Operating System -

1. Defines operating system. Discuss any four services of operating system.

→ An operating system (OS) is the program that, after being initially loaded into the computer by a boot program, manages all of the other application programs in a computer.

Services -
operating system provides services -

- User interface - Almost all operating systems have a user Interface ~~(GUI)~~ (UI). Varies between Command-Line (CLI), Graphics User Interface (GUI), Batch.

- Program execution - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)

- I/o operations - A running programe may require I/o, which may involve a file or an I/o device

- File-system manipulation - The file system is of particular interest. Obviously, Programs need to read and write files and directories, create and delete them, list file Information, permission management

2) Define Process, Explain Process control block with suitable diagram.

~~Process ID~~ Process state
Process State Program Counter
~~Process Priority~~ CPU registers
Accounting information CPU scheduling Information
~~Program Counters~~ memory-Management Information
~~CPU registers~~
PCB Pointers Accounting Information
~~List of openfiles~~ I/O status Information
~~Process I/O status Information~~
.....

Process - A process is a running program that serves as the foundation for all computation.

A process is a pro program at the time of execution.

Process Control Block (PCB) -

A process control block (PCB) is a data structure (table), which contains information related to specific process. Every process or program that runs to needs a PCB. when a user requests to run a particular program, the operating System constructs a process control block for that program. It includes process state, Program Counter, CPU registers, memory limits, list of open files etc.

Typical information that is stored in a process control block is -

- Process state - It contains new, ready, running, waiting & terminated.
- Program Counter - which holds the address of the next instruction to execute.
- CPU registers - Which includes accumulators, index registers, stack pointers, general per purpose registers and condition code information.
- CPU scheduling information - Which includes a process priority, pointers to scheduling queues and other scheduling parameters.
- memory-management information - Which map the address translation table of the process.

is a perfect real-life example (The Person who comes First and stands in the queue gets to buy the ticket First.
रांग

- Accounting information - Which includes amount of CPU and real time used, time limits, account numbers, job or process number.

- I/o status information - Which includes a list of I/o devices allocated to this process, a list of open files, and so on

3) What is CPU Scheduling ? Explain FCFS and Round Robin Scheduling techniques.

→ CPU scheduling is the task performed by the CPU that decides the way and order in which processes should be executed. There are two types of scheduling - preemptive, & non-preemptive.

There are a few common scheduling algorithms -

- First Come, First Served (FCFS) non-Pre
- Shortest Job ft First (SJF) non-Pre
- priority Scheduling (Preemptitive) both non
- Round Robin (RR) Scheduling (Prieemptive)
- ~~First come, First Served (FCFS)~~

First come First Served - Simply schedules the jobs according to their arrival time. The job which comes first in the ready queue will get the CPU first. The lesser कमी the arrival time of the job, the sooner will the job get the CPU
रांग सबसे

criteria : Arrival Time
mode : non-Preenption.

1) maximum CPU utilization
2) for allocation of CPU
3) maximum throughput
4) maximum waiting time

Page No.
Date

Round - Robin (RR) Scheduling -
The round robin is designed for time sharing systems. A small unit of time called time quantum or time slince is defined. The CPU scheduler allocates CPU for a time quantum to first job in ready queue.

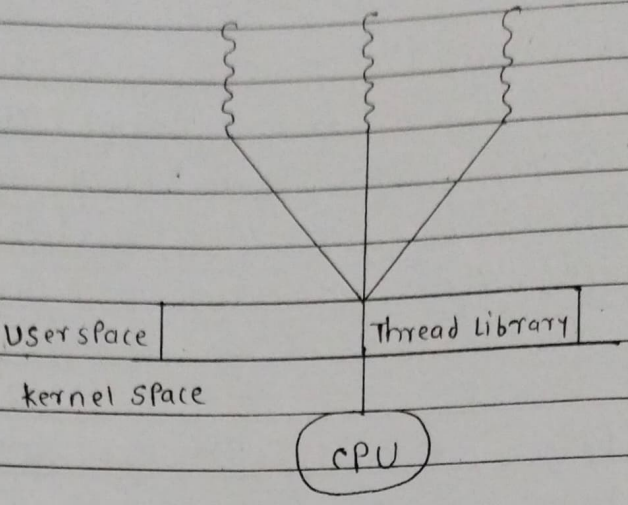
Criteria = "Time Quantum"
Mode = "preemptive".

4 What is thread ? Mention types of thread. Explain any one in detail. Also write advantages of thread

→ A thread refers to a single sequential flow of activities being executed in a process. It is also known as the thread of execution or thread of control.

There are two types of thread -
1) User Threads
2) Kernal Threads

1) User Threads - Thread creation, scheduling, management happen in user space by Thread Library. User Threads are faster to create & manage. If a user thread performs a system call, which blocks it, all the other threads in that process one also automatically blocked, whole process isblocked.



