School of Computing Science and Engineering

Operating System Project

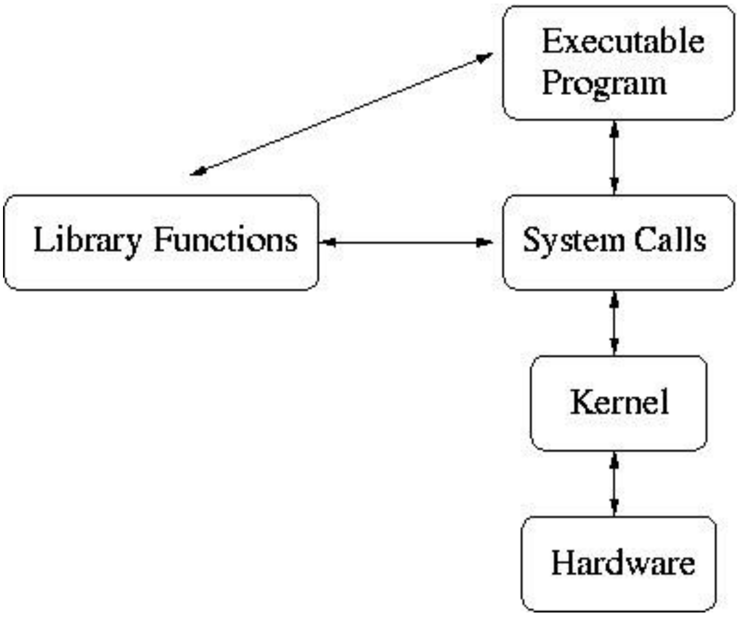
Adding a system call to the Linux kernel

NAME	REGISTRATION NUMBER
SR NAVYA SREE	16BCE0223
RAJDEEPA CHAKRABARTY	16BCE0732
SHAIK DILSHATH	16BCE0669
PANKITHA	16BCE2016

Submitted To
Prof. VIJAYA KUMAR K

ABSTRACT

This project is based on adding a System Call to our operating system i.e UBUNTU. System calls are kernel functions that serve as an interface (for user mode application) to invoke kernel services like drivers, file systems, network stacks and others. System calls are also referred as system entry points since applications can enter kernel mode only through a valid system call interface. Applications can step into system calls using special processor specific soft interrupt instructions.



The aim of this project is to take a string as an argument and print the same when the cat command is executed. In our project, we will be working on Linux kernel **version 4.10.13**. We will create, compile and execute our system call on **UBUNTU**.

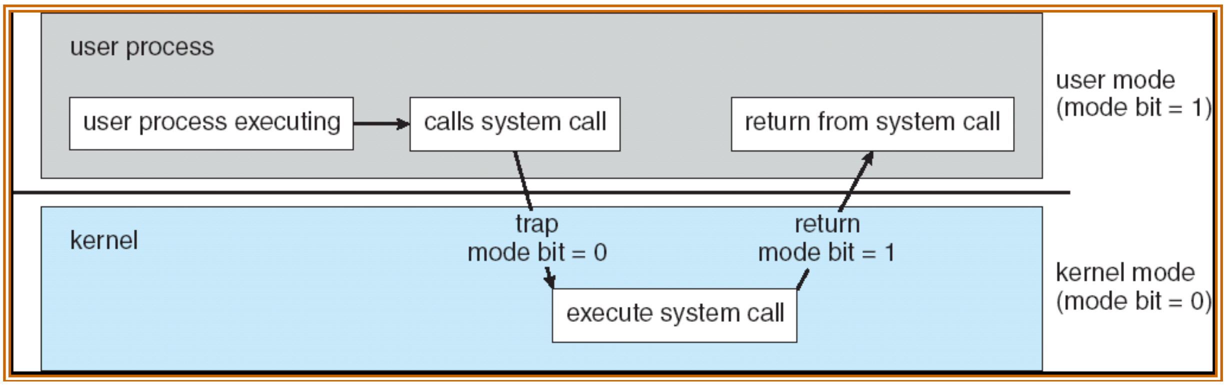
INTRODUCTION

This project is based on adding a System Call to our operating system i.e. UBUNTU. System calls are kernel functions that serve as an interface (for user mode application) to invoke kernel services like drivers, file systems, network stacks and others. System calls are also referred as system entry points since applications can enter kernel mode only through a valid system call interface. Applications can step into system calls using special processor specific soft interrupt instructions.

Background – User vs. Kernel mode

- Hardware provides two modes indicated by bit in PSW.
- Allows OS to protect itself and system components against faulty and malicious processes.
- Some instructions designated as privileged are only executable in kernel mode.
- System call, all traps and interrupts change mode from user to kernel and return from system call resets mode to user.

Transition from user to kernel mode



- Each different system call has its own number or other identity.
- Kernel trap handler uses system call number to index into the table of system call routines.

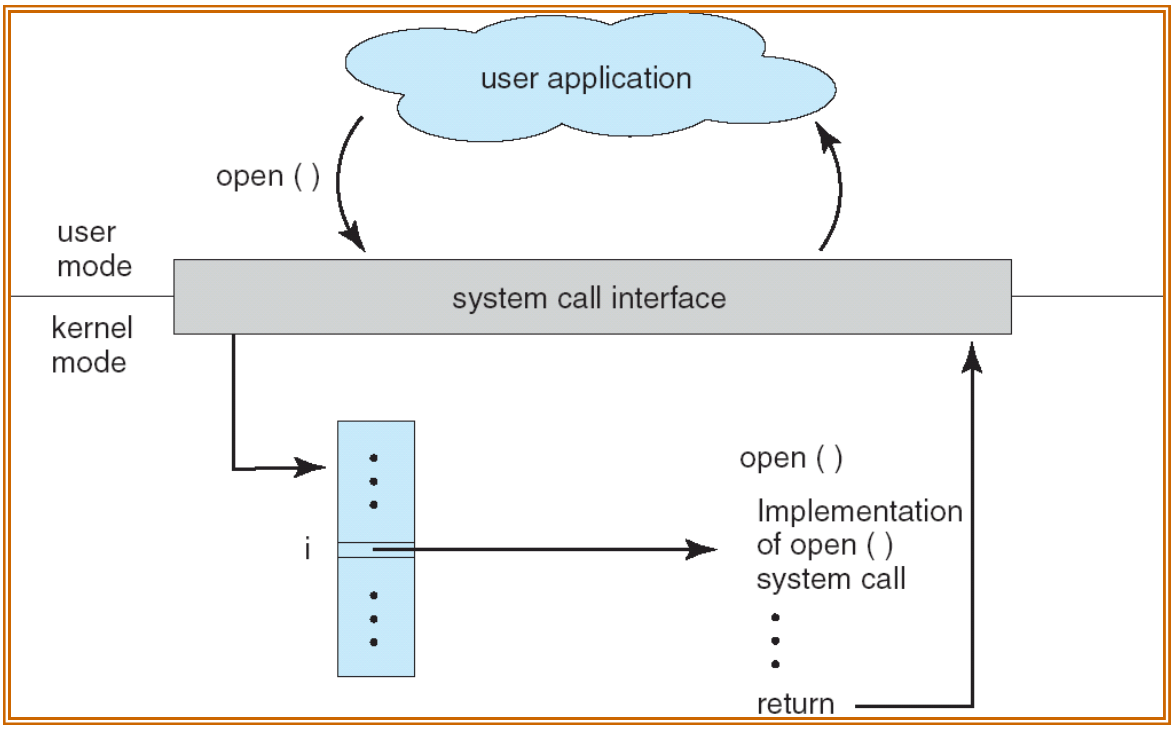
Inside kernel, the OS can

- Read and modify data structures not in user address space.
- Control devices and hardware settings forbidden to user processes.
- Invoke operating system functions not available to user processes.

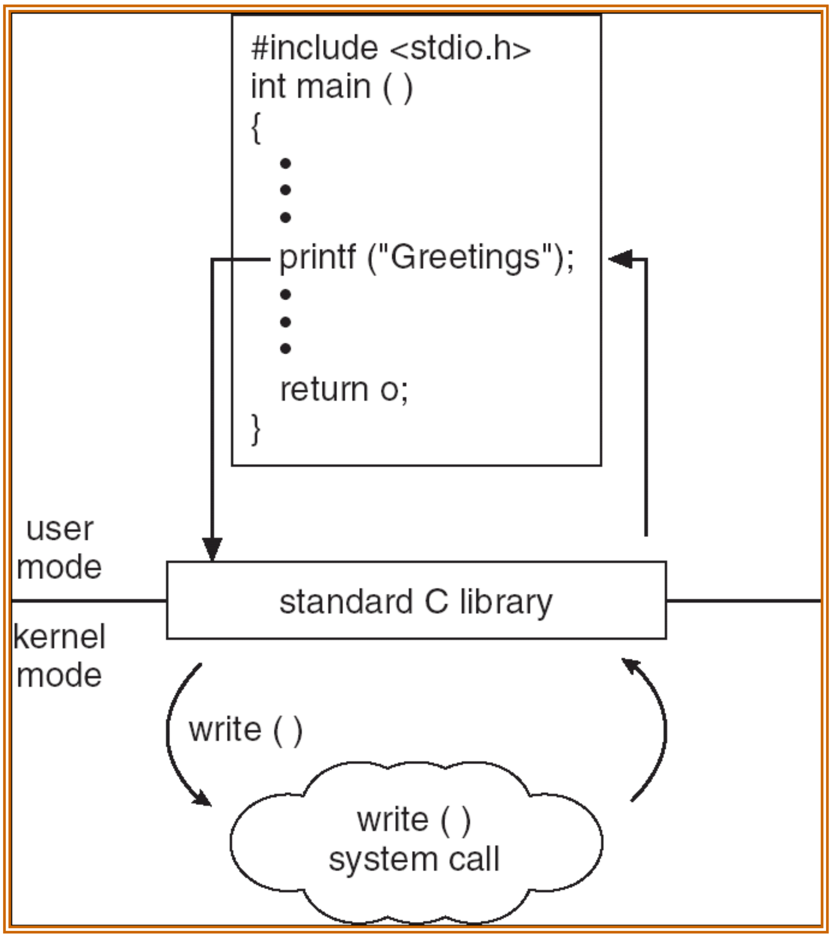
Accessing the kernel via system call

- Normally embedded within a library routine and the user API never makes system calls directly.
- System call mechanism is machine specific and different CPU architectures make system calls in different ways.
- System call numbers are different for various architectures even for same operating system and version.

Accessing kernel via library interface.



