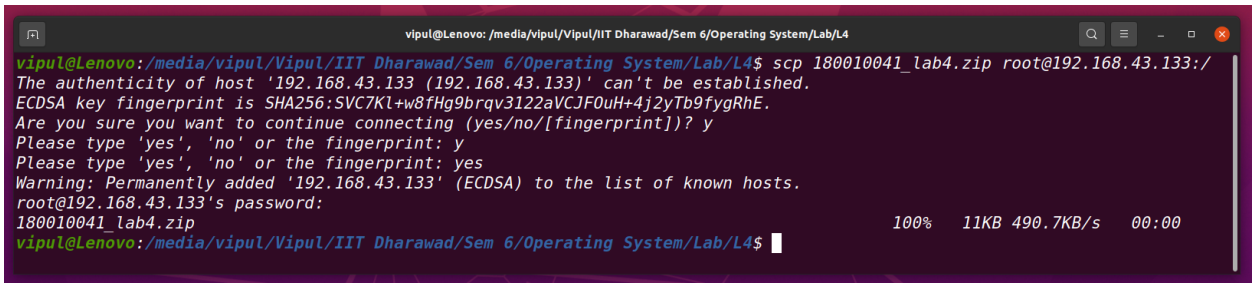
CS314 - Lab 4

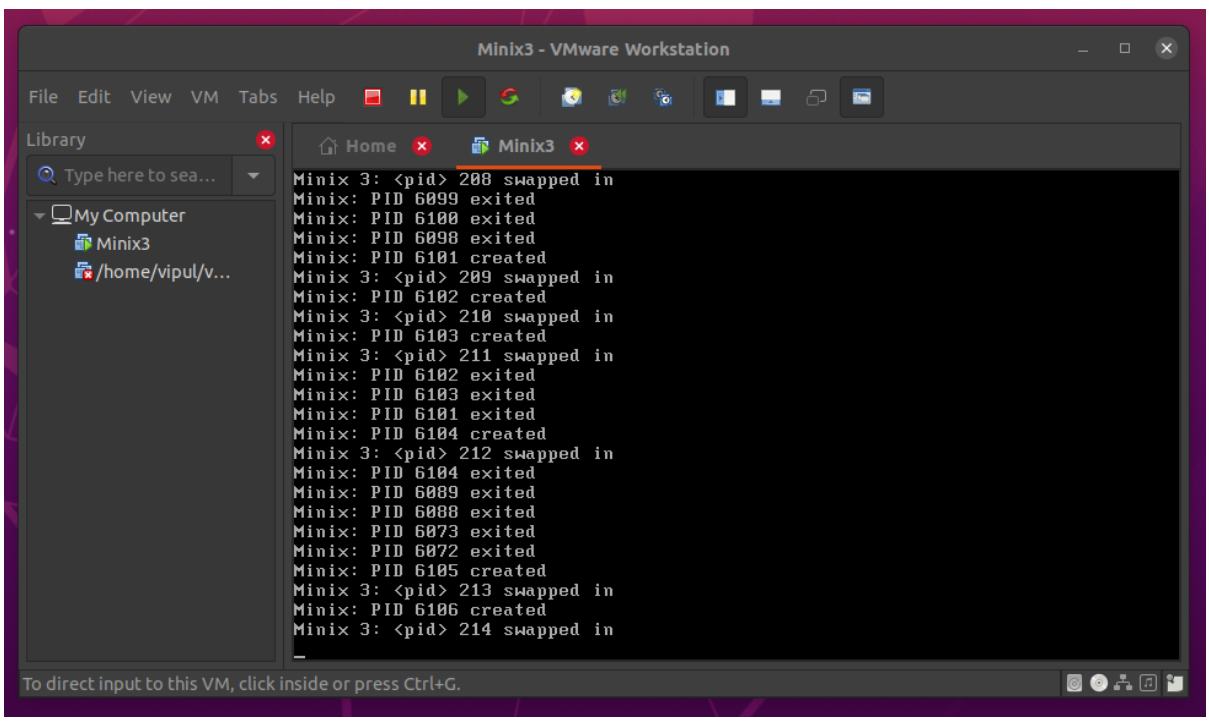
Vipul Yuvraj Nikam
180010041

PART 0 - Transfer:

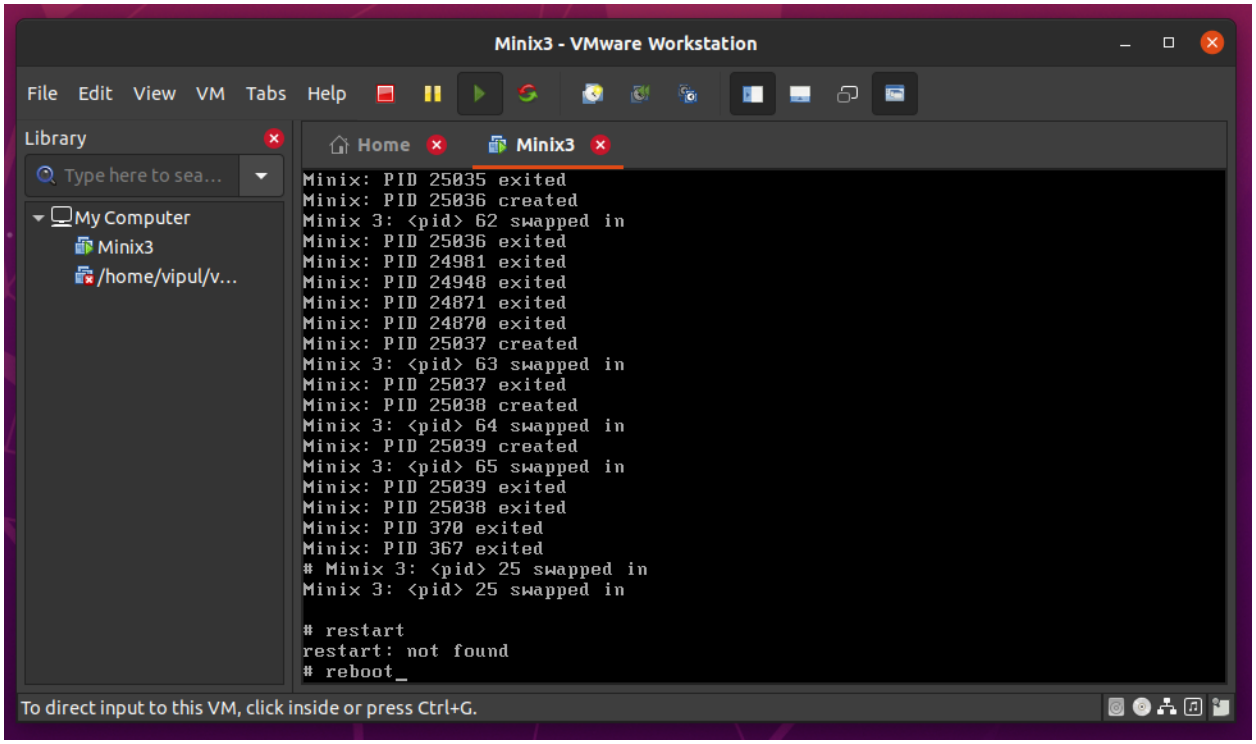
Transferring files from ubuntu to minix



Firstly we transferred the files from ubuntu to minix. Then we unzipped it in minix. And we run the `./run.sh` on minix terminal. And we got the following as output.

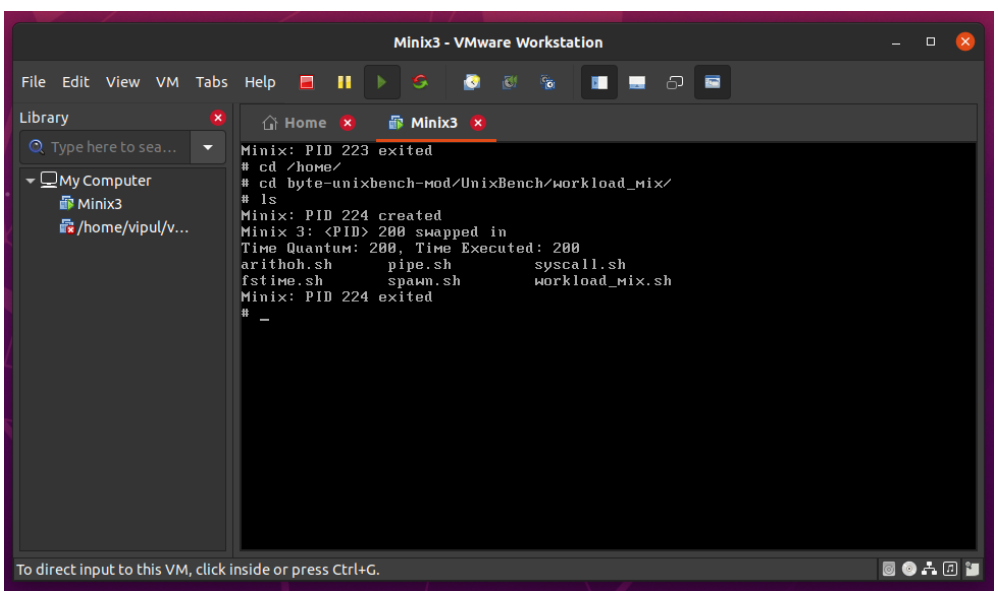


When *./run.sh* was over, we rebooted the system by *reboot*.



We study how the standard Minix3 scheduler schedules our workloads and presents our observations and inferences. In addition, we implement the Pseudo-FIFO policy for the MINIX3 programmer and make observations and inferences.

And now we run the following commands after going in the expected folder of workload_mix and their outputs are just attached below.



PART 1 - Standard Scheduler:

1. Workload Mix 1:

Here we executed the following command of 5 task,

./arithoh.sh & ./arithoh.sh & ./arithoh.sh & ./arithoh.sh & ./arithoh.sh

Observations

We see that the 5 processes are executed in parallel in the form of shifts. The time account, which is 200 by default, is used in full each time the process is scheduled.

Inference

The CPU fairly schedules all processes on a rotational basis. CPU intensive processes do not need to wait for I / O and run completely in the allotted quanta of time.

```
Time Quantum: 200, Time Executed: 0
Minix 3: 203 swapped in
Time Quantum: 200, Time Executed: 200
Minix 3: 206 swapped in
Time Quantum: 200, Time Executed: 200
Minix: PID 228 exited
    1:57.35 real      23.46 user       0.01 sys
Minix: PID 223 exited
arithoh completed
---
Minix: PID 218 exited
Minix: PID 231 exited
    1:57.45 real      23.43 user       0.01 sys
Minix: PID 226 exited
arithoh completed
---
Minix: PID 221 exited
[1]   Done                    ./arithoh.sh
[2]   Done                    ./arithoh.sh
[3]   Done                    ./arithoh.sh
[4]   Done                    ./arithoh.sh
#
```

Workload Mix 2:

Here we executed the following command of 5 task,

./arithoh.sh & ./fstime.sh & ./arithoh.sh & ./fstime.sh & ./arithoh.sh

Observations

We see that the 3 CPU intensive processes and the 2 linked I/O processes run in parallel in a round-robin fashion. However, the other 2 linked I/O processes are waiting for their I/O work and are then scheduled to run on the CPU.

Inference

CPU-bound tasks use all of their quanta, while I/O-bound tasks don't.

```
Time Quantum: 200, Time Executed: 200
Minix 3: 202 swapped in
Time Quantum: 200, Time Executed: 200
Minix 3: 206 swapped in
Time Quantum: 200, Time Executed: 200
Minix 3: 203 swapped in
Time Quantum: 200, Time Executed: 200
Minix 3: 204 swapped in
Time Quantum: 200, Time Executed: 200
Minix 3: 202 swapped in
Time Quantum: 200, Time Executed: 83
Minix 3: 203 swapped in
Time Quantum: 200, Time Executed: 0
Minix 3: 204 swapped in
Time Quantum: 200, Time Executed: 0
Minix 3: 205 swapped in
Time Quantum: 200, Time Executed: 200
Minix 3: 206 swapped in
Time Quantum: 200, Time Executed: 0
Minix 3: 202 swapped in
Time Quantum: 200, Time Executed: 200
```

Workload Mix 3:

Here we executed the following command of 5 task,

./pipe.sh & ./pipe.sh & ./fstime.sh & ./pipe.sh & ./pipe.sh

Observations

Since pipe.sh involves IPC, it invokes system calls and thus gets a higher quantum of 500 ms for system calls.

Inference

All 5 processes run on a rotating basis and fstime.sh rarely uses CPU as it is tied to I/O.

```
[4]   Done                    ./fstime.sh
# Minix 3: 234 swapped in
Time Quantum: 200, Time Executed: 200
Minix 3: 234 swapped in
Time Quantum: 200, Time Executed: 200
Minix 3: 234 swapped in
Time Quantum: 200, Time Executed: 200
Minix 3: 234 swapped in
Time Quantum: 200, Time Executed: 200
Minix 3: 234 swapped in
Time Quantum: 200, Time Executed: 200
Minix 3: 234 swapped in
Time Quantum: 200, Time Executed: 200
Minix 3: 234 swapped in
Time Quantum: 200, Time Executed: 200
Minix: PID 259 exited
    1:20.56 real      23.30 user       0.03 sys
Minix: PID 254 exited
arithoh completed
---
Minix: PID 249 exited
```

Workload Mix 4:

Here we executed the following command of 5 task,

./fstime.sh & ./fstime.sh & ./fstime.sh & ./fstime.sh & ./fstime.sh

Observations

The 5 processes run in shifts. Also, since modifying files is a system call, it has a higher quantum.

Inference

Tasks don't use their quanta to the fullest as they don't consume much CPU.

```
TIME|6.4
TIME|6.4
Time Quantum: 500, Time Executed: 185
Time Quantum: 500, Time Executed: 500
Time Quantum: 500, Time Executed: 135
Time Quantum: 500, Time Executed: 500
Read done: 1000004 in 6.2500, score 40000
Read done: 1000004 in 6.2500, score 40000
Read done: 1000004 in 6.2500, score 40000
Read done: 1000004 in 6.2500, score 40000
Read done: 1000004 in 6.2500, score 40000
COUNT|40000|0|KBps
COUNT|40000|0|KBps
COUNT|40000|0|KBps
COUNT|40000|0|KBps
COUNT|40000|0|KBps
TIME|6.2
TIME|6.2
TIME|6.2
TIME|6.2
TIME|6.2
```

PART 2 - Modified pseudo-FIFO Scheduler:

1. Workload Mix 1:

Here we executed the following command of 5 task,

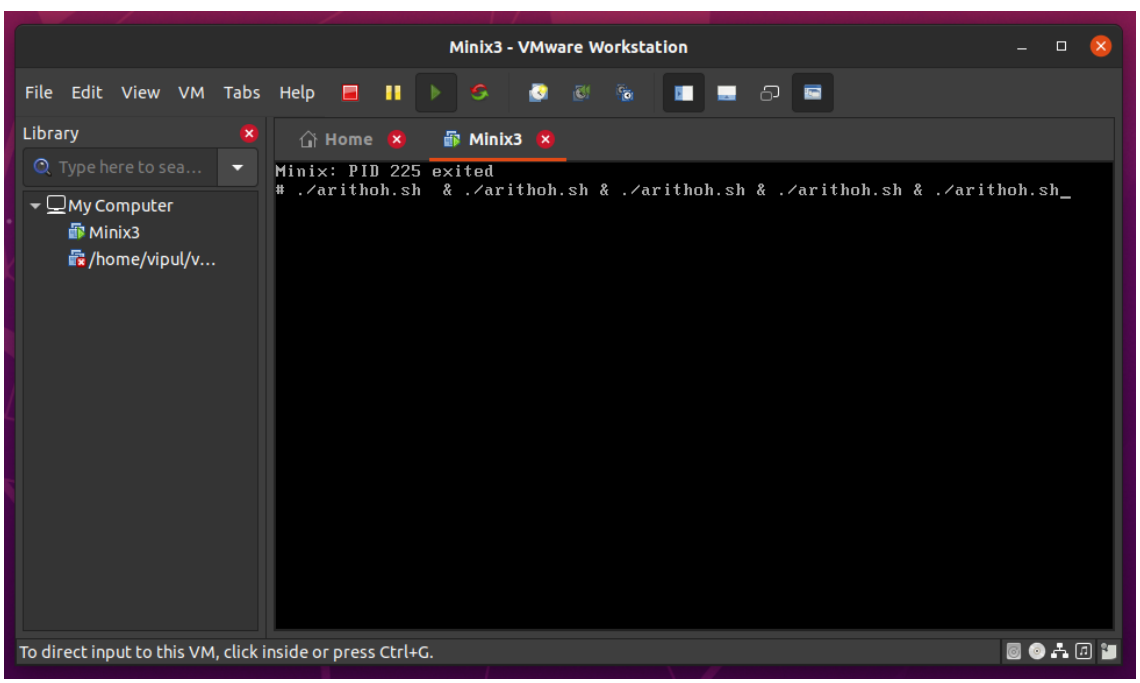
./arithoh.sh & ./arithoh.sh & ./arithoh.sh & ./arithoh.sh & ./arithoh.sh

