Operating System Overview

⇒ Functions of OS
- Process Management
- Process Synchronization / Inter Process Communication
- Memory Management / Resource Allocation / Accounting
- Device Management / I/O Management
- Deadlock Handling / Error Detection & Correction
- Security & Protection
- Interface b/w user & hardware

⇒ Types of OS
- Serial Processing (Before the invention of OS)
- Batch Processing
- Multiprogramming
  - Multitasking
  - Multiuser
  - Multi access
- Time Sharing - (Logical extension of Multiprogramming & Multiuser)
- Multiprocessing / Parallel Processing / Tightly Coupled
  - Symmetric
  - Assymmetric
- Distributed Systems / Loosely Coupled
- Real Time
  - Hard Real Time
  - Soft Real Time

⇒ Imp Questions
- ⇒ Functions / Services of OS
- ⇒ Multiprogramming v/s Multitasking
- ⇒ Multiprogramming v/s Time Sharing
- ⇒ Multiprogramming v/s Multiprocessing
- ⇒ Multiprogramming v/s Distributed

Date

Experiment : _____ Page No. _____

⇒ OS can be viewed from two view points
- User View - Ease of use, User Interface, Performance, Resource utilization
- System View - Functions of OS

⇒ Modes of OS
- User Mode - Mode bit is set to 1
- Kernel Mode - Mode bit is set to 0

⇒ Services of OS
- System Calls
- System Programs

⇒ Types of System Calls
- Process Control - create, terminate, wait, signal, load, execute, abort, allocate, free memory
- File Management - create, delete, open, close, read, write
- Device Management - read, write, request, release
- Information Maintainance - date, time, get data, set data, get process file, set process file
- Communication - create, delete, send, receive, transfer status, attach or detach remote device.

⇒ System Programs - Program is a bundles of usefull system calls

⇒ System Calls - Allows a running process to directly communicate with OS in case of needs. It acts as an interface running process & the OS.

