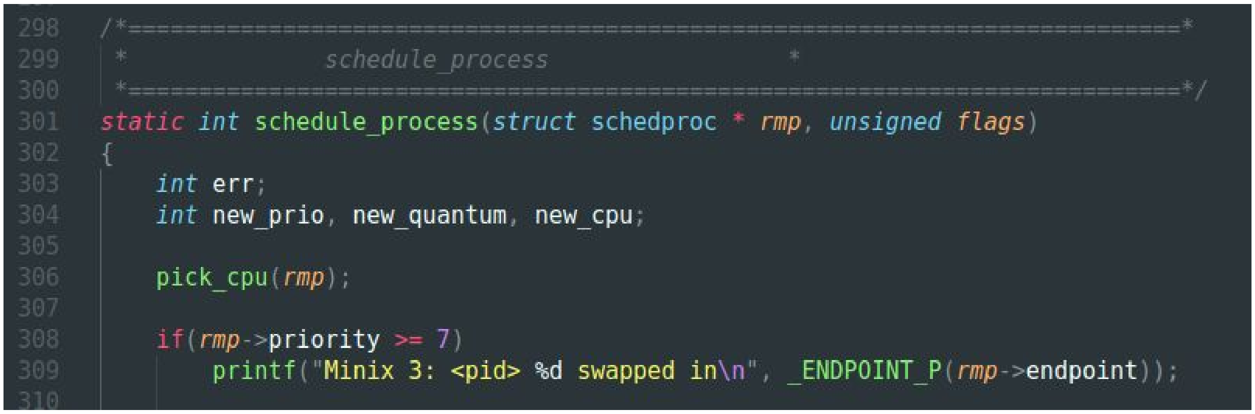
CS 314: Operating Systems Lab

Lab 3 Report

Part I

To print whenever a user-level process is brought in by the scheduler, the code modified in the file: *minix/servers/sched/schedule.c* in the *function schedule_process()*

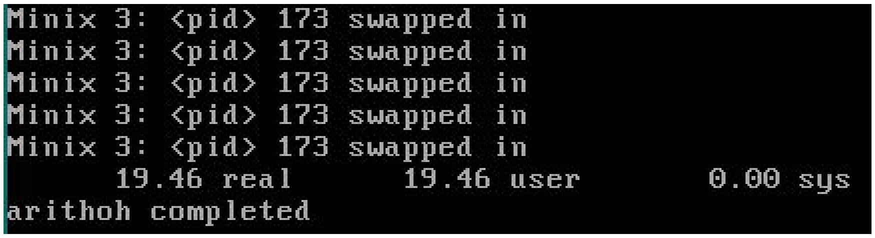


```c
/*===========================================================================*
 *                  schedule_process                          *
 *===========================================================================*/
static int schedule_process(struct schedproc * rmp, unsigned flags)
{
    int err;
    int new_prio, new_quantum, new_cpu;

    pick_cpu(rmp);

    if(rmp->priority >= 7)
        printf("Minix 3: <pid> %d swapped in\n", _ENDPOINT_P(rmp->endpoint));
```

Part 2: Analysis of UnixBench Benchmarks

- ➔ Arithoh.sh
  This is a CPU Bound Benchmark

  - ◆ It is observed that while running 'arithoh' alone, the time taken by real and user is the same. The kernel scheduler log conforms with this and shows that the message to schedule ./arithoh was sent 92 times consecutively.



