Advance Operating System

Assignment 2 Report

By: Shlok Bansal (2021201046) Mtech CSE

Introduction:

In this assignment, we have to create 4 system calls for different purposes for linux Kernel **v4.19.210**

In the following report I have attached all the steps that I have followed to achieve this objective

Downloading And Extracting the Kernel

Downloading The kernel

```
wget https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.19.210.tar.xz
```

The above wget command is used to download linux kernel 4.19.210 from kernel.org website using terminal.

Extracting the Kernel

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

- The tar command is used to extract the tar file which we have downloaded with the help of the above command.
- -x : extract files from an archive.
- -v : requested using the –verbose option, when extracting archives.
- -f : file archive; use archive file or device archive
- -C : extract to the directory specified after it.

The above command will extract the linux-4.19.210.tar.xz file in /usr/src directory. All the kernels are present inside this /usr/src directory. Sudo is written as sudo privileges are required to make any change in this directory.

Note: To avoid writing sudo after every command when we need to make change we are going to go to root via **sudo –s** command. `sudo -s`

Question 1) Write syscall to print welcome message to Linux logs.

Step 1) Go to the kernel directory where we want to make a new system call

```
cd /usr/src/linux-4.19.210/
```

With this we change our and work in the kernel directory.

Step 2) Make the directory in which you will write the source code of the system call and go into that directory.

```
mkdir shlokhello
```

```
cd shlokhello
```

Step 3) Create shlokhello.c in this directory.

```
gedit shlokhello.c
```

- **asmlinkage** is a keyword which is used to indicate that all parameters of the function would be available of stack.
- **void** indicates function that shlokhello() does not take any parameters.
- **printk** is a function like printf but it is used to print in kernel log.

Step 4) Create Makefile

```
gedit Makefile
```

This is to ensure that shlokhello.c is compiled when the kernel is compiled.

Step 5) Edit the Makefile of Kernel

```
cd /usr/src/linux-4.19.210/
```

```
nano Makefile
```

Look for the line which looks like below

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

In this line add shlokhello/

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

- By adding shlokhello/ in this line we are telling the compiler of the kernel that our new system call is defined in shlokhello directory.
- It will tell the compiler to make the Makefile in this directory which was mentioned above.

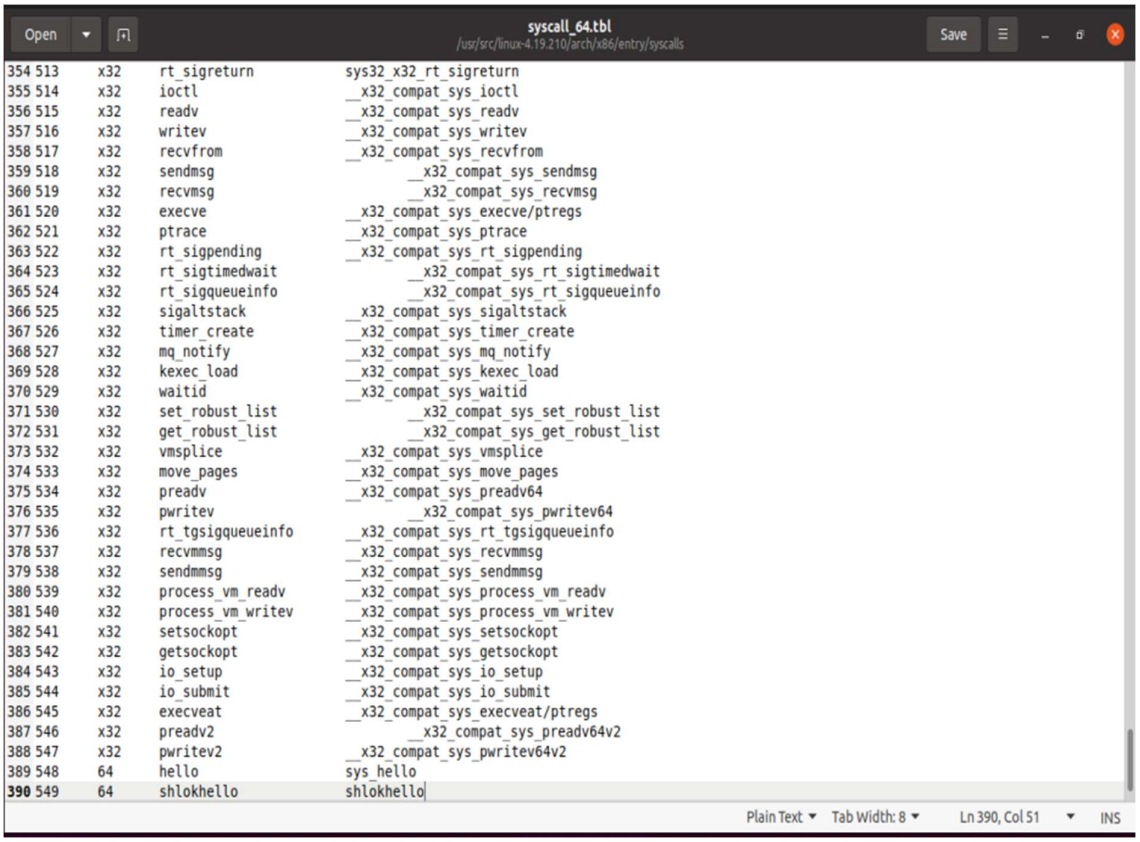
Step 6) Add the new system call to the system call table of kernel

```
cd arch/x86/entry/syscalls/
```

The system call table of the kernel lies in above directory. Change your current directory with the above command.