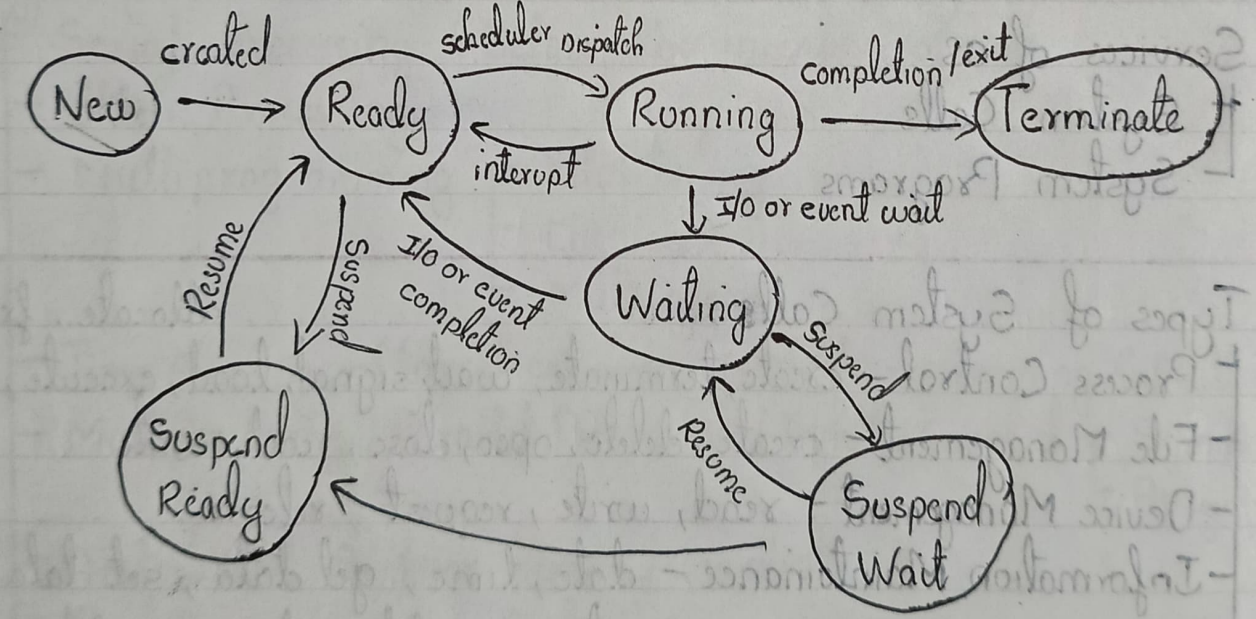
Teacher's Signature _____

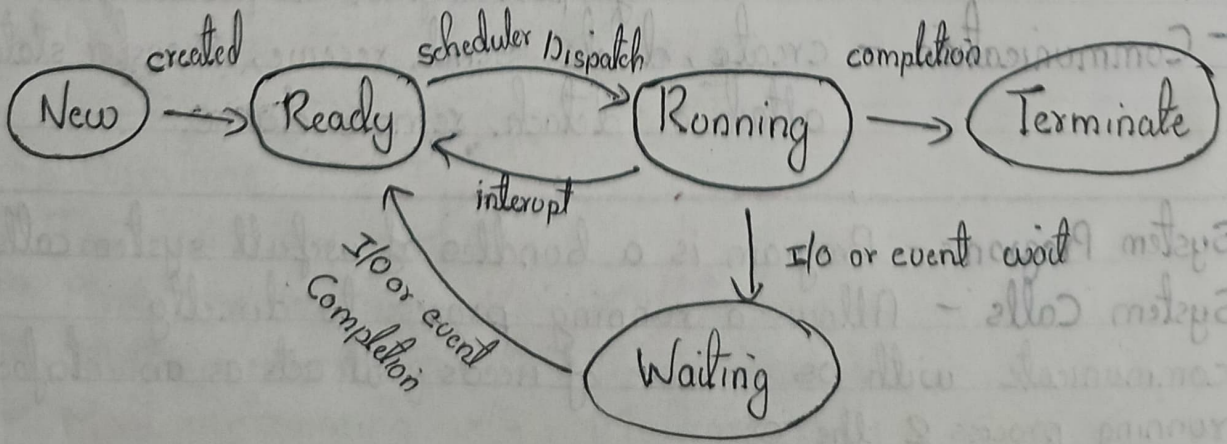
Process Management

⇒ Program v/s Process

⇒ Process State + Diagram

- New - Process in Secondary Memory
- Ready - Process in Primary / Main Memory
- Running - Process with CPU / Execution Phase
- Waiting / Blocked / Suspended - Need I/O or waiting for some event to occur
- Terminate - Completed execution / Ready to leave the CPU
- Suspend Ready - Process is suspended cuz process with high priority has arrived
- Suspend Wait - Same as suspend ready.





Date

Experiment : _____ Page No. 1

⇒ Process Control Block (PCB) + Diagram

- Process ID (Identification Number)
- Process Counter - Pointer to the next instruction to be executed
- Process State - New, Ready, Running, Blocked, Terminate
- Process Priority - High or Low priority
- Accounting Information - CPU time required by the process
- CPU scheduling Information - FCFS, SJF, Round Robin etc.
- I/O status Information - List of I/O devices allocated to the process
- Memory Management Information - Pointer to registers, memory, page table
- Pointer to the Parent process
- Pointer to the child process
- List of open files - While executing some files are opened for Reading & some for writing we should remember which file we have read & which file we have write.

⇒ Context Switching + Diagram

⇒ Cascading Termination

⇒ Throughput

Teacher's Signature _____

Process Scheduling / CPU Scheduling

⇒ Types of Schedulers

- Long Term Scheduler / Job Scheduler - No of process that can be present in the ready queue at maximum also known as degree of multiprogramming is decided by LTS. New to Ready.
- Short Term Scheduler / CPU scheduler - It decides which process must be executed next from the ready queue. Reading to Running. (vice-versa)
- Medium Term Scheduler / Process Swapping Scheduler - It removes process from main memory & put it into secondary memory whenever it needs I/O. Running to wait (vice-versa)

