SYSTEM CALL

Juhi Maheshwari (121020)

Kashyap Patel (121021)

# Project Description

The basic problem statement is to write a system call. The approaching step is to compile and install the latest version of kernel. Followed by, writing a system call and adding it to kernel. Then a kernel needs to be recompiled to test the system call function. A simple C program has to be written to invoke our system call.

# The project is mainly divided into 3 part

1. Downloading and initialization subversion repository with a copy of recent stable version
2. Creation of new branch in that repository within that branch and then write a system call and register it with kernel
3. This part is testing part. In this part we compile and install our modified kernel and then write a simple C code to invoke system call.

# What is System call?

System call is the fundamental interface between application and the Linux kernel. System call provides an essential interface between a process and the operating system.

A system call is an entry point into the Linux kernel. Usually, system calls are not invoked directly: instead, most system calls have corresponding C library wrapper functions which perform the steps required in order to invoke the system call. Thus, making a system call looks the same as invoking a normal library function.

There are 5 different categories of system calls:

Process control, file manipulation, device manipulation, information maintenance and communication.

## Process Control

Process control is basically changing the state of process. Running process, if invokes an I/O request must be blocked or put in a waiting state.

## File Manipulation

## Some common system calls are create*,*delete*,*read*,*writeandclose*.*

## Device Manipulation

Process usually requires several resources to execute. If these resources are available, they will be granted and control returns to the user process. Resources are released by process as soon as it terminates or has finished executing.

## Information Maintenance

Some system calls exist only to transfer information between user program and kernel/OS.

## Communication

It refers to inter-process communication. It can be done by message passing technique. Message-passing uses a common mailbox to pass messages between processes.

# Why do we need system calls?

System calls acts as entry point to OS kernel. There are certain tasks that can only be done if a process is running in kernel mode. Examples of these tasks can be interacting with hardware etc. So if a process wants to do such kind of task then it would require itself to be running in kernel mode which is made possible by system calls.

System calls are also used for switching the execution mode. Example: systenter and sysexit

# Difference between system call and library function

* A library function is linked to the user program and executes in user space while a system call is not linked to a user program and executes in kernel space.
* Library function’s execution time is counted in user level time while a system call execution time is counted as a part of system time.
* Library functions can be debugged easily using a debugger while System calls cannot be debugged as they are executed by the kernel.

# Work Done till 20th March

* Part-1 fully completed (Installation of Kernel)
* Part-2 is in progress (Writing our own System Call)

# Proposed work

* To write a system call and register it with kernel and compile kernel and then call that system call using Simple C code.

# Reference

1. <http://www.tldp.org/LDP/lkmpg/2.4/html/x939.html>
2. <http://en.wikipedia.org/wiki/System_call>
3. <http://www.thegeekstuff.com/2012/07/system-calls-library-functions/>
4. <http://man7.org/linux/man-pages/man2/intro.2.html>
5. <http://faculty.salina.k-state.edu/tim/ossg/Introduction/sys_calls.html>