Exercise #2. Building your own custom kernel

Overview

Many of the key OS concepts covered in this course are typically implemented in a piece of program known as the kernel. For instance, you can find the kernel of Windows at c:\Windows\System32\ntoskrnl.exe. Also, you can find the kernel of Linux at /boot/vmlinuz-*. Throughout the course, it will help your study greatly by relating the concepts you learn to the corresponding portion in the kernel source. As the first step, you need to learn about how to get the kernel source and importantly how to build a custom kernel from the source and set it up.

Tasks

A. Make sure the system has all the necessary packages installed.

```
su -c 'yum install rpmdevtools yum-utils'
su -c 'yum install qt3-devel libXi-devel'
```

B. Prepare a RPM package building environment in your home directory.

```
rpmdev-setuptree
```

C. Get the source.

```
yumdownloader --source kernel
```

At your home directory, you should have the kernel source rpm and the rpmbuild directory as follow:

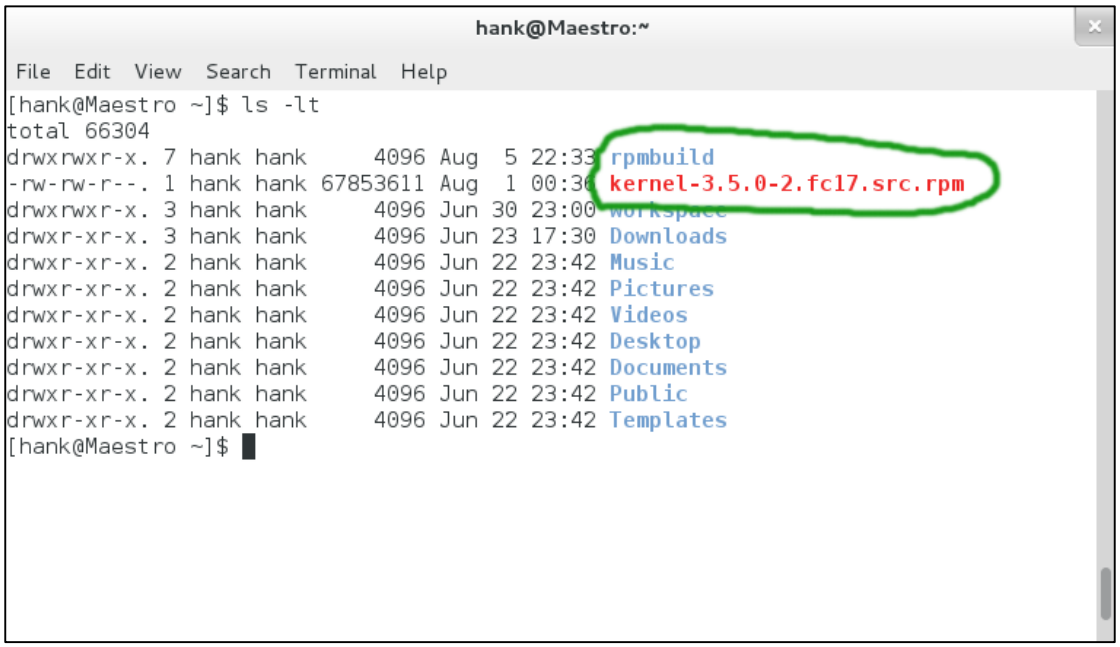


Figure 1. kernel rpm and rpmbuild directory

D. Install build dependencies for the kernel source

```
su -c 'yum-builddep kernel-<version>.src.rpm'
```

Note that you need to replace <version> with the version of the kernel source you downloaded. For instance, the version of the kernel in Figure 1 is "3.5.0-2.fc17".

E. Install the kernel source

```
rpm -Uvh kernel-<version>.src.rpm
```

This command writes the RPM contents into `${HOME}/rpmbuild/SOURCES` and `${HOME}/rpmbuild/SPECS`, where `${HOME}` is your home directory.

