Virginia Tech ❖ Bradley Department of Electrical and Computer Engineering

ECE 4984 / 5984 Linux Kernel Programming
Spring 2017

Small Project: *printk* and adding a new system call

1 Introduction

The objectives of this project is to download the kernel sources using the *git* version control software, and create a new branch based on a particular version of Linux. On that branch we will apply the following code modifications:

1. Add a `printk` statement in the kernel log during the boot process;
2. Add two new, simple, system calls.

All the project steps will be performed in a supplied VirtualBox VM. The notions from the course involved in this project are the following:

- Linux source code exploration and compilation. Installing and running a modified kernel;
- Version control with git: cloning, branching, creating a patch;
- The `printk` primitive;
- User / kernel space communication: system calls.

2 Project steps

1. Grab the VirtualBox VM that will be used for the project here: `http://bit.ly/2bKfKVO`. Import it in VirtualBox (*File → Import Appliance*). Customize the amount of CPUs and memory according to your physical machine. It is recommended, if possible, to give to the VM several CPUs and at least 2GB of RAM. The VM contains a Debian 8 OS. The login is `user` and the password is `a`. There is no graphical interface. Concerning the text editor, it is suggested to use `nano` or `vim`. Al the next steps have to be performed in the VM;
2. Clone, using git, a local copy of the Linux kernel sources. Using the git protocol, the url is:
   `git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git`
   Checkout the `v4.0` git *tag* in the kernel sources, and create a new branch based on that tag. Name that branch as you please;
3. Compile and install that kernel, then reboot to execute it. Check that the currently running kernel is the one we just compiled;
4. Working on that newly created branch, add a `printk` statement that outputs `Hello world` in the kernel log during the kernel boot process. It is asked to print this line as early as possible during the boot process. Compile, install, reboot and check that the line is present in the kernel log;
5. 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;

6. Add a system call named `my_syscall2` that takes as parameter a pointer to a character array containing a string, and returns a signed integer (`int`). If the string size is superior to 128 bytes, the system call should immediately return -1. Otherwise, the system call 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;

7. Write a C program invoking both system calls. In particular, be sure to check for the case where the string size is superior to 128 bytes. Install the new kernel and check that the system calls work correctly.

To get several shells in the VM, and also avoid using the not-so-practical VirtualBox window, you can ssh from the host in the VM. Get the IP of the VM by running `sudo ifconfig` in the VM, then from the host (replace the IP by the one you found through `ifconfig`):

```
$ ssh user@192.168.33.10
```

3 Results to be handed - Deadline: 2017/01/31

The following is expected to be handed by 2017/01/31:

1. Screenshots showing that the `printk` statement and the system calls work as expected;

2. A patch containing the modifications made to the kernel sources. The patch should be applicable to a vanilla Linux sources directory, cloned from `git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git` using the tag `v4.0`. The patch should be applicable via this command:

   ```
   $ cd <linux source folder>
   $ patch -p1 < <patch file name>
   ```

   Or that command:

   ```
   $ cd <linux source folder>
   $ git apply <patch file name>
   ```

   Note that before submission, it is **strongly advised to test** on a clean checkout of linux tag `v4.0` (1) the patch application, and (2) the fact that the functionalities added after the patching process are working correctly.

3. The sources of the C program used to test the newly added system calls.

All of this should be contained in a *tarball*, with the following format: `<VT PID>.project1.tar.gz`.
For example: `johndoe.project1.tar.gz`

4 Additional information

4.1 `printk` usage

This function is used similarly to the way `printf` is called from user space. `printk` outputs on the kernel log. In order to display that log from user space, use the shell command `dmesg`.

4.2 Git mini-guide

- Cloning a remote repository: `git clone <url>`
- Switching to a tag/branch: `git checkout <tag/branch name>`
- Creating a new branch: `git checkout -b <branch name>`
- Creating a patch from the difference between the current branch and a another branch:
  `git diff <another branch name> > <patch file name>`