实验一 Linux内核编译及添加系统调用

17042127 陶逸群

设计要求

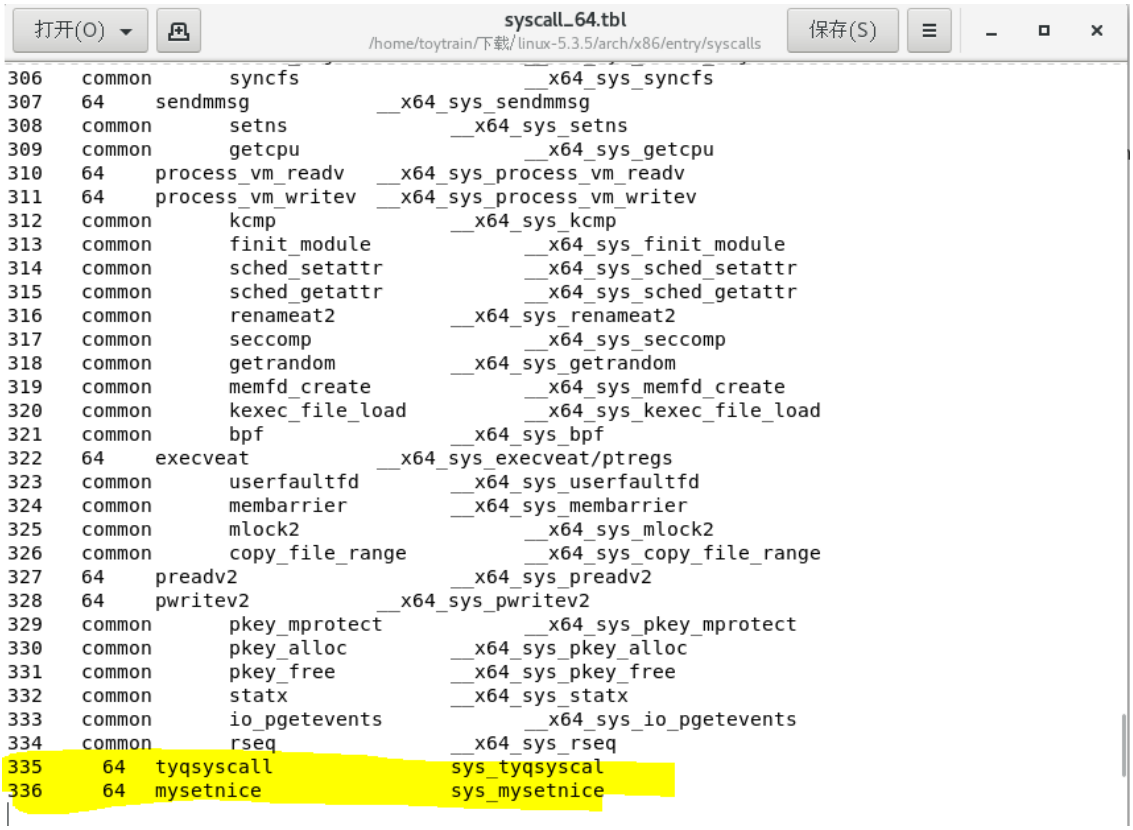
- 添加一个系统调用，实现输出功能
- 添加一个系统调用，实现对指定进程的 nice 值得修改或读取功能
- 写一个简单的应用程序测试添加的修改指定进程nice值的系统调用
- 若程序中调用了 Linux 的内核函数，要求深入阅读相关函数源码

