Assignment 4

Operating Systems Lab, Group 12

Cheryala Rohith	190010012	190010012@iitdh.ac.in
Mohammad Sameer	190010024	190010024@iitdh.ac.in

Part 1

In part 1, we had to prepare 4 workload mixes having different characteristics, ranging from all CPU benchmarks to all I/O-intensive benchmarks, with each workload spawning 5 processes. We had to make observation on the way these processes were scheduled and the time quanta spent in the CPU by the processes.

To find out how much time quanta each process has been allocated and how much time it had spend, we have modified the `sched_proc()` function of the `system.c` present in the `/minix/kernel/` directory. The following code was added:

```
printf("Minix: Allotted Quantum: %d ms\nMinix: Used Quantum: %d ms\n",
p->p_quantum_size_ms,
p->p_quantum_size_ms - cpu_time_2_ms(p->p_cpu_time_left));
```

To copy the modified file and re-build the OS, we have written a bash file named `run1.sh`. Upon executing `run1.sh`, and successful build followed by a reboot, the changes should be reflected.

The workloads used by us and our observations and inferences are presented in the sections below.

1.1 `workload_mix1.sh`

The below are the content of `workload_mix1.sh` file,

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

This workload spawns 5 different instances of `airthoh.sh` process, which is CPU bound.

```
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 137 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 136 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 139 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 138 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 140 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 137 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 136 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 139 swapped in
```

Figure 1: Running `workload_mix1.sh`

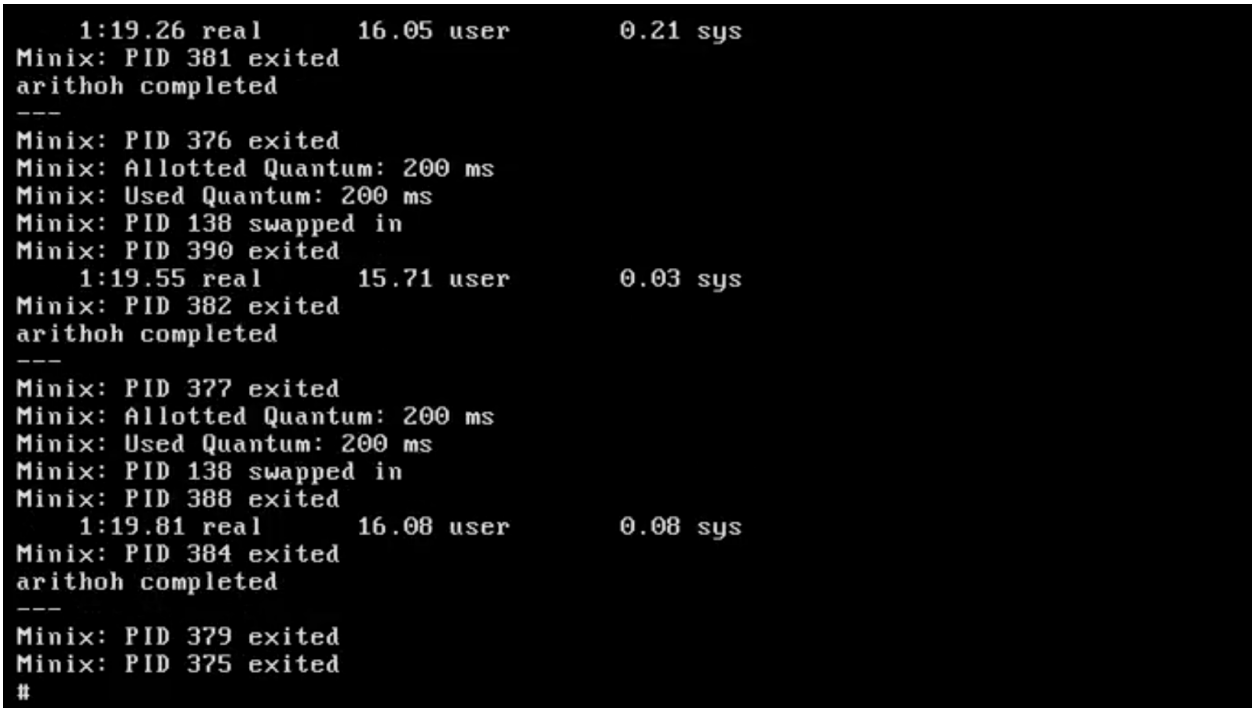

Figure 2: Completed `workload_mix1.sh`

Observations and Inferences: The `airthoh.sh` is a CPU-bound process. As seen from figure 1, the 5 instances of this process have `PID: 136, 137, 138, 139 and 140`. All of these processes were allocated equal quanta of time of `200 ms` by default. And all of these processes, utilized it completely. Then in a round robin manner, these processes execute one after the other. Running a parallel way, it leads to their completion almost at the same time as seen from figure 2.

