Project Report

@ CS353 Linux Kernel

Shen Jiyuan

5130309194

CataLog

Project 0　Project Requirement

1、Project I　Compile the Linux Kernel

- Compile a kernel and run it in the system.

2、Project II-A　Module Program

- Module 1 – load/unload the module can output some info
- Module 2 – module accepts a parameter (an integer)
  – load the module, output the parameter's value
- Module 3 – module creates a proc file, reading the proc file returns some info

3、Project II-B　Process Management

- Add ctx, a new member to task_struct to record the schedule in times of the process; when a task is scheduled in to run on a cpu, increase ctx of the process.
- Export ctx under /proc/xxx/ctx

4、Project III　Memory Management

- Write a module that is called mtest
- When module loaded, module will create a proc fs entry /proc/mtest
- /proc/mtest will accept 3 kinds of input:
  — 'listvma' will print all vma of current process in the format of start-addr end-addr permission e.g. 0x10000 0x20000 rwx
  — 'findpage addr' will find va->pa translation of address in current process's mm context and print it. If there is no va->pa translation, printk 'translation not found'.
  — 'writeval addr val' will change an unsigned long size content in current process's virtual address into val. Note module should write to identity mapping address of addr and verify it from userspace address addr.
- All the print can be done with printk and check result with dmesg.

4、Project IV　File System

- Practice 1: Change romfs code to hide a file/dir with special name
- Practice 2: Change the code of romfs to correctly read info of encrypted romfs
- Practice 3: Change the code of romfs to add 'x'(exe bit) for a special file

Project I   Compile the Linux Kernel

1、VMWare Workstation 11 Installation

Operating System: Windows 10.0 @ x64 processor
Processor: Intel Core i5 - 4200U
Step1. Get exe file from ‘http://www.vmware.com/products/workstation/’;
Step2. Double click the executable file and follow the instructions.

2、Ubuntu 15.10 Installation

Step1. Get iso file from http://www.ubuntu.com/download’;

Step2. Open VMWare Workstation you have installed, and by click ‘add a new virtual machine’ to install the Ubuntu in the workstation.

3、Compile Linux Kernel (version 4.4.4)

Step1. Download the kernel version 4.4.4 from ‘http://www.kernel.org’, notice here you should download the complete source code (not a patch or change log);

Step2. Copy to a directory’/usr/src’, and locate to this directory:

```
# cp linux-4.4.4.tar.gz /usr/src
# cd /usr/src
```

Step3. Extract the kernel:

```
# tar -xzvf linux-4.4.4.tar.gz
```

Step4. Preparation:

```
# cd /usr/src/linux
# make clean
# make mrproper
```

(Notice make clean and make mrproper can be passed with the first compilation.)

Step5. Configure the kernel:

```
# make menuconfig
```

/*create a menu where you can browse options on what the kernel supports*/

Step6. Customize the kernel: add support for NTFS file system from
‘File System>>DOS/FAT/NT/>>select NTFS file system support’.

Step7. Compile and Install the kernel (~3-4 hours!)

```
# make
# make modules_install
# make install
```

Step8. Update GRUB

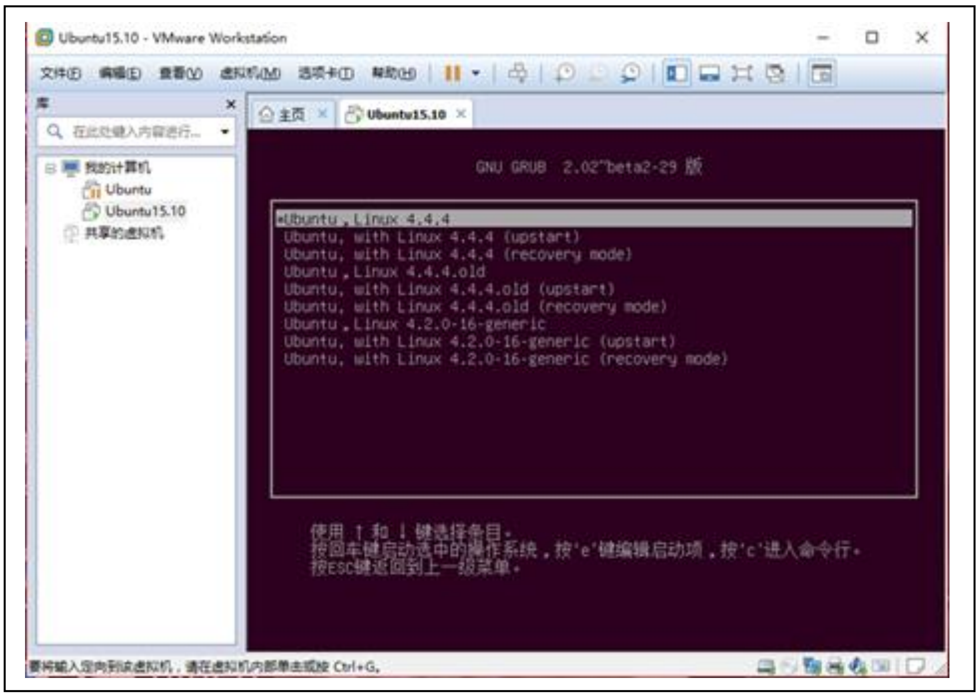
```
# cd /boot
# mkinitrd -o initrd.img-<kernel version>
```

