Operating System Consepts

Template Final Exam

#	Full Name	Group
1	Elen Aruzhan	CS-2107S
2	Seilova Madina	CS-2107S

Link to the repository: https://github.com/Clushka/Operating-System-Final-Exam.git

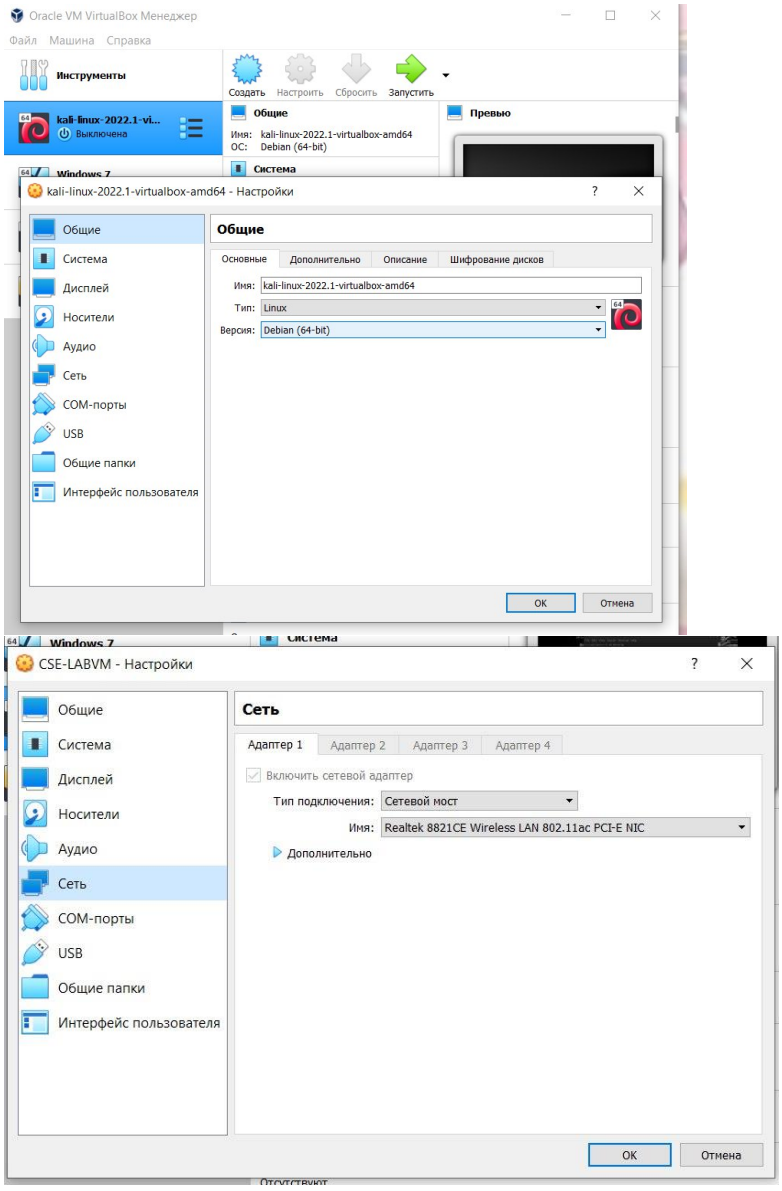
Step-by-step task completion:

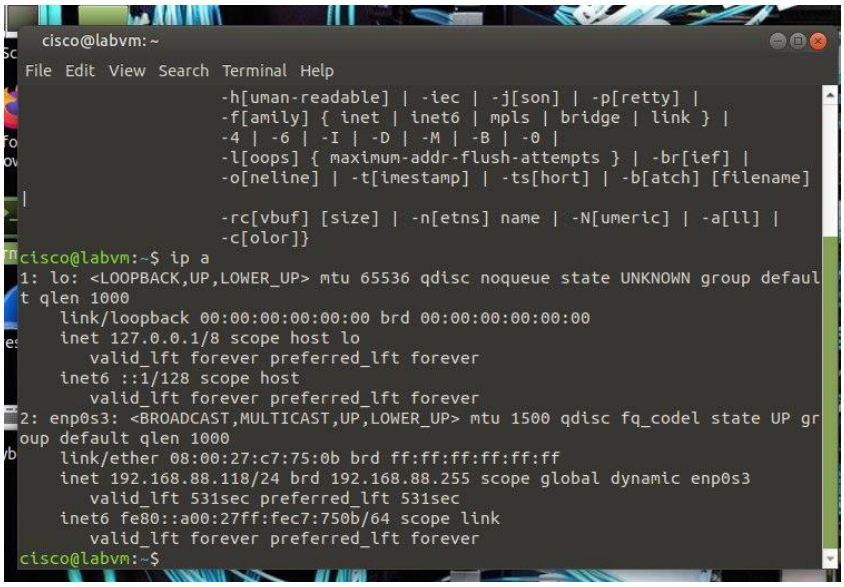
Task 1:
Screenshots of the code compilation result:

Task 2:
Screenshots of the code compilation result:
1)
To set up a "Direct" IP connection to the internet on a virtual machine (VM), we need to configure the VM's network adapter in Bridge mode. Bridge mode allows the VM to share the

host machine's network adapter and obtain its own IP address from the local network. Here are the steps to configure Bridge mode in VirtualBox:

Open VirtualBox and select the VM you want to configure.
Click on "Settings" and select the "Network" tab.
In the "Adapter 1" section, select "Bridged Adapter" from the "Attached to" drop-down menu.
Select the network adapter that we want to use from the "Name" drop-down menu.
Click "OK" to save the changes and close the settings window.
Start the VM.
Once the VM is up and running, it should have its own IP address and be able to access the internet directly. we can verify this by opening a web browser on the VM and visiting a website.



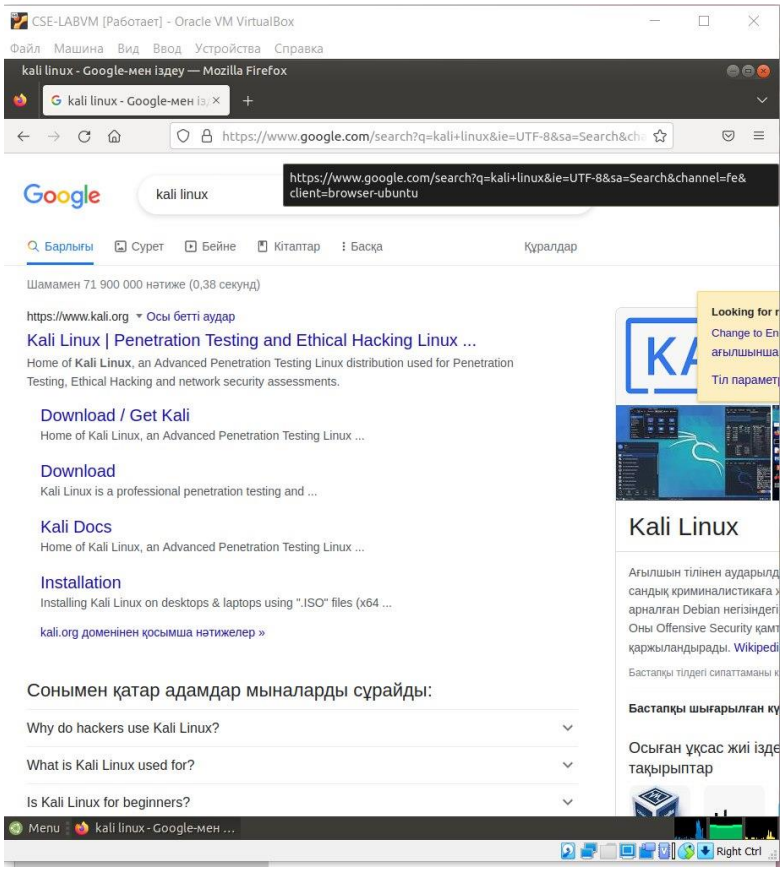


