ASTANA IT UNIVERSITY

Operating System Consepts

Final Exam

#	Full Name	Group
1	Isagalieva Aruzhan	CS2127
2	Akhatova Ayazhan	CS2127
3	Miyatbek Merey	CS2127
4	Nazarova Akbota	CS2127
5	Soltanbekova Tolganay	CS2127

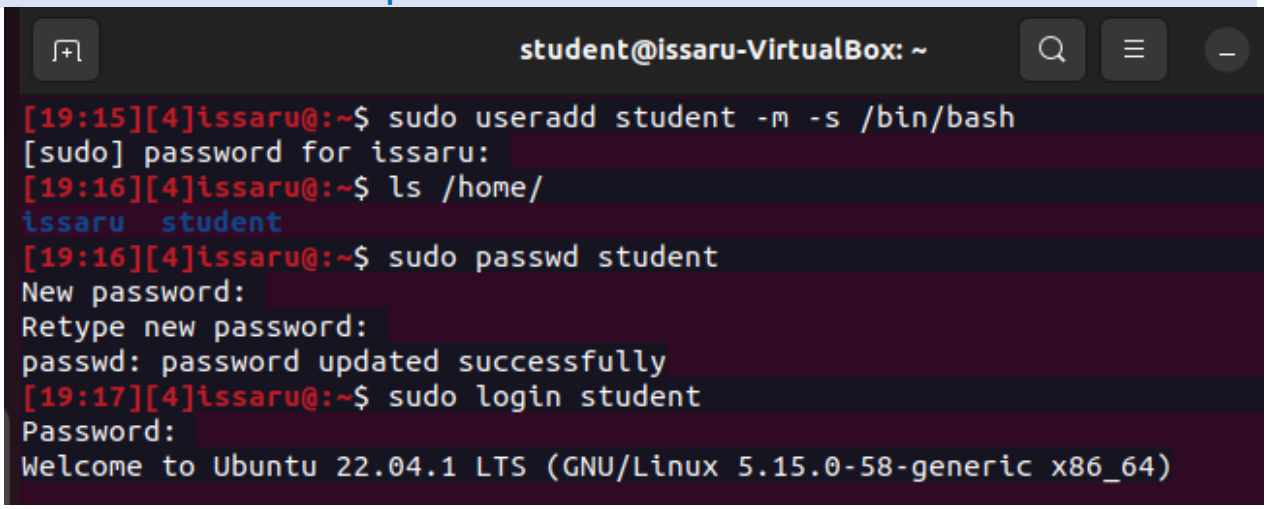
Link to the repository: https://github.com/issaru11/operating-systems-final-cs27

Step-by-step task completion:

Task 1: Here are the steps to add a user named "student" to the Linux system:

1. Type the following command to add a new user named "student":
   sudo useradd student –m –s /bin/bash
2. Set a password for the new user by the following command:
   sudo passwd student
3. Enter a password of at least 5 characters
4. Log in as the new user by running the following command:
   sudo login student
5. Verify that you are logged in as the "student" user by running the following command:
   whoami
6. To log out of the "student" user account, simply type the following command:
   logout

Screenshots of the code compilation result:



```
student@issaru-VirtualBox:~$ whoami
student
student@issaru-VirtualBox:~$ logout
[19:30][4]issaru@:~$
```

Task 2: On virtual machine, implement all three methods of connecting to the Internet (a "Direct" IP connection to the Internet, a connection via NAT and a connection via a proxy server)

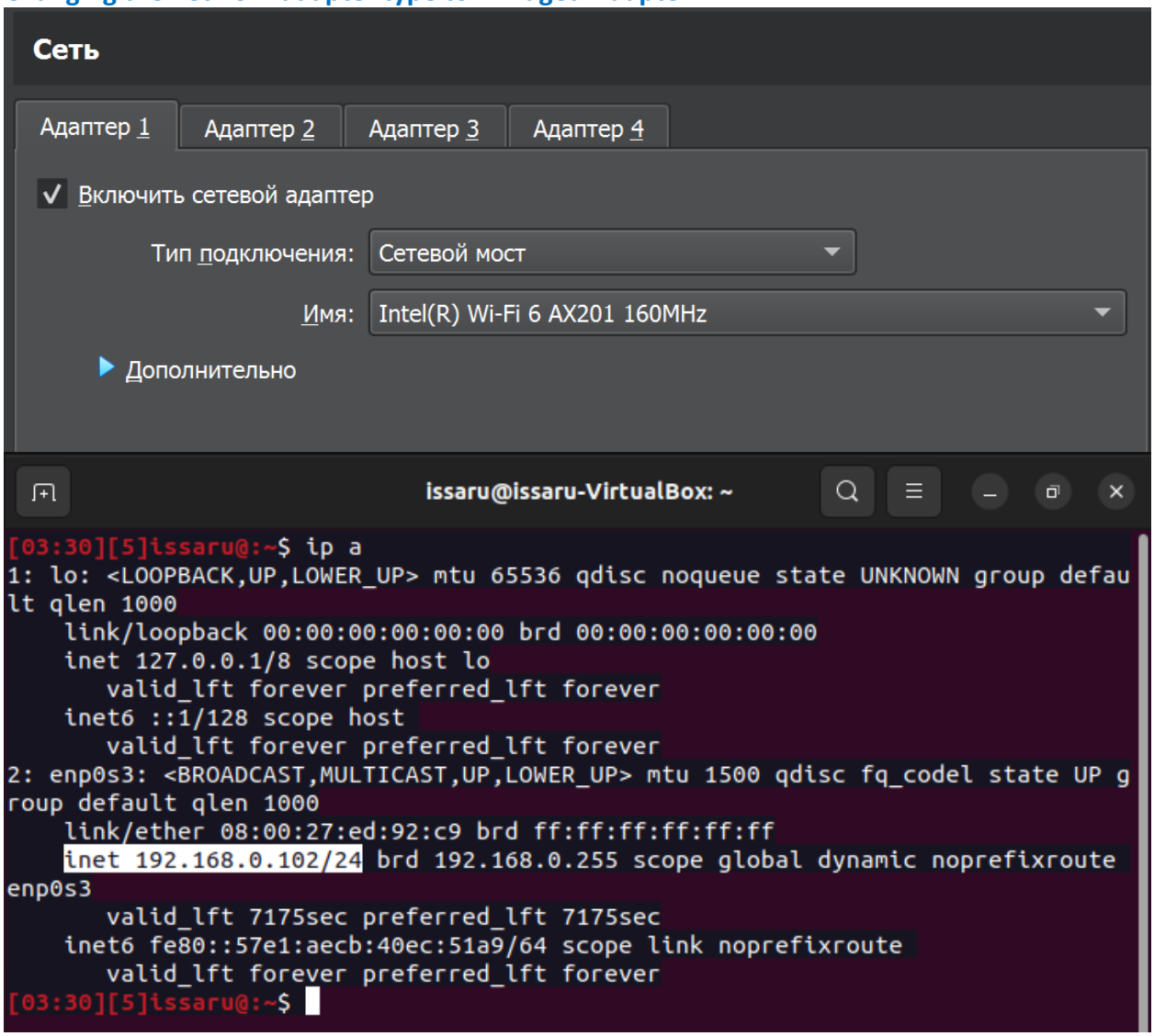
Here are the steps to implement a **"direct" IP connection** to the Internet on an Ubuntu virtual machine:

1. Make sure that your virtual machine is powered off.
2. Open the settings of the virtual machine.
3. Select the "Network" section and change the network adapter type to "Bridged Adapter". This will allow the virtual machine to use the host machine's physical network adapter and connect directly to the Internet.
4. Save the changes and start the virtual machine.
5. Check your network configuration by running the following command:
   ip a
6. Install network-manager with DHCP server.
7. Configure DHCP Client Instance to Obtain a Static IP Address.
8. Restarting Network Manager service by the following command:
   sudo systemctl restart network-manager
9. Check the connection by pinging some services.

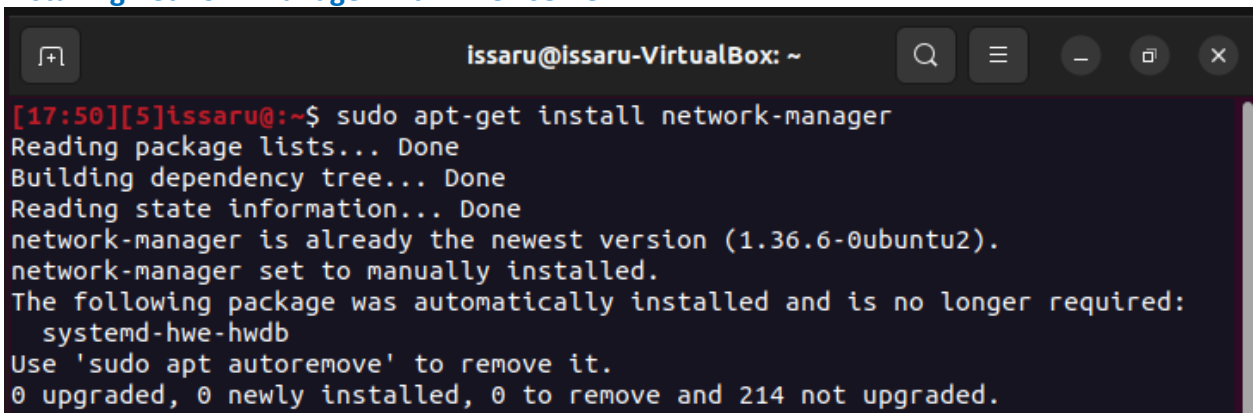
Screenshots of the code compilation result:

DIRECT IP CONNECTION:

Changing the network adapter type to "Bridged Adapter".