Accessing kernel via library interface

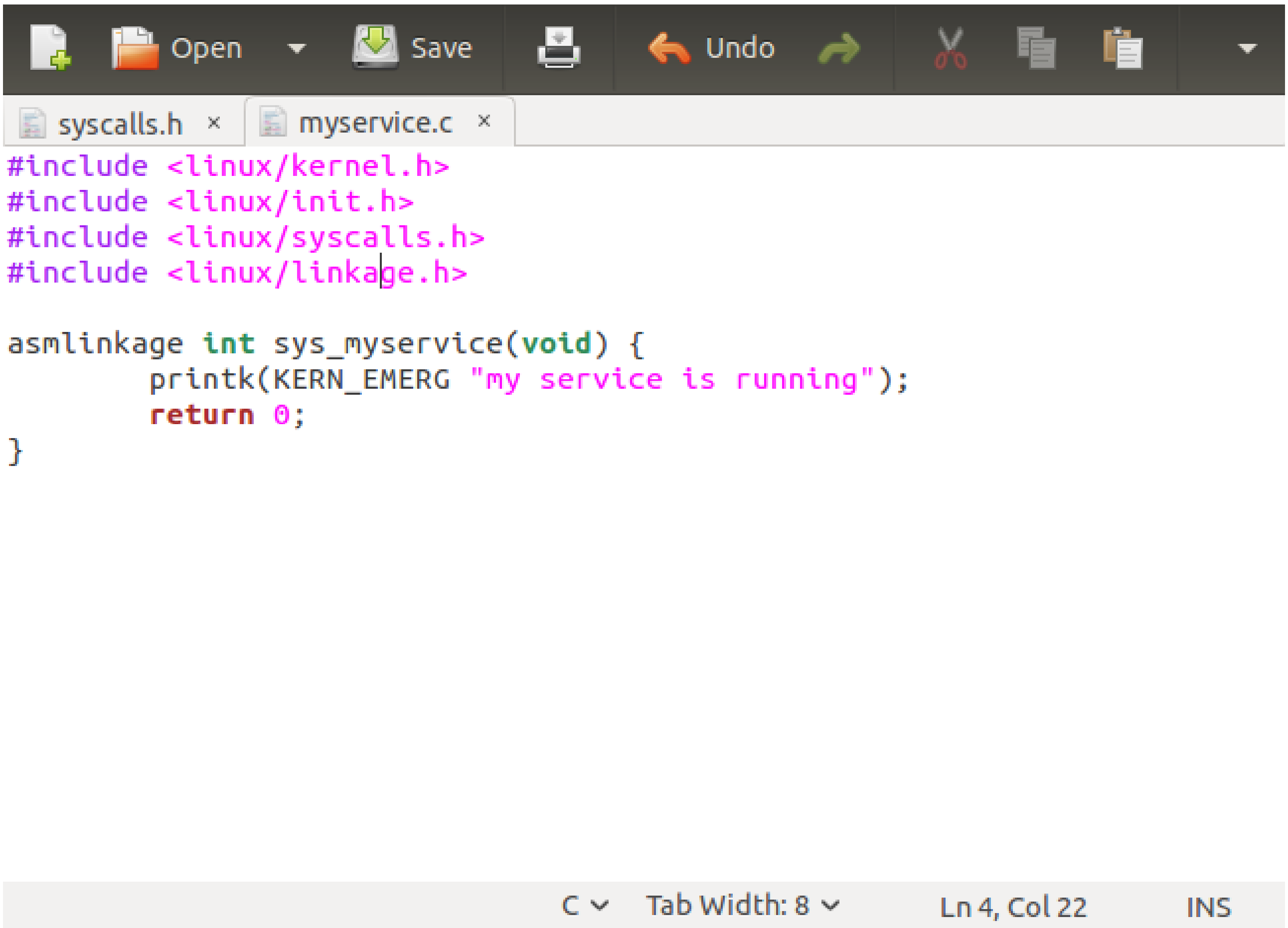


Requirements

- To carry out this project we need Ubuntu on our computers or we can carry this project out by installing VirtualBox on Windows Operating Systems to work on a virtual Linux Terminal.
- We will be using the Terminal interface since on the terminal we can directly install the software.
- We give a terminal command to get started:
  sudo apt_get install packagename

PROPOSED MODEL

Design




```c
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/syscalls.h>
#include <linux/linkage.h>

asmlinkage int sys_myservice(void) {
        printk(KERN_EMERG "my service is running");
        return 0;
}
```

Implementation

1) **Getting Started**

A user-mode procedure call is performed by passing arguments to the called procedure either on the stack or through registers, saving the current state and the value of the program counter, and jumping to the beginning of the code corresponding to the called procedure. The process continues to have the same privileges as before. System calls appear as procedure calls to user programs but result i11 a change in execution context and privileges. In Linux on the Intel386 architecture, a system call is accomplished by storing the system-call number into the EAX register, storing arguments to the system call in other hardware registers, and executing a trap instruction (which is the INT Ox80 assembly instruction). After the trap is executed, the system- call number is used to index into a table of code pointers to obtain the starting address for the handler code implementing the system call. The process then jumps to this address, and the privileges of the process are switched from user to kernel mode. With the expanded privileges, the process can now execute kernel code, which may include privileged instructions that cannot be executed in user mode. The kernel code can then carry out the requested services, such as interacting with I/O devices, and can perform process management and other activities that cannot be performed in user mode. The system call numbers for recent versions of the Linux ker- nel are listed in lusr I src/linux-2. xl include/ asm-i386/unistd. h. (For instance, __ NR_close corresponds to the system call close 0, which is invoked for closin.g a file descriptor, and is defined as value 6.) The list of pointers to system-call handlers is typically stored in the file lusrlsrcllinux-2.x/arch/i386/kernel/entry.S under the head- ing ENTRY (sys_calLtable). Notice that sys_close is stored at entry number 6 in the table to be consistent with the system-call number defined in the unistd. h file. (The keyword .long denotes that the entry will occupy the same number of bytes as a data value of type long.)

2) Building a New Kernel

Before adding a system call to the kernel, you must familiarize yourself with the task of building the binary for a kernel from its source code and booting the machine with the newly built kernel. This activity comprises the following tasks, some of which depend on the particular installation of the Linux operating system in use.

Obtain the kernel source code for the Linux distribution. If the source code package has already been installed on your machine, the corresponding files might be available under lusr I srcllinux or /usr I src/linux-2. x (where the suffix corresponds to the kernel version number). If the package has not yet been installed, it can be downloaded from the provider of your Linux distribution or from http:l/www.kernel.org.

Learn how to configure, compile, and install the kernel binary. This will vary among the different kernel distributions, but some typical commands for

building the kernel (after entering the directory where the kernel source code is stored) include:
o make xconfig
o make dep
o make bzimage
Add a new entry to the set of boatable kernels supported by the system. The Linux operating system typically uses utilities such as lilo and grub to maintain a list ofbootable kernels from which the user can choose during machine boot-up. If your system supports lilo, add an entry to lilo. conf, such as:
image=/boot/bzimage.mykernel label=mykernel root=/dev/hda5 read-only
where lbootlbzimage. my kernel is the kernel image and my kernel is the label associated with the new kernel. This step will allow you to choose the new kernel during the boot-up process. You will then have the option of either booting the new kernel or booting the unmodified kernel if the newly built kernel does not ftmction properly.

3) Extending the Kernel Source

You can now experiment with adding a new file to the set of source files used for compiling the kernel. Typically, the source code is stored in the lusr I srcllinux-2. xlkernel directory, although that location may differ in your Linux distribution. There are two options for adding the system call. The first is to add the system call to an existing source file in this directory. The second is to create a new file in the source directory and modify lusr I srcllinux-2. xlkerneliMakefile to include the newly created file in the compilation process. The advantage of the first approach is that when you modify an existing file that is already part of the compilation process, the Makefile need not be modified.

4) Adding a System Call to the Kernel

Now that you are familiar with the various background tasks corresponding to building and booting Linux kernels, you can begin the process of adding a new system call to the Linux kernel. In this project, the system call will have limited functionality; it will simply transition from user mode to kernel mode, print a message that is logged with the kernel messages, and transition back to user mode. We will call this the myservice system call. While it has only limited functionality, it illustrates the system-call mechanism and sheds light on the interaction between user programs and the kernel.

a) Create a new file called helloworld. c to define your system call. Include the header files linuxllinkage.h and linuxlkernel. h. Add the following code to this file:

```
#include <linuxllinkage.h>
#include <linuxlkernel.h>
asmlinkage int sysJ
myservice()
{ printk(KERN_EMERG "my service is running");
return 1;
}
```

This creates a system call with the name sys_myservice (). If you choose to add this system call to an existing file in the source directory, all that is necessary is to add the sys_myservice () function to the file you choose. In the code, asmlinkage is a rellli<ant from the days when Linux used both C++ and C code and is used to indicate that the code is written in C. The printk () function is used to print messages to a kernel log file and therefore may be called only from the kernel. The kernel mes- sages specified in the parameter to printk () are logged in the file /var/log/kernel/warnings. The function prototype for the printk () call is defined in /usr /include/linux/kernel. h.

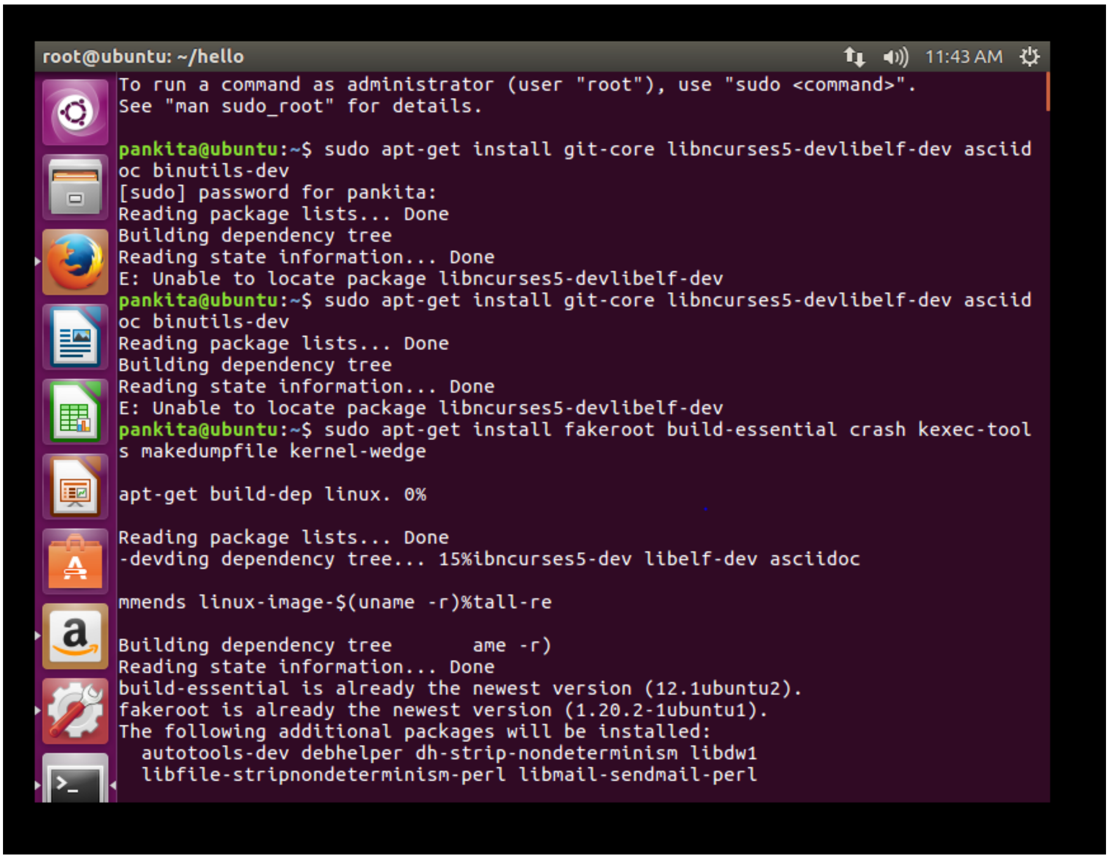
b) Define a new system call number for __ NR_myservice in /usr/src/linux-2.x/include/asm-i386/unistd.h. A user program can use this number to identify the newly added system call. Also be sure to increment the value for __ NR_syscalls, which is stored in the same file. This constant tracks the number of system calls currently defuced in the kernel.
c) Add an entry .long sys_myservice to the sys_calLtable definedinthe/usr/src/linux-2.x/arch/i386/kernel/entry.S file. As discussed earlier, the system-call number is used to index into this table to find the position of the handler code for the invoked system call.
d) Add your file myservice.c to the Makefile (if you created a new file for your system call.) Save a copy of your old kernel binary image (in case there are problems with your newly created kernel). You can now build the new kernet rename it to distinguish it from the unmodified kernet and add an entry to the loader configuration files (such as lilo. conf). After completing these steps, you can boot either the old kernel or the new kernel that contains your system call.

