Harrithha, Shashank P
200010018, 200010048

CS314 Operating Systems Lab
Lab 5

---

1 Part 1

1.1 Changes in the Minix3 source code

In order to study the time quanta spent in the CPU by the processes, the file **system.c** was changed at **minix/kernel/**. The following statement was added in the file at **line number 667** to print the Allotted quantum and Used Quantum:

```
printf("Minix: Allotted quantum is: %d , Used Quantum is : %d, 200010018 200010048\n",
                            p->p_quantum_size_ms,
                            p->p_quantum_size_ms - cpu_time_2_ms(p->p_cpu_time_left));
```

These changes were made in the host machine (Windows). It was transferred to Minix3 Virtual Machine through github. Then, `make build MKUPDATE=yes` was run and the Minix3 Virtual Machine was rebooted. Upon running the `ls` command, the following output was obtained:

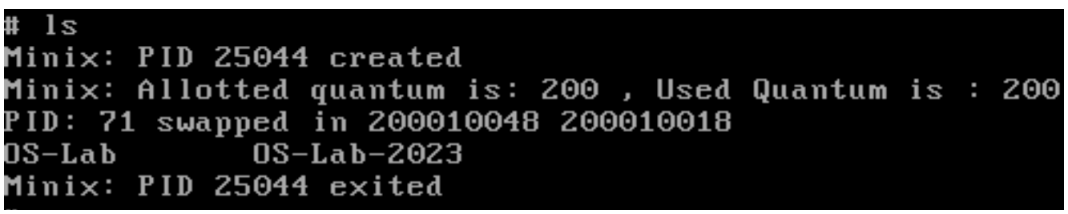


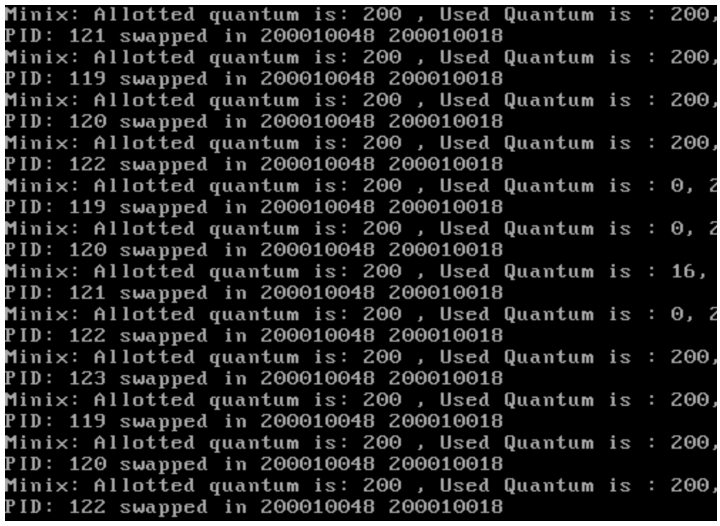
Figure 1: Displaying Allotted Quantum

1.2 Workload mixes

Following workload mixes were created which spawned 5 processes each:

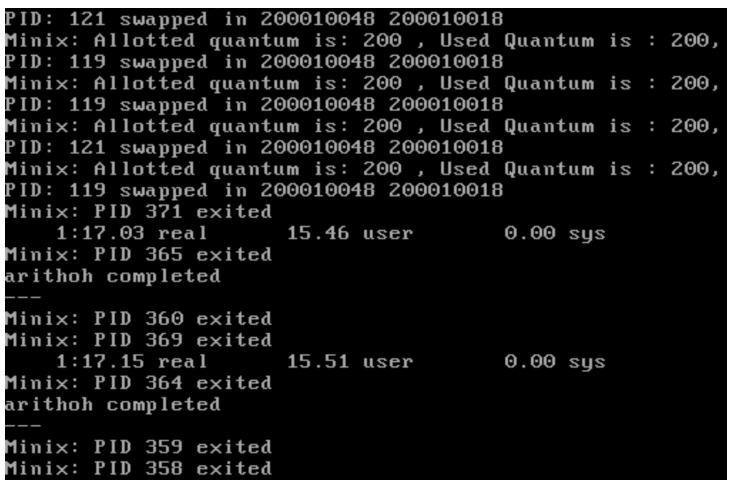
1.2.1 workload_mix1.sh

```
#!/bin/sh
./arithoh.sh &
./arithoh.sh &
./arithoh.sh &
./arithoh.sh &
./arithoh.sh &
wait
```



```
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 121 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 119 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 120 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 122 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 0, 2
PID: 119 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 0, 2
PID: 120 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 16,
PID: 121 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 0, 2
PID: 122 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 123 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 119 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 120 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 122 swapped in 200010048 200010018
```

