

Scheduler Comparison

Neda Amirirad
Abdullah Al Monsur
Ricardo Frumento

March 2024

1 Tests

1.1 Function *ticks_running*

To test if the function returned an acceptable value for ticks, a simple c program was written that starts a process, prints in the screen the message "waiting..." and after 1000 iterations prints the total ticks that process has been running. The result can be seen in figure 1.

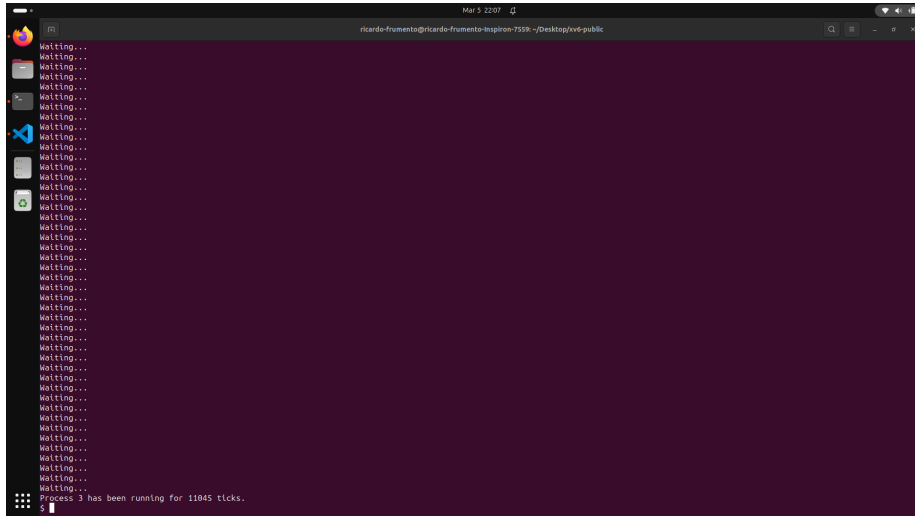


Figure 1: Test result for the ticks running

1.2 Simple Scheduler

After implementing the FIFO scheduler, to check its functionality another short c program was written. This time the program has for loop that runs four times,

each time forking the parent process. It is expected that the processes queue would be [child 1, child 2, child 3, child 4], this can be seen on figure 2. Also it is possible to see that as soon as the process starts running it is not in the queue anymore, and as soon as the current process ends, the next starts running.

```

SeaBIOS (version 1.16.2-debian-1.16.2-1)

IPXE (https://ipxe.org) 00:03:0 CAB0 PCI2.10 PnP PMM+1EFCB160+1EF0B160 CAB0

Booting from Hard Disk...xv6...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ simple_scheduler_test
Child process 4 (parent 3)
Child process 5 (parent 3)
Child process 6 (parent 3)
Child process 7 (parent 3)
Queue while process 4 is running:
Process 5 is in position 1
Process 6 is in position 2
Process 7 is in position 3
Queue while process 5 is running:
Process 6 is in position 1
Process 7 is in position 2
Queue while process 6 is running:
Process 7 is in position 1
Queue while process 7 is running:
Process 3 has been running for 13 ticks.
$

```

Figure 2: FIFO scheduler

1.3 Advanced Scheduler

To show functionality of the Lottery scheduler two runs were made with the same program. As the Lottery scheduler has a chance attribute include, the order of the processes being run should change. That can be seen in figure 3 and 4.

```
SeabIOS (version 1.16.2-debian-1.16.2-1)
IPXE (https://ipxe.org) 00:03:0 CA00 PCI2.10 PnP PMM+1EFCB160+1EF0B160 CA00

Booting from Hard Disk...xv6...
cpu0: Starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ advanced_scheduler_test
Child process 6 (parent 3)
Child process 5 (parent 3)
Child process 4 (parent 3)
Child process 7 (parent 3)
Process 3 has been running for 12 ticks.
$
```

Figure 3: First run of the Lottery scheduler

```
SeabIOS (version 1.16.2-debian-1.16.2-1)
IPXE (https://ipxe.org) 00:03:0 CA00 PCI2.10 PnP PMM+1EFCB160+1EF0B160 CA00

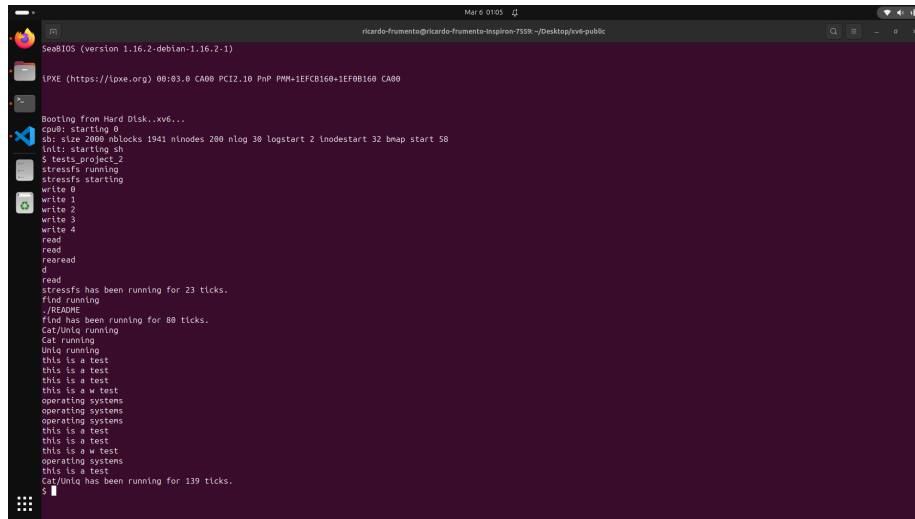
Booting from Hard Disk...xv6...
cpu0: Starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ advanced_scheduler_test
Child process 4 (parent 3)
Child process 6 (parent 3)
Child process 7 (parent 3)
Child process 5 (parent 3)
Process 3 has been running for 12 ticks.
$
```