1.2 `workload_mix2.sh`

The below are the content of `workload_mix2.sh` file,

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

This workload spawns 3 different instances of `airthoh.sh` process, which is CPU bound and 2 different instances of `fstime.sh` process which is I/O bound.

```
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 160 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 162 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 158 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 160 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 162 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 158 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 160 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 162 swapped in
_
```

Figure 3: Running `workload_mix2.sh`

```
Minix: PID 158 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 160 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 196 ms
Minix: PID 162 swapped in
Minix: Allotted Quantum: 500 ms
Minix: Used Quantum: 500 ms
Write done: 1008000 in 2.5333, score 99473
Write done: 1008000 in 2.5333, score 99473
COUNT¦99473¦0¦KBps
COUNT¦99473¦0¦KBps
TIME¦2.5
TIME¦2.5
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 158 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 162 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 160 swapped in
```

Figure 4: Running `workload_mix2.sh` showing write to disk done

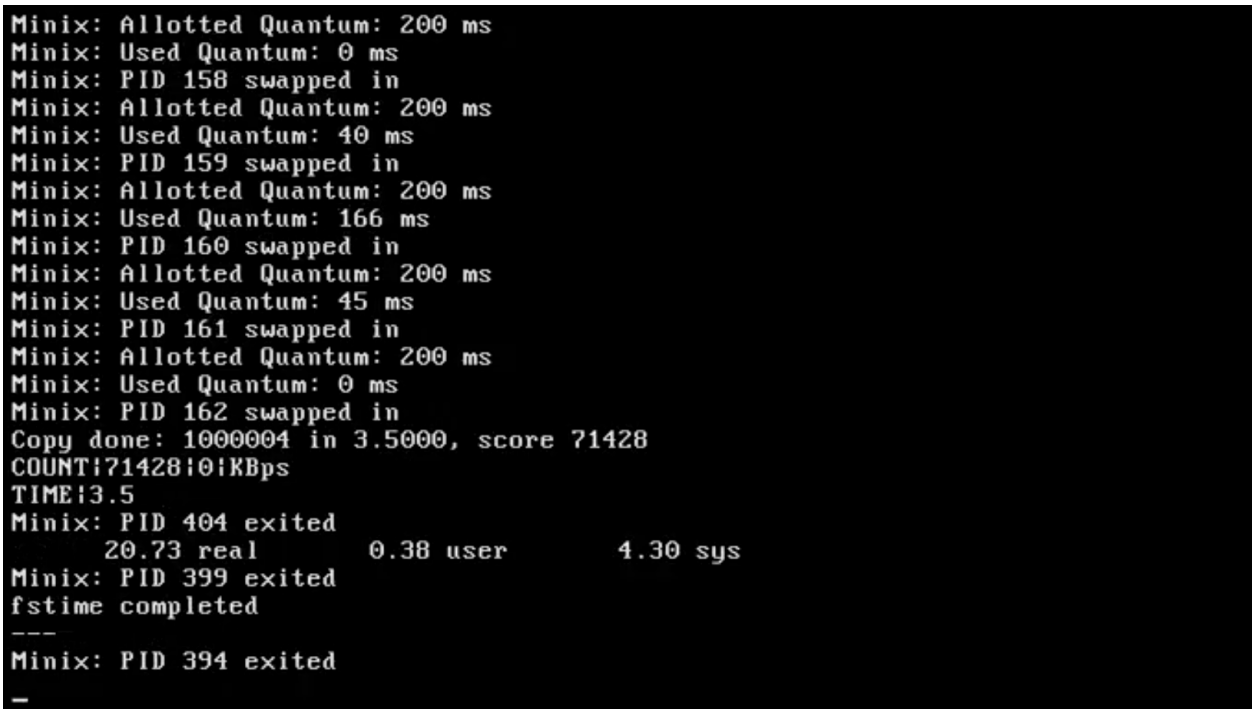

Figure 5: Running `workload_mix2.sh` showing `fstime.sh` completed

Observations and Inferences: The `airthoh.sh` is a CPU-bound process, while the `fstime.sh` is a I/O bound process. As seen from above figures, 3 instances of `airthoh.sh` have `PID: 158, 160 and 162`, while 2 instances of `fstime.sh` have `PID: 159 and 161`. By default, all of these processes, were allocated a time quanta of `200 ms`.

The I/O bound process wait for I/O to respond, while CPU bound process run in a round robin manner as seen from figure 3. Since, `fstime.sh` is I/O bound, so it doesn't completely utilize its time quanta. As a result, next time it has been allocated a time quanta of `500 ms` as seen from figure 4. When I/O responds, the I/O bound processes complete their execution as seen from figure 5. After this, all the remaining CPU bound processes run in round robin manner till completion.

1.3 `workload_mix3.sh`

The below are the content of `workload_mix3.sh` file,

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

This workload spawns 5 different instances of `syscall.sh` process, which is CPU bound but not as CPU intensive as `airthoh.sh`.

```
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 176 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 177 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 178 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 196 ms
Minix: PID 175 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 161 ms
Minix: PID 176 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 198 ms
Minix: PID 177 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 175 ms
Minix: PID 178 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 157 ms
Minix: PID 179 swapped in
_
```

Figure 6: Running `workload_mix3.sh`

```
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 179 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 177 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 175 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 34 ms
Minix: PID 175 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 43 ms
Minix: PID 176 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 34 ms
Minix: PID 177 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 35 ms
Minix: PID 178 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 36 ms
Minix: PID 179 swapped in
```

Figure 7: Running `workload_mix3.sh` showing various time quanta

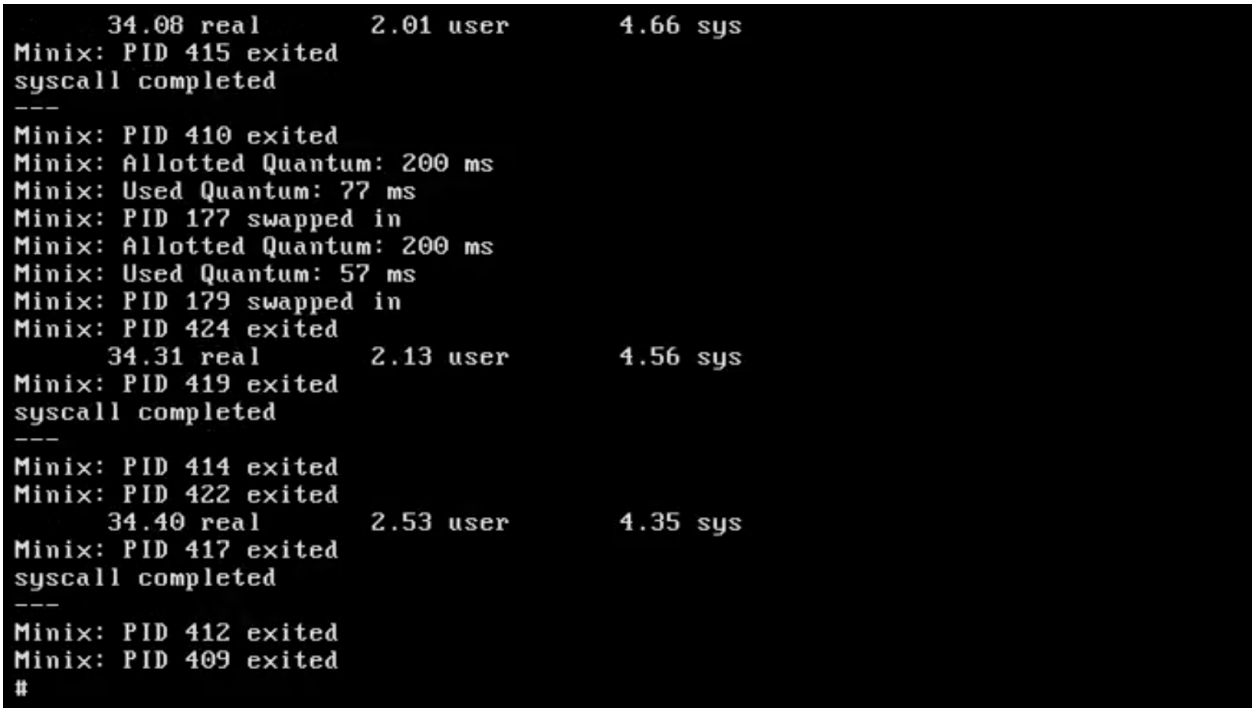

Figure 8: Completed `workload_mix3.sh`

Observations and Inferences: The `syscall.sh` processes have `PID: 175, 176, 177, 178 and 179` as seen from figure 6. By default, these all processes have been allocated a time qunata of `200 ms`. These processes are CPU bound but they not utilize their complete time quanta since they are not as CPU intensive as `airthoh.sh` as seen from figure 7. As a result, they run parallelly in a round robin manner until completion as seen above figures. Also, they all complete execution almost at the same time.

