Final : 2021-22

(1)

(a) Discuss the relationship between hardware, OS, system program a application programs.

⇒

- **Hardware:**
  - These are the physical components of the computer such a the cpu, Memory, input/output devices.
  - Without software, hardware can not perform any tasks.
- **Operating System:**
  - the OS is the core system software that manages all the hardware and the software resources of the computer.
  - It acts as an intermediary between the hardware and other programs, controlling access to the cpu, memory and other peripherals.
- **System programs:**
  - These are the programs that manage and support the computer system and its activities.
  - They are essential for the computer to run and include the operating system itself, as well as compilers, loaders an editors.
- **Application software:**
  - These are programs that uses interact with to perfor specific tasks.
  - They are depend on the system software and OS to function as they can not directly access to the software

(b). What are the main components in the logical organization a computer system?

⇒ * **Central processing Unit: (CPU)**

- **control unit (CU):** Directs operations of the process and co-ordinates between components.
- **Arithmatic Logic Unit (ALU):** performs mathematical and logical operations.

- Registers: small, fast storage locations for immediate data and instructions.

* Memory Unit:
  - Primary memory (RAM): Temporarily stores data and instructions for quick access.
  - Cache memory: High speed memory closer to the cpu for frequently used data.
  - ROM (Read only memory): stores permanent instructions.

* Input/output inputs:
  - Input Unit: converts user data into a format so, the system can process.
  - Output Unit: Converts processed data into human readable form.
  - Controllers: Manage the data between the system bus and the peripheral device.

* Storage Unit:
  - Secondary Storage: provides long term, non volatile storage of data and programs (HDDs, SSDs).
  - Tertiary storage: Backup cloud system cloud storage.

* System Bus:
  - Data bus: tranfers actual data
  - Address bus: carries memory addres
  - Control bus: Sends control signals.

* Software layers:
  - Operating System: Manages hardware & software resources.
  - System software: Utilities and drivers.
  - Application Software: programs for user tasks.

(c). Difference between Kernel mode & user mode.

Kernel mode	user mode
- The program has direct and Unrestricted access to system resources.	- the Application program do not have direct access to system resources. In order to access the resources, a system call must be made.
- the Whole OS might go down if any faults or error occurs	- A single process fails if an interrupt occurs.
- Kernel mode is also known as master mode, privileged mode or system mode.	- User mode is also known as the unprivileged mode, restricte mode.
- All process share a single Virtual address space.	- All process get separate Virtual address space.
- The mode bit of Kernel-mode is 0.	- mode bit of user-mode is 1

(d).

(i) What challenge to face when using a computer without an OS ?

⇒ • No user interface: there would be not desktop, no Start menu, no icons and no Windows, not way to tell the computer what to do.

• Enablity to runs applications: since all applications rely on the OS to run, so without OS no application can run.

• No access to files: there would be no file explorer or finder to nerigate directories.

• Hardware incompatiblity: without OS would need to write software drivers for every piece of hardware to get them work.

• no resource management: the computer's resources would be unmanaged due to a lack of organized memory allocation and cpu scheduling.

ii) What role does the OS play in providing access to applications and files.

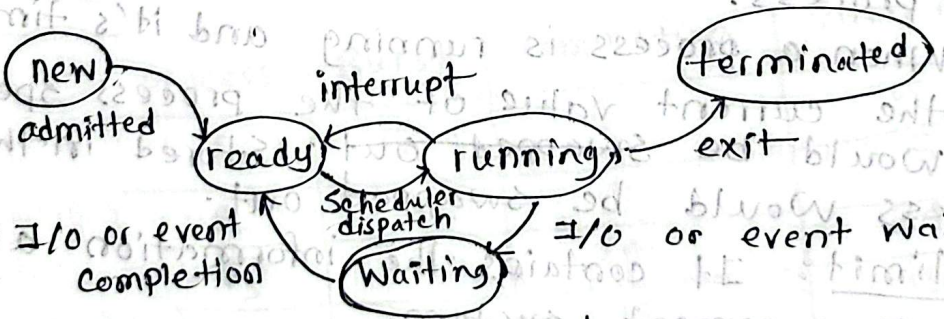
⇒

- Platform of Applications: the OS provides a stable environment and interface that allows applications to be developed and run without needing to know about specific details of the hardware.
- file system management: the OS creates and manages the file system on storage device. This system organizes data into files and directories.
- program execution: the system must be able to load a program into memory and to run that program.
- Error detection: the os needs to be detecting and correcting errors constantly.
- Hardware abstraction: the os uses deriee drivers to communicate with hardware components.
- Resource Allocation: the os ensuros that multiple applications can run simultaneously without interfering with each other.

2

(a). ~~Dfe~~ Define process. States during lifecycle.

A process is a program in execution. Basically it is a sequence of instructions executed in a predefined order.



new: the process is being created.

Running: instruction is being executed.

Ready: the process is waiting to be assigned to a processor.

Waiting: the process it waiting for some event to occur.

terminated: the process has been finished execution.

(b) What is process control Block (PCB)? Illustrate and explain its components with diagram.

A process control block is a data structure used by the OS to keep track of process information and manage execution.

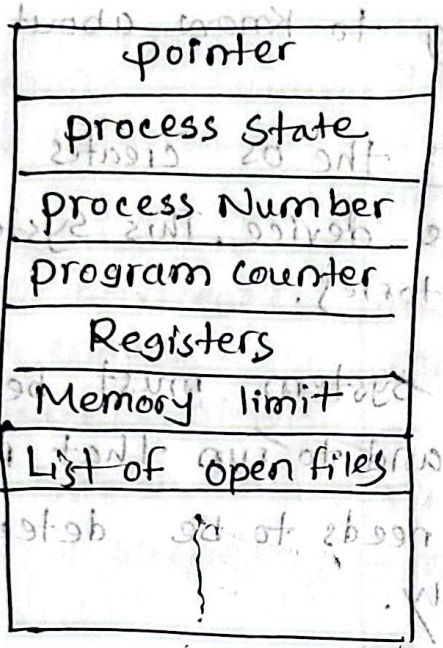


fig: PCB

- pointer: It is a stack pointer that is required to be saved when the process is switched from one state to another.
- process state: It stores the respective state of the process
- process Number: Every process assigned a unique id which stores the process identifier.
- program counter: It stores the counter, which contains the address of the next instruction that is to be executed for the process.
- Register: When a process is running and it's time slice expires, the current value of the process specific register would be ~~swapped out~~ stored in the PCB and the process would be swapped out.
- Memory limit: It contains the information about memory management system.
- list of open files: It contains the list of files opened for a process.

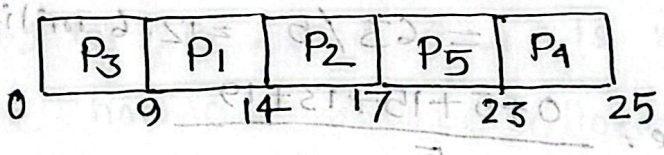
(c)

Process	Burst	~~Arrival~~ Priority prio
P_1	5	3
P_2	3	2
P_3	9	4
P_4	2	1
P_5	6	2

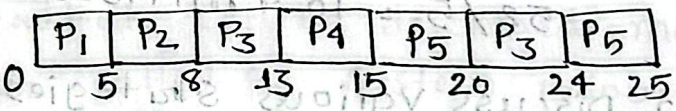
arrival time = 0

i) Gantt chart for preemptive priority (large number higher priority) and Round Robin. (tq = 5)

Priority:



Round Robin:



ii) turn around time for each process:

Priority:

Process	Burst	priority	Completion	Turn Arroun
P_1	5	3	14	14
P_2	3	2	17	17
P_3	9	4	9	9
P_4	2	1	25	25
P_5	6	2	23	23

Round Robin:

process	Burst	Completion	turn Around
P_1	5	5	5
P_2	3	8	8
P_3	9	24	24
P_4	2	15	15
P_5	6	25	25

(iii) Waiting time for each process :-

Waiting time = turn Around time − Burst time.

Priority

$P_1 = 9$
$P_2 = 14$
$P_3 = 0$
$P_4 = 23$
$P_5 = 17$

RR

$P_1 = 5 - 5 = 0$
$P_2 = 8 - 3 = 5$
$P_3 = 24 - 9 = 15$
$P_4 = 15 + 2 = 13$
$P_5 = 25 - 6 = 19$

iv) Average time

priority: average time = $\frac{9+14+0+23+17}{5}$

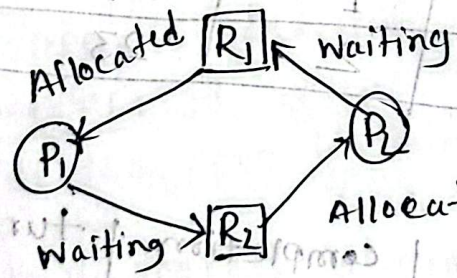
$= 63/5 = 12.6$ miliseconds

RR: average time = $\frac{0+5+15+13+19}{5}$

$= 52/5 = 10.4$ miliseconds.

(3) a) What is deadlock? Discuss various strategies used to prevent it from occuring.

A deadlock is a situation in computing environment where a set of process gets permanently stuck because each process is waiting for a resource hold by another process and none of them can proceed.



Prevention:

- Mutual exclusion: At least one resource must be ho in a non sharable mode, that is only one process at time can use the resource. if another process requ that resource, the requesting process must be delaye until the resource has been released.

- **No preemption**: Resource can not be preempted - that is a resource can be realeased only voluntarily by the process holding it, after that process completed its task.
- **Hold and Wait**: A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other process.
- **Circular Wait**: A set of waiting process must exist such that P_0 is waiting for a resource held by P_1, P_1 is waiting for P_2, ... and P_{n-1} is waiting for P_n.

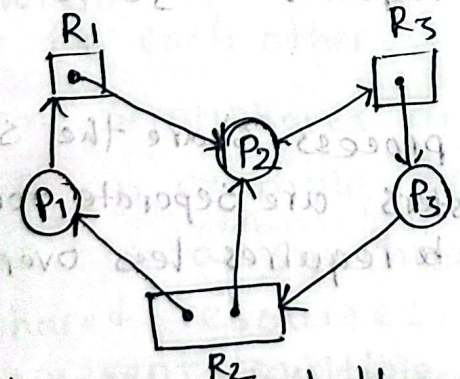
b) Illustrate how a resource Allocation graph (RAG) can be utilized to identify deadlocks in a system?

RAG: Deadlock can be described more precisely in terms of direct graph called a system resource Allocation graph. It consists of a set of vertices V and a set of Edges E. Two different types of nodes can be found : $P = \{P_1, P_2, P_3 \ldots\}$ for Process and resoures: $R = \{R_1, R_2 \ldots R_n\}$.

— if the graph contains no cycle → no deadlock.

— if the graph does contain cycle → deadlock exist.

RAG with deadlock:



$P = \{P_1, P_2, P_3\}$

$R = \{R_1, R_2, R_3\}$

$E = \{P_1 \to R_1, R_1 \to P_2, P_2 \to R$
$R_3 \to P_3, P_3 \to R_2, R_2 \to$
$R_2 \to P_1\}$.

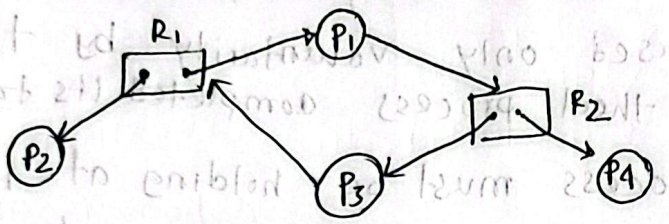
two minimal cylee exist there:

cycle 1: $P_1 \to R_1 \to P_2 \to R_3 \to P_3 \to R_2 \to P_1$

cycle 2: $P_2 \to R_3 \to P_3 \to R_2 \to P_2$

All the processes are deadlock. P_2 is waiting for R_3. wh
P_3, P_3 is waiting for R_2 which held by P_1 a
P_2 and P_1 is waiting for P_2 to release R_1.

RAG Without deadlock:



$P = \{P_1, P_2, P_3, P_4\}$

$R = \{R_1, R_2\}$

$E = \{P_1 \to R_2, R_2 \to P_3, P_3$
$R_1 \to P_1, R_1 \to P_2, P_2 \to$

One minimal cycle is exist:

$P_1 \to R_2 \to P_3 \to R_1 \to P_1$

P_4 or P_2 may release the instance of resource R_2 and R_1 respectively. which can be allocated by P_1 and P_3 respectively, breaking the cycle. There is no deadlock in this system.

c) Define a thread. How does it differ from a process in terms of resource management and execution?

⇒ A thread is the smallest unit of execution that a cpu can schedule.

- Resource management:
  - * process:
    - Has its own independent memory space.
    - Each process maintains its own resources such as open files, registers and PCB.
    - Creating a process requires large overhead.
  - * thread:
    - thread of the same process share the same memory
    - only stack and registers are seperate for each thread
    - Creating a thread requires less overhead.
- Execution:
  - * process:
    - executes independently of other processes.
    - context switching between process is slow.
    - Interprocess communication is complex
  - * threads:
    - execute concurrently within the same process.
    - context switching between threads is fast. because
    - communication between threads is easy.

④ (a). What is process synchronization, and why is it necessary in a multiprogramming environment? provide example of problems caused by lack of synchronization?

⇒ Process synchronization is the technique used to control the order of execution of multiple processes so that they do not access shared resources at the same time in a conflicting way.

Why necessary in multiprogramming:

- Maintain data consistency
- Avoid race condition
- Ensure correct execution order.
- Prevent deadlock and starvation.

problem cause by Lack of synchronization:

- Race condition: Two processes read and update a shared variable at the same time.
- Lost update: two process update the same file/ variabe and one update overWrite's the other's result.
- Inconsistent data: If a process reads shared data while another is modifying it, the result may be inconsistent.
- Deadlock: Without proper synchronization, processes may wait forever for each other.

(b). Describe how semaphores are used for process synchroniz Illustrate with an example of the product-consume proble

⇒ A semaphore is a synchronization tool used to control access to shared resources.

semaphore prevent multiple processes from entiring the critical section at the same time.

omic operation (Wait(S) ⇒ ~~S>0~~ S>0 (Process continue) (S=0, blocked)
Signal(S) ⇒ S=S+1 (waiting process).

producer-consumer:

produce - insert item into the buffer
consumer - removes items from the buffer.

to avoid conflict, we use three semaphores:
mutex = 1 (ensures mutual exclusion);
empty = N (counts empty space in buffer)
full = 0 (counts filled spaces in buffer).

Producer code:

```
do {
    wait(empty); // wait for empty space
    wait(mutex); // lock the buffer
    insert_item(); // critical section
    signal(mutex); // unlock the buffer
    signal(full); // increase number of full slots.
} while (true);
```

Consumer code:

```
do {
    wait(full);
    wait(mutex);
    remove_item();
    signal(mutex);
    signal(empty);
    consume_item();
} while (true);
```

- Mutual exclusion: mutex ensures only one process access the buffer at a time.
- No overflow: empty ensures producer cannot add items when buffer is full.
- No underflow: full ensures consumer can not remove items when buffer is empty.

(c). How does thread scheduling differ from process scheduling? explain with example.

⇒

- **Scheduling level:**

  P: cpu scheduler selects which process will run next.

  T: scheduler selects which thread inside a process will run next.

- **Resource handling:**

  P: each process has its own memory and resources.

  T: thread share the same memory of the process.

- **Context switching cost:**

  P: High cost.

  T: Low cost (only register and stack)

- **Communication:**

  P: is Low

  T: is fast.

- **User level vs Kernel level:**

  P: always doone by OS

  T: User level thread and Kernel level thread.

Examples: Web browser (process)

thread 1: Rendering the webpage

thread 2: Lodding image

thread 3: Running Javascript

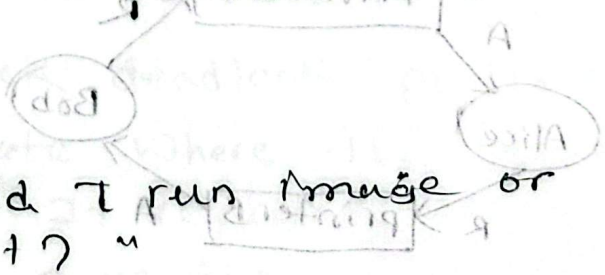
process scheduling:

OS decides "Should I Rus chrome or Ms Word?"

Thread scheduling:

Inside chrome:

OS decides "Should I run Image or Javascript?"



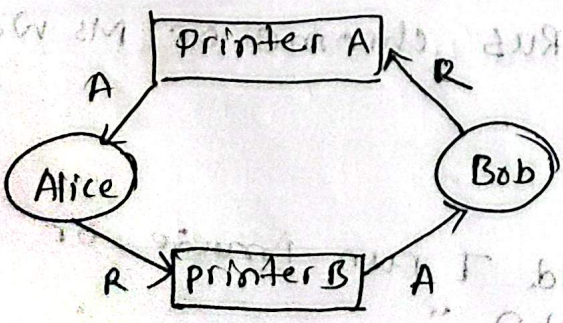
(5) i) Identify and explain the four necessary for deadlock. Which of these are satisfied in the above ~~set~~ scenario?

⇒

Condition	definition	Satisfied/ Unsatisfied	Reason
(i) Mutual exclusion	At least one resource must be non sharable, meaning only one process can use at a time.	Satisfied	The printers are non sha resources.
(ii) Hold & Wait	A process must be holding at least one resource and waiting to acquire additional resources held by other processes.	Satisfied	Alice holds p A waits for printer B. B holds printer Waits for prin
(iii) No preemption	Resource can not be forcibly taken away from a process: they can only be released voluntarily.	satisfied	the system doesn't preem resources.
(iv) Circular Wait	A circular chain of processes exists, where each process is waiting for a resource held by next process in the chain	satisfied	Alice is wait for bob and is waiting Alice.

ii) Draw a Resource Allocation Graph (RAG) to represent the above scenario; Based on your graph, explain how a cycle implies a deadlock?

⇒



cycle Analysis:

The graph contains a cycle: Alice → Printer B → Bob → P A → Alice
In the resource Allocation graph, if every resource in the cycle has only a single instance & the presence of a cycle is both necessary and sufficient for deadlock to exist. The cycle indicates that each process in the loop is waiting for another, creating a permanent circular dependency from which no one can escape.

iii) Describe how a deadlock detection and recovery approach would handle this scenario?

⇒ Deadlock detection: The system detection Algorithm would identify the cycle in the resource Allocation graph. It confirms the deadlock between Alice and Bob processes.

Deadlock Recovery:

- Killing the process: Killing all the processes involve in deadlock one by one. After killing all process check for deadlock again and keep repeating the process till the system recovers from deadloc
- Resource preemption: Resource are preempted from the process involved in deadlock, and preempted resources are allocated to other process so that there is a possibility of recovering the system from deadlock.
- process Rollback: Rollback deadlock processes to a previously saved state where the deadlock condition did not exist. It requires check pointing to periodically save the state of processes.

• Concurrency control: concurrency control machanism are used to prevent data inconsistencies in system in multiple concurrent process. this mechanism ensures that concurrent process do not access the same data at the same time, which can lead to inconsistencies and error.

(iv)

total resources: 2 printers (A,B)
total processes: 2 (Alice, Bob)

Safe and unsafe state check:

process	Allocation		Max need		Remaining need		Availiblity	
	A	B	A	B	A	B	A	B
Alice	1	0	1	1	0	1	0	0
Bob	0	1	1	1	1	0		

No printer's are available: neither Alice and Bob can proceed because Alice needs B and Bob needs A the system would be in an unsafe state.

Conclusion: the Banker's Algorithm would deny Alice request for printer B because it would put the system into a unsafe state, from which deadlock is possible, Alice process would be forced to wait only when Bob finished and realased printer B would Alice's request be granted. There for the system would not have entred deadlock.