Assignment 2: Adding and testing a new system call to the Linux kernel

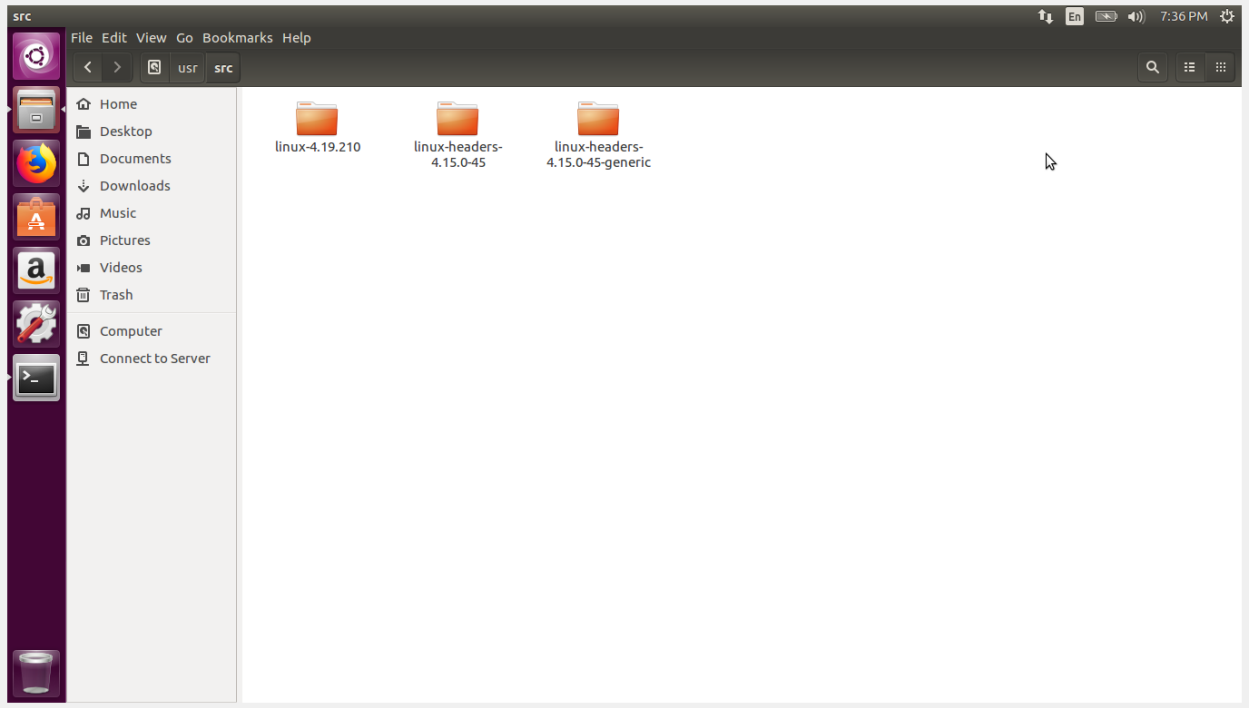
> **Name:** Bhanuj Gandhi
> **Roll no.:** 2022201068

In this assignment, our task was to implement the system calls in the Linux kernel and test them out by compiling the kernel.

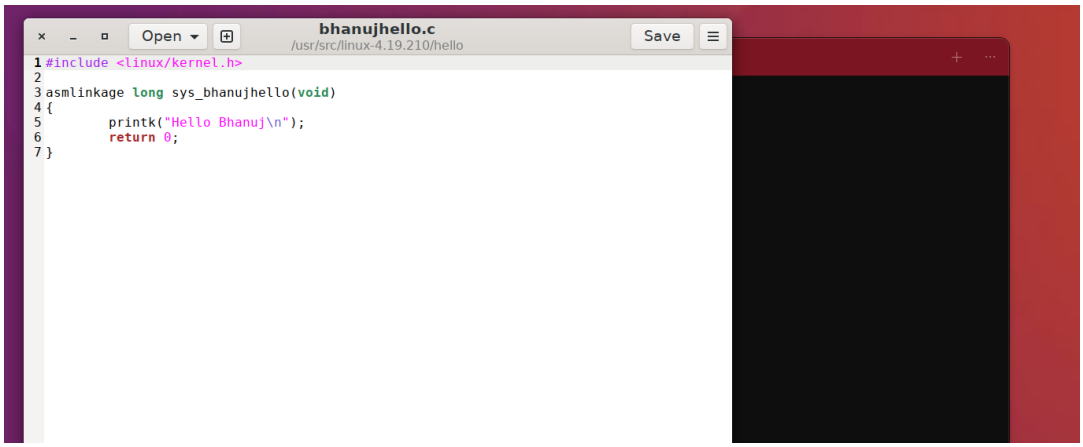
Steps Followed

1. Download the compressed file of *Linux Kernel version 4.19.210*.
2. Extract the downloaded kernel file in */usr/src*.

```
$ sudo tar -xvf linux-4.19.210.tar.xz -C/usr/src
```



1. Go to the extracted folder
2. Implement System Calls
    a. **Question 1. : Write syscall to print a welcome message to Linux logs**
        i. Create a Folder for a new system-call `hello`
        ii. Create a C file inside it. `hello.c`
        iii. Implement the system call as shown in `bhanujhello.c`



iv. Create a make file in the `hello` folder named `Makefile` and include the following line:

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

v. Add function call definition in `./include/syscall.h` header file.

```
asmlinkage long sys_bhanujhello(void);
```

vi. Add the system call entry in `syscalls_32.tbl`

> 💡 The format for syscalls_32.tbl is:
>
> `<number> <abi> <name> <entry point> <compat entry point>`
>
> *(As described in syscalls_32.tbl file itself)*
>
> For 32-bit intel systems, abi is i386
>
> `387 i386 hello sys_bhanujhello`

vii. Add folder name to `Makefile` of kernel

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

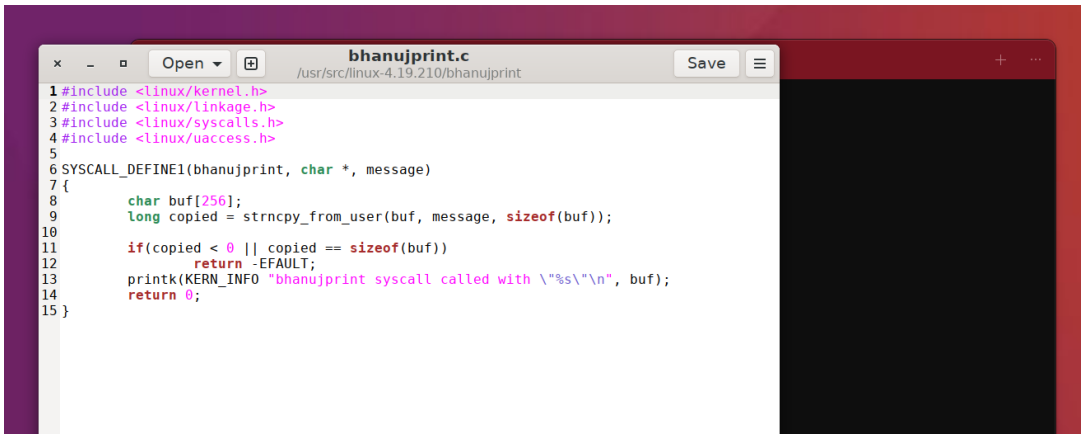
> 💡 *Step iv*: During the kernel recompilation, this Makefile describes the items that must be built and added to the source code.
>
> *Step vii:* This will inform the compiler that we also need to build the files in the directory hello/.
>
> Therefore, the process would go as follows: after running the root makefile, which now lists the hello/ directory, it will look for another makefile in that directory for additional guidance on which objects to build.

b. **Question 2.: Write syscall which will receive string parameter and print it along with some message to kernel logs**

i. Create a Folder for a new system-call `bhanujprint`

ii. Create a C file inside it. `bhanujprint.c`

iii. Implement the system call as shown in `bhanujprint.c`



iv. Create a make file in the `bhanujprint` folder named *`Makefile`* and include the following line:

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

v. Add the system call entry in *`syscalls_32.tbl`*

vi. Add folder name to *`Makefile`* of kernel

```
core-y += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ hello/ bhanujprint/
```

c. **Question 3.: Write system call to print the parent process id and current process id upon calling it**

i. Create a Folder for a new system-call `bhanujprocess`

ii. Create a C file inside it. `bhanujprocess.c`

iii. Implement the system call as shown in `bhanujprocess.c`

iv. Create a make file in the `bhanujprocess` folder named `Makefile` and include the following line:

`obj-y := bhanujprocess.o`

v. Add function call definition in `./include/syscall.h` header file.

`asmlinkage long sys_bhanujprocess(void);`

vi. Add the system call entry in *`syscalls_32.tbl`*

> 💡 The format for syscalls_32.tbl is:
>
> `<number> <abi> <name> <entry point> <compat entry point>`
>
> *(As described in syscalls_32.tbl file itself)*
>
> For 32-bit intel systems, abi is i386
>
> `387 i386 hello sys_bhanujprint _ia32_sys_bhanujprint`
>
> where ia32_sys_#syscallname is a wrapper to a function that helps the system call to parse the arguments passed. As in this system call, we need to pass an argument we have to specify the `compat entry point` also.

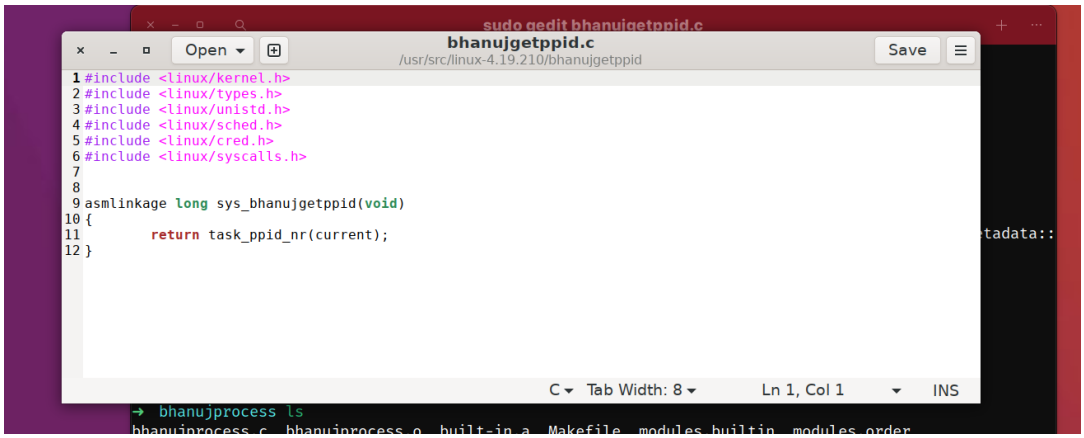
vii. Add folder name to `Makefile` of kernel

`core-y += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ hello/ bhanujprint/ bhanujprocess/`

d. **Question 4: Write system call to execute some predefined system call from your written system call**

In this question, I have implemented `getppid()` function call which is named as `bhanujgetppid()` which will return parent process' process id(pid).

1. Create a Folder for a new system-call *`bhanujgetppid`*
2. Create a C file inside it. *`bhanujgetppid.c`*
3. Implement the system call as shown in `bhanujgetppid.c`



4. Create a make file in the `bhanujgetppid` folder named `Makefile` and include the following line:

   `obj-y := bhanujgetppid.o`

5. Add function call definition in `./include/syscall.h` header file.

   `asmlinkage long sys_bhanujgetppid(void);`

6. Add the system call entry in *`syscalls_32.tbl`*

7. Add folder name to `Makefile` of kernel

   `core-y += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ hello/ bhanujprint/ bhanujprocess/ bhanujgetppid/`

Final `syscalls_32.tbl` file will look like:

Last 4 entries have my implemented system calls for question 1, question 2, question 3, question 4, respectively