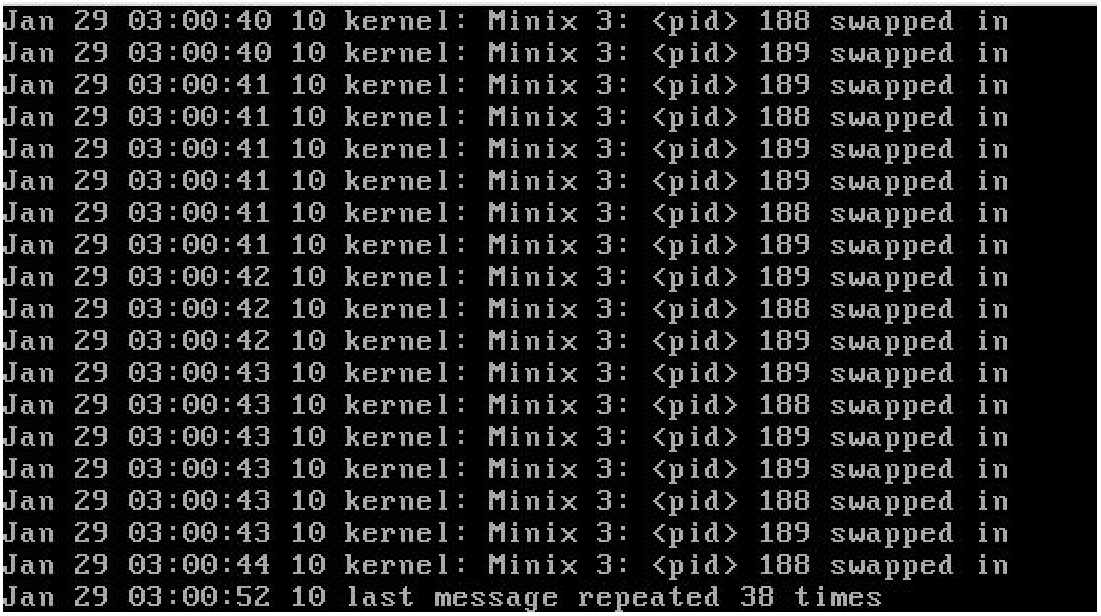
```
Minix 3: <pid> 173 swapped in
Minix 3: <pid> 173 swapped in
Minix 3: <pid> 173 swapped in
Minix 3: <pid> 173 swapped in
Minix 3: <pid> 173 swapped in
      19.46 real       19.46 user        0.00 sys
arithoh completed
```

  - ◆ The sys time taken by this is **0.**

  - ◆ While running two executables of ./arithoh parallelly, we observe the following -

        - A “nearly” round-robin scheduling in action in both the kernel scheduler and our print statements. However, it is important to see that the <pid> of one isn’t always followed by the other, and can instead be swapped in again.

        - We also see that although both processes start at the same time, the second one takes **8s more** after the first one, showing that it was given *lesser priority* during scheduling.



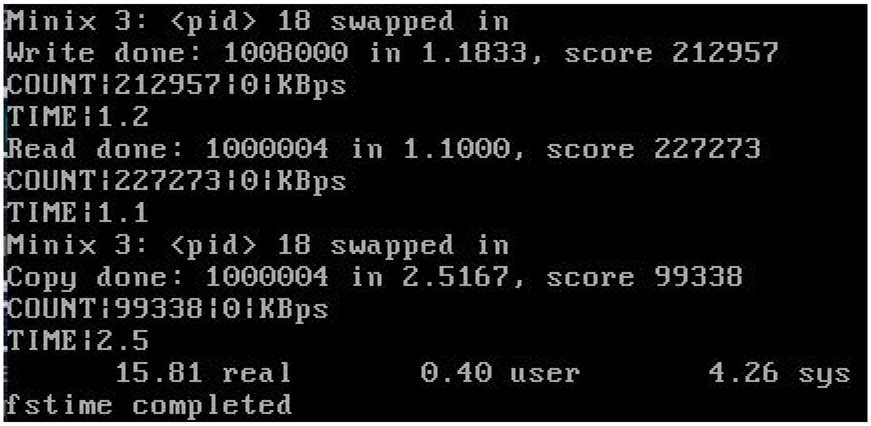
```
Jan 29 03:00:40 10 kernel: Minix 3: <pid> 188 swapped in
Jan 29 03:00:40 10 kernel: Minix 3: <pid> 189 swapped in
Jan 29 03:00:41 10 kernel: Minix 3: <pid> 189 swapped in
Jan 29 03:00:41 10 kernel: Minix 3: <pid> 188 swapped in
Jan 29 03:00:41 10 kernel: Minix 3: <pid> 189 swapped in
Jan 29 03:00:41 10 kernel: Minix 3: <pid> 189 swapped in
Jan 29 03:00:41 10 kernel: Minix 3: <pid> 188 swapped in
Jan 29 03:00:41 10 kernel: Minix 3: <pid> 189 swapped in
Jan 29 03:00:42 10 kernel: Minix 3: <pid> 189 swapped in
Jan 29 03:00:42 10 kernel: Minix 3: <pid> 188 swapped in
Jan 29 03:00:42 10 kernel: Minix 3: <pid> 189 swapped in
Jan 29 03:00:43 10 kernel: Minix 3: <pid> 189 swapped in
Jan 29 03:00:43 10 kernel: Minix 3: <pid> 188 swapped in
Jan 29 03:00:43 10 kernel: Minix 3: <pid> 189 swapped in
Jan 29 03:00:43 10 kernel: Minix 3: <pid> 189 swapped in
Jan 29 03:00:43 10 kernel: Minix 3: <pid> 188 swapped in
Jan 29 03:00:43 10 kernel: Minix 3: <pid> 189 swapped in
Jan 29 03:00:44 10 kernel: Minix 3: <pid> 188 swapped in
Jan 29 03:00:52 10 last message repeated 38 times
```

