Assignment 3:
System Calls and Kernel programming on Linux

Deadline: 24th August 2015, 9:00 pm

Goal:
In this assignment you will study the system-call interface provided by the Linux operating system and learn how user programs communicate with the operating system kernel via this interface. Your task is to include a new system call into the kernel.

Part 1: Getting Started
A user-mode procedure call is performed by passing arguments to the called procedure either on the stack or through registers, saving the current state and the value of the program counter, and jumping to the beginning of the code corresponding to the called procedure. The process continues to have the same privileges as before. System calls appear as procedure calls to user programs but result in a change in execution context and privileges. System calls appear as procedure calls to user programs but result in a change in execution context and privileges. The list of pointers to system-call handlers is typically stored in the file/lusr/src/linux-2.x/arch/i386/kernel/entry.S under the heading ENTRY (sys_call_table).

Part 2: Building a New Kernel
Before adding a system call to the kernel, you must familiarize yourself with the task of building the binary for a kernel from its source code and booting the machine with the newly built kernel.

Part 3: Extending the Kernel Source
You can now experiment with adding a new file to the set of source files used for compiling the kernel. Typically, the source code is stored in the /usr/src/linux-2. x/kernel directory, although that location may differ in your Linux distribution. There are two options for adding the system call. The first is to add the system call to an existing source file in this directory. The second is to create a new file in the source directory and modify /usr/src/linux-2.x/kernel/Makefile to include the newly created file in the compilation process. The advantage of the first approach is that when you modify an existing file that is already part of the compilation process, the Makefile need not be modified.

Part 4: Adding a System Call to the Kernel
Now that you are familiar with the various background tasks corresponding to building and booting Linux kernels, you can begin the process of adding a new system call to the Linux kernel. In this project, the system call will have limited functionality; it will simply transition from user mode to kernel mode, print a message that is logged with the kernel messages, and transition back to user mode. We will call this the *helloworld* system call. While it has only limited functionality, it illustrates the system-call mechanism and sheds light on the interaction between
user programs and the kernel.

Create a new file called helloworld.c to define your system call. Include the header files linux/linkage. h and linux/kernel. h.

```
#include <linux/linkage.h>
#include <linux/kernel.h>
        asmlinkage int sys_helloworld() {
        printk(KERN_EMERG "hello world!");
  return 1;
}
```

This creates a system call with the name sys_helloworld ().

Define a new system call number for __NR_helloworld in /usr/src/linux-2.x/include/asm-i386/unistd.h. A user program can use this number to identify the newly added system call. Also be sure to increment the value for __ NR_syscalls, which is stored in the same file. This constant tracks the number of system calls currently defined in the kernel.
Add an entry .long sys_helloworld to the sys_call_table defined in the /usr/src/linux-2.x/arch/i386/kernel/entry.S file. As discussed earlier, the system-call number is used to index into this table to find the position of the handler code for the invoked system call.

Add your file helloworld. c to the Makefile (if you created a new file for your system call.) Save a copy of your old kernel binary image (in case there are problems with your newly created kernel). You can now build the new kernel rename it to distinguish it from the unmodified kernel and add an entry to the loader configuration files. After completing these steps, you can boot either the old kernel or the new kernel that contains your system call.

Part 5: Using the System Call from a User Program
When you boot with the new kernel it will support the newly defined system call; you now simply need to invoke this system call from a user program. As your new system call is not linked into the standard C library, however, invoking your system call will require manual intervention.

```
#include <linux/errno.h>
#include <sys/syscall.h>
#include <linux/unistd.h>
_syscallO(int, helloworld);
int main()
{
        helloworld();
}
```

Tasks to be done:

You can expand the functionality of your system call. How would you pass an integer value and a character string to the helloworld system call and how to print it into the kernel log file? While implementing this, you must consider the the implications of passing pointers to data stored in the user program's address space as opposed to simply passing an integer value from the user program to the kernel using hardware registers?

The set of deliverables are:

1. Implement the basic system call which prints helloworld on the terminal and also onto the kernel log file.

2. Next, implement a variant in which you will pass two parameters to the helloworld system call: integer (int) and a string (char []).

To test both these system-calls, you can invoke it from the main program like the sample code shown above.

Submission Instructions:

Please submit the following:

1. A report explaining the how you implemented the new system-call. Please mention the version of the kernel you used.

2. The source code of all the functions that you changed and developed for implementing the system-call.

Combine the report and source code into a zip file. Name it as ***assgn3-<rollno>.zip***. Submit your assignment by 24th August 2015, 9:00 pm.

Reading Materials and other Useful Links:

1. Operating System Concepts, Eight Edition by Avi Silberschatz,Peter Baer Galvin,Greg Gagne, Chapter 2, Page 93-97.
   This assignment has been taken from the programing assignment of chapter 2 given in the book.
2. www.tldp.org/LDP/lkmpg/2.6/lkmpg.pdf
3. http://www.tldp.org/HOWTO/html_single/Implement-Sys-Call-Linux-2.6-i386/
4. http://www.cs.ucr.edu/~kishore/cs153_w09/syscall-assignment/how-to-add-system-call.txt
5. https://tssurya.wordpress.com/2014/08/19/adding-a-hello-world-system-call-to-linux-kernel-3-16-0/
6. http://www.cs.rochester.edu/~sandhya/csc256/assignments/adding-a-system-call.html
7. http://www.csee.umbc.edu/courses/undergraduate/CMSC421/fall02/burt/projects/howto_add_system-call.html