

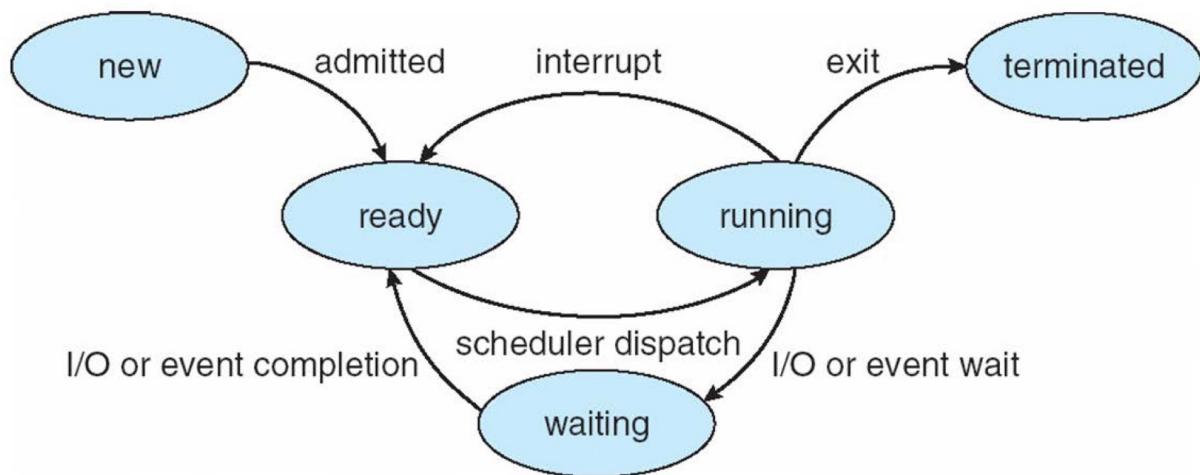
## 1. General outline of the assigned work

CPU Scheduling is a process of determining which process will own CPU for execution while another process is on hold. This makes sure that whenever the CPU is idle, the OS will select one of the processes available in the ready queue for execution. The CPU scheduler selects one of the processes in memory that are ready for execution. In this part of the project we focus on first come first served (FCFS) scheduling algorithm. The helper functions provide the algorithms for reading the process from the file and stores results into a process structure array and sorts the processes array ascending by arrival time. Once program has all the the information from the stdio, the scheduling begins for four CPUs.

FCFS executes queued requests and processes them in order of their arrival, the process that requests for the CPU gets CPU allocation first. A new incoming process is first added to a temporary queue which is then sorted into another queue based on its burst status. The elements of the temporary queue are sorted to add them (enqueue them) to the ready queue in the proper order. Ready queue consists of all processes that are ready to run. The temporary queue is then reset to size to 0. Then program selects a CPU that doesn't have a process currently running on it and schedules the next process on that CPU.

If a currently running process has finished its burst time, we move it to the waiting queue and terminate those who have finished their CPU burst. Waiting queue consists of the processes whose arrival time has been crossed but have yet to be executed. Then we start the process' next I/O burst. If CPU burst is not finished, we move the process to the waiting queue and free the current CPU. If the CPU burst is finished, we terminate the process by setting the end time to the current time. Then we get the first process in the waiting queue and check if its I/O burst is complete. If the current I/O burst is complete, we move on to the next I/O burst and add process to the temporary queue. Then we dequeue the waiting queue and update waiting state by incrementing the current burst's step. We do these steps in a cycle.

For a better visualisation, please refer to the image below from the textbook; Operating System Concepts by Abraham Silberschatz, Peter Galvin and Greg Gagne.



While we go through all the processes, we keep track of their wait times, turnaround times, end time, etc. From these values we can then calculate the average wait time and turn around time. We also keep the clock time moving and it stops when execution of all processes is complete. This gives us the total time CPUs were running and gives us a way to calculate CPU utilisation. For each loop we keep track of how many CPUs were currently running and that gives us a way to calculate CPU utilisation. Since this is a non-preemptive scheduling, there is no context switching. To get the PID of the last process to finish, we compare the PID's of all processes and pick the one that has the latest (largest) end time. To understand the exact calculation method, please refer to the code in the file fcfs.c.

In conclusion, the assigned work is for us to understand how CPU scheduling works, for this part, how FCFS scheduling works and how to schedule processes using queues.

## 2. Assigned work/components done successfully

Program calculates the following and outputs it to stdio :-

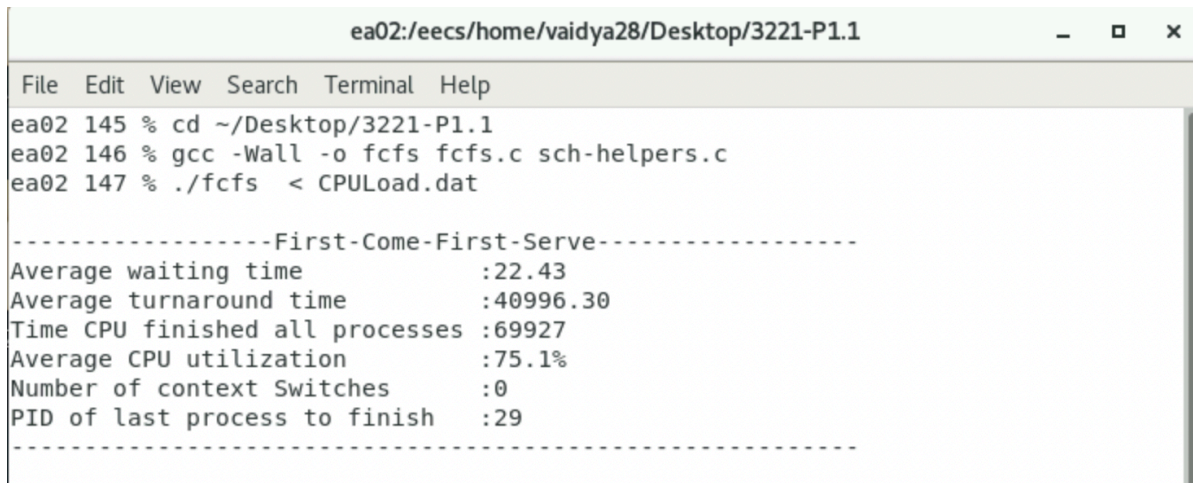
- Average waiting time
- Average turnaround time
- Average CPU utilisation
- Time CPU finished all processes
- Number of context Switches
- PID of last process to finish

## 3. Assigned work/components not done successfully

None, I have completed all the assigned components for this part of the project which involves FCFS basis of scheduling processes using four CPUs.

#### 4. Any other comments

Following is an attachment of the output I got when I ran program on EECS server:-



```
ea02:/eecs/home/vaidya28/Desktop/3221-P1.1
File Edit View Search Terminal Help
ea02 145 % cd ~/Desktop/3221-P1.1
ea02 146 % gcc -Wall -o fcfs fcfs.c sch-helpers.c
ea02 147 % ./fcfs < CPUload.dat

-----First-Come-First-Serve-----
Average waiting time      :22.43
Average turnaround time   :40996.30
Time CPU finished all processes :69927
Average CPU utilization    :75.1%
Number of context Switches :0
PID of last process to finish :29
-----
```