Term Project 1:

Making Your Own System Call for `sysps`

Due: 2nd Apr. (Mon) ~ 15th Apr. (Sun), 11:59 PM
Submit on i-Campus

1. Project Introduction

In this project, we will see how processes are managed in Linux and make a system call to log the information of the current running processes. Your task for this project is to make your own system call to enable the given application **`sysps`** to show the information of the current running processes.

2. Implementations

2.0 Environment

- The project should be performed in Ubuntu environment. (Kernel Version 4.9.x)
- You can install Ubuntu or use a virtual machine to carry out your project.
- If you use a virtual machine, make sure to allocate enough memory to the machine (at least 2 GB).

2.1 Core Function Analysis

- Download the Linux kernel from https://www.kernel.org/

```
$ wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.15.14.tar.xz
```

- Use **`ftrace`** to find the functions that create processes.
- You will have to describe its' functionalities later in your report.
  (You need not explain the functions entirely. We will only check to see that you have properly understood how they work.)

2.2 Output

The user application `sysps` will be provided to print the following.

- UID: user ID (root, $USER)
- PID: process ID
- PPID: process ID of the parent process
- NICE: nice value of the current process (-20 ~ 19)
- STATE: current state of the process (S: sleeping, R: running, Z: zombie)
- CPU#: CPU # of where the process is scheduled
- CMD: the executable name of the process
- nr_switches: # of context switches the processes made
- nr_running: # of running processes in the runqueue where the process is in
- list of siblings: information of the process's siblings
- list of children: information of the process's children

- range of VMA: the start and end addresses of each virtual memory address (CODE, DATA, HEAP, STACK*)
- number of VMA: the number of the virtual memory address

Output Examples

`$ ./sysps` **: print [UID] [PID] [PPID] [NICE] [STATE] [CPU#] [CMD]**

```
hayun@ubuntu:~$ ./sysps
PS:
PS:   UID    PID  PPID NICE STATE CPU# CMD
PS:     0      1     0    0     S    0 systemd
PS:     0      2     0    0     S    0 kthreadd
PS:     0      3     2    0     I    0 kworker/0:0
PS:     0      4     2  -20     I    0 kworker/0:0H
PS:     0      5     2    0     I    0 kworker/u256:0
PS:     0      6     2  -20     I    0 mm_percpu_wq
PS:     0      7     2    0     S    0 ksoftirqd/0
PS:     0      8     2    0     I    0 rcu_sched
PS:     0      9     2    0     I    0 rcu_bh
PS:     0     10     2    0     S    0 migration/0
PS:     0     11     2    0     S    0 watchdog/0
PS:     0     12     2    0     S    0 cpuhp/0
```

`$ ./sysps -C` **: print [CPU#] [nr_switches] [nr_running]**

```
hayun@ubuntu:~$ ./sysps -C
PS:
PS: CPU# nr_switches nr_running
PS:    0       27334          1
PS:    1       20731          0
PS:    2       21414          0
PS:    3       30303          0
```

* STACK may only have the start address

`$ ./sysps -s -p 1195` **: print the siblings of process 1195**

```
hayun@ubuntu:~$ ./sysps -c -p 1195
PS:
PS: Print Children of 1195
PS:   UID   PID  PPID NICE STATE CPU# CMD
PS:   108  1211  1195    0     S    3 indicator-messa
PS:   108  1212  1195    0     S    2 indicator-bluet
PS:   108  1213  1195    0     S    0 indicator-power
PS:   108  1214  1195    0     S    2 indicator-datet
PS:   108  1215  1195    0     S    3 indicator-keybo
PS:   108  1216  1195    0     S    3 indicator-sound
PS:   108  1217  1195    0     S    0 indicator-sessi
PS:   108  1231  1195    0     S    2 indicator-appli
PS:   108  1274  1195  -11     S    1 pulseaudio
```

`$ ./sysps -c -p 1195` **: print the children of process 1195**

```
hayun@ubuntu:~$ ./sysps -s -p 1195
PS:
PS: Print Sibling of 1195
PS:   UID   PID  PPID NICE STATE CPU# CMD
PS:     0   347     1    0     S    1 systemd-journal
PS:     0   363     1    0     S    2 vmware-vmblock-
PS:     0   368     1    0     S    1 systemd-udevd
PS:   100   477     1    0     S    0 systemd-timesyn
PS:     0   801     1    0     S    1 cron
PS:     0   805     1    0     S    1 systemd-logind
PS:   111   812     1    0     S    3 avahi-daemon
PS:   104   813     1    0     S    2 rsyslogd
PS:   106   815     1    0     S    0 dbus-daemon
PS:     0   830     1    0     S    3 vmtoolsd
PS:     0   836     1    0     S    0 acpid
PS:     0   839     1    0     S    3 NetworkManager
PS:     0   841     1    0     S    1 cupsd
```

`$ ./sysps -v` **: print [PID] [CODE] [DATA] [HEAP] [STACK] [# of VMA]**

```
hayun@ubuntu:~$ ./sysps -v
PS:
PS:   PID                    CODE                    DATA                    HEAP          STACK #VMA
PS:     1 0x558ef8035000-0x558ef8190ee0 0x558ef8192c80-0x558ef81b66e8 0x558ef97be000-0x558ef9902000 0x7ffe7e0377c0-   85
PS:   338 0x55bf791a9000-0x55bf791f59d8 0x55bf791f7740-0x55bf791f9078 0x55bf79acd000-0x55bf79b0a000 0x7ffcc94dca30-   63
PS:   357 0x000000400000-0x0000004027cc 0x000000602de8-0x0000006032c0 0x000000baa000-0x000000bcb000 0x7ffe3fc3dc00-   38
PS:   369 0x557a378d0000-0x557a37933be0 0x557a379352e0-0x557a3793f09c 0x557a392d3000-0x557a3948b000 0x7ffd98a85430-   85
PS:   450 0x557f2a10f000-0x557f2a12fa08 0x557f2a1313a0-0x557f2a132078 0x557f2a3cb000-0x557f2a3ec000 0x7ffc49096740-   69
PS:   791 0x559b27a12000-0x559b27a4646c 0x559b27c46d48-0x559b27c48350 0x559b28812000-0x559b288d6000 0x7fff64d0e4e0-   95
PS:   795 0x55aacb360000-0x55aacb3c2b74 0x55aacb5c3128-0x55aacb5c520c 0x55aaccd97000-0x55aacce1f000 0x7ffe7fd74010-  179
PS:   798 0x000000400000-0x000000488054 0x000000688868-0x000000691c80 0x000000d90000-0x000000dcd000 0x7ffe4071f620-   89
PS:   805 0x000000400000-0x0000006bdb7c 0x0000008be450-0x0000008d0900 0x000001534000-0x00000165a000 0x7ffe295759c0-  260
PS:   829 0x000000400000-0x00000040abe4 0x00000060ae10-0x00000060b378 0x000000feb000-0x00000100c000 0x7fff73481150-   18
PS:   831 0x5638dce9d000-0x5638dcf2d368 0x5638dcf2ef80-0x5638dcf345f0 0x5638ddc7a000-0x5638ddcb7000 0x7ffe66034940-   65
PS:   833 0x000000400000-0x00000041d06c 0x00000061dd90-0x00000061e950 0x0000009e4000-0x000000a05000 0x7ffc86ffd270-   98
```

Your task is to make a **system call** to implement the functionalities shown above.

2.3 Tips

- Do NOT modify the given `sysps.c` file
- Compile `sysps.c` with the following command.

```
$ gcc -o sysps sysps.c
```

- The information of a process is managed by the **`task_struct`** structure in Linux.
- The runqueue is managed by the **`rq`** structure in Linux.
- You can use the information from the `/proc` filesystem. If `/proc` does not provide you with the information you inquire, it is possible to retrieve the information from the structures in the kernel.
- **sysps** is a user-level application. Make sure to **make your own system call** to retrieve information from the kernel (use the return values of system calls). To do so, you must find a way to create and use your own system call.
- You may refer to the **ps** application in the open-source project *PROCPS* (https://gitlab.com/procps-ng/procps). It shows the information about the current running processes via the `/proc` filesystem.

```
$ git clone https://gitlab.com/procps-ng/procps
```

Build a binary executable of `ps`. Check the markdown files (e.g., INSTALL.md) to figure out how to build the project code. For this, you will need to install several dependent libraries.

3. Evaluation Policy

- Successfully made your own system call **(5 points)**
- The `sysps` program outputs normal information of all processes **(15 points)**
- The `sysps` program outputs normal information of all CPUs **(15 points)**
- The `sysps` program outputs normal information of all sibling processes **(7.5 points)**
- The `sysps` program outputs normal information of all children processes **(7.5 points)**
- The `sysps` program outputs VMA information of the given process **(15 points)**
- Report (35 points)
  - Analysis of the function path (see section 2.1) using ftrace **(20 points)**
  - Detailed explanation of your implementations (e.g., `task_struct`, `rq`, `mm_struct`) **(15 points)**

Task	System Call	Processes	CPUs	Siblings	Children	VMA	Report		**Total**
							Analysis	Implementation	
Points	5	15	15	7.5	7.5	15	20	15	**100**

4. Submission Guidelines

- Submit a compressed zip file named **[your_student_ID]_proj1.zip**.
  The zip file should contain [your_student_ID]_report.pdf and several codes that you have modified (i.e., kernel codes).
- Do NOT attach the whole kernel code, just the modified ones (i.e., the files you have modified to implement your system calls).

- Please describe the key parts (modified code) of your code explicitly in your report.
  Do NOT attach the whole source code in the report, just the key parts.
- Submit by uploading your zip file on i-Campus.
  If you have trouble uploading it on i-Campus, send it to tk1star2@nate.com. (The title should be **[SWE3004-41] Term project 1, [your_student_id], [your_name]**)

5. Notice

- Your term project must be your original work.
- If you have any questions, describe your problem along with a screenshot and send it to tk1star2@nate.com, or visit 85465 (Research & Business Center).
  (Please include **[SWE3004-41]** in the title.)
- In the case of plagiarism, you will **FAIL** this course.
  (If two similar source codes or reports are found, both students will fail this course.)
- 15 points will be deducted per day in case of late submission.

Have fun!