

# CS314 : Operating Systems Lab

## Lab 3

Sourabh Bhosale (200010004)  
Dibyashu Kashyap (200010013)

January 22, 2023

# 1 Part 1

## Problem

Modify the Minix3 source code such that the string "PID <pid> swapped in" is printed, whenever a user-level process is brought in by the scheduler.

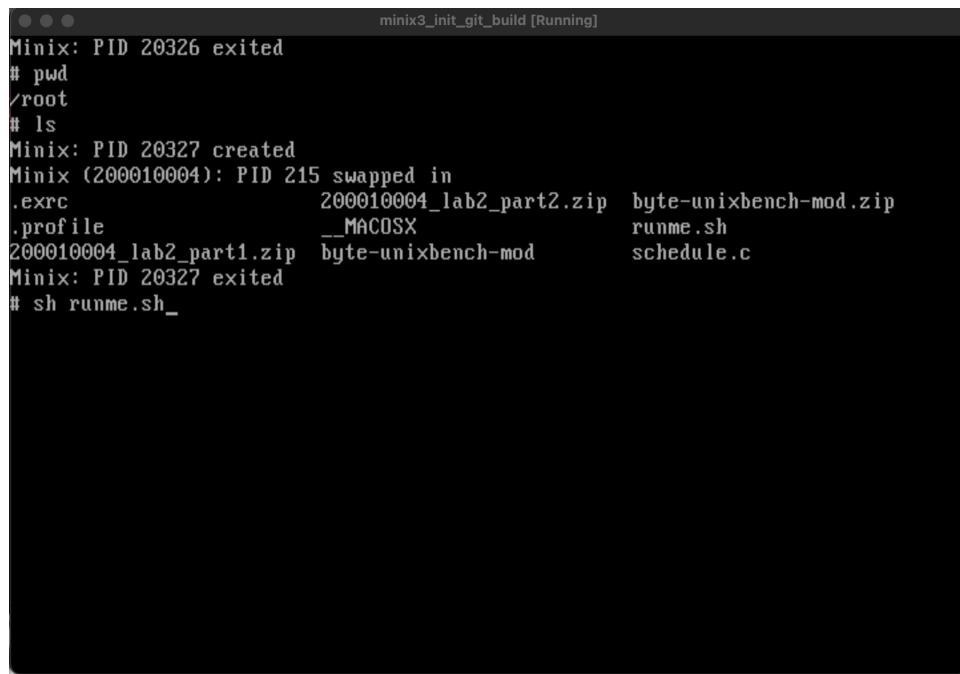
## Answer

In part 1, I have modified the Minix 3 source code such that the string "PID <pid> swapped in" is printed, whenever a user-level process is brought in by the scheduler.

To do this, I have added the following lines of C code in to the `schedule.c` file, located at `/minix/minix/servers/sched` directory.

```
if(rmp->priority >= USER_Q){  
    printf("Minix (200010004): PID %d swapped in\n", _ENDPOINT_P(rmp->endpoint));  
}
```

These lines of code were added to `schedule_process` function present in the said file. To make appropriate changes I have written a `runme.sh` bash file. Which when run, copies the modified `schedule.c` file into appropriate location and rebuilds the minix OS.



```
minix3_init_git_build [Running]  
Minix: PID 20326 exited  
# pwd  
/root  
# ls  
Minix: PID 20327 created  
Minix (200010004): PID 215 swapped in  
.exrc          200010004_lab2_part2.zip  byte-unixbench-mod.zip  
.profile        __MACOSX           runme.sh  
200010004_lab2_part1.zip  byte-unixbench-mod  schedule.c  
Minix: PID 20327 exited  
# sh runme.sh
```

