

# CS314: Lab Report

## Assignment 3 – Scheduling and UnixBench

B Siddharth Prabhu

200010003@iitdh.ac.in

Devdatt N

200010012@iitdh.ac.in

21 January 2023

### 1 Part I – MINIX3 Scheduler Swapping

**Task:** Modify the Minix3 source code such that:

- The string “Minix <roll-no> : PID <pid> swapped in” is printed, whenever a user-level process is brought in by the scheduler.

To do this, understand the working of the scheduler using online documentation of the same.

#### 1.1 Testing and Running

The file `schedule.c` located at `/usr/src/minix/servers/sched` is modified by inserting the desired print statement in the `schedule_process()` function.

#### 1.2 Screenshots

```
# sh arithoh.sh
Minix: PID 20479 created
Roll Number : 200010012 , PID 237 swapped in
Minix: PID 20480 created
Roll Number : 200010012 , PID 238 swapped in
Minix: PID 20481 created
Roll Number : 200010012 , PID 239 swapped in
Minix: PID 20481 exited
      10.21 real      10.03 user      0.18 sys
Minix: PID 20480 exited
arithoh completed
---
Minix: PID 20479 exited
# _
```

Figure 1: Process creation, exiting, and swapping in

```
# ls
Minix: PID 238 created
Minix 200010003: PID 214 swapped in
arithoh.sh      pipe.sh      syscall.sh
fstime.sh       spawn.sh     workload_mix.sh
Minix: PID 238 exited
```

Figure 2: Process creation, exiting, and swapping in

### 1.3 Some Remarks

- The files related to Scheduling are in `/usr/src/minix/servers/sched`, and we observe that the request to start scheduling a process occurs in `schedule.c`.
- As given in the reference textbook, Minix follows a multilevel queue for scheduling processes. Here, we have 16 queues from 0 to 15. To avoid system processes from coming in, we will only print out processes where `rmp>priority` is greater than `USER_Q`.
- A bash script `run.sh` is present in the submitted zip file, which copies the modified source files to the correct directories, and builds the system.

## 2 Part II – Analysis with UnixBench

**Task:** Conduct 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, and also studying online documentation of these benchmarks. Use commands such as “time”. (Note: refrain from using the graphics benchmarks in the suite.)

### 2.1 About UnixBench

UnixBench is the original BYTE UNIX benchmark suite, updated and revised by many people over the years. The main purpose of this tool is to provide a basic indicator of the performance of a Unix-like operating system. UnixBench can be used to evaluate the performance of a system when running a single or multiple tasks. This is a system benchmarking tool, not just a CPU, RAM or disk benchmark tool. The results will depend not only on the hardware, but also on the OS, libraries, and even compiler. In this assignment, we will analyze and use this suite for scheduling.

### 2.2 Initial Remarks

- We will first gauge whether the given benchmark programs for different scripts are I/O-bound or CPU-bound. Then, we will construct workload mixtures and observe the swapping in of processes. The `top` command is used to make these initial remarks.
- We observe that `arithoh` runs more CPU-bound processes. This could be due to the fact that repeated arithmetic operations are rather computationally intensive, and don't have an I/O component.
- The `fstime` script appears to run I/O-bound processes, based on it taking less proportion of the CPU for running. It reads, writes and copies files.
- The `spawn` benchmark goes through 100002 iterations where it forks, and immediately reaps the child that is created. This seems like a memory-bandwidth and process creation benchmark where both the bandwidth of the underlying system and the process creation mechanism of the Unix-based OS is tested. Additionally, this is a CPU bound process.
- The `pipe` script seems to be more I/O-bound; it involves repeated reads and writes to a pipe (which is a mechanism for inter-process communication). It also has less CPU usage, due to its smaller bursts.
- The `syscall` script seems to mainly consist of CPU-bound processes, which does system calls and deals with file descriptors.

