IKO31204
Pemrograman Sistem
Jilid 6: Linux Kernel Module (II) & procfs, syscall

Fakultas Ilmu Komputer - Universitas Indonesia
Sep 2011

topik

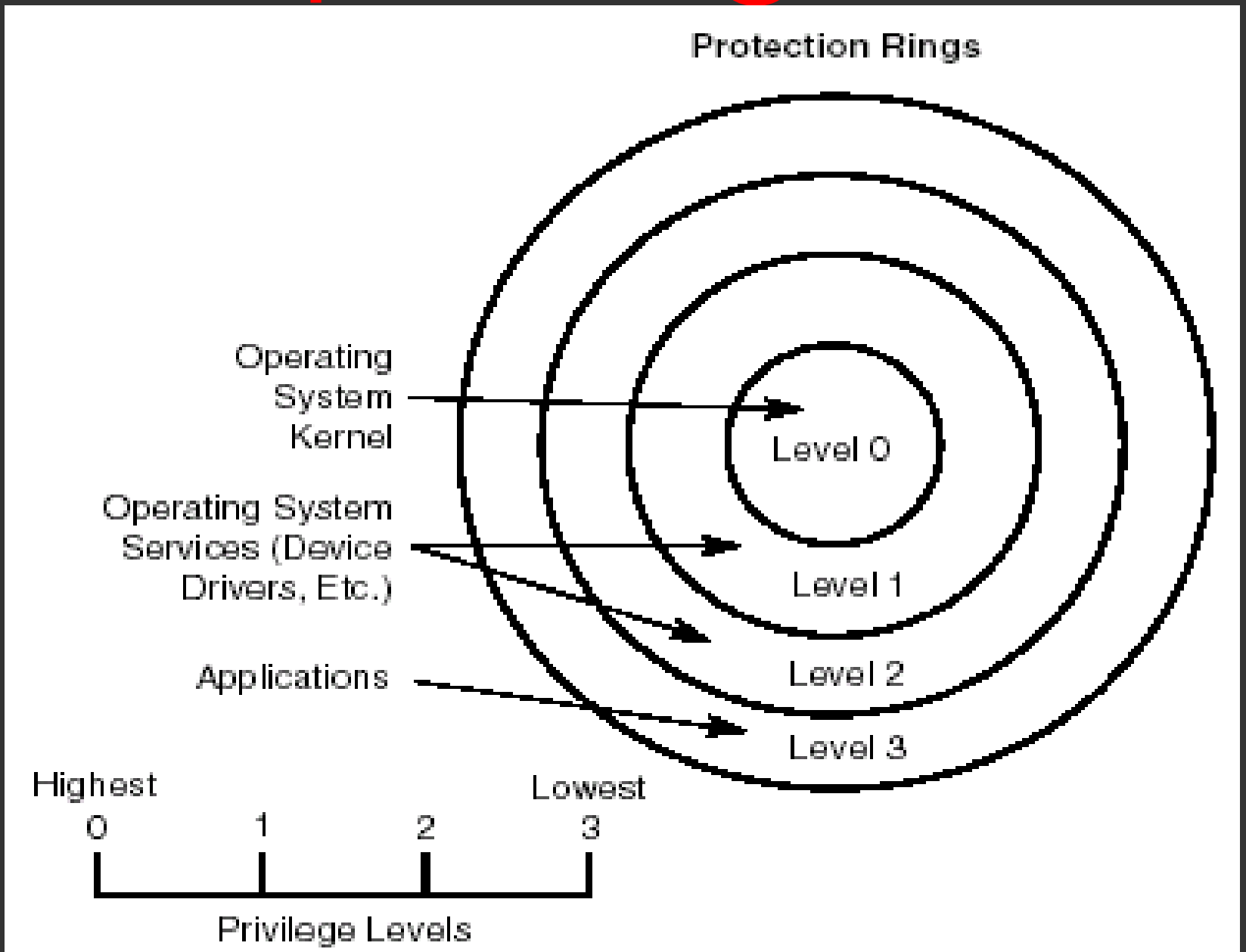
hello module
proc file system
syscall

referensi

The Linux Kernel Module Programming Guide (Google it, now !), Chapter 5

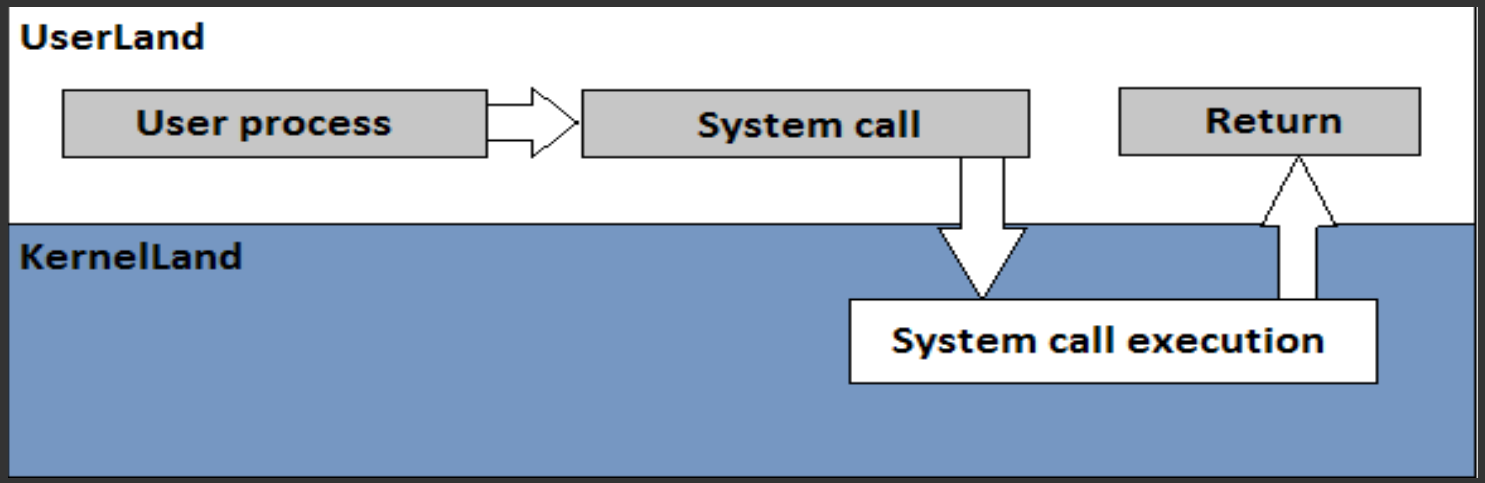
Implementing a System Call on Linux 2.6 for i386 (*Google it, now !*)

privilege level



module vs program

Item	module	program
entry	method init_module dan cleanup_module	method main()
functions / library	system calls, cth: /proc/kallsyms	standard C library, cth: printf (HINT: gunakan strace)
resource	ring 0	ring 3
code space	kernel space / kernel land memory (default 1/4)	user space / user land memory (default 3/4)



hello module

Linux Kernel Module Programming Guide, Ch 2

komponen module

fungsi esensial:
1. start function
2. end function

fungsi tambahan:
1. module parameter
2. module description

(HINT: pelajari dari cth pd buku referensi)

hello-1 module

```
/*
* hello–1.c – The simplest kernel module.
*/

#include <linux/module.h>           /* Needed by all modules */
#include <linux/kernel.h>        /* Needed for KERN_INFO */

int init_module(void)   {
      printk(KERN_INFO "Hello world 1.\n");
      /*    A non 0 return means init_module failed;
            module can't be loaded. */
      return 0;
}
void cleanup_module(void)   {
      printk(KERN_INFO "Goodbye world 1.\n");
}
```

hello-2 module

```
/*
* hello–2.c – Demonstrating the module_init() and module_exit() macros.
* This is preferred over using init_module() and cleanup_module().
*/
#include <linux/module.h>          /* Needed by all modules */
#include <linux/kernel.h>          /* Needed for KERN_INFO */
#include <linux/init.h>            /* Needed for the macros */

static int __init hello_2_init(void)   {
       printk(KERN_INFO "Hello, world 2\n");
       return 0;
}

static void __exit hello_2_exit(void)     {
       printk(KERN_INFO "Goodbye, world 2\n");
}

module_init(hello_2_init);
module_exit(hello_2_exit);
```

Makefile

```
obj-m += hello-1.o
obj-m += hello-2.o
all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

NB:

1. coba lihat cth Makefile pd module dlm kernel src
/usr/src/kernel-source-2.6.8/drivers/input/mouse/Makefile
/usr/src/kernel-source-2.6.8/net/ipv4/Makefile
dll
2. coba variasi2 samples hello pada Linux Kernel Programming Guide (Chapter 2)

Percobaan

mkdir percobaan
cd percobaan
vi hello-1.c
(copy paste seluruh hello-1.c di atas)

vi hello-2.c
(copy paste seluruh hello-2.c di atas)

vi Makefile
(copy paste seluruh isi Makefile di atas)

make
insmod hello-1.ko
insmod hello-2.ko

proc file system

(easy access to kernel)

Linux Kernel Module Programming Guide, Ch 5

procfs di dalam module

1. membuat file /proc/helloworld
   saat init_module()
2. memproses panggilan
   terhadap /proc/helloworld
   a. memproses READ, atau
   b. memproses WRITE
3. men-delete file /proc/helloworld
   saat cleanup_module()

membuat file /proc/helloworld (1)

```
-------------------- START FILE helloworld.c -------------------------
#include <linux/module.h>          /* Specifically, a module */
#include <linux/kernel.h>          /* We're doing kernel work */
#include <linux/proc_fs.h>         /* Necessary because we use the proc fs */

#define procfs_name "helloworld"

struct proc_dir_entry *Our_Proc_File;

int procfile_read(char *buffer, char **buffer_location, off_t offset, int buffer_length, int *eof,
void *data)    {

        int ret;
        printk(KERN_INFO "procfile_read (/proc/%s) called\n", procfs_name);

        if (offset > 0) {
                /* we have finished to read, return 0 */
                ret = 0;
        } else {
                /* fill the buffer, return the buffer size */
                ret = sprintf(buffer, "HelloWorld!\n");
        }
        return ret;
}
```

membuat file /proc/helloworld (1)

```
int init_module()    {
        Our_Proc_File = create_proc_entry(procfs_name, 0644, NULL);
        if (Our_Proc_File == NULL) {
                remove_proc_entry(procfs_name, &proc_root);
                printk(KERN_ALERT "Error: Could not initialize /proc/%s\n", procfs_name);
                return –ENOMEM;
        }
        Our_Proc_File–>read_proc          = procfile_read;
        Our_Proc_File–>owner         = THIS_MODULE;
        Our_Proc_File–>mode          = S_IFREG  |  S_IRUGO;
        Our_Proc_File–>uid                = 0;
        Our_Proc_File–>gid                = 0;
        Our_Proc_File–>size               = 37;
        printk(KERN_INFO "/proc/%s created\n", procfs_name);
        return 0; /* everything is ok */
}

void cleanup_module()       {
        remove_proc_entry(procfs_name, &proc_root);
        printk(KERN_INFO "/proc/%s removed\n", procfs_name);
}

-------------------- END FILE helloworld.c -------------------------
```

Percobaan

mkdir helloworld
cd helloworld
vi helloworld.c
(copy paste seluruh helloworld.c di atas)
vi Makefile
(buat Makefile untuk meng-compile module di atas)
make
insmod helloworld.ko
dmesg
(liat apa yang terjadi)
ls -al /proc/helloworld
cat /proc/helloworld

NOTE: coba buat procfs yang dapat di write

system call

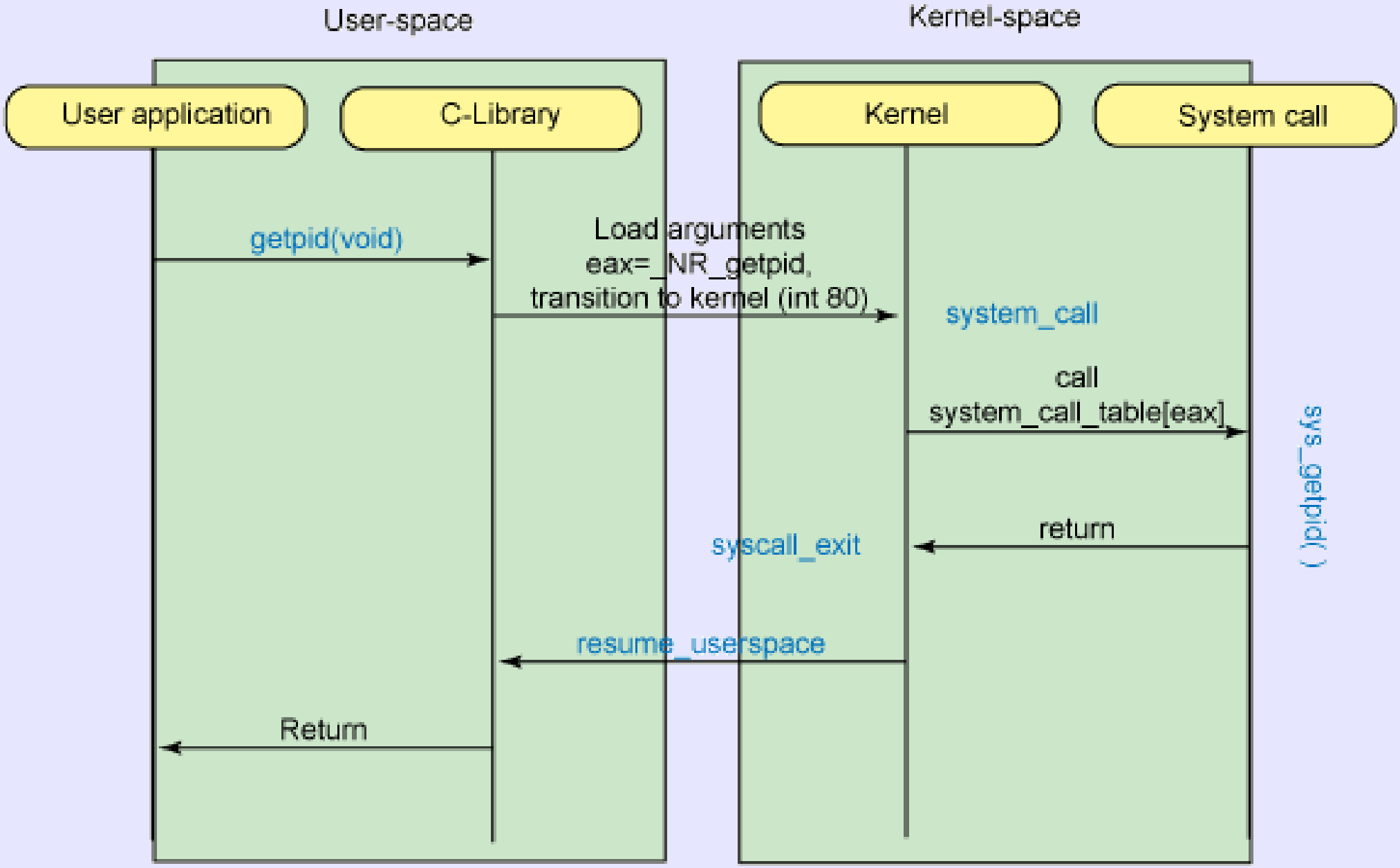
(user access to services in kernel space)

Implementing a System Call on Linux 2.6 for i386

serba serbi System Call

1. system call berjalan didalam kernel space
2. setiap system call punya nomor panggil
3. nomor panggil mjd identifier di-seantero kernel
4. ketika program user space memanggil sys-call, akan dijalankan routing sbb:
   a. trap interrupt 0x80 (interrupt user to kernel)
   b. pass nomor panggil & argumen
   c. kernel execute sys-call
   d. return hasil

cth: getpid()


User-space
Kernel-space
User application
C-Library
Kernel
System call
getpid(void)
Load arguments
eax=_NR_getpid,
transition to kernel (int 80)
system_call
call
system_call_table[eax]
sys_getpid( )
return
syscall_exit
resume_userspace
Return

Latihan

Membuat sebuah syscall "mycall"

Implementing a System Call on Linux 2.6 for i386

Perhatikan:

1. /usr/src/linux =>
   /usr/src/kernel-source-2.6.8
2. syscall_table.S => entry.S

tanya jawab