Q. 5M Basic func^n of OS

Q. 5M or 10M System calls, Types of System calls

Q. 10M What is OS? Goals & functionalities of OS?

- Mod 1 - 20-25M
- Draw diagrams given in the book.
- If you can't complete the paper, adjust your time by decreasing the amt of explaination, & try to complete all questions.
- Mod 2 - 2.1) Processes
  - → imp topic
  - → Transition diagram
  - → PCB diagram.
  - → Start by preparing the difference User level V/S Kernel level threads with explanation.

- Differenciation w respect to certain factors / points.

- You can write theory from the differentiation points & draw diagram if you can.

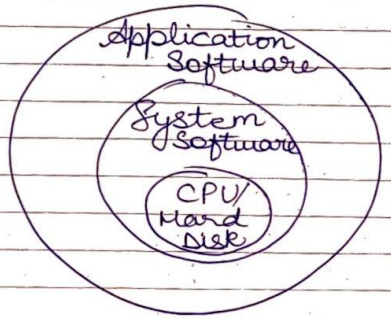
- Mod 5 - 5.2 - Disk Scheduling
  Revise one minute before Jexam the pointers.

* OS – System software

* Interaction of OS & hardware

Scene 1: Recognizing input, sending output

Scene 2: Interaction of the application & hardware.



* Computer System can be divided into 4 components –

→ Hardware
→ OS
→ App software & System ~~software~~ programs
→ The users.

* OS controls & coordinates the use of hardware among the various users & the application programs.

* OS is a resource allocator.

→ Computer system – has many resources

→ Uses priority Queue.

→ Resources = CPU time, memory space, file storage space, i/o devices, etc.

→ OS acts as a manager of all these resources & allocates them to the specific programs & users as needed by their tasks.

→ Proper allocation of resources.

→ OS can also be viewed as a control program

→ ~~Os~~ controls various I/O devices & the users.

★ main goals of OS

→ make computer sys convinient to use
→

★ OS objectives (Stallings)

1) • Convenience
2) • Efficiency
3) • Ability to evolve

1) make computer more efficient

★ OS functions

1) App
2) I/O Management
3) Device Drivers
4) Memory Management
5) CPU Management
6) Hardware

★ OS by the late 1950s → mostly won't come.
1960s
1970s
the history part ↑

★ Early Evolutions Goals & Objectives - Same.

1945: ENIAC

19

★ Terms

• Multicore Processor
- integrated circuit to which 2 or more processors have been attached for enhanced performance, reduced power consumption & more efficient simultaneous processing of multiple tasks.

- is a single computing component with 2 or more indepent actual processing units (called cores) which read the instructions & execute them.

★ OS Services. [Services & functionalities - same]

Range & extent of services provided by an OS depends on a no. of factors.

- The target env that the OS is intended to support.
• It is largely determined by user-visible functions of an OS.

→ OS provides services to both users & programs
- Providing env to the programs for their execution
- Users can execute their programs in an efficient way

[To be Revisited]

★ Types of OS [Write about only these]

→ Batch operating System (User can't directly interact with OS)

→ Time-Sharing OS
- enables many people, located at various terminals, to use a particular Computer system at the same time.

- adv
1) Quick response
2) Avoids duplication of software
3) Reduces CPU idle time

- disadv
1) problem of reliability
2) Problem of data communication
3) System reliability, integrity.

→ Distributed OS

- uses multiple central processors to serve multiple real-time ~~processors~~ applications.

- adv
1) Resource sharing facility
2) Speedup Exchange.

→ Network OS

- runs on a server
- allowed shared file & printer access. typically among LAN.

→ Real Time OS ← Real time systems
← Real time OS (2 definitions)

- data processing system
- response time is very less as compared to online

- 2 types
⇒ hard real-time systems
⇒ soft real-time systems

→ Parallel OS → any os working parallelly. [Geeks for Geeks]