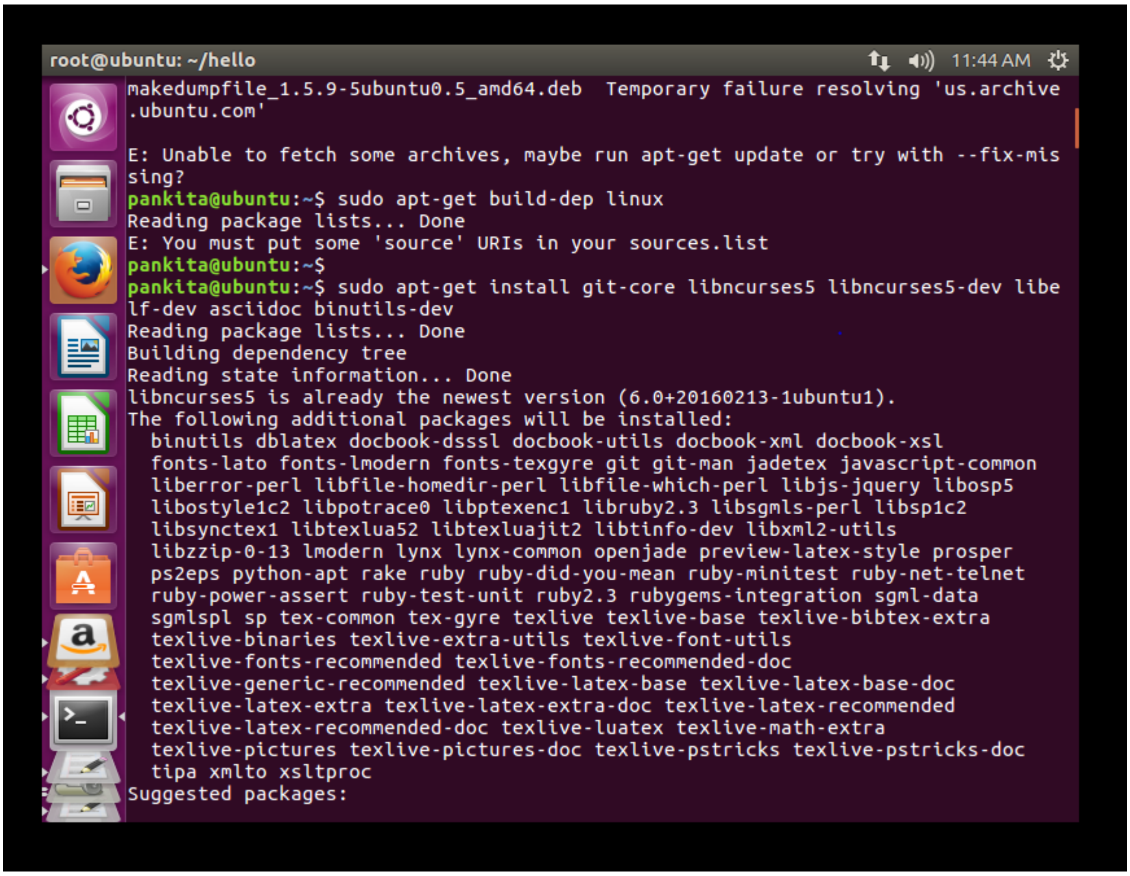
5) Using the System Call from a User Program

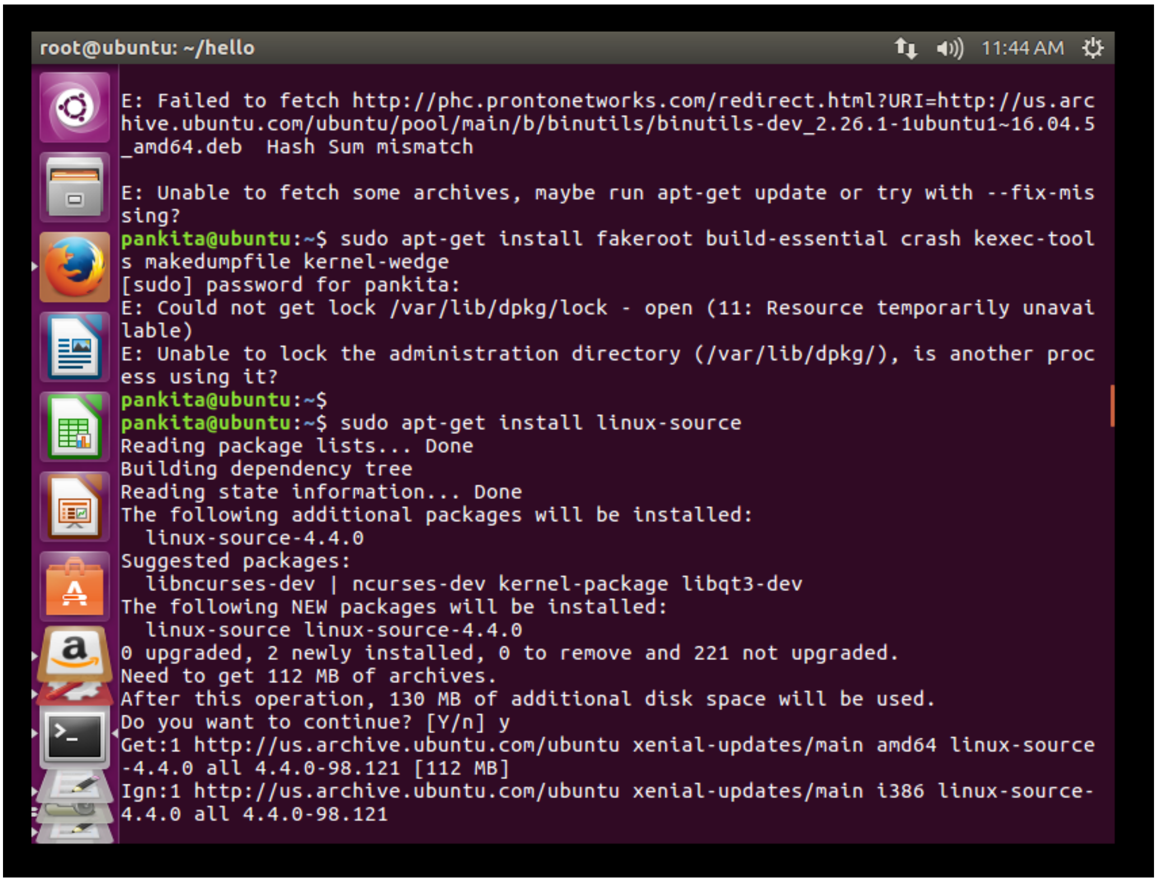
When you boot with the new kernet it will support the newly defined system call; you now simply need to invoke this system call from a user program. Ordinarily, the standard C library supports an interface for system calls defined for the Linux operating system. As your new system call is not linked into the standard C library, however, invoking your system call will require manual intervention. As noted earlie1~ a system call is invoked by storing the appropriate value in a hardware register and performing a trap instruction. Unfortunately, these low-level operations cannot be performed using C language statements and instead require assembly instructions. Fortunately, Linux provides macros for instantiating wrapper functions that contain the appropriate assembly instructions. For instance, the following C program uses the _syscallO () macro to invoke the newly defined system call:

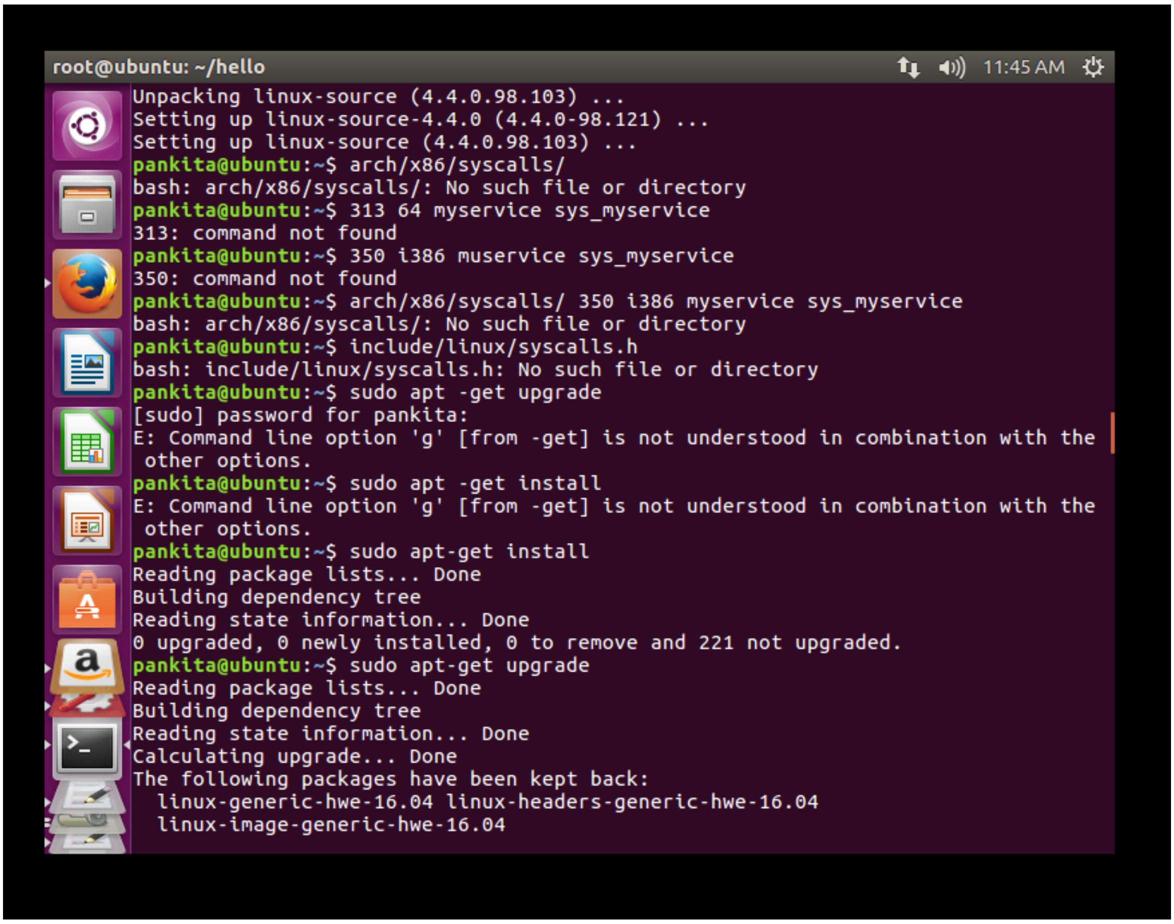
```
#include <linux/errno.h>
#include <sys/syscall.h>
#include <linux/unistd.h>
_syscallO(int, myservice();
main()
{
        myservice ();
 }
```

