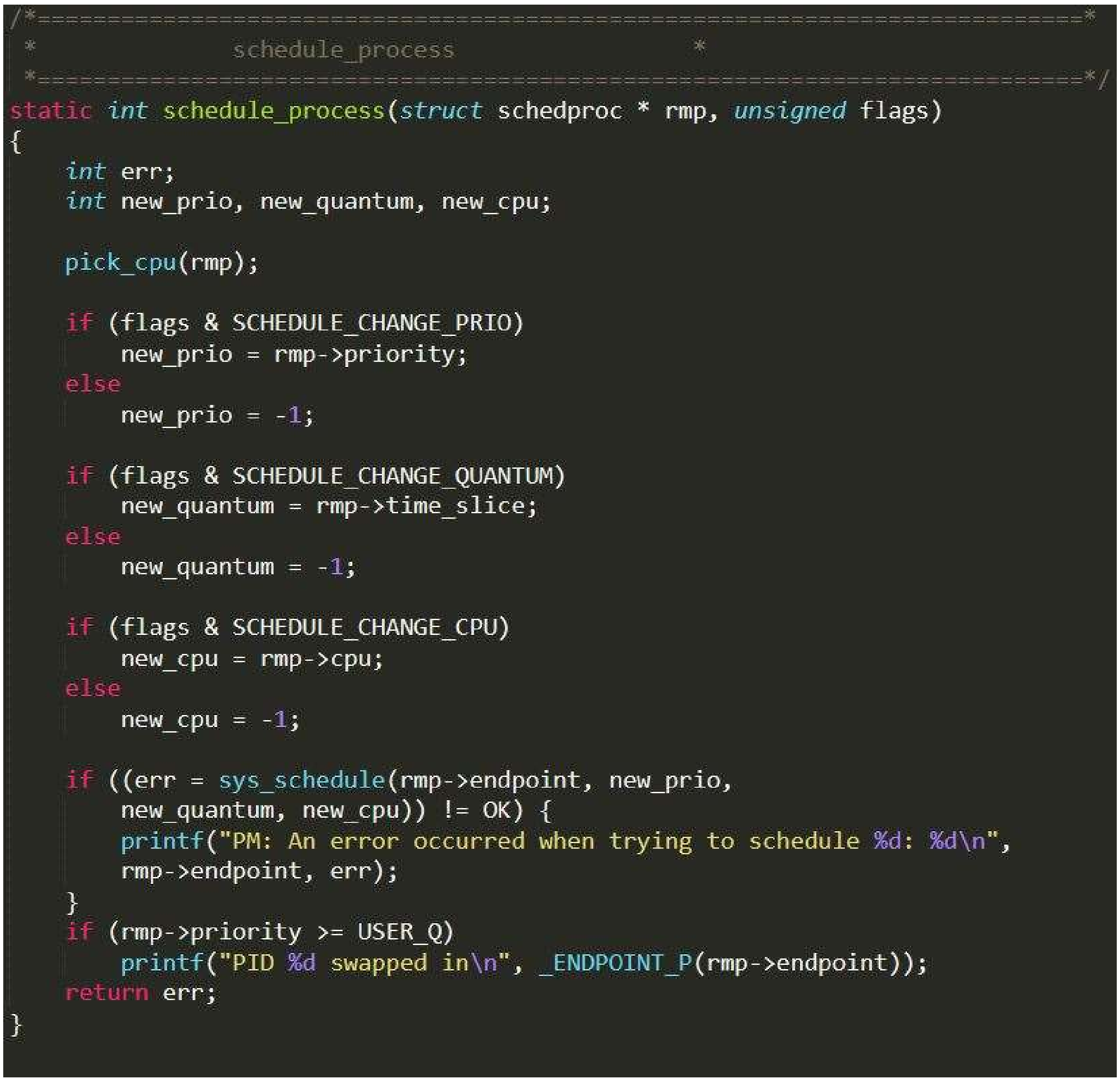
# CS 314 – Operating Systems Lab

# 180010008

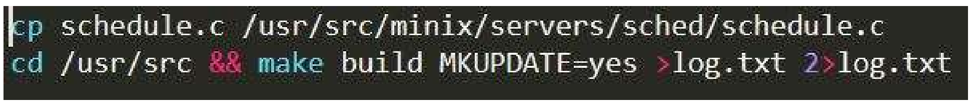
## Part1 →

I read about scheduling in minix and found out that user processes have priority >= 7. In USER_Q, this priority value is saved. So, to filter out only user processes, an if condition is added in schedule_process function. Then, if a process is swapped, then it is put at the end of queue in minix scheduling. So, to find out this swapped process, _ENDPOINT_P(rmp->endpoint) is used.

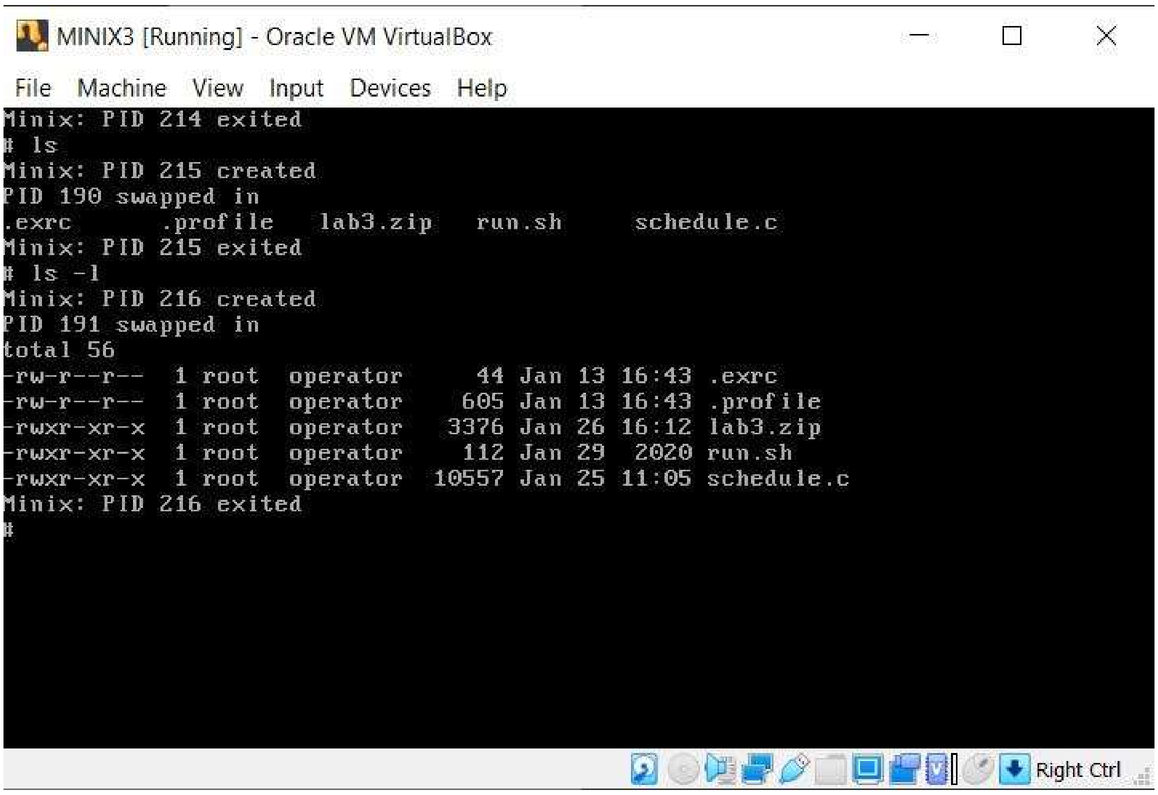


```
/*===========================================================================*
 *              schedule_process                 *
 *===========================================================================*/
static int schedule_process(struct schedproc * rmp, unsigned flags)
{
    int err;
    int new_prio, new_quantum, new_cpu;

    pick_cpu(rmp);

    if (flags & SCHEDULE_CHANGE_PRIO)
        new_prio = rmp->priority;
    else
        new_prio = -1;

    if (flags & SCHEDULE_CHANGE_QUANTUM)
        new_quantum = rmp->time_slice;
    else
        new_quantum = -1;

    if (flags & SCHEDULE_CHANGE_CPU)
        new_cpu = rmp->cpu;
    else
        new_cpu = -1;

    if ((err = sys_schedule(rmp->endpoint, new_prio,
        new_quantum, new_cpu)) != OK) {
        printf("PM: An error occurred when trying to schedule %d: %d\n",
        rmp->endpoint, err);
    }
    if (rmp->priority >= USER_Q)
        printf("PID %d swapped in\n", _ENDPOINT_P(rmp->endpoint));
    return err;
}
```

Then using run.sh file, we replace schedule.c file and make build to update latest minix.



```
cp schedule.c /usr/src/minix/servers/sched/schedule.c
cd /usr/src && make build MKUPDATE=yes >log.txt 2>log.txt
```


MINIX3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

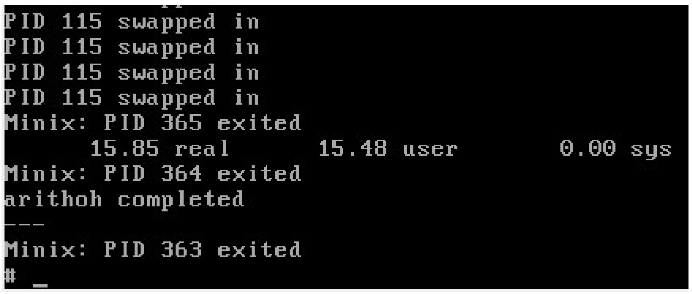
```
Minix: PID 214 exited
# ls
Minix: PID 215 created
PID 190 swapped in
.exrc       .profile   lab3.zip   run.sh     schedule.c
Minix: PID 215 exited
# ls -l
Minix: PID 216 created
PID 191 swapped in
total 56
-rw-r--r--  1 root  operator     44 Jan 13 16:43 .exrc
-rw-r--r--  1 root  operator    605 Jan 13 16:43 .profile
-rwxr-xr-x  1 root  operator   3376 Jan 26 16:12 lab3.zip
-rwxr-xr-x  1 root  operator    112 Jan 29  2020 run.sh
-rwxr-xr-x  1 root  operator  10557 Jan 25 11:05 schedule.c
Minix: PID 216 exited
#
```

PID getting printed for user processes in ls and ls -l command in image above.

## Part2 →

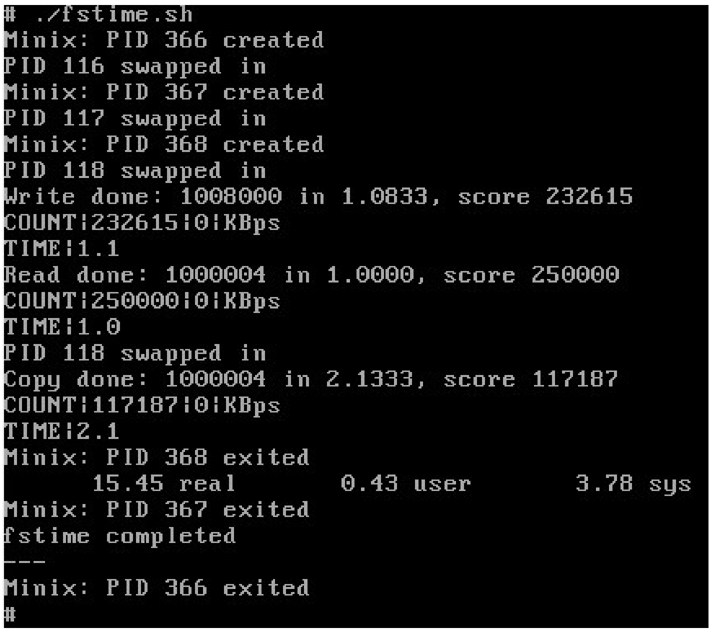
There are 5 different benchmark tests .sh files →

1. arithoh
   This is a CPU Bound Benchmark
   - It is observed that while running 'arithoh' alone, the time taken by real and user is almost same. The kernel scheduler log conforms with this and shows that the message to schedule ./arithoh was sent 92 times consecutively.



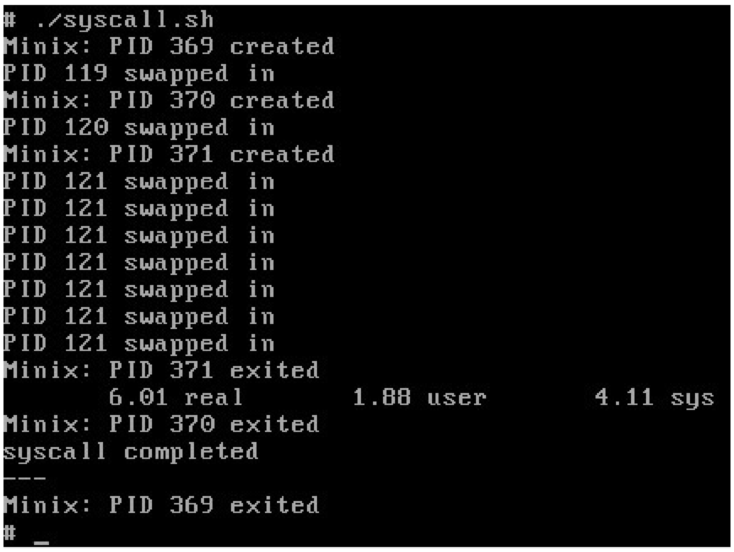
```
PID 115 swapped in
PID 115 swapped in
PID 115 swapped in
PID 115 swapped in
Minix: PID 365 exited
      15.85 real       15.48 user        0.00 sys
Minix: PID 364 exited
arithoh completed
---
Minix: PID 363 exited
# _
```

   - The sys time taken by this is 0.
2. fstime
   This is an IO Bound Benchmark. On executing fstime.sh we observe the following –
   - As it is an IO intensive process the user time is less, and, sys time is moderate compared to the total turnaround time which is large.
   - This is because the process needs to wait for its IO to complete before continuing.



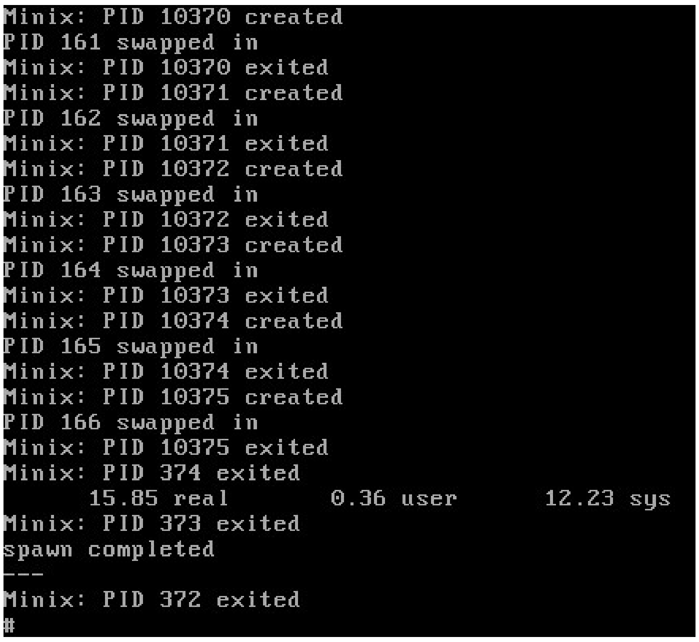
```
# ./fstime.sh
Minix: PID 366 created
PID 116 swapped in
Minix: PID 367 created
PID 117 swapped in
Minix: PID 368 created
PID 118 swapped in
Write done: 1008000 in 1.0833, score 232615
COUNT|232615|0|KBps
TIME|1.1
Read done: 1000004 in 1.0000, score 250000
COUNT|250000|0|KBps
TIME|1.0
PID 118 swapped in
Copy done: 1000004 in 2.1333, score 117187
COUNT|117187|0|KBps
TIME|2.1
Minix: PID 368 exited
      15.45 real       0.43 user       3.78 sys
Minix: PID 367 exited
fstime completed
---
Minix: PID 366 exited
#
```

3. syscall
   - This is a CPU Bound Benchmark
   - The time spent while executing this process is majorly in sys-mode. User mode time spent is also considerable but small.

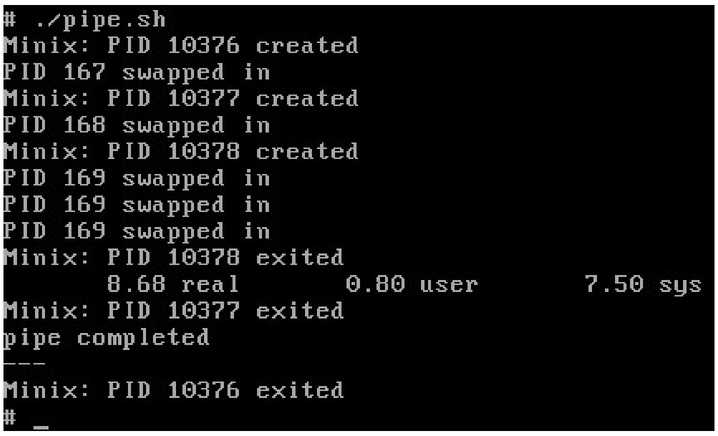


```
# ./syscall.sh
Minix: PID 369 created
PID 119 swapped in
Minix: PID 370 created
PID 120 swapped in
Minix: PID 371 created
PID 121 swapped in
PID 121 swapped in
PID 121 swapped in
PID 121 swapped in
PID 121 swapped in
PID 121 swapped in
PID 121 swapped in
Minix: PID 371 exited
       6.01 real       1.88 user       4.11 sys
Minix: PID 370 exited
syscall completed
---
Minix: PID 369 exited
# _
```

4. spawn
   - This is a CPU Bound Benchmark
   - We see that a huge number of processes ranging from 7 to 255 are swapped into the queue consecutively.

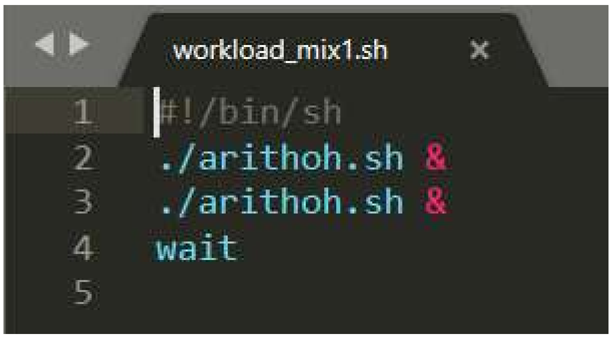

```
Minix: PID 10370 created
PID 161 swapped in
Minix: PID 10370 exited
Minix: PID 10371 created
PID 162 swapped in
Minix: PID 10371 exited
Minix: PID 10372 created
PID 163 swapped in
Minix: PID 10372 exited
Minix: PID 10373 created
PID 164 swapped in
Minix: PID 10373 exited
Minix: PID 10374 created
PID 165 swapped in
Minix: PID 10374 exited
Minix: PID 10375 created
PID 166 swapped in
Minix: PID 10375 exited
Minix: PID 374 exited
      15.85 real        0.36 user       12.23 sys
Minix: PID 373 exited
spawn completed
---
Minix: PID 372 exited
#
```

5. pipe
   - This is a CPU Bound Benchmark based on IPC.
   - A huge amount of time spent by Pipe is in sys mode. This is because the Inter-Process Communication protocols are highly based on the system. The user-mode time spent is very less.


```
# ./pipe.sh
Minix: PID 10376 created
PID 167 swapped in
Minix: PID 10377 created
PID 168 swapped in
Minix: PID 10378 created
PID 169 swapped in
PID 169 swapped in
PID 169 swapped in
Minix: PID 10378 exited
       8.68 real        0.80 user        7.50 sys
Minix: PID 10377 exited
pipe completed
---
Minix: PID 10376 exited
# _
```

I used 4 different workload_mix*.sh (* = 1,2,3,4)

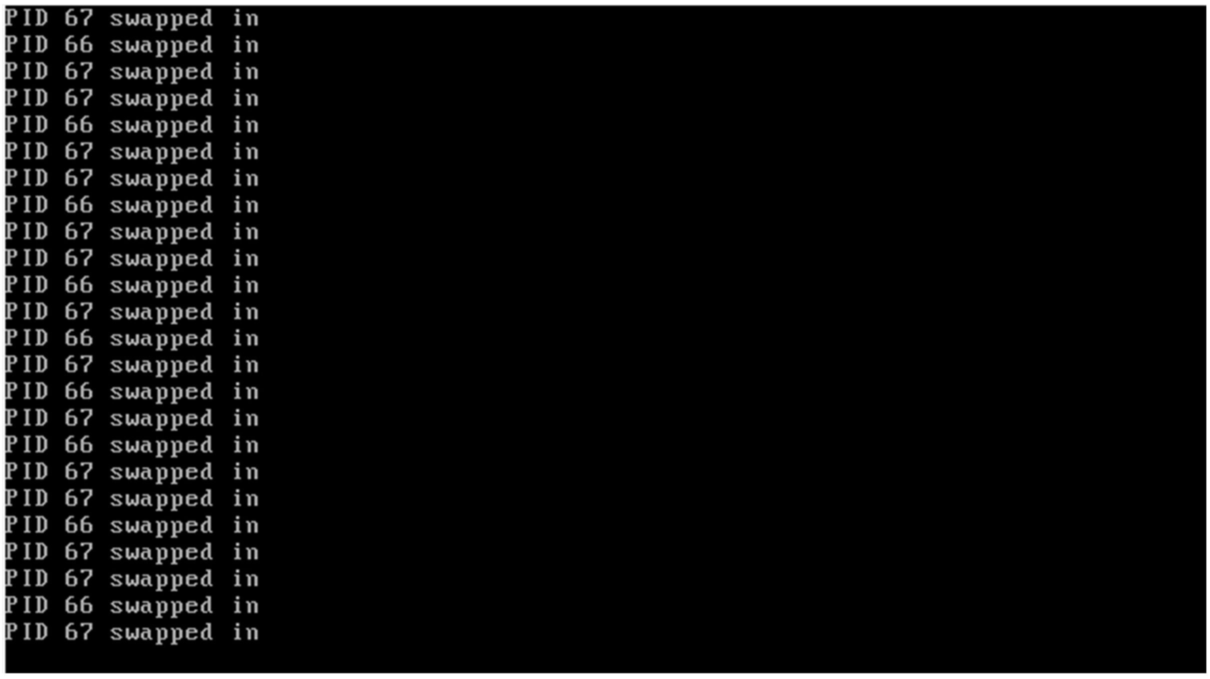
**Workload Mix 1 →**



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

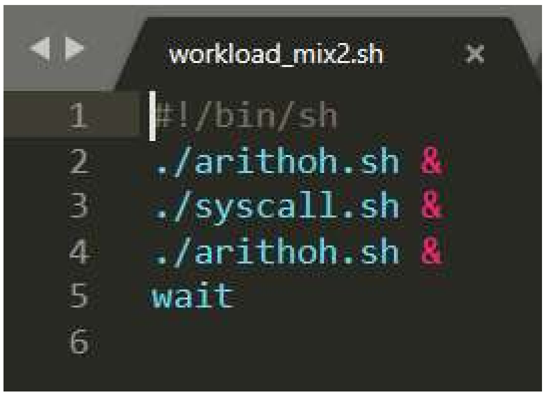
**Output:**

Here, first arithoh.sh has pid 66 and second arithoh.sh has pid 67. If we run multiple instances of arithoh.sh which is cpu intensive, as in this workload, we can see them getting scheduled alternatively.In the given image process with PID 66,67 correspond to two instances of workload in arithoh.sh that are scheduled alternatively until they are completed.



```
PID 67 swapped in
PID 66 swapped in
PID 67 swapped in
PID 67 swapped in
PID 66 swapped in
PID 67 swapped in
PID 67 swapped in
PID 66 swapped in
PID 67 swapped in
PID 67 swapped in
PID 66 swapped in
PID 67 swapped in
PID 66 swapped in
PID 67 swapped in
PID 66 swapped in
PID 67 swapped in
PID 66 swapped in
PID 67 swapped in
PID 67 swapped in
PID 66 swapped in
PID 67 swapped in
PID 67 swapped in
PID 66 swapped in
PID 67 swapped in
```

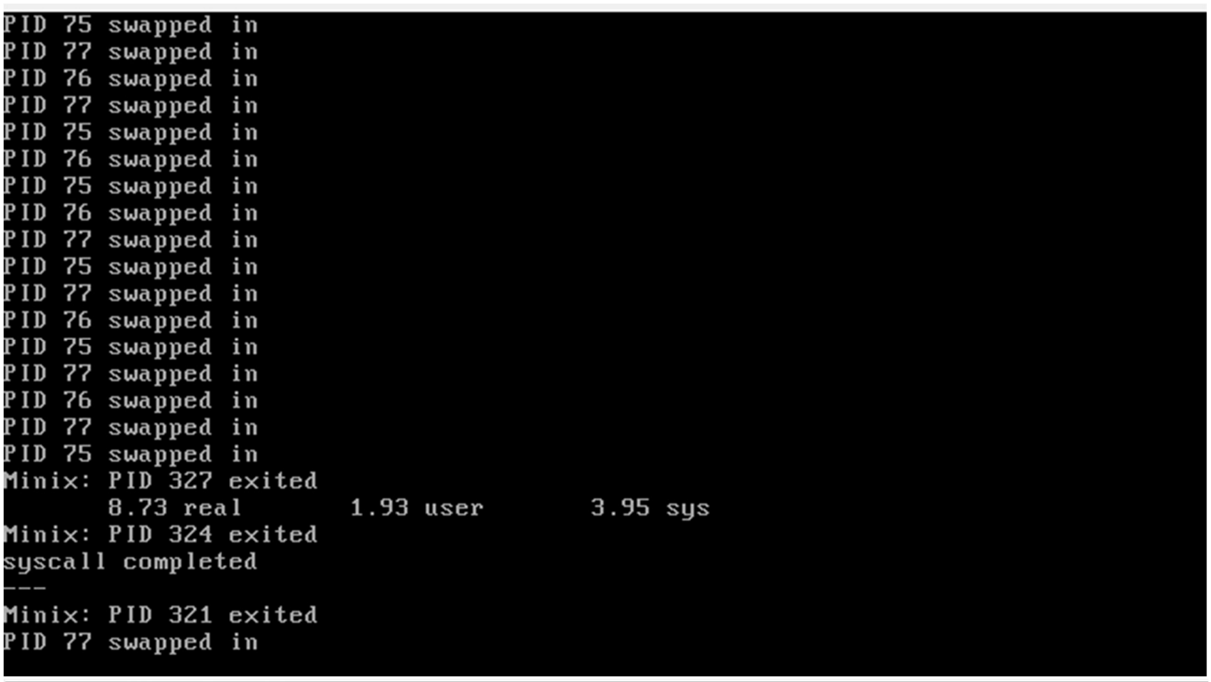
**Workload Mix 2 →**



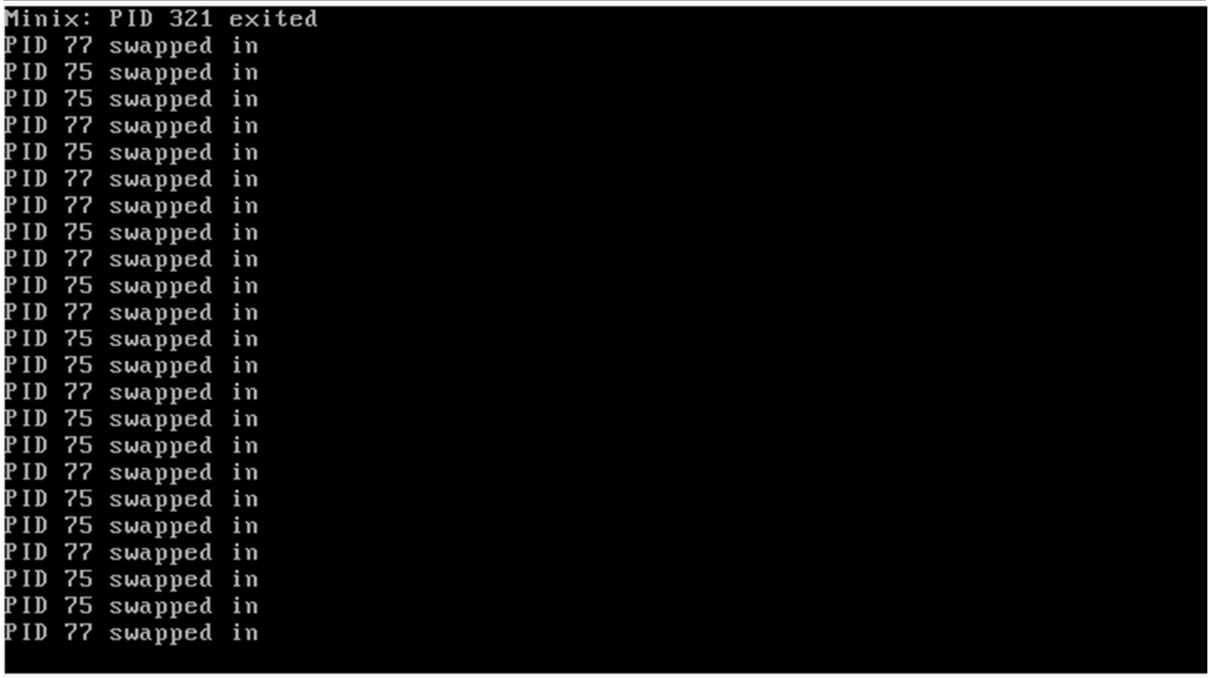
```
#!/bin/sh
./arithoh.sh &
./syscall.sh &
./arithoh.sh &
wait

```

**Output:**

Here, first arithoh.sh has pid 75, syscall.sh has pid 76 and second arithoh.sh has pid 77. We know that arithoh.sh and syscall.sh are different cpu intensive tasks. Here, first all 3 are alternatively scheduled depending upon their intensiveness and then when syscall with pid 76 (which was less cpu intensive as compared to the other two) is completed, then other 2 are alternatively scheduled untill finished.

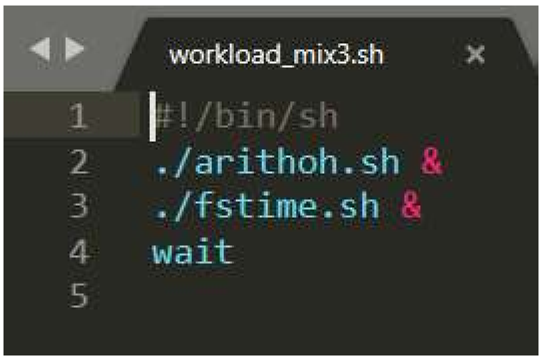


```
PID 75 swapped in
PID 77 swapped in
PID 76 swapped in
PID 77 swapped in
PID 75 swapped in
PID 76 swapped in
PID 75 swapped in
PID 76 swapped in
PID 77 swapped in
PID 75 swapped in
PID 77 swapped in
PID 76 swapped in
PID 75 swapped in
PID 77 swapped in
PID 76 swapped in
PID 77 swapped in
PID 75 swapped in
Minix: PID 327 exited
       8.73 real        1.93 user        3.95 sys
Minix: PID 324 exited
syscall completed
---
Minix: PID 321 exited
PID 77 swapped in
```



```
Minix: PID 321 exited
PID 77 swapped in
PID 75 swapped in
PID 75 swapped in
PID 77 swapped in
PID 75 swapped in
PID 77 swapped in
PID 77 swapped in
PID 75 swapped in
PID 77 swapped in
PID 75 swapped in
PID 77 swapped in
PID 75 swapped in
PID 75 swapped in
PID 77 swapped in
PID 75 swapped in
PID 75 swapped in
PID 77 swapped in
PID 75 swapped in
PID 75 swapped in
PID 77 swapped in
PID 75 swapped in
PID 75 swapped in
PID 77 swapped in
```

**Workload Mix 3 →**

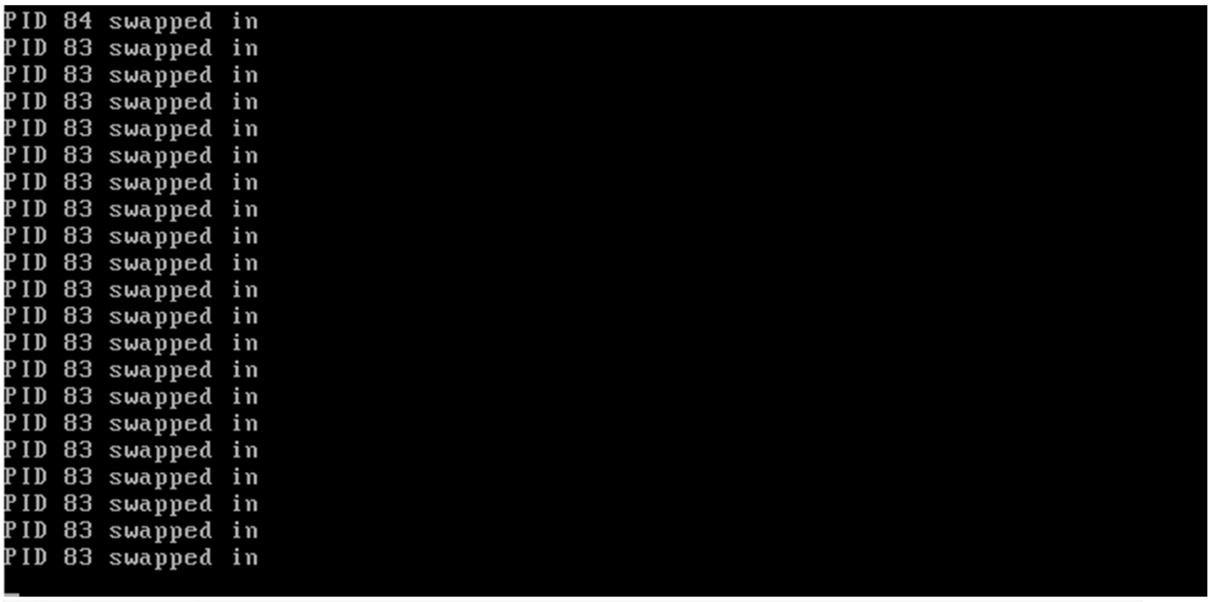


workload_mix3.sh

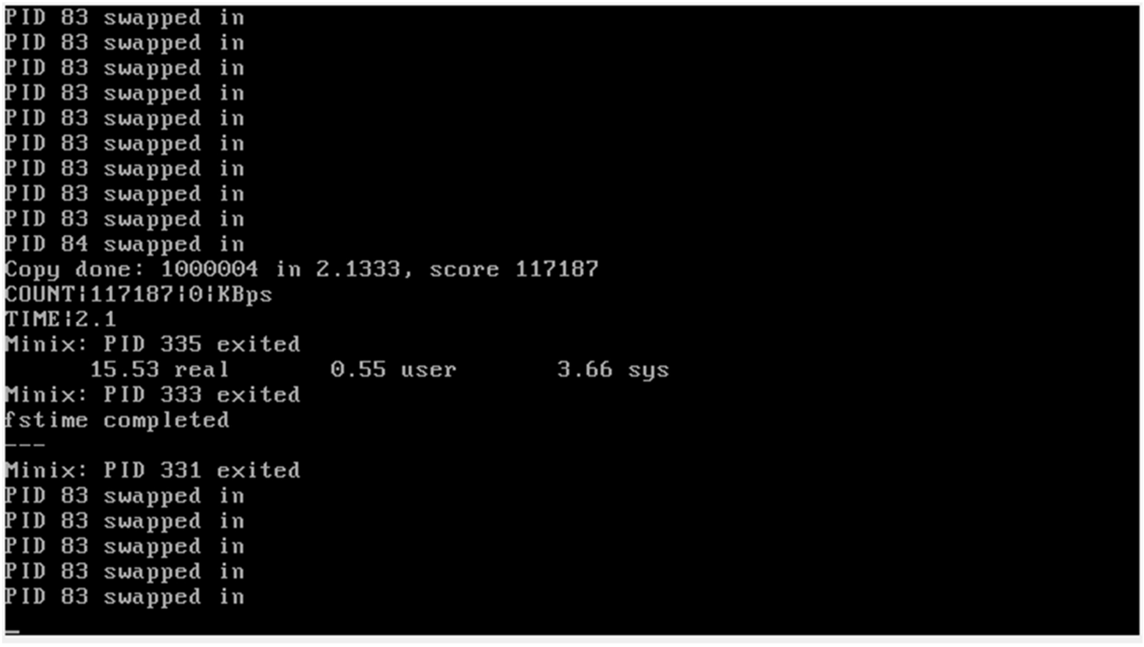
```sh
#!/bin/sh
./arithoh.sh &
./fstime.sh &
wait
```

**Output:**

Here, first arithoh.sh has pid 83 and fstime.sh has pid 84. We know that arithoh.sh is a cpu intensive task and fstime is a i/o task. Here, pid 83 process is scheduled and pid process 84 is waiting for an input till then arithoh.sh is scheduled and is using processor for its cpu intensive tasks. When fstime.sh with pid 84 recieves input then it is scheduled and process is completed and then next 83 pid task is scheduled till it completes.

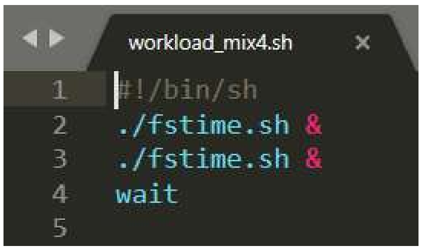


```
PID 84 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
_
```



```
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 84 swapped in
Copy done: 1000004 in 2.1333, score 117187
COUNT|117187|0|KBps
TIME|2.1
Minix: PID 335 exited
      15.53 real        0.55 user        3.66 sys
Minix: PID 333 exited
fstime completed
---
Minix: PID 331 exited
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
PID 83 swapped in
_
```

**Workload Mix 4 →**

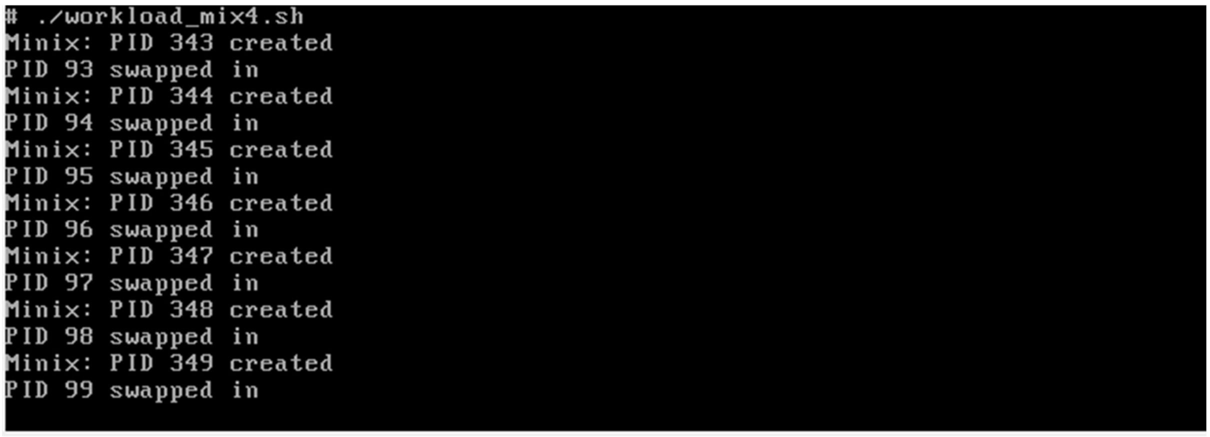


workload_mix4.sh

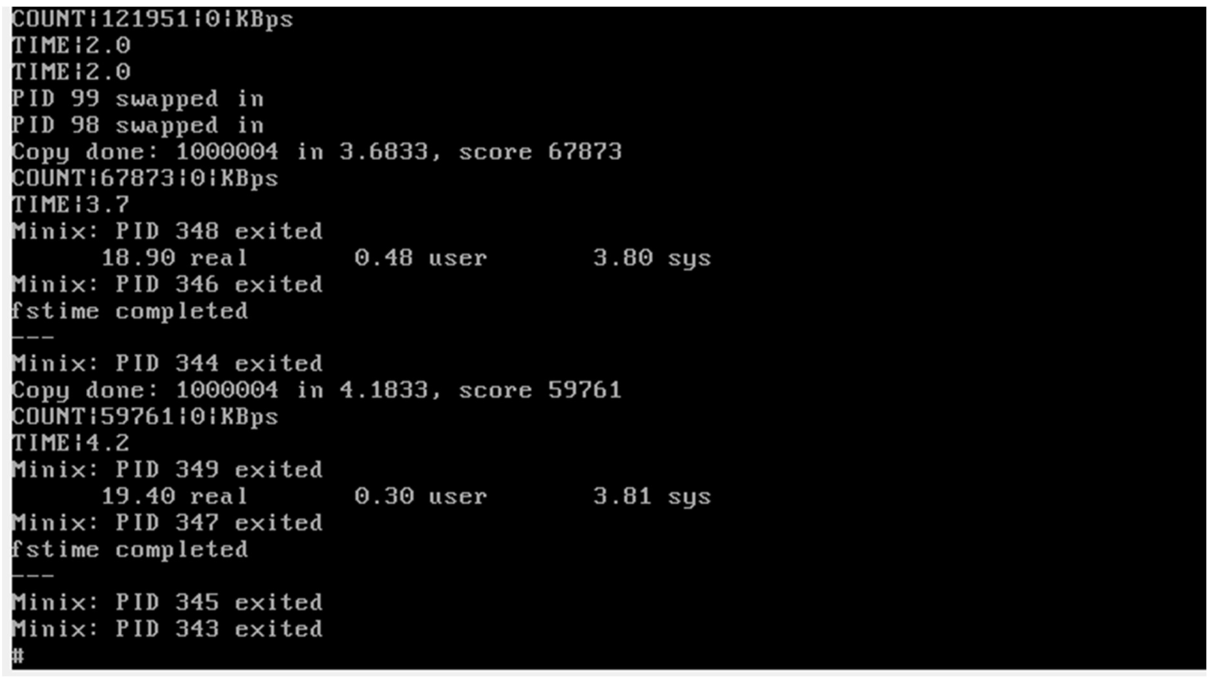
```
#!/bin/sh
./fstime.sh &
./fstime.sh &
wait
```

**Output:**

Here, we have two same workloads fstime.sh with pid 98 and 99. Both are i/o processes that are waiting for input. When input received for one, then that pid process finished and then next pid process scheduled until input received and finished.



```
# ./workload_mix4.sh
Minix: PID 343 created
PID 93 swapped in
Minix: PID 344 created
PID 94 swapped in
Minix: PID 345 created
PID 95 swapped in
Minix: PID 346 created
PID 96 swapped in
Minix: PID 347 created
PID 97 swapped in
Minix: PID 348 created
PID 98 swapped in
Minix: PID 349 created
PID 99 swapped in
```



```
COUNT|121951|0|KBps
TIME|2.0
TIME|2.0
PID 99 swapped in
PID 98 swapped in
Copy done: 1000004 in 3.6833, score 67873
COUNT|67873|0|KBps
TIME|3.7
Minix: PID 348 exited
      18.90 real        0.48 user        3.80 sys
Minix: PID 346 exited
fstime completed
---
Minix: PID 344 exited
Copy done: 1000004 in 4.1833, score 59761
COUNT|59761|0|KBps
TIME|4.2
Minix: PID 349 exited
      19.40 real        0.30 user        3.81 sys
Minix: PID 347 exited
fstime completed
---
Minix: PID 345 exited
Minix: PID 343 exited
#
```