0/8/22

Unit 1 (Part 1)

Introduction : Computer software can be divided into two kinds :

- System Programs which manage the operation of the computer itself
- Application Programs which perform the actual work the user wants

Banking System	Airline Reservation	Web Browser	A/S
Compiler	Editor	Command Interpreter	S/S
OS			
Machine language Micro Architecture Physical Device			H/W

The computer system consist of hardware, system program and application programs

Types of System Softwares

1. Device Drivers : is a computer (software) program

Scanned by TapScanner

that operates or controls a particular type of device that is attached to the computer

2. Middle Ware : is a computer software that provides services to software applications beyond those available from the OS

3. Utility Software : It is a system software designed to help, analyze, configure, optimise, manage, maintain, repair data on a computer

11/8/22

4. Operating System : It is a system software or a program that manages the computer hardware. It is a basis for application programs and acts as an intermediary between the computer user and computer hardware

A computer system can be divided into four components

→ The Hardware (CPU, memory, I/O devices) provide the basic computing resources for the system

→ The application programme such as word processors spreadsheet define the ways in which the resources

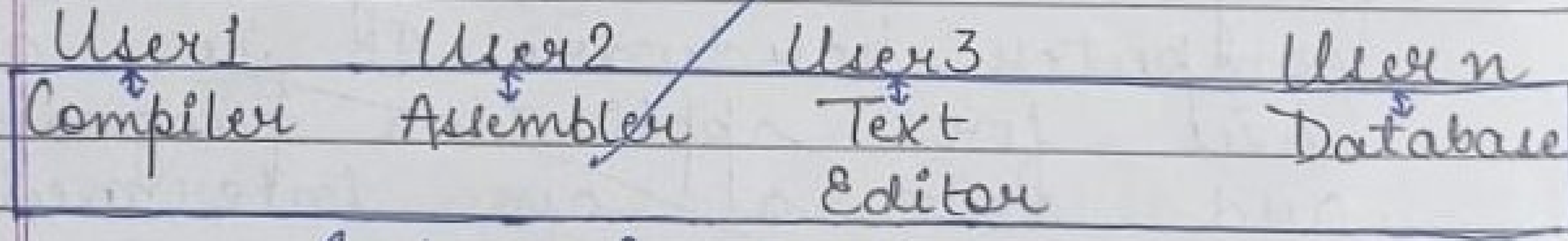
are used to solve computing problems

→ The OS controls the hardware and coordinates its use among the various application programs for the various users

→ The user uses various resources of the system through OS

⇒ Operating System

The most fundamental system program in the OS whose job is to control all computer resources and provide a base upon which the application programs can be written



System & application Programs
↓
Operating System
↓
Computer Hardware

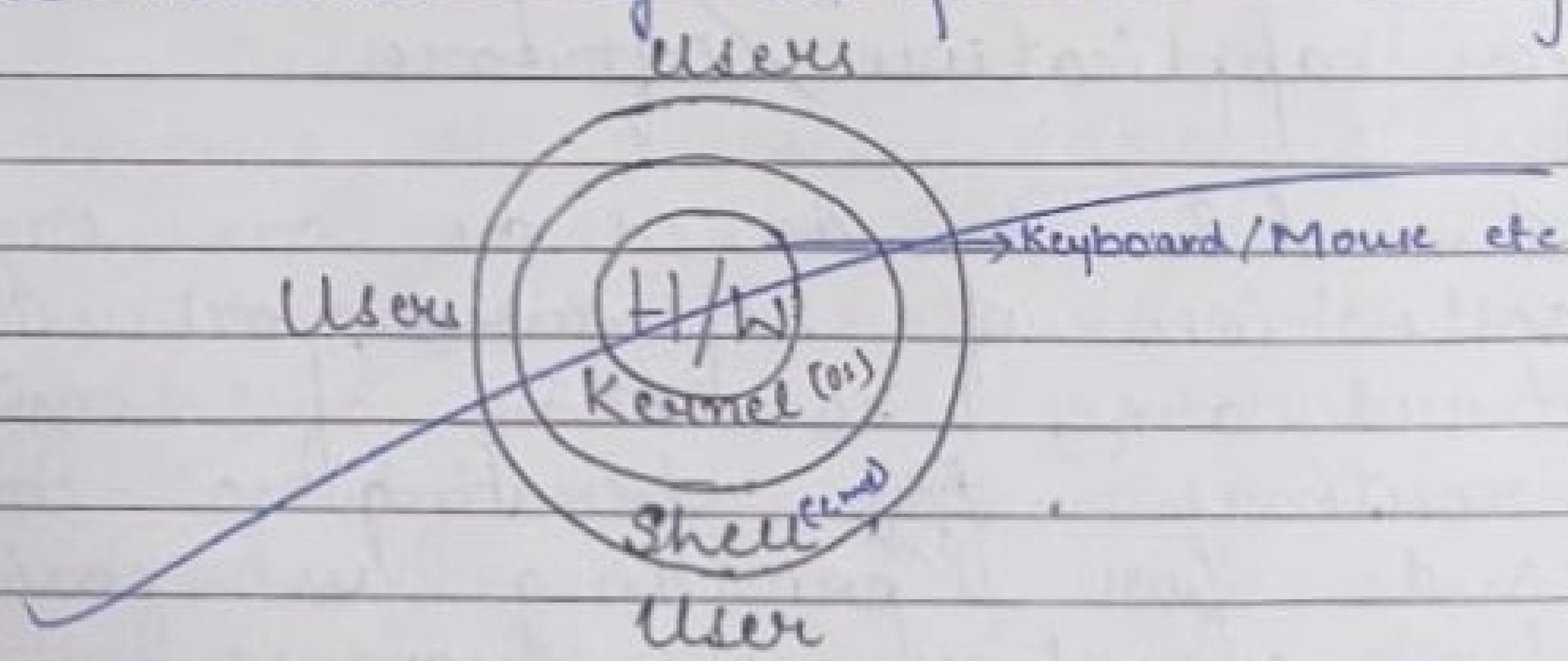
View Of the Componente Of a Computer System

Scanned by TapScanner

An OS is a control program that manages the execution of user programs to prevent errors and improper use of the computer. It is concerned with operation and control of I/O devices. The

The OS is that portion of the software that runs in kernal mode/supervisor mode. It is predicted from user tampering the hardware

Only the top of the OS is the rest of the system software here we find the command interpreter (shell), compilers, editors. They are preinstalled by computer manufacturer



1. Kernel : It is a layer between software and hardware. It is needed to have a complete command over all the activities of the system. It is also called system mode or priviledged mode

2. Shell: It is a layer which acts as an interface between the user and Kernel. Shell interacts with the Kernel and the Kernel interacts with the hardware. A shell is a command line interpreter it interprets the commands which the user types in. Shell cannot directly interact with hardware

An OS acts as an application interface to the hardware. API (Application Program Interface).
The OS has 3 main functions
- To manage the computer resources
- To establish an user interface
- And execute and provide services for application / software

An OS is viewed as an organized collection of software, extension of hardware, consisting of control routines for operating a computer and for providing an environment for execution of programs

OS acts as a resource manager and allocator: OS acts as a resource manager of the computer system such as processor, memory files and I/O

devices in this role the O.S system keeps track of the status of each resource for how long and when. In systems that support concurrent execution of programs the OS resolves conflicting requests of resources in a manner that preserves system integrety and to optimise the performance. The primary objective of operating system is to increase productivity of processing resource such as computer hardware or computer.

Resource Management includes sharing resources in two ways:-

1.) Time : When a resource is time shared it means different programs or users take turn using it first one program or user gets to use the resource then another and so on. For example: with only one CPU and multiple programs the OS allocates the CPU to one program and then another

2.) Space : When a resource is space shared it means that different programs or users, each one gets part of the resource

Scanned by TapScanner

Each one gets for ex- Main memory is normally divided into several running programs so each one can be resident at the same time

The User Interfaces with the Operating System in two ways

1. Command Interpreter : All command line interface that allows users to directly enter commands to be performed by OS. These are also called as shells. The main function of the command interpreter is to and execute the next user specified command

2. Graphical User Interface: a second strategy for interfacing with OS is through a user friendly Graphical User Interface. Rather than entering commands by command like interface mouse based entries are done

note: unix is closed source code. Linux is a open source code. Windows is closed

Scanned by TapScanner

18/8/22

and open source both. Android is open source.

Two view points on OS : From the user view

the OS is designed so that it can be easily used by the user with some attention paid ~~w~~ to resource utilisation. Performance is important to the user

ii.) From the computer point of view the OS is the program i.e. involved with the hardware. Here OS works as resource allocator who manages CPU time, memory space, file storage space, I/O devices etc

⇒ Operating System Services :

1. User Interface: all OS have a user interface that is either command line which uses text commands and the other one is graphical user interface which interacts through icons

2. Program Execution: The must be able to load a program into memory and then

Scanned by TapScanner

to run that program. The program must be able to end its execution either normally or abnormally

3. I/O operations: A running program may require I/O which may involve input output operation device. For efficiency and protection of I/O devices and its proper working OS is used.

4. File System implementation or manipulation: Programs need to be read from files, written into files. Files have to created, deleted, searched, modified etc. Some programs include permission to be allowed or denied

5. Communication: There are many circumstances in which when one process needs to exchange information with other process

6. Error Detection: The OS needs to be constantly aware of possible errors which may occur in the CPU, memory, io devices or in a

Scanned by TapScanner

network for each type of error the ~~os~~ OS needs to take proper action to ensure correct operation.

7. Resource allocation: When there are multiple users or multiple jobs running at the same time resource must be allocated to each of them

8. Accounting: To keep track of the usage of the resources by the user the OS carries a function with it.

9. Protection and Security: Protection involves ensuring that all access to system resources is controlled when many processes run at the same time the process should not interfere the other process.

Security of the system from outside is also important. Security can be maintained through user name and password to gain access to system resources

Scanned by TapScanner

22/8/22

User & other system Prog

↕

GUI command line

↕

User Interface

↕

System Calls

↕

Prog. execution	I/O operation	file system
resource allocation	communication	
accounting	error detection	Protection & security

↕

OS

↕

HW

Designing Issues of OS are divided into user goals and system goal. The OS should be convient to use easy, to learn, reliable, safe and fast for the user

The system should be easy to design, create, maintain and operate the system. It should be reliable flexible error free and efficient

Scanned by TapScanner

The running state of a program is called a process

watch
clock
throughput
turnaround
response

The system should have an effective choice of hardware

OS has mechanism and policies while designing. Mechanism means how to do something. Policy means what will be done

⇒ Main functions of OS

1. Process Management : A process is a program in execution # during execution a process needs certain resources such as CPU time, memory, files and I/O devices. A computer system consists of collection of processes. Process Management takes care of creation and deletion of processes, scheduling of system resources to different processes so that proper synchronization and communication can be there among processes

2. Memory Management : To execute a program it must be loaded in main memory together with the data it accesses. To improve CPU utilisation system keeps several programs in memory. Memory management

Scanned by TapScanner

① Processor ⟷ Register (general purpose memory) Cache ⟷ RAM ⟷ SM (Secondary Memory) ⟷ I/O device

(most powerful machine)

module takes care of allocation and de-allocation of memory space to programs

3. <u>File Management</u>: All computer system store, retrive and share information. A computer stores information in files. This management organizes, stores, retrive names share and protect files

4. <u>Device Management</u>: This management of OS controls all I/O devices it keeps track of I/O request from processes, issue command to I/O devices ensures correct data transmission to and from an I/O device

5. Security

6. Command Interpretation

Measurement of OS can be done through the following parameters

1. Throughput: is the amount of work that a system is able to do per

unit time

2. Turn around Time : It is the interval b/w the time of submission of job to the system for processing and the time of completion of job

3. Response time : is the interval between the time of submission of a job to the system for processing and the time of first response from the system for the job

Evolution of OS

1. Serial processing } uni-programming
2. Batch processing
3. Multi-programming

Types of OS

1. Batch OS : It requires processing generally between program, data and appropriate system commands to be submitted together in the form of a job or process or task. Batch OS usually allows little or no interaction between users and executing programs

Scanned by TapScanner

Batch processing has a greater potential for resource utilization than serial processing serving multiple users. Batch is not very easy for program development. Batch systems provide simple file management. ~~Sed~~ scheduling of job is very simple in first come first serve basis

Job: is the execution of a task

Task: is the excution of a process

Process: is the execution of a program

Program
↓↑
Process
↓↑
Task
↓↑
Job

2. Multi-Programming: Under this concept several sub-concepts are stated

⇒ (a.) Multi-Tasking OS has the ability to support concurrent

execution of two or more active processes

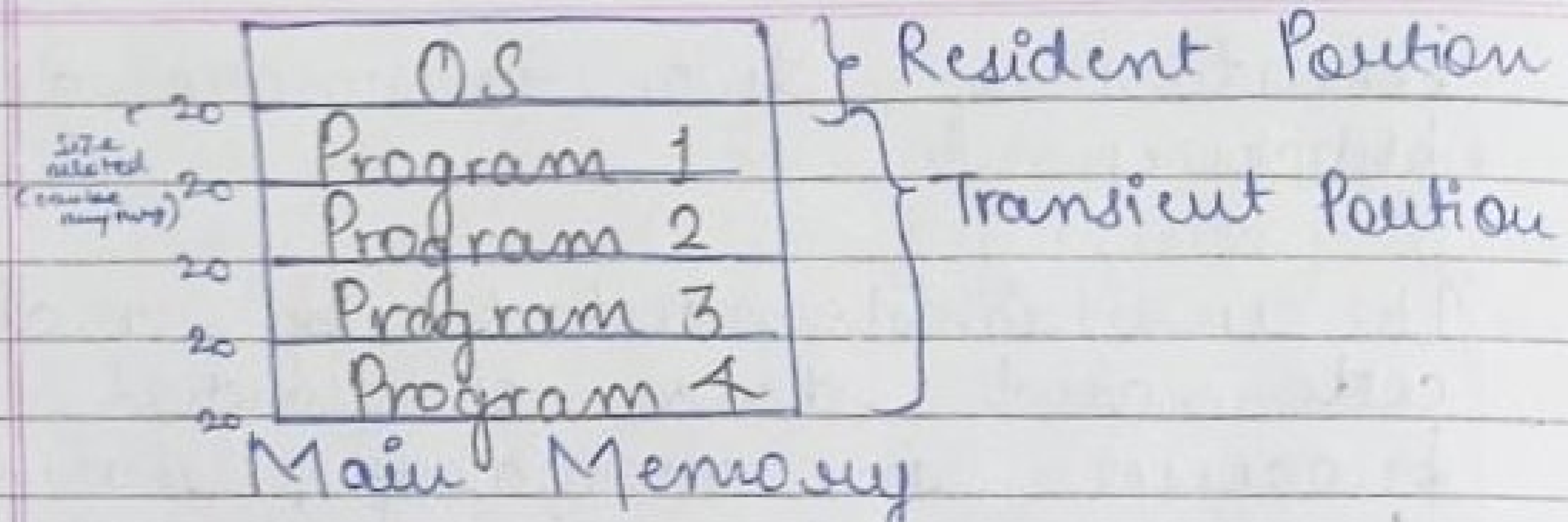
It is implemented by maintaing code and data of several processes in memory, sharing processor and I/O devices among them. Multi tasking supports memory protection in order to prevent ~~arrani~~ erroneous processes from corrupting memory

⇒ (b) Multi User Systems : Multi programming OS in addition to supporting multi tasking provides new forms of memory protection when processes access shared I/O device and files. Multi User OS supports multiple users. These systems provide individual user environment, user authentication for security, per user resource usage accounting

(c) Sub Concept: Multi-Processing manage the operations of computer system through multiple processors. It is a simultaneous execution of two or more processes by a computer system having more than one CPU

Scanned by TapScanner

29/4/22



3. Time Sharing System: It is a representative of multi programming system they are multi tasking system. Its objective is to provide the user one system and a good response time. It provides equal sharing of resources. If the program takes more time than it is schedule, the OS interrupts and places it at the end of program queue so the other programs gets the chance of execution.

4. Real Time System: are used in environments where a large no. of events external to the computer system must be accepted and processed in a short time. These systems are used in flight control, military applications, user convience and resource utilisation

are not so important they deal with thousands of interrupts per second

5 Distributed OS: is a collection of computer systems capable of communication and co-operation via their hardware and software interconnection these systems supports computer networks in which many computers are connected by protocols. The systems are transparent FTP (File Transfer Protocol) and NFS (Network File System) are used

6. Parallel Systems: In this execution of processes are done simultaneously it improves performance. Tasks are broken into sub-tasks that can be processed in-dependently and the results are combined afterwards. Parallel Computers are multiprocessing system having multiple processing elements within a single machine.

