Project 3: AUBatch Report

Jordan Sosnowski

2020-3-9

Contents

Introduction	4
I. Problem Description	4
II. Background	4
Central Processing Unit	4
First Come, First Served	4
Shortest Job First	5
Priority Based	5
Design and Implementation	5
I. Dataflow Diagram	5
II. Project Design	7
III. AUBatch	7
IV. Commandline	9
V. Modules	22
Performance Metrics	34
Instant Arrival	34
First Come First Served, 5 Jobs, Arrival Time 0, Priority Range 0-5, CPU Burst Range 0-10	34
Shortest Job First, 5 Jobs, Arrival Time 0, Priority Range 0-5, CPU Burst Range 0-10	36
Priority Based, 5 Jobs, Arrival Time 0, Priority Range 0-5, CPU Burst Range 0-10	38
Two Second Arrival	40
First Come First Served, 5 Jobs, Arrival Time 2, Priority Range 0-5, CPU Burst Range 0-10	40
Shortest Job First, 5 Jobs, Arrival Time 2, Priority Range 0-5, CPU Burst Range 0-10	42
Priority Based, 5 Jobs, Arrival Time 2, Priority Range 0-5, CPU Burst Range 0-10	44
Max Burst < Arrival Time	46
First Come First Served, 5 Jobs, Arrival Time 5, Priority Range 0-5, CPU Burst Range 0-3	46
Shortest Job First, 5 Jobs, Arrival Time 5, Priority Range 0-5, CPU Burst Range 0-3	48
Priority Based, 5 Jobs, Arrival Time 0, Priority Range 0-5, CPU Burst Range 0-3	50
Number of Jobs > Queue Size	52
First Come First Served, 15 Jobs, Arrival Time 0, Priority Range 0-5, CPU Burst Range 0-3	52
Shortest Job First, 15 Jobs, Arrival Time 0, Priority Range 0-5, CPU Burst Range 0-3	56
Priority Based, 15 Jobs, Arrival Time 0, Priority Range 0-5, CPU Burst Range 0-3	60
Performance Evaluations	64
Lessons Learned	64

Project 3: AUBatch Report	2020-3-9
Conclusion	64
References	65

Introduction

I. Problem Description

Central processing units are the core of any computer. Any program that has to run has to go through the CPU, as without the CPU the program cannot be executed. However, one single program cannot fully utilize a CPU; therefore, if we were to leave a single program on the CPU until it is finished executing we would be wasting valuable time. Imagine if you could only run one program at a time per CPU on a computer, that would be horrendous. Therefore, it is important to keep a CPU as active as possible. For example, if one program is busing doing I/O it should probably be booted off the CPU so a program that can use the CPU's resources can be loaded. But which program should be loaded next? AUBatch is a simulation that looks into process, or job, scheduling. We look into three algorithms: first come, first served, shortest job first, and priority-based.

Additionally, we assume all of our algorithms are non-preemptive, so once a process is loaded onto the CPU it is there until it is completed. Preemptive algorithms are extremely popular and efficient, as state earlier, but to implement one is out of scope for this current project.

II. Background

To fully understand some of the algorithms and technologies discussed in this paper, a background in these methodologies needs to be established.

Central Processing Unit

A central processing unit (CPU)¹ is hardware that executes instructions that make up a computer program. Also referred to as the brain of the computer, without it the computer would not be able to operate. Most modern CPUs have multiple cores, each core can load a single thread of execution. Therefore a CPU with two cores can run two parallel processes.

First Come, First Served

First come, first served (FCFS)² is a scheduling algorithm that loads processes onto the CPU as they arrive. Therefore, if three processes arrive in the following order A, C, B they will execute in the same order.

Shortest Job First

Shortest job first (SJF)³, also known as shortest job next, loads processes based on the remaining CPU burst time. This scheduler minimizes response time as jobs are usually loaded faster.

Priority Based

Priority based scheduling is similar to SJF. Instead of sorting by remaining CPU burst, it will sort based on priority (highest priority first, lowest last). Our implementation of priority based is non-preemptive, most are preemptive.

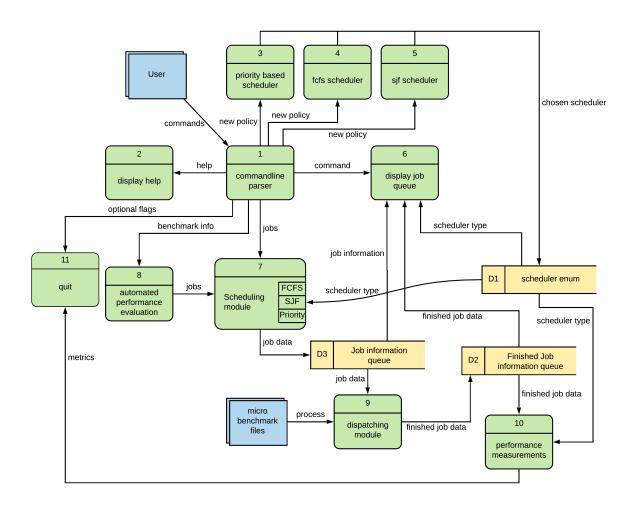
Design and Implementation

I. Dataflow Diagram

Here we have the dataflow diagram for the AUBatch framework. We have two external entities, user, and microbenchmark files. There are 10 processes and 3 data stores. Our user interacts with the commandline parser which will call a function depending on the input.

We can provide different scheduling types, help / h / ?, list/ls, run ..., test ..., and quit.

- run will call the scheduling module which will, in turn, load the job into the job information queue which is used by the dispatching module which when finished will load the finished job onto the finished job information queue.
- list will call the display job queue which pulls information from the scheduler enum, the finished job information queue, and the job information queue.
- test will call the automated performance evaluation process which sends jobs to the scheduling module.
- fcfs / sjf / priority will change the current scheduling algorithm by loading it into the scheduler enum.
- quit can take two optional flags -i or -d. -i will wait until the current job finished, -d will wait until all the jobs finish. If no flag is provided it will simply quit immediately.



II. Project Design

I decided to split up the project into three different files: aubatch.c contains the main method and is the driver for the entire process, commandline.c parses user input and decided how to react, modules.c contains the scheduler and dispatcher modules that are feed data from commandline.c.

Also, I have a helper file called microbatch.c which can be called by the user during benchmarks or runs. It takes a command-line argument, which is handled by run, and will sleep for n seconds, where n is the command-line argument provided by the user. It is matched with cpu_burst_time. Therefore, if a user were to provide run ./microbatch.out 10 1 it would call system with ./ microbatch.out 10 and therefore instruct microbatch to sleep for 10 seconds.

I have two header files commandline.h and modules.h both are imported in aubatch.c and modules.h is imported in commandline.c. Additionally, for error checking I make sure a binary exists before it is run via run, to do this I simply open the file and if it cannot open it warns the user. Since fopen does not use the system path you will have to provide a full or relative path for binaries if you wish to run them. For example run ls 10 1 will not work but run /bin/ls 10 1 will.

Also, packaged with the report are some script files that show me running aubatch under different conditions.

III. AUBatch

I will say here before I get into the code discussion for any function that I was not familiar with I used cplusplus⁵ as reference material as it is an unofficial C/C++ API. I also used GeeksforGeeks⁶ for some fundamental implementations such as how to use qsort. Additionally, I used Dr. Qin's sample code as a base for most of this project.

The first module we will discuss is aubatch.c. It is the driver for the whole framework and the only module with a main function.

We first include commandline.h and modules.h so we can gain the functionality of those two files.

```
1 #include "commandline.h"
2 #include "modules.h"
```

After the include statement, we flesh out main. Within main we print the welcome message, and we declare and instantiate a multitude of variables used throughout commandline.c and modules.c. Following this, we create two threads, one that calls commandline located in commandline.c and another that calls dispatcher which is located in modules.c.

count is used to determine the number of jobs in the waiting queue, plus the job that is currently on the CPU. buf_head and buf_tail point to spots within process_buffer an array of running and waiting processes. buf_head points to the next available spot in the array and buf_tail points to the next process that should be loaded, as specified by the scheduling algorithm, onto the CPU. finished_head points to the next available spot in the finished_process_buffer an array of processes that have finished execution. batch is used as a flag to denote whether we are adding jobs in batch mode or not. This is only set to 1 when we do a benchmark with an arrival rate of 0. This means we are assuming all the jobs are arriving at the same time.

Following this, we wait for the threads to join, and if they have return values we print them out.

```
int main(int argc, char **argv)
2
   {
       printf("Welcome to Jordan Sosnowski's batch job scheduler Version
3
          1.0.\nType 'help' to find more about AUbatch commands.\n");
4
       pthread_t executor_thread, dispatcher_thread; /* Two concurrent
          threads */
5
6
       int iret1, iret2;
8
       policy = FCFS; // default policy for scheduler
9
       /* Initialize count, three buffer pointers */
       count = 0;
12
       buf_head = 0;
13
       buf_tail = 0;
       finished_head = 0;
14
15
       batch = 0;
16
       /* Create two independent threads: executor and dispatcher */
17
18
       iret1 = pthread_create(&executor_thread, NULL, commandline, (void
19
           *)NULL);
       iret2 = pthread_create(&dispatcher_thread, NULL, dispatcher, (void
20
           *)NULL);
21
       /* Initialize the lock the two condition variables */
23
       pthread_mutex_init(&cmd_queue_lock, NULL);
24
       pthread_cond_init(&cmd_buf_not_full, NULL);
25
       pthread_cond_init(&cmd_buf_not_empty, NULL);
27
       /* Wait till threads are complete before main continues. Unless we
            */
       /* wait we run the risk of executing an exit which will terminate
28
            */
       /* the process and all threads before the threads have completed.
             */
       pthread_join(executor_thread, NULL);
31
       pthread_join(dispatcher_thread, NULL);
```

```
if (iret1)
printf("executor_thread returns: %d\n", iret1);
if (iret2)
printf("dispatcher_thread returns: %d\n", iret1);
return 0;
}
```

IV. Commandline

Within commandline.c we include modules.h and commandline.h.

```
1 #include "commandline.h"
2 #include "modules.h"
```

Following this, we declare an array of strings that define the different values help should print out.

```
1 static const char *helpmenu[] = {
      "run <job> <time> <priority>: submit a job named <job>, execution
          time is <time>, priority is <pr>",
3
      "list: display the job status",
      "help: Print help menu",
4
5
      "fcfs: change the scheduling policy to FCFS",
      "sjf: changes the scheduling policy to SJF",
6
      "priority: changes the scheduling policy to priority",
7
8
      "test <benchmark> <policy> <num_of_jobs> <arrival_time> <
          priority_levels> <min_CPU_time> <max_CPU_time>",
      "quit: Exit AUbatch | -i quits after current job finishes | -d
          quits after all jobs finish",
      NULL);
```

Next, we define a custom type cmd which is a struct that houses a string and a function. After that, we define an array of cmds. This will be used by cmd_dispatch to help decide which function to call based on the input value.

```
typedef struct
{
    const char *name;
    int (*func)(int nargs, char **args);
} cmd;

// array of cmds to be used by the command line
static const cmd cmdtable[] = {
    {"?", cmd_helpmenu},
    {"h", cmd_helpmenu},
    {"help", cmd_helpmenu},
    {"run", cmd_run},
}
```

After this, we hit commandline which is called by the executor thread back in aubatch.c. This is where the command line gets input from the user to then determine what to do based on said input.

```
void *commandline(void *ptr)
2
   {
3
4
       char *buffer;
5
6
       buffer = (char *)malloc(MAX_CMD_LEN * sizeof(char));
       if (buffer == NULL)
7
8
9
            perror("Unable to malloc buffer");
            exit(1);
       }
12
13
       while (1)
14
            printf("> [? for menu]: ");
15
            fgets(buffer, MAX_CMD_LEN, stdin);
17
            remove_newline(buffer);
18
            cmd_dispatch(buffer);
19
20
       return (void *)NULL;
21
   }
```

Next, we will look into cmd_dispatch this is the controller of commandline as it helps determine code flow. Within this function we first determine the number of arguments, we assume arguments are space delimitated. To determine the number of arguments we use strtok to tokenize the string. If we send in a command that has more than 8 arguments we remind the user that they have provided more than the tool can handle. If we provided test bench1 fcfs 5 0 6 1 10 the arggs array would be as follows: args[0] = test, args[1] = bench, args[2] = fcfs, args[3] = 5, args[4] = 0, args[5] = 6, args[6] = 1, and args[7] = 10.

After that we loop through cmdtable to see if args [0] equals a function. For example if we provided run ./microbatch.out 10, args[0] would equal run which matches with cmd_run in cmdtable. Once we find the correct function we call it and pass args and nargs as arguments.

```
1 int cmd_dispatch(char *cmd)
```

```
2 {
 3
        char *args[MAXMENUARGS];
 4
        int nargs = 0;
        char *word;
 6
        char *context;
 7
        int i, result;
8
9
        for (word = strtok_r(cmd, " ", &context);
             word != NULL;
             word = strtok_r(NULL, " ", &context))
11
12
        {
13
14
            if (nargs >= MAXMENUARGS)
15
                printf("Command line has too many words\n");
16
17
                return E2BIG;
18
19
            args[nargs++] = word;
20
        }
21
        if (nargs == 0)
22
23
        {
24
            return 0;
25
        }
26
        for (i = 0; cmdtable[i].name; i++)
27
28
29
            if (*cmdtable[i].name && !strcmp(args[0], cmdtable[i].name))
            {
                assert(cmdtable[i].func != NULL);
32
                result = cmdtable[i].func(nargs, args);
33
34
                return result;
            }
        }
37
        printf("%s: Command not found\n", args[0]);
38
        return EINVAL;
40 }
```

What follows next are the implementations for each cmd function.

```
| Snow — /bin/sh — /bin/sh — microbatch.out * aubatch.out — 110×23 |
|bash-3.2$ ./aubatch.out |
|welcome to Jordan Sosnowski's batch job scheduler Version 1.0. |
|Type 'help' to find more about AUbatch commands. |
|? for menu]: run ./microbatch.out 40 1 |
|Job ./microbatch.out was submitted. |
|Total number of jobs in the queue: 1 |
|Expected waiting time: 0 |
|Scheduling Policy: FCFS. |
|? for menu]: run ./microbatch.out 5 1 |
|Job ./microbatch.out was submitted. |
|Total number of jobs in the queue: 2 |
|Expected waiting time: 40 |
|Scheduling Policy: FCFS. |
|? for menu]: |
```

cmd_run, which is called via run or r, will check to ensure you passed the right number of parameters. After that, it will call scheduler and pass it the command line arguments provided.

```
int cmd_run(int nargs, char **args)
2
   {
       if (nargs != 4)
3
4
5
           printf("Usage: run <job> <time> <priority>\n");
           return EINVAL;
6
       }
7
       // ensure file exists first
8
       FILE *f = fopen(args[1], "r");
9
       if (f == NULL)
10
            printf("Error file does not exist. Please use relative or full
12
               path\n");
13
           fclose(f);
14
           return EINVAL;
       }
15
       fclose(f);
16
       scheduler(nargs, args);
17
       return 0; /* if succeed */
18
19 }
```

cmd_quit, which is called via quit or q, will first check and see if you passed any flags with it. If you pass -i aubatch will wait for the current process on the CPU to finish. If you pass -d aubatch will wait for all processes to finish.

