LAB 4
LINUX KERNEL DEVELOPMENT

Fullname: Nguyen Chi Hoang Minh

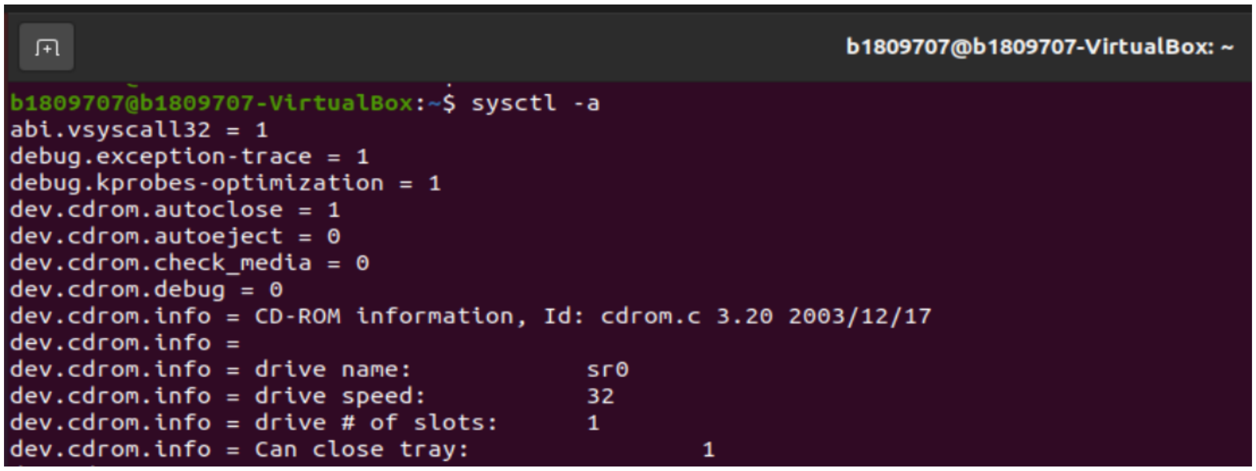
Student ID: B1809707

- Note: screenshots need to be clear and good-looking; submissions must be in PDF format.

1. Modify kernel parameters and install new modules

- List all linux kernel parameters on your OS:

```
sysctl -a
```



```
b1809707@b1809707-VirtualBox:~$ sysctl -a
abi.vsyscall32 = 1
debug.exception-trace = 1
debug.kprobes-optimization = 1
dev.cdrom.autoclose = 1
dev.cdrom.autoeject = 0
dev.cdrom.check_media = 0
dev.cdrom.debug = 0
dev.cdrom.info = CD-ROM information, Id: cdrom.c 3.20 2003/12/17
dev.cdrom.info =
dev.cdrom.info = drive name:             sr0
dev.cdrom.info = drive speed:            32
dev.cdrom.info = drive # of slots:       1
dev.cdrom.info = Can close tray:                 1
```

- List all available TCP congestion control algorithms:

```
sysctl net.ipv4.tcp_available_congestion_control
```

- Show which TCP congestion control algorithm is using:

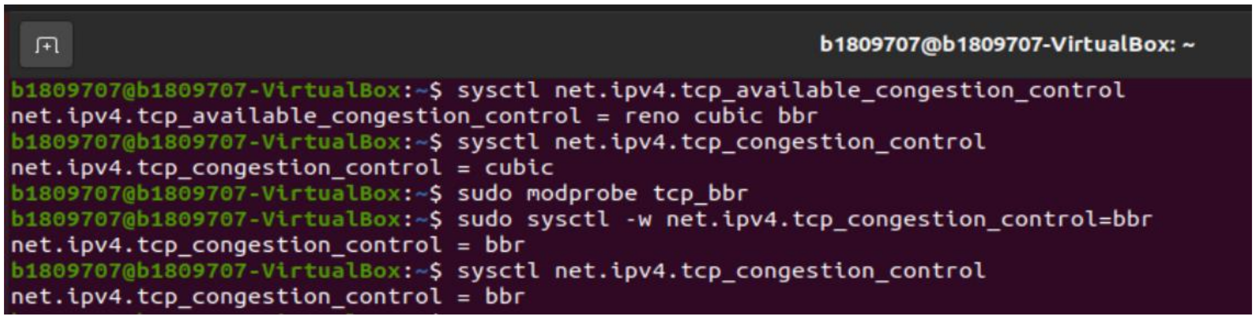
```
sysctl net.ipv4.tcp_congestion_control
```

- Install `bbr` TCP congestion control algorithm module:

```
sudo modprobe tcp_bbr
```

- Switch to the `bbr` TCP congestion control algorithm:

```
sudo sysctl -w net.ipv4.tcp_congestion_control=bbr
sysctl net.ipv4.tcp_congestion_control
```



```
b1809707@b1809707-VirtualBox:~$ sysctl net.ipv4.tcp_available_congestion_control
net.ipv4.tcp_available_congestion_control = reno cubic bbr
b1809707@b1809707-VirtualBox:~$ sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = cubic
b1809707@b1809707-VirtualBox:~$ sudo modprobe tcp_bbr
b1809707@b1809707-VirtualBox:~$ sudo sysctl -w net.ipv4.tcp_congestion_control=bbr
net.ipv4.tcp_congestion_control = bbr
b1809707@b1809707-VirtualBox:~$ sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = bbr
```

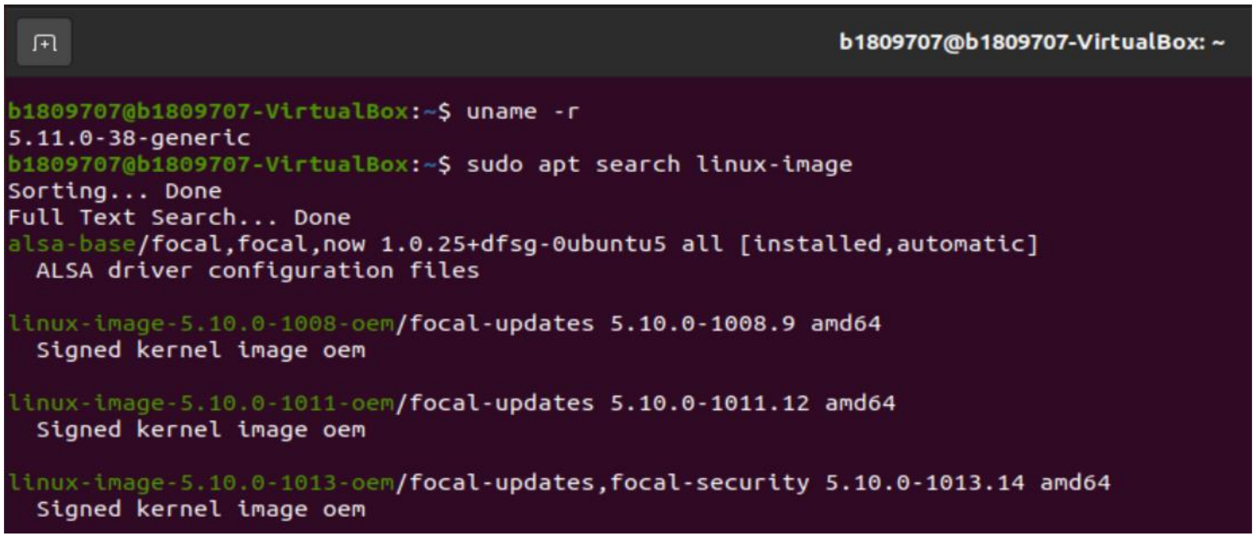
2. Install new kernel version

- Show your current kernel version:

```
uname -r
```

- Search for newer versions:

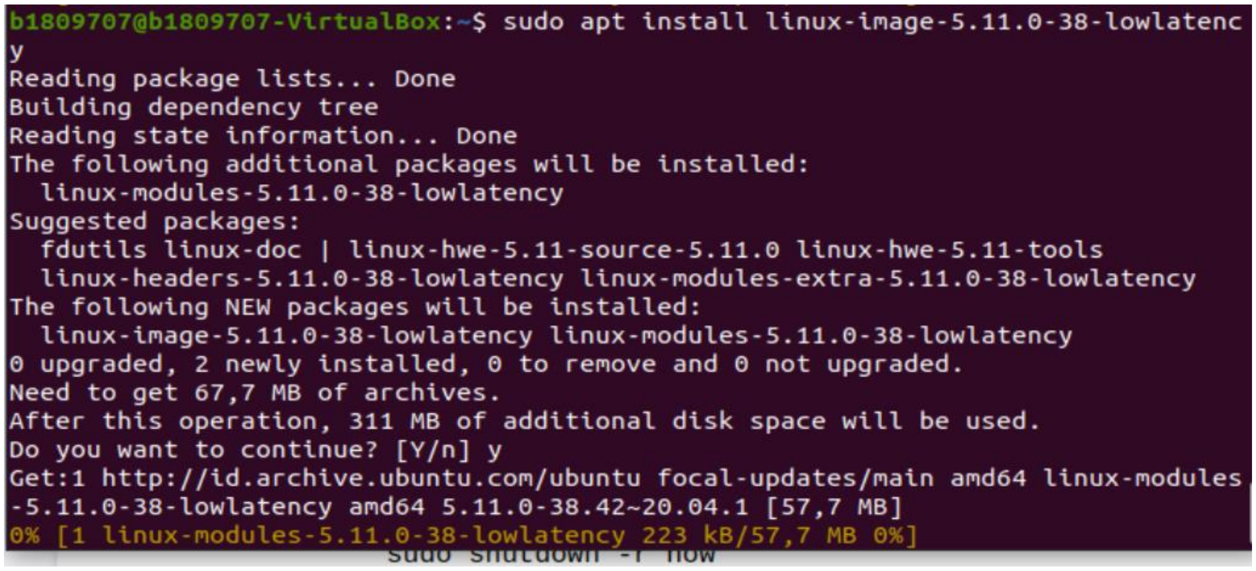
```
sudo apt search linux-image
```



```
b1809707@b1809707-VirtualBox:~$ uname -r
5.11.0-38-generic
b1809707@b1809707-VirtualBox:~$ sudo apt search linux-image
Sorting... Done
Full Text Search... Done
alsa-base/focal,focal,now 1.0.25+dfsg-0ubuntu5 all [installed,automatic]
  ALSA driver configuration files

linux-image-5.10.0-1008-oem/focal-updates 5.10.0-1008.9 amd64
  Signed kernel image oem

linux-image-5.10.0-1011-oem/focal-updates 5.10.0-1011.12 amd64
  Signed kernel image oem

linux-image-5.10.0-1013-oem/focal-updates,focal-security 5.10.0-1013.14 amd64
  Signed kernel image oem
```

- Install the latest version you find:

```
sudo apt install linux-image-x.x.x-x-generic
```



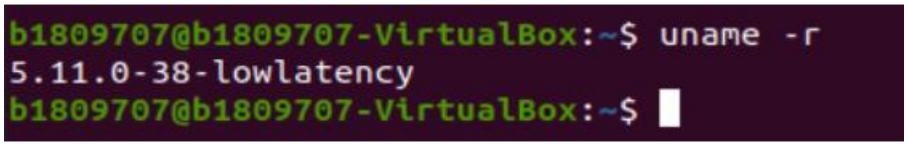
```
b1809707@b1809707-VirtualBox:~$ sudo apt install linux-image-5.11.0-38-lowlatenc
y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  linux-modules-5.11.0-38-lowlatency
Suggested packages:
  fdutils linux-doc | linux-hwe-5.11-source-5.11.0 linux-hwe-5.11-tools
  linux-headers-5.11.0-38-lowlatency linux-modules-extra-5.11.0-38-lowlatency
The following NEW packages will be installed:
  linux-image-5.11.0-38-lowlatency linux-modules-5.11.0-38-lowlatency
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 67,7 MB of archives.
After this operation, 311 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://id.archive.ubuntu.com/ubuntu focal-updates/main amd64 linux-modules
-5.11.0-38-lowlatency amd64 5.11.0-38.42~20.04.1 [57,7 MB]
0% [1 linux-modules-5.11.0-38-lowlatency 223 kB/57,7 MB 0%]
```

- After a kernel upgrade, you must reboot the system. Then, if the device driver you need is in the latest kernel, your hardware will work as expected:

```
sudo shutdown -r now
```

- Show your new current kernel version:

```
uname -r
```



```
b1809707@b1809707-VirtualBox:~$ uname -r
5.11.0-38-lowlatency
b1809707@b1809707-VirtualBox:~$
```

3. Build and install a new kernel version

- Get your system ready

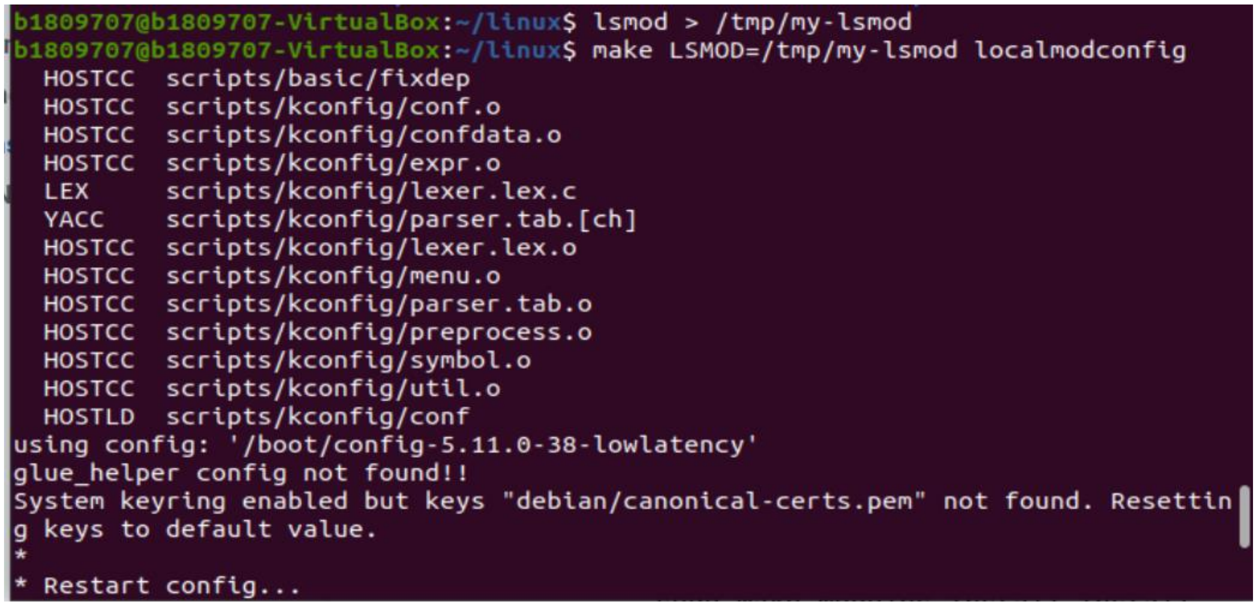
```
sudo apt update
sudo apt-get install build-essential vim git cscope libncurses-dev libssl-dev bison flex libelf-dev bc git-email -y
```

- Clone a mainline kernel source code to your computer:

```
git clone --depth=1 \
https://github.com/torvalds/linux.git
```

- To save time, just create a configuration file based on the list of modules currently loaded on your system (choose default values for other options).

```
lsmod > /tmp/my-lsmod
make LSMOD=/tmp/my-lsmod localmodconfig
```



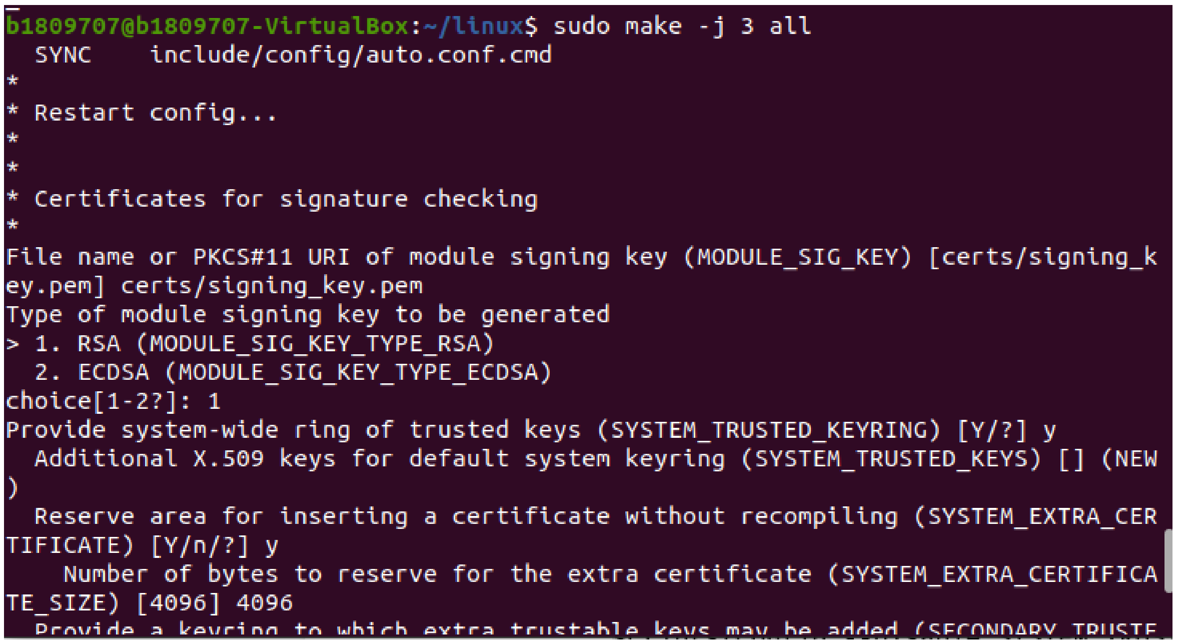
```
b1809707@b1809707-VirtualBox:~/linux$ lsmod > /tmp/my-lsmod
b1809707@b1809707-VirtualBox:~/linux$ make LSMOD=/tmp/my-lsmod localmodconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  HOSTCC  scripts/kconfig/confdata.o
  HOSTCC  scripts/kconfig/expr.o
  LEX     scripts/kconfig/lexer.lex.c
  YACC    scripts/kconfig/parser.tab.[ch]
  HOSTCC  scripts/kconfig/lexer.lex.o
  HOSTCC  scripts/kconfig/menu.o
  HOSTCC  scripts/kconfig/parser.tab.o
  HOSTCC  scripts/kconfig/preprocess.o
  HOSTCC  scripts/kconfig/symbol.o
  HOSTCC  scripts/kconfig/util.o
  HOSTLD  scripts/kconfig/conf
using config: '/boot/config-5.11.0-38-lowlatency'
glue_helper config not found!!
System keyring enabled but keys "debian/canonical-certs.pem" not found. Resettin
g keys to default value.
*
* Restart config...
```

- Disable certificate stuff:

```
scripts/config --disable SYSTEM_TRUSTED_KEYS
scripts/config --disable SYSTEM_REVOCATION_KEYS
```

- Compile the kernel. The process takes about 1 hour, please be patient and enjoy a cup of coffee. It has been tested successfully on Lubuntu 20.04, if any errors occur, please try to fix them by yourself.

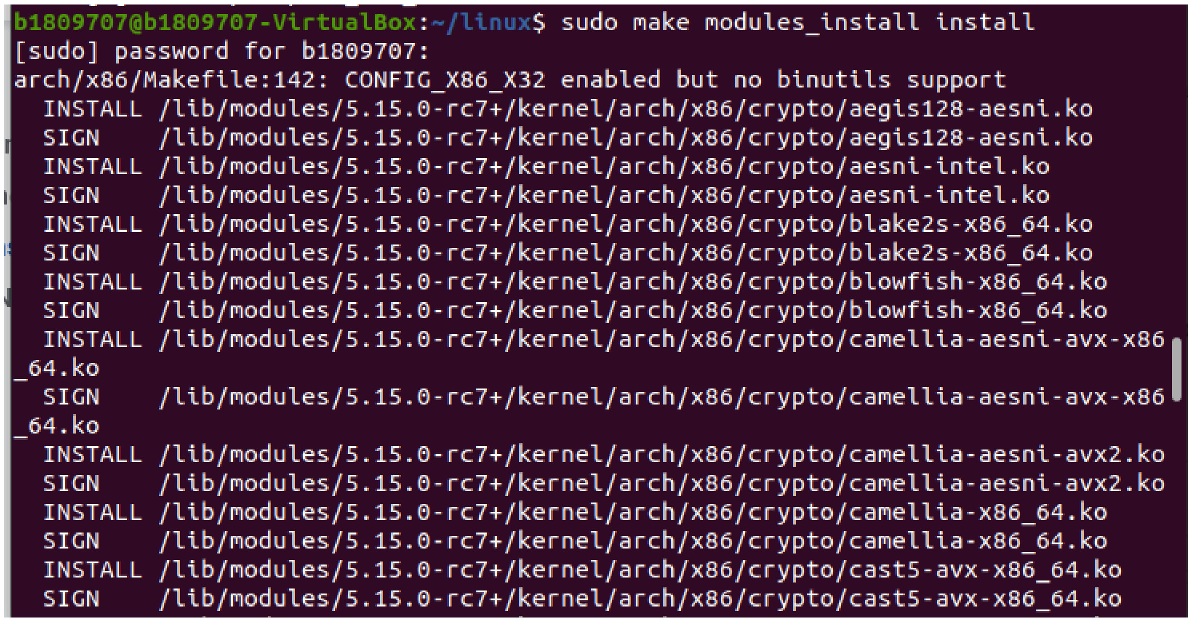
```
make -j3 all
```



```
b1809707@b1809707-VirtualBox:~/linux$ sudo make -j 3 all
  SYNC    include/config/auto.conf.cmd
*
* Restart config...
*
*
* Certificates for signature checking
*
File name or PKCS#11 URI of module signing key (MODULE_SIG_KEY) [certs/signing_k
ey.pem] certs/signing_key.pem
Type of module signing key to be generated
> 1. RSA (MODULE_SIG_KEY_TYPE_RSA)
  2. ECDSA (MODULE_SIG_KEY_TYPE_ECDSA)
choice[1-2?]: 1
Provide system-wide ring of trusted keys (SYSTEM_TRUSTED_KEYRING) [Y/?] y
  Additional X.509 keys for default system keyring (SYSTEM_TRUSTED_KEYS) [] (NEW
)
  Reserve area for inserting a certificate without recompiling (SYSTEM_EXTRA_CER
TIFICATE) [Y/n/?] y
    Number of bytes to reserve for the extra certificate (SYSTEM_EXTRA_CERTIFICA
TE_SIZE) [4096] 4096
  Provide a keyring to which extra trustable keys may be added (SECONDARY_TRUSTE
```

- Install the new kernel:

```
sudo make modules_install install
```



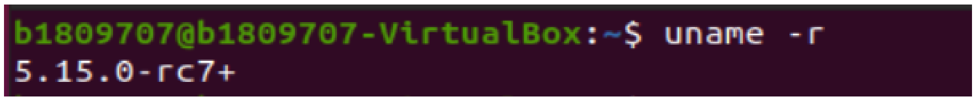
```
b1809707@b1809707-VirtualBox:~/linux$ sudo make modules_install install
[sudo] password for b1809707:
arch/x86/Makefile:142: CONFIG_X86_X32 enabled but no binutils support
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/aegis128-aesni.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/aegis128-aesni.ko
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/aesni-intel.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/aesni-intel.ko
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/blake2s-x86_64.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/blake2s-x86_64.ko
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/blowfish-x86_64.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/blowfish-x86_64.ko
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/camellia-aesni-avx-x86
_64.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/camellia-aesni-avx-x86
_64.ko
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/camellia-aesni-avx2.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/camellia-aesni-avx2.ko
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/camellia-x86_64.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/camellia-x86_64.ko
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/cast5-avx-x86_64.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/cast5-avx-x86_64.ko
```

- Now it is time to reboot the system to boot the newly installed kernel:

```
sudo shutdown -r now
```

- Show your new current kernel version:

```
uname -r
```



```
b1809707@b1809707-VirtualBox:~$ uname -r
5.15.0-rc7+
```

4. Writing Your First Kernel Patch

- Creating a new branch in the linux_mainline repository (has been cloned in exercise 3)

```
git checkout -b first-patch
```

- Update the kernel

```
git fetch origin
```

- Run `lsmod` to see the modules loaded on your system, and pick a driver to change. One driver that's included in all VM images is the `e1000` driver, the Intel ethernet driver, or you can choose another driver depending on your working environment.

