
CS 314 – Operating Systems Laboratory

Assignment-5 Report

Student 1: Manche Pavanitha
Roll-No: 200010027

Student 2: Sahaja Nandyala
Roll-No: 200010032

1 Part 1

To display the time quanta spent by the CPU in the processes, changes have been done in the function `sched_proc()` of the file `system.c` which is located in the directory:

`minix/kernel`

The following changes are done in the file:

```
printf("quantum allotted : %d, quantum used : %d\n", p->p_quantum_size_ms,  
p->p_quantum_size_ms-cpu_time_2_ms(p->p_cpu_time_left));
```

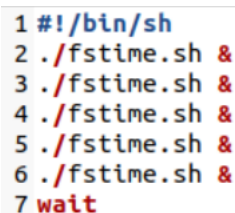


```
# ls  
Minix: PID 600 created  
quantum allotted : 200, quantum used: 200  
200010027:PID 135 swapped in  
bin          boot_monitor home      proc          service       usr  
boot         dev          lib          root          sys          var  
boot.cfg     etc          mnt          sbin          tmp  
Minix: PID 600 exited  
# S<
```

In this Part, we prepared 4 workload mixes having different characteristics, ranging from all compute-intensive benchmarks to all I/O-intensive benchmarks. Each workload is composed of benchmarks from the Unix Bench suite, and are as follows

1.1 workload_mix1.sh

`workload_mix1.sh` contains 5 `fstime.sh` files.



```
1 #!/bin/sh  
2 ./fstime.sh &  
3 ./fstime.sh &  
4 ./fstime.sh &  
5 ./fstime.sh &  
6 ./fstime.sh &  
7 wait
```

Figure 1: `workload_mix1` code

There are 5 `fstime` files that are I/O-bound in this workload mix. I/O processes typically don't take up the entire time slice, thus each process waits for receiving I/O while executing in a round-robin way that is the CPU is evenly distributed among all the processes.

```

quantum allotted : 500, quantum used: 500
quantum allotted : 200, quantum used: 200
200010027:PID 140 swapped in
quantum allotted : 200, quantum used: 200
200010027:PID 139 swapped in
quantum allotted : 200, quantum used: 200
200010027:PID 138 swapped in
quantum allotted : 200, quantum used: 200
200010027:PID 137 swapped in
quantum allotted : 500, quantum used: 500
200010027:PID 23 swapped in
quantum allotted : 500, quantum used: 146
quantum allotted : 200, quantum used: 31
200010027:PID 137 swapped in
quantum allotted : 200, quantum used: 31
200010027:PID 138 swapped in
quantum allotted : 200, quantum used: 31
200010027:PID 139 swapped in
quantum allotted : 200, quantum used: 31
200010027:PID 140 swapped in
quantum allotted : 200, quantum used: 34
200010027:PID 141 swapped in
quantum allotted : 500, quantum used: 500
200010027:PID 23 swapped in

```

```

35.50 real 0.55 user 4.25 sys
Minix: PID 384 exited
fstime completed
---
Minix: PID 379 exited
Copy done: 1000004 in 12.2000, score 20491
COUNT:20491:0:KBps
TIME:12.2
Minix: PID 390 exited
35.51 real 0.36 user 4.21 sys
Minix: PID 385 exited
fstime completed
---
Minix: PID 380 exited
Copy done: 1000004 in 12.5333, score 19946
COUNT:19946:0:KBps
TIME:12.5
Minix: PID 391 exited
35.85 real 0.55 user 3.80 sys
Minix: PID 386 exited
fstime completed
---
Minix: PID 381 exited
Minix: PID 376 exited
#

```

1.2 workload_mix2.sh

workload_mix2.sh contains 3 arithoh.sh and 2 fstime.sh processes and the code for it is as follows:

```

1 #!/bin/sh
2 ./arithoh.sh &
3 ./arithoh.sh &
4 ./arithoh.sh &
5 ./fstime.sh &
6 ./fstime.sh &
7 wait

```

Figure 2: workload_mix2 code

As we know, arithoh.sh are CPU bound processes and hence they are using up the complete time slice allotted and are executing in round robin way whereas fstime.sh are I/O bound in nature, so they don't utilize the entire time slice. The fstime processes are completed first and at the end CPU processes which are arithoh.sh are completed.

```

quantum allotted : 200, quantum used: 0
200010027:PID 181 swapped in
quantum allotted : 200, quantum used: 83
200010027:PID 182 swapped in
quantum allotted : 200, quantum used: 200
200010027:PID 183 swapped in
quantum allotted : 200, quantum used: 200
200010027:PID 182 swapped in
quantum allotted : 200, quantum used: 200
200010027:PID 183 swapped in
quantum allotted : 200, quantum used: 200
200010027:PID 181 swapped in
quantum allotted : 200, quantum used: 200
200010027:PID 183 swapped in
quantum allotted : 200, quantum used: 200
200010027:PID 181 swapped in
quantum allotted : 200, quantum used: 200
200010027:PID 182 swapped in
quantum allotted : 200, quantum used: 200
200010027:PID 181 swapped in
quantum allotted : 200, quantum used: 200
200010027:PID 182 swapped in
quantum allotted : 200, quantum used: 200
200010027:PID 183 swapped in

```

Figure 3: CPU processes using up the entire time slices

```

200010027:PID 204 swapped in
Minix: PID 442 exited
36.31 real      8.28 user      0.00 sys
Minix: PID 437 exited
arithoh completed
---
Minix: PID 432 exited
quantum allotted : 200, quantum used: 200
200010027:PID 203 swapped in
Minix: PID 444 exited
36.45 real      8.20 user      0.06 sys
Minix: PID 439 exited
arithoh completed
---
Minix: PID 434 exited
quantum allotted : 200, quantum used: 200
200010027:PID 203 swapped in
Minix: PID 443 exited
36.80 real      8.01 user      0.01 sys
Minix: PID 438 exited
arithoh completed
---
Minix: PID 433 exited
Minix: PID 431 exited
#

```

Figure 4: arithoh.sh completed

1.3 workload_mix3.sh

The following is the code for workload_mix3.sh which has arithoh.sh and spawn.sh processes alternatively.

```

#!/bin/sh
./arithoh.sh &
./spawn.sh &
./arithoh.sh &
./spawn.sh &
./arithoh.sh &
wait

```

Figure 5: workload_mix3 code

In this workload_mix, arithoh.sh and spawn.sh are called alternatively. They execute in round robin fashion i.e., first arithoh.sh which is CPU bound is executed in allotted time slice of 200 and then spawn.sh is scheduled which contains fork calls resulting in creation of new process which are also added to ready queue and all of these processes are scheduled in round robin manner. So, as arithoh.sh are CPU intensive processes, they are completed at last.

```

Minix: PID 16953 created
quantum allotted : 200, quantum used: 200
200010027:PID 48 swapped in
Minix: PID 16952 exited
Minix: PID 16954 created
quantum allotted : 200, quantum used: 200
200010027:PID 49 swapped in
Minix: PID 16953 exited
Minix: PID 16955 created
quantum allotted : 200, quantum used: 200
200010027:PID 50 swapped in
Minix: PID 16954 exited
Minix: PID 16956 created
quantum allotted : 200, quantum used: 200
200010027:PID 51 swapped in
Minix: PID 16955 exited
Minix: PID 16957 created
quantum allotted : 200, quantum used: 200
200010027:PID 52 swapped in
Minix: PID 16956 exited
Minix: PID 16958 created
quantum allotted : 200, quantum used: 200
200010027:PID 53 swapped in
Minix: PID 16957 exited
Minix: PID 16959 created

