Name: Muhammad Hamza Aslam (F2017266041**)**

**Add Hello World System Call in Linux Kernel**

Note: Don’t Copy Paste Code From File

In your terminal type the following command:

*wget [https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.17.4.tar.xz](https://www.kernel.org/pub/linux/kernel/v3.x/linux-3.16.tar.xz" \t "https://medium.com/anubhav-shrimal/_blank)*

Download a kernel of higher version (4.17.4), this way the kernel gets automatically updated when you reboot the system after compiling.

wget command : GNU Wget is a free utility for non-interactive download of files from the Web.

## **2. Extract the kernel source code**

*sudo tar -xvf linux-4.17.4.tar.xz -C/usr/src/*

Now, we’ll change the directory to where the files are extracted:

cd /usr/src/linux-4.17.4/

## **3. Define a new system call sys\_hello( )**

* Create a directory named hello/ and change the directory to hello/:

mkdir hello  
cd hello

If you get error of not making directory kindly write

Sudo chmod -R 777 /usr/src/linux-4.17.4

Then write above command it change the mod and give you write to make a directory if error still occur kindly close the terminal and again open terminal

Create a file hello.c using your favourite text editor:

gedit hello.c

write the following code in the editor:

#include <linux/kernel.h>

asmlinkage long sys\_hello(void)

{

printk("Hello world\n");

return 0;

}

printk prints to the kernel’s log file.

* Create a “Makefile” in the hello directory:

gedit Makefile

and add the following line to it:

obj-y := hello.o

This is to ensure that the hello.c file is compiled and included in the kernel source code.

*Note: There is no space in between“obj-y”.*

## **4. Adding hello/ to the kernel’s Makefile:**

Go back to the parent dir i.e. cd ../ and open “Makefile”

gedit Makefile

search for core-y in the document, you’ll find this line as the second instance of your search:

*core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/*

Add ‘hello/’ to the end of this line:

*core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ hello/*

*Note: There is a space between “block/” and “hello/”*

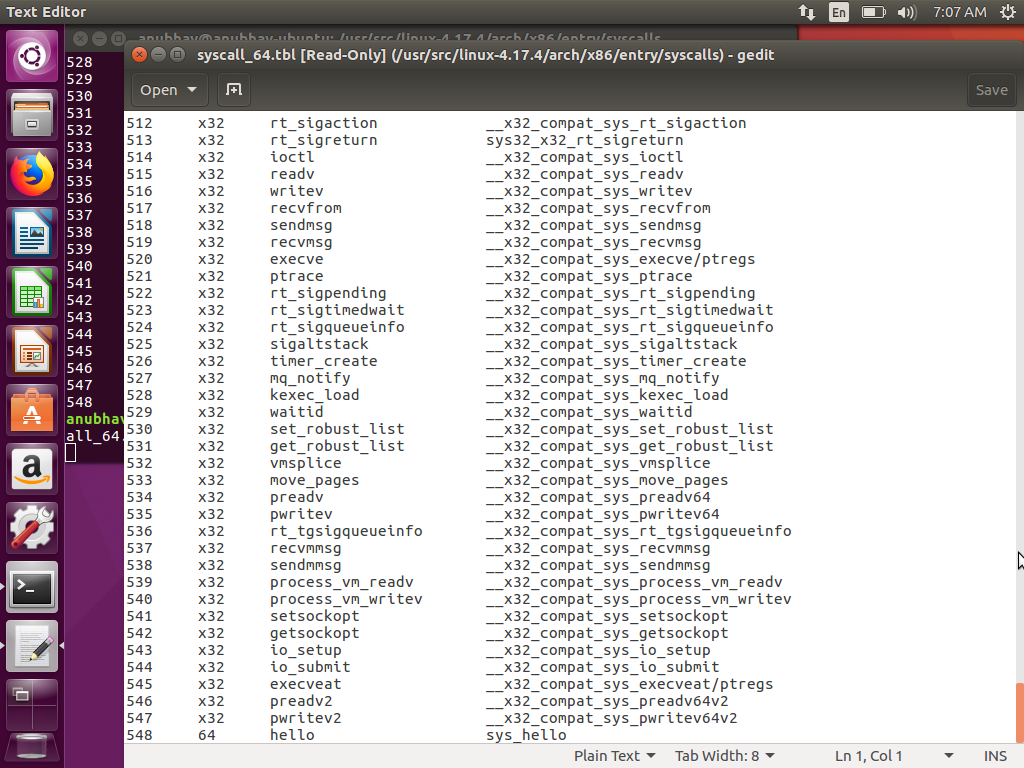
## **5. Add the new system call to the system call table:**

If you are on a 32-bit system you’ll need to change ‘syscall\_32.tbl’. For 64-bit, change ‘syscall\_64.tbl’.