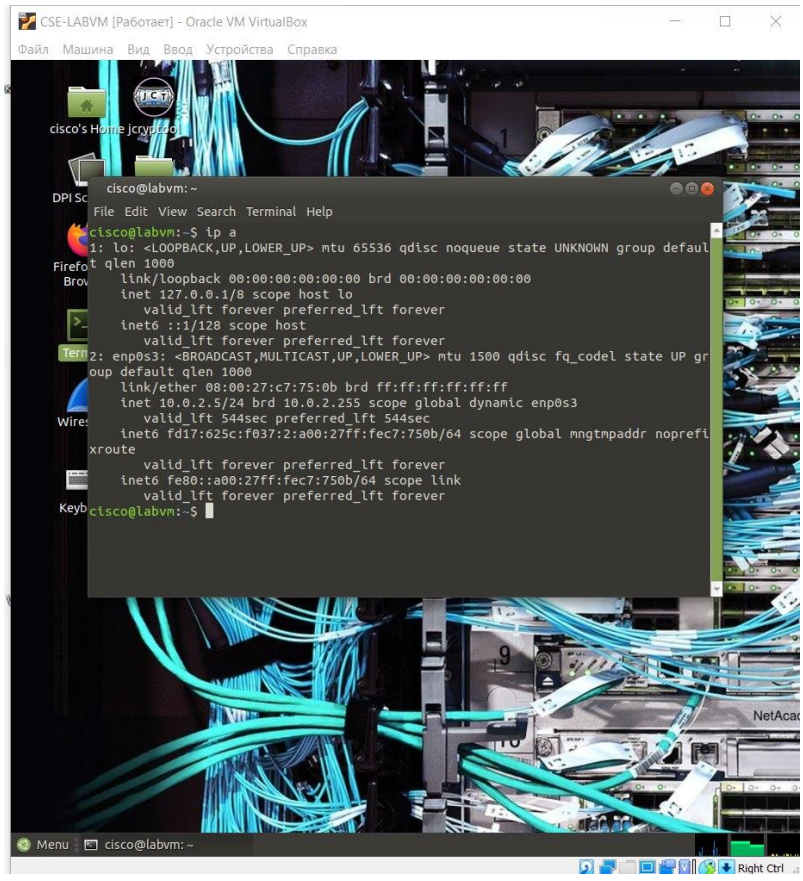
2) To set up a connection via Network Address Translation (NAT) on a virtual machine (VM), we need to configure the VM's network adapter to use NAT. NAT allows the VM to share the host machine's network adapter and obtain an IP address from a private network created by the virtualization software. Here are the steps to configure NAT in VirtualBox:

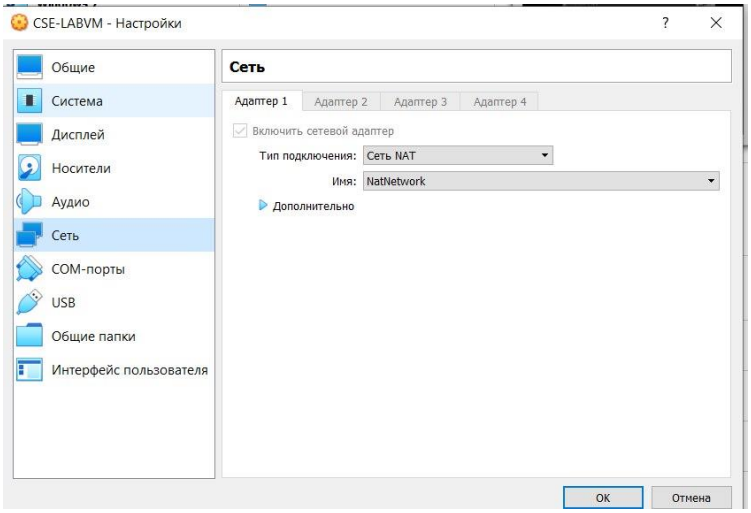
Open VirtualBox and select the VM you want to configure.
Click on "Settings" and select the "Network" tab.
In the "Adapter 1" section, select "NAT" from the "Attached to" drop-down menu.
Click "OK" to save the changes and close the settings window.
Start the VM.
Once the VM is up and running, it should have an IP address assigned by the virtualization software and be able to access the internet through the host machine's network adapter. To verify that the VM is connected to the internet, we opened a web browser on the VM and visit a website.

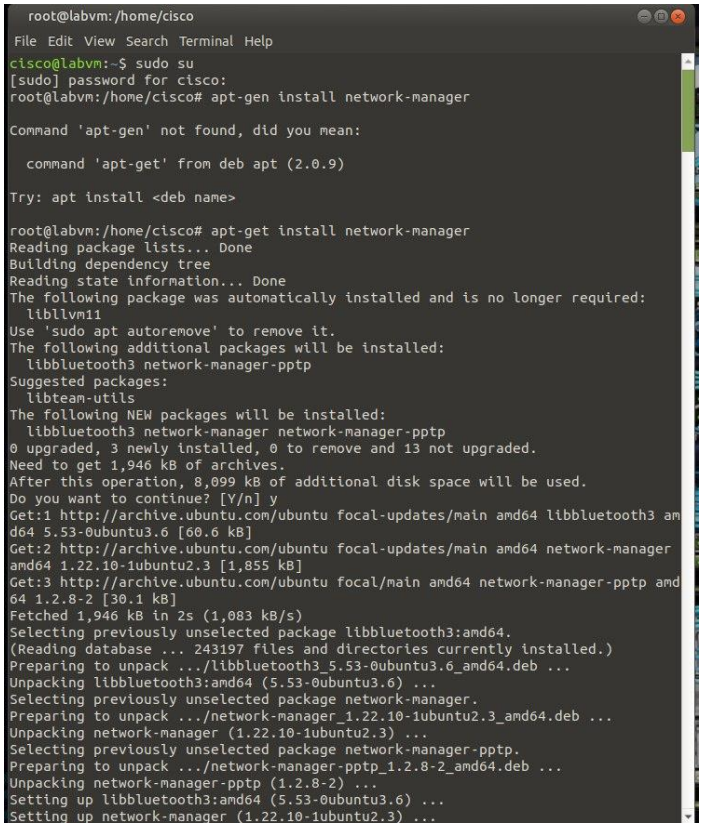
ASTANA IT UNIVERSITY

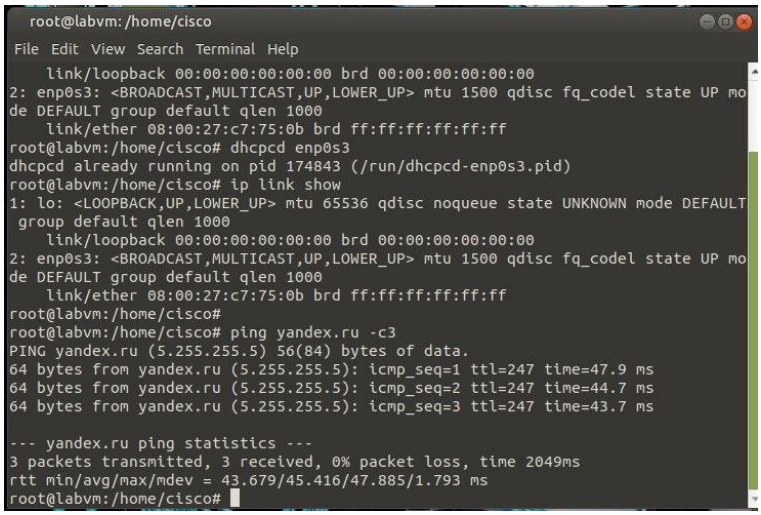
Operating System Consepts



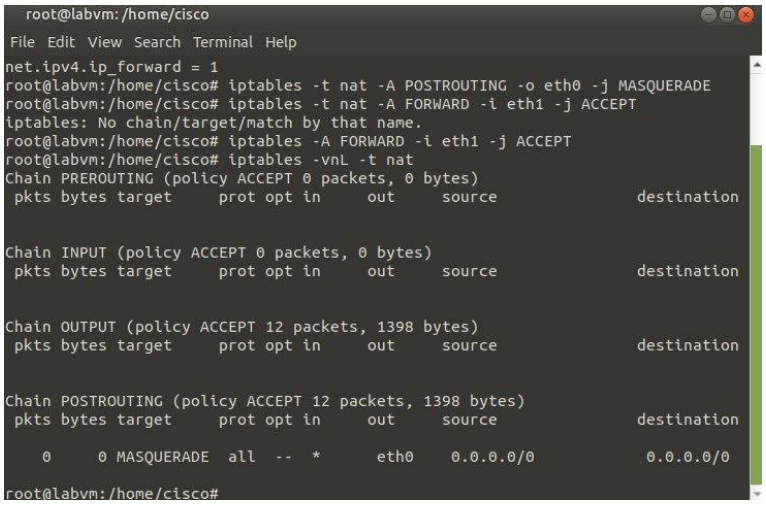



CSE-LABVM - Настройки
Общие
Система
Дисплей
Носители
Аудио
Сеть
COM-порты
USB
Общие папки
Интерфейс пользователя
Сеть
Адаптер 1
Адаптер 2
Адаптер 3
Адаптер 4
Включить сетевой адаптер
Тип подключения:
Сеть NAT
Имя:
NatNetwork
Дополнительно
OK
Отмена