As my system is 64 bit so **add entry in syscall_64.tbl .**

```
gedit syscall_64.tbl
```

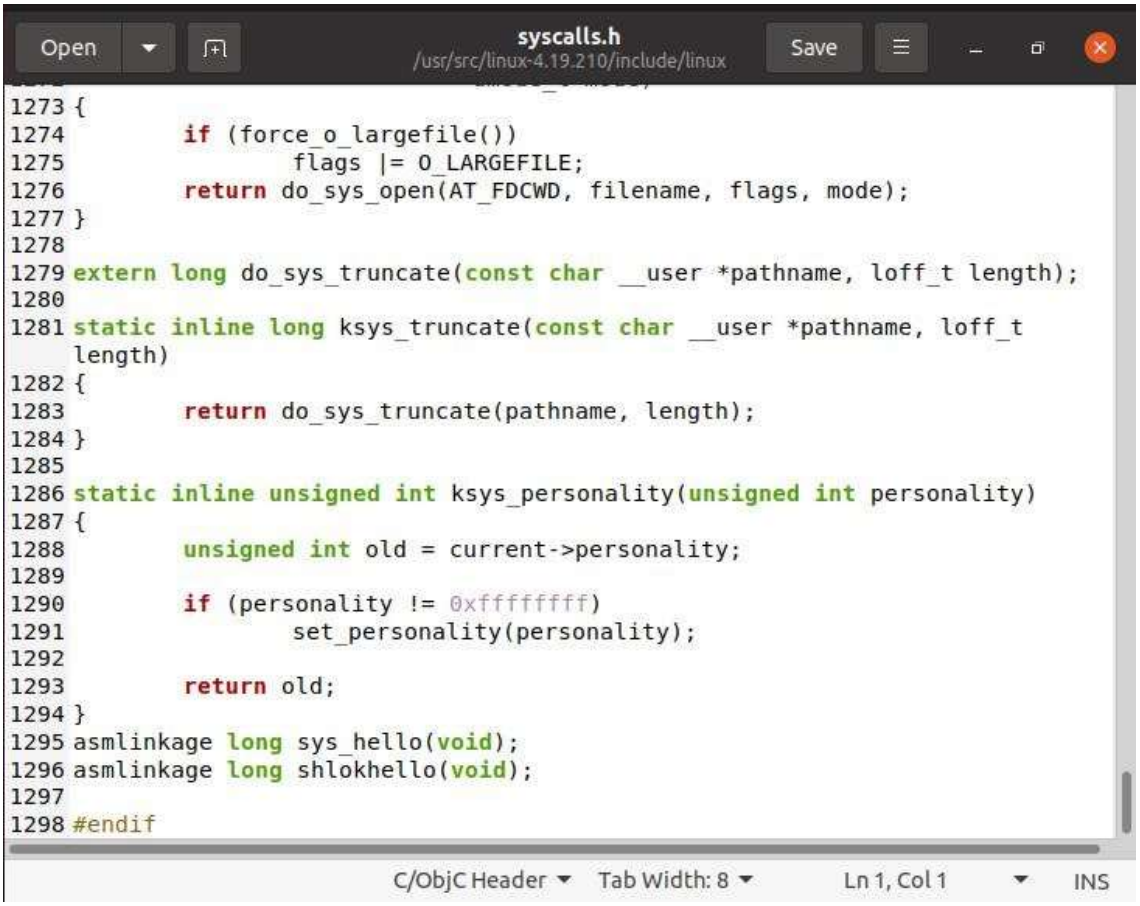


In kernel all system calls are identified by a unique number.

Here we assigned a unique number to our own system function call i.e., **549.**

Step 7) Add new system call to the system call header file

```
cd /usr/src/linux-4.19.210/
cd include/linux/
gedit syscalls.h
```




```
1273 {
1274         if (force_o_largefile())
1275                 flags |= O_LARGEFILE;
1276         return do_sys_open(AT_FDCWD, filename, flags, mode);
1277 }
1278
1279 extern long do_sys_truncate(const char __user *pathname, loff_t length);
1280
1281 static inline long ksys_truncate(const char __user *pathname, loff_t
     length)
1282 {
1283         return do_sys_truncate(pathname, length);
1284 }
1285
1286 static inline unsigned int ksys_personality(unsigned int personality)
1287 {
1288         unsigned int old = current->personality;
1289
1290         if (personality != 0xffffffff)
1291                 set_personality(personality);
1292
1293         return old;
1294 }
1295 asmlinkage long sys_hello(void);
1296 asmlinkage long shlokhello(void);
1297
1298 #endif
```

This defines the prototype of the function of our system call.

Step 8) Compile the Kernel

```
cd /usr/src/linux-4.19.210/
make menuconfig
```

This command is used to configure the Linux kernel. A pop up will come just check that ext4 in file system is included.

```
make -j4
```

This command will start compiling the kernel. Now j4 in **make –j4** makes use of 4 cores in system to make this file. The more cores we use the faster the compilation will take place. -jn tells number of cores to be used and n cannot be greater than number of cores in your machine

Step 9) Update or Install the kernel

```
make modules_install install
```

The above command will make changes in the kernel according to the files which we have compiled above, and the changes will be updated in the 4 files in **/boot** directory: -

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

Now reboot the system to see all the changes in kernel.

Step 10) Test the system call

Code:

Open Q1.c Save

```
#include<stdio.h>
#include<linux/kernel.h>
#include<sys/syscall.h>
#include<unistd.h>
int main()
{
        long int no=8;
        no=syscall(549);
        printf("Ans is %d:",no);
        return 0;
}
```

Output:

```
shlok@shlok:~$ ./q1
Ans is 0:shlok@shlok:~$ dmesg
```

shlok@shlok: ~

```
[    6.589184] systemd[1]: Condition check resulted in Rebuild Hardware Database
 being skipped.
[    6.589338] systemd[1]: Condition check resulted in Platform Persistent Stora
ge Archival being skipped.
[    6.591693] systemd[1]: Starting Load/Save Random Seed...
[    6.597328] systemd[1]: Starting Create System Users...
[    6.599947] systemd[1]: Finished Uncomplicated firewall.
[    6.755779] systemd[1]: Started Journal Service.
[    6.835840] systemd-journald[1173]: Received client request to flush runtime
journal.
[    6.892784] Adding 703976k swap on /swapfile.  Priority:-2 extents:3 across:7
20360k
[   11.934040] ethtool (1248) used greatest stack depth: 13456 bytes left
[   15.968600] e2scrub_all (1342) used greatest stack depth: 13248 bytes left
[   24.783637] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[   24.789655] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 RX
[   24.791592] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[   24.791609] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[   70.412982] rfkill: input handler disabled
[ 2757.974504] kworker/dying (319) used greatest stack depth: 13032 bytes left
[10752.982342] kworker/dying (2990) used greatest stack depth: 12752 bytes left
[11735.885059] Welcome to Linux shlok
```

Question 2) Write syscall which will receive string parameter and print it along with some message to kernel logs

Step 1) Go to the kernel directory where we want to make a new system call

```
cd /usr/src/linux-4.19.210/
```

With this we change our and work in the kernel directory.

