- **Process Concept**

A process is a program in execution. A process is more than the program code, which is sometimes known as the **text section.** It also includes the current activity, as represented by the value of the **program counter** and the contents of the processor's registers. A process generally also includes the process **stack,** which contains temporary data (such as function parameters, return addresses, and local variables), and a **data section,** which contains global variables. A process may also include a **heap,** which is memory that is dynamically allocated during process run time.

A program is a *passive* entity, such as a file containing a list of instructions stored on disk (often called an **executable file),** whereas a process is an *active* entity, with a program counter specifying the next instruction to execute and a set of associated resources. A program becomes a process when an executable file is loaded into memory.

Processes and Programs

Process is a dynamic entity, that is a program in execution. A process is a sequence of information executions. Process exists in a limited span of time. Two or more processes could be executing the same program, each using their own data and resources.

Program is a static entity made up of program statement. Program contains the instructions. A program exists at single place in space and continues to exist. A program does not perform the action by itself.

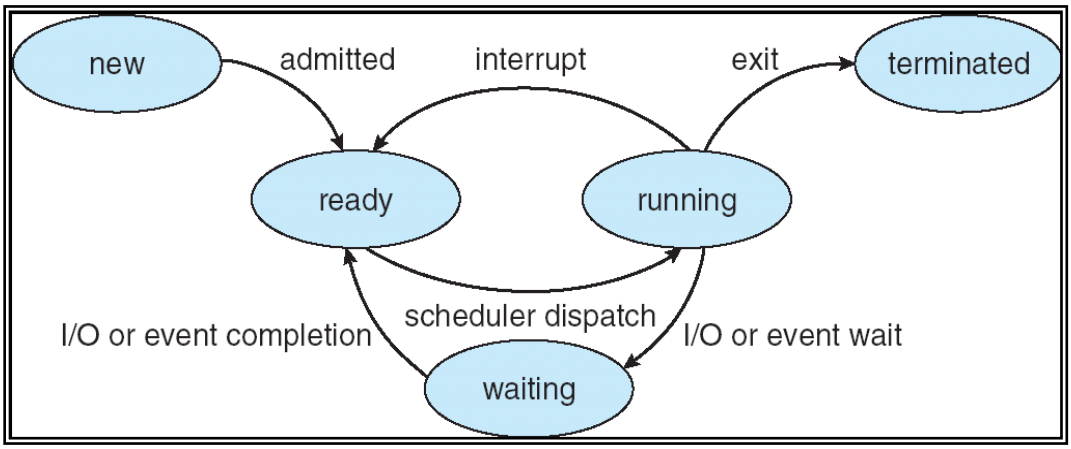
Process State

- When process executes, it changes state. Process state is defined as the current activity of the process. Fig. 3.1 shows the general form of the process state transition diagram. Process state contains five states. Each process is in one of the states. The states are listed below.

1. New
2. Ready
3. Running

4. Waiting
5. Terminated(exist)

1. New : A process that just been created.
2. Ready : Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
3. Running : The process that is currently being executed. A running process possesses all the resources needed for its execution, including theprocessor.
4. Waiting : A process that can not execute until some event occurs such as the completion of an I/O operation. The running process may become suspended by invoking an I/O module.
5. Terminated : A process that has been released from the pool of executable processes by the operating system.



Whenever processes changes state, the operating system reacts by placing the process PCB in the list that corresponds to its new state.

Only one process can be running on any processor at any instant and many processes may be ready and waiting state.

Suspended Processes

Characteristics of suspend process

1. Suspended process is not immediately available for execution.
2. The process may or may not be waiting on an event.
3. For preventing the execution, process is suspend by OS, parent process, process itself and an agent.
4. Process may not be removed from the suspended state until the agent orders the removal.

- Swapping is used to move all of a process from main memory to disk. When all the process by putting it in the suspended state and transferring it todisk.

Reasons for process suspension

1. Swapping
2. Timing
3. Interactive user request
4. Parent process request

Swapping : OS needs to release required main memory to bring in a process that is ready to execute.

Timing : Process may be suspended while waiting for the next time interval.

Interactive user request : Process may be suspended for debugging purpose by user.

Parent process request : To modify the suspended process or to coordinate the activity of various descendants.

Process Control Block (PCB)

- Each process contains the process control block (PCB). PCB is the data structure used by the operating system. Operating system groups a ll information that needs about particular process

Pointer	Proces s State
Process Number	
Program Counter	
CPU registers	
Memory Allocation	
Event Information	
List of open files	

1. Pointer : Pointer points to another process control block. Pointer is used for maintaining the scheduling list.
2. Process State : Process state may be new, ready, running, waiting and so on.

3. Program Counter : It indicates the address of the next instruction to be executed for this process.
4. Event information : For a process in the blocked state this field contains information concerning the event for which the process is waiting.
5. CPU register : It indicates general purpose register, stack pointers, index registers and accumulators etc. number of register and type of register totally depends upon the computer architecture.
6. Memory Management Information : This information may include the value of base and limit register. This information is useful for deallocating the memory when the process terminates.
7. Accounting Information : This information includes the amount of CPU and real time used, time limits, job or process numbers, account numbers etc.

- Process control block also includes the information about CPU scheduling, I/O resource management, file management information, priority and so on.
- The PCB simply serves as the repository for any information that may vary from process to process.
- When a process is created, hardware registers and flags are set to the values provided by the loader or linker. Whenever that process is suspended, the contents of the processor register are usually saved on the stack and the pointer to the related stack frame is stored in the PCB. In this way, the hardware state can be restored when the process is scheduled to run again.

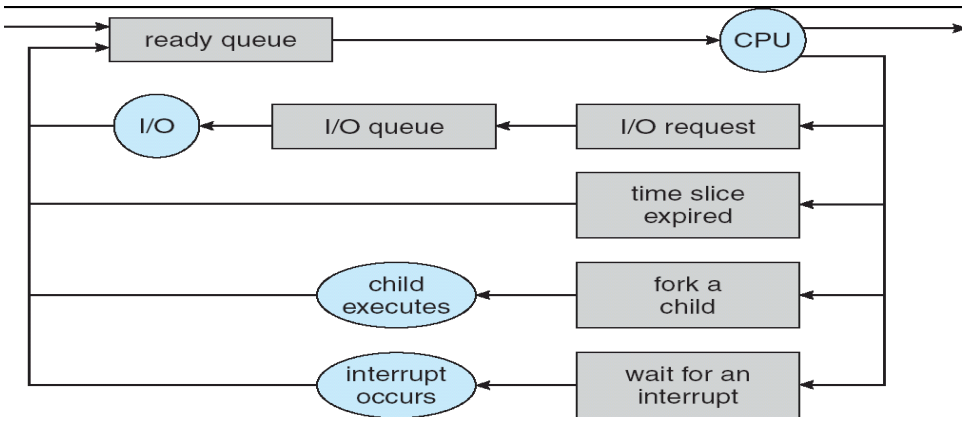
Process Scheduling

- Multiprogramming operating system allows more than one process to be loaded into the executable memory at a time and for the loaded process to share the CPU using time multiplexing.
- The scheduling mechanism is the part of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of particular strategy.

Scheduling Queues

- When the process enters into the system, they are put into a job queue. This queue consists of all processes in the system. The operating system also has other queues.
- Device queue is a queue for which a list of processes waiting for a particular I/O device.

Each device has its own device queue. Fig. 3.3 shows the queuing diagram of process scheduling.



Queues are of two types : ready queue and set of device queues. A newly arrived process is put in the ready queue. Processes are waiting in ready queue for allocating the CPU. Once the CPU is assigned to the process, then process will execute. While executing the process, one of the several events could occur.

- The process could issue an I/O request and then place in an I/O queue.
- The process could create new sub process and waits for its termination.
- The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue.

Two State Process Model

- Process may be in one of two states :
  1. Running
  2. Not Running

- when new process is created by OS, that process enters into the system in the running state.
- Processes that are not running are kept in queue, waiting their turn to execute. Each entry in the queue is a printer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then select a process from the queue to execute.

Schedules

- Schedulers are of three types.

1. Long Term Scheduler
2. Short Term Scheduler
3. Medium Term Scheduler

Long Term Scheduler

- It is also called job scheduler. Long term scheduler determines which programs are admitted to the system for processing. Job scheduler selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduler. The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.
    - On same systems, the long term scheduler may be absent or minimal. Time- sharing operating systems have no long term scheduler. When process changes the state from new to ready, then there is a long termscheduler.

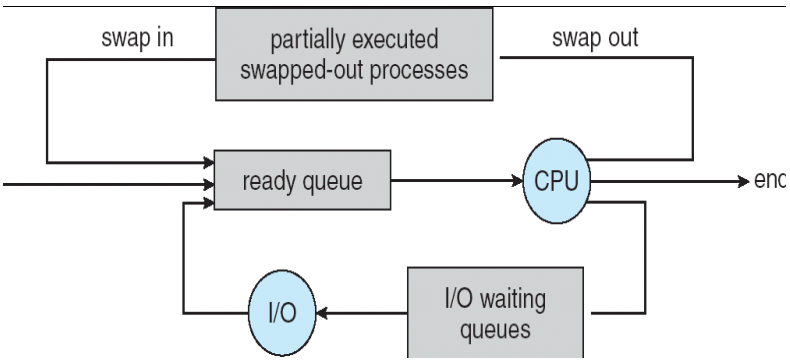
Short Term Scheduler

- It is also called CPU scheduler. Main objective is increasing system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects from among the processes that are ready to execute and allocates the CPU to one of them.

- Short term scheduler also known as dispatcher, execute most frequently and makes the fine grained decision of which process to execute next. Short term scheduler is faster than long tern scheduler.

Medium Term Scheduler

- Medium term scheduling is part of the swapping function. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium term scheduler is in charge of handling the swapped out-processe



Running process may become suspended by making an I/O request. Suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other process. Suspended process is move to the secondary storage is called swapping, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

Comparison between Scheduler

Sr. No.	Long Term	Short Term	Medium Term
1	It is job scheduler	It is CPU Scheduler	It is swapping
2	Speed is less than short term scheduler	Speed is very fast	Speed is in between both
3	It controls degree of multiprogramming	Less control over degree of multiprogramming	Reduce the degree of multiprogramming.
4	Absent or minimal in time sharing system.	Minimal in time sharing system.	Time sharing system use medium term scheduler.
5	It select processes from pool and load them into memory for execution.	It select from among the processes that are ready to execute.	Process can be reintroduced into memory and its execution can be continued.
6	Process state is (New to Ready)	Process state is (Ready to Running)	-
7	Select a good process, mix of I/O bound and CPU bound.	Select a new process for a CPU quite frequently.	-

Context Switch

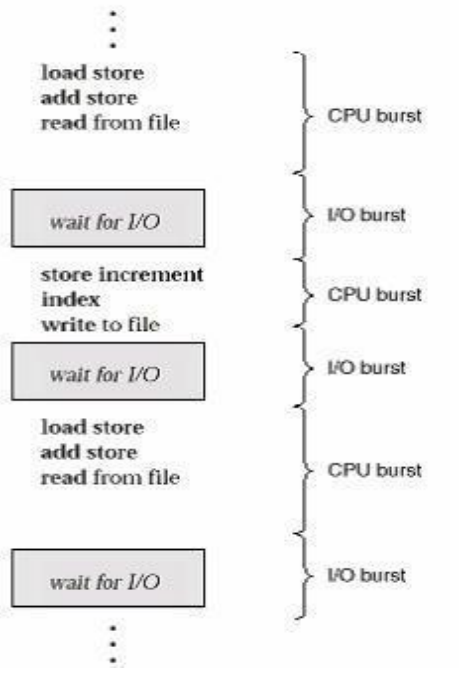
- When the scheduler switches the CPU from executing one process to executing another, the context switcher saves the content of all processor registers for the process being removed from the CPU in its process being removed from the CPU in its process descriptor. The context of a process is represented in the process control block of a process. Context switch time is pure overhead. Context switching can significantly affect performance, since modern computers have a lot of general and status registers to be saved.

- Content switch times are highly dependent on hardware support. Context switch requires (n + m) bXK time units to save the state of the processor with n general registers, assuming b store operations are required to save register and each store instruction requires K time units. Some hardware systems employ two or more sets of processor registers to reduce the amount of context switching time.
- When the process is switched the information stored is :

1. Program Counter
2. Scheduling Information
3. Base and limit register value
4. Currently used register
5. Changed State
6. I/O State
7. Accounting

Burst Cycle

The success of CPU scheduling depends on an observed property of processes: Process execution consists of a cycle of CPU execution and wait. Processes alternate between these two states. Process execution begins with a CPU burst. That is followed by an burst, which is followed by another CPU burst, then another burst, and so on. Eventually, the final CPU burst ends with a system request to terminate execution.



The durations of CPU bursts have been measured extensively. Although they vary greatly from process to process and from computer to computer, they to have a frequency curve similar to that shown in Figure . The curve is generally characterized as exponential or hyperexponential, with a large number of short CPU bursts and a small number of long CPU bursts. An program typically has many short CPU bursts. A CPU-bound program might have a few long CPU bursts. This distribution can be important in the selection of an appropriate CPU-scheduling algorithm.

CPU Scheduler

Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed. The selection process is carried out by the short-term scheduler (or CPU scheduler). The scheduler selects a process from the processes in memory that are ready to execute and allocates the CPU to that process.

Note that the ready queue is not necessarily a first-out queue. As we shall see when we consider the various scheduling algorithms, a ready queue can be implemented as a FIFO queue, a priority queue, a tree, or simply an unordered linked list. Conceptually, however, all the processes in the ready queue are lined up waiting for a chance to run on the CPU. The records in the queues are generally process control blocks (PCBs) of the processes.

Preemptive Scheduling

CPU-scheduling decisions may take place under the following four stances:

1. When a process switches from the running state to the waiting state (for example, as the result of an request or an invocation of wait for the termination of one of the child processes)
2. When a process switches from the running state to the ready state for example, when an interrupt occurs)
3. When a process switches from the waiting state to the ready state (for example, at completion of i/o
4. When a process terminates

For situations 1 and 4, there is no choice in terms of scheduling. A new process (if one exists in the ready queue) must be selected for execution. There is a choice, however, for situations 2 and 3. When scheduling takes place only under circumstances 1 and 4, we say that the scheduling scheme is or cooperative; otherwise, it is preemptive. Under nonpreemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state. This scheduling method was vised by Microsoft Windows 3.x; Windows 95 introduced preemptive scheduling, and all subsequent versions of Windows operating systems have used preemptive scheduling. The Mac OS X operating system for the Macintosh uses preemptive scheduling; previous versions of the Macintosh operating

system relied on cooperative scheduling. Cooperative scheduling is the only method that can be used on certain hardware platforms, because it does not require the special hardware (for example, a timer) needed for preemptive scheduling.

Scheduling Criteria

Different CPU scheduling algorithms have different properties, and the choice of a particular algorithm may favor one class of processes over another. In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms.

Many criteria have been suggested for comparing CPU scheduling algo-rithms. Which characteristics are used for comparison can make a substantial difference in which algorithm is judged to be best. The criteria include the following:

- **CPU utilization**. We want to keep the CPU as busy as possible. Concep-tually, CPU utilization can range from 0 to 100 percent. In a real system, it should range from 40 percent (for a lightly loaded system) to 90 percent (for a heavily used system).

- **Throughput**. If the CPU is busy executing processes, then work is being done. One measure of work is the of processes that are completed per time unit, called throughput. For long processes, this rate may be one process per hour; for short transactions, it may be 10 processes per second.

- **Turnaround time**. From the point of view of a particular process, the important criterion is how long it takes to execute that process. The interval from the time of submission of a process to the time of completion is the turnaround time. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing i/o.

- **Waiting time.** The CPU scheduling algorithm does not affect the amount of time during which a process executes or does it affects only the amount of time that a process spends waiting in the ready queue. Waiting time is the sum of the periods spent waiting in the ready queue.

- **Response time.** In an interactive system, turnaround time may not be the best criterion. Often, a process can produce some output fairly early and can continue computing new results while previous results are being output to the user. Thus, another measure is the time from the submission of a request until the first response is produced. This measure, called response time, is the time it takes to start responding, not the time it takes to output the response. The turnaround time is generally limited by the speed of the output device.