

CMPT-300

Assignment 4

Maoshun Shang
301280479

Please Indicate what components you have completed(Yes,Partial,No):

*(For any partial Completion Please give description in report)

**(Mark penalties will be assessed for claiming a part is present in code when it is not!)

Part A(25%):

Setup Environment(Yes,Partial,No):: Yes

Compile Kernel: Yes

Modified: Kernel: Yes

Part B(75%):

Kernel Space:

Error Handling: Yes (when info is null)

Basic Process Info(Info on input PID): Yes

Linked List Traversal(Relative PID Info): Yes

User-Space:

Testing Code: Yes

General Question:

Briefly Describe What you have learned in this assignment:

1. All the includes are from “include” folder
2. Linux kernel use printk() instead of printf()
3. Linux use doubly linked list to form a list of processes
4. EINVAL returns 22 if the input is null
5. All the field of prinfo does not change except cutime and cstime
6. cutime changes not as frequently as cstime

Part A: Briefly answer these questions(You only need a few sentences each).

1) What is QEMU and how did you use it in this assignment?

QEMU is an emulator, in this assignment, it is used to emulate the Linux kernel. It allows user to modify the kernel without crashing host OS if errors occurs.

2) What is contained in arch/x86/kernel/syscall_table_32.S. How does the operating system use the syscall table?

This table maintains a list of system calls. The operating system looks the system call table when a system call is about to performed in user space.

3) How do you directly call a syscall from user-space?

Include <sys/syscall> library, and define the system call before main function. Then use syscall() function in main function.

4) Run your VM with the modified kernel and run your user-space testing code. Attach a screen shot to this report(The part where we see the diving to kernel level and the rising to user level).

```
Terminal
File Edit View Search Terminal Help

cmpt300 login: cmpt300
Password:
Last login: Fri Apr  7 18:24:33 2017 on ttyS0
Linux cmpt300 2.6.26.5 #127 SMP Fri Apr  7 18:10:32 PDT 2017 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
cmpt300@cmpt300:~$ cd cmpt300/
cmpt300@cmpt300:~/cmpt300$ ls
cmpt300_test.c  prinfo  prinfo.c  prinfo.h
cmpt300@cmpt300:~/cmpt300$ gcc cmpt300_test.c -o cmpt300_test
cmpt300@cmpt300:~/cmpt300$ ./cmpt300_test

Diving to kernel level

Hello World !--syscall arg 300
Rising to user level

cmpt300@cmpt300:~/cmpt300$
```

Part B: Briefly answer these questions(You only need a few sentences each).

1) What is the purpose of a process “nice” value, and how did you find it?

Nice value is a number indicate the priority of the process. It is found by “proc” get the current process and assign its nice value to “info” by using task_nice() function.

2)

a) What data-structure is used in kernel space to traverse between related PIDs information?

Doubly linked list.

b) How did you traverse the data-structure?

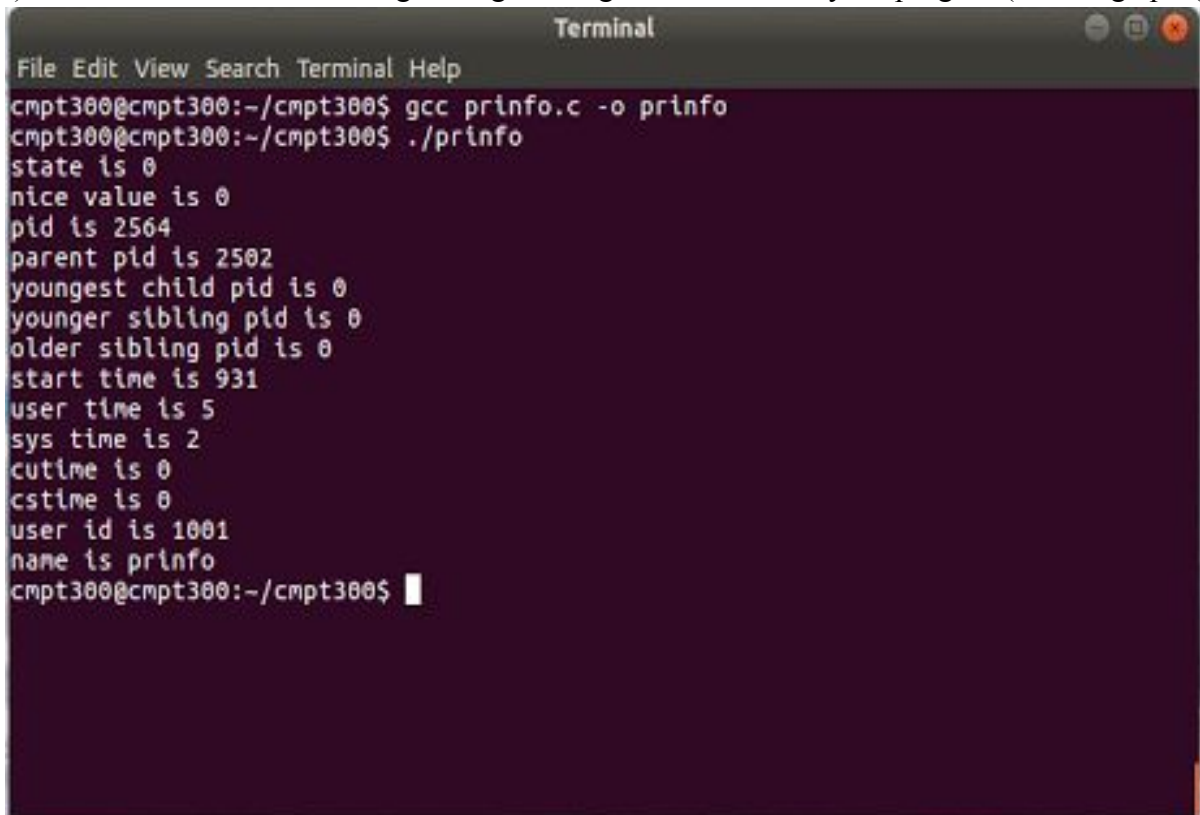
Use list_for_each_entry() to function like a loop until the younger/older sibling or youngest children is found.