Step 2) Make the directory in which you will write the source code of the system call and go into that directory.

```
mkdir shlokprint
```

```
cd shlokprint
```

Step 3) Create shlokprint.c in this directory.

```
gedit shlokprint.c
```

- **SYSCALL_DEFINE1** is used to define function with parameters. Here **1** means the function has only 1 parameter. The first parameter is the function name and it makes function name as **sys_shlokprint(char *msg).**
- The parameters are defines as pairs in SYSCALL_DEFINE(). Now (char* ,msg) is same as char *msg in normal ways.
- When data is passed as a pointer then pointer address is send relative to user space. So when we directly try to access that address in kernel space it gives us error.
- To avoid this error we are taking help of **strncpy_from_user** looks for message pointer in user space and copy the content to the buffer variable of the kernel space .
- **val** tells us how much data is copied in the buffer. If val<0 then there is error in copy and when val=sizeof(buffer) then there is overflow so we have sent EFAULT when this occurs
- **printk** is a function like printf but it is used to print in kernel log.

Step 4) Create Makefile

```
gedit Makefile
```

This is to ensure that shlokprint.c is compiled when the kernel is compiled.

Step 5) Edit the Makefile of Kernel

```
cd /usr/src/linux-4.19.210/
nano Makefile
```

Look for the line which looks like below

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

In this line add shlokprint/

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

- By adding shlokprint/ in this line we are telling the compiler of the kernel that our new system call is defined in shlokprint directory.
- It will tell the compiler to make the Makefile in this directory which was mentioned above.

Step 6) Add the new system call to the system call table of kernel

```
cd arch/x86/entry/syscalls/
```

The system call table of the kernel lies in above directory. Change your current directory with the above command.

As my system is 64 bit so **add entry in syscall_64.tbl .**

```
gedit syscall_64.tbl
```

Open ▾ | syscall_64.tbl — /usr/src/linux-4.19.210/arch/x86/entry/syscalls | Save

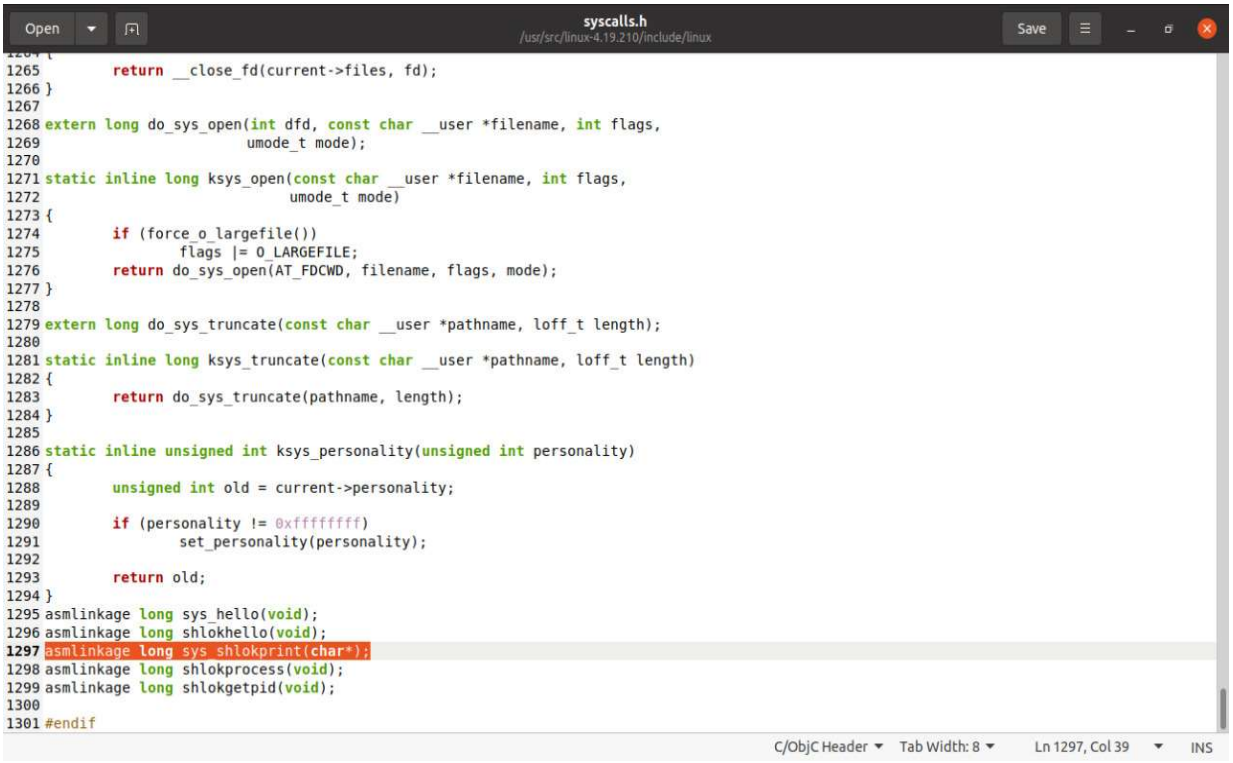
```
355 514     x32     ioctl                   __x32_compat_sys_ioctl
356 515     x32     readv                   __x32_compat_sys_readv
357 516     x32     writev                  __x32_compat_sys_writev
358 517     x32     recvfrom                __x32_compat_sys_recvfrom
359 518     x32     sendmsg                         __x32_compat_sys_sendmsg
360 519     x32     recvmsg                         __x32_compat_sys_recvmsg
361 520     x32     execve                  __x32_compat_sys_execve/ptregs
362 521     x32     ptrace                  __x32_compat_sys_ptrace
363 522     x32     rt_sigpending           __x32_compat_sys_rt_sigpending
364 523     x32     rt_sigtimedwait                 __x32_compat_sys_rt_sigtimedwait
365 524     x32     rt_sigqueueinfo                 __x32_compat_sys_rt_sigqueueinfo
366 525     x32     sigaltstack             __x32_compat_sys_sigaltstack
367 526     x32     timer_create            __x32_compat_sys_timer_create
368 527     x32     mq_notify               __x32_compat_sys_mq_notify
369 528     x32     kexec_load              __x32_compat_sys_kexec_load
370 529     x32     waitid                  __x32_compat_sys_waitid
371 530     x32     set_robust_list                 __x32_compat_sys_set_robust_list
372 531     x32     get_robust_list                 __x32_compat_sys_get_robust_list
373 532     x32     vmsplice                __x32_compat_sys_vmsplice
374 533     x32     move_pages              __x32_compat_sys_move_pages
375 534     x32     preadv                  __x32_compat_sys_preadv64
376 535     x32     pwritev                         __x32_compat_sys_pwritev64
377 536     x32     rt_tgsigqueueinfo       __x32_compat_sys_rt_tgsigqueueinfo
378 537     x32     recvmmsg                __x32_compat_sys_recvmmsg
379 538     x32     sendmmsg                __x32_compat_sys_sendmmsg
380 539     x32     process_vm_readv        __x32_compat_sys_process_vm_readv
381 540     x32     process_vm_writev       __x32_compat_sys_process_vm_writev
382 541     x32     setsockopt              __x32_compat_sys_setsockopt
383 542     x32     getsockopt              __x32_compat_sys_getsockopt
384 543     x32     io_setup                __x32_compat_sys_io_setup
385 544     x32     io_submit               __x32_compat_sys_io_submit
386 545     x32     execveat                __x32_compat_sys_execveat/ptregs
387 546     x32     preadv2                         __x32_compat_sys_preadv64v2
388 547     x32     pwritev2                __x32_compat_sys_pwritev64v2
389 548     64      hello                   sys_hello
390 549     64      shlokhello              shlokhello
391 550     64      shlokprint              __x64_sys_shlokprint
```