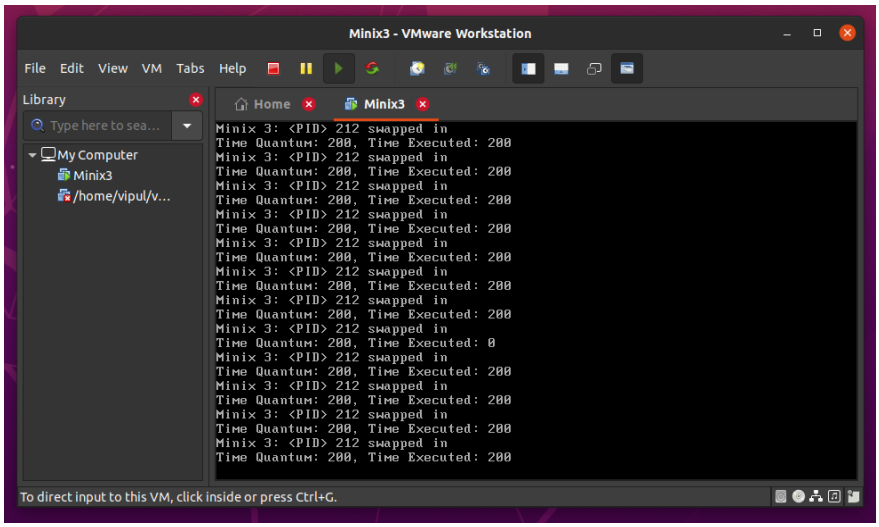
Observations

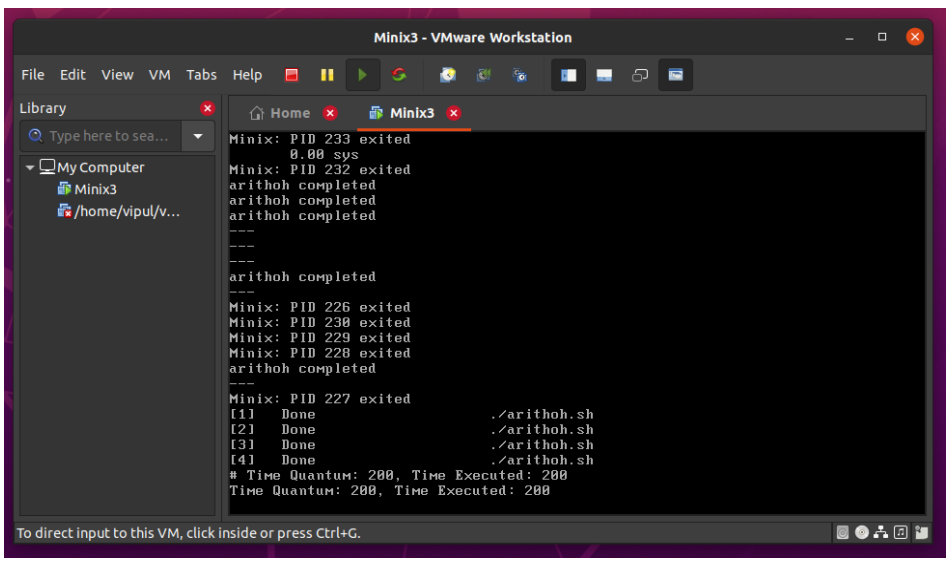
We see that the 5 processes are executed one after another in order of arrival. The time account, which is 200 by default, is used in full each time the process is scheduled. Every now and then, we see a process allocated 500 ms quanta with zero execution time. Most likely it is the execution of a system call.

Inference

The CPU fairly schedules all processes in order of arrival. CPU intensive processes do not need to wait for I / O and run completely in the allotted quanta of time.







2. Workload Mix 2:

Here we executed the following command of 5 task,

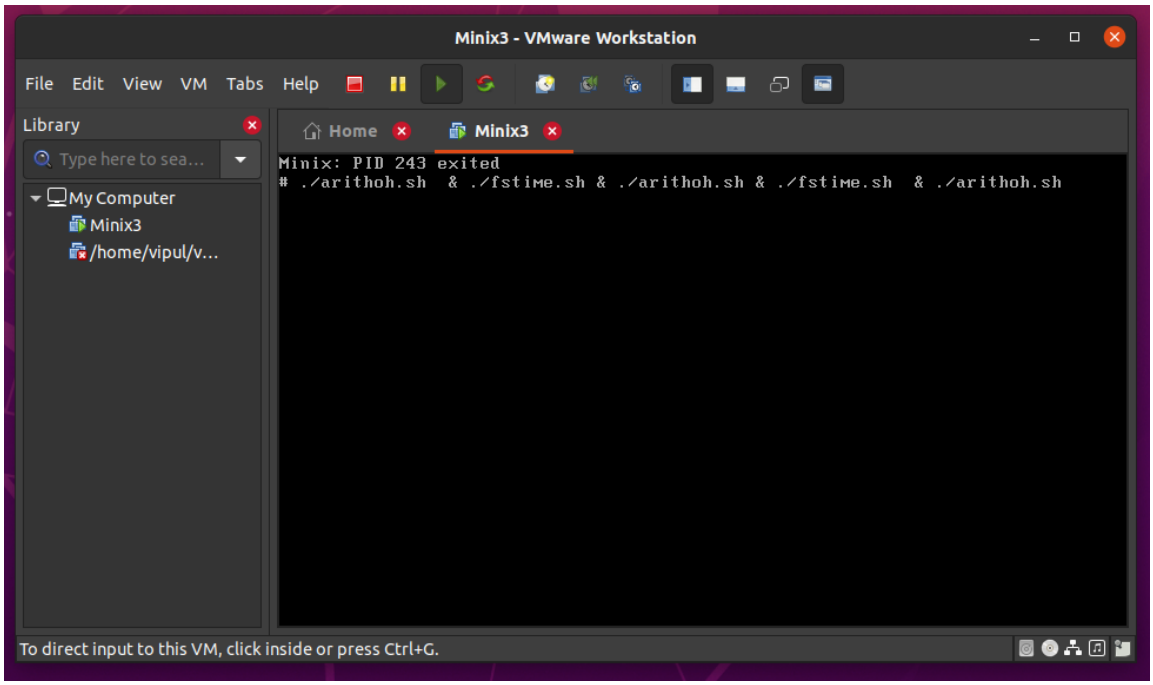
./arithoh.sh & ./fstime.sh & ./arithoh.sh & ./fstime.sh & ./arithoh.sh

