Name : Jayesh Patil
Div : 2
Roll no : T21287

Assignment no :- 9

Study assignment of implementation & addition of a new system call

Aim : Implement a new system call in the kernel space, add this new system call in the Linux kernel by the this compilation of this kernel (any kernel source, any architecture and any linux kernel distribution) & demonstrate the use of this embedded system call using C program in user space

Objective : To study
Linux kernel architecture
System call

Theory :
steps
following are the steps to add a new system call in Linux

1 change to the steps to add a new kernel sources directory using
cd /usr/src/linux-3.17.7/
2. Define a new system call sys_hello()
Create a directly hellow in the kernel source directly

mkdir hello
change into this directory
cd hello

3. create a "hello c" file in this folder & add the definition of the system call to it as given below
gedithello . c

Add the following code :

```
# include <linux/kernel h >
    asmlinkage long sys _ hello (void )
{
        Printk ("Hello world \n") ;
        return 0 ;

}
```

Note that printk prints to the kernel's log file

4. create a 'Makefile' in the hello folder & add the given line to it
gedit makefile
Add the following line to it :-
obj -y = hello . o
This is to ensure that the hello . c file is complied & included in the kernel source code

5. Add the hello directory to the kernel's makefile. change back into the linux -3.17.7. folder & open mark
geditmakefile
Go to line number 842 which says :
" core y+ = kernel / mm / FS / ipc / security / crypton / "
change this to

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

This is to tell the compiler that the source file of our new system call (sys_hello()) are present in the hello directory

6. Add the new system call (sys_hello()) into the system call table (syscall_32.tbl file)
   If your system is a 64 bit system, you will need to alter the syscall_64.tbl file

```
cd arch/x86/syscalls
gedit syscall_32.tbl
```

Add the following line at the end of the file :

```
354    i386    hello    sys_hello
```

354 : It is the number of the system call. It should be one plus the number of the last system call (It was 354 in my system) This has to be noted down to make the system call in the user pace program)

7. Add the new system call (sys_hello()) in the system call header file

```
cd include/linux/
geditsyscalls.h
```

Add the following line to the end of the file just before the #endif statement at the very bottom

```
asmlinkage long sys_hello(void);
```

This define the prototype of the function of our system call "asmlinkage" is a keyword used to indicate that all parameters of the function would be available on the stack

8. Compile this kernel on your system
   To compile linux kernel the following are required to be installed
   1. gcc latest version.
   2. ncurses development package
   3. system packages should be up to date.

To configure your kernel use the following command:

```
sudo make menuconfig
```

once the above command is used to configure the linux kernel, you will get a popup window with the list of menus & you can select the items for the new configuration. If you run familiar with the configuration just check for the file system menu & check whether "ext4" it chosen or not, if not select it & save the configuration

If you like to have your existing configuration, then run the below command

```
sudo make oldconfig
```

Now to compile the kernel ; do make

```
cd / usr/ src/ linux - 3.17.7 /
```

Now compile this program using the following command.

```
gccuserspace c
```

If all goes well you will not have any errors else rectify the errors.
Now run the program using the following command
./a.out
you will see the following line getting printed in the terminal if all the steps were followed correctly.
"system call sys_hello ()".

Now to check the message of the kernel, you can run the following command.
dmesg

This will display "Hello world" at the end of the kernel's message.
Say, we wanted to add our own version of the system call getpid (). let's call our version mygetpid (). This implementation of mygetpid () is :

```
asmlinkage long sys_getpid (void)
 {
     return current -> tgid ;

 }
```

Note : asmlinkage must appear before every system call. It tell compiler to only look on the stack for the functions arguments (aka compiler market magic.)