

Template Final Exam

#	Full Name	Group
1	Abylkair Shaigaliyev	CS-2111
2	Leila Zhuruntayeva	CS-2111
3	Bodambaev Tamerlan	CS-2111

Link to the repository:
github.com/ldhvh/final_osc

Introduction:

Ubuntu, a Linux distribution that is well-known for its user-friendly interface and flexibility, is widely used across various industries. As an open-source operating system, Ubuntu offers a plethora of tools and functions that allow users to perform a wide range of tasks. This article will focus on three distinct tasks that can be carried out in Ubuntu: adding a user and assigning root rights, implementing different methods of connecting to the Internet on virtual machines, and developing a Linux kernel module.

The first task involves adding a new user to the system and granting them root rights. Adding a user to Ubuntu is a straightforward process that can be done through the command line interface. After adding a user, it may be necessary to assign them superuser privileges to perform certain tasks that require elevated permissions. This task is essential as it allows multiple users to access the system while providing an additional layer of security.

The second task involves configuring a virtual machine and implementing three different methods of connecting to the Internet: a direct IP connection, a connection via NAT, and a connection via a proxy server. By implementing these methods, users can gain a better understanding of how virtual networking works in Ubuntu and how to configure various network settings.

The third task involves developing a Linux kernel module, which is a more advanced task that requires a deeper understanding of the operating system's underlying mechanisms. A kernel module is a software component that can be loaded into the Linux kernel to modify or add new functionality. By developing a kernel module, users can tailor their Ubuntu system to meet their specific needs.

Overall, these three tasks provide insight into the versatility and power of Ubuntu as an operating system. Whether you are a novice or an experienced user, Ubuntu offers numerous opportunities for exploration and learning.

Step-by-step task completion:*Task 1:**Add a new user to the system*

To create a new user, you need to register the “sudo adduser student” command, after which we fill out a questionnaire for our user. In our case, we will call the user a student. When creating a user, it is necessary for him to create a password with a length of 8 characters.

```
abylkair@abylkair-VirtualBox:~$ sudo adduser student
Adding user `student' ...
Adding new group `student' (1001) ...
Adding new user `student' (1001) with group `student' ...
Creating home directory `/home/student' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for student
Enter the new value, or press ENTER for the default
  Full Name []: Abylkair Shaigaliyev
  Room Number []: 24
  Work Phone []: 7-707-525-31-80
  Home Phone []: 60-50-57
  Other []: Hello
Is the information correct? [Y/n] y
```

To give root rights, the user must register `sudo usermod -aG sudo student`.

To log in as a student user, we prescribe the command “sudo su student”

```
abylkair@abylkair-VirtualBox:~$ sudo usermod -aG sudo student
abylkair@abylkair-VirtualBox:~$ sudo su student
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

student@abylkair-VirtualBox:/home/abylkair$
```

As you can see, the username has changed to “student”, i.e. we can say we logged in as “student” and to make sure of this, we prescribe the “whoami” command. and in the end, you can exit by writing the “exit” command

```
student@abylkair-VirtualBox:/home/abylkair$ whoami
student
student@abylkair-VirtualBox:/home/abylkair$ exit
exit
abylkair@abylkair-VirtualBox:~$
```

Now you know how to add a user. if you want to delete a user you just need to write the command “userdel username”.

```
userdel: cannot lock /etc/passwd; try again later.
abylkair@abylkair-VirtualBox:~$ sudo userdel student
abylkair@abylkair-VirtualBox:~$
```

```
abylkair@abylkair-VirtualBox:~$ sudo su student
su: user student does not exist or the user entry does not contain all the required fields
```

*Task 2:**"Direct" IP connection to the Internet*

To connect to the Internet, through the command line, we must perform the following steps:

- 1) We need to see the internet devices of the system. Therefore, we execute the command "ip link show". This command will show us "lo" (loop device) and "enp0s3" (Internet network card).

```
tamelan@tamelan-VirtualBox:~$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAUL
T 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 m
ode DEFAULT group default qlen 1000
    link/ether 08:00:27:32:41:18 brd ff:ff:ff:ff:ff:ff
```

- 2) Install "dhcpcd". This is DHCP. It automatically assigns an IP address.

```
tamelan@tamelan-VirtualBox:~$ sudo apt install dhcpcd5
[sudo] password for tamelan:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  openresolv
Suggested packages:
  dhcpcd-gtk
The following NEW packages will be installed:
  dhcpcd5 openresolv
0 upgraded, 2 newly installed, 0 to remove and 71 not upgraded.
Need to get 188 kB of archives.
After this operation, 510 kB of additional disk space will be used.
Do you want to continue? [Y/n]
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/universe amd64 dhcpcd5 amd64 7.
1.0-2build1 [163 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy/universe amd64 openresolv all 3
.12.0-2 [25.6 kB]
Fetched 188 kB in 2s (98.7 kB/s)
Selecting previously unselected package dhcpcd5.
(Reading database ... 200671 files and directories currently installed.)
Preparing to unpack .../dhcpcd5_7.1.0-2build1_amd64.deb ...
Unpacking dhcpcd5 (7.1.0-2build1) ...
Selecting previously unselected package openresolv.
Preparing to unpack .../openresolv_3.12.0-2_all.deb ...
```

- 3) Execute the command "sudo dhcpcd enp0s3". It will solicit an ipv6 for our system.

```
tamelan@tamelan-VirtualBox:~$ sudo dhcpcd enp0s3
DUID 00:04:b4:48:10:a3:71:08:7e:47:a0:4e:d6:c9:1f:79:5c:6c
enp0s3: IAID 27:32:41:18
enp0s3: soliciting an IPv6 router
enp0s3: soliciting a DHCP lease
enp0s3: offered 10.0.2.15 from 10.0.2.2
enp0s3: leased 10.0.2.15 for 86400 seconds
enp0s3: adding route to 10.0.2.0/24
enp0s3: adding default route via 10.0.2.2
forked to background, child pid 3286
```

- 4) Connecting with the "ping" command. Thanks to this, we can check the stability of the network, and the success of sending and receiving 5 packets.

```
tamelan@tamelan-VirtualBox:~$ ping google.com -c5
PING google.com (173.194.220.138) 56(84) bytes of data.
64 bytes from lk-in-f138.1e100.net (173.194.220.138): icmp_seq=1 ttl=102 time=5
5.0 ms
64 bytes from lk-in-f138.1e100.net (173.194.220.138): icmp_seq=2 ttl=102 time=6
4.7 ms
64 bytes from lk-in-f138.1e100.net (173.194.220.138): icmp_seq=3 ttl=102 time=6
4.0 ms
64 bytes from lk-in-f138.1e100.net (173.194.220.138): icmp_seq=4 ttl=102 time=5
4.2 ms
64 bytes from lk-in-f138.1e100.net (173.194.220.138): icmp_seq=5 ttl=102 time=5
6.2 ms

--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 54.211/58.816/64.688/4.567 ms
```

Connection via NAT

NAT (Network Address Translation) is a method for computer networks, thanks to which several devices in one private network share one public IP address to access the Internet.

- 1) We need to use “nano” redactor to get into the “sysctl.conf” file and inside it set the “net.ipv4.ip_forward” parameter to one by uncommenting it

```
tamelan@tamelan-VirtualBox:~$ sudo nano /etc/sysctl.conf
```

```
tamelan@tamelan-VirtualBox: ~
GNU nano 6.2 /etc/sysctl.conf *
# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1

#####
# Additional settings - these settings can improve the network
# security of the host and prevent against some network attacks
# including spoofing attacks and man in the middle attacks through
# redirection. Some network environments, however, require that these

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

- 2) Installation of "iptables-persistent". It is a command-line utility for managing firewall rules.

```
tamelan@tamelan-VirtualBox:~$ sudo apt install iptables-persistent
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  netfilter-persistent
The following NEW packages will be installed:
  iptables-persistent netfilter-persistent
0 upgraded, 2 newly installed, 0 to remove and 71 not upgraded.
```

The command "sudo ip tables - L" was used to enumerate already configured iptable policies.

```
tamelan@tamelan-VirtualBox:~$ 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
```

- 3) Send requests from the local network with the external IP address of the conditional NAT route. After that, save the new iptables rules in the system.

```
tamelan@tamelan-VirtualBox:~$ sudo iptables -t nat -A POSTROUTING -j MASQUERADE
tamelan@tamelan-VirtualBox:~$ 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
tamelan@tamelan-VirtualBox:~$ sudo sh -c "iptables-save > /etc/iptables/rules.v4"
sh: 1: iptables-save: not found
tamelan@tamelan-VirtualBox:~$ sudo sh -c "iptables-save > /etc/iptables/rules.v4"
```

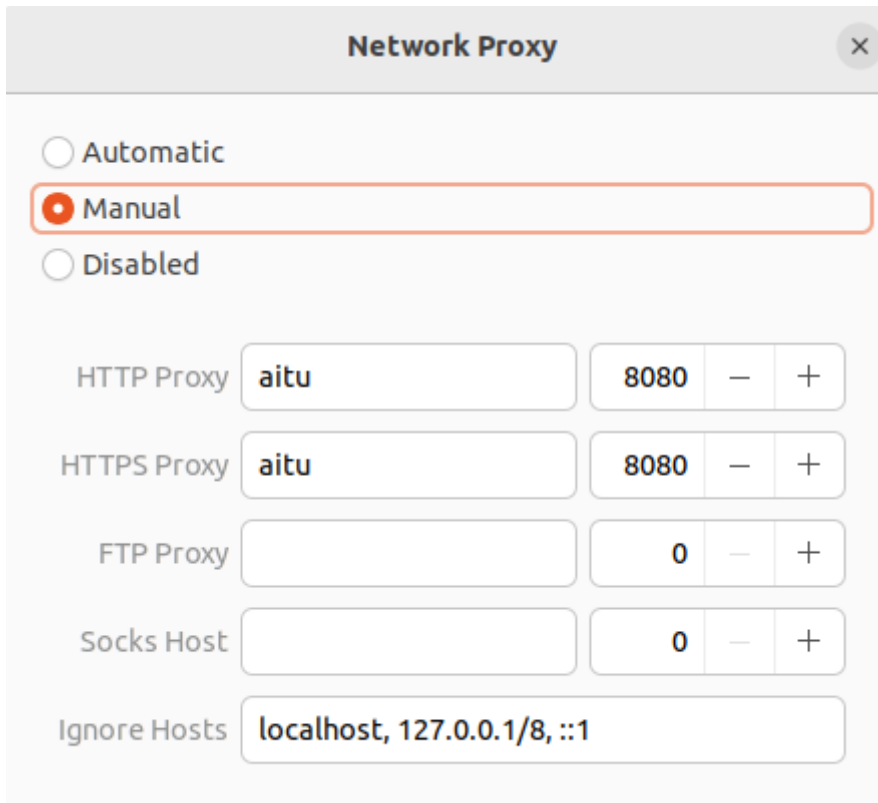
After completing all these steps, the IP address of the device will be replaced by an external router when sending requests to the network. This provides protection by hiding the republican address and storing IP addresses in the local network.

Connection via a proxy server

A proxy server is an intermediary server. It acts as a gateway between the client and the server.

You can manually configure the connection via a proxy server.

- 1) You can configure it in settings > network > proxy network
- 2) Write Proxy server IP and port



The image shows a 'Network Proxy' settings window. It has three radio buttons: 'Automatic', 'Manual' (which is selected and highlighted with a red border), and 'Disabled'. Below these are five rows of settings:

Proxy Type	Host	Port	Minus	Plus
HTTP Proxy	aitu	8080	-	+
HTTPS Proxy	aitu	8080	-	+
FTP Proxy		0	-	+
Socks Host		0	-	+
Ignore Hosts	localhost, 127.0.0.1/8, ::1			

- 3) Execute the command "nano ~/.bashrc" to open the file "~/.bashrc" by "nano"

```
tamelan@tamelan-VirtualBox:~$ nano ~/.bashrc
```

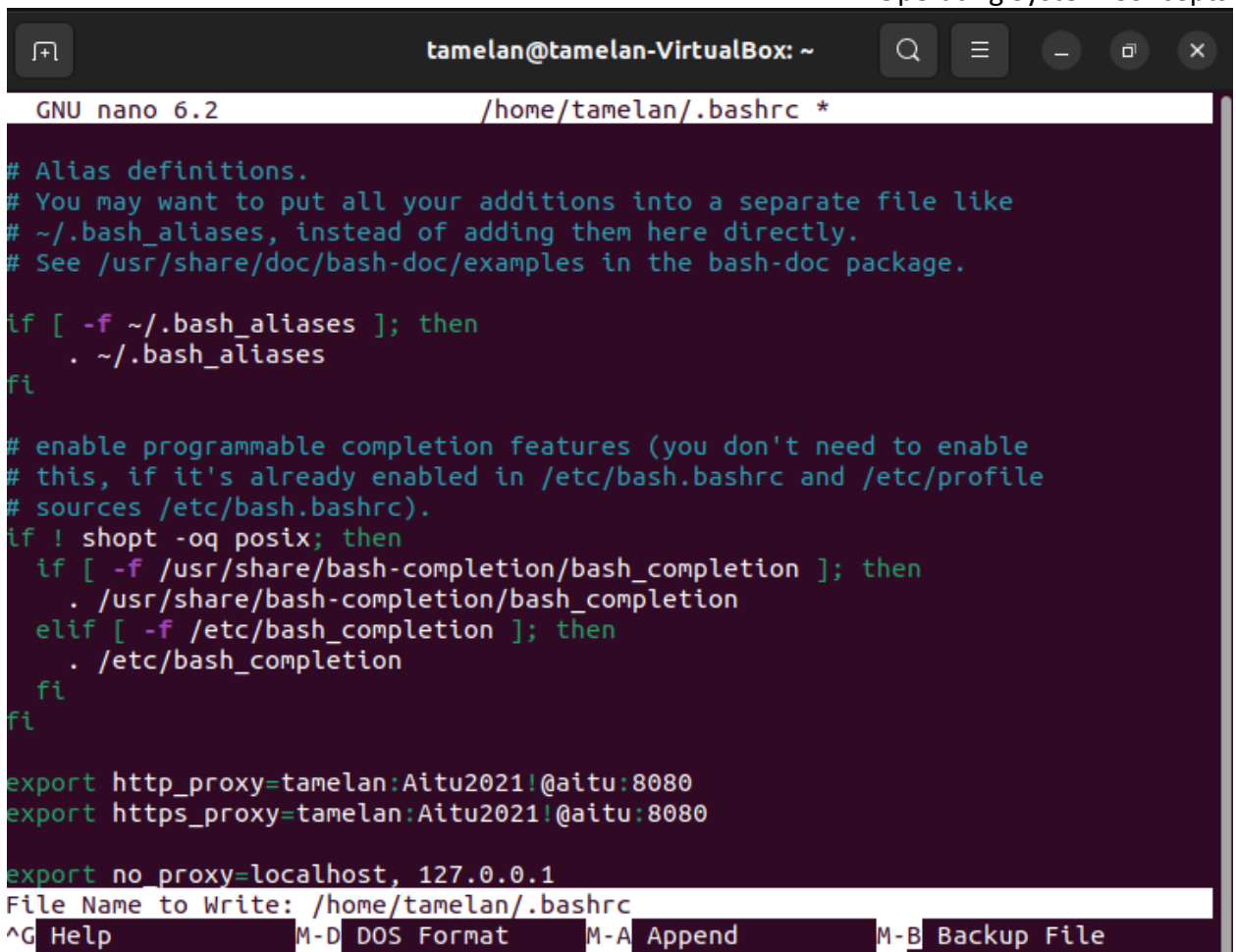
- 4) insert one of the following commands at the end of the file:

```
export http_proxy=username:password@proxy-server-ip:8080
export https_proxy=username:password@proxy-server-ip:8080
export ftp_proxy=username:password@proxy-server-ip:8080
```

and

```
export no_proxy=localhost, 127.0.0.1
```

the save it



```
GNU nano 6.2 /home/tamelan/.bashrc *  
  
# Alias definitions.  
# You may want to put all your additions into a separate file like  
# ~/.bash_aliases, instead of adding them here directly.  
# See /usr/share/doc/bash-doc/examples in the bash-doc package.  
  
if [ -f ~/.bash_aliases ]; then  
    . ~/.bash_aliases  
fi  
  
# enable programmable completion features (you don't need to enable  
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile  
# sources /etc/bash.bashrc).  
if ! shopt -oq posix; then  
    if [ -f /usr/share/bash-completion/bash_completion ]; then  
        . /usr/share/bash-completion/bash_completion  
    elif [ -f /etc/bash_completion ]; then  
        . /etc/bash_completion  
    fi  
fi  
  
export http_proxy=tamelan:Aitu2021!@aitu:8080  
export https_proxy=tamelan:Aitu2021!@aitu:8080  
  
export no_proxy=localhost, 127.0.0.1  
File Name to Write: /home/tamelan/.bashrc  
^G Help M-D DOS Format M-A Append M-B Backup File
```

- 5) Use the "source ~/.bashrc" command to activate your new proxy server settings for the current session.

Task 3: Development of the Linux OS kernel module

To compile our own Linux kernel, the first step is to obtain the source code of the desired version of the kernel, which should be freely available as open-source software.

```
user@user-virtual-machine:~$ wget https://cdn.kernel.org/pub/linux/kernel/v6.x/
linux-6.1.12.tar.xz
--2023-02-25 09:26:56-- https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.1
.12.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 146.75.121.176, 2a04:4e42:8e::432
Connecting to cdn.kernel.org (cdn.kernel.org)|146.75.121.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 134848688 (129M) [application/x-xz]
Saving to: 'linux-6.1.12.tar.xz.1'

linux-6.1.12.tar.xz 100%[=====] 128,60M 8,08MB/s in 16s

2023-02-25 09:27:13 (7,81 MB/s) - 'linux-6.1.12.tar.xz.1' saved [134848688/1348
48688]

user@user-virtual-machine:~$ ls
Desktop  Downloads  linux-6.1.12.tar.xz.1  Public  Videos
develop  linux-6.1.12  Music                 snap
Documents linux-6.1.12.tar.xz  Pictures              Templates
user@user-virtual-machine:~$ tar xvf linux-6.1.12.tar.xz
```

downloading kernel source code

Next, we have to discover what modules are already loaded within the kernel:

```
user@user-virtual-machine:~$ cd linux-6.1.12
user@user-virtual-machine:~/linux-6.1.12$ ls -l
total 968
drwxrwxr-x 24 user user 4096 Apr 15 00:11 arch
```

files list

The primary method for configuring the Linux kernel is through a document configuration file that includes instructions for compiling the kernel, such as kernel functions and modules.

```
user@user-virtual-machine:~/linux-6.1.12$ sudo apt-get install pkg-config
[sudo] password for user:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
pkg-config is already the newest version (0.29.2-1ubuntu3).
user@user-virtual-machine:~/linux-6.1.12$ sudo apt-get install libncurses5-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
libncurses5-dev is already the newest version (6.3-2).
```

configuration settings

The kernel configuration process itself involves selecting the options and features that you want to include in the kernel. This is done by running the "make menuconfig" command, which opens an interactive menu where you can select the desired configuration options.

```
user@user-virtual-machine:~/linux-6.1.12$ make menuconfig -j12
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/confdata.o
HOSTCC scripts/kconfig/expr.o
LEX scripts/kconfig/lexer.lex.c
```

creating configuration file

Next, we have to configure the kernel and its modules through the two following commands:

```
user@user-virtual-machine:~/linux-6.1.12$ make -j12 && sudo make modules_install -j12
***
```

kernel and its modules

The process for installing the kernel headers varies depending on your Linux distribution. In many cases, you can install the headers using the distribution's package manager. Here the command inserts the current kernel version into the package name, so that the correct headers are installed.

```
user@user-virtual-machine:~/Downloads$ sudo apt-get install build-essential linux-headers-$(uname -r)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
build-essential is already the newest version (12.9ubuntu3).
```

install headers

Then, we create new directory for the module with the simple function of displaying "Hello world".

```
user@user-virtual-machine:~/Desktop$ mkdir ~/kernel_module
user@user-virtual-machine:~/Desktop$ cd ~kernel_module
bash: cd: ~kernel_module: No such file or directory
user@user-virtual-machine:~/Desktop$ cd ~/kernel_module
user@user-virtual-machine:~/kernel_module$ nano first.c
user@user-virtual-machine:~/kernel_module$ nano first.c
user@user-virtual-machine:~/kernel_module$ nano Makefile
```

Modules, C files and makefiles

The contents of the files and as follows:

```
GNU nano 6.2 first.c
#include <linux/init.h>
#include <linux/module.h>

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

static void __exit bye_init(void)
{
    print("Goodbye world!\n");
}

module_init(hello_init);
module_exit(bye_init);

MODULE_LICENSE("GPL");
```

first.c

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

Makefile

After configuring the kernel, run the "make" command to build the kernel image. This process can take several minutes to complete. In the screenshot, there can be seen changes in the directory before and after running the command:

```
user@user-virtual-machine:~/kernel_module$ ls
first.c  Makefile
user@user-virtual-machine:~/kernel_module$ nano first.c
user@user-virtual-machine:~/kernel_module$ make
make -C /lib/modules/5.19.0-32-generic/build M=/home/user/kernel_module modules
make[1]: Entering directory '/usr/src/linux-headers-5.19.0-32-generic'
warning: the compiler differs from the one used to build the kernel
The kernel was built by: x86_64-linux-gnu-gcc (Ubuntu 11.3.0-1ubuntu1~22.04)
11.3.0
You are using:          gcc (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0
CC [M] /home/user/kernel_module/first.o
MODPOST /home/user/kernel_module/Module.symvers
CC [M] /home/user/kernel_module/first.mod.o
LD [M] /home/user/kernel_module/first.ko
BTF [M] /home/user/kernel_module/first.ko
Skipping BTF generation for /home/user/kernel_module/first.ko due to unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-5.19.0-32-generic'
user@user-virtual-machine:~/kernel_module$ ls
first.c  first.mod  first.mod.o  Makefile      Module.symvers
first.ko first.mod.c  first.o      modules.order
```

make

Finally, we install the module into kernel and check the message in the log:

```
user@user-virtual-machine:~/kernel_module$ sudo insmod first.ko
[sudo] password for user:
user@user-virtual-machine:~/kernel_module$ sudo dmesg | tail
[ 989.360100] smpboot_thread_fn+0xe0/0x1e0
[ 989.360103] kthread+0xeb/0x120
[ 989.360106] ? kthread_complete_and_exit+0x20/0x20
[ 989.360110] ret_from_fork+0x1f/0x30
[ 989.360115] </TASK>
[ 1005.522897] usb 2-2.1: reset full-speed USB device number 4 using uhci_hcd
[ 1005.687077] psmouse serio1: resync failed, issuing reconnect request
[ 1785.382930] first: loading out-of-tree module taints kernel.
[ 1785.386956] first: module verification failed: signature and/or required key
missing - tainting kernel
[ 1785.402995] Hello world!
```

kernel log

Next, we install the module taking two numbers and outputting their quotient and remainder.

```

user@user-virtual-machine:~/Desktop$ nano second.c
GNU nano 6.2 second.c
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>

MODULE_LICENSE("GPL");

static int div = 25;
static int dis = 4;

static int __init find_quo_rem_init(void)
{
    int quo = div / dis;
    int rem = div % dis;
    printk(KERN_INFO "The quotient and remainder of %d and %d are %d and %d\n",
    return 0;
}

static void __exit find_quo_rem_exit(void)
{
    printk(KERN_INFO "Exiting module\n");
}

module_init(find_quo_rem_init);
module_exit(find_quo_rem_exit);
    
```

second.c

```

user@user-virtual-machine:~/Desktop$ nano Makefile
GNU nano 6.2 Makefile
obj-m += second.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
    
```

Makefile

```

user@user-virtual-machine:~/Desktop$ make
make -C /lib/modules/5.19.0-32-generic/build M=/home/user/Desktop modules
make[1]: Entering directory '/usr/src/linux-headers-5.19.0-32-generic'
warning: the compiler differs from the one used to build the kernel
The kernel was built by: x86_64-linux-gnu-gcc (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3
.0
You are using: gcc (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0
CC [M] /home/user/Desktop/second.o
MODPOST /home/user/Desktop/Module.symvers
CC [M] /home/user/Desktop/second.mod.o
LD [M] /home/user/Desktop/second.ko
BTF [M] /home/user/Desktop/second.ko
Skipping BTF generation for /home/user/Desktop/second.ko due to unavailability of v
Show Applications directory '/usr/src/linux-headers-5.19.0-32-generic'
    
```

build the kernel

```

user@user-virtual-machine:~/Desktop$ nano second.c
user@user-virtual-machine:~/Desktop$ nano Makefile
user@user-virtual-machine:~/Desktop$ sudo insmod second.ko
user@user-virtual-machine:~/Desktop$ sudo dmesg | tail
[ 3576.228020] RSP: 002b:00007f844cc68218 EFLAGS: 00000202
[ 3576.228022] RAX: 0000000000000000 RBX: 0000000000000260 RCX: 0000000000000001
[ 3576.228023] RDX: 0000000000000000 RSI: 0000000000000001 RDI: 00007f844e115fe0
[ 3576.228024] RBP: 00007f844cc68248 R08: 0000000000000014 R09: 000000000000000b
[ 3576.228025] R10: 0000555d3ce80d88 R11: 0000000000000000 R12: 0000000000000025
[ 3576.228025] R13: 00007f8440000b70 R14: 0000000000000025 R15: 00007f84400011c0
[ 3576.228029] </TASK>
[ 3593.416326] usb 2-2.1: reset full-speed USB device number 4 using uhci_hcd
[ 3593.492501] psmouse serio1: resync failed, issuing reconnect request
[ 4151.462663] The quotient and remainder of 25 and 4 are 6 and 1
    
```

install module and check kernel log

Conclusion

Our team's exploration of Linux has been an enriching and diverse experience. We began by learning how to add a new user to the system, an essential step in ensuring access control and security. We then progressed to exploring three different ways of connecting to the internet, including direct IP, via NAT, and proxy server, which has allowed us to gain a deeper understanding of networking in Linux.

Additionally, we had the opportunity to develop a Linux kernel module, which was a challenging but rewarding experience. We learned how to configure and compile the kernel, as well as how to write code that interfaces with the kernel. This project deepened our understanding of the Linux operating system and the underlying technology that powers it.

Overall, our team's experiences have equipped us with valuable skills and knowledge that are essential in our daily work with Linux. We have developed a deeper appreciation for the versatility and complexity of the operating system. We look forward to continuing to contribute to the Linux community and exploring more advanced topics in the future.