Linux 内核实验报告

实验题目：新系统调用设计实验

学号： 200900301236 （辅修号： ）
日期：2012.5.11 班级： 09 软 1 姓名：王添枝
Email：tzwang2012@163.com

实验目的：学习如何产生一个系统调用以及怎样通过往内核中增加一个新函数从内核空间中实现对用户空间的读写。

硬件环境：
软件环境：ubuntu 10.10
linux 内核:2.6.35.13
gcc:4.4.5

实验步骤：

一、实验设计

1. 从网上下载内核源码 linux-2.6.35.13

2. /usr/src/linux-2.6.35.13/arch/x86/kernel/syscall_table_32.S 中增加：

```
.long sys_mysyscall
.long sys_mysyscall1
```

其中:sys_mysyscall 负责读取时间

sys_mysyscall1 负责读取从开机到现在的缺页次数

3. usr/src/linux-2.6.35.13/arch/x86/include/asm/unistd_32.h

中添加：

```
346 #define __NR_mysyscall          338
347 #define __NR_mysyscall1           339
348
349 #ifdef __KERNEL__
350
351 #define NR_syscalls 340
```

注意:要修改 NR_syscalls 的值为最后一个系统调用号加 1

4. /usr/src/linux-2.6.35.13/kernel/time/timekeeping.c 中添加系统调用函数：

```
void my_gettimeofday(struct timeval *tv,struct timespec *sv){
	struct timespec now;
	getnstimeofday(&now);
	tv->tv_sec = now.tv_sec;
	tv->tv_usec = now.tv_nsec/1000;
	sv->tv_sec = now.tv_sec;
	sv->tv_nsec = now.tv_nsec;
}
asmlinkage long sys_mysyscall(struct timeval *v_time,struct timespec *s_time){
	struct timeval kernel_v;
	struct timespec kernel_s;
	my_gettimeofday(&kernel_v,&kernel_s);
	if(copy_to_user(v_time,&kernel_v,sizeof(kernel_v))){
		return -1;
	}
	if(copy_to_user(s_time,&kernel_s,sizeof(kernel_s))){
		return -1;
	}
	return 2;
}
long page_fault_count=0;
EXPORT_SYMBOL(page_fault_count);
asmlinkage long sys_mysyscall1(){
	return page_fault_count;
}
```

其中 sys_mysyscall 用来获取时间，sys_mysyscall1 用来获取缺页中断次数，用 EXPORT_SYMBOL（page_fault_count）来声明一个全局变量来记录缺页次数。

5. **/usr/src/linux-2.6.35.13/arch/x86/mm/fault.c 中 do_page_fault(struct pt_regs *regs, unsigned long error_code)函数中每次调用+1**

```
extern long page_fault_count;
```

```
966	tsk = current;
967	mm = tsk->mm;
968	page_fault_count++;
```

二、编译内核

在 Ubuntu 系统中内核的编译和安装：

1) 首先下载和安装内核开发包

```
sudo apt-get install build-essential kernel-package libncurses5-dev fakeroot
```

2) 下载和安装 xconfig 图形配置界面需要的图形工具包

```
sudo aptitude install libqt3-mt-dev libqt3-compat-headers libqt3-mt
```

3) 把内核源代码解压到/usr/src 目录下

```
sudo tar xzvf linux-2.6.xx.tar.gz  -c /usr/src
cd /usr/src/linux-x.x.xx
```

4) 配置内核编译参数

```
sudo make xconfig
```

5) 编译生成内核和内核初始化解压程序。完成后会在/usr/src/linux-x.x.xx 下生成内核和内核头文件的 deb 安装包

```
sudo make-kpkg --rootcmd fakeroot --initrd  kernel_image kernel_headers
```

6) 把内核的 deb 安装包安装到系统中，并重新配置/boot/grub/grub.conf 启动文件

```
cd /usr/src
sudo dpkg -i linux-headers-2.6.xx.Custom_i386.deb
sudo dpkg -i linux-image-2.6.xx.Custom_i386.deb
```

7) 用新内核重新启动系统

```
sudo reboot
```

注意：在多次安装同一版本的内核的时候，要先删除这一版本的内核，否则可能安装后只有命令行界面而没有图形界面。

三、调试记录

```
wangtianzhi@wangtianzhi-NV48:~/桌面/linux内核实验/实验五$ ./parta
timeval.tv_sec=1336732079
timeval.tv_usec=864073
timespec.tv_sec=1336732079
timespec.tv_nsec=864073923
使用gettimeofday()得到的结果: tv_sec=1336732079,tv_usec=864080
缺页中断次数: strat=2098241,end=2098262,total=21
```

四、结论分析与体会

本次实验做了较长时间，编译一次内核就要两个多小时。当然，如果能够先精简内核，那么可能编译也会快很多。

虽然过程很艰辛，但是最终还是得到了想要的结果。通过此次实验也对内核的编译和在内核中添加系统调用有了较深的了解，巩固了课堂所学的知识。

程序完整源代码：（用户空间访问系统调用源码）

```
#include <stdio.h>
#include <errno.h>
#include <sys/time.h>
#include <linux/unistd.h> //不同内核版本可能位置不同
```

```
struct timeval v_time,now;
struct timespec s_time;
long start,end;
int main()
{
   start=syscall(339);
   // mysyscall(&v_time,&s_time);
   syscall(338,&v_time,&s_time);
   gettimeofday(&now,NULL);
                printf("timeval.tv_sec=%ld\ntimeval.tv_usec=
%ld\n",v_time.tv_sec,v_time.tv_usec);
              printf("timespec.tv_sec=%ld\ntimespec.tv_nsec=
%ld\n",s_time.tv_sec,s_time.tv_nsec);
       printf("使用gettimeofday()得到的结果： tv_sec=
%ld,tv_usec=%ld\n",now.tv_sec,now.tv_usec);
    int i;
    for(i=0;i<10000;i++);
    end=syscall(339);
       printf("缺页中断次数： strat=%ld,end=%ld,total=
%d\n",start,end,end-start);
}
```

参考材料

linux 操作系统内核实习
linux 内核设计与实现