After waiting, or not waiting, it will call report_metrics and then exit.

```
int cmd_quit(int nargs, char **args)
2
   {
3
       if (nargs == 2)
4
           if (!strcmp(args[1], "-i")) // wait for current job to finish
5
               running
6
8
               int cur_count = count;
               printf("Waiting for current job to finish ... n");
9
10
               if (count)
11
                   while (cur_count == count)
                    {
14
                    }
               }
           else if (!strcmp(args[1], "-d")) // wait for all jobs to finish
17
```

```
18
19
                 printf("Waiting for all jobs to finish...\n");
20
                 while (count)
21
                 {
22
                 }
23
            }
24
        }
25
        printf("Quiting AUBatch... \n");
26
27
        report_metrics();
28
29
        exit(0);
30 }
```

```
bash-3.2$ ./aubatch.out

Welcome to Jordan Sosnowski's batch job scheduler Version 1.0.

[Type 'help' to find more about AUbatch commands.

> [? for menu]: help

AUbatch help menu
run <job> <time> <priority>: submit a job named <job>, execution time is <time>, priority is <pr>
list: display the job status
help: print help menu
fefs: change the scheduling policy to FCFS
sjf: changes the scheduling policy to SJF
priority: changes the scheduling policy to priority
test <benchmark> <policy> <num_of_jobs> <arrival_time> <priority_levels> <min_CPU_time> <max_CPU_time>
quit: exit AUbatch | -i quits after current job finishes | -d quits after all jobs finish

> [? for menu]:
```

cmd_helpmenu, which can be called with help, h, or ?, will loop through each helpmenu array element and print to the screen for the user.

```
int cmd_helpmenu(int n, char **a)
2
   {
3
       printf("\n");
4
       printf("AUbatch help menu\n");
6
       int i = 0;
8
       while (1)
9
            if (helpmenu[i] == NULL)
11
12
                break;
13
            printf("%s\n", helpmenu[i]);
14
            i++;
16
       }
       printf("\n");
```

```
18 return 0;
19 }
```

```
math show math shows a show math shows a show math shows math sha
[bash-3.2$ ./aubatch.out
Welcome to Jordan Sosnowski's batch job scheduler Version 1.0.
Type 'help' to find more about AUbatch commands.
> [? for menu]: run ./microbatch.out 60 1
Job ./microbatch.out was submitted.
Total number of jobs in the queue: 1
Expected waiting time: 0
Scheduling Policy: FCFS.
> [? for menu]: run ./microbatch.out 50 2
Job ./microbatch.out was submitted.
Total number of jobs in the queue: 2
Expected waiting time: 60
Scheduling Policy: FCFS.
 > [? for menu]: run ./microbatch.out 40 3
Job ./microbatch.out was submitted.
  Total number of jobs in the queue: 3
Expected waiting time: 110
Scheduling Policy: FCFS.
> [? for menu]: run ./microbatch.out 30 2
Job ./microbatch.out was submitted.
Total number of jobs in the queue: 4
Expected waiting time: 150
Scheduling Policy: FCFS.
  > [? for menu]: run ./microbatch.out 20 1
 Job ./microbatch.out was submitted.
Total number of jobs in the queue: 5
  Expected waiting time: 180
 Scheduling Policy: FCFS.
 > [? for menu]: run ./microbatch.out 10 0
Job ./microbatch.out was submitted.
Total number of jobs in the queue: 6
Expected waiting time: 200
Scheduling Policy: FCFS.
> [? for menu]: list
                                                                   CPU_Time Pri Arrival_time Progress
60 1 Mon Mar 9 16:26:29 2020 running
50 2 Mon Mar 9 16:26:33 2020 ------
40 3 Mon Mar 9 16:26:36 2020 ------
30 2 Mon Mar 9 16:26:38 2020 ------
20 1 Mon Mar 9 16:26:42 2020 ------
10 0 Mon Mar 9 16:26:44 2020 -----
 Name
./microbatch.out 60
./microbatch.out 50
./microbatch.out 40
./microbatch.out 30
./microbatch.out 20
./microbatch.out 10
> [? for menu]: sjf
Scheduling policy is switched to SJF. All the 6 waiting jobs have been rescheduled.
> [? for menu]: list
 J_Time Pri Arrival_time Progress
1 Mon Mar 9 16:26:29 2020 running
0 Mon Mar 9 16:26:44 2020 ------
1 Mon Mar 9 16:26:42 2020 ------
2 Mon Mar 9 16:26:38 2020 -----
3 Mon Mar 9 16:26:36 2020 -----
2 Mon Mar 9 16:26:33 2020 -----
   ./microbatch.out
> [? for menu]: priority
Scheduling policy is switched to Priority. All the 6 waiting jobs have been rescheduled.
> [? for menu]: list
                                                                         st
CPU_Time Pri Arrival_time Progress
60 1 Mon Mar 9 16:26:29 2020 running
40 3 Mon Mar 9 16:26:36 2020 -----
30 2 Mon Mar 9 16:26:38 2020 -----
50 2 Mon Mar 9 16:26:33 2020 -----
20 1 Mon Mar 9 16:26:42 2020 -----
10 0 Mon Mar 9 16:26:44 2020 -----
   ./microbatch.out
                                                                      60
40
   ./microbatch.out
   ./microbatch.out
   ./microbatch.out
                                                                       50
                                                                        20
   ./microbatch.out
   ./microbatch.out
  > [? for menu]:
```

cmd_priority, which is called with priority, will change the current scheduling policy to priority
based.

```
int cmd_priority()
{
    policy = PRIORITY;
    change_scheduler();
    return 0;
}
```

cmd_sjf, which is called with sjf, will change the current scheduling policy to the shortest job first.

```
1 int cmd_sjf()
2 {
3     policy = SJF;
4     change_scheduler();
5     return 0;
6 }
```

cmd_fcfs, which is called with fcfs, will change the current scheduling policy to first come, first served.

```
int cmd_sjf()
{
    policy = FCFS;
    change_scheduler();
    return 0;
}
```

Each change in scheduling algorithm will also call change_scheduler which will print out some information to the screen for the user. It will also resort to the buffer to ensure the processes are in the correct order for the new scheduler.

```
AUbatch — /bin/bash — /bin/bash — microbatch.out • aubatch.out — 92×25
[bash-3.2$ ./aubatch.out
Welcome to Jordan Sosnowski's batch job scheduler Version 1.0.
Type 'help' to find more about AUbatch commands.
> [? for menu]: run ./microbatch.out 10 1
Job ./microbatch.out was submitted.
Total number of jobs in the queue: 1
Expected waiting time: 0
Scheduling Policy: FCFS.
> [? for menu]: run ./microbatch.out 20 1
Job ./microbatch.out was submitted.
Total number of jobs in the queue: 2
Expected waiting time: 10
Scheduling Policy: FCFS.
> [? for menu]: run ./microbatch.out 4 2
Job ./microbatch.out was submitted.
Total number of jobs in the queue: 2
Expected waiting time: 20
Scheduling Policy: FCFS.
> [? for menu]: 1s
                              CPU_Time Pri Arrival_time
                                                                                           Progress
                                          1 Mon Mar 9 16:39:32 2020 finished
1 Mon Mar 9 16:39:37 2020 running
2 Mon Mar 9 16:39:44 2020 -----
 ./microbatch.out
 ./microbatch.out
                              20
 ./microbatch.out
> [? for menu]:
```

cmd_list, which is called vials or list, will list the running process, and all the finished and waiting processes and relevant information about them. If you have no processes waiting, running, or finished it will notify you.

```
int cmd_list()
2
3
       if (finished_head || count)
4
            printf("Name
5
                                         CPU_Time Pri Arrival_time
                            Progress\n");
6
            int i;
            for (i = 0; i < finished_head; i++)</pre>
7
8
9
                finished_process_p process = finished_process_buffer[i];
                char *status = "finished";
11
12
13
                char *time = convert_time(process->arrival_time);
                remove_newline(time);
14
                printf("%-18s %-8d %-3d %s %s\n",
                       process->cmd,
17
                       process->cpu_burst,
18
                       process->priority,
19
                        time,
20
                        status);
21
```

```
22
            for (i = 0; i < buf_head; i++)</pre>
23
24
25
26
                process_p process = process_buffer[i];
                char *status = "----";
27
28
                if (process->cpu_remaining_burst == 0)
29
                {
                    continue;
                }
31
                else if (process->first_time_on_cpu > 0 && process->
32
                    cpu_remaining_burst > 0)
                {
                    status = "running ";
34
                }
37
                char *time = convert_time(process->arrival_time);
38
                remove_newline(time);
                printf("%-18s %-8d %-3d %s %s\n",
40
                       process->cmd,
41
                       process->cpu_burst,
42
                       process->priority,
43
                       time,
44
                       status);
45
            }
            printf("\n");
46
47
        }
48
        else
49
            printf("No processes loaded yet!\n");
50
        return 0;
51 }
```

```
AUbatch — /bin/bash — /bin/bash — aubatch.out — 88×52
bash-3.2$ ./aubatch.out
Welcome to Jordan Sosnowski's batch job scheduler Version 1.0.
Type 'help' to find more about AUbatch commands.
> [? for menu]: test bench fcfs 2 0 1 1 5
Benchmark is running please wait...
=== Reporting Metrics for FCFS ===
Metrics for job ./microbatch.out:
CPU Burst: 4 see
                                          4 seconds
            Interruptions:
                                           0 times
           Arrival Time: Mon Mar 9 16:41:21 2020
First Time on CPU: Mon Mar 9 16:41:21 2020
Finish Time: Mon Mar 9 16:41:25 2020
Turnaround Time: 4 seconds
Waiting Time:
           Waiting Time:
           Response Time:
                                          0 seconds
Metrics for job ./microbatch.out:
CPU Burst: 5 sec
                                    5 seconds
            Interruptions:
                                          0 times
           Alrival Time: Mon Mar 9 16:41:21 2020
First Time on CPU: Mon Mar 9 16:41:25 2020
Finish Time: Mon Mar 9 16:41:30 2020
Turnaround Time: 9 seconds
Waiting Time: 4 seconds
Response Time
                                         4 seconds
            Response Time:
Overall Metrics for Batch:
           Total Number of Jobs Completed: 2
Total Number of Jobs Submitted: 2
           Average Turnaround Time: 6.500 seconds
Average Waiting Time: 2.000 seconds
Average Response Time: 2.000 seconds
Average CPU Burst: 4.500 seconds
Total CPU Burst: 9 seconds
                                                          0.154 No./second
            Throughput:
           Max Turnaround Time:
                                                           9 seconds
           Min Turnaround Time:
                                                           4 seconds
           Max Waiting Time:
                                                           4 seconds
           Min Waiting Time:
                                                           0 seconds
            Max Response Time:
                                                           4 seconds
           Min Response Time:
                                                           0 seconds
            Max CPU Burst:
                                                           5 seconds
           Min CPU Burst:
                                                            4 seconds
> [? for menu]:
```

cmd_test, which is called with test, is the benchmark function. It takes 7 parameters: benchmark_name, policy, num_of_jobs, arrival_rate, priority, min_cpu_burst, and max_cpu_burst. If the user does not provide the right number of arguments or provides logical fallacies such as min_cpu_burst > max_cpu_burst the user will be notified. Additionally, for test to work, it assumes no other jobs have been run or are currently running. It assumes that because if

there are jobs currently on the CPU or jobs that need to be loaded it would mess with the metrics for the benchmark.

It calls test_scheduler which at a high level creates all the jobs needed for the benchmark and notifies dispatcher when appropriate. After the jobs finish, we report the metrics, free all the jobs from the finished buffer, and reset the head and tail variables.

We use **while**(count){} to see if any jobs are waiting to run. For each job that is loaded onto process_buffer count is incremented and for each job that is take off count is decremented. Therefore, once count is 0 we know there are no longer any jobs.

I could have used a conditional variable for this but decided not to.

```
int cmd_test(int nargs, char **argv)
2
   {
3
4
       srand(0); // ensure seed is set to the same value each time to make
            same jobs created
5
       if (nargs != 8)
6
            printf("Usage: test <benchmark> <policy> <num_of_jobs> <</pre>
               arrival_rate> <priority_levels> <min_CPU_time> <max_CPU_time
               >\n");
8
           return EINVAL;
       }
       else if (count || finished_head)
11
            printf("Error: Jobs current in queue / on CPU, no jobs should
               have ran if doing benchmark...\n");
13
           return EINVAL;
14
       }
       char *benchmark = argv[1];
       char *str_policy = argv[2];
16
17
       int num_of_jobs = atoi(argv[3]);
18
       int arrival_rate = atoi(argv[4]);
       int priority_levels = atoi(argv[5]);
20
       int min_cpu_burst = atoi(argv[6]);
21
       int max_cpu_burst = atoi(argv[7]);
22
23
       if (min_cpu_burst >= max_cpu_burst)
24
       {
            printf("Error: <min_CPU_time> cannot be greater than or equal
               to <max_CPU_time>\n");
26
           return EINVAL;
27
       }
       else if (num_of_jobs <= 0 || min_cpu_burst < 0 || max_cpu_burst < 0</pre>
28
            || priority_levels < 0 || arrival_rate < 0)</pre>
29
       {
            printf("Error: <num_of_jobs> cannot be equal or less than zero\
               nError: <min_CPU_time> <max_CPU_time> <arrival_rate> and <</pre>
```

```
priority_levels> must be greater than 0\n");
31
            return EINVAL;
32
       }
34
       if (!strcmp(str_policy, "fcfs"))
            policy = FCFS;
37
       }
       else if (!strcmp(str_policy, "sjf"))
38
39
40
            policy = SJF;
41
       }
42
       else if (!strcmp(str_policy, "priority"))
43
        {
44
            policy = PRIORITY;
       }
45
46
       else
47
            printf("Error: <policy> must be either fcfs, sjf, or priority\n
48
               ");
            return EINVAL;
49
50
       }
51
52
       test_scheduler(benchmark, num_of_jobs, arrival_rate,
           priority_levels, min_cpu_burst, max_cpu_burst);
       printf("Benchmark is running please wait...\n");
54
       while (count)
55
        {
56
       }
57
58
       report_metrics();
59
60
       // clear process queue and finished queue
61
        // ensures that the metrics aren't reported when quitting aubatch
        // also ensures if running metrics again that the prior jobs will
62
           not interfere
63
       int i;
       for (i = 0; i < finished_head; i++)</pre>
64
65
            free(finished_process_buffer[i]);
66
67
68
        finished_head = 0;
69
       buf_head = 0;
       buf_tail = 0;
70
71
72
       return 0;
73 }
```

V. Modules

modules.c contain the schedulers and dispatcher, without this file the process specified in run or test would not be able to run. commandline is simply an interface that interacts with the functions within modules.c

Therefore, within modules.c we only import modules.h.

```
1 #include "modules.h"
```

First we have test_scheduler, which is called by cmd_test in commandline.c. This function will use mutexes and condition variables to ensure there are no issues among threads.

```
pthread_mutex_lock(&cmd_queue_lock);

while (count == CMD_BUF_SIZE)

{
    pthread_cond_wait(&cmd_buf_not_full, &cmd_queue_lock);
}

pthread_mutex_unlock(&cmd_queue_lock);
```

After ensuring there are no issues and it is test_schedulers time to run we will enter a for loop. We will loop for num_of_jobs, this will allow us to create the number of jobs specified by cmd_test. We create process which is of type process_p; a pointer to process_t which is a custom type of a struct that holds all the information we need about a process. For test_scheduler we assume that each cmd to be run by the process will be ./microbatch.out which is a file that simply sleeps for n seconds. This will allow us to get a more accurate CPU burst time. We randomly generate the priority and cpu_burst based on the seed set in cmd_test. We set the seed to 0 for each time we call cmd_test to make sure each time we call benchmark the same jobs are created. If we do not ensure this we cannot accurately compare runs across different scheduling policies. If we provide an arrival_rate > 0 we will need to load the jobs and notify the dispatcher each time to ensure more accurate metrics. After we notify the dispatcher we will need to sleep for arrival_rate seconds. If arrival_rate is 0 we assume all the processes arrive at the same time, therefore, we create them all and load them onto the buffer and only once they are all created will we notify dispatcher.

If our loop increment becomes larger than CMD_BUF_SIZE we need to notify the dispatcher ahead of time to ensure our queue does not overflow.

```
1 if (!arrival_rate)
2   batch = 1;
3 else
4   batch = 0;
5
6 // create jobs based on num_of_jobs
```

```
int i;
   for (i = 0; i < num_of_jobs; i++)</pre>
8
9
       /* lock the shared command gueue */
       pthread_mutex_lock(&cmd_queue_lock);
       while (count == CMD_BUF_SIZE)
14
15
            pthread_cond_wait(&cmd_buf_not_full, &cmd_queue_lock);
       }
16
18
       pthread_mutex_unlock(&cmd_queue_lock);
20
       int priority = (rand() % (priority_levels + 1)) + 1;
21
       int cpu_burst = (rand() % (max_CPU_time + 1)) + min_CPU_time;
22
       process_p process = malloc(sizeof(process_t));
23
       strcpy(process->cmd, "./microbatch.out");
       process->arrival_time = time(NULL);
24
25
       process->cpu_burst = cpu_burst;
26
       process->cpu_remaining_burst = cpu_burst;
27
       process->priority = priority;
28
       process->interruptions = 0;
29
       process->first_time_on_cpu = 0;
31
       if (i >= CMD_BUF_SIZE) // if i is larger than cmd_buff we need to
           notify dispatcher earlier
       // without this we would be stuck forever
34
           pthread_mutex_lock(&cmd_queue_lock);
           while (count == CMD_BUF_SIZE)
            {
37
                pthread_cond_wait(&cmd_buf_not_full, &cmd_queue_lock);
38
           pthread_mutex_unlock(&cmd_queue_lock);
40
           pthread_mutex_lock(&cmd_queue_lock);
41
           process_buffer[buf_head] = process;
42
           count++;
43
            /* Move buf_head forward, this is a circular queue */
44
45
           buf_head++;
46
           sort_buffer(process_buffer);
47
           buf_head %= CMD_BUF_SIZE;
48
            /* Unlock the shared command queue */
49
           pthread_cond_signal(&cmd_buf_not_empty);
51
           pthread_mutex_unlock(&cmd_queue_lock);
52
53
       else if (arrival_rate) // if there is an arrival rate, notify
           dispatcher immediately and then sleep for arrival_rate
54
       {
```

```
56
           pthread_mutex_lock(&cmd_queue_lock);
57
           process_buffer[buf_head] = process;
58
           count++;
            /* Move buf_head forward, this is a circular queue */
           buf_head++;
61
62
           sort_buffer(process_buffer);
63
           buf_head %= CMD_BUF_SIZE;
64
            /* Unlock the shared command queue */
65
            pthread_cond_signal(&cmd_buf_not_empty);
           pthread_mutex_unlock(&cmd_queue_lock);
           sleep(arrival_rate); // wait for the arrival rate
       }
       else
71
       {
72
           pthread_mutex_lock(&cmd_queue_lock);
            process_buffer[buf_head] = process;
74
           count++;
            /* Move buf_head forward, this is a circular queue */
76
77
           buf_head++;
78
           sort buffer(process buffer);
79
           buf_head %= CMD_BUF_SIZE;
80
81
           pthread_mutex_unlock(&cmd_queue_lock);
82
       }
   }
   if (!arrival_rate) // if arrival rate is 0, load all the jobs and then
84
       notify dispatcher
85 {
       pthread_mutex_lock(&cmd_queue_lock);
86
87
       sort_buffer(process_buffer);
       /* Unlock the shared command queue */
91
92
       pthread_cond_signal(&cmd_buf_not_empty);
       pthread_mutex_unlock(&cmd_queue_lock);
94
  }
```

scheduler is called by cmd_run unlike test_scheduler this only loads one job at a time. Immediately after loading the job it always notifies dispatcher so the job can immediately start processing.

We pass get_process argv which simply transforms the provides user arguments into process_p a pointer to process_t which is a custom type of a struct that holds all the information we need about a process. After this, we call submit_job which simply prints some useful information to the screen for the user. After that, we load the process onto the buffer, increment count and buf_head, and sort the buffer with sort_buffer.

After all this, we notify dispatcher with pthread_cond_signal.

```
void scheduler(int argc, char **argv)
2
3
       /* lock the shared command queue */
4
       pthread_mutex_lock(&cmd_queue_lock);
5
6
       while (count == CMD_BUF_SIZE)
7
       {
8
            pthread_cond_wait(&cmd_buf_not_full, &cmd_queue_lock);
9
       }
11
       pthread_mutex_unlock(&cmd_queue_lock);
       process_p process = get_process(argv);
12
13
       // print information about job
14
       submit_job(process->cmd);
17
       process_buffer[buf_head] = process;
18
       pthread_mutex_lock(&cmd_queue_lock);
19
20
       count++;
21
22
       /* Move buf_head forward, this is a circular queue */
23
       buf_head++;
24
       buf_head %= CMD_BUF_SIZE;
25
26
       // ensure buffer is in accordance to current policy
27
       sort_buffer(process_buffer);
28
29
       /* Unlock the shared command queue */
       pthread_cond_signal(&cmd_buf_not_empty);
31
       pthread_mutex_unlock(&cmd_queue_lock);
32 }
```

dispatcher will grab the process from buf_tail as the sorting algorithm places the next process to be scheduled at the bottom of the queue. After we grab it off we call complete_process which runs the process and then loads up a completed process type and loads it onto another buffer.

After we return we decrement the count and move our tail forward. We will also set our running_process to NULL.

```
9
10
           // printf("In dispatcher: count = %d\n", count);
           while (count == 0)
12
13
            {
14
                pthread_cond_wait(&cmd_buf_not_empty, &cmd_queue_lock);
           }
16
           running_process = process_buffer[buf_tail];
           pthread_cond_signal(&cmd_buf_not_full);
18
19
            /* Unlock the shared command queue */
20
           pthread_mutex_unlock(&cmd_queue_lock);
           complete_process(running_process);
22
23
           /* Run the command scheduled in the gueue */
24
           count--;
25
            // printf("In dispatcher: process_buffer[%d] = %s\n", buf_tail,
26
                process_buffer[buf_tail]->cmd);
27
            /* Move buf_tail forward, this is a circular queue */
28
29
           buf_tail++;
           buf_tail %= CMD_BUF_SIZE;
31
32
            running_process = NULL;
       }
34
       return (void *)NULL;
35 }
```

Within calculate_wait which is called by submit_job, we estimate the wait time for the newest process. By wait time we mean the amount of time it will have to wait before it is loaded onto the CPU.

```
int calculate_wait()
2
   {
3
        int wait = 0;
       int i;
4
       for (i = buf_tail; i < buf_head; i++)</pre>
5
6
            wait += process_buffer[i]->cpu_remaining_burst;
7
8
       }
9
       return wait;
10 }
```

get_process which is called by scheduler takes in a string array and loads up the process's variables.

```
process_p get_process(char **argv)
{
    process_p process = malloc(sizeof(process_t));
```

```
remove_newline(argv[3]);
5
6
       // load process structure
7
       strcpy(process->cmd, argv[1]);
       process->arrival_time = time(NULL);
8
9
       process->cpu_burst = atoi(argv[2]);
       process->cpu_remaining_burst = process->cpu_burst;
11
       process->priority = atoi(argv[3]);
12
       process->interruptions = 0;
13
       process->first_time_on_cpu = 0;
14
       return process;
15 }
```

complete_process which is called in dispatcher performs all the commands needed when finishing a process. First, we will run the process, if the process's cmd is ./microbatch.out we will append the burst time to it and call system. microbatch.out expects an additional command and uses that to determine how long to sleep for.

If we don't provide that program we will simply just run it using system. However, we will run the command but pipe its output to /dev/null this unclutters our view. For example, if we were to run /bin/ls when it lists the files of the current directory it sends that to /dev/null instead of standard output.

After that, we create finished_process, of type finished_process_p a pointer to finished_process_t. This is similar to process_t except is has variables that are related to metrics.

After we create this variable we load its fields and then increment finished_head and free the original processes' memory.

```
1 void complete_process(process_p process)
2 {
3
       char cmd[MAX_CMD_LEN * 2];
       if (!strcmp(process->cmd, "./microbatch.out"))
4
           sprintf(cmd, "%s %d", process->cmd, process->
5
               cpu_remaining_burst);
6
       else
7
           sprintf(cmd, "%s > /dev/null", process->cmd);
8
       if (process->first_time_on_cpu == 0)
9
           process->first_time_on_cpu = time(NULL);
12
       system(cmd);
14
       process->cpu_remaining_burst = 0;
       finished_process_p finished_process = malloc(sizeof(
          finished_process_t));
       finished_process->finish_time = time(NULL);
17
18
```

```
//allows more accurate cpu burst, if we run ls 10 1, ls wont
           actually run for 10 seconds, therefore we need to update its
           burst time
       process->cpu_burst = (int)(finished_process->finish_time - process
20
           ->first_time_on_cpu);
21
22
       strcpy(finished_process->cmd, process->cmd);
       finished_process->arrival_time = process->arrival_time;
23
24
       finished_process->cpu_burst = process->cpu_burst;
25
       finished_process->interruptions = process->interruptions;
       finished_process->priority = process->priority;
       finished_process->first_time_on_cpu = process->first_time_on_cpu;
27
       finished_process->turnaround_time = finished_process->finish_time -
28
            finished_process->arrival_time;
29
       if (finished_process->turnaround_time)
           finished_process->waiting_time = finished_process->
               turnaround_time - finished_process->cpu_burst;
       else
           finished_process->waiting_time = 0;
34
       finished_process->response_time = finished_process->
           first_time_on_cpu - finished_process->arrival_time;
       finished_process_buffer[finished_head] = finished_process;
37
       finished_head++;
38
       free(process);
40 }
```

Next, we will discuss report_metrics. This function is called by commandline whenever we quit. It is a lot of code, but all it does it iterate through each finished process and print relative metrics.