Figure 1:

```
● ● ● minix3_init_git_build [Running]
install -M /usr/src/etc -c -p -r ../minix/servers/ds/ds /boot/minix/.temp/mod01_
ds
install -M /usr/src/etc -c -p -r ../minix/servers/rs/rs /boot/minix/.temp/mod02_
rs
install -M /usr/src/etc -c -p -r ../minix/servers/pm/pm /boot/minix/.temp/mod03_
pm
install -M /usr/src/etc -c -p -r ../minix/servers/sched/sched /boot/minix/.temp/
mod04_sched
install -M /usr/src/etc -c -p -r ../minix/servers/vfs/vfs /boot/minix/.temp/mod0
5_vfs
install -M /usr/src/etc -c -p -r ../minix/drivers/storage/memory/memory /boot/mi
nix/.temp/mod06_memory
install -M /usr/src/etc -c -p -r ../minix/drivers/tty/tty/tty /boot/minix/.temp/
mod07_tty
install -M /usr/src/etc -c -p -r ../minix/fs/mfs/mfs /boot/minix/.temp/mod08_mfs
install -M /usr/src/etc -c -p -r ../minix/servers/vm/vm /boot/minix/.temp/mod09_
vm
install -M /usr/src/etc -c -p -r ../minix/fs/pfs/pfs /boot/minix/.temp/mod10_pfs
install -M /usr/src/etc -c -p -r ../sbin/init/init /boot/minix/.temp/mod11_init
rm /dev/c0d0p0s0:/boot/minix/3.3.0r5
Done.
Build started at: Sun Jan 22 11:15:42 GMT 2023
Build finished at: Sun Jan 22 11:23:31 GMT 2023
Minix: PID 14918 exited
#
```

Figure 2:

## 2 Part 2

### Problem

You may study the behavior of the scheduler by seeing the sequence of "PID" prints when these workloads are run.

#### **workload\_mix1.sh**

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

We can expect the following behavior:

The `arithoh.sh` script, is likely to run benchmark tests that measure the performance of the CPU, specifically the overhead of arithmetic operations. examining the code we can easily see there are computationally intensive operations inside two nested loops.

The `fstime.sh` script is likely to run benchmark tests that measure the performance of the file system, specifically the time taken to perform various file operations.

The third command "wait" is used to wait for all background processes to complete. So that the workload mix sh process exits only after all the inside processes are exited.

As both arithoh.sh and fstime.sh are running in parallel and as both are intensive in their own nature, the scheduler will likely have to switch frequently between the two processes in order to allocate resources accordingly. The "wait" command will make sure that the script will wait for both the scripts to complete before exiting.

In terms of schedule orders, the scheduler will likely prioritize the CPU-intensive arithoh.sh process over the I/O-intensive fstime.sh process when the system is under heavy load. However, when the system is less loaded, the scheduler may give more resources to the fstime.sh process in order to complete file operations quickly.

In workload mix1.sh, an instance of file arithoh.sh(217) and an instance of file fstime.sh(218) are run as shown in fig 3-4.

```
● ● ● minix3_init_git_build [Running]
Minix (200010004): PID 207 swapped in
Read done: 1000004 in 1.1500, score 217392
COUNT:217392:0:Kbps
TIME:1.1
Minix (200010004): PID 207 swapped in
Minix: PID 232 exited
    13.20 real      7.61 user      3.11 sys
Minix: PID 230 exited
arithoh completed
---
Minix: PID 228 exited
```

Figure 3:

```
● ● ● minix3_init_git_build [Running]
Minix: PID 259 exited
Minix (200010004): PID 28 swapped in
Minix (200010004): PID 28 swapped in
Minix (200010004): PID 239 swapped in
COUNT:109489:0:Kbps
TIME:2.3
Minix: PID 264 exited
    15.86 real      0.38 user      4.21 sys
Minix: PID 262 exited
Minix: PID 260 exited
Minix: PID 258 exited
# cat log.txt
Minix: PID 265 created
Minix (200010004): PID 240 swapped in
arithoh completed
---
Write done: 1008000 in 1.2167, score 207123
Read done: 1000004 in 1.1000, score 227273
Copy done: 1000004 in 2.2833, score 109489
fstime completed
---
Minix: PID 265 exited
```

Figure 4:

## **workload\_mix2.sh**

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

We can expect the following behavior:

The `./arithoh.sh` script in the background. This script is likely to run benchmark tests that measure the performance of the CPU, specifically the overhead of arithmetic operations.

