**Operating Systems Laboratory Assignment 5**

-Nithin Sabu, 210010032

**Part 1:**

1. workload\_mix1.sh (Completely CPU intensive)

#!/bin/bash

./arithoh.sh &

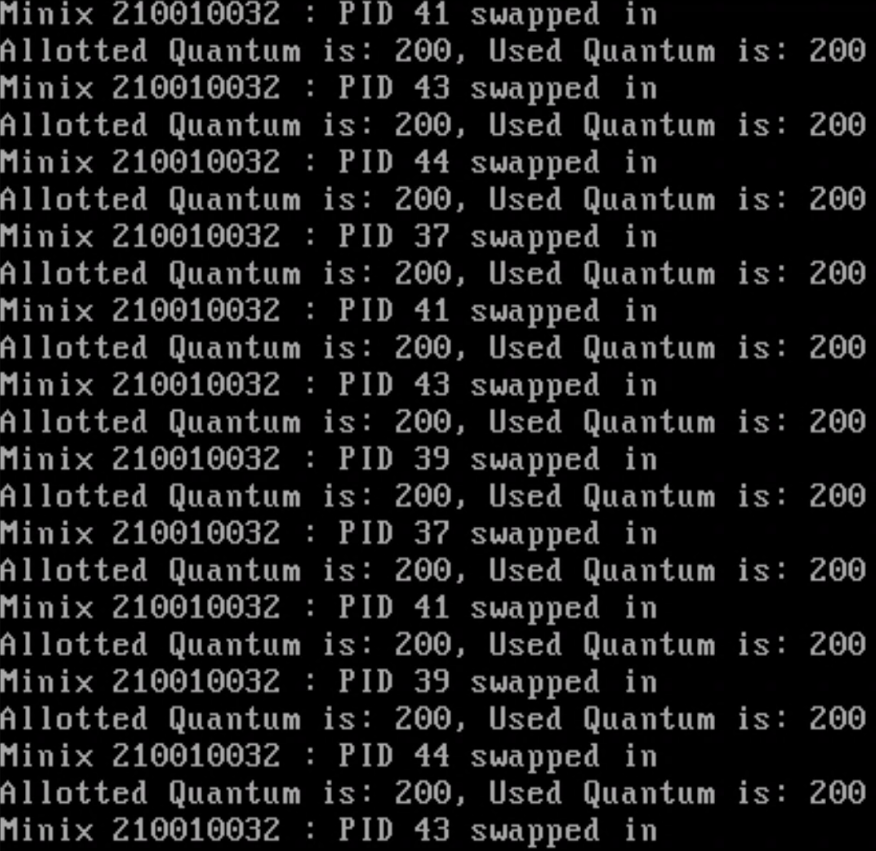
./arithoh.sh &

./arithoh.sh &

./arithoh.sh &

./arithoh.sh &

wait



The above screenshot shows the Processes being executed in a round-robin fashion, each arithoh process using the whole time quanta, 200 allotted, since it’s a completely CPU bound process.

1. workload\_mix2.sh (Completely I/O intensive)