F. Prepare the kernel source tree

```
cd ~/rpmbuild/SPECS
rpmbuild -bp --target=$(uname -m) kernel.spec
```

The kernel source tree is now located in the `~/rpmbuild/BUILD/kernel-<version>/linux-<version>.<arch>` directory. For instance, the kernel source tree directory should look like:

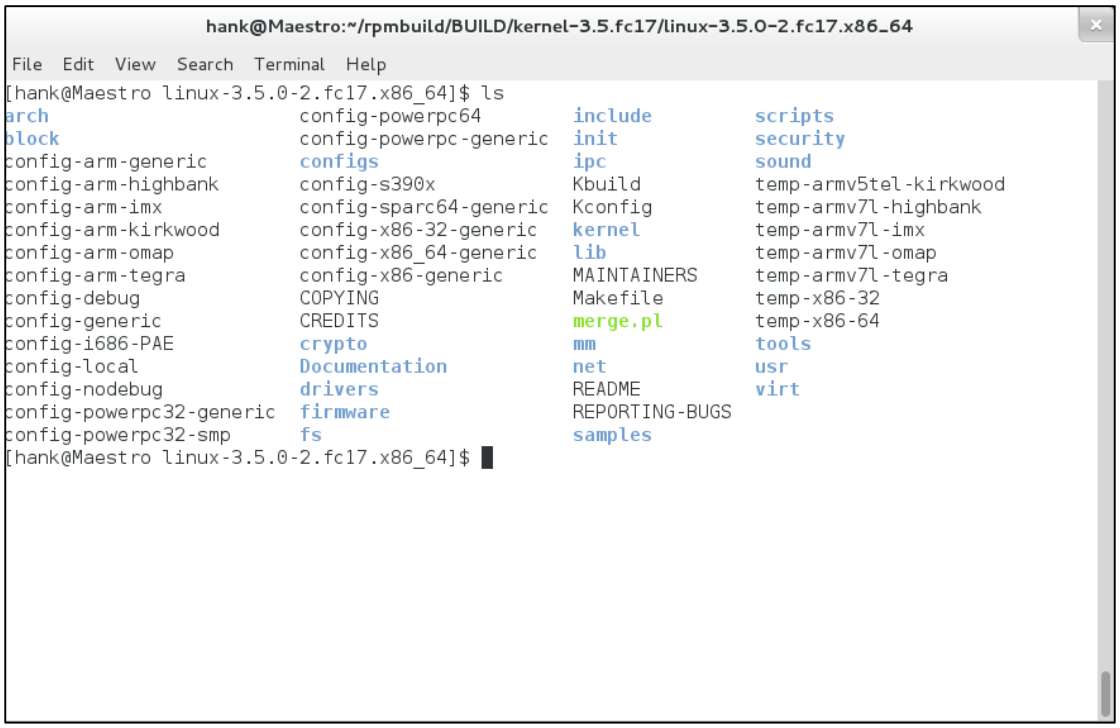


Figure 2. Kernel Source Tree

G. Configure Kernel Options

```
cd ~/rpmbuild/BUILD/kernel-$ver.$fedver/linux-$ver.$arch/
make xconfig
```

H. Give your kernel a unique name by changing EXTRAVERSION in Makefile

```
hank@Maestro:~/rpmbuild/BUILD/kernel-3.5.fc17/linux-3.5.0-2.fc17.x86_64
File  Edit  View  Search  Terminal  Help
VERSION = 3
PATCHLEVEL = 5
SUBLEVEL = 0
EXTRAVERSION = _OS_COURSE_1234567
NAME = Saber-toothed Squirrel

# *DOCUMENTATION*
# To see a list of typical targets execute "make help"
# More info can be located in ./README
# Comments in this file are targeted only to the developer, do not
# expect to learn how to build the kernel reading this file.

# Do not:
# o  use make's built-in rules and variables
#    (this increases performance and avoids hard-to-debug behaviour);
# o  print "Entering directory ...";
MAKEFLAGS += -rR --no-print-directory

# Avoid funny character set dependencies
unexport LC_ALL
LC_COLLATE=C
LC_NUMERIC=C
export LC_COLLATE LC_NUMERIC

# We are using a recursive build, so we need to do a little thinking
# to get the ordering right.
```

