Assignment 2

Due Feb 9 by 11:59pm **Points** 18 **Submitting** a file upload

Some of your programming assignments will be done on the Ubuntu Linux 14.04.1 operating system, which will be running inside a virtual machine (VM). The advantage of doing your OS homework in a VM is that it removes the need for a dedicated physical machine that you can crash when your OS code is buggy. Of course, if you do happen to have a spare physical machine, feel free to do your homework there after installing the 64-bit version of Ubuntu 14.04.1 on it.

The VMM (virtual machine manager) that we will be using is called **VirtualBox (https://www.virtualbox.org/)** .

Getting Started

You have two options: boot up a VM using a virtual hard disk image that we have created, or install Linux in a VM on your own. We'll describe these in turn. We have tested the assignment on CADE, but it can be painful/laggy especially if you try to run VirtualBox remotely. Overall, it's probably best to run VirtualBox on your local machine if you have the resources to do so.

OPTION 1: Running our virtual hard disk image from a CADE lab machine

Login to a CADE lab Linux machine. You can do this remotely but it will probably work best if you're actually at the lab.

Start VirtualBox:

```
virtualbox
```

This will bring up VirtualBox's manager console. Unless you have used VirtualBox before, it won't contain any virtual machines.

Although you can create and start virtual machines using the manager console, some functionality that you need right now is not available from the GUI. So run these commands from a separate terminal, in order to access VirtualBox's functionality at a deeper level. Make sure each command succeeds before proceeding to the next.

Make a new VM:

```
VBoxManage createvm --name "clone1" --ostype Ubuntu_64 --register
```

Give it half a GB of memory:

```
VBoxManage modifyvm "clone1" --memory 512
```

Give the VM a SATA disk:

> VBoxManage storagectl "clone1" --name "sata1" --add sata

Now the magic command which attaches our hard disk image to your virtual machine in such a way that everyone can privately change their own copy of this disk (this is a long single-line command -- make sure you type it that way even if your browser breaks it across lines):

> VBoxManage storageattach "clone1" --storagectl "sata1" --port 0 --device 0 --type hdd --medium /home/cs5460/ubuntu-14.04-for-OS.vdi --mtype multiattach

Now, in the VirtualBox console, double click on the new "clone1" virtual machine to start it. It will boot up an Ubuntu 14.04.1 system with you logged in as user "student". Your password (which you will need in order to become root) is also "student". At any time you can run a command as root like this:

> sudo [command]

We suggest not running things as root unless you actually need to. But it's your virtual machine so do as you like.

If you mess up your virtual machine somehow, you can delete it using the manager console and then create a new one using the instructions above. Of course this will destroy all files that you stored on that virtual machine. As always, you should regularly backup your work.

You may install whatever additional packages you want on this virtual machine using a command line this:

> sudo apt-get install [package name]

Just be careful about using up all of your CADE lab quota. The default virtual machine image that you are using will not use up any of your quota, but anytime you change or create a file on it, the extra material will be stored as part of your quota.

For those who haven't used apt before, **here's a quick summary of the useful commands (https://help.ubuntu.com/12.04/serverguide/apt-get.html)** . A web search will help you find a huge amount of additional information if you require it.

OPTION 2: Running VirtualBox Somewhere Else

If you want to install VirtualBox at home (this should work on Windows, OS X, or Linux) you can do that, and then just follow the instructions above. Of course you'll have to download the .vdi file to your machine first.

OPTION 3: Making Your Own VM Image

If you feel like it, make your own VM image using whatever VMM you like. VMWare Player is an excellent choice.

Now grab this DVD image for Ubuntu 14.04.1 for x86-64:

> http://releases.ubuntu.com/14.04.1/ubuntu-14.04.1-desktop-amd64.iso

Boot up a new virtual machine with at least 1024 MB of RAM and 8 GB of hard disk space. Tell the VMM to mount the Ubuntu 14.04.1 ISO. Install Ubuntu onto the virtual hard drive. There's no need to install any third-party software, just do the default installation.

Reboot the VM and login.

Install updates and also some modules that you will need in order to build kernel code:

> sudo apt-get update
> sudo apt-get dist-upgrade
> sudo apt-get install dkms build-essential

Some packages that we like and that you may find useful:

> sudo apt-get remove vim-common
> sudo apt-get install vim emacs23 git subversion

Reboot the VM and login. Now you are ready to proceed with the lab.

Building and Installing a Kernel Module

NOTE: Instead of working in your home directory, it would be preferable to do this work in a subdirectory. This will avoid clutter, keep you from mixing up files from different homeworks, and make it easier to save your work off of the virtual machine.

NOTE: If the Ubuntu UI gives you brutal lag in VirtualBox (it does for me), switch to a Linux VT (Ctrl-Alt-F1) to make life easier if you're comfortable at a terminal.

From the assignment 2 files directory, download **hello.c** and also create a file called **Makefile** with these contents (both files can be scp'ed or git cloned from /home/cs5460/assignment2 on CADE as well, which isn't behind an authentication portal):

```
obj-m := hello.o
KDIR := /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)

all:
	$(MAKE) -C $(KDIR) M=$(PWD) modules

clean:
	$(MAKE) -C $(KDIR) M=$(PWD) clean
```

Compile the kernel module:

> make

Several files will be created by this, including the actual loadable kernel module "hello.ko". If this file does not now exist, there is a problem. First try to debug it yourself, then talk to Ryan or a TA.

Now, load the module into the kernel:

```
sudo insmod hello.ko
```

Use this command to print the kernel log:

```
dmesg
```

The last message should include the string "Loading the hello module." Look in the hello.c file to see where this was printed.

You can try to do the following instead of viewing the entire output from dmesg:

```
dmesg | grep "hello"
```

Look at the list of currently loaded modules:

```
lsmod
```

The hello module should be one of them.

Now, remove your new module:

```
sudo rmmod hello
```

Make sure that the module is not present anymore and also make sure that "Unloading the hello kernel module." message appears in the log.

```
lsmod
dmesg
```

You may have noticed that in hello.c, we called printk() instead of printf(). To see the full list of functions that can be called from a kernel module:

```
cat /proc/kallsyms | grep ' T '
```

There are more than 16,000 functions available. Moreover, each module that is loaded can export additional functions that other modules can use.

Saving Your Work

Your home directory (and all other files) on the virtual machine are not backed up. To move files off of your virtual machine -- which you should do routinely in case it gets messed up somehow -- there are many options:

- Dropbox or Google Drive
- scp
- a VirtualBox shared folder

- a revision control system such as Git or Subversion (these are already installed for you if you are using our shared VM disk)

Use whichever of these you prefer. **Just make sure to do it.**

Your Assignment

Since the purpose of this homework is simply to debug your virtual machine workflow, your assignment is extremely simple:

- Make a copy of hello.c called my-module.c
- Modify Makefile so that it builds my-module.ko instead of hello.ko
- Modify my-module.c so that when it is loaded, it prints:

  > LOADING my-module, which was written by [name]

  Upon being unloaded, it should print

  > UNLOADING my-module, which was written by [name]

  Of course, [name] should be replaced by your name.

Handin

Submit your my-module.c and Makefile in Canvas before the deadline. Obviously we cannot see, and will not grade, files that live in your VM.