OS LAB - 3

Report

Name: Ashish Kupsad **Roll no:** 180020003

Part I:

The code is added in the function ***schedule_process()*** present in the file *minix/servers/sched/schedule.c*.

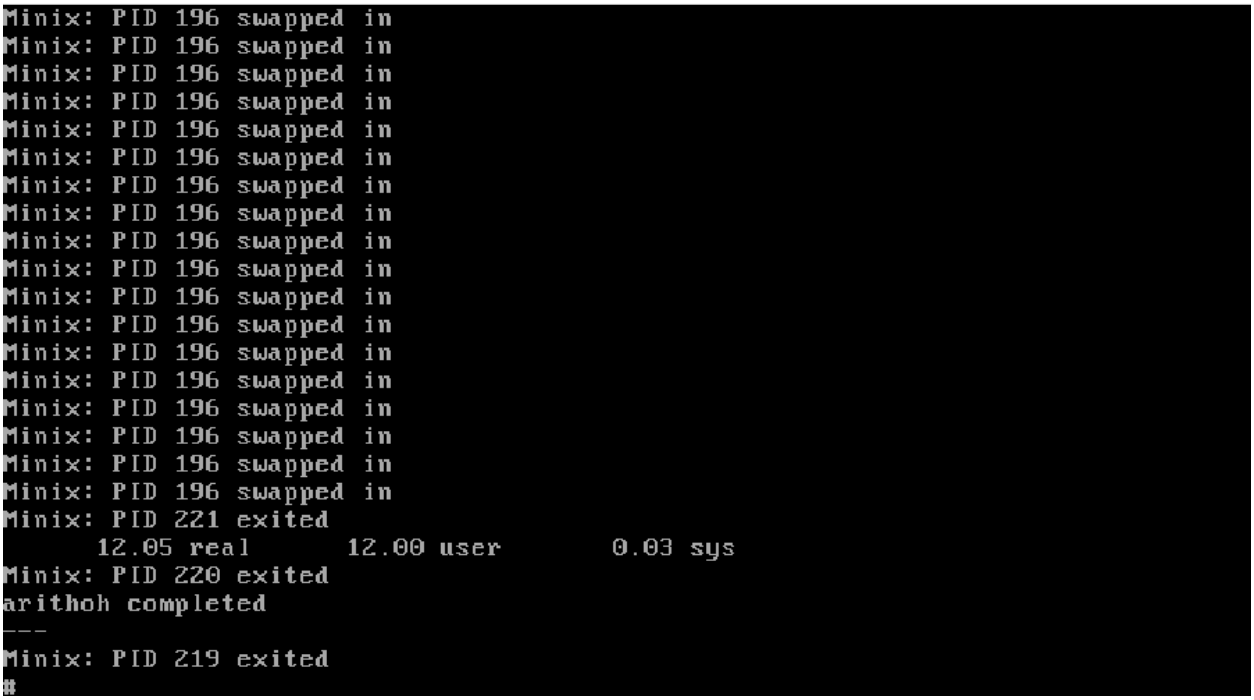
```
if(rmp->priority >= 7)
printf("Minix: PID %d swapped in\n",
_ENDPOINT_P(rmp>endpoint));
```

Part II:

1. **arithoh.sh** (CPU Bound)

   Observations:

a. **Running alone:**
   - When we run arithoh.sh alone, the real and user times taken are almost same. So, the kernel scheduler log indicates that the instruction to schedule Arithoh was sent 92 times consecutively.



b. Running parallely:

- We can observe that, it is following a round-robin kindof scheduling with kernel scheduler and our given print statements.
- Moreover, while both the processes start at the same time, the 2nd one takes around 9 seconds more time than the first one to begin its execution.