Advantages -

- Thread switching does not require kernel mode privileges (विशेष अधिकार)
- User level thread can run on any operating sys
- Scheduling can be application specific in the user level thread.
- User level threads are fast to create and manage.

2) Kernel Threads - kernel creates, Schedules, manages these threads are slower, manage if one thread in a process blocked, over all process need to be blocked.

Advantages -

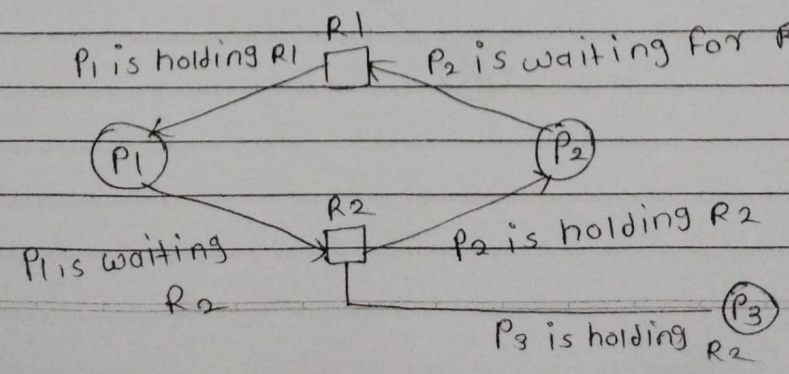
- kernel can simultaneously schedule multiple threads from the same process on multiple process.
- kernel routines themselves can multithre

Advantages of thread -

1. We can execute multiple tasks of an application at a time.
2. Reduces the complexity of a big applications.
3. Helps to improve the performance of an application drastically.
4. Utilizes the max resources of multiprocessor systems.
5. Better user interface in case of GUI based applications.
6. Reduces the development time of an application.

5) Explain Deadlock Avoidance Resource allocation graph with suitable example & diagram.

Deadlock Avoidance - In deadlock avoidance, the request for any resource will be granted if the resulting state of the system doesn't cause deadlock in the system. The state of the system will continuosly be checked for safe or unsafe states.



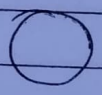
* Resource allocation - graph

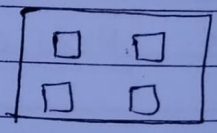
A set of vertices V and a set of edge E. V is partitioned into two types -

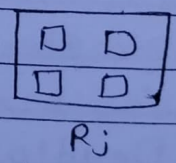
$P = \{P_1, P_2, \ldots P_n\}$, the set consisting of all processes in system

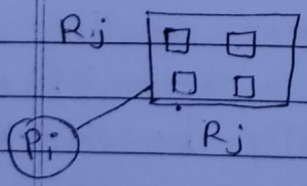
$R = \{R_1, R_2, \ldots R_m\}$, set consisting of all resour types in system.

- request edge - directed edge $P_i \to R_j$
- assignment edge - directed edge $R_j \to P_i$

