CS314

Lab 3

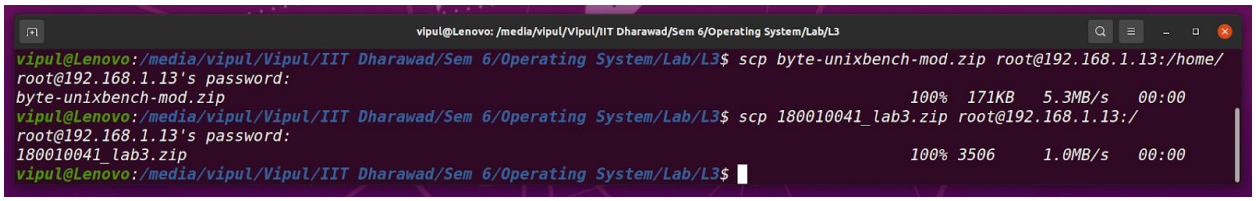
Vipul Yuvraj Nikam
180010041

Q1. Lab 3 Report?

Answer :

Part 0:

Transfering Files from Ubuntu to Minix.



Part I:

To print when the user-level process is brought in by the scheduler, code modified in the file:

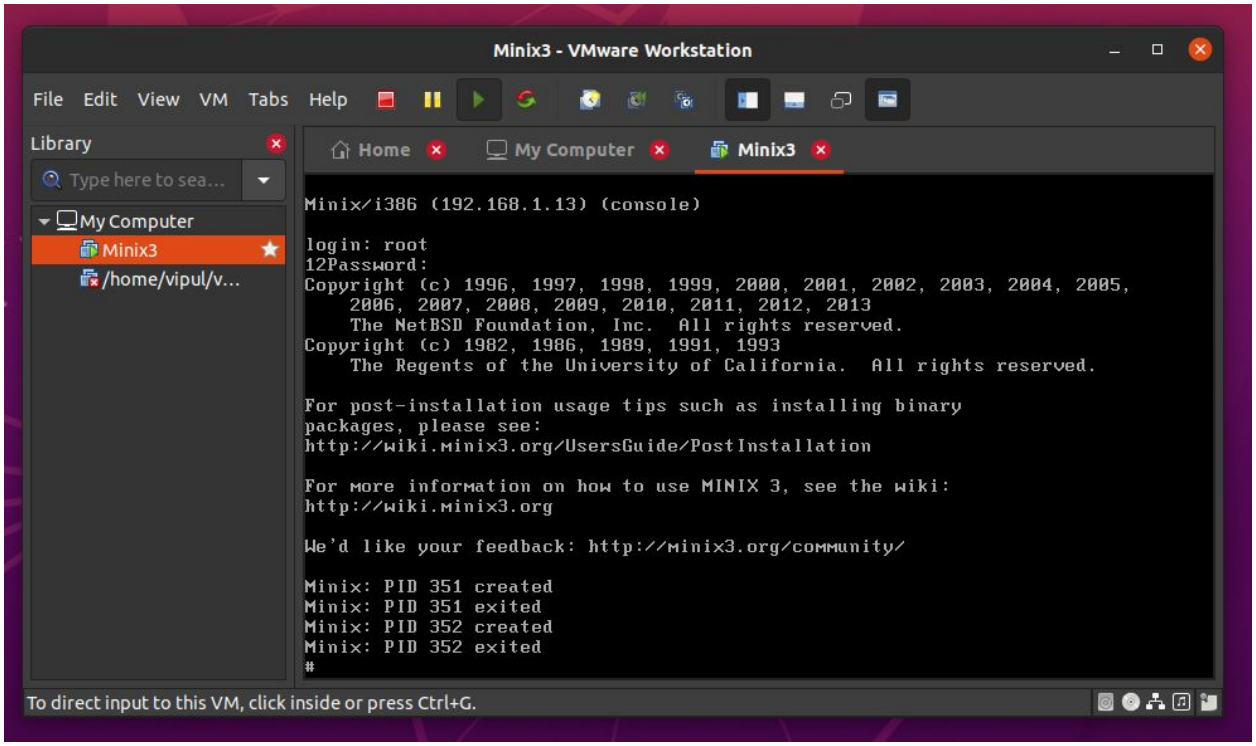
minix/servers/sched/schedule.c

modified function

schedule_process ()