Figure 2: Output in between



```
PID: 121 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 119 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 119 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 121 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 119 swapped in 200010048 200010018
Minix: PID 371 exited
    1:17.03 real      15.46 user       0.00 sys
Minix: PID 365 exited
arithoh completed
---
Minix: PID 360 exited
Minix: PID 369 exited
    1:17.15 real      15.51 user       0.00 sys
Minix: PID 364 exited
arithoh completed
---
Minix: PID 359 exited
Minix: PID 358 exited
```

Figure 3: Final Output

Inferences: This workload mix contains **5 arithoh.sh** processes, all of which are CPU intensive. From the output, we have the following observations,

- Since every process is the same CPU-intensive task, each process is allotted **200** quantum of CPU time
- Also, since it is CPU bound almost always, it uses all 200 of the allotted quantum, although there are a few outliers that do not use all of the quantum.
- Since each process is the same, they all run alternatively one after the other, in a **Round-Robin** fashion

1.2.2 workload_mix2.sh

```sh
#!/bin/sh
./fstime.sh &
./fstime.sh &
./fstime.sh &
./fstime.sh &
./fstime.sh &
wait
```

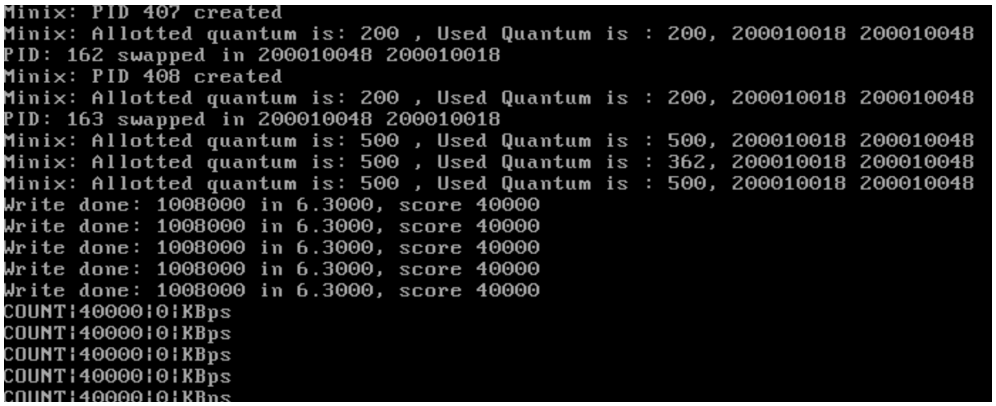


Figure 4: Output in between

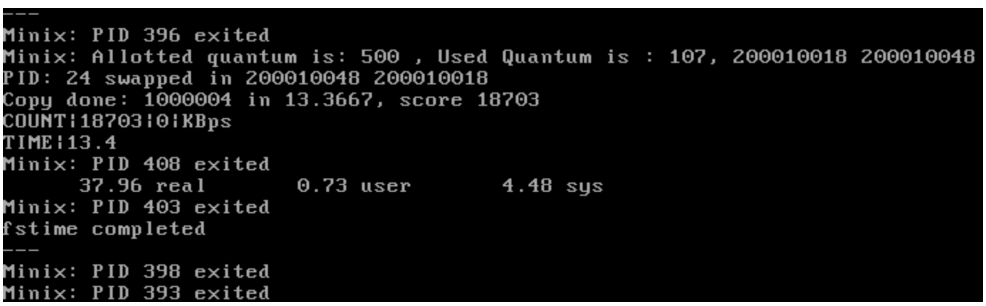


Figure 5: Final Output

Inferences:

This workload mix contains **5 fstime.sh** processes, all of which are IO intensive. From the output, we have the following observations,

- Since every process is the same IO-intensive task, each process is allotted **500** quantum of CPU time.
- Since they are all IO bound, they execute sequentially in a **Round-Robin** fashion.
- The processes wait till they receive I/O and then complete their operations. In practice, they do not utilize the allotted quanta.

1.2.3 workload_mix3.sh

```
#!/bin/sh
./fstime.sh &
./fstime.sh &
./arithoh.sh &
./fstime.sh &
./arithoh.sh &
wait
```

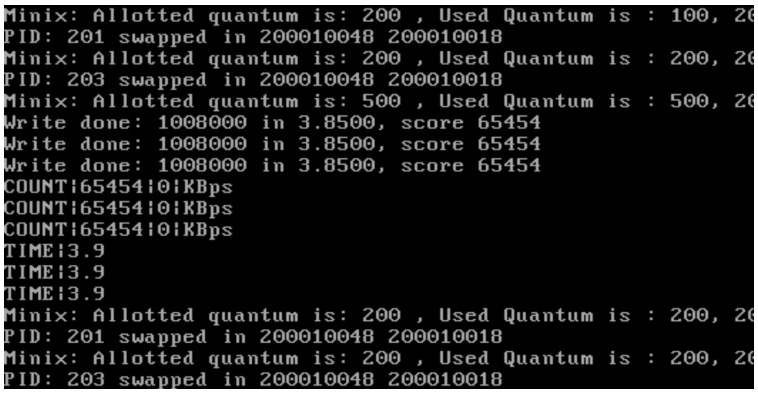


Figure 6: Output in between

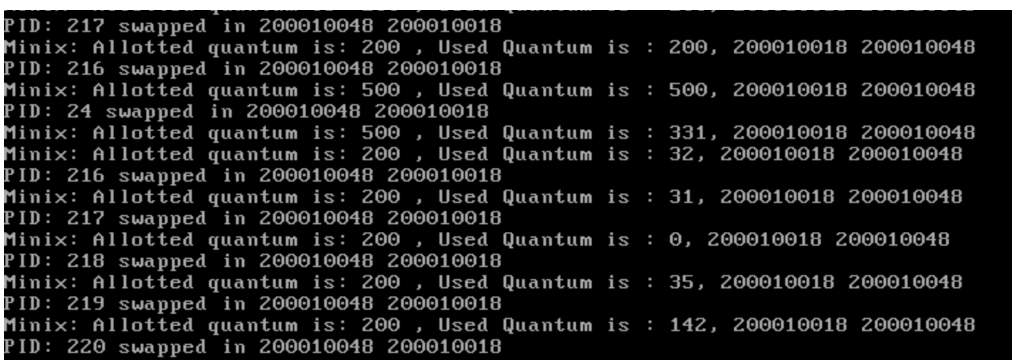


Figure 7: Final Output

Inferences: This workload mix contains **2 arithoh.sh** and **3 fstime.sh** processes, which is a combination of IO and CPU bound jobs. From the output, we have the following observations,

- Since CPU-intensive tasks are allotted **200** quantum of CPU time and IO-intensive tasks are allotted **500** quantum of CPU time, we observe that CPU and IO tasks are being executed alternatively.
- It is observed that the CPU bound processes mostly use all 200 of the allotted quantum, although there are a few outliers that do not use all of the quantum.
- On the other hand, the IO processes generally do not use the full quantum allotted to them and wait for IO inputs to proceed.
- They all run in a **Round-Robin** fashion

1.2.4 workload_mix4.sh

```
#!/bin/sh
./syscall &
./arithoh.sh &
./arithoh.sh &
./loop.sh &
./fstime.sh &
wait
```