syscall_32.tbl
/usr/src/linux-4.19.210/arch/x86/entry/syscalls

```
354 340     i386    prlimit64               sys_prlimit64               __ia32_sys_prlimit64
355 341     i386    name_to_handle_at       sys_name_to_handle_at       __ia32_sys_name_to_handle_at
356 342     i386    open_by_handle_at       sys_open_by_handle_at       __ia32_compat_sys_open_by_handle_at
357 343     i386    clock_adjtime           sys_clock_adjtime           __ia32_compat_sys_clock_adjtime
358 344     i386    syncfs                  sys_syncfs                  __ia32_sys_syncfs
359 345     i386    sendmmsg                sys_sendmmsg                __ia32_compat_sys_sendmmsg
360 346     i386    setns                   sys_setns                   __ia32_sys_setns
361 347     i386    process_vm_readv        sys_process_vm_readv        __ia32_compat_sys_process_vm_readv
362 348     i386    process_vm_writev       sys_process_vm_writev       __ia32_compat_sys_process_vm_writev
363 349     i386    kcmp                    sys_kcmp                    __ia32_sys_kcmp
364 350     i386    finit_module            sys_finit_module            __ia32_sys_finit_module
365 351     i386    sched_setattr           sys_sched_setattr           __ia32_sys_sched_setattr
366 352     i386    sched_getattr           sys_sched_getattr           __ia32_sys_sched_getattr
367 353     i386    renameat2               sys_renameat2               __ia32_sys_renameat2
368 354     i386    seccomp                 sys_seccomp                 __ia32_sys_seccomp
369 355     i386    getrandom               sys_getrandom               __ia32_sys_getrandom
370 356     i386    memfd_create            sys_memfd_create            __ia32_sys_memfd_create
371 357     i386    bpf                     sys_bpf                     __ia32_sys_bpf
372 358     i386    execveat                sys_execveat                __ia32_compat_sys_execveat
373 359     i386    socket                  sys_socket                  __ia32_sys_socket
374 360     i386    socketpair              sys_socketpair              __ia32_sys_socketpair
375 361     i386    bind                    sys_bind                    __ia32_sys_bind
376 362     i386    connect                 sys_connect                 __ia32_sys_connect
377 363     i386    listen                  sys_listen                  __ia32_sys_listen
378 364     i386    accept4                 sys_accept4                 __ia32_sys_accept4
379 365     i386    getsockopt              sys_getsockopt              __ia32_compat_sys_getsockopt
380 366     i386    setsockopt              sys_setsockopt              __ia32_compat_sys_setsockopt
381 367     i386    getsockname             sys_getsockname             __ia32_sys_getsockname
382 368     i386    getpeername             sys_getpeername             __ia32_sys_getpeername
383 369     i386    sendto                  sys_sendto                  __ia32_sys_sendto
384 370     i386    sendmsg                 sys_sendmsg                 __ia32_compat_sys_sendmsg
385 371     i386    recvfrom                sys_recvfrom                __ia32_compat_sys_recvfrom
386 372     i386    recvmsg                 sys_recvmsg                 __ia32_compat_sys_recvmsg
387 373     i386    shutdown                sys_shutdown                __ia32_sys_shutdown
388 374     i386    userfaultfd             sys_userfaultfd             __ia32_sys_userfaultfd
389 375     i386    membarrier              sys_membarrier              __ia32_sys_membarrier
390 376     i386    mlock2                  sys_mlock2                  __ia32_sys_mlock2
391 377     i386    copy_file_range         sys_copy_file_range         __ia32_sys_copy_file_range
392 378     i386    preadv2                 sys_preadv2                 __ia32_compat_sys_preadv2
393 379     i386    pwritev2                sys_pwritev2                __ia32_compat_sys_pwritev2
394 380     i386    pkey_mprotect           sys_pkey_mprotect           __ia32_sys_pkey_mprotect
395 381     i386    pkey_alloc              sys_pkey_alloc              __ia32_sys_pkey_alloc
396 382     i386    pkey_free               sys_pkey_free               __ia32_sys_pkey_free
397 383     i386    statx                   sys_statx                   __ia32_sys_statx
398 384     i386    arch_prctl              sys_arch_prctl              __ia32_compat_sys_arch_prctl
399 385     i386    io_pgetevents           sys_io_pgetevents           __ia32_compat_sys_io_pgetevents
400 386     i386    rseq                    sys_rseq                    __ia32_sys_rseq
401 387     i386    hello                   sys_bhanujhello
402 388     i386    bhanujprint             sys_bhanujprint             __ia32_sys_bhanujprint
403 389     i386    bhanujprocess           sys_bhanujprocess
404 390     i386    bhanujgetppid           sys_bhanujgetppid
```

syscalls_32.tbl file snapshot

Final `Makefile` in root directory of kernel will look like:

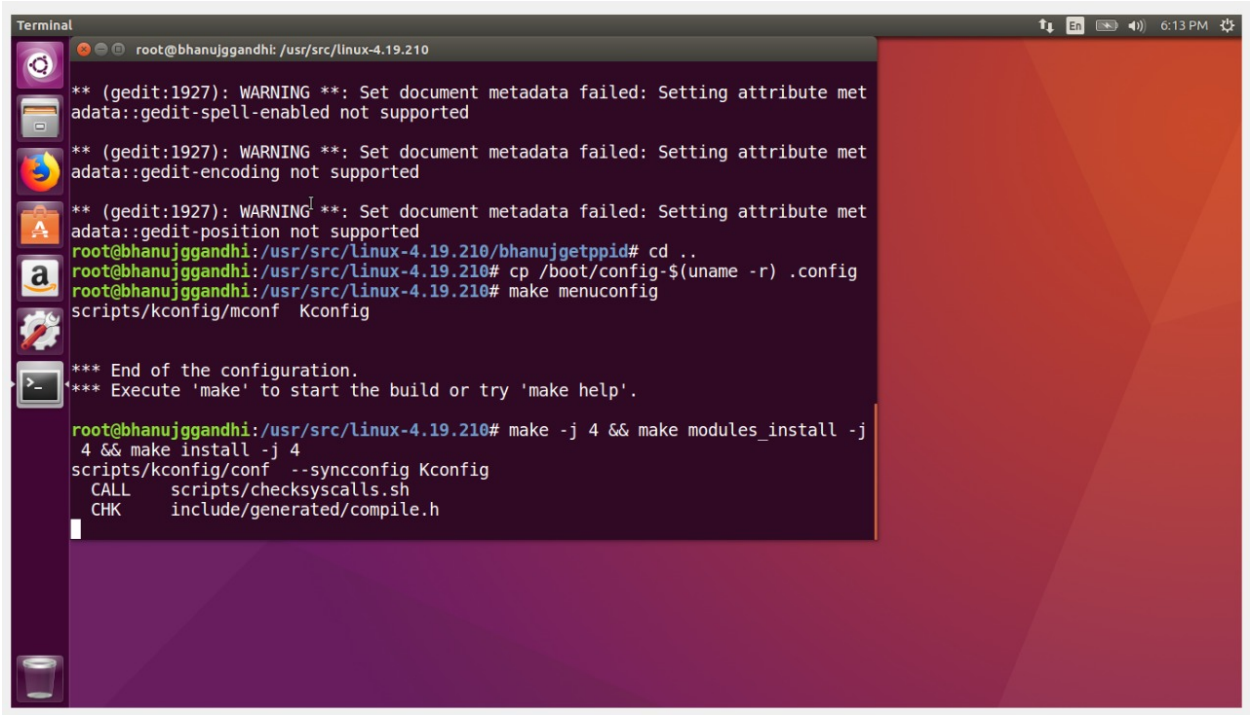
Makefile
/usr/src/linux-4.19.210

```
975             echo "int main() {}" | $(HOSTCC) -xc -o /dev/null $(HOST_LIBELF_LIBS) -,1,0)
976   ifeq ($(has_libelf),1)
977     objtool_target := tools/objtool FORCE
978   else
979     SKIP_STACK_VALIDATION := 1
980     export SKIP_STACK_VALIDATION
981   endif
982 endif
983
984 PHONY += prepare0
985
986 ifeq ($(KBUILD_EXTMOD),)
987 core-y          += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ hello/ bhanujprint/ bhanujprocess/ bhanujgetppid/
988
989 vmlinux-dirs    := $(patsubst %/,%,$(filter %/, $(init-y) $(init-m) \
990                      $(core-y) $(core-m) $(drivers-y) $(drivers-m) \
991                      $(net-y) $(net-m) $(libs-y) $(libs-m) $(virt-y)))
992
993 vmlinux-alldirs := $(sort $(vmlinux-dirs) $(patsubst %/,%,$(filter %/, \
994                      $(init-) $(core-) $(drivers-) $(net-) $(libs-) $(virt-))))
995
996 init-y          := $(patsubst %/, %/built-in.a, $(init-y))
997 core-y          := $(patsubst %/, %/built-in.a, $(core-y))
998 drivers-y       := $(patsubst %/, %/built-in.a, $(drivers-y))
999 net-y           := $(patsubst %/, %/built-in.a, $(net-y))
1000 libs-y1         := $(patsubst %/, %/lib.a, $(libs-y))
1001 libs-y2         := $(patsubst %/, %/built-in.a, $(filter-out %.a, $(libs-y)))
```