- Run `git grep` to look for `e1000` files

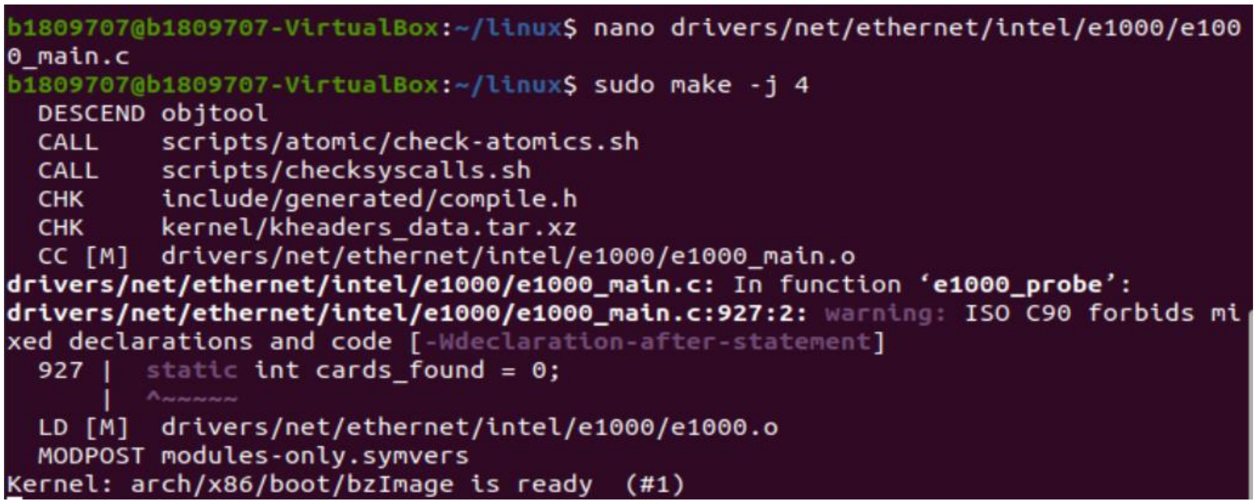
```
git grep e1000 -- '*Makefile'
```

- Make a small change to the probe function of the e1000 driver

```
nano drivers/net/ethernet/intel/e1000/e1000_main.c
# Add a line of code as below
static int e1000_probe(struct pci_dev *pdev, const struct pci_device_id *ent) {
...
struct e1000_hw *hw;
printk(KERN_DEBUG "I can modify the Linux kernel!\n");
static int cards_found = 0;
...
```

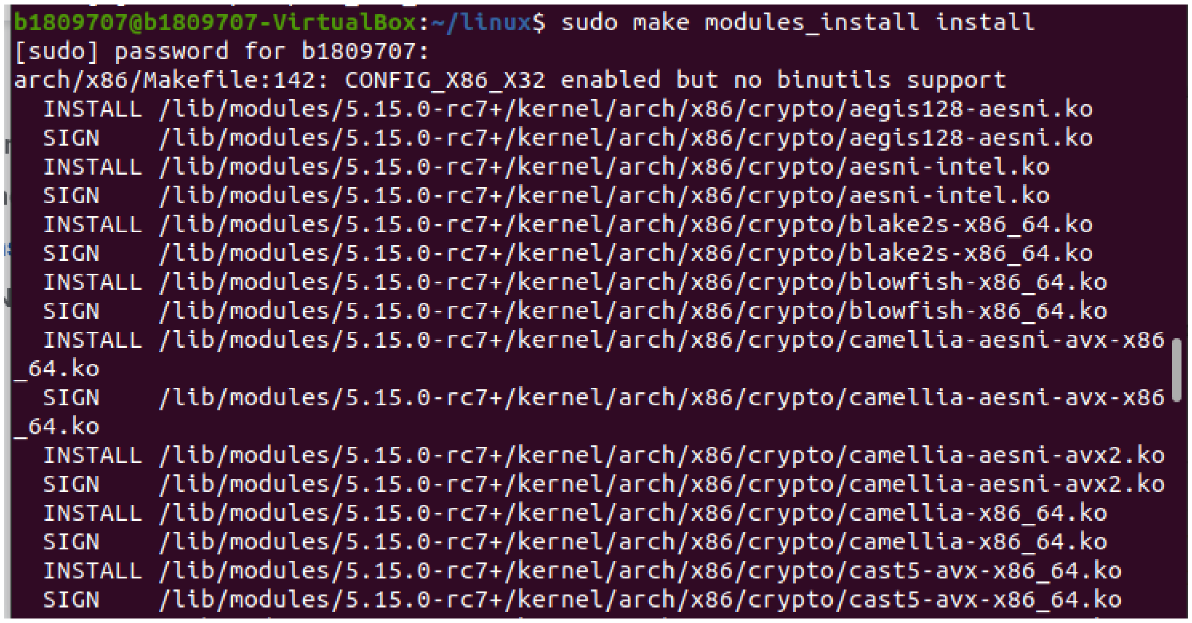
- Compile and install your changes:

```
make -j3
```



```
b1809707@b1809707-VirtualBox:~/linux$ nano drivers/net/ethernet/intel/e1000/e100
0_main.c
b1809707@b1809707-VirtualBox:~/linux$ sudo make -j 4
  DESCEND objtool
  CALL    scripts/atomic/check-atomics.sh
  CALL    scripts/checksyscalls.sh
  CHK     include/generated/compile.h
  CHK     kernel/kheaders_data.tar.xz
  CC [M]  drivers/net/ethernet/intel/e1000/e1000_main.o
drivers/net/ethernet/intel/e1000/e1000_main.c: In function ‘e1000_probe’:
drivers/net/ethernet/intel/e1000/e1000_main.c:927:2: warning: ISO C90 forbids mi
xed declarations and code [-Wdeclaration-after-statement]
  927 |  static int cards_found = 0;
      |  ^~~~~~
  LD [M]  drivers/net/ethernet/intel/e1000/e1000.o
  MODPOST modules-only.symvers
Kernel: arch/x86/boot/bzImage is ready  (#1)
```

```
sudo make modules_install install
```



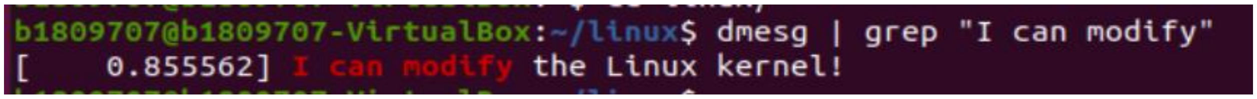
```
b1809707@b1809707-VirtualBox:~/linux$ sudo make modules_install install
[sudo] password for b1809707:
arch/x86/Makefile:142: CONFIG_X86_X32 enabled but no binutils support
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/aegis128-aesni.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/aegis128-aesni.ko
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/aesni-intel.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/aesni-intel.ko
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/blake2s-x86_64.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/blake2s-x86_64.ko
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/blowfish-x86_64.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/blowfish-x86_64.ko
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/camellia-aesni-avx-x86
_64.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/camellia-aesni-avx-x86
_64.ko
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/camellia-aesni-avx2.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/camellia-aesni-avx2.ko
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/camellia-x86_64.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/camellia-x86_64.ko
  INSTALL /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/cast5-avx-x86_64.ko
  SIGN    /lib/modules/5.15.0-rc7+/kernel/arch/x86/crypto/cast5-avx-x86_64.ko
```