Installing network-manager with DHCP server.



```
[17:51][5]issaru@:~$ sudo apt-get install isc-dhcp-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  systemd-hwe-hwdb
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  libirs-export161 libisccfg-export163
Suggested packages:
  isc-dhcp-server-ldap policycoreutils
The following NEW packages will be installed:
  isc-dhcp-server libirs-export161 libisccfg-export163
0 upgraded, 3 newly installed, 0 to remove and 214 not upgraded.
Need to get 528 kB of archives.
After this operation, 1 546 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://kz.archive.ubuntu.com/ubuntu jammy/main amd64 libisccfg-export163
amd64 1:9.11.19+dfsg-2.1ubuntu3 [53,0 kB]
Get:2 http://kz.archive.ubuntu.com/ubuntu jammy/main amd64 libirs-export161 amd
```

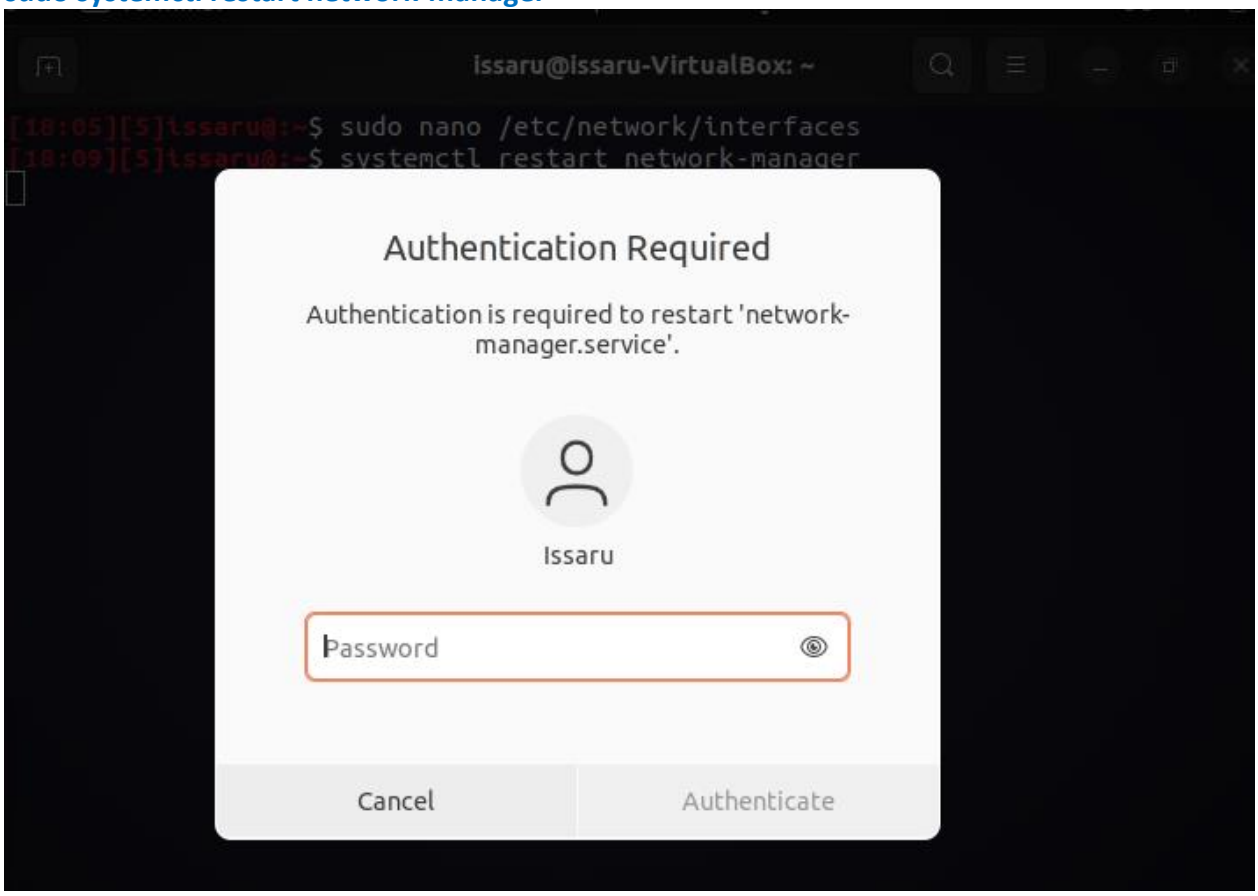
Configuring DHCP Client Instance to Obtain a Static IP Address:

```
issaru@issaru-VirtualBox: ~
[18:05][5]issaru@:~$ sudo nano /etc/network/interfaces
```

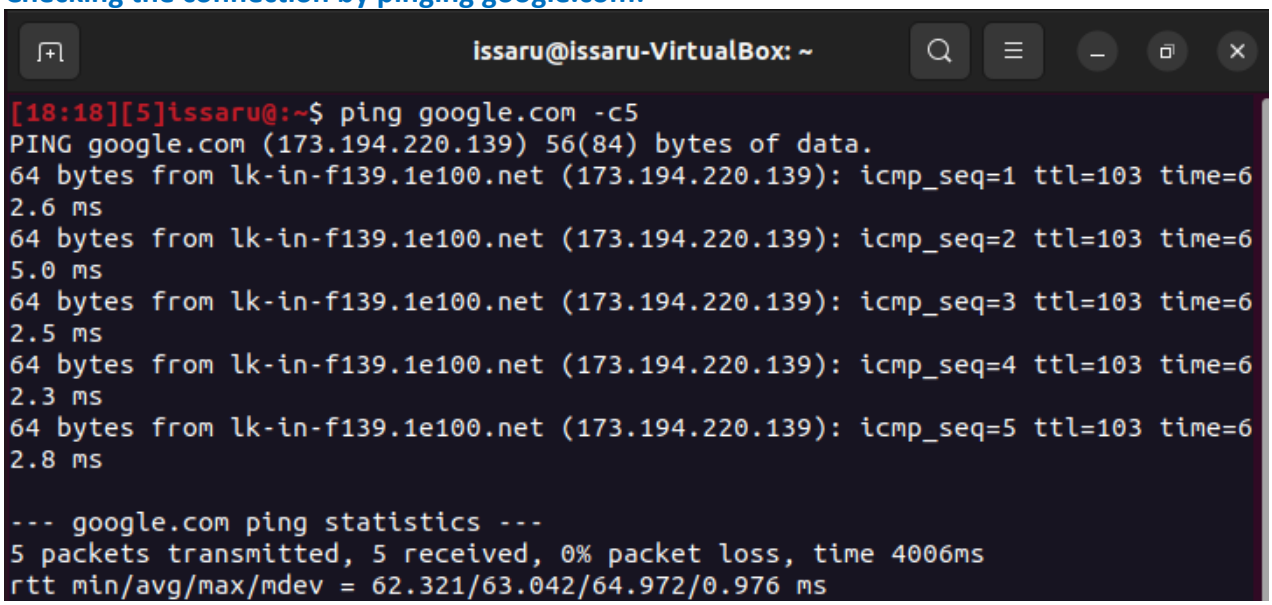
```
issaru@issaru-VirtualBox: ~
  GNU nano 6.2                  /etc/network/interfaces *
auto eth0
iface enp0s3 inet eth0

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```

Restarting Network Manager service by the following command:

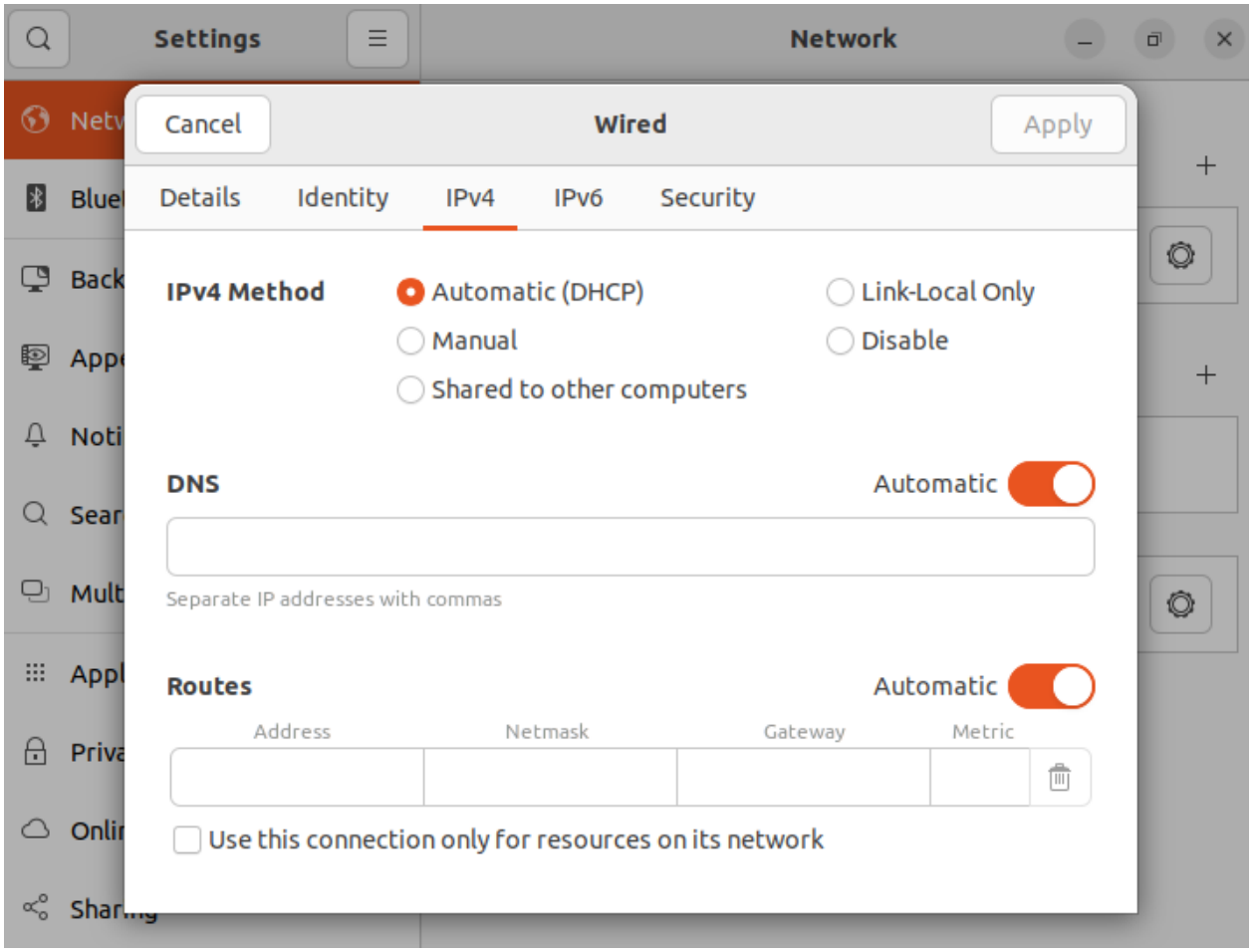
sudo systemctl restart network-manager



Checking the connection by pinging google.com:



```
[18:18][5]issaru@:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.102  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::57e1:aecb:40ec:51a9  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:ed:92:c9  txqueuelen 1000  (Ethernet)
        RX packets 3405  bytes 4374578 (4.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1034  bytes 92314 (92.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 261  bytes 24809 (24.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 261  bytes 24809 (24.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```



This is the way how to implement direct IP connection with the GUI of Ubuntu.

