

操作系统实验一

实验报告

57119101 王晨阳

2021年7月22日

实验目的

实验1 Linux 内核代码分析

实验2：新增系统调用

实验3: Linux进程管理及其扩展

实验总结

实验目的

熟悉 Linux 内核文件结构，学会修改与增加系统调用，掌握进程控制和进程管理相关内容。

实验1 Linux 内核代码分析

依次执行命令

```
1  cd Desktop
2  tar zxvf linux-2.6.21.tar.gz
3  cd linux-2.6.21
4  make mrproper
5  cp /boot/config-2.6.21-1.3194.fc7 ./config
6  make oldconfig
7  make all
8  su
9  make modules_install
10 make install
11 make headers_install
12 vi /boot/grub/menu.lst
```

将

```
1  hiddenmenu
```

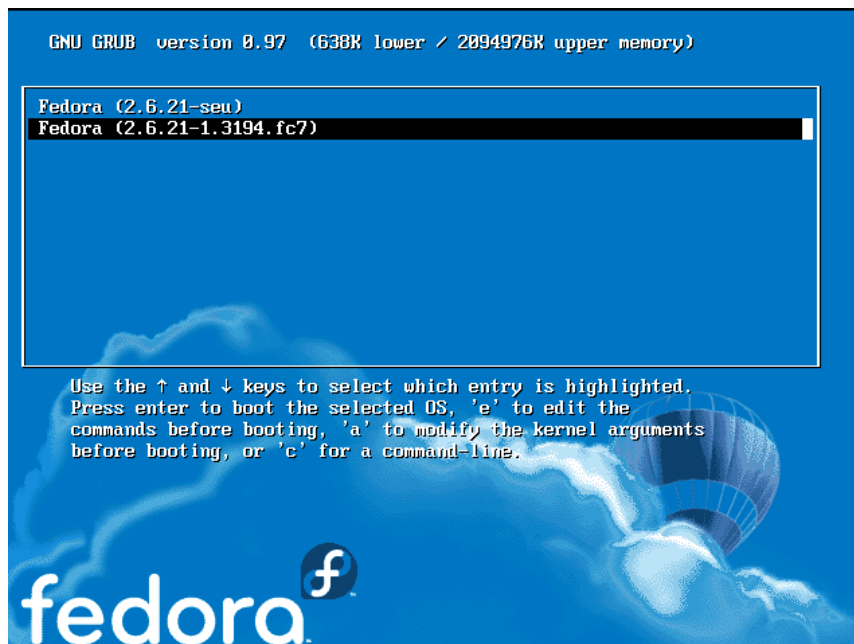
注释掉，如下图所示。

```
seu@localhost:/boot/grub
File Edit View Terminal Tabs Help
[root@localhost grub]# cat /boot/grub/menu.lst
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE: You have a /boot partition. This means that
#         all kernel and initrd paths are relative to /boot/, eg.
#         root (hd0,0)
#         kernel /vmlinuz-version ro root=/dev/sda2
#         initrd /initrd-version.img
#boot=/dev/sda
default=1
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
#hiddenmenu
title Fedora (2.6.21-seu)
    root (hd0,0)
    kernel /vmlinuz-2.6.21-seu ro root=LABEL=/ rhgb quiet
    initrd /initrd-2.6.21-seu.img
title Fedora (2.6.21-1.3194.fc7)
    root (hd0,0)
    kernel /vmlinuz-2.6.21-1.3194.fc7 ro root=LABEL=/ rhgb quiet
    initrd /initrd-2.6.21-1.3194.fc7.img
[root@localhost grub]#
```

重启

1 reboot

看到引导菜单。



选择 seu，系统正常启动。

```
seu@localhost:~/Desktop/linux-2.6.21
File Edit View Terminal Tabs Help
[seu@localhost linux-2.6.21]$ uname -r
2.6.21-seu
[seu@localhost linux-2.6.21]$
```

实验2：新增系统调用

./arch/i386/kernel/syscall_table.S 最后添加一个系统调用

```
1 .long sys_psta
```

```
.long sys_tee          /* 315 */
.long sys_vmsplice
.long sys_move_pages
.long sys_getcpu
.long sys_epoll_pwait
.long sys_psta         /* 320 */
```

./include/linux/psta.h 编辑内容为

```
1 #ifndef _LINUX_PSTA_H
2 #define _LINUX_PSTA_H
3
4 struct pinfo {
5     int nice;
6     pid_t pid;
7     uid_t uid;
8 };
9 #endif
```

```
#ifndef _LINUX_PSTA_H
#define _LINUX_PSTA_H

struct pinfo {
    int nice;
    pid_t pid;
    uid_t uid;
};
#endif
```

./include/linux/Kbuild 添加一行头文件

```
1 header-y += psta.h
```

```
header-y += wireless.h
header-y += x25.h
header-y += psta.h

unifdef-y += acct.h
unifdef-y += adb.h
```

./kernel/psta.c 编辑内容为

```
1  #include <linux/linkage.h>
2  #include <linux/types.h>
3  #include <linux/psta.h>
4  #include <linux/kernel.h>
5  asmlinkage int sys_psta(struct pinfo *buf) {
6      printk("Hello world\n");
7      return 0;
8  }

#include <linux/linkage.h>
#include <linux/types.h>
#include <linux/psta.h>
#include <linux/kernel.h>
asmlinkage int sys_psta(struct pinfo *buf) {
    printk("Hello world\n");
    return 0;
}
```

./kernel/Makefile 增加目标文件 psta.o

```
1  obj-y = psta.o sched.o fork.o exec_domain.o panic.o printk.o profile.o \
2      exit.o itimer.o time.o softirq.o resource.o \
3      sysctl.o capability.o ptrace.o timer.o user.o \
4      signal.o sys.o kmod.o workqueue.o pid.o \
5      rcupdate.o extable.o params.o posix-timers.o \
6      kthread.o wait.o kfifo.o sys_ni.o posix-cpu-timers.o mutex.o \
7      hrtimer.o rwsem.o latency.o nsproxy.o srcu.o

obj-y    = psta.o sched.o fork.o exec_domain.o panic.o printk.o profile.o \
exit.o itimer.o time.o softirq.o resource.o \
sysctl.o capability.o ptrace.o timer.o user.o \
signal.o sys.o kmod.o workqueue.o pid.o \
rcupdate.o extable.o params.o posix-timers.o \
kthread.o wait.o kfifo.o sys_ni.o posix-cpu-timers.o mutex.o \
hrtimer.o rwsem.o latency.o nsproxy.o srcu.o
```

./include/asm-i386/unistd.h 增加宏

```
1  #define __NR_psta 320
```

同时, 将 NR_syscalls 修改为 321

```
---
#define __NR_epoll_pwait          319
#define __NR_psta                 320
---
#ifdef __KERNEL__
#define NR_syscalls 321

```

./include/linux/syscalls.h 增加头文件

```
1  #include <linux/psta.h>

#include <linux/types.h>
#include <linux/aio_abi.h>
#include <linux/capability.h>
#include <linux/list.h>
#include <linux/sem.h>
#include <asm/semaphore.h>
#include <asm/siginfo.h>
#include <asm/signal.h>
#include <linux/quota.h>
#include <linux/key.h>
#include <linux/psta.h>
```

并把函数的定义加进来

```
1  asmlinkage int sys_psta(struct pinfo *buf)
```

```
asmlinkage int sys_psta(struct pinfo *buf);
```

./Makefile 修改内核编号为 seu2

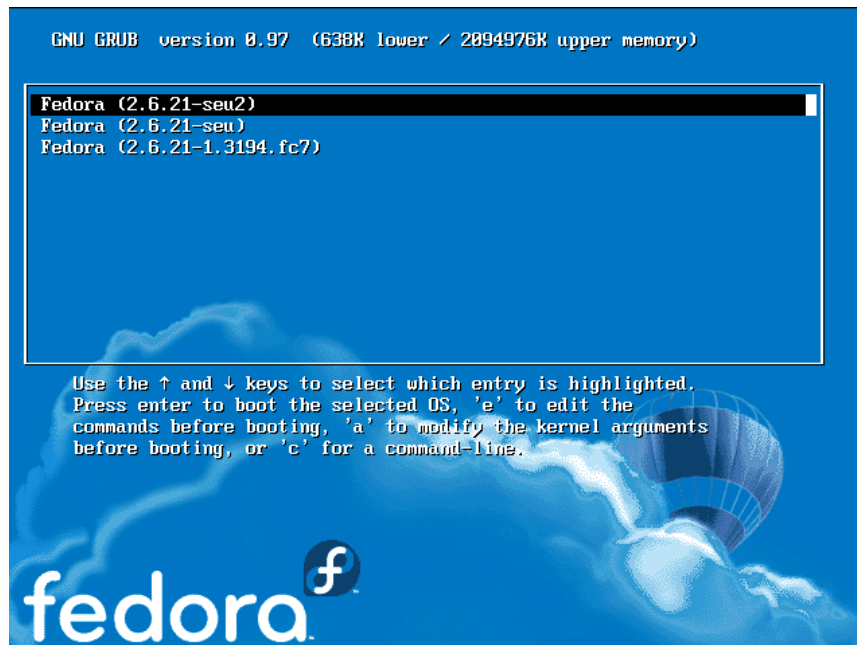
```
1 EXTRAVERSION = -seu2
```

```
VERSION = 2  
PATCHLEVEL = 6  
SUBLEVEL = 21  
EXTRAVERSION = -seu2  
NAME = Nocturnal Monster Puppy
```

然后重新编译重启

```
1 make mrproper  
2 cp /boot/config-2.6.21-1.3194.fc7 ./config  
3 make oldconfig  
4 make all  
5 su  
6 make modules_install  
7 make install  
8 make headers_install  
9 reboot
```

看到引导菜单。



选择 seu2，系统正常启动。

./test.c 编辑内容为

```
1 #include <sys/syscall.h>  
2 #include <unistd.h>  
3 #include <linux/psta.h>  
4  
5 int main()  
6 {  
7     struct pinfo info;  
8     int ret = syscall(320, &info);  
9     return 0;  
10 }
```

```

#include <sys/syscall.h>
#include <unistd.h>
#include <linux/psta.h>

int main()
{
    struct pinfo info;
    int ret = syscall(320,&info);
    return 0;
}

```

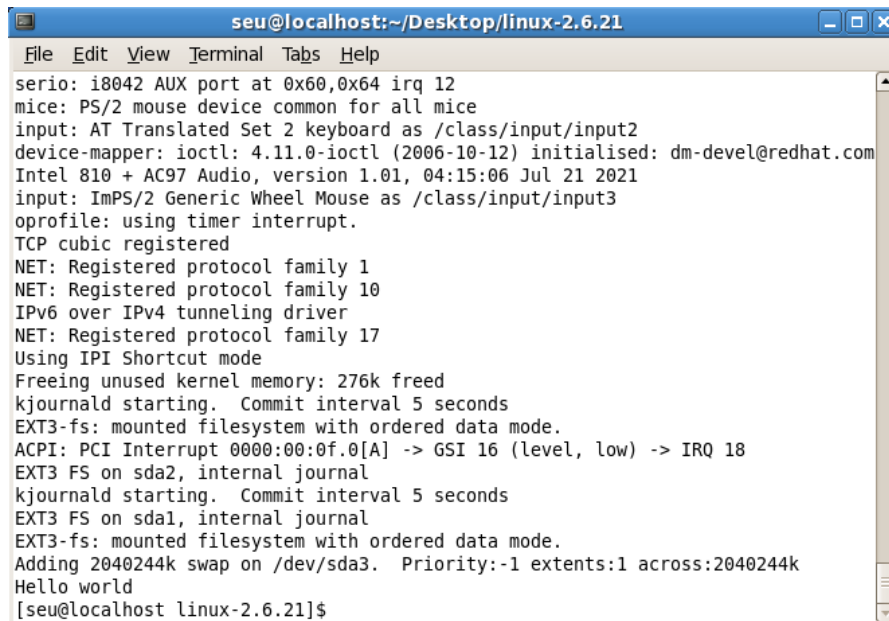
编译并运行，然后查看内核消息列表

```

1 gcc -o test test.c -I/home/seu/Desktop/linux-2.6.21/usr/include
2 ./test
3 dmesg

```

在内核日志的最后看到了 Hello world



```

seu@localhost:~/Desktop/linux-2.6.21
File Edit View Terminal Tabs Help
serio: i8042 AUX port at 0x60,0x64 irq 12
mice: PS/2 mouse device common for all mice
input: AT Translated Set 2 keyboard as /class/input/input2
device-mapper: ioctl: 4.11.0-ioctl (2006-10-12) initialised: dm-devel@redhat.com
Intel 810 + AC97 Audio, version 1.01, 04:15:06 Jul 21 2021
input: ImPS/2 Generic Wheel Mouse as /class/input/input3
oprofile: using timer interrupt.
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 10
IPv6 over IPv4 tunneling driver
NET: Registered protocol family 17
Using IPI Shortcut mode
Freeing unused kernel memory: 276k freed
kjournald starting. Commit interval 5 seconds
EXT3-fs: mounted filesystem with ordered data mode.
ACPI: PCI Interrupt 0000:00:0f.0[A] -> GSI 16 (level, low) -> IRQ 18
EXT3 FS on sda2, internal journal
kjournald starting. Commit interval 5 seconds
EXT3 FS on sda1, internal journal
EXT3-fs: mounted filesystem with ordered data mode.
Adding 2040244k swap on /dev/sda3. Priority:-1 extents:1 across:2040244k
Hello world
[seu@localhost linux-2.6.21]$

```

实验3：Linux进程管理及其扩展

做以下编辑：

./include/linux/sched.h

修改 task_struct

```

1 int cloak;

struct task_struct {
    volatile long state; /* -1 unrunnable, 0 runnable, >0 stopped */
    struct thread_info *thread_info;
    atomic_t usage;
    unsigned long flags; /* per process flags, defined below */
    unsigned long ptrace;

    int lock_depth; /* BKL lock depth */
    int cloak;
}

```

./kernel/fork.c

修改 copy_process

```

1 p->cloak = 0;

```

```

/*
 * Shared signal handlers imply shared VM. By way of the above,
 * thread groups also imply shared VM. Blocking this case allows
 * for various simplifications in other code.
 */
if ((clone_flags & CLONE_SIGHAND) && !(clone_flags & CLONE_VM))
    return ERR_PTR(-EINVAL);

retval = security_task_create(clone_flags);
if (retval)
    goto fork_out;

retval = -ENOMEM;
p = dup_task_struct(current);
if (!p)
    goto fork_out;

p->cloak = 0;

rt_mutex_init_task(p);

```

./fs/proc/base.c

添加语句

```

1  extern int hidden_flag;

                                extern int hidden_flag;

```

创建 sys_hide

```

1  asmlinkage int sys_hide(pid_t pid, int on){
2      if (current->uid == 0){
3          struct task_struct *task = find_task_by_pid(pid);
4          if (on == 0 || hidden_flag == 0)
5              task->cloak = 0;
6          else
7              task->cloak = 1;
8          proc_flush_task(task);
9      }
10     return 0;
11 }

                                asmlinkage int sys_hide(pid_t pid, int on){
                                    if (current->uid == 0){
                                        struct task_struct *task = find_task_by_pid(pid);
                                        if (on == 0 || hidden_flag == 0)
                                            task->cloak = 0;
                                        else
                                            task->cloak = 1;
                                        proc_flush_task(task);
                                    }
                                    return 0;
                                }

```

修改 proc_pid_readdir

```

1  if (hidden_flag == 0 &&
2      proc_pid_fill_cache(filp, dirent, filldir, task, tgid) < 0){
3      put_task_struct(task);
4      goto out;
5  }
6  if (hidden_flag == 1 && task->cloak == 0 &&
7      proc_pid_fill_cache(filp, dirent, filldir, task, tgid) < 0){
8      put_task_struct(task);
9      goto out;
10 }

```

```

for (task = next_tgid(tgid);
     task;
     put_task_struct(task), task = next_tgid(tgid + 1)) {
    tgid = task->pid;
    filp->f_pos = tgid + TGID_OFFSET;
    if (hidden_flag == 0 && proc_pid_fill_cache(filp, dirent, filldir, task, tgid) < 0){
        put_task_struct(task);
        goto out;
    }
    if (hidden_flag == 1 && task->cloak == 0 &&
        proc_pid_fill_cache(filp, dirent, filldir, task, tgid) < 0){
        put_task_struct(task);
        goto out;
    }
}

```

修改 `proc_pid_lookup`

```

1  if (task->cloak == 1 && hidden_flag == 1)
2      goto out;

```

```

        rcu_read_lock();
        task = find_task_by_pid(tgid);
        if (task)
            get_task_struct(task);
        rcu_read_unlock();
        if (!task)
            goto out;
        if (task->cloak == 1 && hidden_flag == 1)
            goto out;

```

创建 `sys_hide_user_processes`

```

1  asmlinkage int sys_hide_user_processes(uid_t uid, char *comm, int on){
2      if (current->uid == 0){
3          struct task_struct *task = NULL;
4          for_each_process(task){
5              char *s = task->comm;
6              if (hidden_flag == 0 && task->uid == uid){ //judge hidden_flag
7                  task->cloak = 0;
8              }
9              else if (comm == NULL){ //hide all
10                 if (task->uid == uid){
11                     task->cloak = on;
12                 }
13             }
14             else if (task->uid == uid && strcmp(s, comm) == 0){ //hide comm
15                 task->cloak = on;
16             }
17             proc_flush_task(task);
18         }
19     }
20     return 0;
21 }

```



```

asmlinkage int sys_hide_user_processes(uid_t uid, char *comm, int on){
    if (current->uid == 0){
        struct task_struct *task = NULL;
        for_each_process(task){
            char *s = task->comm;
            if (hidden_flag == 0 && task->uid == uid){
                task->cloak = 0;
            }
            else if (comm == NULL){
                if (task->uid == uid){
                    task->cloak = on;
                }
            }
            else if (task->uid == uid && strcmp(s, comm) == 0){
                task->cloak = on;
            }
            proc_flush_task(task);
        }
    }
    return 0;
}

```

./fs/proc/proc_misc.c

添加以下变量和函数

```

1  int hidden_flag = 0;
2  EXPORT_SYMBOL(hidden_flag);
3
4  static int proc_read_hidden(char *page, char **start,
5                             off_t off, int count, int *eof, void *data)
6  {
7      int len = 0;
8      len = sprintf(page, "%d", hidden_flag);
9      return len;
10 }
11
12 static int proc_write_hidden(struct file *file, const char *buffer,
13                             unsigned long count, void *data)
14 {
15     hidden_flag = buffer[0] - '0';
16     return count;
17 }
18
19 static int proc_read_hidden_processes(char *page, char **start, off_t off,
20                                      int count, int *eof, void *data){
21     static char buf[1024*8]="";
22     char tmp[128];
23     struct task_struct *p;
24     if (off>0)
25         return 0;
26     sprintf(buf, "%s", "");
27     for_each_process(p){
28         if (p->cloak == 1){
29             sprintf(tmp, "%d", p->pid);
30             strcat(buf, tmp);
31         }
32     }
33     sprintf(page, "%s", buf);
34     return strlen(buf);
35 }

```

```

int hidden_flag = 0;
EXPORT_SYMBOL(hidden_flag);

static int proc_read_hidden(char *page, char **start,
                           off_t off, int count, int *eof, void *data)
{
    int len = 0;
    len = sprintf(page, "%d", hidden_flag);
    return len;
}

static int proc_write_hidden(struct file *file, const char *buffer,
                           unsigned long count, void *data)
{
    hidden_flag = buffer[0] - '0';
    return count;
}

static int proc_read_hidden_processes(char *page, char **start, off_t off, int count, int *eof, void *data){
    static char buf[1024*8]="";
    char tmp[128];
    struct task_struct *p;
    if (off>0)
        return 0;
    sprintf(buf, "%s", "");
    for_each_process(p){
        if (p->cloak == 1){
            sprintf(tmp, "%d", p->pid);
            strcat(buf, tmp);
        }
    }
    sprintf(page, "%s", buf);
    return strlen(buf);
}

```

在 `proc_misc_init` 最后添加

```

1  struct proc_dir_entry *ptr = create_proc_entry("hidden", 0644, NULL);
2  ptr->read_proc = proc_read_hidden;
3  ptr->write_proc = proc_write_hidden;
4  struct proc_dir_entry *hideprocessfile =
5      create_proc_entry("hidden_process", 0644, NULL);
6  hideprocessfile->read_proc=proc_read_hidden_processes;

    struct proc_dir_entry *ptr = create_proc_entry("hidden", 0644, NULL);
    ptr->read_proc = proc_read_hidden;
    ptr->write_proc = proc_write_hidden;
    struct proc_dir_entry *hideprocessfile = create_proc_entry("hidden_process", 0644, NULL);
    hideprocessfile->read_proc=proc_read_hidden_processes;

```

`./arch/i386/kernel/syscall_table.S` 最后添加系统调用

```

1  .long sys_hide
2  .long sys_hide_user_process

    .long sys_tee                /* 315 */
    .long sys_vmsplice
    .long sys_move_pages
    .long sys_getcpu
    .long sys_epoll_pwait
    .long sys_psta              /* 320 */
    .long sys_hide
    .long sys_hide_user_processes

```

`./include/asm-i386/unistd.h` 增加宏

```

1  #define __NR_hide 321
2  #define __NR_hide_user_processes 322

```

同时, 将 `NR_syscalls` 修改为 323

```

#define __NR_getcpu 318
#define __NR_epoll_pwait 319
#define __NR_psta 320
#define __NR_hide 321
#define __NR_hide_user_processes 322

#ifdef __KERNEL__
#define NR_syscalls 323

```

./include/linux/syscalls.h 把函数的定义加进来

```

1  asmlinkage int sys_hide(pid_t pid, int on);
2  asmlinkage int sys_hide_user_processes(uid_t uid, char *comm, int on);

asmlinkage int sys_hide(pid_t pid, int on);
asmlinkage int sys_hide_user_processes(uid_t uid, char *comm, int on);

```

然后重新编译重启

```

1  make mrproper
2  cp /boot/config-2.6.21-1.3194.fc7 ./config
3  make oldconfig
4  make all
5  su
6  make modules_install
7  make install
8  make headers_install
9  reboot

```

启动后在 /proc 文件夹下看到了 hidden 和 hidden_process



hidden



hidden_process

创建 4 个测试文件



hideInit.c



hideRoot.c



hideRootInit.c



recoverInit.c

内容分别为

```

1  //hideInit.c
2  #include <stdlib.h>
3  #include <stdio.h>
4  #include <unistd.h>
5  #include <sys/syscall.h>
6
7  int main()
8  {
9      syscall(321, 1, 1);
10     return 0;
11 }

```

```

1 //recoverInit.c
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <unistd.h>
5 #include <sys/syscall.h>
6
7 int main()
8 {
9     syscall(321, 1, 0);
10    return 0;
11 }

```

```

1 //hideRootInit.c
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <unistd.h>
5 #include <sys/syscall.h>
6
7 int main()
8 {
9     syscall(322, 0, "init", 1);
10    return 0;
11 }

```

```

1 //hideRoot.c
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <unistd.h>
5 #include <sys/syscall.h>
6
7 int main()
8 {
9     syscall(322, 0, NULL, 1);
10    return 0;
11 }

```

分别编译

```

1 cd Desktop
2 gcc -o hideInit hideInit.c
3 gcc -o recoverInit recoverInit.c
4 gcc -o hideRootInit hideRootInit.c
5 gcc -o hideRoot hideRoot.c

```

然后进行测试

```

1 cd /proc
2 su
3 echo "1" > hidden

```

```

[root@localhost proc]# echo "1" > hidden
[root@localhost proc]# cat < hidden
1[root@localhost proc]#

```

然后 top

```
[seu@localhost Desktop]$ top
```

top - 10:15:48 up 51 min, 2 users, load average: 0.00, 0.01, 0.00
Tasks: 111 total, 2 running, 109 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 2074028k total, 310892k used, 1763136k free, 19036k buffers
Swap: 2040244k total, 0k used, 2040244k free, 166600k cached

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3722	root	15	0	3136	864	748	S	0	0.0	0:00.37	hald-addon-stor
1	root	15	0	2132	624	540	S	0	0.0	0:00.56	init

测试 ./hideInit

```
[seu@localhost Desktop]$ ./hideInit
[seu@localhost Desktop]$ top
```

top - 10:16:25 up 52 min, 2 users, load average: 0.00, 0.01, 0.00
Tasks: 111 total, 2 running, 109 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 2074028k total, 311088k used, 1762940k free, 19060k buffers
Swap: 2040244k total, 0k used, 2040244k free, 166608k cached

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3878	root	15	0	149m	12m	6420	S	0	0.6	0:07.86	Xorg
5054	seu	15	0	38348	11m	8436	R	0	0.6	0:00.24	gnome-terminal
1	root	15	0	2132	624	540	S	0	0.0	0:00.56	init

没有成功。

进入 root 权限重新测试

```
[seu@localhost Desktop]$ su
Password:
[root@localhost Desktop]# ./hideInit
[root@localhost Desktop]# top
```

top - 10:18:05 up 53 min, 2 users, load average: 0.07, 0.02, 0.00
Tasks: 112 total, 2 running, 110 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.2%us, 0.1%sy, 0.0%ni, 99.6%id, 0.1%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 2074028k total, 315300k used, 1758728k free, 19100k buffers
Swap: 2040244k total, 0k used, 2040244k free, 166612k cached

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3878	root	15	0	151m	15m	6420	R	2	0.8	0:08.01	Xorg
2	root	RT	0	0	0	0	S	0	0.0	0:00.01	migration/0
3	root	34	19	0	0	0	S	0	0.0	0:00.00	ksoftirqd/0
4	root	RT	0	0	0	0	S	0	0.0	0:00.00	watchdog/0
5	root	RT	0	0	0	0	S	0	0.0	0:00.06	migration/1
6	root	34	19	0	0	0	S	0	0.0	0:00.00	ksoftirqd/1
7	root	RT	0	0	0	0	S	0	0.0	0:00.00	watchdog/1
8	root	10	-5	0	0	0	S	0	0.0	0:00.00	events/0
9	root	10	-5	0	0	0	S	0	0.0	0:00.00	events/1
10	root	20	-5	0	0	0	S	0	0.0	0:00.00	khelper
11	root	10	-5	0	0	0	S	0	0.0	0:00.00	kthread
75	root	16	-5	0	0	0	S	0	0.0	0:00.09	kblockd/0
76	root	10	-5	0	0	0	S	0	0.0	0:00.12	kblockd/1
77	root	15	-5	0	0	0	S	0	0.0	0:00.00	kacpid
245	root	13	-5	0	0	0	S	0	0.0	0:00.00	ata/0
246	root	14	-5	0	0	0	S	0	0.0	0:00.00	ata/1
247	root	13	-5	0	0	0	S	0	0.0	0:00.00	ata_aux
248	root	13	-5	0	0	0	S	0	0.0	0:00.00	ksuspend_usbd
251	root	10	-5	0	0	0	S	0	0.0	0:00.00	khubd
253	root	15	-5	0	0	0	S	0	0.0	0:00.00	kseriod
264	root	13	-5	0	0	0	S	0	0.0	0:00.00	khpsbpkt
291	root	17	0	0	0	0	S	0	0.0	0:00.00	pdflush
292	root	15	0	0	0	0	S	0	0.0	0:00.02	pdflush
293	root	12	-5	0	0	0	S	0	0.0	0:00.00	kswapd0
294	root	12	-5	0	0	0	S	0	0.0	0:00.00	aio/0
295	root	12	-5	0	0	0	S	0	0.0	0:00.00	aio/1
958	root	11	-5	0	0	0	S	0	0.0	0:00.00	scsi_eh_0
1051	root	11	-5	0	0	0	S	0	0.0	0:00.00	kpsmouse
1056	root	11	-5	0	0	0	S	0	0.0	0:00.00	kondemand/0
1057	root	12	-5	0	0	0	S	0	0.0	0:00.00	kondemand/1
1067	root	10	-5	0	0	0	S	0	0.0	0:00.13	kjournald
1130	root	21	-4	2352	632	384	S	0	0.0	0:00.04	udev
2382	root	10	-5	0	0	0	S	0	0.0	0:00.00	kjournald

```
[root@localhost Desktop]# cat /proc/hidden_process
1[root@localhost Desktop]#
```

看到 1 号进程成功屏蔽

测试 ./recoverInit

```
[root@localhost Desktop]# ./recoverInit
[root@localhost Desktop]# top

top - 10:19:44 up 55 min, 2 users, load average: 0.04, 0.02, 0.00
Tasks: 113 total, 2 running, 111 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.7%us, 1.3%sy, 0.0%ni, 97.3%id, 0.0%wa, 0.7%hi, 0.0%si, 0.0%st
Mem: 2074028k total, 313428k used, 1760600k free, 19136k buffers
Swap: 2040244k total, 0k used, 2040244k free, 166608k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3878	root	15	0	149m	12m	6420	S	2	0.6	0:08.32	Xorg
5054	seu	15	0	38604	12m	8436	S	1	0.6	0:00.41	gnome-terminal
4137	seu	15	0	16368	3948	3156	R	1	0.2	0:00.18	gnome-screensav
3701	haldaemo	15	0	5620	3616	2496	S	0	0.2	0:00.06	hald
1	root	15	0	2132	624	540	S	0	0.0	0:00.56	init

```
[root@localhost Desktop]# cat /proc/hidden_process
[root@localhost Desktop]#
```

init 又回来了

测试 ./hideRootInit

```
[root@localhost Desktop]# ./hideRootInit
[root@localhost Desktop]# top

top - 12:26:25 up 2 min, 2 users, load average: 0.04, 0.04, 0.01
Tasks: 110 total, 2 running, 108 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.0%us, 0.0%sy, 0.0%ni, 100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 2074020k total, 281508k used, 1792512k free, 15208k buffers
Swap: 2040244k total, 0k used, 2040244k free, 150392k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2	root	RT	0	0	0	0	S	0	0.0	0:00.05	migration/0
3	root	34	19	0	0	0	S	0	0.0	0:00.00	ksoftirqd/0
4	root	RT	0	0	0	0	S	0	0.0	0:00.00	watchdog/0
5	root	RT	0	0	0	0	S	0	0.0	0:00.06	migration/1
6	root	34	19	0	0	0	S	0	0.0	0:00.00	ksoftirqd/1
7	root	RT	0	0	0	0	S	0	0.0	0:00.00	watchdog/1
8	root	10	-5	0	0	0	S	0	0.0	0:00.00	events/0
9	root	10	-5	0	0	0	S	0	0.0	0:00.00	events/1
10	root	20	-5	0	0	0	S	0	0.0	0:00.00	khelper
11	root	10	-5	0	0	0	S	0	0.0	0:00.00	kthread
75	root	10	-5	0	0	0	S	0	0.0	0:00.07	kblockd/0
76	root	10	-5	0	0	0	S	0	0.0	0:00.11	kblockd/1
77	root	15	-5	0	0	0	S	0	0.0	0:00.00	kacpid
245	root	13	-5	0	0	0	S	0	0.0	0:00.00	ata/0
246	root	14	-5	0	0	0	S	0	0.0	0:00.00	ata/1
247	root	13	-5	0	0	0	S	0	0.0	0:00.00	ata_aux
248	root	13	-5	0	0	0	S	0	0.0	0:00.00	ksuspend_usbd
251	root	10	-5	0	0	0	S	0	0.0	0:00.00	khubb
253	root	15	-5	0	0	0	S	0	0.0	0:00.00	kseriod
264	root	13	-5	0	0	0	S	0	0.0	0:00.00	khpsbpkt
291	root	17	0	0	0	0	S	0	0.0	0:00.00	pdflush
292	root	15	0	0	0	0	S	0	0.0	0:00.01	pdflush
293	root	12	-5	0	0	0	S	0	0.0	0:00.00	kswapd0
294	root	12	-5	0	0	0	S	0	0.0	0:00.00	aio/0
295	root	12	-5	0	0	0	S	0	0.0	0:00.00	aio/1
958	root	11	-5	0	0	0	S	0	0.0	0:00.00	scsi_eh_0
1051	root	11	-5	0	0	0	S	0	0.0	0:00.00	kpsmouse
1056	root	11	-5	0	0	0	S	0	0.0	0:00.00	kondemand/0
1057	root	12	-5	0	0	0	S	0	0.0	0:00.00	kondemand/1
1067	root	10	-5	0	0	0	S	0	0.0	0:00.05	kjournald
1130	root	21	-4	2352	628	384	S	0	0.0	0:00.02	udevd
2384	root	11	-5	0	0	0	S	0	0.0	0:00.00	kjournald
2863	root	15	0	27988	3900	3220	S	0	0.2	0:00.01	vmtoolsd
3182	root	16	0	1800	596	496	S	0	0.0	0:00.00	syslogd
3185	root	25	0	1740	404	332	S	0	0.0	0:00.00	klogd

```
[root@localhost Desktop]# cat /proc/hidden_process
1[root@localhost Desktop]#
```

看到 root 的 init 进程成功屏蔽

测试 ./hideRoot

```

[root@localhost Desktop]# ./hideRoot
[root@localhost Desktop]# top

top - 11:55:19 up 4 min,  2 users,  load average: 0.01, 0.06, 0.02
Tasks: 49 total,  1 running, 48 sleeping,  0 stopped,  0 zombie
Cpu(s):  0.7%us,  0.7%sy,  0.0%ni, 98.2%id,  0.3%wa,  0.1%hi,  0.0%si,  0.0%st
Mem:   2074020k total,  295068k used, 1778952k free,   15512k buffers
Swap:  2040244k total,    0k used,  2040244k free,   154084k cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 3217 rpc        15   0  2196   700   536  S   0.0   0.0   0:00.00 rpcbind
 3288 dbus        15   0  2948   900   616  S   0.0   0.0   0:00.01 dbus-daemon
 3506 smmisp     25   0  7988  1472   644  S   0.0   0.1   0:00.00 sendmail
 3623 xfs         15   0  3868  1652   780  S   0.0   0.1   0:00.00 xfs
 3689 avahi      25   0  2640  1304  1140  S   0.0   0.1   0:00.00 avahi-daemon
 3690 avahi      25   0  2640   312   184  S   0.0   0.0   0:00.00 avahi-daemon
 3701 haldaemon 15   0  5756  3752  2496  S   0.0   0.2   0:00.03 hald
 3712 haldaemon 25   0  2072   788   700  S   0.0   0.0   0:00.00 hald-addon-keyb
 3713 haldaemon 15   0  2072   788   700  S   0.0   0.0   0:00.00 hald-addon-keyb
 3728 haldaemon 25   0  2068   784   696  S   0.0   0.0   0:00.00 hald-addon-acpi
 3900 seu       15   0 31044  5760  4848  S   0.0   0.3   0:00.02 gnome-session
 3991 seu       18   0  4480   516   260  S   0.0   0.0   0:00.00 ssh-agent
 3994 seu       22   0  2832   492   392  S   0.0   0.0   0:00.00 dbus-launch
 3995 seu       17   0  2816   832   592  S   0.0   0.0   0:00.02 dbus-daemon
 4001 seu       17   0  7340  3704  1708  S   0.0   0.2   0:00.12 gconfd-2
 4004 seu       25   0  2772   776   676  S   0.0   0.0   0:00.00 gnome-keyring-d
 4006 seu       15   0  37868  12m  6220  S   0.0   0.6   0:00.09 gnome-settings-
 4010 seu       15   0  17864  8484  6236  S   0.0   0.4   0:00.20 metacity
 4012 seu       15   0  34992  12m  9132  S   0.0   0.6   0:00.14 gnome-panel
 4014 seu       15   0  80516  20m  11m  S   0.0   1.0   0:00.33 nautilus
 4018 seu       15   0  22544  4232  3344  S   0.0   0.2   0:00.00 gnome-volume-ma
 4020 seu       18   0  41644  2740  1988  S   0.0   0.1   0:00.05 bonobo-activati
 4022 seu       15   0  14424  4488  3844  S   0.0   0.2   0:00.01 bluetooth-apple
 4026 seu       15   0  57844  13m  11m  S   0.0   0.6   0:00.12 vmtoolsd
 4038 seu       15   0  39344  9060  7380  S   0.0   0.4   0:00.02 nm-applet
 4055 seu       22   0  11188  2960  2552  S   0.0   0.1   0:00.00 gnome-vfs-daemo
 4058 seu       15   0  24132  12m  7652  S   0.0   0.6   0:00.02 puplet
 4059 seu       17   0  10768  5292  2904  S   0.0   0.3   0:00.02 python
 4063 seu       25   0  15352  2088  1756  S   0.0   0.1   0:00.00 escd
 4065 seu       15   0  13096  2564  2176  S   0.0   0.1   0:00.00 pam-panel-icon
 4066 seu       15   0  24792  5252  3968  S   0.0   0.3   0:00.01 gnome-power-man
 4071 root       15   0  1912   616   520  S   0.0   0.0   0:00.00 pam_timestamp_c
 4074 seu       15   0  34292  10m  7668  S   0.0   0.5   0:00.08 wnck-applet
 4082 seu       15   0  65408  9136  7016  S   0.0   0.4   0:00.01 trashapplet
 4094 seu       18   0  2496  1032   852  S   0.0   0.0   0:00.00 gam_server

```

```

[root@localhost Desktop]# cat /proc/hidden_process
123456789101175767724524624724825125326429129229329429595810511056105710671130238428633182318531973248334633733392340
635003520357936463668367937043728379938043805380638083811381638193874387838794262[root@localhost Desktop]#

```

看到 root 的进程全部被屏蔽

编写程序调用 `syscall(322, 0, NULL, 0);` 恢复 hidden_process。

然后修改 hidden 为 0

```
1 echo "0" > hidden
```

重新测试 `./hideRoot`

```

[root@localhost Desktop]# ./hideRoot
[root@localhost Desktop]# top

top - 12:43:46 up 19 min, 3 users, load average: 0.03, 0.04, 0.00
Tasks: 114 total, 1 running, 113 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.2%us, 0.2%sy, 0.0%ni, 99.7%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 2074020k total, 303256k used, 1770764k free, 16784k buffers
Swap: 2040244k total, 0k used, 2040244k free, 157372k cached

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3728	root	15	0	3132	864	748	S	0	0.0	0:00.14	hald-addon-stor
3879	root	15	0	149m	13m	6800	S	0	0.6	0:04.53	Xorg
1	root	15	0	2136	628	540	S	0	0.0	0:00.57	init
2	root	RT	0	0	0	0	S	0	0.0	0:00.06	migration/0
3	root	34	19	0	0	0	S	0	0.0	0:00.00	ksoftirqd/0
4	root	RT	0	0	0	0	S	0	0.0	0:00.00	watchdog/0
5	root	RT	0	0	0	0	S	0	0.0	0:00.06	migration/1
6	root	34	19	0	0	0	S	0	0.0	0:00.00	ksoftirqd/1
7	root	RT	0	0	0	0	S	0	0.0	0:00.00	watchdog/1
8	root	10	-5	0	0	0	S	0	0.0	0:00.00	events/0
9	root	10	-5	0	0	0	S	0	0.0	0:00.00	events/1
10	root	20	-5	0	0	0	S	0	0.0	0:00.00	khelper
11	root	10	-5	0	0	0	S	0	0.0	0:00.00	kthread
75	root	10	-5	0	0	0	S	0	0.0	0:00.07	kblockd/0
76	root	10	-5	0	0	0	S	0	0.0	0:00.11	kblockd/1
77	root	15	-5	0	0	0	S	0	0.0	0:00.00	kacpid
245	root	13	-5	0	0	0	S	0	0.0	0:00.00	ata/0
246	root	14	-5	0	0	0	S	0	0.0	0:00.00	ata/1
247	root	13	-5	0	0	0	S	0	0.0	0:00.00	ata_aux
248	root	13	-5	0	0	0	S	0	0.0	0:00.00	ksuspend_usbd
251	root	10	-5	0	0	0	S	0	0.0	0:00.00	khubd
253	root	15	-5	0	0	0	S	0	0.0	0:00.00	kseriod
264	root	13	-5	0	0	0	S	0	0.0	0:00.00	khpsbpkt
291	root	17	0	0	0	0	S	0	0.0	0:00.00	pdflush
292	root	15	0	0	0	0	S	0	0.0	0:00.04	pdflush
293	root	12	-5	0	0	0	S	0	0.0	0:00.00	kswapd0
294	root	12	-5	0	0	0	S	0	0.0	0:00.00	aio/0
295	root	12	-5	0	0	0	S	0	0.0	0:00.00	aio/1
958	root	11	-5	0	0	0	S	0	0.0	0:00.00	scsi_eh_0
1051	root	11	-5	0	0	0	S	0	0.0	0:00.00	kpsmoused
1056	root	11	-5	0	0	0	S	0	0.0	0:00.00	kondemand/0
1057	root	12	-5	0	0	0	S	0	0.0	0:00.00	kondemand/1
1067	root	10	-5	0	0	0	S	0	0.0	0:00.08	kjournald
1130	root	21	-4	2352	628	384	S	0	0.0	0:00.02	udev
2384	root	11	-5	0	0	0	S	0	0.0	0:00.00	kjournald

```

[root@localhost Desktop]# cat /proc/hidden_process
[root@localhost Desktop]#

```

可以看到，屏蔽不起效果了。

实验总结

本实验 1 和 2 很容易，依葫芦画瓢即可。实验 3 其实也不难，但我在 `hide_user_processes` 上花了整整 5 个小时：先是 `==` 写成了 `=`，查 bug 查了两个小时；然后，本应该比较两个指针内容的地方，我写成了比较两个指针，又 debug 了两个小时🤦🏻

通过本次实验，熟悉了 Linux 内核文件结构，学会了修改与增加系统调用，掌握了进程控制和进程管理相关内容。