#!/bin/bash

./fstime.sh &

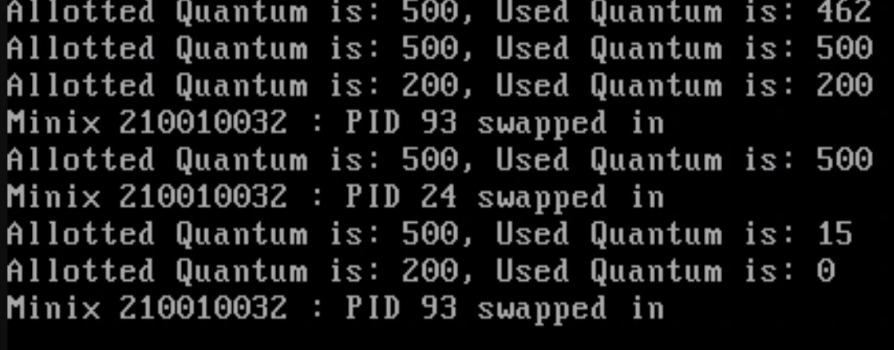
./fstime.sh &

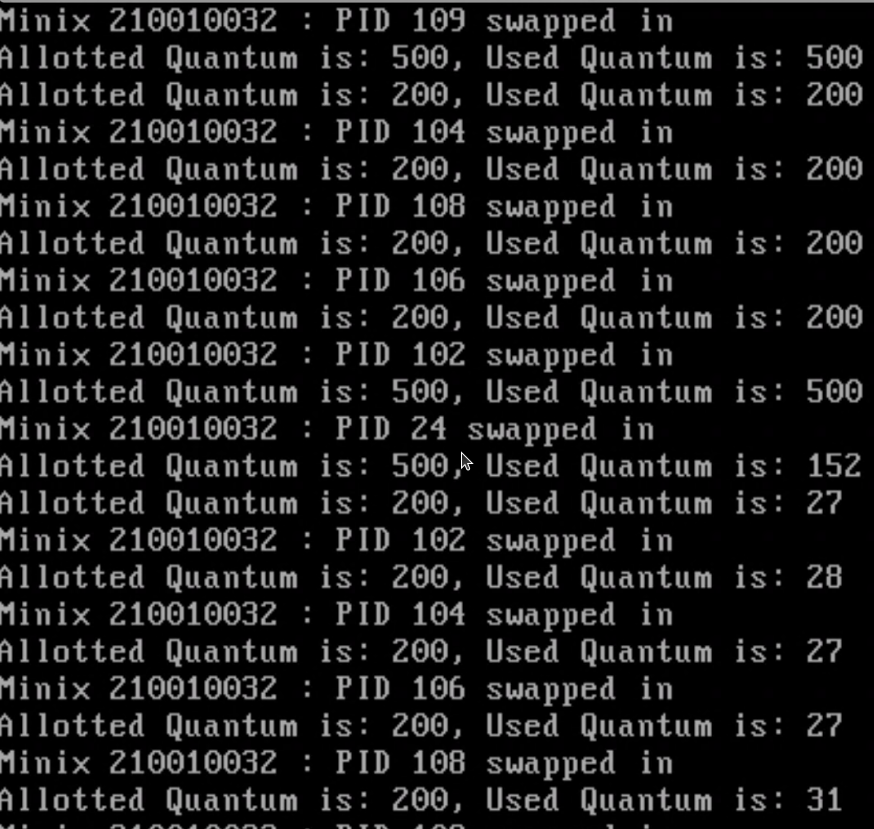
./fstime.sh &

./fstime.sh &

./fstime.sh &

wait





From above screenshots, the processes are running in a round robin fashion. Sometimes, fstime is allotted a time quanta of 500, since it’s more I/O bound. It is also to be noted that there are observable pauses in between.

1. workload\_mix3.sh (only system calls)