The second command, `./arithoh.sh` runs another instance of the arithoh.sh script in the background.

The third command, `./arithoh.sh` runs another instance of the arithoh.sh script in the background. Now, the scheduler have to switch frequently between three processes.

As all the three instances of arithoh.sh are running in parallel and as all are intensive in their own nature, the scheduler will have to switch frequently between the three processes in order to allocate resources accordingly.

In here, first arithoh.sh has PID of 248, second arithoh.sh has PID 249 and the last arithoh.sh has PID 250. Clearly, if multiple instances are run of arithoh.sh which are CPU Intensive in nature as in this workload, it can be seen that th's getting scheduled alternatively. Process with PID 248, 249, 250 correspond to 3 instances of workload in arithoh.sh that are scheduled alternatively until they are completed as shown in fig 5 and 6.

```
● ● ● minix3_init_git_build [Running]
Minix (200010004): PID 248 swapped in
Minix (200010004): PID 249 swapped in
Minix (200010004): PID 250 swapped in
Minix (200010004): PID 249 swapped in
Minix (200010004): PID 248 swapped in
Minix (200010004): PID 250 swapped in
Minix (200010004): PID 248 swapped in
Minix (200010004): PID 250 swapped in
Minix (200010004): PID 249 swapped in
Minix (200010004): PID 250 swapped in
Minix (200010004): PID 248 swapped in
Minix (200010004): PID 250 swapped in
Minix (200010004): PID 249 swapped in
Minix (200010004): PID 248 swapped in
Minix (200010004): PID 250 swapped in
Minix (200010004): PID 249 swapped in
Minix (200010004): PID 250 swapped in
Minix (200010004): PID 248 swapped in
Minix (200010004): PID 249 swapped in
Minix (200010004): PID 250 swapped in
Minix (200010004): PID 248 swapped in
Minix (200010004): PID 249 swapped in
Minix (200010004): PID 250 swapped in
Minix (200010004): PID 248 swapped in
Minix (200010004): PID 249 swapped in
Minix (200010004): PID 250 swapped in
Minix (200010004): PID 248 swapped in
Minix (200010004): PID 249 swapped in
Minix (200010004): PID 250 swapped in
Minix (200010004): PID 248 swapped in
Minix (200010004): PID 249 swapped in
Minix (200010004): PID 250 swapped in
Minix (200010004): PID 248 swapped in
```

Figure 5:

```
● ● ● minix3_init_git_build [Running]
arithoh completed
---
Minix: PID 268 exited
Minix (200010004): PID 248 swapped in
Minix (200010004): PID 250 swapped in
Minix: PID 273 exited
    29.55 real      7.88 user      2.58 sys
Minix: PID 270 exited
arithoh completed
---
Minix: PID 267 exited
Minix (200010004): PID 250 swapped in
Minix: PID 275 exited
    31.33 real      8.10 user      2.61 sys
Minix: PID 272 exited
arithoh completed
---
Minix: PID 269 exited
Minix: PID 266 exited
#
```

Figure 6:

## **workload\_mix3.sh**

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

We can expect the following behavior:

The first command, `./arithoh.sh`, runs the arithoh.sh script in the background.

The second command, `./syscall.sh` runs the syscall.sh script in the background. This script is likely to run benchmark tests that measure the performance of system calls, specifically the overhead of system calls.

As both arithoh.sh and syscall.sh are running in parallel and as both are intensive in their own nature, the scheduler will likely have to switch frequently between the two processes in order to allocate resources accordingly. The "wait" command will make sure that the script will wait for both the scripts to complete before exiting.

In terms of schedule orders, the scheduler will likely prioritize the CPU-intensive arithoh.sh process over the syscall-intensive syscall.sh process when the system is under heavy load. However, when the system is less loaded, the scheduler may give more resources to syscall.sh process in order to complete system calls quickly. as shown in fig 8-11.

