Part I:

CS314 Operating Systems Lab

**Cheedrala Jaswanth-200010008, Altmash Sheikh-200010002**

**Lab Assignment 3**

The code in minix/servers/sched/schedule.c has been modified, so that “<Roll No> PID

<pid> swapped in” is printed whenever a user level process is brought in by the scheduler.

The code given below has been added in the schedule\_process function just above the return statement.

if(rmp->priority >= USER\_Q) {

printf("200010008 PID %d swapped in\n",\_ENDPOINT\_P(rmp->endpoint));

}

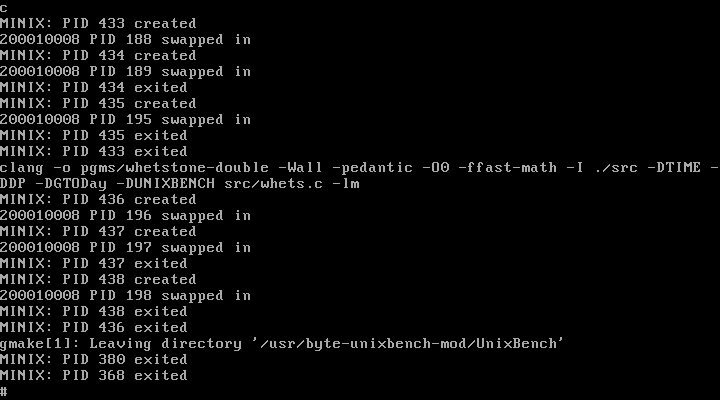
We have written a runme.sh file in order to copy the schedule.c file to its location and to build the changes.

Below is the image of a successful build of the code for this part.



Part II:

UnixBench Benchmark has been downloaded and added into the home folder of MINIX

3. In the Unixbench folder, Makefile has been run using the command gmake. And here is the successful run of the command.

After this, we have gone into the UnixBench/workload\_mix folder in which there are some benchmark programs in .sh files: arithoh.sh, fstime.sh, pipe.sh, spawn.sh, syscall.sh.

We are given a mixture of these files in workload\_mix.sh. It is as follows: #!/bin/sh

./arithoh.sh &

./fstime.sh &

./pipe.sh &

./spawn.sh &

./syscall.sh &

./arithoh.sh &

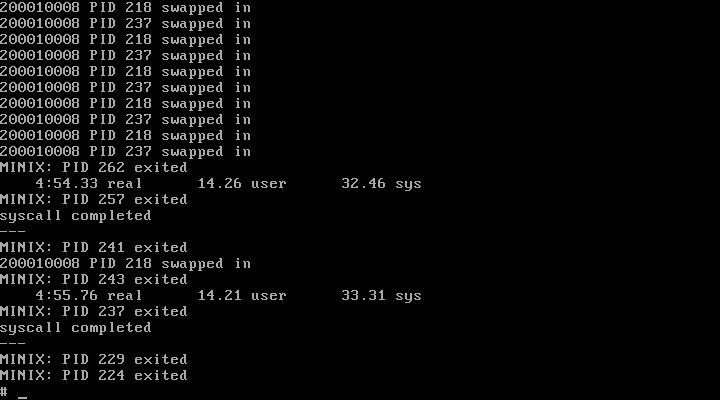
./fstime.sh &

./pipe.sh &

./spawn.sh &

./syscall.sh & wait

This file has been run and successfully like this.



I have written 4 .sh files using the above in different mixtures to observe the behavior of the scheduler by looking at the PID created, excited, swapped in statements printed.

