Joseph Pennington

2912079

EECS 678 Lab03

```
C:\Users\jmp22\Documents\qemu_files2>qemu-img create -f qcow -b eecs678_base.qcow joseph.qcow
```

1. **This command makes a differential image off the local copy of the cycle server base image. The new image is named “joseph.qcow”.**

```
C:\Users\jmp22\Documents\qemu_files2>qemu-system-x86_64 -hda joseph.qcow -smp 2 -m 2048
```

2. **Next, the image was booted up using 2048M of memory.**

```
[~]
root@debian$ adduser joseph
Adding user `joseph' ...
Adding new group `joseph' (1000) ...
Adding new user `joseph' (1000) with group `joseph' ...
Creating home directory `/home/joseph' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for joseph
Enter the new value, or press ENTER for the default
        Full Name []: joseph
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y

[~]
root@debian$
```

3. **After the image finsihed booting up, I added myself as a new user.**

```
[~]
root@debian$ vi /etc/sudoers
```

```
1 root ALL = (ALL) ALL
2 joseph ALL = (ALL) ALL
~
~
```

4. **Next, the command on the left opened the sudoers file where I added the two lines to give the new user system admin rights.**

```
[~]
root@debian$ usermod -a -G sudo joseph
```

5. **Once the user was added to the sudoers file, this command added the new user to the sudo group.**

```
[~]
root@debian$ mkdir /home/joseph/kernel

[~]
root@debian$ mv ~/linux-2.6.32.60/home/joseph/kernel
mv: missing destination file operand after `/root/linux-2.6.32.60/home/joseph/ke
rnel'
Try `mv --help' for more information.

[~]
root@debian$ mv ~/linux-2.6.32.60/home/joseph/kernel/
mv: missing destination file operand after `/root/linux-2.6.32.60/home/joseph/ke
rnel/'
Try `mv --help' for more information.

[~]
root@debian$ mv ~/linux-2.6.32.60/ /home/joseph/kernel

[~]
root@debian$ chown -R joseph:joseph /home/joseph/kernel
```

6. **The next step was to move the kernel source and the config file to the user's home directory. Then the user was granted ownership of those files. Ignore the errors when moving the files.**

```
[~]
root@debian$ apt-get install sudo_
```

7. **This command installs sudo onto the kernel. This enables the new user to call commands by using the sudo rights.**

```
[~]
root@debian$ su joseph
joseph@debian:/root$
```

8. **Next, I switched from the root user to the my new user I finished creating.**

```
joseph@debian:/root$ sudo apt-get install libz-dev_
```

9. **This command installs libz-dev to help prevent any errors that may occur during the build process.**

```
joseph@debian:/root$ cd
joseph@debian:~$ cd /home/joseph/kernel/linux-2.6.32.60/
```

10. **The directory was then changed to the home directory. Then it was changed to the "linux-2.6.32.60" directory.**

```
joseph@debian:~/kernel/linux-2.6.32.60$ mkdir hello
joseph@debian:~/kernel/linux-2.6.32.60$ cd hello
joseph@debian:~/kernel/linux-2.6.32.60/hello$ vi hello.c
```

11. **Once in the proper directory, I made a directory called "hello" and created the hello.c file inside the new directory. I then opened the file using vi.**

```
#include <linux/kernel.h>
asmlinkage long sys_hello(void){
        printk("Hello world\n");
        return 100;
}
~
~
```

12. **Once calling vi, I wrote the script that will print “Hello world” and return the value of 100.**

```
joseph@debian:~/kernel/linux-2.6.32.60/hello$ vi Makefile
```

13. **Next, I created a Makefile to compile the hello.c file.**

```
obj-y := hello.o
~
~
```

14. **I added the following line the Makefile to compile the hello.c file**

```
ifeq ($(KBUILD_EXTMOD),)
core-y          += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ hello/_
```

15. **In the linux-2.6.32.60 directory, I added the “ hello/” to the end of this line in the Makefile.**

```
        .long sys_perf_event_open
        .long sys_sched_other_rr_getquantum
        .long sys_hello
- INSERT --
```

16. **In the syscall_table_32.5 file, I added the last line to the system call table.**

```
#define __NR_perf_event_open    336
#define __NR_sched_other_rr_getquantum  337
#define __NR_hello 338

#ifdef __KERNEL__

#define NR_syscalls 339
```

17. **In the unistd_32.h file, I added the last #define line and changed the number of system calls to 339.**

```
asmlinkage long sys_sched_other_rr_getquantum(void);
asmlinkage long sys_hello(void);_
-- INSERT --
```

18. **The last change was to open the syscalls.h file and add the declaration of the system call at the end of the file.**

```
joseph@debian:~/kernel/linux-2.6.32.60$ sudo vi /usr/bin/kvm-kernel-build _
```

19. **This command opens the kvm-kernel-build script.**

```
1 #!/bin/sh
2 rev=$1
3 if [ -z "$rev" ]; then
4         echo "Usage: build64 <revision>"
5         exit 1
6 fi
7 make-kpkg -j 2 --rootcmd fakeroot --initrd --revision=$rev kernel_image 2>&1
~
~
~
```

20. To prepare for the build process, I added the "-j 2" flag to help optimize the build process.

```
joseph@debian:~/kernel/linux-2.6.32.60$ sudo kvm-kernel-build 1
```

21. Once the flag was added, I ran the kvm-kernel-build script.

```
linux-2.6.32.60/  linux-image-2.6.32.60_1_i386.deb

[/home/joseph/kernel]
root@debian$ dpkg -i linux-image-2.6.32.60_1_i386.deb
Selecting previously deselected package linux-image-2.6.32.60.
(Reading database ... 23905 files and directories currently installed.)
Unpacking linux-image-2.6.32.60 (from linux-image-2.6.32.60_1_i386.deb) ...
Done.
Setting up linux-image-2.6.32.60 (1) ...
Running depmod.
Examining /etc/kernel/postinst.d.
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 2.6.32.60 /boot/vmli
nuz-2.6.32.60
update-initramfs: Generating /boot/initrd.img-2.6.32.60
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 2.6.32.60 /boot/vmlin
uz-2.6.32.60
Generating grub.cfg ...
Found linux image: /boot/vmlinuz-2.6.32.60
Found initrd image: /boot/initrd.img-2.6.32.60
Found linux image: /boot/vmlinuz-2.6.32-5-686
Found initrd image: /boot/initrd.img-2.6.32-5-686
done
```

22. Once the build finished, I installed the kernel as the root user.

```
joseph@debian:~$ uname -a
Linux debian 2.6.32.60 #2 SMP Sun Sep 20 15:31:29 CDT 2020 i686 GNU/Linux
joseph@debian:~$
```

23. Once the kernel was installed, I rebooted the system and selected the newly installed kernel. I then checked to make sure I was running the correct kernel version.

```
#include <stdio.h>
#include <sys/syscall.h>
#include <unistd.h>
#include <linux/kernel.h>
int main()
{
        long int syscall_val = syscall(338);
        printf("System call sys_hello returned %ld\n",syscall_val);
        return 0;
}
~
```

24. **To begin the testing process, I then created the test_syscall.c file and added the short script using vi. This script calls the syscall function on syscall number 338. This refers to the hello.c file I wrote earlier in the report.**

```
joseph@debian:~$ gcc test_syscall.c -o test_syscall
joseph@debian:~$ ./test_syscall
System call sys_hello returned 100
joseph@debian:~$ _
```

25. **After writing the test file, I then compiled and ran the program. The test was successful because it returned that value of 100.**

```
joseph@debian:~$ dmesg | tail -5
[   31.009618] loop: module loaded
[   36.629636] ADDRCONF(NETDEV_UP): eth0: link is not ready
[   36.632603] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: R
X
[   36.636694] ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[  540.076670] Hello world
joseph@debian:~$
```

26. **The last test was to print the last 5 lines from the kernel logs. This test was also successful because “Hello world” was printed.**