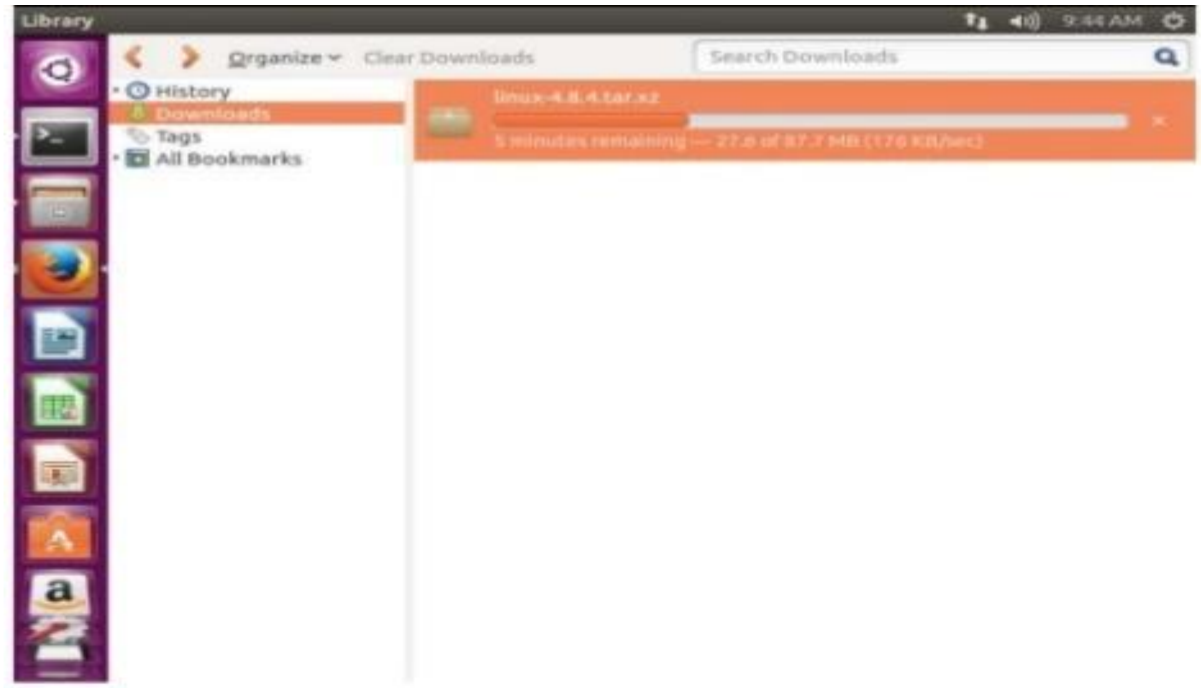
Operating System Project
Report Adding System Call To Kernel

1. . **Step 1**
   As we will have to write sudo in start of approx. every command, So in order to remove this difficulty, write sudo su on terminal and give your password and then you will be free to give password on every command. sudo su Step 2 Download kernel either from https://www.kernel.org/pub/linux/kernel/v3.x/ or https://www.kernel.org/ . Note: In my case, I was using VMware and I downloaded kernel on windows, later when I tried to copy kernel from Windows to VMware Ubuntu, Paste option was hide. So use browser of Ubuntu to download kernel.

