



# 第18章 proc文件系统



## 实验目的

- ❖ 了解proc文件系统的作用和原理
- ❖ 掌握proc编程，并通过该接口获取进程和内核信息



# 主要内容

## ❖ 背景知识

- proc文件系统简介
- proc文件系统数据结构

## • 实验内容

- 向proc文件系统中添加可读写文件
- 通过proc文件系统查看进程信息



## 什么是proc文件系统?

- 早期UNIX系统在设备文件目录/dev下设置了/dev/mem文件，目的是提供一种便捷的用户和内核之间的通信和交互方式。/dev/mem以文件系统作为使用界面，通过它可以读/写系统的整个物理主存，而物理主存的地址就用做读写时文件内的位移量，可像普通文件一样，进行读、写等常规文件操作，从而提供一个在用户空间动态地读写内核数据结构的方法。
- 该方法既可以用于收集状态信息和统计信息，也可以用于程序调试或修改内核数据结构及变量值。在UNIX的发展过程中，该功能被进一步加强，在系统根目录下建立/proc目录，演变成一个特殊的文件系统/proc，该文件系统目录中的特殊文件包含以下内容：





## /proc文件系统目录中的内容(1)

- • 资源管理信息。

如 /proc / slabinfo是主存管理中关于各个slab缓冲块的信息、 /proc / swaps是系统的swap设备的信息、 /proc / partitions是磁盘分区的信息。

- • 设备相关信息。

如 /proc / pci是PCI总线上所有设备的列表信息。

- • 文件系统信息。

如 /proc / mounts是系统中已经安装的文件系统设备的列表， /proc / filesystems则是已经登记的每种文件系统（类型）的清单信息。

- • 中断编号信息。

/proc / interrupts是关于中断源和它们的中断向量编号的清单信息。



## /proc文件系统目录中的内容(2)

- • 模块相关信息。

/proc/modules是系统中已安装的动态模块的清单，  
/proc/ksyms则是内核中供可安装模块动态连接的符号名及其地址的清单信息。

- • 系统版本信息。

包括号系统版本号，及其他各种统计与状态信息。

- • 进程相关信息。

每个进程都有一个子目录（以pid命名），子目录中包括关于该进程命令行、环境变量、cpu占用时间、主存映射表、已打开文件的描述符、进程根目录及当前目录、进程的文件链接、及进程状态信息等。



## /PROC文件系统使用方法

- 应用程序只要有正确的访问权限，就可使用/proc文件系统读写进程空间及操作系统整体组织及状态信息。
- /proc文件系统中，大多数文件都是只读的，用户只能从这些文件中读取内核中的信息，但也有一些/proc文件被设置成可写，用户可以通过写这些/proc文件将参数传递给内核，结果是直接修改了内核相应变量值。
- 除系统已经提供的子目录和文件，/proc还留有接口，允许用户在内核中创建新的子目录或文件，从而与用户程序共享信息。





# /PROC文件系统部分列表

```
min@min-laptop: /proc
File Edit View Terminal Tabs Help
min@min-laptop:/proc$ ls
1      2680  3039  3148  3232  801      driver      meminfo      sysvipc
1016   2685  3040  3150  3237  817      execdmain   misc          tty
115    2697  3041  3152  3239  840      filesystems modules        uptime
172    2725  3042  3159  3241  9        fs           mounts        version
173    2726  3043  3164  3243  acpi      ide          mtrr          vmstat
174    2727  3044  3169  3246  asound    interrupts  net           zoneinfo
175    2756  3066  3171  3247  buddyinfo iomem        partitions
2      2762  3068  3176  3272  bus       ioports      pci
2565   2807  3095  3185  4      cmdline     irq          scsi
2617   2915  3097  3194  5      cpuinfo     kallsyms     self
2643   2937  3139  3198  6      devices     kcore        slabinfo
2645   2950  3142  3208  762    diskstats  kmsg         stat
2664   2963  3143  3213  798    dma         loadavg      swaps
2679   3     3145  3215  8      dri         locks        sys
```

