Assignment No.7

**AIM:** Linux Kernel configuration, compilation, and rebooting from the newly compiled kernel. add your own system call in the kernel.

**STEPS:**

1. **Download the kernel source:**

*wget* [*https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.17.4.tar.xz*](https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.17.4.tar.xz)

# 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/

# Defining a new system call sys\_hello( )

Creating a directory named hello/ and change the directory to hello/: mkdir hello

cd hello

## Creating a file hello.c using text editor:

gedit hello.c

**writing the following code in the editor:** #include <linux/kernel.h> asmlinkage long sys\_hello(void)

{

printk("Hello I am from VIIT\n"); return 0;

}

## 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”.*

1. **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/”.*

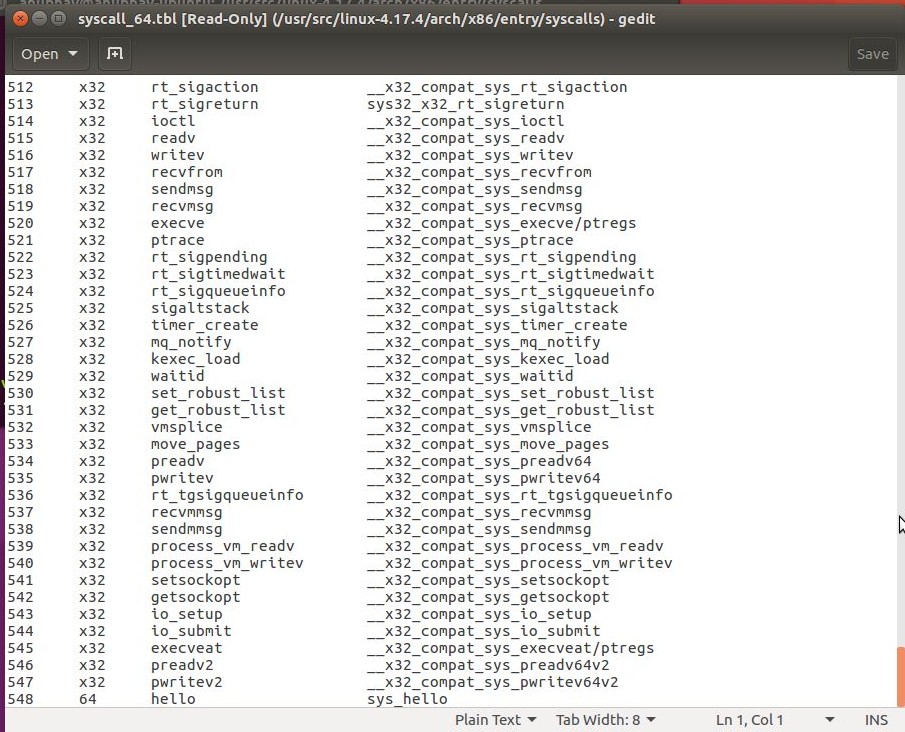
This is to tell the compiler that the source files of our new system call (sys\_hello()) are present in the hello directory.

# Add the new system call to the system call table:

Run the following commands in terminal from linux-4.17.4/ directory: cd arch/x86/entry/syscalls/

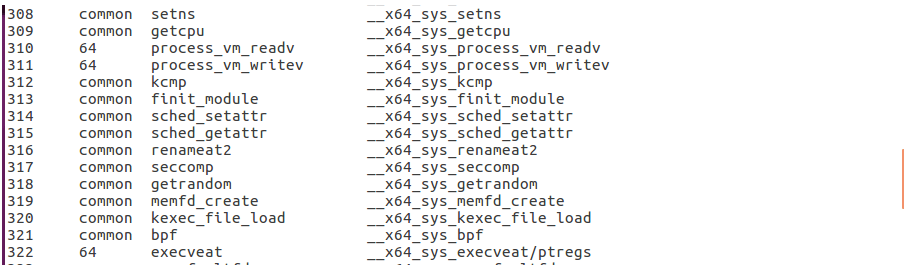
gedit syscall\_64.tbl

We’ll get a file like the following in your edit or:



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

548 64 hello sys\_hello



64 written at 310, 311 and 322 line numbers

This will tell us whether to write i586 or something else. Save and exit.

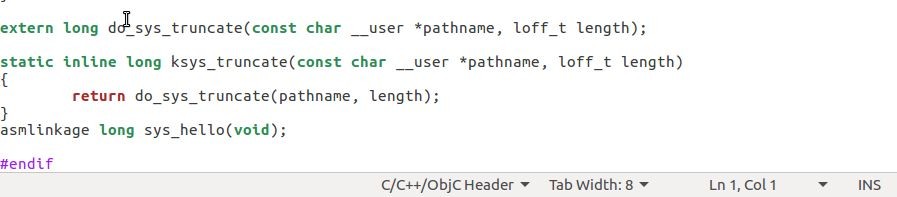
# Add new system call to the system call header file:

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 our 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.

# Compile the kernel:

Before starting to compile we 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 sudo apt-get update

sudo apt-get upgrade

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

sudo make menuconfig

Once the above command is used to configure the Linux kernel, we will get a pop up window with the list of menus and you can select the items for the new configuration. If one is 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

***Tip:*** *The make command can take a lot of time in compiling, to speed up the process you can take advantage of the multiple cores that our systems have these days. Simply type,*

*sudo make -jn*

*where n is the number of cores that you have in your linux system. For example if I have a Quad core(4) processor, I’ll write:*

*sudo make -j4*

*this will speed up my make process 4x times. ;)*

This might take several hours depending on your system.

# 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 an entry in your grub.cfg. To check whether it made the correct entry, check the files under /boot/ directory . If one has 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 . We can use the following command:

shutdown -r now

## After rebooting we can verify the kernel version using the following command:

*uname -r*

## It will display the kernel version like so:

4.17.4

# Test system call:

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

cd ~

gedit userspace.c

## Writing the following code in 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;

}

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

It will display *Hello I am from VIIT* at the end of the kernel’s message.

**OUTPUT:**