- ➔ Fstime.sh

  This is an IO Bound Benchmark. On executing fstime.sh we observe the following -

    - ◆ As it is an IO intensive process the user time is less, and, sys time is moderate compared to the total turnaround time which is large.

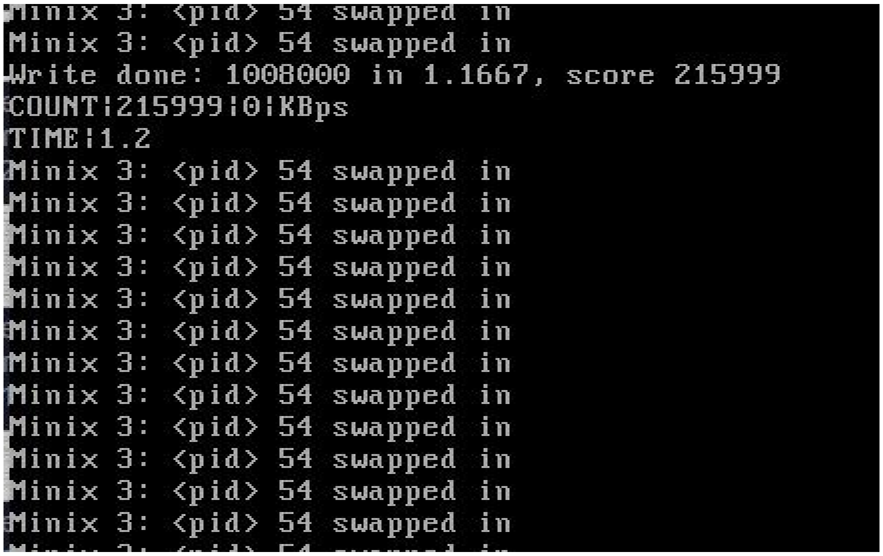
    - ◆ This is because the process needs to wait for its IO to complete before continuing.



```
Minix 3: <pid> 18 swapped in
Write done: 1008000 in 1.1833, score 212957
COUNT¦212957¦0¦KBps
TIME¦1.2
Read done: 1000004 in 1.1000, score 227273
COUNT¦227273¦0¦KBps
TIME¦1.1
Minix 3: <pid> 18 swapped in
Copy done: 1000004 in 2.5167, score 99338
COUNT¦99338¦0¦KBps
TIME¦2.5
      15.81 real        0.40 user        4.26 sys
fstime completed
```

    - ◆ When running ./arithoh and ./fstime, we see an interesting observation -
        - Arithoh always repeats for some time before IO is scheduled.