Run the following commands in your terminal from linux-4.17.4/ directory:

cd arch/x86/entry/syscalls/  
gedit syscall\_64.tbl

You’ll get a file like the following in your editor:



Go to the last of the document and add a new line like so:

548 64 hello sys\_hello

****Note:****

* Here I’ve written 548 because in the previous line the number entry was 547. Remember this number it will be used in the later steps. Also check you have 64 0r 32 bit you find in 310 or 312 number

Save and exit.

## **6. Add new system call to the system call header file:**

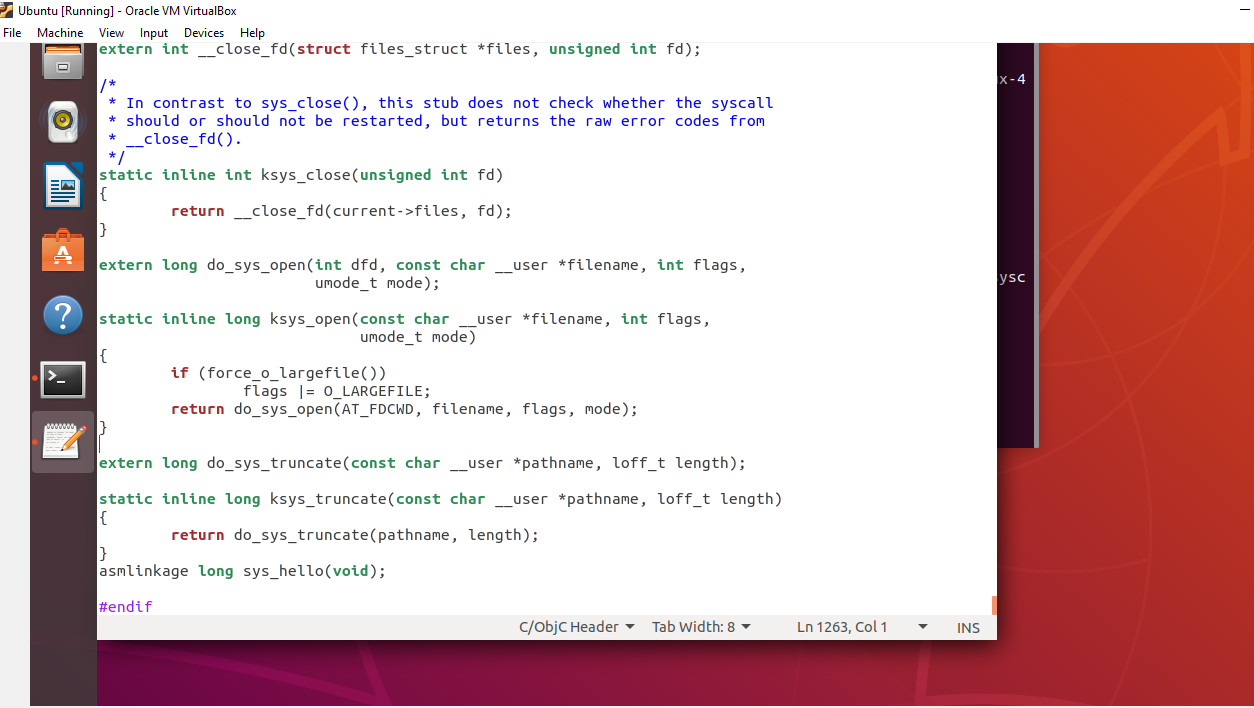
Go to the linux-4.17.4/ directory and type the following commands:

cd include/linux/  
gedit syscalls.h

Add the following line to the end of the document before the #endif statement:

asmlinkage long sys\_hello(void);

After this your file will look like so:



Save and exit.

This defines the prototype of the function of our system call. “asmlinkage” is a key word used to indicate that all parameters of the function would be available on the stack.

## **7. Compile the kernel:**

Before starting to compile you need to install a few packages. Type the following commands in your terminal:

sudo apt-get install gcc  
sudo apt-get install libncurses5-dev  
sudo apt-get install bison  
sudo apt-get install flex  
sudo apt-get install libssl-dev  
sudo apt-get install libelf-dev

to configure your kernel use the following command in your linux-4.17.4/ directory:

sudo make menuconfig

Once the above command is used to configure the Linux kernel, you will get a pop up window with the list of menus and you can select the items for the new configuration. If your unfamiliar with the configuration just check for the file systems menu and check whether “*ext4*” is chosen or not, if not select it and save the configuration.