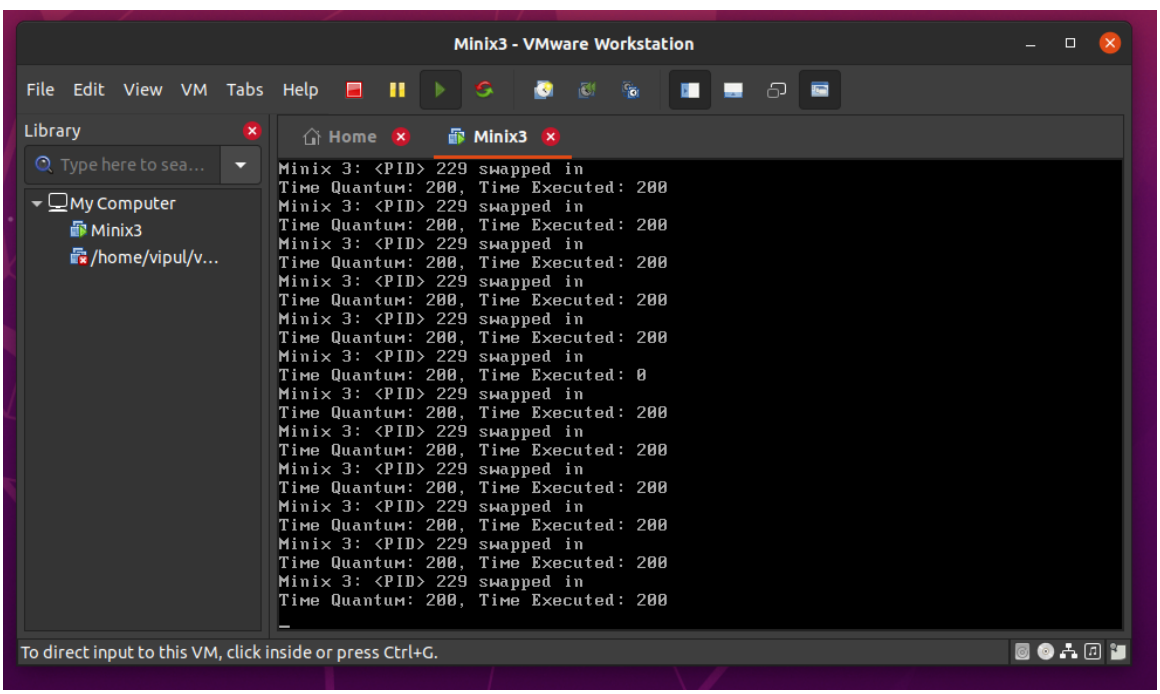
Observations

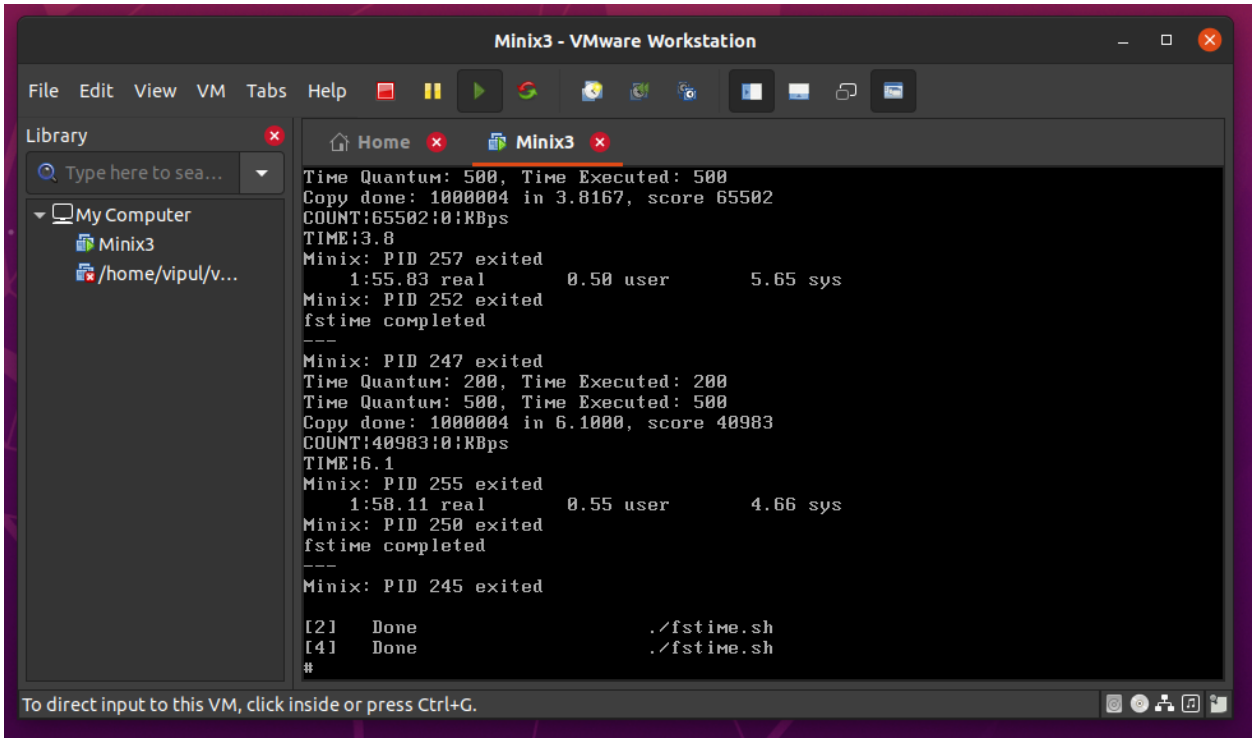
We see that the 3 CPUs tend to complete before the 2 linked I/O processes that are not perfectly in the FCFS order. However this is due to the fact that the 2 I/O linked processes are queued after requesting I/O and then put back in the ready queue and scheduled to run on the CPU.

Inference:

CPU-bound tasks use all of their quanta, while I/O-bound tasks don't always do the same.







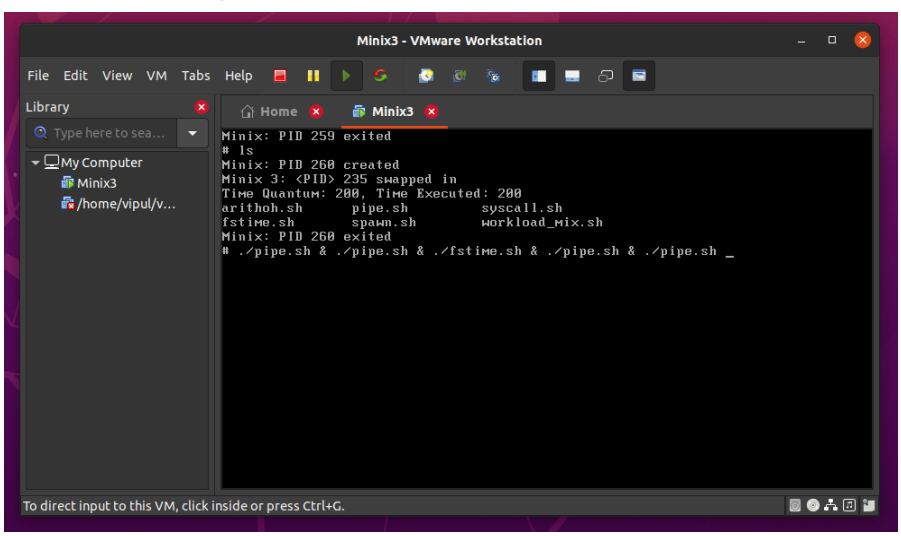
3. Workload Mix 3:

Here we executed the following command of 5 task,

./pipe.sh & ./pipe.sh & ./fstime.sh & ./pipe.sh & ./pipe.s○

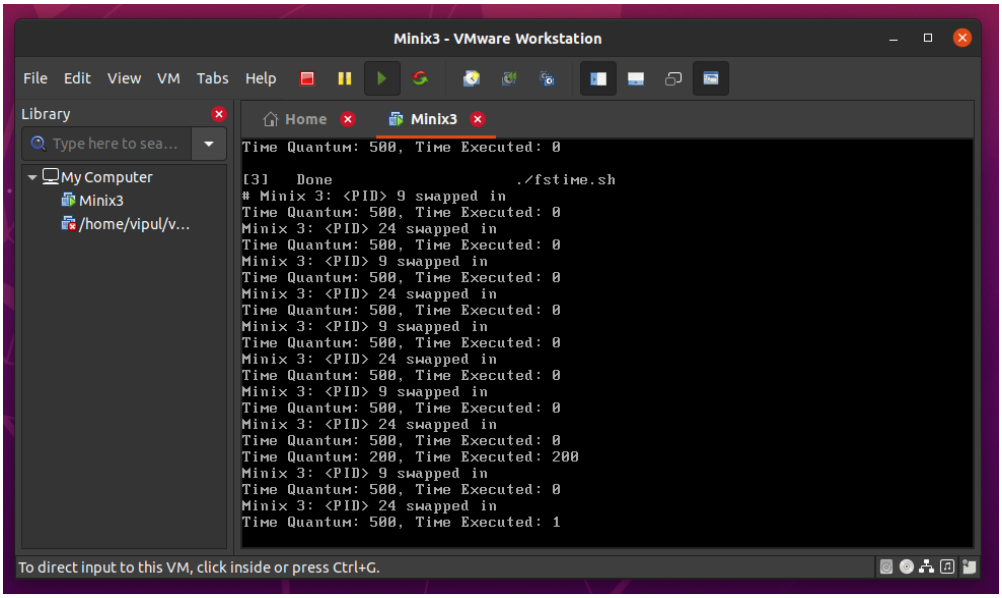
Observations

Since pipe.sh involves IPC, it invokes system calls and therefore gets a quantum greater than 500 ms for system calls. The processes run almost the FCFS way.



Inference:

The 2 linked I / O processes are queued after requesting input and finally run after being scheduled back to the ready queue.



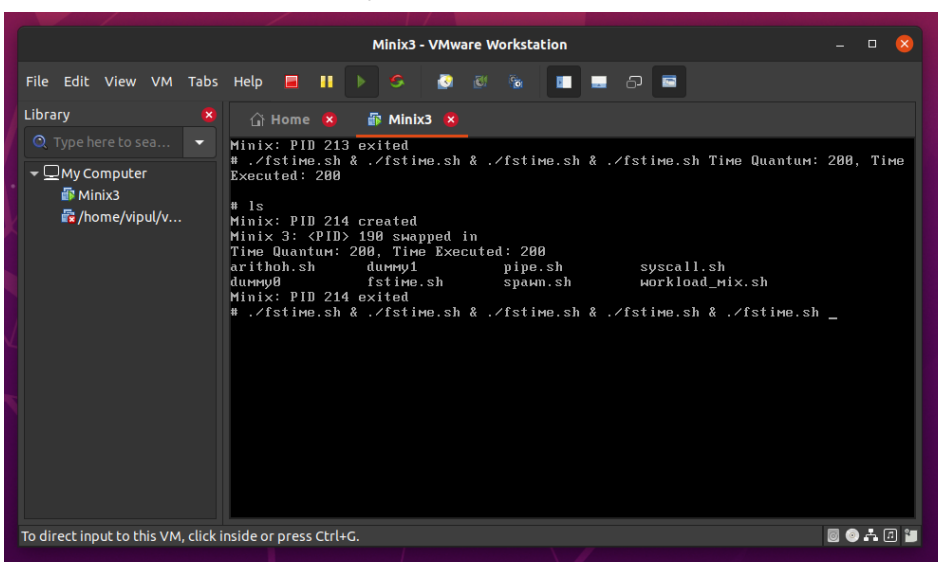
4. Workload Mix 4:

Here we executed the following command of 5 task,

./fstime.sh & ./fstime.sh & ./fstime.sh & ./fstime.sh & ./fstime.sh

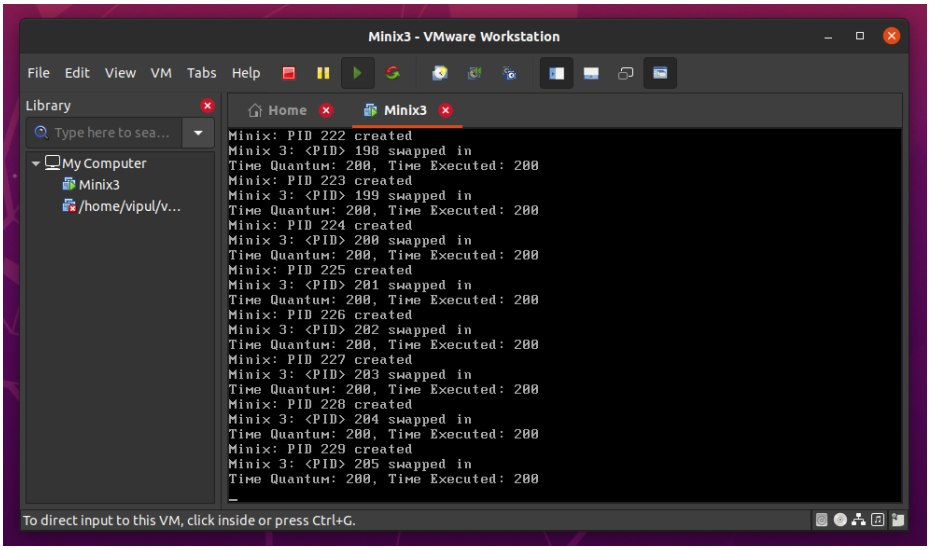
Observations

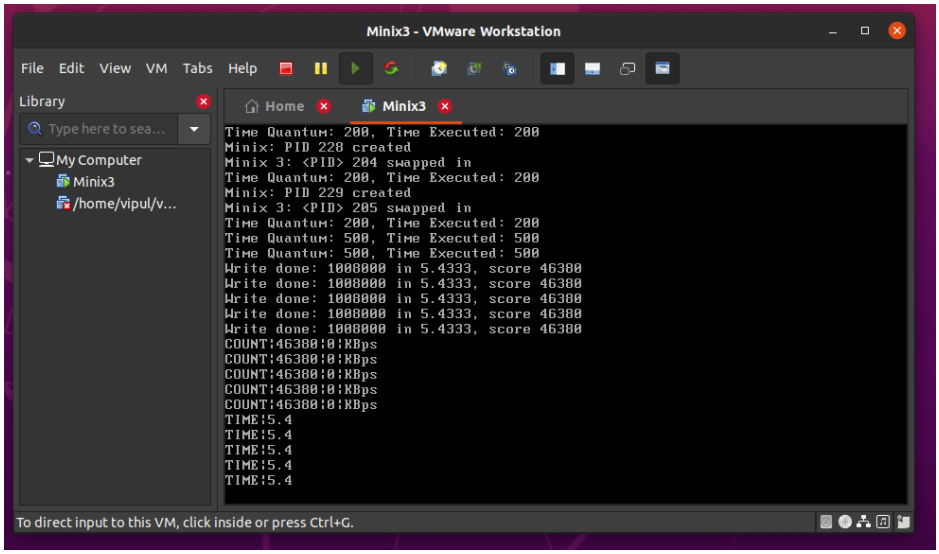
All five processes run FCFS. Here, the quanta assigned to each process are not exactly 500 and are little less than that.



Inference:

This is because it does not require the full quanta when the process is about to complete.





XXX