```

Figure 6

```

200010027:PID 237 swapped in
Minix: PID 23173 exited
1:01.10 real 7.20 user 0.00 sys
Minix: PID 23168 exited
arithoh completed
---
Minix: PID 23163 exited
Minix: PID 23175 exited
1:01.21 real 7.38 user 0.00 sys
Minix: PID 23170 exited
arithoh completed
---
Minix: PID 23165 exited
quantum allotted : 200, quantum used: 200
200010027:PID 237 swapped in
quantum allotted : 200, quantum used: 200
200010027:PID 237 swapped in
Minix: PID 23177 exited
1:01.75 real 7.25 user 0.00 sys
Minix: PID 23172 exited
arithoh completed
---
Minix: PID 23167 exited
Minix: PID 23162 exited
# -

```

Figure 7

1.4 workload_mix4.sh

The following is the code for workload_mix4.sh which contains 5 syscall.sh.

```

1 #!/bin/sh
2 ./syscall.sh &
3 ./syscall.sh &
4 ./syscall.sh &
5 ./syscall.sh &
6 ./syscall.sh &
7 wait

```

Figure 8: workload_mix4 code

The syscall processes are CPU bound processes but not as intensive processes as arithoh processes. So, the syscall processes do not take up that much of the CPU time as shown in Figure 9. In this workload_mix, the syscall processes are created with process ID's 133, 134, 135, 136 and 137 and all these processes finish almost at the same time executing in round robin way.

```

200010027:PID 136 swapped in
Minix: PID 387 created
quantum alloted : 200, quantum used: 200
200010027:PID 137 swapped in
quantum alloted : 200, quantum used: 200
200010027:PID 133 swapped in
quantum alloted : 200, quantum used: 200
200010027:PID 136 swapped in
quantum alloted : 200, quantum used: 200
200010027:PID 135 swapped in
quantum alloted : 200, quantum used: 200
200010027:PID 137 swapped in
quantum alloted : 200, quantum used: 200
200010027:PID 134 swapped in
quantum alloted : 200, quantum used: 12
200010027:PID 133 swapped in
quantum alloted : 200, quantum used: 12
200010027:PID 134 swapped in
quantum alloted : 200, quantum used: 11
200010027:PID 135 swapped in
quantum alloted : 200, quantum used: 12
200010027:PID 136 swapped in
quantum alloted : 200, quantum used: 12
200010027:PID 137 swapped in

```

Figure 9

```

45.15 real      3.10 user      6.25 sys
Minix: PID 378 exited
syscall completed
---
Minix: PID 373 exited
Minix: PID 384 exited
45.26 real      3.26 user      6.05 sys
Minix: PID 379 exited
syscall completed
---
Minix: PID 374 exited
Minix: PID 385 exited
45.36 real      3.46 user      5.88 sys
Minix: PID 380 exited
syscall completed
---
Minix: PID 375 exited
Minix: PID 386 exited
45.38 real      3.61 user      5.30 sys
Minix: PID 381 exited
syscall completed
---
Minix: PID 376 exited
Minix: PID 372 exited
#

```

Figure 10

2 Part 2

In this part, we had to modify the user-level scheduler in Minix3 to the following “Pseudo-FIFO” policy: among the user-level processes that are ready to execute, the one that entered the earliest must be scheduled.

To make the scheduler run in “Pseudo-FIFO” manner, changes have been done in the file `schedule.c` which is located in the directory:

```
minix/servers/sched
```

In the function `do_noquantum` of this file, the priority order is changed as follows i.e., the one which comes first will be given highest priority.

```

rmp->priority -= 1;
/* Giving higher priority to the process that comes first */

```

Also, the following changes were made in the `balance_queues`.

```
// rmp->priority -= 1; /* increase priority */
```

2.1 workload_mix1.sh

`workload_mix1.sh` contains 5 `fstime.sh` files.

```
1 #!/bin/sh
2 ./fstime.sh &
3 ./fstime.sh &
4 ./fstime.sh &
5 ./fstime.sh &
6 ./fstime.sh &
7 wait
```

Figure 11: `workload_mix1` code

In this `workload_mix` we took 5 `fstime.sh` continuously, according to First In First Out scheduling algorithm the 5 processes should be exited in order of their arrival but from the output we can see that FIFO is not happening properly making this scheduling process Pseudo FIFO. This is because `fstime.sh` is I/O bound process which goes to blocked state waiting for response, in the meanwhile the next process is scheduled and this whole results in round robin scheduling. So, there is no difference in output for original minix scheduler and Pseudo FIFO for this `workload_mix`.

```
46.16 real    46.16 real    46.16 real    46.16 real    0.66 user
0.76 user    0.83 user    0.68 user    6.61 sys
Minix: PID 469 exited
6.40 sys
6.70Minix: PID 466 exited
sys
Minix: PID 468 exited
fstime completed
6.00--
sys
Minix: PID 467 exited
fstime completed
Minix: PID 464 exited
---
fstime completed
---
Minix: PID 461 exited
Minix: PID 463 exited
fstime completed
---
Minix: PID 462 exited
Minix: PID 460 exited
# quantum allotted : 500, quantum used: 267
200010027:PID 23 swapped in
```

Figure 12: `workload_mix1`

2.2 workload_mix2.sh

workload_mix2.sh contains 3 arithoh.sh and 2 fstime.sh processes and the code for it is as follows:

```
1 #!/bin/sh
2 ./arithoh.sh &
3 ./arithoh.sh &
4 ./arithoh.sh &
5 ./fstime.sh &
6 ./fstime.sh &
7 wait
```

Figure 13: workload_mix2 code

In this workload, first we are calling 3 arithoh.sh which are CPU intensive process so they follow FIFO policy i.e., they exit in the order the processes have arrived. After that, the 2 fstime.sh which are I/O bound processes are scheduled. These process wait for I/O, in the mean time another process is executed, so in this way these processes execute alternatively waiting for I/O which results in round robin policy.

[illegible]

Figure 14

```

quantum allotted : 200, quantum used: 200
200010027:PID 122 swapped in
quantum allotted : 200, quantum used: 200
200010027:PID 122 swapped in
quantum allotted : 200, quantum used: 25
200010027:PID 122 swapped in
Minix: PID 372 exited
      34.65 real      34.65 real      34.65 real      11.08      11.56 user user
      11.95 user      0.00 sys
      0.01Minix: PID 365 exited
sys
Minix: PID 366 exited
      0.00 sys
Minix: PID 367 exited
arithoh completed
----
arithoh completed
----
Minix: PID 360 exited
Minix: PID 361 exited
arithoh completed
----
Minix: PID 362 exited
quantum allotted : 500, quantum used: 500

```

Figure 15

2.3 workload_mix3.sh

The following is the code for workload_mix3.sh which has arithoh.sh and spawn.sh processes alternatively.

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

Figure 16: workload_mix3 code

Even though the arithoh processes are not continuous in this workload_mix, they still quit one after the other consecutively. The arithoh processes are forked by the spawn processes. All arithoh processes which are CPU bound in nature will be finished in FIFO order in the interim while these child processes are being created, and then the child processes produced by spawn will be completed. Hence, all the arithoh processes will be completed first and the later on spawn processes are finished at the end. So, for this workload_mix, the scheduler gives different output from that of original minix round robin scheduler.

```
200010027:PID 33 swapped in
Minix: PID 10466 created
quantum allotted : 200, quantum used: 200
200010027:PID 35 swapped in
0.00 sys
Minix: PID 10446 exited
0.00 sys
Minix: PID 10448 exited
0.00 userMinix: PID 10465 exited
Minix: PID 10466 exited
arithoh completed
---
Minix: PID 10467 created
quantum allotted : 200, quantum used: 200
200010027:PID 36 swapped in
Minix: PID 10468 created
quantum allotted : 200, quantum used: 200
200010027:PID 37 swapped in
arithoh completed
Minix: PID 10441 exited
---
Minix: PID 10443 exited
Minix: PID 10467 exited
Minix: PID 10468 exited
```