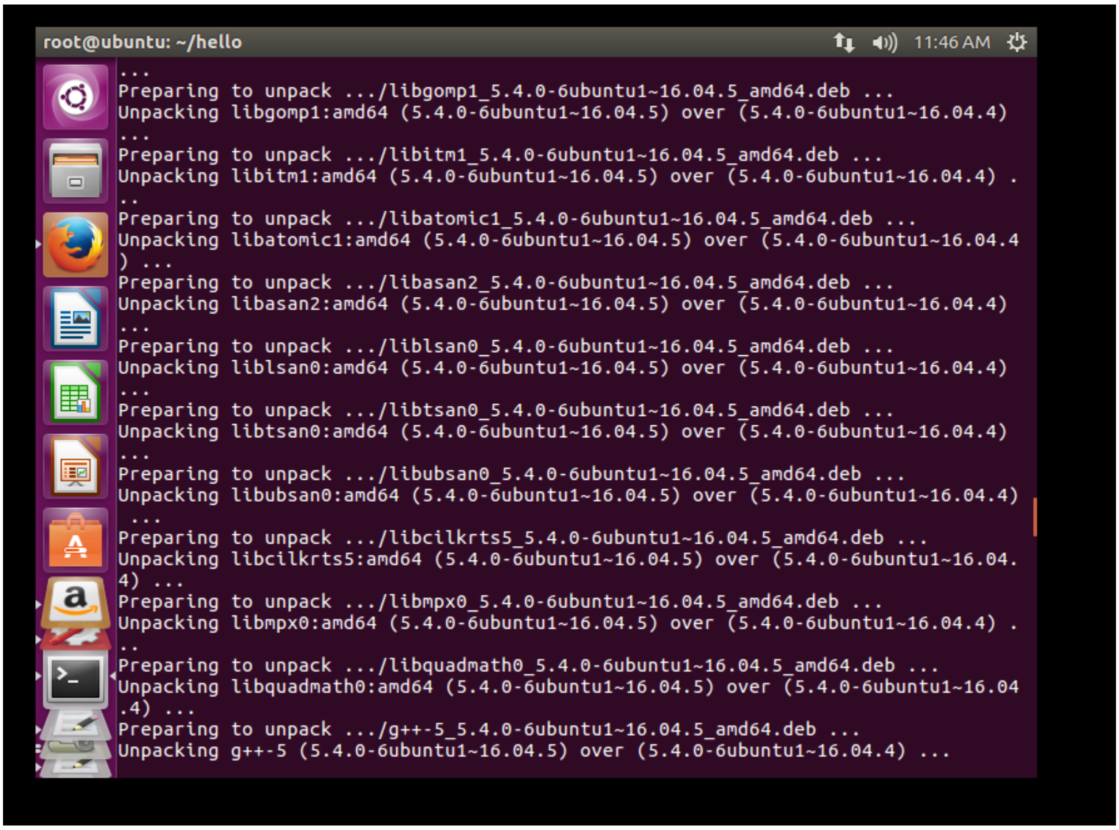
a) The _syscallO macro takes two arguments. The first specifies the type of the value returned by the system call; the second is the name of the system call. The name is used to identify the system- call number that is stored in the hardware register before the trap instruction is executed. If your system call requires arguments, then a different macro (such as _syscallO, where the suffix indicates the number of arguments) could be used to instantiate the assembly code required for performing the system call.
b) Compile and execute the program with the newly built kernel. There should be a message “my service is running" in the kernel log file /var/log/kernel/warnings to indicate that the system call has executed.

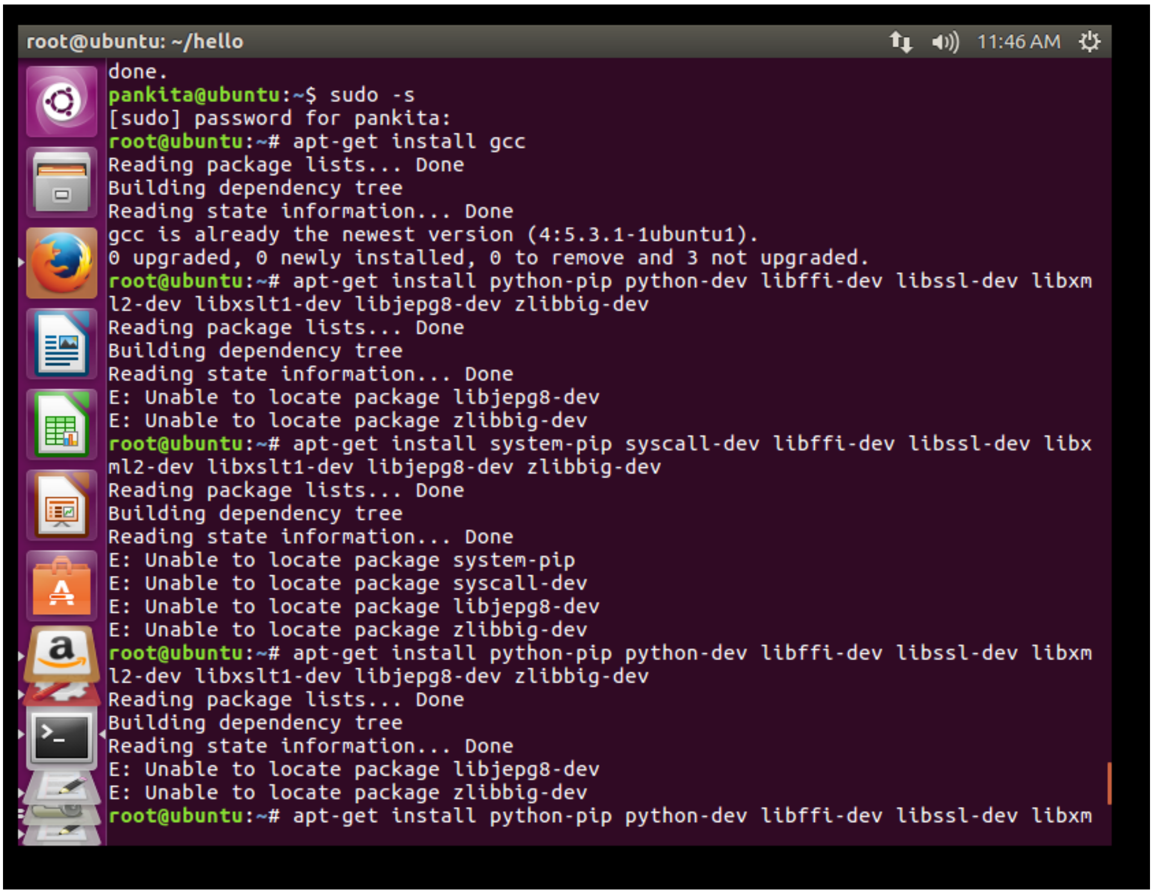

root@ubuntu: ~/hello
11:43 AM
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
pankita@ubuntu:~$ sudo apt-get install git-core libncurses5-devlibelf-dev asciid
oc binutils-dev
[sudo] password for pankita:
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package libncurses5-devlibelf-dev
pankita@ubuntu:~$ sudo apt-get install git-core libncurses5-devlibelf-dev asciid
oc binutils-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package libncurses5-devlibelf-dev
pankita@ubuntu:~$ sudo apt-get install fakeroot build-essential crash kexec-tool
s makedumpfile kernel-wedge
apt-get build-dep linux. 0%
Reading package lists... Done
-devding dependency tree... 15%ibncurses5-dev libelf-dev asciidoc
mmends linux-image-$(uname -r)%tall-re
Building dependency tree ame -r)
Reading state information... Done
build-essential is already the newest version (12.1ubuntu2).
fakeroot is already the newest version (1.20.2-1ubuntu1).
The following additional packages will be installed:
autotools-dev debhelper dh-strip-nondeterminism libdw1
libfile-stripnondeterminism-perl libmail-sendmail-perl

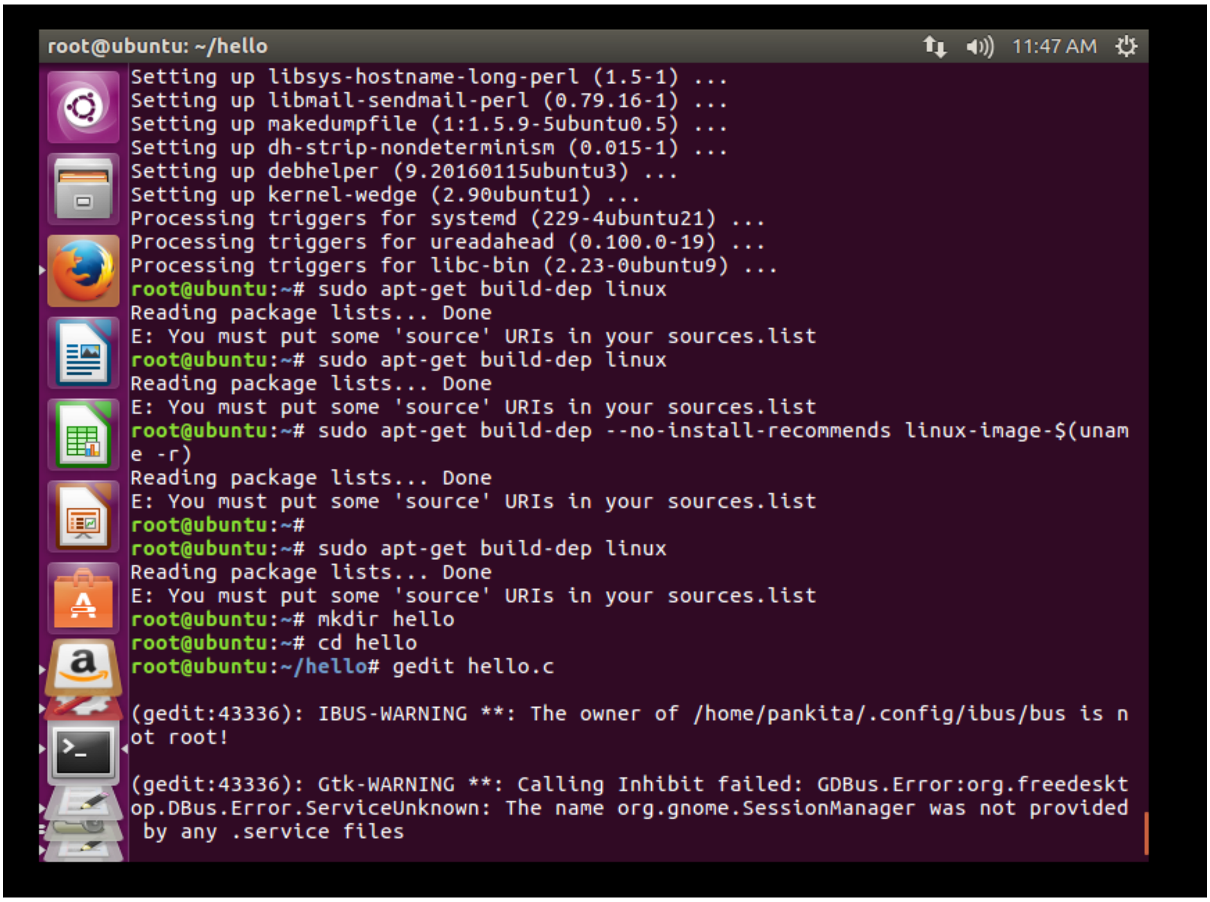

root@ubuntu: ~/hello
11:44 AM
makedumpfile_1.5.9-5ubuntu0.5_amd64.deb Temporary failure resolving 'us.archive
.ubuntu.com'
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-mis
sing?
pankita@ubuntu:~$ sudo apt-get build-dep linux
Reading package lists... Done
E: You must put some 'source' URIs in your sources.list
pankita@ubuntu:~$
pankita@ubuntu:~$ sudo apt-get install git-core libncurses5 libncurses5-dev libe
lf-dev asciidoc binutils-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
libncurses5 is already the newest version (6.0+20160213-1ubuntu1).
The following additional packages will be installed:
binutils dblatex docbook-dsssl docbook-utils docbook-xml docbook-xsl
fonts-lato fonts-lmodern fonts-texgyre git git-man jadetex javascript-common
liberror-perl libfile-homedir-perl libfile-which-perl libjs-jquery libosp5
libostyle1c2 libpotrace0 libptexenc1 libruby2.3 libsgmls-perl libsp1c2
libsynctex1 libtexlua52 libtexluajit2 libtinfo-dev libxml2-utils
libzzip-0-13 lmodern lynx lynx-common openjade preview-latex-style prosper
ps2eps python-apt rake ruby ruby-did-you-mean ruby-minitest ruby-net-telnet
ruby-power-assert ruby-test-unit ruby2.3 rubygems-integration sgml-data
sgmlspl sp tex-common tex-gyre texlive texlive-base texlive-bibtex-extra
texlive-binaries texlive-extra-utils texlive-font-utils
texlive-fonts-recommended texlive-fonts-recommended-doc
texlive-generic-recommended texlive-latex-base texlive-latex-base-doc
texlive-latex-extra texlive-latex-extra-doc texlive-latex-recommended
texlive-latex-recommended-doc texlive-luatex texlive-math-extra
texlive-pictures texlive-pictures-doc texlive-pstricks texlive-pstricks-doc
tipa xmlto xsltproc
Suggested packages:

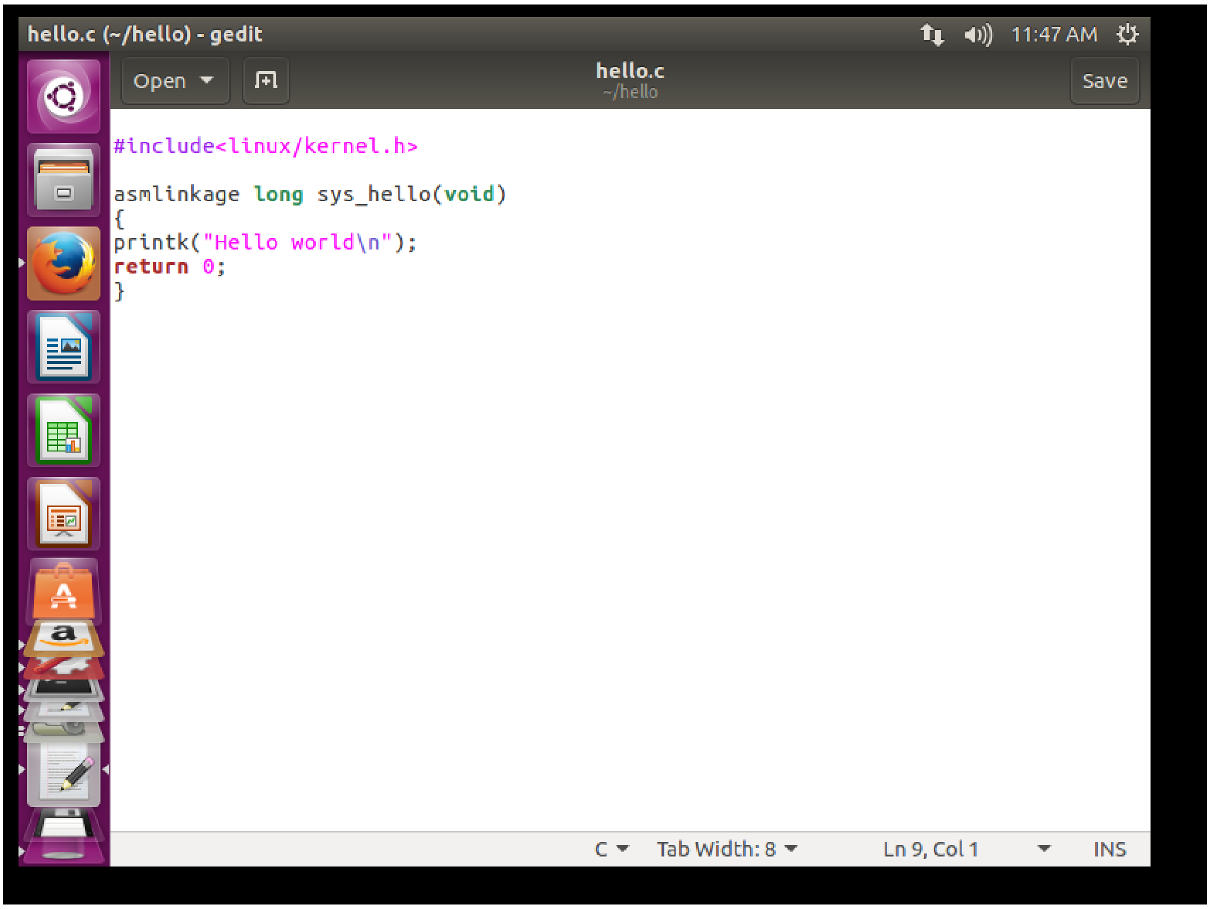

root@ubuntu: ~/hello
11:44 AM
E: Failed to fetch http://phc.prontonetworks.com/redirect.html?URI=http://us.archive.ubuntu.com/ubuntu/pool/main/b/binutils/binutils-dev_2.26.1-1ubuntu1~16.04.5_amd64.deb Hash Sum mismatch
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-missing?
pankita@ubuntu:~$ sudo apt-get install fakeroot build-essential crash kexec-tools makedumpfile kernel-wedge
[sudo] password for pankita:
E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is another process using it?
pankita@ubuntu:~$
pankita@ubuntu:~$ sudo apt-get install linux-source
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
linux-source-4.4.0
Suggested packages:
libncurses-dev | ncurses-dev kernel-package libqt3-dev
The following NEW packages will be installed:
linux-source linux-source-4.4.0
0 upgraded, 2 newly installed, 0 to remove and 221 not upgraded.
Need to get 112 MB of archives.
After this operation, 130 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 linux-source-4.4.0 all 4.4.0-98.121 [112 MB]
Ign:1 http://us.archive.ubuntu.com/ubuntu xenial-updates/main i386 linux-source-4.4.0 all 4.4.0-98.121

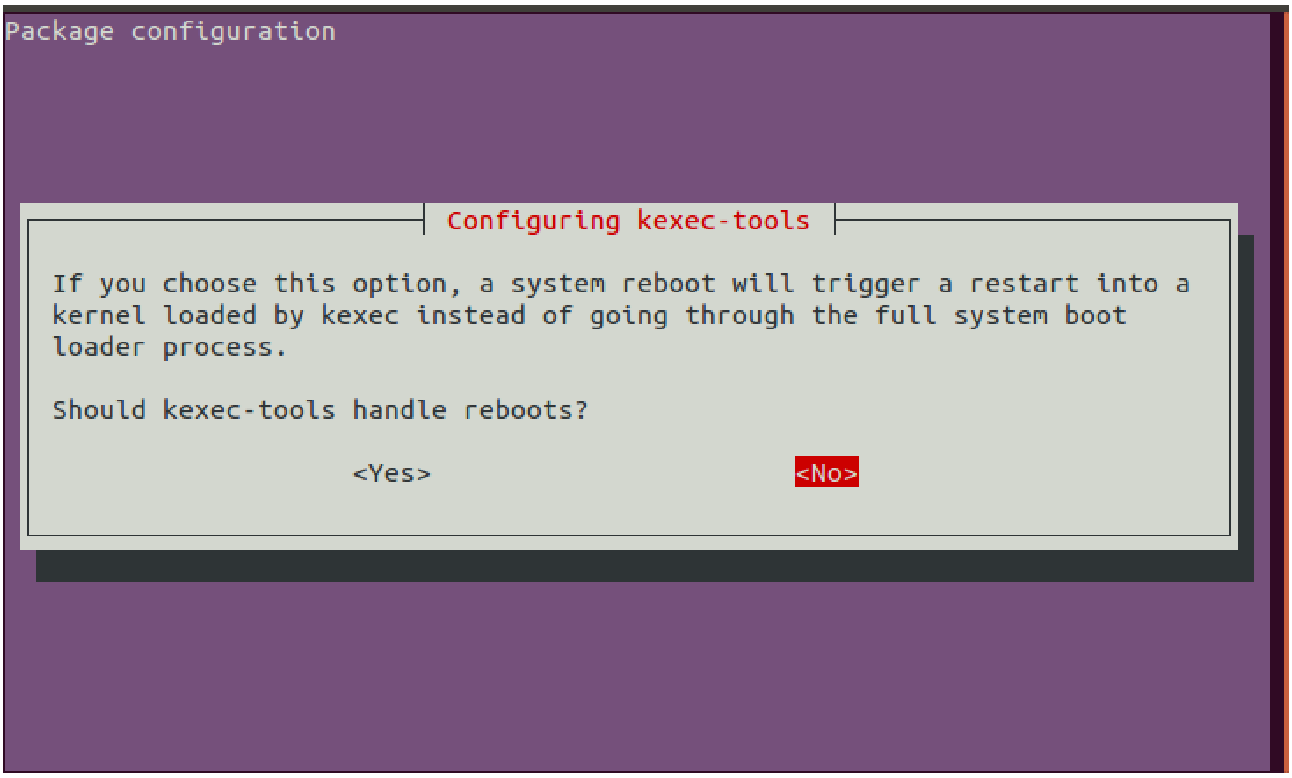

root@ubuntu: ~/hello
11:45 AM
Unpacking linux-source (4.4.0.98.103) ...
Setting up linux-source-4.4.0 (4.4.0-98.121) ...
Setting up linux-source (4.4.0.98.103) ...
pankita@ubuntu:~$ arch/x86/syscalls/
bash: arch/x86/syscalls/: No such file or directory
pankita@ubuntu:~$ 313 64 myservice sys_myservice
313: command not found
pankita@ubuntu:~$ 350 i386 muservice sys_myservice
350: command not found
pankita@ubuntu:~$ arch/x86/syscalls/ 350 i386 myservice sys_myservice
bash: arch/x86/syscalls/: No such file or directory
pankita@ubuntu:~$ include/linux/syscalls.h
bash: include/linux/syscalls.h: No such file or directory
pankita@ubuntu:~$ sudo apt -get upgrade
[sudo] password for pankita:
E: Command line option 'g' [from -get] is not understood in combination with the other options.
pankita@ubuntu:~$ sudo apt -get install
E: Command line option 'g' [from -get] is not understood in combination with the other options.
pankita@ubuntu:~$ sudo apt-get install
Reading package lists... Done
Building dependency tree
Reading state information... Done
0 upgraded, 0 newly installed, 0 to remove and 221 not upgraded.
pankita@ubuntu:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
linux-generic-hwe-16.04 linux-headers-generic-hwe-16.04
linux-image-generic-hwe-16.04

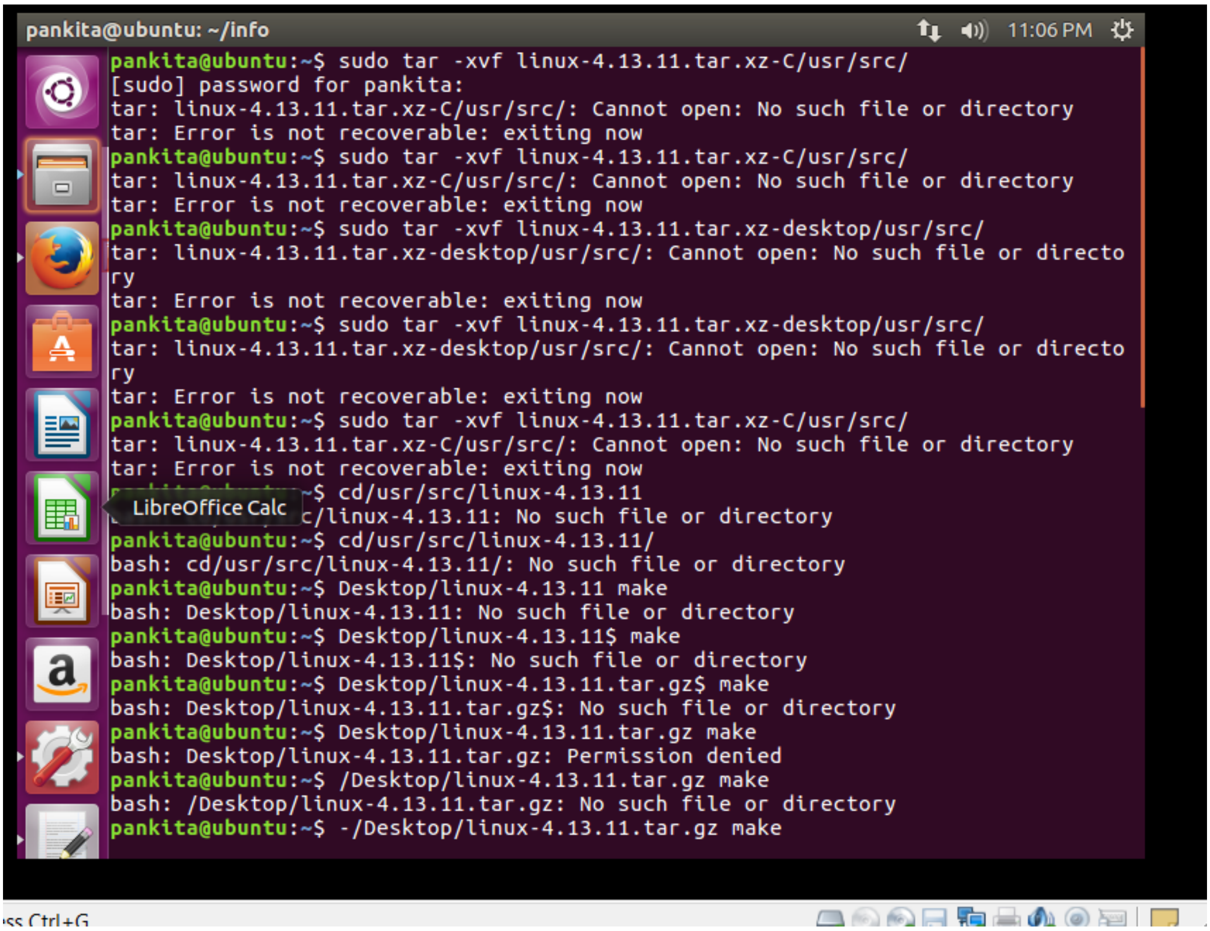

root@ubuntu: ~/hello
11:46 AM
...
Preparing to unpack .../libgomp1_5.4.0-6ubuntu1~16.04.5_amd64.deb ...
Unpacking libgomp1:amd64 (5.4.0-6ubuntu1~16.04.5) over (5.4.0-6ubuntu1~16.04.4)
...
Preparing to unpack .../libitm1_5.4.0-6ubuntu1~16.04.5_amd64.deb ...
Unpacking libitm1:amd64 (5.4.0-6ubuntu1~16.04.5) over (5.4.0-6ubuntu1~16.04.4) .
..
Preparing to unpack .../libatomic1_5.4.0-6ubuntu1~16.04.5_amd64.deb ...
Unpacking libatomic1:amd64 (5.4.0-6ubuntu1~16.04.5) over (5.4.0-6ubuntu1~16.04.4
) ...
Preparing to unpack .../libasan2_5.4.0-6ubuntu1~16.04.5_amd64.deb ...
Unpacking libasan2:amd64 (5.4.0-6ubuntu1~16.04.5) over (5.4.0-6ubuntu1~16.04.4)
...
Preparing to unpack .../liblsan0_5.4.0-6ubuntu1~16.04.5_amd64.deb ...
Unpacking liblsan0:amd64 (5.4.0-6ubuntu1~16.04.5) over (5.4.0-6ubuntu1~16.04.4)
...
Preparing to unpack .../libtsan0_5.4.0-6ubuntu1~16.04.5_amd64.deb ...
Unpacking libtsan0:amd64 (5.4.0-6ubuntu1~16.04.5) over (5.4.0-6ubuntu1~16.04.4)
...
Preparing to unpack .../libubsan0_5.4.0-6ubuntu1~16.04.5_amd64.deb ...
Unpacking libubsan0:amd64 (5.4.0-6ubuntu1~16.04.5) over (5.4.0-6ubuntu1~16.04.4)
...
Preparing to unpack .../libcilkrts5_5.4.0-6ubuntu1~16.04.5_amd64.deb ...
Unpacking libcilkrts5:amd64 (5.4.0-6ubuntu1~16.04.5) over (5.4.0-6ubuntu1~16.04.
4) ...
Preparing to unpack .../libmpx0_5.4.0-6ubuntu1~16.04.5_amd64.deb ...
Unpacking libmpx0:amd64 (5.4.0-6ubuntu1~16.04.5) over (5.4.0-6ubuntu1~16.04.4) .
..
Preparing to unpack .../libquadmath0_5.4.0-6ubuntu1~16.04.5_amd64.deb ...
Unpacking libquadmath0:amd64 (5.4.0-6ubuntu1~16.04.5) over (5.4.0-6ubuntu1~16.04
.4) ...
Preparing to unpack .../g++-5_5.4.0-6ubuntu1~16.04.5_amd64.deb ...
Unpacking g++-5 (5.4.0-6ubuntu1~16.04.5) over (5.4.0-6ubuntu1~16.04.4) ...


root@ubuntu: ~/hello
11:46 AM
done.
pankita@ubuntu:~$ sudo -s
[sudo] password for pankita:
root@ubuntu:~# apt-get install gcc
Reading package lists... Done
Building dependency tree
Reading state information... Done
gcc is already the newest version (4:5.3.1-1ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
root@ubuntu:~# apt-get install python-pip python-dev libffi-dev libssl-dev libxm
l2-dev libxslt1-dev libjepg8-dev zlibbig-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package libjepg8-dev
E: Unable to locate package zlibbig-dev
root@ubuntu:~# apt-get install system-pip syscall-dev libffi-dev libssl-dev libx
ml2-dev libxslt1-dev libjepg8-dev zlibbig-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package system-pip
E: Unable to locate package syscall-dev
E: Unable to locate package libjepg8-dev
E: Unable to locate package zlibbig-dev
root@ubuntu:~# apt-get install python-pip python-dev libffi-dev libssl-dev libxm
l2-dev libxslt1-dev libjepg8-dev zlibbig-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package libjepg8-dev
E: Unable to locate package zlibbig-dev
root@ubuntu:~# apt-get install python-pip python-dev libffi-dev libssl-dev libxm


root@ubuntu: ~/hello
11:47 AM
Setting up libsys-hostname-long-perl (1.5-1) ...
Setting up libmail-sendmail-perl (0.79.16-1) ...
Setting up makedumpfile (1:1.5.9-5ubuntu0.5) ...
Setting up dh-strip-nondeterminism (0.015-1) ...
Setting up debhelper (9.20160115ubuntu3) ...
Setting up kernel-wedge (2.90ubuntu1) ...
Processing triggers for systemd (229-4ubuntu21) ...
Processing triggers for ureadahead (0.100.0-19) ...
Processing triggers for libc-bin (2.23-0ubuntu9) ...
root@ubuntu:~# sudo apt-get build-dep linux
Reading package lists... Done
E: You must put some 'source' URIs in your sources.list
root@ubuntu:~# sudo apt-get build-dep linux
Reading package lists... Done
E: You must put some 'source' URIs in your sources.list
root@ubuntu:~# sudo apt-get build-dep --no-install-recommends linux-image-$(uname -r)
Reading package lists... Done
E: You must put some 'source' URIs in your sources.list
root@ubuntu:~#
root@ubuntu:~# sudo apt-get build-dep linux
Reading package lists... Done
E: You must put some 'source' URIs in your sources.list
root@ubuntu:~# mkdir hello
root@ubuntu:~# cd hello
root@ubuntu:~/hello# gedit hello.c
(gedit:43336): IBUS-WARNING **: The owner of /home/pankita/.config/ibus/bus is not root!
(gedit:43336): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.gnome.SessionManager was not provided by any .service files


hello.c (~/hello) - gedit
11:47 AM
Open
hello.c
~/hello
Save
#include<linux/kernel.h>
asmlinkage long sys_hello(void)
{
printk("Hello world\n");
return 0;
}
C
Tab Width: 8
Ln 9, Col 1
INS


Package configuration
Configuring kexec-tools
If you choose this option, a system reboot will trigger a restart into a kernel loaded by kexec instead of going through the full system boot loader process.
Should kexec-tools handle reboots?
<Yes>
<No>

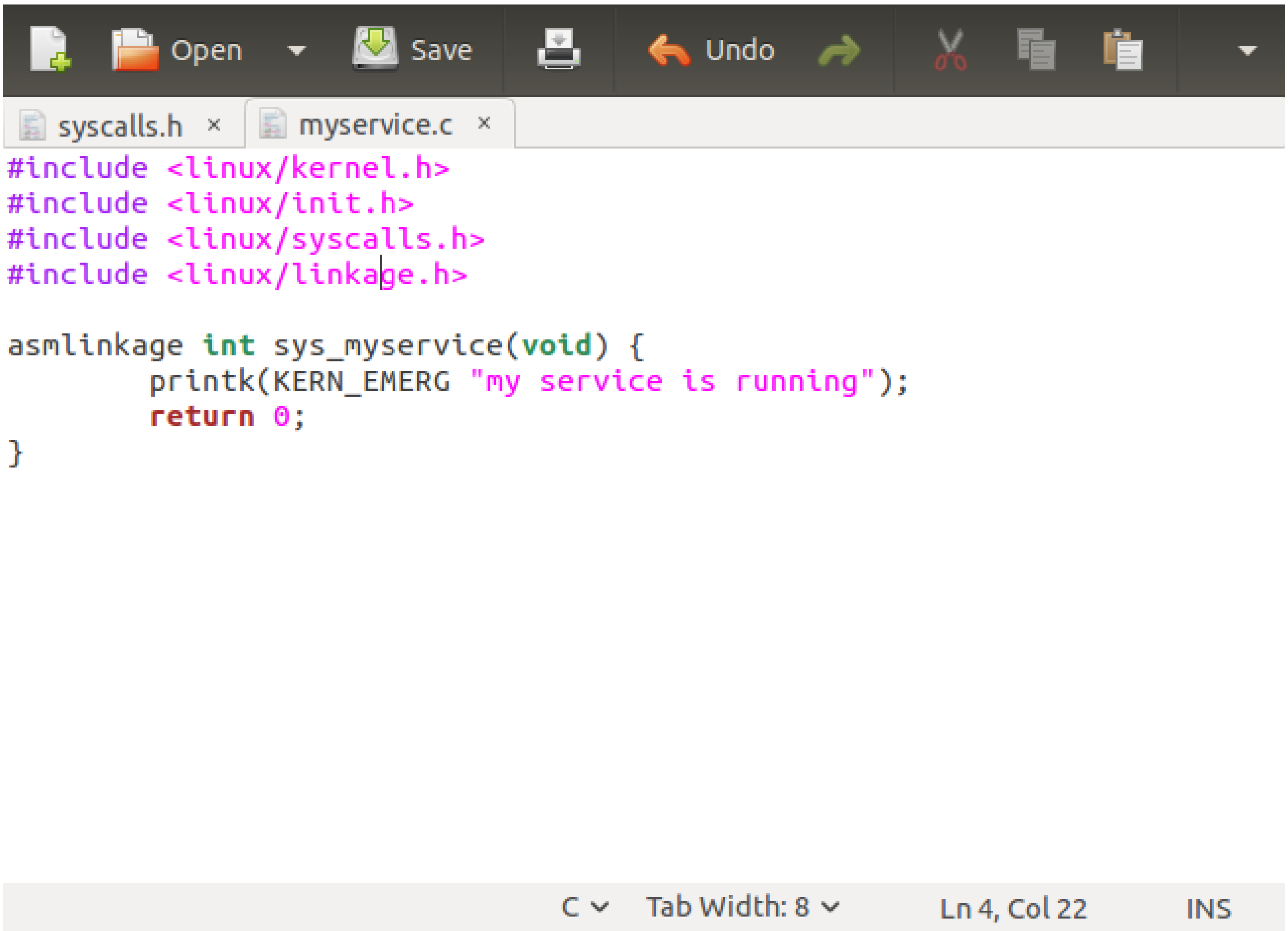

pankita@ubuntu: ~/info
11:06 PM
pankita@ubuntu:~$ sudo tar -xvf linux-4.13.11.tar.xz-C/usr/src/
[sudo] password for pankita:
tar: linux-4.13.11.tar.xz-C/usr/src/: Cannot open: No such file or directory
tar: Error is not recoverable: exiting now
pankita@ubuntu:~$ sudo tar -xvf linux-4.13.11.tar.xz-C/usr/src/
tar: linux-4.13.11.tar.xz-C/usr/src/: Cannot open: No such file or directory
tar: Error is not recoverable: exiting now
pankita@ubuntu:~$ sudo tar -xvf linux-4.13.11.tar.xz-desktop/usr/src/
tar: linux-4.13.11.tar.xz-desktop/usr/src/: Cannot open: No such file or directory
tar: Error is not recoverable: exiting now
pankita@ubuntu:~$ sudo tar -xvf linux-4.13.11.tar.xz-desktop/usr/src/
tar: linux-4.13.11.tar.xz-desktop/usr/src/: Cannot open: No such file or directory
tar: Error is not recoverable: exiting now
pankita@ubuntu:~$ sudo tar -xvf linux-4.13.11.tar.xz-C/usr/src/
tar: linux-4.13.11.tar.xz-C/usr/src/: Cannot open: No such file or directory
tar: Error is not recoverable: exiting now
~$ cd/usr/src/linux-4.13.11
LibreOffice Calc
c/linux-4.13.11: No such file or directory
pankita@ubuntu:~$ cd/usr/src/linux-4.13.11/
bash: cd/usr/src/linux-4.13.11/: No such file or directory
pankita@ubuntu:~$ Desktop/linux-4.13.11 make
bash: Desktop/linux-4.13.11: No such file or directory
pankita@ubuntu:~$ Desktop/linux-4.13.11$ make
bash: Desktop/linux-4.13.11$: No such file or directory
pankita@ubuntu:~$ Desktop/linux-4.13.11.tar.gz$ make
bash: Desktop/linux-4.13.11.tar.gz$: No such file or directory
pankita@ubuntu:~$ Desktop/linux-4.13.11.tar.gz make
bash: Desktop/linux-4.13.11.tar.gz: Permission denied
pankita@ubuntu:~$ /Desktop/linux-4.13.11.tar.gz make
bash: /Desktop/linux-4.13.11.tar.gz: No such file or directory
pankita@ubuntu:~$ -/Desktop/linux-4.13.11.tar.gz make

Open Save Undo

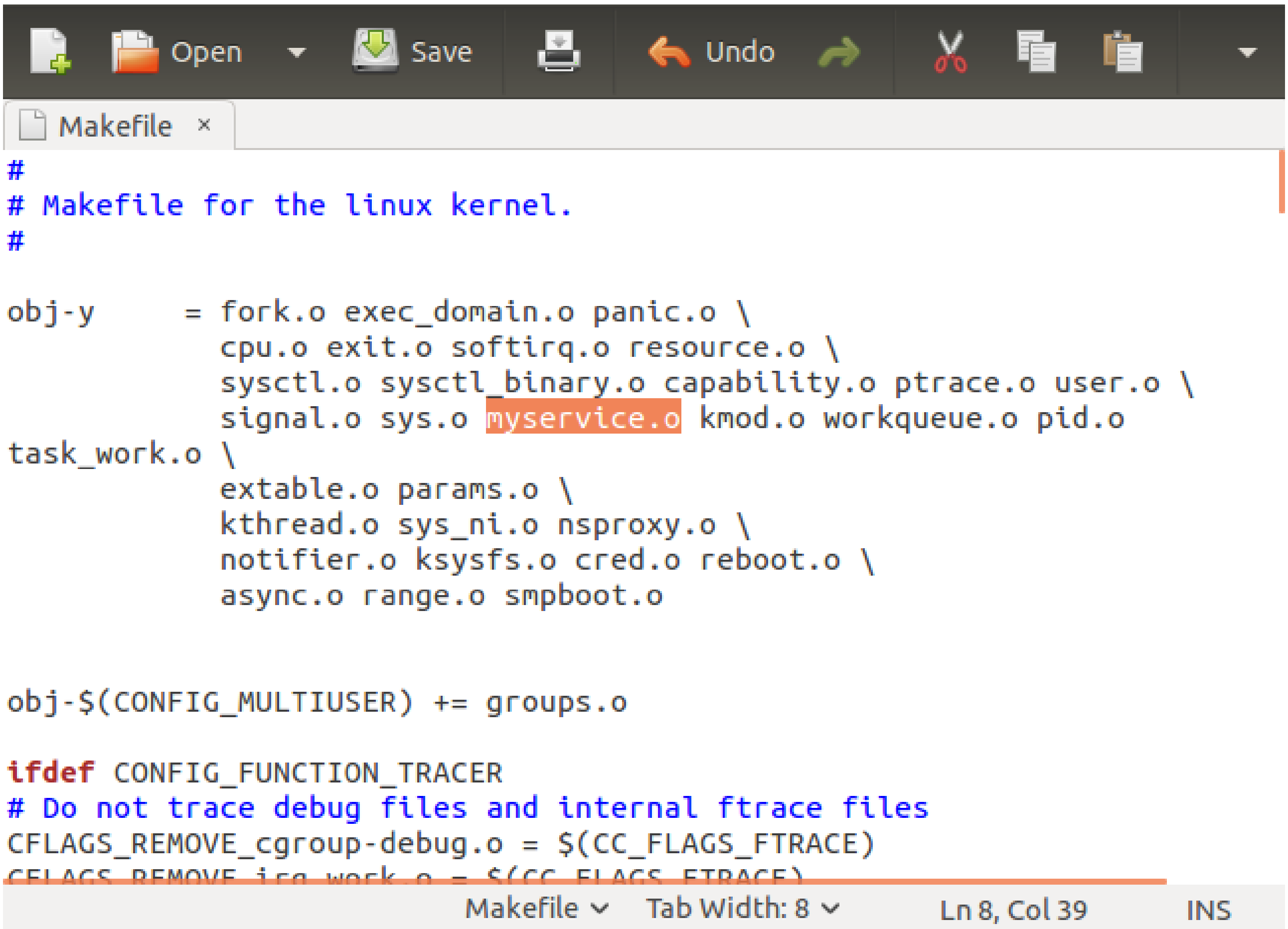
syscalls.h

```c
asmlinkage long sys_name_to_handle_at(int dfd, const char __user *name,
                                      struct file_handle __user *handle,
                                      int __user *mnt_id, int flag);
asmlinkage long sys_open_by_handle_at(int mountdirfd,
                                      struct file_handle __user *handle,
                                      int flags);
asmlinkage long sys_setns(int fd, int nstype);
asmlinkage long sys_process_vm_readv(pid_t pid,
                                     const struct iovec __user *lvec,
                                     unsigned long liovcnt,
                                     const struct iovec __user *rvec,
                                     unsigned long riovcnt,
                                     unsigned long flags);
asmlinkage long sys_process_vm_writev(pid_t pid,
                                      const struct iovec __user *lvec,
                                      unsigned long liovcnt,
                                      const struct iovec __user *rvec,
                                      unsigned long riovcnt,
                                      unsigned long flags);

asmlinkage long sys_kcmp(pid_t pid1, pid_t pid2, int type,
                         unsigned long idx1, unsigned long idx2);
asmlinkage long sys_finit_module(int fd, const char __user *uargs, int flags);
asmlinkage long sys_seccomp(unsigned int op, unsigned int flags,
                            const char __user *uargs);
asmlinkage long sys_getrandom(char __user *buf, size_t count,
                              unsigned int flags);
asmlinkage long sys_bpf(int cmd, union bpf_attr *attr, unsigned int size);

asmlinkage long sys_execveat(int dfd, const char __user *filename,
                        const char __user *const __user *argv,
                        const char __user *const __user *envp, int flags);

asmlinkage int sys_myservice(void);
```

C/C++/ObjC Header   Tab Width: 8   Ln 887, Col 36   INS



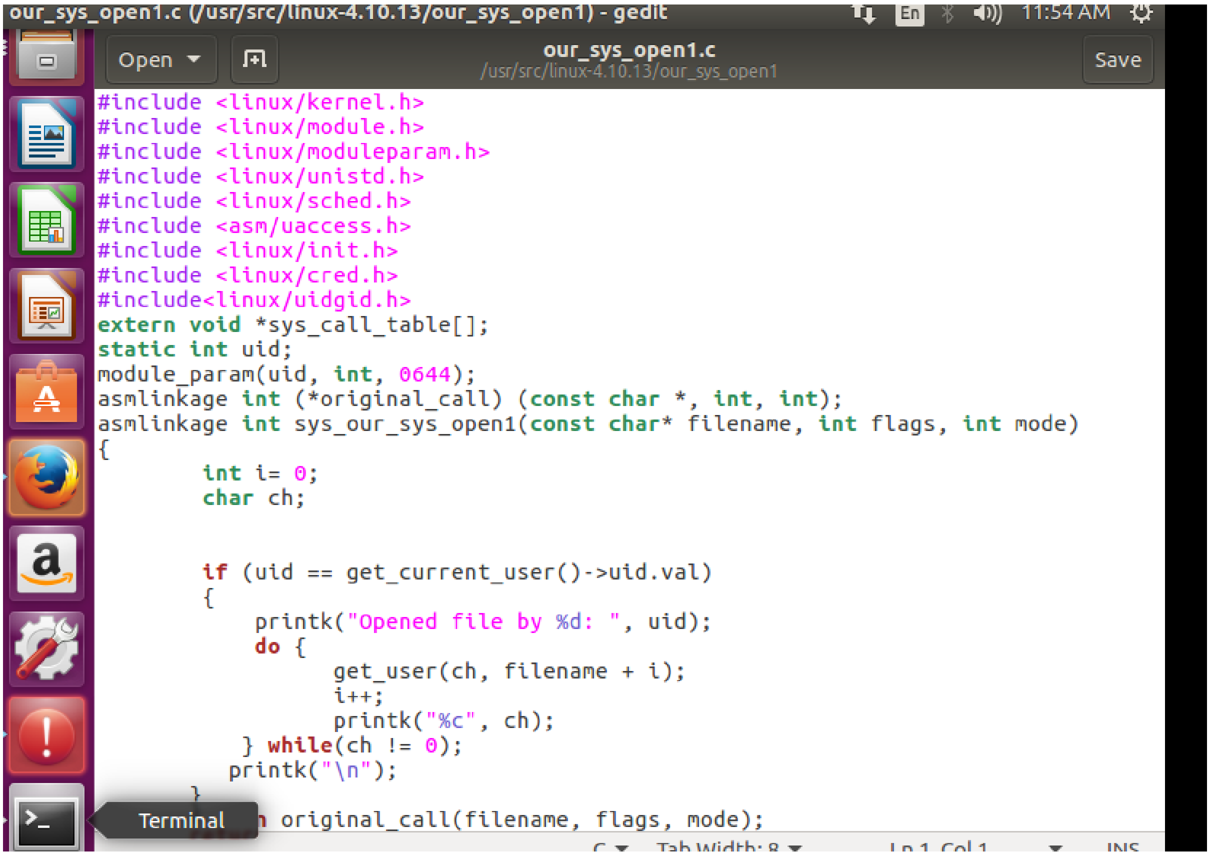
```c
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/syscalls.h>
#include <linux/linkage.h>

asmlinkage int sys_myservice(void) {
        printk(KERN_EMERG "my service is running");
        return 0;
}
```



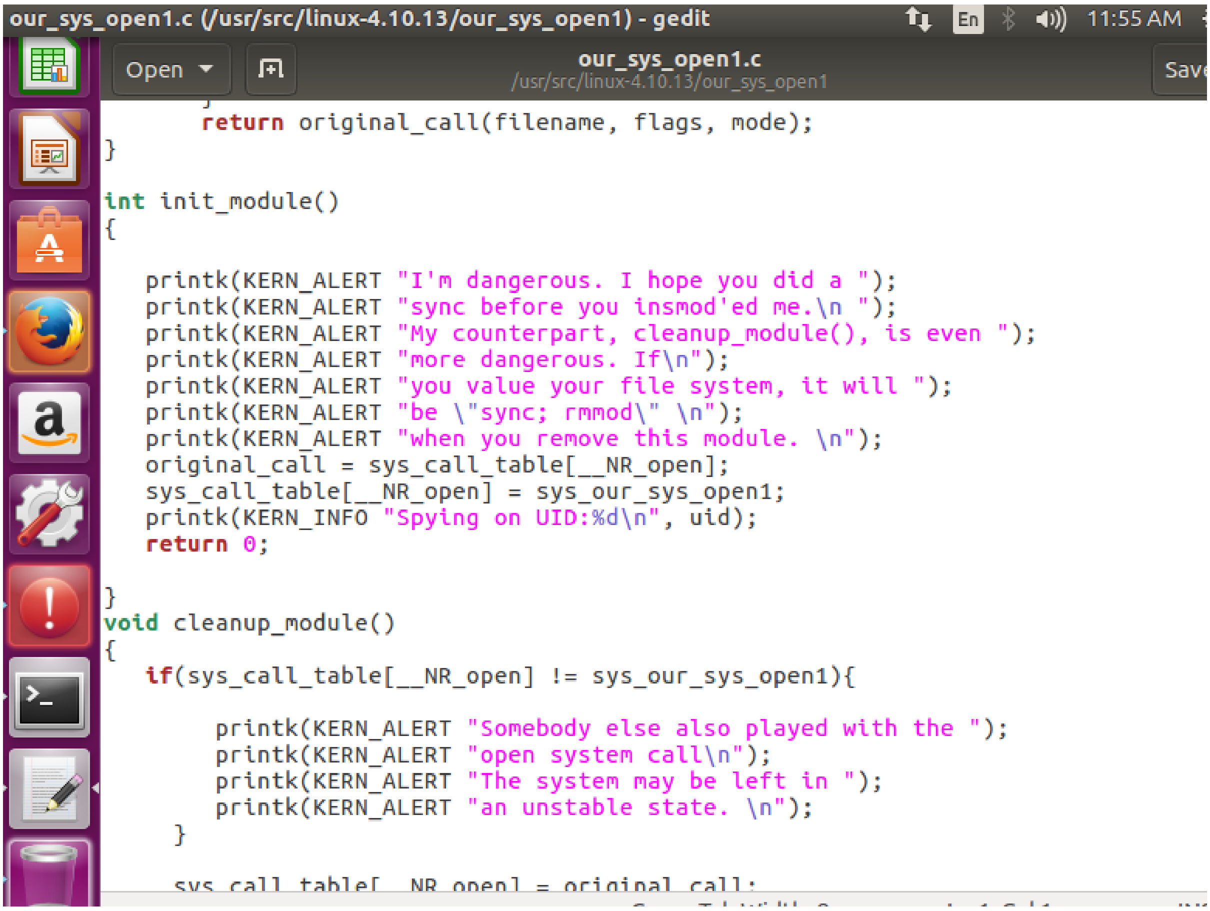
```makefile
#
# Makefile for the linux kernel.
#

obj-y     = fork.o exec_domain.o panic.o \
            cpu.o exit.o softirq.o resource.o \
            sysctl.o sysctl_binary.o capability.o ptrace.o user.o \
            signal.o sys.o myservice.o kmod.o workqueue.o pid.o
task_work.o \
            extable.o params.o \
            kthread.o sys_ni.o nsproxy.o \
            notifier.o ksysfs.o cred.o reboot.o \
            async.o range.o smpboot.o


obj-$(CONFIG_MULTIUSER) += groups.o

ifdef CONFIG_FUNCTION_TRACER
# Do not trace debug files and internal ftrace files
CFLAGS_REMOVE_cgroup-debug.o = $(CC_FLAGS_FTRACE)
CFLAGS_REMOVE_irq_work.o = $(CC_FLAGS_FTRACE)
```

THE SYSTEM CALL WHICH WE ARE TRYING TO ADD

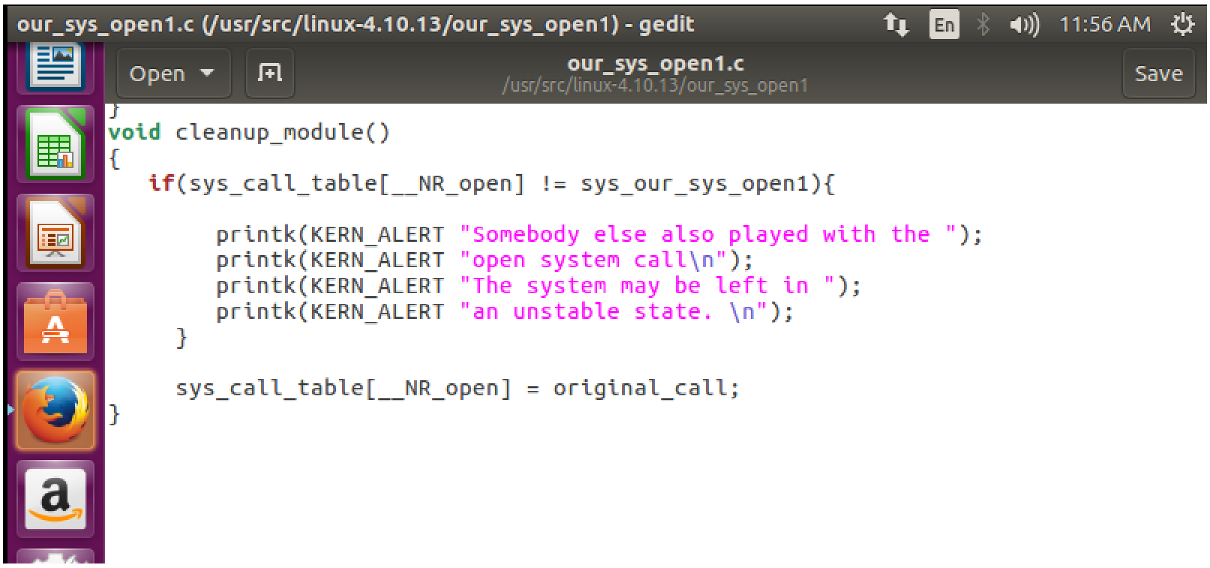
‘




```c
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/unistd.h>
#include <linux/sched.h>
#include <asm/uaccess.h>
#include <linux/init.h>
#include <linux/cred.h>
#include<linux/uidgid.h>
extern void *sys_call_table[];
static int uid;
module_param(uid, int, 0644);
asmlinkage int (*original_call) (const char *, int, int);
asmlinkage int sys_our_sys_open1(const char* filename, int flags, int mode)
{
        int i= 0;
        char ch;


        if (uid == get_current_user()->uid.val)
        {
            printk("Opened file by %d: ", uid);
            do {
                  get_user(ch, filename + i);
                  i++;
                  printk("%c", ch);
           } while(ch != 0);
          printk("\n");
        }
        n original_call(filename, flags, mode);
```




```
        return original_call(filename, flags, mode);
}

int init_module()
{

    printk(KERN_ALERT "I'm dangerous. I hope you did a ");
    printk(KERN_ALERT "sync before you insmod'ed me.\n ");
    printk(KERN_ALERT "My counterpart, cleanup_module(), is even ");
    printk(KERN_ALERT "more dangerous. If\n");
    printk(KERN_ALERT "you value your file system, it will ");
    printk(KERN_ALERT "be \"sync; rmmod\" \n");
    printk(KERN_ALERT "when you remove this module. \n");
    original_call = sys_call_table[__NR_open];
    sys_call_table[__NR_open] = sys_our_sys_open1;
    printk(KERN_INFO "Spying on UID:%d\n", uid);
    return 0;

}
void cleanup_module()
{
    if(sys_call_table[__NR_open] != sys_our_sys_open1){

          printk(KERN_ALERT "Somebody else also played with the ");
          printk(KERN_ALERT "open system call\n");
          printk(KERN_ALERT "The system may be left in ");
          printk(KERN_ALERT "an unstable state. \n");
      }

      sys_call_table[__NR_open] = original_call;
```




```
void cleanup_module()
{
    if(sys_call_table[__NR_open] != sys_our_sys_open1){

          printk(KERN_ALERT "Somebody else also played with the ");
          printk(KERN_ALERT "open system call\n");
          printk(KERN_ALERT "The system may be left in ");
          printk(KERN_ALERT "an unstable state. \n");
      }

      sys_call_table[__NR_open] = original_call;
}
```

Conclusion

Our project on Operating Systems was to add a System call to the Linux operating system. In our system call, we take a string as an argument and print it on the execution of cat command.

We were successfully able to execute our system call and this project led to a deeper understanding about the linux operating system. We got to know about how a new system call is associated to the original source code and we also learned about all the changes that are needed to add a system call.

References

Internet-

- https://www.youtube.com/watch?v=5rr_VoQCOgE&feature=youtu.be
- https://www.youtube.com/watch?v=vZaA2mRT5eg&feature=youtu.be
- https://www.youtube.com/watch?v=1Rido46nao8
- http://www.csee.umbc.edu/courses/undergraduate/CMSC421/fall02/burt/projects/howto_build_kernel.html
- http://www.csee.umbc.edu/courses/undergraduate/CMSC421/fall02/burt/projects/howto_add_systemcall.html
- http://franksthinktank.com/howto/addsyscall/

Book-

- Operating System Concepts, Silberschatz, Galvin, Gagne