Kennesaw State University

CSE 3502 Operating Systems

Project 1 - System call

Instructor: Kun Suo
Points Possible: 100

Assignments

Assignment 0: Build the Linux kernel (50 points)

Create a virtual machine using VirtualBox on your machine. As the kernel compiling is pretty large, please make sure your VM has at least 4GB memory and 80GB storage (if your disk is small, create a 60GB disk for VM). For the Operating System, use Ubuntu 18.04 iso: https://releases.ubuntu.com/18.04/

How to build one ubuntu VM?

Windows 10:
https://www.youtube.com/watch?v=QbmRXJJKsvs
MacOS:
https://www.youtube.com/watch?v=GDoCrfPma2k&t=321s

Step 1: Get the Linux kernel code

Before you download and compile the Linux kernel source, make sure you have development tools installed on your system. We recommend you work this project on your virtual machine.

In Ubuntu, install this software using apt:

```
$ sudo apt-get install -y gcc libncurses5-dev make wget flex bison vim libssl-dev libelf-dev
```

To obtain the version of your current kernel, type:

```
$ uname -r
5.0
```

(For newer distributions of Ubuntu, you can see 5.x or 6.x)

Then, download kernel 5.1 and extract the source:

```
$ wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.1.tar.gz
$ tar xvzf linux-5.1.tar.gz
```

We will refer LINUX_SOURCE to the top directory of the kernel source. Go to the linux source code folder:

$ cd linux-5.1

Step 2: Configure your new kernel

Before compiling the new kernel, a .config file needs to be generated in the top directory of the kernel source. To generate the config file and make possible changes to the default kernel configurations, type:

$ make menuconfig

No changes to the default configuration are needed at this time. Press SAVE and OK, and then exit the configuration menu and a default config file will be generated. You can check .config using the following command under kernel folder. (https://youtu.be/UyOGF4UOoR0)

$ ls -al

Step 3: Compile the kernel

Please keep in mind that the compiling might take 1-2 hours, depending on your machine hardware specs and speed.

In LINUX_SOURCE, compile to create a compressed kernel image:

$ make

You can use "make -j N" to accelerate the compiling. Here N denotes the number of CPUs on your VM.

----------[Possible Error] ----------

For some distributions of Ubuntu, you may see errors like this when compiling:

No rule to make target 'debian/canonical-certs.pem', needed by 'certs/x509_certificate_list'.

[Solution]

Edit the .config file and change the value of CONFIG_SYSTEM_TRUSTED_KEYS to null

$ vim .config

Before:

CONFIG_SYSTEM_TRUSTED_KEYRING=y

CONFIG_SYSTEM_TRUSTED_KEYS="debian/canonical-certs.pem"

After:

CONFIG_SYSTEM_TRUSTED_KEYRING=y

CONFIG_SYSTEM_TRUSTED_KEYS=""

Then recompile the kernel using the make command

To compile kernel modules:

$ make modules

You can use "make modules -j N" to accelerate the compiling. Here N denotes the number of CPUs on your VM.

Step 4: Install the kernel

Install kernel modules (become a root user, use the su command):

```
$ sudo make modules_install
```

Install the kernel:

```
$ sudo make install
```

If you are using Ubuntu, you need to create an init ramdisk manually:

```
$ sudo mkinitramfs -o /boot/initrd.img-5.1.0
$ sudo update-initramfs -c -k 5.1.0
```

The kernel image and other related files have been installed into the /boot directory. You can check it from /boot/grub/grub.cfg. Linux will boot by default using the 1st menu item.

Step 5: Modify grub configuration file

If you are using Ubuntu: change the grub configuration file:

```
$ sudo vim /etc/default/grub
```

Make the following changes:

```
GRUB_DEFAULT=0
GRUB_TIMEOUT=10
```

If your GRUB_HIDDEN_TIMEOUT_QUIET=true, change it to GRUB_HIDDEN_TIMEOUT_QUIET =false. If there is no GRUB_HIDDEN_TIMEOUT_QUIET, just ignore it.

If your GRUB_TIMEOUT_STYLE=hidden, change it to GRUB_TIMEOUT_STYLE=menu. If there is no GRUB_TIMEOUT_STYLE, just ignore it.

