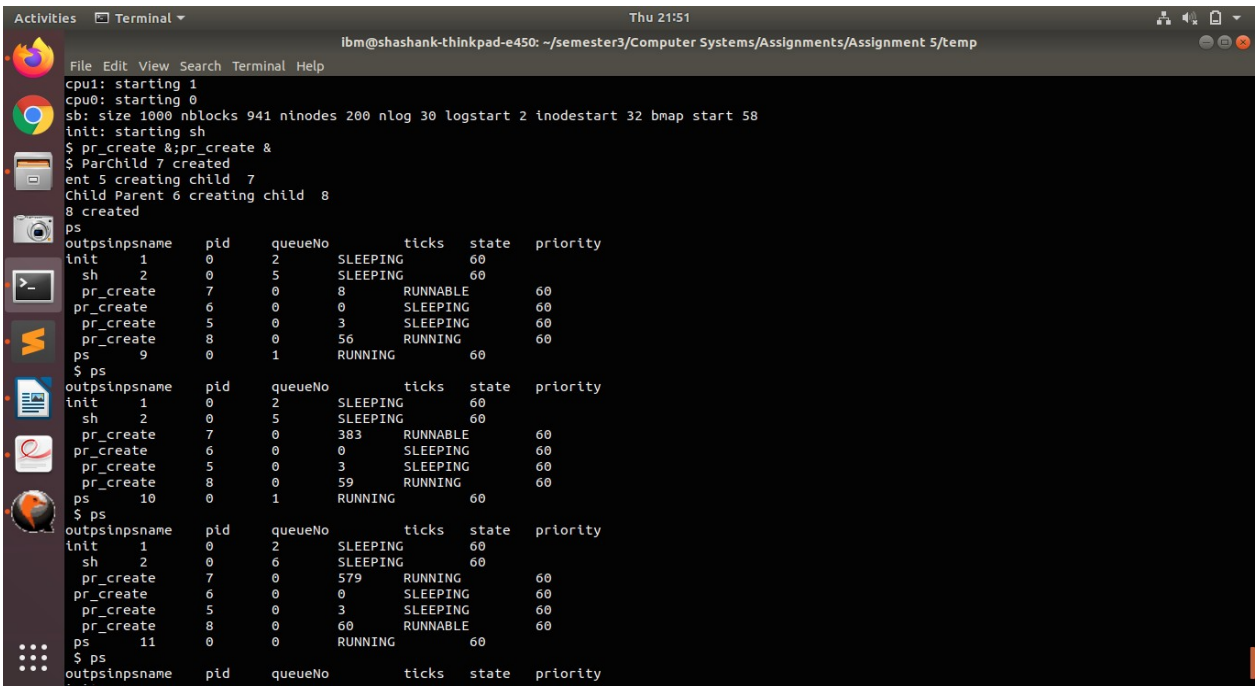
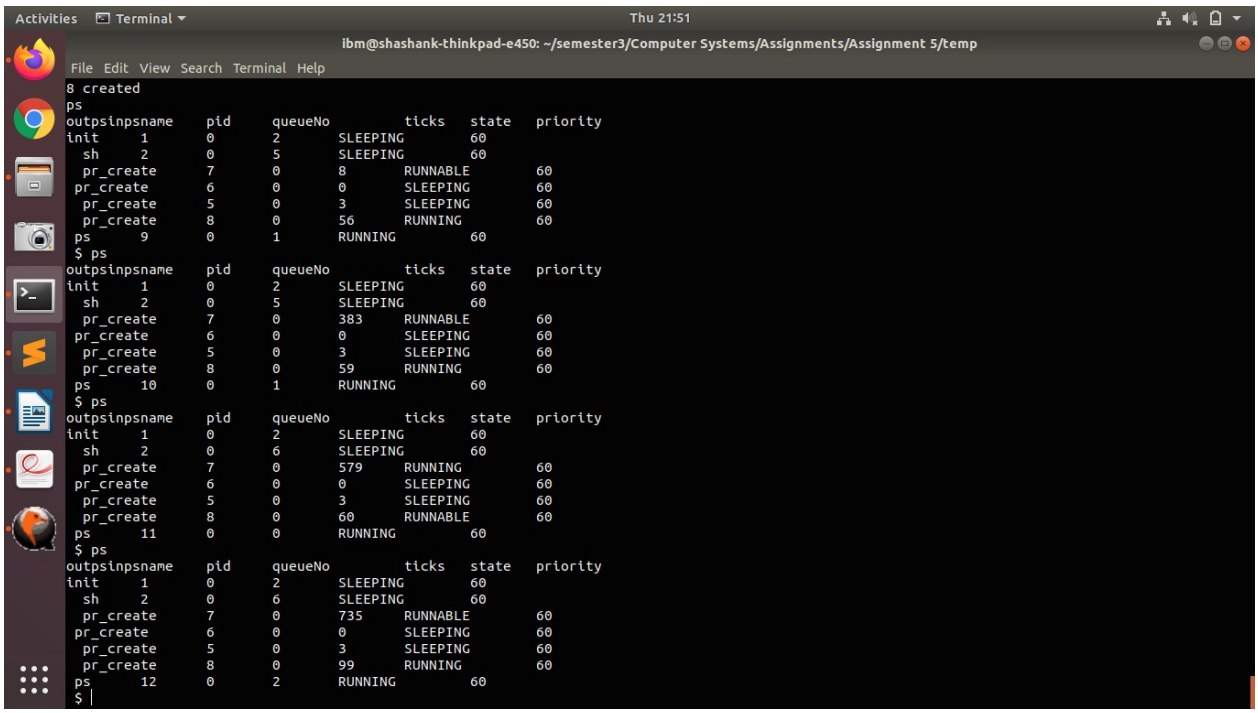
# ASSIGNMENT 5
# 2018111016
# ENHANCING XV6 OS

## Performance Comparision of SCHEDULING POLICIES

### DEFAULT

The default scheduling is Round robin, where each process is provided a fixed time to execute called “QUANTUM”.Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
The screenshots of processes info when DEFAULT is used:



In above images there are two processes with pids 7 and 8 which are RUNNING alternatively ,the “RUNNING” process gets changed once its assigned timequantum is done.

FIRST COME – FIRST SERVED (FCFS)

Processes are executed on first come, first serve basis.Other processes need to wait until the processes which are created before it are done.
The screenshots of processes info when FCFS is used:

```
368+1 records in
368+1 records out
188672 bytes (189 kB, 184 KiB) copied, 0.00110625 s, 171 MB/s
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -
m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ pr_create &
$ Parent 4 creating child  5
Child 5 created
pr_create &
$ Parent 7 creating child  8
Child 8 created
ps
ps
ps
```

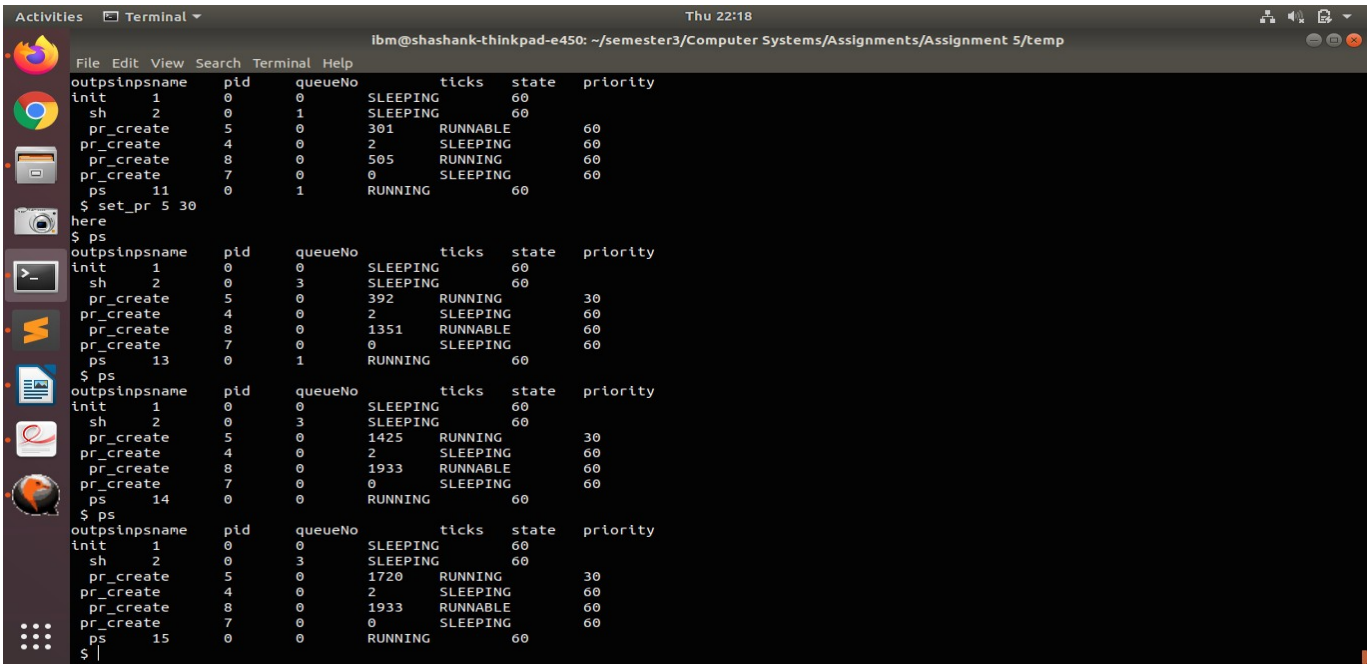
Here two processes are created with pids 5 and 8,and later no other process is getting executed as the previous started processes are not finished and the # of CPU’s used are 2 so atmost 2 processes can execute at a given time.