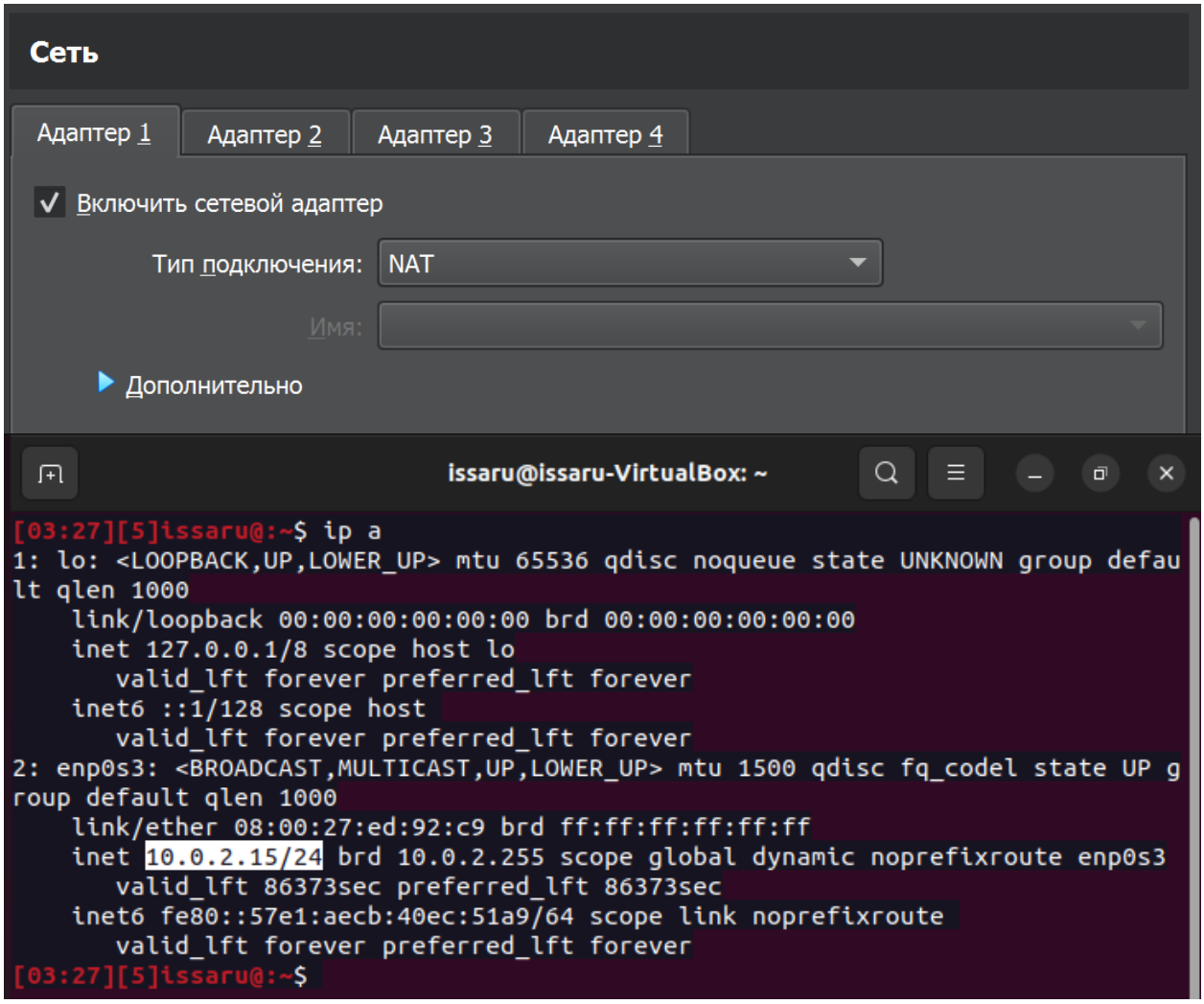
The process of Network Address Translation (NAT) permits numerous computers on a private network to utilize a shared IP address for accessing the internet. This involves the use of two sets of IP addresses, one for internal communication within the organization and the other for presenting itself to the internet. NAT has the dual benefit of conserving limited public IP space and providing security by concealing the private network from direct external access.

Here are the steps to implement a connection via NAT:

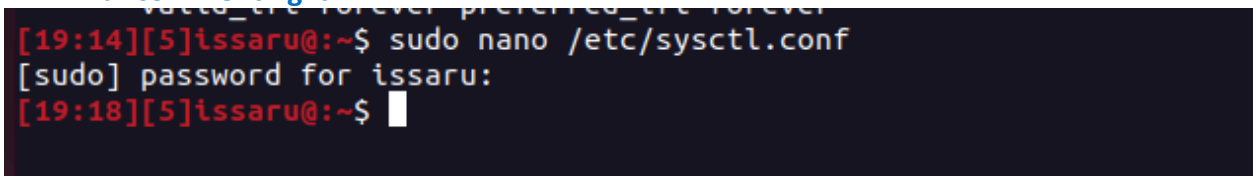
1. **Select "NAT" as the network adapter type. This will create a private network between the virtual machine and the host machine, and the virtual machine will use the host machine's IP address to connect to the Internet.**
2. **Save the changes and start the virtual machine.**
3. **Check your network configuration by running ip a command.**

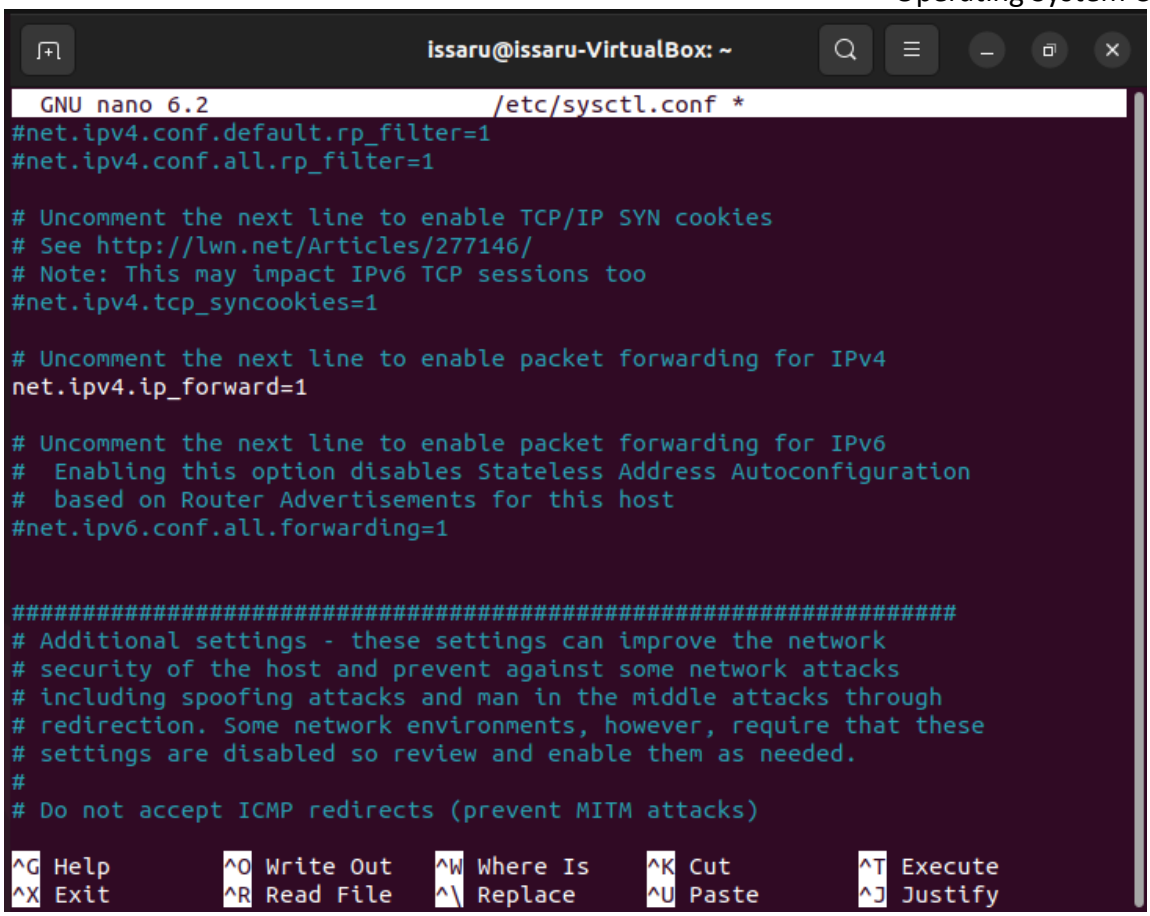
Screenshots of the code compilation result:

CONNECTION VIA NAT



4. **Open the sysctl.conf file and set the "net.ipv4.ip_forward" parameter to 1 by uncommenting it:**





5. **Enable the changes to above file using the command:**

```
[19:18][5]issaru@:~$ sudo sysctl -p
net.ipv4.ip_forward = 1
[19:19][5]issaru@:~$
```

6. **Install the iptables-persistent package using:**

```
[19:19][5]issaru@:~$ sudo apt install iptables-persistent
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  systemd-hwe-hwdb
```

7. **List configured iptable policies by issuing the command:**

```
[19:24][5]issaru@:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[19:25][5]issaru@:~$
```

8. **Mask the requests from inside the LAN with the external IP of NAT and save the iptable rules:**

```
[19:25][5]issaru@:~$ sudo iptables -t nat -A POSTROUTING -j MASQUERADE
[19:28][5]issaru@:~$ sudo iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  anywhere             anywhere
[19:29][5]issaru@:~$ sudo sh -c "iptables-save > /etc/iptables/rules.v4"
[19:30][5]issaru@:~$
```

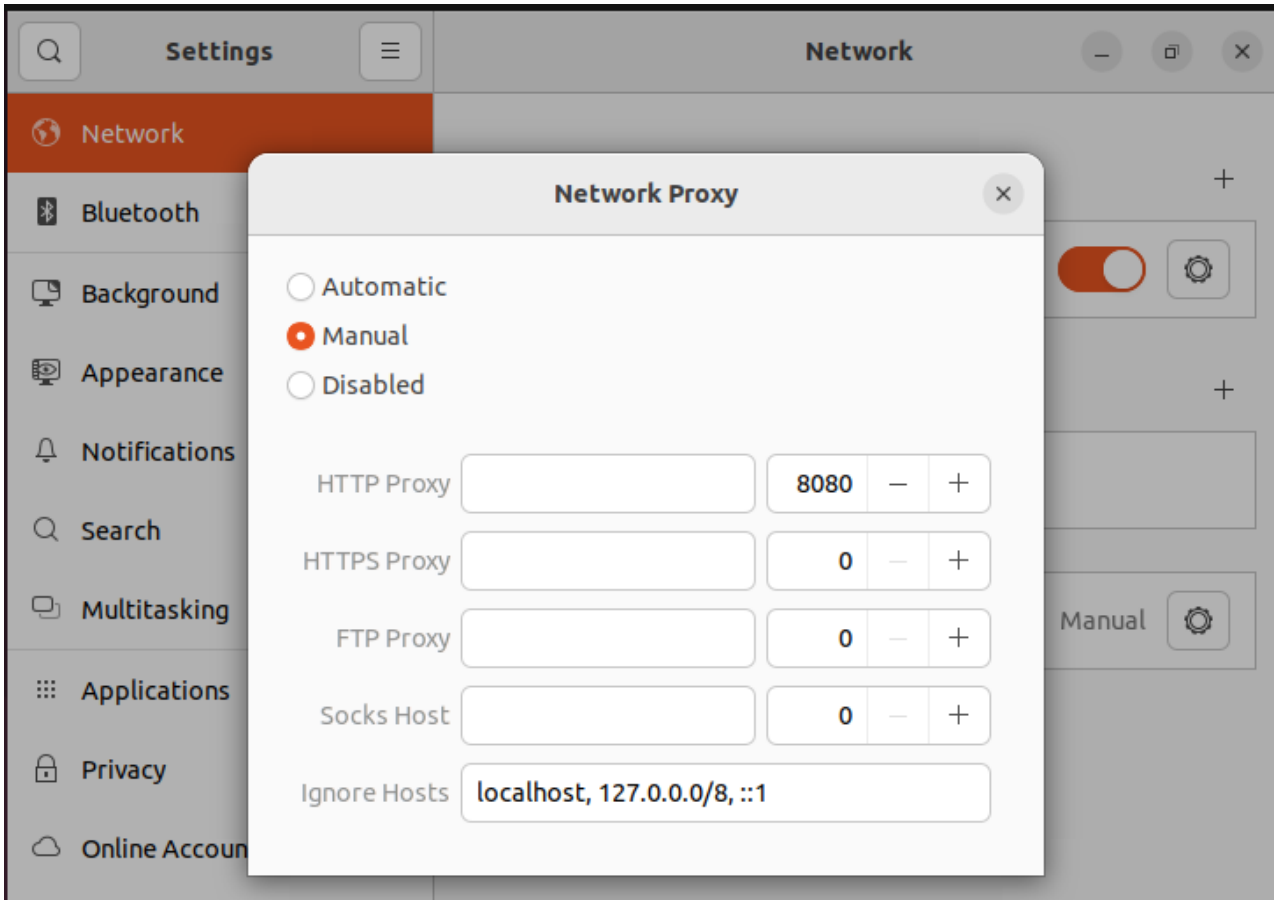
9. **Ping any public IP from the VM:**

```
[19:32][5]issaru@:~$ ping 8.8.8.8 -c 4
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=102 time=69.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=102 time=68.3 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=102 time=77.3 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=102 time=123 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 68.281/84.480/123.066/22.550 ms
[19:32][5]issaru@:~$
```

Here are the steps to implement a connection via a proxy server:

1. **Open the Network Settings dialog by running the following command:**
   sudo gnome-control-center network
2. **Click on the "Network Proxy" tab.**
3. **Select "Manual" under the "Method" section.**
4. **Enter the address and port number of the proxy server that you want to use in the "HTTP proxy" and "HTTPS proxy" fields.**

CONNECTION VIA PROXY SERVER



Task 3: Development of the Linux OS kernel module

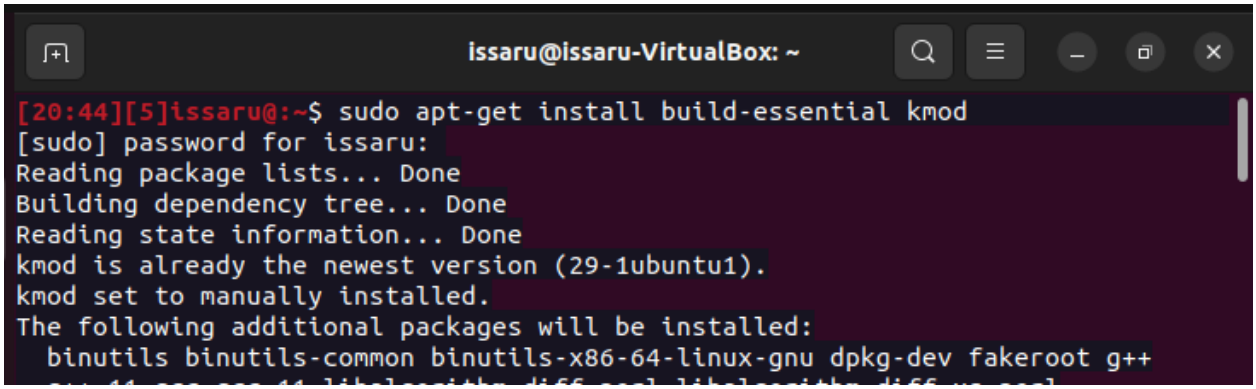
Kernel modules are individual pieces of code that can be loaded and unloaded from the kernel as needed. They extend the functionality of the kernel without having to reboot the system.

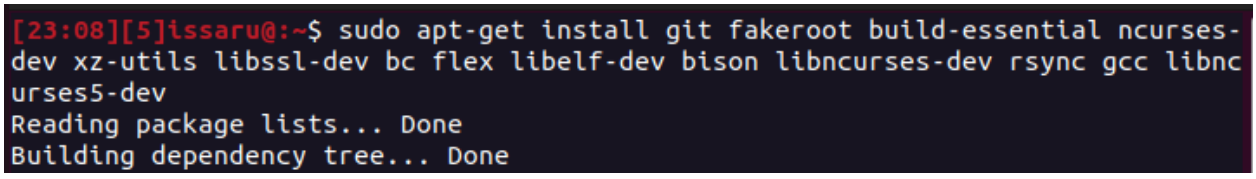
Compile and custom Linux Kernel

1. Install Kernel module packages

Linux distributions provide the commands modprobe, insmod and depmod within a package.

Package	Package description
`git`	Tracks and makes a record of all changes during development in the source code. It also allows reverting the changes.
`fakeroot`	Creates the fake root environment.
`build-essential`	Installs development tools such as C, C++, gcc, and g++.
`ncurses-dev`	Provides API for the text-based terminals.
`xz-utils`	Provides fast file compression and decompression.
`libssl-dev`	Supports SSL and TSL that encrypt data and make the internet connection secure.
`bc` (Basic Calculator)	Supports the interactive execution of statements.
`flex` (Fast Lexical Analyzer Generator)	Generates lexical analyzers that convert characters into tokens.
`libelf-dev`	Issues a shared library for managing ELF files (executable files, core dumps and object code)
`bison`	Converts grammar description to a C program.



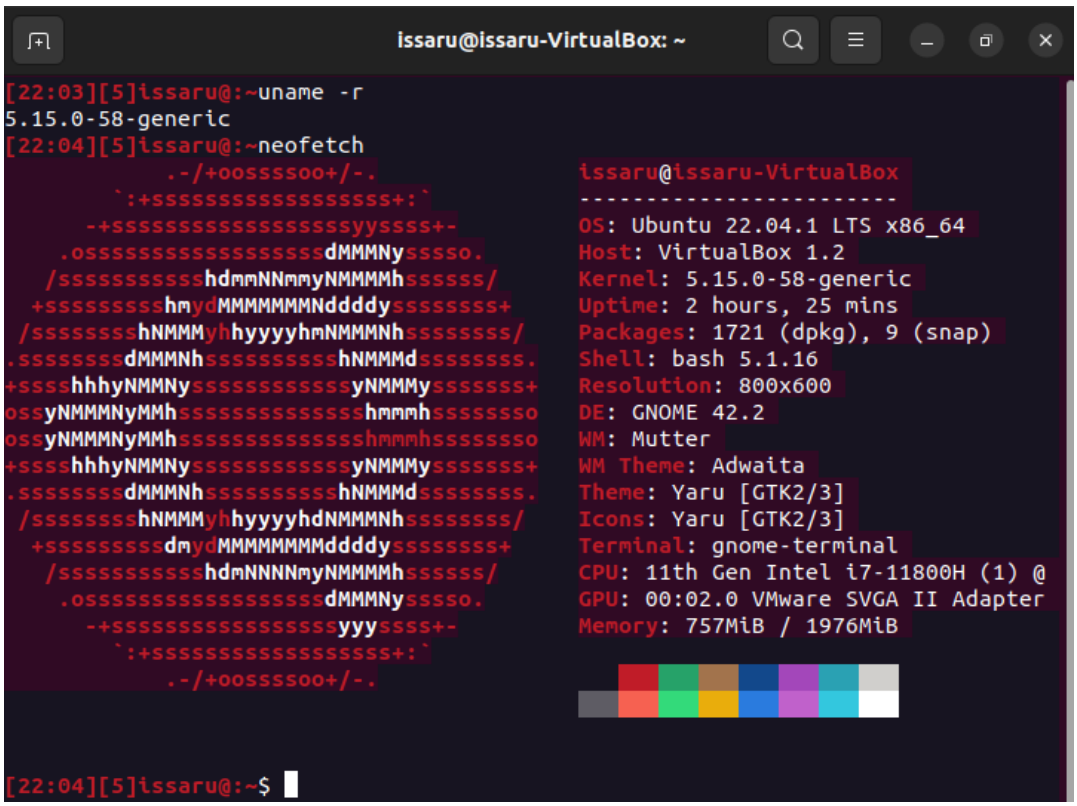


2. Discover what modules are already loaded within your current kernel

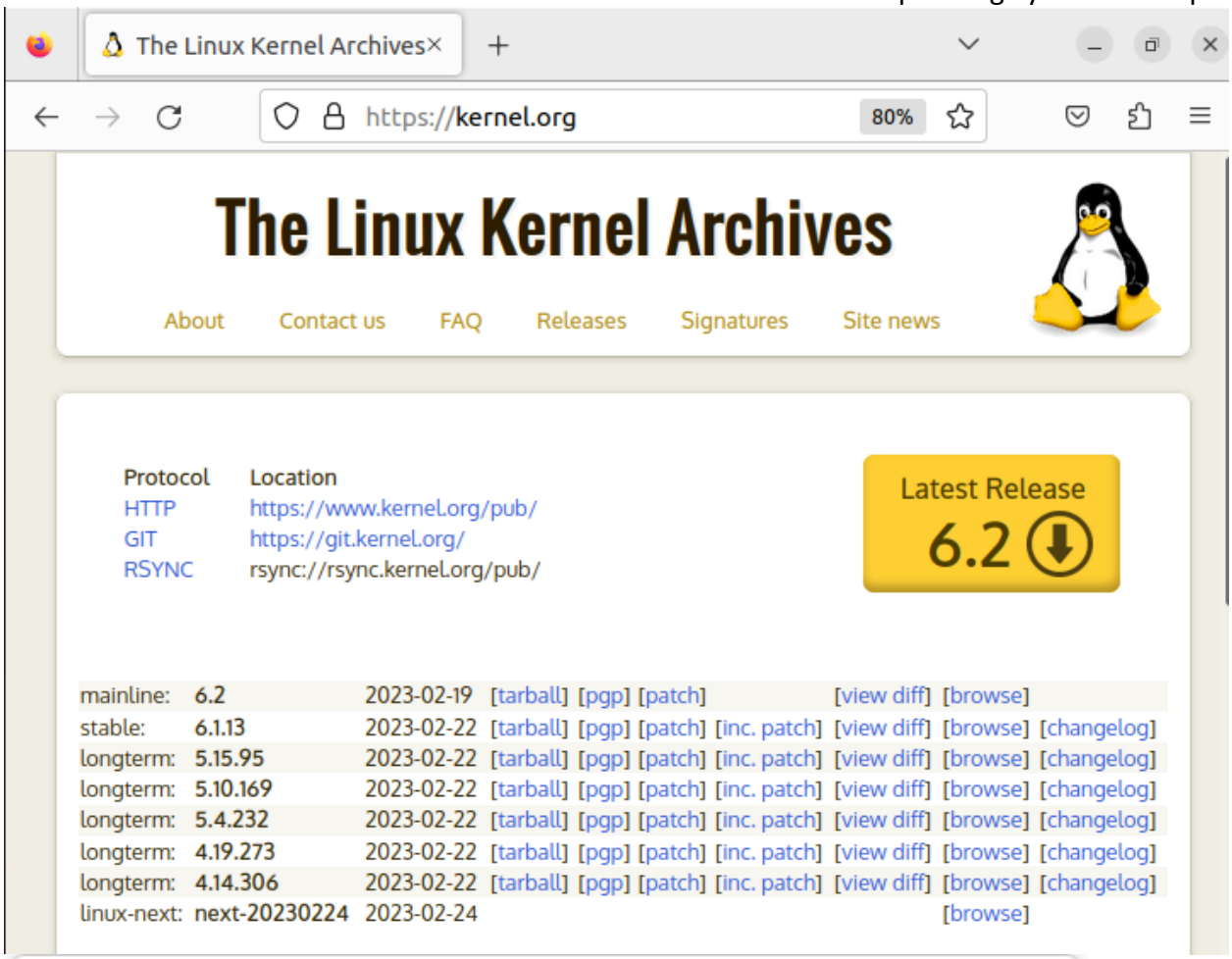
```
[20:51][5]issaru@:~$ sudo lsmod
Module                  Size  Used by
binfmt_misc            24576  1
snd_intel8x0           45056  2
snd_ac97_codec        180224  1 snd_intel8x0
ac97_bus               16384  1 snd_ac97_codec
snd_pcm               143360  2 snd_intel8x0,snd_ac97_codec
snd_seq_midi           20480  0
snd_seq_midi_event     16384  1 snd_seq_midi
snd_rawmidi            49152  1 snd_seq_midi
intel_rapl_msr         20480  0
nls_iso8859_1          16384  1
intel_rapl_common      40960  1 intel_rapl_msr
intel_powerclamp       20480  0
crct10dif_pclmul       16384  1
ghash_clmulni_intel    16384  0
snd_seq                77824  2 snd_seq_midi,snd_seq_midi_event
snd_seq_device         16384  3 snd_seq,snd_seq_midi,snd_rawmidi
snd_timer              40960  2 snd_seq,snd_pcm
aesni_intel           376832  0
crypto_simd            16384  1 aesni_intel
cryptd                 24576  2 crypto_simd,ghash_clmulni_intel
joydev                 32768  0
snd                   106496  11 snd_seq,snd_seq_device,snd_intel8x0,snd_timer,
snd_ac97_codec,snd_pcm,snd_rawmidi
```

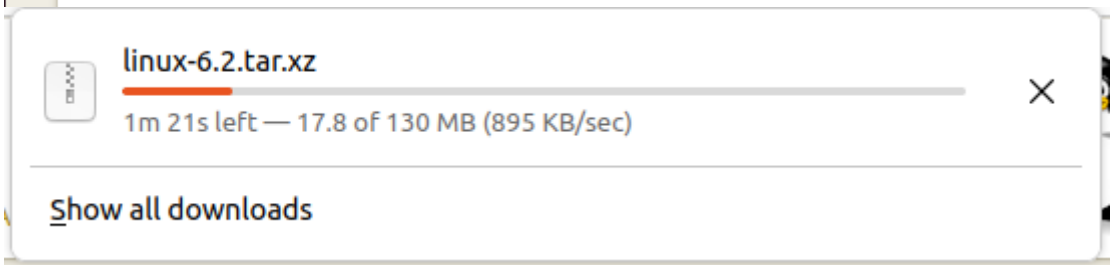
Linux Kernel deployments come in a variety of types, but we will focus on the monolithic kernel via kernel archives and make menuconfig.

3. Check the version of kernel with uname -r or neofetch



4. Download the Source Code (latest kernel version) from official website





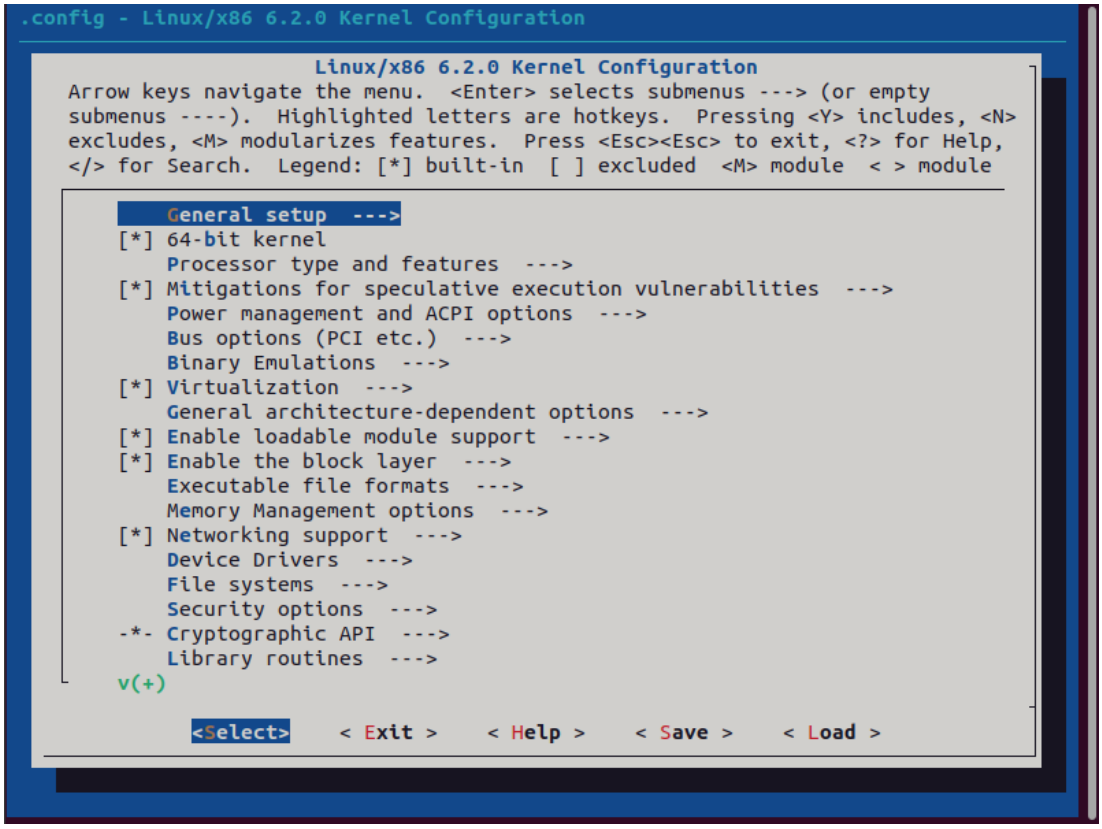

```
[23:38][5]issaru@:~$ cd Downloads
[23:39][5]issaru@:~/Downloads$ ls
linux-6.2.tar.xz
[23:39][5]issaru@:~/Downloads$
```

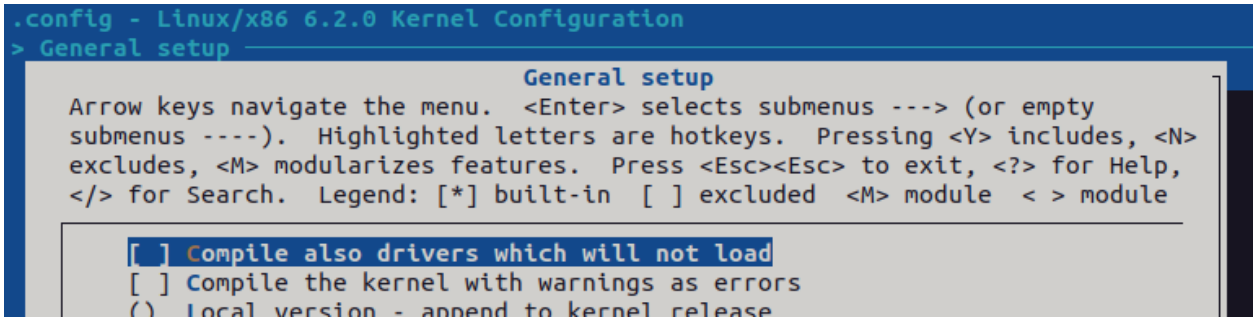
5. Extract the Source Code

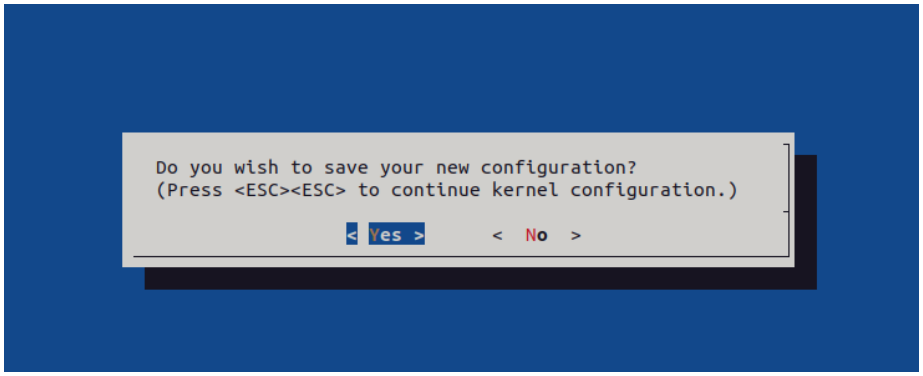
```
[23:43][5]issaru@:~/Downloads$ tar xvf linux-6.2.tar.xz
[23:45][5]issaru@:~/Downloads$ ls
linux-6.2  linux-6.2.tar.xz
[23:45][5]issaru@:~/Downloads$
```

6. Configure Kernel

```
[00:10][6]issaru@:~/Downloads/linux-6.2$ make menuconfig
#
# using defaults found in /boot/config-5.15.0-58-generic
```







```
configuration written to .config

*** End of the configuration.
```

7. Build the Kernel

Start building the kernel by running the following command:

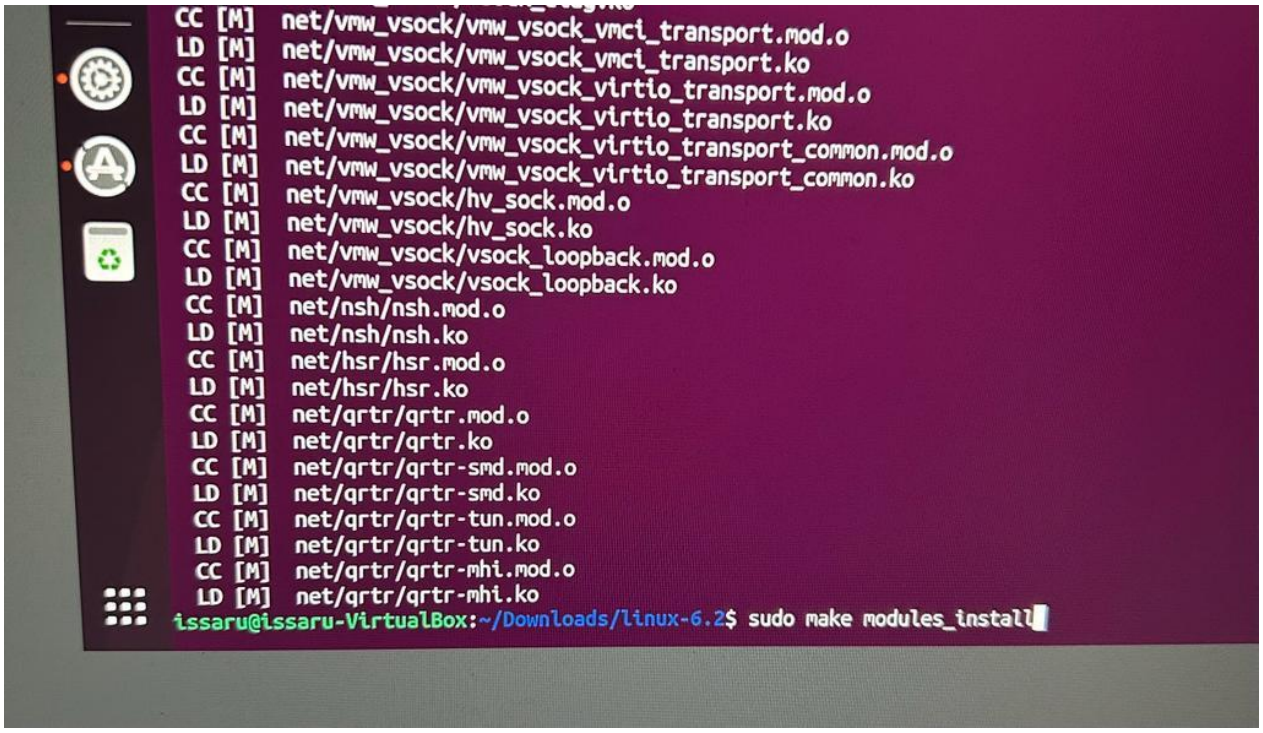
```
make
```

But before using make command disable the conflicting security certificates by executing the two commands below:

