CS492 Lab 2
Adding a new system call

This is an individual assignment. Individual assignments, as the word indicate, are to be done **INDIVIDUALLY**. Any sign of collaboration will result in a 0 and being reported to the Honor Board.

Introduction

The objectives of this project are to familiarize yourself with the kernel system call API and process data structures. Specifically, you will:

- Add two new, simple, system calls;
- Call the system calls from a user-space program.

All the project steps will be performed in the supplied Debian virtual machine. The notions from the course involved in this project are the following:

- User / kernel space communication: system calls;
- Task descriptor.

Project Steps

a) Add a system call named my_syscall that takes two integers as parameters, and returns an integer. That system call computes the sum of the two parameter, then outputs a line on the kernel log, containing the result of the computation. The syscall implementation should be in a separate file from the rest of the kernel, but still in the kernel source code (not implemented as a module). Also use printk() to print out the process id of the calling process, the input arguments to the system call and the output to be returned.

b) Add a system call named my_syscall2 that takes as parameter a pointer to a character array containing a string (terminated by zero), and returns a signed integer (int). If the string size is greater than 128 bytes, the system call should immediately return -1. Otherwise, the system call’s job is to replace all occurrences of the letter "o" by the number "0" (zero) in that string. The system call returns the number of replacements performed. Also use printk() to print out the process id of the calling process, the input argument to the system call and the output to be returned.

c) Write a user-space C program invoking both system calls. In particular, be sure to check for the case where the string size is greater than 128 bytes. The C program must print on standard output: 1) your name and the process id of the running C program; 2) the arguments given to each syscall; 3) the return value of each syscall.

d) Install the new kernel and check that both system calls work correctly by using the user-space C program. You need to provide a screenshot of kernel log and screenshot of the output of the user-space C program.

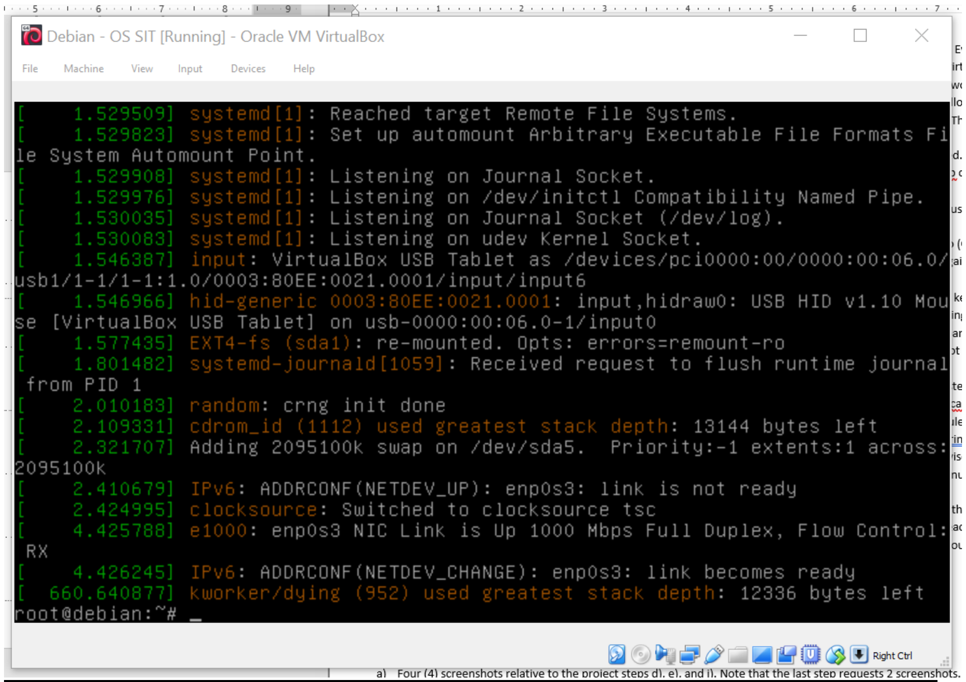
Results to be handed in:

The following is expected to be handed in:

a) Two (2) screenshots relative to the project step c).
b) A patch containing the modifications made to the kernel sources. The patch should be applicable to your original ***linux-4.9/*** directory; therefore, do not overwrite it!
c) The sources of your system call implementations and your modified version of the architecture-specific syscall table, as well as the modified version of the syscalls header file syscalls.h.
d) The sources of the C program used to test the newly added system calls.
e) An additional short report (maximum 2 pages) should be submitted, summarizing how you added the syscalls to the kernel.