#!/bin/bash

./syscall.sh &

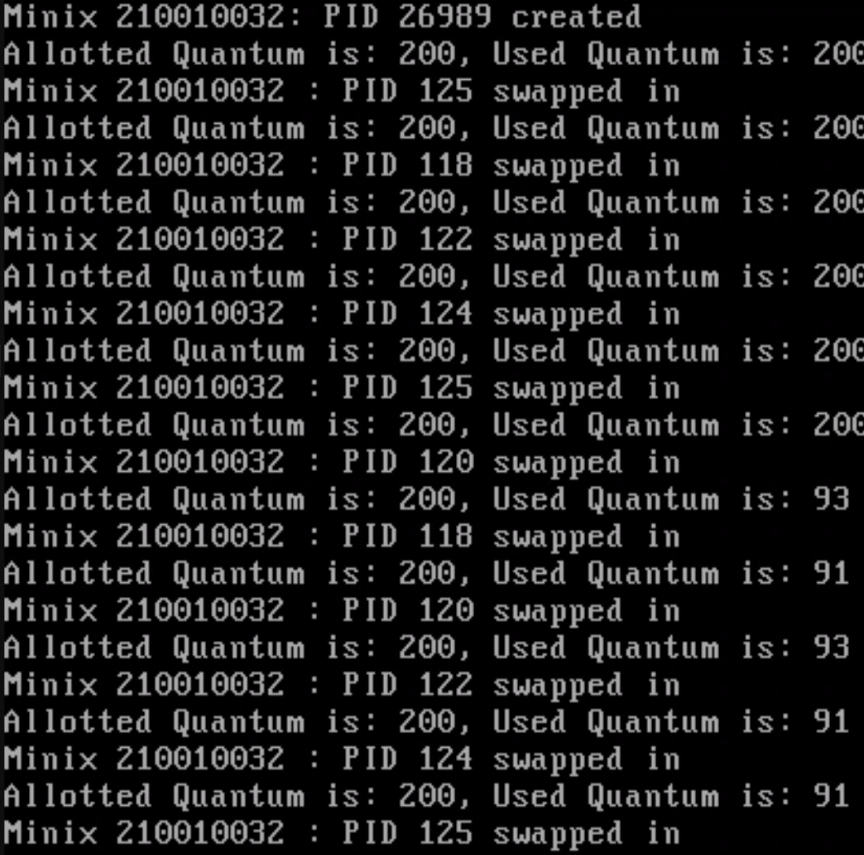
./syscall.sh &

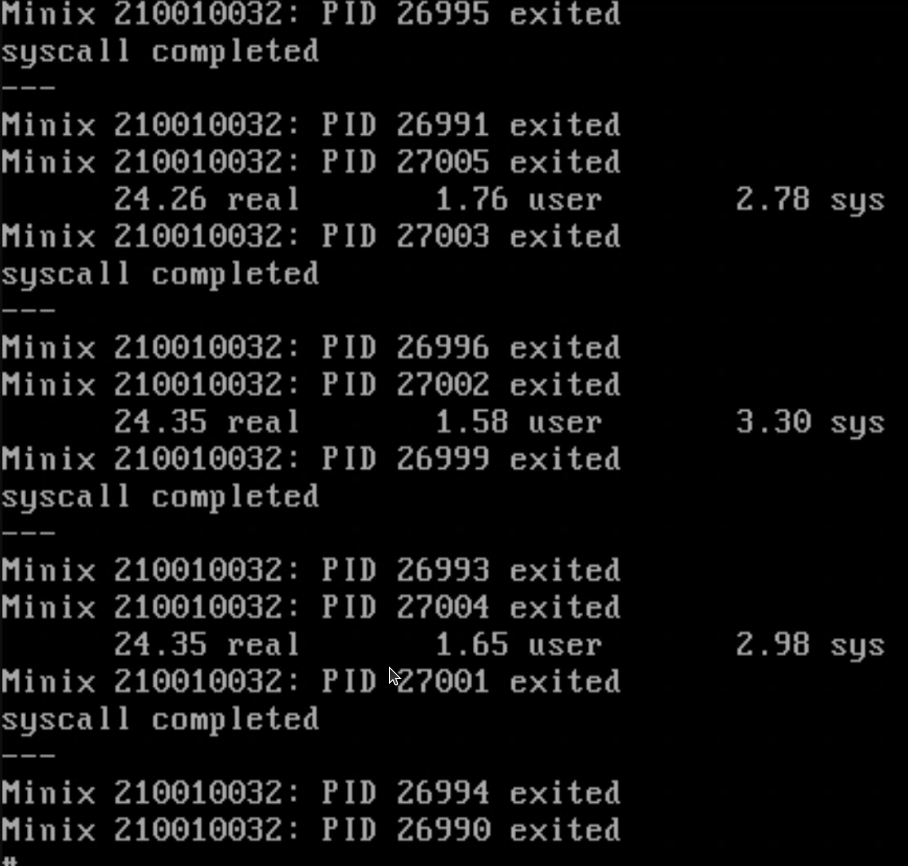
./syscall.sh &

./syscall.sh &

./syscall.sh &

wait





This workload mix has lot of system calls involved. Hence many times, it doesn’t use the complete time quanta allotted to it as observed in above screenshots, which indicates that it is less CPU intensive than arithoh.

