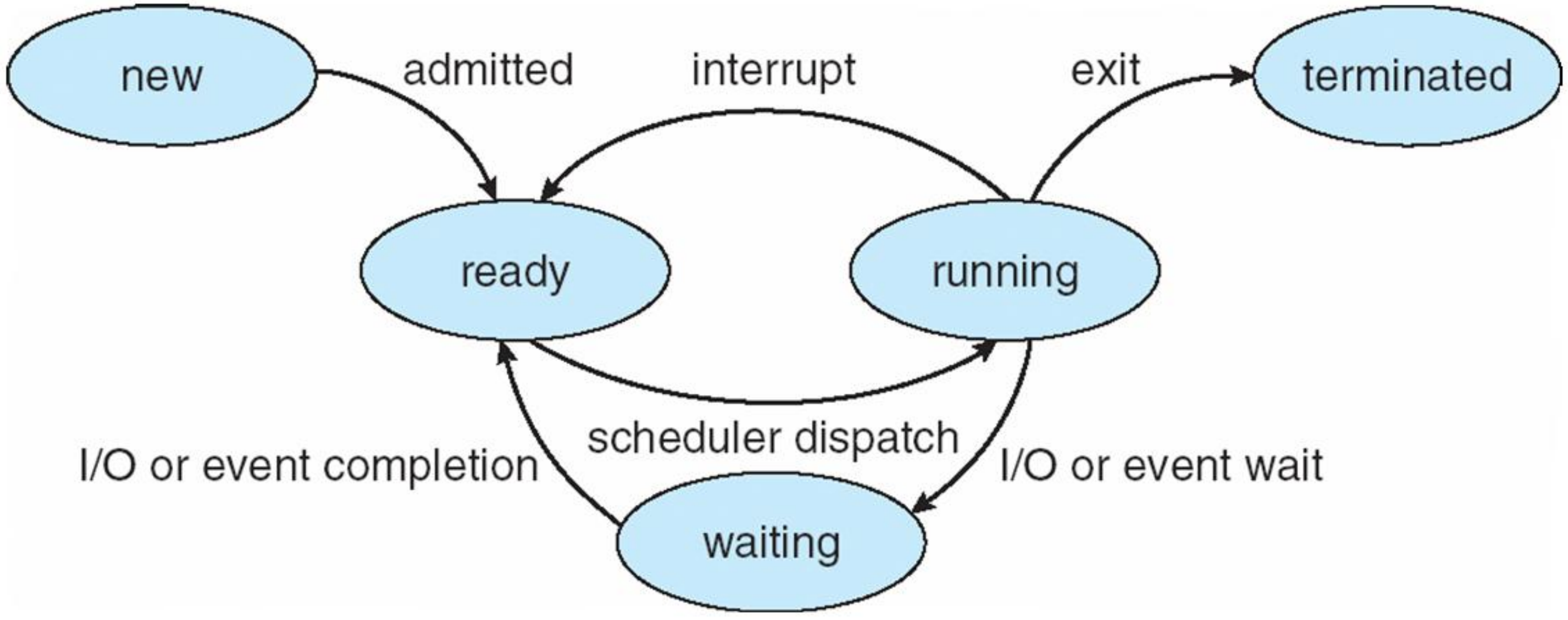
Process Concept

- An operating system executes a variety of programs:
  - Batch system – jobs
  - Time-shared systems – user programs or tasks
- Textbook uses the terms *job* and *process* almost interchangeably
- Process – a program in execution; process execution must progress in sequential fashion
- A process includes:
  - program counter
  - stack
  - data section

Process State

- As a process executes, it changes *state*
  - **new**: The process is being created
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur
  - **ready**: The process is waiting to be assigned to a processor
  - **terminated**: The process has finished execution

Diagram of Process State

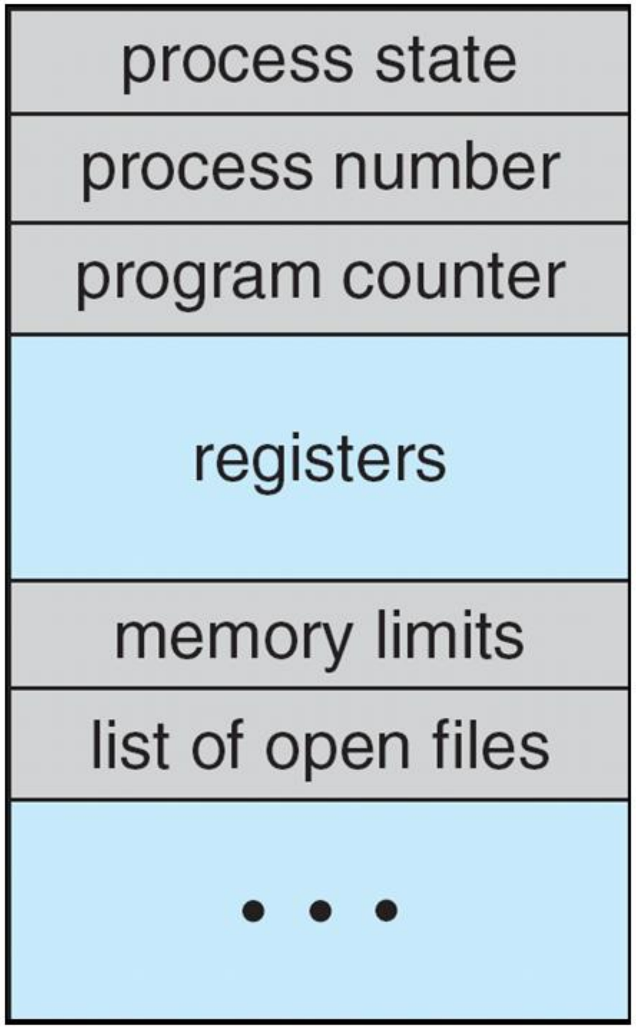


Process Control Block (PCB)

Information associated with each process

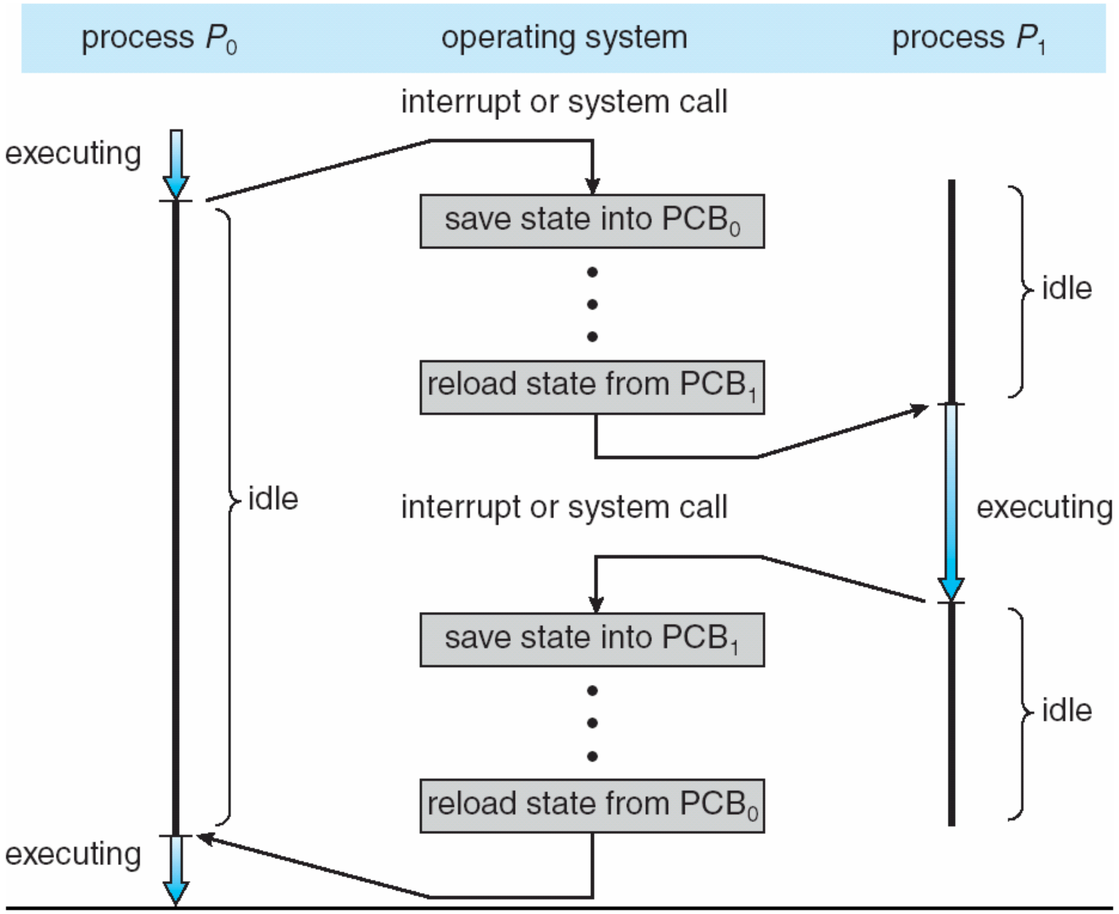
- Process state
- Program counter
- CPU registers
- CPU scheduling information
- Memory-management information
- Accounting information
- I/O status information

Process Control Block (PCB)



- It contains many pieces of information associated with a specific process.
- 1*. Process state:* States may be new ,ready, running, waiting, terminated.
- 2. *CPU registers:* Accumulator, Index registers, general purpose register…
- 3. *CPU scheduling information:* includes process priorities, pointers to queue..
- 4. *Memory management:* Base registers (Starting address of a process), Limit register (length of the Partition).
- 5*. Accounting information:* Time limits, job or process numbers.
- 6*. IO Status information:* List of IO devices allocated to process, List of open files, …

CPU Switch From Process to Process



Process Scheduling Queues

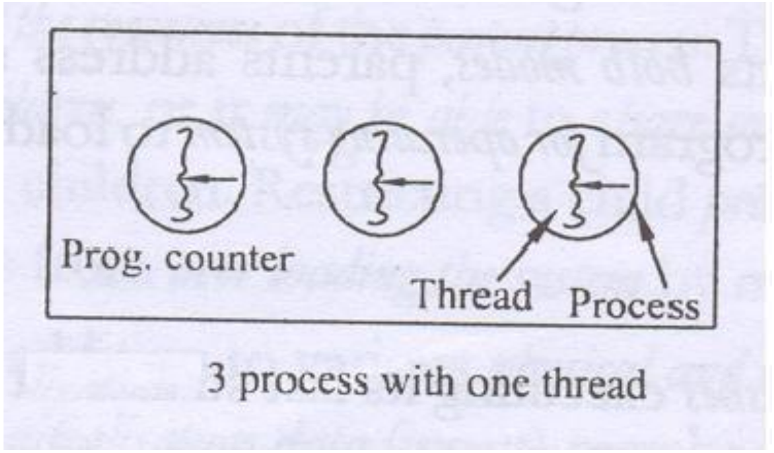
- **Job queue** – set of all processes in the system
- **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
- **Device queues** – set of processes waiting for an I/O device
- Processes migrate among the various queues

Schedulers

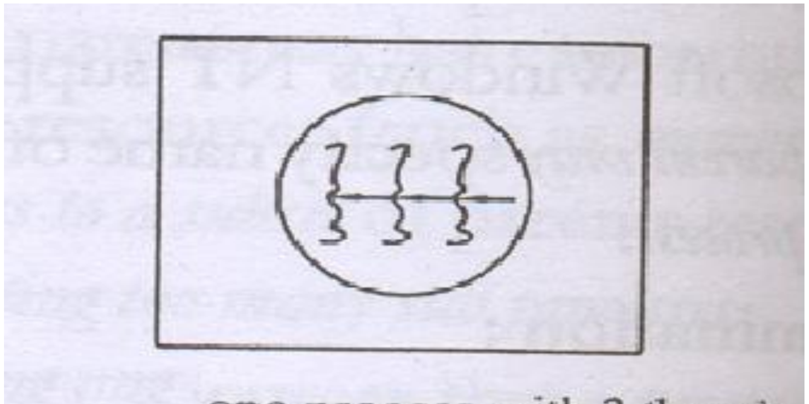
- **Long-term scheduler** (or job scheduler) – selects which processes should be brought into the ready queue
- **Short-term scheduler** (or CPU scheduler) – selects which process should be executed next and allocates CPU

Thread

- *Thread:*
- In a traditional process there is a single thread of control and a single Program counter in each process. Some OS supports for multiple threads of control within a process. These threads of control are called Threads/Light weight processes.
- Multiple threads shared resources as well as state information of a single process.



3 process with one thread



one process with 3 threads

- 1. Each process has shared its own address space and a single thread of control.

  Here each of them operates in its own address space .
- 2. Single process with 3 threads of control.

  They share the same address

- *Examples of multiple threads:*

1. File Server Process.
2. Browser forward www like internet explorer.

- *Advantages of Multithreading:*

1. *Responsiveness* : In multithreading is allow a program to continuing even if part of it is blocked/waiting a lengthy operation.
2. *Resource sharing* : Several threads within same address space.
3. *Economy* : allocation of memory and resources are costly for process creation.
4. *Utilization of multiprocessor architecture* : In multiprocessor each thread may be running in parallel on a different processor.

User and Kernel threads

- *User Threads*:
- *User threads* are supported above the kernel and are implemented by a ***thread library*** at user level. The library provides support for thread *creation, scheduling, and management* with no support from the kernel. Because the kernel is unaware of user-level threads, all thread creation and scheduling are done in user space without need for kernel.
- *Kernel threads*:
- *Kernel threads* are supported directly by the ***OS*** the kernel performs *thread creation, scheduling and management* in a kernel space. Because thread management is done by the OS, kernel threads are generally **slower** to create and manage than the user threads.

Thread Priorities

Priority	**Comment**
Thread.MIN_PRIORITY	Minimum Thread Priority
Thread.MAX_PRIORITY	Maximum Thread Priority
Thread.NORM_PRIORITY	Default Thread Priority

Priorities May Be Set Using setPriority() method:

```
setPriority(Thread.NORM_PRIORITY + 2);
```

Scheduling criteria algorithms

- Objective of multiprogramming is to have some process running at all times, to maximize CPU utilization.
- *CPU Scheduler* :
- When ever the CPU is idle, then the OS must select one of the process from the ready queue to be executed. This selection process is done through CPU scheduler. The ready queue OS not necessarily a FIFO queue.

- ***Preemptive scheduling*** :

- In CPU Scheduling decisions follows 4 circumstances

1. When a process switches from the running state to waiting state.
   (IO request, invocation of waiting for the termination of one of the child process).
2. When a process switches from the running state to the ready state.
   (When an interrupt occur).
3. When a process switches from the waiting state to the ready state. (Completion of IO).
4. When a process terminate.

- *Non-preemptive scheduling:*
- In *Non-preemptive scheduling* once the CPU has been allocated to process, the process keeps the CPU until it releases the CPU either by terminating (OR) by switching to waiting state.
- *Preemptive scheduling* is costly.
- → *Dispatcher*: The dispatcher is the module that gives control of the CPU to the process selected by the job/process scheduler.

- This function involves:
- Switching context.
- Switching to user mode.
- Jumping to the proper location in the user program to restart that program.
- The time it takes for the dispatcher to stop one process and start another running is known as the *dispatcher latency*.

- CPU scheduling algorithms have different properties from one class of process over another. Depending upon the situation, we consider the properties of the various algorithms:
- Some of the characteristics used for comparing different algorithms:
- *CPU utilization:* We want to keep the CPU as busy as possible. In a real system, it should range from 40% (for lightly used system) to 90% (for heavily loaded system).
- *Throughput:* Number of processes completed per time unit is called as throughput. Work is being done.
- *Turnaround time:* It is the time interval between the arrival of the request and completion of the request (Processing time).
- *Waiting time:* Amount of time that a process waiting in the ready queue. OR the time taken to wait a process to connect the CPU.
- *Response time:* Amount of time it takes to start responding.
- OR
- The time from the submission of a request until the first response is produced
- *OR*
- After the completion of the process then the CPU give the response to the user.

Difference between Preemptive and Non-preemptive Scheduling

Non-preemption	Preemption
• 1. In this scheduling the scheduled process is completed then only CPU allocates another process for execution.	1. A new process is schedule before completion of presently handling process.
• 2. Here priority is not considered.	2. Here priority is considered.
• 3. Here response is considering in a negative (slowly) manner.	3. Here getting response fast.
• 4. Here smaller process is waiting for long time.	4. Here smaller process is not waiting for long time

Algorithms

1. First Come First Served scheduling. (FCFS)
2. Shortest Job First scheduling. (SJF)
3. Priority scheduling.
4. Round Robin Scheduling. (RRA)

1. *First Come First Served scheduling*:

- It is *Simple*, It is based on FIFO.
- Here the process which requests the CPU first that is allocated to the CPU. It is Non preemptive.

Drawback: 1. Less performance to shortest job.
2. The average waiting time is long

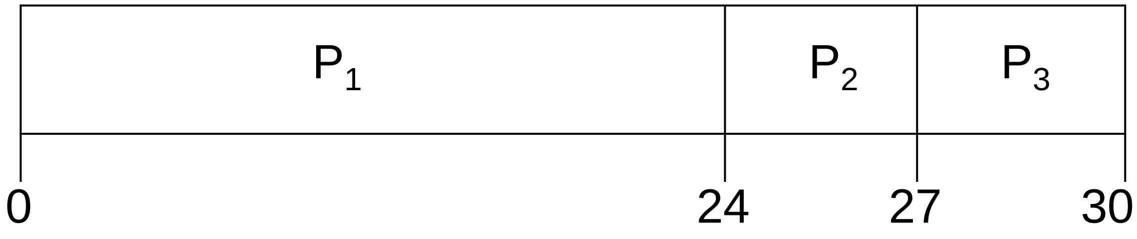
2. *Shortest Job First scheduling*:

- In this processing is of sorted order, where the shortest job will be executed first and then increasing order of execution.
- Shortest job serve first.
- If 2 processes have same length then executes FCFS.
- Due to this average waiting time is a decrease.
- SJF may be preemptive/non preemptive. A preemptive SJF algorithm will preempt the currently executing process but in non preemptive SJF algorithm will allow currently running process over.

First-Come, First-Served (FCFS) Scheduling

Process	Burst Time
P_1	24
P_2	3
P_3	3

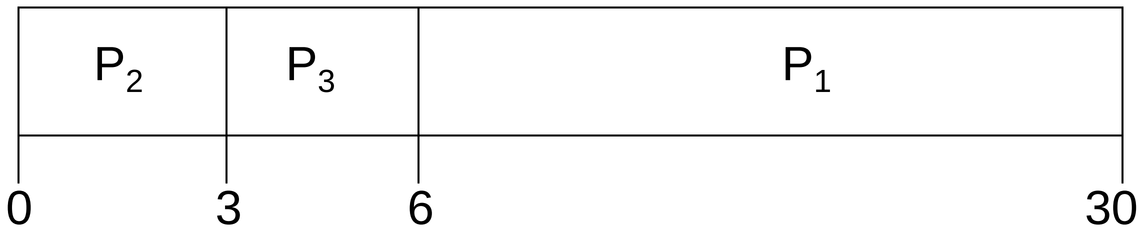
- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule is:



Suppose that the processes arrive in the order

$$P_2 , P_3 , P_1$$

- The Gantt chart for the schedule is:
  Waiting time for $P_1 = 6$; $P_2 = 0$; $P_3 = 3$
- Average waiting time: (6 + 0 + 3)/3 = 3
- Much better than previous case
- *Convoy effect* short process behind long process



SJF

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time
- SJF is optimal – gives minimum average waiting time for a given set of processes
  - The difficulty is knowing the length of the next CPU request

3. Priority scheduling :

Here every process is assigned with priority. The highest priority process is allocated to CPU first.