图 18-1 /proc 文件系统





## proc文件系统部分子目录及文件(1)

- /proc/cmdline文件：给出了内核启动的命令行。
- /proc/cpuinfo文件：提供了有关系统CPU的多种信息。文件列出了CPU的型号，以及能得到的更多特定信息（制造商，型号和版本）。该文件还包含了以bogomips表示的处理器速度。
- /proc/devices文件：列出字符和块设备的主设备号，以及分配到这些设备号的设备名称。
- /proc/dma文件：列出由驱动程序保留的DMA通道和保留它们的驱动程序名称。casade项供用于把次DMA控制器从主控制器分出的DMA行所使用。



## proc文件系统部分子目录及文件(2)

- /proc/filesystems文件：列出可供使用的文件系统类型，一种类型一行。虽然它们通常是编入内核的文件系统类型，但该文件还可以包含可加载的内核模块加入的其它文件系统类型。
- /proc/interrupts文件：每一行都有一个保留的中断。每行中的域有：中断号，本行中断的发生次数，可能带有一个加号的域（SA\_INTERRUPT标志设置），以及登记这个中断的驱动程序的名字。可以在安装新硬件前，像查看/proc/dma和/proc/ioports一样用cat命令手工查看手头的这个文件。
- /proc/ioports文件：这个文件列出了诸如磁盘驱动器，以太网卡和声卡设备等多种设备驱动程序登记的许多I/O端口范围。
- /proc/kcore文件：是系统的物理主存以core文件格式保存的文件。例如，GDB能用它考察内核的数据结构。它不是纯文本，而是/proc目录下为数不多的几个二进制格式的项之一。



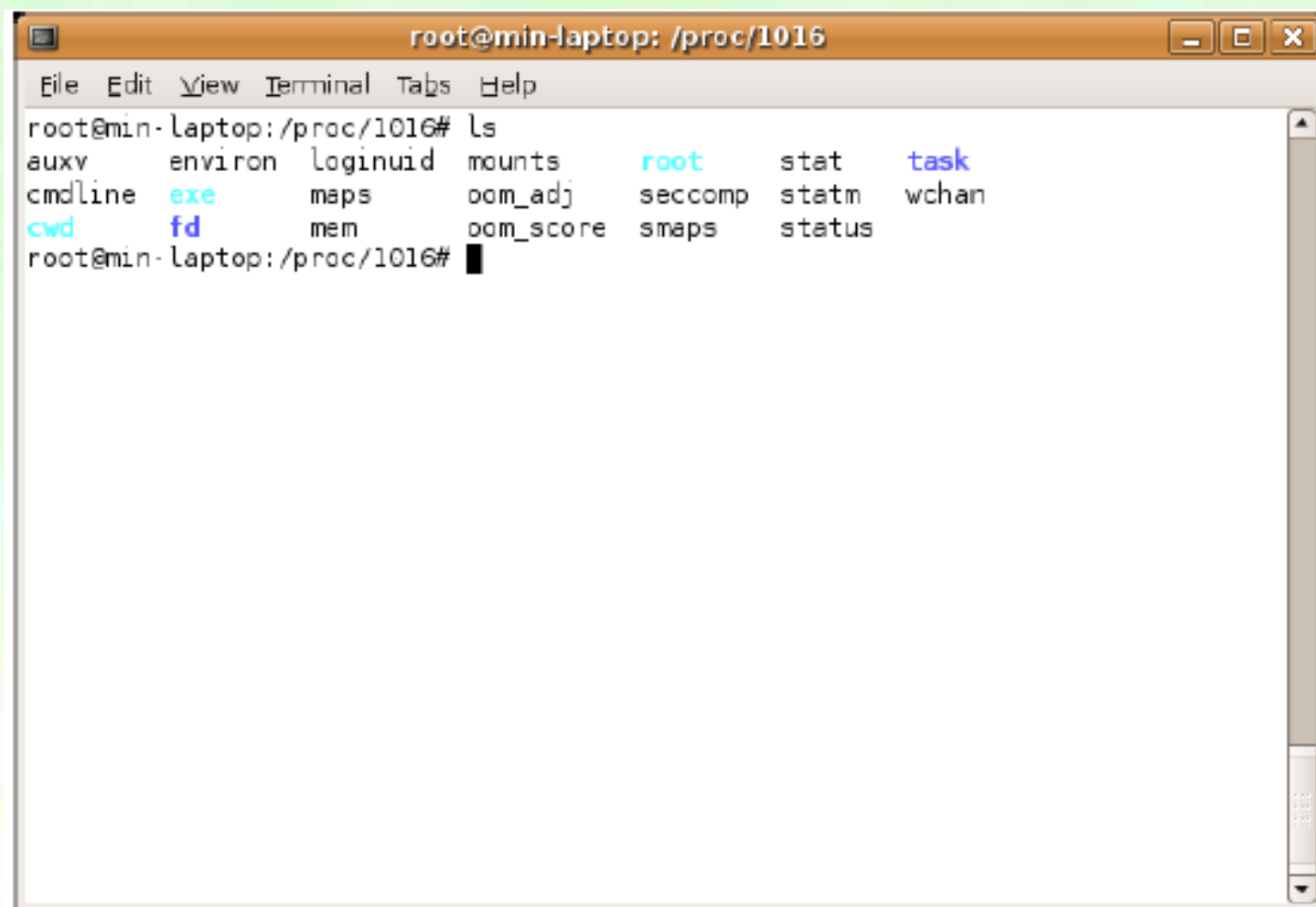
## proc文件系统部分子目录及文件(3)

- /proc/kmsg文件：用于检索用printk生成的内核消息。任何时刻只能有一个具有超级用户权限的进程可以读取这个文件。也可以用系统调用syslog检索这些消息。通常使用工具dmesg或守护进程klogd检索这些消息。
- /proc/ksyms文件：列出了已经登记的内核符号，这些符号给出了变量或函数的地址。每行给出一个符号的地址，符号名称以及登记这个符号的模块。程序ksyms, insmod和kmod使用这个文件。它还列出了正在运行的任务数，总任务数和最后分配的PID。
- /proc/loadavg文件：给出以几个不同的时间间隔计算的系统平均负载，这就如同uptime命令显示的结果那样。前3个数字是平均负载。这是通过计算过去1分钟，5分钟，15分钟里运行队列中的平均任务数得到的，随后是正在运行的任务数和总任务数，最后是上次使用的进程号。
- /proc/locks文件：包含在打开的文件上的加锁信息，文件中的每一行描述了特定文件和文档上的加锁信息以及对文件施加的锁的类型。内核也可以需要时对文件施加强制性锁。





## proc文件系统中进程子目录包含信息(1)



```
root@min-laptop: /proc/1016
File Edit View Terminal Tabs Help
root@min-laptop:/proc/1016# ls
auxv      environ  loginuid  mounts    root      stat      task
cmdline   exe      maps      oom_adj   seccomp   statm     wchan
cwd        fd       mem       oom_score snaps     status
root@min-laptop:/proc/1016#
```

图 18-2 /proc 文件系统中每个进程子目录包含信息





## proc文件系统中进程子目录包含信息(2)

- /proc文件系统中的每个进程子目录中包含的主要内容。  
假定“/proc/pid”为以进程标示pid为第1级子目录，  
每一进程下都包含相应的进程状态文件，
  - /proc/pid/status: 相应进程的状态
  - /proc/pid/ctl: 相应进程的控制文件
  - /proc/pid/psinfo: 相应进程的ps 信息
  - /proc/pid/as: 相应进程的地址空间
  - /proc/pid/map: 相应进程的映射信息