Now to compile the kernel you can use the make command:

sudo make

If you get error type

sudo apt-get install build-essential

This might take several hours depending on your system. It took me 2–3 hours to get this compiled.

In my case it take 7 hours because I select many file in menuconfig file so don’t make mistake only select which mentioned above

## **8. Install / update Kernel:**

Run the following command in your terminal:

sudo make modules\_install install

It will create some files under /boot/ directory and it will automatically make a entry in your grub.cfg. To check whether it made correct entry, check the files under /boot/ directory . If you have followed the steps without any error you will find the following files in it in addition to others.

1. System.map-4.17.4
2. vmlinuz-4.17.4
3. initrd.img-4.17.4
4. config-4.17.4

Now to update the kernel in your system reboot the system . You can use the following command:

shutdown -r now

After rebooting you can verify the kernel version using the following command:

*uname -r*

It will display the kernel version like so:

4.17.4

If it now show 4.17.4 then type

Reboot

After when you system is restart press SHIFT button from your keyboard the advanced options then select the kernel 4.17.4

Note: you may see many option but select only 4.17.4

## **9. Test system call:**

Go to your home(~) directory using the following commands and create a userspace.c file.

cd ~  
gedit userspace.c

Write the following code in this file:

#include <stdio.h>

#include <linux/kernel.h>

#include <sys/syscall.h>

#include <unistd.h>

int main()

{

long int amma = syscall(548);

printf("System call sys\_hello returned %ld\n", amma);

return 0;

}

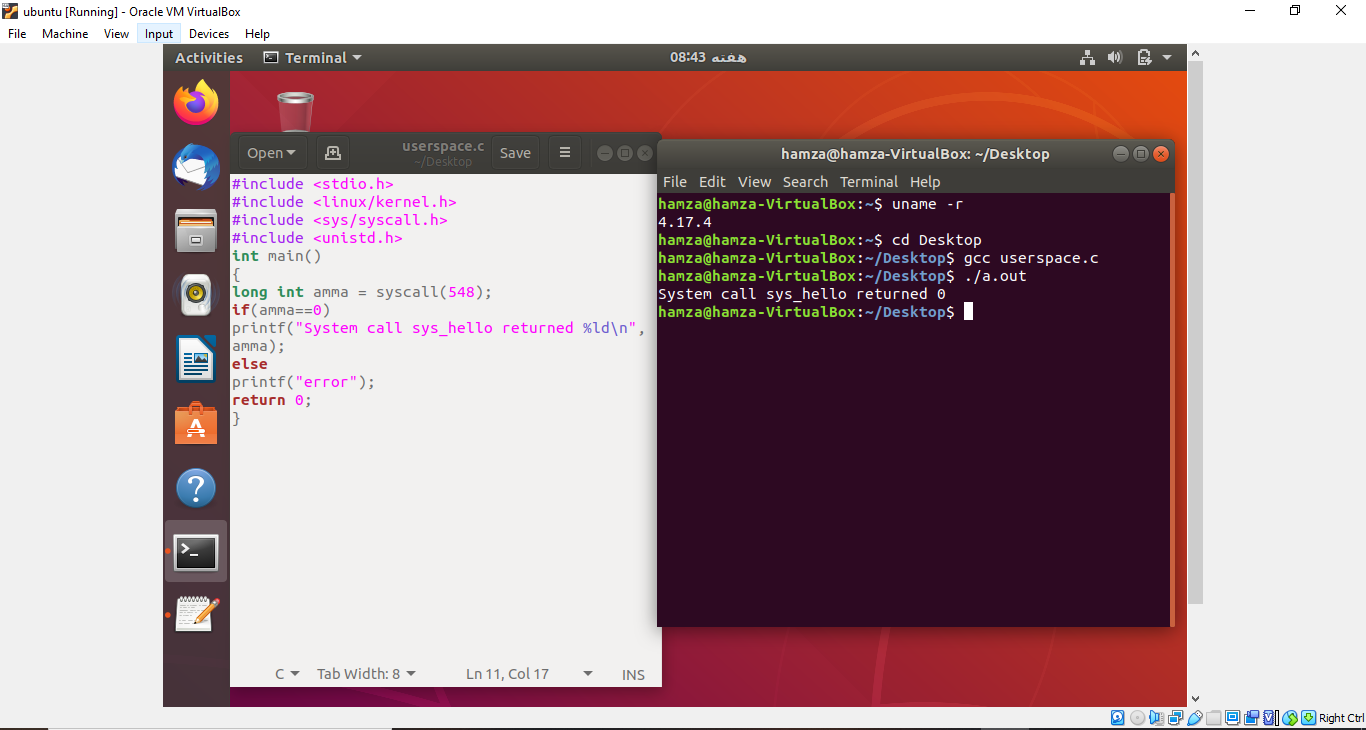
***Note: Remember I told you to keep in mind the number of system call that we wrote in syscalls\_64.tbl? In my case the number was 548. Write that same number in your userspace.c file as an argument in syscall() function.***