It also takes note of specific metrics through each process so it can average them or get the min/max at the end.

```
void report_metrics()
2
   {
3
       if (!finished_head)
4
           printf("No jobs completed!\n");
5
6
           return:
7
       }
8
       int total_waiting_time = 0;
9
       int total_turnaround_time = 0;
       int total_response_time = 0;
11
       int total_cpu_burst = 0;
       int max_waiting_time = INT_MIN;
14
       int min_waiting_time = INT_MAX;
15
       int max_response_time = INT_MIN;
       int min_response_time = INT_MAX;
```

```
17
       int max_turnaround_time = INT_MIN;
18
       int min_turnaround_time = INT_MAX;
19
       int max_cpu_burst = INT_MIN;
20
       int min_cpu_burst = INT_MAX;
21
       printf("\n=== Reporting Metrics for %s ===\n\n", get_policy_string
23
        finished_process_p finished_process;
24
       int i = 0;
25
       for (; i < finished_head; i++)</pre>
26
            finished_process = finished_process_buffer[i];
27
28
            printf("Metrics for job %s:\n", finished_process->cmd);
            printf("\tCPU Burst:
                                            %d seconds\n", finished_process
               ->cpu_burst);
            printf("\tInterruptions:
                                            %d times\n", finished_process->
               interruptions);
            printf("\tPriority:
                                            %d\n", finished_process->
               priority);
34
            printf("\tArrival Time:
                                            %s", convert_time(
               finished_process->arrival_time));
35
            printf("\tFirst Time on CPU:
                                            %s", convert_time(
               finished_process->first_time_on_cpu));
            printf("\tFinish Time:
                                            %s", convert_time(
               finished_process->finish_time));
            printf("\tTurnaround Time:
                                            %d seconds\n", finished_process
               ->turnaround_time);
            printf("\tWaiting Time:
                                            %d seconds\n", finished_process
               ->waiting_time);
                                            %d seconds\n", finished_process
            printf("\tResponse Time:
40
               ->response_time);
           printf("\n");
41
42
43
           if (finished_process->waiting_time < min_waiting_time)</pre>
                min_waiting_time = finished_process->waiting_time;
44
45
            if (finished_process->turnaround_time < min_turnaround_time)</pre>
                min_turnaround_time = finished_process->turnaround_time;
46
47
            if (finished_process->response_time < min_response_time)</pre>
                min_response_time = finished_process->response_time;
48
49
            if (finished_process->cpu_burst < min_cpu_burst)</pre>
50
                min_cpu_burst = finished_process->cpu_burst;
51
            if (finished_process->waiting_time > max_waiting_time)
53
                max_waiting_time = finished_process->waiting_time;
54
            if (finished_process->turnaround_time > max_response_time)
                max_turnaround_time = finished_process->turnaround_time;
            if (finished_process->response_time > max_response_time)
                max_response_time = finished_process->response_time;
```

```
if (finished_process->cpu_burst > max_cpu_burst)
59
               max_cpu_burst = finished_process->cpu_burst;
           total_response_time += finished_process->response_time;
61
           total_waiting_time += finished_process->waiting_time;
62
63
           total_turnaround_time += finished_process->turnaround_time;
64
           total_cpu_burst += finished_process->cpu_burst;
       }
65
       printf("Overall Metrics for Batch:\n");
67
       printf("\tTotal Number of Jobs Completed: %d\n", finished_head);
69
       printf("\tTotal Number of Jobs Submitted: %d\n", finished_head + (
           buf_head - buf_tail));
       printf("\tAverage Turnaround Time:
                                                  %.3f seconds\n",
           total_turnaround_time / (float)i);
71
       printf("\tAverage Waiting Time:
                                                  %.3f seconds\n",
           total_waiting_time / (float)i);
                                                  %.3f seconds\n",
       printf("\tAverage Response Time:
           total_response_time / (float)i);
73
       printf("\tAverage CPU Burst:
                                                  %.3f seconds\n",
           total_cpu_burst / (float)i);
74
       printf("\tTotal CPU Burst:
                                                  %d seconds\n",
          total_cpu_burst);
       printf("\tThroughput:
75
                                                  %.3f No./second\n", 1 / (
           total_turnaround_time / (float)i));
       printf("\tMax Turnaround Time:
                                                  %d seconds\n",
          max_turnaround_time);
       printf("\tMin Turnaround Time:
                                                  %d seconds\n\n",
          min_turnaround_time);
79
80
       printf("\tMax Waiting Time:
                                                  %d seconds\n",
           max_waiting_time);
       printf("\tMin Waiting Time:
                                                  %d seconds\n\n",
           min_waiting_time);
82
83
       printf("\tMax Response Time:
                                                  %d seconds\n",
          max_response_time);
       printf("\tMin Response Time:
84
                                                  %d seconds\n\n",
          min_response_time);
85
       printf("\tMax CPU Burst:
                                                  %d seconds\n",
          max_cpu_burst);
       printf("\tMin CPU Burst:
                                                  %d seconds\n\n",
87
          min_cpu_burst);
88 }
```

sort_buffer is the implementation of the scheduling policy. We first determine which policy we are running, we do this to determine at run-time which sorting algorithm to run.

After this, we use qsort to sort the process_buffer. Note we do something weird with

process_buffer. We get the element at buf_tail, this ensures we get the current process that is not on the CPU. Then we get the address of this and pass it to qsort. We also only run for buf_head - buf_tail iterations. It is for the same reason as the prior command. If we were to remove the process of the buffer once it is run this would solve this but this workaround works.

```
void sort_buffer(process_p *process_buffer)
2
   {
3
       void *sort;
       switch (policy)
4
5
6
       case FCFS:
           sort = fcfs_scheduler;
7
8
           break;
9
       case SJF:
           sort = sjf_scheduler;
11
           break;
12
       case PRIORITY:
           sort = priority_scheduler;
14
       }
15
       int index;
17
       // if we are doing a batch job, aka arrival rate is not 0 then add
18
           1 to buf_tail
       // if we sort ahead of buf_tail for a batch job we will all the
19
           processes even tho
20
        // none are currently on the CPU
21
       if (!batch)
22
           index = buf_tail + 1;
23
24
           index = buf_tail;
       qsort(&process_buffer[index], buf_head - index, sizeof(process_p),
25
           sort);
26 }
```

What follows next is implementations for the sorting algorithms.

First, we have sjf_scheduler this sorts based on cpu_remaining_burst.

Next, we have fcfs_scheduler which sorts based on arrival time.

```
int fcfs_scheduler(const void *a, const void *b)
{

process_p process_a = *(process_p *)a;
process_p process_b = *(process_p *)b;

return (process_a->arrival_time - process_b->arrival_time);
}
```

Finally, we have priority_scheduler which sorts based on priority. Note the final calculation is swapped. This is because we are sorting with the highest priority goes first and the lowest priority goes last.

```
int priority_scheduler(const void *a, const void *b)

{
    process_p process_a = *(process_p *)a;
    process_p process_b = *(process_p *)b;

    return (-process_a->priority + process_b->priority);
}
```

What comes next are some utility functions that just help with ease of use.

First, we have a function that takes in a buffer and removes a trailing newline.

```
void remove_newline(char *buffer)

int string_length = strlen(buffer);

if (buffer[string_length - 1] == '\n')

buffer[string_length - 1] = '\0';

}

buffer[string_length - 1] = '\0';

}
```

Next, we have a function that takes in time and returns the human-readable string version of it.

```
1 char *convert_time(time_t time)
2 {
3    return asctime(localtime(&time));
4 }
```

get_policy_string will return the human readable string of the current policy.

```
char *get_policy_string()

char *get_policy_string()

switch (policy)

case FCFS:
 return "FCFS";
```

```
8
       case SJF:
           return "SJF";
9
       case PRIORITY:
11
12
           return "Priority";
13
14
       default:
15
           return "Unknown";
       }
16
17 }
```

submit_job will print out useful information for the user when submitting a job. This includes the name of the job, the number of jobs in the queue, the expected waiting time, and the scheduling policy.

Performance Metrics

Note: for all performance evaluation I used microbatch.out which is a sample program that simply sleeps for n seconds, while n is provided by the user.

Instant Arrival

With this benchmark, we show how the scheduling algorithms perform when all the jobs arrive at the same time.

First Come First Served, 5 Jobs, Arrival Time 0, Priority Range 0-5, CPU Burst Range 0-10

```
1 > [? for menu]: test bench1 fcfs 5 0 5 0 10
 2 Benchmark is running please wait...
 4 === Reporting Metrics for FCFS ===
 5
6 Metrics for job ./microbatch.out:
7 CPU Burst: 4 seconds
8 Interruptions: 0 times
9 Priority: 1
10 Arrival Time: Sun Mar 8 20:06:21 2020
11 First Time on CPU: Sun Mar 8 20:06:21 2020
9
11
       Finish Time: Sun Mar 8
Turnaround Time: 4 seconds
Waiting Time: 0 seconds
Response Time: 0 seconds
12
                                     Sun Mar 8 20:06:25 2020
13
14
16
17 Metrics for job ./microbatch.out:
       CPU Burst: 5 seconds
Interruptions: 0 times
Priority: 6
Arrival Time: Sun Mar 8 20:06:21 2020
18
     CPU Burst:
19
20
21
        First Time on CPU: Sun Mar 8 20:06:25 2020
22
       Finish Time:

Turnaround Time:

Waiting Time:

Response Time:

Sun Mar 8 20:06:30 2020

9 seconds

4 seconds

4 seconds
23
24
25
26
27
28 Metrics for job ./microbatch.out:
        CPU Burst: 6 seconds
Interruptions: 0 times
29 CPU Burst:
       Priority: 2
Arrival Time: Sun Mar 8 20:06:21 2020
32
33
          First Time on CPU: Sun Mar 8 20:06:30 2020
       Finish Time: Sun Mar 8 20:06:36 2020
34
```

```
Turnaround Time: 15 seconds
       Waiting Time:
                            9 seconds
37
       Response Time:
                            9 seconds
38
39 Metrics for job ./microbatch.out:
40
       CPU Burst:
                            0 seconds
41
       Interruptions:
                            0 times
42
       Priority:
                            5
       Arrival Time:
43
                          Sun Mar 8 20:06:21 2020
       First Time on CPU:
44
                            Sun Mar 8 20:06:36 2020
45
       Finish Time:
                            Sun Mar 8 20:06:36 2020
46
       Turnaround Time:
                            15 seconds
47
       Waiting Time:
                            15 seconds
48
       Response Time:
                            15 seconds
49
50 Metrics for job ./microbatch.out:
51
       CPU Burst:
                            8 seconds
52
       Interruptions:
                            0 times
53
       Priority:
                            3
       Arrival Time: Sun Mar 8 20:06:21 2020
54
       First Time on CPU: Sun Mar 8 20:06:36 2020
55
56
       Finish Time:
                            Sun Mar 8 20:06:44 2020
57
       Turnaround Time:
                          23 seconds
58
       Waiting Time:
                           15 seconds
59
                            15 seconds
       Response Time:
61 Overall Metrics for Batch:
62
       Total Number of Jobs Completed: 5
       Total Number of Jobs Submitted: 5
63
                                   13.200 seconds
64
       Average Turnaround Time:
65
       Average Waiting Time:
                                      8.600 seconds
       Average Response Time:
                                     8.600 seconds
                                      4.600 seconds
67
       Average CPU Burst:
       Total CPU Burst:
                                       23 seconds
68
69
       Throughput:
                                       0.076 No./second
       Max Turnaround Time:
                                       23 seconds
       Min Turnaround Time:
71
                                       4 seconds
72
       Max Waiting Time:
                                       15 seconds
74
       Min Waiting Time:
                                       0 seconds
75
76
       Max Response Time:
                                       15 seconds
77
       Min Response Time:
                                       0 seconds
78
79
       Max CPU Burst:
                                       8 seconds
       Min CPU Burst:
                                       0 seconds
```

Shortest Job First, 5 Jobs, Arrival Time 0, Priority Range 0-5, CPU Burst Range 0-10

```
1 > [? for menu]: test bench2 sjf 5 0 5 0 10
2 Benchmark is running please wait...
3
4 === Reporting Metrics for SJF ===
5
6 Metrics for job ./microbatch.out:
      CPU Burst:
                          0 seconds
8
       Interruptions:
                           0 times
     Priority: 5
Arrival Time: Sun Mar 8 20:10:20 2020
9
10
      First Time on CPU: Sun Mar 8 20:10:20 2020
      Finish Time.
Turnaround Time: 0 seconds
0 seconds
12
      Finish Time: Sun Mar 8 20:10:20 2020
13
     Waiting Time:
14
      Response Time:
                          0 seconds
16
17 Metrics for job ./microbatch.out:
18 CPU Burst: 3 seconds
19
       Interruptions: 0 times
20
       Priority:
                          1
      Arrival Time: Sun Mar 8 20:10:20 2020 First Time on CPU: Sun Mar 8 20:10:20 2020
21
      Finish Time: Sun Mar 8
Turnaround Time: 3 seconds
23
                           Sun Mar 8 20:10:23 2020
24
      Waiting Time:
25
                          0 seconds
26
     Response Time:
                          0 seconds
27
28 Metrics for job ./microbatch.out:
                   5 seconds
29
    CPU Burst:
                          0 times
       Interruptions:
                      6
31
       Priority:
       Arrival Time: Sun Mar 8 20:10:20 2020
32
       First Time on CPU: Sun Mar 8 20:10:23 2020
34
      Finish Time: Sun Mar 8 20:10:28 2020
      Turnaround Time: 8 seconds
35
     Waiting Time:
                          3 seconds
       Response Time: 3 seconds
37
38
39 Metrics for job ./microbatch.out:
                   6 seconds
40 CPU Burst:
41
       Interruptions:
                          0 times
       Priority: 2
Arrival Time: Sun Mar 8 20:10:20 2020
42
43
       First Time on CPU: Sun Mar 8 20:10:28 2020
44
       Finish Time: Sun Mar 8 20:10:34 2020
45
       Turnaround Time: 14 seconds Waiting Time: 8 seconds
46
       Waiting Time: 8 seconds
Response Time: 8 seconds
47
48
49
```

```
50 Metrics for job ./microbatch.out:
51
       CPU Burst:
                           8 seconds
                          0 times
52
       Interruptions:
53
       Priority:
                           3
       Arrival Time: Sun Mar 8 20:10:20 2020
54
       First Time on CPU: Sun Mar 8 20:10:34 2020
55
       Finish Time: Sun Mar 8 20:10:42 2020
57
       Turnaround Time:
                          22 seconds
58
       Waiting Time:
                         14 seconds
59
                          14 seconds
       Response Time:
60
61 Overall Metrics for Batch:
62
       Total Number of Jobs Completed: 5
       Total Number of Jobs Submitted: 5
63
64
       Average Turnaround Time: 9.400 seconds
       Average Waiting Time:
                                     5.000 seconds
65
                                  5.000 seconds
66
       Average Response Time:
       Average CPU Burst:
                                     4.400 seconds
67
68
       Total CPU Burst:
                                      22 seconds
                                    0.106 No./second
69
       Throughput:
       Max Turnaround Time:
70
                                    22 seconds
71
       Min Turnaround Time:
                                     0 seconds
72
                                     14 seconds
73
       Max Waiting Time:
74
       Min Waiting Time:
                                      0 seconds
75
       Max Response Time:
                                      14 seconds
       Min Response Time:
                                      0 seconds
77
78
       Max CPU Burst:
79
                                      8 seconds
80
       Min CPU Burst:
                                      0 seconds
```