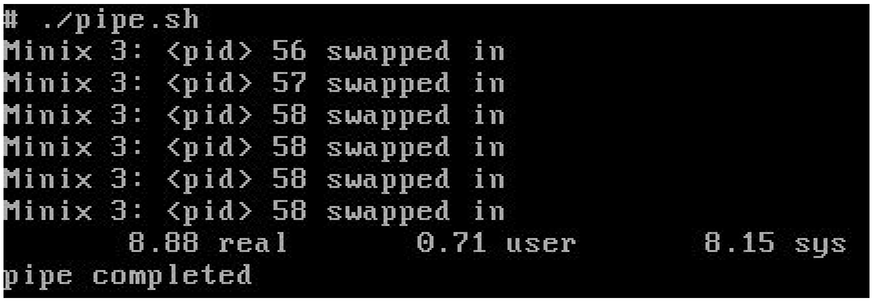
        - This demonstrates the scheduler’s efficiency in utilizing the wait time of ./fstime to schedule a CPU task like ./arithoh.



```
Minix 3: <pid> 54 swapped in
Minix 3: <pid> 54 swapped in
Write done: 1008000 in 1.1667, score 215999
COUNT|215999|0|KBps
TIME|1.2
Minix 3: <pid> 54 swapped in
Minix 3: <pid> 54 swapped in
Minix 3: <pid> 54 swapped in
Minix 3: <pid> 54 swapped in
Minix 3: <pid> 54 swapped in
Minix 3: <pid> 54 swapped in
Minix 3: <pid> 54 swapped in
Minix 3: <pid> 54 swapped in
Minix 3: <pid> 54 swapped in
Minix 3: <pid> 54 swapped in
Minix 3: <pid> 54 swapped in
Minix 3: <pid> 54 swapped in
Minix 3: <pid> 54 swapped in
```

- ➔ Pipe.sh
  This is a CPU Bound Benchmark based on IPC.

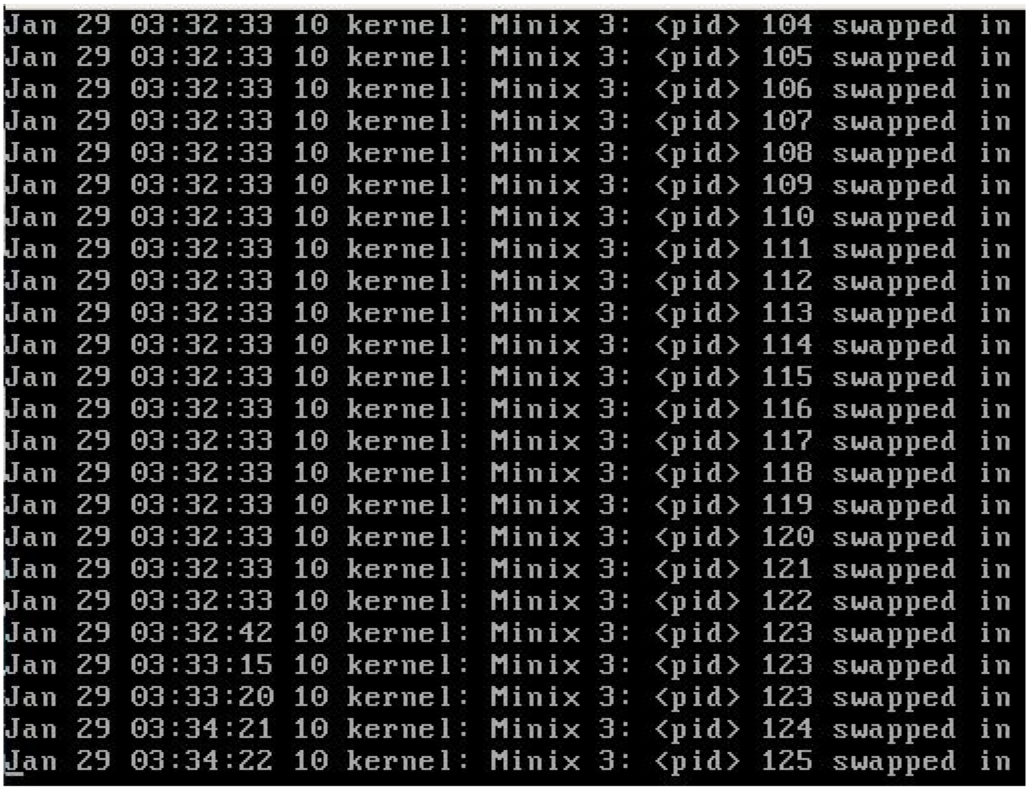
    - ◆ A huge amount of time spent by Pipe is in sys mode. This is because the Inter-Process Communication protocols are highly based on the system. The user-mode time spent is very less.



```
# ./pipe.sh
Minix 3: <pid> 56 swapped in
Minix 3: <pid> 57 swapped in
Minix 3: <pid> 58 swapped in
Minix 3: <pid> 58 swapped in
Minix 3: <pid> 58 swapped in
Minix 3: <pid> 58 swapped in
        8.88 real        0.71 user        8.15 sys
pipe completed
```