## 2.3 Screenshots

```
main memory: 1046972K total, 996000K free, 994476K contig free, 23700K cached
CPU states: 98.85% user, 0.70% system, 0.45% kernel, 0.00% idle
CPU time displayed ('t' to cycle): user ; sort order ('o' to cycle): cpu

  PID USERNAME PRI NICE  SIZE  STATE  TIME   CPU COMMAND
  443 root      15  0    608K   RUN   0:39  98.63% arithoh
    -1 root      0    2802K   0:00  0.45% kernel
    11 root      2  0    5484K   0:13  0.26% vm
    7 root      5  0    1212K   2:09  0.23% vfs
    9 root      1  0     180K   0:14  0.08% tty
    40 root      7  0    1208K   RUN   0:00  0.08% procfs
   139 service   7  0    1152K   0:03  0.05% inet
   444 root      7  0     572K   0:00  0.05% top
    49 service   5  0    8244K   0:00  0.04% mfs
   134 root      7  0     112K   0:10  0.03% lance
    79 root      7  0     200K   0:00  0.02% devman
   107 root      7  0     188K   0:00  0.02% devmand
    5 root      4  0     596K   0:15  0.02% pm
   172 root      7  0     312K   0:01  0.01% syslogd
    73 service   5  0    19076K  0:00  0.01% mfs
   155 service   2  0     148K   0:00  0.00% log
Minix 200010003: PID 229 swapped in
Minix 200010003: PID 229 swapped in
Minix 200010003: PID 229 swapped in
```

Figure 3: CPU Usage of arithoh

```
main memory: 1046972K total, 996076K free, 994476K contig free, 23704K cached
CPU states: 5.27% user, 49.35% system, 24.36% kernel, 21.02% idle
CPU time displayed ('t' to cycle): user ; sort order ('o' to cycle): cpu

  PID USERNAME PRI NICE  SIZE  STATE  TIME   CPU COMMAND
    7 root      5  0    1212K   2:15  31.33% vfs
    -1 root      0    2802K   0:00  24.36% kernel
    76 service   5  0    4756K   0:25  17.32% mfs
   452 root      7  0     612K   RUN   0:00  5.03% fstime
    11 root      2  0    5484K   0:14  0.44% vm
    40 root      7  0    1208K   RUN   0:00  0.13% procfs
    9 root      1  0     180K   0:15  0.09% tty
   453 root      7  0     572K   0:00  0.08% top
   139 service   8  0    1152K   RUN   0:03  0.04% inet
    49 service   5  0    8244K   0:00  0.04% mfs
   134 root      7  0     112K   0:11  0.03% lance
    79 root      7  0     200K   0:00  0.02% devman
   107 root      7  0     188K   0:00  0.02% devmand
    5 root      4  0     596K   0:15  0.02% pm
    32 service   7  0     188K   0:00  0.01% at_wini
    73 service   5  0    19080K  0:00  0.01% mfs
Write done: 1000000 in 1.7000, score 148235
COUNT:148235:0:KBps
TIME:1.7
```

Figure 4: CPU Usage of fstime

```
load averages: 0.17, 0.18, 0.99
48 processes: 2 running, 46 sleeping
main memory: 1046972K total, 996076K free, 994476K contig free, 23704K cached
CPU states: 6.62% user, 64.55% system, 28.84% kernel, 0.00% idle
CPU time displayed ('t' to cycle): user ; sort order ('o' to cycle): cpu

  PID USERNAME PRI NICE  SIZE  STATE  TIME   CPU COMMAND
    7 root      5  0    1212K   2:24  50.24% vfs
    -1 root      0    2802K   0:00  28.84% kernel
    12 service   5  0     480K   0:30  13.85% pfs
   456 root      7  0     608K   RUN   0:00  6.42% pipe
    11 root      2  0    5484K   0:14  0.26% vm
    40 root      7  0    1208K   RUN   0:00  0.08% procfs
   139 service   7  0    1152K   0:03  0.06% inet
    9 root      1  0     180K   0:15  0.06% tty
   457 root      7  0     572K   0:00  0.06% top
   134 root      7  0     112K   0:11  0.04% lance
    49 service   5  0    8244K   0:00  0.04% mfs
    79 root      7  0     200K   0:00  0.03% devman
   107 root      7  0     188K   0:00  0.02% devmand
    5 root      4  0     596K   0:15  0.01% pm
    4 root      4  0    1196K   0:00  0.00% rs
   173 root      7  0     212K   0:00  0.00% dhcpg
Minix 200010003: PID 242 swapped in
```

Figure 5: CPU Usage of pipe

```

load averages: 0.13, 0.03, 0.01
48 processes: 2 running, 46 sleeping
main memory: 1046972K total, 975068K free, 975068K contig free, 43840K cached
CPU states: 21.37% user, 48.57% system, 30.06% kernel, 0.00% idle
CPU time displayed ('t' to cycle): user ; sort order ('o' to cycle): cpu

  PID USERNAME PRI NICE  SIZE  STATE  TIME   CPU COMMAND
    7 root      5    0   1212K   2:45  37.18% vfs
   -1 root      0    0   2802K   0:00  30.06% kernel
21608 root     11    0    608K   0:01  21.15% syscall
    5 root      4    0    596K   0:23  10.86% pm
   11 root      2    0   6252K   0:54   0.32% vm
   40 root      7    0   1208K   0:00   0.09% procfs
  139 service   7    0   1152K   0:07   0.08% inet
    9 root      1    0    180K   1:01   0.06% tty
  134 root      7    0    112K   0:22   0.06% lance
21609 root      7    0    572K   0:00   0.06% top
   49 service   5    0   8244K   0:00   0.04% mfs
   79 root      7    0    200K   0:00   0.02% devman
  107 root      7    0    188K   0:00   0.02% devmand
    6 root      4    0     48K   0:01   0.00% sched
  172 root      7    0    312K   0:05   0.00% syslogd
    4 root      4    0   1196K   0:00   0.00% rs
Minix 200010003: PID 34 swapped in

```

Figure 6: CPU Usage of syscall

## 2.4 Analysis of workload mixtures

### 2.4.1 arithoh and fstime

In this, we run a CPU-bound and an I/O bound process to observe the process swaps. The instructions of arithoh are CPU-intensive, while those of fstime are I/O-bound. We observe that arithoh swaps occur in the time where fstime waits for I/O. The code is as follows:

```

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

```

```

Roll Number : 200010012 , PID 63 swapped in
arithoh.sh      mix_files.zip      workload_mix.sh      workload_mix4.sh
fstime.sh       pipe.sh            workload_mix1.sh     workload_mix5.sh
log.txt         spawn.sh            workload_mix2.sh     workload_mix6.sh
logmix.txt      syscall.sh          workload_mix3.sh
Minix: PID 15140 exited
# sh workload_mix1.sh
Minix: PID 15141 created
Roll Number : 200010012 , PID 64 swapped in
Minix: PID 15142 created
Roll Number : 200010012 , PID 65 swapped in
Minix: PID 15143 created
Roll Number : 200010012 , PID 66 swapped in
Minix: PID 15144 created
Roll Number : 200010012 , PID 67 swapped in
Minix: PID 15145 created
Roll Number : 200010012 , PID 68 swapped in
Minix: PID 15146 created
Roll Number : 200010012 , PID 69 swapped in
Minix: PID 15147 created
Roll Number : 200010012 , PID 70 swapped in
Write done: 1000000 in 1.4000, score 100000
COUNT:100000:0:KBps
TIME:1.4

```

Figure 7: Swaps for workload 1

```

Minix: PID 15147 created
Roll Number : 200010012 , PID 70 swapped in
Write done: 1000000 in 1.4000, score 100000
COUNT:100000:0:KBps
TIME:1.4
Read done: 1000004 in 1.2500, score 200000
COUNT:200000:0:KBps
TIME:1.2
Minix: PID 15146 exited
      13.31 real      10.45 user      0.20 sys
Minix: PID 15144 exited
arithoh completed
---
Minix: PID 15142 exited
Copy done: 1000004 in 2.8833, score 86705
COUNT:86705:0:KBps
TIME:2.9
Minix: PID 15147 exited
      16.80 real      0.66 user      4.88 sys
Minix: PID 15145 exited
fstime completed
---
Minix: PID 15143 exited
Minix: PID 15141 exited
# _

```

Figure 8: Swaps for workload 1

#### 2.4.2 fstime and syscall

In this, we run a syscall and an I/O bound process to observe the process swaps. In this case, the CPU-intensive parts of the syscall code swap in when fstime waits for I/O. The code is as follows:

```

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

```

```

      16.80 real      0.66 user      4.88 sys
Minix: PID 15145 exited
fstime completed
---
Minix: PID 15143 exited
Minix: PID 15141 exited
# sh workload_mix2.sh
Minix: PID 15148 created
Roll Number : 200010012 , PID 71 swapped in
Minix: PID 15149 created
Roll Number : 200010012 , PID 72 swapped in
Minix: PID 15150 created
Roll Number : 200010012 , PID 73 swapped in
Minix: PID 15151 created
Roll Number : 200010012 , PID 74 swapped in
Minix: PID 15152 created
Roll Number : 200010012 , PID 75 swapped in
Minix: PID 15153 created
Roll Number : 200010012 , PID 76 swapped in
Minix: PID 15154 created
Roll Number : 200010012 , PID 77 swapped in
Write done: 1000000 in 1.3333, score 100000
COUNT:100000:0:KBps
TIME:1.3
_

```

Figure 9: Swaps for workload 2

```

Minix: PID 15154 created
Roll Number : 200010012 , PID 77 swapped in
Write done: 1008000 in 1.3333, score 188999
COUNT:188999:0:KBps
TIME:1.3
Minix: PID 15154 exited
      9.26 real      2.71 user      5.21 sys
Minix: PID 15152 exited
syscall completed
---
Minix: PID 15150 exited
Read done: 1000004 in 1.2667, score 197369
COUNT:197369:0:KBps
TIME:1.3
Copy done: 1000004 in 2.8667, score 87209
COUNT:87209:0:KBps
TIME:2.9
Minix: PID 15153 exited
      16.48 real     0.65 user      4.81 sys
Minix: PID 15151 exited
fstime completed
---
Minix: PID 15149 exited
Minix: PID 15148 exited
#

```

Figure 10: Swaps for workload 2

#### 2.4.3 syscall and arithoh

In this, we run a syscall and a CPU-bound process to observe the process swaps. Both of these processes swap alternatively, since both are primarily CPU-bound processes. The code is as follows:

```

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

```

```

COUNT:87209:0:KBps
TIME:2.9
Minix: PID 15153 exited
      16.48 real     0.65 user      4.81 sys
Minix: PID 15151 exited
fstime completed
---
Minix: PID 15149 exited
Minix: PID 15148 exited
# sh workload_mix3.sh
Minix: PID 15155 created
Roll Number : 200010012 , PID 78 swapped in
Minix: PID 15156 created
Roll Number : 200010012 , PID 79 swapped in
Minix: PID 15157 created
Roll Number : 200010012 , PID 80 swapped in
Minix: PID 15158 created
Roll Number : 200010012 , PID 81 swapped in
Minix: PID 15159 created
Roll Number : 200010012 , PID 82 swapped in
Minix: PID 15160 created
Roll Number : 200010012 , PID 83 swapped in
Minix: PID 15161 created
Roll Number : 200010012 , PID 84 swapped in

```

Figure 11: Swaps for workload 3

```

Roll Number : 200010012 , PID 79 swapped in
Minix: PID 15157 created
Roll Number : 200010012 , PID 80 swapped in
Minix: PID 15158 created
Roll Number : 200010012 , PID 81 swapped in
Minix: PID 15159 created
Roll Number : 200010012 , PID 82 swapped in
Minix: PID 15160 created
Roll Number : 200010012 , PID 83 swapped in
Minix: PID 15161 created
Roll Number : 200010012 , PID 84 swapped in
Minix: PID 15161 exited
          9.80 real          2.63 user          5.18 sys
Minix: PID 15159 exited
syscall completed
---
Minix: PID 15157 exited
Minix: PID 15160 exited
          18.43 real         10.33 user          0.26 sys
Minix: PID 15158 exited
arithoh completed
---
Minix: PID 15156 exited
Minix: PID 15155 exited
#

```

Figure 12: Swaps for workload 3

#### 2.4.4 repeated syscall

In this, we run repeated syscalls to observe the process swaps. The code is as follows:

```

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

```

```

---
Minix: PID 15156 exited
Minix: PID 15155 exited
# sh workload_mix4.sh
Minix: PID 15162 created
Roll Number : 200010012 , PID 85 swapped in
Minix: PID 15163 created
Roll Number : 200010012 , PID 86 swapped in
Minix: PID 15164 created
Roll Number : 200010012 , PID 87 swapped in
Minix: PID 15165 created
Roll Number : 200010012 , PID 88 swapped in
Minix: PID 15166 created
Roll Number : 200010012 , PID 89 swapped in
Minix: PID 15167 created
Roll Number : 200010012 , PID 90 swapped in
Minix: PID 15168 created
Roll Number : 200010012 , PID 92 swapped in
Minix: PID 15169 created
Roll Number : 200010012 , PID 93 swapped in
Minix: PID 15170 created
Roll Number : 200010012 , PID 94 swapped in
Minix: PID 15171 created
Roll Number : 200010012 , PID 95 swapped in

```

Figure 13: Swaps for workload 4

```

Roll Number : 200010012 , PID 93 swapped in
Minix: PID 15170 created
Roll Number : 200010012 , PID 94 swapped in
Minix: PID 15171 created
Roll Number : 200010012 , PID 95 swapped in
Minix: PID 15169 exited
      24.50 real      3.35 user      5.16 sys
Minix: PID 15166 exited
syscall completed
---
Minix: PID 15163 exited
Minix: PID 15171 exited
      24.61 real      2.90 user      5.21 sys
Minix: PID 15168 exited
syscall completed
---
Minix: PID 15165 exited
Minix: PID 15170 exited
      24.68 real      3.11 user      4.93 sys
Minix: PID 15167 exited
syscall completed
---
Minix: PID 15164 exited
Minix: PID 15162 exited
# _

```

Figure 14: Swaps for workload 4

#### 2.4.5 repeated arithoh

In this, we run repeated CPU-bound processes to observe the process swaps. The code is as follows:

```

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

```

```

---
Minix: PID 15164 exited
Minix: PID 15162 exited
# sh workload_mix5.sh
Minix: PID 15172 created
Roll Number : 200010012 , PID 96 swapped in
Minix: PID 15173 created
Roll Number : 200010012 , PID 97 swapped in
Minix: PID 15174 created
Roll Number : 200010012 , PID 98 swapped in
Minix: PID 15175 created
Roll Number : 200010012 , PID 99 swapped in
Minix: PID 15176 created
Roll Number : 200010012 , PID 100 swapped in
Minix: PID 15177 created
Roll Number : 200010012 , PID 101 swapped in
Minix: PID 15178 created
Roll Number : 200010012 , PID 102 swapped in
Minix: PID 15179 created
Roll Number : 200010012 , PID 103 swapped in
Minix: PID 15180 created
Roll Number : 200010012 , PID 104 swapped in
Minix: PID 15181 created
Roll Number : 200010012 , PID 105 swapped in

```

Figure 15: Swaps for workload 5



```

Roll Number : 200010012 , PID 103 swapped in
Minix: PID 15180 created
Roll Number : 200010012 , PID 104 swapped in
Minix: PID 15181 created
Roll Number : 200010012 , PID 105 swapped in
Minix: PID 15180 exited
      30.28 real      10.43 user          0.18 sys
Minix: PID 15177 exited
arithoh completed
---
Minix: PID 15174 exited
Minix: PID 15179 exited
      30.61 real      10.38 user          0.23 sys
Minix: PID 15176 exited
arithoh completed
---
Minix: PID 15173 exited
Minix: PID 15181 exited
      31.80 real      10.33 user          0.21 sys
Minix: PID 15178 exited
arithoh completed
---
Minix: PID 15175 exited
Minix: PID 15172 exited
# _

```

Figure 16: Swaps for workload 5

#### 2.4.6 repeated fstime

In this, we run repeated I/O-bound processes to observe the process swaps. The code is as follows:

```

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

```

```

---
Minix: PID 15175 exited
Minix: PID 15172 exited
# sh workload_mix6.sh
Minix: PID 15182 created
Roll Number : 200010012 , PID 106 swapped in
Minix: PID 15183 created
Roll Number : 200010012 , PID 107 swapped in
Minix: PID 15184 created
Roll Number : 200010012 , PID 108 swapped in
Minix: PID 15185 created
Roll Number : 200010012 , PID 109 swapped in
Minix: PID 15186 created
Roll Number : 200010012 , PID 110 swapped in
Minix: PID 15187 created
Roll Number : 200010012 , PID 111 swapped in
Minix: PID 15188 created
Roll Number : 200010012 , PID 112 swapped in
Minix: PID 15189 created
Roll Number : 200010012 , PID 113 swapped in
Minix: PID 15190 created
Roll Number : 200010012 , PID 114 swapped in
Minix: PID 15191 created
Roll Number : 200010012 , PID 115 swapped in

```

Figure 17: Swaps for workload 6

```

      28.00 real      0.68 user      4.75 sys
Minix: PID 15186 exited
fstime completed
---
Minix: PID 15183 exited
Copy done: 1000004 in 8.7667, score 28517
COUNT:28517:0:KBps
TIME:8.8
Minix: PID 15190 exited
      28.10 real      0.65 user      4.60 sys
Minix: PID 15187 exited
fstime completed
---
Minix: PID 15184 exited
Copy done: 1000004 in 8.8333, score 28302
COUNT:28302:0:KBps
TIME:8.8
Minix: PID 15191 exited
      28.13 real      0.71 user      5.71 sys
Minix: PID 15188 exited
fstime completed
---
Minix: PID 15185 exited
Minix: PID 15182 exited
# _

```

Figure 18: Swaps for workload 6

#### 2.4.7 Other output logs

```

Write done: 1008000 in 2.6500, score 95094
Read done: 1000004 in 2.3500, score 106383
Copy done: 1000004 in 5.1500, score 48543
fstime completed
---
arithoh completed
---

```

Figure 19: Output for workload 1

```

Write done: 1008000 in 2.5667, score 98181
Read done: 1000004 in 2.2833, score 109489
Copy done: 1000004 in 5.3500, score 46729
fstime completed
---
syscall completed
---

```

Figure 20: Output for workload 2

```

syscall completed
---
arithoh completed
---

```

Figure 21: Output for workload 3

```
syscall completed
---
syscall completed
---
syscall completed
---
~
```

Figure 22: Output for workload 4

```
arithoh completed
---
arithoh completed
---
arithoh completed
---
~
```

Figure 23: Output for workload 5

```
Write done: 1008000 in 10.6833, score 23588
Read done: 1000004 in 12.8000, score 19531
Copy done: 1000004 in 25.7667, score 9702
Write done: 1008000 in 10.6833, score 23588
Read done: 1000004 in 12.8000, score 19531
Copy done: 1000004 in 25.7667, score 9702
Write done: 1008000 in 10.6833, score 23588
Read done: 1000004 in 12.8000, score 19531
Copy done: 1000004 in 25.7667, score 9702
fstime completed
fstime completed
fstime completed
---
---
```

Figure 24: Output for workload 6