OPERATING SYSTEM

Lecture 2

OBJECTIVES

- Operating System Process Scheduling
- Operating System Scheduling Algorithms
- Operating System Multi - Threading
- Operating System Memory Management
- Operating System Virtual Memory

Process Scheduling

- It is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process.
- It is an essential part of a Multiprogramming operating system.
- It allows more than one process to be loaded into the executable memory at a time and then uses time multiplexing to share the CPU among loaded processes

Process Scheduling Queues

- The OS maintains PCBs in process Scheduling Queues
- The OS maintains the following important process scheduling queues –

– Job queue - This queue keeps all the processes in the system.

– Ready queue - This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.

– Device queues - The processes which are blocked due to unavailability of an I/O device constitute this queue.

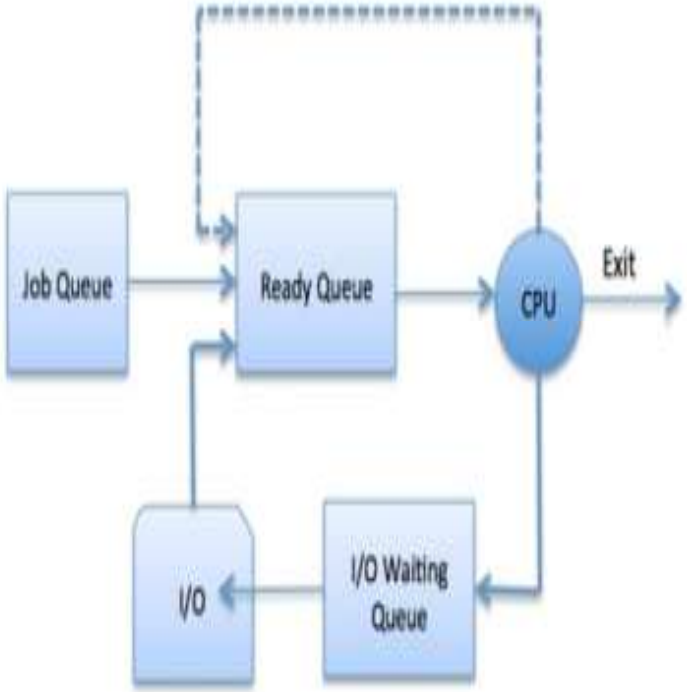


Figure 1 – process scheduling queues

Two-State Process Model

- Two-state process model refers to running and non-running states.

  Running

- When a new process is created, it enters into the system as in the running state.

Two-State Process Model

Not Running

- Processes that are not running are kept in queue, waiting for their turn to execute.
- Each entry in the queue is a pointer to a particular process.
- Queue is implemented by using linked list.
- Use of dispatcher is as follows:

Two-State Process Model

Not Running

- When a process is 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 selects a process from the queue to execute.

Schedulers

- 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 –

– Long-Term Scheduler - It is also called a job scheduler. A long-term scheduler determines which programs are admitted to the system for processing.

– 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.

– Medium-Term Scheduler - Medium-term scheduling is a part of swapping. It removes the processes from the memory. It reduces the degree of multiprogramming.

Comparison among Scheduler

Long-Term Scheduler	**Short-Term Scheduler**	**Medium-Term Scheduler**
It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler
Speed is lesser than short term schedule	Speed is fastest among other two	Speed is in between both short and long term scheduler.
It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming
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

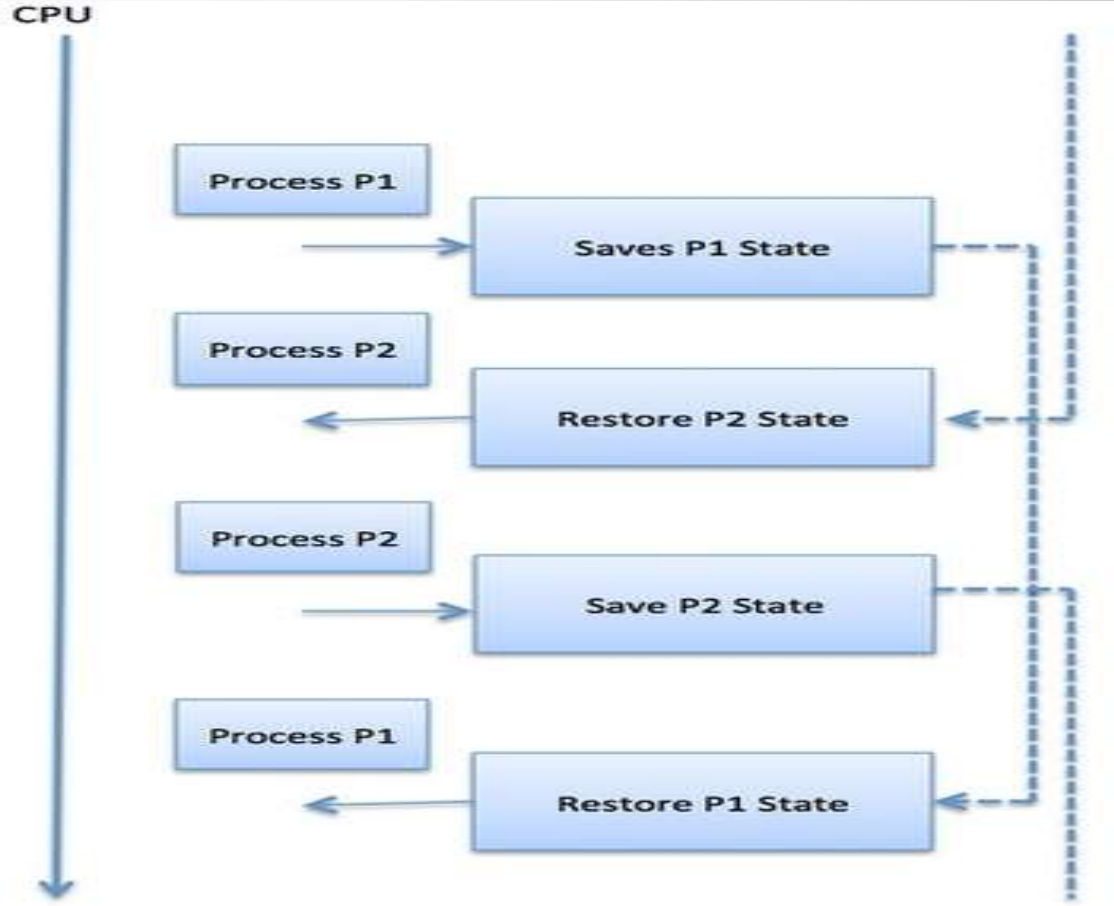
Context Switch

- A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time.
- Using this technique, a context switcher enables multiple processes to share a single CPU.
- Context switching is an essential part of a multitasking operating system features.

Context Switch

- When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block.
- After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc.
- At that point, the second process can start executing.

Context Switch



Context Switch

- Context switches are computationally intensive since register and memory state must be saved and restored.
- To avoid the amount of context switching time, some hardware systems employ two or more sets of processor registers.
- When the process is switched, the following information is stored for later use.

Context Switch

- Program Counter
- Scheduling information
- Base and limit register value
- Currently used register
- Changed State
- I/O State information
- Accounting information

Scheduling Algorithms

- A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms . There are six popular process scheduling algorithms.

– First-Come, First-Served (FCFS) Scheduling
– Shortest-Job-Next (SJN) Scheduling
– Priority Scheduling
– Shortest Remaining Time
– Round Robin(RR) Scheduling
– Multiple-Level Queues Scheduling

Non-preemptive or Preemptive.

- These algorithms are either non-preemptive or preemptive.
- Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state

First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high
- Easy to understand and implement

First Come First Serve (FCFS)

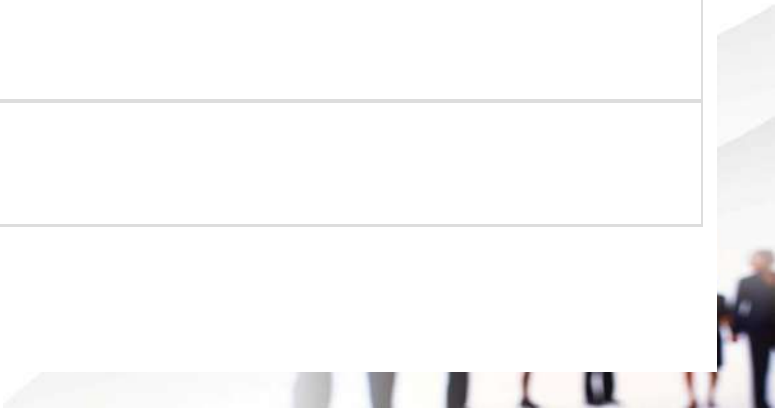
Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16

First Come First Serve (FCFS)

Wait time of each process is as follows −

Process	Wait Time : Service Time - Arrival Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	8 - 2 = 6
P3	16 - 3 = 13

Average Wait Time: (0+4+6+13) / 4 = 5.75



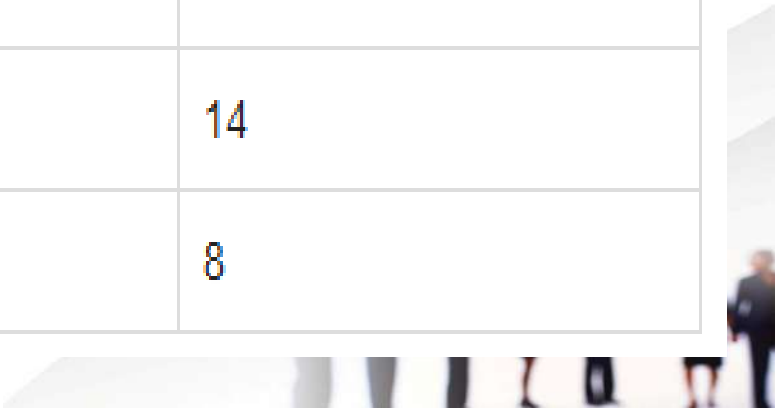
Shortest Job Next (SJN)

- This is also known as shortest job first, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.

Shortest Job Next (SJN)

Given: Table of processes, and their Arrival time, Execution time

Process	Arrival Time	Execution Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	14
P3	3	6	8



Shortest Job Next (SJN)

Waiting time of each process is as follows –

Process	Waiting Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	14 - 2 = 12
P3	8 - 3 = 5

Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

Priority Based Scheduling

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Priority Based Scheduling

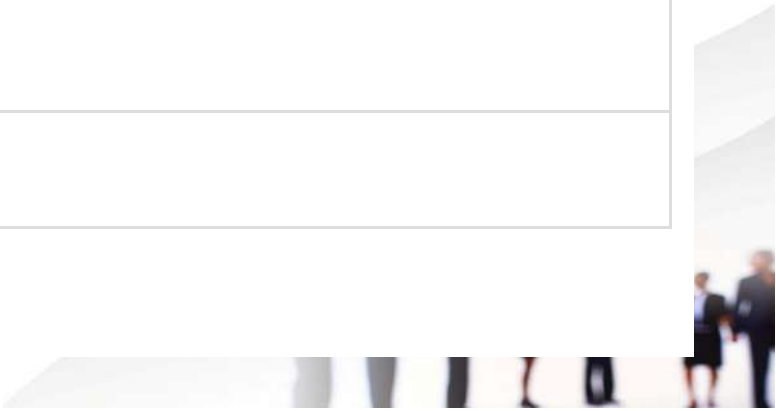
Process	Arrival Time	Execution Time	Priority	Service Time
P0	0	5	1	0
P1	1	3	2	11
P2	2	8	1	14
P3	3	6	3	5

Priority Based Scheduling

Waiting time of each process is as follows –

Process	Waiting Time
P0	0 - 0 = 0
P1	11 - 1 = 10
P2	14 - 2 = 12
P3	5 - 3 = 2

Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6



Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

Round Robin Scheduling

- Round Robin is the preemptive process scheduling algorithm
- Each process is provided a fixed time to execute and is called a quantum.
- It is easy to implement because it does not depend upon burst time
- It deals with all process without any priority.

Multiple-Level Queues Scheduling

- Multiple-level queues are hybrid. They make use of other existing algorithms
- Multiple queues are maintained for processes with common characteristics.
- Each queue can have its own scheduling algorithms.
- Priorities are assigned to each queue

Multi - Threading

- A thread (lightweight process) is a flow of execution through the process code.
- It keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history
- Threads provide a way to improve application performance through parallelism.
- Each thread belongs to one process and represents a separate flow of control.

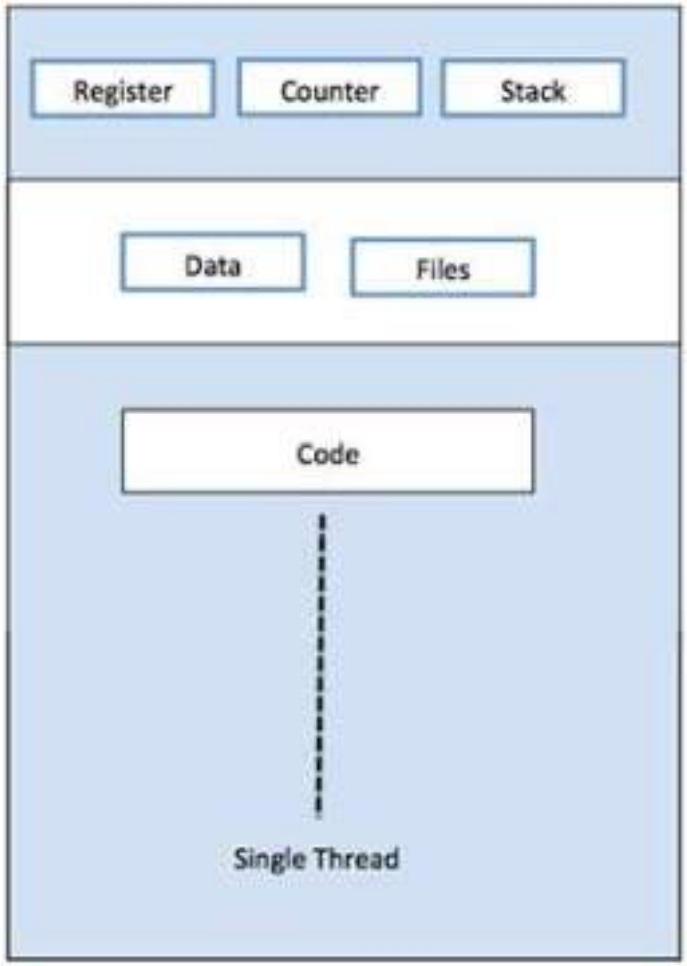


Figure 2 - Single Process with single thread

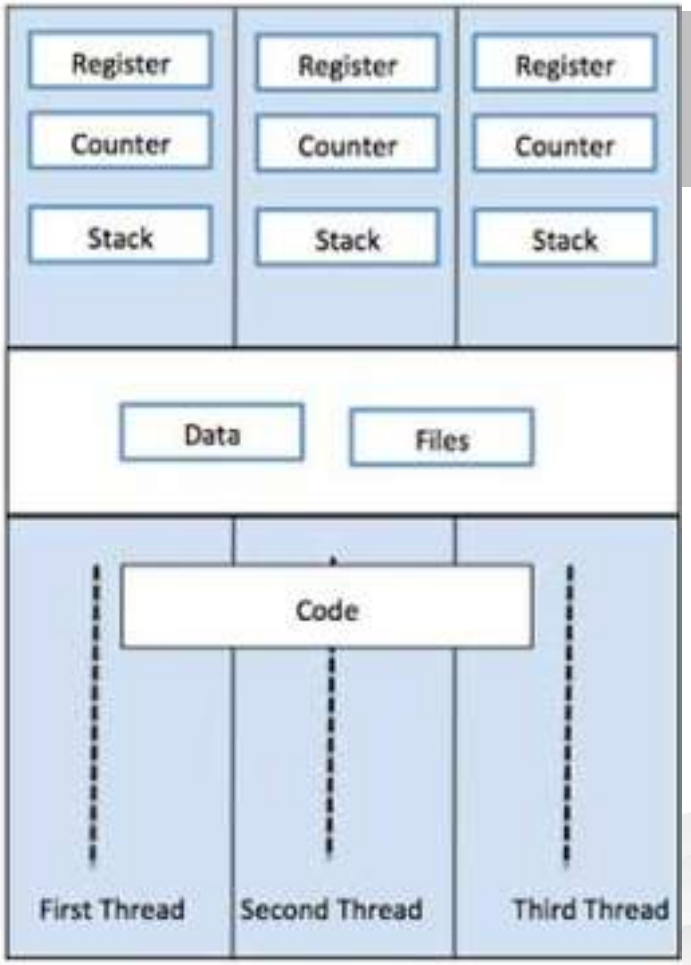


Figure 3 - Single Process with three thread

Difference between Process and Thread

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Merits of Thread

- Use of threads provides concurrency within a process.
- Efficient communication.
- It is more economical to create and context switch threads.
- Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

Multithreading Models

- There are three types of Multithreading models namely:
  - **Many to many relationship -** The many-to-many model multiplexes any number of user threads onto an equal or smaller number of kernel threads
  - **Many to one relationship -** Many-to-one model maps many user level threads to one Kernel-level thread. Thread management is done in user space by the thread library.
  - **One to one relationship -** There is one-to-one relationship of user-level thread to the kernel-level thread. This model provides more concurrency than the many-to-one model. It also allows another thread to run when a thread makes a blocking system call. It supports multiple threads to execute in parallel on microprocessors.

Memory Management

- Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution.
- keeps track of each and every memory location, regardless of either it is allocated to some process or it is free.
- It checks how much memory is to be allocated to processes. It decides which process will get memory at what time.
- It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status

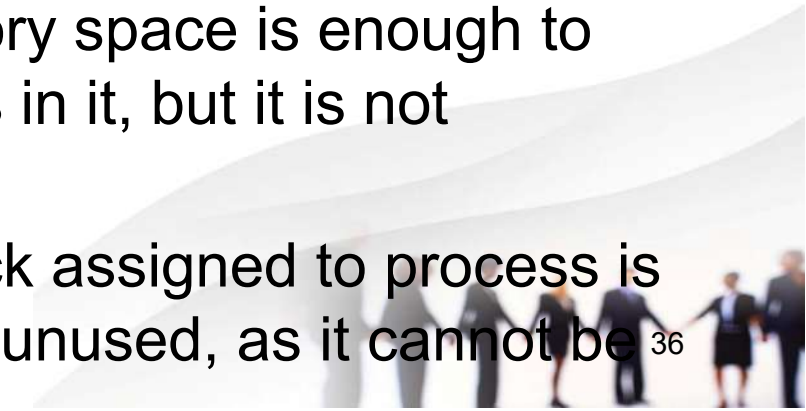
Memory Allocation

- Main memory usually has two partitions –
  - **Low Memory** - Operating system resides in this memory.
  - **High Memory** - User processes are held in high memory.

- The OS uses the following memory allocation mechanism
  - Single-partition allocation – This type of allocation protect user processes from each other .
  - Multiple-partition allocation – This type of allocation divides the main memory in a number if fixed-sized partitions where each partition should contain only one process.

Fragmentation

- It refers to the condition of a disk in which files are divided into pieces scattered around the disk.
- It occurs naturally when you use a disk frequently, creating, deleting, and modifying files.
- It is of two types –
  - **External Fragmentation** - Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used.
  - **Internal fragmentation** - Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process



Paging and Segmentation

- **Paging** is a memory management technique in which process address space is broken into blocks of the same size called **pages.**
- **Segmentation** is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions.
  - Each segment is actually a different logical address space of the program.

Virtual Memory

- It is basically addressing more memory than the physical memory to a system.
- Virtual memory serves two purposes.
  - It allows us to extend the use of physical memory by using disk.
  - It allows us to have memory protection, because each virtual address is translated to a physical address.