CS 314 Lab 3

MV Karthik - 200010030

Josyula V N Taraka Abhishek - 200010021

Jan 2023

Contents

1 Part - I **1**

2 Part-II **2**

2.1 Custom Mix - 1 3
2.2 Custom Mix - 2 4
2.3 Custom Mix - 3 5
2.4 Conclusions 6
2.5 Runnig the mixes 7

List of Figures

1 Process swapped in statements are printed successfully. 2
2 Process scheduling sequence for mix1 or log messages 4
3 Process scheduling sequence for mix2 or log messages 5
4 Process scheduling sequence for mix3 or log messages 6
5 Run mix1 7
6 Run mix2 7
7 Run mix3 8

1 Part - I

We are asked to log a statement "Minix: process PID <pid> swapped in" for each user process that is brought in by the scheduler. We observed that the **sched** server is calling kernel call to schedule a process in the function `schedule_process.c`. Which is used by `do_start_schedule`, `balance_queues`, `do_noquantum` and `do_nice` entry point functions. Hence we included the below if statement in the schedule_process function.

```
if (err == OK && _ENDPOINT_P(rmp->endpoint)>=0)
    printf("200010030 Minix: PID \%d swapped in.\n",
    _ENDPOINT_P(rmp->endpoint));
  }
```

The function _ENDPOINT_P returns the process slot number of the process from the given endpoint, which is similar to the notion of PID. A non-negative process slot number indicates that it is a user process, else it will be a kernel process. (Described in `endpoint.h` file)

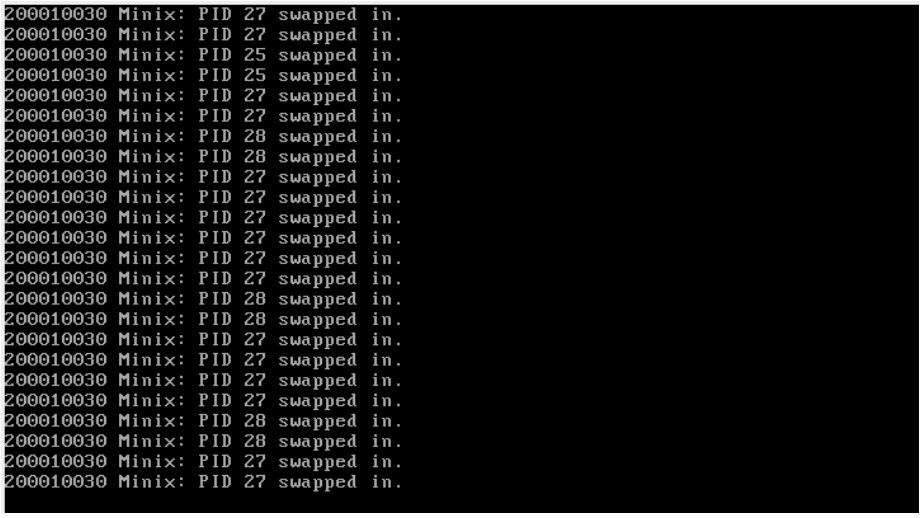

Figure 1: Process swapped in statements are printed successfully.

2 Part-II

We have run the benchmarks on the minix system by various custom workload mixes to find the nature of the unix benchmarks(I/O bound or CPU bound).

1. From the analysis of the code, we see that arithoh.sh is CPU intensive
2. fstime is IO intensive
3. syscall has lots of blocking code (syscalls).

The following code is used to obtain log messages

```
#! /bin/sh

cat "" > /var/log/messages
bash $1
cat /var/log/messages > $1.txt
```

2.1 Custom Mix - 1

Custom mix 1 contains 4 concurrent runs of arithoh.sh and the messages are printed into log.txt

```
    #! /bin/sh

./arithoh.sh &
./arithoh.sh &
./arithoh.sh &
./arithoh.sh &
wait
```

We can see the PID's of four process swapping frequently from log messages

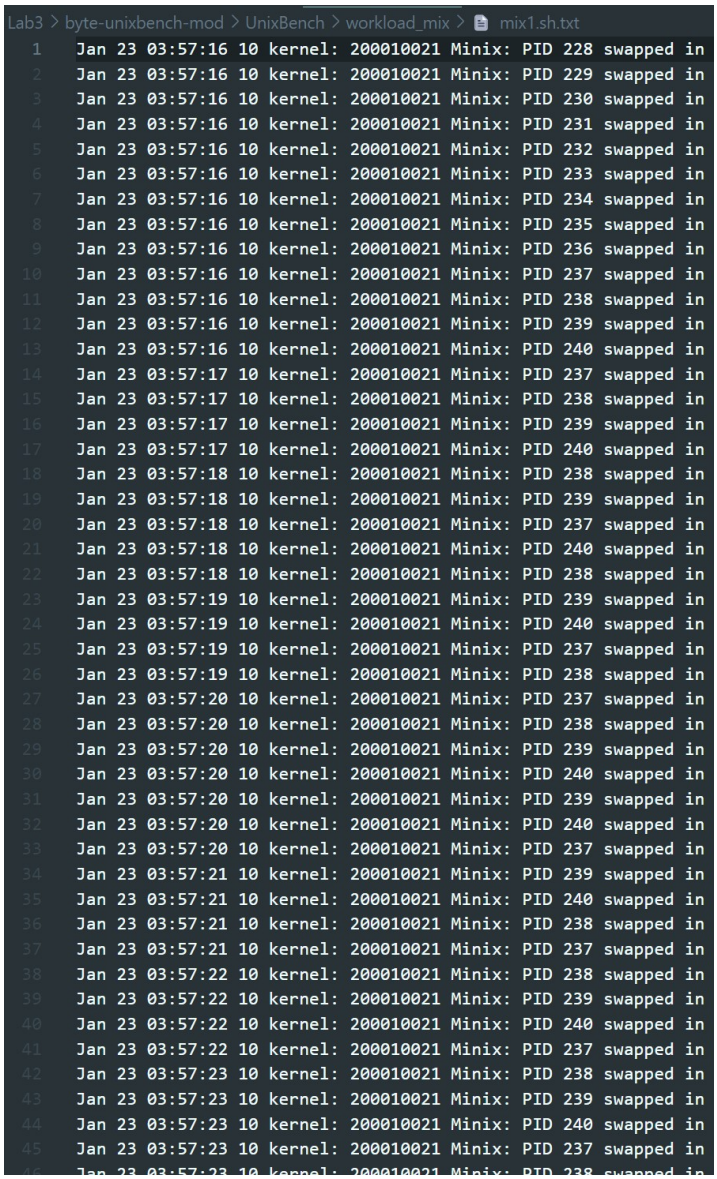

Figure 2: Process scheduling sequence for mix1 or log messages

2.2 Custom Mix - 2

Custom mix 2 contains two benchmarks `arithoh.sh` and `syscall.sh` running in parallel.

From log message we see that PID 47 is swapped in 31 times so it is arithoh.sh and 48 is syscall.sh

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

Lab3 > byte-unixbench-mod > UnixBench > workload_mix > mix2.sh.txt

```
Jan 23 04:05:07 10 kernel: 200010021 Minix: PID 40 swapped in
Jan 23 04:05:07 10 kernel: 200010021 Minix: PID 41 swapped in
Jan 23 04:05:07 10 kernel: 200010021 Minix: PID 42 swapped in
Jan 23 04:05:07 10 kernel: 200010021 Minix: PID 43 swapped in
Jan 23 04:05:07 10 kernel: 200010021 Minix: PID 44 swapped in
Jan 23 04:05:07 10 kernel: 200010021 Minix: PID 45 swapped in
Jan 23 04:05:07 10 kernel: 200010021 Minix: PID 46 swapped in
Jan 23 04:05:07 10 kernel: 200010021 Minix: PID 47 swapped in
Jan 23 04:05:07 10 kernel: 200010021 Minix: PID 48 swapped in
Jan 23 04:05:07 10 kernel: 200010021 Minix: PID 47 swapped in
Jan 23 04:05:08 10 kernel: 200010021 Minix: PID 48 swapped in
Jan 23 04:05:08 10 kernel: 200010021 Minix: PID 47 swapped in
Jan 23 04:05:09 10 kernel: 200010021 Minix: PID 48 swapped in
Jan 23 04:05:09 10 kernel: 200010021 Minix: PID 47 swapped in
Jan 23 04:05:10 10 kernel: 200010021 Minix: PID 47 swapped in
Jan 23 04:05:10 10 kernel: 200010021 Minix: PID 48 swapped in
Jan 23 04:05:10 10 kernel: 200010021 Minix: PID 48 swapped in
Jan 23 04:05:11 10 kernel: 200010021 Minix: PID 47 swapped in
Jan 23 04:05:12 10 kernel: 200010021 Minix: PID 48 swapped in
Jan 23 04:05:12 10 kernel: 200010021 Minix: PID 47 swapped in
Jan 23 04:05:13 10 kernel: 200010021 Minix: PID 48 swapped in
Jan 23 04:05:13 10 kernel: 200010021 Minix: PID 47 swapped in
Jan 23 04:05:19 10 last message repeated 31 times
Jan 23 04:05:19 10 kernel: 200010021 Minix: PID 49 swapped in
```

Figure 3: Process scheduling sequence for mix2 or log messages

2.3 Custom Mix - 3

This workload mix contains benchmarks `arithoh.sh` and `fstime.sh` running in parallel.

From the log message, we see that PID 57 is swapped in 35 times, so it is arithoh.sh and 58 is syscall.sh .

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

```
./fstime.sh &
wait
```

Lab3 > byte-unixbench-mod > UnixBench > workload_mix > mix3.sh.txt

```
Jan 23 04:05:23 10 kernel: 200010021 Minix: PID 52 swapped in
Jan 23 04:05:23 10 kernel: 200010021 Minix: PID 53 swapped in
Jan 23 04:05:23 10 kernel: 200010021 Minix: PID 54 swapped in
Jan 23 04:05:23 10 kernel: 200010021 Minix: PID 55 swapped in
Jan 23 04:05:23 10 kernel: 200010021 Minix: PID 56 swapped in
Jan 23 04:05:23 10 kernel: 200010021 Minix: PID 57 swapped in
Jan 23 04:05:23 10 kernel: 200010021 Minix: PID 58 swapped in
Jan 23 04:05:24 10 kernel: 200010021 Minix: PID 57 swapped in
Jan 23 04:05:32 10 last message repeated 35 times
Jan 23 04:05:33 10 kernel: 200010021 Minix: PID 23 swapped in
Jan 23 04:05:35 10 kernel: 200010021 Minix: PID 23 swapped in
Jan 23 04:05:38 10 kernel: 200010021 Minix: PID 58 swapped in
Jan 23 04:05:38 10 kernel: 200010021 Minix: PID 59 swapped in
```

Figure 4: Process scheduling sequence for mix3 or log messages

2.4 Conclusions

From mix1, we can see that the scheduler is giving the multiple processes of same `arithoh.sh` are getting scheduled alternately(Round robin way).

From mix2, We can see the scheduler is giving `arithoh.sh` less time slice, and it is scheduled more times than the more blocked process of `syscall.sh`

From mix3, We can see the scheduler is giving `arithoh.sh` less time slice, and it is scheduled more times than the more I/O blocked the process of `fstime.sh`

2.5 Runnig the mixes

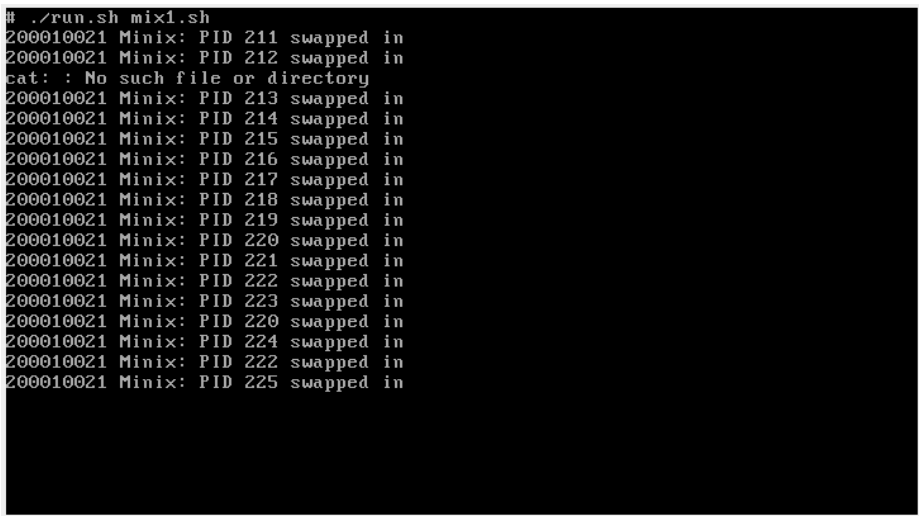

Figure 5: Run mix1

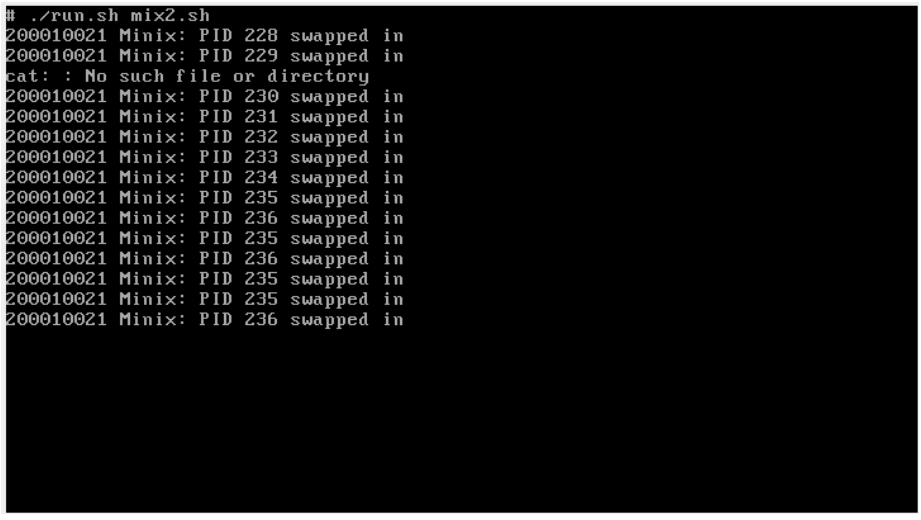

Figure 6: Run mix2

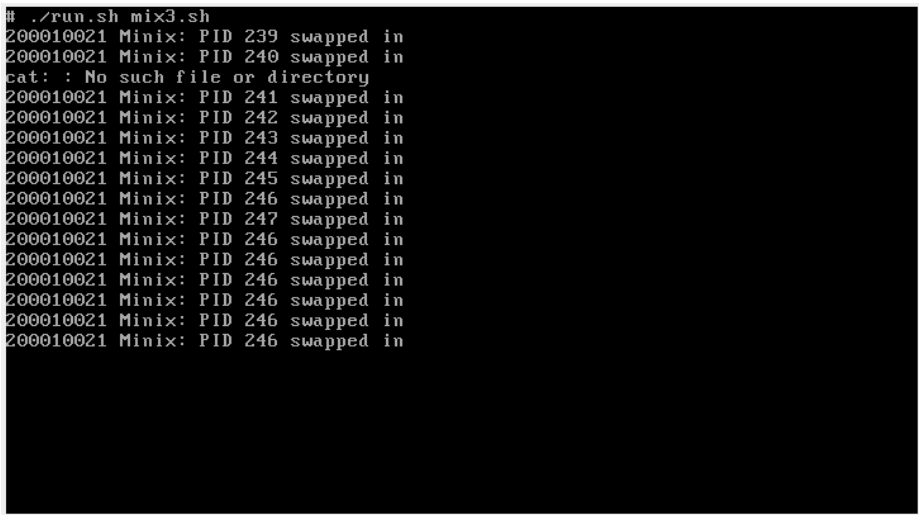

Figure 7: Run mix3