7. Network OS: It provides features such as file sharing across the network

that includes a communication scheme that allows different processes on different computers to exchange messages

8. Multimedia Systems: Multimedia system consist of multimedia data which be audio and video files as well as text files, programs, documents etc.

9. Embedded System: These systems are used in car engine, manufacturing robots and microwave ovens they perform very specific task. They little or no user interface. They spend their time monitoring and managing hardware devices such as automobile engines

- GPS

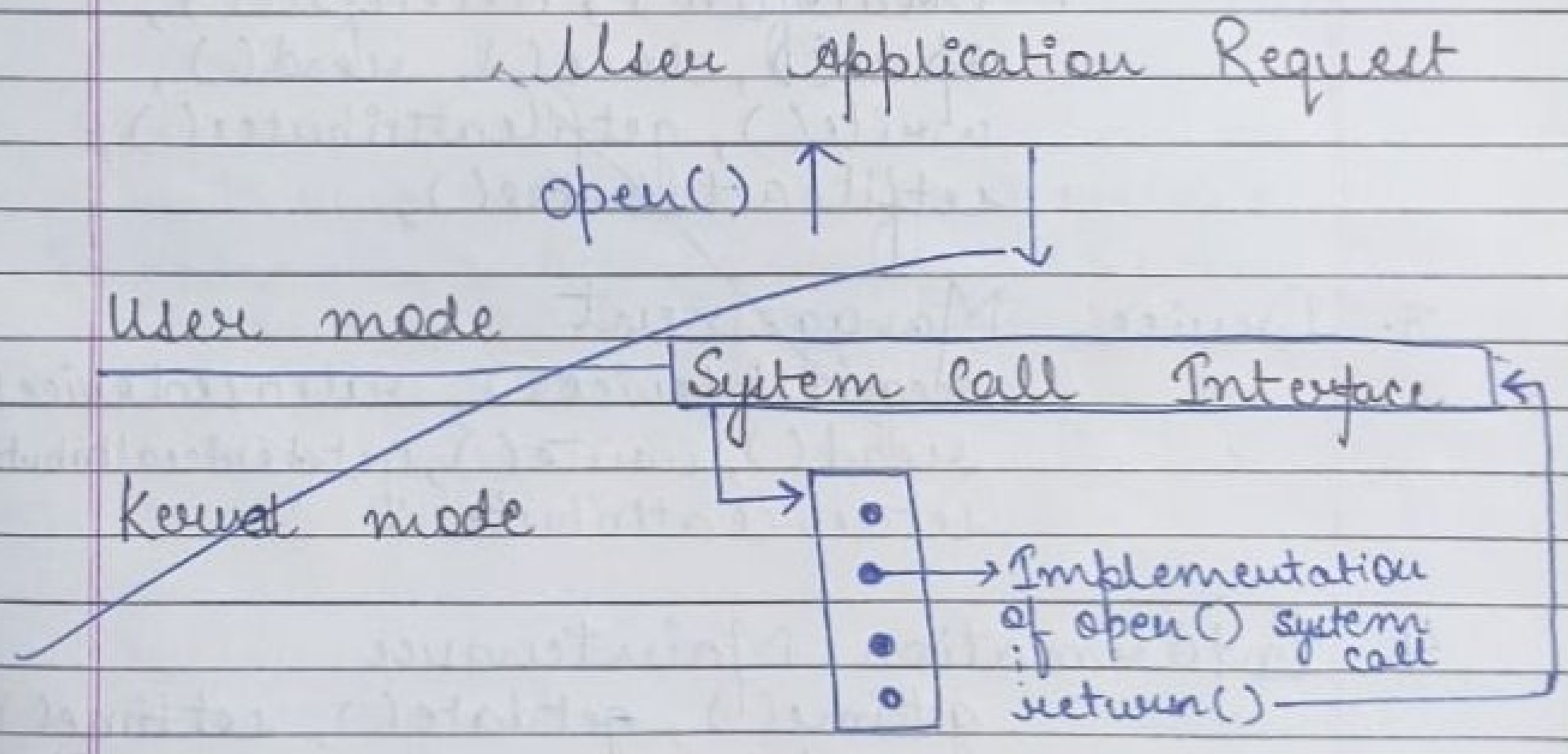
⇒ System Call

There are two ways of ~~interaction~~ interacting with OS
1. System call
2. OS commands

System call provide the interface to a running

Scanned by TapScanner

program and the Operating System Command Interpreter of our OS provides a set of command which users use to give instructions to the computer. Commands supported by command interpretation module are known as System calls when a user gives instruction to a computer by using these system call command interpreter interpret these commands and directs the computer resources to handle the users' request



Example of System Call

Types of System Calls

They are group into 6 major categories

1. Process Management

end(), execute(), abort(), createprocess(), load(), terminateprocess(), getprocessattributes(), setprocessattributes(), waitfortime(), waitevent(), signalevent()

2. File Management

createfile(), deletefile(), open(), close(), read(), write(), getfileattributes(), setfileattributes(),

3. Device Management

reqestdevice(), releasedevice(), read(), write(), getdeviceattribute(), setdeviceattribute

4. Information Maintenance

gettime(), getdate(), settime(), setdate(), getprocess(), getfile(), getdeviceattributes(), setprocess(), setfile()

5. Communication

create(), delete(), send()

Scanned by TapScanner

receive()

6. Protection

setpermission(), getpermission(), allowuser(), denyuser()

Scanned by TapScanner

10/22 Unit 1 Part 2

Memory Management

Memory is one of the central resources required for a computer operating. memory is organized in words. Each word has same size and each memory location is addressable and has unique fixed address. The CPU fetches instructions from memory according to the value of the program counter.

Memory Management is primarily concerned with allocation of main memory of some capacity to requisite processes. No process can even run if certain amount of memory is not allocated to it. The address where the components of the process or programs are loaded is decided, controlled and managed by the Operating System. A program resides on a disk as binary executable file.

Two important features of memory management function are:
i) Protection of processes and sharing of data

Main memory and the registers built into the processor itself are

the only storage that the CPU can access directly instruction and data need by the CPU must be in one of the direct access storage. If the data is not in the memory. they are moved in the memory before the CPU can operate on them. The operations through the registers can be done per clock cycle whereas operations through main memory can take a long time to balance the speed between the main memory and the CPU cache is used. Correct operation must be performed on the processes and they should be protected from other processes this protection must be given by the hardware. Each process has a seperate memory space the processes must determine the legal address space to ensure protection. This protection is given by two registers

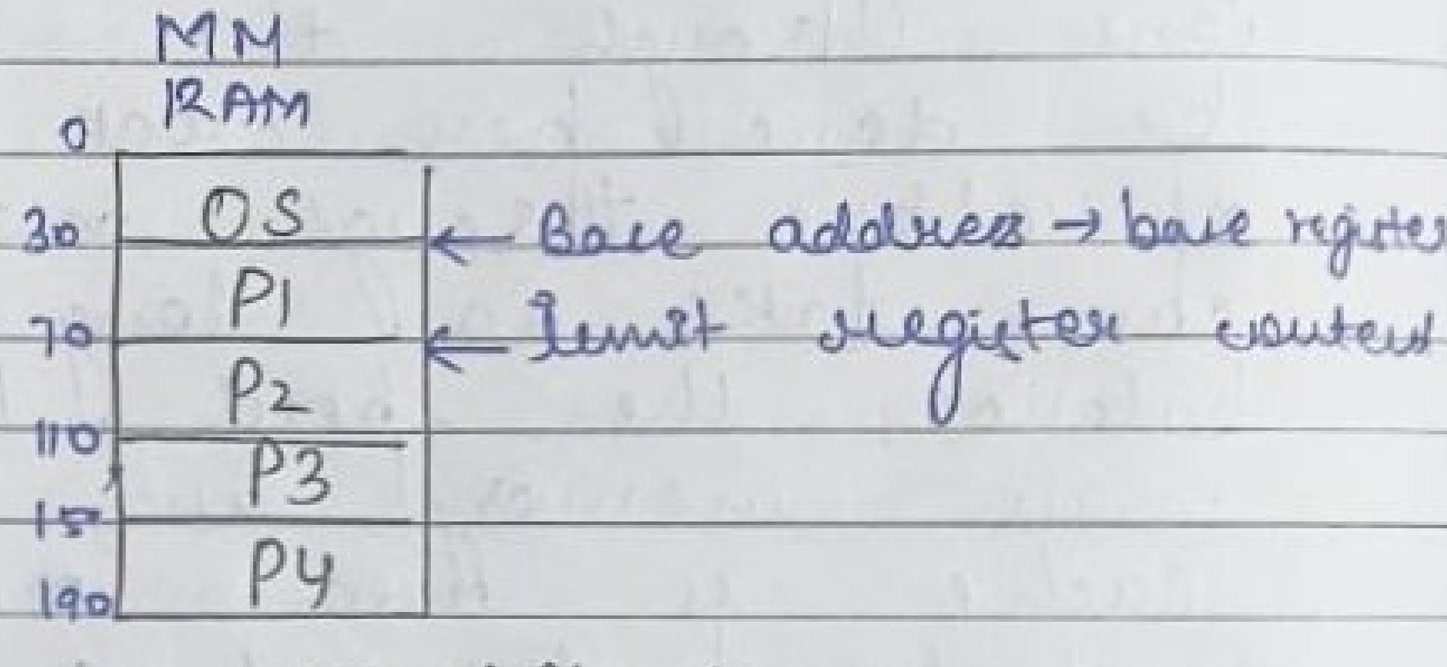
1. The Base register holds the smallest legal physical address also called as relocation register
2. The Limit register specifies the range

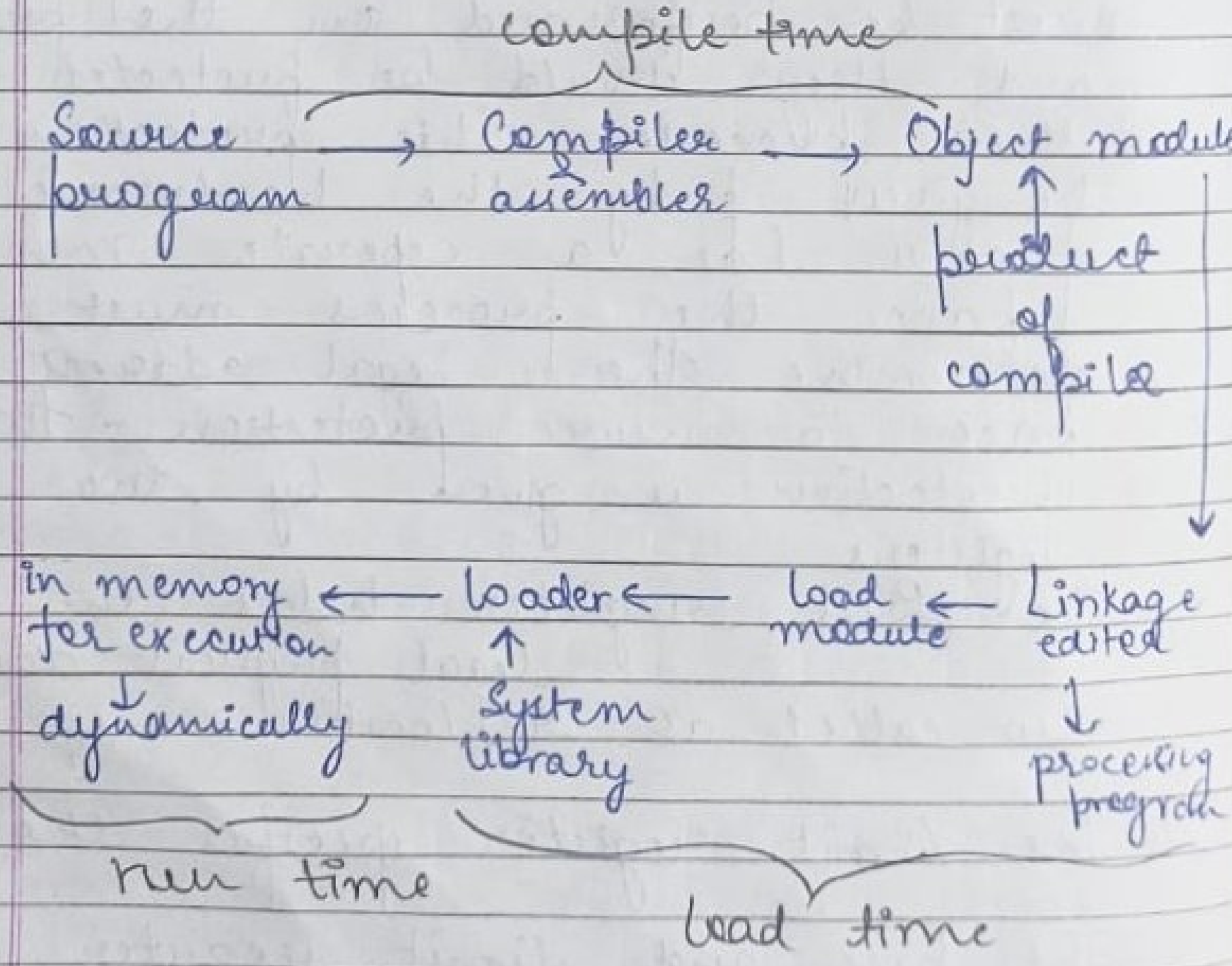
The base and limit register can be

system m address 4 digit (same by m hata 2030H, Hemdecimal manufacture)

loaded only by the OS by using special instruction. OS can change the value of the register but the user program cannot

Assigning an address to an instruction or data is known as address binding





A normal procedure is to select one of the processes in the input queue load that process into memory the process is executed it accesses instructions and data from memory then the process terminated and the memory space is declared available

Addresses can be represented in 3 ways
- Symbolic
- ~~Reat~~ Relocatable
- Absolute

Binding of instructions and data to memory address can be done in 3 steps

1. Compile Time: If you know at compile time where the process will reside in memory then absolute code is generated

2. Load Time: If it is not known at compile time where the process will reside in memory then the compiler must generate relocatable code

3. Execution Time: If the process can be moved during its execution from one memory segment

Scanned by TapScanner

to another then binding must be delayed untill run-time

Logical vs Physical Address

i.) An address generated by the CPU is commonly reffered to as logical address

ii.) An address generated by the memory unit that is the one loaded into the memory address location is reffered to as physical address

iii.) The Compile time and load time address binding methods generate identical logical and physical address

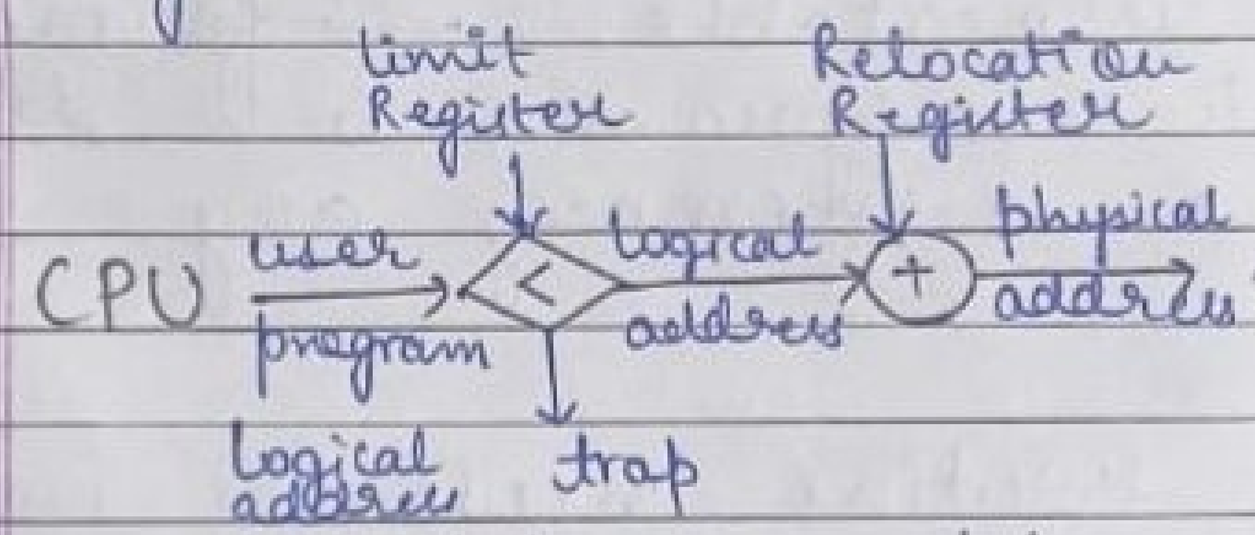
iv.) The execution time address binding scheme results in different logical and physical addresses. Logical address is also called virtual address. The set of all logical address generated by a program is called logical address space

v) The set of all physical addresses corresponding to these logical address is a physical address