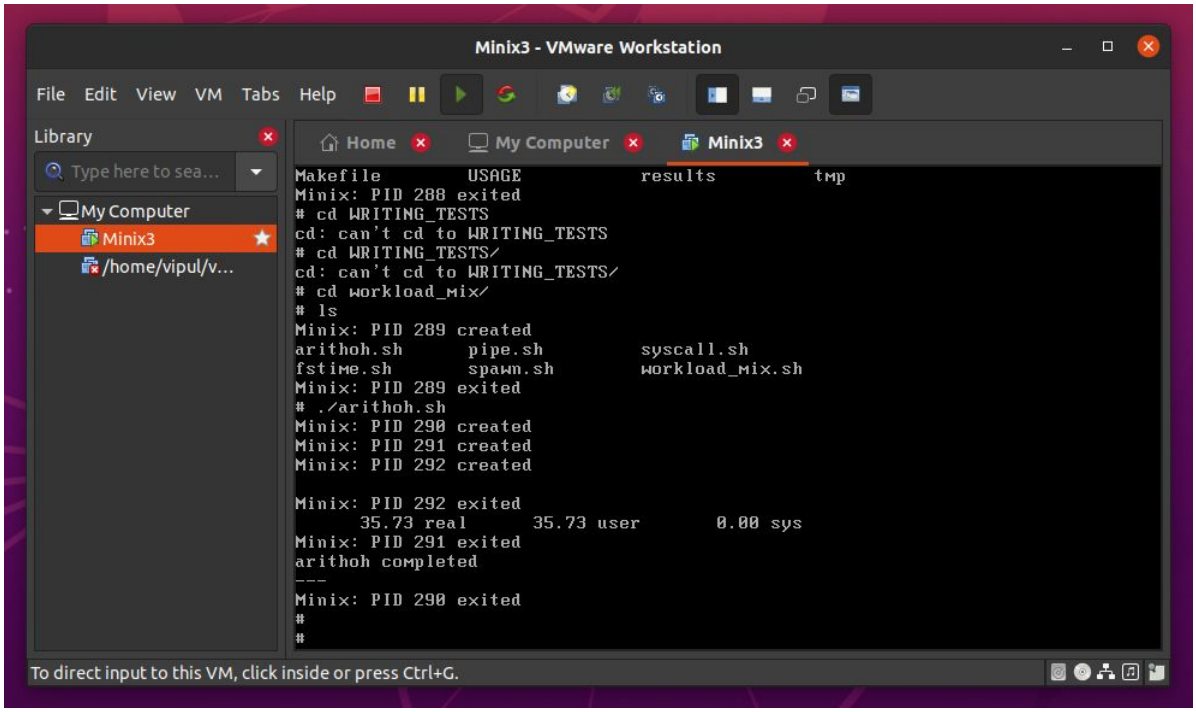
```
297
298  /*===========================================================================*
299   *              schedule_process                                             *
300   *===========================================================================*/
301  static int schedule_process(struct schedproc * rmp, unsigned flags)
302  {
303      int err;
304      int new_prio, new_quantum, new_cpu;
305      int endP = _ENDPOINT_P(rmp->endpoint);
306
307      pick_cpu(rmp);
308
309      if(rmp->priority >= 7)
310      {
311          printf("Minix 3: <pid> %d swapped in\n", endP);
312      }
```