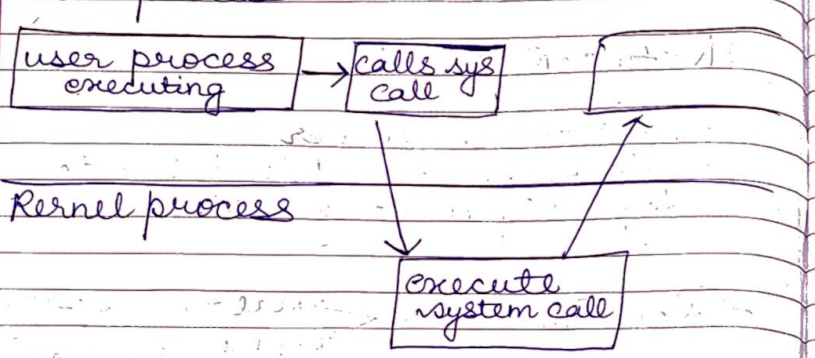
★ 10 M Ques - all types with adv & disadv

★ OS operations —

★ 2 modes of OS

User mode — Kernel mode

Dual mode ~~all~~ operation allows OS to protect itself & other system components.



★ System Calls

- interface b/w user programs & OS
- Varies from OS to OS

Usually types is not asked

File Management — Device Management — Info Mngment — Process Control — Types of System Calls — Communication

31/7/23

Module 2

★ Process (Active) ⟷ Program (Static)

A piece of code while executing

Process == Jobs (used interchangeably)

the no. of times, a code executes the no. of processes increase

★ A process includes
- text section
- stack → containing temporary data - Func^n parameter, return addresses, local variables
- data section containing global variables
- heap → containing memory dynamically allocated during run time.
- Current activity including program counter,

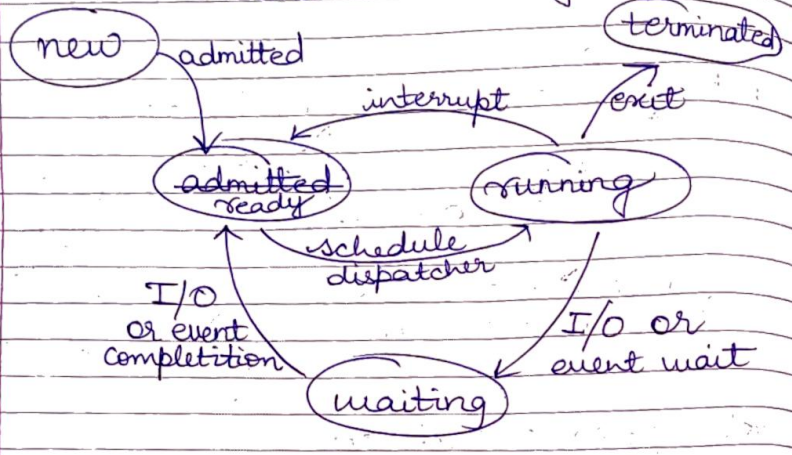
SMQ: Explain process state & diagram (Always draw diagram)

In Galvin BOOK Follow given Women-clature

★ As a process executes, it changes state
- → new: process is being created
- → running: instructions are being executed
- → waiting: process is waiting for some event to occur
- → ready: process waiting to be assigned to a
- → terminated

Very Imp

Process State diagram



Imp ⊙

★ Process Control Block (PCB)
[Descriptive for the process]

Info associated with each process
→ Process state
→ Process counter
→ CPU registers
⋮

[Diagram]

→ Process state: state may be new, ready, running, waiting, halted & so on.

→ Process Counter.
→ CPU-Scheduling information
→ Accounting Information
→ I/O Status Information

★ CPU Switch from Process to Process

[Diagram]