Scanned by TapScanner

space. The run time mapping from virtual to physical address is done by a hardware device called the memory management unit

vii) To protect the memory allocated to the OS from being accessed by the user process two registers are used - Relocation register and limit register



Memory Management Unit (MMU) for Relocation & limit Register

Each logical address must be less than the limit register. MMU will add the logical address to the relocation register. It becomes the physical address and is then send to the memory

data bank → collection of data
database → collection of error free data

Dynamic Loading: It is necessary for a program and data to be in physical memory for the process to execute. The size of the process is limited to the size of physical memory for better memory utilization we can use dynamic loading in this a program is not loaded untill it is called all routines are kept on disk in a relocatable ~~ms~~ format the main program is loaded into the memory any is executed

When a routine needs to call another routine the calling routine first checks that whether the called routine has been loaded if it is not the relocatable linking loader is ~~called~~ called to load ~~th~~ routine into memory. and the address is updated and the control goes to the new routine (a part of a program)

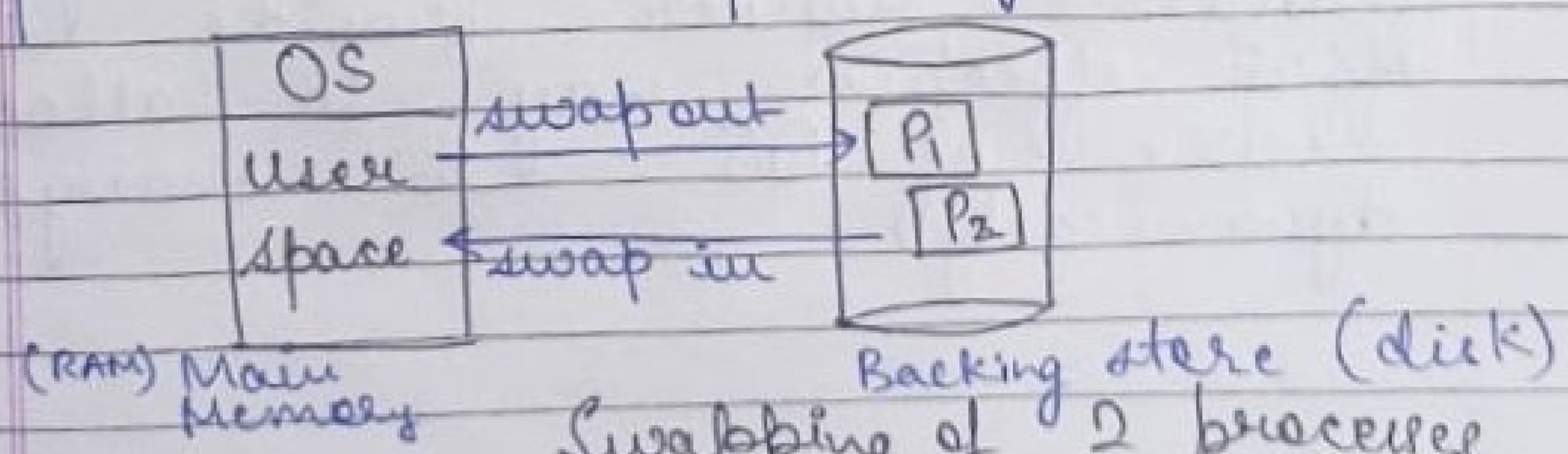
The advantage of dynamic loading is that an unused routine is never loaded which prevents error routines to enter the memory

Scanned by TapScanner

Total program size may be large but the portion that is loaded is much smaller.

Swapping : A process must be in memory to be executed. A process can be swapped temporarily out of memory to a backing store and then brought back into memory for continued execution

Swapping may occur because of the requirement from one memory ~~the~~ to another memory where higher priority process will be swapped in and lower priority process will be swapped out and vice versa. Swapping is done by memory manager it is also called as roll-in roll-out if we want to swap a process we must be sure that it is completly ideal never swap a process with pending I/O



Swapping of 2 processes

MMCBS

Backing store is commonly a fast disk and large enough to collect all the swapped out processes

Contiguous Memory Allocation: The main memory must accomodate both the OS and various users processes Therefore, allocation of main memory should be most efficient

The memory is divided into 2 partitions one for OS and second for the users process - The OS can be either low memory or high memory. Several user processes need to reside in memory. Therefore memory has to be made available. In contiguous memory allocation each process is contained in a single contiguous section of memory. Before allocation of memory mapping and protection is provided by the relocation register and limit register. Relocation register provide an ~~effi~~ effective way to allow the size of OS to change dynamically

8/22

Unit 2 (Part 1)

Processes: Process is a program in execution. A process will need resources such as CPU time, memory, files, and I/O devices. System consists of collection of processes it is an entity to which processors are assigned it can be called as job or task. A process includes the current activity as represented by contents of the registers

A process ~~th~~ contains data section which contains global variables and local variables along with functions ~~The process contains data section which contains global variable~~

A program is a passive entity such as a file containing a list of instruction stored on the disk called as executable file. Whereas a process is an active entity. A program becomes a process when an executable file is loaded into memory

Process is the smallest individually

Program - Passive
Process - Active

discrete (different)

Disk / Disc
↓ magnetism contain
↓ optism contain

schedulable entity consisting of code and data and featured by attributes and dynamic state

Four Principle event that cause processes to be created are as follows

→ System Intialize
→ User Request to create new process
→ Execution of ~~prog~~ process
→ Termination of process

The Process can be terminated due to the following conditions

- Normal exit
  After exit if any message comes
- Error exit
  error that cannot be solved then computer switches off
- Fatal error
- Killed by another process

Process states (conditions), a process goes through a series of discrete process states as a process executes it changes state. Each process may be in one of the following states

1. NEW : A process is being created
2. READY : The process is waiting to be assign to a processor
3. RUNNING : instructions are being executed
4. Blocked (waiting) : The process is waiting for some event to occur

Scanned by TapScanner

Program - Passive
Process - Active

discrete (different)

DUK / DUC
↓ magnetism ↓ optism

schedulable entity consisting of code and data and featured by attributes and dynamic state

Four Principle event that cause processes to be created are as follows

→ System Intialize
→ User Request to create new process
→ Execution of ~~prog~~ process
→ Termination of process

The Process can be terminated due to the following conditions

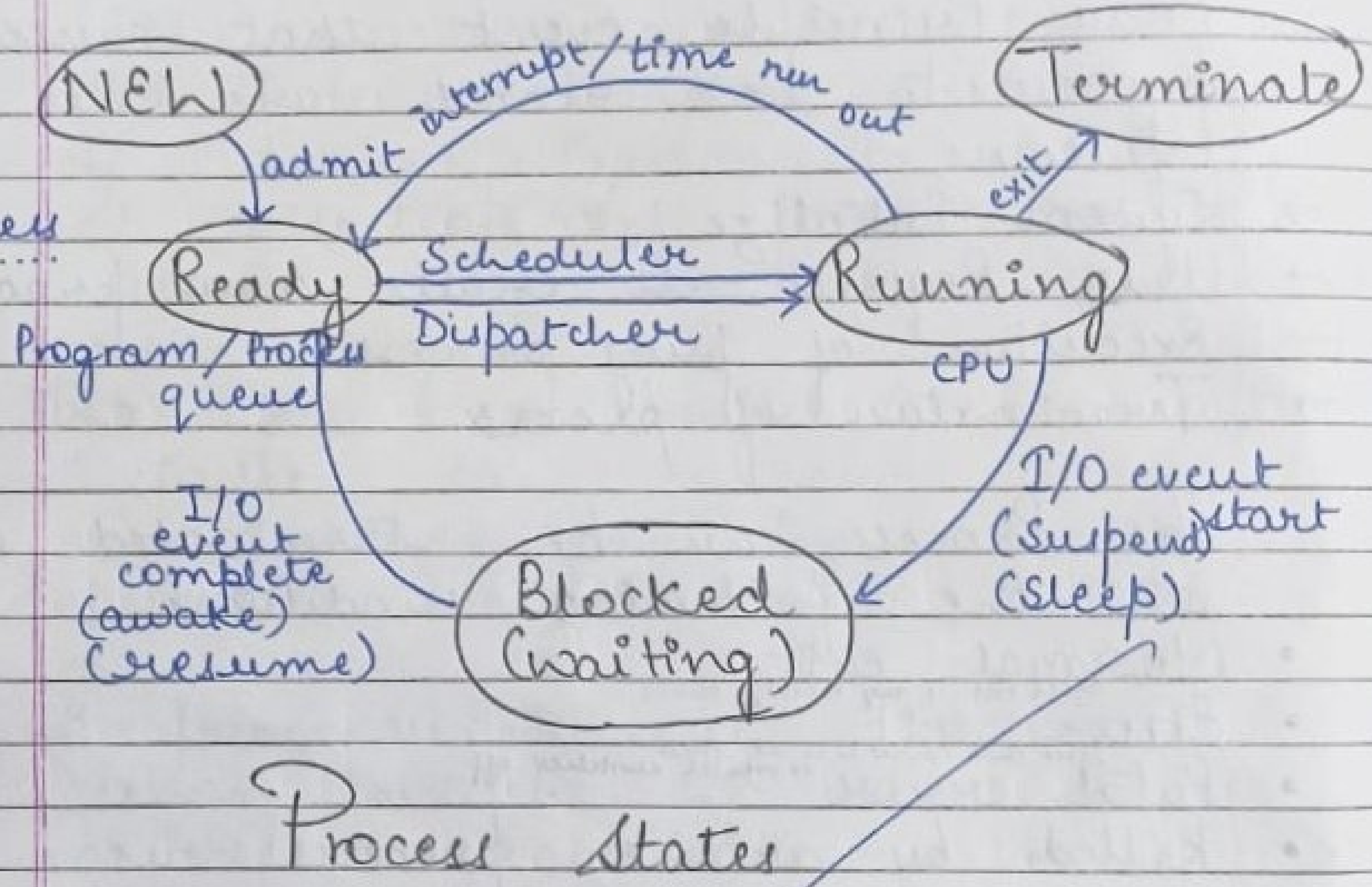
- Normal exit
  After exit if any message comes
- Error exit
  error that cannot be solved shell
- Fatal error
  computer switches off
- Killed by another process

Process states (conditions), a process goes through a series of discrete process states as a process executes it changes state. Each process may be in one of the following states

1. NEW: A process is being created
2. READY: The process is waiting to be assign to a processor
3. RUNNING: instructions are being executed
4. Blocked (waiting): The process is waiting for some event to occur

Scanned by TapScanner

5. Terminated: The process has finished execution



Process States

When a job is admitted to the system a corresponding process is created and normally is inserted at the back of the ready list. The process moves to the head of the ready list as the processes before it complete their term using the CPU

When the process reaches the head of the list and when the CPU becomes available the

process is given to the CPU and it is said to make a state transition from ready state to the running state

The assignment of the CPU to the first process on the ready list is called dispatching and is performed by a system entity called the dispatcher

dispatch(processname): ready → running

To prevent one process from monopolizing the system, the OS sets a hardware interrupting clock to run the process for specific time interval or time slice. If the process does not execute in the specified time interval the clock generates an interrupt the OS makes the previously running process ready and makes the new process running. If the

Time run out(process name): running → ready

Dispatch(process name): ready → running

If the running process waits for an input/output operation or any other event the process

blocks itself.

block(processname) : running → blocked

If the running process completes all the other events the process makes transition from the blocked state to the ready state

wakeup(processname): blocked → ready

⇒ Process Control Block (PCB)

Each process is represented in the OS like PCB or task control block or process descriptor. A PCB is a data structure containing certain important information about the process including:

→ Unique Identification of the process (PID), process name, parent process (the process that created a child process),

→ child process (a process that was created by the parent process), process priority, the process memory and allocated resources

Q. Register is a special purpose memory

i) Process state may be new, ready, running, waiting etc
ii) Program counter indicates the address of the next instruction to be executed for this process
iii) CPU registers include accumulators, index registers, stack pointers and general purpose registers
iv.) CPU scheduling information include the process priority, scheduling queues and any other scheduling parameters
v.) Memory Management Information:

This information may include register size, tables, depending on the memory system used by the OS

vi.) Accounting Information: This includes the amount of the CPU time used
vii) I/O status information: This includes the list of I/O devices allocated to the process

The PCB is a central store of information that allows the OS to locate all key information about a process. PCB is an entity that defines the

Scanned by TapScanner

process to the OS

PCB

Process no.
Process name
Process state
registers
Program counter
Files

Operation on Processes :

create a process, destroy a process, suspend a process, resume a process, change a process priority, block a process, wake up a process, dispatch a process, enable a process and name a process

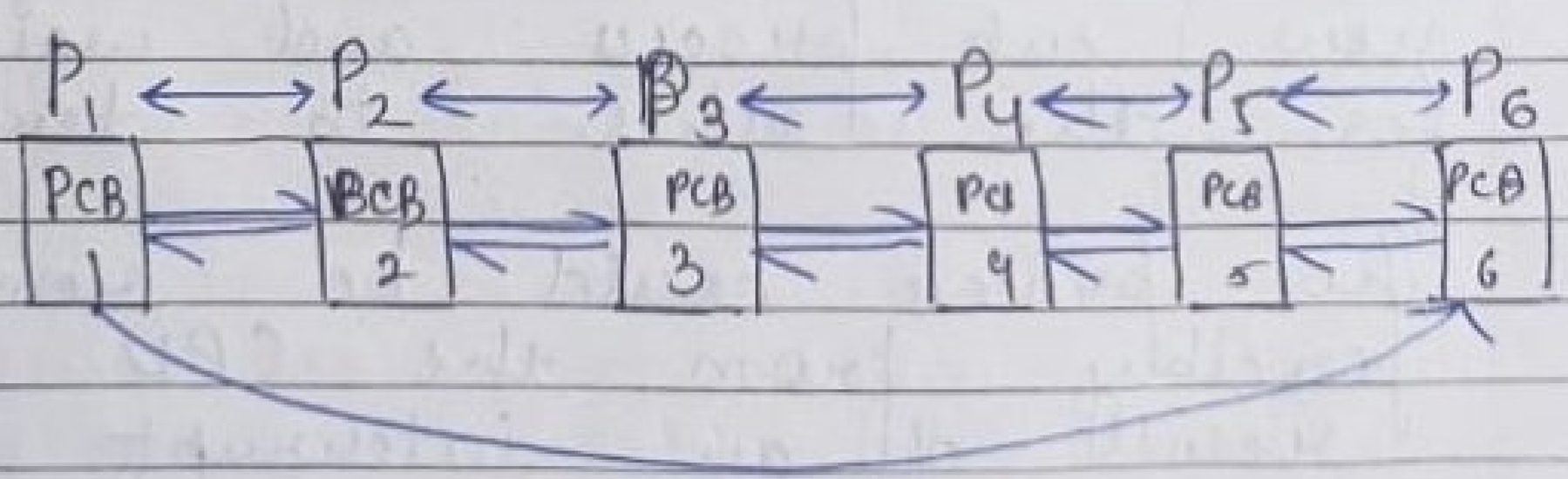
Process scheduling : The objective of multiprogramming is to have some process running at all times to maximize CPU utilization. The objective of timesharing is to switch the CPU among processes so frequently that users can interact with each program while it is running. To meet this objective the process

scheduler selects an available process from a set of processes for program execution on the CPU

Scheduling Queue : As process enters the system they are put into a job queue which consists of all process in the system

- The process that are residing in main memory and are ready and waiting to get executed are kept in a list called ready queue. This queue is generally stored as a linked list
- A ready queue header contains a link between the first and final PCBs in the queue

The common representation of process scheduling is queuing diagram



Scanned by TapScanner

Each rectangular box represents a queue. There are two types of queues - ready queue and set of devices

The circles represent the resources that serve the queues and arrows indicate the flow of processes in the system

A new process is initially put in the ready queue it waits until it is selected for execution or is dispatched once the process is allocated to the CPU and is executing one of the several events may occur

