Bilkent University
Computer Engineering

CS 342
Operating Systems

Project 3.A

Fuad Aghazada
21503691

Can OZGUREL
21400476

Section 1

21.11.2018

Information about the machine that is used:

CPU: 3.1 GHz x 2 Intel Core i7-5557U
Memory: 16GB (Host) / 8GB (Guest) 1867 MHz DDR3
Host OS: macOS High Sierra 10.13.6
Guest OS: Ubuntu 16.04 LTS
Virtual Machine: Oracle VM VirtualBox

Part A: Learning how to build a Linux kernel

In order to build the latest stable Linux kernel (linux-4.19.2), We followed the exact steps from the article [1] that we have found on the Internet.

First of all, as a preparation we needed to run the following commands:

```
sudo apt-get update
sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc
sudo apt-get install bison
sudo apt-get install flex
```

Then we went to kernel.org to download the latest stable Linux kernel, which was the version 4.19.2 as we mentioned above. After the download, we extracted the tar file and placed in my working directory. The preparation steps were done, it was time to configure and compile the downloaded kernel.

The following commands were executed for compilation:

```
cp /boot/config-$(uname -r) .config
make menuconfig
```

The laptop that we used is double-core so we changed 4 to 2 in the tutorial:

```
sudo make -j 2 && sudo make modules_install -j 2 && sudo make install -j 2
```

The execution time of the command above took about 2 hours. After plenty amount of time final commands had been waiting to be executed:

```
update-initramfs -c -k 4.19.2
update-grub
```

And finally the system needed to be restarted:

```
sudo reboot
```

Now when we run ***uname -r*** we see the kernel version of 4.19.2.

Part B: Learning how to write a module and develop a simple Hello World module

The following code has been rewritten by reviewing the tutorial [2] mentioned in the assignment:

```
/*
 * Hello World module
 * @author: Fuad Aghazada & Can Ozgurel
 * @date: 21.11.2018
 */

#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

/* Initializing the module */
static int __init hello(void)
{
        printk(KERN_INFO "Hello, World!\n");

        return 0;                // Module exits successfully
}

/* Cleaning the module */
static void __exit goodbye(void)
{
        printk(KERN_INFO "Module is destroyed!\n");
}

module_init(hello);
module_exit(goodbye);
```

Here is the Makefile for compiling:

```
obj-m += testModule.o

all:
        make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
        make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

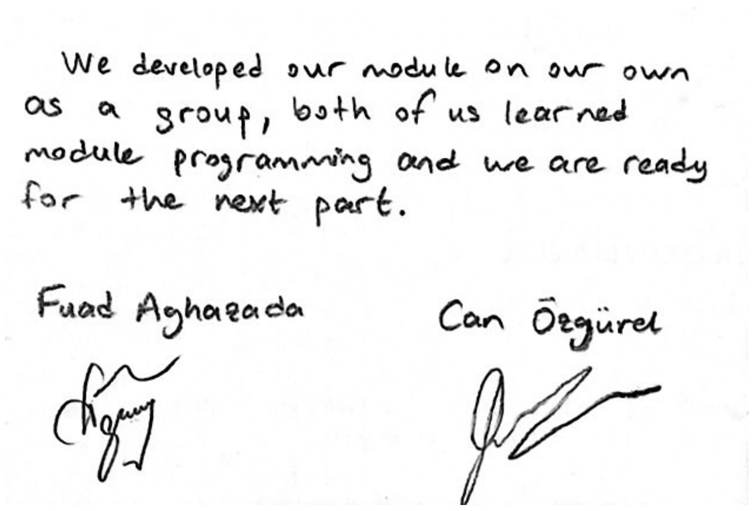
After compiling (executing ***make***), in order to add our fancy module to the kernel, we used the following command:

sudo insmod ./testModule.ko

To see the result, as also mentioned in the tutorial, we changed the directory to /proc/ and opened the 'modules' file. We were able to see the name of our module on the first line of the file. Finally, we removed the module from the kernel using the following command:

sudo rmmod testModule

Here is our signatures:



We developed our module on our own as a group, both of us learned module programming and we are ready for the next part.

Fuad Aghazada [illegible]

Can Özgürel [illegible]

References

[1] Sreehari, "https://medium.freecodecamp.org," Medium, 30 August 2016. [Online]. Available: https://medium.freecodecamp.org/building-and-installing-the-latest-linux-kernel-from-source-6d8df5345980. [Accessed 20 November 2018].

[2] "The Linux Kernel Module Programming Guide," [Online]. Available: http://www.tldp.org/LDP/lkmpg/2.6/html/x181.html. [Accessed 20 November 2018].