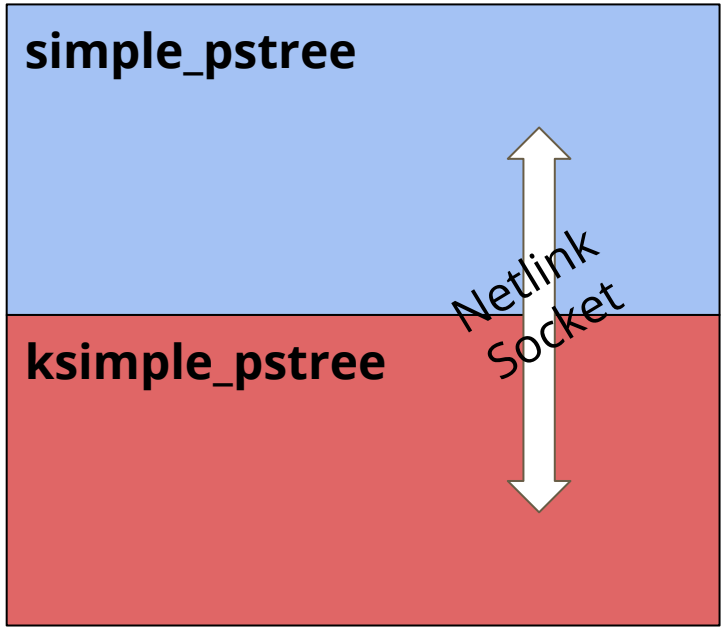
2018 OS Homework 1
Simple Pstree

Due day 11/01 23:59

Objectives

- Understand how to transfer information between the kernel and the user space processes
- Understand the PCB in the kernel and its related macros
- Understand the linked list in the kernel

Overview



1. simple_pstree packages the option and PID into a message and sends it to ksimple_pstree via the socket

2. ksimple_pstree packages the tree of processes into a message and sends it to simple_pstree via the socket

3. simple_pstree displays the received message

Requirement - application

- **simple_pstree [-c|-s|-p][*pid*]**
  - -c : **Display the entire process tree which is spawned by a process**
    - The tree of processes is rooted at **either *pid* or init if *pid* is omitted**
    - This is the **default option**
  - -s : **Display all siblings of a process**
    - The searching PID of the process can be **either *pid* or simple_pstree if *pid* is omitted**
  - -p : **Display all ancestors of a process**
    - The searching PID of the process can be **either *pid* or simple_pstree if *pid* is omitted**
- **The application must send the option and PID to the module by the netlink socket**
- **The application must wait for the module to send the result back and print it out**

Requirement - module

- **After receiving the application's message, the module starts to process the requirement of the application and sends the process tree required by application back to the application by the netlink socket**
- Using ***find_get_pid()*** and ***pid_task()*** in <linux/pid.h> header to find the *struct task_struct*
- Using ***children***, ***sibling***, and ***parent*** fields in the *struct task_struct* to complete the requirement of the process tree for each option
- Using ***list_for_each()*** and ***list_entry()*** in <linux/list.h> header to operate the *struct list_head*
- Each task is represented in the format "***process_name(pid)***"
  - The *process_name* and *pid* are read from the **comm** and **pid** fields in the *struct task_struct* respectively

Requirement - netlink socket

- **You must use the netlink socket to transfer the message between application and module**
  - Application
    - *netlink_socket* = ***socket(AF_NETLINK, SOCK_RAW, ...)***;
  - Module
    - *netlink_socket* = ***netlink_kernel_create(...)***;
- Message format
  - From application to module
    - "***x pid***", where *x* indicates the option (i.e., c, s or p) and *pid* indicates the PID to be queried
  - From module to application
    - The format is **the same as the output format**

(see next page)

Requirement - output format

- -c

  process_name_1(pid_1)
  <4 spaces>process_name_2(pid_2)
  <4 spaces><4 spaces>… ← (if process_name_2 has children)
  <4 spaces>process_name_3(pid_3)
  <4 spaces>…

- -s

  sibling_process_name_1(sibling_pid_1)
  sibling_process_name_2(sibling_pid_2)
  sibling_process_name_3(sibling_pid_3)
  …

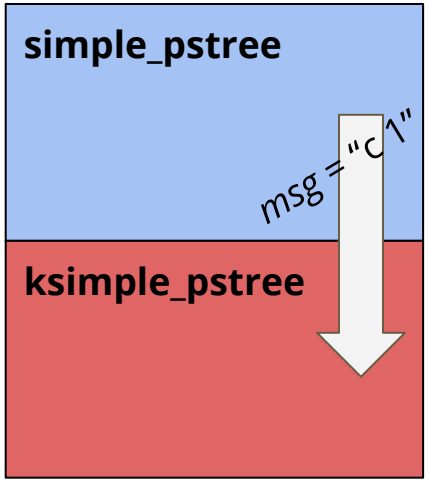
- -p

  process_name_1(pid_1) ← (process_name_2's parent)
  <4 spaces>process_name_2(pid_2) ← (process_name_3's parent)
  <4 spaces><4 spaces>process_name_3(pid_3)
  …

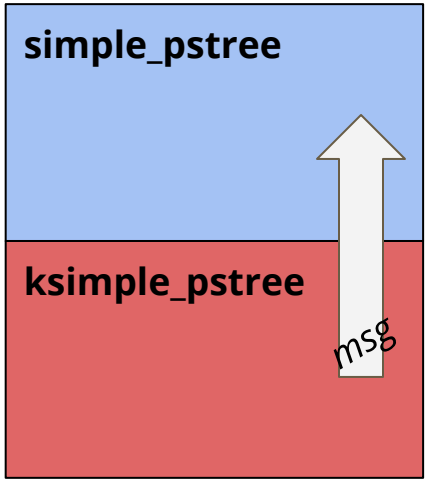
Example

1 $ simple_pstree -c1

2



3



msg = "systemd(1)
ModemManager(1003)
NetworkManager(1250)
ibus-daemon(10514)
ibus-dconf(10518)
ibus-engine-sim(10539)
ibus-ui-gtk3(10520)"

4 Display the *msg* in terminal

Example - default option

```
pohao@linux-5qc1  ~/workplace/simple-pstree  master v1.1  ./simple-pstree
systemd(1)
    systemd-journal(481)
    haveged(487)
    lvmetad(491)
    systemd-udevd(499)
    auditd(897)
    cupsd(991)
    dbus-daemon(997)
    chronyd(1020)
    nscd(1032)
    wickedd-dhcp6(1095)
    wickedd-dhcp4(1096)
    wickedd-auto4(1097)
    firewalld(1098)
    ModemManager(1099)
    irqbalance(1100)
    avahi-daemon(1102)
    smartd(1103)
    rsyslogd(1105)
    mcelog(1106)
    systemd-logind(1107)
    wickedd(1174)
    polkitd(1199)
    wickedd-nanny(1200)
    NetworkManager(1240)
```

Example - option c without PID number

```
 pohao@linux-5qc1  ~/workplace/simple-pstree  master v1.1  ./simple-pstree -c
systemd(1)
    systemd-journal(481)
    haveged(487)
    lvmetad(491)
    systemd-udevd(499)
    auditd(897)
    cupsd(991)
    dbus-daemon(997)
    chronyd(1020)
    nscd(1032)
    wickedd-dhcp6(1095)
    wickedd-dhcp4(1096)
    wickedd-auto4(1097)
    firewalld(1098)
    ModemManager(1099)
    irqbalance(1100)
    avahi-daemon(1102)
    smartd(1103)
    rsyslogd(1105)
    mcelog(1106)
    systemd-logind(1107)
    wickedd(1174)
    polkitd(1199)
    wickedd-nanny(1200)
    NetworkManager(1240)
```

Example - option c with PID number

```
pohao@linux-5qc1 ~/workplace/simple-pstree  master v1.1 ./simple-pstree -c2520
plasmashell(2520)
    ksysguardd(2838)
    firefox(7867)
        kmozillahelper(8026)
        Web Content(7940)
        Web Content(8047)
        Web Content(4197)
        Web Content(18606)
    dolphin(15577)
    konsole(19671)
        zsh(19676)
            simple-pstree(20878)
```

Example - option s without PID number

Example - option s with PID number

```
 pohao@linux-5qc1  ~/workplace/simple-pstree  master v1.1  ./simple-pstree -s2838
firefox(7867)
konsole(18052)
dolphin(15577)
```

Example - option p without PID number

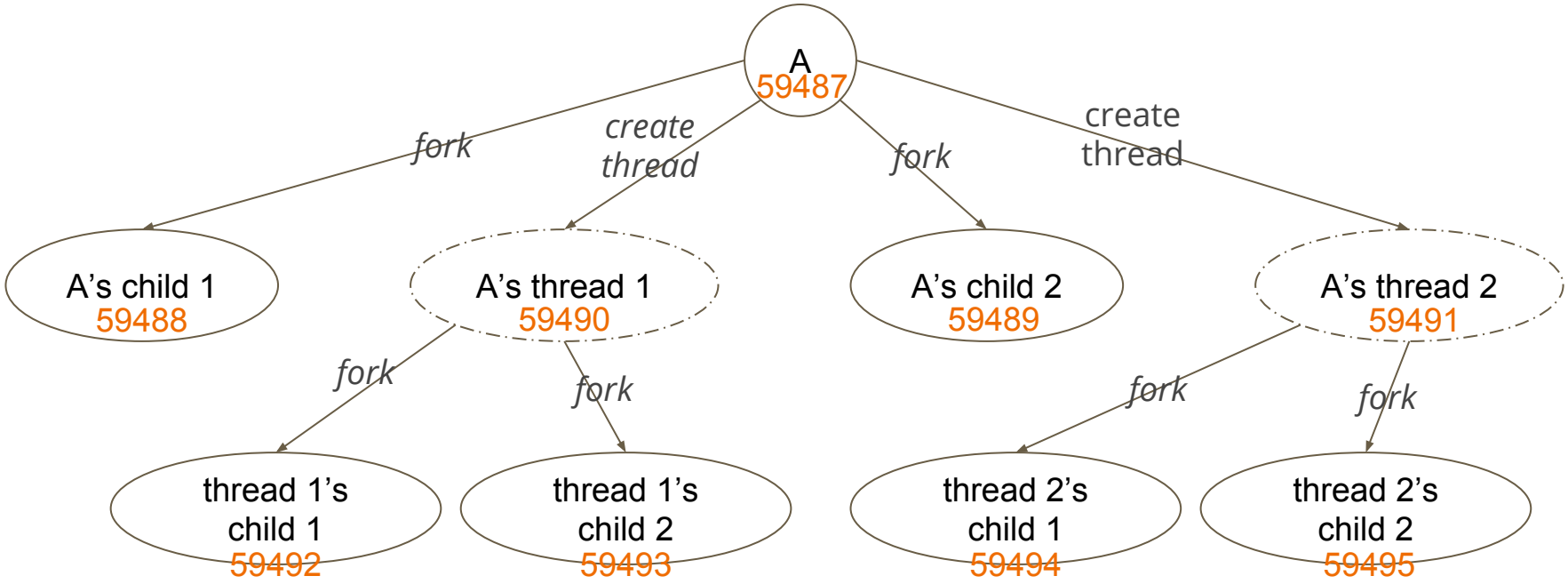
```
 pohao@linux-5qc1  ~/workplace/simple-pstree  master v1.1  ./simple-pstree -p
systemd(1)
    plasmashell(2520)
        konsole(18052)
            zsh(18056)
                simple-pstree(16665)
```

Example - option p with PID number

```
pohao@linux-5qc1 ~/workplace/simple-pstree  master v1.1 ./simple-pstree -p2766
systemd(1)
    kdeinit5(2461)
        akregator(2600)
            QtWebEngineProc(2655)
                QtWebEngineProc(2663)
                    QtWebEngineProc(2766)
```

Example - PID number to be queried does not exist

Bonus - scenario



Bonus

- How to handle the following questions?
  - **When a process creates threads to generate children, we can't find those children through the *children* field of *struct task_struct***
    - When we look for the descendant of process A (59487), we can only find 59488 and 59489, but 59492～59495 are also children of process A
  - **When a process not only generates children by itself, but also generates children through thread, we can't find all siblings from the *sibling* field in a child's *struct task_struct***
    - When we look for the sibling of 59488, we can only find 59489, but 59492～59495 are also the sibling of 59488
  - **When a child is generated by the thread of the process, the *parent* field in the child's *struct task_struct* is pointing to the thread, not the parent process**
    - The parent of the process with PID 59492 is the thread with PID 59490, instead of the process with PID 59487
  - **You can only solve the above problem by using the field in the *struct task_struct***

Reference Materials

- Kernel Space, User Space Interfaces
- NETLINK(7)
- Kernel Korner - Why and How to Use Netlink Socket
- Linux Kernel Development (chapter 3)
- Linux Device Drivers (chapter 2)
- struct task_struct