1. The process put issue an I/O request and then be placed in an I/O queue

2. The process could create a new sub process and wait for the other to terminate

3. The process could be removed forcibly from the CPU as a result of an interrupt and

Scanned by TapScanner

put back in the ready queue

Schedulers

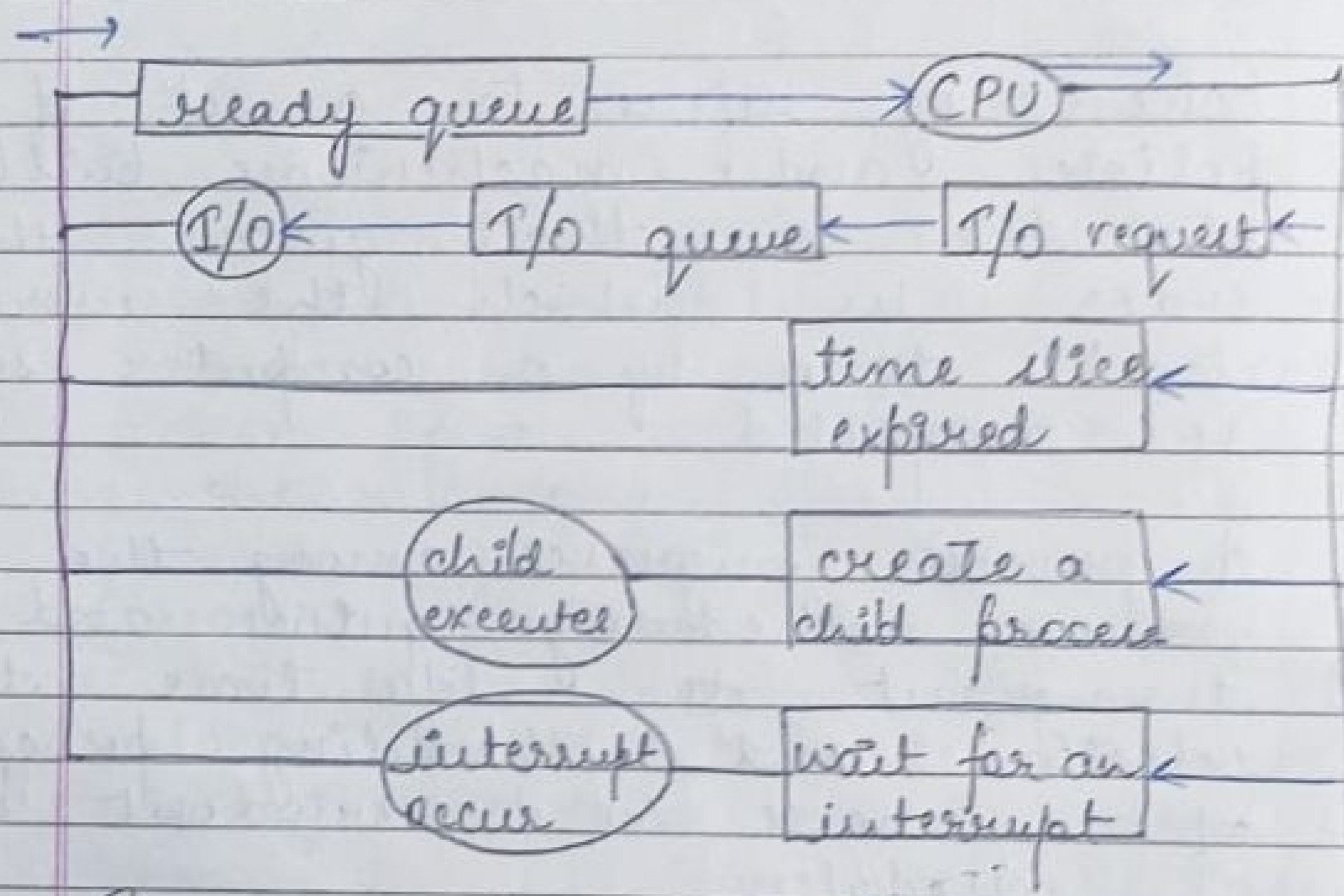
Scheduling refers to a set of policies and machenisms built in to the OS that govern the order in which the work to be done by a computer system is completed

A process migrates among the various scheduling queues and throughout its life time the selection and scheduling procedures of a process is carried out by a scheduler

A scheduler is an OS Module that selects the next Job to be admitted ~~in~~ to the system and next process to run the primary objective of scheduling is to optimize system performance

Scanned by TapScanner

Representation for Process scheduling



Types of Scheduler / Scheduling level

(i.) High level Scheduling / admission scheduling / long term scheduler

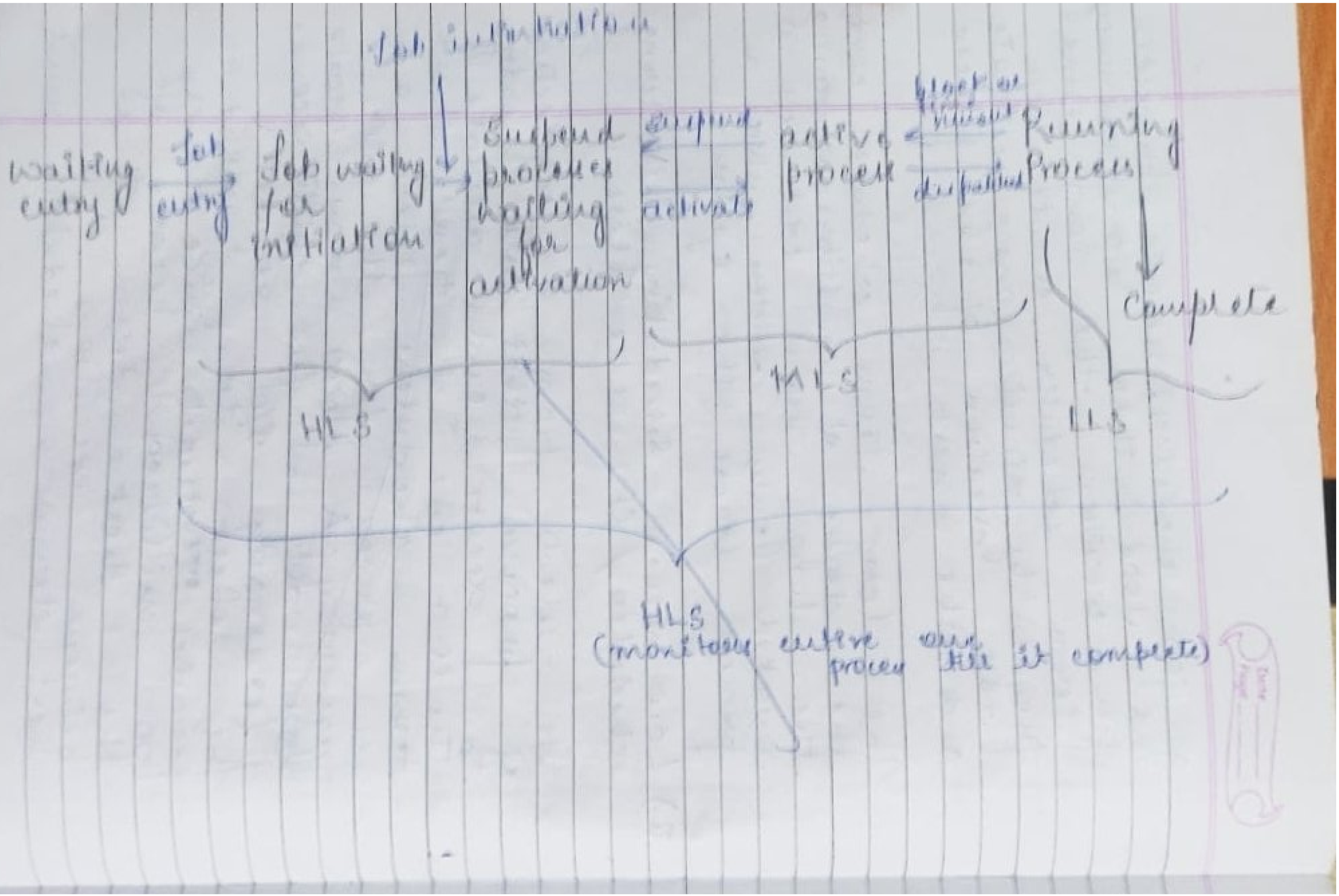
It works with the Batch queue and selects the next batch job to be executed batch is usually reserved for processor time, memory, special i/o devices, low priority programme that may be used to keep the system resources busy also called

a job scheduler, selecte processes and load them into memory for execution the long ~~time~~ term scheduler execute less frequently minutes may seperate the creation of one process and the next

The long term scheduler control the degree of multiprogramming the long term scheduler is invoked only ~~written~~ when a process leaves the system

ii.) Low Level Scheduling / shortterm scheduler / CPU scheduler

Its selects from a collection of processes that are ready to execute and allocate the CPU to one of them. short term scheduler select a new process for the CPU frequently. A process may execute for only a few milli-seconds for any I/O request. The short term scheduler executes ~~once~~ atleast once every 100 millisecond because of the short term time between executions. The short term scheduler must



Scanned by TapScanner

be fast, its main objective is to maximize system performance it is in charge of ready to running state transition. The OS uses the short term scheduler to determine whether another process should be scheduled for execution.

(iii) Intermediate level Scheduling and Medium term scheduler

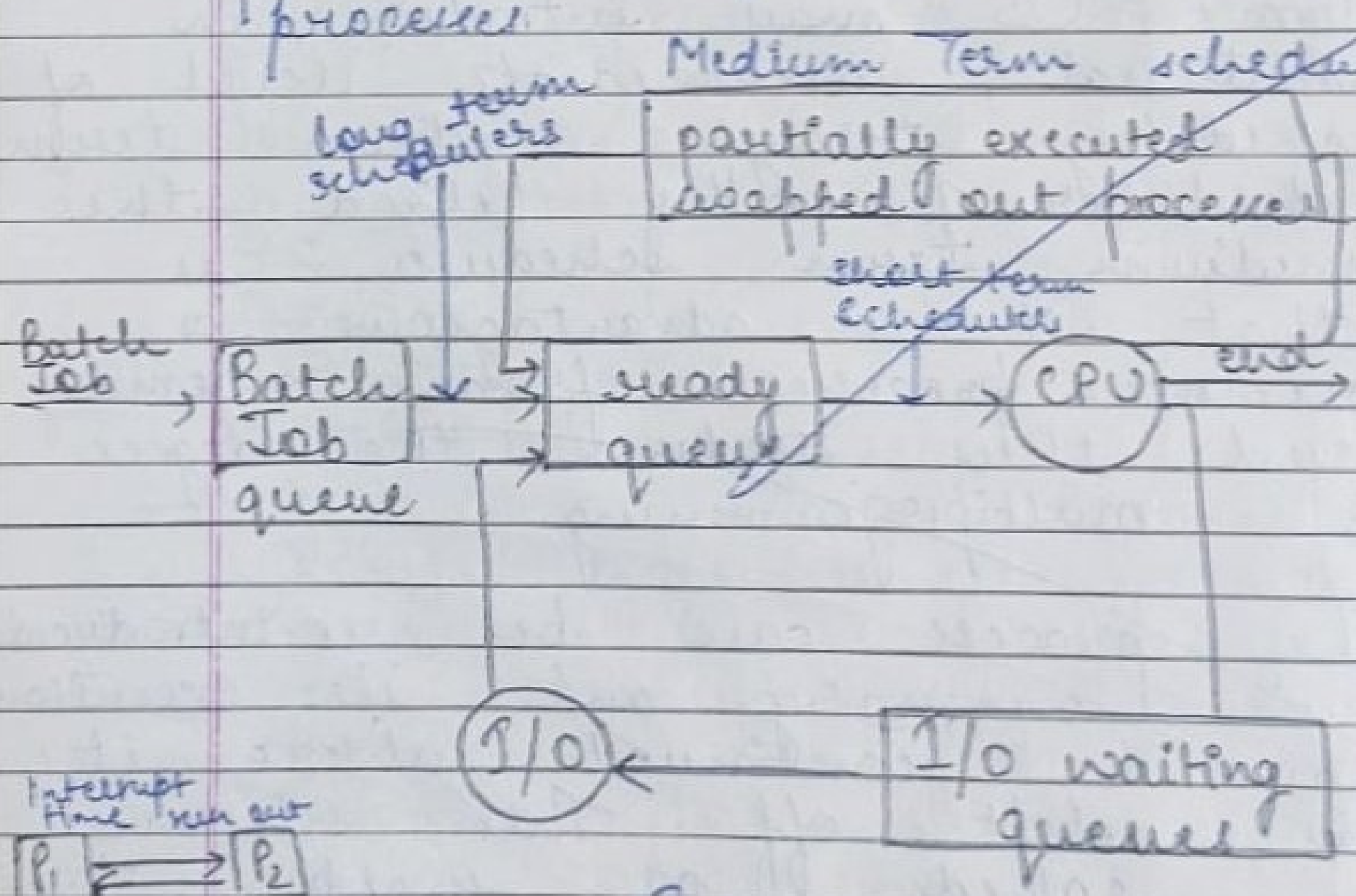
Some OS may introduce an additional, intermediate level of scheduling through medium term scheduler. The key behind the medium term scheduler is that it is advantageous to remove processes from memory and thus reduce the degree of multiprogramming.

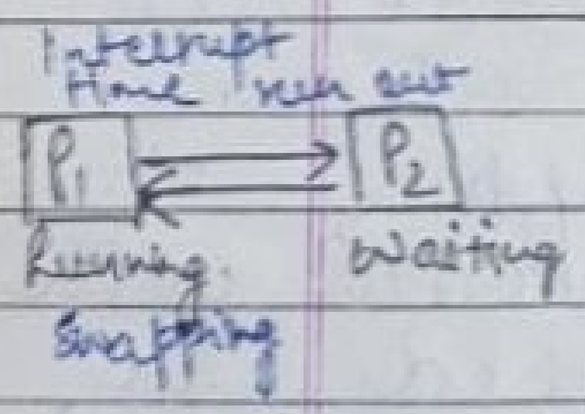
The process can be re-introduced into memory and its execution can be continued where it is left off. This scheme is called as swapping. The process is swapped out with another process swapped in, it is done by medium term scheduler

Standard form
↑
Project
↑
s/w
↑
Program
↑
Process
↑
Job
↑
Task

Swapping is necessary to improve the process mix or a change in memory requirement. Swapping can also be called as rolling. In terms of state transition the medium term scheduler controls the suspended processes

* Process Mix : It is a combination of I/O bound processes and CPU bound processes





Schedulers

Scanned by TapScanner

Context Switch

When an interrupt occurs the system needs to save the current context of the process running on the CPU so that it can restore that context when its processing is done, essentially suspending the process and then resuming it. The context is represented in the process control block of the process. It includes the value of the CPU registers, the process state and memory management information.

Switching the CPU to another process requires performing a state save of the current process and a state restore of a different process. This task is known as Context switch

Scanned by TapScanner

1/22

Unit 2 (Part I)

⇒ Process Scheduling / CPU Scheduling

In a single processor system only one process can run at a time other process must wait untill the CPU is free and can be re-scheduled. The objective of multi-programming is to have some process running at all times to maximize CPU utilization

A Process is executed untill it must wait typically for the completion of some I/O request. In a simple computer system. The CPU just state ideal. All this time waiting time is wasted. With multiprogramming we use this kind several processes are kept in memory at one time. When one process has to wait another process is given to the CPU. This pattern continues every time one process has to weight

CPU is the primary computer resource. The fundamental function of the OS is scheduling. Almost all computer resources are scheduled before use

CPU - I/O Burst Cycle : The success of CPU scheduling depends on an observed property of processes. Process execution consists of a cycle of CPU execution and I/O wait. Processes alternate b/w these two states. Process execution begins with a CPU burst i.e. followed by an I/O burst, which is followed by another CPU burst then another I/O burst and then the final CPU burst ends with a system to terminate the execution. The duration of CPU burst have been measured in milliseconds. They vary from process to process and from computer to computer

CPU Scheduler : Whenever the CPU becomes ideal the OS must select one of the processes in the ready queue to be executed. The

Preempt – जगह ले लेना

selection process is carried out by a short-term scheduler or CPU scheduler. The scheduler selects a process from the processes in the memory that are ready to execute and allocates the CPU to that process. ~~The ready~~

Q. The ready queue can be implemented by first in first out, priority queue and a linked list. All processes in the ready queue are lined up waiting for a chance to run on the CPU

Preemptive & Non-preemptive Scheduling

(i) A scheduling discipline is non-preemptive if once a process has been given to the CPU, the CPU cannot be taken away from process

(ii) A scheduling discipline is ~~preeptiumump~~ tive if the CPU can be taken away

(iii) Preemptive scheduling is usefull

Scanned by TapScanner

8/9/22

in system in which high priority processes require rapid attention.

(iv.) In time sharing system preemptive scheduling is important to get proper response time

(v.) To make preemption affective many processes must be kept in main storage so that the next process is normally ready for the CPU when it becomes available

(vi.) Keeping non-running programs in main storage also involves overhead

(vii.) In non-preemptive systems short job are made to wait for longer jobs

(viii.) Response time are more predictable because incomming high priority job cannot displace waiting job

(ix.) Under Non-preemptive scheduling once the CPU has been allocated to a process the process keeps the CPU untill it

releases the CPU. Either by terminating or by switching to the waiting state

→ Dispatcher: A component involved in the CPU scheduling function is the dispatcher. The dispatcher is the module that gives control to the CPU to the process selected by the short term scheduler.

The Dispatcher should be fast as possible since it is invoked during every process switch. The time it takes for the dispatcher to stop one process and start another process running is known as dispatch latency

Scheduling Criteria: To maximize System performance

1. CPU utilization: We want to keep the CPU as busy as possible. CPU utilization can range from 0 to 100 %. In real system it should range from 40% to 90%.

Scanned by TapScanner

2. Throughput : If the CPU is busy executing processes then work is being done one measure of work is the no of processes that are completed by unit time called throughput. For long processes this rate may be one process per hour. For short transactions it may be ten processes per second

3. Turn around time : From the point of view of a particular process the important criteria is how long it takes to execute the process. The interval from the time of submission of a process to the time of completion is the turn-around time. Turnaround time is the sum of time interval spend waiting to get into memory, waiting in the ready queue, executing on the CPU and doing I/O operations.

4. Waiting time : The CPU scheduling algorithm does not affect the amount of time during which a process

Scanned by TapScanner

executes or does I/O operation it affects only the amount of time that a process spends waiting in the ready queue. Waiting time is the sum of time intervals waiting in the ready queue

Response Time: In an interactive system, the turn around time may not be the best criteria often a process can produced some output very early, may take some time or go for a long session before giving output to the user

Response Time is the measure of the time from the submission of a request until the first response is produced. It is the time it takes to start responding

Inter-process communication: This allows data and information to be exchanged between two processes through sharing of memory and passing of message

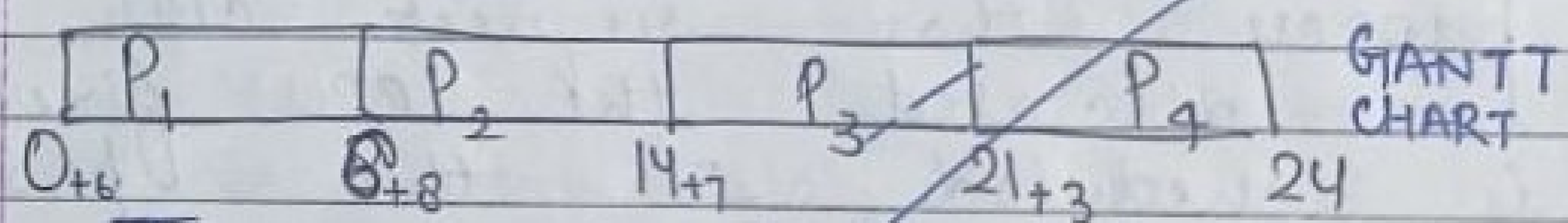
Scheduling Algorithms: CPU scheduling deals with the problem of deciding which of the processes in the ready queue to be allocated to the CPU. There are certain scheduling algorithms

⇒ 1 First Come First Serve Scheduling: This is the simplest CPU scheduling algorithm with this scheme the process that requests CPU first is allocated the CPU first. It is scheduled to the processes in the order of arrival at the ready queue.

The processes are queued up by sequence (sub component of scheduler) in the order of their arrival at the tail of the queue whenever the CPU is available the dispatcher removes a process from the head of the queue and schedules it to the CPU. The implementation of FCFS policy (FIFO) is easily managed with a FIFO (First In First Out) queue when a process enters the ready queue its PCB is linked onto the tail of the queue. When the CPU

is free it is allocated to the process at the head of the queue the running process is then removed from the queue. The code of FCFS scheduling is simple to write and understand.

Process : P_1 P_2 P_3 P_4
Burst Time : 6 8 7 3



GANTT CHART

Turnaround time of P_1 = 0+6 = 6 ms
" " P_2 = 6+8 = 14 ms
" " P_3 = 14+7 = 21 ms
" " P_4 = 21+3 = 24 ms

Average Turnaround Time =
= (6+14+21+24)/4
= 16.25 ms

Waiting Time of P_1 = 0 ms
" " P_2 = 0+6 = 6 ms
" " P_3 = 6+8 = 14 ms
P_4 = 14+7 = 21 ms

Average Waiting Time = (0+6+14+21)/4
= 10.25 ms

Scanned by TapScanner

⇒ 2. Shortest Job First Scheduling (SJF): This algorithm associates with each process the length of the no. of CPU burst needed by it for the execution. The process having the smallest next CPU burst is allocated to the CPU whenever it is available next. In case, there is a tie FCFS scheme is employed to break the tie. Whenever the CPU is available it is assigned to the process that has the smallest next CPU burst. It can also be called as Shortest Next CPU burst Algorithm because scheduling depends on the length of the next CPU burst of a process rather than its total length. SJF Scheduling algorithm is optimal in that it gives the minimum avg waiting time for a given set of processes

Moving a short process before a long one decrease the waiting time of the short process more than it increases the waiting time of the long process. Avg waiting time decreases. This scheduling is used

Scanned by TapScanner

in long term scheduling

Process :	P1	P2	P3	P4
Burst Time :	6	8	7	3

GANTT CHART

P4	P1	P3	P2
0 +3	3 +6	9 +8	16 +7	27

Turn Around Time of $P_1 = 3+6 = 9ms$
" $P_2 = 16+8 = 24ms$
" $P_3 = 9+7 = 16ms$
" $P_4 = 0+3 = 3ms$

Average TT $= (9+24+16+3)/4$
$= 52/4 = 13ms$

Waiting Time of $P1 = 0+3 = 3ms$
" $P2 = 9+7 = 16ms$
$P3 = 3+6 = 9ms$
$P4 = 0ms$

Avg Waiting Time $= (3+16+9+0)/4$
$= 28/4$
$= 7ms$

Scanned by TapScanner

3. Multi-level Queue Scheduling:

Algorithms has been created for situations in which processes are easily classified into different groups. Processes can be divided into ~~FORE~~ foreground (interactive process) and background (batch process) they have different scheduling needs foreground processes may have priority over background processes.

A multilevel queue scheduling algorithm partitions the ready queue into several seperate queues. The processes are permanently assigned to a one queue based on some property of the process such as memory size, process priority or process type. Each queue has its own scheduling algorithm. Seperate queues might be used for foreground and background processes. The foreground process might be scheduled by an shortest Job first scheduling and background process by FCFS. There must be process proper scheduling between the queues

According to the highest priority the five queues are given below:

1. System Processes
2. Interactive Processes
3. Interactive Editing processes
4. Batch Processes
5. Student Processes

Each queue has absolute priority over its lower priority queues i.e. no process in the batch queue could run unless the queues for system processes, interactive processes, interactive editing processes where all empty

If any high priority process arrives, the low priority process will be pointed preemptive

Another possibility is to time slice among the queues

1. Multilevel feedback Queue Scheduling

This type of scheduling allows a process to move b/w queues

The idea is to seperate processes according to the features of their CPU burst (execution time). If a process uses two much CPU time it will be moved to a lower priority queue. In addition a process that waits too long in a lower priority queue may be moved to a higher one. This form of scheduling prevents Starvation

Multilevel feedback queue scheduling is defined by the following parameters

- no of queues
- scheduling algorithm for each queue
- the method used to determine when to upgrade a process to a higher priority queue
- The method used to determine when to degrade a process to a lower priority queue
- The method used to determine which queue a process will enter when that process needs service.

Scanned by TapScanner

5. Priority Scheduling: This algorithm associates a priority with each process on some predetermined basis. The CPU is allocated to the process having the highest priority equal. Priority processes are scheduled in FCFS order. SJF is a special case of priority scheduling where the process with smallest CPU burst requirement having highest priority. low number represent high priority.

Priority scheduling can either be preemptive or non-preemptive when a process arrives at the ready queue its priority is compared with the priority of the currently running process. A preemptive priority scheduling algorithm will preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process. A non-preemptive priority scheduling algorithm will simply put the new process at the head of ready queue

Scanned by TapScanner

A major problem with priority scheduling is indefinite blocking or starvation

Process:	P_1	P_2	P_3	P_4	P_5
Burst time:	10	1	2	1	5
Priority:	3	1	4	5	2

GANTT CHART

P2	P5	P1	P3	P4

0 1 6 16 18 19

Turnaround Time of P_1 = 16 ms
Turnaround Time of P_2 = 1 ms
Turnaround Time of P3 = 18ms
Turnaround Time of P4 = 19ms
Turnaround Time of P5 = 6 ms

Average T.T. = (16+1+18+19+6)/5
= 60/5 = 12 ms

Waiting Time of P_1 = 1+5 = 6 ms.
Waiting Time of P_2 = 0 ms
Waiting Time of P3 = 16ms
Waiting Time of P4 = 18ms
Waiting Time of P5 = 1ms

Average Waiting Time = (6+0+16+18+1)/5
= 8.2 ms

Scanned by TapScanner

⇒ 6. Round Robin Scheduling : It designed for time sharing system it is similar to FCFS scheduling but preemption is added to enable the system to switch b/w processes. A small unit of time is called time slice. The ready queue is lined up in a circular manner. The CPU scheduler goes around the ready queue allocating the CPU to each process for a time interval.

A context switching takes place whenever the running process crosses its allocated time slice at this time the running process is queued up at the tail of the queue if it has not finished execution the next process at the head of the ready queue is allocated ~~that~~ the CPU next

CPU timeslice = 4ms

Process :	P_1	P_2	P_3
Burst time :	24	3	3

Scanned by TapScanner

10/22

GANTT chart

P1	P2	P3	P1	P1	P1	P1	P1

0 4 7 10 14 18 22 26 30

Turnaround Time of P1 = 30ms
Turnaround Time of P2 = 7ms
Turnaround Time of P3 = 10ms

Average Turnaround Time = (30+7+10)/3
= 47/3
= 15.6ms

Waiting time of P1 = 10-4 = 6ms
Waiting time of P2 = 0+4 = 4ms
Waiting time of P3 = 7ms

Average Waiting time = (6+4+7)/3
= 5.6ms

⇒ 7. SRTF (Shortest Remaining Time First) scheduling: The choice arises when a new process arrives at the ready queue while a previous process is still executing the next CPU burst of the newly arrived process may be shorter than what is left of the currently executing process. Preemptive shortest job first scheduling is sometimes called shortest remaining time first scheduling

Scanned by TapScanner

Process	P_1	P_2	P_3	P_4
Arrival time	0	1	2	3
Burst time	8	4	9	5

P1	P2	P4	P1	P3

0 1 5 10 17 26

GANTT chart

Turnaround Time of $P_1 = 17$ ms
" " of $P2 = 1+4 = 5$ ms
" " " $P3 = 17+9 = 26$ ms
" " " $P4 = 5+5 = 10$ ms

Average Turnaround Time $= (17+5+26+10)/4$
$= 58/4$
$= 14.5$ ms

Waiting Time of $P1 = 10-1 = 9$ ms
" " " $P2 = 1-1 = 0$ ms
" " " $P3 = 17-2 = 15$ ms
" " " $P4 = 5-3 = 2$ ms

Average Waiting Time $= (9+0+15+2)/4$
$= 26/4$
$= 6.5$ ms

Scanned by TapScanner

Multiple Processor Scheduling:

If multiple CPU are available load sharing becomes possible but the scheduling problem becomes correspondingly more complex. The systems in which processors are identical in terms of their functions we can then use any available processor to run any process in the queue

One approach to multiple processor scheduling is using asymmetric multiprocessing and symmetric multiprocessing all scheduling decisions, I/O processing and other system activities handled by a single processor which works as master server

Processor Affinity (Liking): In symmetric multi-processing system try to avoid migration of processes from one processor to another and attempt to keep a process running on the same processor this is known as processor affinity i.e. the process has an affinity for the processor on which it is currently running

Scanned by TapScanner

- Soft affinity : When an operating system has a policy of attempting to keep a process running on the same processor but not promising to do so this situation is called as soft affinity
- Hard affinity : A policy of attempting to keep a process running on the same processor and stick to it is called as hard affinity

Load Balancing : Attempts to keep the workload evenly distributed across all processors. Two types of load balancing are as follows:

1. Push Migration : It checks the load on each processor and if it finds imbalance it evenly distributes the load by pushing the process from overloaded processor to less busy processors
2. Pull Migration : occurs when a less busy processor pulls a waiting task from a busy processor

Scanned by TapScanner

6/9/22

Unit 3

Deadlocks : In a multiprogramming environment several process may compete for a finite no. of resources. A process requests resources, if the resources are not available at that time the process enters a waiting state. Sometimes a waiting process is never again able to change state because the resources it has requested are held by other waiting processes this situation is called a deadlock

A deadlock is a situation where a group of processes are permanently blocked as a result of each process having acquired the resources needed for its completion and waiting for release of the remaining resources held by other processes to proceed

Deadlocks can occur in concurrent environment as a result of uncontrol granting of system resources to requesting processes

Scanned by TapScanner

9/22

System Model: A system consists of finite no. of resources to be distributed among a no. of competeing processes. The resources are partitioned into several types of resources like - CPU, files, I/O devices, memory etc

A process must request a resource before using it and must release the resource after using it. A process may request as many resources to carry out its task. The no. of resources may not exceed the total no. of resources available in the system. For ex: a process cannot request three printer if the system has two printers only

Under normal mode of operation a process may utilise a resource in the following sequence

(i.) Request: The process request the resource if the request cannot be granted immediately. Then the requesting process must wait until it can acquire the resource

Scanned by TapScanner

ii) Use: The process can request the resource.

iii) Release: The process releases the resource.

Necessary Conditions: when a deadlock situation can arise in a system.

(i) Mutual Exclusion: At least one resource must be held in a non sharable mode. i.e. only one process at a time can use a resource. If another process requests that resource, the requesting process must be delay until the resource has been released.

(ii) Hold and Wait: A process must be holding atleast one resource and waiting to acquire additional resources that are currently being held by other processes.

(iii) No preemption: Resources cannot be preempted. i.e. a resource can be released only voluntarily by the process holding it. After that process has completed its task.

Scanned by TapScanner

(iv) Circular Wait: A set (P_0, P_1 .... P_n) of waiting processes must exist such that P_0 is waiting for a resource held by P_1, P_1 is waiting for a resource held by P_2 and P_n is waiting for resource held by P_0

Resource Allocation Graph: Deadlock can be describe more precisely in terms of a directed graph called a system resource allocation graph

This graph consists of a set of vertices (V) and set of edges (E) the set of vertices is partitioned into two different types of nodes

$P = \{P_1, P_2 \dots P_n\}$ the set consisting of all the active processes in the system

$R = \{R_1, R_2 \dots R_m\}$ the set consisting of all resource types in the system

A directed edge from process P_i to resource type R_j is denoted by $P_i \rightarrow R_j$ it signifies

Scanned by TapScanner

that process P_i has requested an instance of resource type R_j and is currently waiting for that resource. It is called as request edge

A directed edge from R_j resource type to process P_i is denoted by $R_j \rightarrow P_i$. It signifies that an instance of resource type R_j has been allocated to process P_i it is called as assignment edge. ~~Graph~~

Graphically process P_i is represented as a circle and each resource type R_j as a rectangle. Since a resource type R_j may have more than one instance. Instances are represented by dots within the rectangle

Request edge points to only the rectangle R_j whereas an assignment edge must also designate one of the dots in the rectangle

When process P_i requests an instance of resource type R_j a requested edge is inserted in the resource allocation graph

Scanned by TapScanner

When the request is fulfilled the request edge is instantaneously transform to an assignment edge

When the process ~~know~~ no longer needs access to the resource it releases the resource as a result assignment edge is deleted

A resource allocation graph depicts the following situations

- The set P,R,E
  - → $P = \{P_1, P_2, P_3\}$
  - → $R = \{R_1, R_2, R_3, R_4\}$
  - → $E = \{P_1 \rightarrow R_1, P_2 \rightarrow R_3, R_1 \rightarrow P_2, R_2 \rightarrow P_2, R_2 \rightarrow P_1, R_3 \rightarrow P_3\}$