Figure 4: Second run of the Lottery scheduler

1.4 Workload Test

To compare the three schedulers a more complex c program was created. This has three functions being called in the main function, one that will fork and execute the tests in stressfs, one that will fork and execute the find command, and the last one that will fork and run the command cat and uniq one after the

other. It is possible to see in the output that all three implementations print in different orders in the terminal. The default scheduler shows some weird formatting due to the fact that the fork calls are not handled well.



```

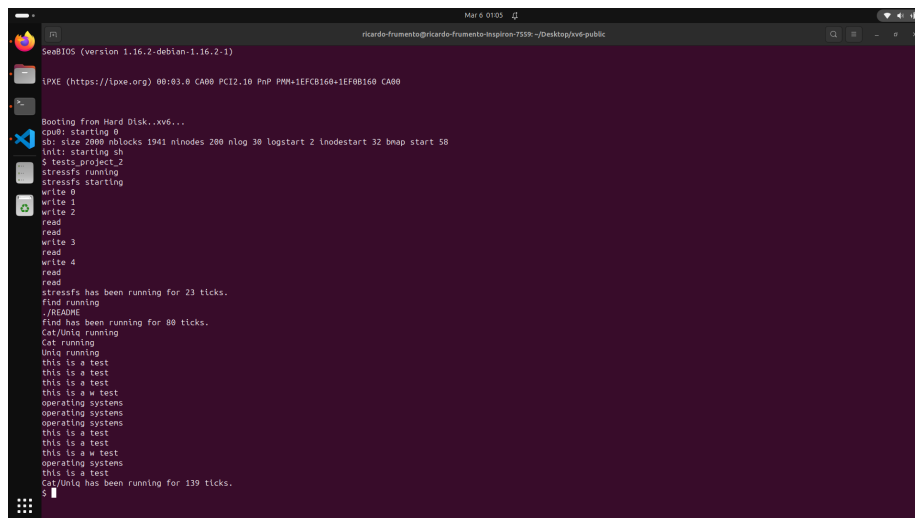
SeabIOS (version 1.16.2-debian-1.16.2-1)
IPXE (https://ipxe.org) 08:03:0 CAB0 PCI2.10 PnP PMM+1EFCB160+1EF0B160 CAB0

Booting from Hard Disk...xv6...
cpu0: starting 0
sb: size 2000 nblocks 1941 nnodes 280 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ tests_project_2
stressfs running
stressfs starting
write 0
write 1
write 2
write 3
write 4
read
read
read
read
read
read
stressfs has been running for 23 ticks.
find running
./README
find has been running for 80 ticks.
cat/uniq running
cat running
uniq running
this is a test
this is a test
this is a test
this is a w test
operating systems
operating systems
operating systems
operating systems
this is a test
this is a test
this is a w test
operating systems
this is a test
cat/uniq has been running for 139 ticks.
$

```

Figure 5: Default scheduler results

The FIFO scheduler has a steady execution, every new process is added to the queue and will wait for it to run until it is dequeued.



```

SeabIOS (version 1.16.2-debian-1.16.2-1)
IPXE (https://ipxe.org) 08:03:0 CAB0 PCI2.10 PnP PMM+1EFCB160+1EF0B160 CAB0

Booting from Hard Disk...xv6...
cpu0: starting 0
sb: size 2000 nblocks 1941 nnodes 280 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ tests_project_2
stressfs running
stressfs starting
write 0
write 1
write 2
read
read
write 3
read
write 4
read
read
stressfs has been running for 23 ticks.
find running
./README
find has been running for 80 ticks.
cat/uniq running
cat running
uniq running
this is a test
this is a test
this is a test
this is a w test
operating systems
operating systems
operating systems
operating systems
this is a test
this is a test
this is a w test
operating systems
this is a test
cat/uniq has been running for 139 ticks.
$

```

Figure 6: FIFO scheduler results

The final Lottery scheduler has also steady execution but does execute out of order as seen in 7 (uniq runs before cat but in the program cat comes first).

```

SeebIOS (version 1.16.2-debian-1.16.2-1)
iPXE (https://ipxe.org) 00:03:0 CAB0 PCI2.10 PnP PMM+1EF0B160+1EF0B160 CAB0

Booting from Hard Disk...xv6...
cpu0: starting 0
sbi: size 2000 nblocks 1941 nnodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
$ tests_project_2
stressfs running
stressfs starting
write 0
write 1
write 2
write 3
read
write 4
read
read
read
stressfs has been running for 23 ticks.
find running
./README
find has been running for 80 ticks.
Cat/Uniq running
Uniq running
Cat running
this is a test
this is a w test
operating systems
this is a test
this is a test
this is a test
this is a test
this is a w test
operating systems
operating systems
operating systems
operating systems
this is a test
Cat/Uniq has been running for 139 ticks.
$

```

Figure 7: Lottery scheduler results

2 Discussion

After observing the two implementations, it is possible to see that both added schedulers have advantages when comparing to the default scheduler. One thing in common about both schedulers is that they run until the end of the process, that is what gives its well formatted output during the simple tests. They are equivalent for the tests here but both could show very bad performance if there is a long process currently running.