CS314 Operating Systems Lab

Cheedrala Jaswanth-200010008, Altmash Sheikh-200010002

Lab Assignment 3

Part I:

The code in minix/servers/sched/schedule.c has been modified, so that “<Roll No> PID <pid> swapped in” is printed whenever a user level process is brought in by the scheduler.

The code given below has been added in the schedule_process function just above the return statement.

```
if(rmp->priority >= USER_Q) {
        printf("200010008 PID %d swapped in\n",_ENDPOINT_P(rmp->endpoint));
    }
```

We have written a runme.sh file in order to copy the schedule.c file to its location and to build the changes.

Below is the image of a successful build of the code for this part.

Part II:

UnixBench Benchmark has been downloaded and added into the home folder of MINIX 3. In the Unixbench folder, Makefile has been run using the command gmake. And here is the successful run of the command.

```
c
MINIX: PID 433 created
200010008 PID 188 swapped in
MINIX: PID 434 created
200010008 PID 189 swapped in
MINIX: PID 434 exited
MINIX: PID 435 created
200010008 PID 195 swapped in
MINIX: PID 435 exited
MINIX: PID 433 exited
clang -o pgms/whetstone-double -Wall -pedantic -O0 -ffast-math -I ./src -DTIME -
DDP -DGTODay -DUNIXBENCH src/whets.c -lm
MINIX: PID 436 created
200010008 PID 196 swapped in
MINIX: PID 437 created
200010008 PID 197 swapped in
MINIX: PID 437 exited
MINIX: PID 438 created
200010008 PID 198 swapped in
MINIX: PID 438 exited
MINIX: PID 436 exited
gmake[1]: Leaving directory '/usr/byte-unixbench-mod/UnixBench'
MINIX: PID 380 exited
MINIX: PID 368 exited
#
```

After this, we have gone into the UnixBench/workload_mix folder in which there are some benchmark programs in .sh files: arithoh.sh, fstime.sh, pipe.sh, spawn.sh, syscall.sh.

We are given a mixture of these files in workload_mix.sh. It is as follows:

```
#!/bin/sh
./arithoh.sh &
./fstime.sh &
./pipe.sh &
./spawn.sh &
./syscall.sh &
./arithoh.sh &
./fstime.sh &
./pipe.sh &
./spawn.sh &
./syscall.sh &
wait
```

This file has been run and successfully like this.