    - ◆ When ./pipe.sh and ./arithoh.sh works similar to ./fstime.sh and ./arithoh.sh, where the pipe finishes earlier followed by consecutive scheduling of arithoh until completion.

- ➔ Spawn.sh
  This is a CPU Bound Benchmark
    - ◆ We see that a huge number of processes ranging from 7 to 255 are swapped into the queue consecutively.

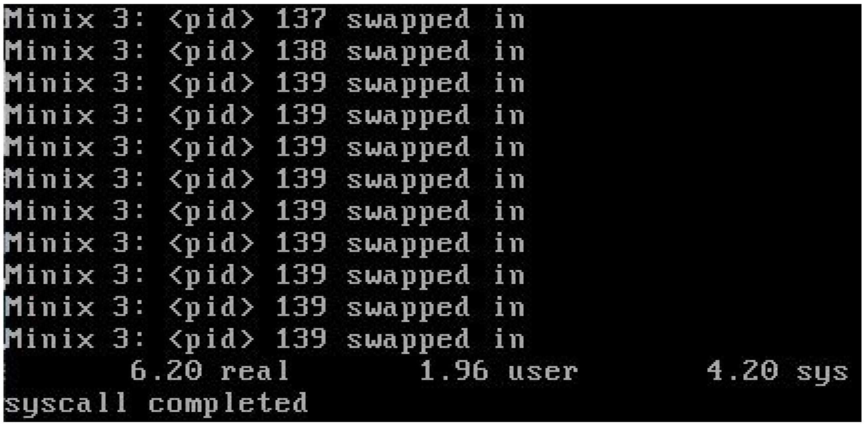


```
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 104 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 105 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 106 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 107 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 108 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 109 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 110 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 111 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 112 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 113 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 114 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 115 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 116 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 117 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 118 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 119 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 120 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 121 swapped in
Jan 29 03:32:33 10 kernel: Minix 3: <pid> 122 swapped in
Jan 29 03:32:42 10 kernel: Minix 3: <pid> 123 swapped in
Jan 29 03:33:15 10 kernel: Minix 3: <pid> 123 swapped in
Jan 29 03:33:20 10 kernel: Minix 3: <pid> 123 swapped in
Jan 29 03:34:21 10 kernel: Minix 3: <pid> 124 swapped in
Jan 29 03:34:22 10 kernel: Minix 3: <pid> 125 swapped in
```

- ◆ In this too, the time spent in sys mode is large. When Spawn and arithoh are executed, spawn finishes earlier and arithoh continues executing until completion.

➔ Syscall.sh
This is a CPU Bound Benchmark

- ◆ The time spent while executing this process is majorly in sys-mode. User mode time spent is also considerable but small.



```
Minix 3: <pid> 137 swapped in
Minix 3: <pid> 138 swapped in
Minix 3: <pid> 139 swapped in
Minix 3: <pid> 139 swapped in
Minix 3: <pid> 139 swapped in
Minix 3: <pid> 139 swapped in
Minix 3: <pid> 139 swapped in
Minix 3: <pid> 139 swapped in
Minix 3: <pid> 139 swapped in
Minix 3: <pid> 139 swapped in
Minix 3: <pid> 139 swapped in
       6.20 real       1.96 user       4.20 sys
syscall completed
```

- ◆ Syscall and arithoh run in a round-robin manner with syscall completing first followed by arithoh.