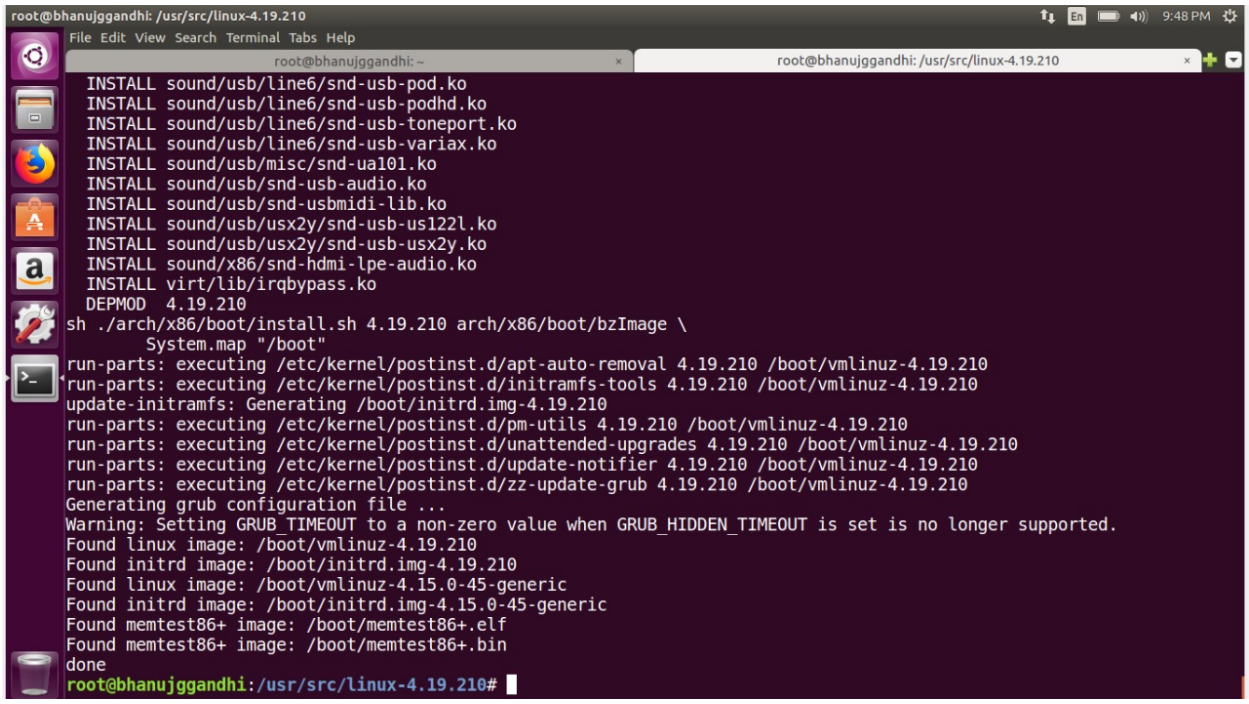
Makefile snapshot

3. Compile the Kernel

```
# Makefile for menuconfig
$ sudo make menuconfig

# Compiling kernel and kernel modules
$ sudo make -j 4 && sudo make modules_install -j 4

# Installing kernel
$ sudo make install -j 4

# Where -j n defines the number of cores to be given to the process
# that will run this command



# This is to update the kernel entries in the grub
$ sudo update-grub
$ sudo shutdown now -r
```