In here, arithoh.sh has PID 7 and syscall.sh has PID 11 as shown in fig 7-9.

```
# ./workload_mix3.sh
Minix: PID 276 created
Minix (200010004): PID 251 swapped in
Minix: PID 277 created
Minix (200010004): PID 252 swapped in
Minix: PID 278 created
Minix (200010004): PID 253 swapped in
Minix: PID 279 created
Minix (200010004): PID 254 swapped in
Minix: PID 280 created
Minix (200010004): PID 255 swapped in
Minix: PID 281 created
Minix (200010004): PID 7 swapped in
Minix: PID 282 created
Minix (200010004): PID 11 swapped in
Minix (200010004): PID 7 swapped in
Minix (200010004): PID 11 swapped in
Minix (200010004): PID 7 swapped in
Minix (200010004): PID 11 swapped in
Minix (200010004): PID 7 swapped in
Minix (200010004): PID 11 swapped in
Minix (200010004): PID 7 swapped in
Minix (200010004): PID 27 swapped in
Minix (200010004): PID 11 swapped in
```

Figure 7:

```
● ● ● minix3_init_git_build [Running]
Minix (200010004): PID 7 swapped in
Minix (200010004): PID 11 swapped in
Minix (200010004): PID 7 swapped in
Minix (200010004): PID 7 swapped in
Minix (200010004): PID 11 swapped in
Minix (200010004): PID 27 swapped in
Minix: PID 282 exited
    8.03 real      1.75 user      4.60 sys
Minix: PID 280 exited
syscall completed
---
Minix: PID 278 exited
Minix (200010004): PID 7 swapped in
```

Figure 8:

```
● ● ● minix3_init_git_build [Running]
Minix (200010004): PID 7 swapped in
Minix: PID 281 exited
    17.20 real      8.08 user      2.75 sys
Minix: PID 279 exited
arithoh completed
---
Minix: PID 277 exited
Minix: PID 276 exited
#
```

Figure 9:

## **workload\_mix4.sh**

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

We can expect the following behavior:

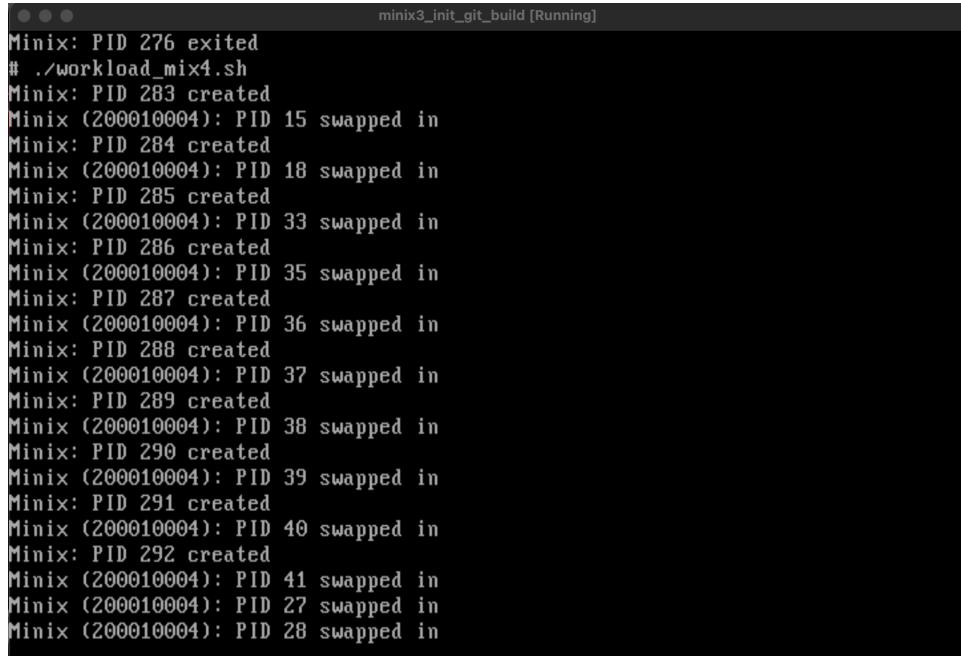
The first command, `./fstime.sh` runs the fstime.sh script in the background. This script is likely to run benchmark tests that measure the performance of the file system, specifically the time taken to perform various file operations.

The second command, `./fstime.sh`, runs another instance of the fstime.sh script in the background. As this script is also I/O-intensive, the scheduler will have to switch frequently between the two processes in order to allocate resources accordingly.

The third command, `./fstime.sh`, runs another instance of the fstime.sh script in the background. Now, the scheduler have to switch frequently between three processes.

The fourth command "wait" is used to wait for all background processes to complete.

As all the three instances of fstime.sh are running in parallel and as all are I/O-intensive in their own nature, the scheduler will have to switch frequently between the three processes in order to allocate resources accordingly. The "wait" command will make sure that the script will wait for all the three scripts to complete before exiting.



```
minix3_init_git_build [Running]
Minix: PID 276 exited
# ./workload_mix4.sh
Minix: PID 283 created
Minix (200010004): PID 15 swapped in
Minix: PID 284 created
Minix (200010004): PID 18 swapped in
Minix: PID 285 created
Minix (200010004): PID 33 swapped in
Minix: PID 286 created
Minix (200010004): PID 35 swapped in
Minix: PID 287 created
Minix (200010004): PID 36 swapped in
Minix: PID 288 created
Minix (200010004): PID 37 swapped in
Minix: PID 289 created
Minix (200010004): PID 38 swapped in
Minix: PID 290 created
Minix (200010004): PID 39 swapped in
Minix: PID 291 created
Minix (200010004): PID 40 swapped in
Minix: PID 292 created
Minix (200010004): PID 41 swapped in
Minix (200010004): PID 27 swapped in
Minix (200010004): PID 28 swapped in
```

Figure 10:

```
minix3_init_git_build [Running]
Minix: PID 292 created
Minix (200010004): PID 41 swapped in
Minix (200010004): PID 27 swapped in
Minix (200010004): PID 28 swapped in
Minix (200010004): PID 27 swapped in
Minix (200010004): PID 28 swapped in
Write done: 1008000 in 3.4000, score 74117
Write done: 1008000 in 3.4000, score 74117
Write done: 1008000 in 3.4000, score 74117
COUNT:74117:0:KBps
COUNT:74117:0:KBps
COUNT:74117:0:KBps
TIME:3.4
TIME:3.4
TIME:3.4
Read done: 1000004 in 3.1667, score 78947
Read done: 1000004 in 3.1667, score 78947
Read done: 1000004 in 3.1667, score 78947
COUNT:78947:0:KBps
COUNT:78947:0:KBps
COUNT:78947:0:KBps
TIME:3.2
TIME:3.2
TIME:3.2
```

Figure 11:

```
minix3_init_git_build [Running]
24.10 real      0.38 user      3.70 sys
Minix: PID 288 exited
fstime completed
---
Minix: PID 285 exited
Copy done: 1000004 in 6.5167, score 38363
COUNT:38363:0:KBps
TIME:6.5
Minix: PID 290 exited
24.18 real      0.36 user      4.63 sys
Minix: PID 287 exited
fstime completed
---
Minix: PID 284 exited
Copy done: 1000004 in 6.9333, score 36057
COUNT:36057:0:KBps
TIME:6.9
Minix: PID 292 exited
24.58 real      0.26 user      4.18 sys
Minix: PID 289 exited
fstime completed
---
Minix: PID 286 exited
Minix: PID 283 exited
#
```

Figure 12:

## **workload\_mix5.sh**

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

We can expect the following behavior:

The first command, `./arithoh.sh`, runs the arithoh.sh script in the background.

The `pipe.sh` script is likely to run benchmark tests that measure the performance of the file system, specifically the time taken to perform various file operations.

As both arithoh.sh and pipe.sh are running in parallel and as both are intensive in their own nature, the scheduler will likely have to switch frequently between the two processes in order to allocate resources accordingly.

