Operating System Project

How to add your system call the Linux OS kernel.

Team members:

1- Salma Mostafa Mostafa Hussien
2- Manar Atef Abd-Altawab
3- Shaza Mohamad Abd-Alaleem

The settings of the virtual machine:

- Number of cores: 2
- The Capacity of memory: 2G
- The version of the Kernel: 5.8.1

The steps of how to add a system call:

1- Make the operating system update by this command:
```
sudo apt update && sudo apt upgrade –y
```
2- Download and install all the packages to compile kernels by this command:
```
sudo apt install build-essential libncurses-dev libssl-dev libelf-dev bison flex –y
```
3- Clean up the installed packages by:
```
sudo apt clean && sudo apt autoremove –y
```
4- Download the source code of the latest stable version of the Linux kernel (which is 5.8.1) to your home folder by this command:
```
wget -P ~/ https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.8.1.tar.xz
```
5- Unpack the tarball which downloaded to the home folder by:
```
tar -xvf ~/linux-5.8.1.tar.xz -C ~/
```
6- Reboot the computer.

7- Change the working directory to the root directory of the recently unpacked source code by:
```
cd ~/linux-5.8.1/
```
8- Create the home directory of the system call by running this command:
```
mkdir osubntu
```
9- Create a C file for the system call by:
```
nano osubntu/osubntu.c
```

- Write the following code in it:

```
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE0(osubntu)

{
   printk("osubntu.\n");
   return 0;
}
```

- And save it by (ctrl+o) and exit the text editor by (ctrl + x).

10- Create a Makefile for the system call by running this command:

```
nano osubntu/Makefile
```

- Write the following code in it:

```
obj-y := osubntu.o
```

- And save it by (ctrl+o) and exit the text editor by (ctrl + x).

11- Add the home directory of the system call to the main Makefile of the kernel. Open the Makefile by the following command:

```
nano Makefile
```

- Search for core-y by (ctrl+ w) . In the second result, you will see a series of directories.

Add the home directory of the system call at the end like the following:

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

- And save it by (ctrl+o) and exit the text editor by (ctrl + x).

12- Add a corresponding function prototype for your system call to the header file of system calls by this command:

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

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

```
asmlinkage long sys_osubntu(void);
```

- And save it by (ctrl+o) and exit the text editor by (ctrl + x).

13- Add the 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:

```
440    common  identity              sys_identity
```

- And save it by (ctrl+o) and exit the text editor by (ctrl + x).

14- Configure the kernel by this command:

```
make menuconfig
```

Note: Make sure the window of your terminal is maximized.

- Use Tab to move between options. Make no changes to keep it in default settings.
- Save and exit.

15- Find out how many logical cores you have by:

```
Nproc
```

Note: Our logical cores are:2

16- Compile the kernel's source code by running source code:

```
make –j2
```

Note: If there are errors run these commands in order :

```
make clean
make mrproper
make menuconfig
make localmodconfig
```

17- Prepare the installer of the kernel by running this command:

```
sudo make modules_install –j2
```

18- Install the kernel by:

```
sudo make install –j2
```

19- Update the bootloader of the operating system with the new kernel by:

```
sudo update-grub
```

20- Reboot your computer.

21- Change the working directory to your home directory by:

```
cd ~
```

22- Create a C file to generate a report of the success or failure of the system call by running this 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_osubntu 440
long osubntu_syscall(void)
{
    return syscall(__NR_osubntu);
}
int main(int argc, char *argv[])
{
    long activity;
    activity = osubntu_syscall();
    if(activity < 0)
    {
        perror("Sorry, Your system failed.");
    }
    else
    {
        printf("Congrats, Your system call added!\n");
    }
    return 0;
}
```

- And save it by (ctrl+o) and exit the text editor by (ctrl + x).

23- Compile the C file you just created by this command:

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

24- Run the C file you just compiled by this command:

```
./report
```

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

```
Congrats, Your system call added!
```

25- Check the last line of the dmesg output by:

```
dmesg
```

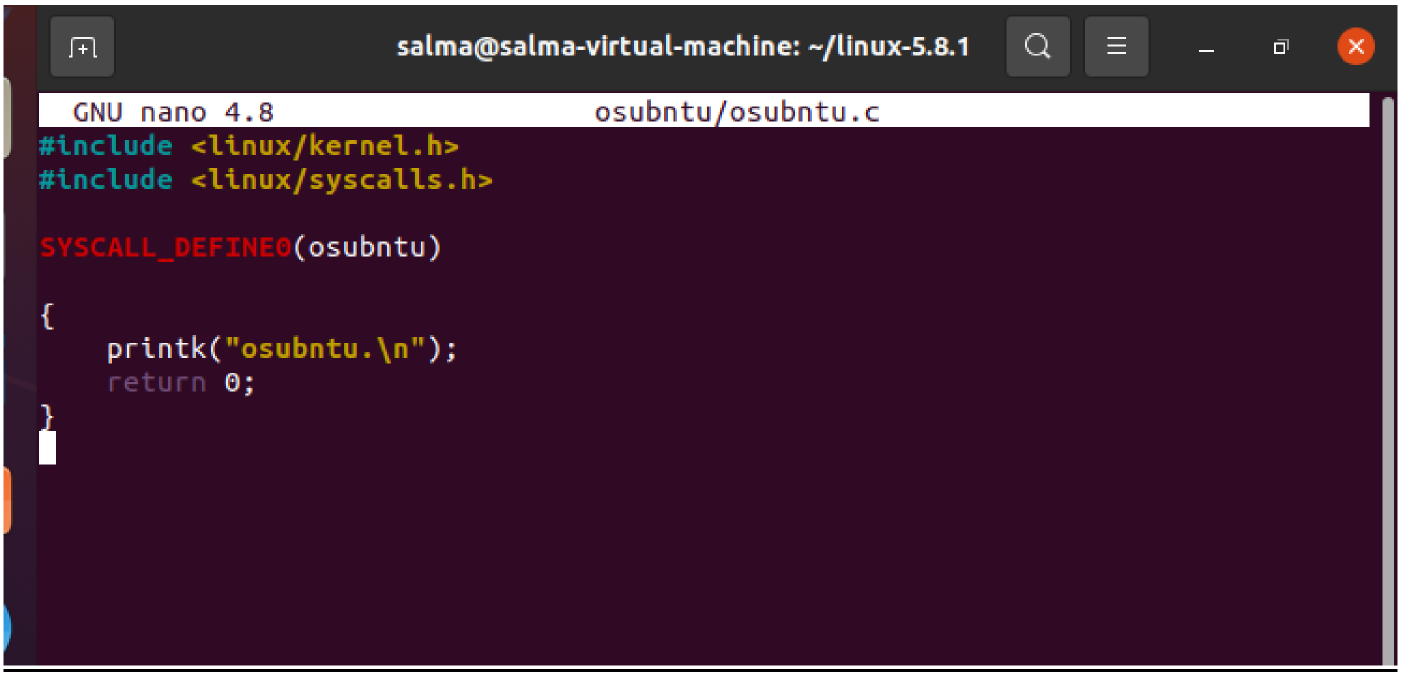
- At the bottom, you should now see the following:

```
Osubntu
```

References: https://dev.to/jasper/adding-a-system-call-to-the-linux-kernel-5-8-1-in-ubuntu-20-04-lts-2ga8

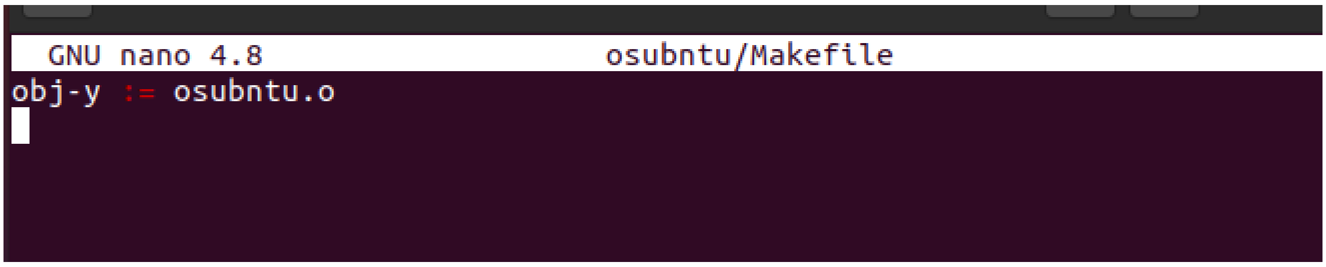
Screenshots From The Project:

1- Create a C file for the system call and write the following code in it:



```
  GNU nano 4.8                    osubntu/osubntu.c
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE0(osubntu)