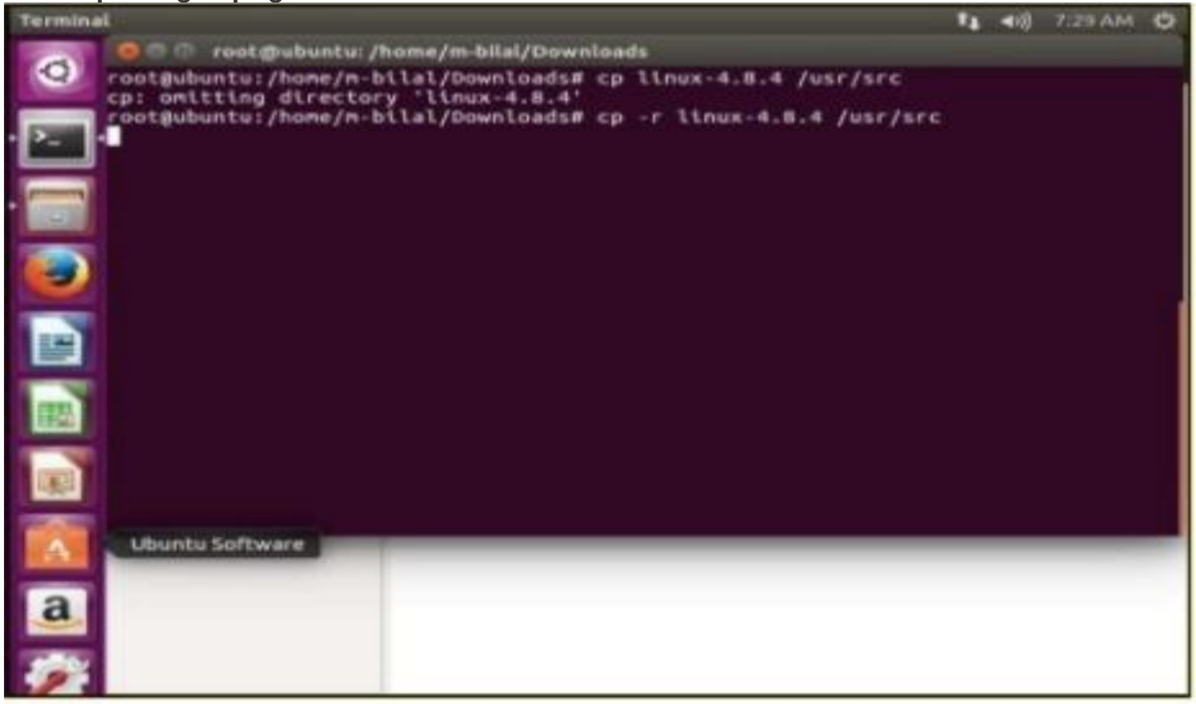
   

   linux-4.8.4.tar.xz
   5 minutes remaining — 27.6 of 87.7 MB (176 KB/sec)

   .

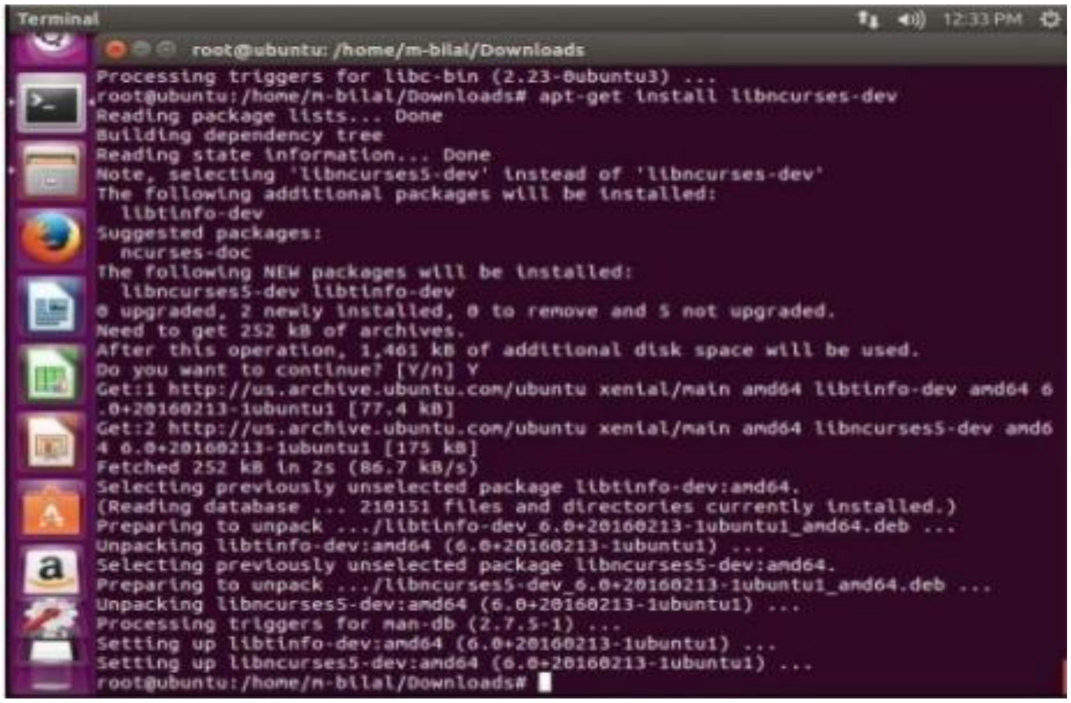
2. . **Now copying the kernel source from the downloads to the usr/src directory. For this first go to Downloads directory cd Downloads Enter the following code:**
   cp –r linux-4.8.4 /usr/src Step 3 Prerequisites: Enter the following codes line by line to make sure you have all the necessary and required packages to add a system call and then compile the kernel successfully. apt-get update apt-get upgrade apt-get install