1. workload\_mix4.sh (mix of arithoh, fstime and syscall)

#!/bin/bash

./arithoh.sh &

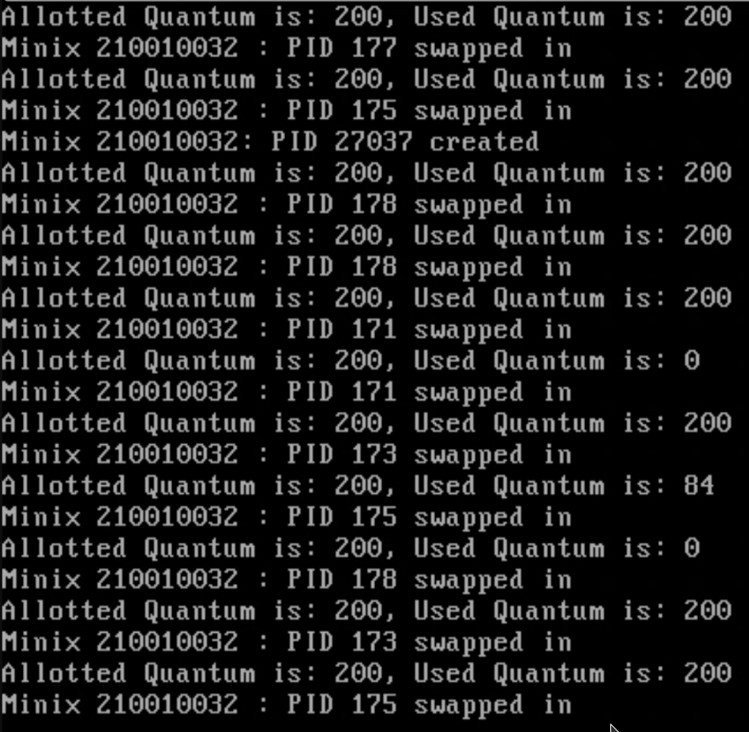
./arithoh.sh &

./arithoh.sh &

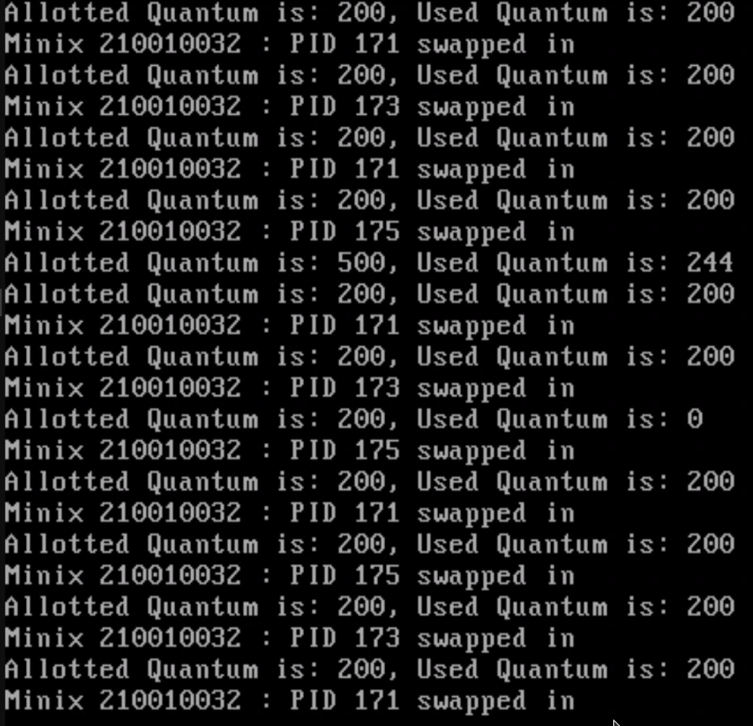
./fstime.sh &

./syscall.sh &

wait



After fstime and syscall is executed:



We observe that syscall is completed first, followed by fstime and then the three arithoh processes execute. Sometimes, fstime gets time quanta of 500.