Maximize CPU use

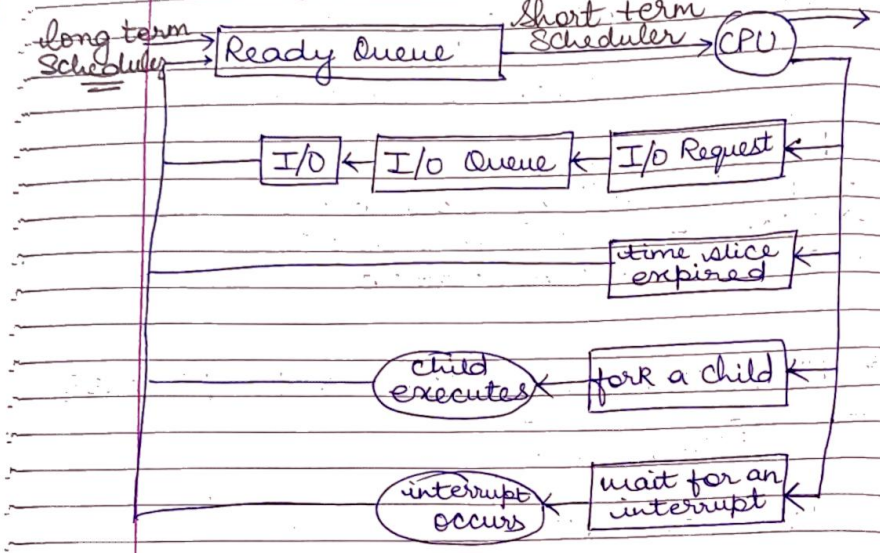
Process Scheduler

Scheduling queues
→ Job queue (set of all)
→ Ready queue (CPU)
→ Device queue (I/O)
} Processes migrate among various queues.

★ Explain Short term, medium, long term

[Repr^n of Process Scheduling diagram]

Repr of Process Scheduling



fork – a process which generates unidentical child.

Q. Diff b/w short term (CPU scheduler) & long term scheduler (Job scheduler)

~~Short~~

Q. Context Switching
- Write how a scheduler works

low number ⇒ high priority.

★ Priority Scheduling

Process	Priority	Arrival Time AT	Burst Time BT
P1	3	0	8
P2	4	1	2
P3	4	3	4
P4	5	4	1
P5	2	5	6
P6	6	6	5
P7	1	10	1

because AT=0

P1	P5	P7	P2	P3	P4	P6

0 8 14 15 17 21 22 27

after 8,
In ready queue, P2, P3, P4, P5, P6 (according to AT)
↑ highest priority

Now, ready queue, → P2, P3, P4, P5, P7
↑ highest

Find CT, TAT, WT, RT

Page No.: Date: YOUVA

* Primitive Scheduling - Execute processes in intervals

For the last ques.

Gantt Chart

P1	P5	P7	P5	P1	P2	P3	P4	P6

0 5 10 11 12 15 17 21 22 27

P1 - 1 P1 - 2 ... P1 - 5

P1, P2 P1, P2, P3 P1, P2, P3, P4, P5

Priority 3, 4 3 3 4 4 5 2

P1 will continue

P5 will execute

P5 - 6 .... P5 - 10 P7 - 11

P1, P2, P3, P4, P5, P6, P7 P1, P2, P3, P4, P5, P6

3 4 4 5 2 6 1

		AT	BT	CT	TAT	WT	RT
P1	3	0	~~3~~	15	15	7	0
P2	4	1	2	17	16	14	14
P3	4	3	4	21	18	14	14
P4	5	4	1	22	18	17	17
P5	2	5	~~6~~ 1	12	7	2 1	0
P6	6	6	5	27	21	15	16
P7	1	10	X	11	1	0	0

~~A~~ ~~B~~ C A D

4 5 2

A ② B

* Round - Robin (By default - primitive)

quant approved for processes

q = 2

Processes	AT	BT	CT
1	0	5	12
2	1	4	11
3	2	2	6
4	4	1	7

~~P1 P2 P3 P1~~

~~P1~~ ~~P2~~ P3 P1 P4 P2

Ready Queue 3 1 2 4

~~P1~~	~~P2~~	~~P3~~	~~P1~~	~~P4~~	~~P2~~	P1

Chart

P1	P2	P3	P4	P1	P2	P1	X wrong

0 2 4 6 7 9 11 12

P1	P2	P3	P1	P4	P2	P1

0 2 4 6 8 9 11 12

Processes	AT	BT
1	0	5 3 1
2	1	3 1
3	2	1
4	3	2
5	4	3 1

P1	P2	P3	P1	P4	P5	P2	P1	P5

P1	P2	P3	P1	P4	P5	P2	P1	P5

0 2 4 5 7 9 11 12 13 14

Processes	AT	BT
1	0	4 2
2	1	5 3 1
3	2	2
4	3	1
5	4	6 4
6	6	3 1

Ready Queue: P1, P2, P3, P1, P4, P5, P2, P6, P2, P6

Grant Chart

P1	P2	P3	P1	P4	P5	P2	P6

0 2 4 6 8 9 11 13 15

★ Process Creation

Parent processes create children processes, which in turn create other processes, forming a tree of processes.