Resource Instances : One instance of resource type R_1

(ii) Two instances for resource type R_2

(iii) One instance of resource type R_3

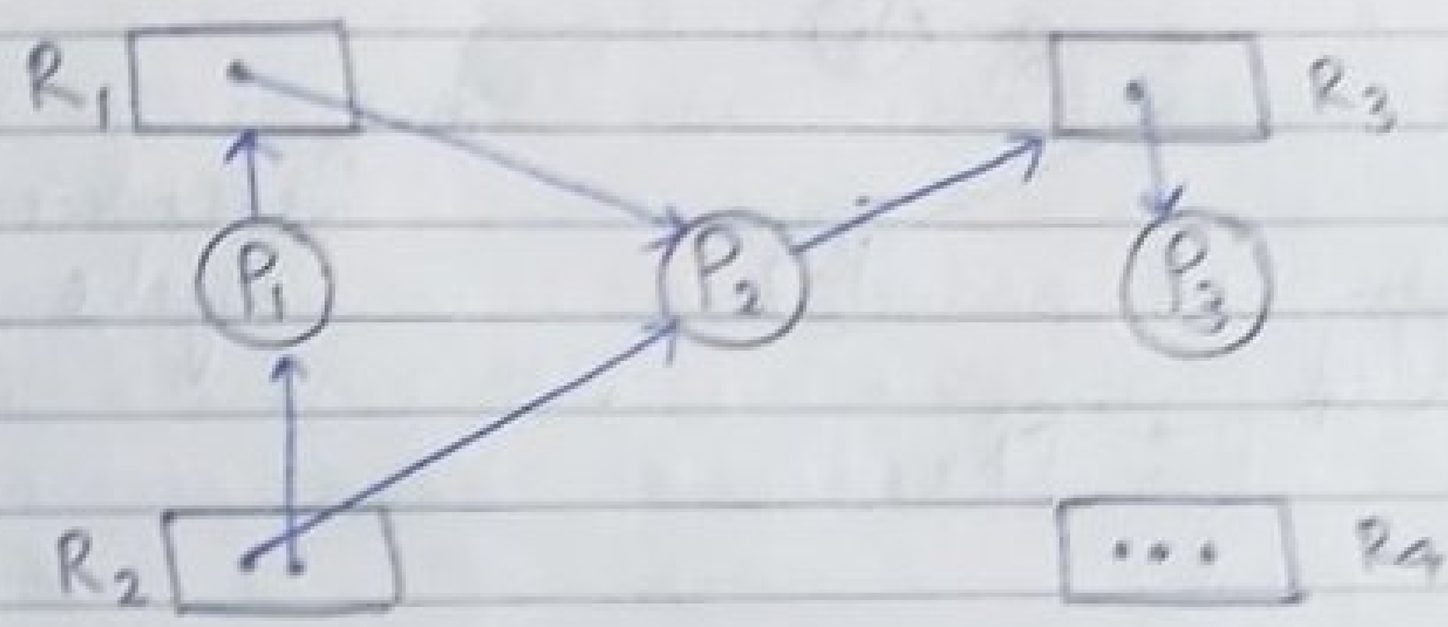
(iv) Three instances of resource type R_4

~~(v)~~ Process States: Process P_1 is holding an instance of resource type R_2 and is waiting for an instance of resource type R_1

Scanned by TapScanner

(ii) Process P_2 is holding an instance of resource R_1 and an instance of R_2 and is waiting for an instance of R_3

(iii) Process P_3 is holding an instance of R_3



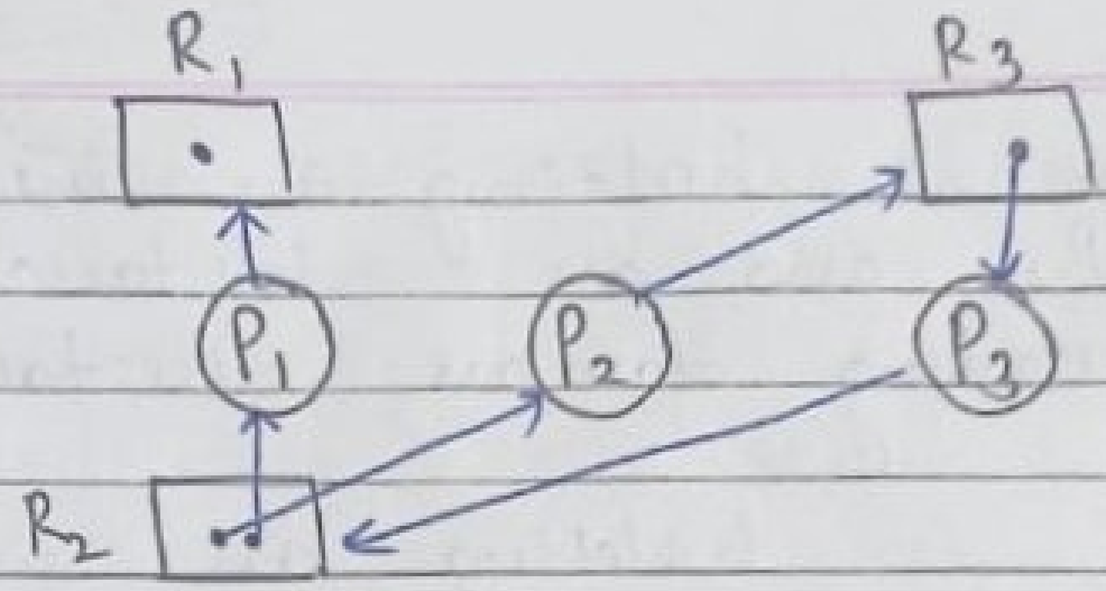
Resource Allocation Graph

If the graph contains no cycles then no process in the system is deadlocked. If the graph does contain a cycle then the deadlock may exist

*1) If the resource type has exactly one instance then a cycle implies that a deadlock has occurred. Each process involved in the cycle is deadlocked

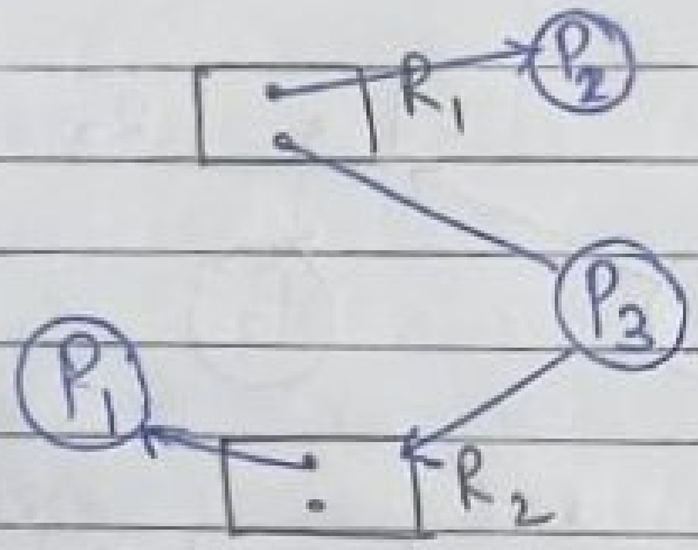
*2) If each resource type has several instances then a cycle does not necessarily imply that a deadlock has occurred

(i)



Resource Allocation Graph
Cycle, deadlock

(ii)



Resource Allocation
cycle, no deadlock

Methods of Handling Deadlocks

We can deal with the deadlock problem in one of the three ways

(i) We can use the a protocol to prevent (रोकना) or avoid (बचना) deadlocks ensuring that the system will never enter a deadlock state

(ii) We can allow the system to enter the deadlock state, detect it and recover from it

(iii) We can ignore (नजरंदाज करना) the problem all together and pretend that the deadlock never occur in

the system

To ensure that deadlocks never occur the system can use either a deadlock prevention or a deadlock ~~avoidation~~ avoidance method

Deadlock avoidance requires that OS should be given advance additional information concerning which resource a process will request so the arrangements can be done before hand

If the system does not follow deadlock prevention or deadlock avoidance then deadlock situation may arise

In the absence of detection and recovery of deadlocks. The system may be in a deadlock state. And if there is no solⁿ the system will stop functioning and will need to be restarted manually if the deadlock is not detected then the system performance will decrease because resources are busy with the processes

Deadlocks Prevention: By ensuring that atleast one of

Scanned by TapScanner

The following conditions cannot hold. We can prevent the occurance of a deadlock

1. Mutual Exclusion : The mutual exclusion condition must ~~for~~ hold for non-sharable resources. for example: a printer cannot be simultaneously shared by several processes. Sharable resources do not require mutually exclusive access and thus cannot be involved in a deadlock. Read only file is a good example of a sharable resource

2. Hold & Wait : To ensure that hold & wait condition never occure in the system we must gurantee that whenever a process requests a resource it does not hold any other resource. One protocol that can be used require each process to request and be allocated all its resources before it begin execution for example: A process copies data from DVD to a file, sorts the file and then prints the results with ~~th~~ a printer if all the resources must be requested at the begining of the process than the process must

initially hold the resources for ~~it~~ its entire execution even though it needs the printer only at the end

Another protocol allows a process to request only the DVD then the file and then the printer one by one. After each and every use of resource the concerned resource is released.

3. <u>No - Preemption</u>: A third necessary condition for deadlocks is that there should be no preemption of resources that have already been allocated to ensure that this condition does not hold the following protocol is used

(i.) If a process is holding some resources and requests another resource that are itself allocated then all resources the processes currently holding are preempted. These resources are added to the list of resources for which the process is waiting. The process will be restarted only when it can regain its old resources as well as the new ones that it is

requesting

(iii) If a process requests some resources we first check whether they are available. If they are available, we allocate them. If they are not available, we check whether they allocated to some other process i.e. waiting for additional ~~waiting~~ resources. If so, we preempt the desired resources from the waiting process and allocate them to the requesting process

4. Circular Wait : The fourth and final condition for deadlock is the circular wait condition. One way to ensure that this condition never holds is to impose a total ordering of all resource types and to require that each process requests resources in an increasing order

Deadlock Recovery : When has deadlock occured one possibility is to deal with the deadlock manually

There are two methods for deadlock

6/10/22

recovery:

1. Process Termination: To eliminate deadlocks by aborting a process we use one of the two methods in both the methods the system re-claims all resources allocated to the terminated process

(a.) Abort All Deadlocked Processes: This method will break the deadlock cycle but at great expense. The deadlocked processes may have computed for a long time and the result of these partial computations must be discarded and probably will have to be re-computed later

(b.) Abort One Process at a Time: Untill the deadlock cycle is eliminated. This method incurs less cost since after each process is aborted. A deadlock detection algorithm must be invoked (आगु (प्रश्न)) to determine whether any process is still deadlocked

Aborting a process may not be easy

if the process is in the middle of updating a file, terminating it will leave that file in an incorrect state. We should abort those processes whose termination will incur minimum cost

4. The factors that affect which process is choosen to abort :

(i) What is the priority of process ?

(ii) How long the process has computed and how much longer the process will compute before completing its task.

(iii) How many and what type of resources the process had used

(iv) How many more resources the process needs in order to complete.

(v) How many processes will need to be terminated ?

(vi) Whether a process is interactive or not ?

2. Resource Preemption : To eliminate deadlocks using resource preemption we preempt some resources from processes and give these resources to the other processes until the deadlock cycle is not broken

Scanned by TapScanner

· To recover from deadlock the following three issues need to be solved

(a.) Selecting a victim: We must determine the order of preemption to minimize cost it includes the no. of resources a deadlock process is holding and the amount of time the process has thus far consumed during its execution

(b.) Roll back: If the resource is preempted from a process · The process cannot continue its normal execution. Since it is missing some resource. It must be rolled back to some safe state and restart it from that state

(c.) Starvation: To ensure that starvation will not occur pick up a victim process only a finite no. of times

Deadlock Avoidance: A method for ~~de~~ avoiding deadlocks is to require additional information about how resources are to be requested · For ex: in a system there is one

0/22

tape drive and printer. The system need to know that process P will request first the tape driver and then the printer whereas process Q will request first the printer and then the tape drive. With this knowledge the system may avoid deadlock.

The system consider the resources available currently allocated to each process, future requests and release of each process. The simplest and most useful model requires that each process declare the maximum no. of resources of each type it may need. It is possible to construct an algorithm that the system will never enter a deadlock state.

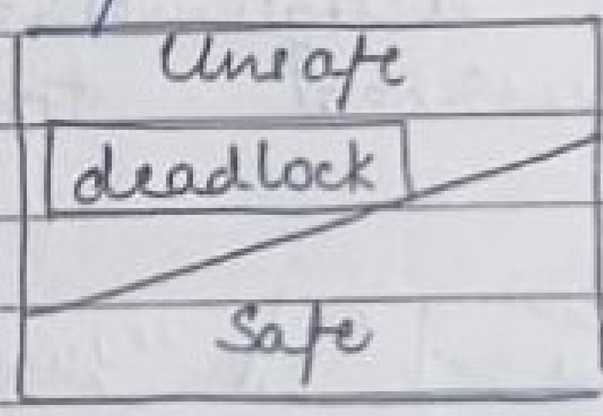
A deadlock avoidance algorithm dynamically examines the resource allocation state to ensure that a circular wait condition can never exist. A resource allocation state is defined by ~~no~~ the no. of available and allocated resources and maximum demands of the process

Scanned by TapScanner

Safe State : A state is safe if the system can allocate resources to each process in some order and still avoid a deadlock. A system is in a safe state only if there exits a safe sequence

A sequence of processes is a safe sequence for the current allocation state if for each process the resource request is made and is satisfied by the currently available resources and the resources held by previous process

A safe state is not a deadlock state. A deadlock state is an unsafe state not all unsafe states are deadlocke an unsafe state may lead to a deadlock. In an unsafe state the OS cannot prevent processes from requesting resources in such a way that a deadlock occurs



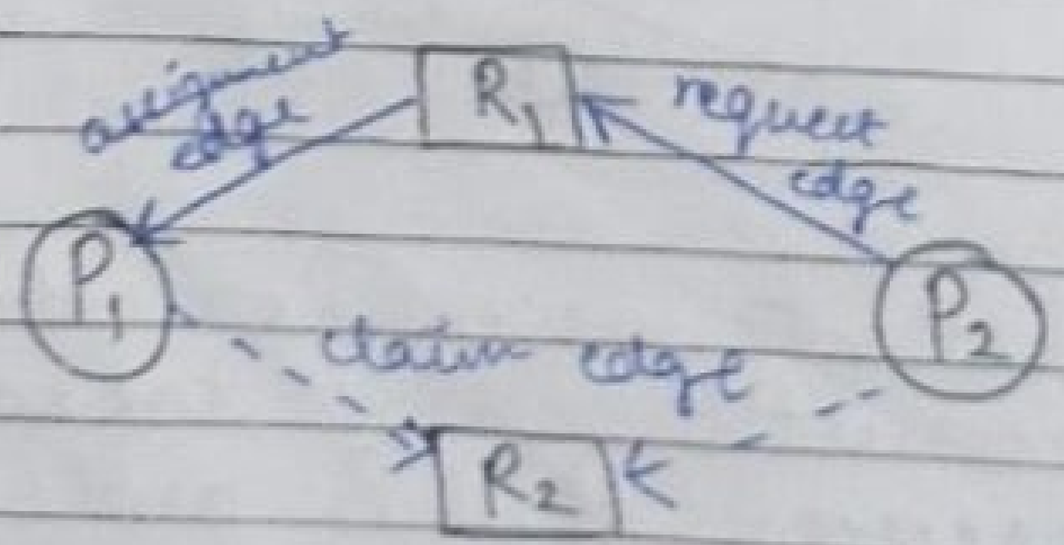
To illustrate we consider a system with 12 tapedrives and 3 processes P_0, P_1 and P_2

Scanned by TapScanner

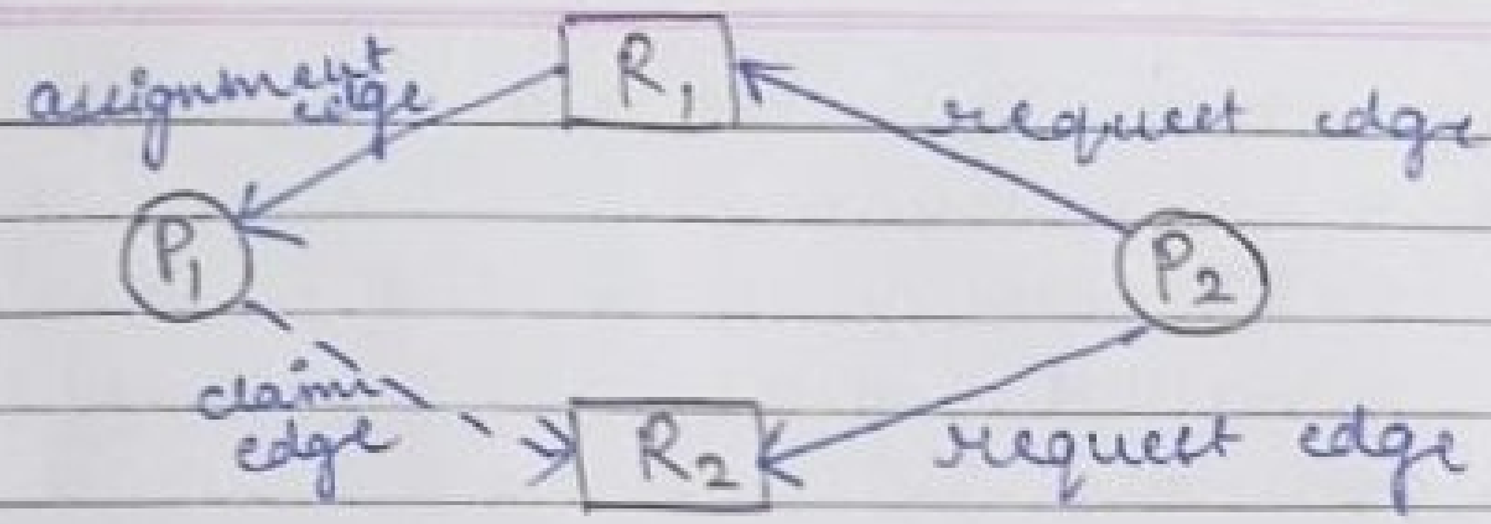
	Maxneed	(maxneed - allocated) Current need	Allocated	Available (3)
P_0	10	5	5	10
P_1	4	2	2	4+1=5
P_2	9	7	2	9+3=12

Two Deadlock avoidance Algorithm

1. Resource Allocation Graph Algorithm
In addition to the request edge and assignment edge a new type of edge called as the claim edge is used. A claim edge indicates that the process P_i may request R_j at some time in the future. This edge resembles a request edge in direction but is represented in the graph by a dotted line. When process P_i requests resource R_j, the claimed edge is converted to a request edge similarly when a resource R_j is released by P_i the assignment edge $R_j \rightarrow P_i$ is reconverted to a claim edge



Scanned by TapScanner



2. Banker's Algorithm : This algorithm is applicable to multiple instances of each resource type but it is less efficient

When a new process enters the system it must declare the maximum no. of instances of each resource type that it may need this no. may not exceed the total no. of resources in the system

Let n is the no. of processes
Let m is the no. of resource types

Some of the data regarding this algorithm is as follows

1. Available : A vector of length m indicates the no. of available resources of each type. If Available[j] = K then K instances of type R_j are available

Scanned by TapScanner

Max : An ~~nxm~~ n×m matrix defines the maximum demand of each process if Max[i][j] = K then process Pi may request atmost K instances of resource type Rj

Allocation : An n×m matrix defines the no. of resources of each type currently allocated to each process if Allocation[i][j] = K then process Pi is currently allocated K instances of resource type Rj

Need : An n×m matrix defines indicates the remaining resources need of each process if Need[i][j] = K then process Pi may need K more instances of resource type Rj to complete its task.

Need[i][j] = Max[i][j] - Allocation[i][j]

* Safety Algorithm :
* Resource request algorithm

Example : Consider with five processes P0 to P4 and three resource types A, B, C

Scanned by TapScanner

resource type A has 10 instances, resource type B has 5 instances, and resource type C has 7 instances. Suppose that at time t_0 the following snapshot is given

*In paper we will make new Table and replace new request (here written with black pen)

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2
P_1	2	0	0 (3,0,2)	3	2	2	(2, 3, 0)		
P_2	3	0	2	9	0	2			
P_3	2	1	1	2	2	2			
P_4	0	0	2	4	3	3			

① Construct the Need Matrix =
Max - Allocation

	Need		
	A	B	C
P_0	7	4	3
P_1	1	2	2 (0,2,0)
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Safety Algorithm: for P_i, if Need$_i$ ≤ Avail then P_i is in safe state

Available = Available + Allocation
otherwise, condition is false, P_i must wa

For P_0, $Need_0 = 7, 4, 3$ Available $= 3, 3, 2$
Condition is false, P_0 must wait

For P_1, $Need_1 = 1, 2, 2$ Available $= 3, 3, 2$
Condition is true
Available $= 3, 3, 2 + 2, 0, 0 = 5, 3, 2$

For P_2, $Need_2 = 6, 0, 0$ Available $= 5, 3, 2$
Condition is false, P_2 must wait

For P_3, $Need_3 = 0, 1, 1$ Available $= 5, 3, 2$
Condition is true
Available $= 5, 3, 2 + 2, 1, 1 = 7, 4, 3$

For P_4, $Need_4 = 4, 3, 1$ Available $= 7, 4, 3$
Condition is ~~false~~ true, ~~P_4 must wait~~
Available $= 7, 4, 3 + 0, 0, 2 = 7, 4, 5$

For P_2, $Need_2 = 6, 0, 0$ Available $= 7, 4, 5$
Condition is true
Available $= 7, 4, 5 + 3, 0, 2 = 10, 4, 7$

For P_0, $Need_0 = 7, 4, 3$ Available $= 10, 4, 7$
Condition is true
Available $= 10, 4, 7 + 0, 1, 0 = 10, 5, 7$

$\langle P_1, P_3, P_4, P_2, P_0 \rangle$
Safe state
Safe sequence

Scanned by TapScanner

Date
Page

Resource Request Algorithm : Suppose now that process P_1 requests one additional instance of resource type A and 2 instances of resource type C therefore $request_1$ = (1, 0, 2). To decide whether this request can be immediately granted we apply resource request algorithm. The following action is taken

① If $Request_i \leq Need_i$ goto step 2
$1, 0, 2 \leq 1, 2, 2$

② If $Request_i \leq Available_i$ goto step 3
$1, 0, 2 \leq 3, 3, 2$

③. Since the above two steps are satisfied the following modification will happen in the matrix

Available = Available - Request
= (3, 3, 2) − (1, 0, 2) = 2, 3, 0

Allocation = Allocation + Request
= (2, 0, 0) + (1, 0, 2) = 3, 0, 2

Need = Need − Request
= (1, 2, 2) − (1, 0, 2) = 0, 2, 0

Scanned by TapScanner

12/10/22

Deadlock Detection : If a system does not employ either a deadlock prevention or a deadlock avoidance algorithm then a deadlock situation may occur in this situation the system may provide an algorithm that that examines the state of the system. To determine whether a deadlock has occured and an algorithm to recover from the deadlock.

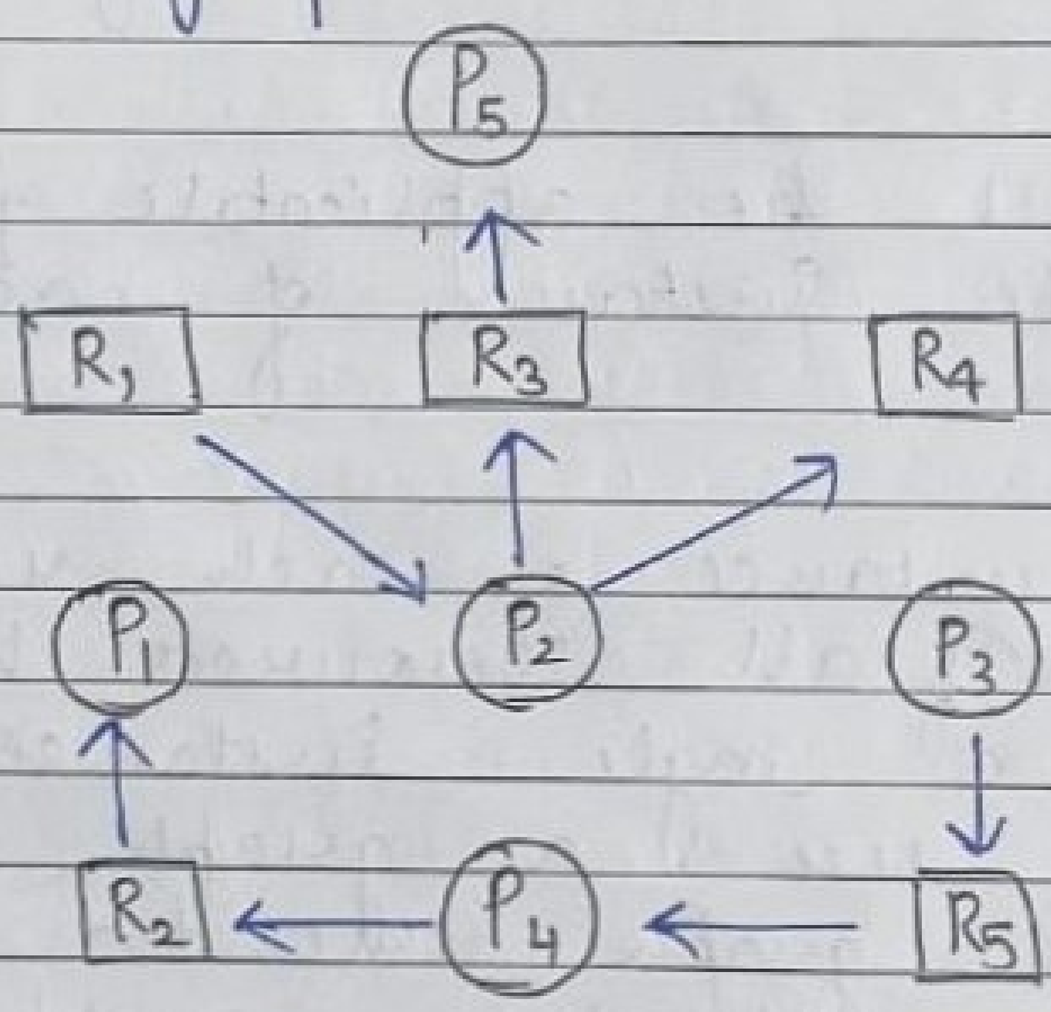
This will be applicable for only a single instance of each resource type

⇒ Single instance of each resource type : If all resources have only a single instance then we can use a graph called wait for graph extended from resource allocation graph by removing the resource nodes and collapsing the appropriate edges i.e. an edge from P_i to P_j in a wait for graph implies that process P_i is wait for process P_j, to release a resource that P_i needs.

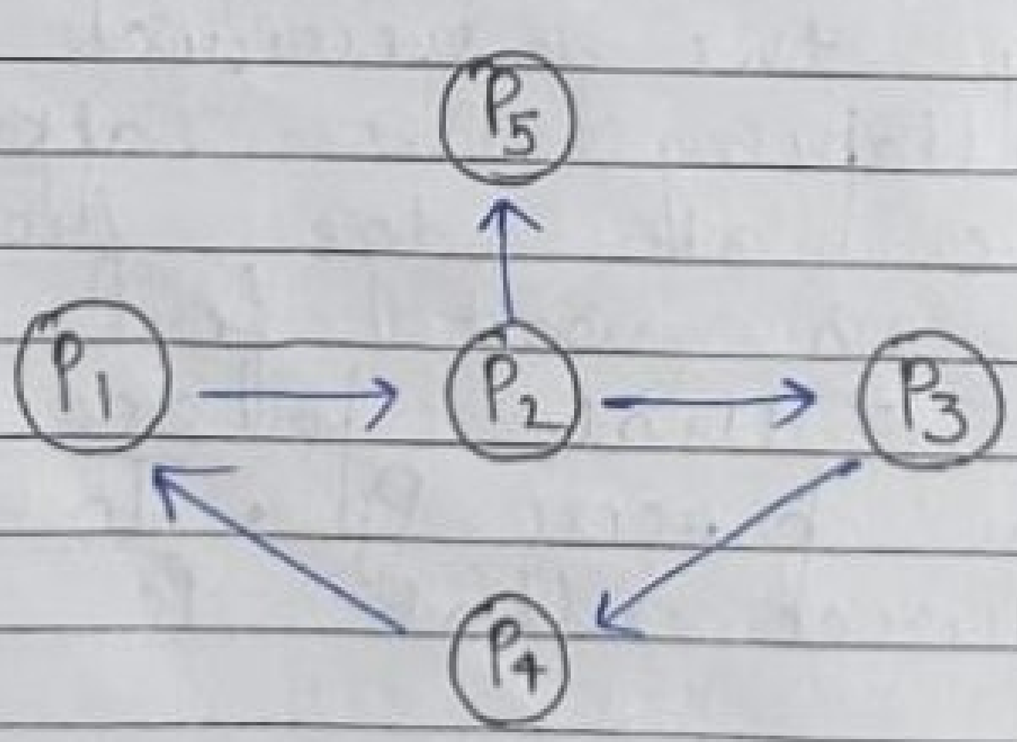
Scanned by TapScanner

An edge $P_i \rightarrow P_j$ exits in a wait for graph if and only if the resource allocation graph contains two edges. $P_i \rightarrow R_q$ and $R_q \rightarrow P_j$ for some resource R_q

A deadlock occurs in the system if and only if wait for graph contains a cycle. To detect deadlock system needs to maintain the wait for graph and check for the cycle in the graph



RAG (Resource Allocation Graph)



wait for graph

Scanned by TapScanner

12/10/22

Unit IV Device Management

Buffering : A buffer is a memory area that stores data being transfered between two devices or between a device and an application

Buffering is done for three reason
- One reason is to balance with a speed mismatched between the producer and consumer of a data stream
- Second reason is to provide adaptation for devices that have different data transfer sizes
- The third is to support logics for application I/O

Whether communication is direct or indirect messages exchanged by communicating processes reside in a temprary queue of three types

1. Zero Capacity Queue : The maximum length of this Queue is zero. The messages cannot wait in it
2. Bounded Capacity Queue : The Queue is of length n i.e. n message can be stored in it

Scanned by TapScanner

disk magnetic
disc optical disk

3. Unbounded Capacity Queue : The Queue is of infinite length thus any no. of messages can wait in it

Secondary Storage Storagerage

1. Disk structure : Disk drives are addressed as logical blocks which is a smallest unit of transfer. The size of logical block is usually 512 bytes and goes higher

The logical blocks is mapped into sectors of the disk. Sector 0 is the first sector of the disk in the first track. They are sequenced. The mapping proceeds in order through that track, then through the rest of the track in the cylinder and then through the rest of the cylinders from outermost to innermost

The no. of sectors per track per cylinder per disk has been always inc. with the advancement of disk technology

2. Disk Attachment : Computers access disk storage in 2 ways

i) Host attached Storage : This storage is (HOST ATTACHED STORAGE) accessed through (ex

I/o ports there ports use several technologies The computer use an architecture called IDE (Integrated Development Enviroment) or ATA (Advance Technology Attachment) SATA (Serial Advance Techology Attachment) PATA (Parallel ATA). Severs generally use more sofesticated I/O architecture such as SCSI (Small Computer System Interface) and fibre channels

ii) NETWORK ATTACHED STORAGE: It is a special purpose storage system that is accessed over a data network. Clients access n/w attached storage via certain procedure calls such as Network File System. iSCSI (internet) is the latest network attached storage protocol

3 ~~iii)~~ DISK ~~form~~ formatting: A new magnetic disk is in a blank state it is a magnetic recording material - Before a disk can store data it must be devided into sectors so that 2 operations read and write can be done. This process is called low level formatting or physical formatting.

Sectors consist of headers, footers and data

Scanned by TapScanner

The OS needs to record its own data on the disk in two steps
(i) The first step is to partition the disk into one or more groups of cylinders
(ii) second step is logical formatting or creation of a file system

File allocation table is the first file system

Boot Block : The computer starts running from the ROM which has a program called Bootstrap loader which initializes all aspects of the system from CPU to registers to device controllers and the contents of main memory and then starts the OS. The bootstrap finds the OS on the disk and goes to the starting address to begin the OS execution. The bootstrap has a boot code which is kept in the system disc. The bootcode in the first sector of the hard disk is called as master boot record. Master boot record has more partitions containing the OS and device drivers. Once the system identifies the boot partition it reads the first sector from that

Scanned by TapScanner

partition.

Bad Blocks : Because disks have moving parts and small tolerances they can fail sometimes they need to be replaced and its contents restored from backup media to the new disk

In other case, one or more sectors become effective they are called bad blocks. These blocks are handled in variety of ways. One of the ways is through sector sparing in which a bad block sector replaced by by spared sector in this process data may be lost

Swap Space Management : Swapping occurs when the physical amount of memory reaches a critically low point and processes are moved from memory to swap space to free available memory

Swap space Management is the task of the OS. Systems use swap space to hold the entire process including the code and data segments. The amount of swap space

~~ROM.~~

needed on a system may vary from MB to GB depending on the amount of physical memory and virtual memory. A swap space can reside in the normal file system.

Disk Reliability : When using a disk drive there may be problem due to a power failure during a write operation. Reliability is an attribute of an computer related component that consistently performs according to its specifications. Reliability, Availability and Serviceability are considered as important aspects of reliability.

Disk Scheduling : One of the responsibilities of the OS is to use the efficiently for the disk should have fast access time and large disk bandwidth.

The access time has two major components

1. Seek time : It is the time for disk arm to move the head to the cylinder containing the desired sector.

2. Rotational latency : It is additional time for the disk to

rotate the desired sector to the disk head

The disk bandwidth is the total no. of bytes transferred divided by the total time between the first request for service and completion of the last transfer

All of the above can be improved by managing the order of the I/O requests. When one request is completed the OS chooses the pending request to service next through various scheduling algorithms as follows:

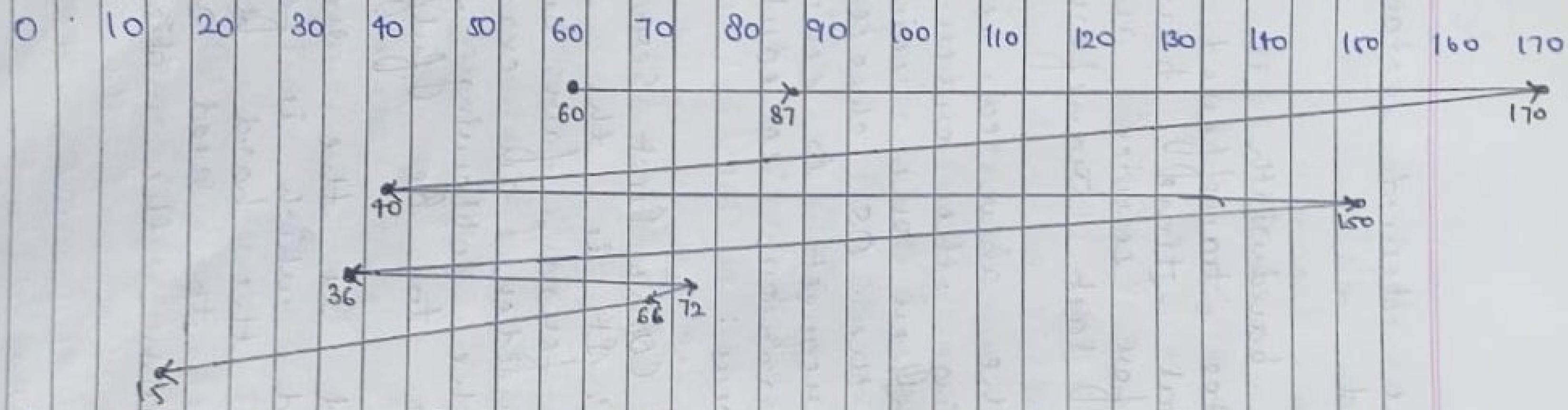
1. FCFS (First Come First Server) Disk Scheduling: It is the simplest form, fair but not the fastest. Consider an example where the following request of I/O needs to be fulfilled

* Disk head is the read or write head which is found the arm of the hard drive that is used to read and write data from the disk

Scanned by TapScanner

queue : 87, 170, 40, 150, 36, 72, 66, 15

disk head = 60 ~~disk~~ total cylinders = 180



(60 to 87) + (87 to 170) + (170 to 40) + (40 to 150) + (150 to 36) + (36 to 72) + (72 to 66) + (66 to 15)

27 + 83 + 130 + 110 + 114 + 36 + 6 + 51 = 557 cylinders = total head movements

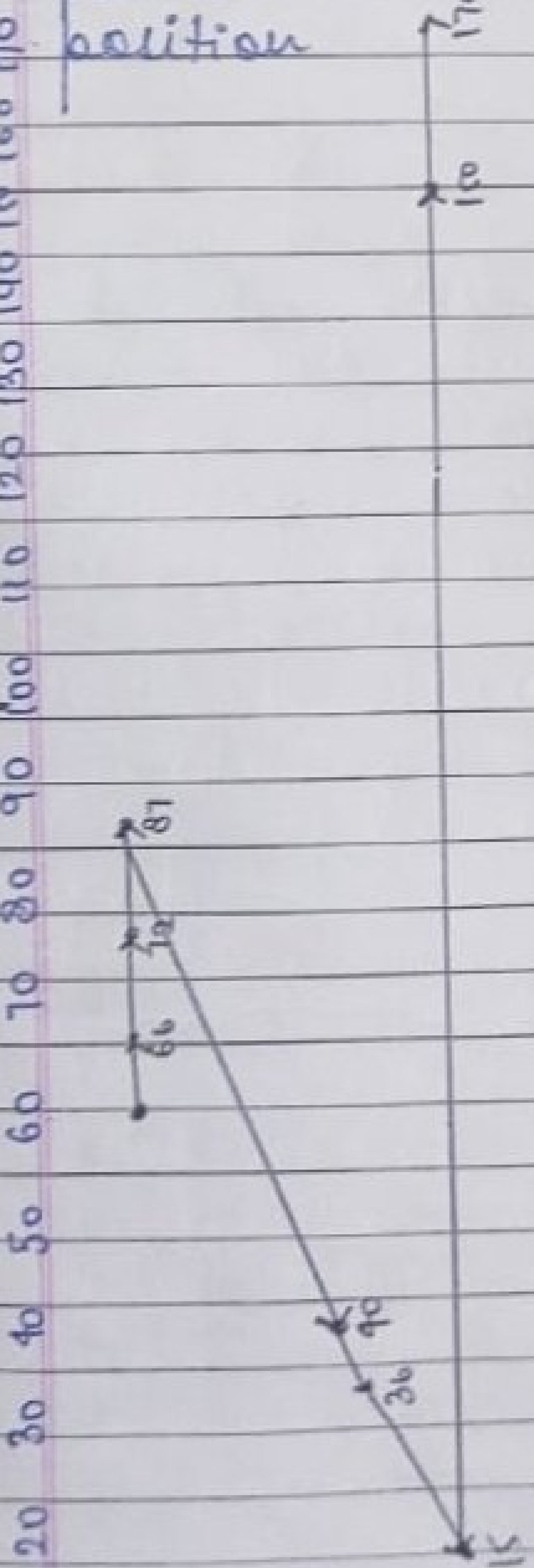
557/8 = Avg. head movement

69.6 cylinders

Scanned by TapScanner

2 Shortest seek time first (SSTF) disk scheduling: This algorithm selects the request with the minimum seek time first from current head position. SSTF chooses the pending request closes to the current head position

queue: 87, 170, 40, 150, 36, 72, 66, 15
disk head = 60
total cylinder = 180

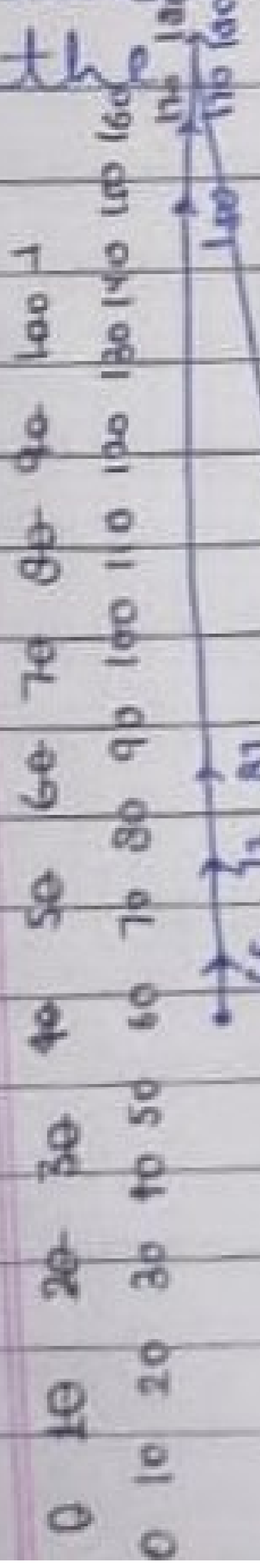


(60 to 66) + (66 to 72) + (72 to 87) + (87 to 40) + (40 to 36) + (36 to 15) + (15 to 170)

6 + 6 + 15 + 47 + 21 + 135 + 20 = 250 = Total cylinder

Avg. head movement = 250/8 = 31.75 = Total head movement cylinder

Scan Scheduling: In this disk arm start at one end of the disk and moved towards other end of servicing request as is reached each cylinder untill it gets to the other end of the disk. At the other end, the direction of the head momentum is reversed and servicing continues. The head continuously scans back and fourth across the disk it is also called Elevator Algorithm. Since, the disk arm behaves just like an elevator in a building first serving all the requests going up and then reversing to service requests the other way



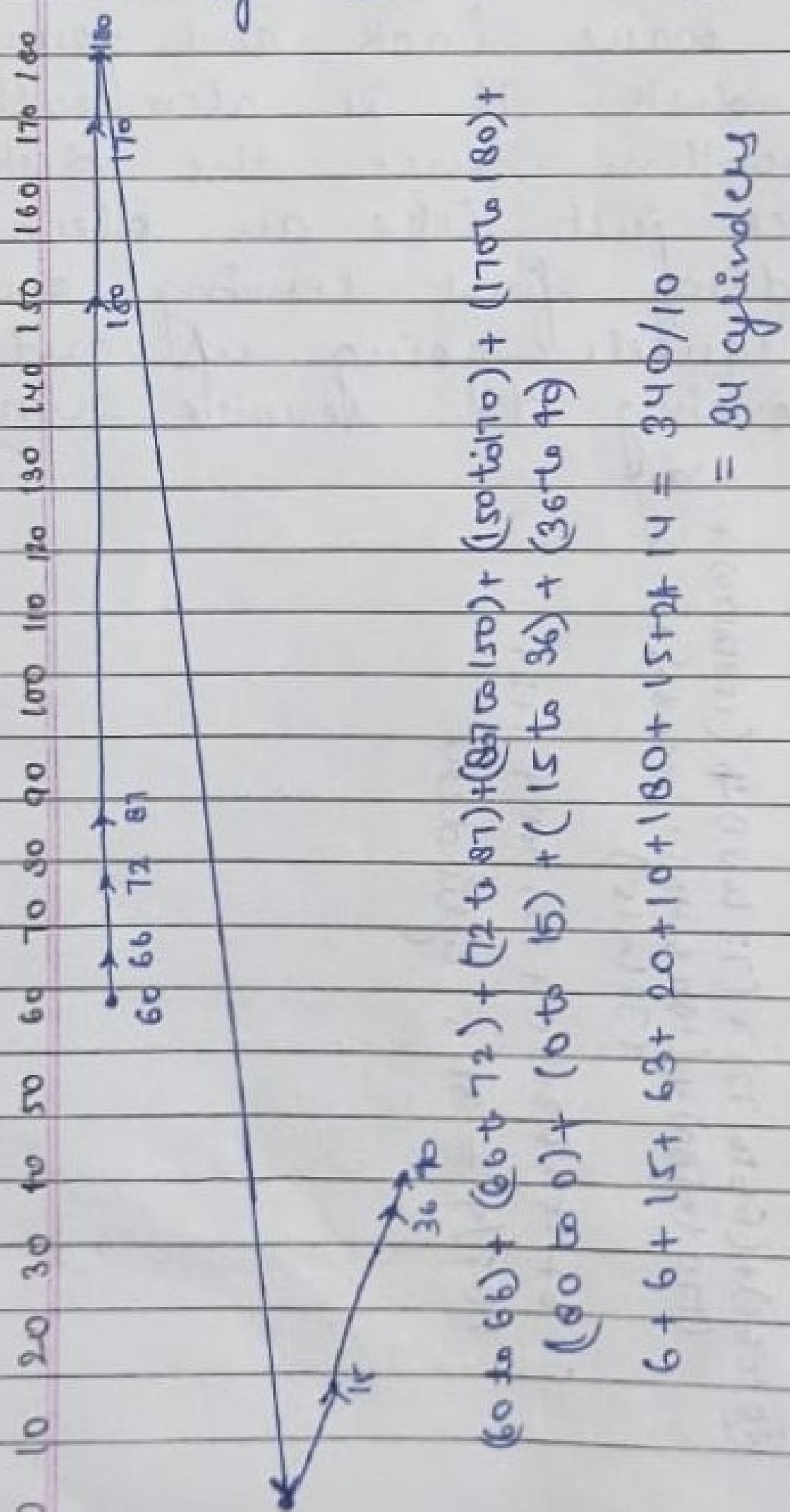
(a) $(26+66)+(66+72)+(72+87)+(87+150)+(150+170)+(170+190)$

(b) $(190-40)+(40-36)+(36-15)$

$6+6+12+6+20+10+140+4+21$

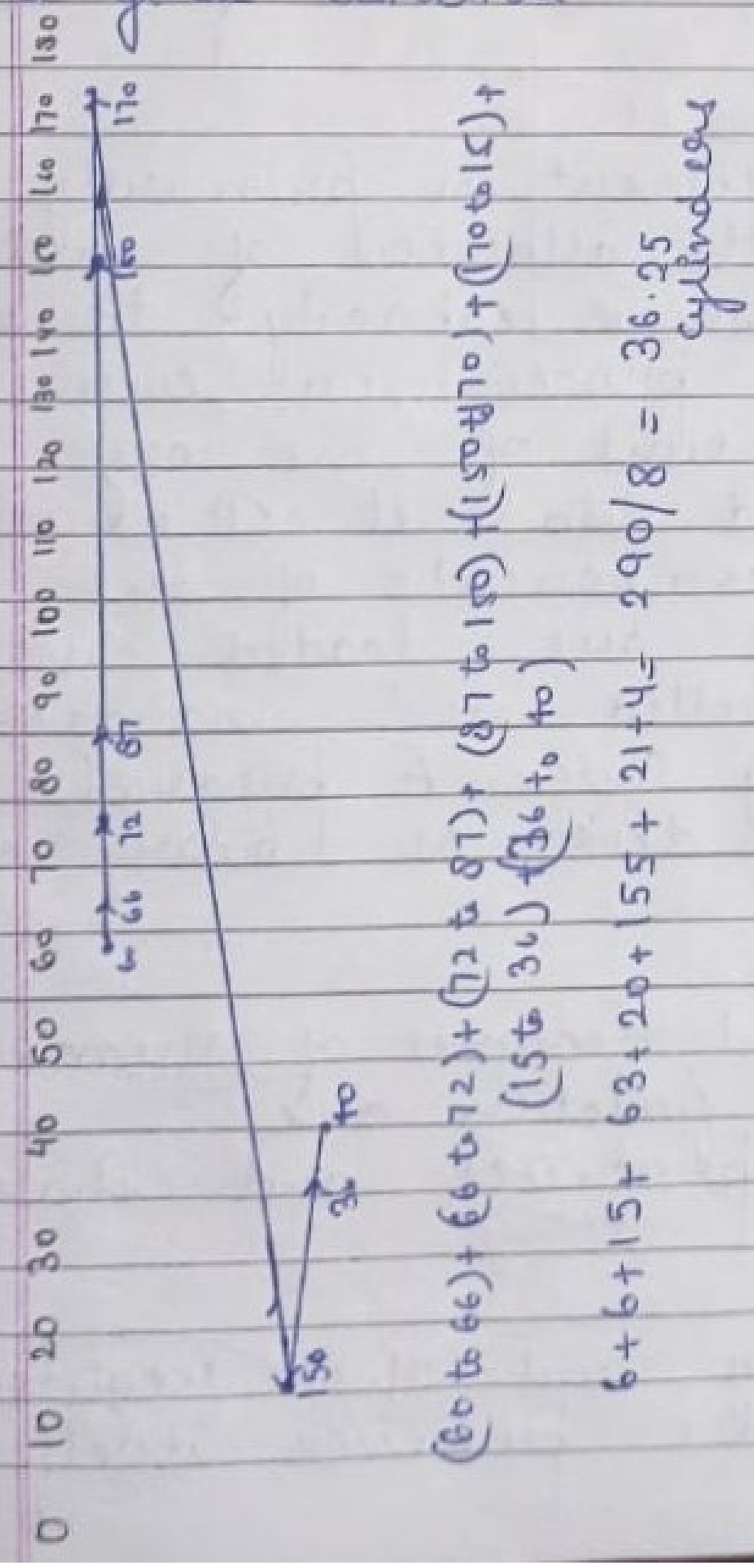
$285/9 = 31.6$ cylinders

C Scan : Circular scan is an extension of scan designed to provide a more uniform wait time. C-Scan move the head from one-end of the disk to the other servicing requests along the way when the head reaches the other end it immediately returns to the begining of the disk without servicing any request on the return trip. It treat the cylinder as the circular list



$(60+66)+(66+72)+(72+87)+(87+150)+(150+160)+(160+170)+(170+180)+$
$(180-0)+(0+12)+(12+36)+(36+60)$
$6+6+15+63+20+10+180+15+24 = 340/10$
$= 34$ cylinder

Look Scheduling: In the previous algorithm the disk arm moved across the full width of the disk, but in this algorithm the arm goes only as far as the final request in each direction then it reverses immediately without going all the way to the end of the disk this is called as C look scheduling because they look for a request before continuing to move in a given direction



$(50 \to 66) + (66 \to 72) + (72 \to 87) + (87 \to 98) + (98 \to 150) + (150 \to 170) + (170 \to 183)$

$(15 \to 36) + (36 \to 53)$

$6 + 6 + 15 + 63 + 29 + 155 + 21 + 4 = 299/8 = 36.25$ cylinders