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

8. Install the required modules with this command:

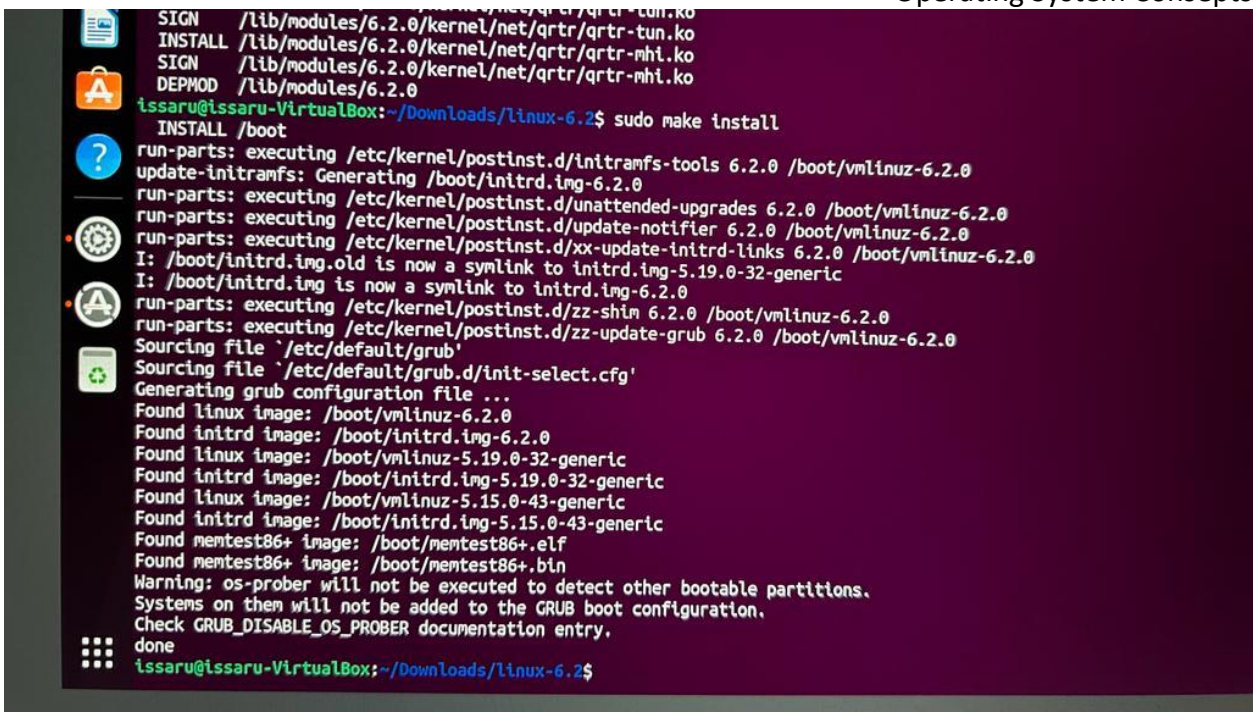
```
sudo make modules_install
```



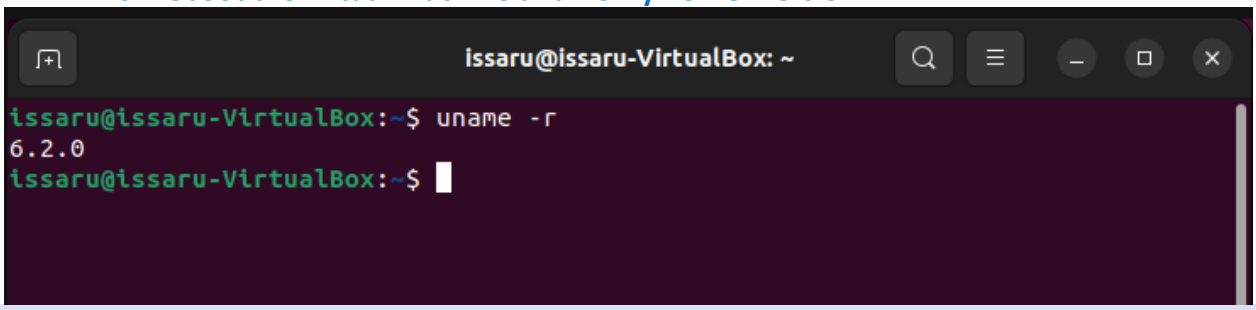
9. Install the kernel by typing:

```
sudo make install
```

The make install command performs updating the bootloader automatically.

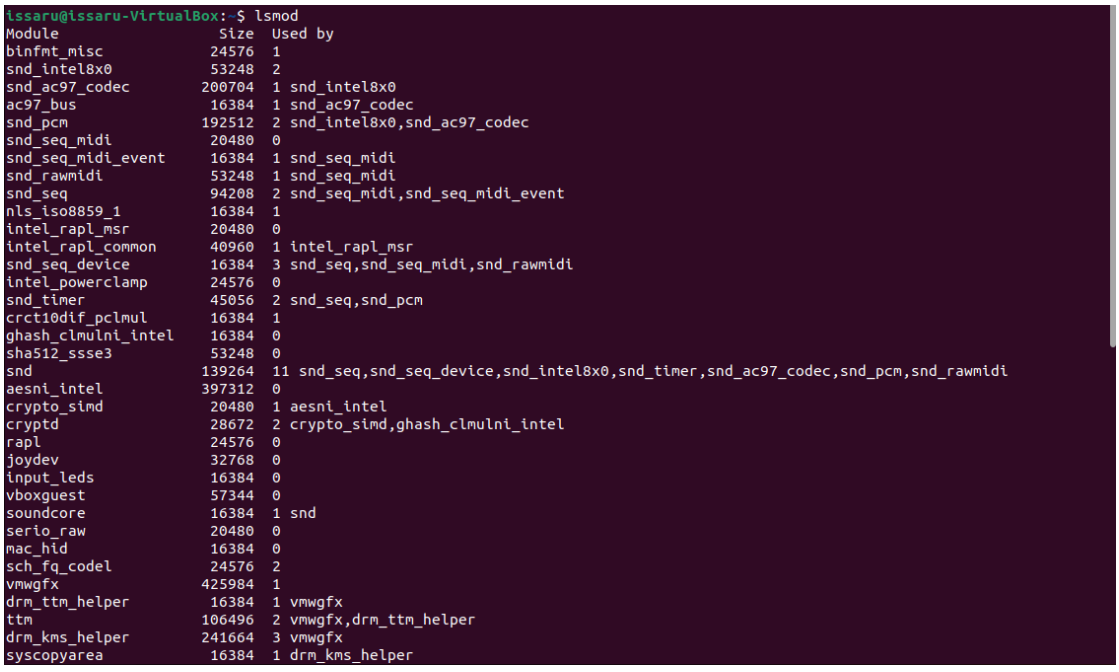


10. Reboot the virtual machine and Verify Kernel Version



MODULE # 1

NOW WE WILL DEVELOP “HELLO WORLD” KERNEL MODULE



1. **Create a new directory for our kernel module:**
2. **Change into the new directory:**

```
issaru@issaru-VirtualBox: ~/hello
issaru@issaru-VirtualBox:~$ mkdir hello
issaru@issaru-VirtualBox:~$ ls
Desktop  Documents  Downloads  hello  Music  Pictures  Public  snap  Templates  Videos
issaru@issaru-VirtualBox:~$ cd hello
issaru@issaru-VirtualBox:~/hello$
```

3. **Create a new file called hello.c and open it in a text editor:**

```
issaru@issaru-VirtualBox:~/hello$ nano helloworld.c
issaru@issaru-VirtualBox:~/hello$
```

```
issaru@issaru-VirtualBox: ~/hello
  GNU nano 6.2                                  helloworld.c *
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/printk.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Issaru");
MODULE_DESCRIPTION("Hello world module");

int init_module(void)
{
 printk(KERN_INFO "Hello World!\n");
 return 0;
}

void cleanup_module(void)
{
 printk(KERN_INFO "Goodbye World!\n");
}
```

This code defines a simple kernel module that prints "Hello World!" to the system log when it is loaded and "Goodbye World!" when it is unloaded.

4. **Create a Makefile for the kernel module:**

```
issaru@issaru-VirtualBox:~/hello$ nano Makefile
```

This Makefile will compile the kernel module and create the .ko file.

```
issaru@issaru-VirtualBox: ~/hello
  GNU nano 6.2                                  Makefile *
obj-m += helloworld.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
```

5. **Compile the kernel module using the following command:**

```
issaru@issaru-VirtualBox:~/hello$ make
make -C /lib/modules/6.2.0/build M=/home/issaru/hello modules
make[1]: Entering directory '/home/issaru/Downloads/linux-6.2'
  CC [M]  /home/issaru/hello/helloworld.o
  MODPOST /home/issaru/hello/Module.symvers
  CC [M]  /home/issaru/hello/helloworld.mod.o
  LD [M]  /home/issaru/hello/helloworld.ko
make[1]: Leaving directory '/home/issaru/Downloads/linux-6.2'
issaru@issaru-VirtualBox:~/hello$
```

This will create the helloworld.ko file in the hello directory.

6. **Load the kernel module into the kernel using the following command:**

```
issaru@issaru-VirtualBox:~/hello$ sudo insmod helloworld.ko
[sudo] password for issaru:
issaru@issaru-VirtualBox:~/hello$
```

```
issaru@issaru-VirtualBox:~/hello$ lsmod | grep hello
helloworld             16384  0
```

7. **Check the system log to see the output of our kernel module using the following command:**

```
issaru@issaru-VirtualBox:~/hello$ sudo dmesg | tail
[ 1885.786489] audit: type=1400 audit(1677296059.931:130): apparmor="STATUS" operation="profile_replace" profile="unconfined
" name="snap.firefox.firefox" pid=3294 comm="apparmor_parser"
[ 1886.364343] audit: type=1400 audit(1677296060.511:131): apparmor="STATUS" operation="profile_replace" profile="unconfined
" name="snap.firefox.geckodriver" pid=3295 comm="apparmor_parser"
[ 1886.473979] audit: type=1400 audit(1677296060.619:132): apparmor="STATUS" operation="profile_replace" profile="unconfined
" name="snap.firefox.hook.configure" pid=3296 comm="apparmor_parser"
[ 1886.556715] audit: type=1400 audit(1677296060.703:133): apparmor="STATUS" operation="profile_replace" profile="unconfined
" name="snap.firefox.hook.connect-plug-host-hunspell" pid=3297 comm="apparmor_parser"
[ 1886.642766] audit: type=1400 audit(1677296060.787:134): apparmor="STATUS" operation="profile_replace" profile="unconfined
" name="snap.firefox.hook.disconnect-plug-host-hunspell" pid=3298 comm="apparmor_parser"
[ 1886.730439] audit: type=1400 audit(1677296060.875:135): apparmor="STATUS" operation="profile_replace" profile="unconfined
" name="snap.firefox.hook.post-refresh" pid=3299 comm="apparmor_parser"
[ 1886.850771] audit: type=1400 audit(1677296060.995:136): apparmor="DENIED" operation="open" class="file" profile="snap-upd
ate-ns.firefox" name="/var/lib/" pid=3312 comm="6" requested_mask="r" denied_mask="r" fsuid=0 ouid=0
[ 5035.470758] helloworld: loading out-of-tree module taints kernel.
[ 5035.470788] helloworld: module verification failed: signature and/or required key missing - tainting kernel
[ 5035.470884] Hello World!
issaru@issaru-VirtualBox:~/hello$
```

8. **Unload the kernel module from the kernel using the following command:**

```
issaru@issaru-VirtualBox:~/hello$ sudo rmmod helloworld
issaru@issaru-VirtualBox:~/hello$ sudo dmesg | tail
[ 1886.364343] audit: type=1400 audit(1677296060.511:131): apparmor="STATUS" operation="profile_replace" profile="unconfined
" name="snap.firefox.geckodriver" pid=3295 comm="apparmor_parser"
[ 1886.473979] audit: type=1400 audit(1677296060.619:132): apparmor="STATUS" operation="profile_replace" profile="unconfined
" name="snap.firefox.hook.configure" pid=3296 comm="apparmor_parser"
[ 1886.556715] audit: type=1400 audit(1677296060.703:133): apparmor="STATUS" operation="profile_replace" profile="unconfined
" name="snap.firefox.hook.connect-plug-host-hunspell" pid=3297 comm="apparmor_parser"
[ 1886.642766] audit: type=1400 audit(1677296060.787:134): apparmor="STATUS" operation="profile_replace" profile="unconfined
" name="snap.firefox.hook.disconnect-plug-host-hunspell" pid=3298 comm="apparmor_parser"
[ 1886.730439] audit: type=1400 audit(1677296060.875:135): apparmor="STATUS" operation="profile_replace" profile="unconfined
" name="snap.firefox.hook.post-refresh" pid=3299 comm="apparmor_parser"
[ 1886.850771] audit: type=1400 audit(1677296060.995:136): apparmor="DENIED" operation="open" class="file" profile="snap-upd
ate-ns.firefox" name="/var/lib/" pid=3312 comm="6" requested_mask="r" denied_mask="r" fsuid=0 ouid=0
[ 5035.470758] helloworld: loading out-of-tree module taints kernel.
[ 5035.470788] helloworld: module verification failed: signature and/or required key missing - tainting kernel
[ 5035.470884] Hello World!
[ 5447.553271] Goodbye World!
issaru@issaru-VirtualBox:~/hello$
```

The "Goodbye World!" message was printed to the system log by running the dmesg | tail command again.