{
    printk("osubntu.\n");
    return 0;
}
```

2- Create a Makefile for the system call and write the following code in it:



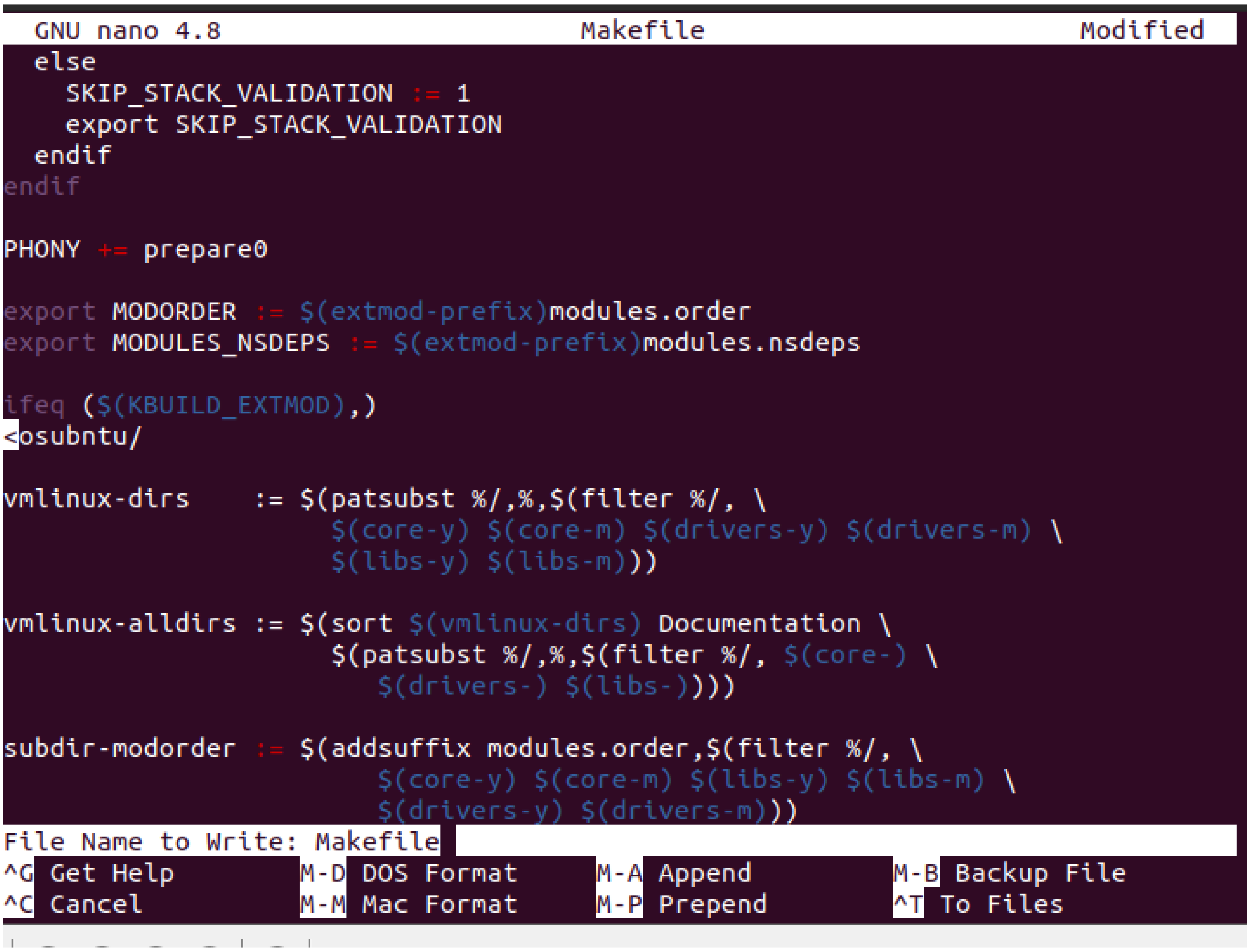
```
  GNU nano 4.8                    osubntu/Makefile
obj-y := osubntu.o
```

3- Add the home directory of the system call to the main Makefile of the kernel. Open the Makefile.

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

- Add the home directory of the system call at the end like the following:

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

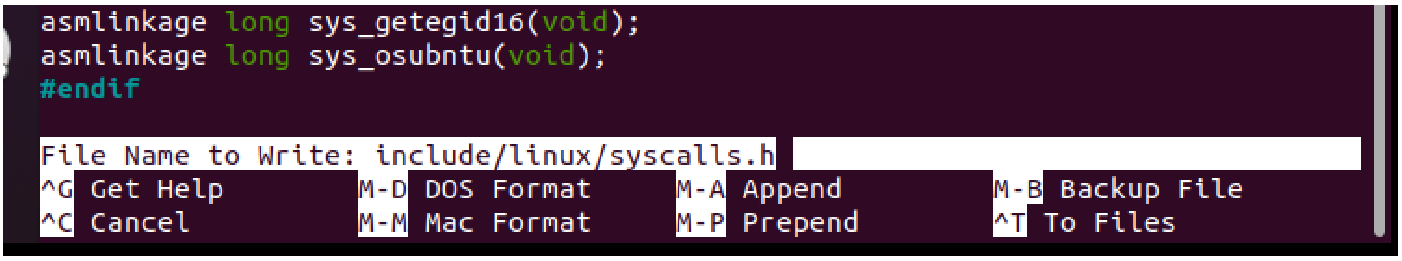


```
  GNU nano 4.8                          Makefile                          Modified
  else
    SKIP_STACK_VALIDATION := 1
    export SKIP_STACK_VALIDATION
  endif
endif

PHONY += prepare0

export MODORDER := $(extmod-prefix)modules.order
export MODULES_NSDEPS := $(extmod-prefix)modules.nsdeps

ifeq ($(KBUILD_EXTMOD),)
<osubntu/

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)))
File Name to Write: Makefile
^G Get Help          M-D DOS Format       M-A Append           M-B Backup File
^C Cancel            M-M Mac Format       M-P Prepend          ^T To Files
```

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

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

asmlinkage long sys_osubntu(void);

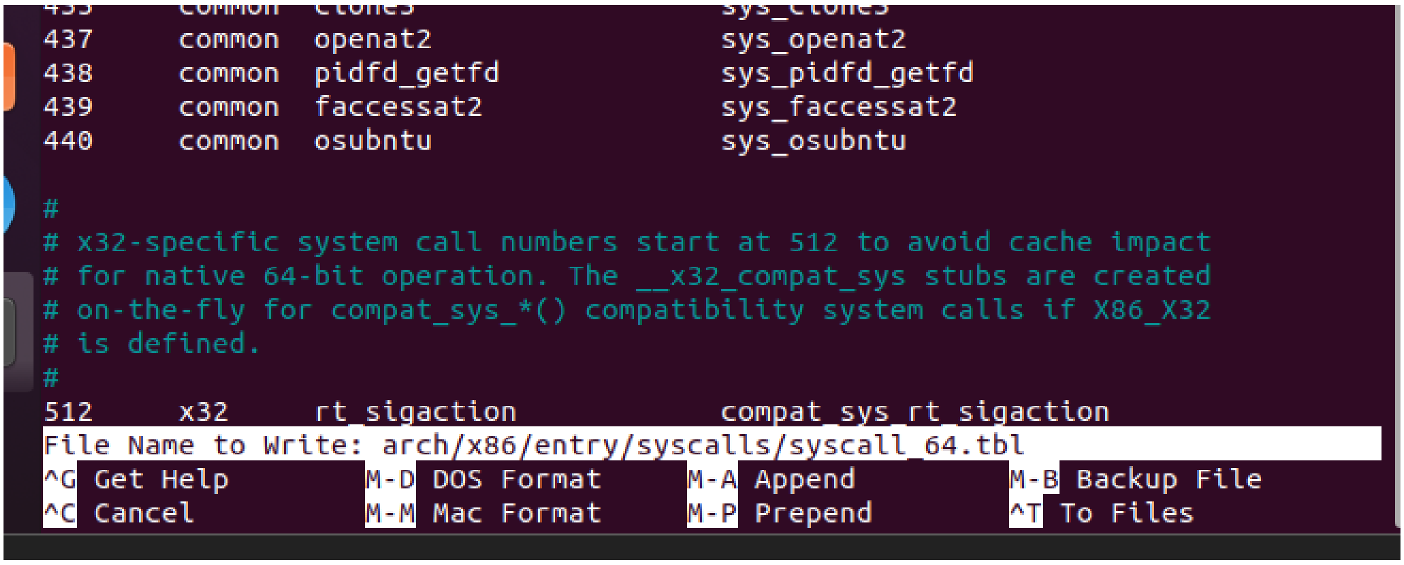


```
asmlinkage long sys_getegid16(void);
asmlinkage long sys_osubntu(void);
#endif

File Name to Write: include/linux/syscalls.h
^G Get Help          M-D DOS Format       M-A Append           M-B Backup File
^C Cancel            M-M Mac Format       M-P Prepend          ^T To Files
```

5- Add the system call to the kernel's system call table.

- 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:

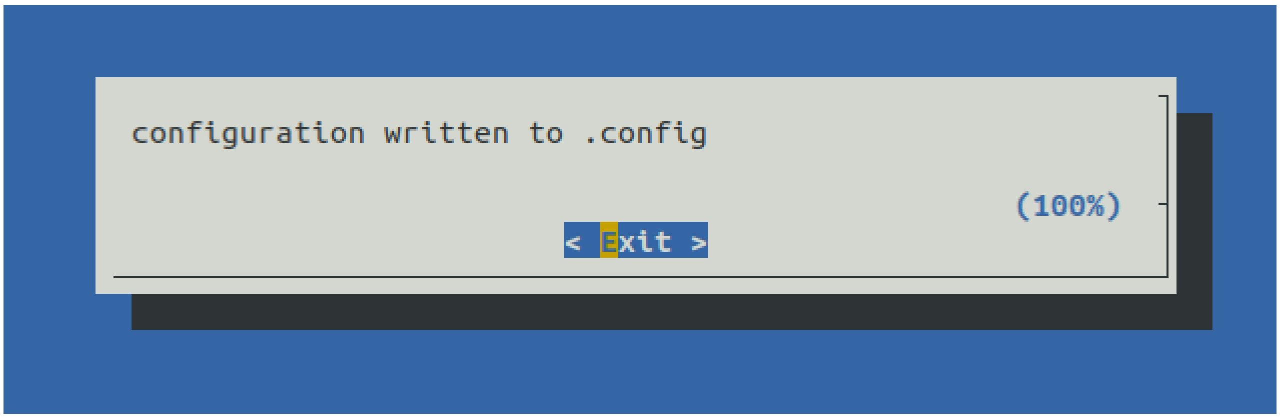
440 common identity sys_identity



```
437     common  openat2                 sys_openat2
438     common  pidfd_getfd             sys_pidfd_getfd
439     common  faccessat2              sys_faccessat2
440     common  osubntu                 sys_osubntu