⇒ Dispatcher -

⇒ Criteria For CPU scheduling

- CPU utilization
- Throughput
- Turnaround time
- Waiting time
- Response time

Experiment : _____ Date _____

Page No. _____

⇒ Time Parameters of a Process

- Arrival Time (AT)
- Burst Time (BT)
- Completion Time (CT)
- Turnaround Time (TAT) TAT = CT - AT
- Waiting Time (WT) WT = TAT - BT
- Response Time (RT)

AT - Time at which process enters the CPU

BT - Total time required by a process for its execution

CT - Time at which process exit the CPU or completes its execution

TAT - CT - AT

WT - TAT - BT

⇒ Diagram of Scheduling Table

Process	Priority	AT	BT	CT	TAT	WT	RT

Note ⇒ Lower no. means higher priority

Note ⇒ When AT is given, then do calculate CT, then calculate TAT

Note ⇒ When AT is not given, then do not calculate CT your CT will become TAT

Teacher's Signature _____

⇒ Process Scheduling Algorithm

- ⇒ FCFS (First Come First Serve) (Without Arrival Time) (Non Pscemptive)
- ⇒ FCFS (First come First Serve) (With Arrival Time) (Non Preemptive)
- ⇒ SJF (Shortest Jab First) (Without Arrival Time) (Non Preemption)
- ⇒ SJF (Shortest Jab First) (With Arrival Time) (Preemption)
- ⇒ Priority Scheduling (Without Arrival Time)
- ⇒ Priority Scheduling (With Arrival Time)
- ⇒ Round Rabin (Without Arrival & Priority) (With Arrival & Priority)
- ⇒ Highest Response Ratio

Process Synchronization

Date _____

Experiment : _____ Page No. _____

⇒ Types of Process
- Independent Process
- Cooperative Process (When two process share variable, memory, code, resources)

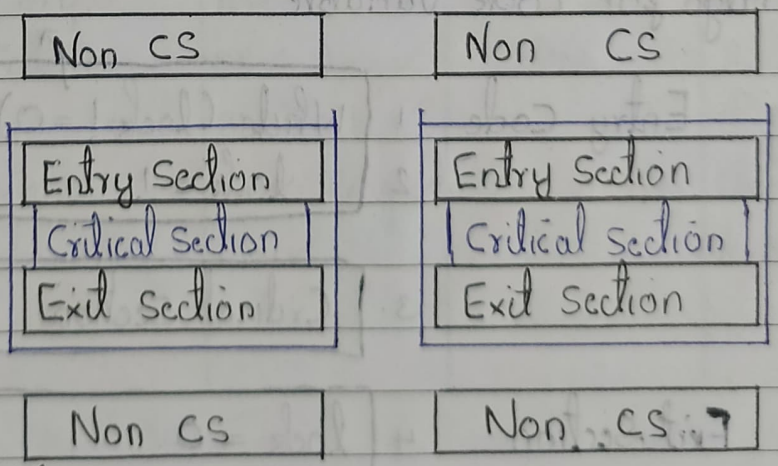
⇒ Need for Process Synchronization

Cooperative processes needs to be sychronized because they to do inter process communication or used some shared variable, memory, code or a resource which if not synchronized lead to some inconsistency or may create the problem of race conditions

Cooperating Processes run in parralled an single CPU.

⇒ Race Condition

⇒ Critical Section



⇒ Critical Section Problem

Teacher's Signature _____

⇒ Condition for Process Synchronization

⇒ Primary
- Mutual Exclusion
- Progress

⇒ Secondary
- Bounded Waiting
- Architectural Neutrality

⇒ Process Synchronization Mechanism
- Lock Variable
- Test Set Lock (Logical extension of Lock Variable)
- Disabbling Interrupt
- Turn Variable
- Interested Variable
- Peterson Solution / Dekker's Solution (Logical extension of Turn & Interested Variable)
- Semaphore
  - Counting Semaphore
  - Binary Semaphore
- Bakery Algo