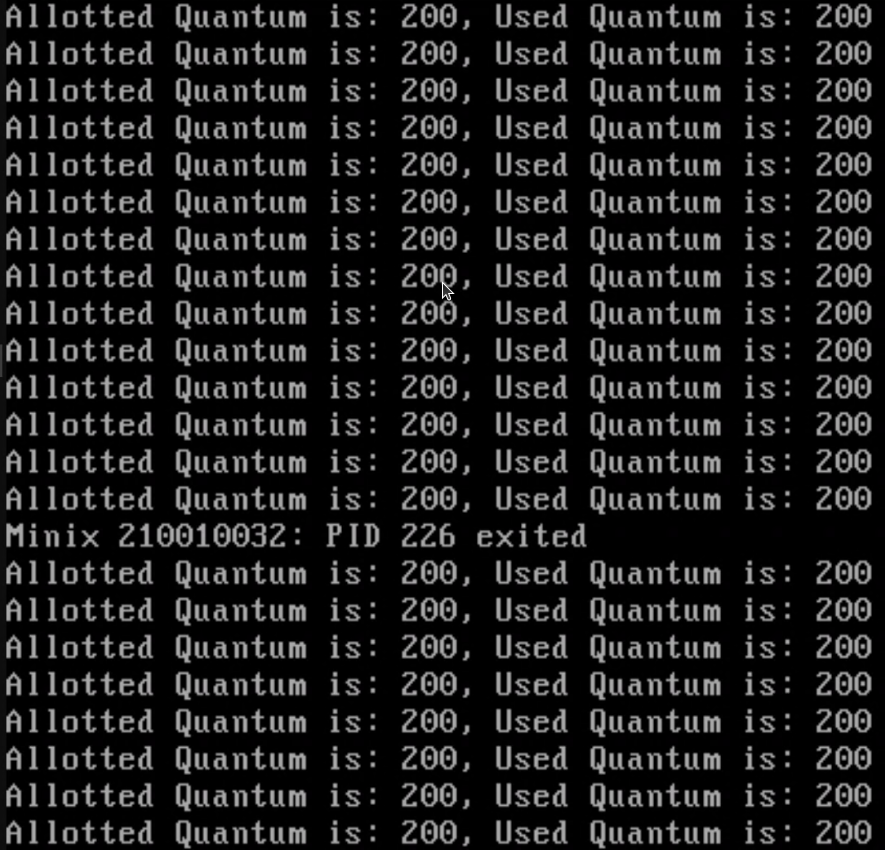
**Part 2:**

To make a pseudo FIFO scheduler, the following changes were made:

In minix/servers/sched/schedule.c, function do\_noquantum() had a line rmp->priority += 1; which was changed to rmp->priority -= 1. This makes the scheduler to decrease the priority of the new incoming processes to pre emptively get time slices.

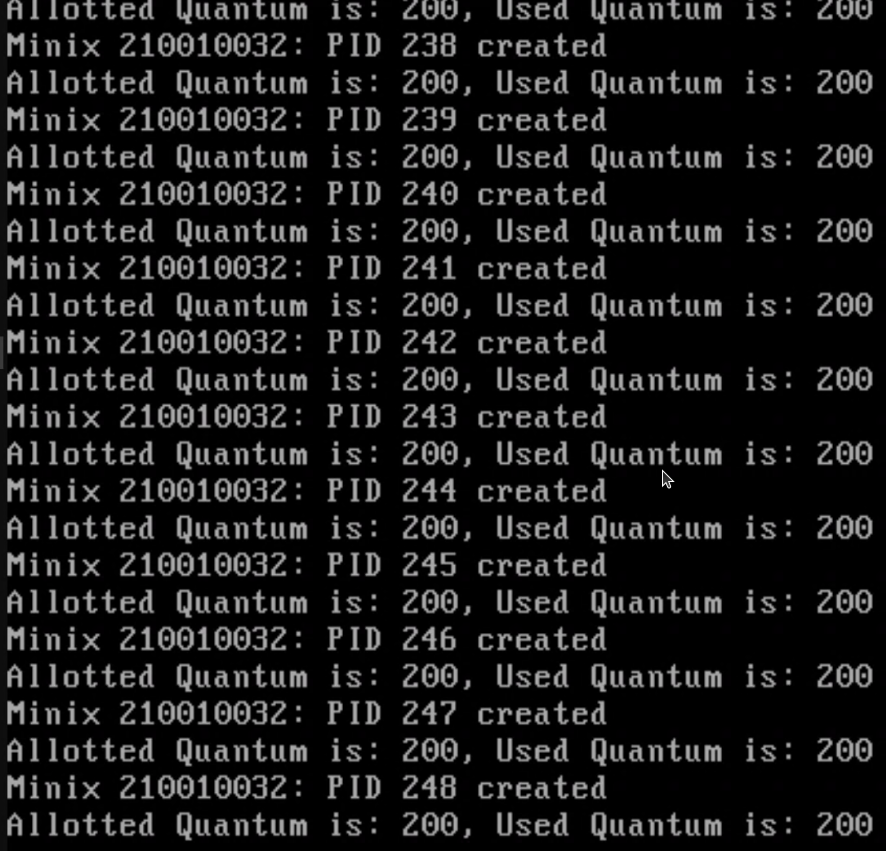
Also the function balance\_queues() in the same file was changed by commenting out the rmp->priority -=1; line to not increase the priority of processes, to prevent overflow of the priority queue.

