Date: / / Page :

Multiprogramming	Multiprocessing
① multiple programs ~~execute~~ are taken to RAM at a same time on a single device.	① A Computer using more than one CPU at a time.
② It uses batch OS.	② It carries multiple processors to execute the task
③ One cpu is used.	③ More than one CPU used
④ More time is taken to process the jobs.	④ Less time is taken for job processing
⑤ One process is executed at a time.	⑤ More than one process can be executed at a time.
⑥ One user at a time	⑥ Can be one or more than one user
⑦ Throughput is less	⑦ Throughput maximum
⑧ Less efficiency	⑧ Maximum efficiency
⑨	⑨

A Process is divided into several different sub processes called threads which has its own path of execution

Date: / / Page:

Multithreading	Multitasking
① Multithreading is an extended form of multitasking.	① The execution of more than one task simultaneously.
② The tasks are always further divided into sub tasks.	② It uses time sharing as the task assigned switches regularly.
③ Can be one or more than one CPU used	③ One CPU is used
④ Moderate amount of time is taken for job processing.	④ Moderate amount of time is taken for job processing.
⑤ Various components of the same process are being executed at a time	⑤ One by one job is being executed at a time.
⑥ Usually one user	⑥ More than one users.
⑦ Throughput is moderate	⑦ Throughput is moderate.
⑧ Moderate efficiency	⑧ Moderate efficiency
⑨	⑨

Date: / / Page :

Binary Semaphores

① Binary Semaphores is a semaphore whose integer value range over 0 and 1.

② 0 means that a process or a thread is accessing the critical section, other process should wait for it → 1 represents the critical section is free.

③ It guarantees mutual exclusion, since just one process or thread can enter the critical section at a time.

④ No waiting queue is present then FCFS is not followed so, Starvation is possible and busy wait present.

⑤ It doesn't guarantees bounded wait.

⑥ It can be used only for 2 processes

Structure Implementation →

⑦
```
typedef struct {
    int semaphore_variable;
} binary_semaphore;
```

Counting Semaphores

① A Counting semaphores has multiple values of the counter and the value can range over an unrestricted domain

② The value can range from 0 to N, where N is the number of process or thread that has to enter the critical section.

③ It doesnot guarantees mutual exclusion, since more than one process or thread can enter the critical section at a time.

④ Waiting queue is present then FCFS is followed so, no Starvation is possible and no busy wait present.

⑤ It guarantees bounded wait.

⑥ It can be used for any number of processes.

Structure Implementation →

⑦
```
typedef struct {
    int semaphore_variable;
    Queue list;
} Counting semaphore;
```

Synchronous	Asynchronous
① In Synchronous transmission, data is sent in form of blocks or frames.	① In Asynchronous transmission, data is sent in form of bytes or characters.
② It is faster	② It is slower
③ It is costly	③ It is economical
④ There is no gap present b/w data	④ There is gap present b/w data.
⑤ It is easy to design	⑤ It is complex to design
⑥ The start and stop bits are not used in transmitting data	⑥ The start and stop bits are used in transmitting data.
⑦ The time interval of transmission is constant.	⑦ The time interval of transmissi is not constant, it is random.
⑧ It is implemented by hardware and software	⑧ It is only implemented by hardware.
⑨ It does not need any storage at the terminal end.	⑨ It require local buffer storages at the two ends of the line to construct/assemble blocks.
⑩ Voice-band and broad-band channels are mostly used in Synchronous transmission	⑩ Voice band channels that have a narrow type are used in asynchronous transfer.

It determines which programs will enter into the RAM for processing by the CPU.

