Owner’s Manual for p4-test

CS333 Course Staff

February 12, 2019

Welcome To Your New `p4-test` Program!

This `p4-test` program has been designed to make it easy for you to construct tests of your MLFQ. This document will introduce you to `p4-test` and hopefully give you a good start to testing your MLFQ scheduler’s functionality.

This document will describe the code and then walk through the execute of the `p4-test`.

Keep in mind that you are under no obligation to use this test program. If you do use `p4-test.c`, do not modify the code without first reaching out to course staff.

About `p4-test.c`

The code in `p4-test.c` has been written with an eye for readability. Extensive documentation has been provided to help you understand what the code is doing. In turn, you can use this understanding of the code when documenting your own testing.

The test program operates by creating `P4T_TO_CREATE` programs every second; the default value of `P4T_TO_CREATE` is 5. This should make it easy to observe processes moving through multiple priority queues—some processes will have performed more work than others. You should be able to observe these programs by pressing `C-s`. However, you might occasionally see them on one of the runnable lists. Think about why that might happen…

Once all of the processes (60 by default) have been created, `p4-test` will print a message every 10 seconds prompting you to test your system.

Child Processes

Every 7th process (`PID % 7 ==0`) will be put to sleep for a very long time before finally exiting.

The remaining processes will repeatedly count from `COUNTER_START` to `COUNTER_END` until `P4T_SECONDS` have passed. By default, each active process will run for 60 seconds.

Using `p4-test`

Make sure you have copied the new `p4-test.c` into your source tree and that it's added to `CS333_TPROGS` as well as `runoff.list`.

Running `p4-test`

Build your kernel and run `p4-test` on the command line. The program will display the message "Created 5 processes. Sleeping for 1 seconds." 12 times. If you were to press `C-p` during the start up process, you might see something like the following.

```
Created 5 processes. Sleeping for 1 seconds.

PID     Name            UID     GID     PPID    Prio        Elapsed     CPU     State   Size    PCs
1       init            0       0       1       6           8.699       0.100   sleep   12288   8010491a 80104a7e 80105
e99 80107025 80106e5c
2       sh              0       0       1       6           8.591       0.066   sleep   16384   8010491a 80104a7e 80105
e99 80107025 80106e5c
4       p4-test         0       0       2       6           2.046       0.163   runble  12288
5       p4-test         0       0       4       4           2.001       0.568   runble  12288
6       p4-test         0       0       4       4           1.987       0.557   runble  12288
7       p4-test         0       0       4       6           1.973       0.000   runble  12288
8       p4-test         0       0       4       4           1.953       0.550   runble  12288
9       p4-test         0       0       4       4           1.941       0.550   runble  12288
10      p4-test         0       0       4       4           0.869       0.403   run     12288
11      p4-test         0       0       4       4           0.860       0.400   run     12288
12      p4-test         0       0       4       4           0.846       0.400   runble  12288
13      p4-test         0       0       4       4           0.825       0.400   runble  12288
14      p4-test         0       0       4       6           0.806       0.000   runble  12288
Created 5 processes. Sleeping for 1 seconds.
Created 5 processes. Sleeping for 1 seconds.
Created 5 processes. Sleeping for 1 seconds.
Created 5 processes. Sleeping for 1 seconds.
Created 5 processes. Sleeping for 1 seconds.
Created 5 processes. Sleeping for 1 seconds.
```

Here, the original `p4-test` has a PID of 4. There are 10 total child processes (`p4-test` isn't done starting up). Once `p4-test` has finished its start up process you should see 61 processes with the name `p4-test`

Testing with `p4-test`

After `p4-test` is done warming up, you will see a message prompting you to use `C-p`, `C-r`, and `C-s` to test your implementation of the MLFQ.

```
Now verify that your system is working by pressing C-p and then C-r.


Ready List Processes:

Priority 6: (4, 200) --> (10, 140) --> (34, 141) --> (12, 140) --> (27, 140) -->
 (40, 140) --> (43, 140) --> (44, 140) --> (36, 140) --> (18, 140) --> (5, 140)
--> (46, 140) --> (47, 140) --> (17, 140) --> (26, 140) --> (8, 140) --> (37, 14
0) --> (25, 140) --> (38, 140) --> (9, 140) --> (59, 140) --> (19, 140) --> (41,
 140) --> (11, 140) --> (24, 140) --> (51, 140) --> (53, 140) --> (54, 140) -->
(52, 140) --> (16, 140) --> (55, 140) --> (60, 140) --> (30, 140) --> (15, 140)
--> (45, 140) --> (29, 140) --> (22, 140) --> (39, 140) --> (64, 140) --> (48, 1
40) --> (20, 140) --> (6, 140) --> (61, 140) --> (57, 150) --> (23, 150) --> (50
, 150) --> (32, 150) --> (31, 150) --> (33, 151) --> (58, 151)
Priority 5: None.
Priority 4: None.
Priority 3: None.
Priority 2: None.
Priority 1: None.
Priority 0: None.

Sleep List Processes:
1 --> 2 --> 21 --> 7 --> 49 --> 42 --> 56 --> 28 --> 63 --> 14 --> 35 --> 4
```

p4-test

In this screen shot, we can see that there are a number of processes on the highest priority queue. In addition, we can verify that a number of processes are asleep, including the parent `p4-test` process. The parent is asleep for one of two reasons. One possibility is that the parent process is sleeping between prompting you to use control sequences. The other possibility is that it is waiting for child processes to exit.

Observing Promotion and Demotion

Promotion and demotion should not be difficult to observe. If you are having problems, adjust the values of `BUDGET` and `TICKS_TO_PROMOTE`. Course staff recommend values of 200 and 2000, respectively, as a good starting point.

Take a look at this screen shot:

```
Ready List Processes:

Priority 6: None.
Priority 5: None.
Priority 4: None.
Priority 3: None.
Priority 2: None.
Priority 1: (45, 142) --> (58, 138) --> (60, 140) --> (64, 141) --> (61, 136) --> (9, 131) --> (16, 125) --> (5, 132) --
> (15, 144) --> (29, 129) --> (11, 135) --> (31, 134) --> (40, 140) --> (51, 123) --> (6, 138) --> (8, 139) --> (25, 146
) --> (13, 132) --> (30, 135) --> (32, 130) --> (12, 128) --> (26, 134) --> (19, 132) --> (55, 133) --> (18, 135) --> (3
9, 129) --> (37, 131) --> (43, 124) --> (41, 140) --> (44, 130) --> (47, 129) --> (48, 126) --> (46, 122) --> (54, 125)
--> (53, 136) --> (17, 132) --> (50, 132) --> (10, 121) --> (23, 139) --> (59, 132) --> (22, 145) --> (20, 130) --> (33,
 127) --> (27, 141) --> (34, 128) --> (36, 121) --> (38, 130) --> (52, 137) --> (24, 152) --> (57, 140)
Priority 0: None.


Ready List Processes:

Priority 6: None.
Priority 5: (7, 34) --> (42, 23) --> (28, 29)
Priority 4: None.
Priority 3: None.
Priority 2: (4, 112)
Priority 1: (46, 98) --> (48, 106) --> (53, 115) --> (54, 95) --> (17, 106) --> (50, 104) --> (10, 90) --> (23, 101) -->
 (22, 125) --> (59, 103) --> (20, 100) --> (33, 95) --> (27, 112) --> (34, 99) --> (36, 91) --> (38, 104) --> (52, 107)
--> (24, 124) --> (57, 108) --> (62, 123) --> (45, 100) --> (60, 102) --> (58, 101) --> (64, 100) --> (61, 99) --> (16,
84) --> (9, 92) --> (29, 98) --> (5, 88) --> (11, 103) --> (15, 110) --> (31, 75) --> (51, 92) --> (40, 100) --> (6, 104
) --> (8, 99) --> (25, 108) --> (13, 95) --> (30, 95) --> (32, 89) --> (12, 82) --> (26, 95) --> (19, 95) --> (55, 101)
--> (18, 93) --> (37, 92) --> (39, 93) --> (41, 109) --> (43, 92) --> (44, 93)
Priority 0: None.

Sleep List Processes:
1 --> 2 --> 35 --> 14 --> 56 --> 21 --> 63
Now verify that your system is working by pressing C-p and then C-r.
Now verify that your system is working by pressing C-p and then C-r.
```

In this screen shot, we can observe multiple things happening:

1. Three sleeping processes (`PID % 7 ==0`) have been woken up.
2. A number of processes on different priority queues.
3. 7 sleeping processes (`init`, `sh`, and 5 instances of `p4-test`).

Although you and I know that a demotion has occurred (processes are not at the default priority), you will need to capture processes moving between lists. Likewise, you should capture processes moving between lists to prove that promotion has occurred.

Cool Down

After the `p4-test` workers are done running, the sleeping processes will remain. It may take a while for them to go away—remember that they've been set up to sleep for a very long time.

During the cool down process, `p4-test` will sleep for one second while repeatedly calling `wait()` until all child processes have exited. We know when all child processes are done running because the call to `wait()` will return -1.