1. workload1-mix.sh



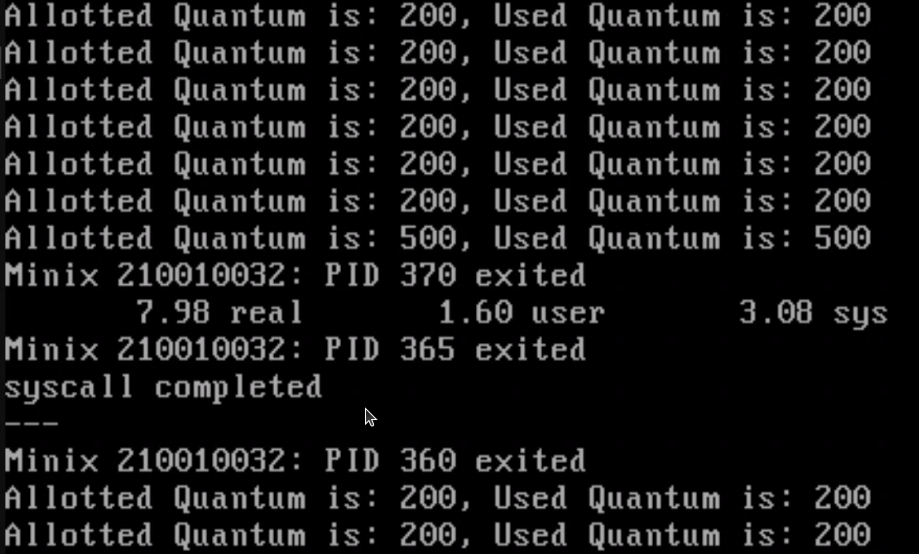
Since all arithoh are CPU intensive, the processes are executed sequentially, hence the processes are not swapped in until the process in ready queue is completed.