1.4 `workload_mix4.sh`

The below are the content of `workload_mix4.sh` file,

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

This workload spawns 5 different instances of `fstime.sh` process, which is I/O bound.

```
Minix: PID 201 swapped in
Minix: Allotted Quantum: 500 ms
Minix: Used Quantum: 500 ms
Minix: Allotted Quantum: 500 ms
Minix: Used Quantum: 432 ms
Minix: Allotted Quantum: 500 ms
Minix: Used Quantum: 500 ms
Write done: 1008000 in 4.9833, score 50568
Write done: 1008000 in 4.9833, score 50568
Write done: 1008000 in 4.9833, score 50568
Write done: 1008000 in 4.9833, score 50568
Write done: 1008000 in 4.9833, score 50568
COUNT¦50568¦0¦KBps
COUNT¦50568¦0¦KBps
COUNT¦50568¦0¦KBps
COUNT¦50568¦0¦KBps
COUNT¦50568¦0¦KBps
TIME¦5.0
TIME¦5.0
TIME¦5.0
TIME¦5.0
TIME¦5.0
Minix: Allotted Quantum: 500 ms
Minix: Used Quantum: 163 ms
_
```

Figure 9: Running `workload_mix4.sh`

```
      33.48 real       0.43 user       3.01 sys
Minix: PID 434 exited
fstime completed
---
Minix: PID 429 exited
Copy done: 1000004 in 12.5000, score 20000
COUNT¦20000¦0¦KBps
TIME¦12.5
Minix: PID 440 exited
      33.48 real       0.53 user       5.01 sys
Minix: PID 435 exited
fstime completed
---
Minix: PID 430 exited
Copy done: 1000004 in 12.5500, score 19920
COUNT¦19920¦0¦KBps
TIME¦12.5
Minix: PID 437 exited
      33.55 real       0.46 user       3.20 sys
Minix: PID 432 exited
fstime completed
---
Minix: PID 427 exited
Minix: PID 426 exited
# _
```

Figure 10: Completed `workload_mix4.sh`

Observations and Inferences: The `fstime.sh` process is I/O bound. By default, these all processes have been allocated a time quanta of `200 ms`. But since they all are I/O

bound, they do not utilize the complete time quanta. As a result, next time they all get a allocated time quanta of `500 ms` as seen from figure 9. They wait until the I/O responds, and then run in round robin manner until completion as seen from figure 10.

Part 2

In part 2 we had to modify the user-level scheduler in minix 3 to the following "Pseudo-FIFO" policy: among the user-level processes that are ready to execute, the one that entered the earliest must be scheduled.

The minix 3 scheduler by default following Round Robin policy inside each queue. Whenever a process uses complete quanta of time allocated to it, it will be send the subsequent lower priority queue. To avoid starvation, periodically, all the processes will get the priority boosted after certain interval of time.