Priority Based, 5 Jobs, Arrival Time 0, Priority Range 0-5, CPU Burst Range 0-10

```
1 > [? for menu]: test bench3 priority 5 0 5 0 10
2 Benchmark is running please wait...
3
4 === Reporting Metrics for Priority ===
5
6 Metrics for job ./microbatch.out:
      CPU Burst:
                           5 seconds
8
       Interruptions:
                           0 times
9
       Priority:
                           6
      Arrival Time: 5 Sun Mar 8 20:13:07 2020
10
      First Time on CPU: Sun Mar 8 20:13:07 2020
12
      Finish Time: Sun Mar 8 20:13:12 2020
      Turnaround Time:
                         5 seconds
13
                         0 seconds
     Waiting Time:
14
      Response Time:
                          0 seconds
16
17 Metrics for job ./microbatch.out:
18
    CPU Burst: 0 seconds
19
       Interruptions: 0 times
20
       Priority:
                          5
      Arrival Time: Sun Mar 8 20:13:07 2020 First Time on CPU: Sun Mar 8 20:13:12 2020
21
      Finish Time: Sun Mar & Turnaround Time: 5 seconds
23
                           Sun Mar 8 20:13:12 2020
24
      Waiting Time:
25
                          5 seconds
26
     Response Time:
                          5 seconds
27
28 Metrics for job ./microbatch.out:
                   8 seconds
29
    CPU Burst:
      Interruptions:
                          0 times
                          3
31
      Priority:
32
       Arrival Time: Sun Mar 8 20:13:07 2020
       First Time on CPU: Sun Mar 8 20:13:12 2020
34
      Finish Time: Sun Mar 8 20:13:20 2020
35
      Turnaround Time: 13 seconds
      Waiting Time:
                          5 seconds
       Response Time: 5 seconds
37
38
39 Metrics for job ./microbatch.out:
                   6 seconds
40 CPU Burst:
41
       Interruptions:
                          0 times
      Priority: 2
Arrival Time: Sun Mar 8 20:13:07 2020
42
43
       First Time on CPU: Sun Mar 8 20:13:20 2020
44
                          Sun Mar 8 20:13:26 2020
45
       Finish Time:
      Turnaround Time: 19 seconds Waiting Time: 13 seconds
46
      Waiting Time: 13 seconds
Response Time: 13 seconds
47
48
49
```

```
50 Metrics for job ./microbatch.out:
51
       CPU Burst:
                           3 seconds
       Interruptions:
                          0 times
52
53
       Priority:
                           1
       Arrival Time: Sun Mar 8 20:13:07 2020
54
       First Time on CPU: Sun Mar 8 20:13:26 2020
55
       Finish Time: Sun Mar 8 20:13:29 2020
57
      Turnaround Time:
                         22 seconds
58
       Waiting Time:
                         19 seconds
59
                          19 seconds
       Response Time:
60
61 Overall Metrics for Batch:
62
       Total Number of Jobs Completed: 5
       Total Number of Jobs Submitted: 5
63
64
       Average Turnaround Time: 12.800 seconds
       Average Waiting Time:
                                    8.400 seconds
65
                                  8.400 seconds
66
       Average Response Time:
       Average CPU Burst:
                                     4.400 seconds
67
68
       Total CPU Burst:
                                     22 seconds
                                    0.078 No./second
69
       Throughput:
       Max Turnaround Time:
70
                                    22 seconds
                                    5 seconds
71
       Min Turnaround Time:
72
73
                                     19 seconds
       Max Waiting Time:
74
       Min Waiting Time:
                                     0 seconds
75
       Max Response Time:
                                      19 seconds
       Min Response Time:
                                      0 seconds
77
78
       Max CPU Burst:
79
                                      8 seconds
80
       Min CPU Burst:
                                      0 seconds
```

Two Second Arrival

Here we show how the scheduling algorithms differ when they do not arrive at the same time.

First Come First Served, 5 Jobs, Arrival Time 2, Priority Range 0-5, CPU Burst Range 0-10

```
1 > [? for menu]: test bench1 fcfs 5 2 5 0 10
 2 Benchmark is running please wait...
   === Reporting Metrics for FCFS ===
 5
 6 Metrics for job ./microbatch.out:
 7 CPU Burst: 3 seconds
8
        Interruptions:
                                 0 times
       Priority: 1
Arrival Time: Sun Mar 8 20:14:50 2020
9
10
      First Time on CPU: Sun Mar 8 20:14:50 2020
Finish Time: Sun Mar 8 20:14:53 2020
Turnaround Time: 3 seconds
Waiting Time: 0 seconds
Response Time: 0 seconds
11
12
13
14
16
17 Metrics for job ./microbatch.out:
        CPU Burst: 5 seconds
Interruptions: 0 times
18
19
      Priority: 6
Arrival Time: Sun Mar 8 20:14:52 2020
21
       First Time on CPU: Sun Mar 8 20:14:53 2020
22
      Finish Time: Sun Mar 8 20:14:58 2020
Turnaround Time: 6 seconds
Waiting Time: 1 seconds
23
24
25
       Response Time:
                                 1 seconds
26
27
28 Metrics for job ./microbatch.out:
29 CPU Burst: 6 seconds
       Interruptions: 0 times
Priority: 2
Arrival Time: Sun Mar 8 20:14:54 2020
First Time on CPU: Sun Mar 8 20:14:58 2020
30
31
32
33
       Finish Time:

Turnaround Time:

Waiting Time:

Response Time:

Sun Mar 8

10 seconds

4 seconds

4 seconds
                                 Sun Mar 8 20:15:04 2020
34
37
38
39 Metrics for job ./microbatch.out:
40
        CPU Burst:
                         0 seconds
         Interruptions:
                                 0 times
41
42
                                 5
         Priority:
43
         Arrival Time: Sun Mar 8 20:14:56 2020
```

```
First Time on CPU: Sun Mar 8 20:15:04 2020
45
       Finish Time:
                           Sun Mar 8 20:15:04 2020
                          8 seconds
       Turnaround Time:
46
47
       Waiting Time:
                            8 seconds
                           8 seconds
48
       Response Time:
49
50 Metrics for job ./microbatch.out:
                       8 seconds
51
       CPU Burst:
       Interruptions:
52
                           0 times
       Priority: 3
Arrival Time: Sun Mar 8 20:14:58 2020
53
54
       First Time on CPU: Sun Mar 8 20:15:04 2020
55
       Finish Time: Sun Mar 8 20:15:12 2020
Turnaround Time: 14 seconds
Waiting Time: 6 seconds
56
57
58
59
                           6 seconds
       Response Time:
61 Overall Metrics for Batch:
       Total Number of Jobs Completed: 5
62
       Total Number of Jobs Submitted: 5
63
       Average Turnaround Time: 8.200 seconds
64
                                      3.800 seconds
65
       Average Waiting Time:
66
       Average Response Time:
                                      3.800 seconds
       Average CPU Burst:
67
                                      4.400 seconds
       Total CPU Burst:
                                      22 seconds
68
       Throughput:
                                       0.122 No./second
69
                                     14 seconds
70
       Max Turnaround Time:
       Min Turnaround Time:
                                       3 seconds
71
72
73
       Max Waiting Time:
                                      8 seconds
74
       Min Waiting Time:
                                      0 seconds
75
76
       Max Response Time:
                                       8 seconds
       Min Response Time:
77
                                       0 seconds
78
79
       Max CPU Burst:
                                      8 seconds
       Min CPU Burst:
80
                                       0 seconds
```

Shortest Job First, 5 Jobs, Arrival Time 2, Priority Range 0-5, CPU Burst Range 0-10

```
1 > [? for menu]: test bench2 sjf 5 2 5 0 10
2 Benchmark is running please wait...
3
4 === Reporting Metrics for SJF ===
5
6 Metrics for job ./microbatch.out:
          CPU Burst:
7
                              3 seconds
8
          Interruptions:
                              0 times
          Priority: 1
Arrival Time: Mon Mar 9 11:33:48 2020
9
10
          First Time on CPU: Mon Mar 9 11:33:48 2020
12
          Finish Time: Mon Mar 9 11:33:51 2020
13
          Turnaround Time:
                             3 seconds
                             0 seconds
14
          Waiting Time:
          Response Time:
                             0 seconds
16
17 Metrics for job ./microbatch.out:
         CPU Burst: 5 seconds
18
19
          Interruptions: 0 times
20
          Priority:
                             6
          Arrival Time: Mon Mar 9 11:33:50 2020 First Time on CPU: Mon Mar 9 11:33:51 2020
21
23
          Finish Time:
                              Mon Mar 9 11:33:56 2020
          Turnaround Time: Mon Mar S
24
25
          Waiting Time:
                              1 seconds
26
          Response Time:
                             1 seconds
27
28 Metrics for job ./microbatch.out:
          CPU Burst: 0 seconds
29
          Interruptions:
                             0 times
31
          Priority:
                              5
32
          Arrival Time:
                             Mon Mar 9 11:33:54 2020
          First Time on CPU: Mon Mar 9 11:33:56 2020
          Finish Time: Mon Mar 9 11:33:56 2020
34
35
          Turnaround Time: 2 seconds
          Waiting Time:
                              2 seconds
          Response Time: 2 seconds
37
38
39 Metrics for job ./microbatch.out:
40
          CPU Burst:
                              6 seconds
41
          Interruptions:
                             0 times
42
          Priority:
          Priority: 2
Arrival Time: Mon Mar 9 11:33:52 2020
43
          First Time on CPU: Mon Mar 9 11:33:56 2020
44
                             Mon Mar 9 11:34:02 2020
45
          Finish Time:
          Turnaround Time:
Waiting Time:
Response Time:
46
                              10 seconds
                              4 seconds
47
48
                              4 seconds
49
```

```
50 Metrics for job ./microbatch.out:
            CPU Burst:
51
                                  8 seconds
            Interruptions:
52
                                0 times
53
            Priority:
                                 3
           Arrival Time: Mon Mar 9 11:33:56 2020
54
            First Time on CPU: Mon Mar 9 11:34:02 2020
55
           Finish Time: Mon Mar 9 11:34:10 2020
Turnaround Time: 14 seconds
Waiting Time: 6 seconds
57
58
            Waiting Time:
            Response Time:
59
                                 6 seconds
60
61 Overall Metrics for Batch:
62
           Total Number of Jobs Completed: 5
           Total Number of Jobs Submitted: 5
63
           Average Turnaround Time: 7.000 seconds
Average Waiting Time: 2.600 seconds
64
           Average Waiting Time:
65
                                         2.600 seconds
66
            Average Response Time:
            Average CPU Burst:
                                            4.400 seconds
67
            Total CPU Burst:
68
                                             22 seconds
                                            0.143 No./second
69
            Throughput:
70
            Max Turnaround Time:
                                            14 seconds
71
            Min Turnaround Time:
                                            2 seconds
72
73
            Max Waiting Time:
                                             6 seconds
74
            Min Waiting Time:
                                             0 seconds
75
            Max Response Time:
                                             6 seconds
            Min Response Time:
                                             0 seconds
77
78
            Max CPU Burst:
79
                                             8 seconds
80
            Min CPU Burst:
                                             0 seconds
```

Priority Based, 5 Jobs, Arrival Time 2, Priority Range 0-5, CPU Burst Range 0-10

```
1 > [? for menu]: test bench3 priority 5 2 5 0 10
2 Benchmark is running please wait...
3
4 === Reporting Metrics for Priority ===
5
6 Metrics for job ./microbatch.out:
          CPU Burst:
7
                              3 seconds
8
          Interruptions:
                              0 times
          Priority: 1
Arrival Time: Mon Mar 9 11:34:57 2020
9
10
          First Time on CPU: Mon Mar 9 11:34:57 2020
12
          Finish Time: Mon Mar 9 11:35:00 2020
13
          Turnaround Time:
                             3 seconds
                             0 seconds
14
          Waiting Time:
          Response Time:
                             0 seconds
16
17 Metrics for job ./microbatch.out:
         CPU Burst: 5 seconds
18
19
          Interruptions: 0 times
20
         Priority:
                             6
          Arrival Time: Mon Mar 9 11:34:59 2020 First Time on CPU: Mon Mar 9 11:35:00 2020
21
                              Mon Mar 9 11:35:05 2020
23
          Finish Time:
          Turnaround Time: Mon Mar S
24
25
          Waiting Time:
                              1 seconds
26
         Response Time:
                             1 seconds
27
28 Metrics for job ./microbatch.out:
          CPU Burst: 0 seconds
29
          Interruptions:
                             0 times
31
          Priority:
                              5
32
          Arrival Time:
                             Mon Mar 9 11:35:03 2020
          First Time on CPU: Mon Mar 9 11:35:05 2020
          Finish Time: Mon Mar 9 11:35:05 2020
34
35
          Turnaround Time: 2 seconds
          Waiting Time:
                             2 seconds
          Response Time: 2 seconds
37
38
39 Metrics for job ./microbatch.out:
40
          CPU Burst:
                              8 seconds
41
          Interruptions:
                             0 times
42
          Priority:
                             3
          Arrival Time: Mon Mar 9 11:35:05 2020
43
          First Time on CPU: Mon Mar 9 11:35:05 2020
44
                             Mon Mar 9 11:35:13 2020
45
          Finish Time:
          Turnaround Time: 8 seconds Waiting Time: 0 seconds
46
          Waiting Time:
Response Time:
47
48
                              0 seconds
49
```

```
50 Metrics for job ./microbatch.out:
51
            CPU Burst:
                                  6 seconds
            Interruptions:
52
                                  0 times
53
            Priority:
                                  2
            Arrival Time: Mon Mar 9 11:35:01 2020
54
            First Time on CPU: Mon Mar 9 11:35:13 2020
55
           Finish Time: Mon Mar 9 11:35:19 2020
Turnaround Time: 18 seconds
Waiting Time: 12 seconds
57
58
            Response Time:
                                  12 seconds
59
60
61 Overall Metrics for Batch:
62
            Total Number of Jobs Completed: 5
            Total Number of Jobs Submitted: 5
63
            Average Turnaround Time: 7.400 seconds
Average Waiting Time: 3.000 seconds
64
            Average Waiting Time:
65
                                          3.000 seconds
4.400 seconds
66
            Average Response Time:
            Average CPU Burst:
67
            Total CPU Burst:
68
                                              22 seconds
                                              0.135 No./second
69
            Throughput:
            Max Turnaround Time:
70
                                             18 seconds
71
            Min Turnaround Time:
                                             2 seconds
72
                                              12 seconds
73
            Max Waiting Time:
74
            Min Waiting Time:
                                              0 seconds
75
            Max Response Time:
                                              12 seconds
            Min Response Time:
                                              0 seconds
77
78
            Max CPU Burst:
79
                                              8 seconds
80
            Min CPU Burst:
                                              0 seconds
```

Max Burst < Arrival Time

Whenever the max CPU burst is less than the arrival time the schedulers will all act like FCFS. This is because as each job enters the queue, they finish before the next arrives.

First Come First Served, 5 Jobs, Arrival Time 5, Priority Range 0-5, CPU Burst Range 0-3

```
> [? for menu]: test bench1 fcfs 5 5 5 0 3
2 Benchmark is running please wait...
3
4 === Reporting Metrics for FCFS ===
6 Metrics for job ./microbatch.out:
    CPU Burst: 3 seconds
         Interruptions: 0 times
Priority: 1
Arrival Time: Mon Mar 9 17:51:14 2020
8
9
10
          First Time on CPU: Mon Mar 9 17:51:14 2020
         Finish Time: Mon Mar 9 17:51:17 2020
Turnaround Time: 3 seconds
Waiting Time: 0 seconds
12
13
         Waiting Time: 0 seconds
Response Time: 0 seconds
14
15
16
17 Metrics for job ./microbatch.out:
18 CPU Burst: 0 seconds
19 Interruptions: 0 times
         Priority: 6
Arrival Time: Mon Mar 9 17:51:19 2020
20
21
          First Time on CPU: Mon Mar 9 17:51:19 2020
22
                              Mon Mar 9 17:51:19 2020
23
          Finish Time:
          Turnaround Time: 0 seconds Waiting Time: 0 seconds
24
25
          Waiting Time:
          Response Time: 0 seconds
26
27
28 Metrics for job ./microbatch.out:
29
          CPU Burst: 1 seconds
          Interruptions:
                              0 times
         Priority:
                               2
31
         Priority: 2
Arrival Time: Mon Mar 9 17:51:24 2020
32
          First Time on CPU: Mon Mar 9 17:51:24 2020
33
34
          Finish Time: Mon Mar 9 17:51:25 2020
          Turnaround Time:
                              1 seconds
          Waiting Time:
                              0 seconds
37
          Response Time:
                              0 seconds
38
39 Metrics for job ./microbatch.out:
40
          CPU Burst: 1 seconds
41
           Interruptions:
                              0 times
42
                               5
           Priority:
```

```
Arrival Time: Mon Mar 9 17:51:29 2020
            First Time on CPU: Mon Mar 9 17:51:29 2020
44
            Finish Time: Mon mar Turnaround Time: 1 seconds
                                   Mon Mar 9 17:51:30 2020
45
46
            Waiting Time:
47
                                   0 seconds
48
            Response Time:
                                   0 seconds
49
50 Metrics for job ./microbatch.out:
                             1 seconds
51
           CPU Burst:
            Interruptions:
                                 0 times
52
53
            Priority:
                                  3
            Arrival Time: Mon Mar 9 17:51:34 2020
54
            First Time on CPU: Mon Mar 9 17:51:34 2020
55
            Finish Time: Mon Mar 9 17:51:35 2020
Turnaround Time: 1 seconds
Waiting Time: 0 seconds
57
            Waiting Time:
58
59
            Response Time: 0 seconds
60
61 Overall Metrics for Batch:
62
            Total Number of Jobs Completed: 5
            Total Number of Jobs Submitted: 5
63
           Average Turnaround Time: 1.200 seconds
Average Waiting Time: 0.000 seconds
Average Response Time: 0.000 seconds
64
65
66
            Average CPU Burst:
                                              1.200 seconds
67
                                           6 seconds
0.833 No./second
1 seconds
            Total CPU Burst:
68
69
            Throughput:
70
            Max Turnaround Time:
                                              0 seconds
            Min Turnaround Time:
71
72
73
            Max Waiting Time:
                                               0 seconds
74
            Min Waiting Time:
                                               0 seconds
75
            Max Response Time:
                                               0 seconds
                                               0 seconds
            Min Response Time:
77
78
            Max CPU Burst:
79
                                               3 seconds
            Min CPU Burst:
                                               0 seconds
```

Shortest Job First, 5 Jobs, Arrival Time 5, Priority Range 0-5, CPU Burst Range 0-3

```
1 > [? for menu]: test bench2 sjf 5 5 5 0 3
2 Benchmark is running please wait...
3
4 === Reporting Metrics for SJF ===
5
6 Metrics for job ./microbatch.out:
      CPU Burst:
7
                          3 seconds
8
       Interruptions:
                           0 times
9
     Priority: 1
Arrival Time: Mon Mar 9 16:56:21 2020
       Priority:
                           1
10
      First Time on CPU: Mon Mar 9 16:56:21 2020
      Finish Time.
Turnaround Time: 3 seconds
0 seconds
12
      Finish Time: Mon Mar 9 16:56:24 2020
13
                          3 seconds
     Waiting Time:
14
      Response Time:
                          0 seconds
16
17 Metrics for job ./microbatch.out:
18 CPU Burst: 0 seconds
       Interruptions: 0 times
19
20
       Priority:
                          6
      Arrival Time: Mon Mar 9 16:56:26 2020 First Time on CPU: Mon Mar 9 16:56:26 2020
21
      Finish Time: Mon Mar 9
Turnaround Time: 0 seconds
23
                           Mon Mar 9 16:56:26 2020
24
      Waiting Time:
25
                          0 seconds
26
     Response Time:
                          0 seconds
27
28 Metrics for job ./microbatch.out:
                   1 seconds
29
    CPU Burst:
       Interruptions:
                          0 times
                       2
31
       Priority:
       Arrival Time: Mon Mar 9 16:56:31 2020
32
       First Time on CPU: Mon Mar 9 16:56:31 2020
      Finish Time: Mon Mar 9 16:56:32 2020
34
      Turnaround Time: 1 seconds
35
     Waiting Time:
                          0 seconds
       Waiting Time: 0 seconds
Response Time: 0 seconds
37
38
39 Metrics for job ./microbatch.out:
                   1 seconds
40 CPU Burst:
41
       Interruptions:
                          0 times
       Priority: 5
Arrival Time: Mon Mar 9 16:56:36 2020
42
43
       First Time on CPU: Mon Mar 9 16:56:36 2020
44
                          Mon Mar 9 16:56:37 2020
45
       Turnaround Time: 1 seconds

Time: 0 seconds
       Finish Time:
46
                          1 seconds
       Waiting Time: 0 seconds
Response Time: 0 seconds
47
48
49
```

```
50 Metrics for job ./microbatch.out:
51
       Interruptions:
                             1 seconds
                            0 times
52
53
       Priority:
                             3
       Arrival Time: Mon Mar 9 16:56:41 2020
54
       First Time on CPU: Mon Mar 9 16:56:41 2020
55
       Finish Time: Mon Mar 9 16:56:42 2020 Turnaround Time: 1 seconds
57
58
       Waiting Time:
                           0 seconds
59
       Response Time:
                            0 seconds
60
61 Overall Metrics for Batch:
62
       Total Number of Jobs Completed: 5
       Total Number of Jobs Submitted: 5
63
       Average Turnaround Time: 1.200 seconds
Average Waiting Time: 0.000 seconds
64
       Average Waiting Time:
65
                                    0.000 seconds
       Average Response Time:
66
       Average CPU Burst:
                                       1.200 seconds
67
68
       Total CPU Burst:
                                       6 seconds
                                       0.833 No./second
69
       Throughput:
                                      1 seconds
0 seconds
       Max Turnaround Time:
70
71
       Min Turnaround Time:
72
73
                                       0 seconds
       Max Waiting Time:
74
                                        0 seconds
       Min Waiting Time:
75
       Max Response Time:
                                        0 seconds
       Min Response Time:
                                        0 seconds
77
78
       Max CPU Burst:
79
                                        3 seconds
80
       Min CPU Burst:
                                        0 seconds
```

Priority Based, 5 Jobs, Arrival Time 0, Priority Range 0-5, CPU Burst Range 0-3

```
1 > [? for menu]: test bench3 priority 5 5 5 0 3
2 Benchmark is running please wait...
3
4 === Reporting Metrics for Priority ===
5
6 Metrics for job ./microbatch.out:
      CPU Burst:
                           3 seconds
8
       Interruptions:
                           0 times
9
      Priority: 1
Arrival Time: Mon Mar 9 16:58:07 2020
       Priority:
                           1
10
      First Time on CPU: Mon Mar 9 16:58:07 2020
12
      Finish Time: Mon Mar 9 16:58:10 2020
      Turnaround Time:
13
                          3 seconds
                          0 seconds
     Waiting Time:
14
      Response Time:
                          0 seconds
16
17 Metrics for job ./microbatch.out:
18 CPU Burst: 0 seconds
       Interruptions: 0 times
19
20
       Priority:
                          6
      Arrival Time: Mon Mar 9 16:58:12 2020 First Time on CPU: Mon Mar 9 16:58:12 2020
21
      Finish Time: Mon Mar 9
Turnaround Time: 0 seconds
23
                           Mon Mar 9 16:58:12 2020
24
       Waiting Time:
25
                          0 seconds
26
     Response Time:
                          0 seconds
27
28 Metrics for job ./microbatch.out:
                   1 seconds
29
    CPU Burst:
       Interruptions:
                          0 times
                          2
31
       Priority:
       Arrival Time: Mon Mar 9 16:58:17 2020
32
       First Time on CPU: Mon Mar 9 16:58:17 2020
      Finish Time: Mon Mar 9 16:58:18 2020
34
      Turnaround Time: 1 seconds
35
     Waiting Time:
                          0 seconds
       Waiting Time: 0 seconds
Response Time: 0 seconds
37
38
39 Metrics for job ./microbatch.out:
                   1 seconds
40 CPU Burst:
       Interruptions:
41
                          0 times
       Priority: 5
Arrival Time: Mon Mar 9 16:58:22 2020
42
43
       First Time on CPU: Mon Mar 9 16:58:22 2020
44
                          Mon Mar 9 16:58:23 2020
45
      Turnaround Time: 1 seconds

Time: 0 seconds
       Finish Time:
46
                          1 seconds
       Waiting Time: 0 seconds
Response Time: 0 seconds
47
48
49
```

```
50 Metrics for job ./microbatch.out:
51
       Interruptions:
                             1 seconds
                            0 times
52
53
       Priority:
                             3
       Arrival Time: Mon Mar 9 16:58:27 2020
54
       First Time on CPU: Mon Mar 9 16:58:27 2020
55
       Finish Time: Mon Mar 9 16:58:28 2020
Turnaround Time: 1 seconds
57
58
       Waiting Time:
                           0 seconds
59
       Response Time:
                            0 seconds
60
61 Overall Metrics for Batch:
62
       Total Number of Jobs Completed: 5
       Total Number of Jobs Submitted: 5
63
       Average Turnaround Time: 1.200 seconds
Average Waiting Time: 0.000 seconds
64
       Average Waiting Time:
65
                                    0.000 seconds
       Average Response Time:
66
       Average CPU Burst:
                                       1.200 seconds
67
68
       Total CPU Burst:
                                       6 seconds
                                       0.833 No./second
69
       Throughput:
                                      1 seconds
0 seconds
       Max Turnaround Time:
70
71
       Min Turnaround Time:
72
73
                                       0 seconds
       Max Waiting Time:
74
                                        0 seconds
       Min Waiting Time:
75
       Max Response Time:
                                         0 seconds
       Min Response Time:
                                         0 seconds
77
78
       Max CPU Burst:
79
                                         3 seconds
80
       Min CPU Burst:
                                         0 seconds
```

Number of Jobs > Queue Size

Whenever we have more jobs than the queue can fit they will have to wait to be loaded. Below are metrics showing that in action.