Then, update the grub entry:

```
$ sudo update-grub2
```

Step 6: Reboot your VM

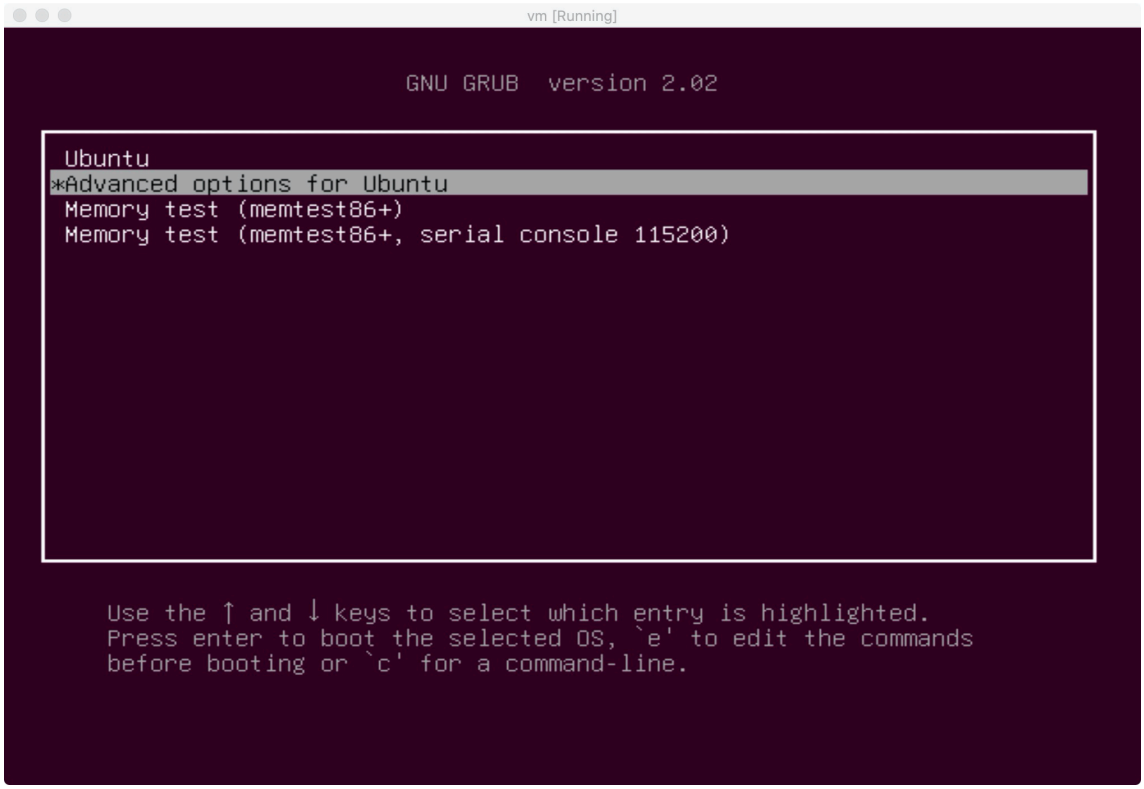
Reboot to the new kernel:

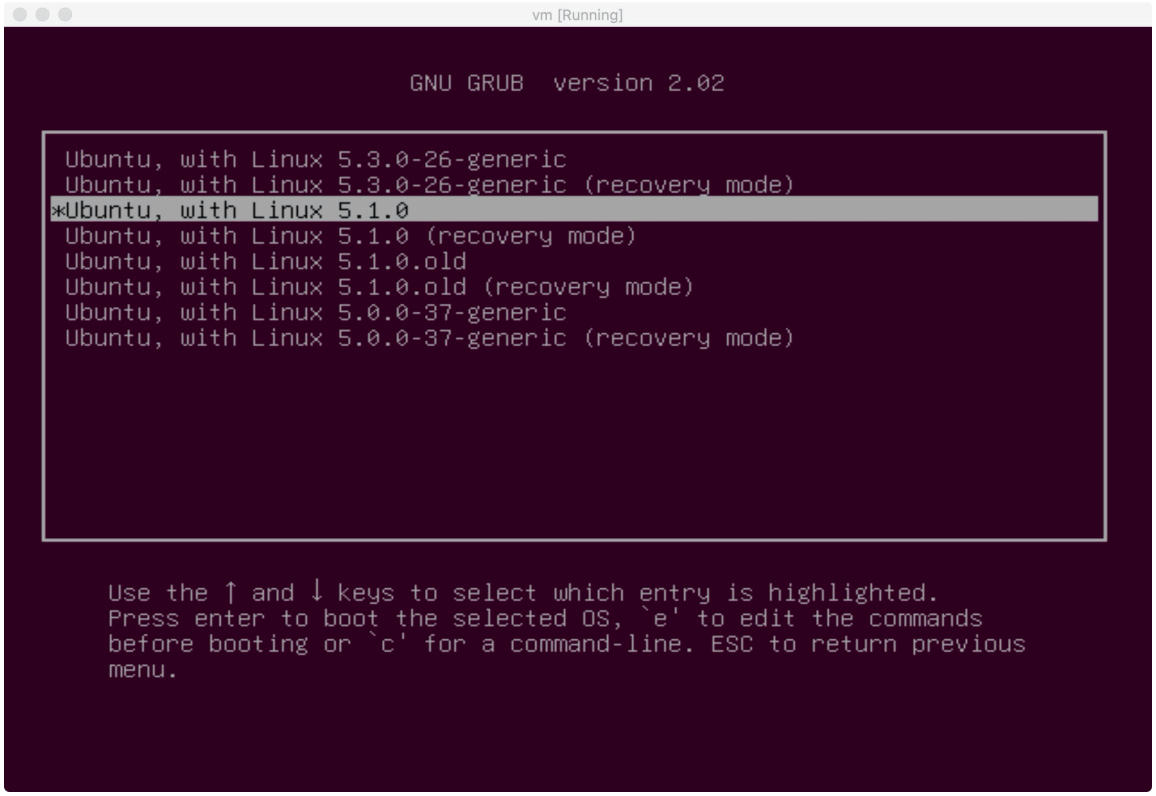
```
$ sudo reboot
```

(If you are using university VM, ignore the following steps. It only works for local VM)

Immediately after the BIOS/UEFI splash screen during boot, with BIOS, quickly press and hold the Shift key, which will bring up the GNU GRUB menu. (If you see the Ubuntu logo, you've missed the point where you can enter the GRUB menu.)

Select the following option:





After boot, check if you have the new kernel:

$ uname -r

5.1.0

Submission of assignment 0:

Please submit the screenshot of $uname -r

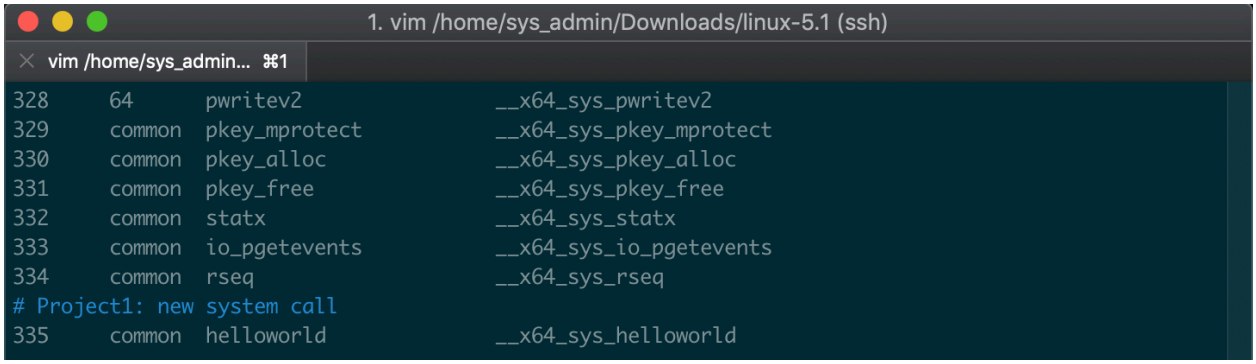
Assignment 1: Add a new system call into the Linux kernel (50 points)

In this assignment, we add a simple system call helloworld to the Linux kernel. The system call prints out a hello world message to the syslog. You need to implement the system call in the kernel and write a user-level program to test your new system call. Go to the kernel source code folder linux-5.1.