Figure 17

```
200010027:PID 233 swapped in
Minix: PID 10437 exited
Minix: PID 10439 created
quantum allotted : 200, quantum used: 200
200010027:PID 234 swapped in
Minix: PID 10438 exited
Minix: PID 10439 exited
Minix: PID 20392 exited
Minix: PID 20394 exited
1:05.83 real 1:05.83 real 0.20 user 0.30 user 6.30 sys
Minix: PID 20387 exited
6.13 sys
Minix: PID 20389 exited
spawn completed
---
Minix: PID 20382 exited
spawn completed
---
Minix: PID 20384 exited
Minix: PID 20380 exited
# quantum allotted : 200, quantum used: 1
200010027:PID 32 swapped in
quantum allotted : 200, quantum used: 2
200010027:PID 151 swapped in
```

Figure 18

2.4 workload_mix4.sh

The following is the code for workload mix4.sh which contains 5 syscall.sh

```
1 #!/bin/sh
2 ./syscall.sh &
3 ./syscall.sh &
4 ./syscall.sh &
5 ./syscall.sh &
6 ./syscall.sh &
7 wait
```

Figure 19: workload_mix4 code

In this workload_mix, we are calling 5 syscall.sh processes continuously. As all the processes are CPU bound, a process is scheduled only after completion of the running process. Thus they are executed one by one in FIFO manner which we can see from the output. So, pseudo FIFO and minix round robin schedulers are not same for this workload.

```
Minix: PID 390 created
quantum allotted : 200, quantum used: 200
200010027:PID 137 swapped in
Minix: PID 391 created
quantum allotted : 200, quantum used: 200
200010027:PID 138 swapped in
Minix: PID 392 created
quantum allotted : 200, quantum used: 200
200010027:PID 139 swapped in
Minix: PID 393 created
quantum allotted : 200, quantum used: 200
200010027:PID 140 swapped in
Minix: PID 394 created
quantum allotted : 200, quantum used: 200
200010027:PID 141 swapped in
Minix: PID 395 created
quantum allotted : 200, quantum used: 200
200010027:PID 143 swapped in
Minix: PID 396 created
quantum allotted : 200, quantum used: 200
200010027:PID 145 swapped in
Minix: PID 397 created
quantum allotted : 200, quantum used: 200
200010027:PID 146 swapped in
```

Figure 20: workload_mix4 code

```
Minix: PID 376 exited
33.20 real    2.50 user    5.50 sys
Minix: PID 371 exited
syscall completed
---
Minix: PID 366 exited
quantum allotted : 200, quantum used: 200
quantum allotted : 200, quantum used: 200
quantum allotted : 200, quantum used: 200
quantum allotted : 200, quantum used: 200
quantum allotted : 500, quantum used: 500
200010027:PID 37 swapped in
quantum allotted : 200, quantum used: 200
quantum allotted : 200, quantum used: 200
quantum allotted : 200, quantum used: 200
quantum allotted : 200, quantum used: 200
200010027:PID 128 swapped in
Minix: PID 378 exited
40.51 real    2.65 user    5.68 sys
Minix: PID 373 exited
syscall completed
---
Minix: PID 368 exited
Minix: PID 363 exited
```

Figure 21: workload_mix4 code

3 Why Pseudo FIFO?

I/O bound processes when requested for I/O are sent to the waiting queue until they receive and once after receiving I/O, they are scheduled to work on CPU. Also, I/O bound processes do not utilize the entire time slice. When I/O processes wait for I/O, in the meantime other processes can be scheduled. So, I/O bound processes follow round robin even though the FIFO policy is applied to the scheduler. Therefore, since all the processes do not follow exact FIFO order, this implementation of scheduler is called **Pseudo FIFO**.