CS 314 – Operating Systems Lab

180010008

Lab-4

Part1 →

Workload mix 1:

workload_mix1.sh

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

```

```
Quantum Alloted: 200, Quantum Used: 200
PID 126 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 128 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 124 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 125 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 126 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 128 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 127 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 125 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 126 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 124 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 128 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 127 swapped in
```

Here, there are 5 arithoh.sh processes with pids: 124, 125, 126, 127, 128.

```
# ./workload_mix1.sh
    1:41.35 real      22.05 user       0.00 sys
arithoh completed
---
    1:45.40 real      22.11 user       0.00 sys
arithoh completed
---
    1:46.26 real      22.03 user       0.00 sys
arithoh completed
---
    1:49.91 real      22.33 user       0.00 sys
arithoh completed
---
    1:50.68 real      22.15 user       0.00 sys
arithoh completed
---
# _
```

Observations: We see that the 5 processes execute parallelly in round-robin fashion. The time quanta which is 200 by default is fully used every time the process is scheduled.

Inference: CPU fairly schedules all the processes in a round-robin fashion. CPU intensive processes do not need to wait for I/O and they execute fully in the time quanta they are allocated.

Here, all 5 CPU Intensive processes finish in almost same time as they run fairly in round robin fashion while fully utilizing their assigned quantum as they do not wait for any I/O.

Workload mix 2:

workload_mix2.sh

```
1  #!/bin/sh
2  ./syscall.sh &
3  ./syscall.sh &
4  ./syscall.sh &
5  ./syscall.sh &
6  ./syscall.sh &
7  wait
8
```

```
Quantum Alloted: 200, Quantum Used: 9
PID 145 swapped in
Quantum Alloted: 200, Quantum Used: 9
PID 146 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 146 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 145 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 140 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 143 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 141 swapped in
Quantum Alloted: 200, Quantum Used: 11
PID 140 swapped in
Quantum Alloted: 200, Quantum Used: 11
PID 141 swapped in
Quantum Alloted: 200, Quantum Used: 11
PID 143 swapped in
Quantum Alloted: 200, Quantum Used: 10
PID 145 swapped in
Quantum Alloted: 200, Quantum Used: 10
PID 146 swapped in
```

Here, there are 5 syscall.sh processes.

```
# ./workload_mix2.sh
      45.08 real       3.00 user       6.05 sys
syscall completed
---
      45.23 real       2.65 user       5.95 sys
syscall completed
---
      45.26 real       2.55 user       6.58 sys
syscall completed
---
      45.30 real       3.13 user       6.25 sys
syscall completed
---
      45.36 real       2.85 user       6.35 sys
syscall completed
---
```

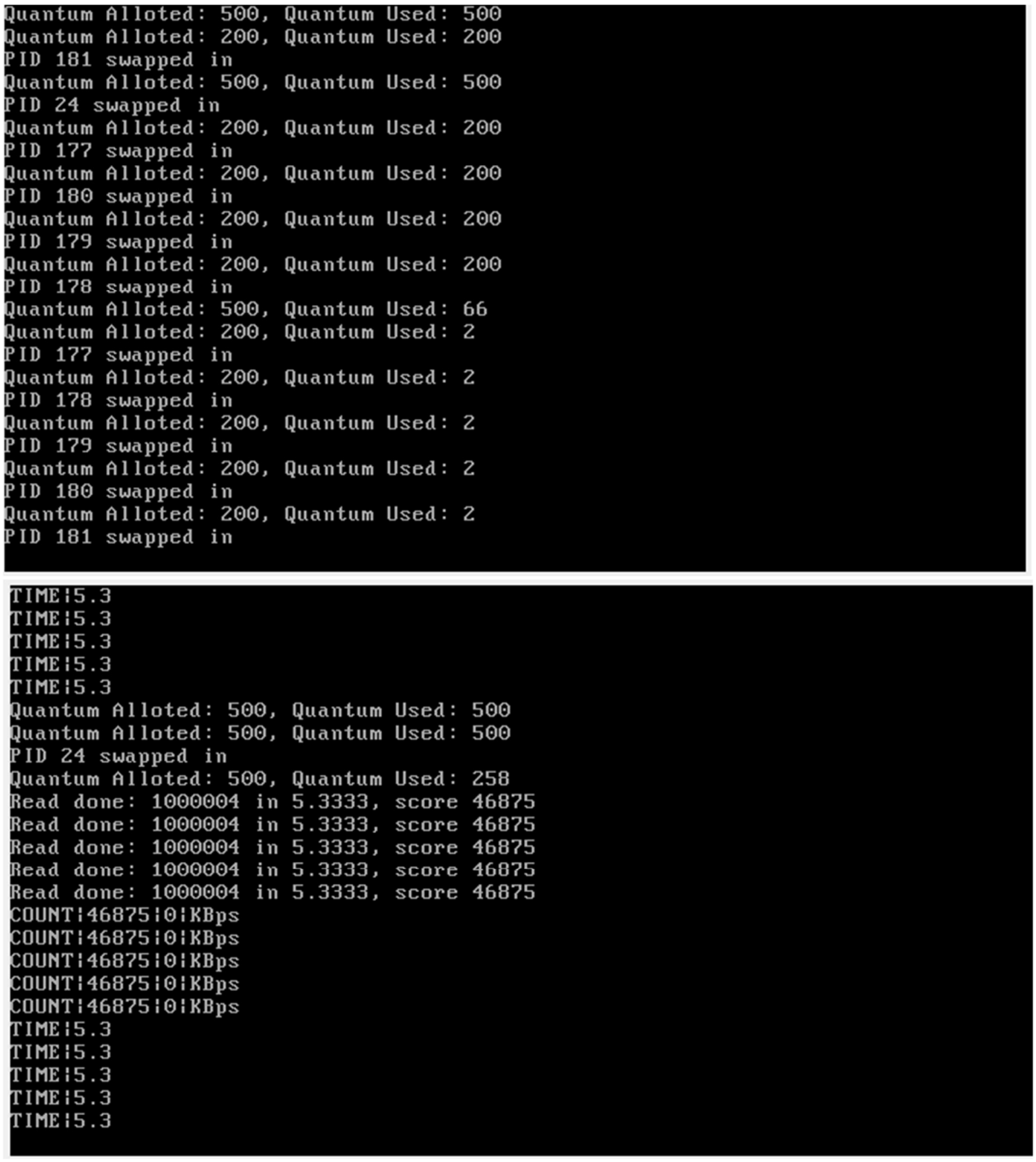
Observations: We see that the 5 processes execute parallelly in round-robin fashion. The time quanta which is 200 by default is not fully used every time the process is scheduled as syscall is less CPU Intensive as compared to arithoh.sh.

Inference: CPU fairly schedules all the processes in a round-robin fashion. CPU intensive processes do not need to wait for I/O and they execute fully in the time quanta they are allocated.

Workload mix 3:

workload_mix3.sh

```
1  #!/bin/sh
2  ./fstime.sh &
3  ./fstime.sh &
4  ./fstime.sh &
5  ./fstime.sh &
6  ./fstime.sh &
7  wait
8
```



Here all 5 processes are I/O bound processes which wait for I/O before they can complete.

Observations & Inference: We see that the 5 I/O bound processes execute parallelly in round-robin fashion. However, they wait for their I/O work and then are scheduled to work on the CPU. As seen in the screenshot, I/O bound tasks do not utilize all allotted quanta.