- Here equal priority processes are scheduled in FCFS basis.
- We use low number to represent highest priority.
- Priority scheduling may be preemptive/non preemptive.
- Drawback: Indefinite blocking (Starvation), Here Low priority process waiting more time for CPU.
- Solution of the problem of indefinite blocking of low priority process is aging.
- Aging is a technique of gradually increasing the priority of process that wait in the system for long time.

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)
  - Preemptive
  - nonpreemptive
- SJF is a priority scheduling where priority is the predicted next CPU burst time
- Problem ≡ **Starvation** – low priority processes may never execute
- Solution ≡ **Aging** – as time progresses increase the priority of the process

4. *Round Robin Scheduling:*

It is similar to FCFS Scheduling with time slice, CPU switches from one process to another process.

- For every process to be executed for a fixed amount of time slice is given. If any process was not completed at time, then it is preempted in to ready queue. After completion of next processes in the ready queue again preempted process can be executed.
- Number of time slices=CPU burst time/time slice.

→ Smaller time quantum increases the context switch.

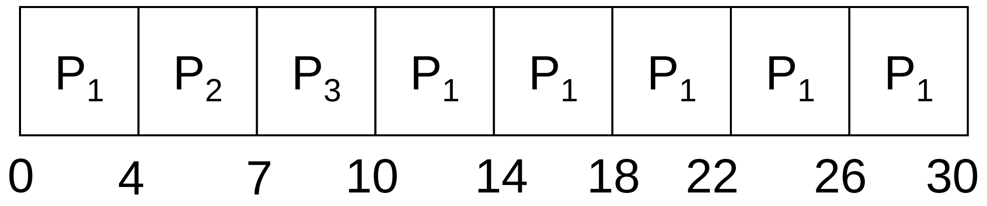
→Performance of RRA depends on size of time quantum

→If time quantum is very large (Infinite) the RRA is same as FCFS.

→ if time slice is very small (1 ms) the RRA called as "*Process sharing*". And TAT is depends on Time slice.

- Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are *n* processes in the ready queue and the time quantum is *q*, then each process gets 1/*n* of the CPU time in chunks of at most *q* time units at once. No process waits more than (*n*-1)*q* time units.
- Performance
  - *q* large ⇒ FIFO
  - *q* small ⇒ *q* must be large with respect to context switch, otherwise overhead is too high

Process	Burst Time
P1	24
P2	3
P3	3



The Gantt chart is:

Typically, higher average turnaround than SJF, but better response

Time-Slicing

Since the JVM Doesn’t Ensure Time-Slicing, the yield() Method

May Be Used:

```
while (true) {
    // perform CPU-intensive task
    . . .
    Thread.yield();
}
```

Evaluation

- ***Best CPU Utilization:***

  → Maximized: Process utilization, throughput.

  → Minimized: Turn around time, waiting time, and response time.

  → Arrival: The request from user is arrive to the system.

  → Completion: The processing of the request gets completed then the scheduler will select another process for scheduling.

- ***Advantages of Scheduling:***

1. Minimize the wastage of resources.
2. Effective utilization of CPU.
3. Provide service to their maximum number of used at a given time.
4. The resources are used in a balances manner.

Deterministic modeling

- Example:
- Calculate Average turn around time, average waiting time, throughput of 20 ms time unit on FCFS,SJF, Priority and RRA scheduling Algorithms.
- 1.

Process	P1, P2, P3, P4
CPU burst time	5, 10, 7, 6
Priority	2, 3, 4, 1

2.

Process	:	P1, P2, P3, P4, and P5
CPU Burst time	:	10, 29, 3, 7, and 12
Priority	:	4, 1, 3, 5, and 2