

Operating Systems Laboratory

Lab 3 Report

Pavan Kumar V Patil 200030041
Karthik J Ponarkar 200010022

January 21, 2023

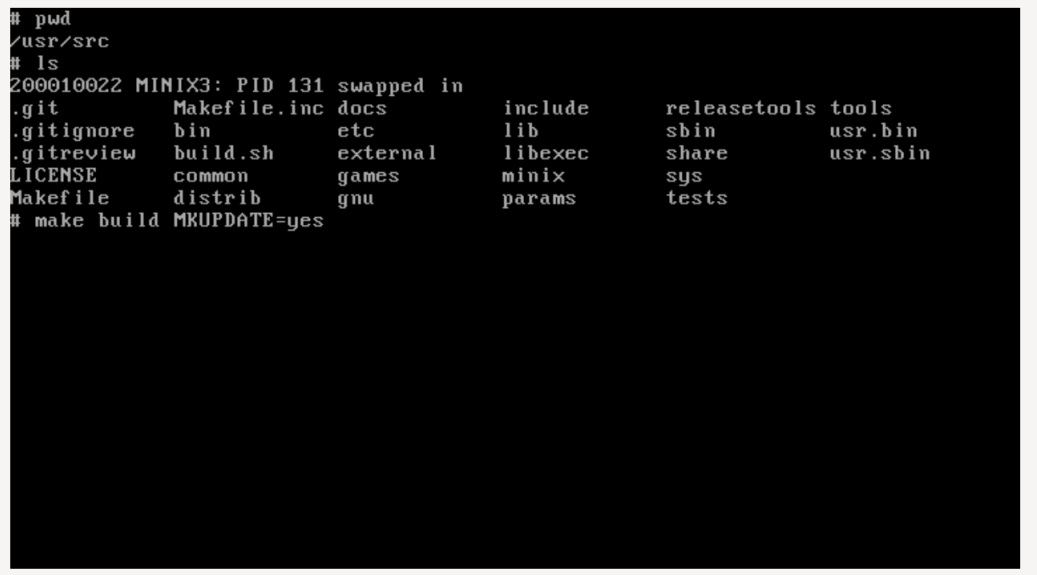
1 Part 1

Task: Modify the Minix 3 source code such that the string “PID <pid> swapped in” is printed, whenever the scheduler brings in the user-level process.

- By referring to `src/minix/servers/sched/schedule.c` and `src/minix/kernel/proc.c` files, we were able to accomplish the above task by modifying some of its content.
- The following piece of code was added to the `schedule.c` file in the `do_start_scheduling` function.
- We print the process id of the swapped-in process when it is successfully scheduled and allocated a CPU to perform its tasks by the `schedule_process` function. We print only if it is a user process (`priority >= USER_Q`) and return OK.
- The value of `USER_Q` is 7. There are a total of 16 ready queues and lower the queue index, greater is the priority of the process it contains.
- Executing the script file modifies the changes made into `schedule.c` into the original `src` directory and builds using `make build MKUPDATE=yes` command. While running the script, both `runme.sh` and modified `schedule.c` must be in the same directory.

```
1 // schedule.c
2 if (rmp->priority >= USER_Q) {
3     int pid = _ENDPOINT_P(rmp->endpoint);
4     printf("200030041 PID %d swapped in\n", pid);
5 }
```

```
6 # runme.sh
7 rm -f /usr/src/minix/servers/sched/schedule.c
8 mv schedule.c /usr/src/minix/servers/sched/
9 make build MKUPDATE=yes -C /usr/src/
```



```
# pwd
/usr/src
# ls
200010022 MINIX3: PID 131 swapped in
.git          Makefile.inc  docs          include       releasetools  tools
.gitignore    bin           etc           lib           sbin          usr.bin
.gitreview    build.sh      external      libexec      share         usr.sbin
LICENSE       common        games         minix        sys
Makefile      distrib      gnu           params        tests
# make build MKUPDATE=yes
```