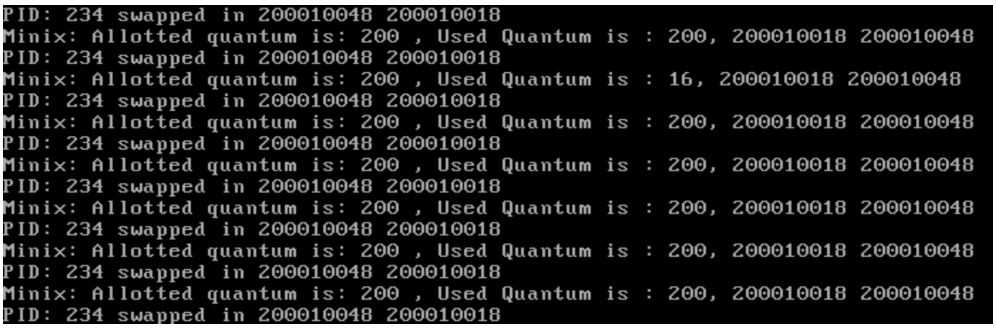


Figure 8: Output in between

Inferences: This workload mix contains **2 arithoh.sh** processes, **1 fstime.sh** process, **1 loop.sh** process, **1 syscall.sh** process. From the output, we have the following observations,

- This is a mixture of CPU, IO and infinite loop processes. We observe that all these run alternatively in a **Round-Robin** fashion.
- Loop does not complete and uses all of its allotted quanta, CPU and syscall also use their allotted quanta where as IO processes don't fully use their allotted quanta.

2 Psedo FIFO Implementation

In order to implement **Pseudo FIFO** policy, we studied the `schedule.c` file given. The file **schedule.c** was changed at **minix/servers/sched/**
location. The following changes were performed to achieve the goal.

The entire balance queues function was commented out as we did not require jobs to increase their priority with time.

```
static void balance_queues(minix_timer_t *tp)
{
    // struct schedproc *rmp;
    // int proc_nr;

    // for (proc_nr = 0, rmp = schedproc; proc_nr < NR_PROCS; proc_nr++, rmp++)
    // {
    //  if (rmp->flags & IN_USE)
    //  {
    //    if (rmp->priority > rmp->max_priority)
    //    {
    //      rmp->priority -= 1; /* increase priority */
    //      schedule_process_local(rmp);
    //    }
    //  }
    // }

    // set_timer(&sched_timer, balance_timeout, balance_queues, 0);
}
```

In line no. 108, instead of lowering the priority of a job that uses the entire quanta, we increase the priority thereby "encouraging" the process to continue without a context switch.

```
rmp = &schedproc[proc_nr_n];
if (rmp->priority < MIN_USER_Q)
{
    rmp->priority -= 1; /* increase priority */
}
```

These changes were made in the host machine (Windows). It was transferred to Minix3 Virtual Machine through GitHub. These changes approximately simulate the behaviour of First In, First Out algorithm. Then, `make build MKUPDATE=yes` was run and the Minix3 Virtual Machine was rebooted.

2.1 Workload mixes

Following workload mixes were created which spawned 5 processes each:

2.1.1 workload_mix1.sh

```
#!/bin/sh
./arithoh.sh &
./arithoh.sh &
./arithoh.sh &
./arithoh.sh &
./arithoh.sh &
wait
```

Figure 9: Output in between

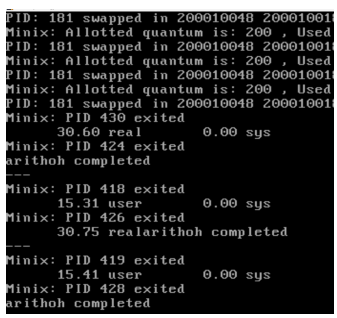


Figure 10: Final Output

Inferences: This workload mix contains **5 arithoh.sh** processes, all of which are CPU intensive. From the output, we have the following observations,

- Since every process is the same CPU-intensive task, each process is allotted **200** quantum of CPU time which it uses.
- The change we observe in the new implementation is that one process starts only after the previous process is completed as in **FIFO**.