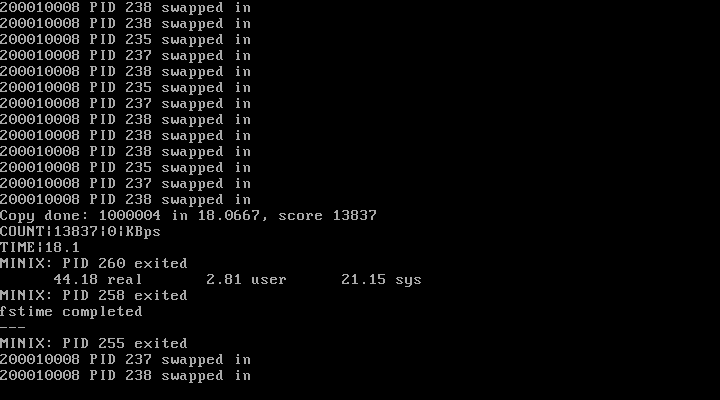
1. mymix1.sh: #!/bin/bash

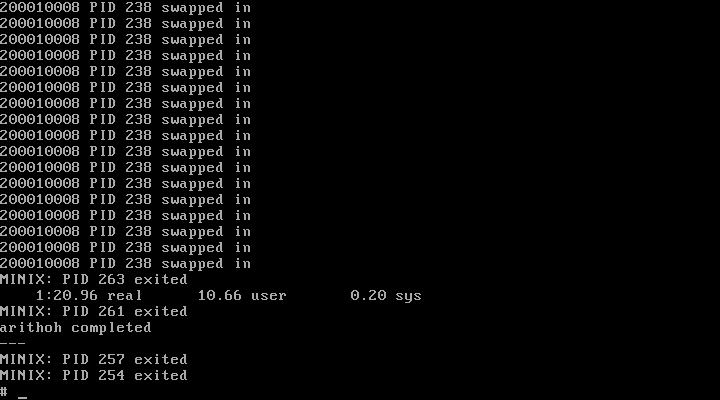
./fstime.sh &

./syscall.sh &

./arithoh.sh & wait



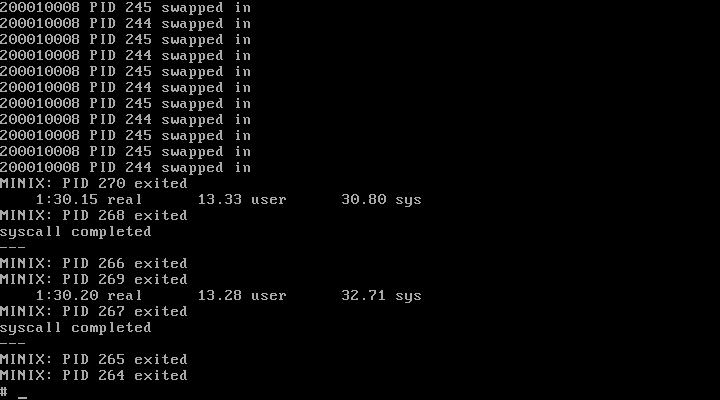




In this execution, fstime completed first, after that syscall completed, and finally arithoh got completed. This is because, fstime is I/O bound, and it is given high priority in the queue, whereas arithoh is highly CPU bound with lot of computations within, so it takes a lot of time slices to get completed, syscall contains system calls and it executes in between.

1. mymix2.sh:

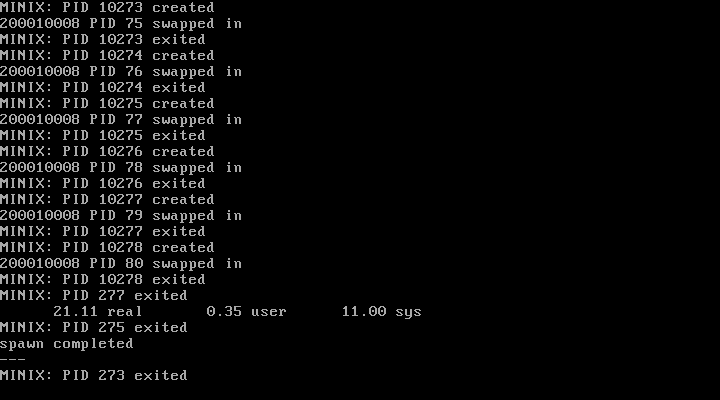
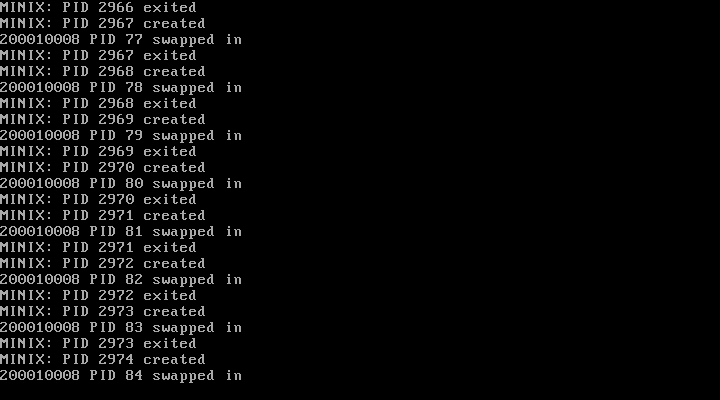
./syscall.sh &

./syscall.sh & wait

In this, the two syscall, will get executed switching in between every time, and then get completed consecutively.

1. mymix3.sh:

./pipe.sh &

./spawn.sh & wait

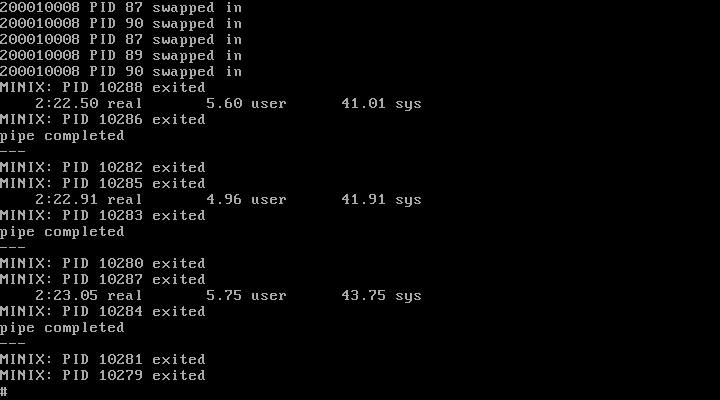
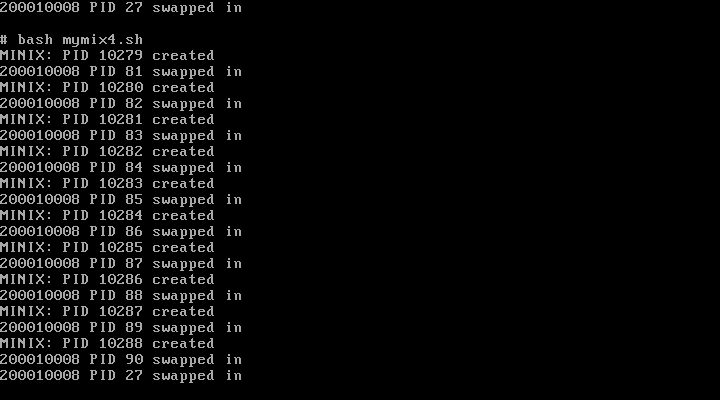
We can see that pipe get completed first before spawn, and enormous no. of processes are created because of the loop execution of fork(). But the created child processes are exited without taking much time, which means there is not much work to the child processes.

1. mymix4.sh:

#!/bin/bash

./pipe.sh &

./pipe.sh &

./pipe.sh & wait

We can see that the 3 pipe sh files get executed and completed consecutively, alternating in between equally. But the time taken before each swapping is more compared to other processes, that means it have more time slice.