Long-term Scheduler

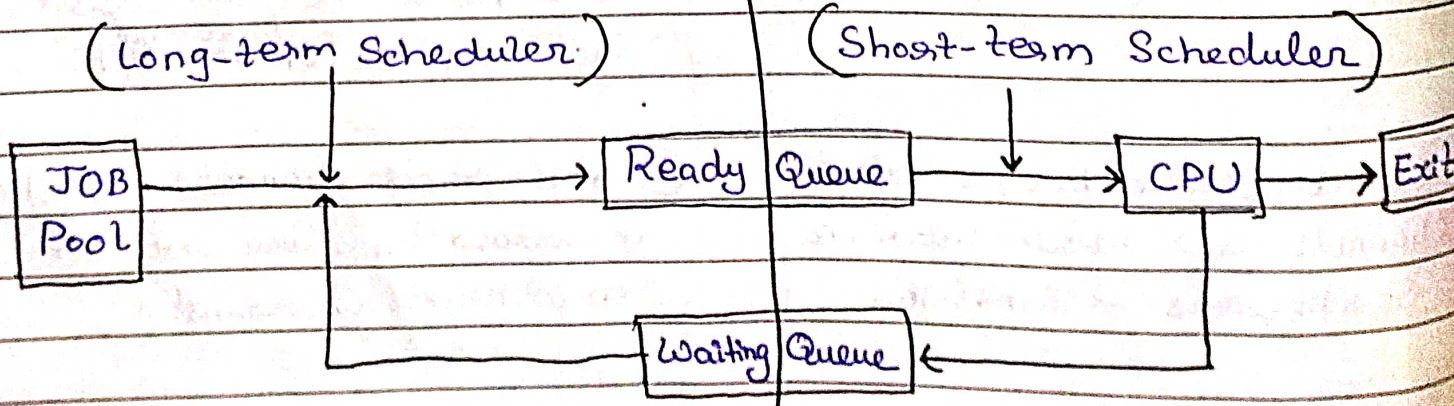
1. Long-term Scheduler is responsible for bringing processes from JOB queue (Secondary memory) into the READY queue (or main memory).
2. It is also known as Job Scheduler
3. Speed is less than ~~than~~ the Short term Scheduler
4. It changes the process state from New to Ready.
5. It control Multi-Programming
6. It controls the more DOM (Degree of Multiprogramming)
7. It controls the programs which are selected to system for processing.

It determines which program is suitable or important for processing.

Date: / / Page :

Short-term Scheduler

1. Short-term Scheduler is responsible for chooses one job from READY queue and then sends it to the CPU for processing.
2. It is also known as CPU Schedule
3. Speed is very fast as compared to long-term scheduler.
4. It changes the process state from Ready to Running
5. It control multitasking
6. It controls the less DOM (Degree of Multi-programmin
7. It ensures which program is suitable or important for processing.



Preemptive	Non-Preemptive
① In this processes are allocated for a limited time.	① Once the processor starts its execution, it must finish it before executing the other.
② Its process may be paused in the middle of the execution.	② It may not be interrupted or paused in the middle.
③ Response time is less	③ Response time is high
④ Waiting time is less	④ Waiting time is high
⑤ Cost associated	⑤ No cost associated
⑥ CPU Utilization is high	⑥ CPU Utilization is low
⑦ It affects the design of the Operating System kernel.	⑦ It doesn't affect the design of the Operating System kernel.
⑧ It has overheads associated with process scheduling	⑧ It doesn't have overhead
⑨ It is flexible	⑨ It is rigid
⑩ Round Robin and Shortest Remaining Time First are example.	⑩ Example → FCFS & SJF

Date: / / Page :

Direct Addressing Mode	Indirect Addressing Mode
① Address field contains the effective address of operand.	① Address field contains reference of effective address
② Requires only one memory reference	② Requires two memory reference.
③ It is faster than Indirect Addressing Mode	③ It is slower than direct addressing mode.
④ It occupies a smaller amount of space than the indirect mode.	④ It occupies a large amount of space than direct mode.
⑤ No additional overhead is involved while searching for operand	⑤ Additional overhead involved while searching for operand.
⑥ Address space is restricted	⑥ Requires more number of memory references.
⑦ It easily access memory	⑦ It takes times to access memory.

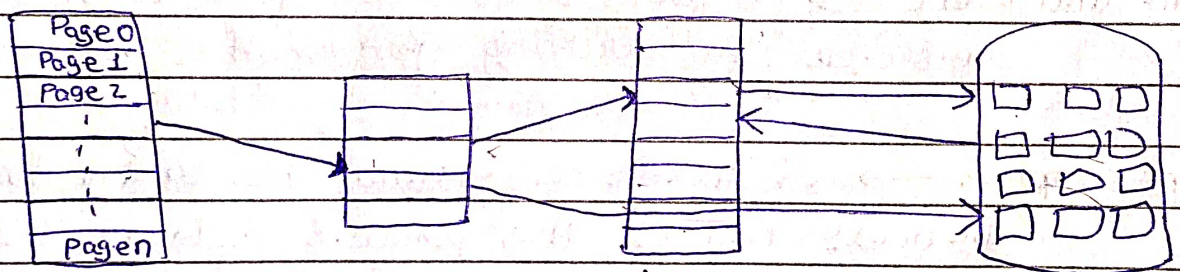
Virtual Memory.