$ cd linux-5.1

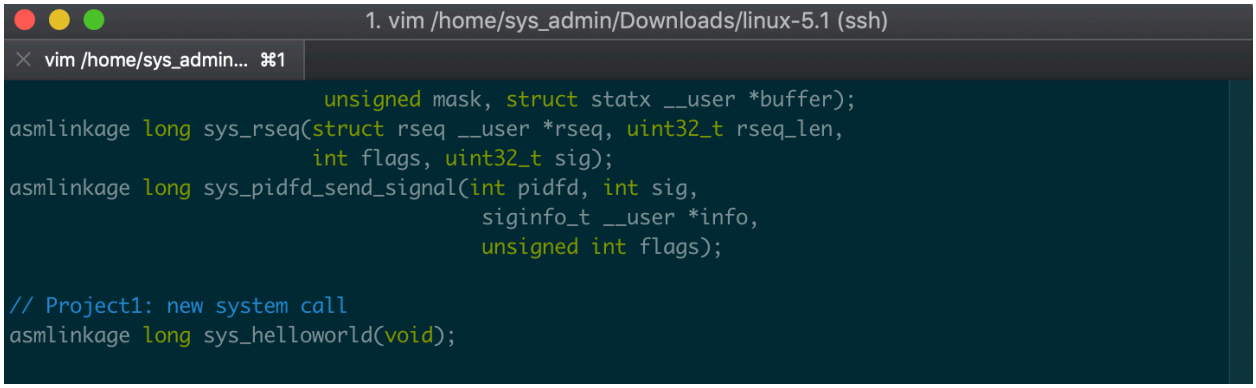
Step 1: register your system call

$ vim arch/x86/entry/syscalls/syscall_64.tbl



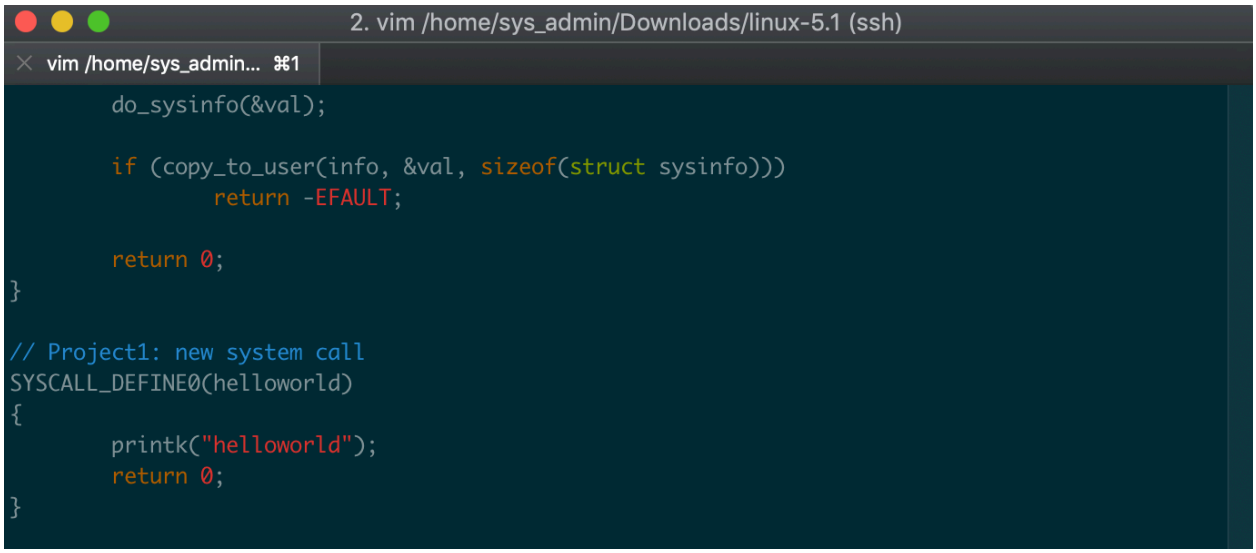
Step 2: declare your system call in the header file