First Come First Served, 15 Jobs, Arrival Time 0, Priority Range 0-5, CPU Burst Range 0-3

```
> [? for menu]: test bench1 fcfs 15 0 5 0 3
2 Benchmark is running please wait...
3
4 === Reporting Metrics for FCFS ===
6 Metrics for job ./microbatch.out:
    CPU Burst: 2 seconds
          Interruptions: 0 times
Priority: 1
Arrival Time: Mon Mar 9 17:55:32 2020
8
9
10
          First Time on CPU: Mon Mar 9 17:55:32 2020
          Finish Time: Mon Mar 9 17:55:34 2020
Turnaround Time: 2 seconds
Waiting Time: 0 seconds
12
13
          Waiting Time: 0 seconds
Response Time: 0 seconds
14
15
16
17 Metrics for job ./microbatch.out:
18 CPU Burst: 0 seconds
19 Interruptions: 0 times
          Priority: 6
Arrival Time: Mon Mar 9 17:55:32 2020
20
21
          First Time on CPU: Mon Mar 9 17:55:34 2020 Finish Time: Mon Mar 9 17:55:34 2020
23
           Turnaround Time: 2 seconds
Waiting Time: 2 seconds
Response Time: 2 seconds
24
25
26
27
28 Metrics for job ./microbatch.out:
29
           CPU Burst: 1 seconds
           Interruptions:
                                0 times
30
          Priority: 2
Arrival Time: Mon Mar 9 17:55:32 2020
31
32
          First Time on CPU: Mon Mar 9 17:55:34 2020
33
34
           Finish Time: Mon Mar 9 17:55:35 2020
           Turnaround Time:
                                3 seconds
          Waiting Time:
                                2 seconds
37
           Response Time:
                                2 seconds
38
39 Metrics for job ./microbatch.out:
40
           CPU Burst: 1 seconds
41
           Interruptions:
                                0 times
42
                                 5
           Priority:
```

```
43
           Arrival Time: Mon Mar 9 17:55:32 2020
44
           First Time on CPU:
                               Mon Mar 9 17:55:35 2020
           Finish Time:
                               Mon Mar 9 17:55:36 2020
45
46
           Turnaround Time:
                               4 seconds
47
           Waiting Time:
                               3 seconds
48
           Response Time:
                               3 seconds
49
50 Metrics for job ./microbatch.out:
           CPU Burst:
51
                               1 seconds
52
           Interruptions:
                               0 times
53
           Priority:
                               3
54
           Arrival Time:
                               Mon Mar 9 17:55:32 2020
55
           First Time on CPU: Mon Mar 9 17:55:36 2020
           Finish Time:
                               Mon Mar 9 17:55:37 2020
57
           Turnaround Time:
                               5 seconds
58
           Waiting Time:
                               4 seconds
59
           Response Time:
                               4 seconds
60
61 Metrics for job ./microbatch.out:
62
           CPU Burst:
                               0 seconds
63
           Interruptions:
                               0 times
64
           Priority:
                               5
65
           Arrival Time:
                              Mon Mar 9 17:55:32 2020
66
           First Time on CPU: Mon Mar 9 17:55:37 2020
           Finish Time:
                               Mon Mar 9 17:55:37 2020
67
68
           Turnaround Time:
                               5 seconds
69
           Waiting Time:
                               5 seconds
70
           Response Time:
                               5 seconds
71
72 Metrics for job ./microbatch.out:
73
           CPU Burst:
                               0 seconds
74
           Interruptions:
                               0 times
75
           Priority:
                               3
           Arrival Time:
                               Mon Mar 9 17:55:32 2020
           First Time on CPU: Mon Mar 9 17:55:37 2020
77
           Finish Time:
78
                               Mon Mar 9 17:55:37 2020
79
           Turnaround Time:
                               5 seconds
           Waiting Time:
                               5 seconds
81
           Response Time:
                               5 seconds
82
83 Metrics for job ./microbatch.out:
84
           CPU Burst:
                               3 seconds
85
           Interruptions:
                               0 times
86
           Priority:
                               1
87
           Arrival Time:
                               Mon Mar 9 17:55:32 2020
           First Time on CPU:
                               Mon Mar 9 17:55:37 2020
88
89
           Finish Time:
                               Mon Mar 9 17:55:40 2020
           Turnaround Time:
                               8 seconds
           Waiting Time:
                               5 seconds
92
           Response Time:
                               5 seconds
```

```
94 Metrics for job ./microbatch.out:
            CPU Burst:
                                 0 seconds
                                 0 times
            Interruptions:
            Priority:
97
                                 1
98
            Arrival Time:
                                 Mon Mar 9 17:55:32 2020
99
            First Time on CPU:
                                 Mon Mar 9 17:55:40 2020
            Finish Time:
                                 Mon Mar 9 17:55:40 2020
            Turnaround Time:
                                 8 seconds
102
            Waiting Time:
                                 8 seconds
103
            Response Time:
                                 8 seconds
104
105 Metrics for job ./microbatch.out:
            CPU Burst:
                                 1 seconds
                                 0 times
107
            Interruptions:
108
            Priority:
                                 6
            Arrival Time:
109
                                 Mon Mar 9 17:55:32 2020
            First Time on CPU:
                                 Mon Mar 9 17:55:40 2020
            Finish Time:
                                 Mon Mar 9 17:55:41 2020
112
            Turnaround Time:
                                 9 seconds
113
            Waiting Time:
                                 8 seconds
114
            Response Time:
                                 8 seconds
116 Metrics for job ./microbatch.out:
117
            CPU Burst:
                                 1 seconds
118
            Interruptions:
                                 0 times
119
            Priority:
                                 1
            Arrival Time:
                                 Mon Mar 9 17:55:34 2020
                                 Mon Mar 9 17:55:41 2020
            First Time on CPU:
121
            Finish Time:
                                 Mon Mar 9 17:55:42 2020
123
            Turnaround Time:
                                 8 seconds
124
            Waiting Time:
                                 7 seconds
125
            Response Time:
                                 7 seconds
126
127 Metrics for job ./microbatch.out:
            CPU Burst:
128
                                 2 seconds
            Interruptions:
                                 0 times
129
            Priority:
                                 2
131
            Arrival Time:
                                 Mon Mar 9 17:55:34 2020
            First Time on CPU:
132
                                 Mon Mar 9 17:55:42 2020
                                 Mon Mar 9 17:55:44 2020
            Finish Time:
134
            Turnaround Time:
                                 10 seconds
135
            Waiting Time:
                                 8 seconds
            Response Time:
                                 8 seconds
138 Metrics for job ./microbatch.out:
139
            CPU Burst:
                                 1 seconds
140
                                 0 times
            Interruptions:
141
            Priority:
                                 4
142
            Arrival Time:
                                 Mon Mar
                                          9 17:55:35 2020
143
            First Time on CPU:
                                          9 17:55:44 2020
                                 Mon Mar
144
            Finish Time: Mon Mar 9 17:55:45 2020
```

```
Turnaround Time:
                                10 seconds
146
            Waiting Time:
                                 9 seconds
147
            Response Time:
                                 9 seconds
148
149 Metrics for job ./microbatch.out:
            CPU Burst:
                                 1 seconds
            Interruptions:
                                 0 times
152
            Priority:
                                 1
153
            Arrival Time:
                                 Mon Mar 9 17:55:36 2020
            First Time on CPU:
154
                                 Mon Mar 9 17:55:45 2020
            Finish Time:
                                 Mon Mar 9 17:55:46 2020
            Turnaround Time:
                                 10 seconds
            Waiting Time:
                                 9 seconds
157
158
            Response Time:
                                 9 seconds
160 Metrics for job ./microbatch.out:
            CPU Burst:
                                 3 seconds
162
            Interruptions:
                                 0 times
            Priority:
                                 4
            Arrival Time:
164
                                 Mon Mar 9 17:55:37 2020
            First Time on CPU: Mon Mar 9 17:55:46 2020
                                 Mon Mar 9 17:55:49 2020
166
            Finish Time:
167
            Turnaround Time:
                                 12 seconds
168
            Waiting Time:
                                 9 seconds
169
                                 9 seconds
            Response Time:
171 Overall Metrics for Batch:
            Total Number of Jobs Completed: 15
172
            Total Number of Jobs Submitted: 15
173
174
            Average Turnaround Time:
                                         6.733 seconds
175
            Average Waiting Time:
                                           5.600 seconds
176
            Average Response Time:
                                           5.600 seconds
                                           1.133 seconds
177
            Average CPU Burst:
            Total CPU Burst:
178
                                            17 seconds
                                            0.149 No./second
            Throughput:
179
            Max Turnaround Time:
                                            12 seconds
181
            Min Turnaround Time:
                                            2 seconds
182
            Max Waiting Time:
                                            9 seconds
184
            Min Waiting Time:
                                            0 seconds
            Max Response Time:
                                            9 seconds
187
            Min Response Time:
                                            0 seconds
188
            Max CPU Burst:
                                            3 seconds
            Min CPU Burst:
                                            0 seconds
```

Shortest Job First, 15 Jobs, Arrival Time 0, Priority Range 0-5, CPU Burst Range 0-3

```
1 > [? for menu]: test bench2 sjf 15 0 5 0 3
2 Benchmark is running please wait...
3
4 === Reporting Metrics for SJF ===
5
6 Metrics for job ./microbatch.out:
          CPU Burst:
7
                             0 seconds
8
          Interruptions:
                             0 times
          Priority: 3
Arrival Time: Mon Mar 9 17:57:17 2020
9
10
          First Time on CPU: Mon Mar 9 17:57:17 2020
12
         Finish Time: Mon Mar 9 17:57:17 2020
13
         Turnaround Time:
                            0 seconds
                            0 seconds
14
         Waiting Time:
         Response Time:
                            0 seconds
16
17 Metrics for job ./microbatch.out:
        CPU Burst: 0 seconds
18
19
          Interruptions: 0 times
20
         Priority:
                            5
          Arrival Time: Mon Mar 9 17:57:17 2020
21
          First Time on CPU: Mon Mar 9 17:57:17 2020
23
          Finish Time:
                             Mon Mar 9 17:57:17 2020
          Turnaround Time: Mon Mar S

O seconds
24
25
          Waiting Time:
                             0 seconds
26
         Response Time:
                             0 seconds
27
28 Metrics for job ./microbatch.out:
         CPU Burst: 0 seconds
29
          Interruptions:
                            0 times
31
          Priority:
                             1
32
          Arrival Time:
                            Mon Mar 9 17:57:17 2020
          First Time on CPU: Mon Mar 9 17:57:17 2020
         Finish Time: Mon Mar 9 17:57:17 2020
34
          Turnaround Time: 0 seconds
          Waiting Time:
                             0 seconds
37
          Response Time:
                            0 seconds
38
39 Metrics for job ./microbatch.out:
40
         CPU Burst:
                             0 seconds
41
          Interruptions:
                             0 times
42
         Priority:
                            6
          Arrival Time: Mon Mar 9 17:57:17 2020
43
          First Time on CPU: Mon Mar 9 17:57:17 2020
44
                            Mon Mar 9 17:57:17 2020
45
          Finish Time:
          Turnaround Time:
46
                            0 seconds
          Waiting Time:
Response Time:
47
                            0 seconds
48
                             0 seconds
49
```

```
50 Metrics for job ./microbatch.out:
51
           CPU Burst:
                                1 seconds
52
                                0 times
           Interruptions:
           Priority:
53
                                5
           Arrival Time:
54
                               Mon Mar 9 17:57:17 2020
55
           First Time on CPU:
                               Mon Mar 9 17:57:17 2020
           Finish Time:
                               Mon Mar 9 17:57:18 2020
57
           Turnaround Time:
                               1 seconds
58
           Waiting Time:
                                0 seconds
59
           Response Time:
                                0 seconds
60
61 Metrics for job ./microbatch.out:
           CPU Burst:
62
                               1 seconds
                                0 times
63
           Interruptions:
64
           Priority:
                               6
           Arrival Time:
65
                               Mon Mar 9 17:57:17 2020
66
           First Time on CPU:
                               Mon Mar 9 17:57:18 2020
           Finish Time:
                                Mon Mar 9 17:57:19 2020
67
68
           Turnaround Time:
                                2 seconds
69
           Waiting Time:
                                1 seconds
70
           Response Time:
                                1 seconds
71
72 Metrics for job ./microbatch.out:
73
           CPU Burst:
                               1 seconds
74
           Interruptions:
                                0 times
75
           Priority:
                                3
           Arrival Time:
                               Mon Mar 9 17:57:17 2020
           First Time on CPU:
                               Mon Mar 9 17:57:19 2020
77
78
           Finish Time:
                               Mon Mar 9 17:57:20 2020
79
           Turnaround Time:
                                3 seconds
80
           Waiting Time:
                                2 seconds
81
           Response Time:
                                2 seconds
82
83 Metrics for job ./microbatch.out:
           CPU Burst:
84
                                1 seconds
           Interruptions:
85
                                0 times
           Priority:
86
                                2
87
           Arrival Time:
                               Mon Mar 9 17:57:17 2020
88
           First Time on CPU: Mon Mar 9 17:57:20 2020
                               Mon Mar 9 17:57:21 2020
89
           Finish Time:
           Turnaround Time:
                                4 seconds
91
           Waiting Time:
                                3 seconds
92
           Response Time:
                                3 seconds
93
94 Metrics for job ./microbatch.out:
           CPU Burst:
                               2 seconds
           Interruptions:
                                0 times
97
           Priority:
                                1
98
           Arrival Time:
                                Mon Mar
                                        9 17:57:17 2020
                               Mon Mar 9 17:57:21 2020
99
           First Time on CPU:
           Finish Time: Mon Mar 9 17:57:23 2020
```

```
Turnaround Time: 6 seconds
102
           Waiting Time:
                                4 seconds
           Response Time:
                                4 seconds
104
105 Metrics for job ./microbatch.out:
           CPU Burst:
106
                                3 seconds
           Interruptions:
                                0 times
108
           Priority:
                                1
109
           Arrival Time:
                                Mon Mar 9 17:57:17 2020
110
           First Time on CPU:
                                Mon Mar 9 17:57:23 2020
111
           Finish Time:
                                Mon Mar 9 17:57:26 2020
112
           Turnaround Time:
                                9 seconds
                                6 seconds
113
           Waiting Time:
           Response Time:
114
                                6 seconds
116 Metrics for job ./microbatch.out:
117
           CPU Burst:
                                1 seconds
118
           Interruptions:
                                0 times
119
           Priority:
                                1
           Arrival Time:
                                Mon Mar 9 17:57:17 2020
           First Time on CPU: Mon Mar 9 17:57:26 2020
121
           Finish Time:
                                Mon Mar 9 17:57:27 2020
123
           Turnaround Time:
                                10 seconds
124
           Waiting Time:
                                9 seconds
125
                                9 seconds
           Response Time:
127 Metrics for job ./microbatch.out:
128
           CPU Burst:
                          2 seconds
129
           Interruptions:
                                0 times
           Priority:
                                2
131
           Arrival Time:
                                Mon Mar 9 17:57:17 2020
132
           First Time on CPU:
                                Mon Mar 9 17:57:27 2020
                                Mon Mar 9 17:57:29 2020
133
           Finish Time:
                                12 seconds
134
           Turnaround Time:
           Waiting Time:
                                10 seconds
           Response Time:
                                10 seconds
138 Metrics for job ./microbatch.out:
           CPU Burst:
                                1 seconds
                                0 times
140
           Interruptions:
141
           Priority:
                                4
142
           Arrival Time:
                                Mon Mar 9 17:57:17 2020
           First Time on CPU:
143
                                Mon Mar 9 17:57:29 2020
144
           Finish Time:
                                Mon Mar 9 17:57:30 2020
                                13 seconds
145
           Turnaround Time:
146
           Waiting Time:
                                12 seconds
147
           Response Time:
                                12 seconds
148
149 Metrics for job ./microbatch.out:
           CPU Burst:
                                1 seconds
           Interruptions:
                                0 times
```

```
152
            Priority:
153
            Arrival Time:
                                 Mon Mar 9 17:57:17 2020
            First Time 5..
Finish Time:
            First Time on CPU:
                                Mon Mar 9 17:57:30 2020
154
                                 Mon Mar 9 17:57:31 2020
                                 14 seconds
157
            Waiting Time:
                                 13 seconds
158
            Response Time:
                                13 seconds
159
160 Metrics for job ./microbatch.out:
           CPU Burst:
161
                           3 seconds
            Interruptions:
                                 0 times
163
            Priority:
                                4
164
            Arrival Time:
                               Mon Mar 9 17:57:18 2020
            First Time on CPU: Mon Mar 9 17:57:31 2020
            Finish Time:
                              Mon Mar 9 17:57:34 2020
            Turnaround Time:
                                16 seconds
167
168
            Waiting Time:
                                13 seconds
169
            Response Time:
                                13 seconds
170
171 Overall Metrics for Batch:
           Total Number of Jobs Completed: 15
172
173
            Total Number of Jobs Submitted: 15
174
            Average Turnaround Time:
                                          6.000 seconds
175
            Average Waiting Time:
                                           4.867 seconds
                                           4.867 seconds
176
            Average Response Time:
            Average CPU Burst:
177
                                            1.133 seconds
178
            Total CPU Burst:
                                            17 seconds
179
            Throughput:
                                            0.167 No./second
            Max Turnaround Time:
                                           16 seconds
            Min Turnaround Time:
                                            0 seconds
182
183
            Max Waiting Time:
                                            13 seconds
184
            Min Waiting Time:
                                            0 seconds
            Max Response Time:
                                            13 seconds
187
            Min Response Time:
                                            0 seconds
188
            Max CPU Burst:
                                            3 seconds
            Min CPU Burst:
                                            0 seconds
```

Priority Based, 15 Jobs, Arrival Time 0, Priority Range 0-5, CPU Burst Range 0-3

```
1 > [? for menu]: test bench3 priority 15 0 5 0 3
2 Benchmark is running please wait...
3
4 === Reporting Metrics for Priority ===
5
6 Metrics for job ./microbatch.out:
          CPU Burst:
7
                             2 seconds
8
          Interruptions:
                             0 times
          Priority: 6
Arrival Time: Mon Mar 9 17:58:12 2020
9
10
          First Time on CPU: Mon Mar 9 17:58:12 2020
12
         Finish Time: Mon Mar 9 17:58:14 2020
13
          Turnaround Time:
                            2 seconds
                            0 seconds
14
         Waiting Time:
         Response Time:
                             0 seconds
16
17 Metrics for job ./microbatch.out:
         CPU Burst: 0 seconds
18
                          0 times
19
          Interruptions:
20
         Priority:
                            6
          Arrival Time: Mon Mar 9 17:58:12 2020
21
          First Time on CPU: Mon Mar 9 17:58:14 2020
23
          Finish Time:
                             Mon Mar 9 17:58:14 2020
          Turnaround Time:
24
                             2 seconds
25
          Waiting Time:
                             2 seconds
26
         Response Time:
                             2 seconds
27
28 Metrics for job ./microbatch.out:
         CPU Burst: 0 seconds
29
          Interruptions:
                            0 times
31
          Priority:
                             5
32
          Arrival Time:
                            Mon Mar 9 17:58:12 2020
          First Time on CPU: Mon Mar 9 17:58:14 2020
         Finish Time: Mon Mar 9 17:58:14 2020
34
35
          Turnaround Time: 2 seconds
          Waiting Time:
                             2 seconds
          Response Time: 2 seconds
37
38
39 Metrics for job ./microbatch.out:
40
          CPU Burst:
                             1 seconds
41
          Interruptions:
                            0 times
42
         Priority:
                            5
          Arrival Time: Mon Mar 9 17:58:12 2020
43
          First Time on CPU: Mon Mar 9 17:58:14 2020
44
                            Mon Mar 9 17:58:15 2020
45
          Finish Time:
          Turnaround Time: 3 seconds
2 seconds
46
                             3 seconds
          Waiting Time:
Response Time:
47
48
                             2 seconds
49
```

```
50 Metrics for job ./microbatch.out:
51
           CPU Burst:
                                1 seconds
52
           Interruptions:
                                0 times
53
           Priority:
                                3
           Arrival Time:
54
                                Mon Mar 9 17:58:12 2020
55
           First Time on CPU:
                                Mon Mar 9 17:58:15 2020
           Finish Time:
                                Mon Mar 9 17:58:16 2020
57
           Turnaround Time:
                                4 seconds
                                3 seconds
58
           Waiting Time:
59
           Response Time:
                                3 seconds
60
61 Metrics for job ./microbatch.out:
           CPU Burst:
62
                                0 seconds
                                0 times
63
           Interruptions:
64
           Priority:
                                3
           Arrival Time:
65
                                Mon Mar 9 17:58:12 2020
66
           First Time on CPU:
                                Mon Mar 9 17:58:16 2020
           Finish Time:
                                Mon Mar 9 17:58:16 2020
67
68
           Turnaround Time:
                                4 seconds
69
           Waiting Time:
                                4 seconds
70
                                4 seconds
           Response Time:
71
72 Metrics for job ./microbatch.out:
73
           CPU Burst:
                               1 seconds
74
           Interruptions:
                                0 times
75
           Priority:
                                2
           Arrival Time:
                                Mon Mar 9 17:58:12 2020
           First Time on CPU:
                                Mon Mar 9 17:58:16 2020
77
78
           Finish Time:
                                Mon Mar 9 17:58:17 2020
79
           Turnaround Time:
                                5 seconds
80
           Waiting Time:
                                4 seconds
81
           Response Time:
                                4 seconds
82
83 Metrics for job ./microbatch.out:
           CPU Burst:
84
                                3 seconds
           Interruptions:
85
                                0 times
           Priority:
86
                                1
87
           Arrival Time:
                              Mon Mar 9 17:58:12 2020
88
           First Time on CPU: Mon Mar 9 17:58:17 2020
                                Mon Mar 9 17:58:20 2020
89
           Finish Time:
           Turnaround Time:
                                8 seconds
91
           Waiting Time:
                                5 seconds
92
           Response Time:
                                5 seconds
93
94 Metrics for job ./microbatch.out:
           CPU Burst:
                                0 seconds
                                0 times
           Interruptions:
97
           Priority:
                                1
98
           Arrival Time:
                                Mon Mar
                                         9 17:58:12 2020
                                Mon Mar 9 17:58:20 2020
99
           First Time on CPU:
           Finish Time: Mon Mar 9 17:58:20 2020
```

```
Turnaround Time:
                                8 seconds
102
           Waiting Time:
                                 8 seconds
           Response Time:
                                 8 seconds
104
105 Metrics for job ./microbatch.out:
           CPU Burst:
106
                                2 seconds
           Interruptions:
                                0 times
108
           Priority:
                                1
109
           Arrival Time:
                                Mon Mar 9 17:58:12 2020
           First Time on CPU:
110
                                Mon Mar 9 17:58:20 2020
111
           Finish Time:
                                Mon Mar 9 17:58:22 2020
112
           Turnaround Time:
                                10 seconds
           Waiting Time:
                                 8 seconds
113
114
           Response Time:
                                 8 seconds
116 Metrics for job ./microbatch.out:
117
           CPU Burst:
                                1 seconds
118
           Interruptions:
                                0 times
119
           Priority:
                                1
           Arrival Time:
                                Mon Mar 9 17:58:14 2020
           First Time on CPU: Mon Mar 9 17:58:22 2020
121
           Finish Time:
                                Mon Mar 9 17:58:23 2020
123
           Turnaround Time:
                                9 seconds
124
           Waiting Time:
                                8 seconds
125
                                8 seconds
           Response Time:
127 Metrics for job ./microbatch.out:
128
           CPU Burst:
                           2 seconds
129
           Interruptions:
                                0 times
           Priority:
                                2
131
           Arrival Time:
                                Mon Mar 9 17:58:14 2020
132
           First Time on CPU:
                                Mon Mar 9 17:58:23 2020
                                Mon Mar 9 17:58:25 2020
133
           Finish Time:
134
           Turnaround Time:
                                11 seconds
           Waiting Time:
                                 9 seconds
           Response Time:
                                9 seconds
138 Metrics for job ./microbatch.out:
           CPU Burst:
                                1 seconds
                                0 times
140
           Interruptions:
141
           Priority:
                                4
142
           Arrival Time:
                                Mon Mar 9 17:58:14 2020
           First Time on CPU:
143
                                Mon Mar 9 17:58:25 2020
144
           Finish Time:
                                Mon Mar 9 17:58:26 2020
145
                                12 seconds
           Turnaround Time:
146
           Waiting Time:
                                11 seconds
147
           Response Time:
                                11 seconds
148
149 Metrics for job ./microbatch.out:
           CPU Burst:
                                1 seconds
           Interruptions:
                                0 times
```

```
152
            Priority:
            Arrival Time:
153
                                 Mon Mar 9 17:58:15 2020
            First Time C...
Finish Time:
            First Time on CPU:
                                Mon Mar 9 17:58:26 2020
154
                                 Mon Mar 9 17:58:27 2020
                                 12 seconds
157
            Waiting Time:
                                 11 seconds
158
            Response Time:
                                11 seconds
159
160 Metrics for job ./microbatch.out:
           CPU Burst:
161
                           3 seconds
            Interruptions:
                                 0 times
163
            Priority:
                                4
164
            Arrival Time:
                                Mon Mar 9 17:58:16 2020
            First Time on CPU: Mon Mar 9 17:58:27 2020
            Finish Time:
                                Mon Mar 9 17:58:30 2020
            Turnaround Time:
                                14 seconds
167
168
            Waiting Time:
                                11 seconds
169
            Response Time:
                                 11 seconds
170
171 Overall Metrics for Batch:
           Total Number of Jobs Completed: 15
172
173
            Total Number of Jobs Submitted: 15
174
            Average Turnaround Time:
                                        7.067 seconds
175
            Average Waiting Time:
                                          5.867 seconds
                                           5.867 seconds
176
            Average Response Time:
            Average CPU Burst:
177
                                            1.200 seconds
178
            Total CPU Burst:
                                            18 seconds
179
            Throughput:
                                            0.142 No./second
            Max Turnaround Time:
                                           14 seconds
            Min Turnaround Time:
                                            2 seconds
182
            Max Waiting Time:
                                            11 seconds
184
            Min Waiting Time:
                                            0 seconds
            Max Response Time:
                                            11 seconds
187
            Min Response Time:
                                            0 seconds
188
            Max CPU Burst:
                                            3 seconds
            Min CPU Burst:
                                            0 seconds
```

Performance Evaluations

For each benchmark shortest job first performed the best. This is to be expected as it will achieve the highest response and waiting time. However with shortest job first it is possible to have job starvation as if you keep filling the queue with short jobs a longer job may have to wait forever to execute.

Priority based comes in a close second, but as these numbers are randomly generated it is hard to give accurate metrics. Also note that when providing an arrival time > max CPU burst, the jobs are scheduled in a first come, first served way as the current job is completed before the next job can arrive.

Lessons Learned

Before this project, I had "okay" proficiency with the C language. I had never dealt with programming with mutexes, conditional variables or threads so those new additions were a challenge. Dr. Qin's source code aided in the creation process as a base to go off of. I feel though after this project my C language understanding has doubled, if not tripled. The biggest hurdle I had was dealing with double pointers, my custom type process_p and process_t and dealing with thread synchronization. For example the day of submission I realized I had a large bug with the benchmark code. I was not ensuring a small edge case code was locking the mutex to ensure it was synced with the other thread.

Conclusion

This project was very interesting and taught me a whole deal about multi-threading and job scheduling algorithms. I wish I used more automatic testing frameworks such as CUnit to speed up my testing process. For example, whenever I made a new change I would manually test to make sure I did not break anything, with an automatic testing framework it would have lessened the time spent manually testing my code.

Additionally, it would have been interesting to have implemented a preemptive scheduling algorithm as the metrics for that would have been very interesting to see.

References

- 1: https://en.wikipedia.org/wiki/Central_processing_unit
- 2: https://en.wikipedia.org/wiki/Scheduling_(computing)#First_come,_first_served
- 3: https://en.wikipedia.org/wiki/Shortest_job_next
- 4: https://en.wikipedia.org/wiki/Scheduling_(computing)#Fixed_priority_pre-emptive_scheduling
- 5: http://www.cplusplus.com/
- 6: https://www.geeksforgeeks.org/