094210 – computer organization and operating systems

version	date	change	comment
1.3	2020-06-09	Fix def of set_priority()	
2.0	2021-05-18		Try to make it easier
2.1	2021-05-25	Add example slides	
2.11	2021-06-01	Submit a patch	
2.12	2021-06-08		Ready for review
2.13	2021-06-08	Fix comments	
2.14	2021-06-09	Clarify algorithm	RR behavior

Assignment 2:

implement Multi Level Queue (MLQ) scheduler in xv6

In this task you will replace the existing process scheduler with an MLQ scheduler. Note that this is a different policy than the 'multilevel feedback queue' that we learned in class. There are diagrams that demonstrate this policy at the bottom of this document.

The objectives:

1. Understand how context switching is done in the kernel
2. Hands-on with a scheduling algorithm

Contents

Preparations 2

Reading 2

Use the correct source code in the linux Virtual Machine 2

Need to start over? 2

The Task 3

EXAMPLES 5

Testing yourself 7

Preparations

Reading

- Read chapter 6 (Scheduling) in the guide:

https://pdos.csail.mit.edu/6.828/2019/xv6/book-riscv-rev0.pdf

- Read the current implementation in XV6 of the round-robin scheduler: the function

```
void scheduler(void)
```

in the file `proc.c`. This is the main function that you will modify in this exercise. We recommend that before you start doing so, keep on the side the original code, and the original executable, so you will have a reference.

Use the correct source code in the linux Virtual Machine

You will use the same Azure virtual machine from previous homework.

The directory `xv6-public` in the virtual machine has to be prepared with the correct version of the source code. FAILING TO FOLLOW THESE STEPS WILL CAUSE YOUR SUBMISSION TO FAIL.

```
cd xv6-public
git reset --hard
git clean -f
git remote set-url origin https://github.com/noam1023/xv6-public
git pull
git checkout mlq
make clean qemu-nox
```

Now run the command 'sanity' in the xv6 shell.

Delete a place holder file

Delete the file "set_priority.c"

Need to start over?

If you made a terrible mess and you want to start from a completely new directory, **erasing** the old directory:

```
cd /
rm -rf xv6-public # this will delete the directory!
```

```
git clone https://github.com/noam1023/xv6-public
```

and then continue with the above steps

The Task

Your MLQ scheduler must follow these rules:

- There should be three priority levels, numbered from 2 (highest) down to 0 (lowest). At creation, a process starts with priority 1.
- Whenever the xv6 timer tick occurs (by default this happens every 10 ms), the highest priority process which is ready ('RUNNABLE') is scheduled to run. That is, this is a preemptive scheduler.
- The highest priority ready process is scheduled to run whenever the previously running process exits, sleeps, or otherwise 'yields' the CPU.
- Your scheduler should schedule all the processes at each priority level in a round robin fashion.
- When a timer tick occurs, whichever process was currently using the CPU should be considered to have used up an entire timer tick's worth of CPU, even if it did not start at the previous tick (note that a timer tick is different than the time-slice).

- Time-slices:

Priority	Timer ticks in one time slice:
2	8
1	16
0	32

- If a process voluntarily relinquishes the CPU before its time-slice expires at a particular priority level, its time-slice is reset; the next time that that process is scheduled, it will have a new time-slice at that priority level.
- **A process in a given priority completes its time slice before other processes in the same priority can run**. In other words, a new process will not preempt the running process in this priority. The running process will stop running if preempted by a higher priority process, or it relinquish (i.e. give away) the CPU.

Implement a system call to set the current process priority and a user space function that will call the system call:

```
/**
  set the current process priority (0..2)
```

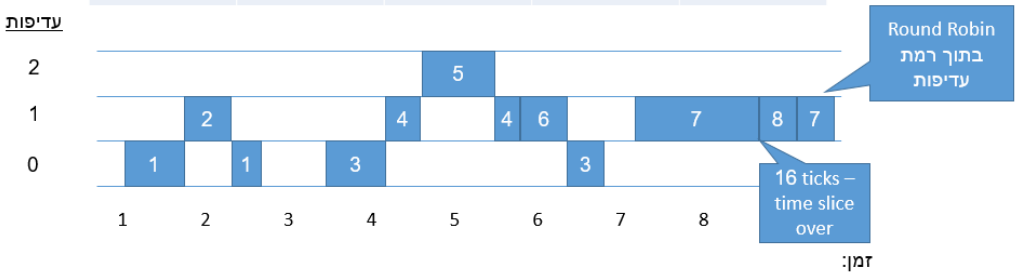
```
  @return 0 if success, non zero if error
 */
int set_priority(int new_priority);
```

NOTE: there is currently a placeholder called “set_priority.c” . Delete this file.

EXAMPLES

דוגמה 1.
Multi-level Queue (MLQ)
הנחות:
זמן בשניות
כל tick = 0.1 שניה.

תהליך	עדיפות	הגעה	התחלה	סיום
1	0	1	1	2.7
2	1	1.65	1.7	2.2
3	0	3.5	3.5	6.7
4	1	4.06	4.1	5.7
5	2	4.6	4.6	5.5
6	1	4.5	5.7	6.4
7	1	7.1	7.1	9.2
8	1	7.2	8.7	8.8



Pay attention to process 2: it arrives at 1.65, which is inside a time tick interval. The scheduler will look which process to run at the next tick which is 1.7

הגעה: הזמן שהתהליך מבקש להתחיל לרוץ

התחלה: הזמן בו התהליך מקבל את המעבד

סיום: הזמן בו התהליך הסתיים

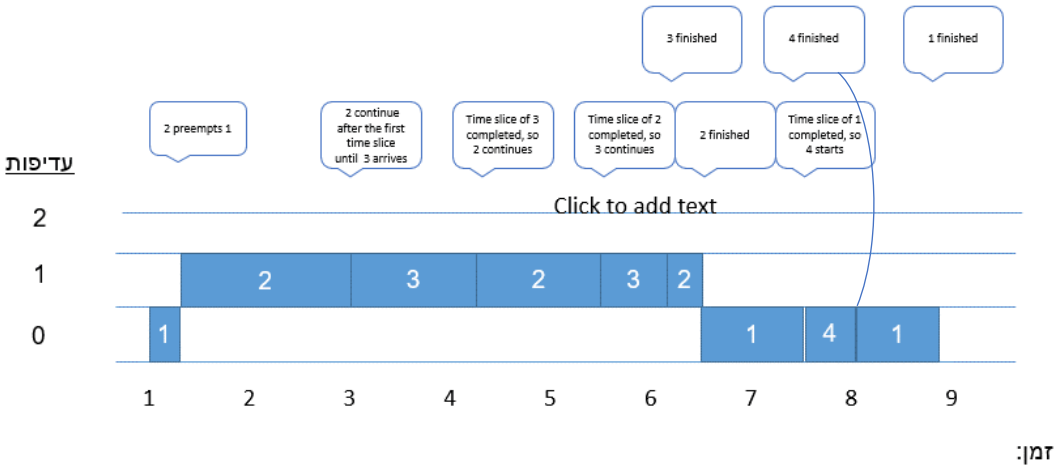
TICK TIME לצורך הפשטות ניתן כ 0.1 שניה

TIMESLICE בעדיפות 1 הוא 16

דוגמה 2.
Multi-level Queue (MLQ)

הנחות:
זמן בשניות
כל tick = 0.1 שניה.

תהליך	עדיפות	הגעה	התחלה	סיום
1	0	1	1	8.9
2	1	1.2	1.2	6.5
3	1	3.0	3	6.1
4	0	3.5	7.6	8.0



In this example, we show that process 2 runs after it completed its first time slice because nobody with its priority (or higher) requests the CPU. **When Process 3 arrives, it waits until process 2 for completed the current time slice**.

The same goes for processes 1 and 4: process 1 resumes at 6.5 and completes its time slice. Only then, process 4 (who is waiting impatiently since time 3.5) can run. Process 4 finishes at 8.0 and then process 1 continues (because there are no other processes at this or higher priority)

Testing yourself

Use the command `fairness`. Read the source code of `check_proc_order.c` to learn what is the expected output.

When the homework checker runs, it removes the comment in init.c

// #define SINGLESHOT

You can try it too:

Run the xv6 and it will exit after a single call to check_proc().

Do NOT add the change in init.c, to the patch !

The Checker link:

http://homework-tester.westeurope.cloudapp.azure.com/94210/submit/hw/2

Submission

1. Clean the directory by running `make clean`.
2. Create a patch file named HW5_ID1_ID2.patch
    a. Creating the patch is done the same way as in the previous assignment:
        i. make clean
        ii. git diff > HW5_ID1_ID2.patch
3. Upload the patch to the test server. Once you are happy with the result, submit to Moodle.

Due date: 2021-06-24