

Scanned by CamScanner

has made its request to enter the critical section.

Algorithm 1

Process i

```
do
{ while (turn == j);
   critical section
   turn == j;
} while (1)
```

Process j

```
do
{ while (turn == i);
   critical section
   turn == i;
} while (1)
```

Algorithm 2

```
do
{ flag[i] = true;
  while (flag[j]);
  critical section
  flag[i] = false
} while(1)
```

```
do
{ flag[j] = true;
  while(flag[i]);
  Critical section;
  flag[j] = false
} while (1)
```

Peterson's Solution

```
do
{ flag[i] = true;
  turn = j;
  while (flag[j] == true && turn == j);
  critical section
  flag[i] = false;
} while(1)
```

```
do
{ flag[j] = true;
  turn = i;
  while(flag[i] == true && turn == i);
  critical section
  flag[j] = false;
} while (1)
```

www.imrankhatri.in

Scanned by CamScanner

If we want that execution order of instruction should be as 1st 2nd.

[Instruction 1
Instruction 2] [Instruction 3
Instruction 4]

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

Process 1
Instruction 1;
Instruction 2;
Signal();

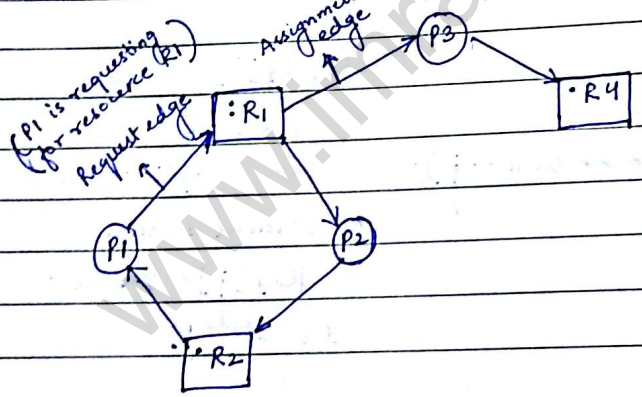
Process 2
wait(S);
Instruction 3;
Instruction 4;

Deadlock

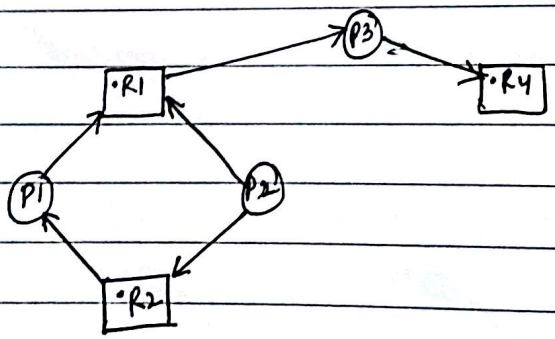
For Deadlock to occur the following should be present.

1. Mutual Exclusion
2. Hold and Wait
3. No preemption
4. Circular Wait

Resource Allocation Graph



Q.



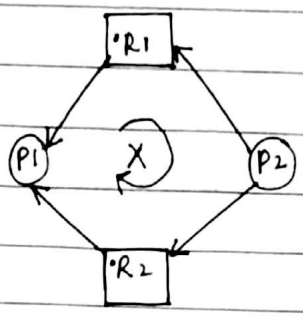
www.imrankhalid.in

Scanned by CamScanner

classmate
Date
Page

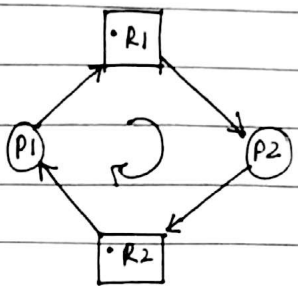
(1) When R1 is assigned to P1 first.
(2) When R1 is assigned to P2 first.

(1)



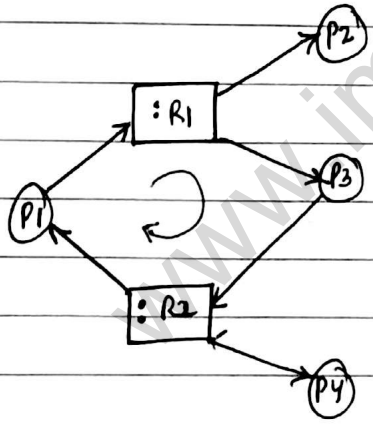
No Deadlock situation occurs
(1) P3 will be completed.
(2) P1 will be completed.
(3) P2 will be completed.
Cycle does not exist.

(2)



Cycle exists and deadlock situation arises.

Q.



Cycle exists and no deadlock situation occurs.

www.imrankhalid.in

Scanned by CamScanner

Magnetic Tapes = 12

Process	Maximum Needs	Current Needs	Needs left
P_0	10	5	5
P_1	4	2	2
P_2	9	2	7

12 − 9(current Needs) = 3

Available = 3

P1 → 2 → 3-2 = Available 1

P1 will release 4 Magnetic Tapes after terminating

So available = 1+4 = 5

Po will execute now → 5 ⇒ Available=0

Po will release 10 Magnetic Tapes.

Now P2 will execute → 7 ⇒ 10 - 7 = 3 = available

(P1, Po, P2) → safe sequence

Safety Algorithm

① Work = Available and Finish[i] = false for all i.

② Find an i such that both

Finish[i] = false, need ≤ work

if no such i exists goto step 4.

③ work = work + allocation[i]

finish[i] = true

goto step 2.

④ if finish[i] = true for all i then the system is in safe state.

www.imrankhalid.in

Scanned by CamScanner

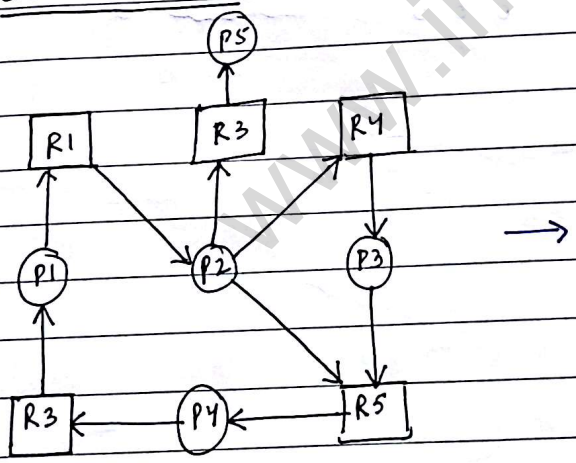
Process	Allocation	Maximum	Available	Need left
	A B C	A B C	A B C	A B C
P0	0 1 0	7 5 3	3 3 2	7 4 3
P1	2 0 0	3 2 2		1 2 2
P2	3 0 2	9 0 2		6 0 0
P3	2 1 1	2 2 2		0 1 1
P4	0 0 2	4 3 3		4 3 1

P1, P3, P4, P0, P2

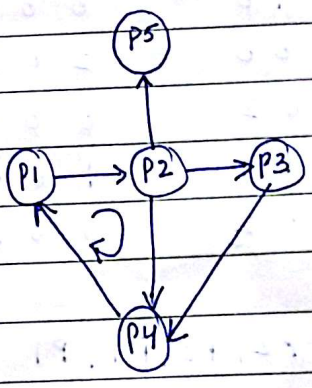
Request-Process Algorithm

1. if request ≤ need goto step 2. otherwise raise an error condition.
2. if request ≤ Available goto step 3
3. Available = Available − Request.
   Allocation = Allocation + Request
   need i = need i − Request

Deadlock Detection



Resource Allocation Graph



Wait for Graph

As cycle exists and each process has only one instance. so deadlock exists.

www.imrankhalid.in

Scanned by CamScanner

Algorithm

1. Let work = Available for $i = 0, 1, n-1$
   if Allocation $\neq 0$ then Finish [i] = false.
   otherwise Finish [i] = true
2. Find an index i such that both
   (a) Finish [i] = false.
   (b) Request i $\leq$ work
   if no such i go to Step 4.
3. work = work + Allocation i
   Finish [i] = true
   Go to Step 2
4. if Finish [i] = false for some i $0 \leq i < n$
   then the system is deadlocked.
   if Finish [i] = false the process P is deadlocked.

Process	Allocation	Request	Available
	A B C	A B C	A B C
P0	0 1 0	0 0 0	0 0 0
P1	2 0 0	2 0 2	
P2	3 0 3	0 0 0	
P3	2 1 1	1 0 0	
P4	0 0 2	0 0 2	

P0, P2, P3, P4, P1

www.imranikhalid.in

Scanned by CamScanner

Unit - 4 (File Management)

File

A File is a named collection of related information that is recorded on secondary storage.

A File is the smallest allotment of logical secondary storage. It means that data cannot be written unless it is within a file. A file can represent both program and data. Files maybe text files or maybe formatted in the form of a table.

Many different types of information maybe stored in a file.

1. Source Program (High Level Language Program)
2. Object Program (Machine Language Program)
3. Executable Program (Linked Program)
4. Numeric Data
5. Text
6. Pay Roll
7. Graphic Images
8. Sound Recordings.

File Attributes

The file attributes are the property of a file by which we identify the file

1. Name
2. Identifier (Unique id or primary key)
3. type (.exe, .dat, etc)
4. location (where it is stored)
5. Size
6. Protection (Access control information)
7. Time, date and user identification.

File Operations

Creating a File (Two steps are necessary to create a File)

1. Space in the file system must be available.
2. Entry for the new file must be made in the directory.

www.imrankhatri.in

Scanned by CamScanner

classmate
Date
Page

Writing to a File

To write a file we make a system call for both the name of the file and the information to be written into the file. With the name of the file the file system searches the directory to find the file location. The system must keep a write pointer to the location in the file where the next write is to take place.

Reading a File

To read from a file we use a system call that specifies the name of the file again the directory is searched for the associated entry and the system keeps a read pointer to the location in the file where the next read is to take place...

Repositioning within a File

The directory is searched for the appropriate entry and the current file position pointer is repositioned to a given value. Repositioning within a file need not involve any I/O.

Deleting a File

To delete a file we search a directory for the named file. After finding the associated directory we release all file space that can be used by other files

Truncating a File

By this we erase the contents of a file but we keep its attributes

~~Several pieces~~

www.imrankhalid.in

Scanned by CamScanner

several pieces

1. File Pointer (same as the read/write pointer)
2. File Open Count (How many process have opened the file)
3. Disk location of the file
4. Access Rights

* File Types (DO it Yourself)

Directory Structure

The file system consists of two distinct parts -
A collection of file and a directory structure which organises and provides information about all the files in a system.
The File system is the most visible aspect of an operating system.
It provides the mechanism for online storage and access to both data and programs of the operating system and all the users of the computer system.

Directory

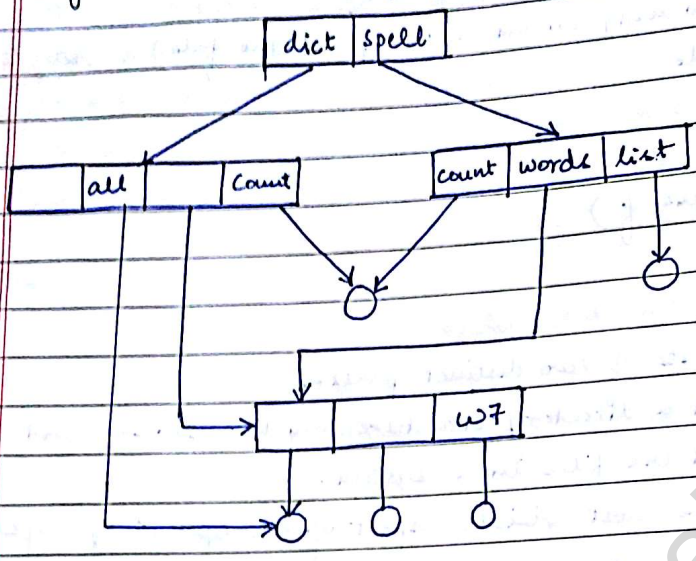
The directory is a table that translates files name into their directory entries. The various operations that are performed on a directory are :-

1. Search for a File
2. Create a File
3. Delete a File
4. List a directory
5. Rename a File
6. Traverse the File System

www.imrankhalid.in

Scanned by CamScanner

Acyclic Graph



A common subdirectory is said to be a shared sub-directory between two directories. A share directory or file will exist in the file system in two places at one time.

Tree directory structure prohibits the sharing of files or directories and acyclic graph is a graph with no cycles and allows directories to share sub-directories and files.

A link is a pointer to another file or sub-directory. The problem with links lies in deletion whenever the original file is deleted it leads to dangling pointers.

These dangling pointers can either be deleted or another approach is to preserve the file until all references are deleted.

www.mrankhalid.in

Scanned by CamScanner

Disk Scheduling

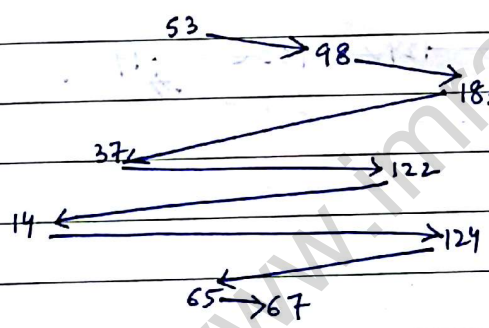
Access time = Seek time + Latency time

Time taken by the hard disk to read/write data

Time taken by the read/write head to move to a

Q: Track no. = 98, 183, 37, 122, 14, 124, 65, 67.

for FCFS order remain same



98	183	183	122	122	124	124	67
53	98	37	37	14	14	65	65
45	85	146	85	108	110	59	2 → Track movement

Shortest Seek Time First

According to above track no.

Q. 53, 65, 67, 37, 14, 98, 122, 124, 183

www.imankhalid.in

Scanned by CamScanner

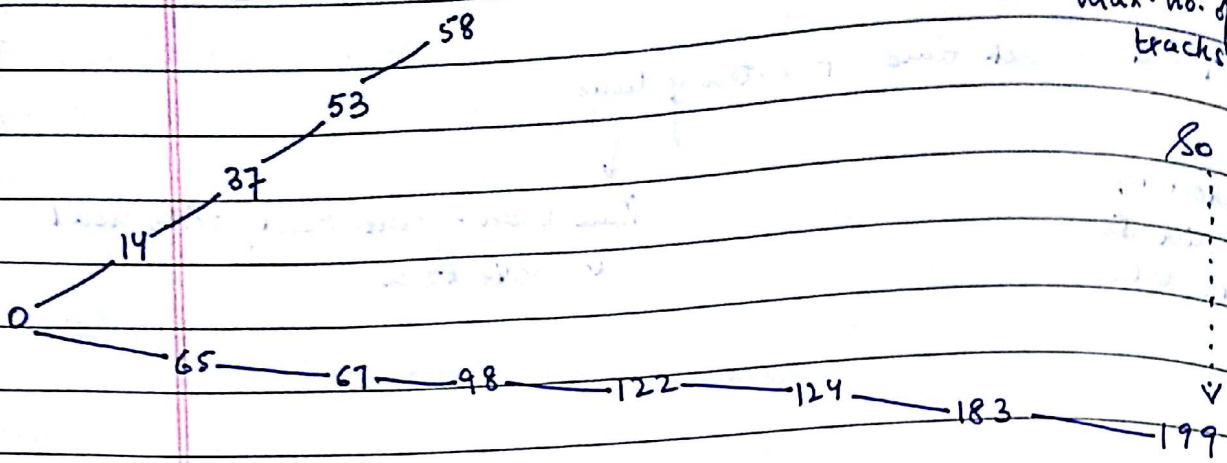
Track No = 98, 183, 37, 122, 14, 124, 65, 67

Scan.

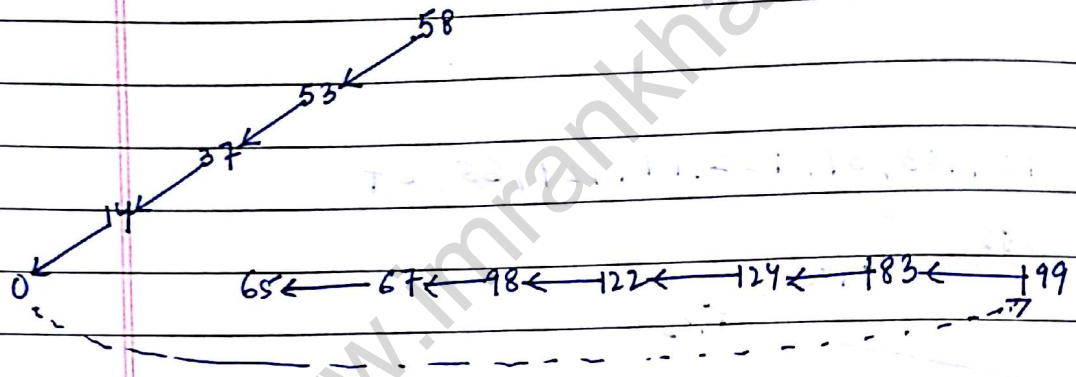
as 58 → 53 it is in decreasing order.

max. no. of tracks = 200

So in last we write.



Circular Scan



C look.

same as Circular Scan but after 14 it goes to 183 and difference between their seek time is not considered.

www.imrankhalid.in

Scanned by CamScanner