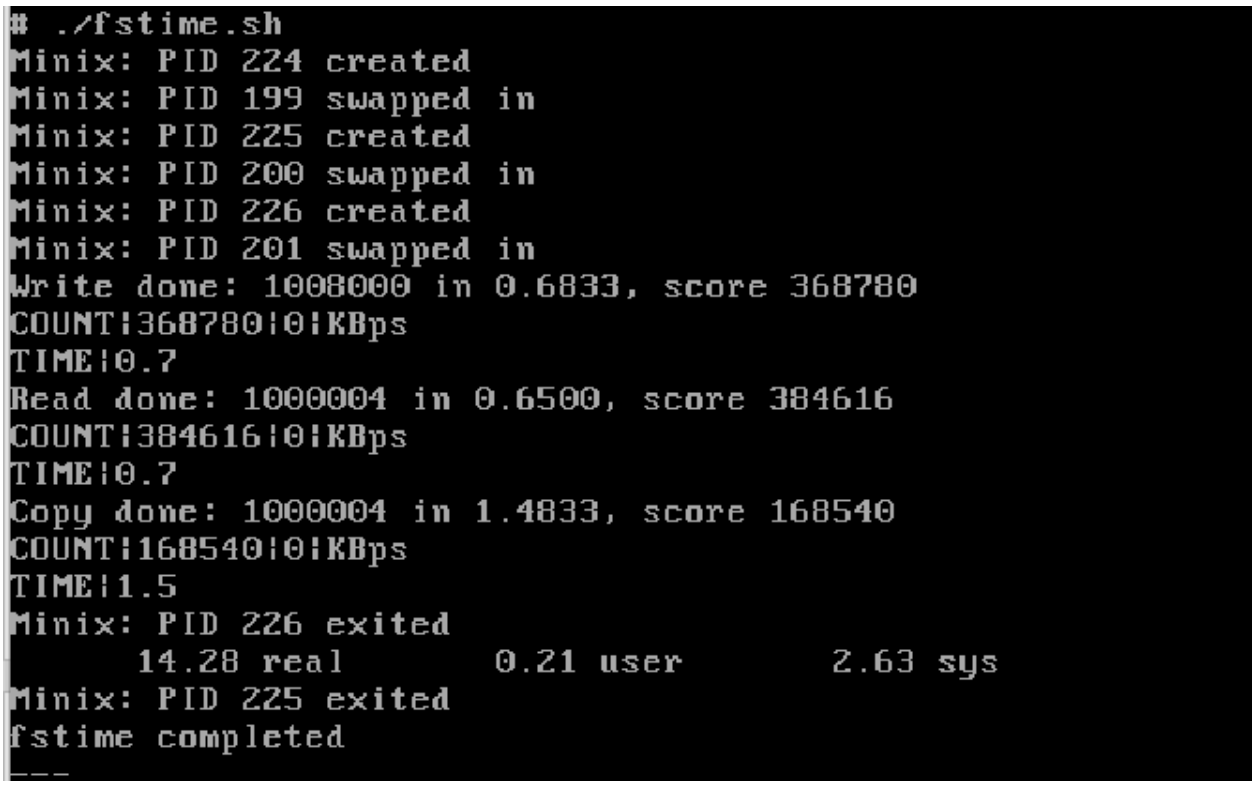
2. <u>fstime.sh</u> <u>(IO Bound)</u>

Observations:

a. Running alone:

- Since it looks like an IO Bound Benchmark process, the user time is less than sys time (moderate) which is less then the total turnaround time (which is largest).

- This occurs because, the process needs to wait for the completion of IO before executing.



b. Running ./arithoh.sh and fstime.sh together:

- Arithoh.sh keeps repeating for some time when IO is scheduled.
- Clearly, arithoh.sh is trying to utilize the CPU when, fstime is waiting for IO. This increases the efficiency of scheduler.

3. **pipe.sh** (CPU Bound)

Observations:

a. Running Alone:

- Lot of time spent by pipe is on sys mode, because the communications between different processes (through pipe) are based on the system itself.

- Time spent in user mode is less, because there's not much user IO in it.

```
# ./pipe.sh
Minix: PID 230 created
Minix: PID 205 swapped in
Minix: PID 231 created
Minix: PID 206 swapped in
Minix: PID 232 created
Minix: PID 207 swapped in
Minix: PID 207 swapped in
Minix: PID 232 exited
       5.55 real       0.40 user       5.13 sys
Minix: PID 231 exited
pipe completed
---
```

b. Running with arithoh.sh together:

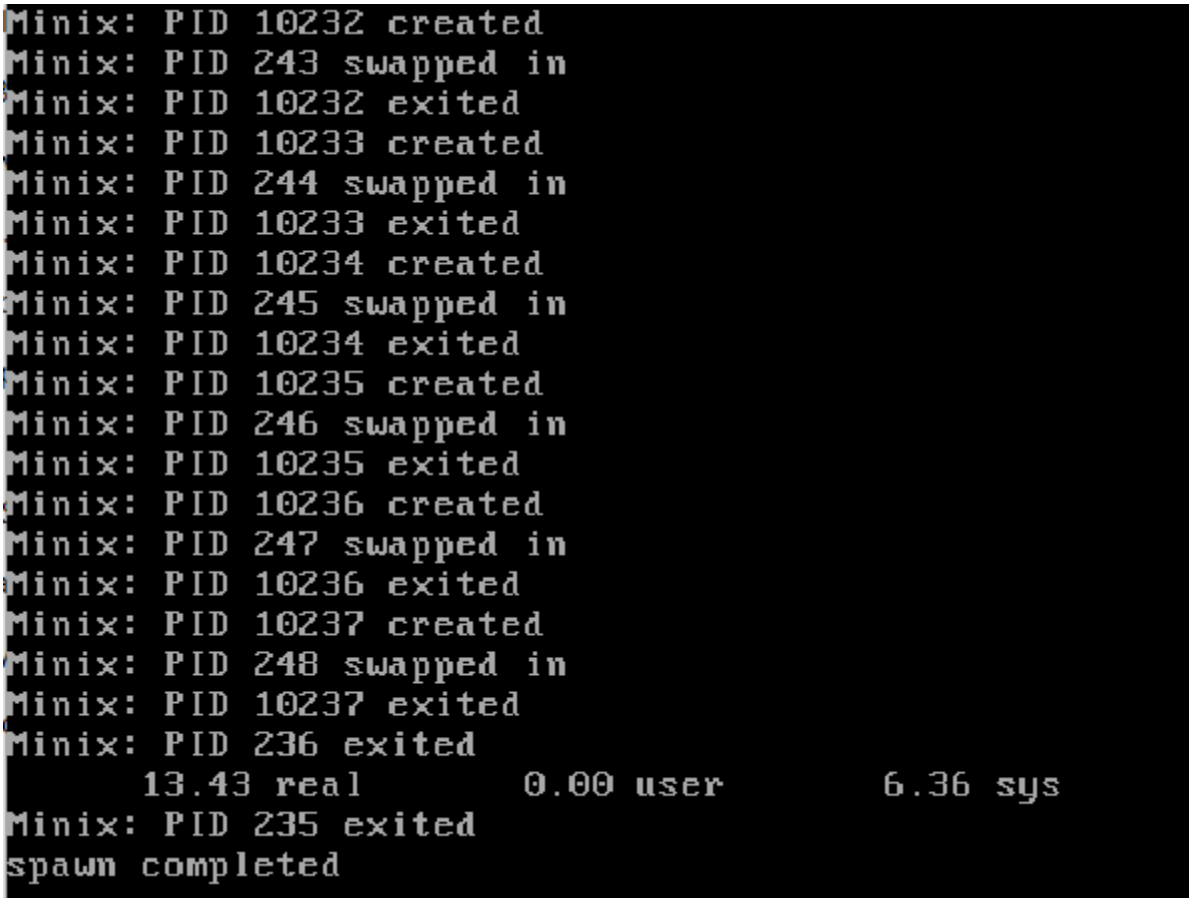
- They run similar to [fstime – arithoh] together, but the only difference is the pipe finishes earlier which is followed by consecutive scheduling of arithoh until it completes.

4. **spawn.sh** (CPU bound)

Observations

a. Running Alone:

- Time spent in sys mode is large (because CPU does most of the work), but total turnaround time (real time) is largest.
- Large number of swaps happening between different processes (in the queue).



b. Running with arithoh together:

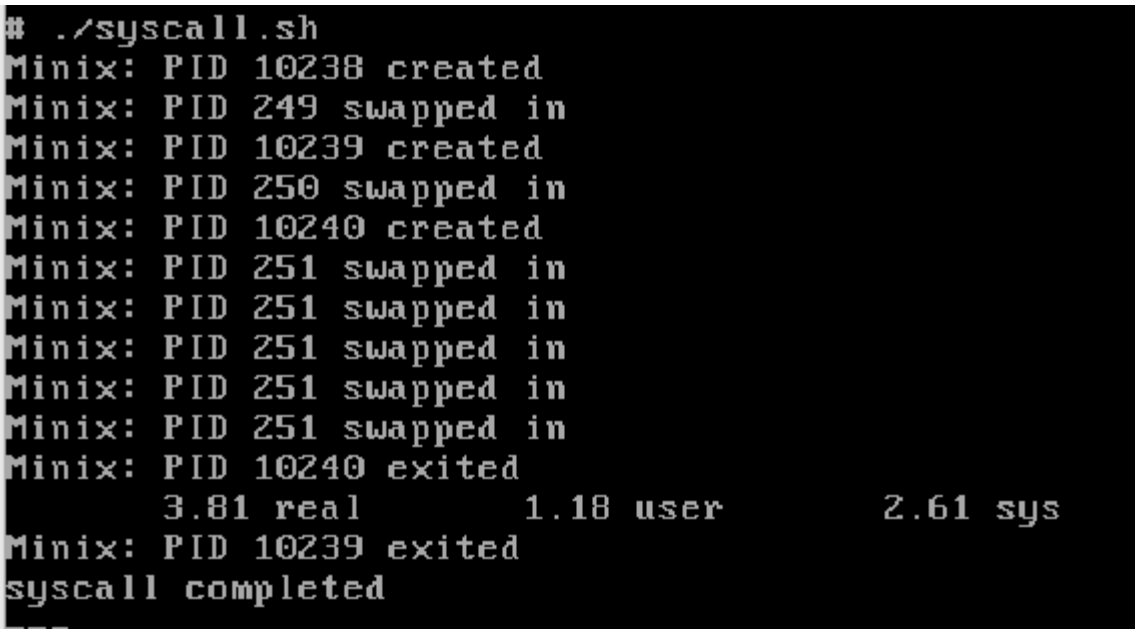
- Similar to [pipe-arithoh], spawn completes early and arithoh continues its execution till it completes.

5. **<u>syscall.sh</u>** <u>(CPU bound)</u>

Observations:

a. Running alone:

- Again, major time spent is on sys mode whereas time spent by user is smaller, but not too small.



Running with arithoh together:

- Both follow a round-robin fashion. syscall completes its execution first and then arithoh is completed.