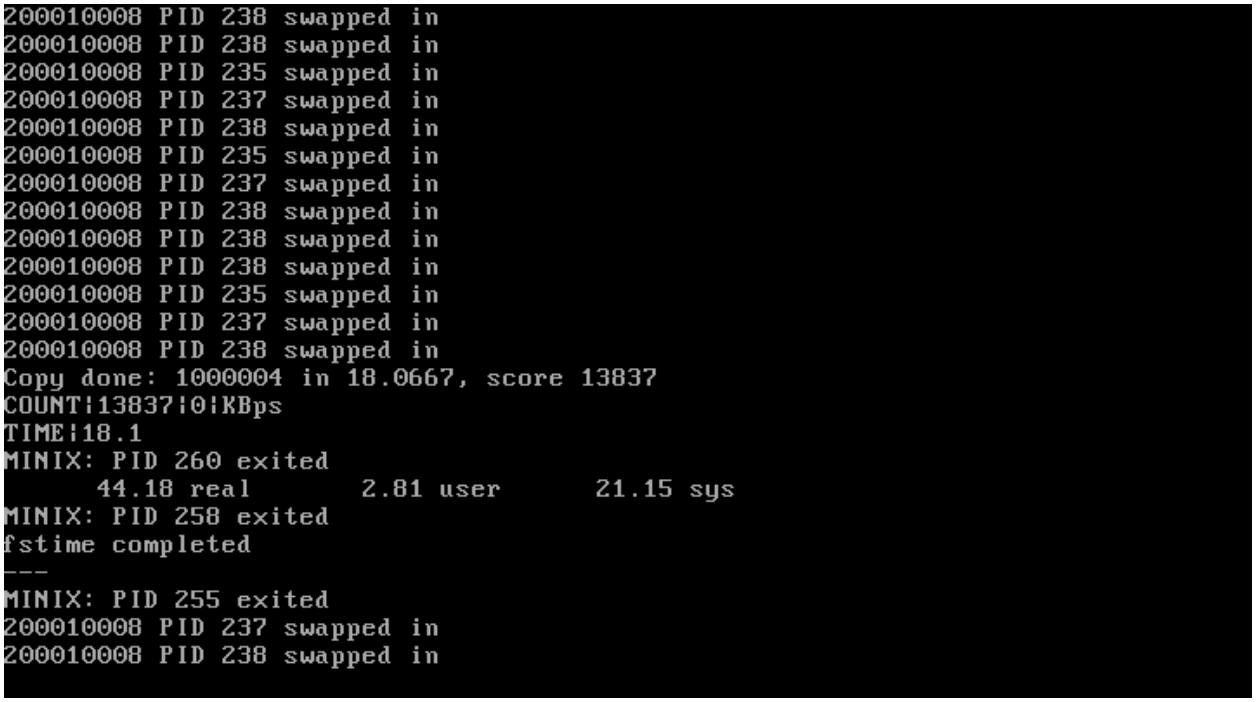
```
200010008 PID 218 swapped in
200010008 PID 237 swapped in
200010008 PID 218 swapped in
200010008 PID 237 swapped in
200010008 PID 218 swapped in
200010008 PID 237 swapped in
200010008 PID 218 swapped in
200010008 PID 237 swapped in
200010008 PID 218 swapped in
200010008 PID 237 swapped in
MINIX: PID 262 exited
    4:54.33 real      14.26 user      32.46 sys
MINIX: PID 257 exited
syscall completed
---
MINIX: PID 241 exited
200010008 PID 218 swapped in
MINIX: PID 243 exited
    4:55.76 real      14.21 user      33.31 sys
MINIX: PID 237 exited
syscall completed
---
MINIX: PID 229 exited
MINIX: PID 224 exited
# _
```

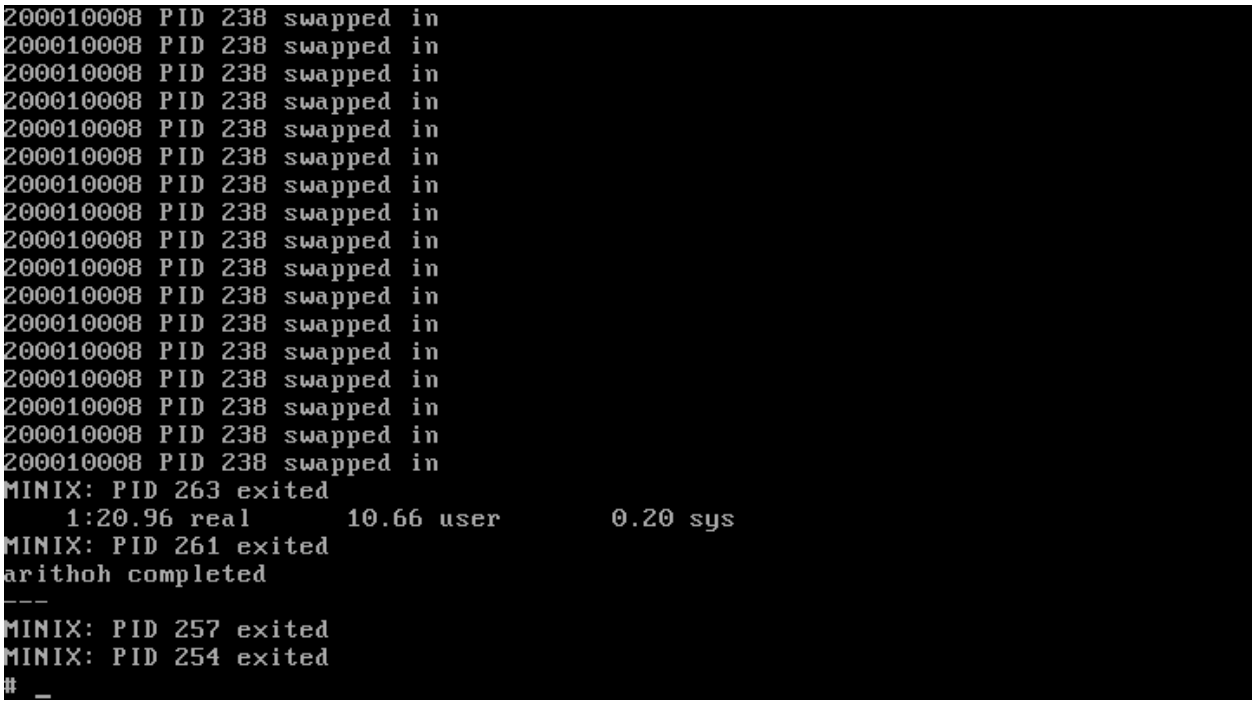
I have written 4 .sh files using the above in different mixtures to observe the behavior of the scheduler by looking at the PID created, excited, swapped in statements printed.

i) mymix1.sh:

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

```
# bash mymix1.sh
MINIX: PID 254 created
200010008 PID 229 swapped in
MINIX: PID 255 created
200010008 PID 230 swapped in
MINIX: PID 256 created
200010008 PID 231 swapped in
MINIX: PID 257 created
200010008 PID 232 swapped in
MINIX: PID 258 created
200010008 PID 233 swapped in
MINIX: PID 259 created
200010008 PID 234 swapped in
MINIX: PID 260 created
200010008 PID 235 swapped in
MINIX: PID 261 created
200010008 PID 236 swapped in
MINIX: PID 262 created
200010008 PID 237 swapped in
MINIX: PID 263 created
200010008 PID 238 swapped in
200010008 PID 238 swapped in
200010008 PID 237 swapped in
200010008 PID 238 swapped in
```





In this execution, fstime completed first, after that syscall completed, and finally arithoh got completed. This is because, fstime is I/O bound, and it is given high priority in the queue, whereas arithoh is highly CPU bound with lot of computations within, so it takes a lot of time slices to get completed, syscall contains system calls and it executes in between.

ii)mymix2.sh:

#!/bin/bash

./syscall.sh &
./syscall.sh &
wait

```
200010008 PID 27 swapped in
200010008 PID 27 swapped in
200010008 PID 91 swapped in
200010008 PID 91 swapped in
200010008 PID 27 swapped in
200010008 PID 27 swapped in
200010008 PID 27 swapped in
200010008 PID 27 swapped in

# bash mymix2.sh
MINIX: PID 264 created
200010008 PID 239 swapped in
MINIX: PID 265 created
200010008 PID 240 swapped in
MINIX: PID 266 created
200010008 PID 241 swapped in
MINIX: PID 267 created
200010008 PID 242 swapped in
MINIX: PID 268 created
200010008 PID 243 swapped in
MINIX: PID 269 created
200010008 PID 244 swapped in
MINIX: PID 270 created
200010008 PID 245 swapped in
_
```

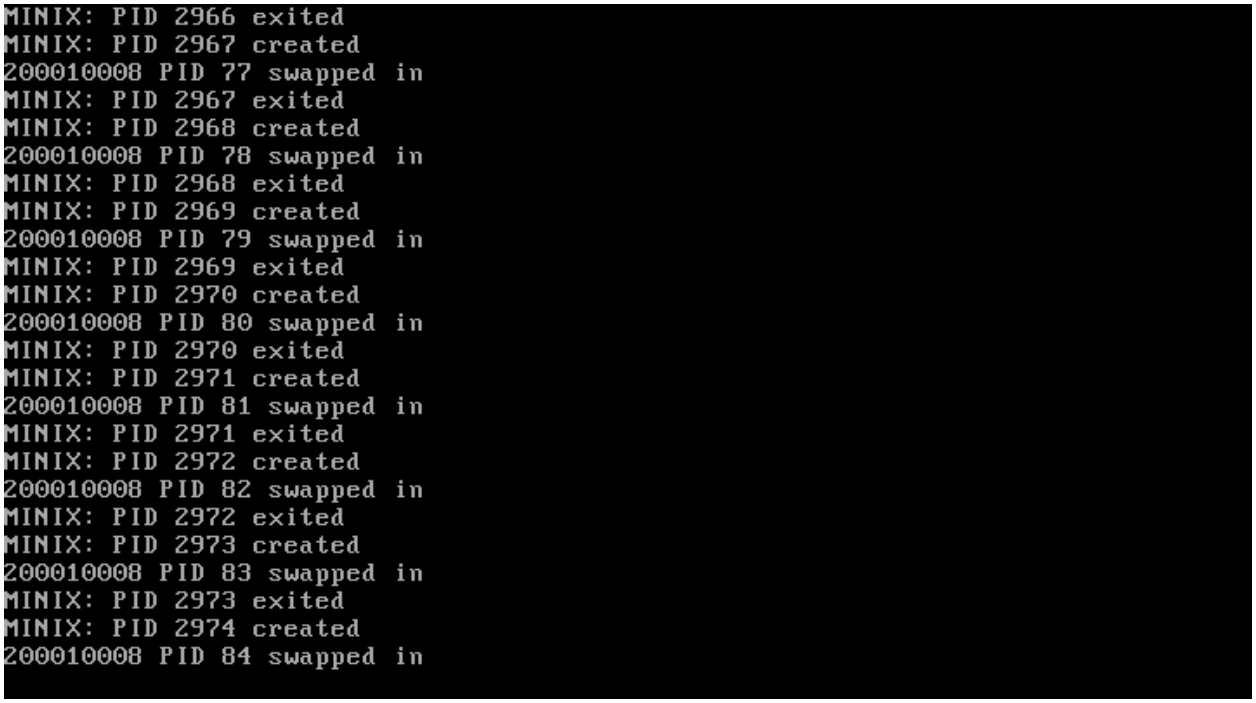
```
200010008 PID 245 swapped in
200010008 PID 244 swapped in
200010008 PID 245 swapped in
200010008 PID 244 swapped in
200010008 PID 245 swapped in
200010008 PID 244 swapped in
200010008 PID 245 swapped in
200010008 PID 244 swapped in
200010008 PID 245 swapped in
200010008 PID 245 swapped in
200010008 PID 244 swapped in
MINIX: PID 270 exited
    1:30.15 real      13.33 user      30.80 sys
MINIX: PID 268 exited
syscall completed
---
MINIX: PID 266 exited
MINIX: PID 269 exited
    1:30.20 real      13.28 user      32.71 sys
MINIX: PID 267 exited
syscall completed
---
MINIX: PID 265 exited
MINIX: PID 264 exited
# _
```

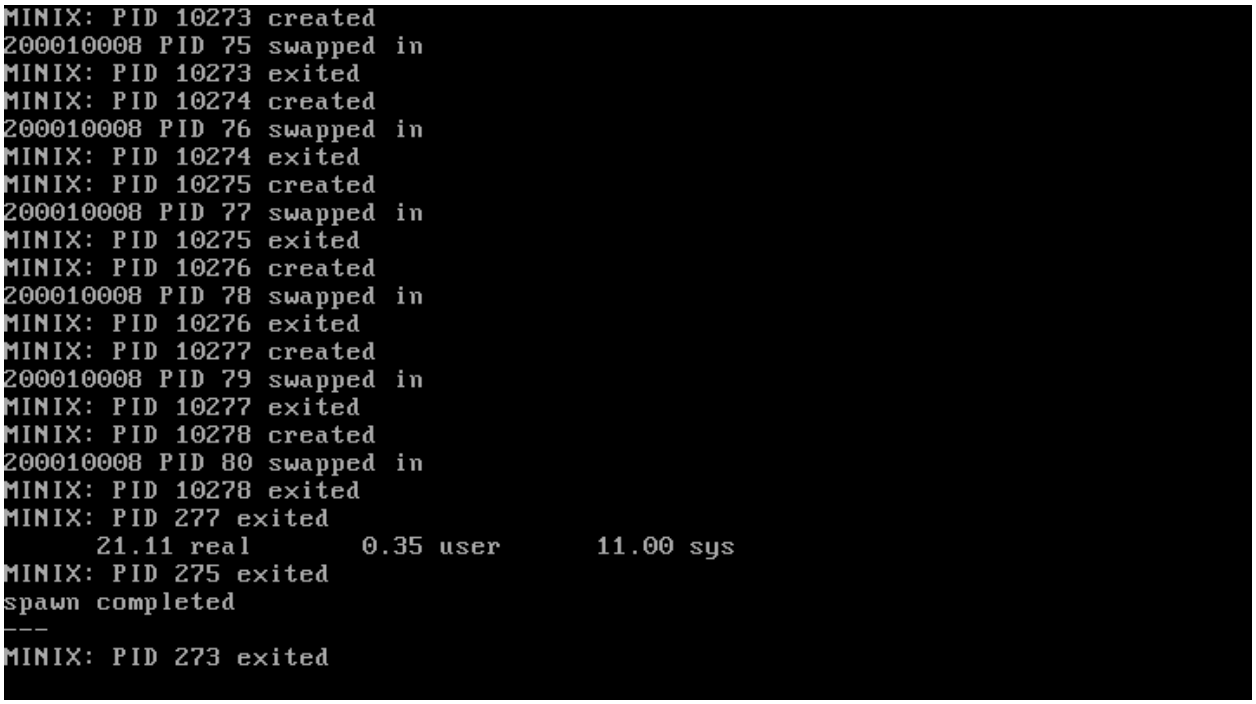
In this, the two syscall, will get executed switching in between every time, and then get completed consecutively.

iii)mymix3.sh:

#!/bin/bash

./pipe.sh &
./spawn.sh &
wait





We can see that pipe get completed first before spawn, and enormous no. of processes are created because of the loop execution of fork(). But the created child processes are exited without taking much time, which means there is not much work to the child processes.

iv)mymix4.sh:

```
#!/bin/bash
./pipe.sh &
./pipe.sh &
./pipe.sh &
wait
```

```
200010008 PID 27 swapped in

# bash mymix4.sh
MINIX: PID 10279 created
200010008 PID 81 swapped in
MINIX: PID 10280 created
200010008 PID 82 swapped in
MINIX: PID 10281 created
200010008 PID 83 swapped in
MINIX: PID 10282 created
200010008 PID 84 swapped in
MINIX: PID 10283 created
200010008 PID 85 swapped in
MINIX: PID 10284 created
200010008 PID 86 swapped in
MINIX: PID 10285 created
200010008 PID 87 swapped in
MINIX: PID 10286 created
200010008 PID 88 swapped in
MINIX: PID 10287 created
200010008 PID 89 swapped in
MINIX: PID 10288 created
200010008 PID 90 swapped in
200010008 PID 27 swapped in
```

```
200010008 PID 87 swapped in
200010008 PID 90 swapped in
200010008 PID 87 swapped in
200010008 PID 89 swapped in
200010008 PID 90 swapped in
MINIX: PID 10288 exited
    2:22.50 real       5.60 user      41.01 sys
MINIX: PID 10286 exited
pipe completed
---
MINIX: PID 10282 exited
MINIX: PID 10285 exited
    2:22.91 real       4.96 user      41.91 sys
MINIX: PID 10283 exited
pipe completed
---
MINIX: PID 10280 exited
MINIX: PID 10287 exited
    2:23.05 real       5.75 user      43.75 sys
MINIX: PID 10284 exited
pipe completed
---
MINIX: PID 10281 exited
MINIX: PID 10279 exited
#
```

We can see that the 3 pipe sh files get executed and completed consecutively, alternating in between equally. But the time taken before each swapping is more compared to other processes, that means it have more time slice.