kernel-package apt-get install libncurses-dev



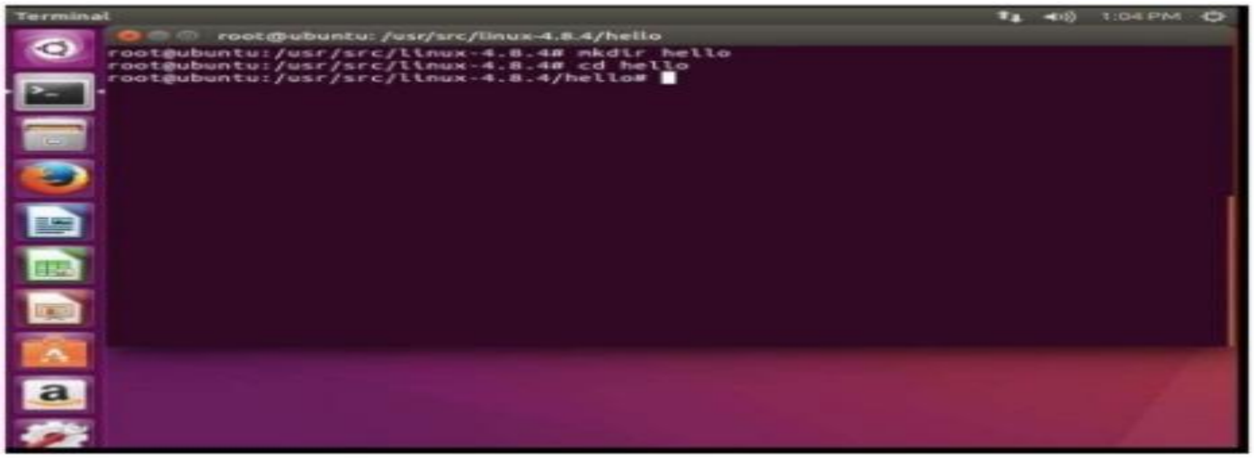
```
root@ubuntu:/home/m-bilal/Downloads# cp linux-4.8.4 /usr/src
cp: omitting directory 'linux-4.8.4'
root@ubuntu:/home/m-bilal/Downloads# cp -r linux-4.8.4 /usr/src
```

3. . **If prompted for a yes/No enter Y for yes and press enter to allow the download again this may take some time depending on the internet speed. You will get a screen like this after all the packages have been installed Step 4 Now direct yourself to the directory of the linux-4.8.4 in usr/src with the following command: cd /usr/src/linux-4.8.4 Now create a new folder named hello in linux-4.8.4 directory using mkdir hello. NOTE:You canmake this hello directory manually by going to linux-4.2.8 directory and then right click on screenand create a new folder and rename it to hello. Now change into hello directory to proceed further. cd hello and Create a “hello.c” file in the hello folder with the following code: vim hello.c NOTE:In my case, whenI wrote vim hello.c, I got error that vim is not installed, so for installing vim, you have to write apt install vim on terminal.**



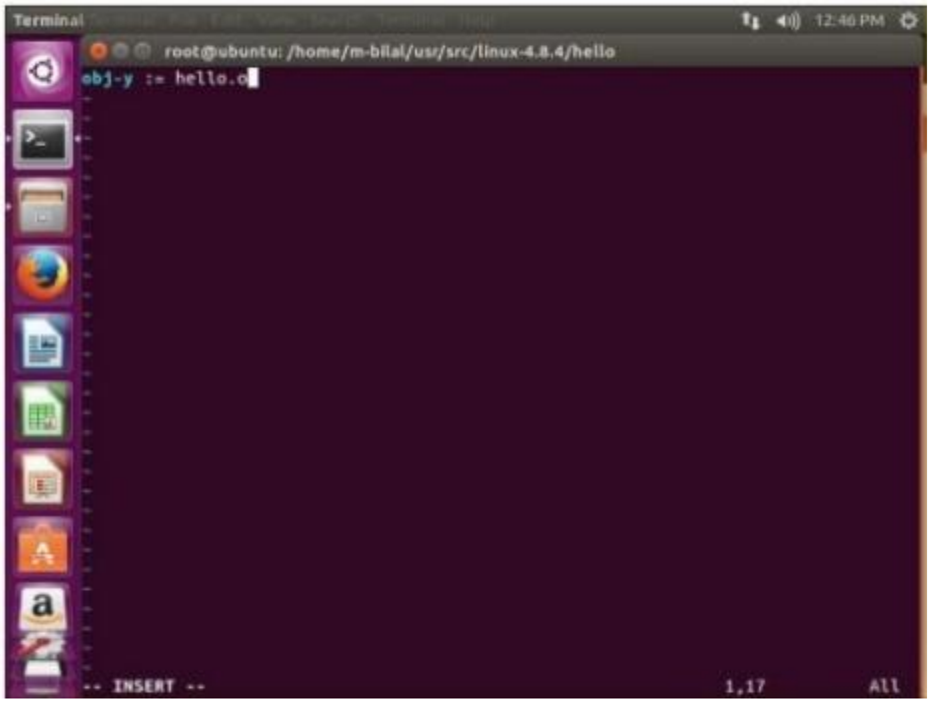
```
Processing triggers for libc-bin (2.23-0ubuntu3) ...
root@ubuntu:/home/m-bilal/Downloads# apt-get install libncurses-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'libncurses5-dev' instead of 'libncurses-dev'
The following additional packages will be installed:
  libtinfo-dev
Suggested packages:
  ncurses-doc
The following NEW packages will be installed:
  libncurses5-dev libtinfo-dev
0 upgraded, 2 newly installed, 0 to remove and 5 not upgraded.
Need to get 252 kB of archives.
After this operation, 1,461 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 libtinfo-dev amd64 6
.0+20160213-1ubuntu1 [77.4 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 libncurses5-dev amd6
4 6.0+20160213-1ubuntu1 [175 kB]
Fetched 252 kB in 2s (86.7 kB/s)
Selecting previously unselected package libtinfo-dev:amd64.
(Reading database ... 210151 files and directories currently installed.)
Preparing to unpack .../libtinfo-dev_6.0+20160213-1ubuntu1_amd64.deb ...
Unpacking libtinfo-dev:amd64 (6.0+20160213-1ubuntu1) ...
Selecting previously unselected package libncurses5-dev:amd64.
Preparing to unpack .../libncurses5-dev_6.0+20160213-1ubuntu1_amd64.deb ...
Unpacking libncurses5-dev:amd64 (6.0+20160213-1ubuntu1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up libtinfo-dev:amd64 (6.0+20160213-1ubuntu1) ...
Setting up libncurses5-dev:amd64 (6.0+20160213-1ubuntu1) ...
root@ubuntu:/home/m-bilal/Downloads#
```