Figure 3. Customizing kernel name

I. Build Kernel Image

```
make bzImage
```

If the build is successful, you should see the following screen

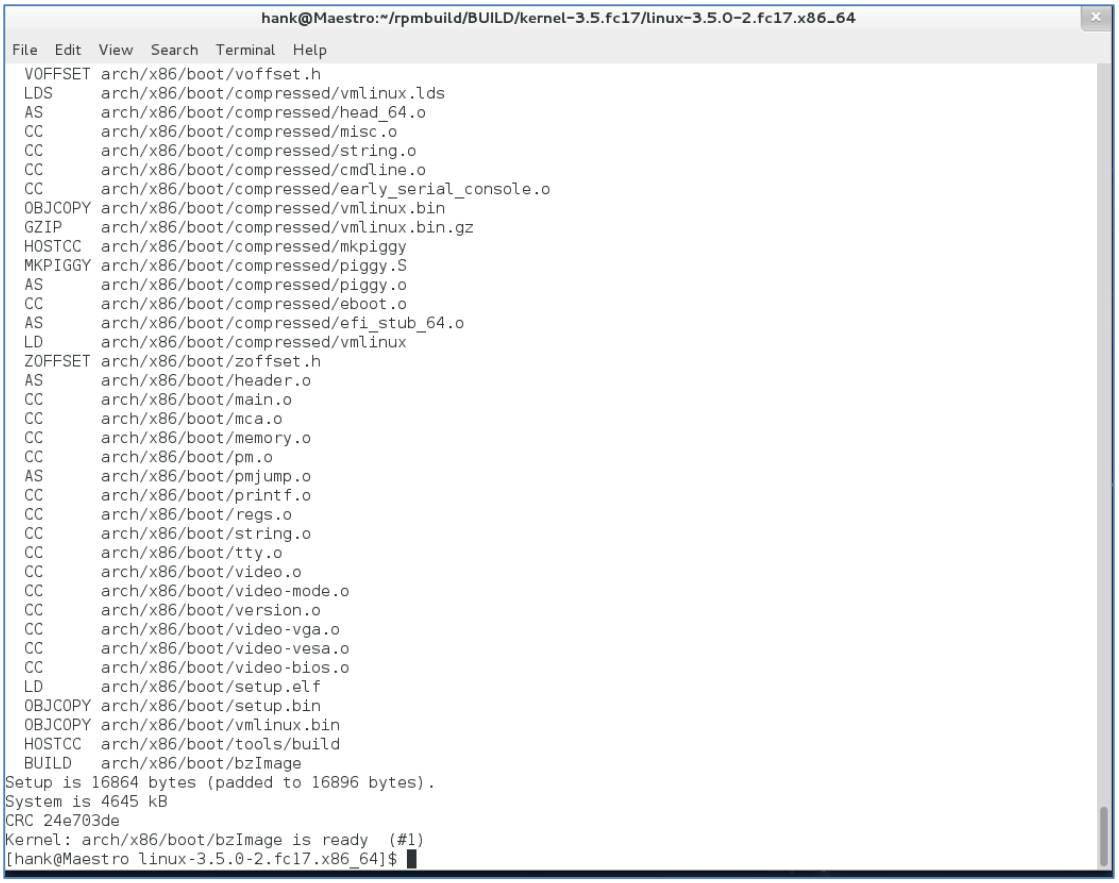


Figure 4. Build kernel succesfully

J. Build modules

```
make modules
```

If the build is successful, you will see the following screen

```
hank@Maestro:~/rpmbuild/BUILD/kernel-3.5.fc17/linux-3.5.0-2.fc17.x86_64
File  Edit  View  Search  Terminal  Help
  IHEX    firmware/qlogic/sd7220.fw
  IHEX    firmware/korg/k1212.dsp
  IHEX    firmware/ess/maestro3_assp_kernel.fw
  IHEX    firmware/ess/maestro3_assp_minisrc.fw
  IHEX    firmware/yamaha/ds1_ctrl.fw
  IHEX    firmware/yamaha/ds1_dsp.fw
  IHEX    firmware/yamaha/ds1e_ctrl.fw
  IHEX    firmware/tehuti/bdx.bin
  IHEX    firmware/tigon/tg3.bin
  IHEX    firmware/tigon/tg3_tso.bin
  IHEX    firmware/tigon/tg3_tso5.bin
  IHEX    firmware/3com/typhoon.bin
  HOSTCC  firmware/ihex2fw
  IHEX2FW firmware/emi26/loader.fw
  IHEX2FW firmware/emi26/firmware.fw
  IHEX2FW firmware/emi26/bitstream.fw
  IHEX2FW firmware/emi62/loader.fw
  IHEX2FW firmware/emi62/bitstream.fw
  IHEX2FW firmware/emi62/spdif.fw
  IHEX2FW firmware/emi62/midi.fw
  IHEX    firmware/kaweth/new_code.bin
  IHEX    firmware/kaweth/trigger_code.bin
  IHEX    firmware/kaweth/new_code_fix.bin
  IHEX    firmware/kaweth/trigger_code_fix.bin
  IHEX    firmware/ti_3410.fw
  IHEX    firmware/ti_5052.fw
  IHEX    firmware/mts_cdma.fw
  IHEX    firmware/mts_gsm.fw
  IHEX    firmware/mts_edge.fw
  H16TOFW firmware/edgeport/boot.fw
  H16TOFW firmware/edgeport/boot2.fw
  H16TOFW firmware/edgeport/down.fw
  H16TOFW firmware/edgeport/down2.fw
  IHEX    firmware/edgeport/down3.bin
  IHEX2FW firmware/whiteheat_loader.fw