$ vim include/linux/syscalls.h



Step 3: implement your system call

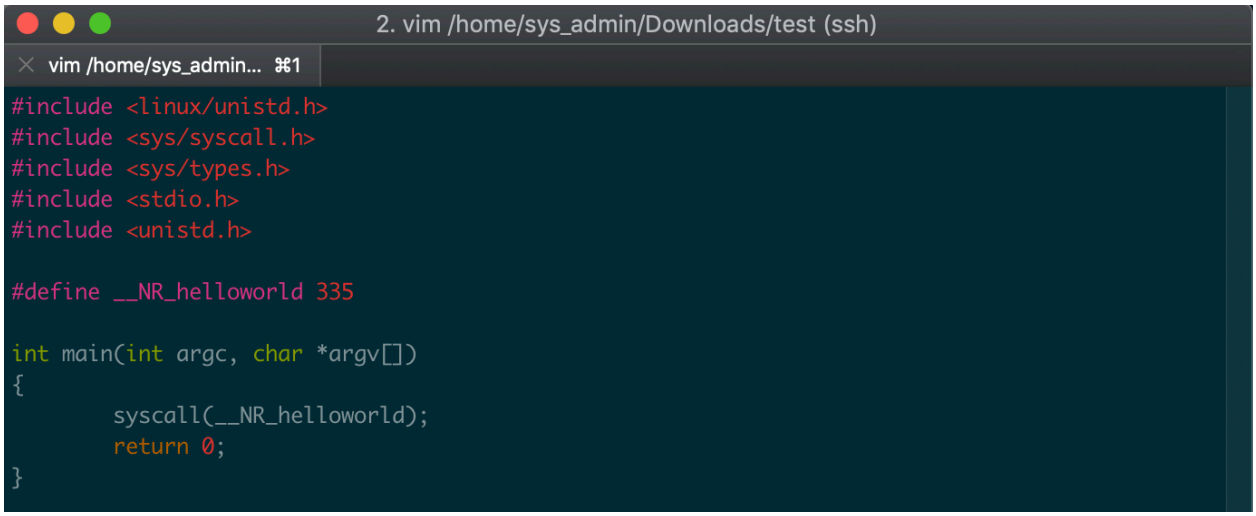
$ vim kernel/sys.c



Repeat step 3 and 4 in assignment 0 to re-compile the kernel and reboot to the new kernel.

Step 4: write a user-level program to test your system call

Go to your home directory and create a test program test_syscall.c



Compile the user level program:

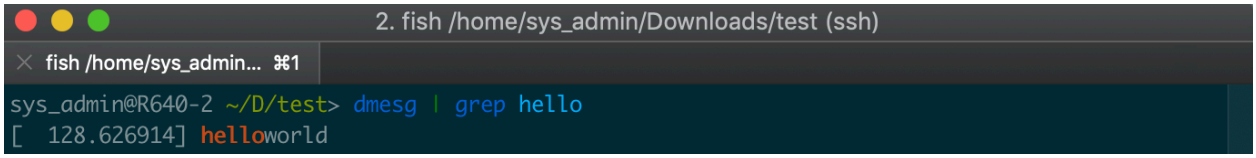
$ gcc test_syscall.c -o test_syscall

Test the new system call by running:

$ sudo ./test_syscall

The test program will call the new system call and output a helloworld message at the tail of the output of dmesg.

$ dmesg | grep hello



Submission of assignment 1:

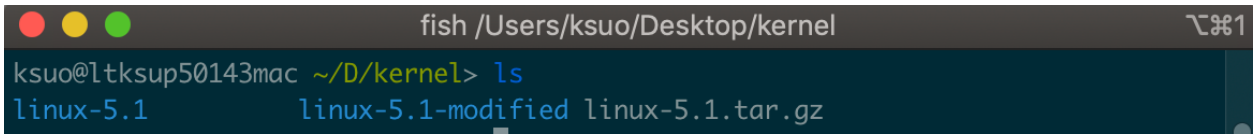
Please have two copies of kernel source code: 1) the original kernel source code without any modification; 2) the kernel source code you modified. You can define the folder name based on your need. Here I use *linux-5.1* as the original source code without modification and *linux-5.1-modified* as the source code I worked on.

Instruction:

Change your linux-5.1 folder name into linux-5.1-modified.
$ mv linux-5.1 linux-5.1-modified

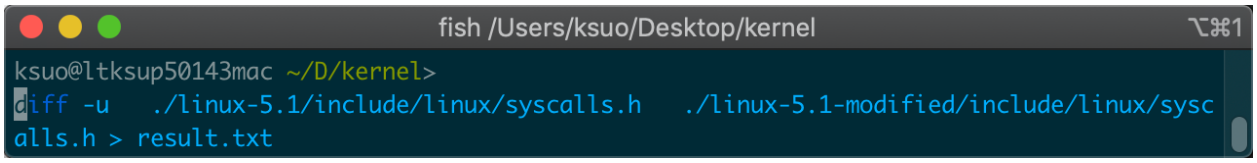
Unzip the source code again
$ tar xvzf linux-5.1.tar.gz

Then you should have two folders of the Linux source code.



Please use diff command to highlight your modification (Here the original_file.c refers the file or file path of the original file source code; the modified_file.c refers the file or file path of the file source code you have modified):
$ diff -u original_file.c modified_file.c > result.txt

For example, to show the difference between file include/linux/syscalls.h, just use the command below:



For the kernel code, please do not add the entire kernel source code. Just add your modification code, e.g., result1.txt, result2.txt, result3.txt, …

Please submit the screenshot of $ dmesg | grep hello and result1.txt, result2.txt, result3.txt, …