- Reboot the system:

```
sudo shutdown -r now
```

- Show kernel buffer log:

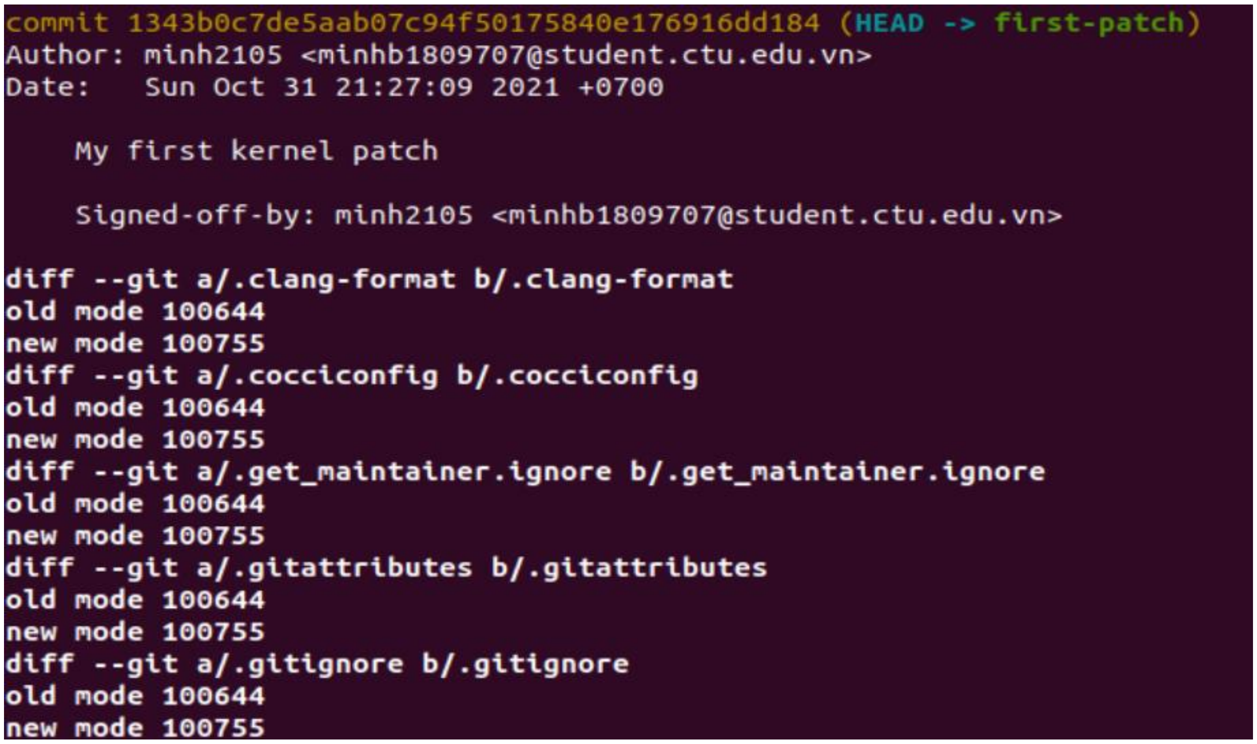
```
    dmesg | less
    # Search for your printk in the log file by typing "/I
can modify"
```



```
b1809707@b1809707-VirtualBox:~/linux$ dmesg | grep "I can modify"
[    0.855562] I can modify the Linux kernel!
```

- Committing changes, and view your commit

```
    git add .
    git commit -s -v -m "My first kernel patch"
    git show HEAD
```



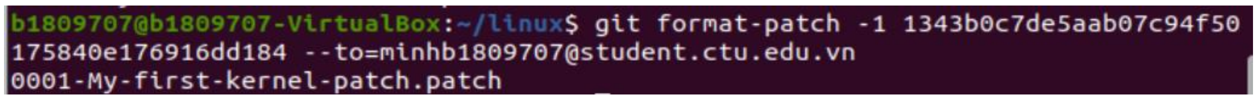
```
commit 1343b0c7de5aab07c94f50175840e176916dd184 (HEAD -> first-patch)
Author: minh2105 <minhb1809707@student.ctu.edu.vn>
Date:   Sun Oct 31 21:27:09 2021 +0700

    My first kernel patch

    Signed-off-by: minh2105 <minhb1809707@student.ctu.edu.vn>

diff --git a/.clang-format b/.clang-format
old mode 100644
new mode 100755
diff --git a/.cocciconfig b/.cocciconfig
old mode 100644
new mode 100755
diff --git a/.get_maintainer.ignore b/.get_maintainer.ignore
old mode 100644
new mode 100755
diff --git a/.gitattributes b/.gitattributes
old mode 100644
new mode 100755
diff --git a/.gitignore b/.gitignore
old mode 100644
new mode 100755
```

- Find whom to send the patch to

```
    git show HEAD | scripts/get_maintainer.pl
```

- Create a patch

```
    git format-patch -1 <commit ID> --to=<your email> Note:
    Please do not send your patch to a maintainer, send it
    to yourself instead.
```



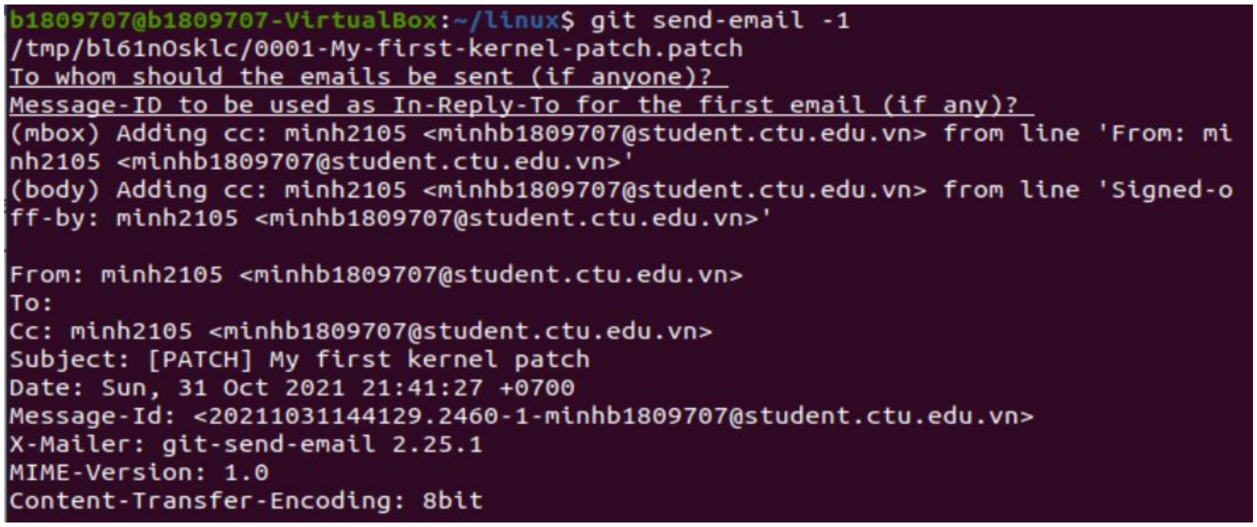
```
b1809707@b1809707-VirtualBox:~/linux$ git format-patch -1 1343b0c7de5aab07c94f50
175840e176916dd184 --to=minhb1809707@student.ctu.edu.vn
0001-My-first-kernel-patch.patch
```

- Modify `./git/config` file to configure send-email

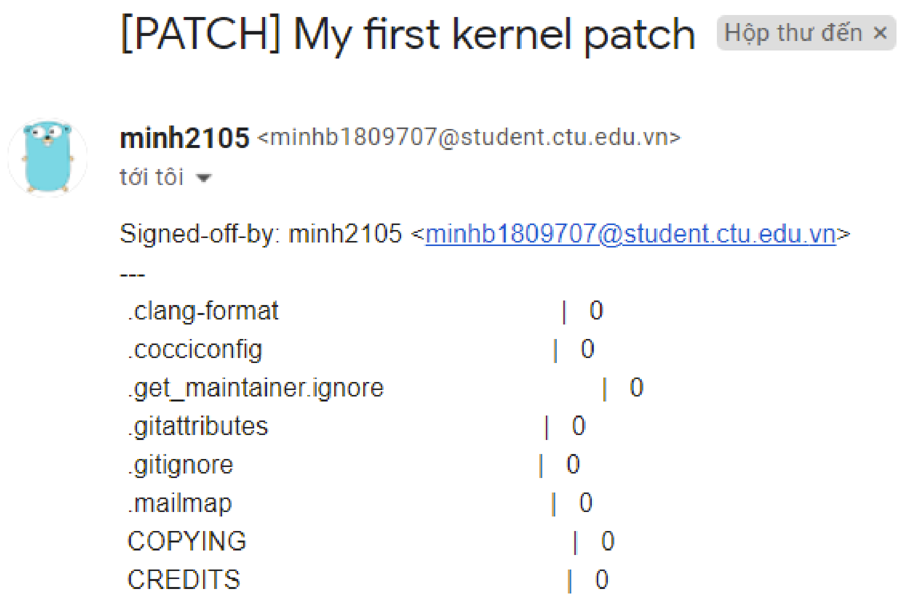
```
 #.git/config
[sendemail]
    smtpserver = smtp.googlemail.com
    smtpencryption = tls
    smtpserverport = 587
    smtpuser = your gmail address (CTU student email is
OK
```

- Send the patch

```
git send-email <patch_file>
```



```
b1809707@b1809707-VirtualBox:~/linux$ git send-email -1
/tmp/bl61nOsklc/0001-My-first-kernel-patch.patch
To whom should the emails be sent (if anyone)?
Message-ID to be used as In-Reply-To for the first email (if any)?
(mbox) Adding cc: minh2105 <minhb1809707@student.ctu.edu.vn> from line 'From: mi
nh2105 <minhb1809707@student.ctu.edu.vn>'
(body) Adding cc: minh2105 <minhb1809707@student.ctu.edu.vn> from line 'Signed-o
ff-by: minh2105 <minhb1809707@student.ctu.edu.vn>'

From: minh2105 <minhb1809707@student.ctu.edu.vn>
To:
Cc: minh2105 <minhb1809707@student.ctu.edu.vn>
Subject: [PATCH] My first kernel patch
Date: Sun, 31 Oct 2021 21:41:27 +0700
Message-Id: <20211031144129.2460-1-minhb1809707@student.ctu.edu.vn>
X-Mailer: git-send-email 2.25.1
MIME-Version: 1.0
Content-Transfer-Encoding: 8bit
```



[PATCH] My first kernel patch Hộp thư đến ×

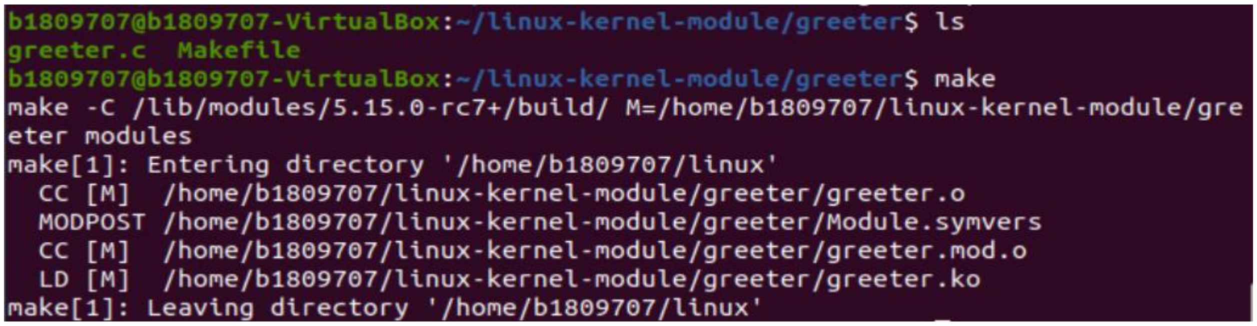
minh2105 <minhb1809707@student.ctu.edu.vn>
tới tôi

```
Signed-off-by: minh2105 <minhb1809707@student.ctu.edu.vn>
---
 .clang-format                | 0
 .cocciconfig                 | 0
 .get_maintainer.ignore       | 0
 .gitattributes               | 0
 .gitignore                   | 0
 .mailmap                     | 0
 COPYING                      | 0
 CREDITS                      | 0
```

5. Writing a simple Linux kernel module: Greeter sample

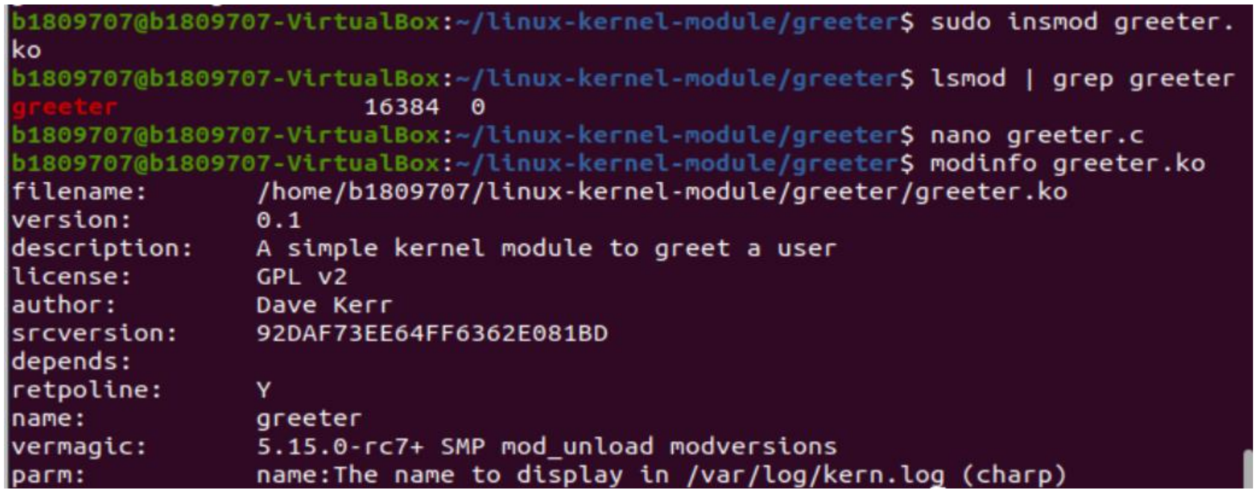
This module simply takes a name as a parameter, and writes a greeting to the kernel log (/var/log/kern.log):

- Clone this repository to your computer:
  https://github.com/TuanThai/linux-kernel-module.git
- Move into `greeter/` directory.
- Build the module using `make` command. The module is compiled to `greeter.ko`



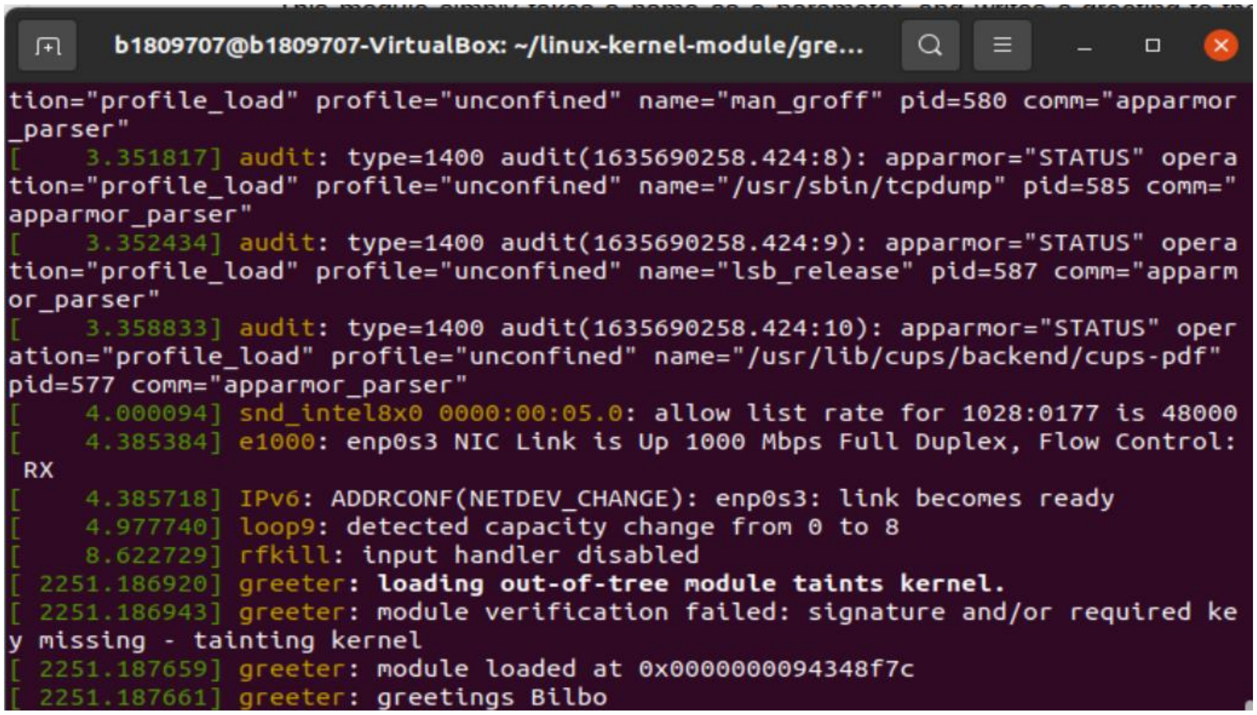
```
b1809707@b1809707-VirtualBox:~/linux-kernel-module/greeter$ ls
greeter.c  Makefile
b1809707@b1809707-VirtualBox:~/linux-kernel-module/greeter$ make
make -C /lib/modules/5.15.0-rc7+/build/ M=/home/b1809707/linux-kernel-module/gre
eter modules
make[1]: Entering directory '/home/b1809707/linux'
  CC [M]  /home/b1809707/linux-kernel-module/greeter/greeter.o
  MODPOST /home/b1809707/linux-kernel-module/greeter/Module.symvers
  CC [M]  /home/b1809707/linux-kernel-module/greeter/greeter.mod.o
  LD [M]  /home/b1809707/linux-kernel-module/greeter/greeter.ko
make[1]: Leaving directory '/home/b1809707/linux'
```

- Install the module using `insmod greeter.ko` command, then show that the module has been installed using `lsmod | grep greeter` command
- Show the information of the module using `modinfo greeter.ko`



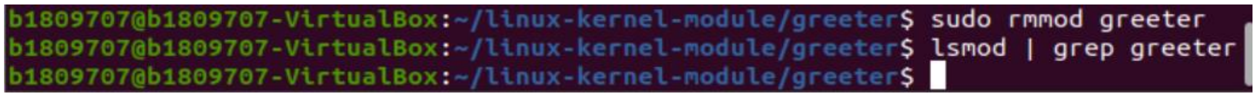
```
b1809707@b1809707-VirtualBox:~/linux-kernel-module/greeter$ sudo insmod greeter.
ko
b1809707@b1809707-VirtualBox:~/linux-kernel-module/greeter$ lsmod | grep greeter
greeter                16384  0
b1809707@b1809707-VirtualBox:~/linux-kernel-module/greeter$ nano greeter.c
b1809707@b1809707-VirtualBox:~/linux-kernel-module/greeter$ modinfo greeter.ko
filename:       /home/b1809707/linux-kernel-module/greeter/greeter.ko
version:        0.1
description:    A simple kernel module to greet a user
license:        GPL v2
author:         Dave Kerr
srcversion:     92DAF73EE64FF6362E081BD
depends:
retpoline:      Y
name:           greeter
vermagic:       5.15.0-rc7+ SMP mod_unload modversions
parm:           name:The name to display in /var/log/kern.log (charp)
```

- Show kernel log with `dmesg`



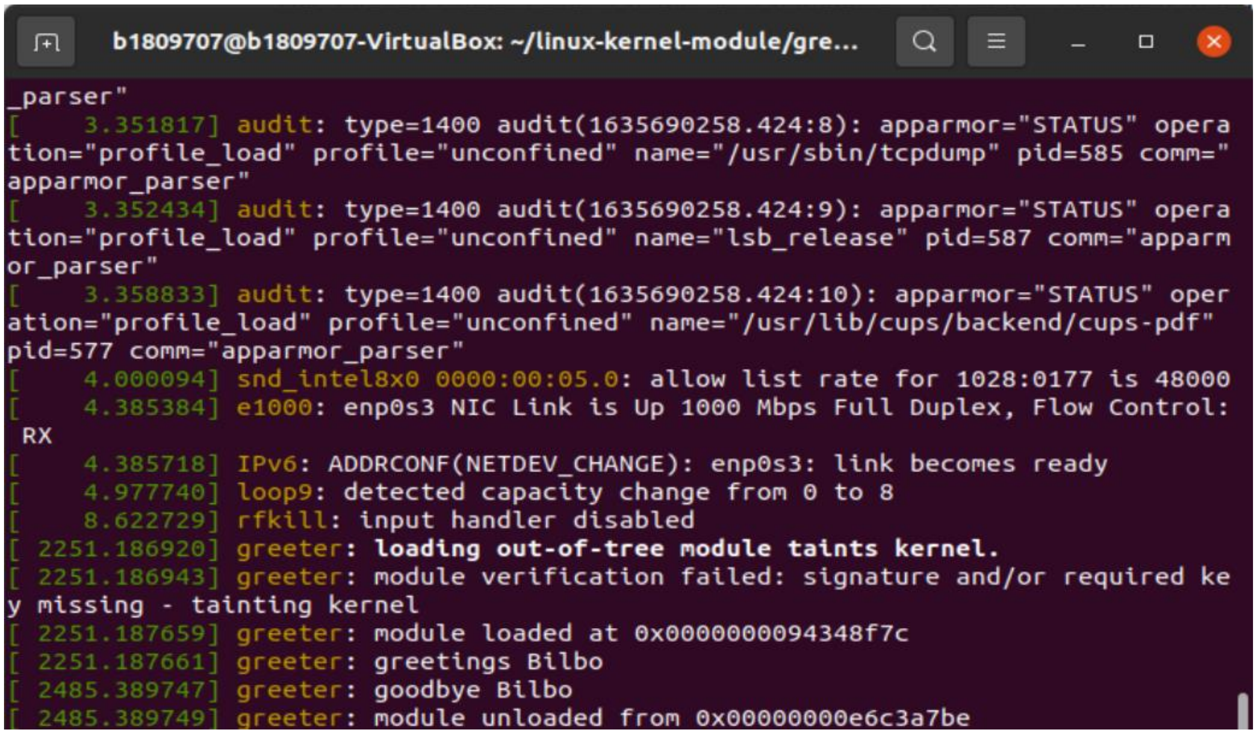
```
tion="profile_load" profile="unconfined" name="man_groff" pid=580 comm="apparmor
_parser"
[    3.351817] audit: type=1400 audit(1635690258.424:8): apparmor="STATUS" opera
tion="profile_load" profile="unconfined" name="/usr/sbin/tcpdump" pid=585 comm="
apparmor_parser"
[    3.352434] audit: type=1400 audit(1635690258.424:9): apparmor="STATUS" opera
tion="profile_load" profile="unconfined" name="lsb_release" pid=587 comm="apparm
or_parser"
[    3.358833] audit: type=1400 audit(1635690258.424:10): apparmor="STATUS" oper
ation="profile_load" profile="unconfined" name="/usr/lib/cups/backend/cups-pdf"
pid=577 comm="apparmor_parser"
[    4.000094] snd_intel8x0 0000:00:05.0: allow list rate for 1028:0177 is 48000
[    4.385384] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 RX
[    4.385718] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[    4.977740] loop9: detected capacity change from 0 to 8
[    8.622729] rfkill: input handler disabled
[ 2251.186920] greeter: loading out-of-tree module taints kernel.
[ 2251.186943] greeter: module verification failed: signature and/or required ke
y missing - tainting kernel
[ 2251.187659] greeter: module loaded at 0x0000000094348f7c
[ 2251.187661] greeter: greetings Bilbo
```

- Remove the module using `rmmod greeter.ko` command, then show that the module has been removed using `lsmod | grep greeter` command.



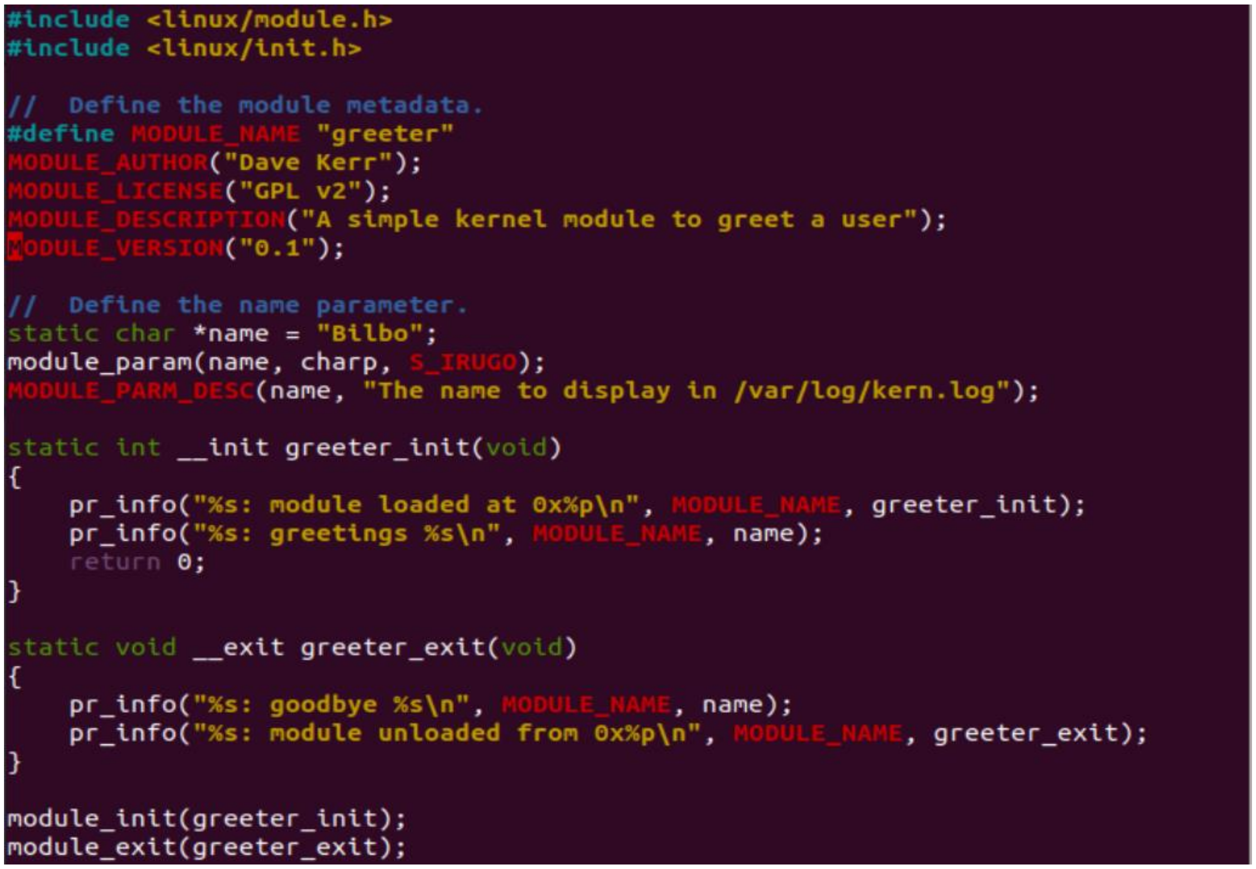
```
b1809707@b1809707-VirtualBox:~/linux-kernel-module/greeter$ sudo rmmod greeter
b1809707@b1809707-VirtualBox:~/linux-kernel-module/greeter$ lsmod | grep greeter
b1809707@b1809707-VirtualBox:~/linux-kernel-module/greeter$
```

- Show kernel log with `dmesg`



```
b1809707@b1809707-VirtualBox: ~/linux-kernel-module/gre...
_parser"
[    3.351817] audit: type=1400 audit(1635690258.424:8): apparmor="STATUS" opera
tion="profile_load" profile="unconfined" name="/usr/sbin/tcpdump" pid=585 comm="
apparmor_parser"
[    3.352434] audit: type=1400 audit(1635690258.424:9): apparmor="STATUS" opera
tion="profile_load" profile="unconfined" name="lsb_release" pid=587 comm="apparm
or_parser"
[    3.358833] audit: type=1400 audit(1635690258.424:10): apparmor="STATUS" oper
ation="profile_load" profile="unconfined" name="/usr/lib/cups/backend/cups-pdf"
pid=577 comm="apparmor_parser"
[    4.000094] snd_intel8x0 0000:00:05.0: allow list rate for 1028:0177 is 48000
[    4.385384] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 RX
[    4.385718] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[    4.977740] loop9: detected capacity change from 0 to 8
[    8.622729] rfkill: input handler disabled
[ 2251.186920] greeter: loading out-of-tree module taints kernel.
[ 2251.186943] greeter: module verification failed: signature and/or required ke
y missing - tainting kernel
[ 2251.187659] greeter: module loaded at 0x0000000094348f7c
[ 2251.187661] greeter: greetings Bilbo
[ 2485.389747] greeter: goodbye Bilbo
[ 2485.389749] greeter: module unloaded from 0x00000000e6c3a7be
```

- Move to `greeter.c` file, then briefly explain below functions:



```c
#include <linux/module.h>
#include <linux/init.h>

//  Define the module metadata.
#define MODULE_NAME "greeter"
MODULE_AUTHOR("Dave Kerr");
MODULE_LICENSE("GPL v2");
MODULE_DESCRIPTION("A simple kernel module to greet a user");
MODULE_VERSION("0.1");

//  Define the name parameter.
static char *name = "Bilbo";
module_param(name, charp, S_IRUGO);
MODULE_PARM_DESC(name, "The name to display in /var/log/kern.log");

static int __init greeter_init(void)
{
    pr_info("%s: module loaded at 0x%p\n", MODULE_NAME, greeter_init);
    pr_info("%s: greetings %s\n", MODULE_NAME, name);
    return 0;
}

static void __exit greeter_exit(void)
{
    pr_info("%s: goodbye %s\n", MODULE_NAME, name);
    pr_info("%s: module unloaded from 0x%p\n", MODULE_NAME, greeter_exit);
}

module_init(greeter_init);
module_exit(greeter_exit);
```

greeter_init. The __init macro causes the init function to be discarded and its memory(vmalloc) freed once the init function finishes for built-in drivers.

greeter_exit. The __exit macro causes the omission of the function when the module is built into the kernel.

module_init(greeter_init). The module_init() macro defines which function is to be called at module insertion time (if the file is compiled as a module), or at boot time.

module_exit(greeter_exit). This macro defines the function to be called at module removal time (or never, in the case of the file compiled into the kernel).

---END---