Harrithha, Shashank P
200010018, 200010048

CS314 Operating Systems Lab
Lab 3

Part 1

In order to perform the given task, the file **main.c** was changed at **minix/servers/sched/** location. The following statement was added in main.c file at **line number 65** when a process is swapped in:

```
if(rmp->priority >= USER_Q)
{
    printf("PID: %d swapped in 200010048 200010018\n", _ENDPOINT_P(rmp->endpoint));
}
return OK;
```

These changes were made in the host machine (Windows). It was transferred to Minix3 Virtual Machine through github. Then, `make build MKUPDATE=yes` was run and the Minix3 Virtual Machine was rebooted. Upon running the `ls` command, the following output was obtained:

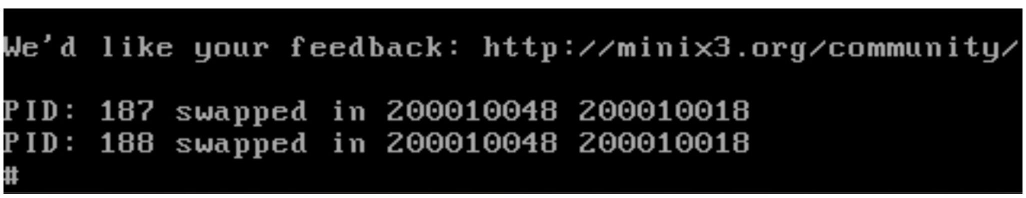


Figure 1: After Reboot

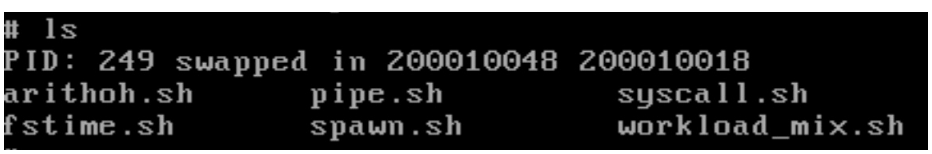
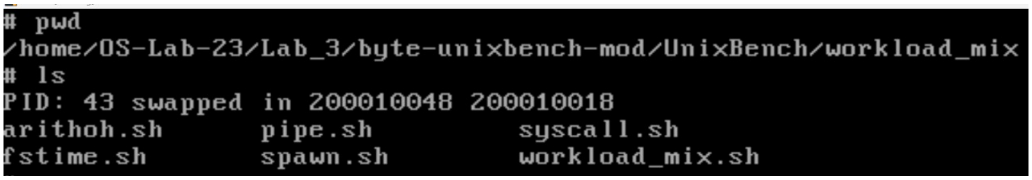


Figure 2: Sample Command

Part 2

Individual Benchmarks

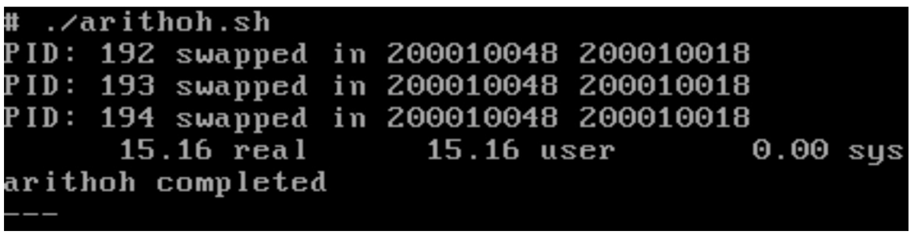
- We got the source code from the link mentioned in the assignment and copied it to the home folder in the Minix3 VM.
- `gmake` command was run to build the benchmarks.



```
# pwd
/home/OS-Lab-23/Lab_3/byte-unixbench-mod/UnixBench/workload_mix
# ls
PID: 43 swapped in 200010048 200010018
arithoh.sh      pipe.sh        syscall.sh
fstime.sh       spawn.sh       workload_mix.sh
```

Figure 3: Benchmarks

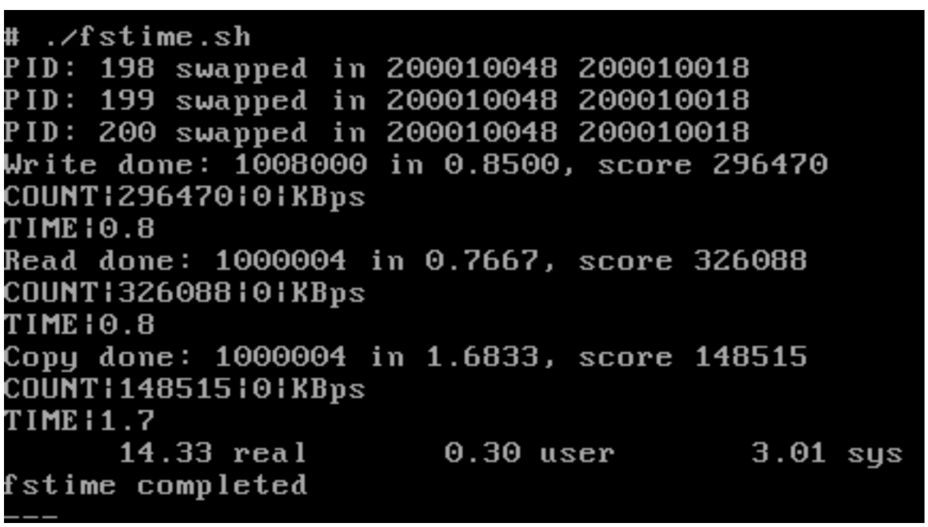
- **arithoh.sh**



```
# ./arithoh.sh
PID: 192 swapped in 200010048 200010018
PID: 193 swapped in 200010048 200010018
PID: 194 swapped in 200010048 200010018
      15.16 real       15.16 user       0.00 sys
arithoh completed
---
```

Figure 4: arithoh.sh

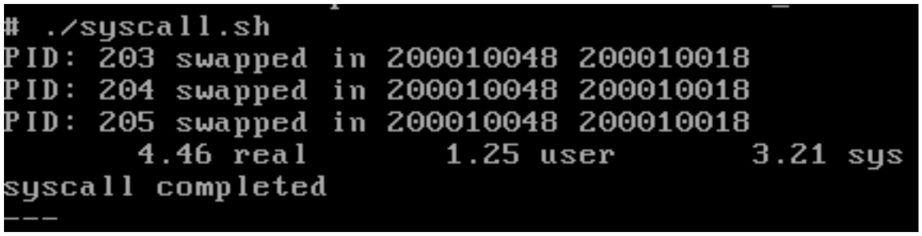
- **fstime.sh**



```
# ./fstime.sh
PID: 198 swapped in 200010048 200010018
PID: 199 swapped in 200010048 200010018
PID: 200 swapped in 200010048 200010018
Write done: 1008000 in 0.8500, score 296470
COUNT|296470|0|KBps
TIME|0.8
Read done: 1000004 in 0.7667, score 326088
COUNT|326088|0|KBps
TIME|0.8
Copy done: 1000004 in 1.6833, score 148515
COUNT|148515|0|KBps
TIME|1.7
      14.33 real       0.30 user       3.01 sys
fstime completed
---
```

Figure 5: fstime.sh

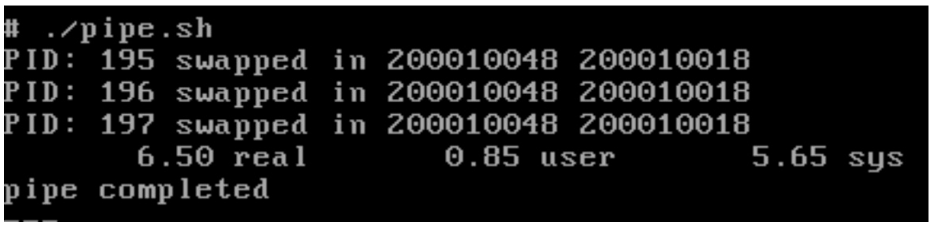
- **syscall.sh**



```
# ./syscall.sh
PID: 203 swapped in 200010048 200010018
PID: 204 swapped in 200010048 200010018
PID: 205 swapped in 200010048 200010018
        4.46 real        1.25 user        3.21 sys
syscall completed
```

Figure 6: syscall.sh

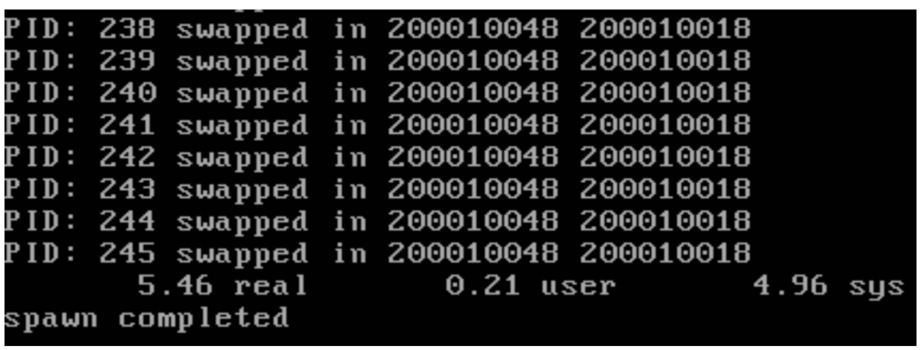
- **pipe.sh**



```
# ./pipe.sh
PID: 195 swapped in 200010048 200010018
PID: 196 swapped in 200010048 200010018
PID: 197 swapped in 200010048 200010018
        6.50 real        0.85 user        5.65 sys
pipe completed
```

Figure 7: pipe.sh

- **spawn.sh**



```
PID: 238 swapped in 200010048 200010018
PID: 239 swapped in 200010048 200010018
PID: 240 swapped in 200010048 200010018
PID: 241 swapped in 200010048 200010018
PID: 242 swapped in 200010048 200010018
PID: 243 swapped in 200010048 200010018
PID: 244 swapped in 200010048 200010018
PID: 245 swapped in 200010048 200010018
        5.46 real        0.21 user        4.96 sys
spawn completed
```

Figure 8: spawn.sh

Plots

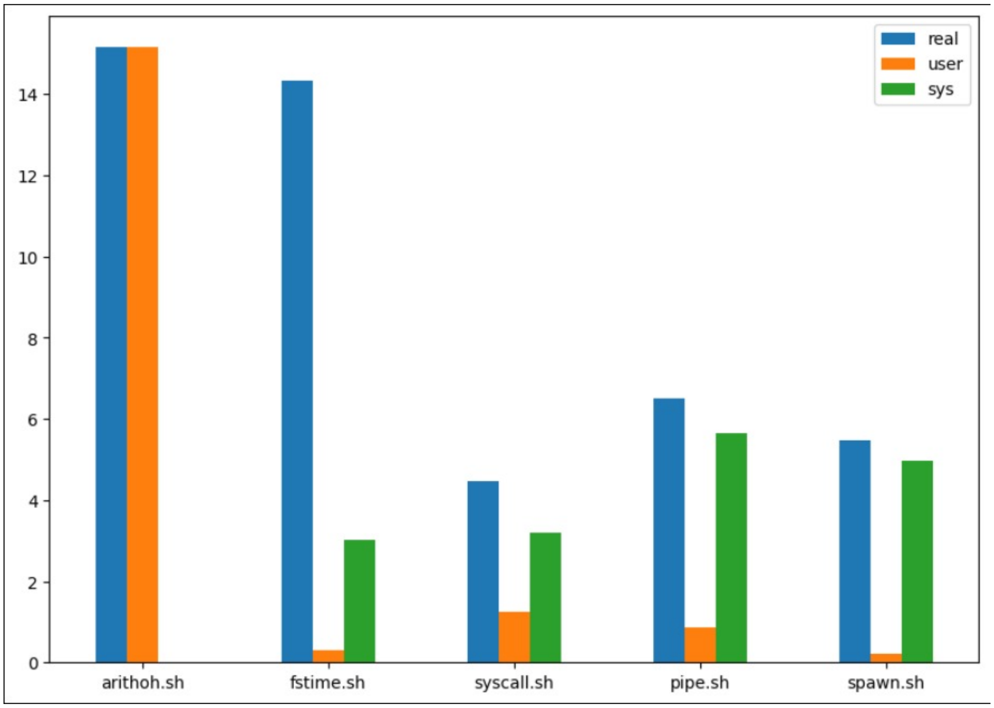


Figure 9: Plots comparing the real, user and system times for above processes

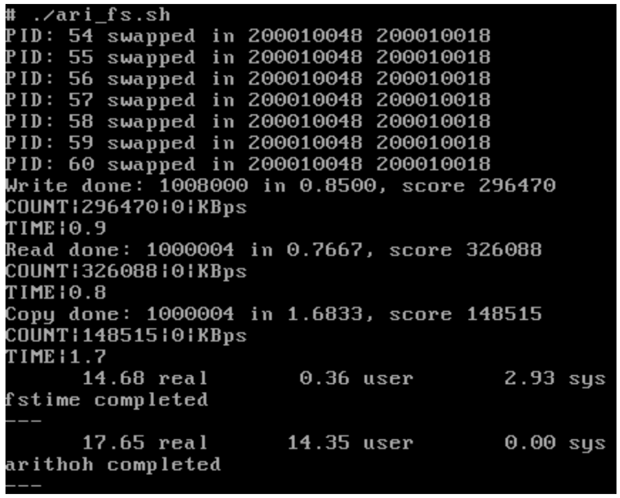
Inferences

- **arithoh.sh:** It is observed that there is no system time used. The entire process is run in user mode as they are cpu intensive processes.
- **fstime.sh:** Here, the real time is high as compared to user and sys time because the process consists of several calls to time, date, sleep etc. (real time processes whose deadlines have to be met).
- **syscall.sh, pipe.sh, spawn.sh:** These three benchmarks mostly consist of system calls. Hence, we can see that most of the process is run in kernel mode rather than user mode.

Workload Mix Benchmarks

- **arithoh.sh & fstime.sh**

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



```
# ./ari_fs.sh
PID: 54 swapped in 200010048 200010018
PID: 55 swapped in 200010048 200010018
PID: 56 swapped in 200010048 200010018
PID: 57 swapped in 200010048 200010018
PID: 58 swapped in 200010048 200010018
PID: 59 swapped in 200010048 200010018
PID: 60 swapped in 200010048 200010018
Write done: 1008000 in 0.8500, score 296470
COUNT|296470|0|KBps
TIME|0.9
Read done: 1000004 in 0.7667, score 326088
COUNT|326088|0|KBps
TIME|0.8
Copy done: 1000004 in 1.6833, score 148515
COUNT|148515|0|KBps
TIME|1.7
      14.68 real        0.36 user        2.93 sys
fstime completed
---
      17.65 real       14.35 user        0.00 sys
arithoh completed
---
```

Figure 10: ari_fs.sh

Inferences : It is observed that, fstime completes its execution before arithoh as fstime mostly consists of **real time processes** which have the **highest priority** when compared to arith operations.

- **Our Benchmarks : loop.c**

```c
int main(argc, argv)
int argc;
char  *argv[];
{
  while(1)
    {
    }

return(0);
}
```

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

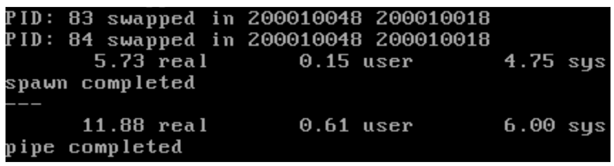


Figure 11: spawn_pipe.sh

Inferences : We observe that even though **loop.c** consists of an infinite loop and will never complete execution, the scheduler gives CPU time to the arithmetic process in order to complete its execution. Hence, the scheduler does not wait for a process to be completed in order to run the next one.