## 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 |
|  |

## 实验步骤

1. 增加新文件类型

|  |
| --- |
| 2020-05-21 00-21-20 的屏幕截图2020-05-21 00-21-51 的屏幕截图 |

1. 修改mknod() 支持新文件类型

|  |
| --- |
| 2020-05-21 00-25-11 的屏幕截图 |

1. 修改初始化函数

|  |
| --- |
| 2020-05-21 00-26-21 的屏幕截图2020-05-21 00-26-48 的屏幕截图2020-05-21 00-29-14 的屏幕截图 |

1. 修改sys\_read() ，增加处理分支

|  |
| --- |
| 2020-05-21 00-33-54 的屏幕截图 |

注：实验指导中S\_IFPROC()应为S\_ISPROC()

1. proc 文件的处理函数

|  |
| --- |
| #include <stdarg.h>  #include <linux/fs.h>  #include <linux/sched.h>  #include <asm/segment.h>  char proc\_buf[4096];  int sprintf(char \*buf, const char \*fmt, ...)  {  va\_list args; int i;  va\_start(args, fmt);  i=vsprintf(buf, fmt, args);  va\_end(args);  return i;  }  int get\_psinfo()  {  int ans=0;  ans+=sprintf(proc\_buf+ans,"%s","PID\tPPID\tS\tPRI\tTTY\tTIME\n");  struct task\_struct \*\*p=NULL;  for(p = &LAST\_TASK ; p > &FIRST\_TASK ; --p)  {  if (\*p)  {  ans+=sprintf(proc\_buf+ans, "%d\t", (\*p)->pid);  ans+=sprintf(proc\_buf+ans, "%d\t", (\*p)->father);  ans+=sprintf(proc\_buf+ans, "%d\t", (\*p)->state);  ans+=sprintf(proc\_buf+ans, "%d\t", (\*p)->priority);  ans+=sprintf(proc\_buf+ans, "%d\t", (\*p)->tty);  ans+=sprintf(proc\_buf+ans, "%d\n", (\*p)->cutime+(\*p)->cstime);  }  }  return ans;  }  int get\_hdinfo()  {  struct super\_block \* sb;  unsigned int ans=0;  unsigned int i,j;  unsigned int blockused=0;  unsigned char tmp;  sb = get\_super(current->root->i\_dev);  for(i=0;i<sb->s\_zmap\_blocks;i++)  for(j=0;j<1024;j++)  for(tmp=sb->s\_zmap[i]->b\_data[j];tmp;tmp>>=1)  blockused+=1&tmp;  ans+=sprintf(proc\_buf+ans,"total\_blocks :%u\n", sb->s\_nzones);  ans+=sprintf(proc\_buf+ans,"Free blocks :%d\n", sb->s\_nzones-blockused);  ans+=sprintf(proc\_buf+ans,"Total inodes :%u\n", sb->s\_ninodes);  return ans;  }  int get\_inodeinfo()  {  int ans=0;  struct super\_block \* sb;  sb = get\_super(current->root->i\_dev);  ans += sprintf(proc\_buf+ans,"total counr:%d\n",sb->s\_ninodes);  return ans;  }  int proc\_read(int dev, unsigned long \* pos, char \* buf, int count)  {  int i;  int ans;  if(0 == \*pos)  if(dev==0)ans=get\_psinfo();  else if(dev==1)ans=get\_hdinfo();  else if(dev==2)ans=get\_inodeinfo();  else return 0;  for(i=0;i<count;i++,(\*pos)++)  if(\*pos>ans)return i;  else put\_fs\_byte(proc\_buf[\*pos], &buf[i]);  return i;  } |

1. 修改makefile文件

|  |
| --- |
| 2020-05-22 22-21-51 的屏幕截图2020-05-22 22-22-51 的屏幕截图 |

1. 测试

|  |
| --- |
| 2020-05-22 22-19-07 的屏幕截图2020-05-22 22-20-15 的屏幕截图2020-05-22 22-20-23 的屏幕截图2020-05-22 22-20-30 的屏幕截图 |