schedule.c



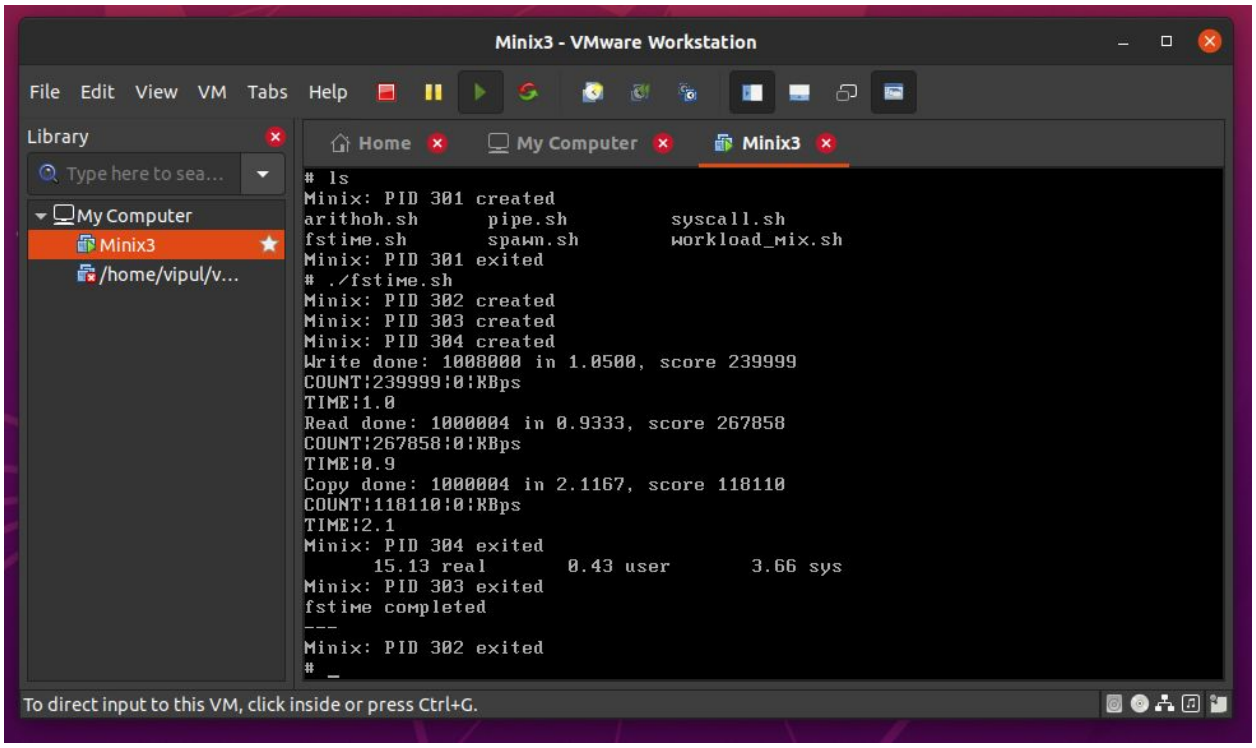
Part II:

- **Arithoh.sh:**



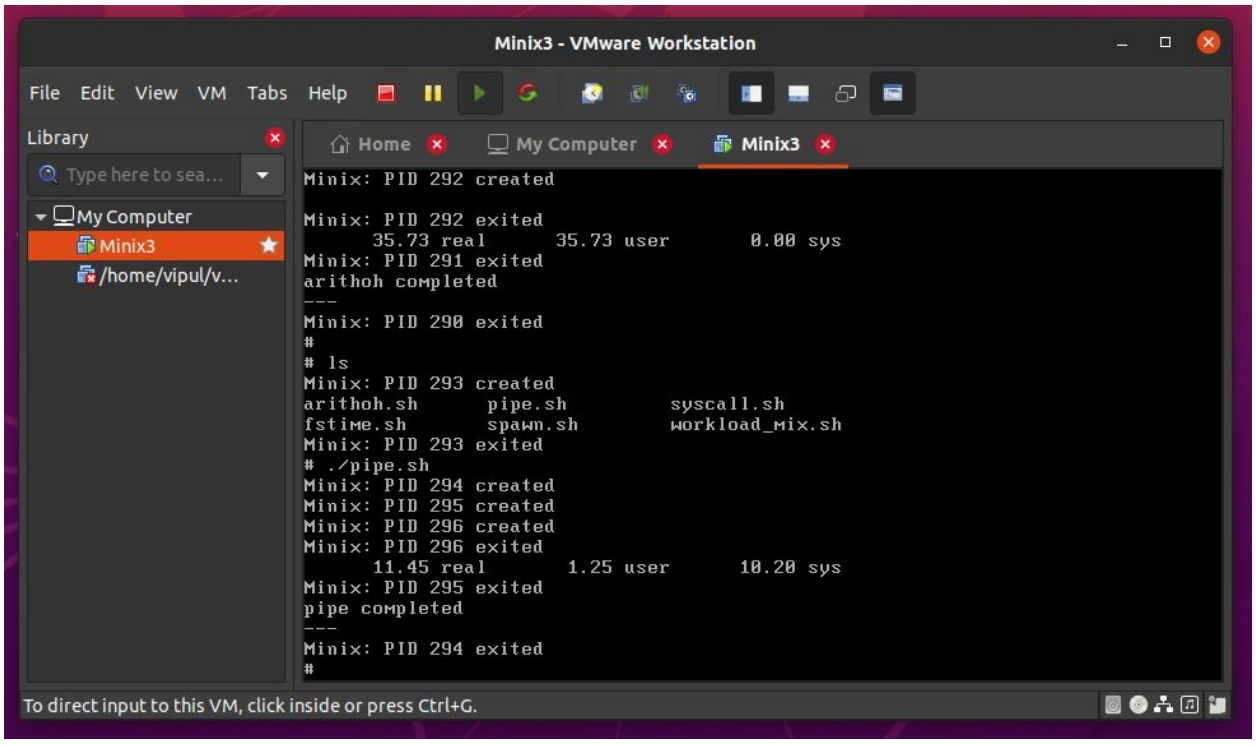
- CPU Bound Benchmark.
- It is observed that while running ‘./arithoh.sh’ command, the real & the user took the same time of ***35.73*** whereas the sys time taken is ***0*** as we can clearly see it in the Screenshot. The kernel scheduler log conforms with this and shows the message to schedule ./arithoh was sent 92 times consecutively.
- We can observe after running two executables of ./arithoh parallelly:
  - Almost turn-based programming in both the kernel scheduler and our print statements. However, the <pid> of one is not always followed by the other, and is sometimes changed again.
  - We also see that although both processes start at the same time, the second takes about 8 seconds longer after the first, which shows that it was given lower priority during scheduling.

- **Fstime.sh:**

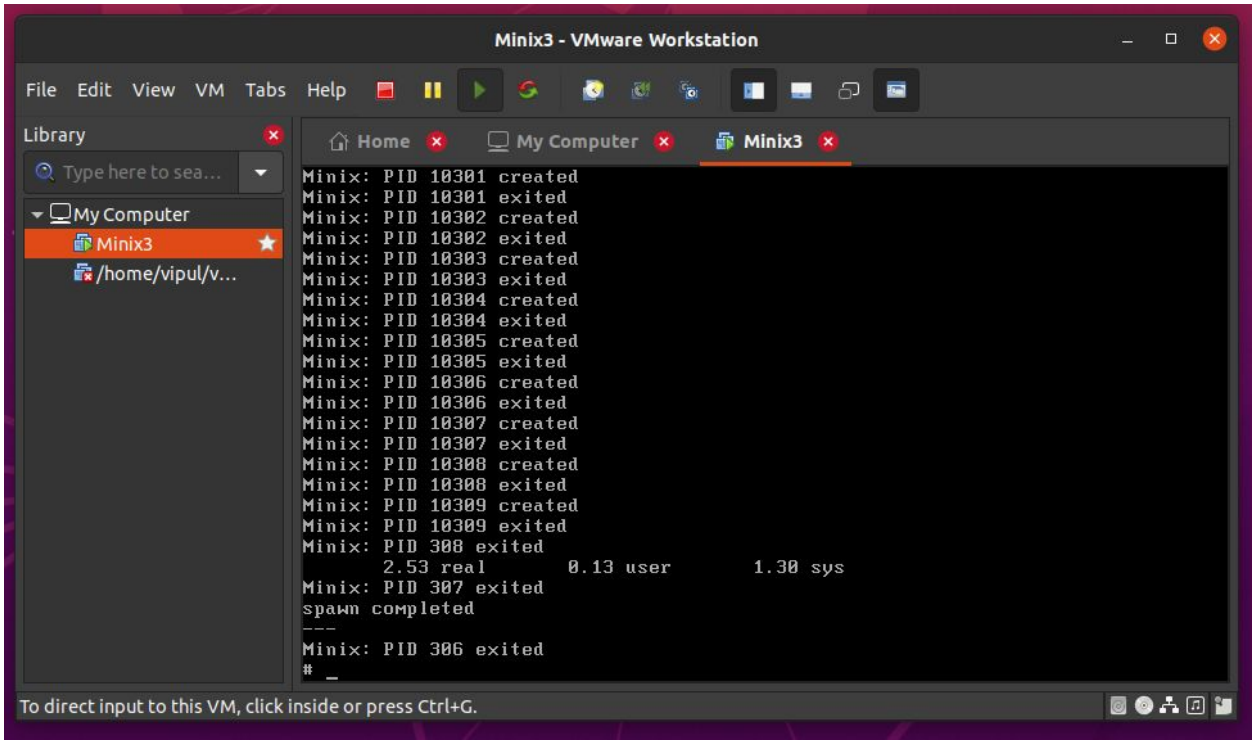


- IO Bound Benchmark.
- We can observe on running fstime.sh that *'total turnaround time'* is greater than *'sys time'* is greater than *'user time'* which occurs because the process needs to wait for its IO to complete before continuing.
- And if ./arithoh and ./fstime are executed at the same time we can see that Arithoh repeats for some time before IO is scheduled. Which demonstrates the scheduler's efficiency in utilizing the wait time of ./fstime to schedule a CPU task like ./arithoh.

