**14. Adding a New System call to Linux Kernel**

**Aim**

**Adding out own developed system call to linus kernel**

**Procedure:**

1. Download the latest version of the 2.6 Linux kernel from [www.kernel.org](http://www.kernel.org).

2. Unzip and untar the kernel directory into /usr/src/.

3. In /usr/src/Linux-x.x.x/kernel/, Create a new file myservice.c to define your

system call.

#include <Linux/linkage.h> //for linking a system call

#include <Linux/kernel.h> //for the printk

asmlinkage int sys\_myservice (int arg1, char\* arg2) {

printk(KERN\_EMERG “my service is running”);

//kernel messages logged to /var/log/kernel/warnings

return(1);

}

**4.** In /usr/src/Linux-x.x.x/include/asm/unistd.h, define an index for your system call.

**Your index should be the number after the last system call defined in the list.**

#define \_\_NR\_myservice 274

5. Also, you should increment the system call count.

#define \_\_NR\_syscalls 275

**6.** In /usr/src/Linux-x.x.x/arch/i386/kernel/entry.S, you should define a pointer to

hold a reference to your system call routine. **It is important that your data entry**

**placement corresponds to the index you assigned to your system call.**

.long sys\_myservice

7. Add your system call to the Makefile in /usr/src/Linux-x.x.x/kernel/Makefile.

Add your object after the other kernel objects have been declared.

obj-y += myservice.o

8. Make your system from /usr/src/Linux- x.x.x

make xconfig //save the defaults

make dep //make dependency list

make bzImage //build your kernel

9. Add a new boot image to Lilo, by editing /etc/lilo.conf. Your lilo configuration

will vary slightly. After saving, run lilo –v to install your settings. **Don’t just**

**modify an existing lilo entry; you may need it if your new kernel has bugs.**

image=/usr/src/Linux-x.x.x/arch/i386/boot/bzImage

label=”Linux-test”

root=/dev/hda5

read-only

10. Making a user test file. You also need to copy your edited unistd.h from

/usr/src/Linux- x.x.x/include/asm/ to /usr/include/kernel/ because it contains your

system call’s index.

#include <Linux/errno.h>

#include<sys/syscall.h>

#include <Linux/unistd.h>

long errno; //this is the return code from the system call

//this is a macro defined in unistd.h to help prototype sys calls

\_syscall2(int, myservice, int, arg1, char\*, arg2);

main() {

myservice(1, "hi");

}

11. Reboot into your new kernel and compile your user test program to try out

your system call. You will know if it worked if you see a kernel message in

/var/log/kernel/warnings announcing that your service is running.

**Result**

**The process of adding a new system call to Linux is studied**