```
TIME:9.7
      37.16 real         0.68 user         5.71 sys
fstime completed
---
      38.25 real         0.66 user         6.80 sys
fstime completed
---
Copy done: 1000004 in 17.1333, score 14591
COUNT:14591:0:KBps
TIME:17.1
      44.50 real         0.60 user         6.83 sys
fstime completed
---
Copy done: 1000004 in 17.1500, score 14577
COUNT:14577:0:KBps
TIME:17.2
      44.51 real         0.71 user         6.08 sys
fstime completed
---
Copy done: 1000004 in 17.1833, score 14549
COUNT:14549:0:KBps
TIME:17.2
      44.55 real         0.65 user         5.01 sys
fstime completed
---
```

Here, we can see that all processes wait until they receive I/O and then all complete their operations.

Workload mix 4:

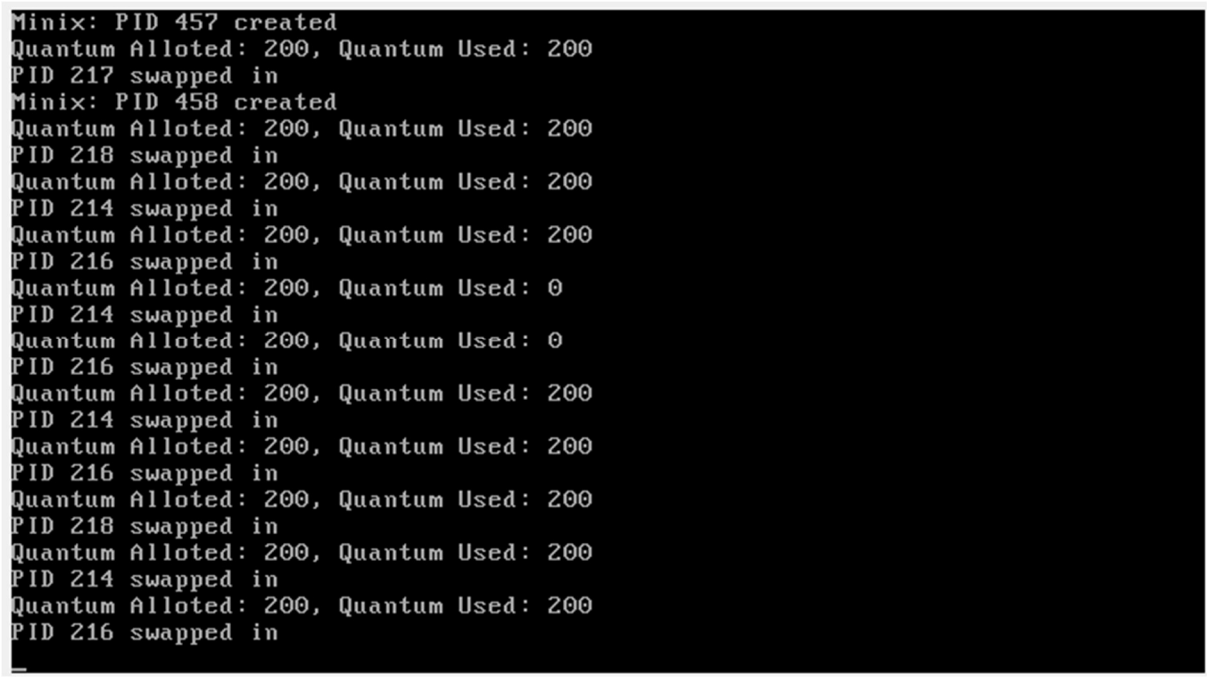
workload_mix4.sh

```
1  #!/bin/sh
2  ./arithoh.sh &
3  ./fstime.sh &
4  ./arithoh.sh &
5  ./fstime.sh &
6  ./syscall.sh &
7  wait
8
```

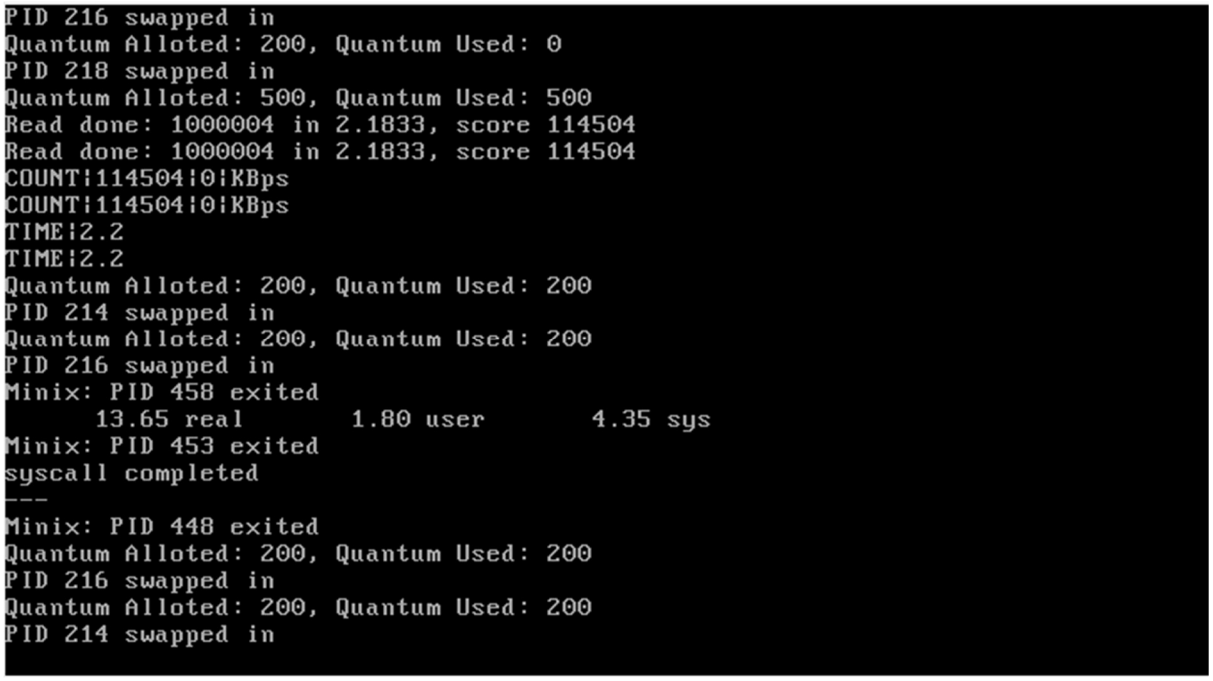
We can see that above workload mix consists of: arithoh.sh (CPU Intensive), fstime.sh (I/O Bound) and syscall.sh (System CPU Intensive).

PID 214: arithoh.sh, PID 215: fstime.sh, PID 216: arithoh.sh,

PID 217: fstime.sh, PID 218: syscall.sh



Here, we see that 214, 216 and 218 are scheduled in a round robin manner whereas 215 and 217 wait for I/O as they are I/O bound processes.



Here, syscall is a CPU Intensive Process which completely utilizes its allotted quanta and finishes its task and exits.

```
PID 215 swapped in
Quantum Alloted: 500, Quantum Used: 500
Copy done: 1000004 in 3.6667, score 68182
COUNT¦68182¦0¦KBps
TIME¦3.7
Minix: PID 455 exited
      19.61 real       0.50 user       3.88 sys
Minix: PID 450 exited
fstime completed
---
Minix: PID 445 exited
Copy done: 1000004 in 4.2333, score 59055
COUNT¦59055¦0¦KBps
TIME¦4.2
Minix: PID 457 exited
      20.16 real       0.33 user       3.90 sys
Minix: PID 452 exited
fstime completed
---
Minix: PID 447 exited
Quantum Alloted: 200, Quantum Used: 200
PID 214 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 216 swapped in
```

Here, fstime.sh processes with pid 215 and 217 is allotted more quanta than a CPU Intensive task, i.e, allotted 500 quanta and doesn't utilize all quanta every time.

```
Quantum Alloted: 200, Quantum Used: 200
PID 216 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 214 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 216 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 216 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 214 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 216 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 216 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 214 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 216 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 214 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 216 swapped in
Quantum Alloted: 200, Quantum Used: 200
PID 214 swapped in
```

After all other processes are completed and then both arithoh.sh with pid 214 and 216 are scheduled in round robin manner till they are completed.

```
TIME¦5.9
      23.55 real         0.36 user         5.76 sys
fstime completed
---
Copy done: 1000004 in 6.3333, score 39473
COUNT¦39473¦0¦KBps
TIME¦6.3
      23.98 real         0.56 user         5.83 sys
fstime completed
---
      24.73 real         2.56 user         6.00 sys
syscall completed
---

    1:14.45 real       27.10 user         0.06 sys
arithoh completed
---
    1:15.63 real       27.36 user         0.00 sys
arithoh completed
---
```

Here is the time taken by each process to complete.

More CPU Intensive processes take more time where as I/O bound processes wait for I/O to be completed.

Part2 →

Minix follows **Round Robin Scheduling** by default which was changed to **FIFO** by following modifications.

1. Do not lower priority in "**do_noquantum**" function that is responsible to be called on behalf of process' that run out of quantum.

```
if (rmp->priority < MIN_USER_Q) {
    rmp->priority -= 1;
}
```

2. Do not change priority or balance queues in "**balance_queues**" function that is responsible for balancing queues to change priority when they utilize their quantum fully.

```
if (rmp->priority > rmp->max_priority) {
    // rmp->priority -= 1; /* increase priority */
    schedule_process_local(rmp);
}
```

Workload mix 1:

workload_mix1.sh

```
1  #!/bin/sh
2  ./arithoh.sh &
3  ./arithoh.sh &
4  ./arithoh.sh &
5  ./arithoh.sh &
6  ./arithoh.sh &
7  wait
8
```

```
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
```

Observation and inference: We observed that 5 CPU intensive processes run one after the other in first come first serve fashion while fully utilizing their quantum of 200 as they do not wait for I/O.

Workload mix 2:

workload_mix2.sh

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

```
PID swapped in 143
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Minix: PID 391 exited
      9.90 real       1.85 user       4.33 sys
Minix: PID 386 exited
syscall completed
---
Minix: PID 381 exited
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Minix: PID 392 exited
     15.05 real       1.76 user       4.33 sys
Minix: PID 387 exited
syscall completed
---
Minix: PID 382 exited
Quantum Alloted: 200, Quantum Used: 200
```

Observation and inference: We observed that syscall is also a CPU intensive process that is less intensive than arithoh.sh. Here, all 5 CPU intensive processes run one after the other in first come first serve fashion while fully utilizing their quantum of 200 as they do not wait for I/O. Only when one process completes its execution, next process is scheduled.

Workload mix 3:

```
Minix: PID 425 created
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 179
Quantum Alloted: 500, Quantum Used: 1
PID swapped in 24
Quantum Alloted: 500, Quantum Used: 500
PID swapped in 24
Quantum Alloted: 500, Quantum Used: 298
PID swapped in 24
Write done: 1008000 in 5.7500, score 43826
Write done: 1008000 in 5.7500, score 43826
Write done: 1008000 in 5.7500, score 43826
Write done: 1008000 in 5.7500, score 43826
Write done: 1008000 in 5.7500, score 43826
COUNT|43826|0|KBps
COUNT|43826|0|KBps
COUNT|43826|0|KBps
COUNT|43826|0|KBps
COUNT|43826|0|KBps
TIME|5.8
TIME|5.8
TIME|5.8
TIME|5.8
TIME|5.8
```

Observation and Inference: We see that here FIFO order is not followed properly in I/O bound processes. It happens due to the fact that the I/O bound processes are sent to the waiting queue after requesting for I/O and are then placed back in the ready queue and scheduled to work on the CPU when I/O received. As seen in the screenshot, I/O bound tasks don't always utilize complete quanta allotted.

Workload mix 4:

workload_mix4.sh

```
1  #!/bin/sh
2  ./arithoh.sh &
3  ./fstime.sh &
4  ./arithoh.sh &
5  ./fstime.sh &
6  ./syscall.sh &
7  wait
8
```

```
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 126
Minix: PID 377 created
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 127
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 123
_
```

Here, arithoh.sh is continuously scheduled till its computation is completed as this is FIFO scheduling.

```
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 124
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 124
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 124
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 124
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 124
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 124
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 124
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 124
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 124
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 124
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 124
Quantum Alloted: 200, Quantum Used: 200
PID swapped in 124
_
```

After pid 123 (first arithoh.sh) was completed next arithoh.sh is continuously scheduled.

```
PID swapped in 35
Quantum Alloted: 500, Quantum Used: 500
PID swapped in 36
      31.63 real      31.63 real      15.81 user      15.81 user       0.00 sys
Minix: PID 368 exited
       0.00 sys
Minix: PID 370 exited
arithoh completed
---
Minix: PID 363 exited
arithoh completed
---
Minix: PID 365 exited
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
^[[DQuantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 200, Quantum Used: 200
Minix: PID 377 exited
      37.96 real       2.16 user       4.15 sys
Minix: PID 372 exited
syscall completed
---
Minix: PID 367 exited
```

Here, syscall is completed before fstime (which came earlier from syscall) because I/O bound processes are sent to the waiting queue after requesting for I/O and are then placed back in the ready queue and scheduled to work on the CPU. Thus, syscall is completed before fstime.

```
TIME¦2.3
Quantum Alloted: 500, Quantum Used: 500
Quantum Alloted: 200, Quantum Used: 200
Quantum Alloted: 500, Quantum Used: 500
Copy done: 1000004 in 3.3333, score 75000
COUNT¦75000¦0¦KBps
TIME¦3.3
Minix: PID 376 exited
      54.71 real       0.51 user       3.90 sys
Minix: PID 371 exited
fstime completed
---
Minix: PID 366 exited
Quantum Alloted: 200, Quantum Used: 200
Copy done: 1000004 in 4.7000, score 53191
COUNT¦53191¦0¦KBps
TIME¦4.7
Minix: PID 375 exited
      56.08 real       0.40 user       4.30 sys
Minix: PID 369 exited
fstime completed
---
Minix: PID 364 exited
Minix: PID 362 exited
# _
```

fstime completed at the end when I/O received.

Observations & Inference: We see that fstime came before syscall so in FIFO order, fstime was to be completed before syscall but however, this is due to

the fact that the I/O bound processes are sent to the waiting queue after requesting for I/O and are then placed back in the ready queue and scheduled to work on the CPU.