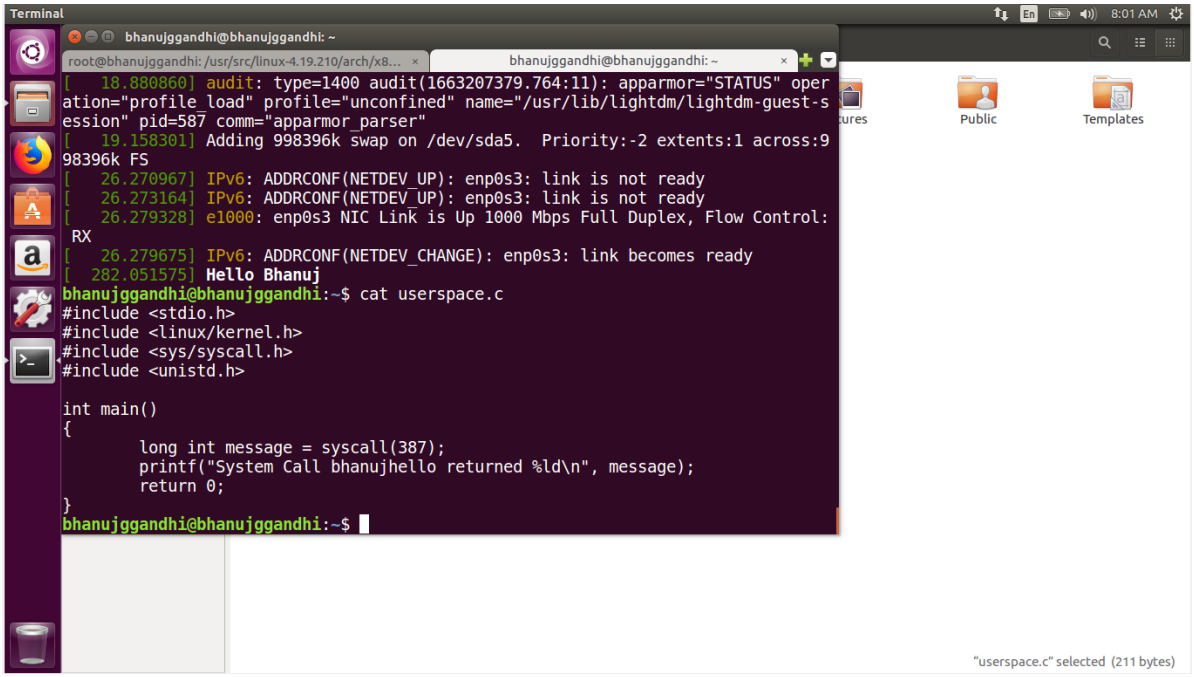
Execution snapshot of *menuconfig and make command*



