Operating System Project

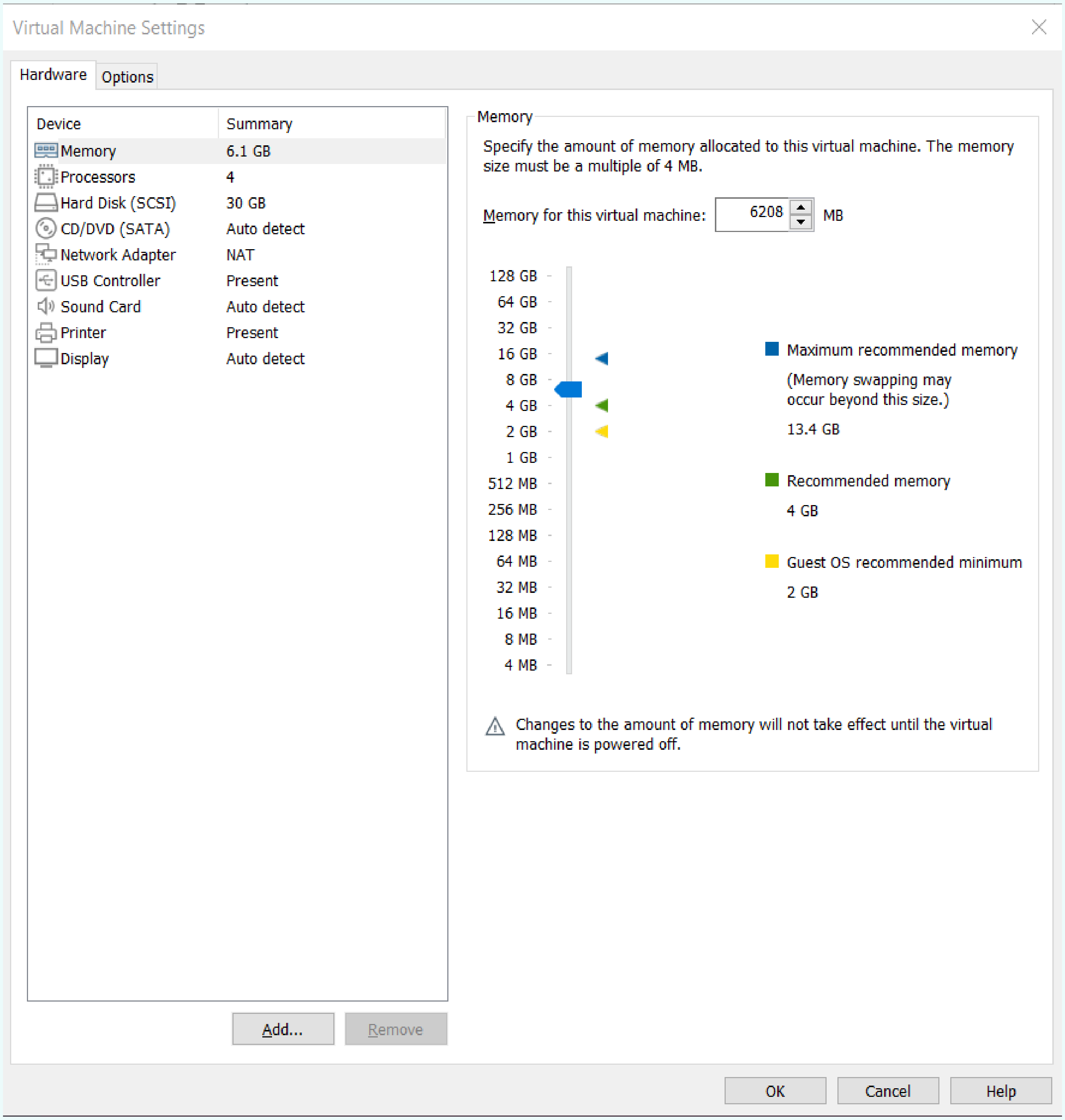
Mostafa Medhat Mohamed
Mohamed Essam Sayed
Yusuf Ahmed Yusuf

How to add your system call the Linux OS kernel.

Operating System Project

System Descriptions:

Ram: 6 GB
Processors: 4 cores
Hard Disk: 30 GB
Kernel Version: 5.8.0-55-generic



Device	Summary
Memory	6.1 GB
Processors	4
Hard Disk (SCSI)	30 GB
CD/DVD (SATA)	Auto detect
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB.

Memory for this virtual machine: 6208 MB

128 GB, 64 GB, 32 GB, 16 GB, 8 GB, 4 GB, 2 GB, 1 GB, 512 MB, 256 MB, 128 MB, 64 MB, 32 MB, 16 MB, 8 MB, 4 MB

🟦 Maximum recommended memory
(Memory swapping may occur beyond this size.)
13.4 GB

🟩 Recommended memory
4 GB

🟨 Guest OS recommended minimum
2 GB

⚠ Changes to the amount of memory will not take effect until the virtual machine is powered off.

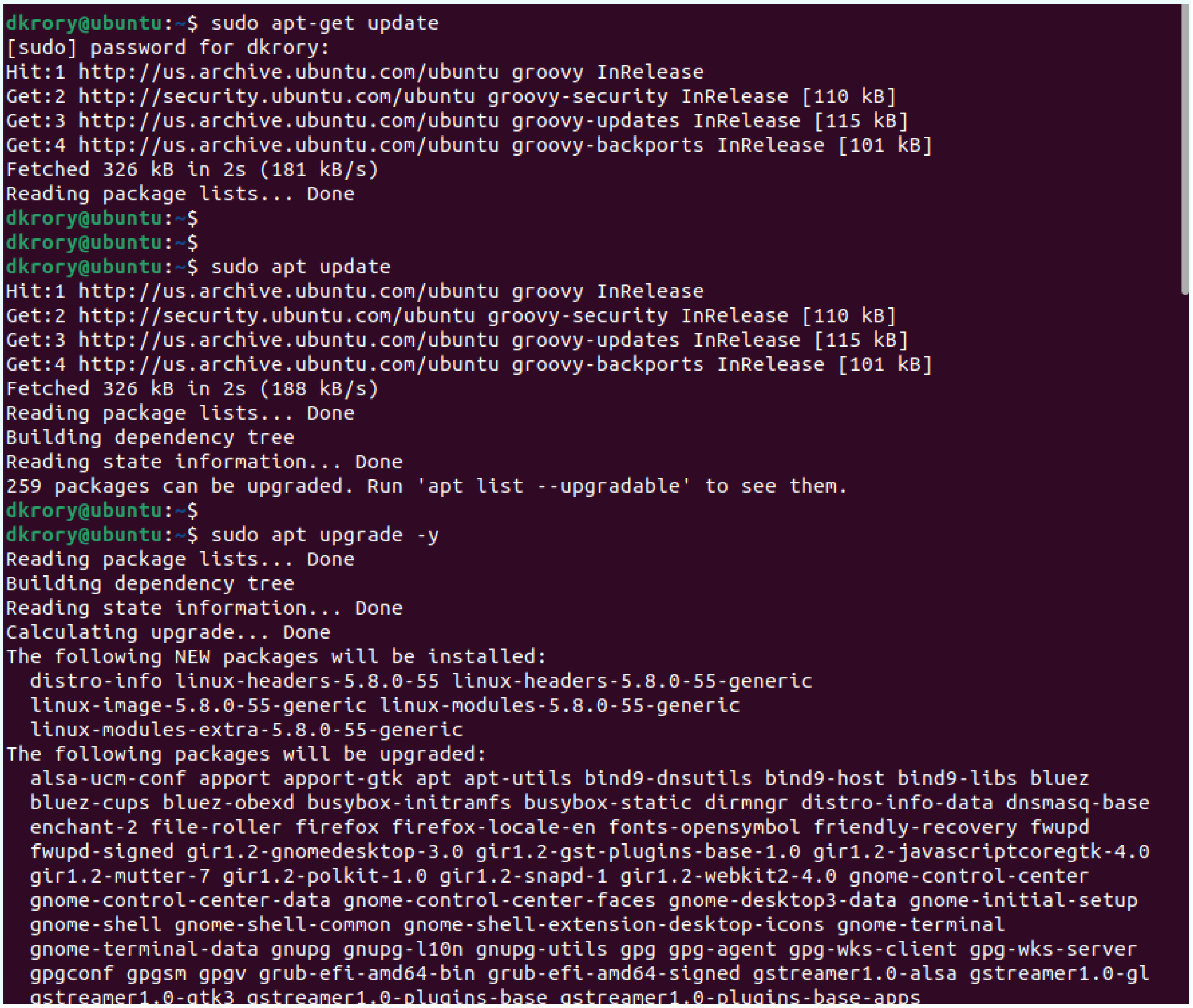
Steps of How Adding the system call:

Section 1 – Preparation

In this section, you will download all necessary tools to add a basic system call to the Linux kernel and run it. This is the only part of the entire process where network connectivity is necessary.

1.1 - Fully update your operating system.

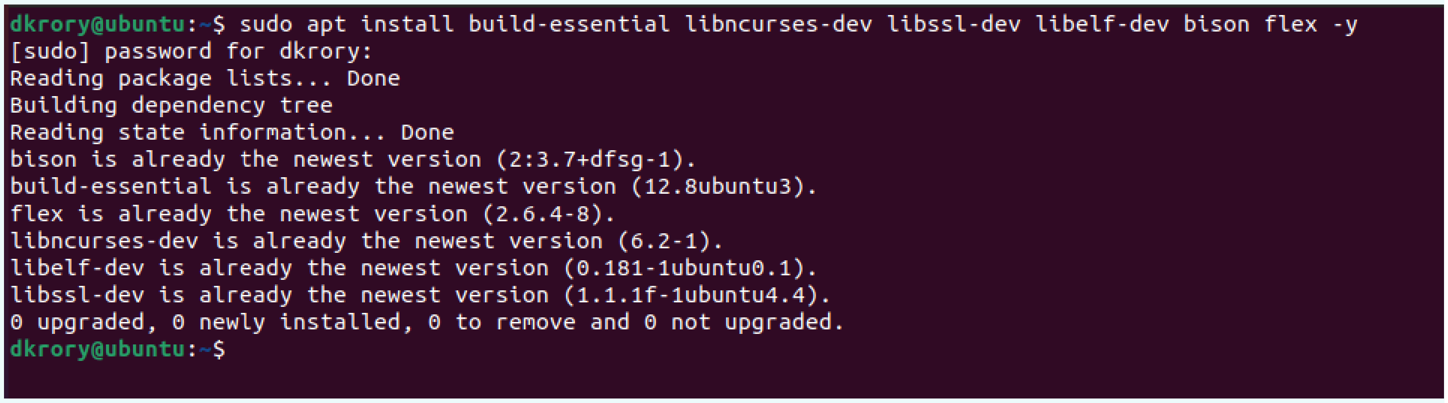
```
sudo apt update && sudo apt upgrade -y
```



```
dkrory@ubuntu:~$ sudo apt-get update
[sudo] password for dkrory:
Hit:1 http://us.archive.ubuntu.com/ubuntu groovy InRelease
Get:2 http://security.ubuntu.com/ubuntu groovy-security InRelease [110 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu groovy-updates InRelease [115 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu groovy-backports InRelease [101 kB]
Fetched 326 kB in 2s (181 kB/s)
Reading package lists... Done
dkrory@ubuntu:~$
dkrory@ubuntu:~$
dkrory@ubuntu:~$ sudo apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu groovy InRelease
Get:2 http://security.ubuntu.com/ubuntu groovy-security InRelease [110 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu groovy-updates InRelease [115 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu groovy-backports InRelease [101 kB]
Fetched 326 kB in 2s (188 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
259 packages can be upgraded. Run 'apt list --upgradable' to see them.
dkrory@ubuntu:~$
dkrory@ubuntu:~$ sudo apt upgrade -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following NEW packages will be installed:
  distro-info linux-headers-5.8.0-55 linux-headers-5.8.0-55-generic
  linux-image-5.8.0-55-generic linux-modules-5.8.0-55-generic
  linux-modules-extra-5.8.0-55-generic
The following packages will be upgraded:
  alsa-ucm-conf apport apport-gtk apt apt-utils bind9-dnsutils bind9-host bind9-libs bluez
  bluez-cups bluez-obexd busybox-initramfs busybox-static dirmngr distro-info-data dnsmasq-base
  enchant-2 file-roller firefox firefox-locale-en fonts-opensymbol friendly-recovery fwupd
  fwupd-signed gir1.2-gnomedesktop-3.0 gir1.2-gst-plugins-base-1.0 gir1.2-javascriptcoregtk-4.0
  gir1.2-mutter-7 gir1.2-polkit-1.0 gir1.2-snapd-1 gir1.2-webkit2-4.0 gnome-control-center
  gnome-control-center-data gnome-control-center-faces gnome-desktop3-data gnome-initial-setup
  gnome-shell gnome-shell-common gnome-shell-extension-desktop-icons gnome-terminal
  gnome-terminal-data gnupg gnupg-l10n gnupg-utils gpg gpg-agent gpg-wks-client gpg-wks-server
  gpgconf gpgsm gpgv grub-efi-amd64-bin grub-efi-amd64-signed gstreamer1.0-alsa gstreamer1.0-gl
  gstreamer1.0-gtk3 gstreamer1.0-plugins-base gstreamer1.0-plugins-base-apps
```

1.2 - Download and install the essential packages to compile kernels.

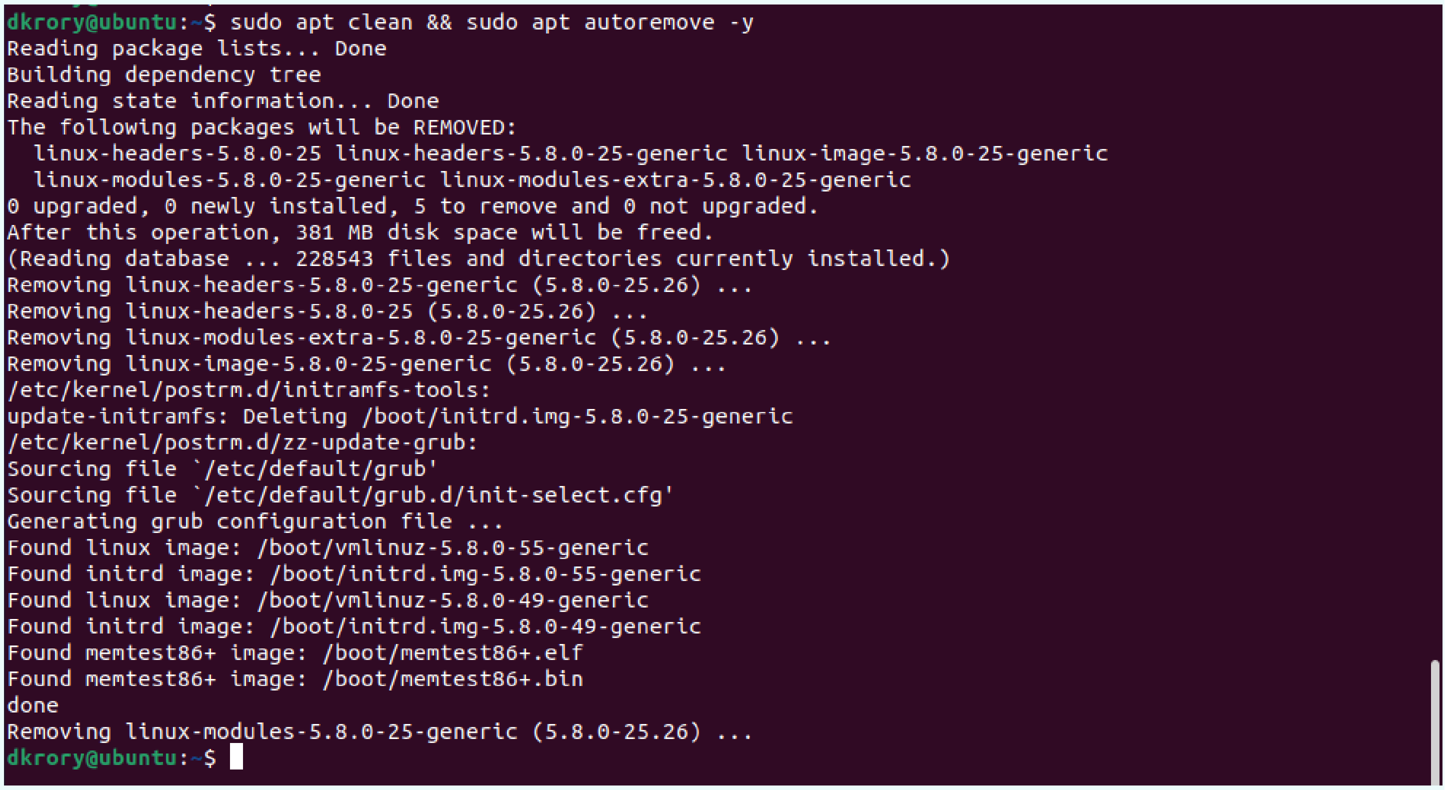
sudo apt install build-essential libncurses-dev libssl-dev libelf-dev bison flex -y



```
dkrory@ubuntu:~$ sudo apt install build-essential libncurses-dev libssl-dev libelf-dev bison flex -y
[sudo] password for dkrory:
Reading package lists... Done
Building dependency tree
Reading state information... Done
bison is already the newest version (2:3.7+dfsg-1).
build-essential is already the newest version (12.8ubuntu3).
flex is already the newest version (2.6.4-8).
libncurses-dev is already the newest version (6.2-1).
libelf-dev is already the newest version (0.181-1ubuntu0.1).
libssl-dev is already the newest version (1.1.1f-1ubuntu4.4).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
dkrory@ubuntu:~$
```

1.3 - Clean up your installed packages.

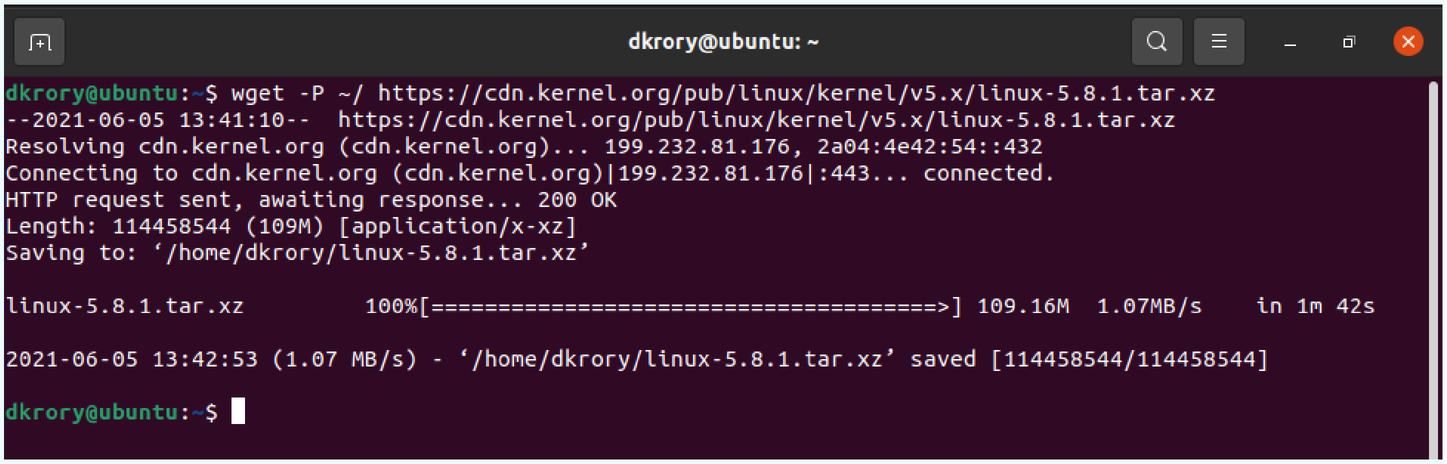
sudo apt clean && sudo apt autoremove -y



```
dkrory@ubuntu:~$ sudo apt clean && sudo apt autoremove -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  linux-headers-5.8.0-25 linux-headers-5.8.0-25-generic linux-image-5.8.0-25-generic
  linux-modules-5.8.0-25-generic linux-modules-extra-5.8.0-25-generic
0 upgraded, 0 newly installed, 5 to remove and 0 not upgraded.
After this operation, 381 MB disk space will be freed.
(Reading database ... 228543 files and directories currently installed.)
Removing linux-headers-5.8.0-25-generic (5.8.0-25.26) ...
Removing linux-headers-5.8.0-25 (5.8.0-25.26) ...
Removing linux-modules-extra-5.8.0-25-generic (5.8.0-25.26) ...
Removing linux-image-5.8.0-25-generic (5.8.0-25.26) ...
/etc/kernel/postrm.d/initramfs-tools:
update-initramfs: Deleting /boot/initrd.img-5.8.0-25-generic
/etc/kernel/postrm.d/zz-update-grub:
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.8.0-55-generic
Found initrd image: /boot/initrd.img-5.8.0-55-generic
Found linux image: /boot/vmlinuz-5.8.0-49-generic
Found initrd image: /boot/initrd.img-5.8.0-49-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
Removing linux-modules-5.8.0-25-generic (5.8.0-25.26) ...
dkrory@ubuntu:~$
```

1.4 - Download the source code of the latest stable version of the Linux kernel.

```
wget -P ~/ https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.8.1.tar.xz
```



```
dkrory@ubuntu:~$ wget -P ~/ https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.8.1.tar.xz
--2021-06-05 13:41:10--  https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.8.1.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 199.232.81.176, 2a04:4e42:54::432
Connecting to cdn.kernel.org (cdn.kernel.org)|199.232.81.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 114458544 (109M) [application/x-xz]
Saving to: '/home/dkrory/linux-5.8.1.tar.xz'

linux-5.8.1.tar.xz         100%[======================================>] 109.16M  1.07MB/s    in 1m 42s

2021-06-05 13:42:53 (1.07 MB/s) - '/home/dkrory/linux-5.8.1.tar.xz' saved [114458544/114458544]

dkrory@ubuntu:~$ 
```

1.5 - Unpack the tarball you just downloaded to your home folder.

tar -xvf ~/linux-5.8.1.tar.xz -C ~/

1.6 - Reboot your computer.

Section 2 - Creation

In this section, you will write a basic system call in C and integrate it into the new kernel.

2.1 - Check the version of your current kernel.

```
uname -r
```

2.2 - Change your working directory to the root directory of the recently unpacked source code.

```
cd ~/linux-5.8.1/
```

2.3 - Create the home directory of your system call.

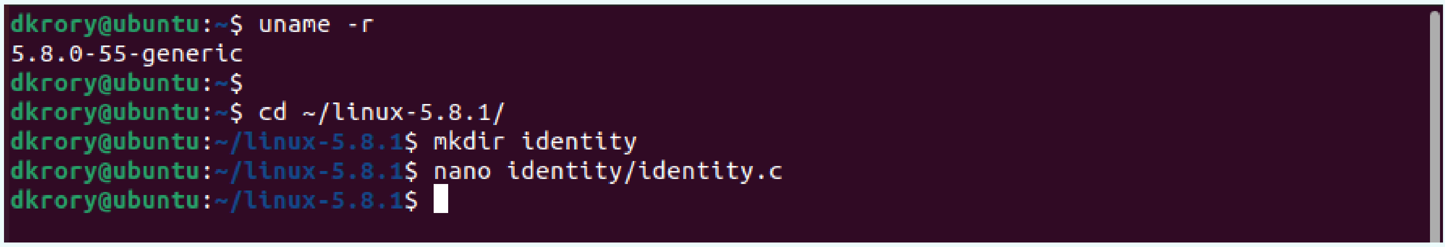
Decide a name for your system call, and keep it consistent from this point onwards. I have chosen identity.

```
mkdir identity
```

2.4 - Create a C file for your system call.

Create the C file with the following command.

```
nano identity/identity.c
```



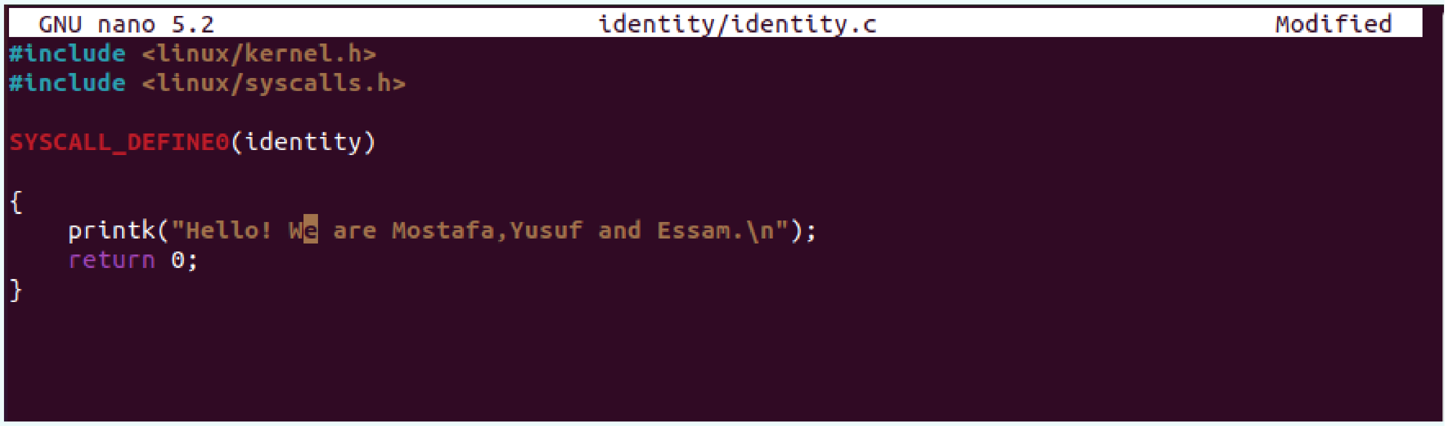
```
dkrory@ubuntu:~$ uname -r
5.8.0-55-generic
dkrory@ubuntu:~$
dkrory@ubuntu:~$ cd ~/linux-5.8.1/
dkrory@ubuntu:~/linux-5.8.1$ mkdir identity
dkrory@ubuntu:~/linux-5.8.1$ nano identity/identity.c
dkrory@ubuntu:~/linux-5.8.1$
```

Write the following code in it.

```
#include <linux/kernel.h>
#include <linux/syscalls.h>
SYSCALL_DEFINE0(identity){
    printk("I am Jihan Jasper Al-rashid.\n");
    return 0;
}
```

You can write anything you like here.

Save it and exit the text editor.



```
  GNU nano 5.2                 identity/identity.c                 Modified
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE0(identity)

{
    printk("Hello! We are Mostafa,Yusuf and Essam.\n");
    return 0;
}
```

2.5 - Create a Makefile for your system call.

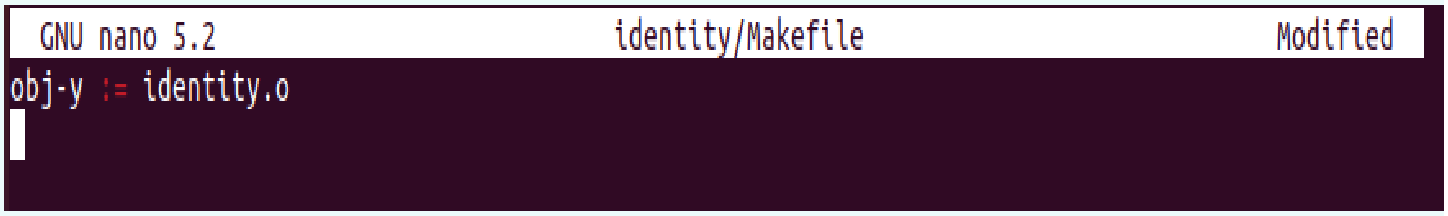
Create the Makefile with the following command.

nano identity/Makefile

Write the following code in it.

obj-y := identity.o

Save it and exit the text editor.



```
  GNU nano 5.2                      identity/Makefile                        Modified
obj-y := identity.o
```

2.6 - Add the home directory of your system call to the main Makefile of the kernel.

Open the Makefile with the following command.

nano Makefile

Search for core-y. In the second result, you will see a series of directories.

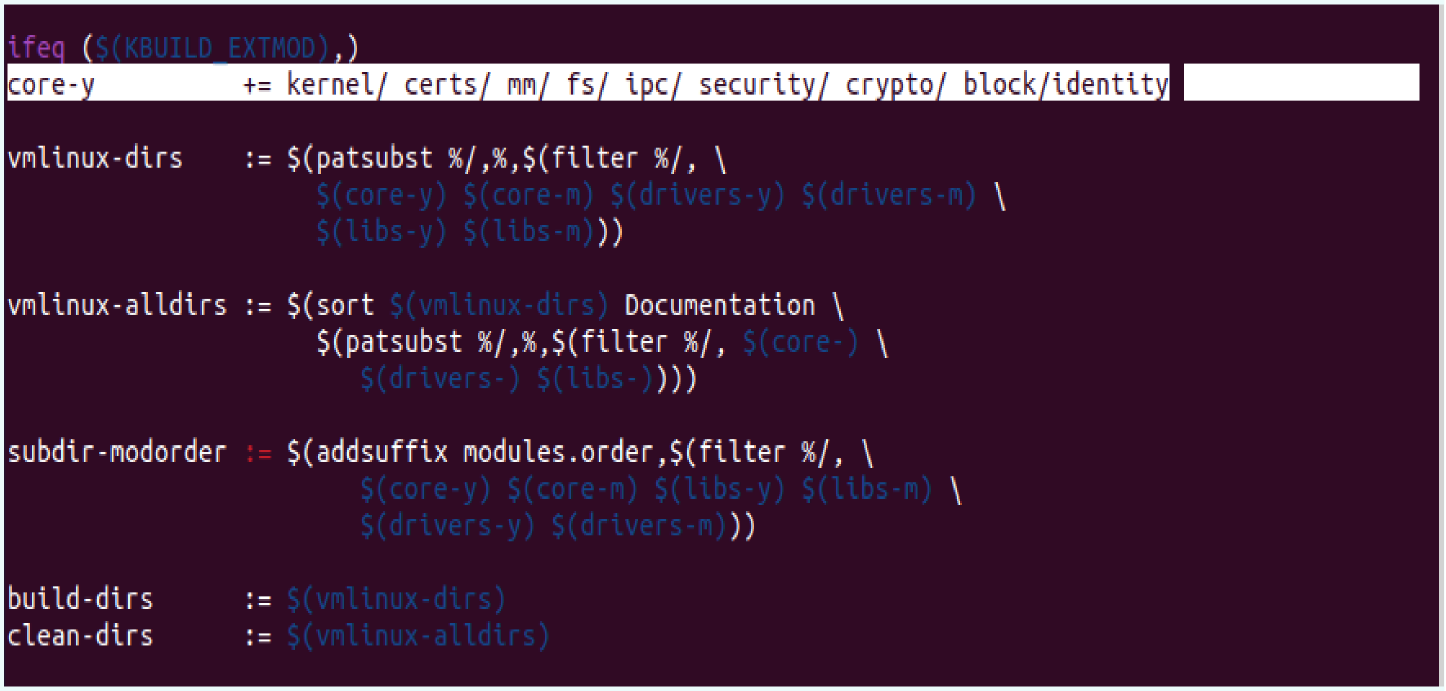
kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/

In the fresh source code of Linux 5.8.1 kernel, it should be in line 1073.

Add the home directory of your system call at the end like the following.

kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ identity/

Save it and exit the editor.



```
ifeq ($(KBUILD_EXTMOD),)
core-y          += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/identity

vmlinux-dirs    := $(patsubst %/,%,$(filter %/, \
                     $(core-y) $(core-m) $(drivers-y) $(drivers-m) \
                     $(libs-y) $(libs-m)))

vmlinux-alldirs := $(sort $(vmlinux-dirs) Documentation \
                     $(patsubst %/,%,$(filter %/, $(core-) \
                        $(drivers-) $(libs-))))

subdir-modorder := $(addsuffix modules.order,$(filter %/, \
                        $(core-y) $(core-m) $(libs-y) $(libs-m) \
                        $(drivers-y) $(drivers-m)))

build-dirs      := $(vmlinux-dirs)
clean-dirs      := $(vmlinux-alldirs)
```

2.7 - Add a corresponding function prototype for your system call to the header file of system calls.

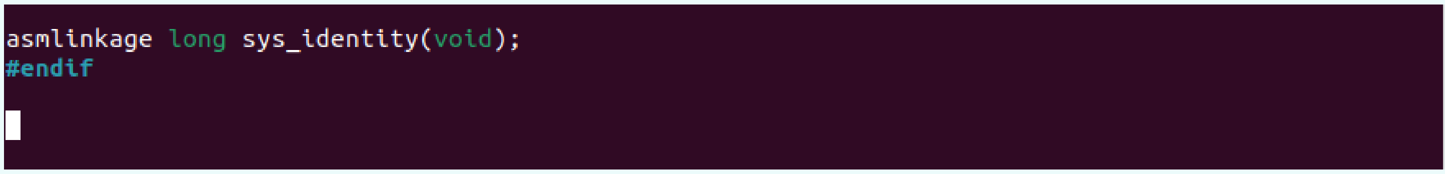
Open the header file with the following command.

```
nano include/linux/syscalls.h
```

Navigate to the bottom of it and write the following code just above #endif.

```
asmlinkage long sys_identity(void);
```

Save it and exit the editor.



```
asmlinkage long sys_identity(void);
#endif
```

2.8 - Add your system call to the kernel's system call table.

Open the table with the following command.

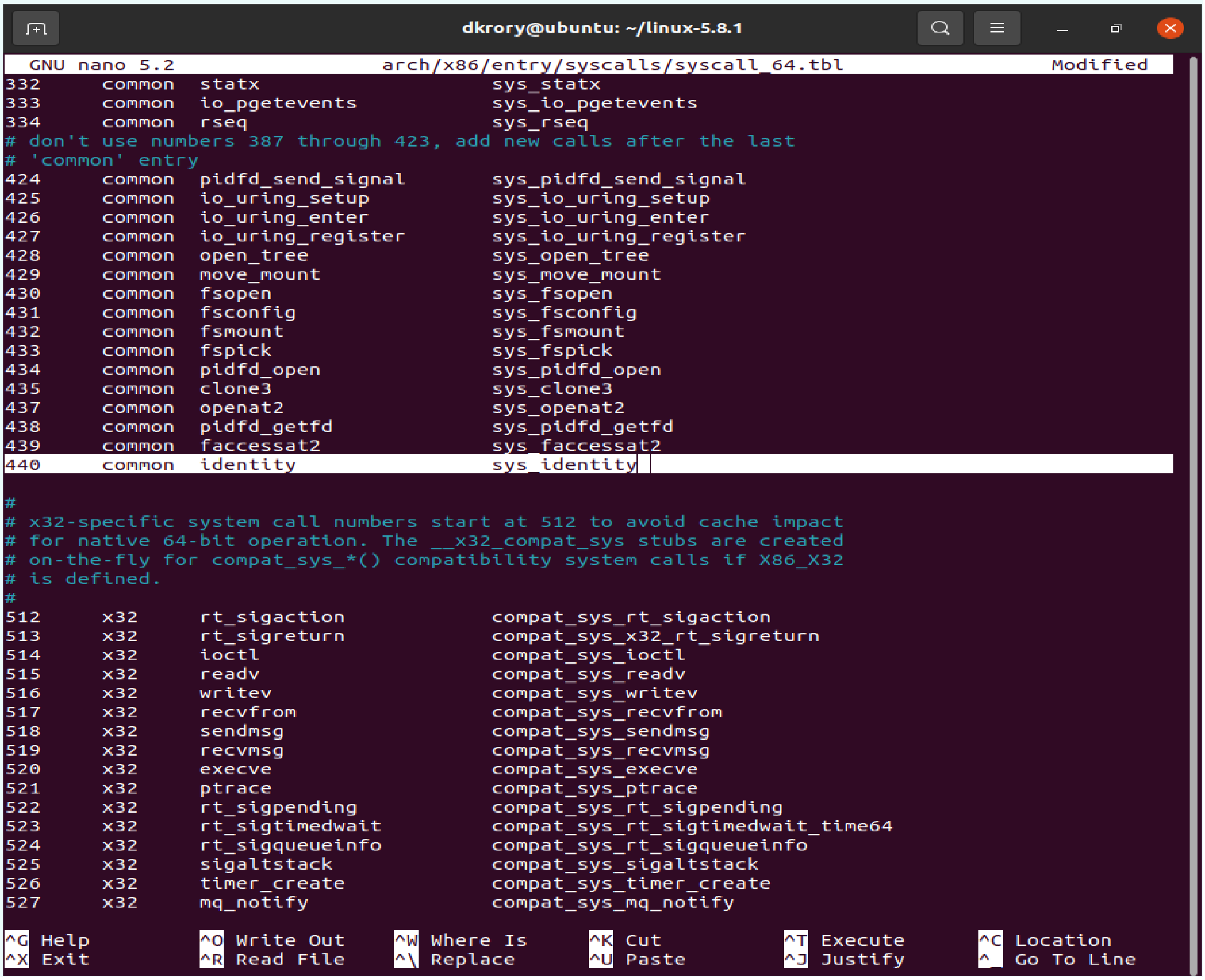
nano arch/x86/entry/syscalls/syscall_64.tbl

Navigate to the bottom of it. You will find a series of x32 system calls. Scroll to the section above it. This is the section of your interest. Add the following code at the end of this section respecting the chronology of the row as well as the format of the column. Use Tab for space.

```
440		common	identity				sys_identity
```

In the fresh source code of Linux 5.8.1 kernel, the number for your system call should be 440.

Save it and exit the editor.



```
dkrory@ubuntu: ~/linux-5.8.1
  GNU nano 5.2                 arch/x86/entry/syscalls/syscall_64.tbl                 Modified
332     common  statx                   sys_statx
333     common  io_pgetevents           sys_io_pgetevents
334     common  rseq                    sys_rseq
# don't use numbers 387 through 423, add new calls after the last
# 'common' entry
424     common  pidfd_send_signal       sys_pidfd_send_signal
425     common  io_uring_setup          sys_io_uring_setup
426     common  io_uring_enter          sys_io_uring_enter
427     common  io_uring_register       sys_io_uring_register
428     common  open_tree               sys_open_tree
429     common  move_mount              sys_move_mount
430     common  fsopen                  sys_fsopen
431     common  fsconfig                sys_fsconfig
432     common  fsmount                 sys_fsmount
433     common  fspick                  sys_fspick
434     common  pidfd_open              sys_pidfd_open
435     common  clone3                  sys_clone3
437     common  openat2                 sys_openat2
438     common  pidfd_getfd             sys_pidfd_getfd
439     common  faccessat2              sys_faccessat2
440     common  identity                sys_identity

#
# x32-specific system call numbers start at 512 to avoid cache impact
# for native 64-bit operation. The __x32_compat_sys stubs are created
# on-the-fly for compat_sys_*() compatibility system calls if X86_X32
# is defined.
#
512     x32     rt_sigaction            compat_sys_rt_sigaction
513     x32     rt_sigreturn            compat_sys_x32_rt_sigreturn
514     x32     ioctl                   compat_sys_ioctl
515     x32     readv                   compat_sys_readv
516     x32     writev                  compat_sys_writev
517     x32     recvfrom                compat_sys_recvfrom
518     x32     sendmsg                 compat_sys_sendmsg
519     x32     recvmsg                 compat_sys_recvmsg
520     x32     execve                  compat_sys_execve
521     x32     ptrace                  compat_sys_ptrace
522     x32     rt_sigpending           compat_sys_rt_sigpending
523     x32     rt_sigtimedwait         compat_sys_rt_sigtimedwait_time64
524     x32     rt_sigqueueinfo         compat_sys_rt_sigqueueinfo
525     x32     sigaltstack             compat_sys_sigaltstack
526     x32     timer_create            compat_sys_timer_create
527     x32     mq_notify               compat_sys_mq_notify
^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^_ Go To Line
```

Section 3 – Installation

In this section, you will install the new kernel and prepare your operating system to boot into it.

3.1 - Configure the kernel.

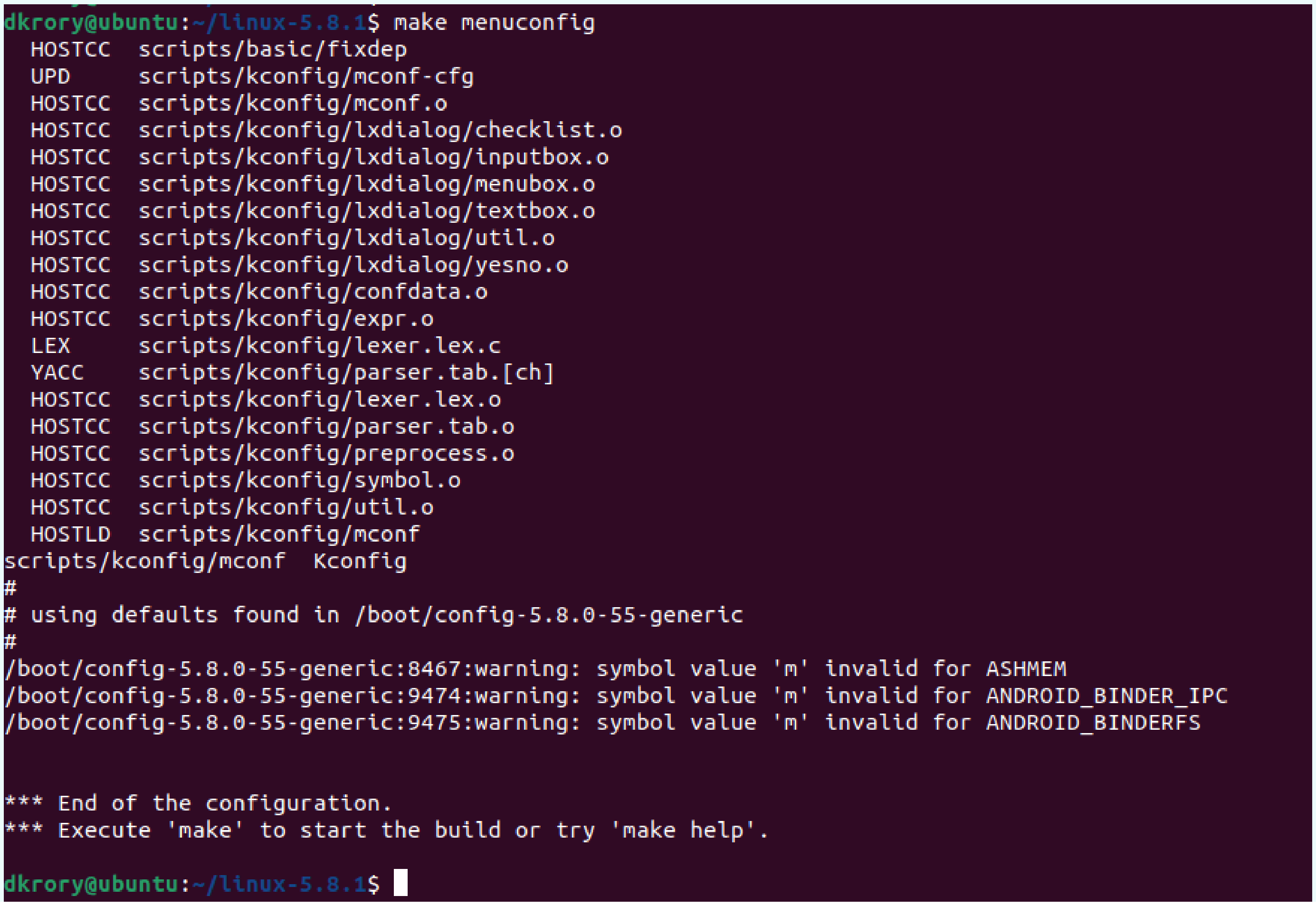
Make sure the window of your terminal is maximized.

Open the configuration window with the following command.

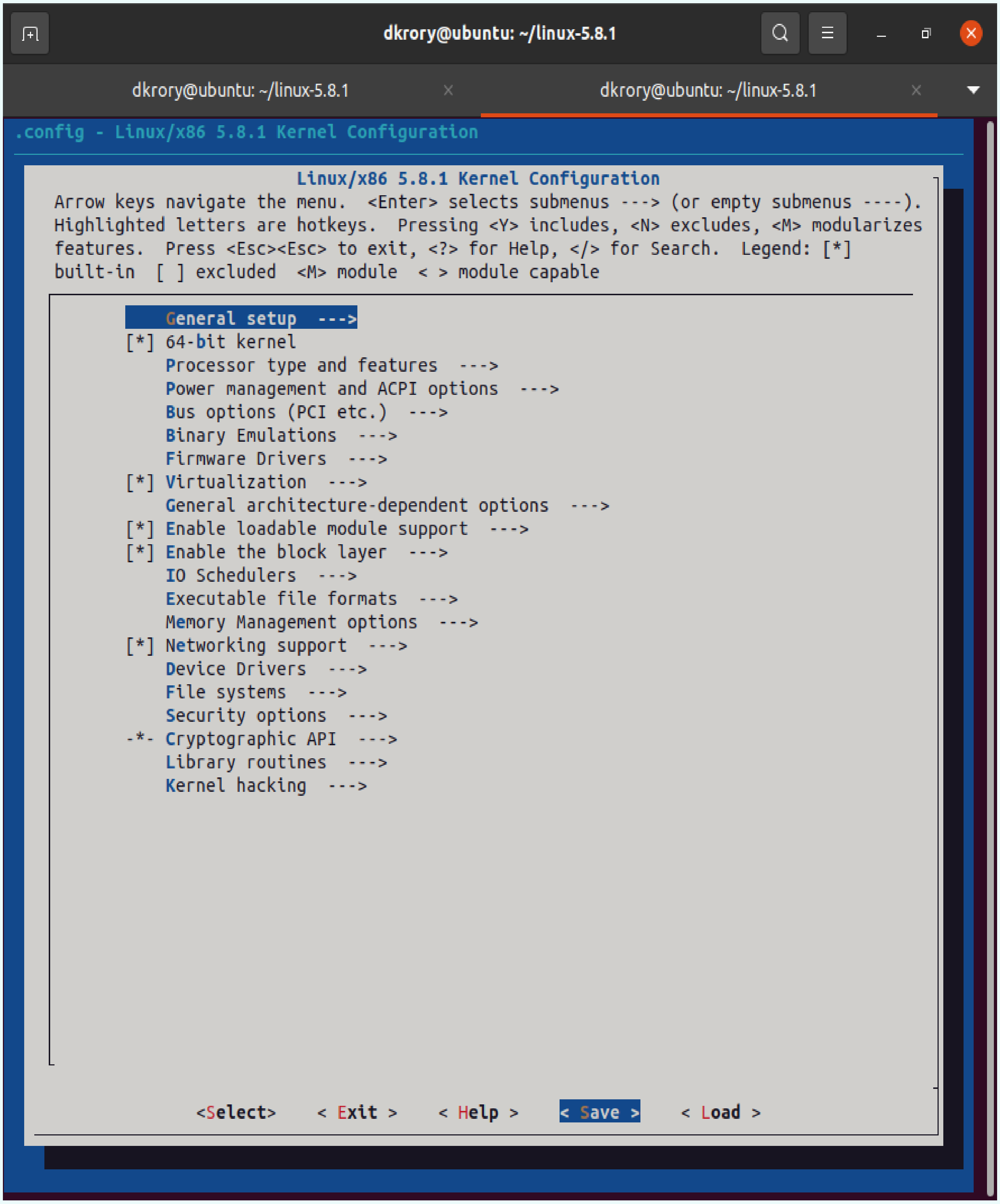
```
make menuconfig
```

Use **Tab** to move between options. Make no changes to keep it in default settings.

Save and exit.



```
dkrory@ubuntu:~/linux-5.8.1$ make menuconfig
  HOSTCC  scripts/basic/fixdep
  UPD     scripts/kconfig/mconf-cfg
  HOSTCC  scripts/kconfig/mconf.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.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/parser.tab.o
  HOSTCC  scripts/kconfig/preprocess.o
  HOSTCC  scripts/kconfig/symbol.o
  HOSTCC  scripts/kconfig/util.o
  HOSTLD  scripts/kconfig/mconf
scripts/kconfig/mconf  Kconfig
#
# using defaults found in /boot/config-5.8.0-55-generic
#
/boot/config-5.8.0-55-generic:8467:warning: symbol value 'm' invalid for ASHMEM
/boot/config-5.8.0-55-generic:9474:warning: symbol value 'm' invalid for ANDROID_BINDER_IPC
/boot/config-5.8.0-55-generic:9475:warning: symbol value 'm' invalid for ANDROID_BINDERFS


*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.

dkrory@ubuntu:~/linux-5.8.1$ 
```


dkrory@ubuntu: ~/linux-5.8.1
dkrory@ubuntu: ~/linux-5.8.1
dkrory@ubuntu: ~/linux-5.8.1

```
.config - Linux/x86 5.8.1 Kernel Configuration
                    Linux/x86 5.8.1 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
  features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*]
  built-in  [ ] excluded  <M> module  < > module capable

        General setup  --->
    [*] 64-bit kernel
        Processor type and features  --->
        Power management and ACPI options  --->
        Bus options (PCI etc.)  --->
        Binary Emulations  --->
        Firmware Drivers  --->
    [*] Virtualization  --->
        General architecture-dependent options  --->
    [*] Enable loadable module support  --->
    [*] Enable the block layer  --->
        IO Schedulers  --->
        Executable file formats  --->
        Memory Management options  --->
    [*] Networking support  --->
        Device Drivers  --->
        File systems  --->
        Security options  --->
    -*- Cryptographic API  --->
        Library routines  --->
        Kernel hacking  --->

       <Select>    < Exit >    < Help >    < Save >    < Load >
```

3.2 - **Find out how many logical cores you have.**

```
nproc
```

The following few commands require a long time to be executed. Parallel processing will greatly speed them up. For me, it is 4. Therefore, I will put 4 after -j in the following commands.

3.3 - **Compile the kernel's source code**.

```
make -j4
```

3.4 - Prepare the installer of the kernel.

```
sudo make modules_install -j4
```

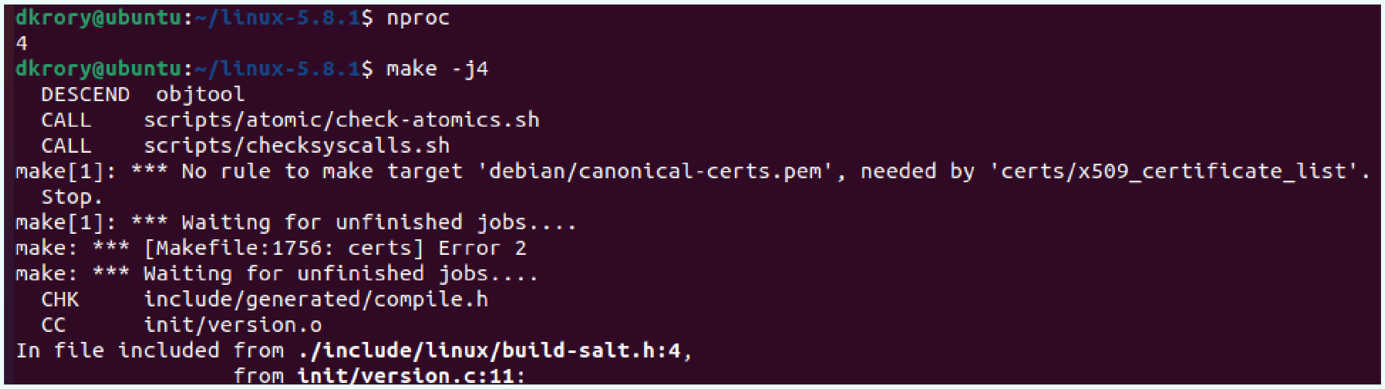
3.5 - **Install the kernel**.

```
sudo make install -j4
```

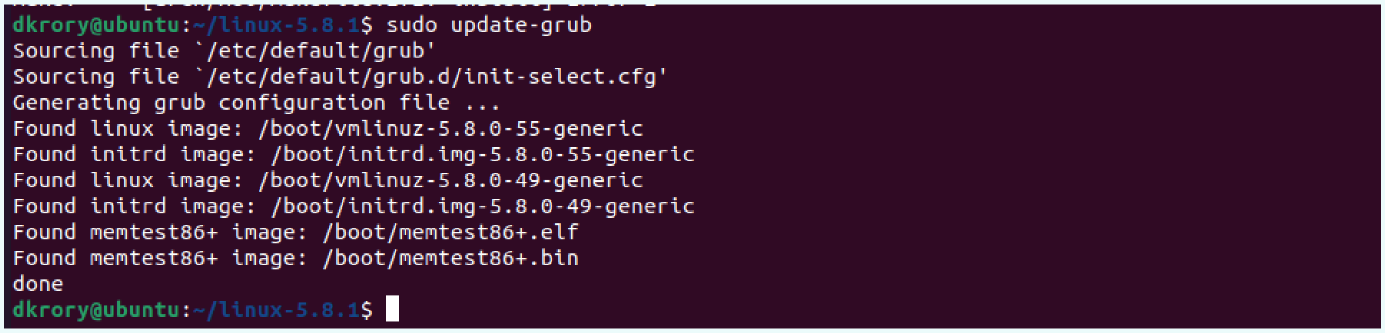
3.6 - **Update the bootloader of the operating system with the new kernel.**

```
sudo update-grub
```

3.7 - **Reboot your computer.**



```
dkrory@ubuntu:~/linux-5.8.1$ nproc
4
dkrory@ubuntu:~/linux-5.8.1$ make -j4
  DESCEND  objtool
  CALL    scripts/atomic/check-atomics.sh
  CALL    scripts/checksyscalls.sh
make[1]: *** No rule to make target 'debian/canonical-certs.pem', needed by 'certs/x509_certificate_list'.
  Stop.
make[1]: *** Waiting for unfinished jobs....
make: *** [Makefile:1756: certs] Error 2
make: *** Waiting for unfinished jobs....
  CHK     include/generated/compile.h
  CC      init/version.o
In file included from ./include/linux/build-salt.h:4,
                 from init/version.c:11:
```



```
dkrory@ubuntu:~/linux-5.8.1$ sudo update-grub
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.8.0-55-generic
Found initrd image: /boot/initrd.img-5.8.0-55-generic
Found linux image: /boot/vmlinuz-5.8.0-49-generic
Found initrd image: /boot/initrd.img-5.8.0-49-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
dkrory@ubuntu:~/linux-5.8.1$
```

Section 4 - Result

In this section, you will write a C program to check whether your system call works or not. After that, you will see your system call in action.

4.1 - Check the version of your current kernel.

```
uname -r
```

4.2 - Change your working directory to your home directory.

```
cd ~
```

4.3 - **Create a C file to generate a report of the success or failure of your system call**.

Create the C file with the following command.

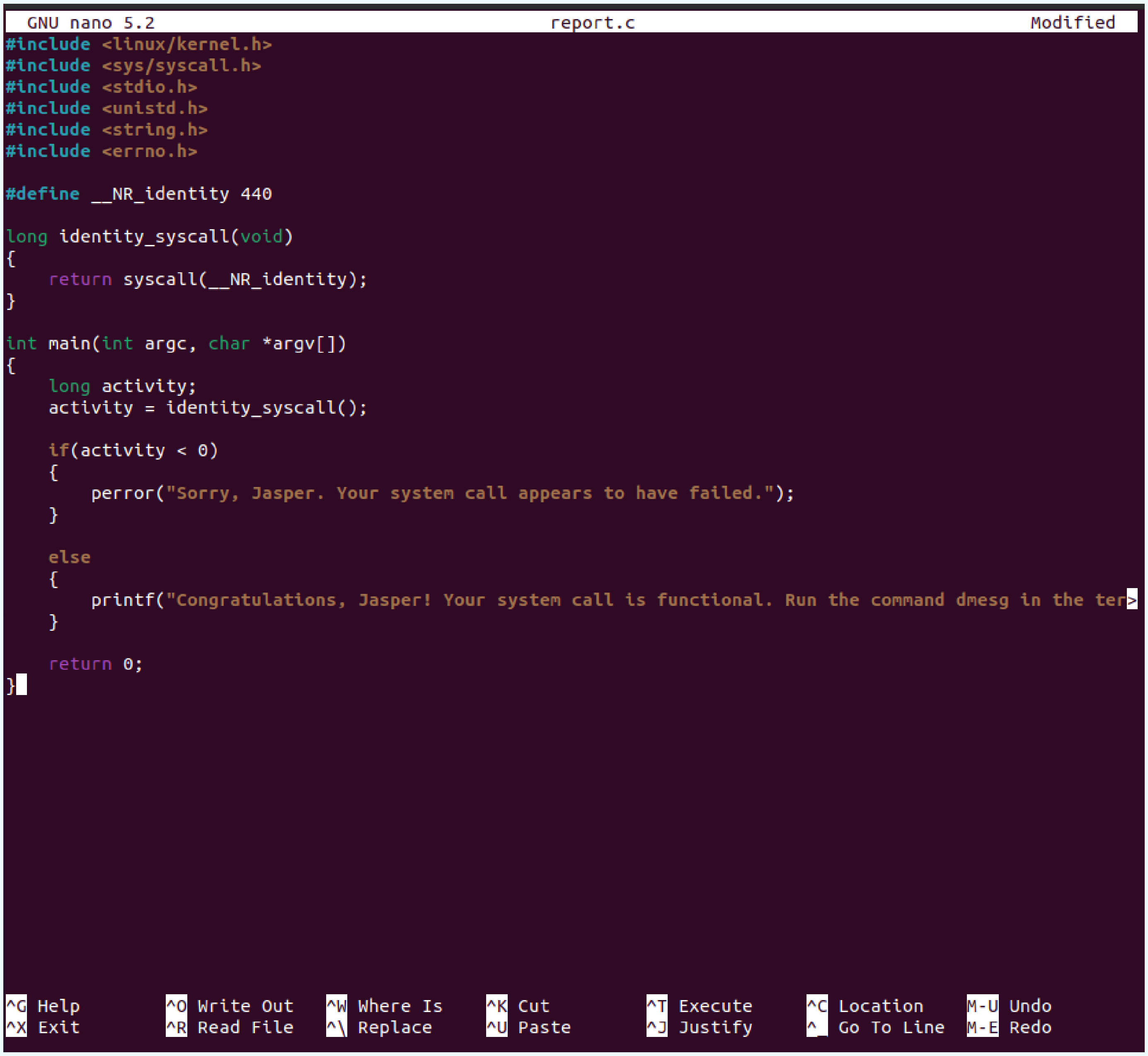
```
nano report.c
```

Write the following code in it.

```
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>

#define __NR_identity 440

long identity_syscall(void)
{
   return syscall(__NR_identity);
}

int main(int argc, char *argv[])
{
   long activity;
   activity = identity_syscall();

   if(activity < 0)
   {
      perror("Sorry, Jasper. Your system call appears to have failed.");
   }

   else
   {
      printf("Congratulations, Jasper! Your system call is functional. Run the command dmesg in the terminal
and find out!\n");
   }

   return 0;
}
```

You can customize the messages for failure and success anyhow you like.

Save it and exit the editor.



```
  GNU nano 5.2                        report.c                        Modified
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>

#define __NR_identity 440

long identity_syscall(void)
{
    return syscall(__NR_identity);
}

int main(int argc, char *argv[])
{
    long activity;
    activity = identity_syscall();

    if(activity < 0)
    {
        perror("Sorry, Jasper. Your system call appears to have failed.");
    }

    else
    {
        printf("Congratulations, Jasper! Your system call is functional. Run the command dmesg in the ter>
    }

    return 0;
}
```

4.4 - Compile the C file you just created.

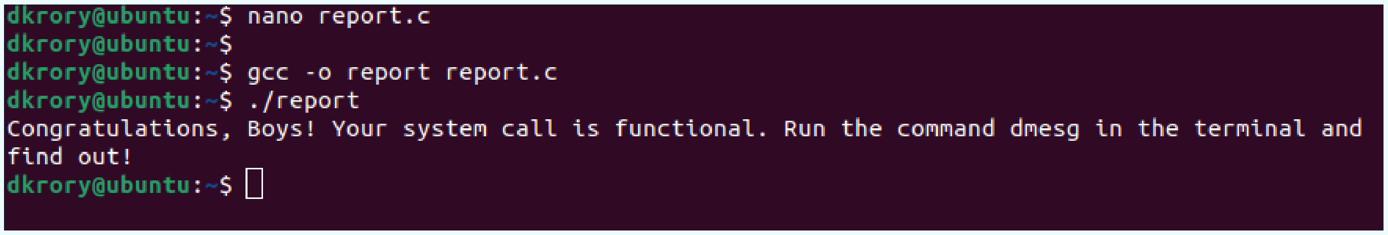
```
gcc -o report report.c
```

4.5 - Run the C file you just compiled.

```
./report
```

If it displays the following, everything is working as intended.

Congratulations, Jasper! Your system call is functional. Run the command dmesg in the terminal and find out!



```
dkrory@ubuntu:~$ nano report.c
dkrory@ubuntu:~$
dkrory@ubuntu:~$ gcc -o report report.c
dkrory@ubuntu:~$ ./report
Congratulations, Boys! Your system call is functional. Run the command dmesg in the terminal and
find out!
dkrory@ubuntu:~$ 
```

References

- https://dev.to/jasper/adding-a-system-call-to-the-linux-kernel-5-8-1-in-ubuntu-20-04-lts-2ga8
- https://medium.com/anubhav-shrimal/adding-a-hello-world-system-call-to-linux-kernel-dad32875872
- https://www.kernel.org/doc/html/latest/process/adding-syscalls.html