Generally, process identified & managed via a process identifier (pid)

Resource Sharing

- Parent & children share all processes
- Children share subset of parent's resources
- Parent & child share no resources

Execution

- Parent & children

Address Space

UNIX examples
- fork system call creates new pro.
- exec

forK()

★ Process Termination

[Diff b/w threads & processes]

★ Threads & Processes

defines a sequential execution stream within a process

defines the address space & general purpose attributes.

A thread is bound to a single process
Threads are the unit of scheduling

Q. Diff b/w User-level thread & Kernel-level thread

Q. Thread models

★ Thread Overview

Single-threaded process → heavy weight process

Multi-threaded process → light weight process

Threads share
→ Global memory
→ Process id & parent process id
→ Controlling terminal
→ Process credentials (user)
→ open file info
→ Timers

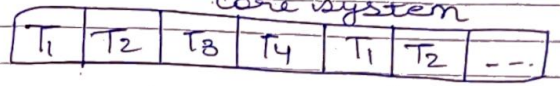
Thread specific attributes
→ Thread id
→ Thread specific data
→ CPU affinity

Benefits
→ Responsiveness
→ Resource Sharing
→ Economy
→

★ MultiCore Programming

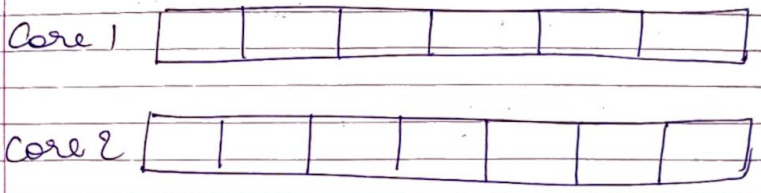
Concurrent execution on a single-core system

Single core

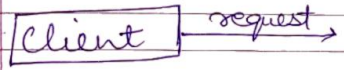


time →

Parallel execution on a Multicore System



Challenges
→ Dividing activities
→ Balance
→ Data Splitting
→ Data dependency
→ Testing & Debugging is harder.



★ Multithreading Models

- User-level → User-threads
  Supported above the Kernel & managed w/o Kernel support.
- Kernel-level → Kernel threads
  Supported & managed by the OS

★ Context Switching

Process level
Kernel level
User level

↑ Time reqd

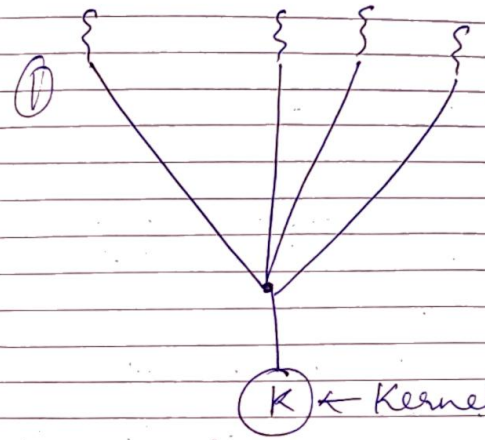
★ User Threads

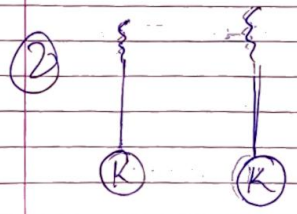
★ Kernel Thread

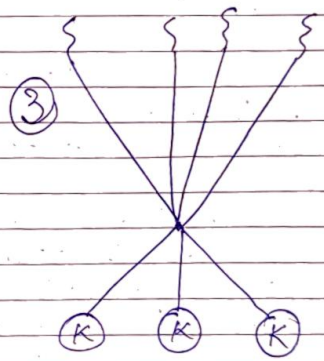
All OS
→ Windows
→ Mac OS

★ Multithreading Models
user level to Kernel level

① → Many-to-One
② → One-to-One
③ → Many-to Many







★ POSIX Compilation on linux

POSIX Thread Creation
" Thread ()
" Thread Termination

Thread Cancellation

★ fork(), exec(), exit()

exec()
All threads, except the one calling exec(), vanish immediately

★ Interprocess Communication

Process within a System → independent / cooperating

Cooperating process can affect or be affected by other processes, including sharing data.

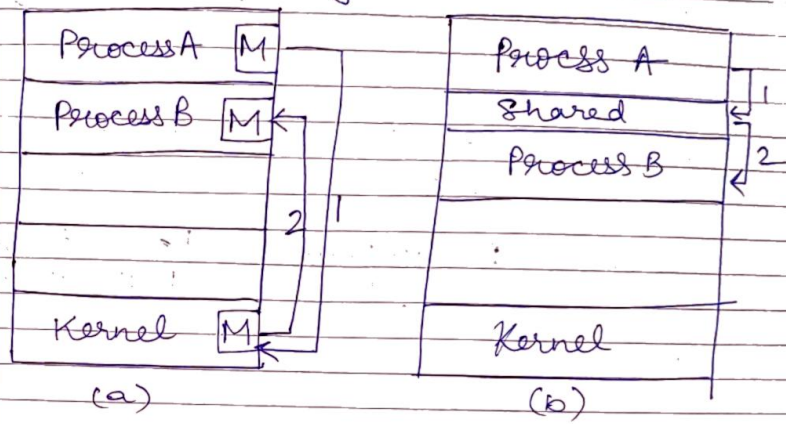
Reasons for cooperating process
→ info sharing
→ computation speedup
→ Modularity
→ Convenience

Cooperating processes need interprocess Communication (IPC)

Two models of IPC
→ Shared memory (b)
→ Message passing (a)

★ Communication Models

M = Messages



★ Producer - Consumer Problem

★ Multiprocessor Systems

Classifications

1) Loosely coupled or distributed multiprocessor, or cluster
2) Functionally specialized processors
3) Tightly coupled multiprocessor

Synchronization Granularity & Processes

Grain Size	Description	Sync interval (instructions)
Fine		
Medium		

★ Independent Parallelism

No explicit synchronization among processes.
- each represents a separate, independent app or job.

Typical use in a time sharing system.

★ Coarse & Very-Coarse Grained Parallelism

Sync. among processes, but at a very gross level.

Good for concurrent processes.

★ Medium-Grained Parallelism.

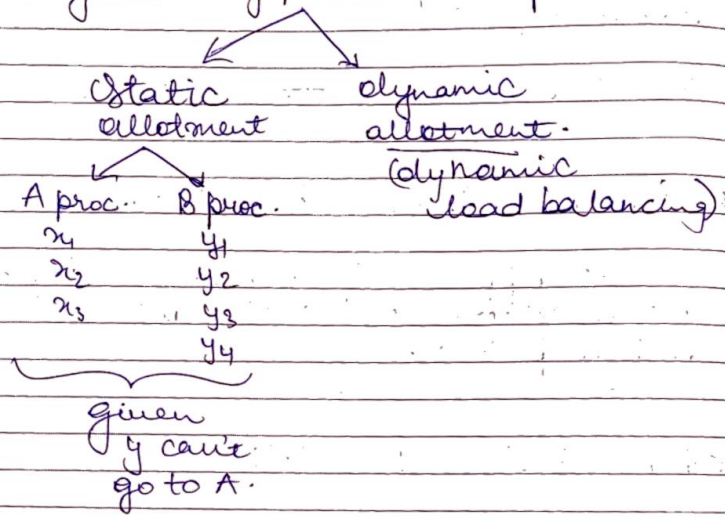
Single app can be effectively implemented as a collection of threads within a single process.

★ Fine-Grained Parallelism

★ Design Issues.

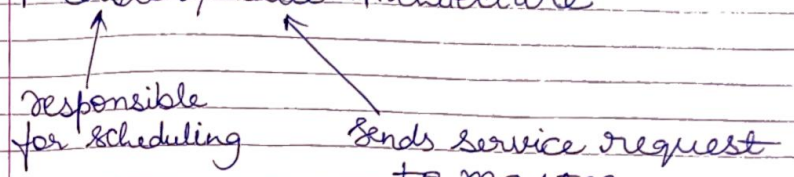
approach - depends on degree of granularity & no. of processors available.

Imp points
- actual dispatching
- scheduling of a process
- use of multiprogramming on individual processor.
- assignment of processes to processor



⇝ Master/Slave
⇝ Peer.

⇝ Master/Slave Architecture



⇝ Peer Architecture

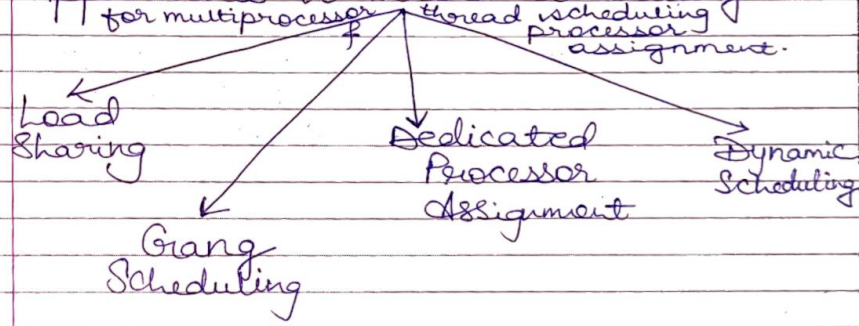
Each processor - self scheduling from pool of available processors

Kernel can execute on any processor

★ Process Scheduling.

★ Thread Scheduling.

Approaches to Thread Scheduling for multiprocessor & thread scheduling processor assignment.



★ Real-Time Systems

Ex: Robotics
air traffic control
telecommunication

Hard real-time task

must meets its deadline

Soft real-time task

desirable but not mandatory to meet its deadline

★ Periodic & Aperiodic tasks

Periodic

once per period T

exactly T units apart

Aperiodic

has a deadline by which it must finish or start

may have constraint on both start & finish.

★ Determinism → only can be provided to a certain extent.

Concerned with how long an OS delays before acknowledging an interrupt

Operations are performed at fixed, predetermined times or within predetermined time intervals.

★ Responsiveness

Concerned with how long, after acknowledgement, it takes an OS to service the interrupt.

ISR: Interrupt Service Routine
→ predetermined
→ Ex. exceptions → a specific ISR executed.

★ User Control

essential to allow the user fine-grained control over task priority

Allows user to specify —
- paging or process swapping
- what processes must always be in main memory

★ Reliability

Real-time systems respond to & control events in real time ~~to~~ so loss or degradation of performance

★ Fail-Soft Operation

ability of a system to fail in such a way as to preserve as much capability & data as possible

imp aspect - Stability

Stable if it will meet the deadlines

★ Scheduling of Real-time process

depends on 3 factors

→ whether a system performs schedulability analysis

★ Classes of Real-time scheduling

① Static table-driven approach
static analysis

② Static priority-driven preemptive approaches
static analysis - priority decided

③ Dynamic planning based approaches
run time par analysis ho rha hai

④ Dynamic best effort approaches
feasibility - not checked
jo aa rha - woh execute

★ Deadline Scheduling

Real-time OS are designed with the objective of starting real-time tasks as rapidly as possible & emphasize rapid interrupt

Info used -
- Ready time
- Start deadline
- Completion deadline
- Processing time
- Resource requirement
- Priority
- Subtask Scheduler

Rate Monotonic Scheduling
[Graph]

Not asked much

* Priority Inversion

occur in any priority-based preemptive scheduling scheme

Occurs when system forces a higher priority task to wait for lower " "

Unbounded priority inversion.

* [redacted]

* Linux Scheduling [Mostly not asked upper upar se dekhna]

I Classes —
1) FIFO
2)
3)

Book Mapping. [Follow Stallings might be easier]

1.1 → Both galvin & stallings work

* Focus on Keywords
Use pointers
Try to stick to the book

Computer Hardware & Software Structure — Galvin

OS Services — Stalling

Key Interfaces —
①
① App Binary In.
② App Programming In.

Mod 1.3 : Monolithic Operating System.

Any module can call any module w/o any reservation.

Kernel mode – has priviledged access & priviledged mode

User mode – only priviledged mode.

does not enforce data hiding.

Ex – MS-DOS
– CP/M

* Layered Operating System.

Each layer is defined & communication dir^n is defined

layer one on top of the other.

Each layer of code is given access to only the lower-level interfaces.

Ex: VAX/VMS
UNIX

Module 1

* 1.1

Types of System calls – 2 in detail baki ainway

→ Process & files (R/W/O/C)

functions associated for only 2

file management f(O) ← don't do this for System calls

file System calls (O/R/W/C)

1.2 → Stallin / notes (Adv & Disadv & Examples.

* Draw diagrams for all *

1.3 → What is Kernel, use
types of Kernel
What's ~~sharing~~ Shell
Types of ~~sharing~~ Shell
(Read upar upar se)

not in syllabus or notes be but can be asked.

→ 3 shell names.

* Process Scheduling & Real time Scheduling numericals.

Mod 1.3

→ Client-Server or Microkernel Operating System.

load on Kernel is distributed

Expand the scope of main memory.

*Characteristics of OS

Ex- C-DAC

Modularity is less complex & expansion is easy because the core functionalities of an OS are considered as a sub system on the user side.

* Visualization / Virtual Machines
↑ ↑ ↑ OS on any → Mention both
use of software to allow a piece of hardware to run multiple OS images at the same time.

→ installing multiple OS on top of an existing one.

Guest OS | Guest OS | Guest OS
Host OS
Hardware

* Virtual Machines
↑
takes layered approach to its logical conclusion

which is

→ Hardware → simulated in software
all resources → virtualized
⇒ individual OS runs on virtualized resources

OS creates the illusion of multiple processes

[System Models & all this → Galvin]
Host is the underlying Hardware ~~is the~~

VMM → Virtual Machine Manager (Hypervisor)

(Mostly asked)

Implementations of VMMs

Type 0 — firmware
Type 1 — os-like Software
Type 2

Benefits & Features of VMMs

- Cloud computing

Mod 1.4 : Model UNIX System

2nd unit → Pg - 110/111

First, see what is UNIX System.

InterProcess Communication.

COF → Common Object File

Draw Diagram, Meaning of the Common functions (the subpoints) & Examples for Modern UNIX System.

Schedules (do difference)

1) long-term schedule (Job schedule)
2) Short-term schedule (CPU Schedule)

Proces
- I/O bound process
- CPU bound process

* Addition of medium term scheduling difference

Long-term	Short-term	Medium-term

* Scheduling algo

1) First come, First serve (FCFS) (preemptive)

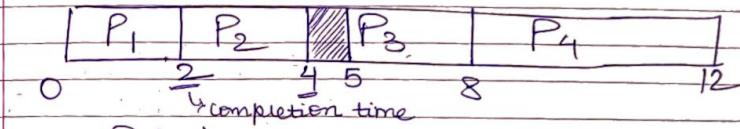
- Arrival time
- Burst time
- Completion time
- Turn around time (TAT) = completion − arrival
- Waiting time = TAT − Burst time
- Response time = the time at which the process get CPU for the first time − Arrival time

Show all calculations in exam

(Processor number)

PN	Arrival time	Burst time	Completion time	TAT	WT	RT
1	0	2	2	2	0	0
2	1	2	4	3	1	1
3	5	3	8			
4	6	4	12			

Gantt Chart



for PN = 1,
TAT = completion time − arrival time
= 2 − 0 = 2
WT = TAT − Burst time
= 2 − 2 = 0
RT = 0 − 0 = 0

for PN = 2,
TAT = Comp. − Arrival
= 4 − 1 = 3
WT = TAT − Burst
= 3 − 2 = 1
RT = [ ] − Arrival
= 2 − 1 = 1

for PN = 3,
TAT = Comp − Arr = 8 − 5 = 3
WT = TAT − Burst = 3 − 3 = 0
RT = 5

* Rate Monotonic Scheduling Algo

Real Time OS

✓ before applying

Check given processes can be scheduled or not

$$\sum_{R=1}^{n} \frac{C_i}{T_i} \leq U = n\left(2^{1/n} - 1\right)$$

C_i → Capacity/Execution time

T_i → also represented as Time Period

Processes	Execution time (C)	Time Period (T)
P_1	3	20
P_2	2	5
P_3	2	10

No. of processes → n

$n(2^{1/n} - 1) = 3(2^{1/3} - 1) = 0.7977$

Value of utilisation → $U = 3/20 + 2/5 + 2/10 = 0.75$

$0.75 < 1$

(∵ the whole scale is not being utilised)

① Feasibility check

Satisfy this eqn → $n(2^{1/n} - 1) > U$

∴ Real time OS scheduling can be done

Scheduling time = LCM (of Time Period)

