ADDING AN EXAMPLE SYSTEM CALL

The example system call takes two integer variables as its arguments, finds their sum and returns this sum.

(Please note that this system call is an only example for illustrating the steps of adding a system call. So, this is not what we expect for Phase 1 of the Project. You should write the system call which is explained in the main document!!)

Note: Code in this document means both C files and header files.

1) For the implementation of system call, you will update several files in the system. So, define an exact name for the system call. In this document, we explain the procedure for adding a system call named ***mysyscall.***

2) First you should extract the kernel-sources in the **/usr/src** directory.

Go to **/usr/src** and type this command: “tar xjvf kernel-source-2.4.27.tar.bz2”

3) The directories for adding a new module in the kernel:

a) **/usr/include/** -> Contains user space headers. We will add a header file for the system call for the user space under a directory in this space.

Important: The user space programs can include header files under /usr/include/ directory.Example: An user space code can include /usr/include/stdio.h as #include<stdio.h>. An user space code can include /usr/include/sys/types.h as #include<sys/types.h>.

In the further process for the system call, **/usr/include/** directory will be named as **{USER_SPACE}**.

b) **/usr/src/kernel-source-2.4.27** -> Contains the files for kernel header and functions. This directory defines all of the functionalities of the kernel. The kernel will be compiled under this directory. It contains the function codes of all system calls.

Important: The modules under kernel can include header files under /usr/src/kernel-source-2.4.27/ directory. Example: A kernel code can include /usr/src/kernel-source-2.4.27/include/linux/sched.h as #include<linux/sched.h>. A kernel code can include /usr/src/kernel-source-2.4.27/include/asm-i386/uaccess.h as #include<asm/uaccess.h>.

The header files under kernel space could not be accessed by user space, or vice versa.

In the further process for the system call, **/usr/src/kernel-source-2.4.27** directory will be named as **{KERNEL_SPACE}**.

4) Adding system call.

a) Adding system call in the system call trap tables:

i) Kernel Space:

1) In *{KERNEL_SPACE}/arch/i386/kernel/entry.S*, add the following **red** line in the ***ENTRY(sys_call_table)*** before the line starting with ***.rept NR_...***

```
…
.long SYMBOL_NAME(sys_ni_syscall)    /* sys_set_tid_address */
.long SYMBOL_NAME(sys_mysyscall)   /* 259 */

.rept NR_syscalls-(.-sys_call_table)/4
…
```

This is done for linking the system call under x86 assembly.

2) In *{KERNEL_SPACE}/include/asm-i386/unistd.h*, add the following under the system call list.

after line

```
#define __NR_exit_group        252
```

line,

Add

```
#define __NR_mysyscall         259
```

This file contains the system call numbers for the operating system. If we add this line, mysyscall becomes identifiable for the kernel.

3) Adding system call under user space:

In *{USER_SPACE}/asm/unistd.h* file, delete lines 267-283, then, after

```
#define __NR_set_tid_address   258
```

line,

Add

```
#define __NR_mysyscall         259
```

b) Creating the Kernel Space System Call header.

Under *{KERNEL_SPACE}/include/linux*, create ***mysyscall.h***, and write the following lines

```
-------------------------------------------

#ifndef __LINUX_MYSYSCALL_H
#define __LINUX_MYSYSCALL_H
#include <linux/linkage.h>
#endif
--------------------------------------------
```

c) Creating the function codes of Kernel Space System Call.

Under *{KERNEL_SPACE}/fs*, create ***mysyscall.c*** and write the following lines:

```
---------------------------------------------------------

#include <linux/mysyscall.h>
asmlinkage int sys_mysyscall(int arg1,int arg2){
     return arg1+arg2;
}

----------------------------------------------------------
```

The function codes of a System call can be written under kernel space. User space can only call the system call.

Change ***{KERNEL_SPACE}/fs/Makefile*** file

```
obj-y :=    open.o read_write.o devices.o file_table.o buffer.o \
            super.o block_dev.o char_dev.o stat.o exec.o pipe.o namei.o \
            fcntl.o ioctl.o readdir.o select.o fifo.o locks.o \
            dcache.o inode.o attr.o bad_inode.o file.o iobuf.o dnotify.o \
            filesystems.o namespace.o seq_file.o xattr.o quota.o
```

to

```
obj-y :=    open.o read_write.o devices.o file_table.o buffer.o \
            super.o block_dev.o char_dev.o stat.o exec.o pipe.o namei.o \
            fcntl.o ioctl.o readdir.o select.o fifo.o locks.o \
            dcache.o inode.o attr.o bad_inode.o file.o iobuf.o dnotify.o \
            filesystems.o namespace.o seq_file.o xattr.o quota.o \
            mysyscall.o
```

If you do not do this, the function codes for ***mysyscall*** could not be compiled and added to the system.

d) How to add the system call header under user space:

Create ***mysyscall.h*** under ***{USER_SPACE}/linux/,*** and write the following lines:

```
#include <linux/unistd.h>
#include <errno.h>
extern int errno;
_syscall2(int, mysyscall, int, arg1, int , arg2);
```

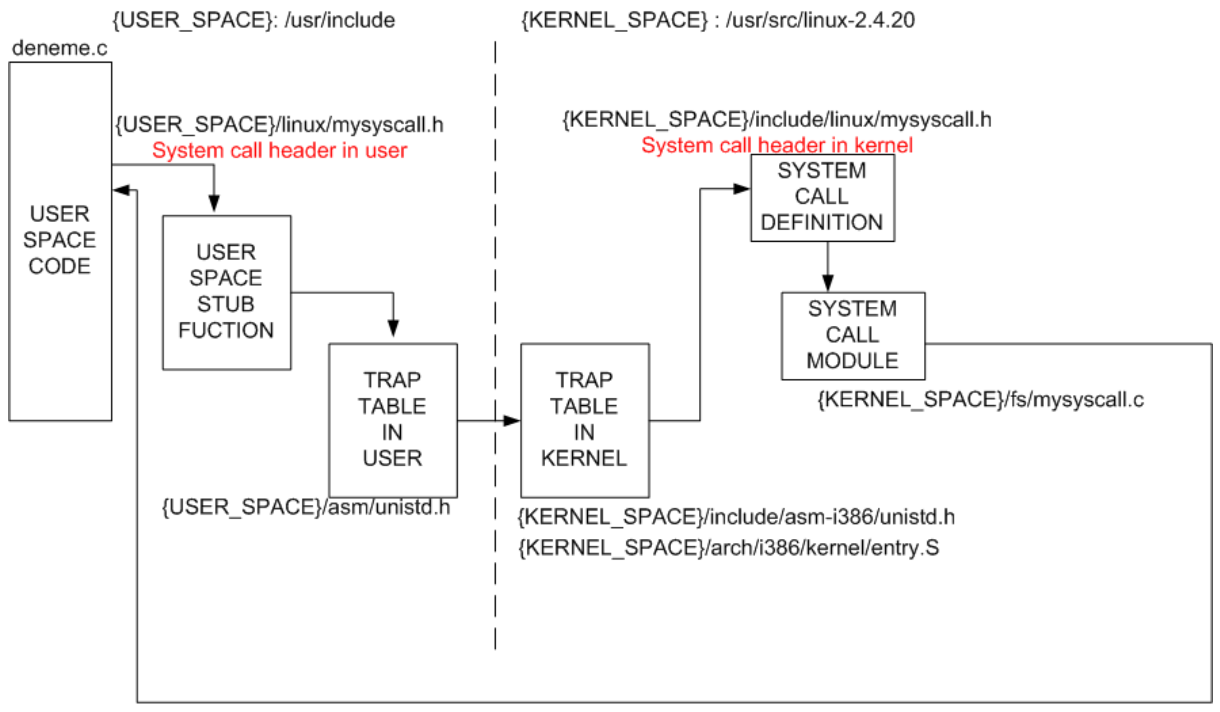
This creates a user stub system call function under user space and it will create the bridge between user space and kernel space.

e) User Space Stub Program:

For calling system call by an ordinary user function write the following C file:
deneme.c (**you should create this C file in your home folder**):

```
#include <linux/mysyscall.h>

main(){

int sum;

sum=mysyscall(10,10);

printf("%d",sum);

}
```

After that, you should compile the kernel. Refer to the Kernel Compilation file for it.


{USER_SPACE}: /usr/include
{KERNEL_SPACE} : /usr/src/linux-2.4.20
deneme.c
USER SPACE CODE
{USER_SPACE}/linux/mysyscall.h
System call header in user
USER SPACE STUB FUCTION
TRAP TABLE IN USER
{USER_SPACE}/asm/unistd.h
{KERNEL_SPACE}/include/linux/mysyscall.h
System call header in kernel
SYSTEM CALL DEFINITION
SYSTEM CALL MODULE
{KERNEL_SPACE}/fs/mysyscall.c
TRAP TABLE IN KERNEL
{KERNEL_SPACE}/include/asm-i386/unistd.h
{KERNEL_SPACE}/arch/i386/kernel/entry.S