⇒ Algo for Lock Variable

While (Lock == 1)

```
Entry Code  1  While (lock != 0)
            2      lock = 1
            3  Critical Section
Exit Section 4  lock = 0
```

Date _____

Experiment : _____ Page No. _____

⇒ Classical problem in Process Synchronization
- ⇒ Producer / Consumer
- ⇒ Reader / Writer
- ⇒ Dinning Philosopher

} Algo are very IMP 5★

	Mode	Synchronization Mechanism	Condition For SM	Busy Waiting	No. of Processes
	User	Lock Variable	ME × Progress ×	BW	More than 2 Process
	User	Test set Lock	ME ✓ Prog ×	BW	
	User	Disabling Interrupt	ME ✓ Prog ✓ BW × AN ×	BW	
5★	User	Turn Variable	ME ✓ Prog × BW ✓ AN ✓	BW	2 Process
5★	User	Interested Variable	ME ✓ Prog ✓ BW × AN ✓	BW	2 Process
5★	User	Peterson Solution	ME ✓ Prog ✓ BW ✓ AN ✓	BW	2 Process
5★	Kernel	Semaphore	ME ✓ Prog ✓ BW ✓		More than 2

Teacher's Signature _____

Algo 1 (Turn Variable)

```
type
who = (proc1, proc2);
var turn : who ;
process P1
begin
while true do
begin
 while turn = proc2 do (KeepTesting);
 critical section ;
 turn = proc2 ;
 other - p1 - processing
 end { while }
 end { p1 }
```

```
Parent Process
begin { mutex }
turn = . . . ;
initiate p1, p2 ;
end { mutex }
```

```
process P2
begin
while true do
begin
while turn = proc2 do (Keep testing)
critical section ;
turn = proc1 ;
other - p2 - processing ;
end { while }
end { p1 }
```

Algo 2 (Interested Variable / Flag Variable / Boolean)

```
var
  p1using, p2using : Boolean ;
process P1
begin
while true do
begin
while p2using do (Keep testing)
p1using := True ;
critical section ;
p1using := False
other - p1 - processing
end { while }
end { p1 }
```

```
process P2 :
begin
while true do
begin
while p1using do (Keep Testing)
p2using := True
critical section
p2using := False ;
other - p2 - processing
end { while }
end { p2 }
```

```
Parent process
begin { mutex }
p1using := False ;
p2using := False ;
initiate P1, P2
end { mutex }
```

MEMORY MANAGEMENT

Page No.
Date

- Need for Memory Management
- Object code
- Linker
- Loader

Memory Manager
(Diagram + Limit register
+ Relocation register)

Terminology

- Logical Address
- Physical Address
- Absolute Address
- Relocatable Address
- Internal Fragmentation
- External Fragmentation
- Limit register
- Relocation register
- ① Degree of multiprogramming

Memory Allocation

Continous

- Fixed partitioning / static
- Dynamic partitioning / variable
  (compaction is the solution)

- Data structures
- Bit Map
- Linked List
  - Best Fit
  - First Fit
  - Worst Fit
  - Next Fit
  - Quick Fit

Non Continous

- Paging ← Multilevel Paging, Inverted Page
- Segmentation - Segmental
- Virtual Memory
  (size of process > size of MM)
  (Load only part of process in MM)

- Compaction / Defragmentation
- Overlays

5*

Virtual Memory

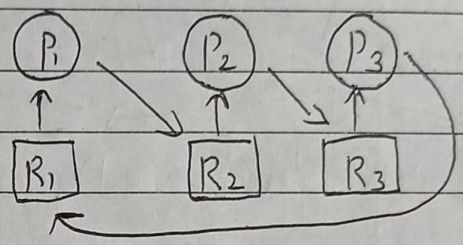
- Frame Allocation
  (Max & Min no. of frames that can be allocated to a process)
  - Equal
  - Weighted
  - Dynamic
- Page replacement Algorithm
  - FIFO
  - OPTIMAL page replacement
  - LRU (Least recently used)
  - Numerical based on FIFO, OPTIMAL, LRU

Deadlock

⇒ Conditions for Deadlock
- Mutual Exclusion
- Hold & Wait
- No Preemption
- Circular Wait

Conditions for Deadlock have different meaning as compare to conditions for synchronization

$P_1 \rightarrow P_2 \rightarrow P_3$



⇒ Deadlock Handling Mechanizm
- Deadlock ignorance
- Deadlock prevention
- Deadlock avoidance
- Deadlock detection & recovery

⇒ Deadlock Prevention

Conditions	Approach
- Mutual Exclusion	Spool everything
- Hold & Wait	Request all resources initially
- No preemption	Take resources away
- Circular Wait	Process should only ask for the resources in the increasing order.

NOTE - We connot implement ME, Hold & wait, No preemption practically
- Circular wait is the solution for deadlock prevention

⇒ Deadlock Ignorance

If there is a deadlock, we act as if there is no deadlock and what happens is system starts to hang or screen blackout and the system will hault and we restart the system & it starts fresh. It is used in most os including UNIX & Windows

⇒ Deadlock avoidance

Banker's Algorithm → Safe state (No deadlock state)
↘ Unsafe state (Deadlock state)

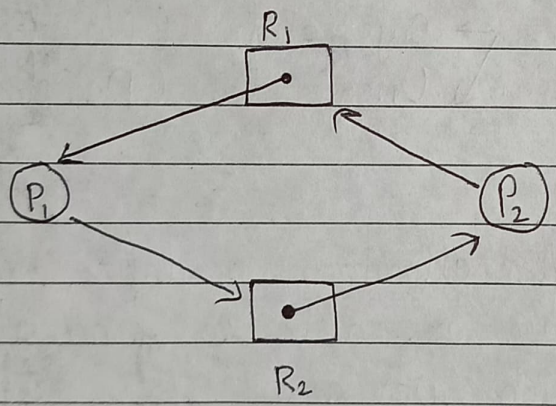
Safe state - If after allocating available resources to the process we are able to find a combination or order in which we can satisfy their needs and there is no deadlock then it is a safe state.

Resource allocation Graph

Representation

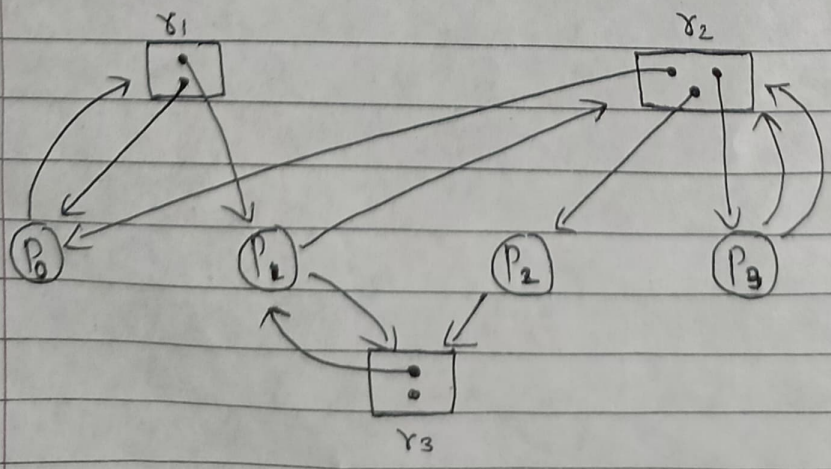
- Process = (P)
- Resource = [R]
- Assign edge [R] → (P)
- Request edge (P) → [R]
- Single instance [·]
- Multi instance [· · ·]

NOTE- If there is a cycle in a single instance graph then it is a deadlock



- R_1 is allocated to P_1
- R_2 is allocated to P_2
- P_1 is waiting for R_2
- P_2 is waiting for R_1

Example



Page No.
Date

Convert Graph into table for better understanding

	Allocated			Required		
	R_1	R_2	R_3	R_1	R_2	R_3
P_0	1	1	0	1	0	0
P_1	1	0	1	0	1	1
P_2	0	1	0	0	0	1
P_3	0	1	0	0	2	0