CMPE 283 Section 1

Assignment - 1

013779246 Jinzhou Tao

013734214 Hemang Behl

Q1: Teamwork Responsibility

Jinzhou:

Made the half of the code changes for the .c file. Added extra comment and reference information with the code. Implemented kernel module installation and execution in VirtualBox environment with Ubuntu. Wrote the installation steps. Troubleshooted the errors faced and made the git diff file ready.

Hemang:

Made half of the code changes for the .c file. Searched for the control list and mask in SDM. Cloned git repo, tried building the new kernel (which corrupted the Ubuntu in dual boot), tried inserting the module in Ubuntu (in dual boot mode) and inserted the module successfully in Ubuntu (VM Ware Workstation 15). Troubleshooted the different errors (solved by installing the required libraries like flex, msr-tools or enabling 'nested VT-x').

Q2: Steps for Custom Kernel module programming, compile, installation

Requirement:

Linux environment (user should have a superuser privilege)

At least 6 GB spare disk space

A copy of the desired Linux flavor (say Ubuntu 18 LTS) to use the ISO image to run the virtual OS in VM Ware Workstation 15 (for windows)

Steps:

1. Get Linux source code.

Since each module would use packages in Linux source folder, we need to grab the source code first by following command

$ git clone https://github.com/torvalds/linux.git

2. Editing your module

Move the cmpe283-1.c & Makefile into the Linux source folder, creation of an individual folder would be recommended. And then we could use any editor to program our code in .c file.

$ gedit cmpe283-1.c

3. Compile

If the makefile is in the module source code folder, simply run:

$ make

Or run:

$ make -C /lib/modules/$(shell uname -r)/build

To compile and build the module. After the execution we could get a .ko (kernel object) file with the same name with the source file.

4. Insert and test module.

   First insert the Kernel module into module folder and load it. Recommended way is to create a symbolic link in the module folder.

   $ sudo ln -s cmpe283-1.ko /var/modules/$(uname -r)/

   $ sudo modprobe cmpe283-1

   Or we can also insert it using directly

   $ sudo insmod cmpe283-1.ko

   Now we could check if kernel module is loaded by following command

   $ lsmod | grep cmpe283-1

5. Get module execution information.

   Simply run the following information

   $ dmesg

6. Remove kernel module

   Run the following command

   $ sudo modprobe -r cmpe283-1

   Or alternatively:

   $ sudo rmmod cmpe283-1.ko

Extra Information:

1. Nested VT-X is not supported in VM

   While testing the module with an ubuntu OS on VirtualBox VM. The dmesg outputs shows following information:

   [41.784989] Run detect_vmx_features
   [41.784992] True Controls mask is NOT supported!!!
   [41.784994] Pinbased Controls MSR: 0x0
   ...
   [41.784999] Procbased Controls MSR: 0x0
   ...
   [41.785012] Exit Controls MSR: 0x0
   ...
   [41.785020] Entry Controls MSR: 0x0
   ...

   Which indicates the VMX function is not supported in the VirtualBox VM instance. After investigate on VirtualBox's manual and forum, we've learned the information that VirtualBox seems only support nested VT-x on AMD CPUs, which is why we cannot get VMX features support on a VirtualBox. Since both of our team member don't have any pc with AMD CPU, we're unable to test if the module would able to execute in a VMX available environment.

   [src: https://docs.oracle.com/cd/E97728_01/F12469/html/nested-virt.html]

CMPE 283 Section 1

2. For Windows 10 users

We can use VM Ware Workstation 15 and start a virtual machine running Ubuntu with 'nested VT-x features' enabled. For this assignment we allocated 8 GB RAM out of the available 16 GB to the VM along with 40 GB HDD space. In this case the Ubuntu installation had a few libraries missing such as msr-tools, flex, libdev. Using 'sudo su' we were able to install and use them easily.

Sample Output:

```
[  183.906700] CMPE 283 Assignment 1 Module Start
[  183.906701] Run detect_vmx_features
[  183.906705] True controls capability is supported--
[  183.906708] Pinbased Controls MSR: 0x3f00000016
[  183.906710]      External Interrupt Exiting: Can set=Yes, Can clear=Yes
[  183.906711]      NMI Exiting: Can set=Yes, Can clear=Yes
[  183.906713]      Virtual NMIs: Can set=Yes, Can clear=Yes
[  183.906714]      Activate VMX Preemption Timer: Can set=No, Can clear=Yes
[  183.906715]      Process Posted Interrupts: Can set=No, Can clear=Yes
[  183.906718] Procbased Controls MSR: 0xfff9fffe04006172
[  183.906719]      Interrupt-window exiting: Can set=Yes, Can clear=Yes
[  183.906720]      Use TSC Offsetting: Can set=Yes, Can clear=Yes
[  183.906721]      HLT Exiting: Can set=Yes, Can clear=Yes
[  183.906723]      INVLPG Exiting: Can set=Yes, Can clear=Yes
[  183.906724]      MWAIT Exiting: Can set=Yes, Can clear=Yes
```