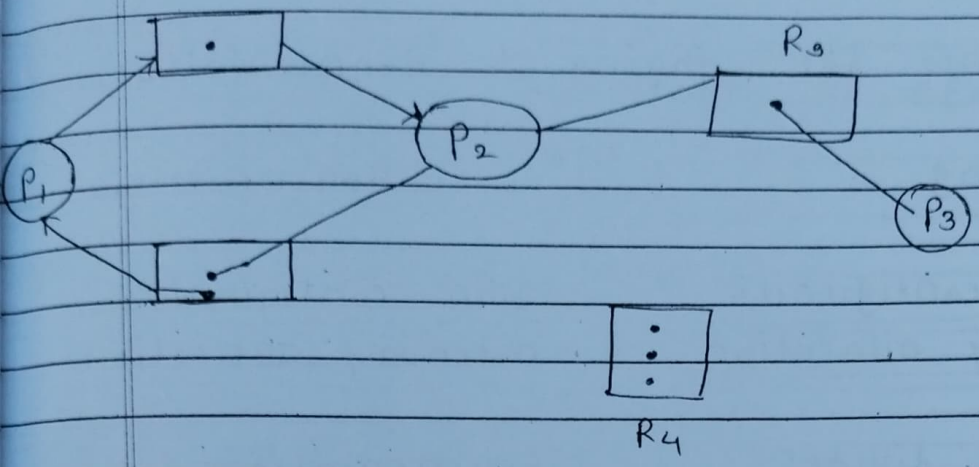
Process 

Resource Type with 4 instances 

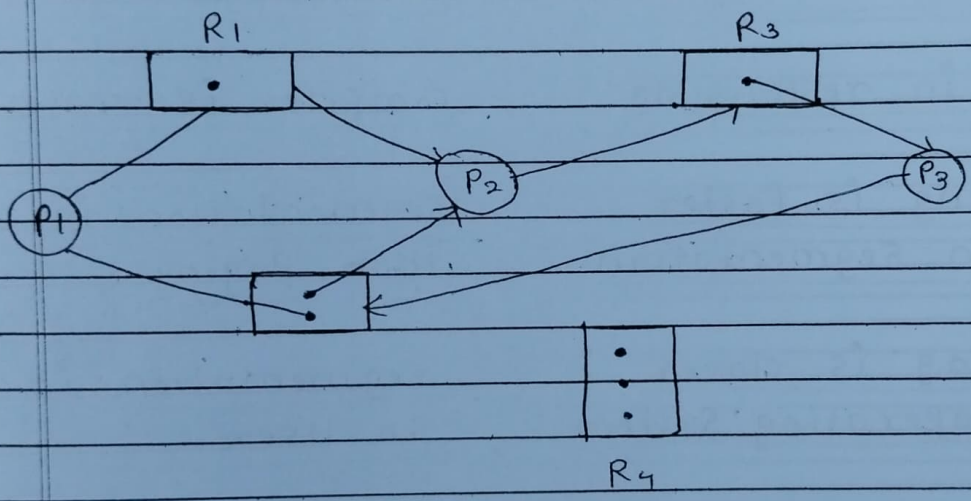
P_i request instance of R_j 

P_i is holding an instance an instance of R_j 

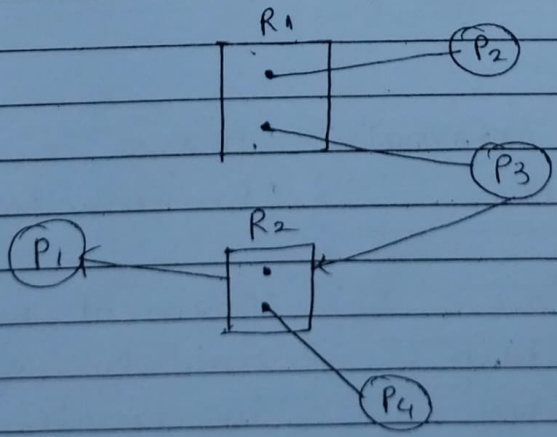
Example of a resource allocation Graph



Resource Allocation Graph with a Deadlock



- Graph with A cycle But No Deadlock



6. Difference bet^n Paging & Segmentation.

	Paging	Segmentation
1.	Non - contiguous memory allocation	Non - contiguous memory allocation
2.	Paging divides program into fixed size pages	Segmentation divides program into variable size segments.
3.	OS is responsible	Compiler is responsible
4.	Paging is faster than segmentation	Segmentation is slower than paging.
5.	Paging is closer to operating system	Segmentation is closer to user.
6.	It suffers from internal fragmentation.	It suffers from external fragmentation
7.	There is no external fragmentation.	There is no external fragmentation.
8)	Page table is used to maintain the page information.	Segment table is maintains the segment information.

* 8. Define thread. Explain multithreaded model.

* 9. What is deadlock ? Explain deadlock characteristization.

→ A deadlock is a situation in which two computer programs sharing the same resource are effectively preventing each other (प्रतिबंध) form accessing (प्रवेश) resource, resulting in both programs ceasing (बंद करना) to function. Deadlock can arise if four conditions hold simultaneously.

- Mutual exclusion - only one process at a time can use a resource (संसाधन)
- Hold and wait - A process holding at least one resource is waiting to acquire additional resources held by other process.

one process hold some process demand

- No preemption - a resource can be released only voluntarily by process holding it, after that process has completed its task.
- Circular wait - there exists a set {$P_0, P_1, \ldots, P_0$} of waiting process such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by $P_2, \ldots, P_{n-1}$ is waiting for a resource that is held by P_n, & P_0 is waiting for a resource that is held by P_0.

10) Paging in details -

Paging is a storage mechanism used in OS to retrive (परत मिळवणे) processes from secondary storage to the main memory as pages. The primary concept behind paging is to break each process into individual pages. Thus the primary memory would also be separated into frames

- ATG Search supports two types of paging, - normal paging and fast paging.

Paging used in OS -

Paging is used for faster access to data, & it is a logical concept. Paging is used to address the issue of external fragmentation. This allows a process's logical address space to be noncontiguous, allowing the process to be allocated physical memory.

*

11) What is an operating System? Explain functi of operating system

→ A program that acts as an intermediary between a user of a computer and the computer hardware operating system goals.
• Execute

An operating system (os) is the program that, after being initially loaded into the computer by a boot program, manages all of the other application program in a computer.

Functions -

1) Booting - Booting is a process of starting the computer, operating system starts the computer to work. It checks the computer and makes it ready to work.

2) Loading and Execution - A program is loaded in the memory before it can be executed. Operating system provides the facility to load programs in memory easily and the execute it.

Datasecurity -

3) Data is an important part of computer system. The operating system protects the data stored on the computer from illegal use, modificatio or deletion.