2.1.2 workload_mix2.sh

```
#!/bin/sh
./fstime.sh &
./fstime.sh &
./fstime.sh &
./fstime.sh &
./fstime.sh &
wait
```

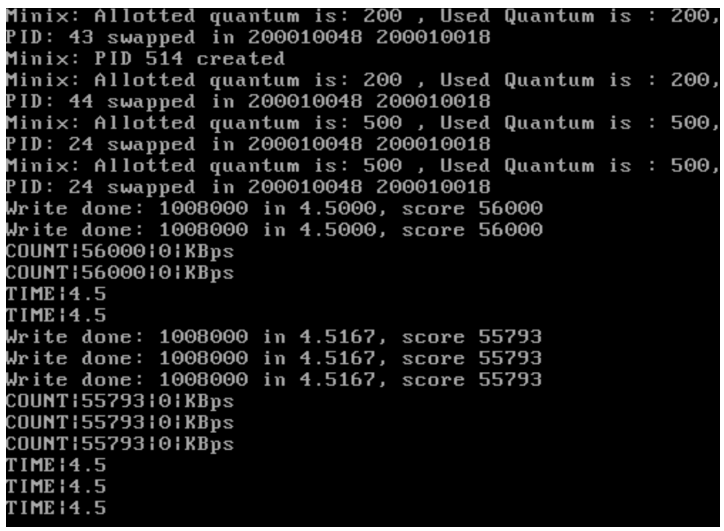
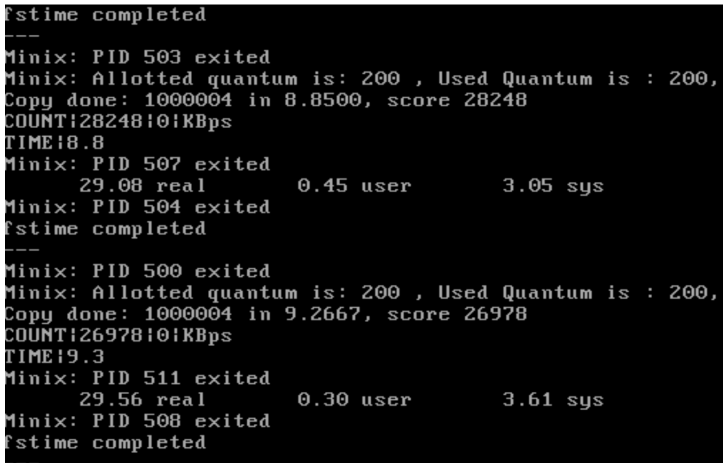


Figure 11: Output in between



```
fstime completed
---
Minix: PID 503 exited
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
Copy done: 1000004 in 8.8500, score 28248
COUNT|28248|0|KBps
TIME|8.8
Minix: PID 507 exited
      29.08 real       0.45 user       3.05 sys
Minix: PID 504 exited
fstime completed
---
Minix: PID 500 exited
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
Copy done: 1000004 in 9.2667, score 26978
COUNT|26978|0|KBps
TIME|9.3
Minix: PID 511 exited
      29.56 real       0.30 user       3.61 sys
Minix: PID 508 exited
fstime completed
```

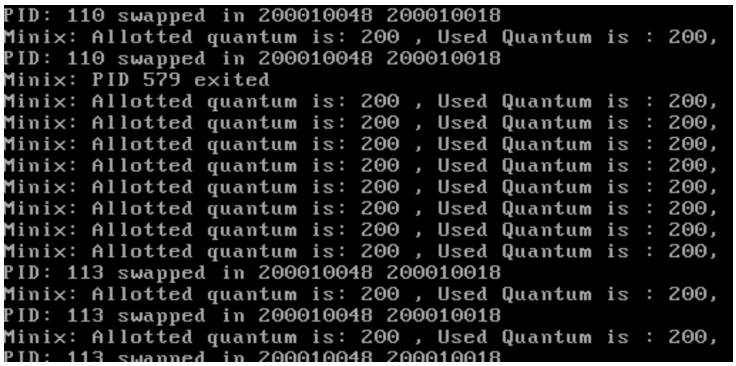
Figure 12: Final Output

Inferences: This workload mix contains **5 fstime.sh** processes, all of which are IO intensive. From the output, we have the following observations,

- Since every process is the same IO-intensive task, each process is allotted **500** quantum of CPU time, which is utilized completely as per observation.
- The I/O bound processes are sent to the waiting queue after requesting for I/O and then once done, are placed in the ready queue. They follow the normal Round Robin fashion as they can be executed only when an input is given. Until that, they are in blocked state.
- Thus, we observe that I/O processes are unable to follow the FIFO policy due to which it is called **Pseudo FIFO**.

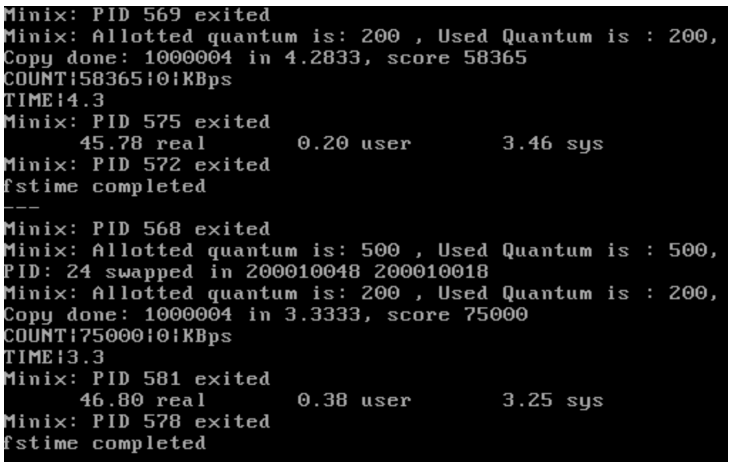
2.1.3 workload_mix3.sh

```
#!/bin/sh
./fstime.sh &
./fstime.sh &
./arithoh.sh &
./fstime.sh &
./arithoh.sh &
wait
```



```
PID: 110 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 110 swapped in 200010048 200010018
Minix: PID 579 exited
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 113 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 113 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
PID: 113 swapped in 200010048 200010018
```

Figure 13: Output in between



```
Minix: PID 569 exited
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
Copy done: 1000004 in 4.2833, score 58365
COUNT¦58365¦0¦KBps
TIME¦4.3
Minix: PID 575 exited
      45.78 real       0.20 user       3.46 sys
Minix: PID 572 exited
fstime completed
---
Minix: PID 568 exited
Minix: Allotted quantum is: 500 , Used Quantum is : 500,
PID: 24 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200,
Copy done: 1000004 in 3.3333, score 75000
COUNT¦75000¦0¦KBps
TIME¦3.3
Minix: PID 581 exited
      46.80 real       0.38 user       3.25 sys
Minix: PID 578 exited
fstime completed
```

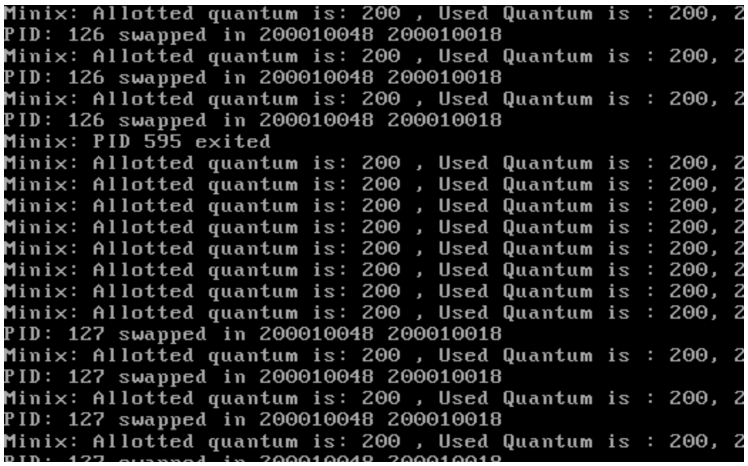
Figure 14: Final Output

Inferences: This workload mix contains **2 arithoh.sh** and **3 fstime.sh** processes, which is a combination of IO and CPU bound jobs. From the output, we have the following observations,

- Since CPU-intensive tasks are allotted **200** quantum of CPU time and IO-intensive tasks are allotted **500** quantum of CPU time, we observe that CPU tasks are executed first in FIFO fashion as the IO tasks are sent to the waiting queue for inputs.
- It is observed that the CPU and IO bound processes mostly use all of the allotted quantum, although there are a few outliers that do not use all of the quantum.
- Even though fstime comes before arithoh, arithoh finishes first because of the FIFO policy while fstime is sent to waiting queue for receiving inputs.

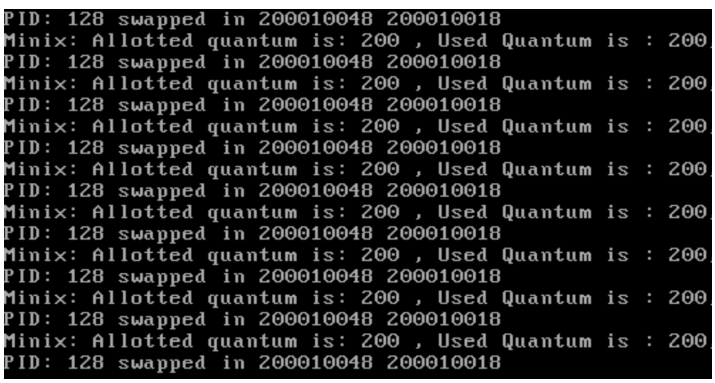
2.1.4 workload_mix4.sh

```
#!/bin/sh
./syscall &
./arithoh.sh &
./arithoh.sh &
./loop.sh &
./fstime.sh &
wait
```



```
Minix: Allotted quantum is: 200 , Used Quantum is : 200, 2
PID: 126 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200, 2
PID: 126 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200, 2
PID: 126 swapped in 200010048 200010018
Minix: PID 595 exited
Minix: Allotted quantum is: 200 , Used Quantum is : 200, 2
Minix: Allotted quantum is: 200 , Used Quantum is : 200, 2
Minix: Allotted quantum is: 200 , Used Quantum is : 200, 2
Minix: Allotted quantum is: 200 , Used Quantum is : 200, 2
Minix: Allotted quantum is: 200 , Used Quantum is : 200, 2
Minix: Allotted quantum is: 200 , Used Quantum is : 200, 2
Minix: Allotted quantum is: 200 , Used Quantum is : 200, 2
Minix: Allotted quantum is: 200 , Used Quantum is : 200, 2
PID: 127 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200, 2
PID: 127 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200, 2
PID: 127 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200, 2
PID: 127 swapped in 200010048 200010018
```

Figure 15: Output in between



```
PID: 128 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200
PID: 128 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200
PID: 128 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200
PID: 128 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200
PID: 128 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200
PID: 128 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200
PID: 128 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200
PID: 128 swapped in 200010048 200010018
Minix: Allotted quantum is: 200 , Used Quantum is : 200
PID: 128 swapped in 200010048 200010018
```

Figure 16: Output in between

Inferences: This workload mix contains **2 arithoh.sh** processes, **1 fstime.sh** process, **1 loop.sh** process, **1 syscall.sh** process. From the output, we have the following observations,

- This is a mixture of CPU, IO and infinite loop processes.
- **syscall** comes first, and is executed first by utilizing all of its allotted quanta, then arithoh executes in a similar way after which we have loop which starts executing but never finishes as it is an infinite loop.
- Since it is a FIFO policy, until this process finishes, the next process fstime is not executed until loop completes.

- Thus we observe that real time and CPU intensive processes somewhat follow the FIFO policy but the IO bound processes are unable to as they are sent to the waiting queue.