Process

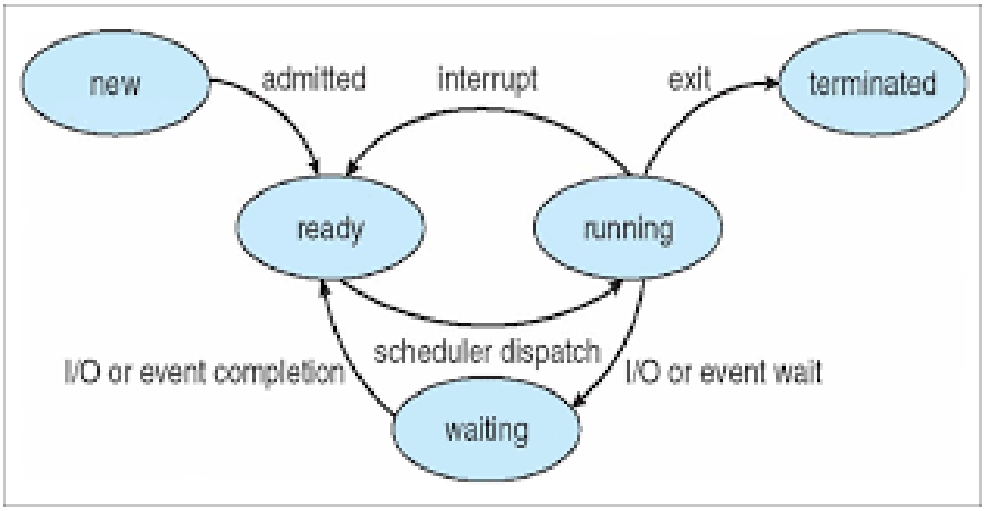
A program in execution is called process. A process is more than the set of instructions, besides it contains memory area, local, global variable, CPU registers, Program Counters.

- Process is an active entity that requires a set of resources, including a processors, program counters, registers, registers to perform its functions. Multiple process may associate with one program
- Each process has its own address space. The address space is divided into following regions :
  1. **Text Region:** It contains the code that the process executes.
  2. **Data Region:** It stores the global variables and dynamically allocated memory that process uses during execution.
  3. **Stack Region:** It stores instructions and global variable for active procedure call.
- Process contains the Heap area for dynamic memory allocations and Stack contains the program counter.
- Process is an Active entity.

Process State Transitions

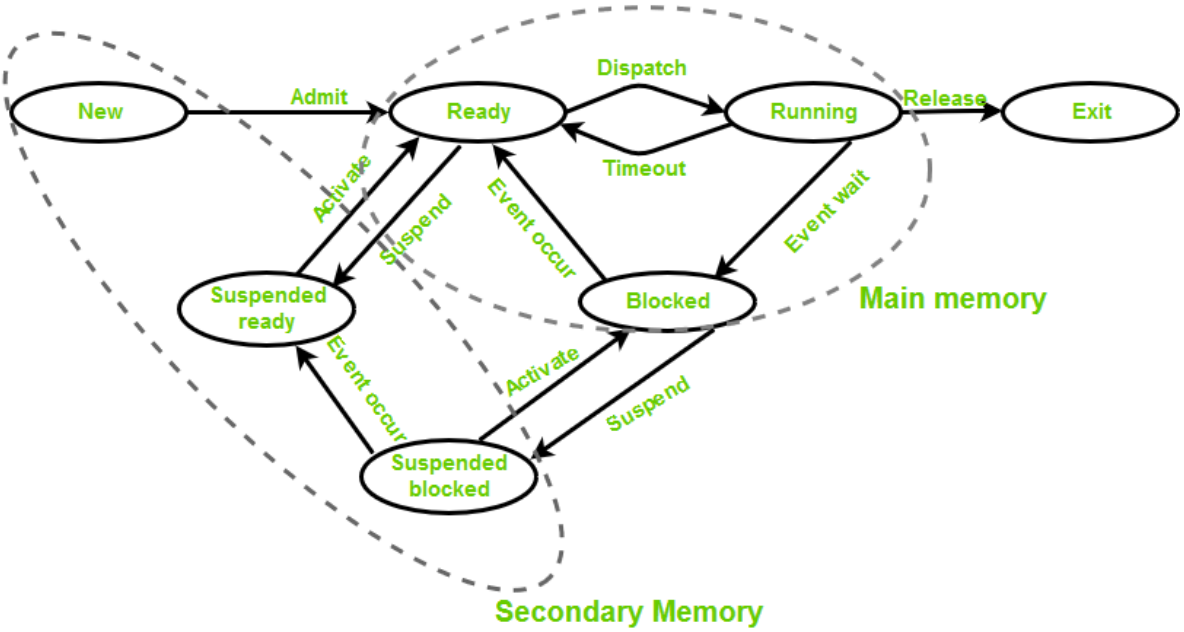
A process remains in one of the following state during its life time.

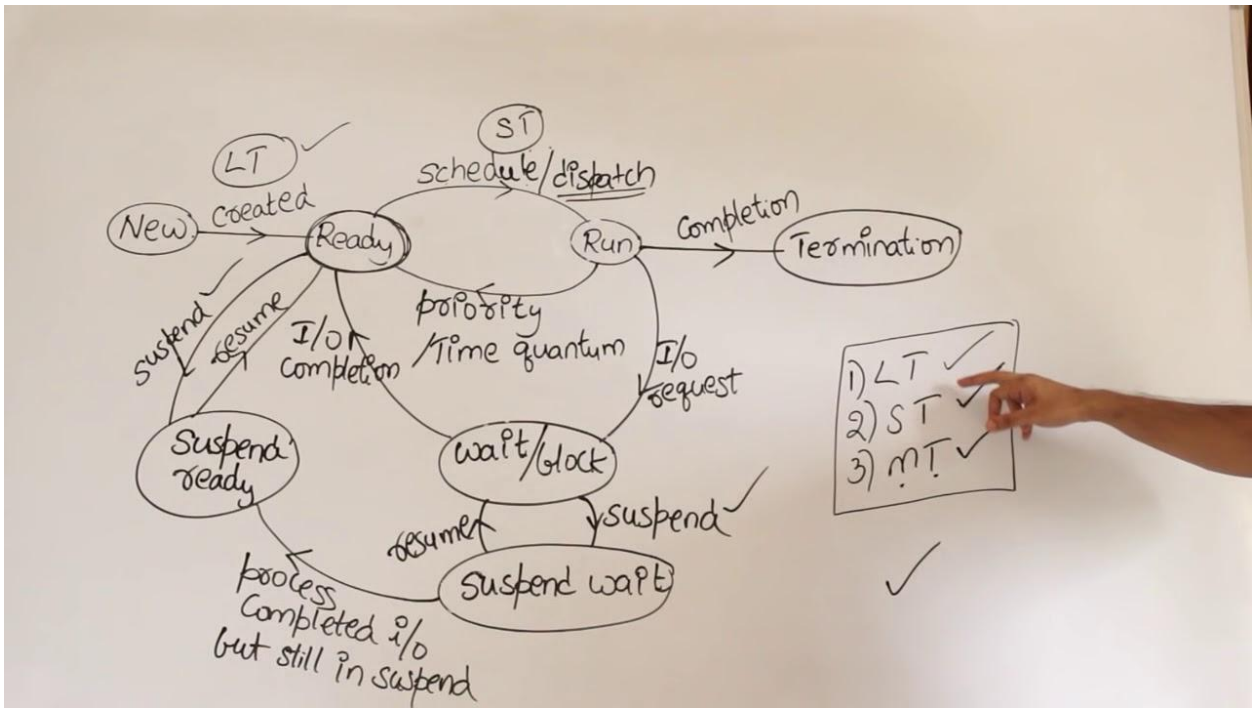
- **NEW:** The Process is being created.
- **READY:** Process is competing for the CPU. Process is ready and waiting for processor's turn.
- **RUNNING:** Process is currently being executed. OS allocate all H/W and S/W resources to the process.
- **WAITING:** Process is waiting for some I/O device and for event to occur in the queue.
- **EXIT:** A process is completed and releases all the allocated resources.



Process transition states:

- Awake (Process-name) : New -> Ready
- Dispatch (Process-name): ready -> running
- Timerrunout (Process-name): running -> ready
- Block(Process-name): running -> blocked
- wakeup(Process-name): blocked-> ready





- **New (Create) –** In this step, the process is about to be created but not yet created, it is the program which is present in secondary memory that will be picked up by OS to create the process.
- **Ready –** New -> Ready to run. After the creation of a process, the process enters the ready state i.e. the process is loaded into the main memory. The process here is ready to run and

is waiting to get the CPU time for its execution. Processes that are ready for execution by the CPU are maintained in a queue for ready processes.

- **Run –** The process is chosen by CPU for execution and the instructions within the process are executed by any one of the available CPU cores.
- **Blocked or wait –** Whenever the process requests access to I/O or needs input from the user or needs access to a critical region(the lock for which is already acquired) it enters the blocked or wait state. The process continues to wait in the main memory and does not require CPU. Once the I/O operation is completed the process goes to the ready state.
- **Terminated or completed –** Process is killed as well as PCB is deleted.
- **Suspend ready –** Process that was initially in the ready state but were swapped out of main memory (refer Virtual Memory topic) and placed onto external storage by scheduler are said to be in suspend ready state. The process will transition back to ready state whenever the process is again brought onto the main memory.
- **Suspend wait or suspend blocked –** Similar to suspend ready but uses the process which was performing I/O operation and lack of main memory caused them to move to secondary memory. When work is finished it may go to suspend ready.

PROCESS CONTROL BLOCK:

Each process is represented in the OS by a PROCESS CONTROL BLOCK (PCB)- also called Task control Block.

It contains many piece of information associated with a specific process, including these:

- **Process State:** The state may be New, Ready, Running, Waiting, Halted and so on.
- **Program Counter:** The Program Counter contains the address of the next instruction to be executed.

- **CPU Registers:** The registers vary in size, numbers and type depending on the computer architecture. They include accumulator, index registers, stack pointer and general purpose register and any conditional code information.

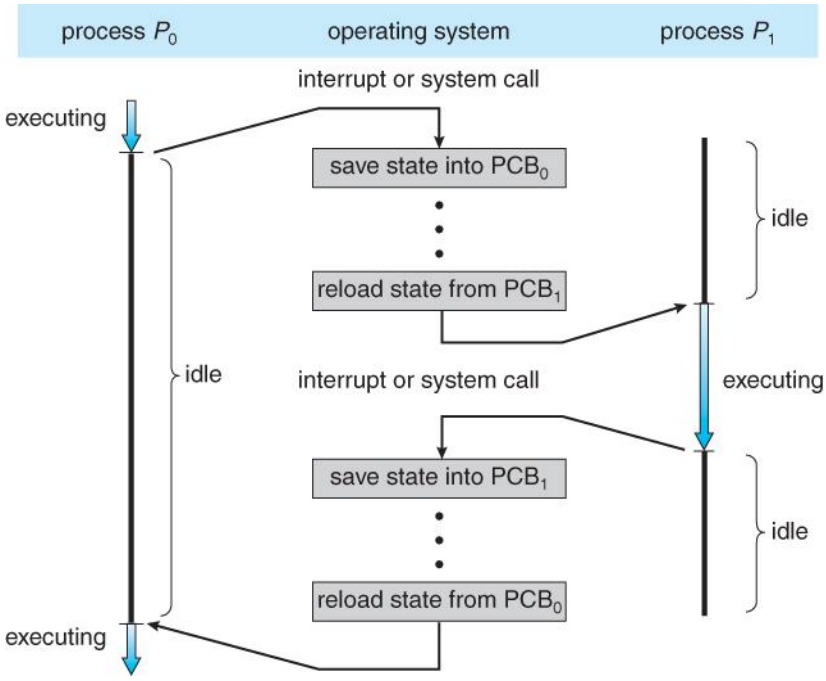
  Also with the program counter, this state information is saved when an interrupt occurs to allow the process to be continued correctly afterwards.
- **CPU scheduling Information:** This includes a process priority, pointers to scheduling queues and any other scheduling parameters.
- **Memory Management Information:** This information may include the value of the BASE and LIMIT registers, page table, or segment tables depending on the memory system used by OS.
- **Accounting Information** : This includes the amount of CPU and real time used, time limits, account numbers, job and process number and so on…
- **I/O status Information:** The information includes the list of I/O devices allocated to this process, a list of open files and so on.

The PCB serves as a repository for any information that may ary from process to process. Because PCBs need to be manipulated quickly by the OS, many computer systems contain a H/W register that always points to the PCB of the currently executing process.

Hardware instructions are often available that load state information into the PCB and restore the information quickly.

PROCESS SCHEDULING:

To maximize CPU utilization some process should continuously running at all times, which is the objective of multiprogramming. The objective of the time sharing is to switch the CPU among processes so frequently the users can interact with each program while it is running. For a uni-processor system, there will never be more processes, the rest will have to wait until the CPU is free and can be rescheduled.

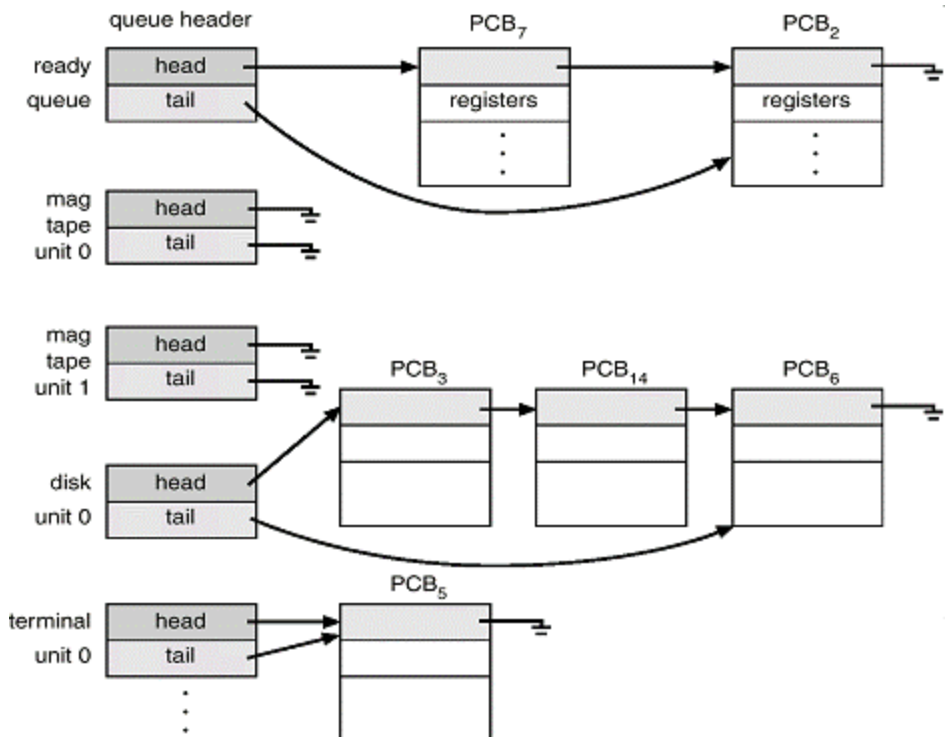


Scheduling Queues:

As processed enter the system, they are put into the job queue. Thus queue consists of all the processed in the system. The processes that are residing in main memory and ready to execute are kept in Ready queue. This queue is generally stored as a linked list. A ready queue Header will contain pointers to the first and last PCB of the list.

“Each PCB has a pointer field that points to the next process in the ready queue.”

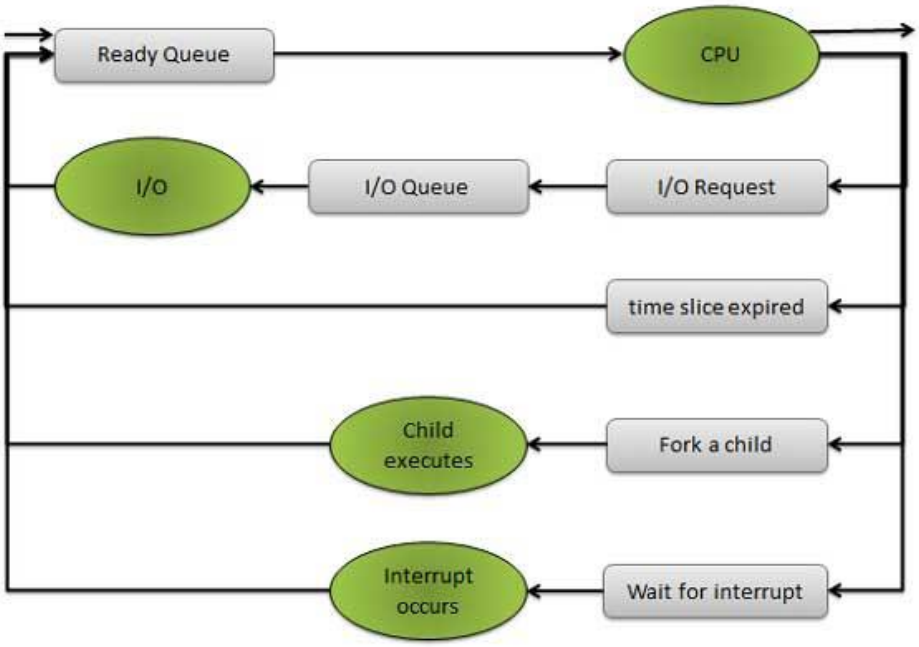
The list of processes waiting for a particular I/O device is called a Device Queue. Each device has its own Device Queues.



(Ready Queue and Various I/O Device Queue)

A Common representation of Process scheduling is a queuing diagram.

- Each Rectangle Box Represents a Queue.
- Two types of Queue are present: Ready Queue and Device Queue.
- The circles represent the queue that serves the queue.
- The arrow indicates the flow of process in the system.
- A new process is initially put in the ready queue. It waits in the ready queue until it is selected for execution (Dispatched) and is given the CPU. Once the CPU is allocated to the process and is executing one of the several events could occur:
  - **i.** The process could issue an I/O request and then be placed in I/O Queue.
  - **ii.** The process could create a new sub process and waits for its termination.
  - **iii.** The Process could be removed forcibly from the CPU as a result of interrupt, and be put back in the ready queue.



In the first two cases, the process eventually switched from the waiting state to the ready state, and is then put back in the ready queue. A process continues this cycle until it terminates, at which time it is removed from the queues and has its PCB and resources reallocated.

SCHEDULERS

Schedulers are special system software which handles 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 −

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

Long Term Scheduler

It is also called a **job scheduler**. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

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 some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

Short Term Scheduler

It is also called as **CPU scheduler**. Its main objective is to increase 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 a process among the processes that are ready to execute and allocates CPU to one of them.

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

Medium Term Scheduler

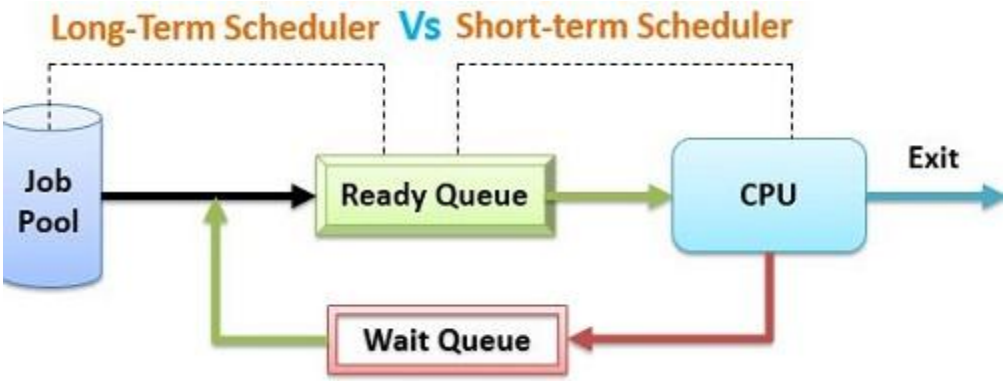
Medium-term scheduling is a part of **swapping**. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.

A running process may become suspended if it makes an I/O request. A suspended process cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called **swapping**, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

A process is selected from queues by the appropriate scheduler.

- In a batch system, more processes are submitted than can be executed immediately. These process are spooled to a mass storage device (typically a disk), where they are kept for later execution.
- The **short term scheduler** (**CPU SCHEDULER**) select from among the process that ready to execute, and allocate the CPU to one of them.
- The Primary distinction between these schedulers is the frequency of their execution. The short term schedulers must select a new process for the CPU quite frequently. Because of the short duration of time between executions, the short term scheduler must be very fast. Short term scheduler execute at least once in every 100 milliseconds.

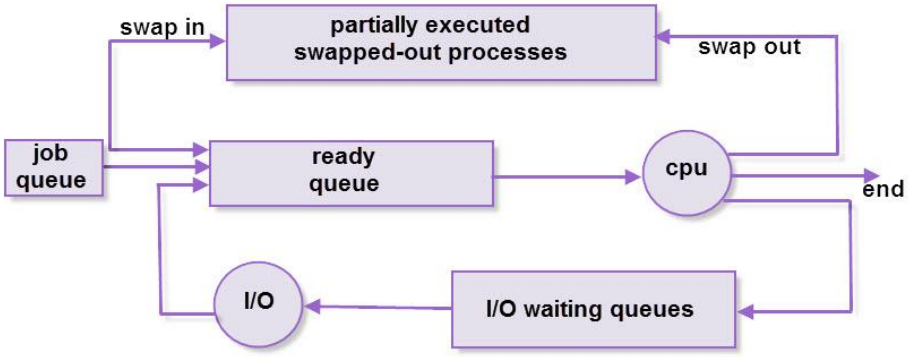
- The long term scheduler on the other hand execute much less frequently. There may be minutes between the creations of new processes in the system. The long term scheduler controls the degree of multiprogramming (the no. of processes in the memory). If the degree of multiprogramming is stable, then the average rate of processes creation must be equal to the average departure rate of processes leaving the system.
- Thus, the long term schedulers may need to be invoked only when a process leaves the system. Because of the longer interval between executions the long term scheduler can afford to take more time to decide which process should be selected for execution.
- The long term scheduler select good process mix of I/O bound and CPU Bound processes. If all processes are I/O bound, the ready queue will almost be empty and short term scheduler will have little to do.



- If all the processes are CPU bound, the I/O waiting queue will almost be empty, devices will go unused and again the system will be unbalanced.
- The System with best performance will have a combination of CPU bound and I/O bound processes.
- On Some system long term scheduler is absent or minimal. For example Time Sharing systems often have no long term scheduler but simply put every new process in memory for the short term scheduler.
- Some Operating System such as Time Sharing System may introduce an additional intermediate level of scheduling. This **Medium Term Scheduler** sometimes removes

processes from memory (and From active contention for the CPU) and thus reduce the degree of multiprogramming.

- At Later time, the process can be reintroduced into memory and its execution can be continued where it is left off. This scheme is **Swapping**. The Process is swapped out and swapped in later by **Medium Term Scheduler.**
- **Swapping** may be necessary to improve process mix or because a change in memory requirements has over committed available memory, requiring memory to be freed up.



Addition of medium-term scheduling to the queuing diagram

Comparison among Scheduler

S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.

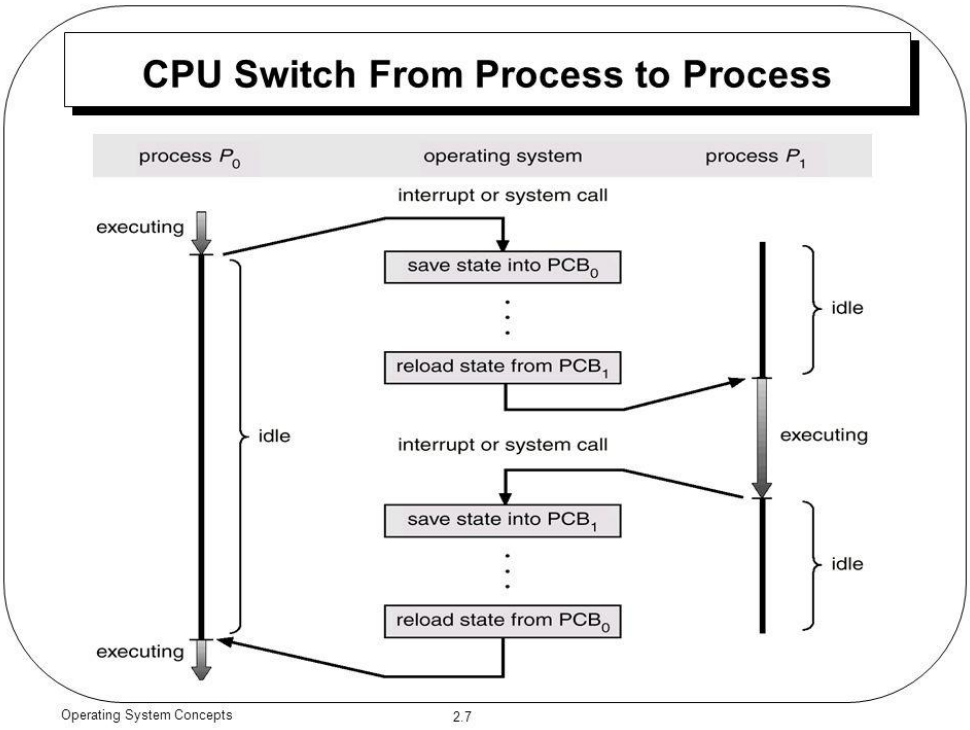
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

- **Long term – performance –** Makes a decision about how many processes should be made to stay in the ready state, this decides the degree of multiprogramming. Once a decision is taken it lasts for a long time hence called long term scheduler.
- **Short term – Context switching time –** Short term scheduler will decide which process to be executed next and then it will call dispatcher. A dispatcher is a software that moves process from ready to run and vice versa. In other words, it is context switching.
- **Medium term – Swapping time –** Suspension decision is taken by medium term scheduler. Medium term scheduler is used for swapping that is moving the process from main memory to secondary and vice versa.

Context Switch

"Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process. This task is known as context switch."

- Context switch time is pure overhead because system does no useful work while switching.



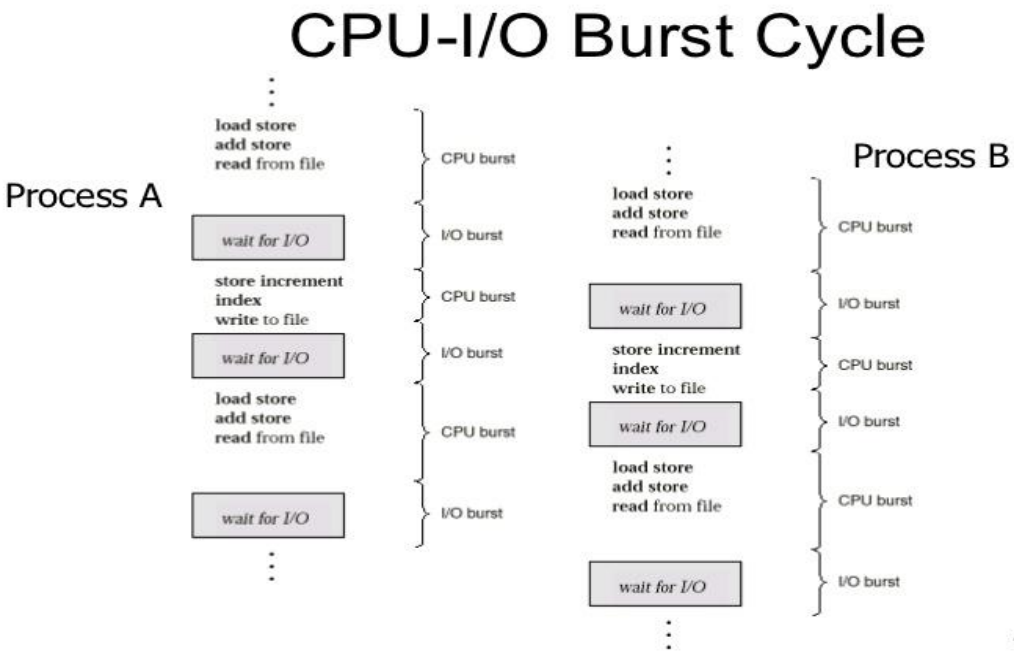
- Context switch speed various from machine to machine depending on the memory speed, the numbers of registers which must be copied and the existence of special instructions (such as a single instruction to load or store all registers)
- Typically the speed ranges from 1 to 100 milliseconds.
- Context switch time is highly dependent on H/W support. For instance, some processors provide multiple type of registers.
- A context switch simply includes changing the pointer to the current register set.

Process Scheduling

The **process scheduling** is the activity 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 a particular strategy. **Process scheduling** is an essential part of Multiprogramming **operating systems**. CPU Scheduling the Basic of multiprogramming.

In a single Processor system, only one process can run at a time; any other must wait until the CPU is free and can be rescheduled.

The Objective of Multiprogramming is to have some process running at all times to maximize the CPU utilization.



Process Scheduling:

The process scheduling is the activity 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 a particular strategy.

Process scheduling is an essential part of Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

Dispatcher

Another component involved in the CPU scheduling function is the **Dispatcher**. The dispatcher is the module that gives control of the CPU to the process selected by the **short-term scheduler**. This function involves:

- Switching context
- Switching to user mode
- Jumping to the proper location in the user program to restart that program from where it left last time.

Types of CPU Scheduling

CPU scheduling decisions may take place under the following four circumstances:

1. When a process switches from the **running** state to the **waiting** state (for I/O request or 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, completion of I/O).
4. When a process **terminates**.

In circumstances 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 in circumstances 2 and 3.

When Scheduling takes place only under circumstances **1** and **4**, we say the scheduling scheme is **non-preemptive**; otherwise the scheduling scheme is **preemptive**.

Non-Preemptive Scheduling

Under non-preemptive 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.

Preemptive Scheduling

In this type of Scheduling, the tasks are usually assigned with priorities. At times it is necessary to run a certain task that has a higher priority before another task although it is running. Therefore, the running task is interrupted for some time and resumed later when the priority task has finished its execution.

CPU Scheduling: Scheduling Criteria

There are many different criteria's to check when considering the **"best"** scheduling algorithm, they are:

- **Throughput:** It is the total number of processes completed per unit time or rather say total amount of work done in a unit of time. This may range from 10/second to 1/hour depending on the specific processes*.*
- **Turnaround Time:** It is the amount of time taken to execute a particular process, i.e. the interval from time of submission of the process to the time of completion of the process (Wall clock time).
- **Waiting Time:** The sum of the periods spent waiting in the ready queue amount of time a process has been waiting in the ready queue to acquire get control on the CPU.
- **Load Average:** It is the average number of processes residing in the ready queue waiting for their turn to get into the CPU.
- **Response Time:** Amount of time it takes from when a request was submitted until the first response is produced. Remember, it is the time till the first response and not the completion of process execution (final response).

  In general CPU utilization and Throughput are maximized and other factors are reduced for proper optimization.
- **CPU Utilization***:* To make out the best use of CPU and not to waste any CPU cycle, CPU would be working most of the time (Ideally 100% of the time). Considering a real system, CPU usage should range from 40% (lightly loaded) to 90% (heavily loaded).