New York University
Tandon School of Engineering
Department of Electrical & Computer Engineering

Introduction to Operating Systems (CS-GY6233)
Fall 2021

Assignment 3
(15 points)

Part 1 (2 points)

If you create a main() routine that calls fork() three times, i.e. if it includes the following code:

```
pid_t pid1=33, pid2, pid3;
pid1 = fork();
if(pid1>0) pid2 = fork();
if(pid2>0) pid3 = fork();
```

Draw a process tree similar to that on slide **22** of lecture 4, clearly indicating the values of pid1,pid2 and pid3 for each process in the tree (i.e. whether 0, smaller than 0 or larger than 0).

Note that the process tree should only have one node for each process and thus the number of nodes should be equal to the number of processes.

The process tree should be a snapshot just after all forks completed but before any process exists.

Each line/arrow in the process tree diagram shall represent a creation of a process, or alternatively a parent/child relationship.

Part 2 (13 points)

In this assignment, we shall build on what we did in assignment 2 and further extend it. You shall write a dynamically loadable module that allows us to get the time since system boot from a user-mode applications. This time should be recorded when your module is inserted with insmod (i.e. computed in the init function, just as in assignment #2).

Of course the "libc" library contains API functions that allows us to obtain the time since boot, but we are going to implement our own version instead, that's the goal of this assignment.

When reading from the module using the following command:

>> cat /dev/lab3

it shall return:

"Hello world, the time since boot ss.mmm seconds".

Your module should:

a) Use the `misc_register()` functionality to register itself as a character device. This requires that you create a structure (`struct file_operations`) containing the function pointers (open, release, read, etc.) and pass it to `misc_register()`.
b) Define the open(), release() and read() functions. Make sure you populate their function pointers in `struct file_operations`.
c) you may test your code by either writing your own user-mode program that reads from your device (useful for debugging) or by using:
> cat /dev/lab3
Hello world, the time now is 12:47:54
(// that's if you named your device lab3 during misc_reg())

Some Linux Notes:

- Devices (or more correctly device drivers) are organized as character, block or network devices.
- Devices have major and minor numbers. The major number specifies the device type whereas the minor number specifies an instance of that type. As such, it is common that a single device driver takes care of an entire major number.
- Most major numbers are already assigned to specific device drivers so you are not supposed to use them.
- Linux uses the concept of virtual file systems, one of those is the devfs.
    - Generally, when you register your device driver (i.e. your module) with the devfs using a major and a minor number,
    - You can then create a node for it in the /dev directory (using the mknod command), e.g. /dev/lab3
    - You can then treat it as a file and file operations such as fopen(), fprint, fread, etc. are routed to functions that your module handles.
- Instead of using this complex registration and node creation process, we shall use the `misc_register()`, which handles both steps automatically. It registers us under device major number 10 (amongst with many other devices besides us using the same major number but different minor numbers).
    - You need to provide a minor number to `misc_register()`, the device's name ("lab3") and a pointer to the file operations.
    - In order to avoid collisions, use the `MISC_DYNAMIC_MINOR` macro as the minor number when you register your device.
- For information regarding misc_register(), see: http://www.linuxjournal.com/article/2920.
- You may find further information at kernel.org or lwn.net
- For reasons that shall be explained when we study virtual memory (towards lectures 8-9), you cannot directly use the pointers provided by the read() function in `struct_file_operstions`. Instead you need to use:

```
unsigned long copy_to_user(void __user *to, const void *from, unsigned long n);
```

What to hand in (using NYU Classes):

- Your ".c" and ".h" files (with appropriate comments).
- Your Makefile(s).

- A screen shot of your terminal window showing the current directory, the command used to compile your program, the command used to run your program and the output of your program.

RULES:

- You may consult with other students about GENERAL concepts or methods but copying code (or code fragments) or algorithms to solve your coding assignment is NOT ALLOWED and is considered cheating (whether copied form other students, the internet or any other source).
- If you are having trouble, please ask your teaching assistant for help.
- You must submit your assignment prior to the deadline.