添加系统调用

实验目的

学习 Linux 内核的系统调用，理解、掌握 Linux 系统调用的实现框架、用户界面、参数传递、进入/返回过程。阅读 Linux 内核源代码，通过添加一个简单的系统调用实验，进一步理解 Linux 操作系统处理系统调用的统一流程。

实验内容

在现有的系统中添加一个不用传递参数的系统调用。这个系统调用的功能是实现遍历进程。实验主要内容：

- 添加系统调用的名字
- 利用标准 C 库进行包装
- 添加系统调用号
- 在系统调用表中添加相应表项
- sys_mysyscall 的实现
- 编写用户态测试程序

实验指导

一个添加新的系统调用的步骤：

1. 下载并部署内核源代码

2. 添加系统调用号

系统调用号在文件 unistd.h 里面定义。这个文件可能在你的系统上会有两个版本：一个是 C 库文件版本，出现的地方是在/usr/include/unistd.h 和/usr/include/asm/unistd.h（注意：不同的发行版本文件名可能不一样）；另外还有一个版本是内核自己的 unistd.h，出现的地方是在你解压出来的 2.6.15 内核代码的对应位置（比如/usr/src/linux/include/asm/unistd.h）（注意：不同内核版本的文件名可能不一样）。当然，也有可能这个 C 库文件只是一个到对应内核文件的连接。现在，你要做的就是在文件 unistd.h 中添加我们的系统调用号：__NR_mysyscall，如下所示：

include/asm-i386/unistd.h

/usr/include/asm/unistd.h

```
231 #define __NR_mysyscall              223     /* mysyscall adds here */
```

/* 注意：不同版本的内核系统调用号不一样，您可以根据内核版本不同对系统调用号进行修改*/

添加系统调用号之后，系统才能根据这个号，作为索引，去找 syscall_table 中的相应表项。所以说，我们接下来的一步就是：

3. 在系统调用表中添加相应表项

我们前面讲过，系统调用处理程序（system_call）会根据 eax 中的索引到系统调用表（sys_call_table）中去寻找相应的表项。所以，我们必须在那里添加我们自己的一个值。

arch/i386/kernel/syscall_table.S

```
……
233     .long sys_mysyscall
234     .long sys_gettid
235     .long sys_readahead             /* 225 */
……
```

到现在为止，系统已经能够正确地找到并且调用 sys_mysyscall。剩下的就只有一件事情，那就是 sys_mysyscall 的实现。

4. sys_mysyscall 的实现

我们把这一小段程序添加在 kernel/sys.c 里面。在这里，我们没有在 kernel 目录下另外添加自己的一个文件，这样做的目的是为了简单，而且不用修改 Makefile，省去不必要的麻烦。

mysyscall 系统调用实现遍历系统中所有进程，并打印每个进程的进程名字(name)、进程标识符(pid)、进程的状态和父进程的名字。

进程名字、pid、进程状态、父进程的指针在 task-struct 结构的字段中。在内核中使用 printk 函数打印有关变量的值。遍历进程可以使用 next_task 宏，init_task 进程为 1 号进程。

```
asmlinkageintsys_mysyscall(void)
{
        //在此处加入遍历进程的代码;
        return 0;
}
```

5. 重新编译内核

一定要重新编译内核。内核编译完成后，重新启动编译后的新内核。

6. 编写用户态程序

要测试新添加的系统调用，需要编写一个用户态测试程序（test.c）调用 mysyscall 系统调用。mysyscall 系统调用中 printk 函数输出的信息在/var/log/messages 文件中。

用户态程序

```
#include <linux/unistd.h>
_syscall0(int,mysyscall)          /* 注意这里没有分号 */
int main()
{
        mysyscall();      /*这个系统调用的作用是实现遍历进程 */
        //在此加入在屏幕输出每个进程相关信息的代码；
}
```

在 Linux 2.6.25 以后内核中，宏_syscall0()至_syscall6()不再定义在/usr/include/asm/unistd.h 中，因此需要使用 syscall()函数实现系统调用。

用户态测试程序可以用如下方法实现

```
#include <linux/unistd.h>
# include <sys/syscall.h>
#define __NR_mysyscall 223
int main()
{
        syscall(__NR_mysyscall);   /*或 syscall(223)  */
        //在此加入在屏幕输出每个进程相关信息的代码；
}
```

- 用 gcc 编译源程序

  # gcc –o test test.c

- 运行程序

  # ./test

思考题（选做）

用内核模块（Kernel Module）方法完成替换某一个系统调用。该系统调用的功能由你自己定义。

撰写实验报告的要求

1. 按照实验报告模板格式撰写；

2. 整个实验过程的截图；

3. 源程序的修改部分，运行结果的截图；

4. 实验过程中遇到的问题及解决方法等；

5. 心得体会。