Step9. Reboot the system

4、Result Checking

To check whether the new kernel has been successfully compiled or not, we can actually see the version by checking the grub when your Ubuntu starts. That is the following change should be done to assure the grub:

(1) Open the terminal and execute 'sudo gedit /etc/default/grub';
(2) Change in Line 5: originally it is 'GRUB_HIDDEN_TIMEOUT=0', to any >0 value, save the change and exit;
(3) Execute 'sudo update-grub';
(4) Reboot the Ubuntu and press 'shift' or 'esc' when it starts.
(5) You will see the grub as follows:


Ubuntu15.10 - VMware Workstation
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
库
在此处键入内容进行...
我的计算机
Ubuntu
Ubuntu15.10
共享的虚拟机
主页
Ubuntu15.10

```
GNU GRUB  2.02~beta2-29 版

*Ubuntu，Linux 4.4.4
 Ubuntu, with Linux 4.4.4 (upstart)
 Ubuntu, with Linux 4.4.4 (recovery mode)
 Ubuntu，Linux 4.4.4.old
 Ubuntu, with Linux 4.4.4.old (upstart)
 Ubuntu, with Linux 4.4.4.old (recovery mode)
 Ubuntu，Linux 4.2.0-16-generic
 Ubuntu, with Linux 4.2.0-16-generic (upstart)
 Ubuntu, with Linux 4.2.0-16-generic (recovery mode)

使用 ↑ 和 ↓ 键选择条目。
按回车键启动选中的操作系统，按'e'键编辑启动项，按'c'进入命令行。
按ESC键返回到上一级菜单。
```

要将输入定向到该虚拟机，请在虚拟机内部单击或按 Ctrl+G。

4、Problems and Solutions

Pro1. When I use my original VMware and Ubuntu, I always come with the error that says there is no more disk space?

Solution: After trying various methods from the internet, the error is still not solved, then I install a new VMware and Ubuntu.

Pro2. How to operate the 'menuconfig'.

Solution: Follow the principle that 'do not change anything if you are not sure about your operation'. There are three choices for user: first, <*> or [*] stands for 'this function is compiled into kernel'; [] stands for 'this function not compiled into kernel'; [M] stands for 'this function can be dynamically compiled into kernel'.

Project II-A   Module program

1、Module1 – load/unload the module can output some info

According to the requirement, we need to print some information when we load a module and also print some information when we unload it. That means in the function which is called when module_init should be printk something, and same for the function which is called when module_exit. Therefore, by following the class ppt, this program is not hard:

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>

static int __init hello_init(void){
printk("Name: ShenJIyuan.Hello~\n");
return 0;
}

static void __exit hello_exit(void){
printk("Student_No.:5130309194.Bye~\n");
}

module_init(hello_init);
module_exit(hello_exit);
MODULE_LICENSE("GPL");
```

- After naming the file as 'hello.c' and writing a Makefile (discussed later), test it:

```
# make
# sudo insmod hello.ko
# dmesg
#sudo rmmod hello.ko
# dmesg
```

2、Module2 – module accepts a parameter (an integer)

– load the module, output the parameter's value

According to the requirement, we need to get the input (here it's an integer) from user when a module is loaded and output this value when unload the module. That means in the function which is called when module_init should be put something into an integer value, and printk this value for the function which is called when module_exit. Therefore, by following the class ppt involving parameter programming, this program is also not hard: pay attention to the head files that are included should add a '/linux/moduleparam.h'.

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/moduleparam.h>

static int test;
module_param(test, int, 0644);

void hello_foo(void)
{
printk("Hello\n");
}

EXPORT_SYMBOL(hello_foo);

static int __init hello_init(void)
{
printk(KERN_INFO "Hello parameter world\n");
printk(KERN_INFO "Params:test:%d;\n",test);
return 0;
}

static void __exit hello_exit(void)
{
printk(KERN_INFO "Goodbye parameter world\n");
}

MODULE_LICENSE("GPL");
module_init(hello_init);
module_exit(hello_exit);
```

- After naming the file as 'para.c' and writing a Makefile (discussed later), test it:

```
# make
# sudo insmod para.ko test=6
# dmesg
# sudo rmmod para.ko
# dmesg
```

3、Module3 – module creates a proc file

– reading the proc file returns some info

This program needs to deal with proc file operations including creating, read trigger and removing. For reading, here I use ‘proc_create’, and pay attention to the defined file operations for this proc. Then when the module is loaded, a proc file is created by the user, and it will do read operation and ‘printk’ the checking situation of proc creation. And when the module is unloaded, the proc will be removed.

- Module loaded, create a proc

```
static int __init hello_init(void){
        struct proc_dir_entry* entry;
        entry = proc_create("hello", 0, NULL, &hello_proc_fops);
        if (!entry){
                printk(KERN_INFO"Creating proc_read_entry failed!\n");
                return -1; }
        else {
                printk(KERN_INFO"Creating proc_read_entry finished!\n");
                return 0;}
}
```

- Define proc file operations

```
static const struct file_operations hello_proc_fops = {
        .owner = THIS_MODULE,
        .open = hello_proc_open,
        .read = seq_read
};
```

- Implement the open proc file operations

```
static int hello_proc_show(struct seq_file* m, void* v){
        seq_printf(m, "This is a test for proc file!\n");
        return 0;
}
static int hello_proc_open(struct inode* inode, struct file* file){
        return single_open(file, hello_proc_show, NULL);
};
```

```
static void __exit hello_exit(void) {
        remove_proc_entry("hello", NULL);
        printk(KERN_INFO"Removing proc_read_entry finished!\n");
}
```

- After naming the file as ‘proc.c’ and writing a Makefile (discussed later), test it:

```
# make
# sudo insmod proc.ko
# cat /proc/modules | grep proc        /*you can find ‘hello’ file in the /proc*/
# dmesg
# sudo rmmod proc.ko
# dmesg
```

4、 Makefile

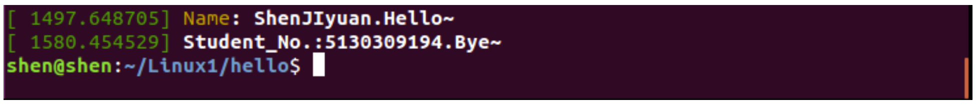
All the three small module programming are using a same Makefile, and by following the class ppt I write this Makefile as:

```
obj-m := proc.o
      KDIR := /lib/modules/$(shell uname -r)/build
      PWD := $(shell pwd)
all:
      make -C $(KDIR) M=$(PWD) modules
clean:
      rm *.o *.ko *.mod.c Module.symvers modules.order -f
```

5、 Result Checking

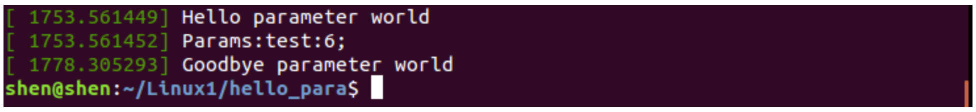
By following the test methods mentioned above, the result can be got:

- Module 1 — simple load/unload printk



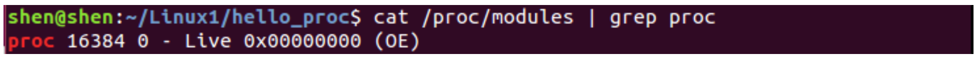
```
[ 1497.648705] Name: ShenJIyuan.Hello~
[ 1580.454529] Student_No.:5130309194.Bye~
shen@shen:~/Linux1/hello$ 
```

- Module 2 — printk input parameter

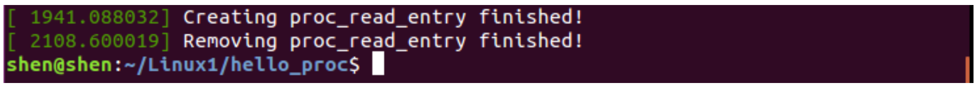


```
[ 1753.561449] Hello parameter world
[ 1753.561452] Params:test:6;
[ 1778.305293] Goodbye parameter world
shen@shen:~/Linux1/hello_para$ 
```

- Module 3 — proc file operation



```
shen@shen:~/Linux1/hello_proc$ cat /proc/modules | grep proc
proc 16384 0 - Live 0x00000000 (OE)
```



```
[ 1941.088032] Creating proc_read_entry finished!
[ 2108.600019] Removing proc_read_entry finished!
shen@shen:~/Linux1/hello_proc$ 
```

Project II-B   Process Management

1、How to implement

Step1. Change file ‘include/linux/sched.h’:

— add ‘int ctx;’ to ‘struct task_struct’.

Step2. Change file ‘kernel/sched/core.c’

— increase ‘task_struct->ctx’ by 1, in function ‘__schedule’

Step3. Change file ‘kernel/fork.c’:

— initiate ‘p->ctx=0;’ to function ‘long _do_fork’.
Notice here need to add it exactly after ‘p’ is assigned.

Step4. Change file ‘fs/exec.c’:

— initiate ‘tsk->ctx=0;’ to function ‘static int exec_mmap’;

Next steps refer to the implementation of ‘statm’

Step5. Change file ‘/fs/proc/internal.h’:

— define a function by referring to the function ‘proc_pid_state’:

```
int proc_pid_ctx(struct seq_file *m, struct pid_namespace *ns,
                    struct pid *pid, struct task_struct *task)
{
        seq_printf(m,"%d\n",task->ctx);
        return 0;
}
```

Step6. Change file ’fs/proc/base.c’:

— add ‘ONE(“ctx”,S_IRUGO,proc_pid_ctx),’ in thread groups
(can search for ’statm’ and add it properly to a near line)

Step7. Remake the kernel:

```
# make clean
# make mrproper
# make oldconfig
# make
# make modules_install
# make install
```

2、How to test (refer to the ‘experiment manual’)

(1) write following codes into file ‘block.c’:

```
# include <stdio.h>
int main () {
   while (1)   getchar();
   return 0;
}
```

(2) compile it to exe file 'block' by

```
# gcc block.c -o block -Wall
```

(3) open a terminal and execute

```
# ./block
```

(4) open another terminal and execute

```
# ps -e | grep block
```

(5) find the number corresponding to 'block' (assume it is 7711):

```
# cd /proc/7711
cat ctx
```

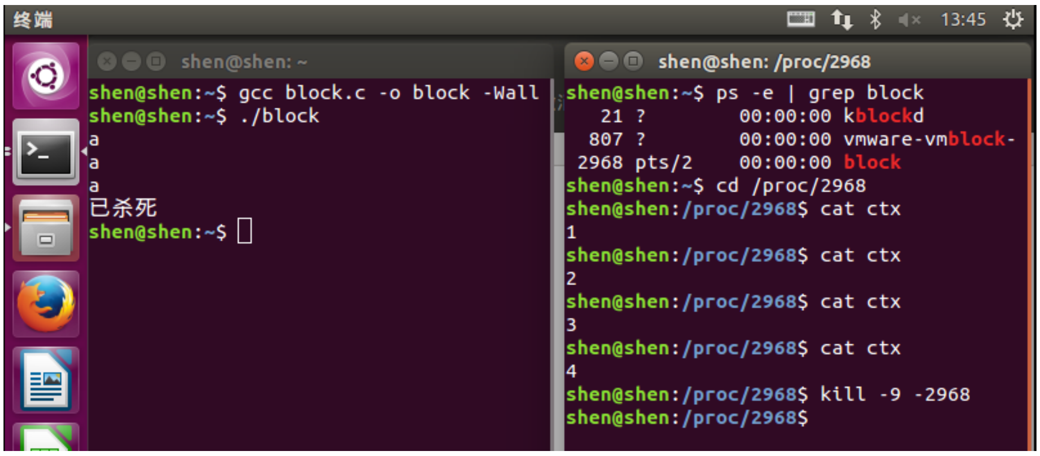
Keep put 'a' in the first terminal, and 'cat ctx' in the second terminal, then you can observe the 'ctx' increase.

---

(6) terminate the process

```
# kill -9 -7711
```

3、Result Checking



```
shen@shen:~$ gcc block.c -o block -Wall
shen@shen:~$ ./block
a
a
a
已杀死
shen@shen:~$ 
```

```
shen@shen:~$ ps -e | grep block
   21 ?        00:00:00 kblockd
  807 ?        00:00:00 vmware-vmblock-
 2968 pts/2    00:00:00 block
shen@shen:~$ cd /proc/2968
shen@shen:/proc/2968$ cat ctx
1
shen@shen:/proc/2968$ cat ctx
2
shen@shen:/proc/2968$ cat ctx
3
shen@shen:/proc/2968$ cat ctx
4
shen@shen:/proc/2968$ kill -9 -2968
shen@shen:/proc/2968$
```

4、Problem and Solution

Pro1. It is hard to find the lines or functions that I need to place my lines to, the how can I find where a variable or function is defined or mentioned?

Solution: Here is a website: http://lxr.free-electrons.com/

Project III   Memory Management

1、How to implement

According to the requirement, we need to write a proc module where three kinds of input are accepted: (1) ‘listvma’ prints all vma of current process; (2) ‘findpage addr’ finds va->pa translation in current process’s mm context and prints it; (3) ‘writeval addr val’ changes the content of current process’s va into val. And considering for a whole framework, I define the file operation in the module as a function that will analyze the input and execute corresponding operations by calling each function. (Since proc creating and removing have been discussed in ProjectII-A, here omitted)

Step1. Define file operation

```
static const struct file_operations mtest_proc_fops = {
      .write    =    mtest_proc_write
};
```

Step2. Implement the main function that analyze inputs (mtest_proc_write)

```
if (memcmp(buf, "listvma", 7) == 0)                                /*listvma*/
            mtest_list_vma();

else if (memcmp(buf, "findpage", 8) == 0){                         /*findpage*/
        if (sscanf(buf + 8, "%lx", &val) == 1)
            mtest_find_page(val);
}

else if (memcmp(buf, "writeval", 8) == 0){                         /*writeval*/
        if (sscanf(buf + 8, "%lx %lx", &val, &val2) == 2)
            mtest_write_val(val, val2);
}

else
        printk("No such command. Nothing to be done.\n");
```

Step3. Implement for ‘listvma’.

Here we should first lock current mm and do a traversal to print info.

```
down_read(&mm->mmap_sem);                                              //lock
for (vma = mm->mmap;vma; vma = vma->vm_next) {
    printk("VMA 0x%lx-0x%lx ", vma->vm_start, vma->vm_end);
    if (vma->vm_flags & VM_READ)      printk("r");     else    printk("-");
    if (vma->vm_flags & VM_WRITE)     printk("w");     else    printk("-");
    if (vma->vm_flags & VM_EXEC)      printk("x");     else    printk("-");     printk("\n");
    }
up_read(&mm->mmap_sem);                                                //release lock
```

Step4. Implement for 'findpage'.

```
pte = pte_offset_map_lock(mm, pmd, addr, &ptl);
if (!pte)     return NULL;
if (!pte_present(*pte)){
        pte_unmap_unlock(pte, ptl);
        return NULL; }

page = pfn_to_page(pte_pfn(*pte));
if (!page){
        pte_unmap_unlock(pte, ptl);
        return NULL; }

get_page(page);
```

Step5. Implement for 'writeval'

```
page = mtest_seek_page(vma, addr);
kernel_addr = (unsigned long)page_address(page);
kernel_addr += (addr&~PAGE_MASK);
*(unsigned long *)kernel_addr = val;
```

2、 How to test

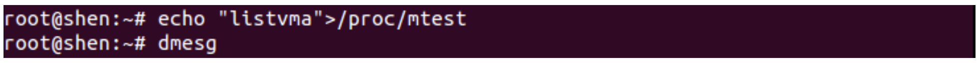
(1) echo "listvma">/proc/mtest
(2) echo "findpage0xb7787000">/proc/mtest
(3) echo "writeval0xb7787000 123456">/proc/mtest

3、 Result Checking

==

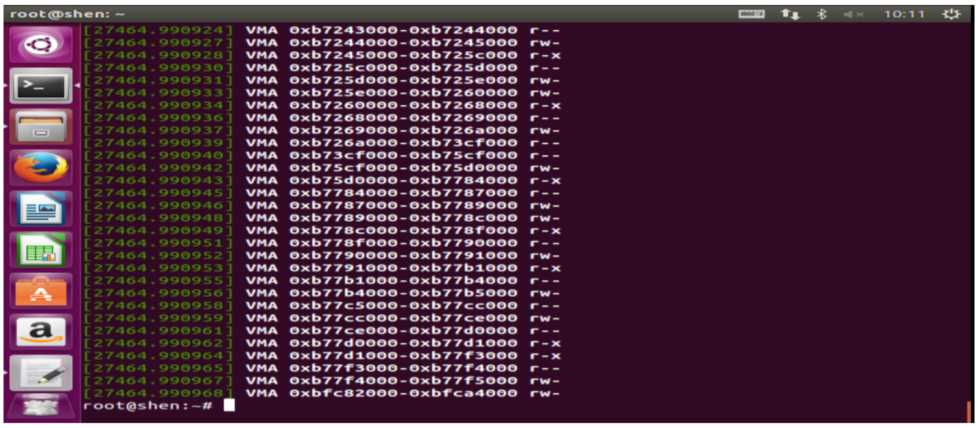
- for 'listvma'

  (1) command



```
root@shen:~# echo "listvma">/proc/mtest
root@shen:~# dmesg
```

  (2) result

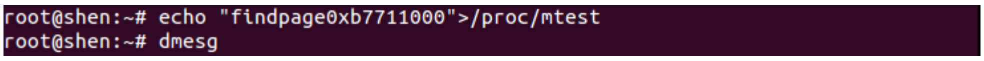


```
root@shen: ~
[27464.990924] VMA 0xb7243000-0xb7244000 r--
[27464.990927] VMA 0xb7244000-0xb7245000 rw-
[27464.990928] VMA 0xb7245000-0xb725c000 r-x
[27464.990930] VMA 0xb725c000-0xb725d000 r--
[27464.990931] VMA 0xb725d000-0xb725e000 rw-
[27464.990933] VMA 0xb725e000-0xb7260000 rw-
[27464.990934] VMA 0xb7260000-0xb7268000 r-x
[27464.990936] VMA 0xb7268000-0xb7269000 r--
[27464.990937] VMA 0xb7269000-0xb726a000 rw-
[27464.990939] VMA 0xb726a000-0xb73cf000 r--
[27464.990940] VMA 0xb73cf000-0xb75cf000 r--
[27464.990942] VMA 0xb75cf000-0xb75d0000 rw-
[27464.990943] VMA 0xb75d0000-0xb7784000 r-x
[27464.990945] VMA 0xb7784000-0xb7787000 r--
[27464.990946] VMA 0xb7787000-0xb7789000 rw-
[27464.990948] VMA 0xb7789000-0xb778c000 rw-
[27464.990949] VMA 0xb778c000-0xb778f000 r-x
[27464.990951] VMA 0xb778f000-0xb7790000 r--
[27464.990952] VMA 0xb7790000-0xb7791000 rw-
[27464.990953] VMA 0xb7791000-0xb77b1000 r-x
[27464.990955] VMA 0xb77b1000-0xb77b4000 r--
[27464.990956] VMA 0xb77b4000-0xb77b5000 rw-
[27464.990958] VMA 0xb77c5000-0xb77cc000 r--
[27464.990959] VMA 0xb77cc000-0xb77ce000 rw-
[27464.990961] VMA 0xb77ce000-0xb77d0000 r--
[27464.990962] VMA 0xb77d0000-0xb77d1000 r-x
[27464.990964] VMA 0xb77d1000-0xb77f3000 r-x
[27464.990965] VMA 0xb77f3000-0xb77f4000 r--
[27464.990967] VMA 0xb77f4000-0xb77f5000 rw-
[27464.990968] VMA 0xbfc82000-0xbfca4000 rw-
root@shen:~#
```

==

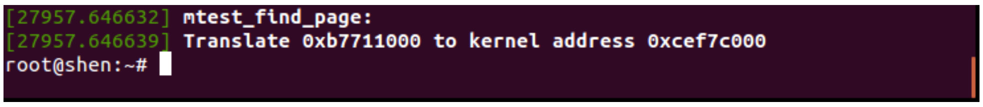
- for ‘findpage’

  (1) command



```
root@shen:~# echo "findpage0xb7711000">/proc/mtest
root@shen:~# dmesg
```

  (2) result

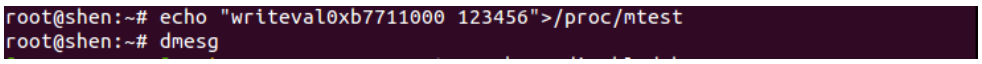


```
[27957.646632] mtest_find_page:
[27957.646639] Translate 0xb7711000 to kernel address 0xcef7c000
root@shen:~#
```

==

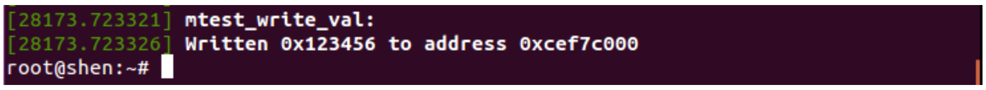
- for ‘findpage’

  (1) command



```
root@shen:~# echo "writeval0xb7711000 123456">/proc/mtest
root@shen:~# dmesg
```

  (2) result



```
[28173.723321] mtest_write_val:
[28173.723326] Written 0x123456 to address 0xcef7c000
root@shen:~#
```

==

Project IV　File System

1、Img File Creating

1、Create a folder: file{aa, bb, ft, fo[aa]}
2、Install tool ‘genromfs’: $ sudo apt-get install genromfs
3、Generate the img file: $ genromfs -f test.img -d file
Then you can see ‘test.img’ in your /home directory.

2、How to implement

- Hide a file with a special name

Change the function ‘romfs_readdir()’ in file ‘super.c’, since this function is used to read the files of a directory. In this function, by checking the file name we can successfully hide files with a specified name.

```
If (memcmp(fsname,spec,num)==0)    goto out;
```

- Encrypt specified files

Change the function ‘romfs_readpage()’ in file ‘super.c’, since this function is used to read the content of a file. Then we can simply change what we have read from this file.

```
encryptFile(buf, fillsize);
```

Add the definitions of called function before ‘readpage’ function:

```
void encryptFile(char* buf, unsigned long size){
        int I;
        for (i=0; i<size; i++)
              if (buf[i]!=’\n’ && buf[i]!=’\r’)
                     buf[i]=buf[i]+2;
}
```

- Add ‘x’ to a specified file

Change the function ‘romfs_iget()’ in file ‘super.c’, since this function is used to query inode.

```
If (name_len>=num && memcpy(fsname, spec, num)==0) mode |= S|IXUGO;
```

3、How to test

- Create an empty file in /home named ‘mnt’. (mounted directory)
- For HIDING

```
$ cd romfs
$ sudo insmod romfs.ko hide_file_name=”aa”
$ sudo mount -o loop ../test.img ../mnt
$ ls ../mnt
```

- For ENCRYPTION

```
$ cd romfs
$ sudo insmod romfs.ko encry_file_name="aa"
$ sudo mount -o loop ../test.img ../mnt
$ cat ../mnt/aa
```

- For ADDEXE

```
$ cd romfs
$ sudo insmod romfs.ko addex_file_name="aa"
$ sudo mount -o loop ../test.img ../mnt
$ ls -l ../mnt/aa
```

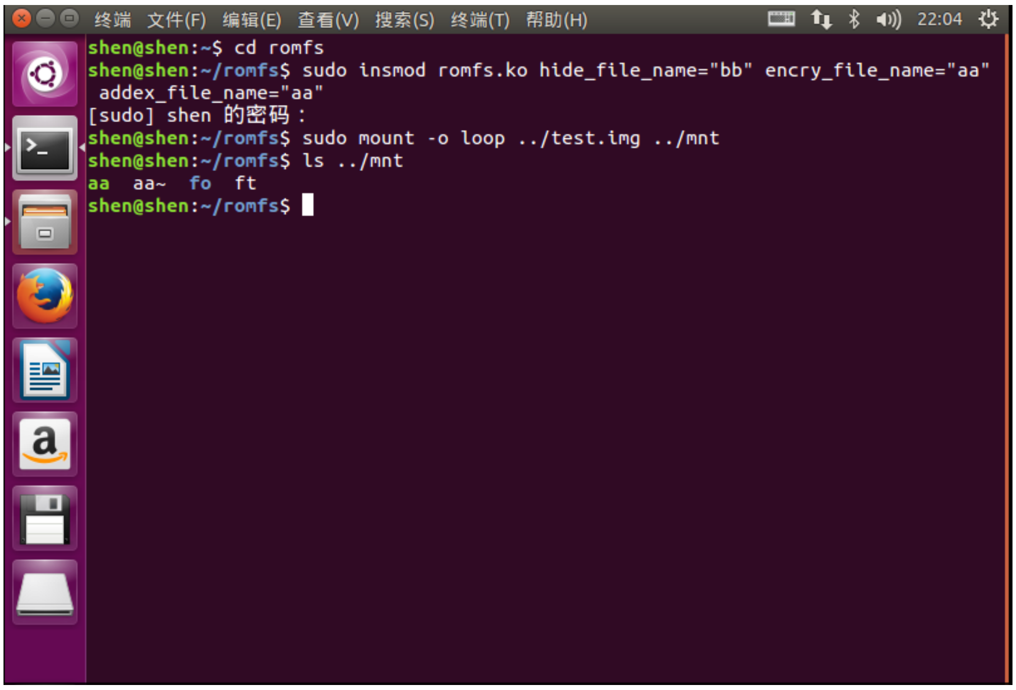
- ALL TEST

We can take all parameters in one insmod action.

```
$ cd romfs
$ sudo insmod romfs.ko hide_file_name="bb" encry_file_name="aa"
addex_file_name="aa"
$ sudo mount -o loop ../test.img ../mnt
$ ls ../mnt
$ cat ../mnt/aa
$ls -l ../mnt/aa
```

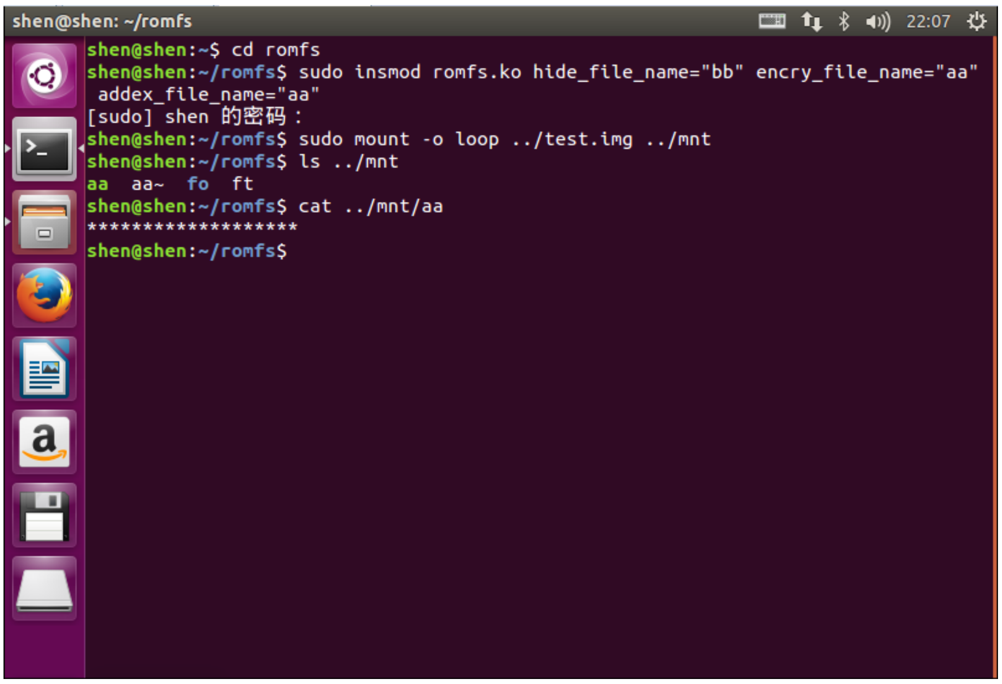
4、 Result Checking

(1) Hide "bb"



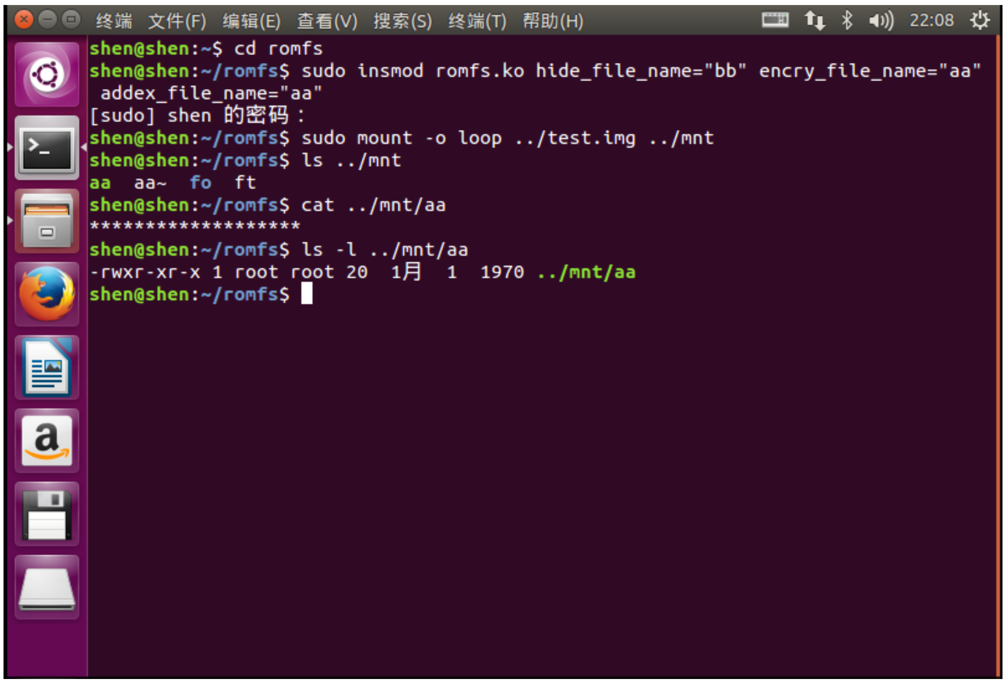
```
shen@shen:~$ cd romfs
shen@shen:~/romfs$ sudo insmod romfs.ko hide_file_name="bb" encry_file_name="aa"
 addex_file_name="aa"
[sudo] shen 的密码：
shen@shen:~/romfs$ sudo mount -o loop ../test.img ../mnt
shen@shen:~/romfs$ ls ../mnt
aa  aa~  fo  ft
shen@shen:~/romfs$
```

(2) Encrypt “aa”



```
shen@shen:~$ cd romfs
shen@shen:~/romfs$ sudo insmod romfs.ko hide_file_name="bb" encry_file_name="aa"
 addex_file_name="aa"
[sudo] shen 的密码：
shen@shen:~/romfs$ sudo mount -o loop ../test.img ../mnt
shen@shen:~/romfs$ ls ../mnt
aa  aa~  fo  ft
shen@shen:~/romfs$ cat ../mnt/aa
*******************
shen@shen:~/romfs$
```

(3) AddEX “aa”



```
shen@shen:~$ cd romfs
shen@shen:~/romfs$ sudo insmod romfs.ko hide_file_name="bb" encry_file_name="aa"
 addex_file_name="aa"
[sudo] shen 的密码：
shen@shen:~/romfs$ sudo mount -o loop ../test.img ../mnt
shen@shen:~/romfs$ ls ../mnt
aa  aa~  fo  ft
shen@shen:~/romfs$ cat ../mnt/aa
*******************
shen@shen:~/romfs$ ls -l ../mnt/aa
-rwxr-xr-x 1 root root 20  1月  1  1970 ../mnt/aa
shen@shen:~/romfs$ 
```