Чтение атрибутов файлов в ОС Unix

А. А. Вылиток

Функции *stat*, *lstat* и *fstat* возвращают атрибуты заданного файла:

```
#include <sys/types.h>
#include <sys/stat.h>
stat  ( char *path, struct stat *buf )
lstat ( char *path, struct stat *buf ) // не преобразует символич.ссылки
fstat ( int fd, struct stat *buf )     // возвращают 0 или -1
```

Усеченное описание структуры «*stat*» (полное см. в «*sys/stat.h*») выглядит следующим образом:

```
struct stat
{
   dev_t     st_dev;     // идентификатор файловой системы
   ino_t     st_ino;     // номер индексного дескриптора файла
   u_short   st_mode;    // содержит тип файла и флаги доступа
   short     st_nlink;   // значение счетчика жестких связей
   uid_t     st_uid;     // идентификатор владельца файла
   gid_t     st_gid;     // идентификатор группы
   dev_t     st_rdev;    // содержит старший и младший номера устройства
   off_t     st_size;    // размер файла в байтах
   time_t    st_atime;   // время последнего доступа
   time_t    st_mtime;   // время последней модификации
   time_t    st_ctime;   // время последнего изменения индексного дескриптора
};
```

Следующие макросы служат для работы с флагами поля «*st_mode*»:

```
#define S_IFMT    0170000 // позволяет “вырезать” тип файла
#define S_IFDIR   0040000 // каталог
```

```
#define S_IFCHR   0020000 // байт-ориентированный
#define S_IFBLK   0060000 // блок-ориентированный
#define S_IFREG   0100000 // обычный
#define S_IFLNK   0120000 // символическая ссылка
#define S_IFSOCK  0140000 // сокет
#define S_ISUID   0004000 // установка идентификатора пользователя при выполнении (изменяется eUID процесса)
#define S_ISGID   0002000 // установка идентификатора группы при выполнении (изменяется eGID процесса)
#define S_ISVTX   0001000 // для обычного файла - сохранить текст программы (машинный код) в ОП после завершения процесса;
/* для каталога - запретить обычному пользователю, не являющемуся владельцем данного каталога, удалять или переименовывать
```

Функция «*getpwuid*» позволяет преобразовать идентификатор пользователя в его имя

```
#include <pwd.h>

struct passwd *getpwuid ( int uid )
```

Структура *passwd* описана следующим образом (полное описание в «pwd.h»):

```
struct passwd
{
    char *pw_name;    // имя пользователя
    char *pw_passwd;  // пароль
    int   pw_uid;     // идентификатор пользователя
    int   pw_gid;     // идентификатор группы
    char *pw_dir;     // домашний каталог пользователя
    char *pw_shell;   // shell запускаемый после регистрации (по умолчанию "/bin/sh")
};
```

Программа эмуляции команды «*ls –l* $file_1$ $file_2$ ... $file_N$»:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <pwd.h>
#include <stdio.h>

#ifndef  MAJOR
  #define MINOR_BITS
  #define MAJOR(dev) ((unsigned) dev >> MINOR_BITS)
  #define MINOR(dev) (dev & MINOR_BITS)
```

```
#endif

// показать тип файла в первой позиции выходной строки
void display_file_type ( int st_mode )
{
    switch ( st_mode & S_IFMT )
    {
        case S_IFDIR:  putchar ( 'd' ); return;
        case S_IFCHR:  putchar ( 'c' ); return;
        case S_IFBLK:  putchar ( 'b' ); return;
        case S_IFREG:  putchar ( '-' ); return;
        case S_IFLNK:  putchar ( 'l' ); return;
        case S_IFSOCK: putchar ( 's' ); return;
    }
}

// показать права доступа для владельца, группы и прочих пользователей, а также все спец.флаги
void display_permission ( int st_mode )
{
  static const char xtbl[10] = "rwxrwxrwx";
  char     amode[10];
  int      i, j;

  for ( i = 0, j = ( 1 << 8 ); i < 9; i++, j >>= 1 )
    amode[i] = ( st_mode&j ) ? xtbl[i]: '-';
  if ( st_mode & S_ISUID )   amode[2]= 's';
  if ( st_mode & S_ISGID )   amode[5]= 's';
  if ( st_mode & S_ISVTX )   amode[8]= 't';
  amode[9]='\0';
  printf ( "%s ",amode );
}

// перечислить атрибуты одного файла
void long_list ( char * path_name )
{
  struct stat     statv;
  struct passwd  *pw_d;

  if ( lstat ( path_name, &statv ) )
  {
    perror ( path_name );
    return;
  }
  display_file_type ( statv.st_mode );
  display_permission ( statv.st_mode );
```

```
  printf ( "%d ",statv.st_nlink );  // значение счетчика жестких связей
  pw_d = getpwuid ( statv.st_uid ); // преобразовать UID в имя пользователя
  printf ( "%s ",pw_d->pw_name );   // и напечатать его

  if (
      ( statv.st_mode & S_IFMT) == S_IFCHR  ||
      ( statv.st_mode & S_IFMT) == S_IFBLK
     )
    // показать старший и младший номера устройства
    printf ( "%d, %d",MAJOR(statv.st_rdev), MINOR(statv.st_rdev) );
  else
    // или размер файла
    printf ( "%d", statv.st_size );
  //  показать имя файла
  printf ( "     %s\n", path_name );
}

// главный цикл отображения атрибутов для каждого файла
int main ( int argc, char * argv[] )
{
  if ( argc == 1 )
    fprintf ( stderr, "usage: %s <path name> ...\n", argv[0] );
  else
    while ( argc-- != 1 )
      long_list ( *++argv );
  return 0;
}
```