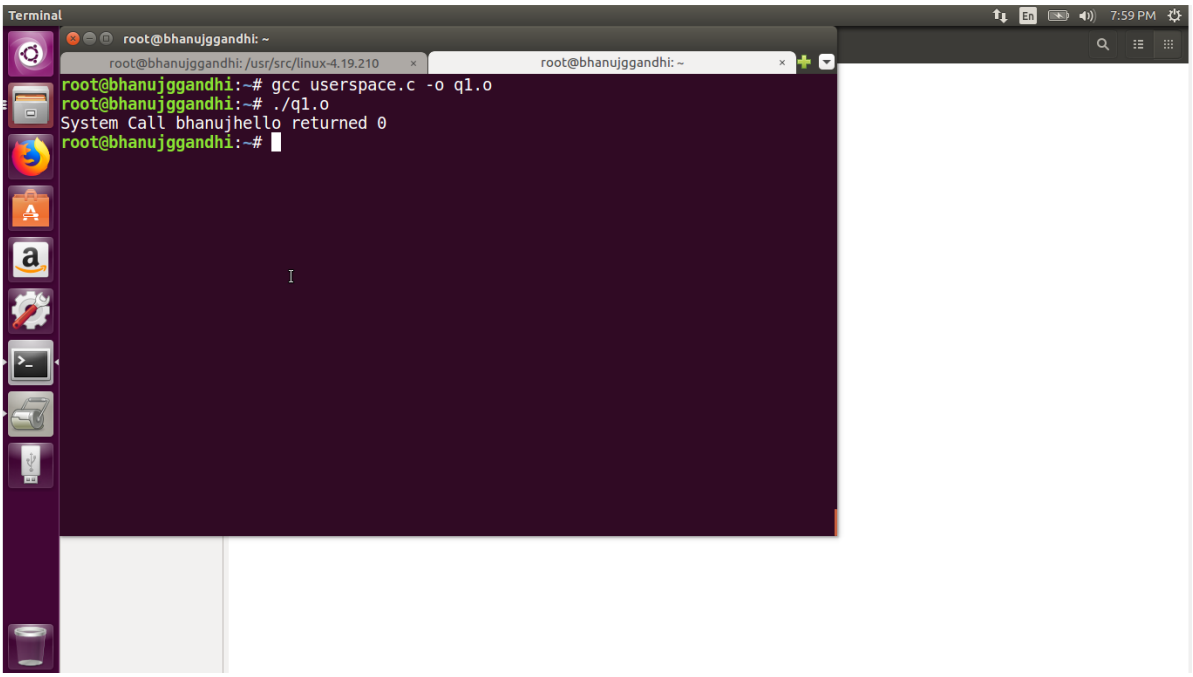
Completion of make install command

4. Test the system calls

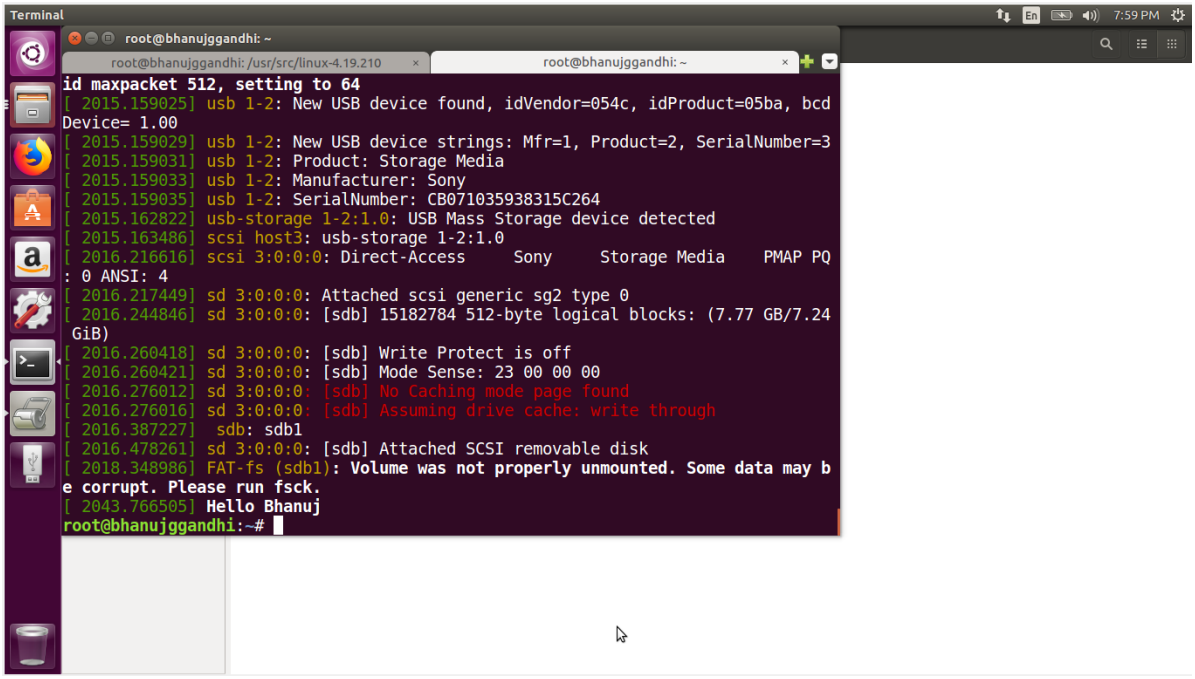
   a. **Question 1:**



Driver Code to Test question 1 sys_bhanujhello system call



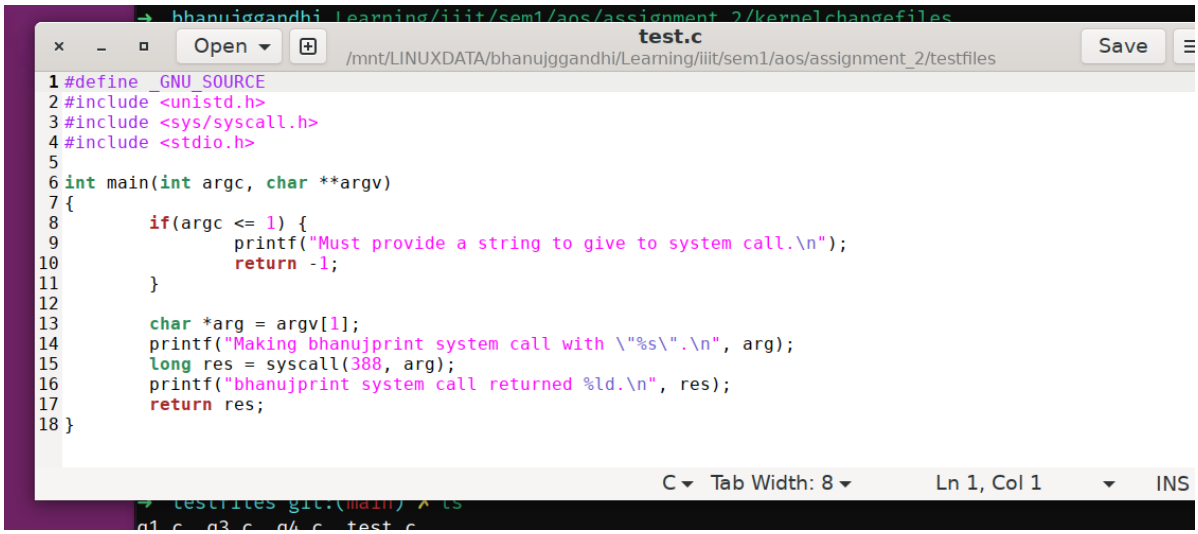
Upon running the out file after compiling



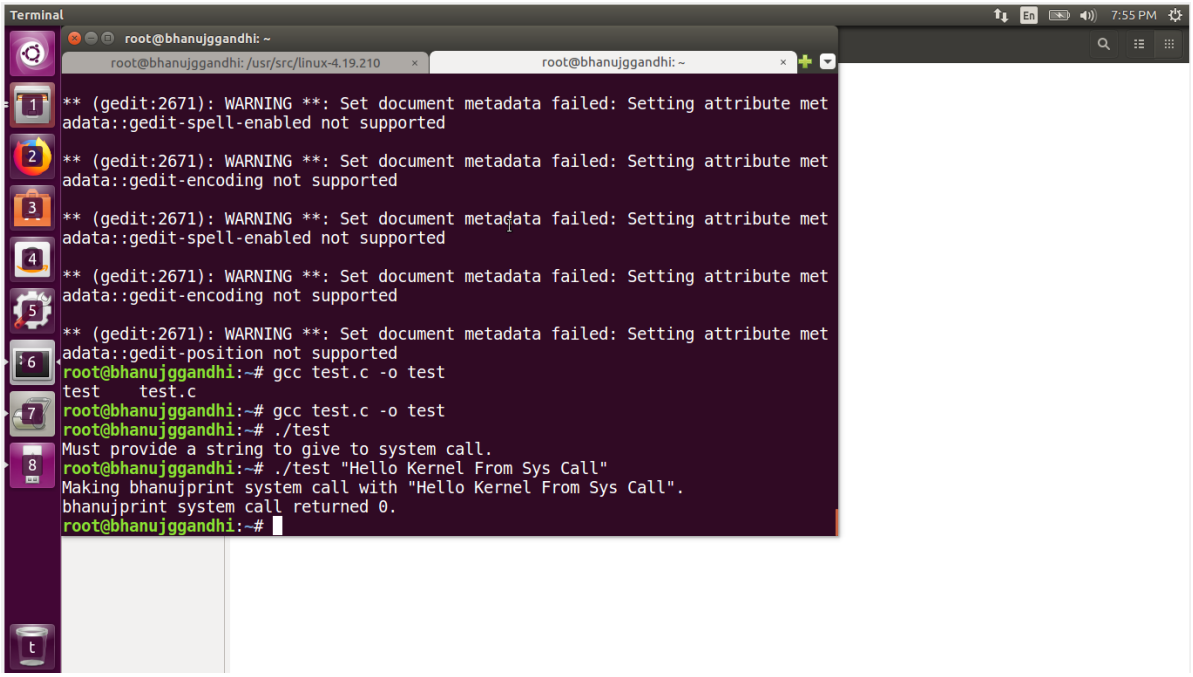
Output in dmesg by sys_bhanujhello system call

---

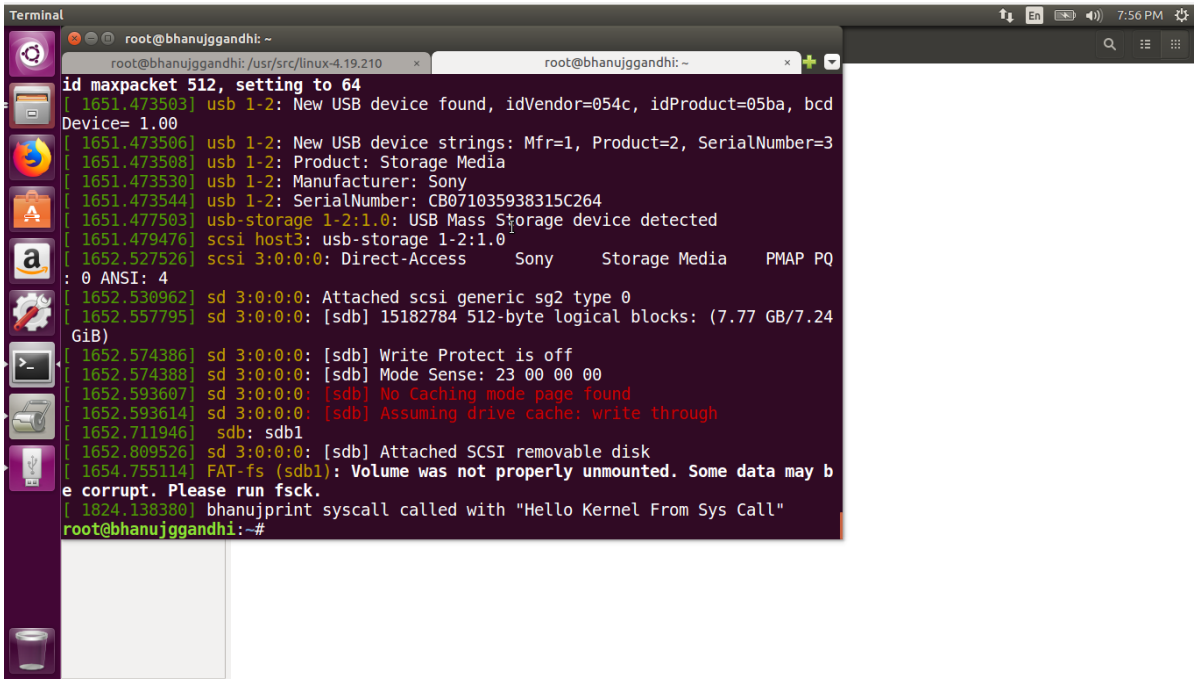
b. **Question 2:**



Driver code for bhanujprint system call

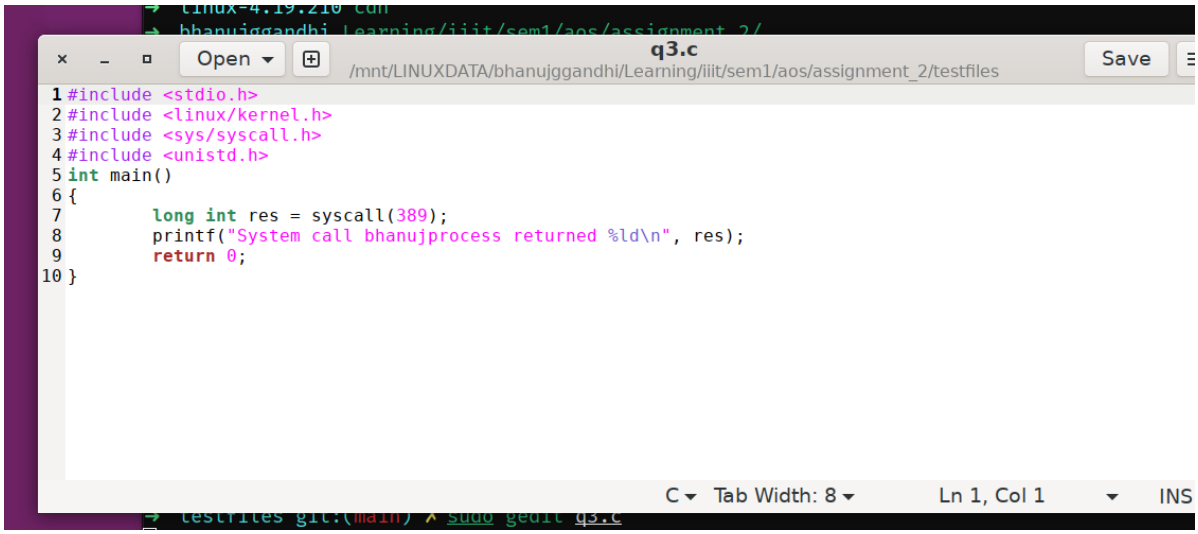


Upon compiling and running .out file

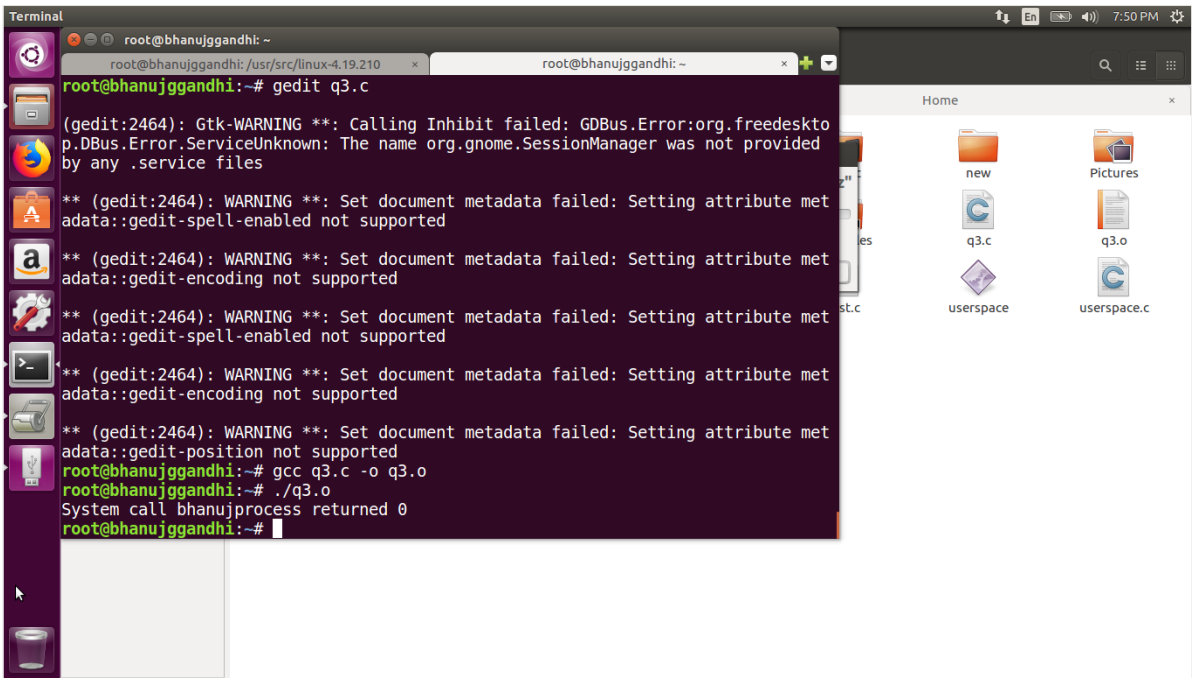


Output in dmesg by sys_bhanujprint system call

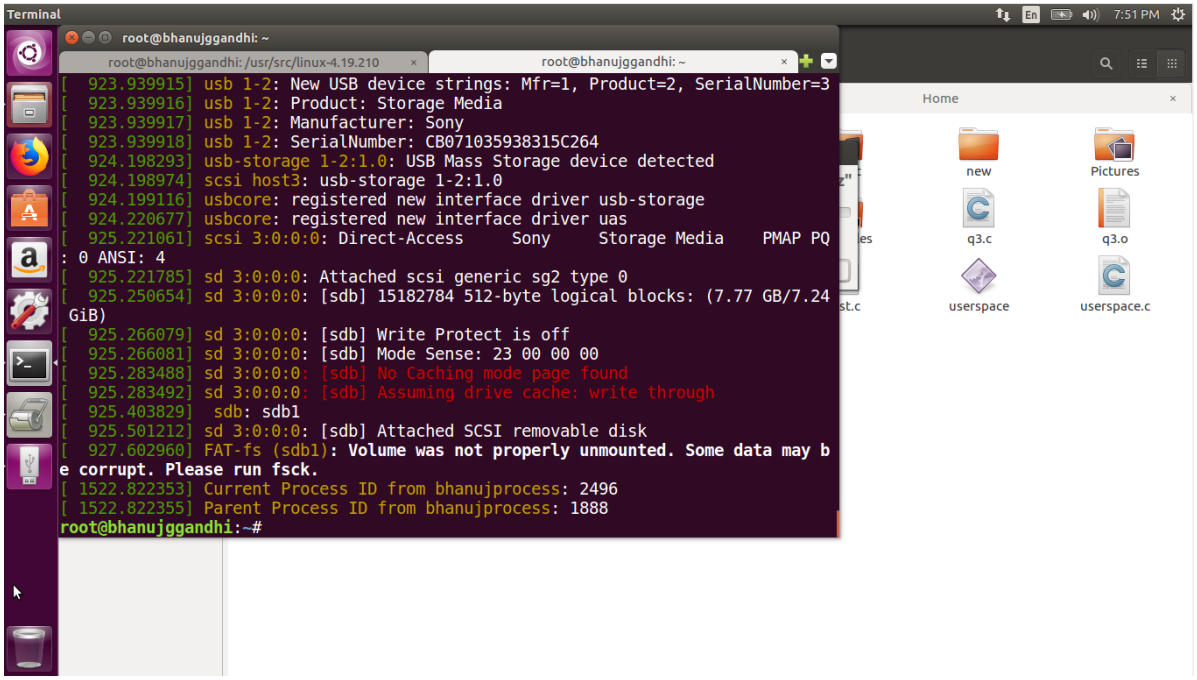
c. **Question 3:**



Driver code for sys_bhanujprocess system call



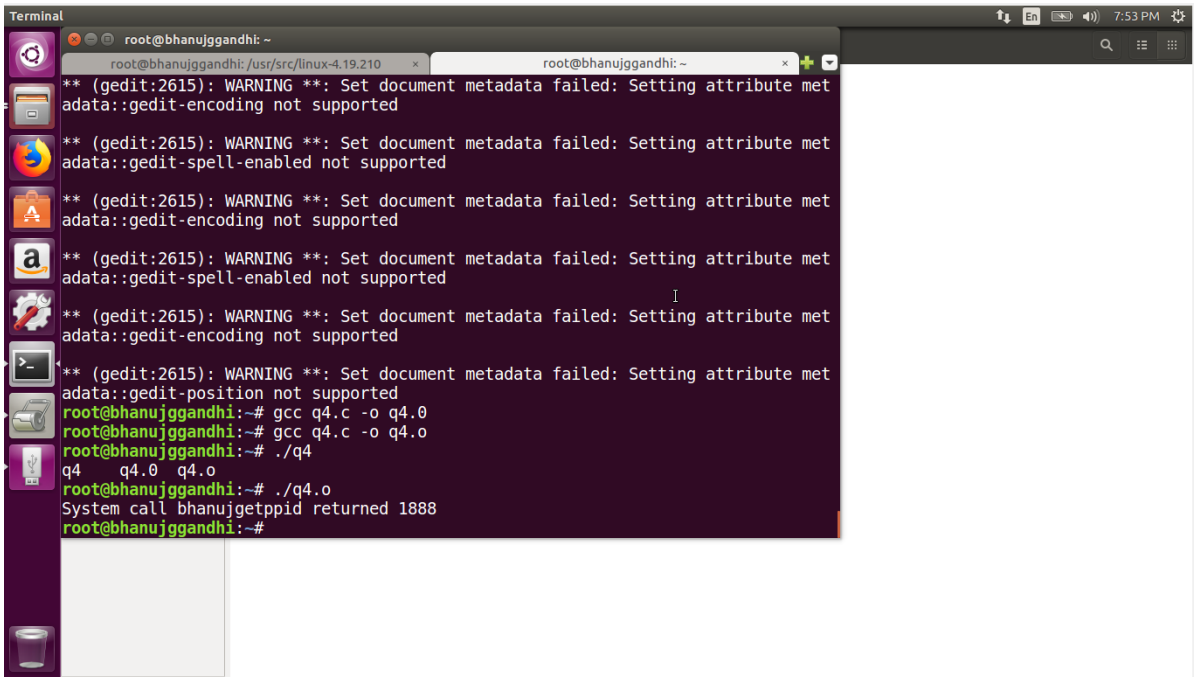
Upon compiling and running .out file for question 3 system call



Output in dmesg by sys_bhanujprocess system call

d. **Question 4:**

Driver code for sys_bhanujgetppid system call



Upon compiling and running the out file, the process outputs the PID of the parent of the *./q4.o* process.

> **Q) Are both process IDs the same or different? Why? What are your observations?**

> No current process ID and parent process ID are not the same, as they both are different. Shell we are in right now will execute a process by executing the instance of *a.out* binary. The process of *a.out* binary will be the child process of the current shell process. Parent ID will be the process ID of the shell and current process ID will be the instance of *a.out* process ID.
>
> The process ID is a unique ID, thus it won't correspond to any running processes.
>
> The child process inherits some attributes like *Time Limit, Region Size*, *current working directory*, etc.