LCM (20, 10, 5) = 20 time units

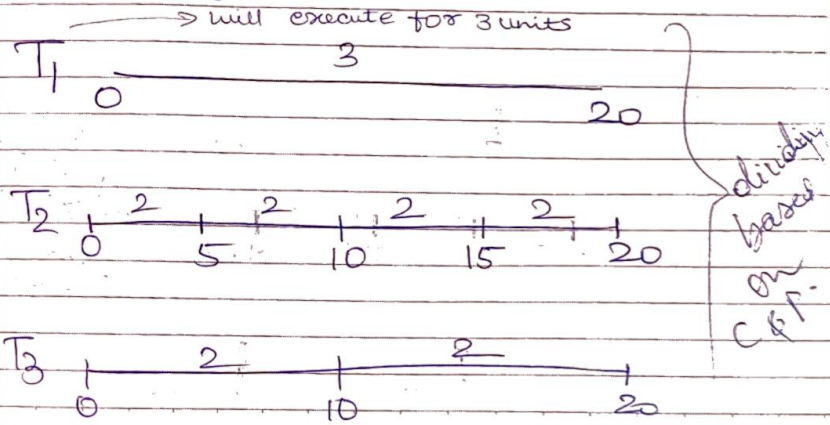
Rate Monotonic

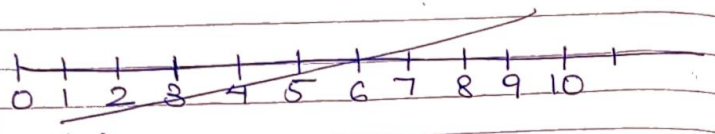
	C	P
T_1	3	20
T_2	2	5
T_3	2	10

We have to consider priority which is based on P.

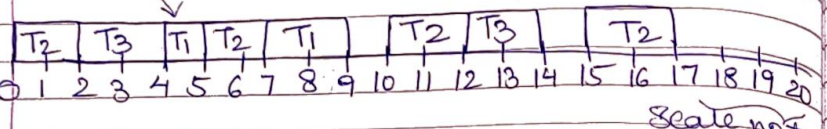
P ↓ = priority ↑

∴ Priority = $T_2 > T_3 > T_1$





Primitive



Scale not utilised fully

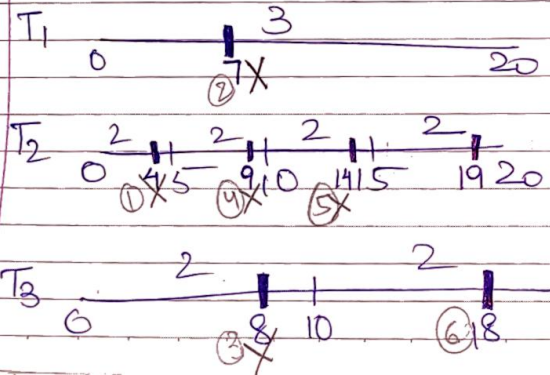
★ Earliest Deadline First (EDF)

	C	Deadline	P
T_1	3	7	20
T_2	2	4	5
T_3	2	8	10

① Do it

Deadline will act as our priority

② Scheduling time = LCM(20, 5, 10) = 20

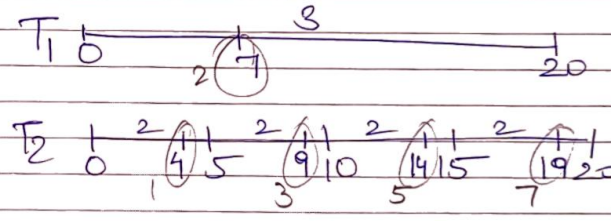


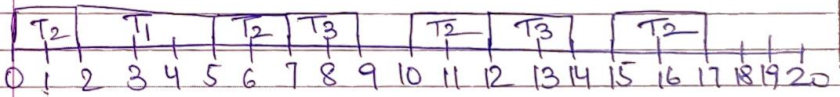
Priority = 4 > 7 > 8 > 9
$T_2 > T_1 > T_3$
14 > 18 > 19
$T_2 > T_3 > T_2$

★ Deadline Monotonic Scheduling

	C	Monotonic Deadline	P
T_1	3	7	20
T_2	2	4	5
T_3	2	9	10

Priority → 4 > 7 > 9
$T_2 > T_1 > T_3$
high low





Process - Stalling & PPt

Suspension, creation, term - Stalling

PCB - galvin/Ppt

2.2 - galvin

2.3 - mix (ppt - galvin) Adv - Disadv

SRTF = Primitive Edge F
STF = Non Primitive (To chhota kaa
Priority Inversion (brief) priority → woh pehl

2.4 - Galvin

2.5 - Malware (Stalling/PPt)

A div - Sems
ISE papers