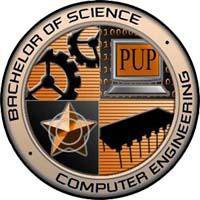
****Republic of the Philippines

POLYTECHNIC UNIVERSITY OF THE PHILIPPINES

College of Engineering

COMPUTER ENGINEERING DEPARTMENT

CPU SCHEDULING

In Partial Fulfillment

of the Requirements for the Subject

Operating Systems

by

Asuncion, Harold T.

Eresuela, Michael Joshua G.

Obmerga, Gladys T.

Pal-lingayan, Pinky S.

March 2015

***PROJECT DESCRIPTION***

An operating system provides services to both users and to programs. It provides programs, an environment to execute while providing the users, services to execute the programs in a convenient manner. Program in execution is often called process. Process is an entity which represents the basic unit of work to be implemented in the system. To maximize the system, we must keep the CPU as busy as possible, often called CPU Utilization.

Part of CPU Utilization is CPU scheduling or 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 on the basis of a particular strategy. Process scheduling is an essential part of a Multiprogramming operating system for it allows more than one process to be loaded into the executable memory at a time and loaded process shares the CPU using time multiplexing.

This project is designed primarily to create a CPU scheduling GUI application. This will improve our understanding of a typical process scheduling algorithm and how it works. It also incorporates skills and knowledge that we have earned in previous class discussions pertaining to the said topic.

In this project, the behavior of the different scheduling algorithm was taken into consideration:

**First Come First Serve (FCFS)**

This non-preemptive scheduling algorithm follows the first-in, first-out (FIFO) policy. As each process becomes ready, it joins the ready queue. When the current running process finishes execution, the oldest process in the ready queue is selected to run next.

**Shortest Job First (SJF)**

This non-preemptive scheduling algorithm favors processes with the shortest expected process time. As each process becomes ready, it joins the ready queue. When the current running process finishes execution, the process in the ready queue with the shortest expected processing time (or service time) is selected to run next.

**Shortest Remaining Time (SRT)**

This preemptive scheduling algorithm favors processes with the shortest remaining expected process time. As each process becomes ready, it joins the ready queue. This triggers an interrupt which preempts the current running process back into the ready queue. The process in the ready queue with the shortest remaining service time is selected to run next.

**Round Robin (RR)**

The CPU is allocated to the process at the head the ready queue. Then, one of two things will happen. If the process has a CPU burst of less than 1 time quantum, the process itself will release the CPU voluntarily. The CPU is then assigned to the next process in the ready queue. Otherwise, if the CPU burst of the currently running job is longer than 1 time quantum, the process is preempted after 1 time quantum and put at the tail of the ready queue. The CPU is then assigned to the next process in the ready queue.

**Priority (Prio)**

A priority is associated with each processes and the CPU is allocated to the process with the highest priority.

***THEORITICAL FRAMEWORK***

Modern general purpose computers, including personal computers and mainframes, have an operating system to run other programs, such as application software. An operating system (OS) is a software program that manages the hardware and software resources of a computer. The OS performs basic tasks, such as controlling and allocating memory, prioritizing the processing of instructions, controlling input and output devices, facilitating networking, and managing files.

To maximize CPU usage, OS have CPU scheduler which makes a sequence of “moves” that determines the interleaving of threads. The scheduler’s moves are dedicated by a scheduling policy.

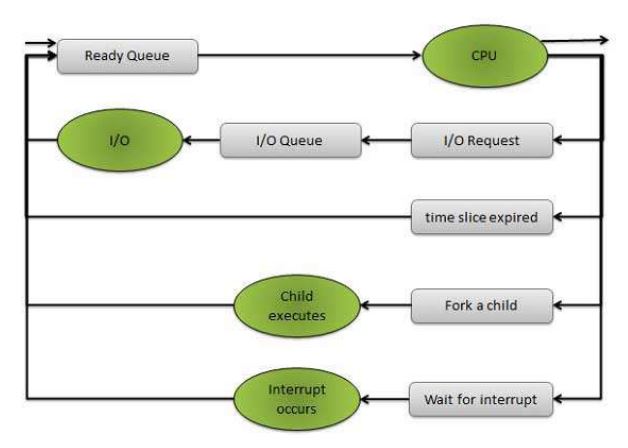


The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

Scheduling queues refers to queues of processes or devices. When the process enters into the system, then this process is put into a job queue. This queue consists of all processes in the system. The operating system also maintains other queues such as device queue. Device queue is a queue for which multiple processes are waiting for a particular I/O device. Each device has its own device queue.

This figure shows the queuing diagram of process scheduling.

* Queue is represented by rectangular box.
* The circles represent the resources that serve the queues.
* The arrows indicate the process flow in the system.



Queues are of two types

* Ready queue
* Device queue

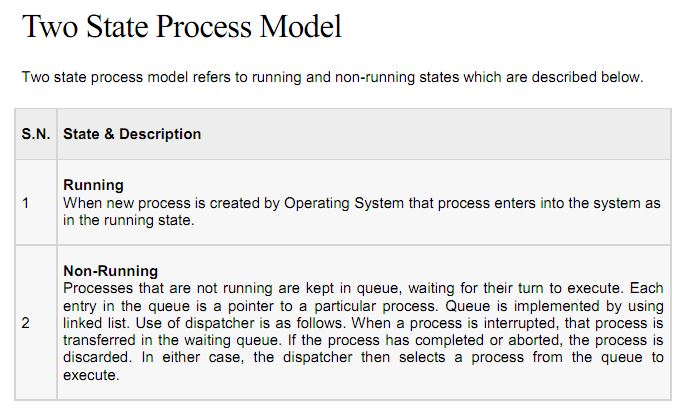
A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.

* The process could issue an I/O request and then it would be placed in an I/O queue.
* The process could create new sub process and will wait 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

Two state process model refers to the running and non-running states which are described below.



***SYSTEM WALKTHROUGH***

This project is designed primarily to create a CPU scheduling GUI application.

1. The program is not a system program, it is just a typical user application.
2. The program simulator was programmed in Java language.
3. Java Swing was used for the Graphical User Interface to show the following components:
4. The description of the simulated process includes the following information:
   * Number of processes
   * Arrival time of the process
   * Number of CPU burst
5. The application was capable of receiving input values from the user for the number of processes, arrival time, burst time and priority.
6. Import data was also implemented.
7. At the end of simulation, the simulator shall display the Gantt Chart

***SIMULATION OUTPUT***

