Embedded System Software Design
Project 2

Problem Definition:

Since multi-core architectures are widely developed in most of hardware platforms, feasible task management mechanisms for multi-core systems are also proposed. In this project to input a real user application, and runs on a multi-core system. By parallelizing some regions in a program, we can reduce the execution duration, but make higher scheduling complexity. We compare FIFO, Round-Robin and CFS scheduler performance and temperature management in Linux system.

Experimental Environment:

✓ PC: i7 (8 cores) or 4 cores

✓ OS: Ubuntu 15.14

✓ Compiler: GCC 5.2.1

Scheduler setting

Linux supports several schedulers, such as FIFO、Round-Robin and CFS. We can use the function “`sched_setscheduler(pid_t pid, int policy, const struct sched_param *param)`” to set the scheduling policy of the process specified by `pid` to policy and the scheduling parameters to “`param`”.

If `pid` is 0, the policy and parameters are set for the calling process. The following policies are available:

- SCHED_FIFO

First in first out. Processes are allowed on the CPU in the order in which they were added to the queue of processes to be run, for each priority.

- SCHED_RR

Round-Robin. Identical to SCHED_FIFO except that a process runs only for a set time slice (see `sched_rr_get_interval()`). Once the process has completed its time slice it is placed on the tail of the queue of processes to be run, for its priority.

- SCHED_OTHER

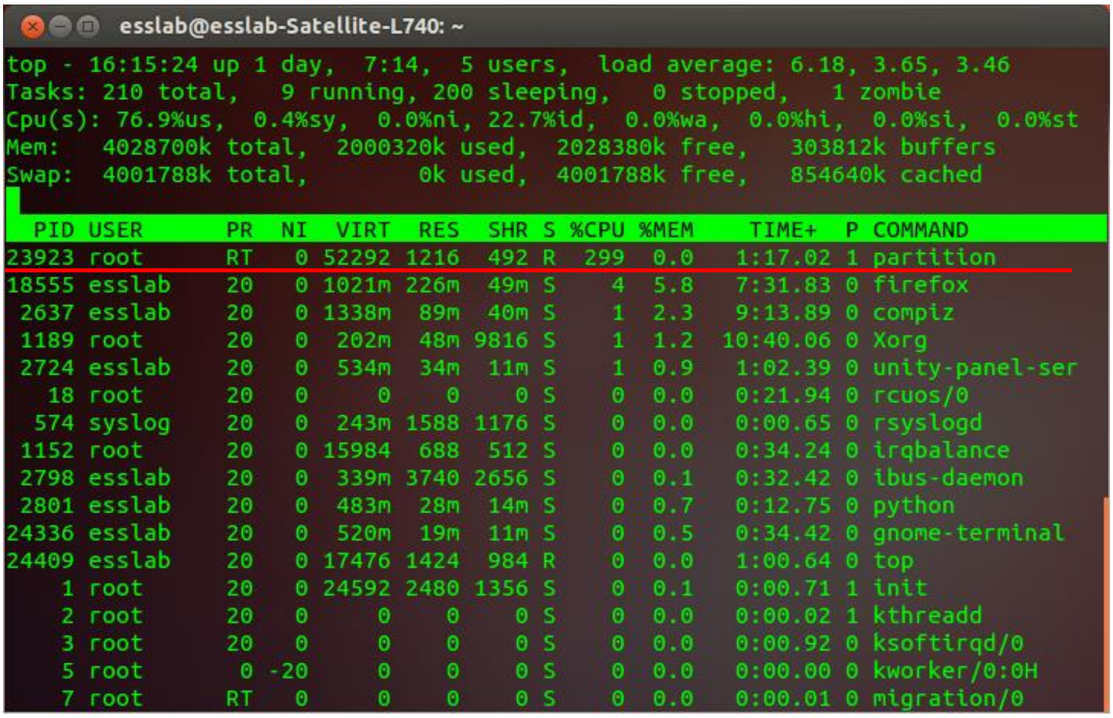
Non-realtime scheduling. This uses the traditional UNIX scheduler.

```
#include <sched.h>

struct sched_param sp;

sp.sched_priority = sched_get_priority_max(SCHED_FIFO);
```

```
ret = sched_setscheduler(0, SCHED_FIFO, &sp);
```

In real-time scheduling policies, we need to set priority of the process. Linux allows the **static priority** value range 1 to 99 for SCHED_FIFO and SCHED_RR and the priority 0 for SCHED_OTHER (non-realtime). We can ues the function "sched_get_priority_max" to get the range of the policies.

In the result, we can figure out that the priority of the process has been "RT" or negative values. It represents the process is running in real time policies and the priority is higher than normal processes.

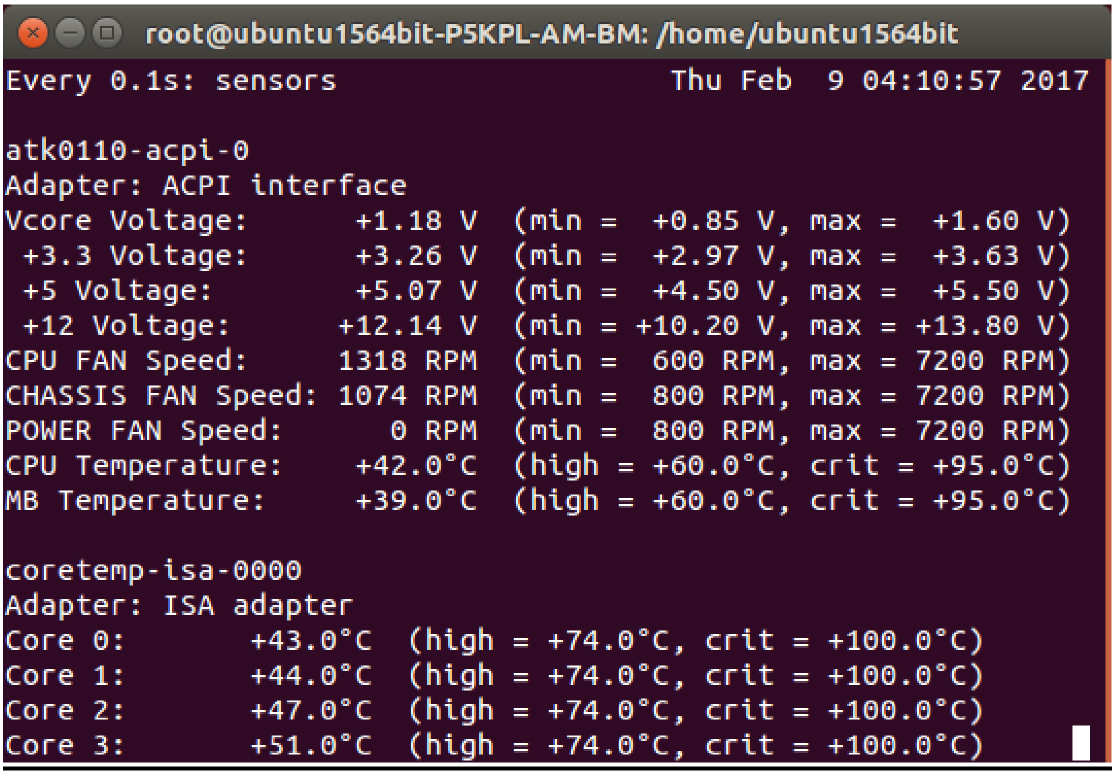


```
top - 16:15:24 up 1 day,  7:14,  5 users,  load average: 6.18, 3.65, 3.46
Tasks: 210 total,   9 running, 200 sleeping,   0 stopped,   1 zombie
Cpu(s): 76.9%us,  0.4%sy,  0.0%ni, 22.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   4028700k total,  2000320k used,  2028380k free,   303812k buffers
Swap:  4001788k total,        0k used,  4001788k free,   854640k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  P COMMAND
23923 root      RT   0 52292 1216  492 R  299  0.0   1:17.02 1 partition
18555 esslab    20   0 1021m 226m  49m S    4  5.8   7:31.83 0 firefox
 2637 esslab    20   0 1338m  89m  40m S    1  2.3   9:13.89 0 compiz
 1189 root      20   0  202m  48m 9816 S    1  1.2  10:40.06 0 Xorg
 2724 esslab    20   0  534m  34m  11m S    1  0.9   1:02.39 0 unity-panel-ser
   18 root      20   0     0    0    0 S    0  0.0   0:21.94 0 rcuos/0
  574 syslog    20   0  243m 1588 1176 S    0  0.0   0:00.65 0 rsyslogd
 1152 root      20   0 15984  688  512 S    0  0.0   0:34.24 0 irqbalance
 2798 esslab    20   0  339m 3740 2656 S    0  0.1   0:32.42 0 ibus-daemon
 2801 esslab    20   0  483m  28m  14m S    0  0.7   0:12.75 0 python
24336 esslab    20   0  520m  19m  11m S    0  0.5   0:34.42 0 gnome-terminal
24409 esslab    20   0 17476 1424  984 R    0  0.0   1:00.64 0 top
    1 root      20   0 24592 2480 1356 S    0  0.1   0:00.71 1 init
    2 root      20   0     0    0    0 S    0  0.0   0:00.02 1 kthreadd
    3 root      20   0     0    0    0 S    0  0.0   0:00.92 0 ksoftirqd/0
    5 root       0 -20     0    0    0 S    0  0.0   0:00.00 0 kworker/0:0H
    7 root      RT   0     0    0    0 S    0  0.0   0:00.01 0 migration/0
```

Temperature Management & CPU Frequencies Scaling:

Running the applications will generate thermal for CPU, and if the CPU frequencies are getting higher the temperature will also be higher. We can use the thermal sensors in the CPUs to read the temperature value, and scale the CPU Frequencies to manage the temperature.

Use this command "#sudo apt-get install lm-sensors" to install the package and use command "#watch –n 0 sensors" to sense the CPU's voltage & temperature



```
Every 0.1s: sensors                          Thu Feb  9 04:10:57 2017

atk0110-acpi-0
Adapter: ACPI interface
Vcore Voltage:       +1.18 V  (min =  +0.85 V, max =  +1.60 V)
 +3.3 Voltage:       +3.26 V  (min =  +2.97 V, max =  +3.63 V)
 +5 Voltage:         +5.07 V  (min =  +4.50 V, max =  +5.50 V)
 +12 Voltage:       +12.14 V  (min = +10.20 V, max = +13.80 V)
CPU FAN Speed:      1318 RPM  (min =  600 RPM, max = 7200 RPM)
CHASSIS FAN Speed:  1074 RPM  (min =  800 RPM, max = 7200 RPM)
POWER FAN Speed:       0 RPM  (min =  800 RPM, max = 7200 RPM)
CPU Temperature:     +42.0°C  (high = +60.0°C, crit = +95.0°C)
MB Temperature:      +39.0°C  (high = +60.0°C, crit = +95.0°C)

coretemp-isa-0000
Adapter: ISA adapter
Core 0:         +43.0°C  (high = +74.0°C, crit = +100.0°C)
Core 1:         +44.0°C  (high = +74.0°C, crit = +100.0°C)
Core 2:         +47.0°C  (high = +74.0°C, crit = +100.0°C)
Core 3:         +51.0°C  (high = +74.0°C, crit = +100.0°C)
```

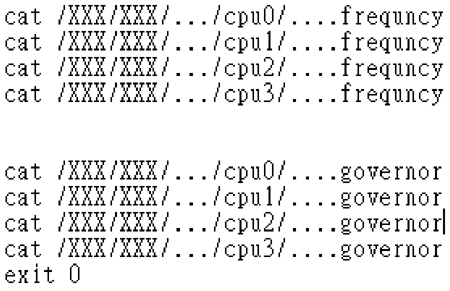
Check the **scaling_available_frequencies** file to know how many frequency levels that your CPU can be scaled.

Change the **scaling_governor** file to change the governor which being used right now.

Change the **scaling_setspeed** file to change frequency for current CPUs.

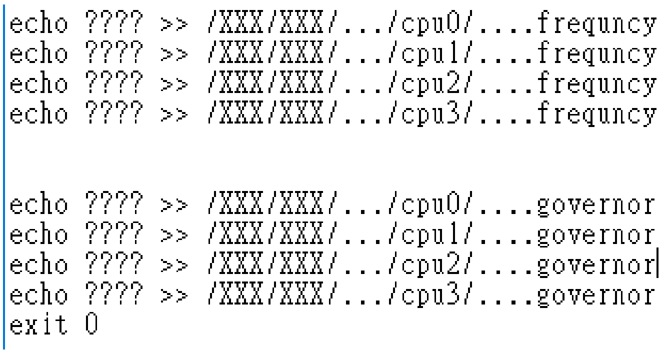
Change the governor to **“userspace”** that we can scale the frequency by ourselves.

Using “cat” to print the information in the file.



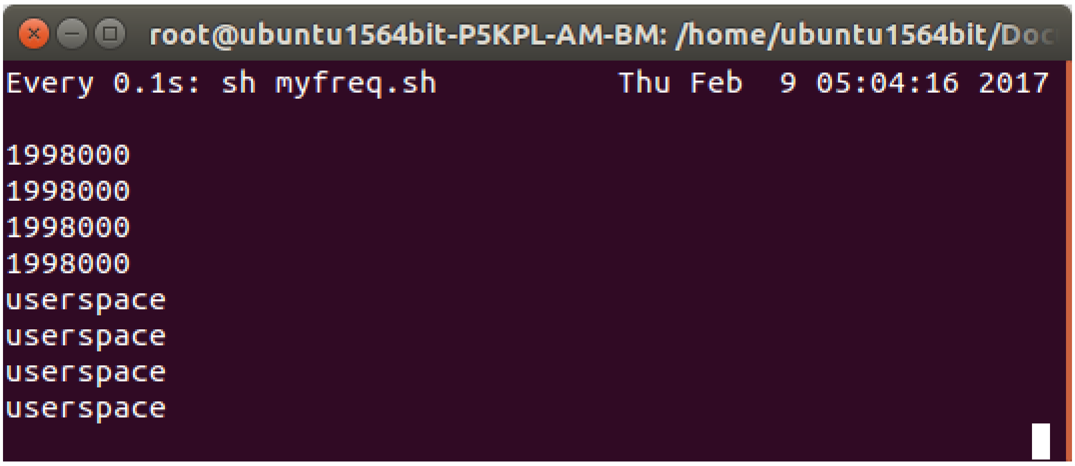
```
cat /XXX/XXX/.../cpu0/....frequncy
cat /XXX/XXX/.../cpu1/....frequncy
cat /XXX/XXX/.../cpu2/....frequncy
cat /XXX/XXX/.../cpu3/....frequncy


cat /XXX/XXX/.../cpu0/....governor
cat /XXX/XXX/.../cpu1/....governor
cat /XXX/XXX/.../cpu2/....governor
cat /XXX/XXX/.../cpu3/....governor
exit 0
```

Using “echo” to change the value in the file



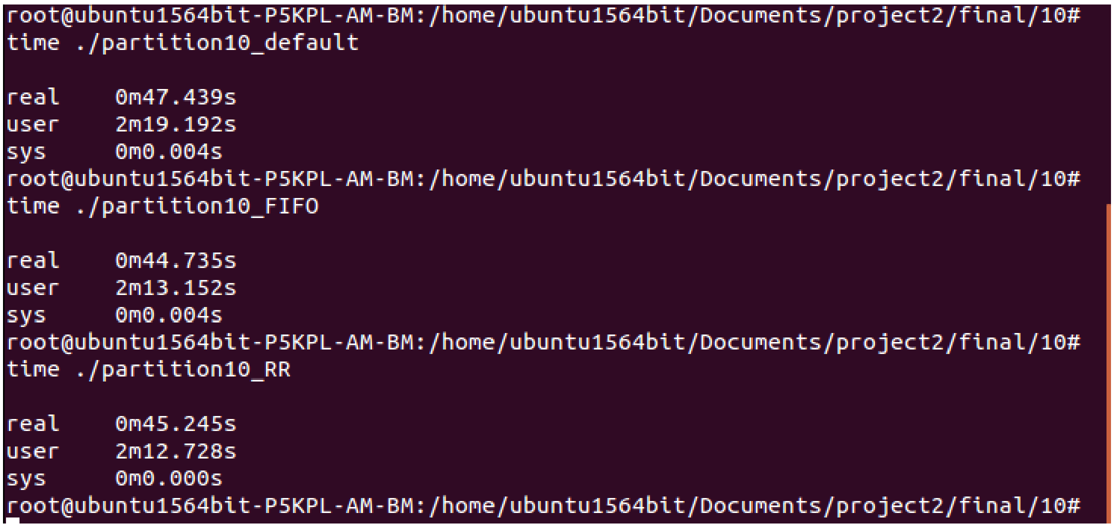
```
echo ???? >> /XXX/XXX/.../cpu0/....frequncy
echo ???? >> /XXX/XXX/.../cpu1/....frequncy
echo ???? >> /XXX/XXX/.../cpu2/....frequncy
echo ???? >> /XXX/XXX/.../cpu3/....frequncy


echo ???? >> /XXX/XXX/.../cpu0/....governor
echo ???? >> /XXX/XXX/.../cpu1/....governor
echo ???? >> /XXX/XXX/.../cpu2/....governor
echo ???? >> /XXX/XXX/.../cpu3/....governor
exit 0
```

Use #watch –n 0 sh XXXX.sh to run the scripts continuously.

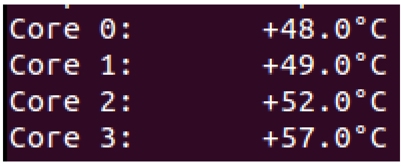


```
Every 0.1s: sh myfreq.sh                 Thu Feb  9 05:04:16 2017

1998000
1998000
1998000
1998000
userspace
userspace
userspace
userspace
```

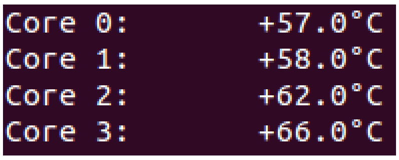
Low Frequency Setting:



```
root@ubuntu1564bit-P5KPL-AM-BM:/home/ubuntu1564bit/Documents/project2/final/10#
time ./partition10_default

real    0m47.439s
user    2m19.192s
sys     0m0.004s
root@ubuntu1564bit-P5KPL-AM-BM:/home/ubuntu1564bit/Documents/project2/final/10#
time ./partition10_FIFO

real    0m44.735s
user    2m13.152s
sys     0m0.004s
root@ubuntu1564bit-P5KPL-AM-BM:/home/ubuntu1564bit/Documents/project2/final/10#
time ./partition10_RR

real    0m45.245s
user    2m12.728s
sys     0m0.000s
root@ubuntu1564bit-P5KPL-AM-BM:/home/ubuntu1564bit/Documents/project2/final/10#
```

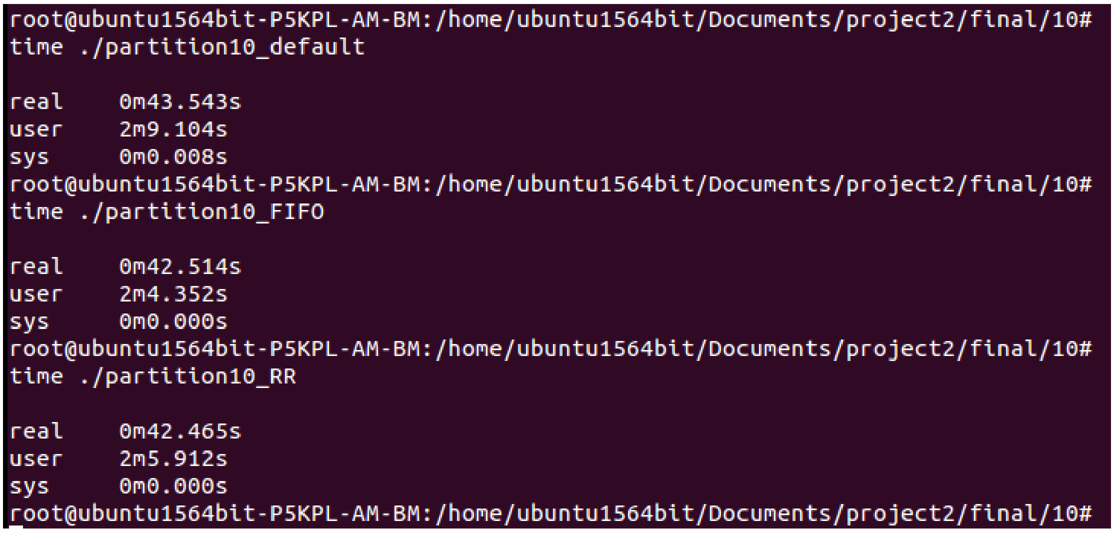


```
Core 0:         +48.0°C
Core 1:         +49.0°C
Core 2:         +52.0°C
Core 3:         +57.0°C
```

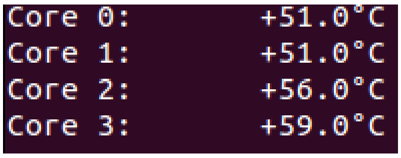


```
Core 0:         +57.0°C
Core 1:         +58.0°C
Core 2:         +62.0°C
Core 3:         +66.0°C
```

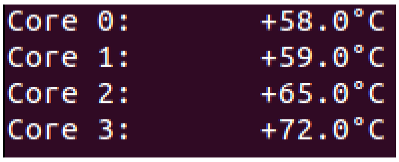
High Frequency Setting:



```
root@ubuntu1564bit-P5KPL-AM-BM:/home/ubuntu1564bit/Documents/project2/final/10#
time ./partition10_default

real	0m43.543s
user	2m9.104s
sys	0m0.008s
root@ubuntu1564bit-P5KPL-AM-BM:/home/ubuntu1564bit/Documents/project2/final/10#
time ./partition10_FIFO

real	0m42.514s
user	2m4.352s
sys	0m0.000s
root@ubuntu1564bit-P5KPL-AM-BM:/home/ubuntu1564bit/Documents/project2/final/10#
time ./partition10_RR

real	0m42.465s
user	2m5.912s
sys	0m0.000s
root@ubuntu1564bit-P5KPL-AM-BM:/home/ubuntu1564bit/Documents/project2/final/10#
```



```
Core 0:         +51.0°C
Core 1:         +51.0°C
Core 2:         +56.0°C
Core 3:         +59.0°C
```



```
Core 0:         +58.0°C
Core 1:         +59.0°C
Core 2:         +65.0°C
Core 3:         +72.0°C
```

Crediting :

The number of created threads must be twice as the number of CPUs, and the number of CPUs should be more than 3.

[Scheduler Implementation. 40%]

- Describe how to implement the scheduler setting (FIFO, RR, Default) **20%**
- Show the scheduling states of tasks **20%**

[Temperature Watch & Frequency Scaling. 40%]

- Show that your computer's frequency levels and write the scripts 20%
- Show that CPU's temperature in different frequency setting. 20%

[Result. 20%]

- Compare the response time of the program in three scheduler setting (FIFO, RR, Default) **10%**
- Analyze the performance of three scheduler setting (FIFO, RR, Default) and in different frequency settings. **10%**

Project submit

Submit deadline: 09 :00, May. 12, 2017

Submission : Moodle

File name format : ESSD_Student ID_HW2.rar

※ Strictly prohibited copying !

ESSD _Student ID_HW2.rar must inculde the report and source code.

嚴禁抄襲，發生該類似情況者，一律以零分計算