Now, compile and run the program:

gcc userspace.c  
./a.out

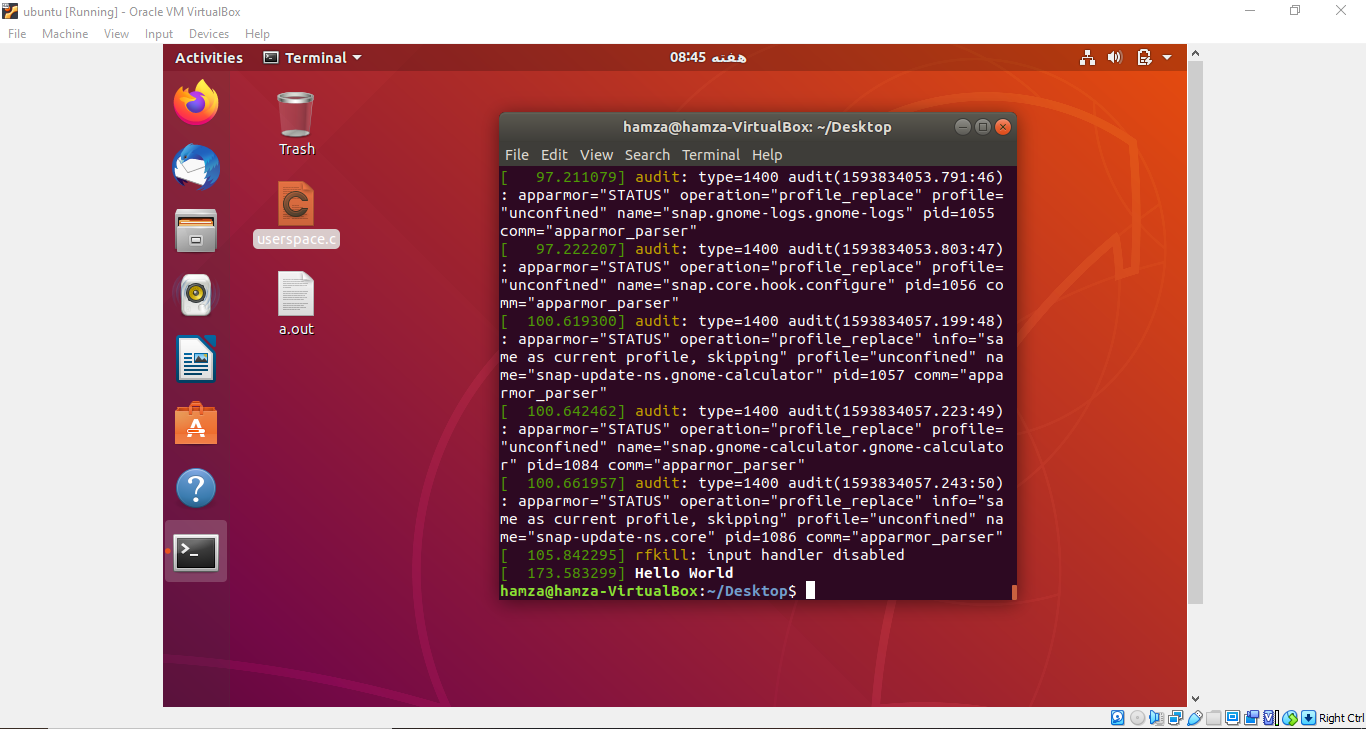
If all the steps are done correctly you’ll get an output like below:

System call sys\_hello returned 0



Now, to check the message of your kernel run the following command:

Dmesg



This will display *Hello world* at the end of the kernel’s message.

Note: if your first try fail don’t be sad try again and again I successful in adding system call in 5 try.