Assignment 3 - System calls and kernel programming in linux

Written by Akilesh B, CS13B1042

August 23, 2015

Tasks

1. Implement a basic system call which prints helloworld on the terminal and also onto the kernel log file.
2. Next, implement a variant in which I pass two parameters to the helloworld system call: integer (int) and a string (char []).
3. Test these system calls by invoking it from the main program.

Download kernel source

1. Give this command: apt-get source linux-image-$(uname -r). This downloads the source code for your version of linux installed in system. Mine is Ubuntu 14.04 LTS 32-bit version.
2. I continued this assignment in the same kernel in which I did my Assignment 1.

Part 1: helloworld

1.1 Creating new system call

1. Go to /usr/src/linux-lts-vivid-3.19.0/ and create a directory hello, by giving mkdir hello
2. Move into this directory ie cd hello
3. Create a hello.c file in this directory

   #include <linux/kernel.h>

   #include <linux/linkage.h>

   asmlinkage long sys_hello(void) {

   printk(KERN_ALERT "hello world\n");

   return 0;

   }

   KERN_ALERT is a higher priority to make sure that printk() messages are printed to the console rather than just getting logged inside log file. This can also be ensured by changing dmesg level.

   Now, create a Makefile in this directory and add the below line:

   obj-y := hello.o

   This ensures that hello.c file is compiled and included in source code of kernel.

1.2 Adding created directory to kernel's Makefile

- Go to /usr/src/linux-lts-vivid-3.19.0 and open Makefile.
- Change this line "core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ " to "core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ hello/". This informs the compiler that source files of new system call (sys_hello()) is present in hello directory.

1.3 Adding sys_hello() to system call table

- Go to /usr/src/linux-lts-vivid-3.19.0/arch/x86/syscalls and open syscall_32.tbl. Add this line to the end of file :-

  359 i386 hello sys_hello

  Here 359 is the system call number which is one more than the last system call number.

1.4 Adding sys_hello() in the system call header

- cd /usr/src/linux-lts-vivid-3.19.0/include/linux/ and open syscalls.h . Now add the below line

  asmlinkage long sys_hello(void);

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

1.5 Compiling the kernel on my system

Pre-steps

Go to /usr/src/linux-lts-vivid-3.19.0 and execute below commands

make clean

make localmodconfig

Actual compilation

make

1.6 Install and update the kernel

- Once kernel is successfully compiled without any errors, this modified kernel should be installed by giving the below command.

  make modules_install install

- To update the kernel, reboot the system.

1.7 Testing the system call

- Create a test.c program in Desktop or home folder.

  #include <stdio.h>

  #include <linux/kernel.h>

  #include <sys/syscall.h>

  #include <unistd.h>

  int main()

  {

  long int test = syscall(359);

  printf("System call sys_hello returned %ld\n", test);

  return 0;

  }

  Compile this program: gcc test.c

- When this program is run by giving ./a.out, we will see the below lines getting printed in full terminal mode (Ctrl+Alt+F1).

  executing ./a.out

  hello world

  System call sys_hello returned 0

- I'm getting "executing ./a.out" as I used the same modified kernel of Assignment 1.

- Hence, part 1 was successfully implemented.

Part 2: Variant of helloworld

In this section most of the steps remain the same except a few changes as mentioned below.

- In 1.1, create a new directory modhello and inside this directory create a modhello.c as below

  #include <linux/kernel.h>

  #include <linux/linkage.h>

  asmlinkage long sys_modhello(int a, char* b) {

  printk(KERN_ALERT "hello world\n");

  printk(KERN_ALERT "integer is %d\n", a);

  printk(KERN_ALERT "string is %s\n", b);

return 0;

}

Inside the Makefile give

obj-y := modhello.o

- In 1.2, instead of "core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ hello/" give "core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ modhello/"
- In 1.3, add this line

  360 i386 modhello sys_modhello
- In 1.4, add this line

  asmlinkage long sys_modhello(int a, char* b);
- Steps 1.5 and 1.6 remains the same.
- To test the new system call, create modtest.c in Desktop or home folder.

  #include <stdio.h>

  #include <linux/kernel.h>

  #include <sys/syscall.h>

  #include <unistd.h>

  int main()

  {

  long int test = syscall(360, 123, "akilesh");

  printf("System call sys_hello returned %ld\n", test);

  return 0;

  }

  Compile this program: gcc modtest.c
- When this program is run by giving ./a.out, we will see the below lines getting printed in full terminal mode (Ctrl+Alt+F1).

  executing ./a.out

  hello world

  integer is 123

  string is akilesh

  System call sys_hello returned 0
- Hence, part 2 is also successfully implemented.

  The below is the screen shot. I'm getting executing ./a.out as I have modified the same kernel I used for Operating Systems Assignment 1.

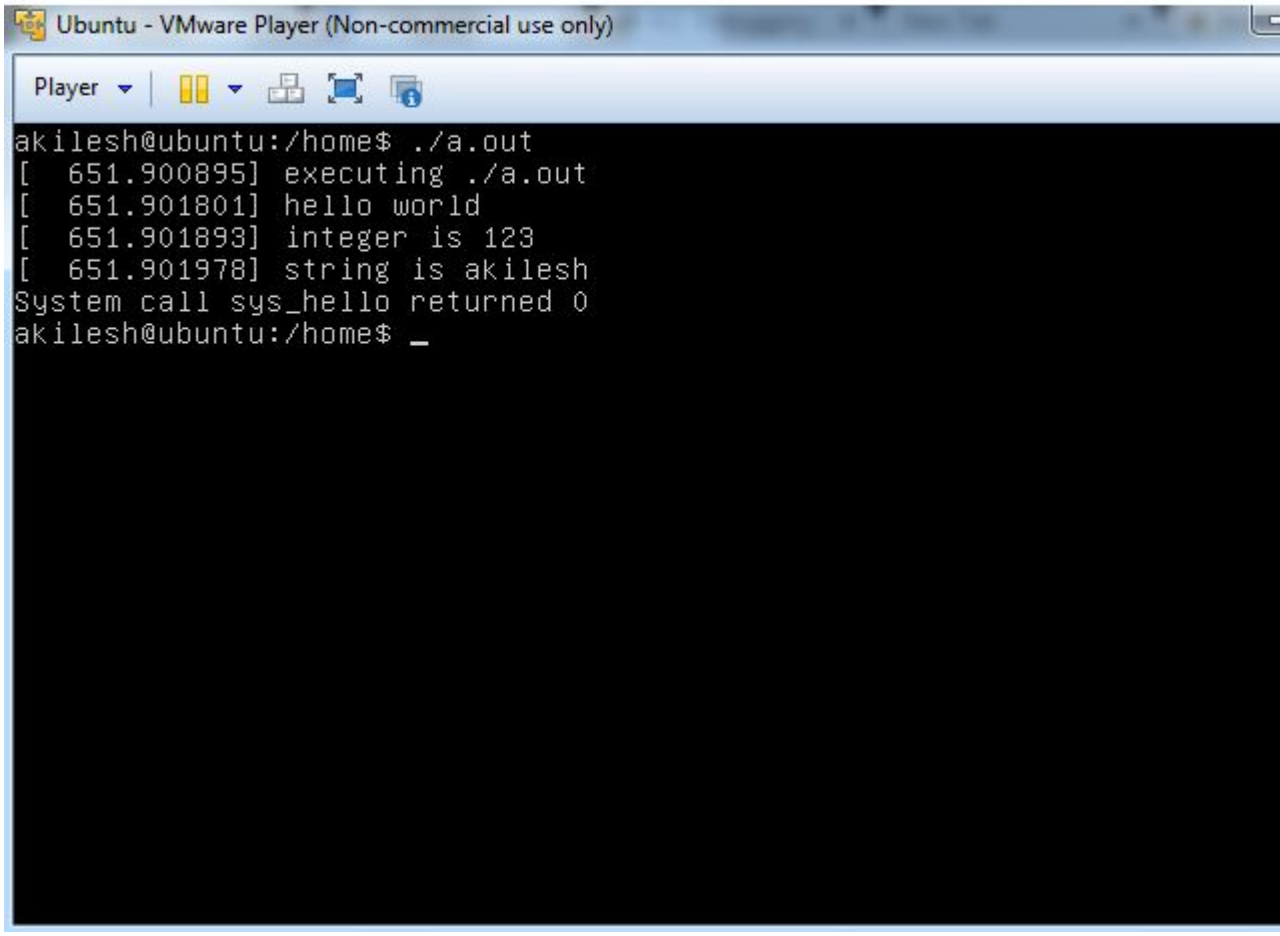

Figure 1: Screenshot of terminal