4) Disk Management - Operating System manages the disk space. It manages the stored files and folders in proper way.

5) Process Management - CPU can perform one task at the one time. if there are many tasks, operating system decides. Which task should get the CPU.

12) Explain Banker's Algorithm in details - with ex.

→ • Banker's Algorithm is a deadlock avoidance strategy.
• It is called so beause it is used in banking systems to decide whether a loan can be granted or not.
• Banker's Algorithm requires : whenever a new process is created, It specifies the maximum number of instances of each resource type that is exactly needs.

• Banker's algorithm, following four data structur are used -
1) Available
2) Max
3) Allocation
4) Need

Working - Banker's ~~alog~~ algorithm is executed whenever any process puts forward the request for allog allocating the resources.
• It involves the following steps -

Step - 1 -

- Banker's Algorithm checks whether the request made by the process is valid or not.
- If the request is invalid, it aborts the request
- If the request is valid, it follows step - 2

Step - 2 - • Banker's algorith checks if the number of requested instances of each resource type is less than the number of available instances of each type.

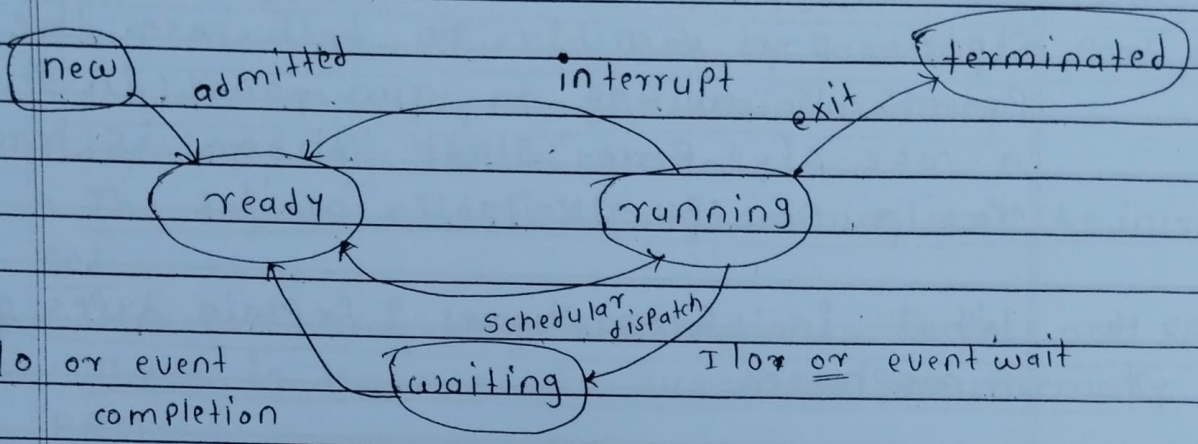
- If the sufficient number of instances are not available, it asks the process to wait longer.
- If the sufficient number of instances are available, it follows step - 3.

Step - 3 -

- Banker's algorithm makes an assumption that the requested resources have been allocated to the process.
- Then, it modifies its data structures accordingly and moves from one state the other state.
- Now, Banker's Algorithm follows the Safety algorithm to check whether the resulting state it has entered in is a safe state or not.
- If it is a safe state, then it rollbacks to its previous state & asks the process to wait longer.

13. Process state -

A process is a running program that serves as the foundation for all computation. A process is a the program at the time of execution.



* Diagram of process state.

- New state - The process being created.
- Terminated state - The process has finished execution
- waiting or Blocked state - When a process blocks it does not show because logically it cannot continue, typically because it is waiting for input that is not yet available. In this state a process is unable to run until some external event happens.
- Running state - A process is said to be running if it currently has the CPU. that is, actually using the CPU at that particular instant.

. . Ready State - A process is ~~sta~~ said to be ready if it use a CPU if one were available. It is runable but temporarily stop to let another process run.

Logically, the 'Running' & 'Ready' states are similar. In both cases the process is willing to run, only in the a case of 'Ready' state, there is temporarily no CPU available for it.

14. What is Symaphores? Explain types of Symaphores.

Q. 15 Explain terms - Virtual Memory & Demand paging.

Virtual Memory -

- Virtual memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory.

Advantages of Virtual Memory -

- The degree of Multiprogramming will be increased.
- User can run large application with less real RAM
- There is no need to buy more ~~mo~~ memory RAMs.

Disadvantages of Virtual Memory -

- The system becomes slower since swapping takes time.
- It takes more time in switching between applications.
- The user will have the lesser hard disk space for its use.