4. **Once you enter the above command it will open a text editor in the terminal. Simply paste these lines of codethere #include <linux/kernel.h> Asmlinkage long sys_hello(void) { Printk("Hello World!n"); //printk prints the message to the kernels logs return 0; } Your screen should now look like this: Now just press Esc key to leave insert mode and then type :wq and press enter which will save the file and exit VM editor.**
5. **Create the Makefile in the hello directory. vim Makefile Using the same process as we did before, add the following line in the opened up vim text editor and save it by first hitting the Esc key then typing :wq and then press enter. obj-y := hello.o Now Add the hello directory (recently you have created in linux-4.8.4 directory) to the Main kernel Makefile. For this first you have to step out of the hello folder back to the linux directory and open the Makefile there for editing: $ cd .. OR $ cd /usr/src/linux-4.8.4 $ vim Makefile Navigate to line 893 (may vary but will be somewhere close to this number) to find the following line: Core-y += kernel/ mm/ fs/ ipc/ security/ cryptop/block/ Now add "hello/" at the end of it: Core-y += kernel/ mm/ fs/ ipc/ security/ cryptop/block/ hello/ NOTE:There is a space after every slash (/).**



```
root@ubuntu:/usr/src/linux-4.8.4# mkdir hello
root@ubuntu:/usr/src/linux-4.8.4# cd hello
root@ubuntu:/usr/src/linux-4.8.4/hello#
```

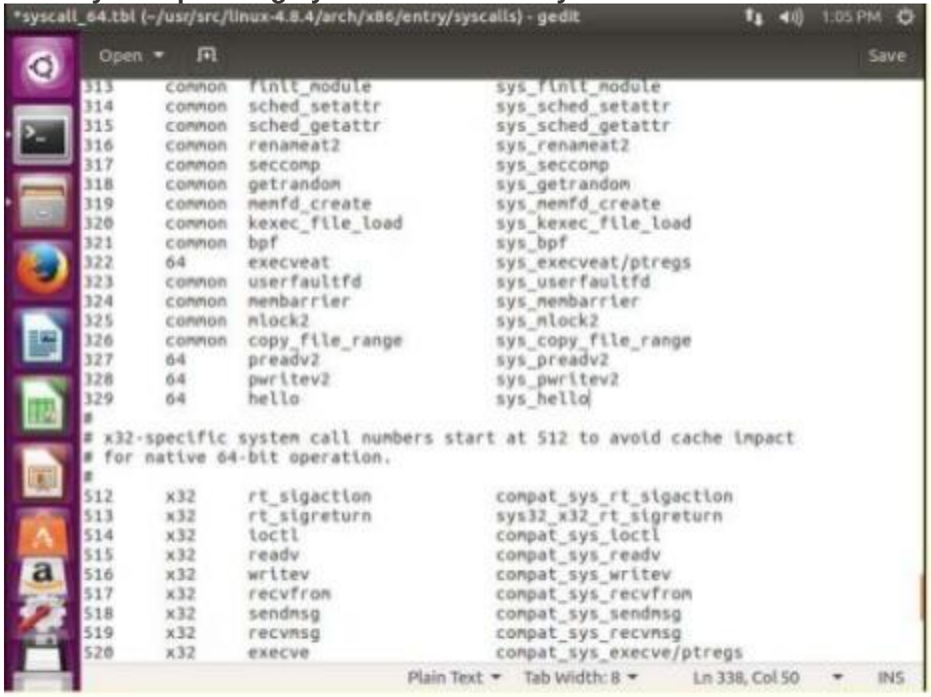
6. **This tells our compiler that the source codeof sys_hello() lies in the hello directory. Step 10 Add sys_hello() into the syscall table Navigate to this directory: $ cd usr/src/linux-4.8.4/arch/x86/entry/syscalls**
7. **. Now if your OS is 32-bit. Write this on terminal gedit syscall_32.tbl Then add the following line at the end where integer count ends and increment integers by one (In this case, if integers end at 379, than write this line with the start of 380) and write it 380 i386 hello sys_hello Now if your OS is 64-bit. Write this on terminal gedit syscall_64.tbl Then add the following line at the end where integer count ends and increment integers by one (In this case, if integers end at 379, than write this line with the start of 380) and write it 329 64 hello sys_hello**



```
obj-y := hello.o
-- INSERT --                                  1,17          All
```

8. **Add the sys_hello() in the sysc-alls.h header file and open the syscalls.h file for editing: Navigate to this directory: $ cd /usr/src/linux-4.8.4/include/linux gedit syscalls.h**
9. **Now add the following line just before the #endif statement: asmlinkage long sys_hello(void);**
10. **Save it and close it , Now create the config file: Navigate to this directory: $ cd /usr/src/linux-4.8.4/ and write this in terminal make menuconfig Incase you get an error stating that you do not have the curses.hfile then simply input the following command to download and install it and then try the previous step: apt-get install libncurses5-dev libncursesw5-dev**
11. **Build the Kernel This is the longest step of the process.The build will take 2 to 3 HOURS. Navigate to this directory: $ cd /usr/src/linux-4.8.4 To increase the speed of the build enters the command below: export CONCURRENCY_LEVEL=3 # 1+number of cores on your processor Now enter the following command to build the kernel make NOTE:After writing make command to build my kernel. I facedthis error of openssl/opensslv.h:No such file or directory, Compilation terminated.**
12. **. SOLUTION: Navigate to your home directory and now install these: apt-get install openssl apt-get install cl-plus-ssl apt-get install libssl-ocaml apt-get install libsslcommon2 apt-get install libsslcommon2-dev apt-get install libssl-dev apt-get install libssl-doc apt install openssn**

13. . **Now navigate to linux-4.8.4 directory and write make. Compilation will start Step 14 Once the build is complete it will return to the main linux-4.8.4 folder directory Now install the kernel with the following command: Make modules_install install Now rebootyour operating system for the newly installed linux kernel to take effect. Reboot**



*syscall_64.tbl (~/usr/src/linux-4.8.4/arch/x86/entry/syscalls) - gedit

```
313     common  finit_module            sys_finit_module
314     common  sched_setattr           sys_sched_setattr
315     common  sched_getattr           sys_sched_getattr
316     common  renameat2               sys_renameat2
317     common  seccomp                 sys_seccomp
318     common  getrandom               sys_getrandom
319     common  memfd_create            sys_memfd_create
320     common  kexec_file_load         sys_kexec_file_load
321     common  bpf                     sys_bpf
322     64      execveat                sys_execveat/ptregs
323     common  userfaultfd             sys_userfaultfd
324     common  membarrier              sys_membarrier
325     common  mlock2                  sys_mlock2
326     common  copy_file_range         sys_copy_file_range
327     64      preadv2                 sys_preadv2
328     64      pwritev2                sys_pwritev2
329     64      hello                   sys_hello
#
# x32-specific system call numbers start at 512 to avoid cache impact
# for native 64-bit operation.
#
512     x32     rt_sigaction            compat_sys_rt_sigaction
513     x32     rt_sigreturn            sys32_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/ptregs
```

14.

15. **After rebootto chec5k for your version of kernel that is installed type the following command: uname -r My result returned: 4.8.4 Test the system call: Create a program that uses your system call and you may call it what ever you like. I'm going to call mine "test.c" Add the following codeto it: #include <stdio.h> #include <linux/kernel.h> #include <sys/syscall.h> #include <unistd.h> int main() { long int sys = syscall(380); // 380 is the sys_hello number I used to add the sys_call printf("System call sys_hello returned %ldn", sys); // 0 shows that our program returns 0 and works return 0; } Then compile it using the standard gcc compiler with the following code: gcc –o test test.c And execute with this: ./test The result should show this: System call sys_hello returned 0. Now type the following command to display the kernel message which should show "Hello World": dmesg**

16. 22. THE END