PRIORITY BASED SCHEDULER (PBS)

A priority-based scheduler selects the process with the highest priority for execution.Until the processes of higher priority are done lower priority are not done.
The screenshots of processes info when PBS is used:

```
Activities  Terminal  Thu 22:18
ibm@shashank-thinkpad-e450: ~/semester3/Computer Systems/Assignments/Assignment 5/temp
File  Edit  View  Search  Terminal  Help
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ pr_create &
$ Parent 4Child 5 created
 creating child  5
pr_create &
$ ParenCt 7 creating child  8
hild 8 created
ps
outpsinpsname    pid     queueNo         ticks   state   priority
init     1       0       0       SLEEPING        60
  sh     2       0       1       SLEEPING        60
  pr_create      5       0       299     RUNNING         60
 pr_create       4       0       2       SLEEPING        60
  pr_create      8       0       106     RUNNABLE        60
 pr_create       7       0       0       SLEEPING        60
  ps     9       0       0       RUNNING         60
 $ ps
outpsinpsname    pid     queueNo         ticks   state   priority
init     1       0       0       SLEEPING        60
  sh     2       0       1       SLEEPING        60
  pr_create      5       0       300     RUNNING         60
 pr_create       4       0       2       SLEEPING        60
  pr_create      8       0       254     RUNNABLE        60
 pr_create       7       0       0       SLEEPING        60
  ps     10      0       1       RUNNING         60
 $ ps
outpsinpsname    pid     queueNo         ticks   state   priority
init     1       0       0       SLEEPING        60
  sh     2       0       1       SLEEPING        60
  pr_create      5       0       301     RUNNABLE        60
 pr_create       4       0       2       SLEEPING        60
  pr_create      8       0       505     RUNNING         60
 pr_create       7       0       0       SLEEPING        60
  ps     11      0       1       RUNNING         60
```

Above two processes with pids 5 and 8 are created and they both run alternatively following Round robin as their priorities are same (i.e 60)
Now set_pr command is used to update the priority of process with pid 5 to 30.




Now only process of pid 5 is RUNNING and pid 8 process is not RUNNING as one CPU is assigned to pid 5 and other is for present command ,so pid 8 remains in RUNNABLE until process of pid 5 is done its execution.

MULTI-LEVEL FEEDBACK QUEUE SCHEDULING

MLFQ scheduler allows processes to move between different priority queues based
on their behavior and CPU bursts. If a process uses too much CPU time, it is pushed
to a lower priority queue, leaving I/O bound and interactive processes for higher priority queues. Also, to prevent starvation, it implements aging.
The screenshots of processes info when MLFQ is used:

Here initially only one process is created with pid 5, and can observe that it changes its “CURRENT QUEUE” based on the ticks performed.

```
Activities  Terminal  Thu 22:43
ibm@shashank-thinkpad-e450: ~/semester3/Computer Systems/Assignments/Assignment 5/temp
File  Edit  View  Search  Terminal  Help
  pr_create      13      4       1576    RUNNING         60
 $ QEMU 2.11.1 monitor - type 'help' for more information
(qemu) q
(base) ibm@shashank-thinkpad-e450:~/semester3/Computer Systems/Assignments/Assignment 5/temp$ make qemu SCHEDULER=MLFQ
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -
m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ pr_create &
$ Parent 4 creating child  5
Child 5 created
ps
outpsinpsname    pid     queueNo         ticks   state   priority
init     1       1       1       SLEEPING        60
  sh     2       0       4       SLEEPING        60
  pr_create      5       2       3       RUNNING         60
 pr_create       4       0       0       SLEEPING        60
  ps     6       0       3       RUNNING         60
 $ ps
outpsinpsname    pid     queueNo         ticks   state   priority
init     1       1       1       SLEEPING        60
  sh     2       0       4       SLEEPING        60
  pr_create      5       2       7       RUNNING         60
 pr_create       4       0       0       SLEEPING        60
  ps     7       0       1       RUNNING         60
 $ ps
outpsinpsname    pid     queueNo         ticks   state   priority
init     1       1       1       SLEEPING        60
  sh     2       0       4       SLEEPING        60
  pr_create      5       3       29      RUNNING         60
 pr_create       4       0       0       SLEEPING        60
  ps     8       0       1       RUNNING         60
 $ pr_create &
Parent 10 creating child  11
$ Child 11 created
```

Now another process with pid 11 is created and is in RUNNING state as it lies in queue 0 and can observe the change of queue of that process based on the ticks performed.

```
Activities  Terminal  Thu 22:44
ibm@shashank-thinkpad-e450: ~/semester3/Computer Systems/Assignments/Assignment 5/temp
File  Edit  View  Search  Terminal  Help
init     1       1       1       SLEEPING        60
  sh     2       0       19      SLEEPING        60
  pr_create      5       4       3475    RUNNABLE        60
 pr_create       4       0       0       SLEEPING        60
  pr_create      11      0       9       RUNNING         60
 pr_create       10      0       0       SLEEPING        60
  ps     26      0       0       RUNNING         60
 $ ps
outpsinpsname    pid     queueNo         ticks   state   priority
init     1       1       1       SLEEPING        60
  sh     2       1       4       SLEEPING        60
  pr_create      5       4       3475    RUNNABLE        60
 pr_create       4       0       0       SLEEPING        60
  pr_create      11      2       21      RUNNING         60
 pr_create       10      0       0       SLEEPING        60
  ps     27      0       1       RUNNING         60
 $ pps
exec: fail
exec pps failed
$ ps
outpsinpsname    pid     queueNo         ticks   state   priority
init     1       1       1       SLEEPING        60
  sh     2       0       25      SLEEPING        60
  pr_create      5       4       4079    RUNNABLE        60
 pr_create       4       0       0       SLEEPING        60
  pr_create      11      2       21      RUNNING         60
 pr_create       10      0       0       SLEEPING        60
  ps     29      0       1       RUNNING         60
 $ ps
outpsinpsname    pid     queueNo         ticks   state   priority
init     1       1       1       SLEEPING        60
  sh     2       0       26      SLEEPING        60
  pr_create      5       4       4384    RUNNABLE        60
 pr_create       4       0       0       SLEEPING        60
  pr_create      11      2       21      RUNNING         60
 pr_create       10      0       0       SLEEPING        60
  ps     30      1       0       RUNNING         60
 $
```

CONCLUSION:

In FCFS no preemption is allowed so the process created should WAIT until all the processes created before it are executed.Not suitable for REAL TIME COMPUTING and NOT INTERACTIVE.
But it is easy to implement...
Avg waiting time of processes can be large if the first created process runs for a long time.

In PBS higher priority processes are executed so the processes which need to meet a deadline can be given a higher priority and can be done.
But a process with low priority keeps WAITING until all other higher priority processes are done which leads to STARVATION.....

In RR, STARVATION doesn't occur because for each round robin cycle, every process is given a fixed time to execute. No process is left behind.
But if time quantum is longer than needed, it tends to exhibit the same behavior as FCFS and if time quantum is shorter than needed, the number of times that CPU switches from one process to another process, increases. This leads to decrease in CPU efficiency as number of CONTEXT SWITCHES are increased.

In MLFQ , processes are RESPONSIVE but STARVATION of processes cant be avoided as processes in lower priority queues need to wait indefinitely until higher priority queues are done.
Additional overhead due to MULTIPLE QUEUES as process needs to be switched between the queues.