KERNEL ARCHITECTURE

BLG413E – System Programming, Practice Session 2

Contents

- System Calls
- Kernel Modules

1-Adding a system call

Requirements (for Ubuntu OS): linux-source, kernel-package, fakeroot, libncurses5-dev (all of them are available in the provided Lubuntu 14.04 image)

Steps:

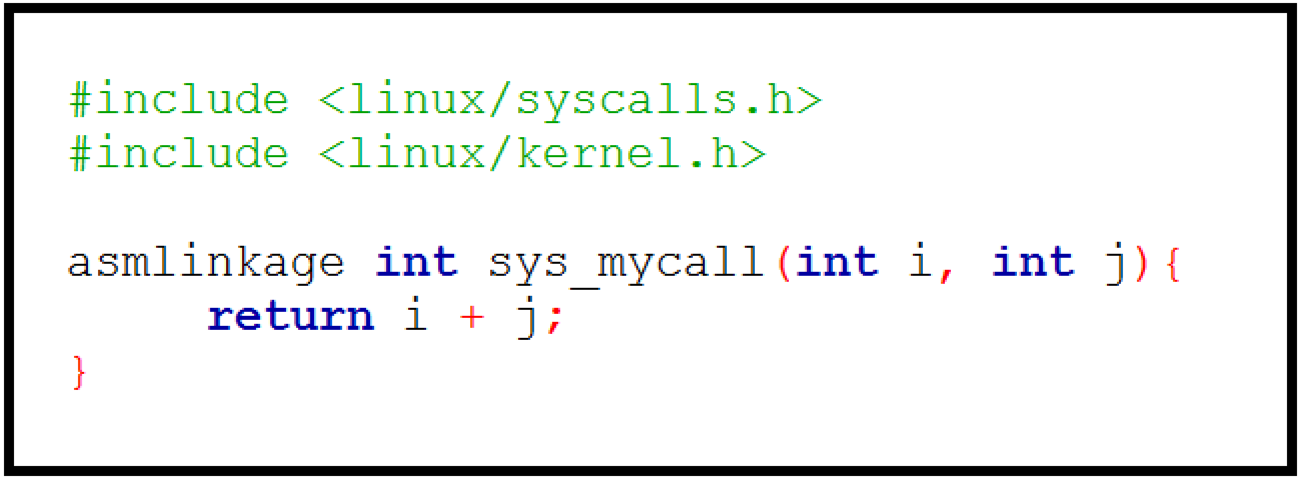
- extract linux source
- write new system call
- modify Makefiles
- modify system call table
- modify system call header file
- compile and install new kernel
- reboot to new kernel
- test new system call

Extracting linux source

- move linux source archive file (available in the provided Lubuntu 14.04 image) to desktop
  - cd Desktop
  - sudo mv /usr/src/linux-source-3.13.0/linux-source-3.13.0.tar.bz2 linux-source-3.13.0.tar.bz2
- and extract it
  - tar -xjvf linux-source-3.13.0.tar.bz2
- enter linux source folder
  - cd linux-source-3.13.0

Writing a system call

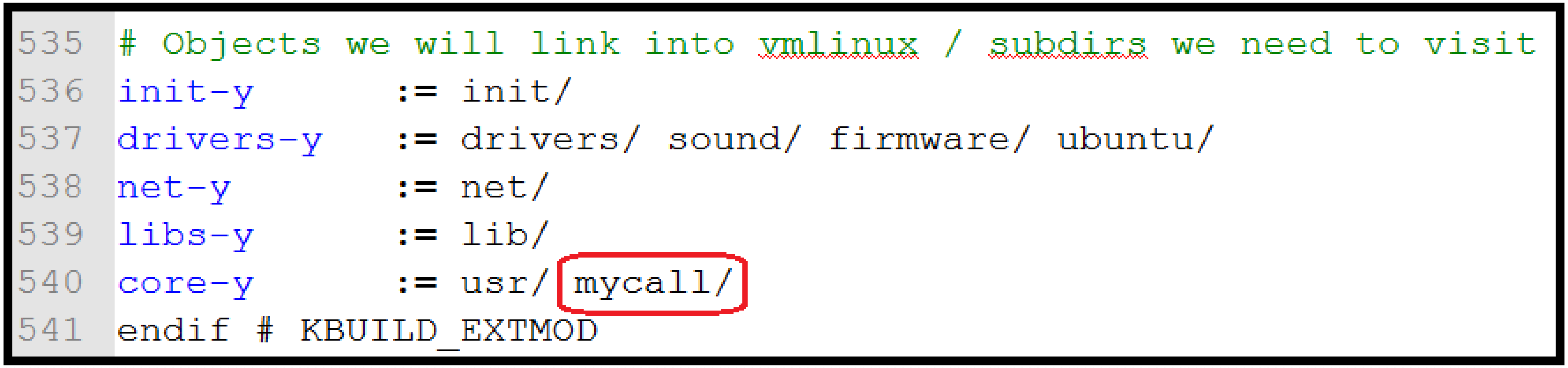
- mkdir mycall
- **mycall.c:** under /mycall



```c
#include <linux/syscalls.h>
#include <linux/kernel.h>

asmlinkage int sys_mycall(int i, int j){
     return i + j;
}
```

Modifying Makefiles

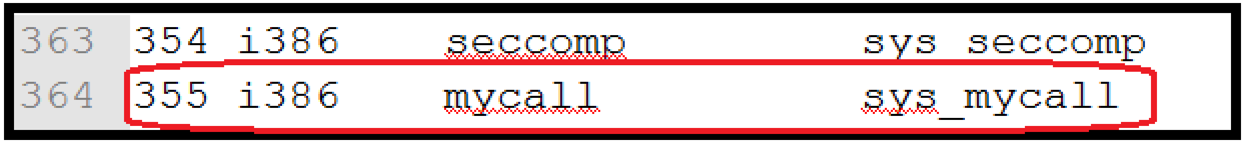
- create **Makefile** under /mycall
  - write "obj-y := mycall.o" into this file
- modify Makefile under /linux-source-3.13.0 by adding "mycall/" to core-y



```
# Objects we will link into vmlinux / subdirs we need to visit
init-y          := init/
drivers-y       := drivers/ sound/ firmware/ ubuntu/
net-y           := net/
libs-y          := lib/
core-y          := usr/ mycall/
endif # KBUILD_EXTMOD
```

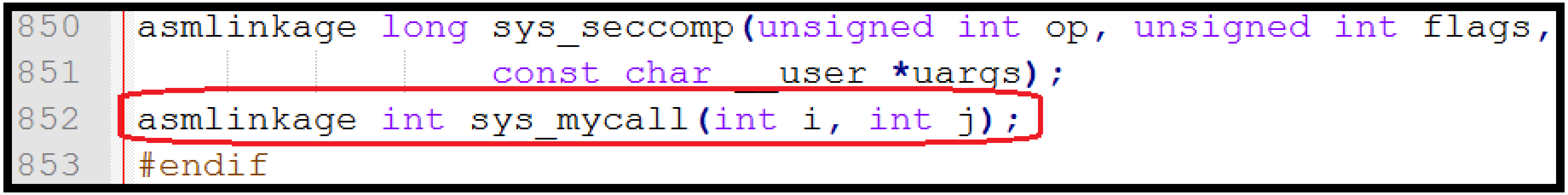
Modifying system call table and system call header files

- open arch/x86/syscalls/syscall_32.tbl
  - add "355 i386 mycall sys_mycall" to the end of file



```
354 i386        seccomp                 sys_seccomp
355 i386        mycall                  sys_mycall
```

- open include/linux/syscalls.h
  - add "asmlinkage int sys_mycall(int i, int j);" to the end of file before #endif



```
asmlinkage long sys_seccomp(unsigned int op, unsigned int flags,
                            const char __user *uargs);
asmlinkage int sys_mycall(int i, int j);
#endif
```

Compiling linux kernel

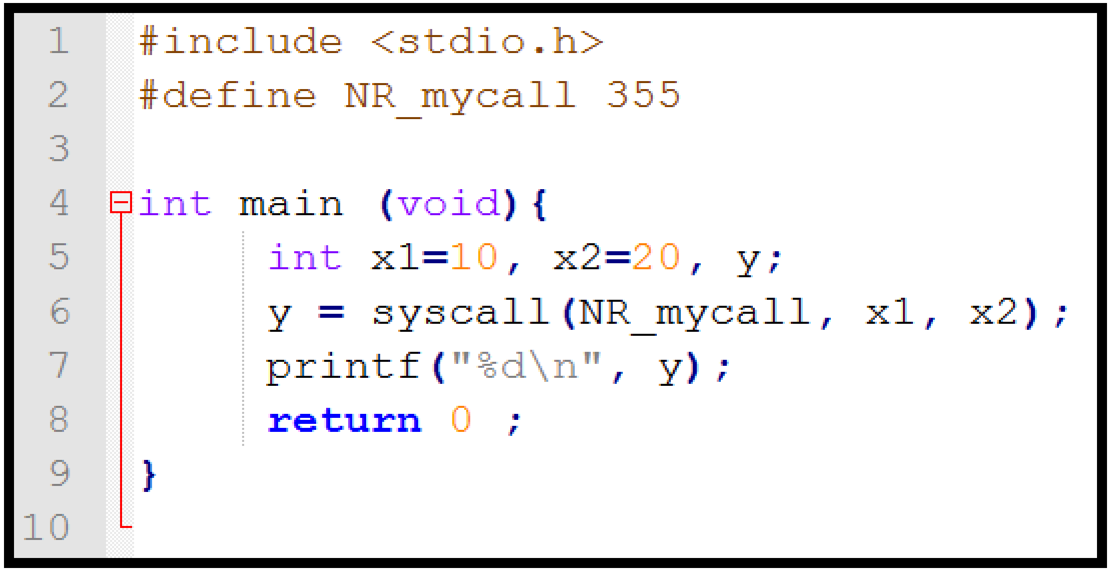
- make localmodconfig → include only the modules that are used in the current system
- make-kpkg clean → cleans up all from previous kernel compiles
- **Compilation (Warning: It may take 1-2 hours)**: fakeroot make-kpkg --initrd --append-to-version=-custom kernel_image kernel_headers
- **Output:** two files in parent directory (i.e., Desktop):
  - linux-image-3.13...deb
  - linux-headers-3.13...deb

Installing compiled kernel

- sudo dpkg -i linux-image-3.13...
- sudo dpkg -i linux-headers-3.13...
- Then reboot to open from the new kernel:
  - sudo reboot

Testing new system call

- A simple C program using our new system call to add 2 numbers and printing out the result



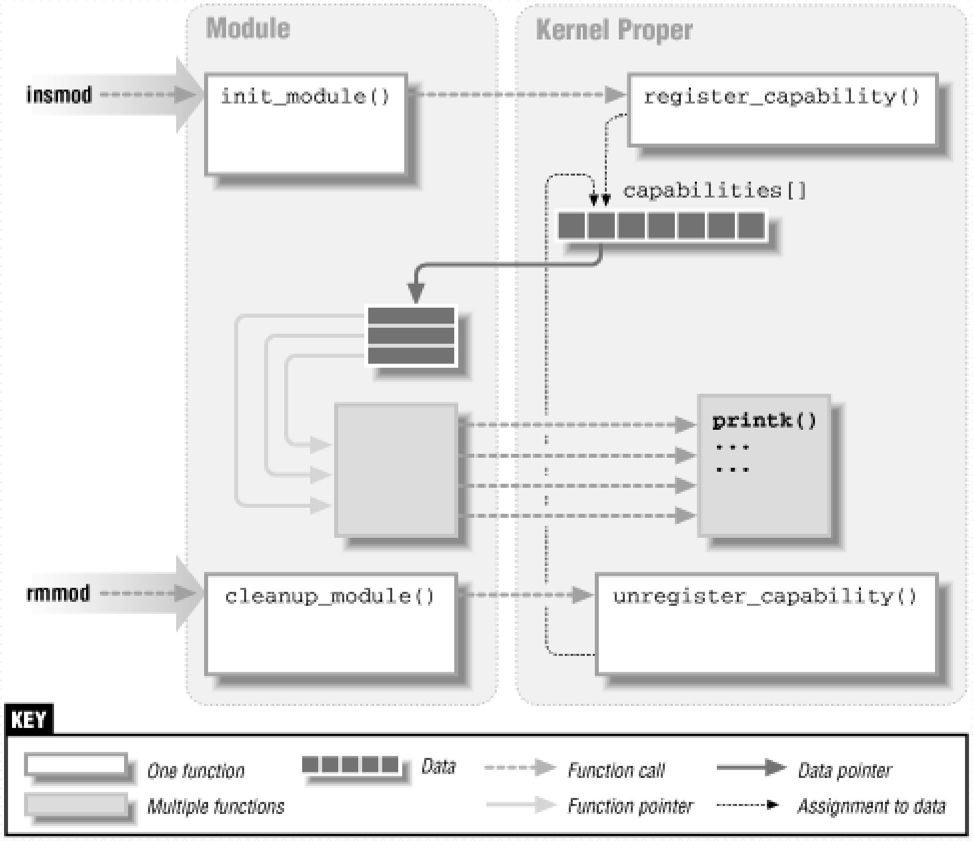
```c
#include <stdio.h>
#define NR_mycall 355

int main (void){
        int x1=10, x2=20, y;
        y = syscall(NR_mycall, x1, x2);
        printf("%d\n", y);
        return 0 ;
}

```

Uninstalling compiled kernel

- When you need to recompile the kernel, **first boot from the original kernel (from Advanced Options for Ubuntu in the boot menu)** and uninstall the kernel you have compiled before by using following commands
  - sudo dpkg -r linux-image-3.13...custom
  - sudo dpkg -r linux-headers-3.13...custom

2-Kernel modules

- A way to add new features to the kernel without rebuilding it.
- Unlike applications, modules register themselves for serving future requests.
- Applications can access the capabilities of a module through system calls.



http://www.xml.com/ldd/chapter/book/figs/ldr2_0201.gif

An example module: hello

- **hello.c:**

```
#include <linux/init.h> /* for module_init and module_exit */
#include <linux/module.h> /* needed by all modules */
MODULE_LICENSE("Dual BSD/GPL"); /* a macro to declare that this module is open source */

static int hello_init(void) /* static: unvisible outside the module */
{                           /* to avoid namespace pollution */
   printk(KERN_ALERT "Hello, world\n"); /* printk: kernel print function (macros for priority) */
   return 0;                            /* KERN_ALERT: a situation requiring immediate action */
}

static void hello_exit(void)
{
   printk(KERN_ALERT "Goodbye, cruel world\n");
}

module_init(hello_init);
module_exit(hello_exit);
```

- **Makefile:**

```
obj-m := hello.o
all:
     make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
```

M=$(PWD) is to build external module in the working directory

Using hello module

- **Compiling:**
  - *make*
- **Loading** (check with ***dmesg*** which is used to write the kernel messages):
  - *sudo insmod ./hello.ko*
- **Unloading** (check with ***dmesg***):
  - *sudo rmmod hello*
- check with ***lsmod*** (which prints the contents of the /proc/modules file) before and after loading and unloading

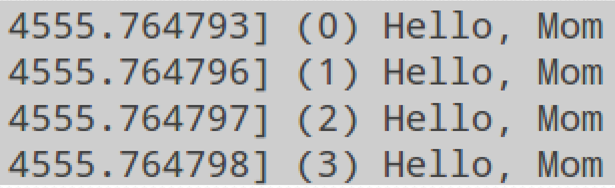
An example module using load time parameters

- **hellop.c:**

```
/* $Id: hellop.c,v 1.4 2004/09/26 07:02:43 gregkh Exp $  */
#include <linux/init.h>
#include <linux/module.h>
#include <linux/moduleparam.h> /* to enable passing parameters at loadtime */
MODULE_LICENSE("Dual BSD/GPL");

/* A couple of parameters that can be passed in: how many times we say hello, and to whom */
static char *whom = "world";
static int howmany = 1;
module_param(howmany, int, S_IRUGO); /* S_IRUGO: read by the world but cannot be changed */
module_param(whom, charp, S_IRUGO);

static int hello_init(void){
   int i;
   for (i = 0; i < howmany; i++)
      printk(KERN_ALERT "(%d) Hello, %s\n", i, whom);
   return 0;
}

static void hello_exit(void){
   printk(KERN_ALERT "Goodbye, cruel world\n");
}

module_init(hello_init);
module_exit(hello_exit);
```

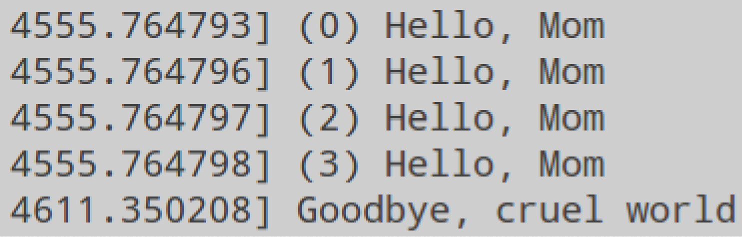
Specifying module parameters

- sudo insmod ./hellop.ko whom=‘Mom’ howmany=4
- dmesg



```
4555.764793] (0) Hello, Mom
4555.764796] (1) Hello, Mom
4555.764797] (2) Hello, Mom
4555.764798] (3) Hello, Mom
```

- sudo rmmod hellop
- dmesg



```
4555.764793] (0) Hello, Mom
4555.764796] (1) Hello, Mom
4555.764797] (2) Hello, Mom
4555.764798] (3) Hello, Mom
4611.350208] Goodbye, cruel world
```