## 实验目的

* 掌握虚拟文件系统的实现原理；
* 实践文件、目录、文件系统等概念。

## 实验内容

在 Linux 0.11 上实现 procfs（proc 文件系统）内的 psinfo 结点。当读取此结点的内容时，可得到系统当前所有进程的状态信息。

## 报告

|  |
| --- |
| * 如果要求你在 psinfo 之外再实现另一个结点，具体内容自选，那么你会实现一个给出什么信息的结点？为什么？ * 一次 read() 未必能读出所有的数据，需要继续 read()，直到把数据读空为止。而数次 read() 之间，进程的状态可能会发生变化。你认为后几次 read() 传给用户的数据，应该是变化后的，还是变化前的？ + 如果是变化后的，那么用户得到的数据衔接部分是否会有混乱？如何防止混乱？ + 如果是变化前的，那么该在什么样的情况下更新 psinfo 的内容？ |
| 最多三个参数，也就是通过寄存器ebx，ecx，edx，可以通过寄存器传递参数位置，也就是寄存器间接寻址来传递多个参数 |
| 修改 include/linux/sys.h  在sys\_call\_table最后加入sys\_foo，  再加上extern rettype sys\_foo();  修改include/unistd.h  添加 #define \_\_NR\_foo 系统调用号  修改 kernel/system\_call.s  nr\_system\_calls =系统调用总数目加1  在kernel中添加 foo.c  最后修改kernel的Makefile，以链接foo.c  调用处添加有  #define \_\_LIBRARY\_\_  #include <unistd.h>  \_syscallN宏展开系统调用 |

## 实验步骤

1. 修改 bootsect.s 更改字符串及其长度，运行并显示

|  |
| --- |
| 修改文件 |
|  |
| 运行效果 |
|  |

1. 修改bootsect.s和setup.s显示Now we are in SETUP

|  |
| --- |
| bootsect.s文件 |
|  |
| setup.s文件 |
|  |
| 运行效果 |
|  |

1. 修改 build.c 方便使用

|  |
| --- |
|  |

1. 读取内存大小、光标位置并显示

|  |
| --- |
| 关键代码 |
|  |
| 运行效果 |
|  |