To implement Pseudo-FIFO, we have to first not decrease the priority of the process that uses its complete time quanta. Instead, we have to give it higher priority so that it completes its execution, in a FIFO manner. To to this, we have modified the `do_noquantum()` function of the `schedule.c` file present in the `/minix/servers/sched/` directory as follows:

```
rmp->priority -= 1; /* higher priority */
```

Next, to avoid overflowing of queue, we have to ensure periodic priority boosting doesn't occur. To do this we have commented out the following code present in the `balance_queues()` function of the same file.

```
// rmp->priority -= 1; /* higher priority */
```

To copy the modified file and re-build the OS, we have written a bash file named `run2.sh`. Upon executing `run2.sh`, and successful build followed by reboot, the changes should be reflected.

After this, we ran the workloads of part 1 again and our observations and inferences are presented in sections below.

2.1 `workload_mix1.sh`

The below are the content of `workload_mix1.sh` file,

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

```
./arithoh.sh &
wait
```

This workload spawns 5 different instances of `airthoh.sh` process, which is CPU bound.

```
Minix: PID 373 created
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 123 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 119 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 119 swapped in
_
```

Figure 11: Running `workload_mix1.sh` where `PID 123` is running

```
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 0 ms
Minix: PID 120 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 120 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 120 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 120 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 120 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 120 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 120 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 120 swapped in
_
```

Figure 12: Running `workload_mix1.sh` where `PID 120` is running

```
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 122 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 122 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 122 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 122 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 122 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 122 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 122 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 122 swapped in
```

Figure 13: Running `workload_mix1.sh` where `PID 122` is running

Observations and Inferences: The `airthoh.sh` is CPU bound process. All of these processes were allocated a time quanta of `200 ms` by default. Since they are CPU bound,

they utilize the complete time quanta. Unlike round robin manner, in the Pseudo-FIFO, one process starts its execution and runs till completion. In the figure 11, we can see process with `PID 123`, is being scheduled back to back indicating a FIFO policy. Similar observations can be made from figure 12 and 13.

2.2 `workload_mix2.sh`

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

This workload spawns 3 different instances of `airthoh.sh` process, which is CPU bound and 2 different instances of `fstime.sh` process which is I/O bound.

```
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 138 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 138 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 138 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 138 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 138 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 138 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 138 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 138 swapped in
```

Figure 14: Running `workload_mix2.sh` where `PID 138` is running

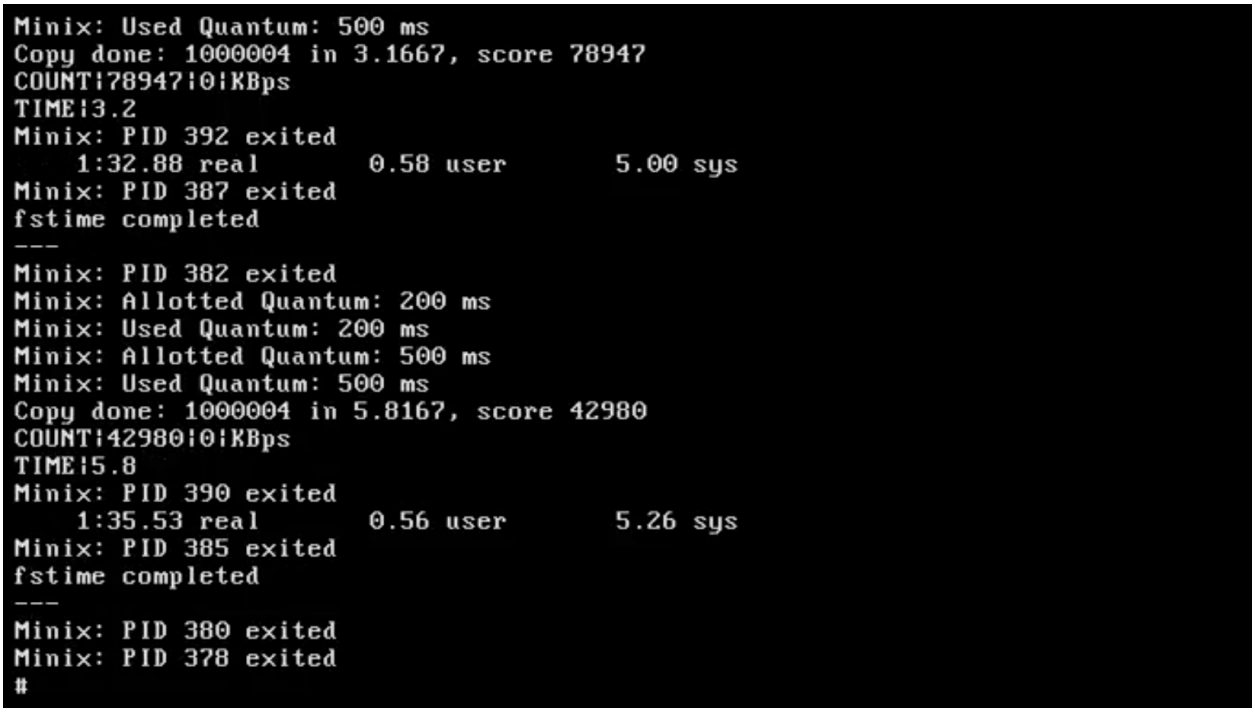

Figure 15: Completed `workload_mix2.sh`

Observations and Inferences: The `airthoh.sh` is CPU bound, where as `fstime.sh` is I/O bound. By default, all of these process have been allocated a time quanta of 200 ms. In this case, we can observe the working of actual "Pseudo" FIFO. In an actual FIFO policy, when an I/O bound process is scheduled, the CPU has to wait until I/O responds, then finish the process and schedule some other process. But in our case, whenever a process waits for I/O response, we simply preempt it and schedule some other process, and let it run. This is the reason we call it "Pseudo" FIFO.

In our case, while the I/O bound `fstime.sh` processes wait for I/O to respond, the scheduler schedules CPU bound `airthoh.sh` processes. But unlike the default minix scheduler, this scheduler follows a FIFO policy in case of CPU bound processes. As seen from figure 14, process with `PID 138` is scheduled back to back indicating a FIFO policy. The `fstime.sh` processes being I/O bound do not completely utilize the allocated time quanta and they are allocated a time quanta of `500 ms` next time. After the I/O responds, the `fstime.sh` processes are scheduled and completed as seen from figure 15.

2.3 `workload_mix3.sh`

```
#!/bin/sh
./syscall.sh &
./syscall.sh &
./syscall.sh &
./syscall.sh &
./syscall.sh &
```

```
wait
```

This workload spawns 5 different instances of `syscall.sh` process, which is CPU bound but not as CPU intensive as `airthoh.sh`.

```
Minix: PID 410 created
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 162 swapped in
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: PID 409 exited
      10.28 real       2.28 user       5.33 sys
Minix: PID 404 exited
syscall completed
---
Minix: PID 399 exited
_
```