In kernel all system calls are identified by a unique number.

Here we assigned a unique number to our own system function call i.e., **550.**

Step 7) Add new system call to the system call header file

```
cd /usr/src/linux-4.19.210/
cd include/linux/
gedit syscalls.h
```




```
1265        return __close_fd(current->files, fd);
1266 }
1267
1268 extern long do_sys_open(int dfd, const char __user *filename, int flags,
1269                         umode_t mode);
1270
1271 static inline long ksys_open(const char __user *filename, int flags,
1272                              umode_t mode)
1273 {
1274         if (force_o_largefile())
1275                 flags |= O_LARGEFILE;
1276         return do_sys_open(AT_FDCWD, filename, flags, mode);
1277 }
1278
1279 extern long do_sys_truncate(const char __user *pathname, loff_t length);
1280
1281 static inline long ksys_truncate(const char __user *pathname, loff_t length)
1282 {
1283         return do_sys_truncate(pathname, length);
1284 }
1285
1286 static inline unsigned int ksys_personality(unsigned int personality)
1287 {
1288         unsigned int old = current->personality;
1289
1290         if (personality != 0xffffffff)
1291                 set_personality(personality);
1292
1293         return old;
1294 }
1295 asmlinkage long sys_hello(void);
1296 asmlinkage long shlokhello(void);
1297 asmlinkage long sys_shlokprint(char*);
1298 asmlinkage long shlokprocess(void);
1299 asmlinkage long shlokgetpid(void);
1300
1301 #endif
```

This defines the prototype of the function of our system call.

Step 8) Compile the Kernel

```
cd /usr/src/linux-4.19.210/
make menuconfig
```

This command is used to configure the Linux kernel. A pop up will come just check that ext4 in file system is included.

```
make -j4
```

This command will start compiling the kernel. Now j4 in **make –j4** makes use of 4 cores in system to make this file. The more cores we use the faster the compilation will take place. -jn tells number of cores to be used and n cannot be greater than number of cores in your machine

Step 9) Update or Install the kernel

```
make modules_install install
```

The above command will make changes in the kernel according to the files which we have compiled above, and the changes will be updated in the 4 files in **/boot** directory: -

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

Now reboot the system to see all the changes in kernel.

Step 10) Test the system call

Code:

Q2.c

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/syscall.h>


int main(int argc, char **argv)
{
  char str[20] = "Omkar";
  // Make the greet_o system call
  int ret_val = syscall(550, str);
  if (ret_val < 0)
  {
    printf("[ERR] Ret val: %d\n", ret_val);
    return 1;
  }
  printf("greet_o system call message: %d\n", ret_val);
  return 0;
}
```

Output:

```
shlok@shlok:~$ gcc Q2.c -o q2
shlok@shlok:~$ ./q2
greet_o system call message: 0
```

```
shlok@shlok:~$ gcc Q2.c -o q2
shlok@shlok:~$ ./q2
greet_o system call message: 0
 RX
[    24.791592] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[    24.791609] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[    70.412982] rfkill: input handler disabled
[ 2757.974504] kworker/dying (319) used greatest stack depth: 13032 bytes left
[10752.982342] kworker/dying (2990) used greatest stack depth: 12752 bytes left
[11735.885059] Welcome to Linux shlok
[11758.476060] Welcome to Linux shlok
[11863.018553] Omkar
```

Question 3) Write system call to print the parent process id and current process id upon calling it

Step 1) Go to the kernel directory where we want to make a new system call

```
cd /usr/src/linux-4.19.210/
```

With this we change our and work in the kernel directory.

Step 2) Make the directory in which you will write the source code of the system call and go into that directory.

```
mkdir shlokprocess
```

```
cd shlokprocess
```

Step 3) Create shlokprocess.c in this directory.

```
gedit shlokprocess.c
```

- **asmlinkage** is a keyword which is used to indicate that all parameters of the function would be available of stack.
- **void** indicates function that shlokprocess() does not take any parameters.
- **printk** is a function like printf but it is used to print in kernel log.
- **current** is global pointer variable which of **struct task_struct type**. It points to the current process' data and with the help of pointer we are easily accessing the process' pid which is available in it's tast_struct data and return its value.
- **task_ppid_nr** helps us to get parent id of the pointer which we pass to it.

Step 4) Create Makefile

```
gedit Makefile
```

This is to ensure that shlokprocess.c is compiled when the kernel is compiled.

Step 5) Edit the Makefile of Kernel

```
cd /usr/src/linux-4.19.210/
```

```
nano Makefile
```

Look for the line which looks like below

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

In this line add shlokprocess/

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

- By adding shlokprocess/ in this line we are telling the compiler of the kernel that our new system call is defined in shlokprocess directory.
- It will tell the compiler to make the Makefile in this directory which was mentioned above.

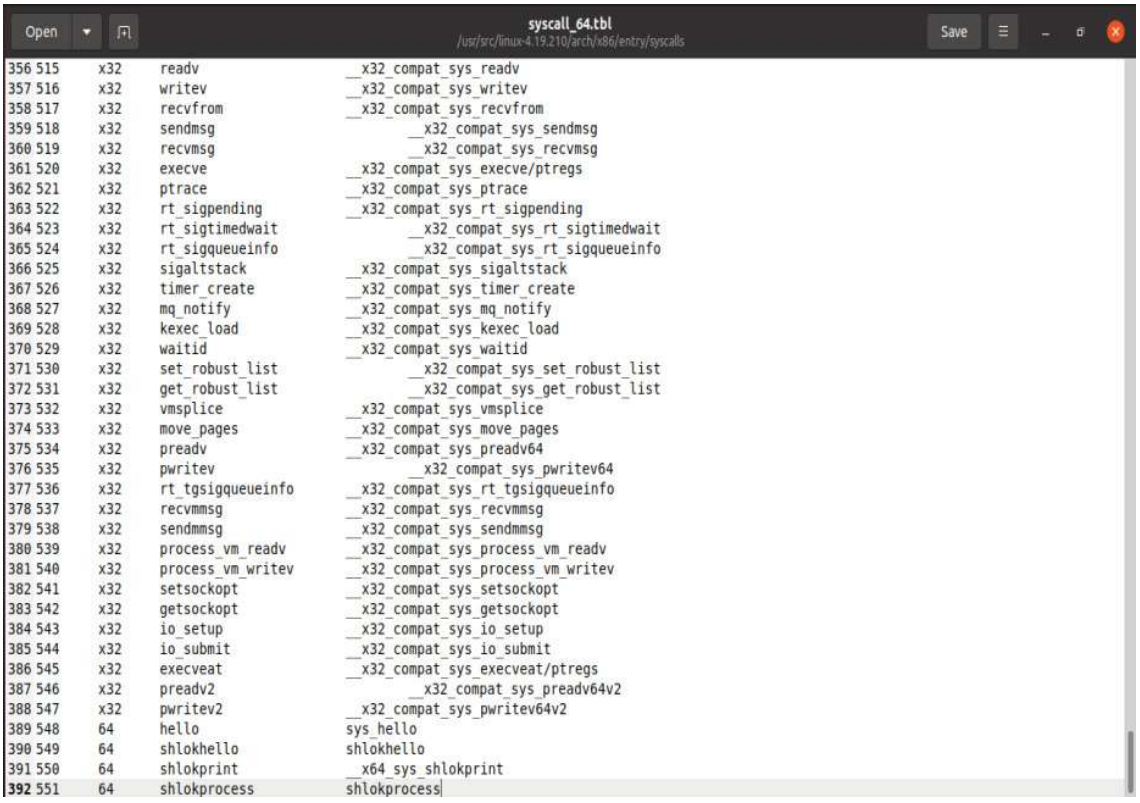
Step 6) Add the new system call to the system call table of kernel

```
cd arch/x86/entry/syscalls/
```

The system call table of the kernel lies in above directory. Change your current directory with the above command.

As my system is 64 bit so **add entry in syscall_64.tbl .**

