EECS 750 Homework #2

Playing with Linux CGROUP

In this homework, you will learn to interact with Linux's CGROUP (control group).
You should submit three files: hw2-1.png, hw2-2.png, hw2-3.png

You need to have an access to a Linux computer. Alternatively, you can download VirtualBox and install Ubuntu 16.04 there. In addition, you need a root shell access to the computer to complete the homework.

Part 0. Preparation

On a terminal, create the 'cpuhog' program and copy the binary as follows.

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

$ gcc cpuhog.c -o cpuhog
$ cp -v cpuhog phd
$ cp -v cpuhog master
$ cp -v cpuhog under
```

Execute them twice as follows.

```
$ under &
$ under &
$ phd &
$ phd &
$ master &
$ master &
```

Now, they will be scheduled on any available cores in your computer at the time. For example, on my computer, the 'top' result after that was as follows. Notice that the 6 processes we launched are all over the cores. This is because Linux's load balancer distributed the workload.

```
top - 05:13:38 up 17 days, 14:47, 17 users,  load average: 5.34, 3.26, 2.28
Tasks: 366 total,   8 running, 353 sleeping,   4 stopped,   1 zombie
%Cpu0  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu2  :  0.3 us,  0.0 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.3 si,  0.0 st
%Cpu3  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu4  :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu5  :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu6  :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu7  :  0.0 us,  0.7 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu8  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu9  :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu10 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu11 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32781088 total, 13861440 free,  2890424 used, 16029224 buff/cache
KiB Swap: 34078712 total, 34078712 free,        0 used. 28991852 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
16708 heechul   20   0    4216    620    552 R 100.0  0.0   2:02.43 under
16709 heechul   20   0    4216    628    560 R 100.0  0.0   2:01.74 under
16777 heechul   20   0    4216    640    572 R 100.0  0.0   0:42.66 master
16778 heechul   20   0    4216    632    564 R 100.0  0.0   0:41.77 master
16772 heechul   20   0    4216    728    664 R  99.7  0.0   0:48.86 phd
16773 heechul   20   0    4216    632    564 R  99.7  0.0   0:48.14 phd
```

Part 1. Using ‘cpuset’ controller (subsystem) of CGROUP

The ‘cpuset’ controller can be used to control which cores and memory controllers of the tasks in a cgroup. This is similar to what ‘taskset’ can do but for a group. Here, we use the cpuset controller to consolidate all the previously launched programs on core 0.

First, become a root user

```
$ sudo bash
```

Now, create a ‘core0’ cgroup using ‘cpuset’ controller and assign the core 0 and the first memory controller to the cgroup.

```
# cd /sys/fs/cgroup/cpuset/
# mkdir core0
# echo 0 > core0/cpuset.cpus
# echo 0 > core0/cpuset.mems
```

Now, let’s assign all processes to the created cgroup as follows.

```
# for p in `pidof phd`; do echo $p > core0/tasks ; done
# for p in `pidof master` ; do echo $p > core0/tasks ; done
# for p in `pidof under`; do echo $p > core0/tasks ; done
```

Now, if you look at the top screen, you will see all the processes are now running only on core 0.

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
16708 heechul   20   0    4216    620    552 R  16.9  0.0  12:19.43 under
16772 heechul   20   0    4216    728    664 R  16.9  0.0  11:05.86 phd
16773 heechul   20   0    4216    632    564 R  16.9  0.0  11:05.15 phd
16777 heechul   20   0    4216    640    572 R  16.9  0.0  10:59.66 master
16709 heechul   20   0    4216    628    560 R  16.6  0.0  12:18.73 under
16778 heechul   20   0    4216    632    564 R  16.6  0.0  10:58.77 master
```

Capture the terminal screen of the 'top' and save it as 'hw2-1.png'. You should return the file as a proof.

Part 2. Using 'cpu' controller of CGROUP

The 'cpu' controller can be used to interact with the scheduler as a group (of processes). You can assign CPU share or limit the maximum usage.

First, let's create the following group hierarchy.

```
# cd /sys/fs/cgroup/cpu
# mkdir grad
# mkdir grad/phd
# mkdir grad/master
# mkdir under
```

Then, assign the previously launched phd, master, and under processes to their respective cgroups as follows.

```
# for p in `pidof phd`; do echo $p > grad/phd/tasks; done
# for p in `pidof master`; do echo $p > grad/master/tasks; done
# for p in `pidof under`; do echo $p > under/tasks; done
```

At this point, notice that 'under' processes are using 50% while 'phd' and 'master' are using 25% cpu each. This is because the scheduler assign 50% to 'grad' cgroup and the other 50% to 'under' cgroup.

```
16708 heechul   20   0    4216    620    552 R  25.2  0.0  13:48.78 under
16709 heechul   20   0    4216    628    560 R  24.8  0.0  13:48.07 under
16772 heechul   20   0    4216    728    664 R  12.6  0.0  12:35.40 phd
16773 heechul   20   0    4216    632    564 R  12.6  0.0  12:34.69 phd
16777 heechul   20   0    4216    640    572 R  12.6  0.0  12:21.35 master
16778 heechul   20   0    4216    632    564 R  12.6  0.0  12:20.47 master
```

The current situation seems unfair to grad students who may need more computing resources to conduct research. So, next, we will assign 80% share to ‘grad’ group and only 20% to ‘under’ cgroup as follows.

```
# echo 80 > grad/cpu.shares
# echo 20 > under/cpu.shares
```

The result will be something like the following.

```
  PID USER       PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
16772 heechul    20   0    4216    728    664 R  19.9  0.0  13:36.46 phd
16773 heechul    20   0    4216    632    564 R  19.9  0.0  13:35.76 phd
16777 heechul    20   0    4216    640    572 R  19.9  0.0  13:22.42 master
16778 heechul    20   0    4216    632    564 R  19.9  0.0  13:21.54 master
16709 heechul    20   0    4216    628    560 R  10.3  0.0  15:39.39 under
16708 heechul    20   0    4216    620    552 R  10.0  0.0  15:40.08 under
```

Next, among phd and master groups, we decide to give a bit more cpu share to the phd cgroup (60% phd vs. 40% master) as follows.

```
# echo 60 > grad/phd/cpu.shares
# echo 40 > grad/master/cpu.shares
```

The result will be something like the following.

```
  PID USER       PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
16773 heechul    20   0    4216    632    564 R  24.3  0.0  14:52.82 phd
16772 heechul    20   0    4216    728    664 R  23.9  0.0  14:53.52 phd
16777 heechul    20   0    4216    640    572 R  15.9  0.0  14:40.81 master
16778 heechul    20   0    4216    632    564 R  15.9  0.0  14:39.92 master
16709 heechul    20   0    4216    628    560 R  10.3  0.0  16:18.25 under
16708 heechul    20   0    4216    620    552 R  10.0  0.0  16:18.95 under
```

Notice that phd processes are getting about 48% (100 * 0.8 * 0.6) and the master processes are getting 32% (100 * 0.8 * 0.4) of the cpu time while the under processes are still getting the same 20% of the total cpu share.

Capture the terminal screen of the ‘top’ and save it as ‘hw2-2.png’. You should return the file as a proof.

Now, undergrad students were given a major project (Quash project!). So, we decided to limit the grad group’s maximum cpu share to 50%. To achieve this, you will use CFS bandwidth controller to limit the maximum budget to 50ms (over 100ms default sampling period) as follows.

```
# echo 50000 > grad/cpu.cfs_quota_us
```

Now, the final state will look like the following. Notice that under processes are now getting 50% CPU time, while phd processes are getting 30% and master processes are getting 20%.

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
16709 heechul   20   0    4216    628    560 R  25.6  0.0  18:42.40 under
16708 heechul   20   0    4216    620    552 R  24.6  0.0  18:43.09 under
16773 heechul   20   0    4216    632    564 R  15.3  0.0  18:18.26 phd
16772 heechul   20   0    4216    728    664 R  15.0  0.0  18:18.96 phd
16777 heechul   20   0    4216    640    572 R  10.0  0.0  16:57.77 master
16778 heechul   20   0    4216    632    564 R  10.0  0.0  16:56.88 master
```

Capture the terminal screen of the 'top' and save it as 'hw2-3.png'. You should return the file as a proof.