Name: Pratyanshu Pandey
Roll No: 2019101025

Explanation of Implementation:

The process handling is implemented in form of an array of structures. Each structure contains relevant data for a process. Similarily we have a structure for CPU.

Process Structure

```
// Per-process state
struct proc
{
uint sz; // Size of process memory (bytes)
pde_t *pgdir; // Page table
char *kstack; // Bottom of kernel stack for this process
enum procstate state; // Process state
int pid; // Process ID
struct proc *parent; // Parent process
struct trapframe *tf; // Trap frame for current syscall
struct context *context; // swtch() here to run process
void *chan; // If non-zero, sleeping on chan
int killed; // If non-zero, have been killed
struct file *ofile[NOFILE]; // Open files
struct inode *cwd; // Current directory
char name[16]; // Process name (debugging)
int ctime; // Creation time of the process
int etime; // End time of the process
int rtime; // Run time of the process
int iotime; // time spent in I/O operation
int cur_wait_time; // time spent in CPU ready queue since last execution/IO operation
int total_wait_time; // Total Waiting time spent in CPU ready queue
int priority; // priority in [0,100]. Lower value = Higher priority. Default = 60
int n_run; // Number of times the process was picked by the scheduler
int cur_q; // Current queue for MLFQ
int q[5]; //  Number of ticks the process has received at each of the 5 queues
int cur_rtime_ticks; // number of runtime clockticks in current time slice
};
```

Implemented Functions:

```
void proc_time_update()
```

After every tick we update the relevant variables in the proc structure in the function. This function is called from trap.c.

```
// print info of all the running processes
int ps()
```

Prints all the relevant data for all the processes. We acquire the ptable spinlock and go through all the processes in it and print all the relevant variables from the proc structure.

```
// set a new priority for PBS scheduler
int set_priority(int new_priority, int pid)
```

Just sets the priority of a process. We acquire the ptable spinlock and go through all the processes in it till we find the required process and then we change its priority to new_priority and return old priority.

Scheduling Algorithms

Note: Before every algorithm looks through the process table the ptable spinlock is acquired and released after we are done. Once the scheduler has decided upon a process the following commands are run irrespective of the scheduling algorithm where p is the chosen process. Also a scheduler is called when a process goes for I/O, exits or by the timer interrupt.

```
// Switch to chosen process. It is the process's job
// to release ptable.lock and then reacquire it
// before jumping back to us.

c->proc = p;
switchuvm(p);
p->state = RUNNING;
p->n_run++;
p->cur_wait_time = 0;

swtch(&(c->scheduler), p->context);

switchkvm();
```

1.Default(Similar to Round Robin)

The default algorithm is not exactly Round Robin as there is no implementation of a circular queue. But nevertheless trap.c contains a function with an if condition that yields the CPU to the scheduler after certain constant time (1 tick). During yielding the state of the current running process is changed to “RUNNABLE”. The scheduler then acquires the ptable spinlock, looks through the process table and just allocates the CPU to the first process that it finds “RUNNABLE”.

2. First Come First Serve (FCFS)

In this case first we modify the if condition in trap.c to not yield the CPU to scheduler periodically. The if condition on satisfaction performs no operation now. Now when the scheduler is called we look through all the “RUNNABLE” processes and pick the one with the highest cur_wait_time value. Higher cur_wait_time value means the process was created first or arrived in queue first and demanded first and thus should be alloted the CPU.

3. Priority Based Scheduling(PBS)

In this case after every tick we check if there is a “RUNNABLE” process with lower or equal priority than the current process. In any case we yield the CPU to scheduler otherwise we continue running the process. The scheduler on being called first identifies the lowest “RUNNABLE” priority. Then for all the processes with that priority it choses the one with the highest cur_wait_time. This ensures that same priority processes are implemented in a Round Robin fashion. So after expiration of time slice if there are more than one “RUNNABLE” processes with same priority we choose the next one in queue and if there is only one process with the priority that process is run.

4. Multilevel Feedback Queue

Depending upon which queue the process belongs to we call yield function from trap.c after the alloted time slice is over. This is done using

```
int cur_rtime_ticks; // number of runtime clockticks in current time slice
```

Now when the scheduler is called it

1. Checks if there are any “RUNNABLE” processes that have value of cur_rtime_ticks greater than the alloted time slice. This would mean that the scheduler was called because of this process crossing its limits and we need to set cur_time_ticks = 0 , cur_wait_time = 0and demote the process’s queue if possible.
2. Similarily if a “RUNNABLE” process has cur_wait_time>AGING_TIME the process is promoted to a higher queue and cur_wait_time is set to 0. If the process is already in queue 0 nothing happens.
3. Now we go through all the queues in order of highest to lowest priority. For a certain queue if the queue is not empty we pick a process that has highest cur_wait_time and allot the CPU to it.

This satisfies all the requirements mentioned in the document. Note that if a process forfiets the CPU before expiration of time slice its cur_rtime_ticks becomes 0 and it is not demoted but rather remains in the same queue.

Question from PDF:
If a process voluntarily relinquishes control of the CPU, it leaves the queuing network, and when the process becomes ready again after the I/O, it is inserted at the tail of the same queue, from which it is relinquished earlie .(Explain inthe report how could this be exploited by a process).

Ans:
If a process voluntarily gives up the CPU then it means that it will remain in the same queue and not get demoted to a lower level. Had it been demoted to a lower priority queue it would have to wait for all the processes in the higher queues to finish or wait for aging to take effect. But if it remains in the same queue it will get the CPU back to itself much sooner due to round robin. This improves response time for the process and also allows the process to try to obtain as much CPU as possible.

Performance Comparison:

Benchmark: The benchmark creates 10 processes and assigns some priority to them. 9 out of 10 processes call exec to a programme called bcnt.c which is just a programme that does big calculations i.e. CPU intensive jobs. 1 remaining programme just sleeps simulating an I/O intensive job. The parent programme then prints the results for all the processes.

Case 1 : bcnt.c contains relatively short process

Activities Terminal ▾ Nov 2 21:32 77 %

pratyanshu@pratyanshu-Inspiron-5570: ~/Sem 3/Operating Systems and Networks/Assignments/Assignment 5/2019101025/xv6-public

```
SeaBIOS (version 1.13.0-1ubuntu1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA10+1FECCA10 CA00



xv6...g from Hard Disk...
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
$ benchmark 2
For this benchmark we have calculated sum of runtimes
and waiting times of all the sub processes and
 divided by number of subprocesses to get the average
Process with pid = 4 exited with rtime = 876 and wtime = 679 and total time = 1555
Process with pid = 13 exited with rtime = 0 and wtime = 568 and total time = 568
Process with pid = 5 exited with rtime = 875 and wtime = 1272 and total time = 2147
Process with pid = 6 exited with rtime = 888 and wtime = 1801 and total time = 2689
Process with pid = 7 exited with rtime = 903 and wtime = 2030 and total time = 2933
Process with pid = 8 exited with rtime = 914 and wtime = 2095 and total time = 3009
Process with pid = 9 exited with rtime = 918 and wtime = 2142 and total time = 3060
Process with pid = 10 exited with rtime = 929 and wtime = 2113 and total time = 3042
Process with pid = 11 exited with rtime = 930 and wtime = 2084 and total time = 3014
Process with pid = 12 exited with rtime = 938 and wtime = 2085 and total time = 3023
Average Run time(time running on CPU i.e. RUNNING state) = 817
Average Wait time(time spent in RUNNABLE state) = 1686
Average Total time = 2504
$ For Round Robin (Default)
```

Activities Terminal ▾ Nov 2 21:35 80 %

pratyanshu@pratyanshu-Inspiron-5570: ~/Sem 3/Operating Systems and Networks/Assignments/Assignment 5/2019101025/xv6-public

```
SeaBIOS (version 1.13.0-1ubuntu1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA10+1FECCA10 CA00



Booting from Hard Disk..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
$ benchmark 2
For this benchmark we have calculated sum of runtimes
and waiting times of all the sub processes and
 divided by number of subprocesses to get the average
Process with pid = 13 exited with rtime = 0 and wtime = 0 and total time = 0
Process with pid = 11 exited with rtime = 939 and wtime = 0 and total time = 939
Process with pid = 12 exited with rtime = 942 and wtime = 0 and total time = 942
Process with pid = 10 exited with rtime = 992 and wtime = 740 and total time = 1732
Process with pid = 9 exited with rtime = 945 and wtime = 1142 and total time = 2087
Process with pid = 8 exited with rtime = 943 and wtime = 1732 and total time = 2675
Process with pid = 7 exited with rtime = 942 and wtime = 2088 and total time = 3030
Process with pid = 6 exited with rtime = 947 and wtime = 2676 and total time = 3623
Process with pid = 5 exited with rtime = 931 and wtime = 3031 and total time = 3962
Process with pid = 4 exited with rtime = 933 and wtime = 3624 and total time = 4557
Average Run time(time running on CPU i.e. RUNNING state) = 851
Average Wait time(time spent in RUNNABLE state) = 1503
Average Total time = 2354
$ For PBS(Priority Based Scheduler)
```

```
Activities   Terminal ▾                    Nov 2  21:37                         82 %
pratyanshu@pratyanshu-Inspiron-5570: ~/Sem 3/Operating Systems and Networks/Assignments/Assignment 5/2019101025/xv6-public
SeaBIOS (version 1.13.0-1ubuntu1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA10+1FECCA10 CA00



Booting from Hard Disk..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
$ benchmark 2
For this benchmark we have calculated sum of runtimes
and waiting times of all the sub processes and
 divided by number of subprocesses to get the average
Process with pid = 4 exited with rtime = 949 and wtime = 812 and total time = 1761
Process with pid = 13 exited with rtime = 0 and wtime = 1385 and total time = 1385
Process with pid = 5 exited with rtime = 1050 and wtime = 1325 and total time = 2375
Process with pid = 7 exited with rtime = 951 and wtime = 2530 and total time = 3481
Process with pid = 6 exited with rtime = 953 and wtime = 2847 and total time = 3800
Process with pid = 11 exited with rtime = 944 and wtime = 2708 and total time = 3652
Process with pid = 10 exited with rtime = 948 and wtime = 3003 and total time = 3951
Process with pid = 8 exited with rtime = 950 and wtime = 3071 and total time = 4021
Process with pid = 12 exited with rtime = 949 and wtime = 3015 and total time = 3964
Process with pid = 9 exited with rtime = 976 and wtime = 3207 and total time = 4183
Average Run time(time running on CPU i.e. RUNNING state) = 867
Average Wait time(time spent in RUNNABLE state) = 2390
Average Total time = 3257
$ For MLFQ(MultiLevel Feedback Queue)
```

```
Activities   Terminal ▾                    Nov 2  21:33                         79 %
pratyanshu@pratyanshu-Inspiron-5570: ~/Sem 3/Operating Systems and Networks/Assignments/Assignment 5/2019101025/xv6-public
SeaBIOS (version 1.13.0-1ubuntu1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA10+1FECCA10 CA00



Booting from Hard Disk..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
$ benchmark 2
For this benchmark we have calculated sum of runtimes
and waiting times of all the sub processes and
 divided by number of subprocesses to get the average
Process with pid = 4 exited with rtime = 897 and wtime = 0 and total time = 897
Process with pid = 5 exited with rtime = 918 and wtime = 0 and total time = 918
Process with pid = 6 exited with rtime = 940 and wtime = 697 and total time = 1637
Process with pid = 7 exited with rtime = 941 and wtime = 718 and total time = 1659
Process with pid = 8 exited with rtime = 935 and wtime = 1437 and total time = 2372
Process with pid = 9 exited with rtime = 934 and wtime = 1459 and total time = 2393
Process with pid = 10 exited with rtime = 936 and wtime = 2173 and total time = 3109
Process with pid = 11 exited with rtime = 939 and wtime = 2194 and total time = 3133
Process with pid = 13 exited with rtime = 0 and wtime = 2934 and total time = 2934
Process with pid = 12 exited with rtime = 945 and wtime = 2910 and total time = 3855
Average Run time(time running on CPU i.e. RUNNING state) = 838
Average Wait time(time spent in RUNNABLE state) = 1452
Average Total time = 2290
$ For FCFS(First Come First Serve)
```

When the bcnt.c contains small process we see by waiting time and total time the ranking is:

1. FCFS
2. PBS
3. RR
4. MLFQ

Case 2: When bcnt.c contains a relatively longer process

The ranking becomes (screenshots below)

1.PBS
2.FCFS
3.MLFQ
4.RR

Reasoning:

When bcnt.c is smaller then FCFS dominates as it has very little overhead and since processes are not that long the waiting time is not adversely affected. When bcnt.c becomes a bit larger FCFS takes a hit and falls down to second position.

PBS dominates falls behind FCFS in case 1 because of its overhead but since processes become longer in case 2 the low priority of I/O process allows PBS to take lead in waiting times as I/O finishes fast leaving less processes in queue.

MLFQ is worst in case 1 due to its huge overhead. MLFQ though provides the best responsiveness and response times. The waiting times are also equally high due to increased overheads and increased waitings in lower priority queues.

RR has a huge problem that it is interrupted quite often when all the processes are CPU bound this gives it very good responsiveness but a big overhead which creates huge waiting times. This is further reflected when bcnt.c becomes longer and RR drops to last position.

Activities Terminal Nov 2 21:46 86 %

pratyanshu@pratyanshu-Inspiron-5570: ~/Sem 3/Operating Systems and Networks/Assignments/Assignment 5/2019101025/xv6-public

```
SeaBIOS (version 1.13.0-1ubuntu1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA10+1FECCA10 CA00



Booting from Hard Disk..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
$ benchmark 2
For this benchmark we have calculated sum of runtimes
and waiting times of all the sub processes and
 divided by number of subprocesses to get the average
Process with pid = 13 exited with rtime = 0 and wtime = 575 and total time = 575
Process with pid = 4 exited with rtime = 2371 and wtime = 5903 and total time = 8274
Process with pid = 5 exited with rtime = 2337 and wtime = 6412 and total time = 8749
Process with pid = 6 exited with rtime = 2319 and wtime = 6851 and total time = 9170
Process with pid = 7 exited with rtime = 2347 and wtime = 7111 and total time = 9458
Process with pid = 8 exited with rtime = 2360 and wtime = 7183 and total time = 9543
Process with pid = 9 exited with rtime = 2362 and wtime = 7228 and total time = 9590
Process with pid = 10 exited with rtime = 2383 and wtime = 7203 and total time = 9586
Process with pid = 11 exited with rtime = 2385 and wtime = 7174 and total time = 9559
Process with pid = 12 exited with rtime = 2378 and wtime = 7175 and total time = 9553
Average Run time(time running on CPU i.e. RUNNING state) = 2124
Average Wait time(time spent in RUNNABLE state) = 6281
Average Total time = 8405
$ For Round Robin(Default)
```

Activities Terminal Nov 2 21:49 88 %

pratyanshu@pratyanshu-Inspiron-5570: ~/Sem 3/Operating Systems and Networks/Assignments/Assignment 5/2019101025/xv6-public

```
SeaBIOS (version 1.13.0-1ubuntu1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA10+1FECCA10 CA00



Booting from Hard Disk..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
$ benchmark 2
For this benchmark we have calculated sum of runtimes
and waiting times of all the sub processes and
 divided by number of subprocesses to get the average
Process with pid = 13 exited with rtime = 0 and wtime = 0 and total time = 0
Process with pid = 11 exited with rtime = 2304 and wtime = 0 and total time = 2304
Process with pid = 12 exited with rtime = 2266 and wtime = 0 and total time = 2266
Process with pid = 10 exited with rtime = 2315 and wtime = 2104 and total time = 4419
Process with pid = 9 exited with rtime = 2265 and wtime = 2467 and total time = 4732
Process with pid = 8 exited with rtime = 2158 and wtime = 4420 and total time = 6578
Process with pid = 7 exited with rtime = 2190 and wtime = 4732 and total time = 6922
Process with pid = 6 exited with rtime = 2642 and wtime = 6579 and total time = 9221
Process with pid = 5 exited with rtime = 2673 and wtime = 6923 and total time = 9596
Process with pid = 4 exited with rtime = 2555 and wtime = 9223 and total time = 11778
Average Run time(time running on CPU i.e. RUNNING state) = 2136
Average Wait time(time spent in RUNNABLE state) = 3644
Average Total time = 5781
$ For PBS(Priority Based Scheduler)
```

Activities Terminal Nov 2 21:43 85 %

pratyanshu@pratyanshu-Inspiron-5570: ~/Sem 3/Operating Systems and Networks/Assignments/Assignment 5/2019101025/xv6-public

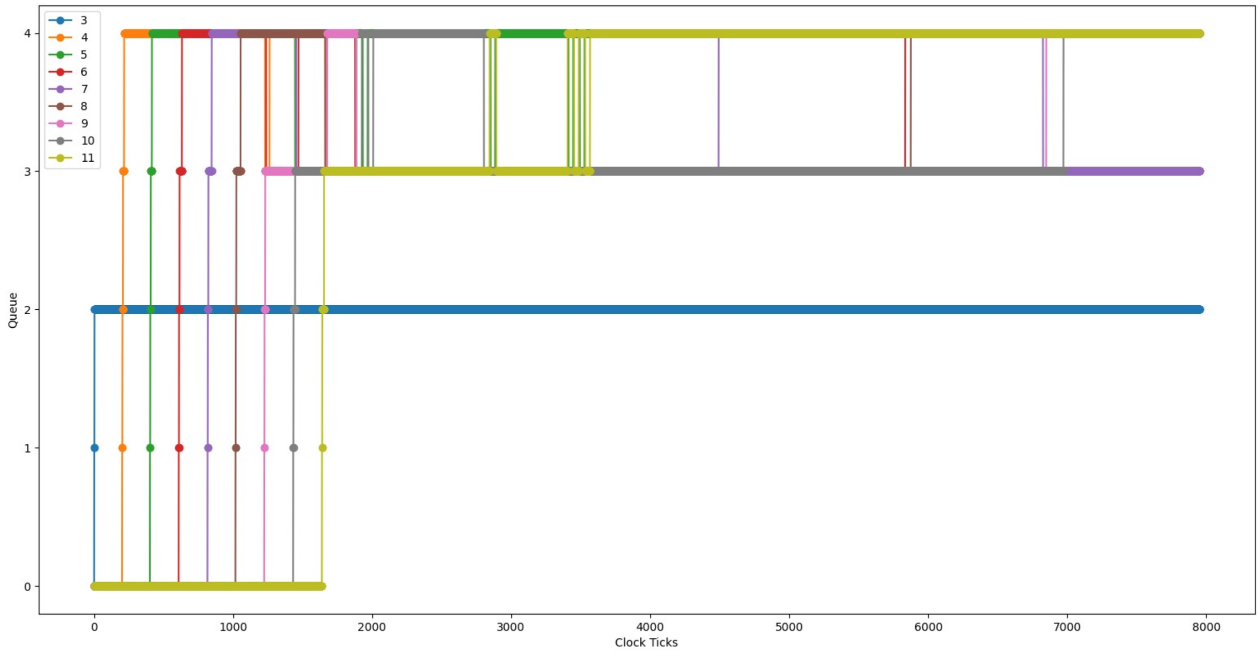
```
SeaBIOS (version 1.13.0-1ubuntu1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA10+1FECCA10 CA00



Booting from Hard Disk..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
$ benchmark 2
For this benchmark we have calculated sum of runtimes
and waiting times of all the sub processes and
 divided by number of subprocesses to get the average
Process with pid = 13 exited with rtime = 0 and wtime = 1385 and total time = 1385
Process with pid = 5 exited with rtime = 1926 and wtime = 3645 and total time = 5571
Process with pid = 4 exited with rtime = 1932 and wtime = 4748 and total time = 6680
Process with pid = 11 exited with rtime = 2171 and wtime = 5590 and total time = 7761
Process with pid = 6 exited with rtime = 2058 and wtime = 5895 and total time = 7953
Process with pid = 7 exited with rtime = 2079 and wtime = 6942 and total time = 9021
Process with pid = 12 exited with rtime = 2189 and wtime = 6691 and total time = 8880
Process with pid = 10 exited with rtime = 2153 and wtime = 6897 and total time = 9050
Process with pid = 8 exited with rtime = 2112 and wtime = 7018 and total time = 9130
Process with pid = 9 exited with rtime = 2126 and wtime = 7102 and total time = 9228
Average Run time(time running on CPU i.e. RUNNING state) = 1874
Average Wait time(time spent in RUNNABLE state) = 5591
Average Total time = 7465
$ For MLFQ(Multilevel Feedback Queue)
```

Activities Terminal Nov 2 21:52 89 %

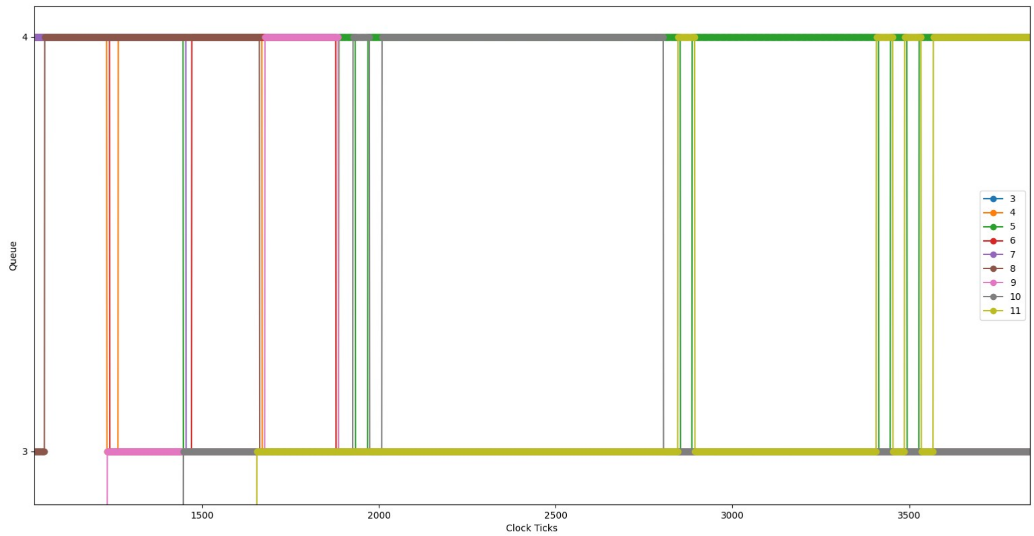
pratyanshu@pratyanshu-Inspiron-5570: ~/Sem 3/Operating Systems and Networks/Assignments/Assignment 5/2019101025/xv6-public

```
SeaBIOS (version 1.13.0-1ubuntu1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA10+1FECCA10 CA00



Booting from Hard Disk..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
$ benchmark 2
For this benchmark we have calculated sum of runtimes
and waiting times of all the sub processes and
 divided by number of subprocesses to get the average
Process with pid = 4 exited with rtime = 2260 and wtime = 0 and total time = 2260
Process with pid = 5 exited with rtime = 2237 and wtime = 0 and total time = 2237
Process with pid = 6 exited with rtime = 2113 and wtime = 2062 and total time = 4175
Process with pid = 7 exited with rtime = 2133 and wtime = 2062 and total time = 4195
Process with pid = 8 exited with rtime = 2118 and wtime = 3976 and total time = 6094
Process with pid = 9 exited with rtime = 2139 and wtime = 3996 and total time = 6135
Process with pid = 10 exited with rtime = 2448 and wtime = 5895 and total time = 8343
Process with pid = 11 exited with rtime = 2501 and wtime = 5937 and total time = 8438
Process with pid = 13 exited with rtime = 0 and wtime = 8239 and total time = 8239
Process with pid = 12 exited with rtime = 2287 and wtime = 8144 and total time = 10431
Average Run time(time running on CPU i.e. RUNNING state) = 2023
Average Wait time(time spent in RUNNABLE state) = 4031
Average Total time = 6054
$ For FCFS (First Come First Serve)
```

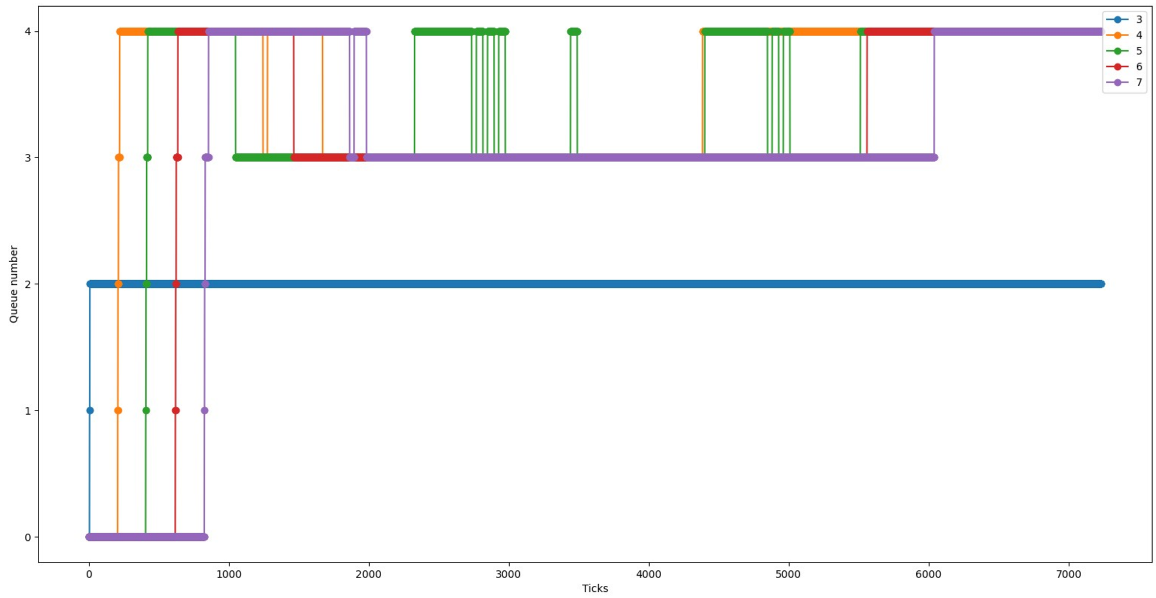
Bonus Graphs For MLFQ:



The zoomed version displaying the points where aging affects the current queue of process



With a shorter aging time:



Note: More images are in Graph Images folder. Please look at them.