  IHEX2FW firmware/whiteheat.fw
  IHEX2FW firmware/keyspan_pda/keyspan_pda.fw
  IHEX2FW firmware/keyspan_pda/xircom_pgs.fw
  IHEX    firmware/cpia2/stv0672_vp4.bin
  IHEX    firmware/yam/1200.bin
  IHEX    firmware/yam/9600.bin
[hank@Maestro linux-3.5.0-2.fc17.x86_64]$
```

Figure 5. Build modules successfully

K. Install modules

```
su –c 'make modules_install'
```

Modules will be installed under /lib/modules. For instance, following the example above, the modules will be installed under /lib/modules/3.5.0_OS_COURSE_1234567 (Figure 6).

```
hank@Maestro:/lib/modules
File  Edit  View  Search  Terminal  Help
[hank@Maestro modules]$ ls -l /lib/modules/3.5.0_OS_COURSE_1234567/
total 2452
lrwxrwxrwx.  1 root root     67 Aug 10 20:56 build -> /home/hank/rpmbuild/BUILD/kernel-3.5.fc17/linux-3.5.0-2.fc17.x86
_64
drwxrwxr-x. 12 root root   4096 Aug 10 20:59 kernel
-rw-r--r--.  1 root root 639919 Aug 10 21:01 modules.alias
-rw-r--r--.  1 root root 630139 Aug 10 21:01 modules.alias.bin
-rw-rw-r--.  1 root root   6523 Aug 10 20:56 modules.builtin
-rw-r--r--.  1 root root   8682 Aug 10 21:02 modules.builtin.bin
-rw-r--r--.  1 root root 232540 Aug 10 21:00 modules.dep
-rw-r--r--.  1 root root 337499 Aug 10 21:00 modules.dep.bin
-rw-r--r--.  1 root root    274 Aug 10 21:03 modules.devname
-rw-rw-r--.  1 root root  90524 Aug 10 20:56 modules.order
-rw-r--r--.  1 root root    131 Aug 10 21:02 modules.softdep
-rw-r--r--.  1 root root 231816 Aug 10 21:02 modules.symbols
-rw-r--r--.  1 root root 293450 Aug 10 21:02 modules.symbols.bin
lrwxrwxrwx.  1 root root     67 Aug 10 20:56 source -> /home/hank/rpmbuild/BUILD/kernel-3.5.fc17/linux-3.5.0-2.fc17.x8
6_64
[hank@Maestro modules]$
```

Figure 6. Kernel modules installation location

L. Install kernel image

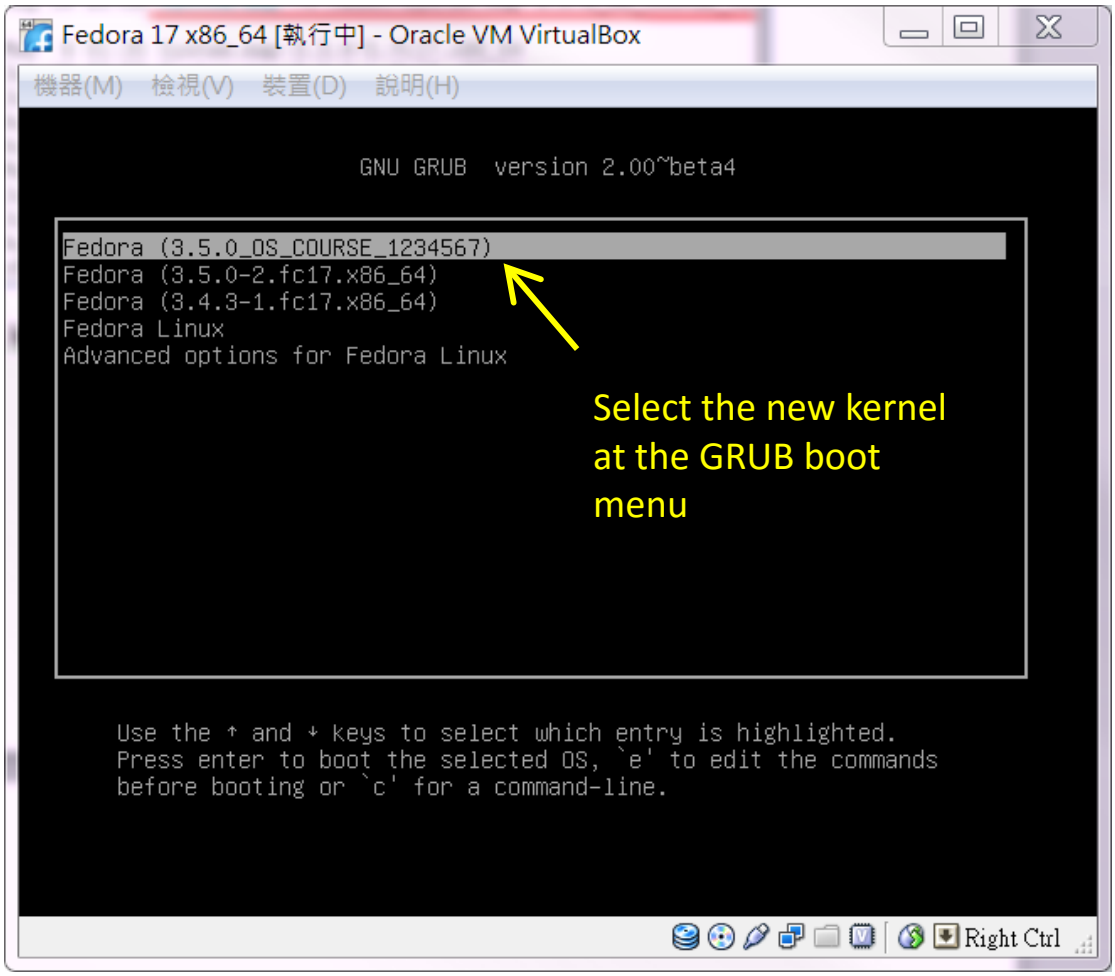
```
su –c 'make install'
```

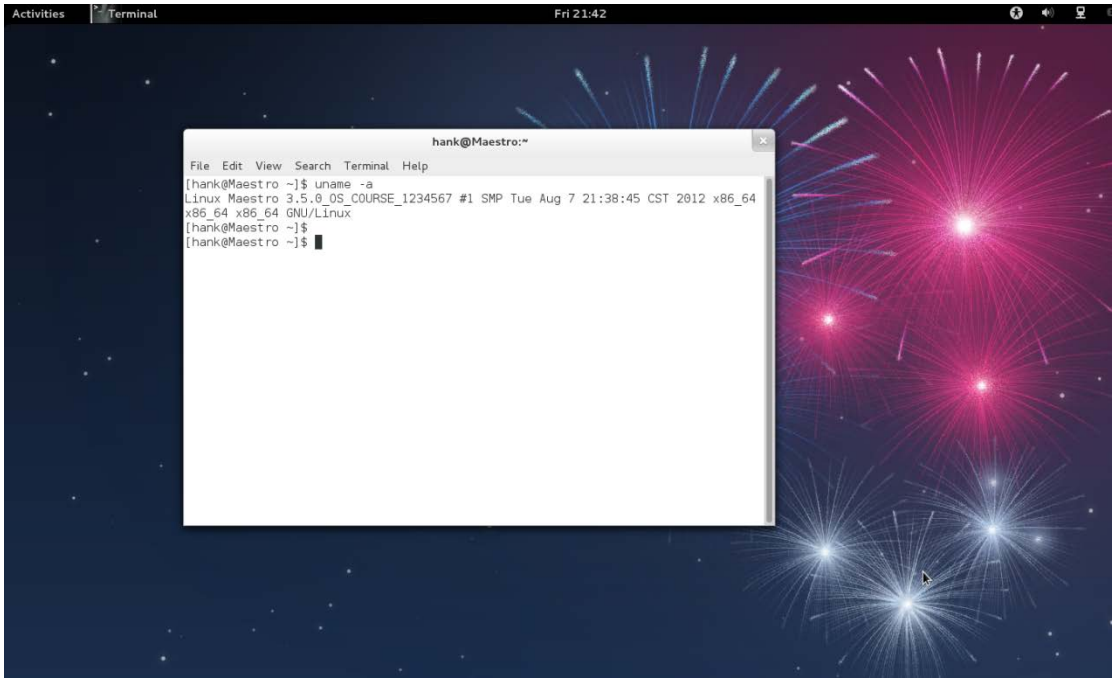
The kernel image will be installed under /boot as shown in Figure 7.

```
hank@Maestro:~/rpmbuild/BUILD/kernel-3.5.fc17/linux-3.5.0-2.fc17.x86_64
File  Edit  View  Search  Terminal  Help
        System.map "/boot"
[hank@Maestro linux-3.5.0-2.fc17.x86_64]$ ls -l /boot/
total 90164
-rw-r--r--. 1 root root   115179 May  8 01:35 config-3.3.4-5.fc17.x86_64
-rw-r--r--. 1 root root   116714 Jun 19 03:58 config-3.4.3-1.fc17.x86_64
-rw-r--r--. 1 root root   118625 Jul 30 23:05 config-3.5.0-2.fc17.x86_64
drwxr-xr-x. 4 root root     1024 May 23 04:40 efi
-rw-r--r--. 1 root root   178436 Mar 27 17:11 elf-memtest86+-4.20
drwxr-xr-x. 2 root root     1024 May 23 04:40 grub
drwxr-xr-x. 6 root root     1024 Aug 10 21:19 grub2
-rw-rw-r--. 1 root root 17472729 Jun 22 12:00 initramfs-3.3.4-5.fc17.x86_64.img
-rw-r--r--. 1 root root 17772445 Jun 23 11:10 initramfs-3.4.3-1.fc17.x86_64.img
-rw-r--r--. 1 root root 17923915 Aug  3 08:51 initramfs-3.5.0-2.fc17.x86_64.img
-rw-rw-r--. 1 root root  9703029 Aug 10 21:19 initramfs-3.5.0_OS_COURSE_1234567.img
drwx------. 2 root root    12288 Jun 22 11:58 lost+found
-rw-r--r--. 1 root root   176760 Mar 27 17:11 memtest86+-4.20
lrwxrwxrwx. 1 root root       40 Aug 10 21:16 System.map -> /boot/System.map-3.5.0_OS_COURSE_12
34567
-rw-------. 1 root root  2412391 May  8 01:35 System.map-3.3.4-5.fc17.x86_64
-rw-------. 1 root root  2440456 Jun 19 03:58 System.map-3.4.3-1.fc17.x86_64
-rw-------. 1 root root  2466258 Jul 30 23:05 System.map-3.5.0-2.fc17.x86_64
-rw-rw-r--. 1 root root  2466258 Aug 10 21:16 System.map-3.5.0_OS_COURSE_1234567
lrwxrwxrwx. 1 root root       37 Aug 10 21:16 vmlinuz -> /boot/vmlinuz-3.5.0_OS_COURSE_1234567
-rwxr-xr-x. 1 root root  4662160 May  8 01:35 vmlinuz-3.3.4-5.fc17.x86_64
-rwxr-xr-x. 1 root root  4711472 Jun 19 03:58 vmlinuz-3.4.3-1.fc17.x86_64
-rwxr-xr-x. 1 root root  4772672 Jul 30 23:05 vmlinuz-3.5.0-2.fc17.x86_64
-rw-rw-r--. 1 root root  4772832 Aug 10 21:16 vmlinuz-3.5.0_OS_COURSE_1234567
[hank@Maestro linux-3.5.0-2.fc17.x86_64]$
```

Figure 7. kernel image installation location

M. Boot with the new kernel





Try this

Under Linux, you can delete a file by the *rm* or *unlink* commands (Figure 8). Assume that you have implanted a malware named '*sticky_malware*' on a machine. Now, you want the malware to stay forever on that machine so you have to make sure that the malware cannot be trivially removed by the rm or unlink command by the root (as shown in Figure 9). One way to achieve the effect is by tampering the kernel code to prevent the *unlink* and *unlinkat* system calls from operating on any file named '*sticky_malware*' (you can use strace or ltrace to trace the *rm* / *unlink* commands to confirm that they are indeed the underlying system calls for file deletion). The two system calls are defined in fs/name.c (Figure 10).

Please modify the kernel to achieve the effect of undeletable file for any file with the name '*sticky_malware*'. For the exercise, please submit a report detailing your modifications and also present the results demonstrating the effect of undeletable file (e.g. the screenshots from running your system with the tampered kernel).

```
File  Edit  View  Search  Terminal  Help
[hank@Maestro test]$ ls -al
total 8
drwxrwxr-x.  2 hank hank 4096 Aug 11 22:34 .
drwxrwxrwt. 56 root root 4096 Aug 11 22:34 ..
[hank@Maestro test]$ touch a.txt
[hank@Maestro test]$
[hank@Maestro test]$ ls
a.txt
[hank@Maestro test]$ rm a.txt
[hank@Maestro test]$
[hank@Maestro test]$ ls -al
total 8
drwxrwxr-x.  2 hank hank 4096 Aug 11 22:34 .
drwxrwxrwt. 56 root root 4096 Aug 11 22:34 ..
[hank@Maestro test]$
[hank@Maestro test]$
```

Figure 8. delete a file with *rm* command

```
hank@Maestro:/tmp/test
File  Edit  View  Search  Terminal  Help
[hank@Maestro test]$ ls -al
total 8
drwxrwxr-x.  2 hank hank 4096 Aug 11 22:39 .
drwxrwxrwt. 56 root root 4096 Aug 11 22:39 ..
-rw-rw-r--.  1 hank hank    0 Aug 11 22:39 sticky_malware
[hank@Maestro test]$
[hank@Maestro test]$ rm sticky_malware
rm: cannot remove `sticky_malware': Operation not permitted
[hank@Maestro test]$
[hank@Maestro test]$ unlink sticky_malware
unlink: cannot unlink `sticky_malware': Operation not permitted
[hank@Maestro test]$
[hank@Maestro test]$ ls -al
total 8
drwxrwxr-x.  2 hank hank 4096 Aug 11 22:39 .
drwxrwxrwt. 56 root root 4096 Aug 11 22:41 ..
-rw-rw-r--.  1 hank hank    0 Aug 11 22:39 sticky_malware
[hank@Maestro test]$
```

Figure 9. Undeletable *sticky_malware*

```
LXR linux/fs/namei.c
lxr.linux.no/linux+v3.5.1/fs/namei.c
2960            return error;
2961
2962    slashes:
2963            error = !dentry->d_inode ? -ENOENT :
2964                    S_ISDIR(dentry->d_inode->i_mode) ? -EISDIR : -ENOTDIR;
2965            goto exit2;
2966    }
2967
2968    SYSCALL_DEFINE3(unlinkat, int, dfd, const char __user *, pathname, int, flag)
2969    {
2970            if ((flag & ~AT_REMOVEDIR) != 0)
2971                    return -EINVAL;
2972
2973            if (flag & AT_REMOVEDIR)
2974                    return do_rmdir(dfd, pathname);
2975
2976            return do_unlinkat(dfd, pathname);
2977    }
2978
2979    SYSCALL_DEFINE1(unlink, const char __user *, pathname)
2980    {
2981            return do_unlinkat(AT_FDCWD, pathname);
2982    }
2983
2984    int vfs_symlink(struct inode *dir, struct dentry *dentry, const char *oldname)
2985    {
2986            int error = may_create(dir, dentry);
2987
2988            if (error)
2989                    return error;
2990
2991            if (!dir->i_op->symlink)
2992                    return -EPERM;
```

Figure 10. unlink / unlinkat system calls