Date: / / Page :

The virtual memory is a memory management technique which works on the concept of breaking a program into no. of small pieces and also perform swapping operation.

The concept of virtual memory helps us in executing application program whose complete size, having data storage and stack is larger than the physical memory. In this technique only those portion of the program is kept in the memory which is necessary to execute program and the remaining portion of program is kept on the disk.

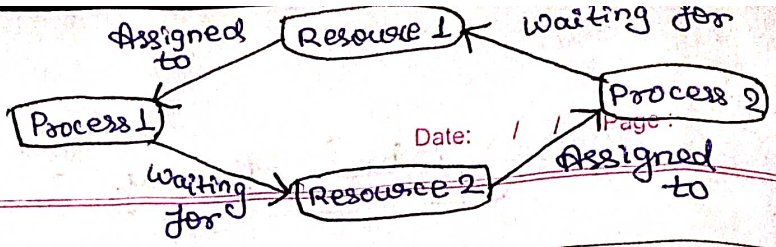
Virtual memory makes the work of programming much simpler because the programmer do not have to take care about the amount of physical memory available.



In Virtual Memory →
- CPU Utilization and throughput increases as the user utilize less physical memory.
- programmers having very large memory than physical memory.

Virtual memory can be implemented as an extension of paged or segmented memory management schome, also called demand Paging or demand segmentation.

Deadlock



Date: / /
Page:

A deadlock in OS is a situation in which more than one process is blocked because it is holding a resource and also requires some resource that is used by some other process.

Four necessary conditions for a deadlock situation to occurs are ⟶ (1) Mutual Exclusion
(2) Resource Holding and Wait
(3) No Preemption
(4) Circular Wait.

(1) Mutual Exclusion → Two or more resources are non-shareable Only one process can use at a time.

(2) Hold and Wait ⟶ A process is holding at least one resource and waiting for another resource.

(3) No preemption ⟶ A resource cannot be taken from a process unless the process release the resource

(4) Circular Wait ⟶ A set of processes are waiting for each other in circular form

Deadlock Prevention ⟶ A method assures that at least one of the four deadlock conditions never occurs.

Deadlock Avoidance ⟶ A mechanism prevents the system from coming to an unsafe state.

Banker's algorithm is the mostly used mechanism to avoid deadlock.

Date: / / Page :

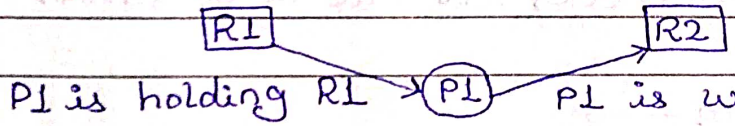
Prevention of Deadlocks →

① Eliminate Mutual Exclusion → If a ~~process~~ resource can be shared simultaneously by all processes, it cannot cause a deadlock.

It is not possible to dissatisfy the mutual exclusion because some resources, such as the tape drive and printer, are inherently non-shareable.

② Eliminate Hold and wait → If a process requests all the resources it needs at one time, it cannot be in involved in a deadlock.

Deadlock is characterised by a process holding one resource while requesting and having to wait on another. If all resources are obtained simultaneously no wait and hence no deadlock, can occur.



③ Eliminate No Preemption → If one process could remove resources from another, it would not need to wait for the resources and hence, a deadlock could not occur.

④ Eliminate Circular wait → The circular wait condition can be prevented, if resources are organised into a particular order and to require that resource requests have to follow this order.

Each resource will be assigned with a numerical number. A process can request the resources increasing/decreasing order of numbering.

Deadlock Avoidance → A System is considered safe when all processes may be assigned resources in any order without causing a deadlock.

The Deadlock Avoidance mechanism prevents the system from coming to an unsafe state. The System should be aware of the number of available resources, availability and requests to prevent the system from entering an unsafe condition.

Banker's Algorithm

Banker's Algorithm is resource allocation and deadlock avoidance algorithm which test all the request made by processes for resources, it checks for the safe state.

Inputs to Banker's Algorithm →

1. Max need of resources by each process
2. Currently, allocated resources by each process.
3. Max free available resources in the system.

The request will only be granted under the below condition→

1. If the request made by the process is less than equal to max need to that process.
2. If the request made by the process is less than equal to the freely available resource in the system.