#
# 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
File Name to Write: arch/x86/entry/syscalls/syscall_64.tbl
^G Get Help          M-D DOS Format       M-A Append           M-B Backup File
^C Cancel            M-M Mac Format       M-P Prepend          ^T To Files
```

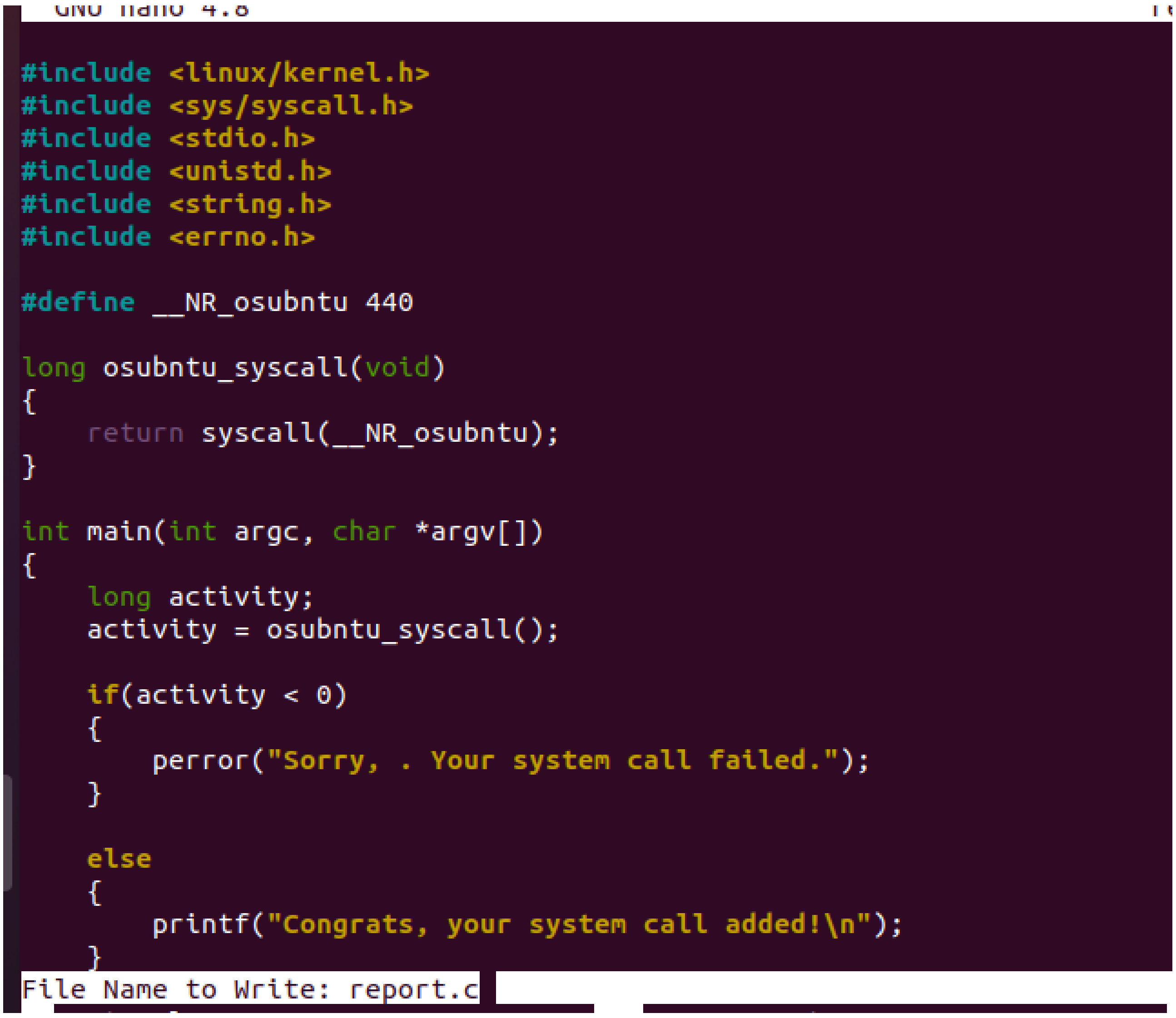
6- Configure the kernel .

- Use Tab to move between options. Make no changes to keep it in default settings.
- Save and exit.



```
configuration written to .config

(100%)
< Exit >
```

7- Create a C file to generate a report of the success or failure of the system call and write this code in it:

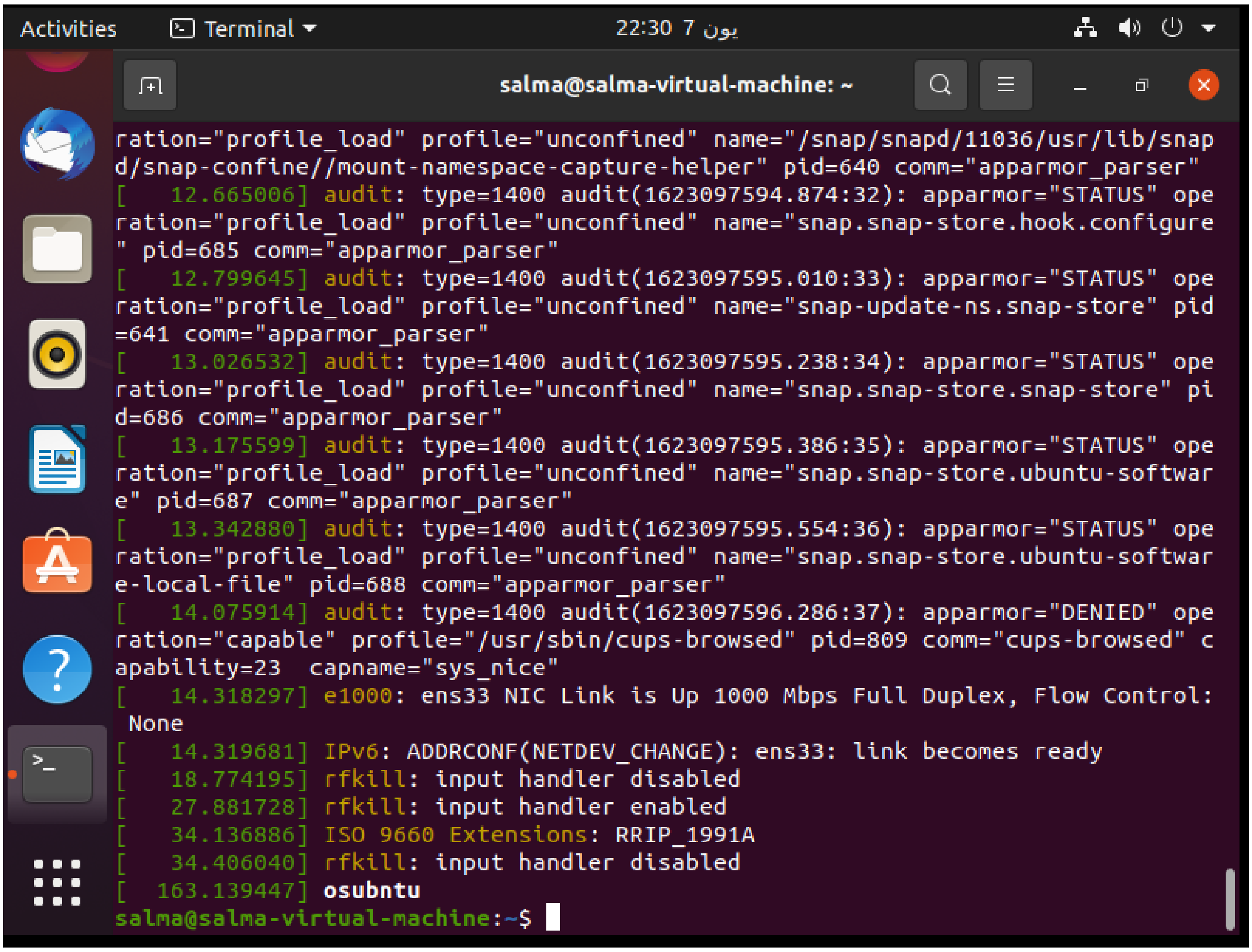


```
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>

#define __NR_osubntu 440

long osubntu_syscall(void)
{
    return syscall(__NR_osubntu);
}

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

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

    else
    {
        printf("Congrats, your system call added!\n");
    }
File Name to Write: report.c
```

8- Check the last line of the dmesg output.

- At the bottom, you should now see the following:

Osubntu



```
ration="profile_load" profile="unconfined" name="/snap/snapd/11036/usr/lib/snap
d/snap-confine//mount-namespace-capture-helper" pid=640 comm="apparmor_parser"
[   12.665006] audit: type=1400 audit(1623097594.874:32): apparmor="STATUS" ope
ration="profile_load" profile="unconfined" name="snap.snap-store.hook.configure
" pid=685 comm="apparmor_parser"
[   12.799645] audit: type=1400 audit(1623097595.010:33): apparmor="STATUS" ope
ration="profile_load" profile="unconfined" name="snap-update-ns.snap-store" pid
=641 comm="apparmor_parser"
[   13.026532] audit: type=1400 audit(1623097595.238:34): apparmor="STATUS" ope
ration="profile_load" profile="unconfined" name="snap.snap-store.snap-store" pi
d=686 comm="apparmor_parser"
[   13.175599] audit: type=1400 audit(1623097595.386:35): apparmor="STATUS" ope
ration="profile_load" profile="unconfined" name="snap.snap-store.ubuntu-softwar
e" pid=687 comm="apparmor_parser"
[   13.342880] audit: type=1400 audit(1623097595.554:36): apparmor="STATUS" ope
ration="profile_load" profile="unconfined" name="snap.snap-store.ubuntu-softwar
e-local-file" pid=688 comm="apparmor_parser"
[   14.075914] audit: type=1400 audit(1623097596.286:37): apparmor="DENIED" ope
ration="capable" profile="/usr/sbin/cups-browsed" pid=809 comm="cups-browsed" c
apability=23  capname="sys_nice"
[   14.318297] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 None
[   14.319681] IPv6: ADDRCONF(NETDEV_CHANGE): ens33: link becomes ready
[   18.774195] rfkill: input handler disabled
[   27.881728] rfkill: input handler enabled
[   34.136886] ISO 9660 Extensions: RRIP_1991A
[   34.406040] rfkill: input handler disabled
[  163.139447] osubntu
salma@salma-virtual-machine:~$ 
```