3) How do you pass the target PID from user-space to kernel space?

Use `copy_from_user()` to send information from user to kernel.

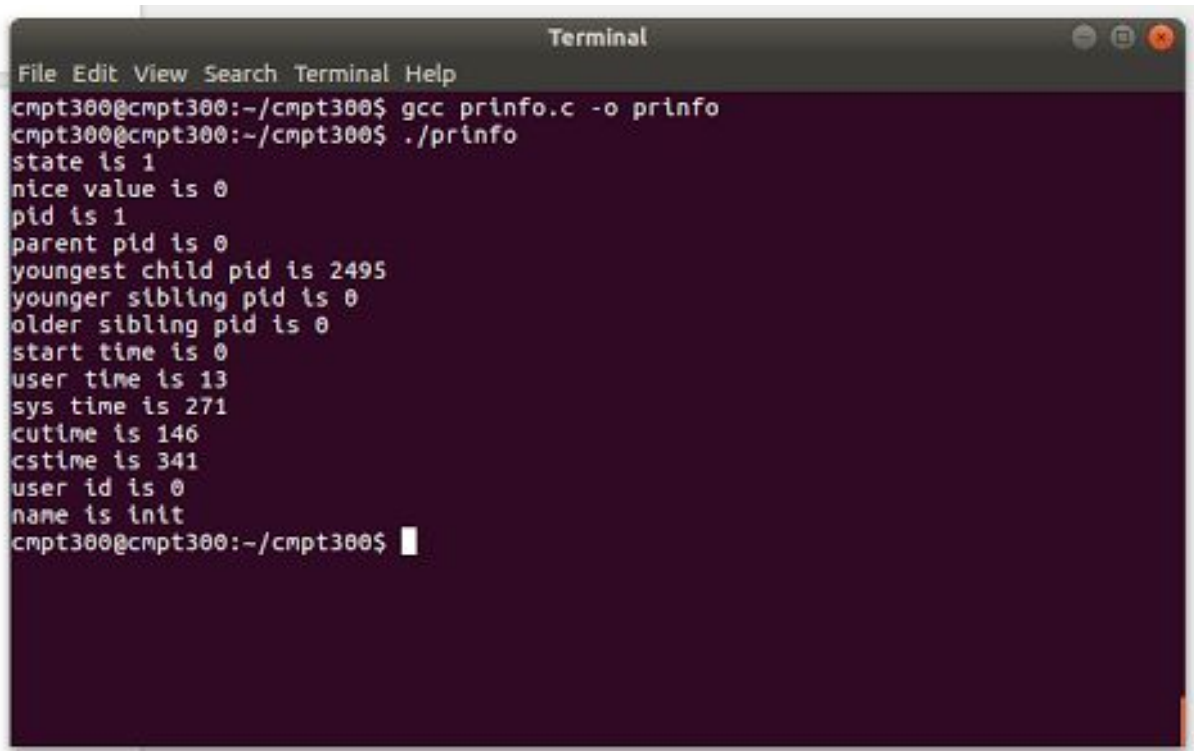
4) Attach three screen shots of your user-space testing code being run.

a) One screen shot of the testing coding running with the PID of your program(ex. use `getpid()`).

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is "cmpt300@cmpt300:~/cmpt300\$". The user enters "gcc prinfo.c -o prinfo". The prompt changes to "cmpt300@cmpt300:~/cmpt300\$". The user enters "./prinfo". The program outputs the following information: "state is 0", "nice value is 0", "pid is 2564", "parent pid is 2502", "youngest child pid is 0", "younger sibling pid is 0", "older sibling pid is 0", "start time is 931", "user time is 5", "sys time is 2", "cutime is 0", "cstime is 0", "user id is 1001", "name is prinfo". The prompt returns to "cmpt300@cmpt300:~/cmpt300\$".

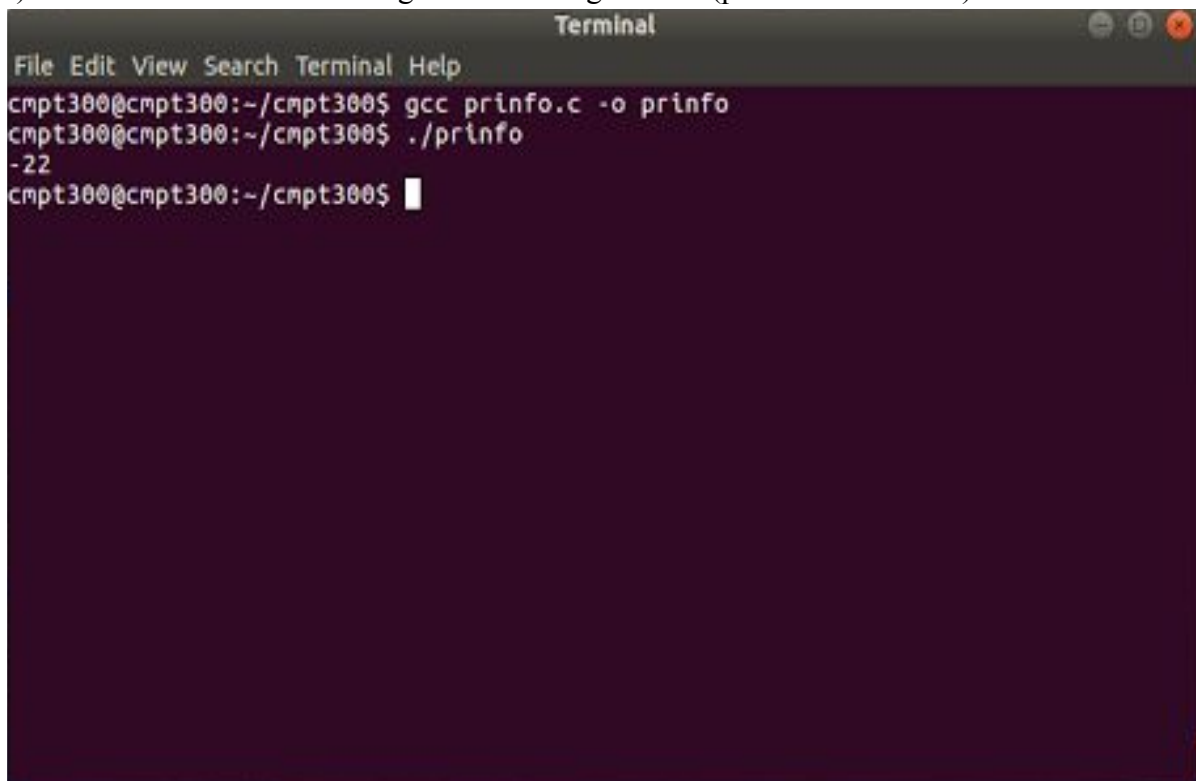
```
cmpt300@cmpt300:~/cmpt300$ gcc prinfo.c -o prinfo
cmpt300@cmpt300:~/cmpt300$ ./prinfo
state is 0
nice value is 0
pid is 2564
parent pid is 2502
youngest child pid is 0
younger sibling pid is 0
older sibling pid is 0
start time is 931
user time is 5
sys time is 2
cutime is 0
cstime is 0
user id is 1001
name is prinfo
cmpt300@cmpt300:~/cmpt300$
```

b) One screen shot of the testing coding using the PID of init (pid = 1).

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is "cmpt300@cmpt300:~/cmpt300\$". The user enters "gcc prinfo.c -o prinfo" and then "./prinfo". The program outputs the following information: state is 1, nice value is 0, pid is 1, parent pid is 0, youngest child pid is 2495, younger sibling pid is 0, older sibling pid is 0, start time is 0, user time is 13, sys time is 271, cutime is 146, cstime is 341, user id is 0, name is init. The prompt returns to "cmpt300@cmpt300:~/cmpt300\$".

```
cmpt300@cmpt300:~/cmpt300$ gcc prinfo.c -o prinfo
cmpt300@cmpt300:~/cmpt300$ ./prinfo
state is 1
nice value is 0
pid is 1
parent pid is 0
youngest child pid is 2495
younger sibling pid is 0
older sibling pid is 0
start time is 0
user time is 13
sys time is 271
cutime is 146
cstime is 341
user id is 0
name is init
cmpt300@cmpt300:~/cmpt300$
```

c) One screen shot of the testing code handling an error(prinfo struct = null).

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is "cmpt300@cmpt300:~/cmpt300\$". The user enters "gcc prinfo.c -o prinfo" and then "./prinfo". The program outputs "-22". The prompt returns to "cmpt300@cmpt300:~/cmpt300\$".

```
cmpt300@cmpt300:~/cmpt300$ gcc prinfo.c -o prinfo
cmpt300@cmpt300:~/cmpt300$ ./prinfo
-22
cmpt300@cmpt300:~/cmpt300$
```