Figure 16: Running `workload_mix3.sh` where `PID 162` is running

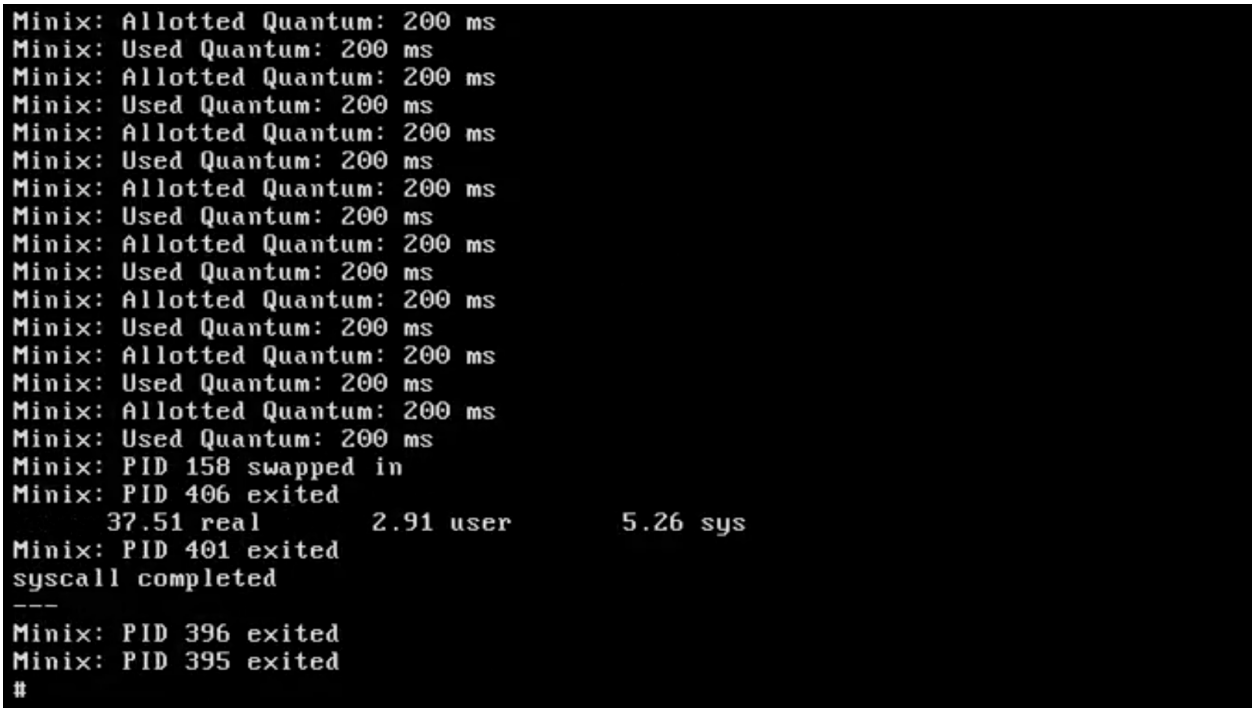

Figure 17: Completed `workload_mix3.sh`

Observations and Inferences: The `syscall.sh` is CPU bound process. By default these processes have been allocated a time quanta of `200 ms`. These processes run one after the other in a FIFO manner, unlike round robin manner of default minix. From figure 16, we can see that process with `PID 162` is being scheduled back to back until completion. Similar observation can be made from figure 17 as well. Unlike round robin, here once a process completes its execution, only then the next process is scheduled.

2.4 `workload_mix4.sh`

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

This workload spawns 5 different instances of `fstime.sh` process, which is I/O bound.

```
Read done: 1000004 in 7.9833, score 31315
COUNT|31315|0|KBps
TIME|8.0
Minix: Allotted Quantum: 500 ms
Minix: Used Quantum: 500 ms
Minix: PID 24 swapped in
Read done: 1000004 in 8.4667, score 29527
Read done: 1000004 in 8.4667, score 29527
Read done: 1000004 in 8.4667, score 29527
COUNT|29527|0|KBps
COUNT|29527|0|KBps
COUNT|29527|0|KBps
TIME|8.5
TIME|8.5
TIME|8.5
Minix: Allotted Quantum: 500 ms
Minix: Used Quantum: 93 ms
Minix: PID 24 swapped in
Minix: Allotted Quantum: 500 ms
Minix: Used Quantum: 500 ms
Minix: PID 24 swapped in
Minix: Allotted Quantum: 500 ms
Minix: Used Quantum: 235 ms
Minix: PID 24 swapped in
```

Figure 18: Running `workload_mix4.sh`

```
Minix: PID 426 exited
      36.90 real       0.68 user       5.48 sys
Minix: PID 421 exited
fstime completed
---
Minix: PID 416 exited
Minix: Allotted Quantum: 200 ms
Minix: Used Quantum: 200 ms
Minix: Allotted Quantum: 500 ms
Minix: Used Quantum: 500 ms
Minix: PID 24 swapped in
Minix: Allotted Quantum: 500 ms
Minix: Used Quantum: 190 ms
Minix: PID 24 swapped in
Copy done: 1000004 in 8.1167, score 30800
COUNT|30800|0|KBps
TIME|8.1
Minix: PID 425 exited
      39.48 real       0.41 user       5.35 sys
Minix: PID 420 exited
fstime completed
---
Minix: PID 415 exited
Minix: PID 414 exited
#
```

Figure 19: Completed `workload_mix4.sh`

Observations and Inferences: Again in this workload we can observe the "pseudo" nature of the Pseudo-FIFO policy. The `fstime.sh` is I/O bound process. By default these

processes were allocated a time quanta of 200 ms, but don't completely utilize it. So next time they are allocated a time quanta of 500 ms.

Unlike an actual FIFO policy, in our case when ever a process waits for I/O to respond, we simply preempt it. This was also observed in the case of round robin policy of default minix. Once the I/O responds, the processes are scheduled again until completion in a FIFO manner assuming they do not depend on I/O anymore. This is the reason we call our policy "Pseudo" FIFO. This policy follows FIFO, only in the case of CPU bound processes or when there is no dependency on I/O devices.