CS6223 - Introduction to Operating Systems.

Assignment 2

The main purpose of this assignment is to familiarize you with linux kernel and system calls. In particular, you will add a new system call and hook existing system calls in linux kernel. Note that the below instruction works for Ubuntu 11.04 and Linux Kernel 2.6.39. If you work on other Linux distribution or kernel version, the steps might change. For example, in Linux kernel version 3.x, you need to modify the file syscall\_32.tbl instead of syscall\_table\_32.S

**Submit a report in PDF format along with code that details the following:**

1. What are kernel APIs? What is the difference between kernel APIs and system calls?

2. What are roles of files in /boot/ (vmlinuz-\*, initrd.img-\*, grub, config\*)?

3. Include the screenshots **AND all modified files** from tasks 1, 2 and 3 below.

**Task 1. Add “Hello World” system call**

1. Prepare the virtual machine.

- Install VirtualBox

- Download Ubuntu 11.04 Image: <http://virtualboxes.org/images/ubuntu/>

- You will work on this virtual machine.

- To reduce time for compiling Linux kernel so you could use multiple CPU cores for your virtual machine and use $make -jn when compiling the kernel, where n is number of CPU cores.

2. Download Linux Kernel: <https://www.kernel.org/> (2.6.39):

ftp://[ftp.kernel.org/pub/linux/kernel/v2.6/linux-2.6.39.tar.bz2](http://ftp.kernel.org/pub/linux/kernel/v2.6/linux-2.6.39.tar.bz2)

3. Extract the kernel to /usr/src/

4. Write your own system call, named my\_system\_call.c and place it in the *linux-2.6.39/kernel* folder

For example:

#include<linux/linkage.h>

#include<linux/kernel.h>

asmlinkage long sys\_hello(const char \*msg)

{

printk(KERN\_INFO “Hello I am in kernel space, %s”, msg);

    return 0;

}

4. Declare your new system call in the kernel source.

- Open the file *linux-2.6.39/arch/x86/kernel/syscall\_table\_32.S* and add the below line in the end of the file:

.long sys\_hello

- Open the file *linux-2.6.39/arch/x86/include/asm/unistd\_32.h* and add a new line after the line #define \_\_NR\_syncfs:

#define NR\_syshello 345

//Change 345 to other number if it is already defined.

      Change the line “#define NR\_syscalls 345” to “#define NR\_syscalls 346”

- Open the file *linux-2.6.39/arch/x86/include/asm/unistd\_64.h,* find line “\_\_SYSCALL(\_\_NR\_syncfs, sys\_syncfs)” and add two following lines after that line:

#define \_\_NR\_syshello 307

\_\_SYSCALL(\_\_NR\_syshello, sys\_hello)

     //Change 307 to other number if it is already defined.

- Open the file *linux-2.6.39/include/linux/syscalls.h* and add the below line in the end of the file and before #endif:

asmlinkage long sys\_hello(const char \*msg);

5. Declare your new system call in the Makefile:

- Open the file *linux-2.6.39/kernel/Makefile* and find out the line: obj-y += groups.o. After this line, add a new line:

obj-y += my\_system\_call.o

6. Compile and install your new kernel. Assume you are in the folder /usr/src/*linux-2.6.39.*

You could strip down unnecessary modules in the kernel by following this instruction: <http://linux-hacks.blogspot.com/2009/06/build-your-kernel-faster.html>

- Generate new configure file and save it as .config

$make oldconfig

$make or $make -jn, where n is number of CPU cores in your machine.

$make modules

$make modules\_install

$make install

$cd /boot

$mkinitramfs -o initrd.img-2.6.39 2.6.39

$update-grub

7. Test your new system call.

- Reboot your machine, choose your new kernel to boot.

- Compile and run this code:

//test\_syscall.c

#include <stdio.h>

#include <linux/unistd.h>

#include <sys/syscall.h>

#define sys\_hello 345

int main(void)

{

   char \*msg = “Hello System Call”;

   syscall(sys\_hello , msg);

   return 0;

}

- Check the output of the program: $dmesg (You should take screenshot of the output of this command)

**Task 2: Try to hook the system call: sys\_open.**

When a program call sys\_open on a file, let’s say my\_file.txt, the kernel print out a line: “File my\_file.txt is being opened”. After booting on a new modified kernel, use the command $dmesg

- Open my\_file.txt then using the commands $dmesg or $tail -f /var/log/syslog to see the result.

Note that, after you done compiling the kernel for the first task, it should take much less time to recompile the kernel latter for other tasks. So be patient :-)

**Task 3: Try to hook the system call: sys\_write.**

When a program call sys\_write on file my\_file.txt, add to the end of this file a line: “File my\_file.txt is being hacked”.

- Write a program that writes to a file then using the commands $dmesg to check the result.