Linux操作系统编程

文件属性管理

粘着位

- **在UNIX早期版本中，有一位被称为粘住位，如果一可执行程序文件的这一位被设置了，那么在该程序第一次执行并结束时，该程序正文被保存在交换区中，这使得下次执行该程序时能较快地将其装入内存。**
- **现今较新的UNIX系统大多数都具有虚存系统以及快速文件系统，所以不再需要使用这种技术**
- **如果对一个目录设置了粘住位，则只有对该目录具有写许可权的用户并且满足下列条件之一，才能删除或更名该目录下文件：**
  - 拥有此文件
  - 拥有此文件
  - 是超级用户
- **目录/tmp和/var/spool/uucppublic是设置粘住位的候选者。这两个目录是任何用户都可在其中创建文件的目录，对任一用户(用户、组和其他)的许可权通常都是读、写和执行。但是用户不应能删除或更名属于其他人的文件，为此在这两个目录的文件方式中都设置了粘住位。**

文件长度

- **stat结构的成员st_size包含了以字节为单位的该文件的长度。此字段只对普通文件、目录文件和符号连接有意义**
- **对于普通文件，其文件长度可以是0，在读这种文件时，将得到文件结束指示**
- **对于目录，文件长度通常是一个数，例如16或512的整倍数**
- **对于符号连接，文件长度是在文件名中的实际字节数。例如**：
  - lrwxrwxrwx 1 root 7 Sep 25 07:14 lib -> usr/lib
  - 其中，文件长度7就是路径名usr/lib的长度

文件截断

- **函数原型**：

```
int truncate(const char* pathname, off_t length);
int ftruncate(int fd, off_t length);
```

- **函数说明**：

用于改变文件的长度。pathname：欲改变长度的文件的文件名；fd：欲改变长度的文件的文件描述符；length：要设置的文件的新长度

- **返回值**：

返回值：成功返回0，出错返回-1

- **注意事项**：

当文件以前的长度>length时，则超过length以外的数据将不复存在

当文件以前的长度<length时，在文件以前长度到length之间，将形成空洞，读该区域，将返回0

文件截断

- **常见问题**:

  truncate和ftruncate函数并未实质性的向磁盘写入数据，只是分配了一定的空间供当前文件使用。当fd<length时，此时如果使用十六进制编辑工具打开该文件，会发现文件末尾多了很多00，这就是执行这个函数后的效果。如果发生系统复位或者装置掉电以后，该函数所产生的作用将被文件系统忽略，也就是说它所分配的空间将不能被识别，文件的大小将会是最后一次写入操作的区域大小，而非ftruncate分配的空间大小，也就是说，文件大小有可能会被改变

- **解决方案**:

  可以在执行完ftruncate之后，在新空间的末尾写入一个或以上字节的数据（不为0x00），这样新空间则不为空，文件系统会把这部分空间当成这个文件的私有空间处理，而不会出现文件大小改变的错误。

根据用户ID获取用户属性

- **常用函数**：getpwuid
- **头文件**： sys/types.h，pwd.h
- **函数原型**：struct passwd *getpwuid(uid_t uid);
- **函数说明**：

输入用户ID，返回用户属性信息（passwd结构）

```
struct  passwd{
        char * pw_name;       /* 用户名*/
        char * pw_passwd;     /* 密码.*/
        __uid_t pw_uid;       /* 用户ID.*/
        __gid_t pw_gid;       /*组ID.*/
        char *  pw_gecos;     /*真实名*/
        char *  pw_dir;       /* 主目录.*/
        char *  pw_shell;     /*使用的shell*/};
```

根据组ID获取组属性

- **常用函数**： getgrgid
- **头文件**： sys/types.h，grp.h
- **函数原型**： struct passwd *getgrgid(gid_t gid);
- **函数说明**：

输入用户组ID，返回用户组属性信息（group结构）

```
struct group{
    char *gr_name; /*组名称*/
    char *gr_passwd; /* 组密码*/
    gid_t gr_gid; /*组ID*/
    char **gr_mem; /*组成员账号*/ }
```

设备特殊文件

- **每一个文件系统所在的存储设备都由其主、次设备号表示。设备号所用的数据类型是基本系统数据类型dev_t**
- **主设备号标识设备驱动程序，次设备号标识特定的子设**
- **通常可以使用两个宏major和minor来访问主、次设备号**
  - 早期系统用16位整型存放设备号：8位用于主设备号，8位用于次设备号
  - FreeBSD 8.0和Mac OS X 10.6.8使用32位整型，其中8位表示主设备号，24位表示次设备号。
  - 32位系统中，Solaris 10用32位整型表示dev_t，其中14位用于主设备号，18位用于次设备
  - 64位系统中，Solaris 10用64位整型表示dev_t，主设备号和次设备号各为32位
  - 在Linux 3.2.0上，dev_t是64位整型，12位用于主设备号，20位用于次设备号
- **系统中与每个文件名关联的st-dev值是文件系统的设备号，该文件系统包含了这一文件名以及与其对应的索引结点**
- **只有字符特殊文件和块特殊文件才有st-rdev值，此值包含实际设备的设备号**