In terms of schedule orders, the scheduler will likely prioritize the CPU-intensive arithoh.sh process over the pipe-intensive pipe.sh process when the system is under heavy load. However, when the system is less loaded, the scheduler may give more resources to pipe.sh process in order to complete pipe operations quickly. 93 is the PID of arithoh.sh and pipe.sh has PID 94 as shown in fig 13-14

```
minix3_init_git_build [Running]
Minix (200010004): PID 93 swapped in
Minix (200010004): PID 93 swapped in
Minix (200010004): PID 93 swapped in
Minix (200010004): PID 94 swapped in
Minix (200010004): PID 93 swapped in
Minix (200010004): PID 93 swapped in
Minix (200010004): PID 94 swapped in
Minix: PID 10328 exited
    9.08 real      0.85 user      7.58 sys
Minix: PID 10326 exited
pipe completed
---
Minix: PID 10324 exited
Minix (200010004): PID 93 swapped in
```

Figure 13:

```
minix3_init_git_build [Running]
Minix (200010004): PID 93 swapped in
Minix: PID 10327 exited
    19.05 real      9.95 user      0.66 sys
Minix: PID 10325 exited
arithoh completed
---
Minix: PID 10323 exited
Minix: PID 10322 exited
#
```

Figure 14:

## **workload\_mix6.sh**

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

In this workload, we have used two workloads, namely `arithoh.sh` and `spawn.sh`. After looking at their source code and the order of scheduling, We came to know that both `arithoh.sh` and `spawn.sh` are CPU bound.

The `arithoh.sh` does some CPU bound arithmetic calculations where as `spawn.sh` keeps creating (forks) new process which are exited immediately.

From the figure, we can see that new process are continuously created by `spawn.sh` and are being scheduled as well to minimize the response time. These newly created process exit immediately, which is observed from the source code.

After some time, the `spawn.sh` with along with the newly created process terminate as seen from figure 16-18.

Then, the `arithoh.sh`, with PID 65 is scheduled until completion as seen from figure 15-17.

```
minix3_init_git_build [Running]
Minix: PID 4398 created
Minix (200010004): PID 206 swapped in
Minix: PID 4398 exited
Minix: PID 4399 created
Minix (200010004): PID 207 swapped in
Minix: PID 4399 exited
Minix: PID 4400 created
Minix (200010004): PID 208 swapped in
Minix: PID 4400 exited
Minix: PID 4401 created
Minix (200010004): PID 209 swapped in
Minix: PID 4401 exited
Minix: PID 4402 created
Minix (200010004): PID 210 swapped in
Minix: PID 4402 exited
Minix: PID 4403 created
Minix (200010004): PID 211 swapped in
Minix: PID 4403 exited
Minix: PID 4404 created
Minix (200010004): PID 212 swapped in
Minix: PID 4404 exited
Minix: PID 4405 created
Minix (200010004): PID 213 swapped in
Minix: PID 4405 exited
```

Figure 15:

```
● ● ● minix3_init_git_build [Running]
Minix: PID 10315 created
Minix (200010004): PID 80 swapped in
Minix: PID 10315 exited
Minix: PID 10316 created
Minix (200010004): PID 81 swapped in
Minix: PID 10316 exited
Minix: PID 10317 created
Minix (200010004): PID 82 swapped in
Minix: PID 10317 exited
Minix: PID 10318 created
Minix (200010004): PID 83 swapped in
Minix: PID 10318 exited
Minix: PID 317 exited
    14.56 real      0.18 user      7.43 sys
Minix: PID 315 exited
spawn completed
---
Minix: PID 313 exited
Minix (200010004): PID 65 swapped in
```

Figure 16:

```
● ● ● minix3_init_git_build [Running]
Minix (200010004): PID 65 swapped in
Minix: PID 316 exited
    23.60 real      9.11 user      0.76 sys
Minix: PID 314 exited
arithoh completed
---
Minix: PID 312 exited
Minix: PID 311 exited
#
```

Figure 17: