

CMPT-300

Assignment 4

Name
ID

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

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:

I learned that all processes in the OS is stored in a doubly linked list hierarchy structure and it is very easy to traverse through all processes. I also learned that one cannot just pass pointer to/from user space and kernel space directly. I spent a lot of time looking up `copy_from_user()` and `copy_to_user()`.

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 a open source virtual machine (/emulator).

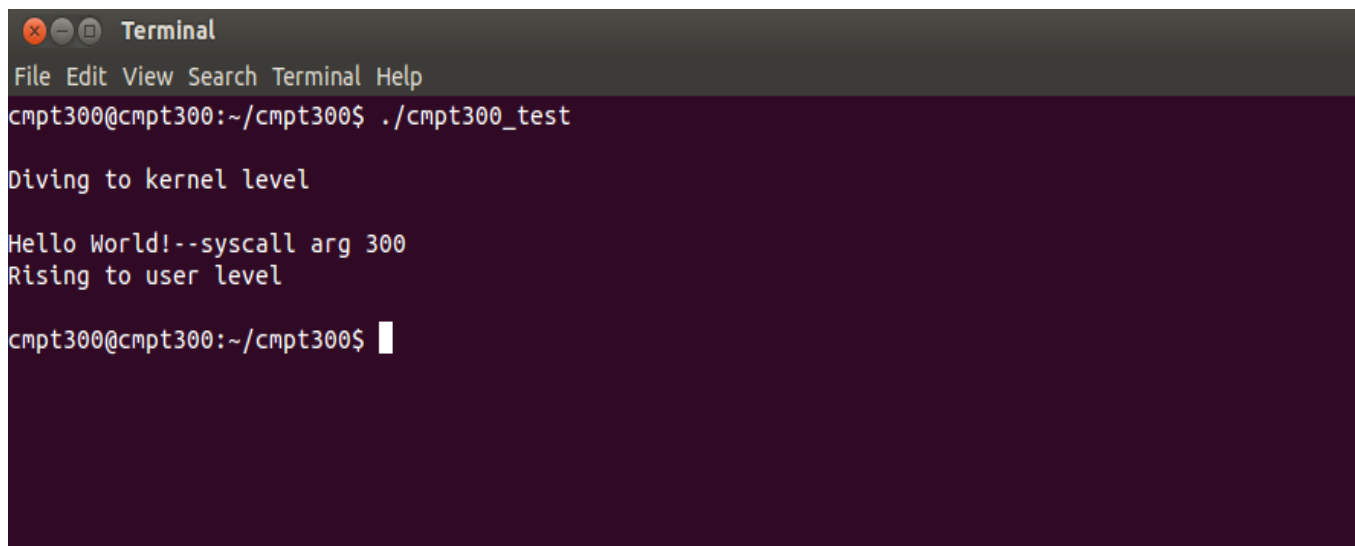
2) What is contained in arch/x86/kernel/syscall_table_32.S. How does the operating system use the syscall table?

It adds a pointer to the function we defined in cmpt300/cmpt300_test.c. Inside QEMU we can invoke `syscall(327,(value/pointer))` to call our implemented function.

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

`syscall(syscall_number, (value/pointer));`

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@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” is an priority number of CPU time. The higher nice value is the more CPU time its process gets. In kernel, priority ranges from 0 to 139 and nice ranges from -20 to 19 so we can use the current pointer to get the values with some subtraction after. The nice value is default to be 0.

`(current->prio) - 120`

2)

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

`list_for_each(pos, head)`

b) How did you traverse the data-structure?

Iterate all the process using `list_for_each`, then inside the loop use “`list_entry`” to extract any data

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

Using “`copy_from_user`” to pass any information in a pointer from user space to kernel space

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()`).

```

state = 0
nice = 0
pid = 2510
parent_pid = 2489
uid = 1001
youngest_child_pid = 2510
younger_sibling_pid = 0
older_sibling_pid = 0
start_time = 131949571
user_time = 6
sys_time = 1
cutime = 0
cstime = 0
uid = 1001
comm = cmpt300_test

```

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

```

state = 1
nice = 0
pid = 1
parent_pid = 0
uid = 0
youngest_child_pid = -1014858944
younger_sibling_pid = 2
older_sibling_pid = 0
start_time = 196000249
user_time = 14
sys_time = 206
cutime = 293
cstime = 0
uid = 0
comm = init

```

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

```

-----
state = 0
nice = 200
pid = 0
parent_pid = 0
uid = 0
youngest_child_pid = 0
younger_sibling_pid = 0
older_sibling_pid = 0
start_time = 0
user_time = 0
sys_time = 0
cutime = 0
cstime = 0
uid = 0
comm =

```

Note: the system call returns -22 and these are all printf statements in user space