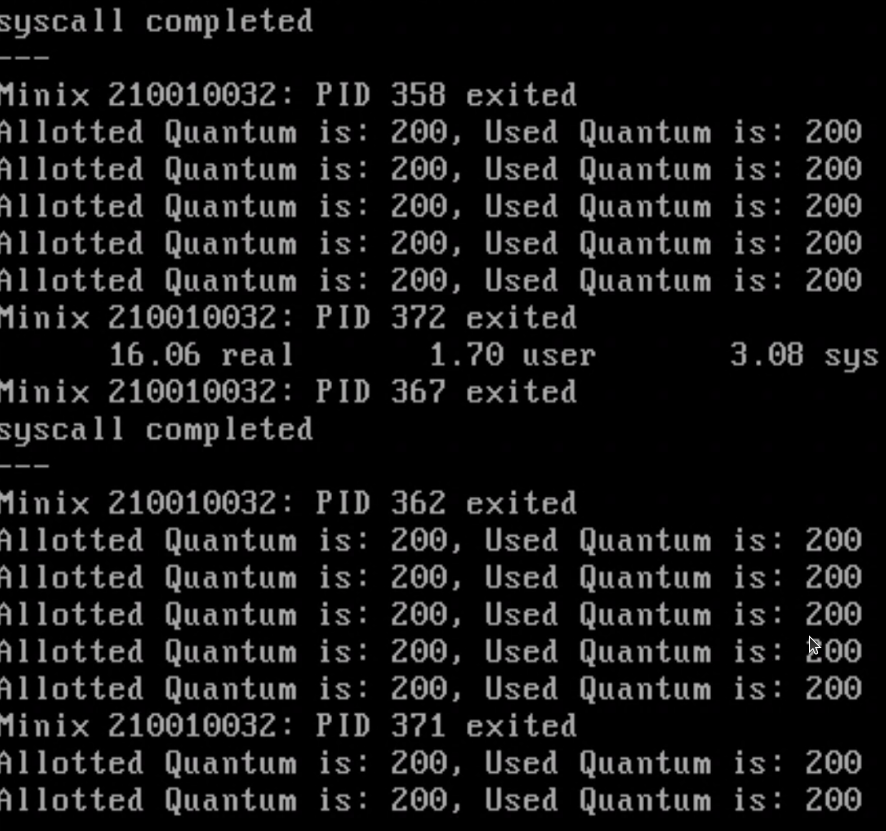
1. workload\_mix2.sh



Here, the scheduling resembles round robin, since fstime is mostly I/O bound. Because, When a fstime process is waiting for I/O, another fstime process begins execution. Following FIFO will be inefficient in this case since CPU will be idle if the second process waits till completion of first process.