```
gedit syscall_64.tbl
```



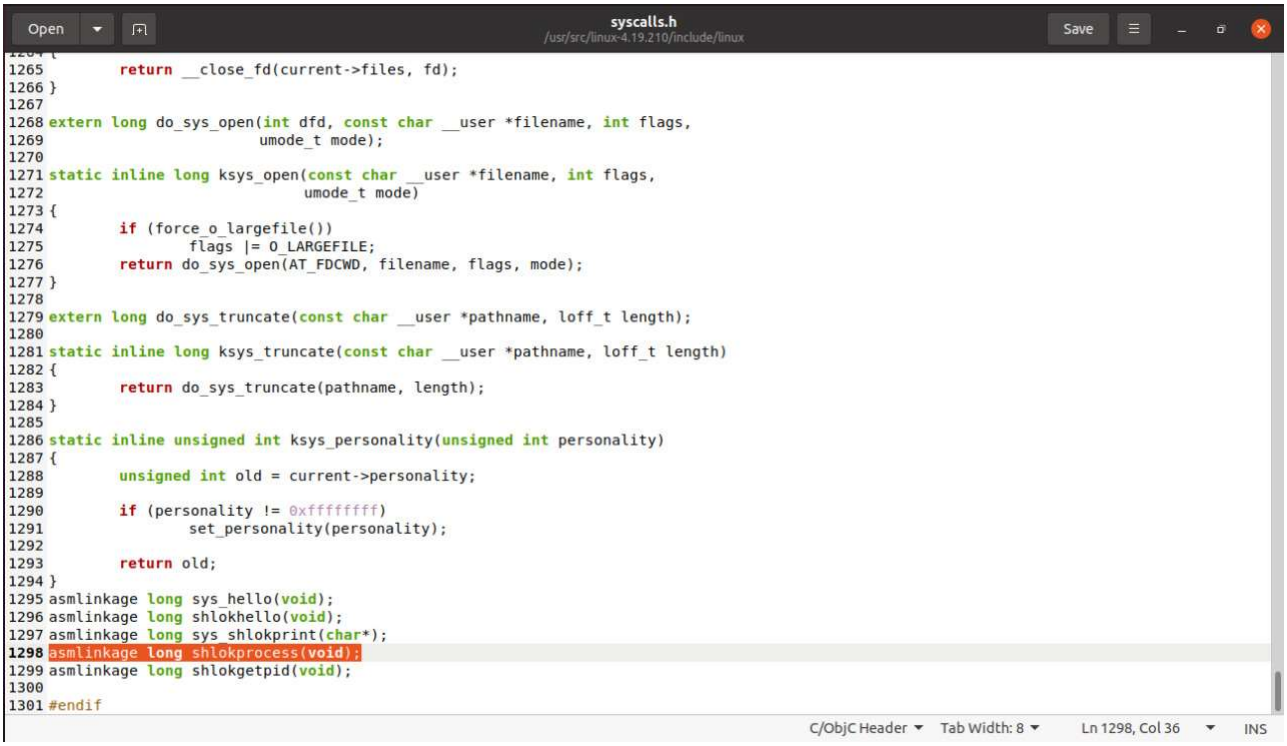

```
356 515     x32     readv                   __x32_compat_sys_readv
357 516     x32     writev                  __x32_compat_sys_writev
358 517     x32     recvfrom                __x32_compat_sys_recvfrom
359 518     x32     sendmsg                         __x32_compat_sys_sendmsg
360 519     x32     recvmsg                         __x32_compat_sys_recvmsg
361 520     x32     execve                  __x32_compat_sys_execve/ptregs
362 521     x32     ptrace                  __x32_compat_sys_ptrace
363 522     x32     rt_sigpending           __x32_compat_sys_rt_sigpending
364 523     x32     rt_sigtimedwait                 __x32_compat_sys_rt_sigtimedwait
365 524     x32     rt_sigqueueinfo                 __x32_compat_sys_rt_sigqueueinfo
366 525     x32     sigaltstack             __x32_compat_sys_sigaltstack
367 526     x32     timer_create            __x32_compat_sys_timer_create
368 527     x32     mq_notify               __x32_compat_sys_mq_notify
369 528     x32     kexec_load              __x32_compat_sys_kexec_load
370 529     x32     waitid                  __x32_compat_sys_waitid
371 530     x32     set_robust_list                 __x32_compat_sys_set_robust_list
372 531     x32     get_robust_list                 __x32_compat_sys_get_robust_list
373 532     x32     vmsplice                __x32_compat_sys_vmsplice
374 533     x32     move_pages              __x32_compat_sys_move_pages
375 534     x32     preadv                  __x32_compat_sys_preadv64
376 535     x32     pwritev                         __x32_compat_sys_pwritev64
377 536     x32     rt_tgsigqueueinfo       __x32_compat_sys_rt_tgsigqueueinfo
378 537     x32     recvmmsg                __x32_compat_sys_recvmmsg
379 538     x32     sendmmsg                __x32_compat_sys_sendmmsg
380 539     x32     process_vm_readv        __x32_compat_sys_process_vm_readv
381 540     x32     process_vm_writev       __x32_compat_sys_process_vm_writev
382 541     x32     setsockopt              __x32_compat_sys_setsockopt
383 542     x32     getsockopt              __x32_compat_sys_getsockopt
384 543     x32     io_setup                __x32_compat_sys_io_setup
385 544     x32     io_submit               __x32_compat_sys_io_submit
386 545     x32     execveat                __x32_compat_sys_execveat/ptregs
387 546     x32     preadv2                         __x32_compat_sys_preadv64v2
388 547     x32     pwritev2                __x32_compat_sys_pwritev64v2
389 548     64      hello                   sys_hello
390 549     64      shlokhello              shlokhello
391 550     64      shlokprint              __x64_sys_shlokprint
392 551     64      shlokprocess            shlokprocess
```

In kernel all system calls are identified by a unique number.

Here we assigned a unique number to our own system function call i.e., **551.**

Step 7) Add new system call to the system call header file

```
cd /usr/src/linux-4.19.210/
cd include/linux/
gedit syscalls.h
```




```
1265        return __close_fd(current->files, fd);
1266 }
1267
1268 extern long do_sys_open(int dfd, const char __user *filename, int flags,
1269                         umode_t mode);
1270
1271 static inline long ksys_open(const char __user *filename, int flags,
1272                              umode_t mode)
1273 {
1274         if (force_o_largefile())
1275                 flags |= O_LARGEFILE;
1276         return do_sys_open(AT_FDCWD, filename, flags, mode);
1277 }
1278
1279 extern long do_sys_truncate(const char __user *pathname, loff_t length);
1280
1281 static inline long ksys_truncate(const char __user *pathname, loff_t length)
1282 {
1283         return do_sys_truncate(pathname, length);
1284 }
1285
1286 static inline unsigned int ksys_personality(unsigned int personality)
1287 {
1288         unsigned int old = current->personality;
1289
1290         if (personality != 0xffffffff)
1291                 set_personality(personality);
1292
1293         return old;
1294 }
1295 asmlinkage long sys_hello(void);
1296 asmlinkage long shlokhello(void);
1297 asmlinkage long sys_shlokprint(char*);
1298 asmlinkage long shlokprocess(void);
1299 asmlinkage long shlokgetpid(void);
1300
1301 #endif
```

This defines the prototype of the function of our system call.

Step 8) Compile the Kernel

```
cd /usr/src/linux-4.19.210/
make menuconfig
```

This command is used to configure the Linux kernel. A pop up will come just check that ext4 in file system is included.

```
make -j4
```

This command will start compiling the kernel. Now j4 in **make –j4** makes use of 4 cores in system to make this file. The more cores we use the faster the compilation will take place. -jn tells number of cores to be used and n cannot be greater than number of cores in your machine

Step 9) Update or Install the kernel

```
make modules_install install
```

The above command will make changes in the kernel according to the files which we have compiled above, and the changes will be updated in the 4 files in **/boot** directory: -

3. System.map-4.17.4
4. vmlinuz-4.17.4
5. initrd.img-4.17.4
6. config-4.17.4

Now reboot the system to see all the changes in kernel.

Code:

Q3.c

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/syscall.h>

int main()
{
        pid_t process_id=getpid();
        pid_t parent_pro_id=getppid();
        long int num=syscall(551);
        printf("%ld\n",num);
        printf("Process id:%d",process_id);
        printf("\nParent Process id:%d",parent_pro_id);
        return 0;
}
```

Output:

```
shlok@shlok:~$ ./q3
0
Process id:4824
Parent Process id:2255shlok@shlok:~$ dmesg
```

```
[   24.791592] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[   24.791609] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[   70.412982] rfkill: input handler disabled
[ 2757.974504] kworker/dying (319) used greatest stack depth: 13032 bytes left
[10752.982342] kworker/dying (2990) used greatest stack depth: 12752 bytes left
[11735.885059] Welcome to Linux shlok
[11758.476060] Welcome to Linux shlok
[11863.018553] Omkar
[11874.622972] Omkar
[12020.839314] CID:4824  PID:2255
```

Q) Are both process ids i.e. is current process id and parent process id same or different? Why? What are your observations.

Ans:

Every process has a unique process id which is assigned by OS during the process creation.

Now as we know that when a function call takes place during execution of process the executing code changes but the running process remains same. Similarly when a system call is called no new process is created so the currently running process does not changes so Process ID also not changes and the system call is part of this process.

As you can see that in above example the system call shlokprocess() was called by q3. The process id of q3 and the CID (Current ID) of system call is same i.e. 4824. Therefore parent of q3 is same as parent of system call as systemcall is part of that is why the PID (Parent id) printed in system call is same as the parent of q3 i.e. 2255.

Question 4) Write system call to execute some predefined system call from your written system call.

Step 1) Go to the kernel directory where we want to make a new system call

```
cd /usr/src/linux-4.19.210/
```

With this we change our and work in the kernel directory.

Step 2) Make the directory in which you will write the source code of the system call and go into that directory.

```
mkdir shlokgetpid
```

```
cd shlokgetpid
```

Step 3) Create shlokgetpid.c in this directory.

```
gedit shlokgetpid.c
```


```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/unistd.h>
#include <linux/syscalls.h>
#include <linux/fs.h>
#include <linux/sched.h>


asmlinkage long shlokgetpid(void)
{
        return current->pid;
}
```

- **asmlinkage** is a keyword which is used to indicate that all parameters of the function would be available of stack.
- **void** indicates function that shlokgetpid() does not take any parameters.
- **current** is global pointer variable which of **struct task_struct type**. It points to the current process' data and with the help of pointer we are easily accessing the process' pid which is available in it's tast_struct data and return its value.

Step 4) Create Makefile

```
gedit Makefile
```

This is to ensure that shlokgetpid.c is compiled when the kernel is compiled.

Step 5) Edit the Makefile of Kernel

```
cd /usr/src/linux-4.19.210/
```

```
nano Makefile
```

Look for the line which looks like below

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

In this line add shlokgetpid/

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

- By adding shlokgetpid/ in this line we are telling the compiler of the kernel that our new system call is defined in shlokgetpid directory.
- It will tell the compiler to make the Makefile in this directory which was mentioned above.

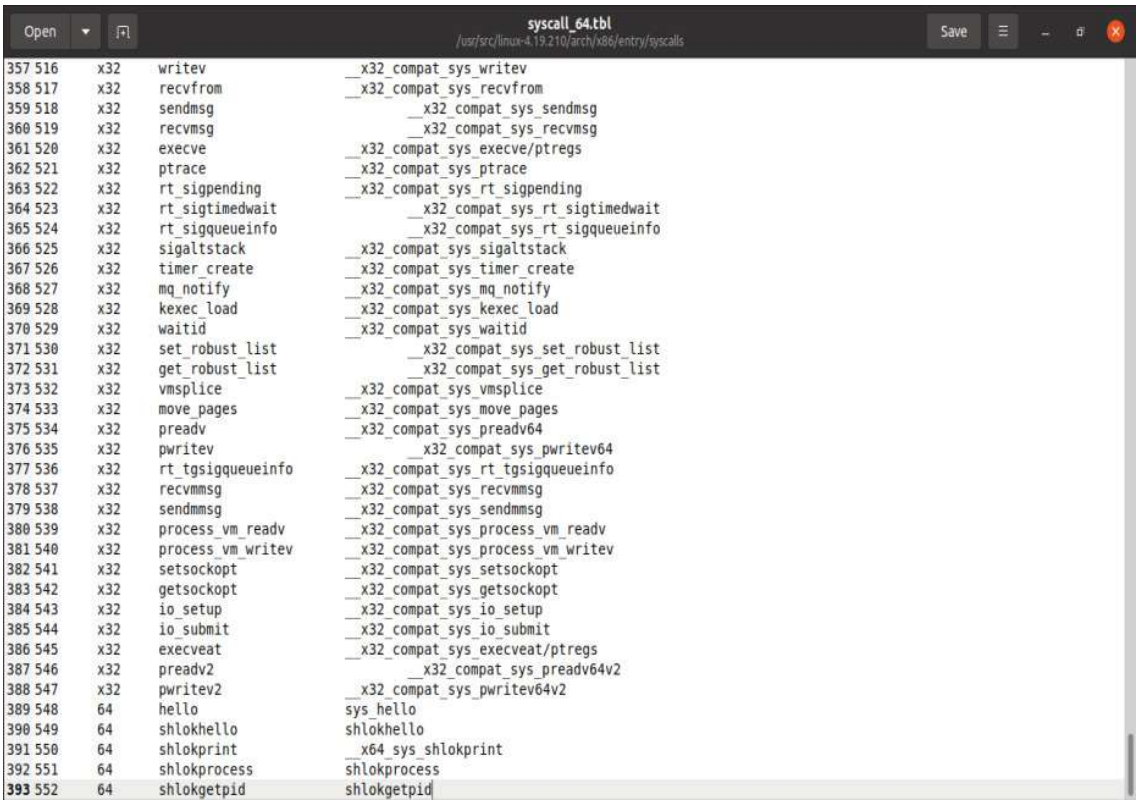
Step 6) Add the new system call to the system call table of kernel

```
cd arch/x86/entry/syscalls/
```

The system call table of the kernel lies in above directory. Change your current directory with the above command.

As my system is 64 bit so **add entry in syscall_64.tbl .**

```
gedit syscall_64.tbl
```




```
357 516     x32     writev                  __x32_compat_sys_writev
358 517     x32     recvfrom                __x32_compat_sys_recvfrom
359 518     x32     sendmsg                         __x32_compat_sys_sendmsg
360 519     x32     recvmsg                         __x32_compat_sys_recvmsg
361 520     x32     execve                  __x32_compat_sys_execve/ptregs
362 521     x32     ptrace                  __x32_compat_sys_ptrace
363 522     x32     rt_sigpending           __x32_compat_sys_rt_sigpending
364 523     x32     rt_sigtimedwait                 __x32_compat_sys_rt_sigtimedwait
365 524     x32     rt_sigqueueinfo                 __x32_compat_sys_rt_sigqueueinfo
366 525     x32     sigaltstack             __x32_compat_sys_sigaltstack
367 526     x32     timer_create            __x32_compat_sys_timer_create
368 527     x32     mq_notify               __x32_compat_sys_mq_notify
369 528     x32     kexec_load              __x32_compat_sys_kexec_load
370 529     x32     waitid                  __x32_compat_sys_waitid
371 530     x32     set_robust_list                 __x32_compat_sys_set_robust_list
372 531     x32     get_robust_list                 __x32_compat_sys_get_robust_list
373 532     x32     vmsplice                __x32_compat_sys_vmsplice
374 533     x32     move_pages              __x32_compat_sys_move_pages
375 534     x32     preadv                  __x32_compat_sys_preadv64
376 535     x32     pwritev                         __x32_compat_sys_pwritev64
377 536     x32     rt_tgsigqueueinfo       __x32_compat_sys_rt_tgsigqueueinfo
378 537     x32     recvmmsg                __x32_compat_sys_recvmmsg
379 538     x32     sendmmsg                __x32_compat_sys_sendmmsg
380 539     x32     process_vm_readv        __x32_compat_sys_process_vm_readv
381 540     x32     process_vm_writev       __x32_compat_sys_process_vm_writev
382 541     x32     setsockopt              __x32_compat_sys_setsockopt
383 542     x32     getsockopt              __x32_compat_sys_getsockopt
384 543     x32     io_setup                __x32_compat_sys_io_setup
385 544     x32     io_submit               __x32_compat_sys_io_submit
386 545     x32     execveat                __x32_compat_sys_execveat/ptregs
387 546     x32     preadv2                         __x32_compat_sys_preadv64v2
388 547     x32     pwritev2                __x32_compat_sys_pwritev64v2
389 548     64      hello                   sys_hello
390 549     64      shlokhello              shlokhello
391 550     64      shlokprint              __x64_sys_shlokprint
392 551     64      shlokprocess            shlokprocess
393 552     64      shlokgetpid             shlokgetpid
```

In kernel all system calls are identified by a unique number.

Here we assigned a unique number to our own system function call i.e., **552.**

Step 7) Add new system call to the system call header file

```
cd /usr/src/linux-4.19.210/
cd include/linux/
gedit syscalls.h
```


```
1265        return __close_fd(current->files, fd);
1266 }
1267
1268 extern long do_sys_open(int dfd, const char __user *filename, int flags,
1269                         umode_t mode);
1270
1271 static inline long ksys_open(const char __user *filename, int flags,
1272                              umode_t mode)
1273 {
1274        if (force_o_largefile())
1275                flags |= O_LARGEFILE;
1276        return do_sys_open(AT_FDCWD, filename, flags, mode);
1277 }
1278
1279 extern long do_sys_truncate(const char __user *pathname, loff_t length);
1280
1281 static inline long ksys_truncate(const char __user *pathname, loff_t length)
1282 {
1283        return do_sys_truncate(pathname, length);
1284 }
1285
1286 static inline unsigned int ksys_personality(unsigned int personality)
1287 {
1288        unsigned int old = current->personality;
1289
1290        if (personality != 0xffffffff)
1291                set_personality(personality);
1292
1293        return old;
1294 }
1295 asmlinkage long sys_hello(void);
1296 asmlinkage long shlokhello(void);
1297 asmlinkage long sys_shlokprint(char*);
1298 asmlinkage long shlokprocess(void);
1299 asmlinkage long shlokgetpid(void);
1300
1301 #endif
```

This defines the prototype of the function of our system call.

Step 8) Compile the Kernel

```
cd /usr/src/linux-4.19.210/
make menuconfig
```

This command is used to configure the Linux kernel. A pop up will come just check that ext4 in file system is included.

```
make -j4
```

This command will start compiling the kernel. Now j4 in **make –j4** makes use of 4 cores in system to make this file. The more cores we use the faster the compilation will take place. -jn tells number of cores to be used and n cannot be greater than number of cores in your machine

Step 9) Update or Install the kernel

```
make modules_install install
```

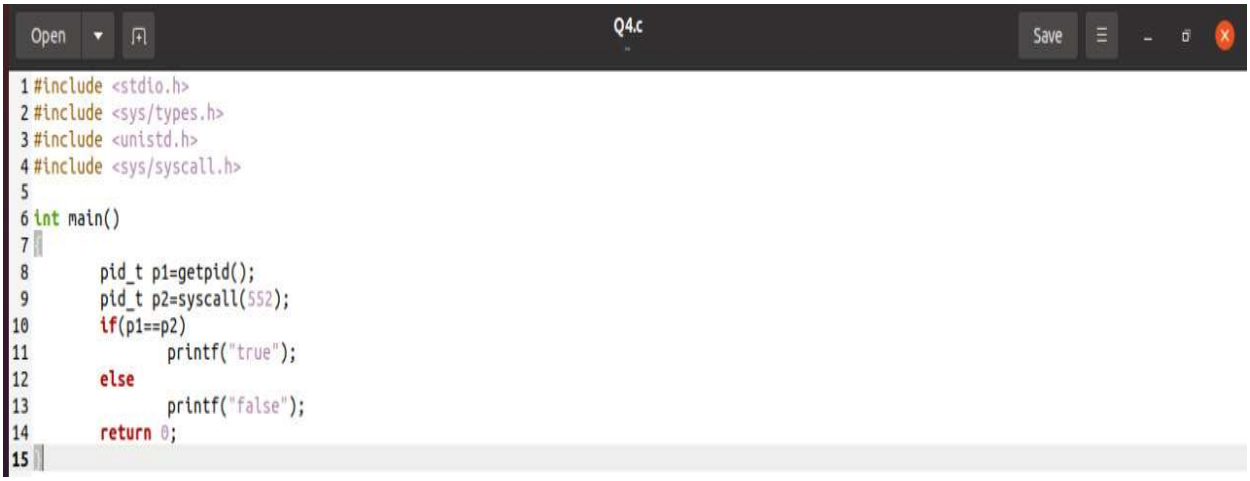
The above command will make changes in the kernel according to the files which we have compiled above, and the changes will be updated in the 4 files in **/boot** directory: -

4. System.map-4.17.4
5. vmlinuz-4.17.4
6. initrd.img-4.17.4
7. config-4.17.4

Now reboot the system to see all the changes in kernel.

Step 10) Test the system call

Code:



Output: