Assignment No 10

SYSTEM CALL

P V Talele

Instruction int 0x80

- int means interrupt, and the number 0x80 is the interrupt number.
- An interrupt transfers the program flow to whomever is handling that interrupt, which is interrupt 0x80 in this case.
- In Linux, 0x80 interrupt handler is the kernel, and is used to make system calls to the kernel by other programs.
- The kernel is notified about which system call the program wants to make.

System Call

- What is a system call table ?
- System calls in Linux are stored in syscall.tbl in arch/syscalls
  - 32-bit system call table – syscall_32.tbl
  - 64-bit system call table – syscall_64.tbl
- Ways to write a system call to kernel
  - Adding a kernel module
  - Change the existing kernel code

Steps

- Setting-up our system call directory
- cd /usr/src/linux-3.17.7
- mkdir addnum
- cd addnum
- Write the code for the system call in the source file (e.g. addnum.c)
- Create the Makefile for our system call in the same directory (e.g. Makefile).

addnum.c

```
/* addnum.c system call code */
#include <linux/kernel.h>
/* asmlinkage indicates we use the kernel stack to pass
   parameters */
asmlinkage long sys_addnum(int i, int j)
{
printk(KERN_INFO "Addnum is working! Now
   adding %d and %d", i, j);
return i+j;
}
```

Makefile

- obj-y := addnum.o

Steps Continued…

- Make changes to kernel Makefile so as to accommodate our directory of a system call.
  - Find the following line in the Makefile
    - core-y += kernel/mm/fslipc/security/crypto/block
  - Update it to
    - core-y += kernel/ mm/ fslipc/ security/ crypto/ block/ addnum/
- Adding our system call to the 32-bit or 64-bit system call table

Continued…

- Make changes to
  - /usr/src/linux-3.17.7/arch/x86/syscalls/syscall_64.tbl
  - At line no. 321, add the following line
    - 321 COMMON addnum sys_addnum
  - Save and exit
- Make changes to
  - /usr/src/linux-3.17.7/include/linux/syscalls.h to add a prototype of our system call.
    - asmlinkage long sys_addnum(void);

Continued…

- Compile, link and install the kernel
  - sudo make menuconfig
    - Configure the system
  - sudo make –j5
    - Complies all source files
    - j stands for no. of jobs
    - (4 cores + 1)

Continued…

- sudo make modules_install
  - Install modules at /lib/modules
- sudo make install
  - Makes an entry in grub (updates the grub)
- sudo reboot
  - Restarts the system

Continued…

- Now implement the helper program that will call our system call.
- Write a helper program (e.g. helper.c in any directory)
- Use syscall function known as indirect system call to invoke our system call.
- syscall is needed because we have no wrapper function to call our system call through that wrapper function.

helper.c

```
#include <stdio.h>
#include <ctype.h>
#include <syscall.h>

int main()
{
   int var1, var2 = 0;
   long res = 0;

   printf("Please enter 2 numbers:  ");
   scanf("%d%d", &var1, &var2);
    res = syscall(321, var1, var2);
   printf("\nYou entered %d and %d. Result is: %ld\n", var1, var2, res);

   return 0;
}
```