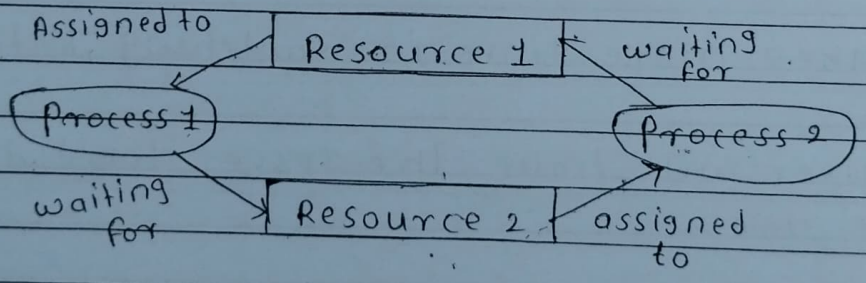
Demand Paging - Demand paging is identical to the paging system with swapping. In demand paging, a page is delivered into the memory on demand i.e., only when a reference is made to a location on that page, Demand paging combines the feature of simple paging & implement virtual memory as it has a large virtual memory. Lazy Swapper concept is implemented in demand paging in which a page is not swapped into the memory unless it is required

IMP

16) What is deadlock and Explain methods for handling deadlock.

→ Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

A similar situation occurs in operating systems when there are two or more processes that hold some resources and wait for resources held by other(s). For example, in the below diagram, Process 1 is holding Resource 1 & waiting for resource 2 which is acquired by process 2, & process 2 is waiting for resource 1.



- Methods for of handling deadlocks - There are three approaches to deal with deadlocks -
1) Deadlock prevention
2) Deadlock avoidance
3) Deadlock detection

1) Deadlock prevention : The posibility of deadlock is excluded before making requests, by eliminating one of the necessary conditions for dadlock.

Example only allowing traffic from one direction, will exclude the possibility of blocking the road.

2) Deadlock avoidance - Operating system runs an algorithm on requests to check for a safe state Any request that may result in a deadlock is not granted.

Example - checking each car and not allowing any car that can block the road. if there is already traffic on road, then a car coming from the opposite direction can cause blockage.

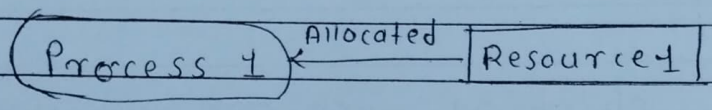
3) Deadlock detection & recovery : OS detects deadlock by regularly checking the system state, & recovers to a safe state using recovery techniques.

Example - Unblocking the road by backing cars from one side.

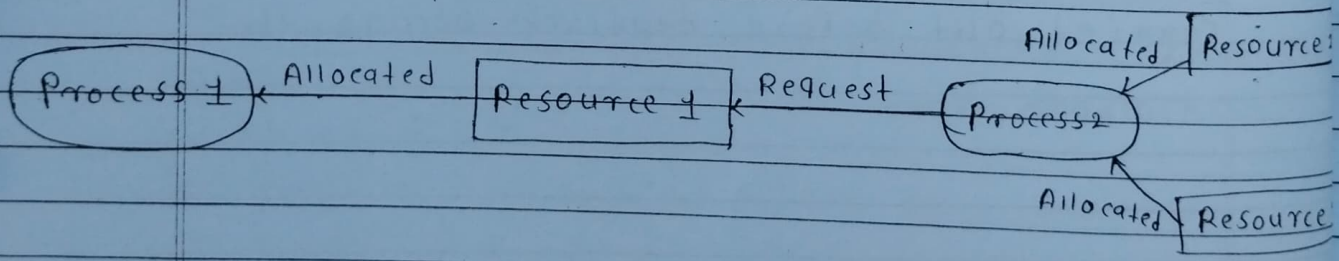
Deadlock prevention & deadlock avoidance are carried out before deadlock occurs.

17. What is deadlock ? necessar cond^n to cause it There are four conditions that must be m in order to achive deadlock.

1. Mutual Exclusion -
At least one resource must be kept in a no shareable state ; if another process requests it it must wait for it to be released.

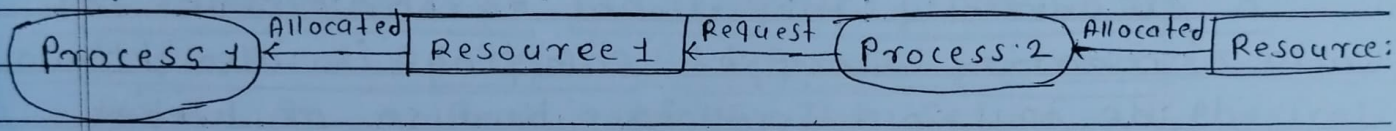


2. Hold & Wait -
A process must hold at least one resource while also waiting for at least one resource that another process is currently holding: A process can hold multiple resources & still request more resources from other processes which are holding them. In the diagram given below, process 2 holds Resource 2 & Resource 3 and is requesting the given resource 1 which is held by process 1.

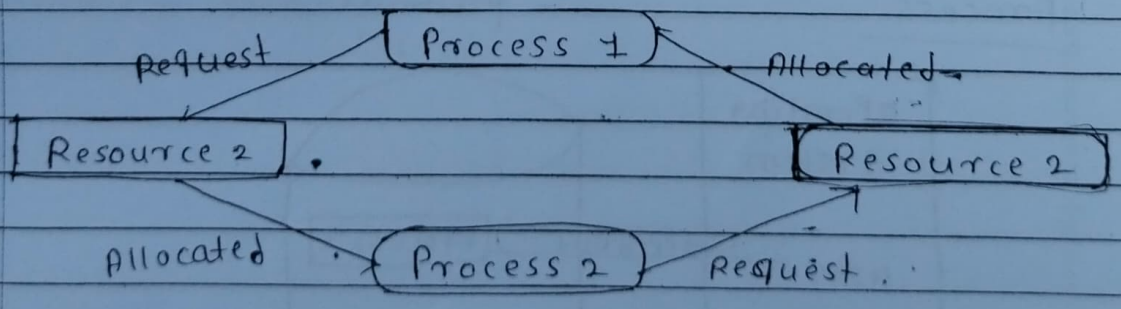


3. No Preemption - Once a process holds a resource that resource cannot be taken away from that process voluntarily release it.

A resource cannot be preempted from a process by force. A process can only release a resource voluntarity. In the diagram below, process cannot preempt Resource 1 from Process 1. It will only be released when process 1 relinquishes it voluntarity after its execution is complete.

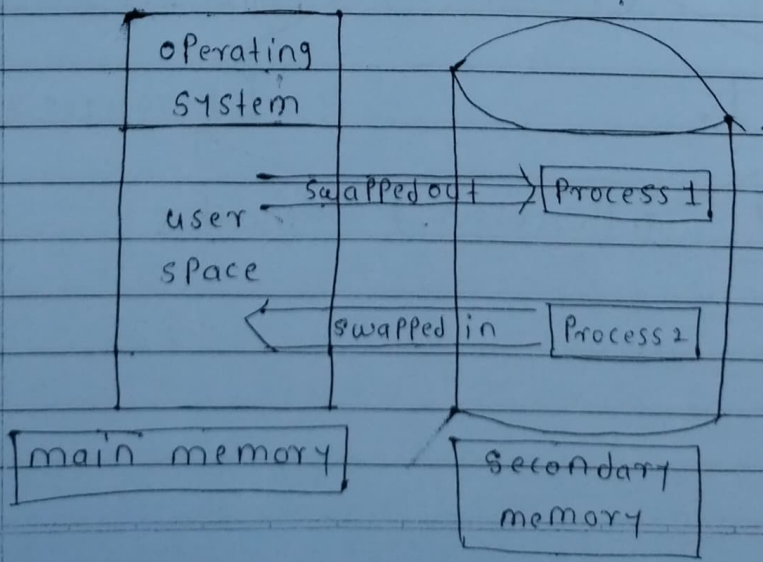


4) Circular Wait - There must be a set of processes P0, P1, P2, ----, PN such that every P[I] is waiting for P[(I+1) percent (N+1)]. (It is important to note that ~~is~~ this ~~comp~~ condition implies the hold-and-wait condition, but dealing with the four conditions is easier if they are considered separately)



18) What is Swapping & Logical and physical Address space.

Swapping - When a process is executed it may have resided in memory. Swapping is a process of swapping a ~~for~~ Process temporily into a secondary memory from the main memory, which is fast as compared to secondary memory. A swapping allows more processes to be run and can be fit into memory at one time. The main part of swapping is transferred time & the total tim is directly proportional to the amount of memory swapped. Swapping is also known as roll-out, roll in, because of higher priority process arrives and wants service, the memory manager can swap out the lower priority process & then load & execute the higher priority process. After finishing higher priority work, the lower priority process. After finishing higher priority work, the lower priority process swapped back in memory & continued to the executio process.



logical & physical address space -

- logical Address Space - An address generated by the CPU is known as a "logical Address". It is also known as a virtual address. Logical address space can be defined as the size of the porar process. A logical address can be changed.

- Physical Address Space - An address generated by the CPU i seen by the memory unit (i.e. the one loaded into the momory address register of the memory) is commonly known as a "physical Address". A physical address is also know as a Real address. The set of all physical addresses corresponding to these logical addresses is known as physical address space. A physical address is computed by NMU. The run-Time mapping from virtual to physical addresses is done by a hardware device memory management Unit (MMU). The physical address always remains constant.

- tinking and loading -

2. Counting Semaphores - Counting Semaphores are signaling integers that can take on any integer value. Using these Semaphores we can coordinate access to resources and here the semaphore count is the number of resources available. If the value of the Semaphore is anywhere above 0, processes can access the critical section, or the shared resources. The number of processes that can access the resources / code is the value the Semaphore. Howeever, if the value is 0, it means that there aren't any resources that are available or the critical section is already being accessed by number of processes and cannot be ~~acced~~ accessed by more processes.

Counting Semaphores are generally used When the number of instances of a resource are more than 1, and multiple processes can access the resource.

20. Diff - Process and Thread

	Process	Thread
1)	Process takes more time to create.	Thread takes less time to create.
2)	It takes more time to complete execution and terminate.	less time to terminate.
3)	execution is very slow	execution is vary fast
4)	It takes more time to switch b/w two processes	It takes less time to switch b/w two threads
5)	Communication b/w two processes is difficult.	Communication b/w Two threads is easy
6)	Process can't share the same memory area.	Threads can share same memory area.
7)	System calls are requested to communicate each other.	System calls are not required.
8)	Process is loosely coupled	Threads are tightly coupled.
9)	It requires more resources to execute.	Threads are requires few resources to execute.

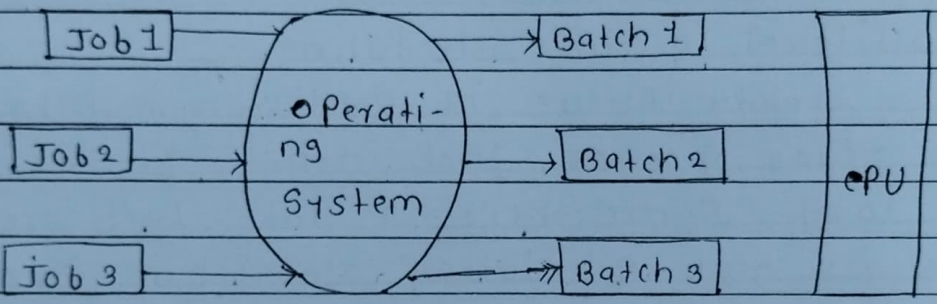
21. Define Scheduler and Explain the concept of multilevel queue scheduling.

→ Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types - 1) Long term 2) Short term 3) Medium-term.

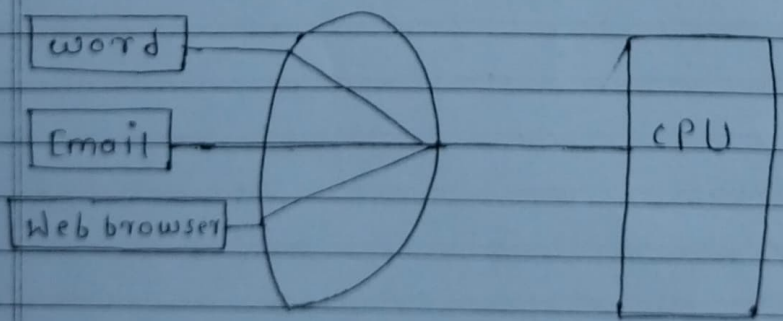
- Multilevel queue scheduling (विभाजन)
  ready queue is partitioned into number of ready queues. Each ready queue is capable to load same type of jobs. Each ready queue has its own scheduling-algorithm: ready queue is partitioned into 4 ready queues. one ready queue for system processes, one ready queue for background processes, one ready queue for fore-ground processes, background process were all empty.
  each queue gets a certain portion of the CPU time, which it can then schedule among its various processes.

22 1) Batch 2) Time Sharing 3) Real-Time.

→ 1) Batch - A batch operating system is a type of operating system that allows multiple users to use it at the same time, without direct communication between them. This is done by having the users submit their jobs to the operating system, which then processes them one at a time.

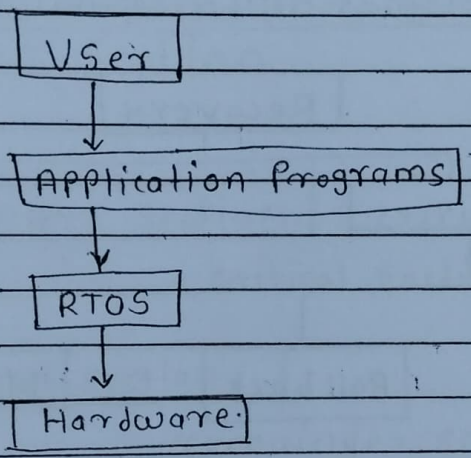


2) Time - Sharing -
The time-sharing operating system is a type of operating system in which the user can perform more than one task and each task gets the same amount of time to execute. It is also called as multitasking operating system.



3) Real - Time -

The structure A real - time system is a software system where the correct functioning of the system system depends on the results produced by the system and the time at which these results are produced.



23 recovery from deadlock -

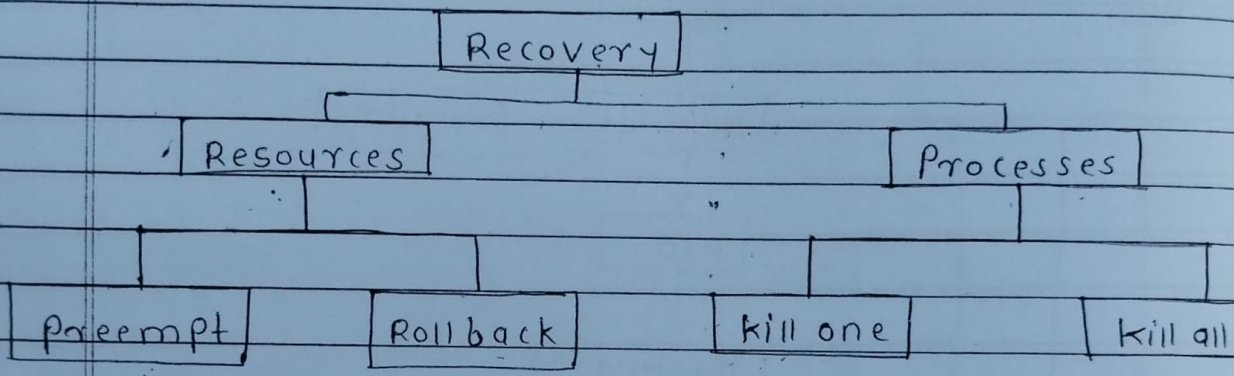
→ Deadlock recovery -

A traditional operating system such as windows doesn't deal with deadlock recovery as it is a time and space - consuming process Real - time operating systems used deadlock recovery.

1. Killing the process -

Killing all the processes involved in the deadlock killing process one by one After killing each process check for deadlock again keep repeating the process till the system recovers from deadlock killing all the processes one by one helps a system to break circular wait condition

2. Resource preemption -
Resources are preempted from the proc
involved in the deadlock, preempted resou
are allocated to other processes so that
there is possibility of recovering the
system from deadlock. In this case, th
System goes into starvation.



** 24) Critical Section problem -
→ eg. Railway Reservation System Two persons from different stations want to reserve their tickets the train number, destination is common, two persons try t get reservation at same time unfortunatel The available berths are only one, both are trying for that berths.

It is also called the critical section Problem. Solution is when one processes is executing in its critical section, no other process to be allowed to execute in its critical section.

The Critical Section problem is to design protocal that processes can use to cooperate. each processes must request permission to enter its critical section. The section of code implementing this request is the entry section. The critical section may be followed by an exist section. The remaining code is the remainder section.

```
do {
        entry section
        critical section
        exit section
        remainder section
} while (TRUE);
```

General structure of typical process Pi

Consider a system assume that consisting of n processes. Each process having a segment of code. This segment of code is said to be critical section.

* Explain at least two classic problem of Synchronization

1) classical problem of Synchronization as examples of a large class of concurrency-control problems.

1) Bounded - buffer (or producer - consumer) prob

1) Bounded Buffer problem is also called producer consumer problem.

2) This problem is generalized in terms o the proceducer - consumer problem.

3) Solution to this problem is, creating tw counting Semaphores "full" & "empty" to ke tracting Semaphores "full" & "empty" to ke tract of the current number of full & emp buffers of the current number of full & empty buffers resp.

we require to employ three Semaphores which are defined in the following table.

Semaphore	Purpose	Initial
free	mutual exclusion for buffer access	1
space	space available in buffer	N
data	data available in buffer	0

Proce producer	Explanation
produce item	· Application produces data item.
· wait (space)	It buffer full, wait for space signal.
· wait (free)	· It buffer ~~beig~~ being used wait for free signal.
· add item to buffer	· All clear : put item in next buffer slot.
· signal (free)	· signal that buffer no longer in use.

• Signal (data)	• Signal that data has been put in buffer.
* Consumer	
• wait (data)	• wait until at least one item in buffer.
• wait (free)	• If buffer being used, wait for free signal.
• get item from buffer	• All clear get item from buffer.
• Signal (free)	• Signal that buffer no longer in use.
Signal (space)	• Signal that at least one space exists in buffer.
• Consume item	• Application specific processing of item.

2. Readers and writers problem -

1) Suppose that a database is to be shared among several concurrent process.

2) Some of these processes may want only to read the database, where as others may want to update the database.

3) We distinguish between these two types of processes by refering to former as readers and to the letter as writters.

(1) First readers - writers problem - No readers be kept waiting unless a writer has already obtained permission to use the shared resource.

(2) Second readers - writers problem - once writer is ready, that writer performs its write as soon as possible.

```
/* Initialize Semaphore Variables */

integer mutex = 1;
integer DB = 1;
integer RC = 0;

1. Reader ()
Repeat continuosly
DoWN (mutex);
RC = RC + 1;
if (RC = 1) DoWN (DB);

up (mutex);
Read - Database ();
DoWN (mutex);
RC = RC - 1;
if (RC = 0) UP (DB);
up (mutex)

0. Writer ()
Repeat continuously
Down (DB);
write - Database ();
UP (DB);
End.
```