实验步骤

1. 下载Linux源码

   我下载了Linux-5.3.5的源码。

2. 在系统调用表中添加系统调用号




```
306   common      syncfs                  __x64_sys_syncfs
307   64    sendmmsg            __x64_sys_sendmmsg
308   common      setns                   __x64_sys_setns
309   common      getcpu                  __x64_sys_getcpu
310   64    process_vm_readv    __x64_sys_process_vm_readv
311   64    process_vm_writev   __x64_sys_process_vm_writev
312   common      kcmp                    __x64_sys_kcmp
313   common      finit_module            __x64_sys_finit_module
314   common      sched_setattr           __x64_sys_sched_setattr
315   common      sched_getattr           __x64_sys_sched_getattr
316   common      renameat2           __x64_sys_renameat2
317   common      seccomp                 __x64_sys_seccomp
318   common      getrandom           __x64_sys_getrandom
319   common      memfd_create            __x64_sys_memfd_create
320   common      kexec_file_load         __x64_sys_kexec_file_load
321   common      bpf                 __x64_sys_bpf
322   64    execveat            __x64_sys_execveat/ptregs
323   common      userfaultfd         __x64_sys_userfaultfd
324   common      membarrier          __x64_sys_membarrier
325   common      mlock2                  __x64_sys_mlock2
326   common      copy_file_range         __x64_sys_copy_file_range
327   64    preadv2             __x64_sys_preadv2
328   64    pwritev2            __x64_sys_pwritev2
329   common      pkey_mprotect           __x64_sys_pkey_mprotect
330   common      pkey_alloc          __x64_sys_pkey_alloc
331   common      pkey_free           __x64_sys_pkey_free
332   common      statx               __x64_sys_statx
333   common      io_pgetevents           __x64_sys_io_pgetevents
334   common      rseq                __x64_sys_rseq
335     64    tyqsyscall              sys_tyqsyscal
336     64    mysetnice               sys_mysetnice
```

我在系统调用表中添加了两个系统调用号，其中 335 号用于在内核中输出我的信息，证明我实现了对内核的修改。336号系统调用用于读取或修改进程的 nice 值。

3. 申明系统调用服务例程原型

```
/*
 *
 * 我的修改
 */
asmlinkage long sys_tyqsyscall(void);
asmlinkage int  sys_mysetnice(pid_t pid,int flag,int nicevalue,void __user* prio,void __user* nice);
#endif
```

4. 实现系统调用服务例程

```
SYSCALL_DEFINE0(tyqsyscall){
        printk("陶逸群的系统调用\n ");
        return 0;
}
SYSCALL_DEFINE5(mysetnice,pid_t,pid,int,flag,int,nicevalue,void __user
*,prio,void __user *,nice){
    struct pid * kpid;//进程描述符指针，指向一个枚举类型
    struct task_struct * task;//任务描述符信息
    kpid = find_get_pid(pid);//根据进程号返回kpid
    task = pid_task(kpid, PIDTYPE_PID);//返回task_struct
    int n;
    n = task_nice(task);//返回进程当前nice值
    int p;
    p = task_prio(task);//返回进程当前prio值
    if(flag == 1){
        set_user_nice(task, nicevalue);//修改进程nice值
        n = task_nice(task);//重新取得进程nice值
        p = task_prio(task);//重新取得进程当前prio值
        copy_to_user(nice,&n,sizeof(n));//将nice值拷贝到用户空间
        copy_to_user(prio,&p,sizeof(p));//将prio值拷贝到用户空间
        return 0;
    }
    else if(flag == 0){
        copy_to_user(nice,&n,sizeof(n));//将nice值拷贝到用户空间
        copy_to_user(prio,&p,sizeof(p));//将prio值拷贝到用户空间
        return 0;
```

```
    }
    return EFAULT;
}
```

内核函数的具体作用见以上注释部分。

5. 重新编译内核
6. 编写用户态程序来测试新添加的系统调用
   - 测试打印信息

```c
#include <unistd.h>
#include <sys/syscall.h>
int main(){
    syscall(335);
    return 0;
}
```

   - 测试修改nice值

```c
#define _GNU_SOURCE
#include <unistd.h>
#include<sys/syscall.h>
#include<stdio.h>
#include<stdlib.h>
int main(){
    pid_t pid;
    int nicevalue;
    int flag;
    int p = 0;
    int n = 0;
    int *prio;
    int *nice;
    prio = &p;
    nice = &n;

    printf("请输入pid: \n");
    scanf("%d",&pid);
    printf("请输入nice: \n");
    scanf("%d",&nicevalue);

    printf("请输入flag: \n");
    scanf("%d",&flag);

    syscall(336,pid,flag,nicevalue,prio,nice);

    printf("现在的nice为%d\n,prio为%d\n",n,p);
    return 0;
}

```

结果演示

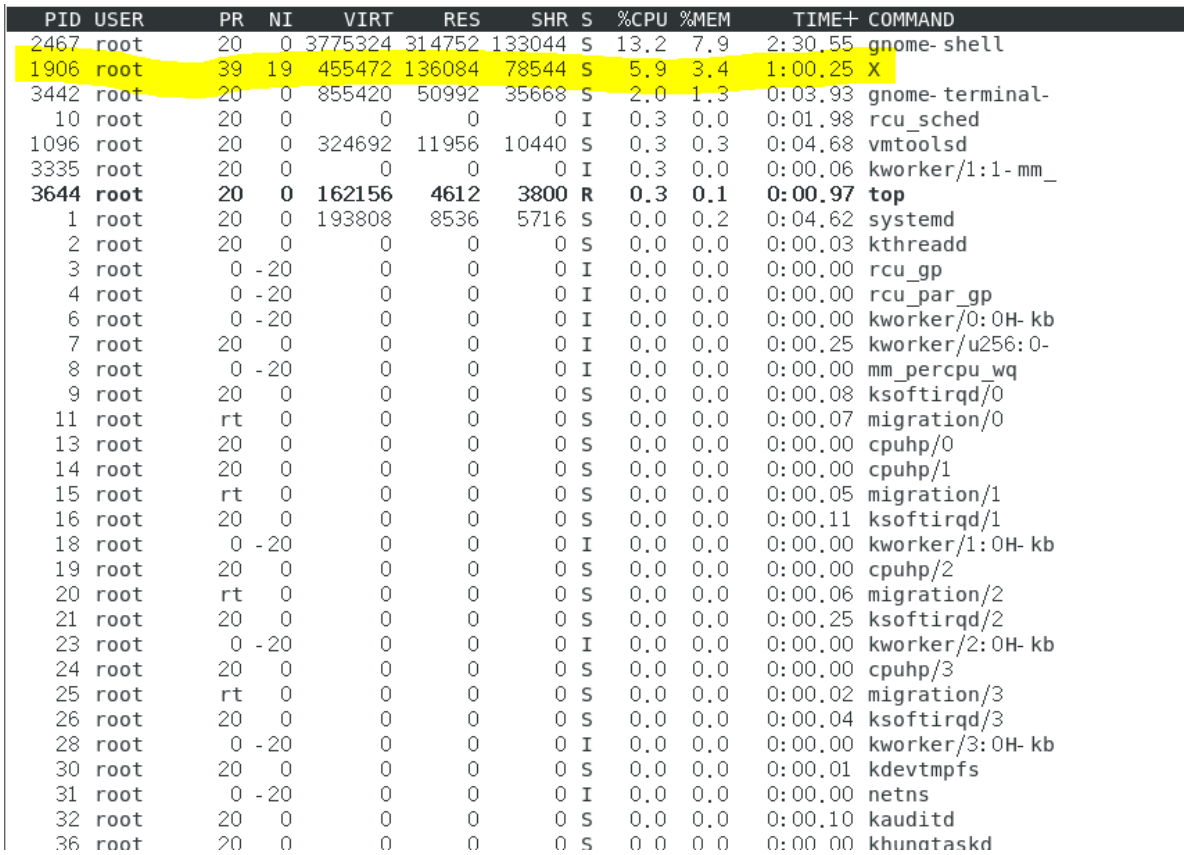
```
[   86.795192] Bluetooth: RFCOMM socket layer initialized
[   86.795307] Bluetooth: RFCOMM ver 1.11
[ 1706.542628] 陶逸群的系统调用
```

修改前：

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 2467 root      20   0 3773596 313984 131312 R   7.3  7.8   2:20.52 gnome-shell
 1906 root      20   0  453744 134560  77020 R   5.9  3.4   0:57.17 X
 3442 root      20   0  843656  46944  33116 R   2.6  1.2   0:02.82 gnome-terminal-
 2728 root      20   0  641732  38812  31120 S   1.7  1.0   0:05.46 vmtoolsd
 3596 root      20   0       0      0      0 I   0.7  0.0   0:00.09 kworker/0:1-eve
 3644 root      20   0  162156   4612   3800 R   0.7  0.1   0:00.13 top
 1061 root      20   0   90576   5504   4640 R   0.3  0.1   0:04.25 rngd
 1096 root      20   0  324692  11956  10440 S   0.3  0.3   0:04.37 vmtoolsd
 2630 root      20   0  779908  25308  19888 S   0.3  0.6   0:00.65 gsd-color
 2701 root      20   0  905816  45540  34940 D   0.3  1.1   0:01.54 nautilus-deskto
 2830 root      20   0 1084204  75704  50048 S   0.3  1.9   0:24.94 nautilus
 3241 root      20   0       0      0      0 I   0.3  0.0   0:00.08 kworker/2:0-mm_
    1 root      20   0  193808   8536   5716 S   0.0  0.2   0:04.60 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.03 kthreadd
    3 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 rcu_par_gp
    6 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/0:0H-kb
    7 root      20   0       0      0      0 I   0.0  0.0   0:00.25 kworker/u256:0-
    8 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 mm_percpu_wq
    9 root      20   0       0      0      0 S   0.0  0.0   0:00.08 ksoftirqd/0
   10 root      20   0       0      0      0 R   0.0  0.0   0:01.84 rcu_sched
   11 root      rt   0       0      0      0 S   0.0  0.0   0:00.07 migration/0
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/0
   14 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/1
   15 root      rt   0       0      0      0 S   0.0  0.0   0:00.05 migration/1
   16 root      20   0       0      0      0 S   0.0  0.0   0:00.11 ksoftirqd/1
   18 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/1:0H-kb
   19 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/2
   20 root      rt   0       0      0      0 S   0.0  0.0   0:00.06 migration/2
```

修改:

修改后：



```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 2467 root      20   0 3775324 314752 133044 S  13.2   7.9   2:30.55 gnome-shell
 1906 root      39  19  455472 136084  78544 S   5.9   3.4   1:00.25 X
 3442 root      20   0  855420  50992  35668 S   2.0   1.3   0:03.93 gnome-terminal-
   10 root      20   0       0      0      0 I   0.3   0.0   0:01.98 rcu_sched
 1096 root      20   0  324692  11956  10440 S   0.3   0.3   0:04.68 vmtoolsd
 3335 root      20   0       0      0      0 I   0.3   0.0   0:00.06 kworker/1:1-mm_
 3644 root      20   0  162156   4612   3800 R   0.3   0.1   0:00.97 top
    1 root      20   0  193808   8536   5716 S   0.0   0.2   0:04.62 systemd
    2 root      20   0       0      0      0 S   0.0   0.0   0:00.03 kthreadd
    3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
    6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-kb
    7 root      20   0       0      0      0 I   0.0   0.0   0:00.25 kworker/u256:0-
    8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
    9 root      20   0       0      0      0 S   0.0   0.0   0:00.08 ksoftirqd/0
   11 root      rt   0       0      0      0 S   0.0   0.0   0:00.07 migration/0
   13 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
   14 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
   15 root      rt   0       0      0      0 S   0.0   0.0   0:00.05 migration/1
   16 root      20   0       0      0      0 S   0.0   0.0   0:00.11 ksoftirqd/1
   18 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/1:0H-kb
   19 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/2
   20 root      rt   0       0      0      0 S   0.0   0.0   0:00.06 migration/2
   21 root      20   0       0      0      0 S   0.0   0.0   0:00.25 ksoftirqd/2
   23 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/2:0H-kb
   24 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/3
   25 root      rt   0       0      0      0 S   0.0   0.0   0:00.02 migration/3
   26 root      20   0       0      0      0 S   0.0   0.0   0:00.04 ksoftirqd/3
   28 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/3:0H-kb
   30 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kdevtmpfs
   31 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
   32 root      20   0       0      0      0 S   0.0   0.0   0:00.10 kauditd
   36 root      20   0       0      0      0 S   0.0   0.0   0:00.00 khungtaskd
```

体会

通过本次实验，我对 Linux 有了初步的了解，能够掌握 Linux 系统的基本使用，同时，通过本次实验我了解了 Linux 内核的修改、添加系统调用以及内核编译方法，并通过自己的实验实现了两个系统调用。同时通过本次实验我还掌握了一些内核函数比如 set_user_nice()以及 get _task()，同时对进程的相关知识有了更加深入的理解。