**Mid term paper**



Submitted by:  **Muhammad Ali**

Registration No:- **19PWCSE1801**

Class Section: A

Submitted to:

**Sir Tariq Kamal**

Data:(24,05,2021)

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

* First Come First Serve (FCFS), 2) Shortest-Job-First (SJF) Scheduling 3) Shortest Remaining Time 4) Priority Scheduling 5) Round Robin Scheduling 6) Multilevel Queue Scheduling

there are six different scheduling algorithms.

**1.First Come First Serve (FCFS): -**

First Come First Serve is the full form of FCFS. It is the easiest and most simple CPU scheduling algorithm. In this type of algorithm, the process which requests the CPU gets the CPU allocation first. This scheduling method can be managed with a FIFO queue.

As the process enters the ready queue, its PCB (Process Control Block) is linked with the tail of the queue. So, when CPU becomes free, it should be assigned to the process at the beginning of the queue.

.It offers non-preemptive and pre-emptive scheduling algorithm.

.Jobs are always executed on a first-come, first-serve basis

**2) Shortest-Job-First (SJF) Scheduling : -**

SJF is a full form of (Shortest job first) is a scheduling algorithm in which the process with the shortest execution time should be selected for execution next. This scheduling method can be preemptive or non-preemptive. It significantly reduces the average waiting time for other processes awaiting execution.

**3) Shortest Remaining Time : -**

The full form of SRT is Shortest remaining time. It is also known as SJF preemptive scheduling. In this method, the process will be allocated to the task, which is closest to its completion. This method prevents a newer ready state process from holding the completion of an older process.

4)Priority Scheduling : -

Priority scheduling is a method of scheduling processes based on priority. In this method, the scheduler selects the tasks to work as per the priority.

Priority scheduling also helps OS to involve priority assignments. The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis. Priority can be decided based on memory requirements, time requirements, etc.

**5) Round Robin Scheduling : -**

Round robin is the oldest, simplest scheduling algorithm. The name of this algorithm comes from the round-robin principle, where each person gets an equal share of something in turn. It is mostly used for scheduling algorithms in multitasking. This algorithm method helps for starvation free execution of processes.

**6) Multilevel Queue Scheduling: -**

This algorithm separates the ready queue into various separate queues. In this method, processes are assigned to a queue based on a specific property of the process, like the process priority, size of the memory, etc.

However, this is not an independent scheduling OS algorithm as it needs to use other types of algorithms in order to schedule the jobs.

Multiple queues should be maintained for processes with some characteristics.

. Every queue may have its separate scheduling algorithms.

. Priorities are given for each queue.

**Code: -**

#include <stdio.h> /\* standard I/O routines \*/

#include <pthread.h> /\* pthread functions and data structures \*/

/\* function to be executed by the new thread \*/

void\* PrintHello(void\* data)

{

int my\_data = (int)data; /\* data received by thread \*/

[pthread\_detach](http://pubs.opengroup.org/onlinepubs/009695399/functions/pthread_detach.html)([pthread\_self](http://pubs.opengroup.org/onlinepubs/009695399/functions/pthread_self.html)());

printf("Hello from new thread - got %d\n", my\_data);

[pthread\_exit](http://pubs.opengroup.org/onlinepubs/007908775/xsh/pthread_exit.html)(NULL); /\* terminate the thread \*/

}

/\* like any C program, program's execution begins in main \*/

int main(int argc, char\* argv[])

{

int rc; /\* return value \*/

pthread\_t thread\_id; /\* thread's ID (just an integer) \*/

int t = 10; /\* data passed to the new thread \*/

/\* create a new thread that will execute 'PrintHello' \*/

rc = [pthread\_create](http://pubs.opengroup.org/onlinepubs/007908775/xsh/pthread_create.html)(&thread\_id, NULL, PrintHello, (void\*)t);

if(rc) /\* could not create thread \*/

{

printf("\n ERROR: return code from pthread\_create is %d \n", rc);

exit(1);

}

printf("\n Created new thread (%u) ... \n", thread\_id);

[pthread\_exit](http://pubs.opengroup.org/onlinepubs/007908775/xsh/pthread_exit.html)(NULL); /\* terminate the thread \*/

}

**Process state machine: -**

Each process being managed by an OS is represented by a dedicated data structure referred to as a “process control block” (PCB). Like the process state machine, the PCB will vary from OS to OS.

However, there are a minimum number of basic parameters common to the PCBs of all OSs. The PCB contains the data that specifies the existence of a particular process and the information necessary to permit the process to make forward progress.

From the previous subsection, it is clear that the process state is a critical parameter determining the modality of a process at any point in time and thus the possible states to which it may transit. The PCB contains a field that specifies an encoding of all possible process states, and holds the value associated with the state of the process as the process evolves throughout its lifecycle.