## 18.2 背景知识

### proc文件系统中进程子目录包含信息(3)

- /proc/pid/object: 相应进程的对象信息
- /proc/pid/sigact: 相应进程的信号量操作
- /proc/pid/sysent: 相应进程的系统调用信息
- /proc/pid/lwp/tid: 相应进程的核心线程标示符目录
- /proc/pid/lwp/tid/lwpstatus: 核心线程的状态
- /proc/pid/lwp/tid/lwpctl: 核心线程控制文件
- /proc/pid/lwp/tid/lwpsinfo: 核心线程的ps 信息



# 主要内容

## ❖ 背景知识

- proc文件系统简介
- **proc文件系统数据结构**

## • 实验内容

- 向proc文件系统中添加可读写文件
- 通过proc文件系统查看进程信息



# proc文件系统数据结构(1)

- /proc文件系统中，代表各个文件的是proc\_dir\_entry结构，该结构和文件系统中的dir\_entry类似，管理对用户空间和核心空间的/proc文件的驱动。
- proc\_dir\_entry是/proc文件系统中最重要的数据结构，系统初始化时，建立/proc文件系统树，/proc树保存完成读写/proc文件系统所需要的几乎所有关系、属性、操作函数指针等。
- 由于/proc文件系统只存在主存里，在读操作时，不像其他文件系统一样从辅存中获取inode信息，而是从/proc树中读取对应inode，然后再调用inode中登记的函数，动态地从内核读取所需信息。





## proc文件系统数据结构(2)

- 系统启动后，创建由proc\_dir\_entry结构形成的文件系统树，每当从用户空间读取/proc目录下的文件时，内核根据读取的文件映射到对应的函数，动态地获得内核数据。
- 除提供读出功能，/proc文件系统的部分文件还提供写入功能。/proc文件系统的超级块和普通文件系统的超级块不同，它不需要从硬件中获得超级块中的数据，而是在内核启动时直接初始化超级块数据，从而完成系统对/proc文件系统操作函数的初始化过程。
- /proc文件系统的编程接口与VFS类似，大部分/proc文件系统中的函数是VFS的函数名加一个“proc\_”。例如，在/proc文件系统中创建目录的函数是proc\_mkdir()，创建符号链接的函数是proc\_symlink()，创建设备文件的函数是proc\_mknod()。



## proc文件系统数据结构(3)

- /proc文件系统中，proc\_dir\_entry数据结构的定义：
  - struct proc\_dir\_entry { /\*描述/proc文件系统的目录结构\*/
  - unsigned short low\_ino;
  - unsigned short namelen;
  - const char \*name;
  - mode\_t mode;
  - nlink\_t nlink;
  - uid\_t uid;
  - gid\_t gid;
  - unsigned long size;
  - struct inode\_operation \*proc\_iops;
  - struct file\_operations \*proc\_fops;
  - get\_info\_t \*get\_info;



## 文件系统数据结构(4)

- /proc文件系统中，proc\_dir\_entry数据结构的定义(续)：

- struct module \*owner;
- struct proc\_dir\_entry \*next, \*parent, \*subdir;
- void \*data;
- read\_proc\_t \*read\_proc;
- write\_proc\_t \*write\_proc;
- atomic\_t count;
- int deleted;
- kdev\_t rdev;
- };



## proc文件系统数据结构(5)

- 在该结构中，常用到的一些成员：

- name: /\*proc文件名\*/
- namelen: /\* proc文件名长度\*/
- mode: /\*文件的类型和权限\*/
- nlink: /\*读文件的链接数\*/
- count: /\*用户计数\*/
- proc\_iops: /\*inode节点操作函数\*/
- proc\_fops : /\*file文件操作函数\*/
- read\_proc: /\*读操作函数\*/
- write\_proc: /\*写操作函数\*/
- data: /\*保存与目录相关的数据\*/
- owner: /\*该文件的拥有者模块\*/





## proc文件系统数据结构(6)

- 一个这样的数据结构代表一个/proc文件，很多成员的含义与普通文件的相同。当用户读写/proc文件时，该文件所关联的读/写函数就会被激活，所以proc\_dir\_entry结构中的读/写函数应该预先设置好。
- 读文件接口的函数原型是：

```
int (*read_proc) (char *page, char **start,  
off_t off, int count, int *eof, void *data);
```

参数page指针是写数据给用户的数据缓冲区；start指明要把数据写在哪一个位置；offset指定写入数据相对于缓冲区的偏移量；count为写入的字节数；eof是一个返回参数，当文件指针已经指向文件尾时，该标志被置位；data一般用作保存私有数据。



## proc文件系统数据结构(7)

- 在读文件接口中有两个偏移量：start和offset，它们有各自的用途。/proc文件只有单个主存页面供用户数据传输。这样就把用户文件的大小限制在4KB。start参数在这里就是用来实现大数据量文件的。如果read\_proc()函数不对start指针进行设置，内核就会假定offset参数被忽略，并且数据页包含返回给用户空间的整个文件。反之，如果需要通过多个片段创建更大的文件，则可以把start赋值为page，这样内核就能知道新数据位于缓冲区的开始位置，并会跳过前offset字节的数据。
- 写文件接口的函数原型是：  

```
int (*write_proc) (struct file *file, const char *buffer, unsigned long count, void *data);
```

该函数将从buffer开始的缓冲区中的count个字节写入file。由于buffer一般是用户空间的指针，指向的是用户空间缓冲区。因此，应该使用copy\_from\_user()将数据拷贝到核心空间中来。



## proc文件系统数据结构(8)

- 定义read\_proc()和write\_proc()函数后，需要将它们加载到/proc文件系统中。加载的第1步是获得一个proc\_dir\_entry结构，获得该结构的函数是：

```
struct proc_dir_entry *create_proc_entry(const  
char *name, mode_t mode, struct proc_dir_entry  
*parent );
```

参数name是要创建的文件名；mod是文件的保护掩码；parent指出要创建的文件目录(如果parent是NULL, 文件在/proc根目录下创建)；函数的返回值是要创建的/proc文件所对应的proc\_dir\_entry。

加载的第2步就是对数据结构设置read\_proc()和write\_proc()指针。如果想创建一个只读的proc文件，还有一个更加方便的函数：

```
struct proc_dir_entry  
*create_proc_read_entry(const char *name, mode_t  
mode, struct proc_dir_entry *base, read_proc_t  
*read_proc, void *data);
```





## proc文件系统数据结构(9)

- 该函数实际上就是调用create\_proc\_entry( )，并将返回结构中的read\_proc域设置成read\_proc，data域设置成data。
- /proc文件系统还提供多个函数方便在/proc文件系统中创建或者删除文件或目录，主要的函数有：
  - proc\_mkdir( )                   /\*创建目录\*/
  - remove\_proc\_entry( )       /\*删除文件或目录\*/
  - proc\_symlink( )           /\*创建符号链接\*/
  - proc\_mknod ( )           /\*创建设备文件\*/





# 主要内容

## ❖ 背景知识

- proc文件系统简介
- proc文件系统数据结构

## • 实验内容

- 向proc文件系统中添加可读写文件
- 通过proc文件系统查看进程信息



# • 实验1 向proc文件系统中添加可读写文件

## • 实验说明

- 编写一个模块，当模块被加载时，该模块会向/proc文件系统中添加一个文件。用户可以往该文件中输入一个字符串。当用户读取该文件时，返回用户前一次输入的字符串。

## • 解决方案

- 定义数据结构
- 创建、删除/PROC文件
- 验证正确性





# 主要内容

## ❖ 背景知识

- proc文件系统简介
- proc文件系统数据结构

## • 实验内容

- 向proc文件系统中添加可读写文件
- 通过proc文件系统查看进程信息



## • 实验2 通过proc文件系统查看进程信息

### • 实验说明

- 编写一个模块，当模块被加载时，模块会向/proc文件系统中添加一个文件（/proc/task）。用户可以往该文件中输入需要查看的进程pid，当用户读取该文件时，返回进程控制块中的信息。

### • 解决方案

- /PROC文件操作
- 传递字符串形式的进程pid
- 显示进程信息