EECS 750 Homework #1

Playing with Linux Schedulers

In this homework, you will learn to interact with Linux's CPU schedulers and monitor their behaviors. In the process, you will also learn to use several standard tools and scheduling related system call APIs. **You should submit four files: hw1-1.png, hw1-2.png, hw1-3.png and hw1-4.c**

You need to have an access to a Linux computer. Alternatively, you can download VirtualBox and install Ubuntu 16.04 there. *Note that the virtual machine must be configured to have at least two CPU cores.*

Part 1. Using scheduler related tools

On a terminal, create the following 'cpuhog' program and compile it.

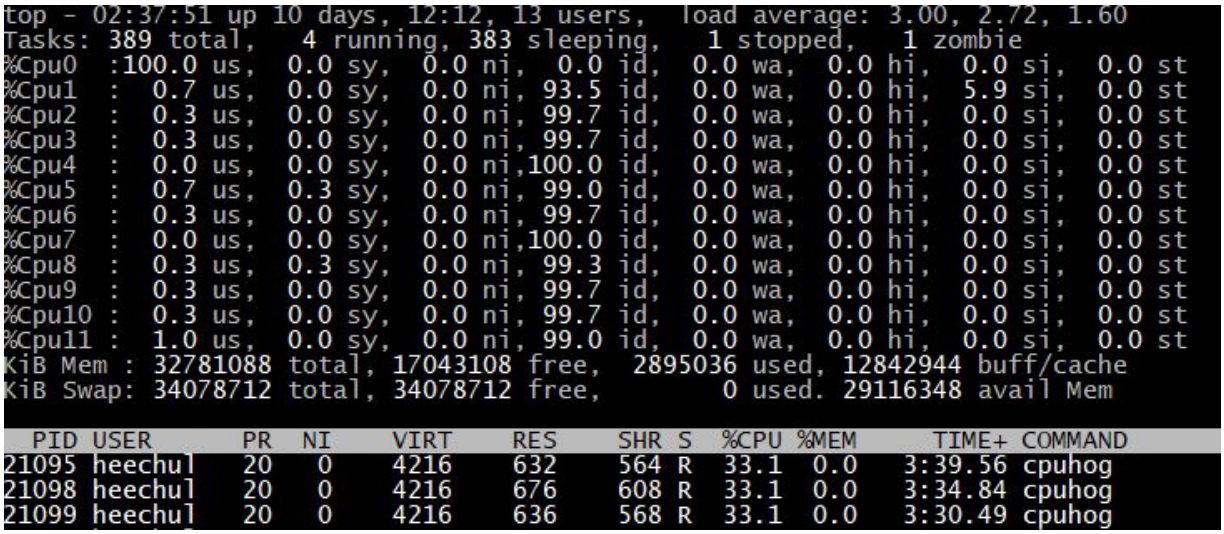
```
$ cat cpuhog.c
int main()
{
        while(1);
}

$ gcc cpuhog.c -o cpuhog
```

Now, you shall use the 'taskset' utility to launch three instances of the 'cpuhog' program as follows.

```
$ taskset -c 0 ./cpuhog &
[1] 21095
$ taskset -c 0 ./cpuhog &
[2] 21098
$ taskset -c 0 ./cpuhog &
[3] 21099
```

The taskset utility controls which CPU core (cores) to execute the given program. In this case, it forces to schedule at core 0 (due to "-c 0").

On the terminal, execute the 'top' program and press '1'. You will see something like the following.



Next, you will change cpuhog instances' nice values (i.e., CFS priority values) using the 'renice' tool. Open up a new terminal so that we can keep monitoring the output of the top.

Check the PID values of the three cpuhog instances shown on the top screen. In the example above, they are 21095, 21098, 21099. Your PID values may be different. You can also check the pid values by using the 'pidof' tool as follows.

```
$ pidof cpuhog
21099 21098 21095
```

Now, let's change the nice value of the first cpuhog instance.

```
$ renice 5 21095
```

On the terminal executing top, monitor the CPU utilization the cpuhog instances. You should see that the CPU utilization of the reniced cpuhog is dropped to around 14% while each of the other two cpuhog instances utilizes around 43%.

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
21098 heechul   20   0    4216    676    608 R  43.0  0.0   8:46.10 cpuhog
21099 heechul   20   0    4216    636    568 R  43.0  0.0   8:41.75 cpuhog
21095 heechul   25   5    4216    632    564 R  13.9  0.0   8:20.28 cpuhog
```

Capture the terminal screen of the 'top' and save it as 'hw1-1.png'. You should return the file as a proof.

Next, you again change the second cpuhog instance's nice value as follows.

```
$ renice 5 21098
```

Then, again monitor the ‘top’ screen. You should see each of the reniced cpuhog instances (21095 and 21098) consumes around 20% and the remaining ‘normal’ instance is consuming around 60%.

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
21099 heechul   20   0    4216    636    568 R  60.5  0.0  15:40.46 cpuhog
21095 heechul   25   5    4216    632    564 R  19.9  0.0  10:37.26 cpuhog
21098 heechul   25   5    4216    676    608 R  19.9  0.0  15:06.33 cpuhog
```

Capture the terminal screen of the ‘top’ and save it as ‘hw1-2.png’. You should return the file as a proof.

```
$ renice 5 21099
```

Monitor the ‘top’ screen. You should now see all cpuhog instances equally share the CPU---33% each..

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
21098 heechul   25   5    4216    676    608 R  33.6  0.0  15:56.59 cpuhog
21095 heechul   25   5    4216    632    564 R  33.2  0.0  11:27.52 cpuhog
21099 heechul   25   5    4216    636    568 R  33.2  0.0  18:01.04 cpuhog
```

Capture the terminal screen of the ‘top’ and save it as ‘hw1-3.png’. You should return the file as a proof.

Next, we will use a real-time scheduler to schedule one of the cpuhog instance. *This requires a root permission. If you don’t have a root access, you can skip this part.*

On the terminal, use the ‘chrt’ tool to change a cpuhog instance’s scheduler from CFS to a real-time scheduler (SCHED_FIFO) with a real-time priority value 1 as follows.

```
$ sudo chrt -f -p 1 21098
```

You should see that the cpuhog instance occupies 100% cpu time, while the other two use 0% .

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
21098 heechul   -2   5    4216    676    608 R 100.0  0.0  22:31.53 cpuhog
```

Then, change the task’s scheduler back to the CFS as follows.

```
$ sudo chrt -o -p 0 21098
```

Part 2. Using scheduler related system calls

So far, you have used ‘taskset’ and ‘renice’ tools to control cpu core and nice values of your cpuhog program. Now, instead of using these external tools, you need to modify the cpuhog.c code and directly use system calls.

First, modify cpuhog.c to be always scheduled on core 0. You need to use ‘sched_setaffinity’ system call. See the manual.

```
$ man sched_setaffinity
```

Then, use the ‘setpriority’ system call to change the task’s nice value to 5. Again, see the manual for usage.

```
$ man setpriority
```

Save the modified code as hw1-4.c and return it.