That's it! We have developed and tested a "Hello World" kernel module on Ubuntu. This is a simple example, but kernel modules can be used to add new functionality to the Linux kernel and to interact with hardware devices or system resources.

MODULE #2

Develop kernel module that generates a random number:

```
issaru@issaru-VirtualBox:~$ mkdir random
issaru@issaru-VirtualBox:~$ cd random
issaru@issaru-VirtualBox:~/random$ nano myrandommodule.c
issaru@issaru-VirtualBox:~/random$ nano Makefile
issaru@issaru-VirtualBox:~/random$ make
make -C /lib/modules/6.2.0/build M=/home/issaru/random modules
make[1]: Entering directory '/home/issaru/Downloads/linux-6.2'
  CC [M]  /home/issaru/random/myrandommodule.o
  MODPOST /home/issaru/random/Module.symvers
  CC [M]  /home/issaru/random/myrandommodule.mod.o
  LD [M]  /home/issaru/random/myrandommodule.ko
make[1]: Leaving directory '/home/issaru/Downloads/linux-6.2'
```

```
issaru@issaru-VirtualBox:~/random$ sudo insmod myrandommodule.ko
issaru@issaru-VirtualBox:~/random$ sudo dmesg | tail
[ 1886.850771] audit: type=1400 audit(1677296060.995:136): apparmor="DENIED" operation="open" class=
refox" name="/var/lib/" pid=3312 comm="6" requested_mask="r" denied_mask="r" fsuid=0 ouid=0
[ 5035.470758] helloworld: loading out-of-tree module taints kernel.
[ 5035.470788] helloworld: module verification failed: signature and/or required key missing - taint
[ 5035.470884] Hello World!
[ 5447.553271] Goodbye World!
[11484.119489] kauditd_printk_skb: 1 callbacks suppressed
[11484.119492] audit: type=1326 audit(1677305658.234:138): auid=1000 uid=1000 gid=1000 ses=3 subj=sn
"firefox" exe="/snap/firefox/2356/usr/lib/firefox/firefox" sig=0 arch=c000003e syscall=314 compat=0
[15044.390861] audit: type=1326 audit(1677309218.471:139): auid=1000 uid=1000 gid=1000 ses=3 subj=sn
="firefox" exe="/snap/firefox/2356/usr/lib/firefox/firefox" sig=0 arch=c000003e syscall=314 compat=0
[16255.295063] myrandmodule module loaded
[16255.295066] Random number: -381715719
```

```
issaru@issaru-VirtualBox:~/random$ sudo rmmod myrandommodule
issaru@issaru-VirtualBox:~/random$ sudo dmesg | tail
[ 5035.470758] helloworld: loading out-of-tree module taints kernel.
[ 5035.470788] helloworld: module verification failed: signature and/or required key missing - tainting kernel
[ 5035.470884] Hello World!
[ 5447.553271] Goodbye World!
[11484.119489] kauditd_printk_skb: 1 callbacks suppressed
[11484.119492] audit: type=1326 audit(1677305658.234:138): auid=1000 uid=1000 gid=1000 ses=3 subj=snap.firefox.fire
"firefox" exe="/snap/firefox/2356/usr/lib/firefox/firefox" sig=0 arch=c000003e syscall=314 compat=0 ip=0x7f58afa667
[15044.390861] audit: type=1326 audit(1677309218.471:139): auid=1000 uid=1000 gid=1000 ses=3 subj=snap.firefox.fire
="firefox" exe="/snap/firefox/2356/usr/lib/firefox/firefox" sig=0 arch=c000003e syscall=314 compat=0 ip=0x7f0e18022
[16255.295063] myrandmodule module loaded
[16255.295066] Random number: -381715719
[16320.320718] myrandmodule module unloaded
issaru@issaru-VirtualBox:~/random$
```

```
 GNU nano 6.2                                          myrandommodule.c *
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/random.h>
#include <linux/printk.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Issaru");
MODULE_DESCRIPTION("Module that generates random number");

static int __init myrandmodule_init(void)
{
    int random;

    printk(KERN_INFO "myrandmodule module loaded\n");

    get_random_bytes(&random, sizeof(random));
    printk(KERN_INFO "Random number: %d\n", random);

    return 0;
}

static void __exit myrandmodule_exit(void)
{
    printk(KERN_INFO "myrandmodule module unloaded\n");
}

module_init(myrandmodule_init);
module_exit(myrandmodule_exit);
```

```
 GNU nano 6.2                                                  Makefile *
obj-m += myrandommodule.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
```

MODULE #3

Develop a kernel module on Ubuntu that can calculate the sum of two integers entered by the user:

```
issaru@issaru-VirtualBox:~$ mkdir sumofnumbers
issaru@issaru-VirtualBox:~$ cd sumofnumbers
issaru@issaru-VirtualBox:~/sumofnumbers$ nano sumoftwointegers.c
issaru@issaru-VirtualBox:~/sumofnumbers$ nano Makefile
issaru@issaru-VirtualBox:~/sumofnumbers$ make
make -C /lib/modules/6.2.0/build M=/home/issaru/sumofnumbers modules
make[1]: Entering directory '/home/issaru/Downloads/linux-6.2'
  CC [M]  /home/issaru/sumofnumbers/sumoftwointegers.o
  MODPOST /home/issaru/sumofnumbers/Module.symvers
  CC [M]  /home/issaru/sumofnumbers/sumoftwointegers.mod.o
  LD [M]  /home/issaru/sumofnumbers/sumoftwointegers.ko
make[1]: Leaving directory '/home/issaru/Downloads/linux-6.2'
issaru@issaru-VirtualBox:~/sumofnumbers$ sudo insmod sumoftwointegers.ko first_number=1 second_number=10
issaru@issaru-VirtualBox:~/sumofnumbers$ sudo dmesg | tail
[ 5035.470884] Hello World!
[ 5447.553271] Goodbye World!
[11484.119489] kauditd_printk_skb: 1 callbacks suppressed
[11484.119492] audit: type=1326 audit(1677305658.234:138): auid=1000 uid=1000 gid=1000 ses=3 subj=snap.firefox.f
"firefox" exe="/snap/firefox/2356/usr/lib/firefox/firefox" sig=0 arch=c000003e syscall=314 compat=0 ip=0x7f58afa
[15044.390861] audit: type=1326 audit(1677309218.471:139): auid=1000 uid=1000 gid=1000 ses=3 subj=snap.firefox.f
="firefox" exe="/snap/firefox/2356/usr/lib/firefox/firefox" sig=0 arch=c000003e syscall=314 compat=0 ip=0x7f0e18
[16255.295063] myrandmodule module loaded
[16255.295066] Random number: -381715719
[16320.320718] myrandmodule module unloaded
[17920.323344] The sum of 5 and 7 is 12
[18781.326794] The sum of 1 and 10 is 11
issaru@issaru-VirtualBox:~/sumofnumbers$
```

```
 GNU nano 6.2                                    sumoftwointegers.c *
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/moduleparam.h>
#include <linux/printk.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Issaru");
MODULE_DESCRIPTION("Module that calculates sum of two integers");

static int first_number;
static int second_number;

module_param(first_number, int, S_IRUSR | S_IWUSR);
module_param(second_number, int, S_IRUSR | S_IWUSR);

int sum(int a, int b) {
    return a + b;
}

static int __init summodule_init(void)
{
    int result = sum(first_number, second_number);

    printk(KERN_INFO "The sum of %d and %d is %d\n", first_number, second_number, result);

    return 0;
}

static void __exit summodule_exit(void)
{
    printk(KERN_INFO "Sum module unloaded\n");
}

module_init(summodule_init);
```

```
 GNU nano 6.2                                          Makefile *
obj-m += sumoftwointegers.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
```

CONCLUSION

The final project for the operating systems course involved a series of tasks designed to develop hands-on experience with important aspects of Linux operating systems.
These tasks included adding a new user to the system using the useradd command, implementing three methods of connecting to the Internet (a "Direct" IP connection, a connection via NAT, and a connection via a proxy server), and developing a Linux OS kernel module.
Through completing these tasks, we gained a deeper understanding of how to work with user accounts and network connections in Linux.
In addition, the kernel module development provided an opportunity to gain insight into the underlying workings of the Linux OS and its system architecture.
Overall, this project provided valuable experience with some of the fundamental concepts and tools used in Linux administration and kernel development.
It served as a challenging and rewarding capstone to the course, and we feel well-prepared to tackle similar tasks in the future.

REFERENCES

https://wiki.archlinux.org/title/Kernel_module
https://sysprog21.github.io/lkmpg/#device-drivers
https://www.altlinux.org/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D0%B5%D0%B9_%D1%8F%D0%B4%D1%80%D0%B0#%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B