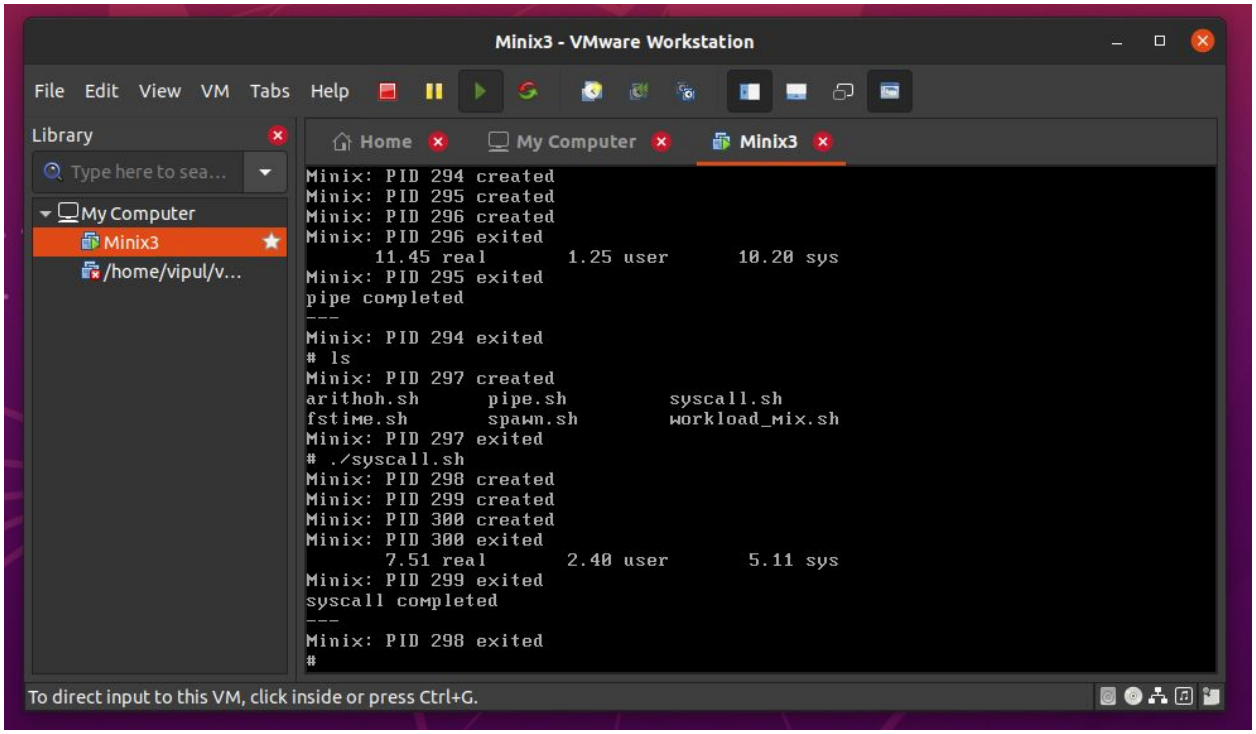
- **Pipe.sh:**
  - CPU Bound Benchmark. Because of Inter-Process Communication protocols, it takes a longer amount of time in sys compared to usr.
  - If executed together with ./arithoh.sh, works similar to previous case of ./fstime.sh, where the pipe finishes earlier followed by consecutive scheduling of arithoh until Completion



- **Spawn.sh:**
    - CPU Bound Benchmark.
    - Like in pipe, here too it takes a longer in sys compared to usr. We can see that a large number of processes ranging from 12 to 320 are exchanged in the queue consecutively.
    - When executed together with arithoh.sh run, spawn ends earlier whereas arithoh continues to run until complete.



- **Syscall.sh:**
    - CPU Bound Benchmark.
    - *'Real time'* is greater than *'sys time'* is greater than *'usr time'.*
    - When executed together with ./arithoh.sh, runs in a round-robin manner with syscall completing first followed by arithoh.


Minix3 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to sea...
My Computer
Minix3
/home/vipul/v...
Home
My Computer
Minix3
Minix: PID 294 created
Minix: PID 295 created
Minix: PID 296 created
Minix: PID 296 exited
11.45 real 1.25 user 10.20 sys
Minix: PID 295 exited
pipe completed

Minix: PID 294 exited
ls
Minix: PID 297 created
arithoh.sh pipe.sh syscall.sh
fstime.sh spawn.sh workload_mix.sh
Minix: PID 297 exited
./syscall.sh
Minix: PID 298 created
Minix: PID 299 created
Minix: PID 300 created
Minix: PID 300 exited
7.51 real 2.40 user 5.11 sys
Minix: PID 299 exited
syscall completed

Minix: PID 298 exited
#
To direct input to this VM, click inside or press Ctrl+G.

XXX