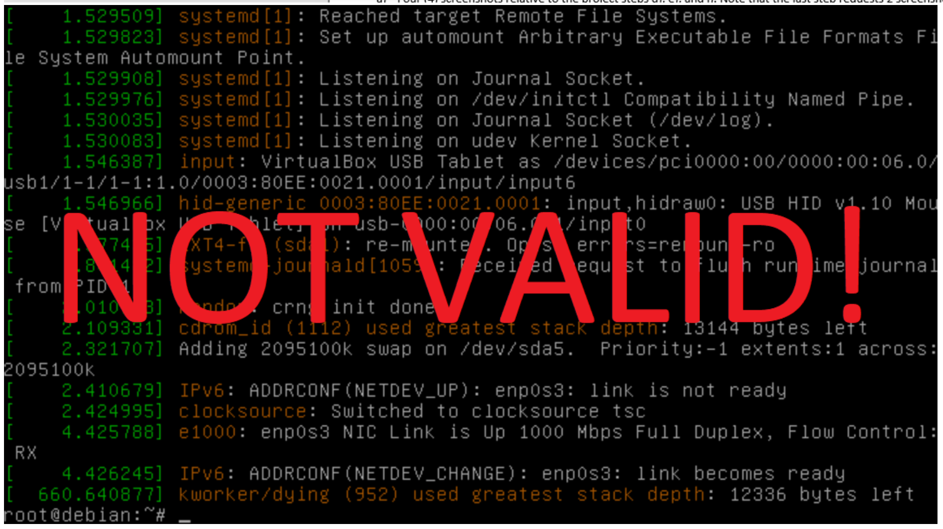
All of these should be contained in a tarball, with the following format: STUDENT_ID.lab2.tar.gz. Where STUDENT_ID is your student identification number. The tarball must be submitted on Canvas before the deadline.

Every screenshot should be taken with the entire VirtualBox window, an example of a valid screenshot and a non-valid one follow.



```
[    1.529509] systemd[1]: Reached target Remote File Systems.
[    1.529823] systemd[1]: Set up automount Arbitrary Executable File Formats Fi
le System Automount Point.
[    1.529908] systemd[1]: Listening on Journal Socket.
[    1.529976] systemd[1]: Listening on /dev/initctl Compatibility Named Pipe.
[    1.530035] systemd[1]: Listening on Journal Socket (/dev/log).
[    1.530083] systemd[1]: Listening on udev Kernel Socket.
[    1.546387] input: VirtualBox USB Tablet as /devices/pci0000:00/0000:00:06.0/
usb1/1-1/1-1:1.0/0003:80EE:0021.0001/input/input6
[    1.546966] hid-generic 0003:80EE:0021.0001: input,hidraw0: USB HID v1.10 Mou
se [VirtualBox USB Tablet] on usb-0000:00:06.0-1/input0
[    1.577435] EXT4-fs (sda1): re-mounted. Opts: errors=remount-ro
[    1.801482] systemd-journald[1059]: Received request to flush runtime journal
 from PID 1
[    2.010183] random: crng init done
[    2.109331] cdrom_id (1112) used greatest stack depth: 13144 bytes left
[    2.321707] Adding 2095100k swap on /dev/sda5.  Priority:-1 extents:1 across:
2095100k
[    2.410679] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[    2.424995] clocksource: Switched to clocksource tsc
[    4.425788] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 RX
[    4.426245] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[  660.640877] kworker/dying (952) used greatest stack depth: 12336 bytes left
root@debian:~# _
```

a) Four (4) screenshots relative to the project steps d), e), and i). Note that the last step requests 2 screenshots.



```
[    1.529509] systemd[1]: Reached target Remote File Systems.
[    1.529823] systemd[1]: Set up automount Arbitrary Executable File Formats Fi
le System Automount Point.
[    1.529908] systemd[1]: Listening on Journal Socket.
[    1.529976] systemd[1]: Listening on /dev/initctl Compatibility Named Pipe.
[    1.530035] systemd[1]: Listening on Journal Socket (/dev/log).
[    1.530083] systemd[1]: Listening on udev Kernel Socket.
[    1.546387] input: VirtualBox USB Tablet as /devices/pci0000:00/0000:00:06.0/
usb1/1-1/1-1:1.0/0003:80EE:0021.0001/input/input6
[    1.546966] hid-generic 0003:80EE:0021.0001: input,hidraw0: USB HID v1.10 Mou
se [VirtualBox USB Tablet] on usb-0000:00:06.0-1/input0
[    1.577435] EXT4-fs (sda1): re-mounted. Opts: errors=remount-ro
[    1.801482] systemd-journald[1059]: Received request to flush runtime journal
 from PID 1
[    2.010183] random: crng init done
[    2.109331] cdrom_id (1112) used greatest stack depth: 13144 bytes left
[    2.321707] Adding 2095100k swap on /dev/sda5.  Priority:-1 extents:1 across:
2095100k
[    2.410679] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[    2.424995] clocksource: Switched to clocksource tsc
[    4.425788] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 RX
[    4.426245] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[  660.640877] kworker/dying (952) used greatest stack depth: 12336 bytes left
root@debian:~# _
```

Further Notes

In order to create a patch you can use one of the following solutions.

- Use git. You can find *original_commit_hash*, the hash of your initial git commit, using *git log*

  ```
  $ git diff original_commit_hash
  ```

- Use diff. ***linux-4.9/*** is the vanilla Linux kernel source code and ***lab1-linux-4.9/*** is a copy of it, which you modify for this homework

  ```
  $ diff -ur linux-4.9/ lab2-linux-4.9/
  ```