1. workload\_mix3.sh

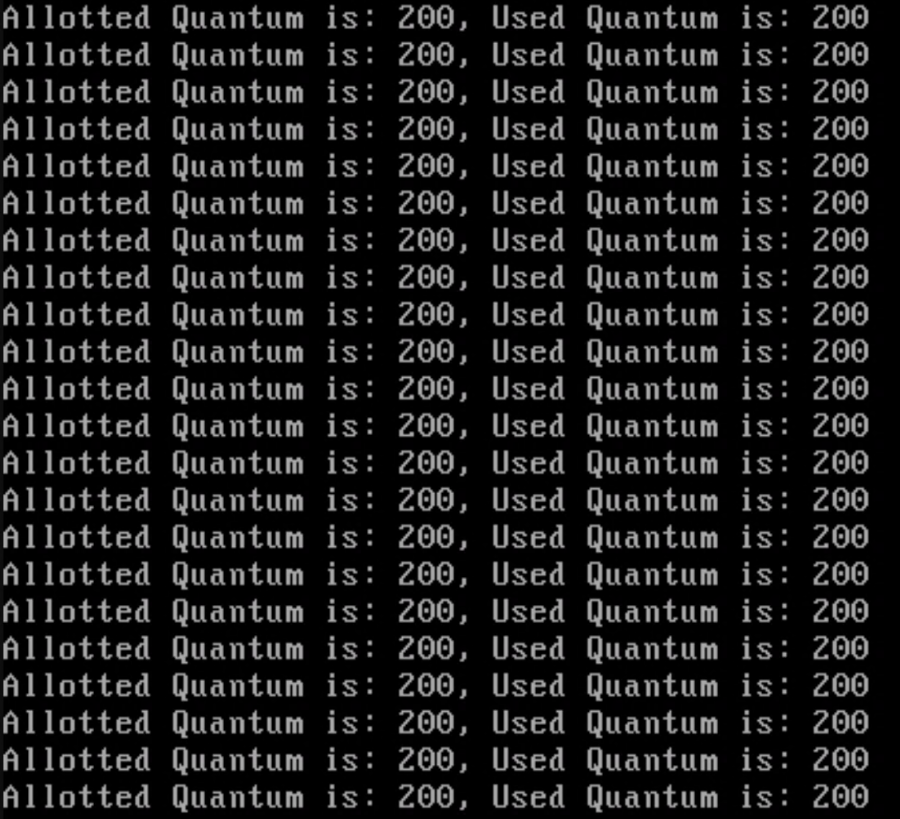




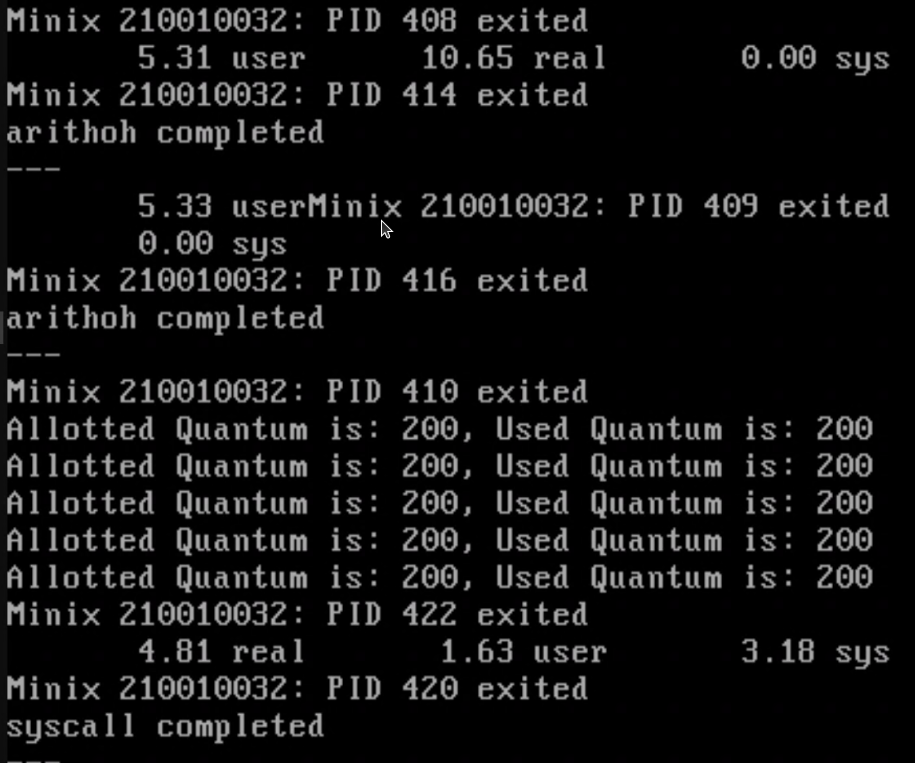
We observe that since syscall is CPU intensive (but not as much as aritho), hence it follows FIFO and not the round robin we observe when fstime was executed multiple times.

1. workload4.sh

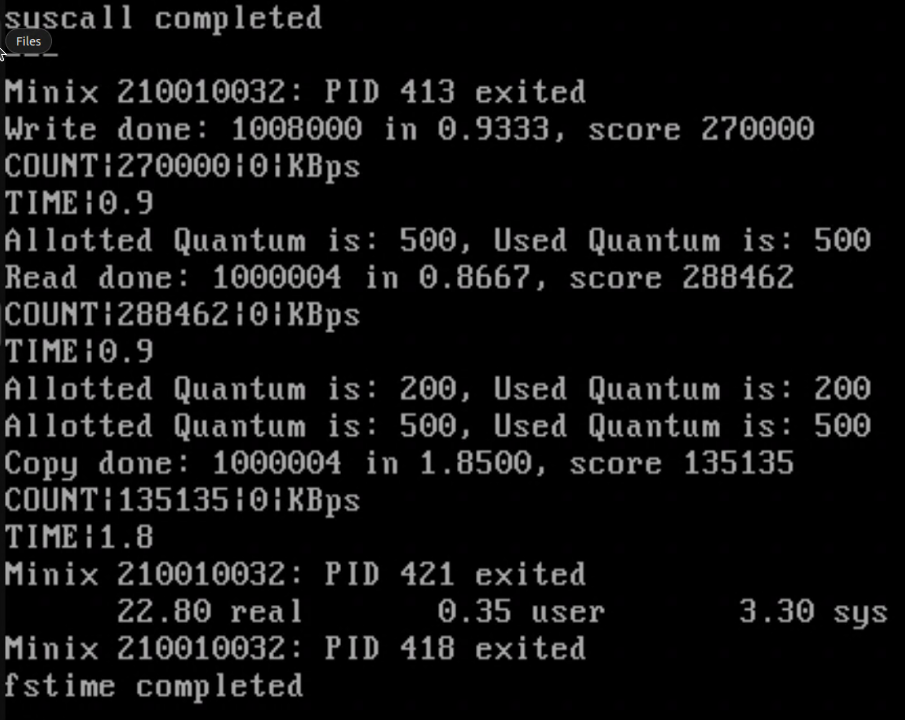
execution of arithoh:



After arithoh is completed:



After syscall is completed:



We observe that arithoh gets executed in a FIFO manner first, (all three of arithoh), then syscall is executed followed by fstime. It is observed that CPU first executed the more CPU bound processes first (arithoh), then less CPU bound (syscall) and then least CPU bound (fstime). This is because even if fstime enters queue first, it gets removed since it waits for I/O and more CPU bound process such as aritho gets inside queue and stays until it completes.