root@labvm: /home/cisco
File Edit View Search Terminal Help
cisco@labvm:~$ sudo su
[sudo] password for cisco:
root@labvm:/home/cisco# apt-gen install network-manager
Command 'apt-gen' not found, did you mean:
command 'apt-get' from deb apt (2.0.9)
Try: apt install <deb name>
root@labvm:/home/cisco# apt-get install network-manager
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
libllvm11
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
libbluetooth3 network-manager-pptp
Suggested packages:
libteam-utils
The following NEW packages will be installed:
libbluetooth3 network-manager network-manager-pptp
0 upgraded, 3 newly installed, 0 to remove and 13 not upgraded.
Need to get 1,946 kB of archives.
After this operation, 8,099 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libbluetooth3 amd64 5.53-0ubuntu3.6 [60.6 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 network-manager amd64 1.22.10-1ubuntu2.3 [1,855 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/main amd64 network-manager-pptp amd64 1.2.8-2 [30.1 kB]
Fetched 1,946 kB in 2s (1,083 kB/s)
Selecting previously unselected package libbluetooth3:amd64.
(Reading database ... 243197 files and directories currently installed.)
Preparing to unpack .../libbluetooth3_5.53-0ubuntu3.6_amd64.deb ...
Unpacking libbluetooth3:amd64 (5.53-0ubuntu3.6) ...
Selecting previously unselected package network-manager.
Preparing to unpack .../network-manager_1.22.10-1ubuntu2.3_amd64.deb ...
Unpacking network-manager (1.22.10-1ubuntu2.3) ...
Selecting previously unselected package network-manager-pptp.
Preparing to unpack .../network-manager-pptp_1.2.8-2_amd64.deb ...
Unpacking network-manager-pptp (1.2.8-2) ...
Setting up libbluetooth3:amd64 (5.53-0ubuntu3.6) ...
Setting up network-manager (1.22.10-1ubuntu2.3) ...


root@labvm: /home/cisco
File Edit View Search Terminal Help
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mo
de DEFAULT group default qlen 1000
link/ether 08:00:27:c7:75:0b brd ff:ff:ff:ff:ff:ff
root@labvm:/home/cisco# dhcpcd enp0s3
dhcpcd already running on pid 174843 (/run/dhcpcd-enp0s3.pid)
root@labvm:/home/cisco# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mo
de DEFAULT group default qlen 1000
link/ether 08:00:27:c7:75:0b brd ff:ff:ff:ff:ff:ff
root@labvm:/home/cisco#
root@labvm:/home/cisco# ping yandex.ru -c3
PING yandex.ru (5.255.255.5) 56(84) bytes of data.
64 bytes from yandex.ru (5.255.255.5): icmp_seq=1 ttl=247 time=47.9 ms
64 bytes from yandex.ru (5.255.255.5): icmp_seq=2 ttl=247 time=44.7 ms
64 bytes from yandex.ru (5.255.255.5): icmp_seq=3 ttl=247 time=43.7 ms
--- yandex.ru ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2049ms
rtt min/avg/max/mdev = 43.679/45.416/47.885/1.793 ms
root@labvm:/home/cisco#

3)


root@labvm: /home/cisco
File Edit View Search Terminal Help
net.ipv4.ip_forward = 1
root@labvm:/home/cisco# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
root@labvm:/home/cisco# iptables -t nat -A FORWARD -i eth1 -j ACCEPT
iptables: No chain/target/match by that name.
root@labvm:/home/cisco# iptables -A FORWARD -i eth1 -j ACCEPT
root@labvm:/home/cisco# iptables -vnL -t nat
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
Chain OUTPUT (policy ACCEPT 12 packets, 1398 bytes)
pkts bytes target prot opt in out source destination
Chain POSTROUTING (policy ACCEPT 12 packets, 1398 bytes)
pkts bytes target prot opt in out source destination
0 0 MASQUERADE all -- * eth0 0.0.0.0/0 0.0.0.0/0
root@labvm:/home/cisco#

Task 3:

```
ubuntu@ubuntu:~$ gcc --version
make --version
gcc (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

GNU Make 4.3
Built for x86_64-pc-linux-gnu
Copyright (C) 1988-2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
ubuntu@ubuntu:~$ mkdir hello
ubuntu@ubuntu:~$ cd hello
ubuntu@ubuntu:~/hello$
```

we'll develop a simple "hello world" kernel module that prints a message when it is loaded.