Figure 1: Running `make build MKUPDATE=yes`

```

200010022 MINIX3: PID 248 swapped in
200010022 MINIX3: PID 249 swapped in
200010022 MINIX3: PID 250 swapped in
200010022 MINIX3: PID 251 swapped in
200010022 MINIX3: PID 252 swapped in
200010022 MINIX3: PID 253 swapped in
200010022 MINIX3: PID 254 swapped in
200010022 MINIX3: PID 255 swapped in
200010022 MINIX3: PID 7 swapped in
200010022 MINIX3: PID 11 swapped in
200010022 MINIX3: PID 15 swapped in
200010022 MINIX3: PID 18 swapped in
200010022 MINIX3: PID 33 swapped in
200010022 MINIX3: PID 35 swapped in
200010022 MINIX3: PID 36 swapped in
200010022 MINIX3: PID 37 swapped in
200010022 MINIX3: PID 38 swapped in
200010022 MINIX3: PID 39 swapped in
200010022 MINIX3: PID 40 swapped in
200010022 MINIX3: PID 41 swapped in
Build started at: Sat Jan 21 12:14:45 GMT 2023
200010022 MINIX3: PID 42 swapped in
Build finished at: 200010022 MINIX3: PID 43 swapped in
Sat Jan 21 12:20:38 GMT 2023
# _

```

Figure 2: Build finished successfully

```

# ls
200010022 MINIX3: PID 190 swapped in
.exrc                .ssh                byte-unixbench-mod.zip
.profile             byte-unixbench-mod
# cat
200010022 MINIX3: PID 191 swapped in
=

```

Figure 3: Output of part 1 after system reboot

2 Part 2

Task: Submit a study of the nature of the benchmarks in the UnixBench suite by analyzing the schedule orders. You may supplement your analysis by studying the source code of the benchmarks (in the folder `src`) and also studying online documentation of these benchmarks.

- The main purpose of the UnixBench (a benchmarking suite) tool is to provide an indication of the overall performance of a Unix-like operating system.
- UnixBench can be used to evaluate the performance of your system when running single or multiple tasks.
- This tool is a system benchmarking tool, not just a CPU, RAM, or disk benchmark tool. The results will depend not only on the hardware of the system, but also on the operating system, libraries, and even compiler.
- Unixbench will do the following benchmark tests on your system:
 - **arithmetic:** It consists of highly computational processes (CPU bound).
 - **fstime:** It consists of file read and write (IO bound) processes.
 - **syscall:** It consists of processes of entering and leaving the operating system kernel.
 - **pipe:** It consists of processes that read and write into a pipe.
 - **swan:** It consists of processes that usually spawns a lot of child processes.

2.a Setting up the UnixBench

```
# pwd
/root/byte-unixbench-mod/UnixBench
# ls
200030041 MINIX3: PID 193 swapped in
.cproject      README      WRITING_TESTS  src          workload_mix
.project       Run        pgms          testdir
Makefile      USAGE      results       tmp
# gmake_
```

Figure 4: run gmake command in UnixBench directory

```

200030041 MINIX3: PID 197 swapped in
gmake[11]: Entering directory '/root/byte-unixbench-mod/UnixBench'
200030041 MINIX3: PID 198 swapped in
200030041 MINIX3: PID 199 swapped in
200030041 MINIX3: PID 200 swapped in
Checking distribution of files
200030041 MINIX3: PID 201 swapped in
./pgms exists
200030041 MINIX3: PID 202 swapped in
./src exists
200030041 MINIX3: PID 203 swapped in
./testdir exists
200030041 MINIX3: PID 204 swapped in
./tmp exists
200030041 MINIX3: PID 205 swapped in
./results exists
gmake[11]: Leaving directory '/root/byte-unixbench-mod/UnixBench'
gmake programs
200030041 MINIX3: PID 206 swapped in
gmake[11]: Entering directory '/root/byte-unixbench-mod/UnixBench'
200030041 MINIX3: PID 207 swapped in
200030041 MINIX3: PID 208 swapped in
gmake[11]: Nothing to be done for 'programs'.
gmake[11]: Leaving directory '/root/byte-unixbench-mod/UnixBench'
#

```

Figure 5: Executables will created in UnixBench/pgms

2.b Executing the script files individually

```

# ./arithoh.sh
200030041 MINIX3: PID 215 swapped in
200030041 MINIX3: PID 216 swapped in
200030041 MINIX3: PID 217 swapped in
    14.66 real    14.60 user    0.06 sys
arithoh completed
----
#

```

Figure 6: arithoh.sh

```

# ./fstime.sh
200030041 MINIX3: PID 219 swapped in
200030041 MINIX3: PID 220 swapped in
200030041 MINIX3: PID 221 swapped in
Write done: 1008000 in 1.1833, score 212957
COUNT:212957:0:KBps
TIME:1.2
Read done: 1000004 in 0.9167, score 272728
COUNT:272728:0:KBps
TIME:0.9
Copy done: 1000004 in 1.9833, score 126050
COUNT:126050:0:KBps
TIME:2.0
      15.11 real      0.38 user      3.70 sys
fstime completed
---
# -

```

Figure 7: fstime.sh

```

# ./syscall.sh
200030041 MINIX3: PID 46 swapped in
200030041 MINIX3: PID 47 swapped in
200030041 MINIX3: PID 48 swapped in
      6.05 real      1.73 user      4.31 sys
syscall completed
---
# -

```

Figure 8: syscall.sh

```

# ./pipe.sh
200030041 MINIX3: PID 225 swapped in
200030041 MINIX3: PID 226 swapped in
200030041 MINIX3: PID 227 swapped in
        7.41 real        0.70 user        6.71 sys
pipe completed
---
#

```

Figure 9: pipe.sh

```

200030041 MINIX3: PID 250 swapped in
200030041 MINIX3: PID 251 swapped in
200030041 MINIX3: PID 252 swapped in
200030041 MINIX3: PID 253 swapped in
200030041 MINIX3: PID 254 swapped in
200030041 MINIX3: PID 255 swapped in
200030041 MINIX3: PID 7 swapped in
200030041 MINIX3: PID 11 swapped in
200030041 MINIX3: PID 15 swapped in
200030041 MINIX3: PID 18 swapped in
200030041 MINIX3: PID 33 swapped in
200030041 MINIX3: PID 35 swapped in
200030041 MINIX3: PID 36 swapped in
200030041 MINIX3: PID 37 swapped in
200030041 MINIX3: PID 38 swapped in
200030041 MINIX3: PID 39 swapped in
200030041 MINIX3: PID 40 swapped in
200030041 MINIX3: PID 41 swapped in
200030041 MINIX3: PID 42 swapped in
200030041 MINIX3: PID 43 swapped in
200030041 MINIX3: PID 44 swapped in
        6.93 real        0.08 user        5.88 sys
spawn completed
---
#

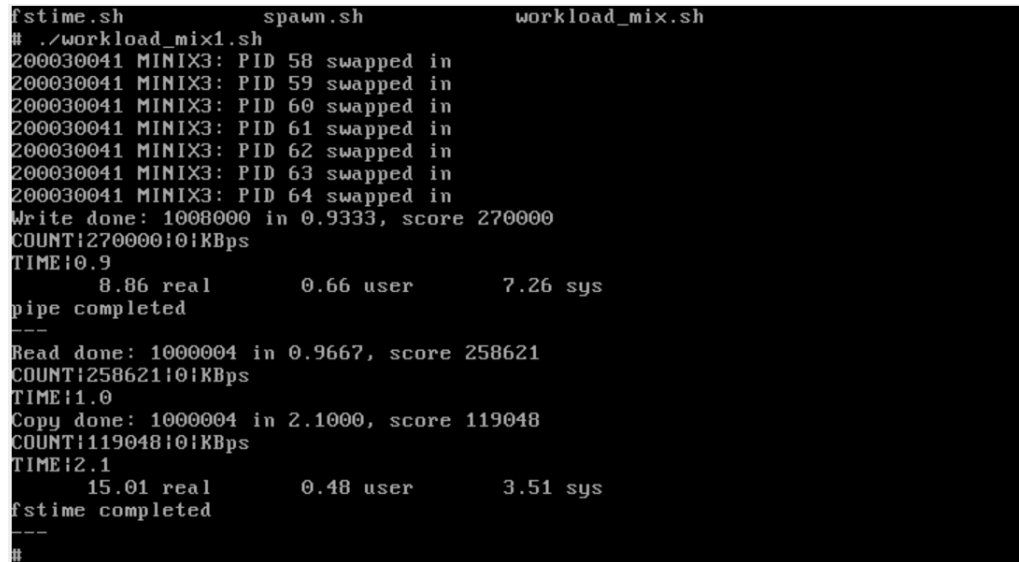
```

Figure 10: spawn.sh

2.c Customized workloads

2.c.1 work_load_mix1

```
10 #!/bin/sh
11 ./fstime.sh &
12 ./pipe.sh &
13 wait
```



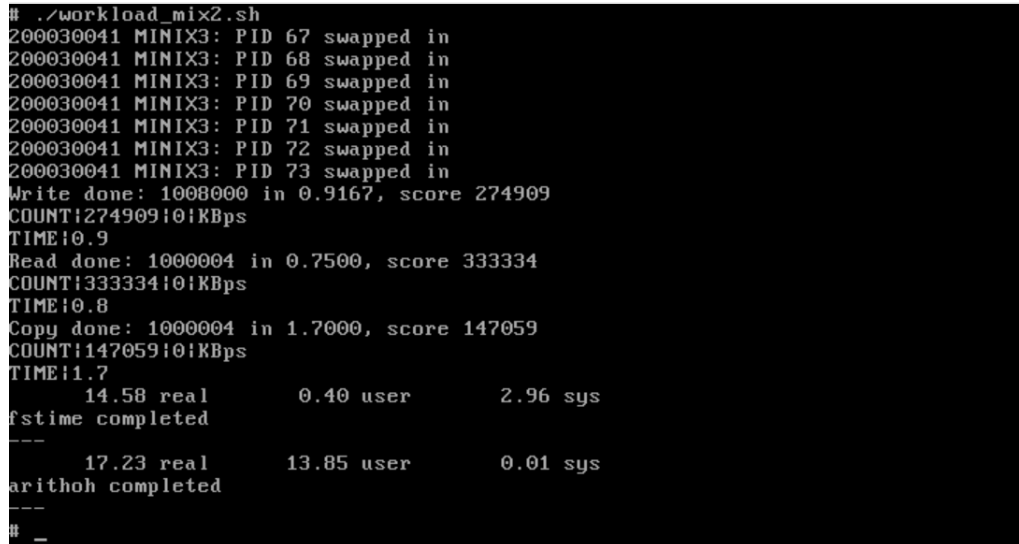
```
fstime.sh      spawn.sh      workload_mix.sh
# ./workload_mix1.sh
200030041 MINIX3: PID 58 swapped in
200030041 MINIX3: PID 59 swapped in
200030041 MINIX3: PID 60 swapped in
200030041 MINIX3: PID 61 swapped in
200030041 MINIX3: PID 62 swapped in
200030041 MINIX3: PID 63 swapped in
200030041 MINIX3: PID 64 swapped in
Write done: 1008000 in 0.9333, score 270000
COUNT:1270000:0:KBps
TIME:0.9
      8.86 real      0.66 user      7.26 sys
pipe completed
---
Read done: 1000004 in 0.9667, score 258621
COUNT:1258621:0:KBps
TIME:1.0
Copy done: 1000004 in 2.1000, score 119048
COUNT:119048:0:KBps
TIME:2.1
     15.01 real      0.48 user      3.51 sys
fstime completed
---
```

Figure 11: mix1

- The workload mix1 file consists of fstime(I/O bound) and pipe processes.
- Pipe processes complete earlier compared to I/O bound processes even though pipe.sh is placed below the fstime.sh in workload mix1 file.
- From the above point, we can infer that pipe processes have smaller CPU burst time as compared to I/O (fstime) bound processes.

2.c.2 work_load_mix2

```
14 #!/bin/sh
15 ./arithoh.sh &
16 ./fstime.sh &
17 wait
```



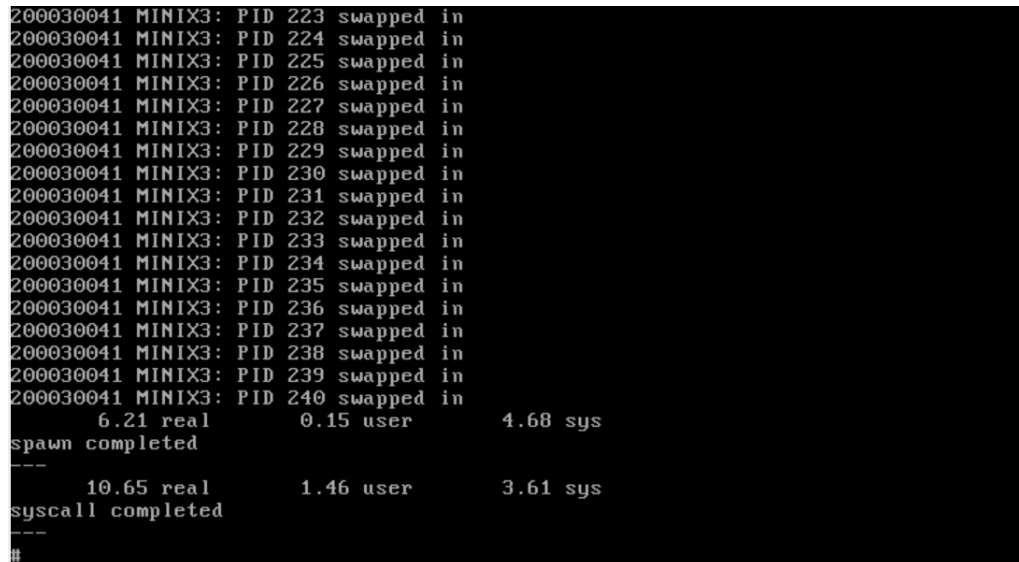
```
# ./workload_mix2.sh
200030041 MINIX3: PID 67 swapped in
200030041 MINIX3: PID 68 swapped in
200030041 MINIX3: PID 69 swapped in
200030041 MINIX3: PID 70 swapped in
200030041 MINIX3: PID 71 swapped in
200030041 MINIX3: PID 72 swapped in
200030041 MINIX3: PID 73 swapped in
Write done: 1008000 in 0.9167, score 274909
COUNT:1274909:0:KBps
TIME:0.9
Read done: 1000004 in 0.7500, score 333334
COUNT:1333334:0:KBps
TIME:0.8
Copy done: 1000004 in 1.7000, score 147059
COUNT:147059:0:KBps
TIME:1.7
    14.58 real        0.40 user        2.96 sys
fstime completed
---
    17.23 real       13.85 user        0.01 sys
arithoh completed
---
# -
```

Figure 12: mix2

- The workload mix2 file consists of computational intensive (CPU bound processes) and fstime (I/O bound) processes.
- From the output, we can observe that I/O bound processes have completed before the CPU bound processes because CPU bound processes have significantly more CPU burst time as compared to I/O bound processes.
- I/O bound processes have been given more priority compared to CPU bound processes to give a better response experience to the user.
- Even though fstime is below arithoh in the mix file, it finishes earlier as compared to processes in the arithoh exe file.

2.c.3 work_load_mix3

```
18 #!/bin/sh
19 ./spawn.sh &
20 ./syscall.sh &
21 wait
```



```
200030041 MINIX3: PID 223 swapped in
200030041 MINIX3: PID 224 swapped in
200030041 MINIX3: PID 225 swapped in
200030041 MINIX3: PID 226 swapped in
200030041 MINIX3: PID 227 swapped in
200030041 MINIX3: PID 228 swapped in
200030041 MINIX3: PID 229 swapped in
200030041 MINIX3: PID 230 swapped in
200030041 MINIX3: PID 231 swapped in
200030041 MINIX3: PID 232 swapped in
200030041 MINIX3: PID 233 swapped in
200030041 MINIX3: PID 234 swapped in
200030041 MINIX3: PID 235 swapped in
200030041 MINIX3: PID 236 swapped in
200030041 MINIX3: PID 237 swapped in
200030041 MINIX3: PID 238 swapped in
200030041 MINIX3: PID 239 swapped in
200030041 MINIX3: PID 240 swapped in
      6.21 real      0.15 user      4.68 sys
spawn completed
---
      10.65 real      1.46 user      3.61 sys
syscall completed
---
```

Figure 13: mix3

- The workload mix3 file consists of system call and recursively spawning processes.
- From the output, we can observe that spawning processes have completed before the system processes.
- System call will take more time to complete because when a system call is made, the user level process is changed to kernel level process and this takes time (mode switching overhead).