We have created a new directory for the module and created a new file called hello.c with the following code:

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

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Your Name");
MODULE_DESCRIPTION("A simple hello world kernel module");

static int __init hello_init(void)
{
    printk(KERN_INFO "Hello, world!\n");
    return 0;
}

static void __exit hello_exit(void)
{
    printk(KERN_INFO "Goodbye, world!\n");
}

module_init(hello_init);
module_exit(hello_exit);
```

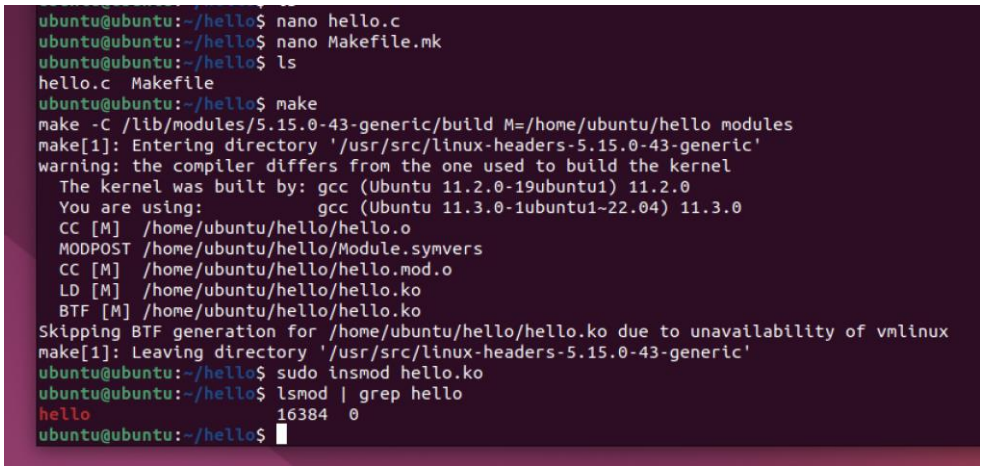
This code defines the module and includes the necessary kernel headers. It also defines two functions, hello_init and hello_exit, which will be called when the module is loaded and unloaded, respectively. The printk function is used to print messages to the kernel log.

Then we will create a Makefile with the following code:

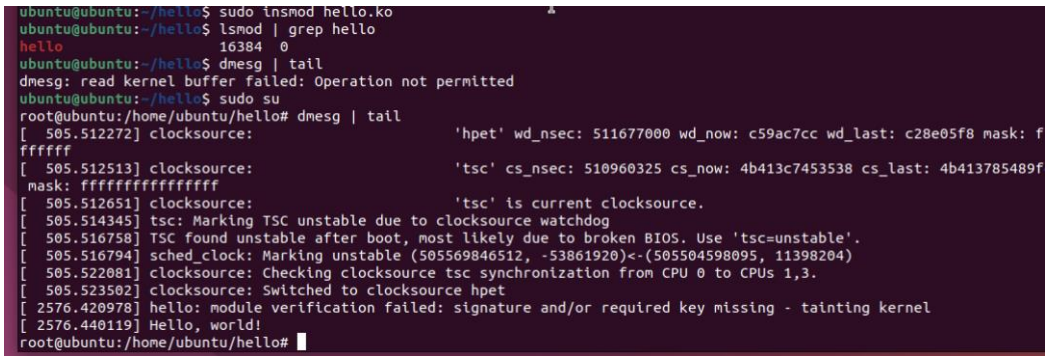
```
obj-m += hello.o

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

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

This Makefile specifies that the target is the hello.o module, and it uses the make command to build the module. The clean target can be used to remove any build artifacts.

Then we compile the module and run the make command in the module directory to build the module.
We use the insmod command to load the module into the kernel
Then we used the lsmod command to verify that the module is loaded



After that we are running the dmesg command to view the kernel log and verify that the "Hello, world!" message was printed:



We used the rmmod command to unload the module
And then the lsmod command to verify that the module is no longer loaded: