2.2 cp 命令设计与实现 (二)_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 2.2 cp 命令设计与实现（二）涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

- 基本思路:
  - step 1 : 调用 stat 函数，获取文件属性, 并会保存到 struct stat 结构体中
  - step 2 : 将文件属性中的文件类型信息保存到自定义结构体中 struct cp_file_info
- 获取文件属性一般使用 stat 函数 与 lstat 函数
  - stat 函数适用于通用文件
  - lstat 专门针对 链接文件
  -
- stat 函数

函数头文件

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

函数功能

获取文件属性，并将文件属性信息保存到 struct stat 结构体中

函数原型

int stat(const char *pathname, struct stat *statbuf);

函数参数

- pathname : 文件绝对路径
- statbuf : 文件属性结构体的指针, 具体定义如下

struct stat { dev_t st_dev; /* ID of device containing file / ino_t st_ino; / Inode number / mode_t st_mode; / 文件类型与权限 / nlink_t st_nlink; / 硬链接 / uid_t st_uid; / 用户 ID / gid_t st_gid; / 用户组 ID / dev_t st_rdev; / Device ID (if special file) / off_t st_size; / 文件大小 / blksize_t st_blksize; blkcnt_t st_blocks; struct timespec st_atime; / 最后访问时间 / struct timespec st_mtime; / 最后修改时间 */ struct timespec st_ctime; /* 最后状态修改时间 */ }

函数返回值

成功：返回 0

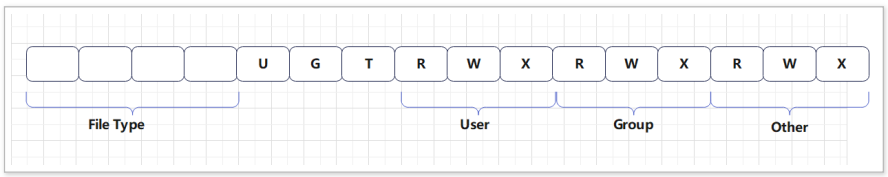
失败：返回 –1, 并设置 errno

stat 函数的使用

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

```
#include <unistd.h>


int main(void)
{
    int ret;
    struct stat statbuf;

    ret = stat("./test.txt",&statbuf);
    if (ret == -1){
        perror("stat(): ");
        return -1;
    }

    printf("inode number : %ld\n",statbuf.st_ino);
    printf("size : %ld\n",statbuf.st_size);

    return 0;
}
```

- 文件类型保存在 struct stat 结构体中的 st_mode 成员中 ， 具体存储在第 [15:12] bit 中, 剩下 [8:0] 为权限位




```
#define S_IFMT  00170000
#define S_IFSOCK 0140000
#define S_IFLNK  0120000
#define S_IFREG  0100000
#define S_IFBLK  0060000
#define S_IFDIR  0040000
#define S_IFCHR  0020000
#define S_IFIFO  0010000


#define S_ISLNK(m)  (((m) & S_IFMT) == S_IFLNK)
#define S_ISREG(m)  (((m) & S_IFMT) == S_IFREG)
#define S_ISDIR(m)  (((m) & S_IFMT) == S_IFDIR)
#define S_ISCHR(m)  (((m) & S_IFMT) == S_IFCHR)
#define S_ISBLK(m)  (((m) & S_IFMT) == S_IFBLK)
#define S_ISFIFO(m) (((m) & S_IFMT) == S_IFIFO)
#define S_ISSOCK(m) (((m) & S_IFMT) == S_IFSOCK)
```

- 在具体业务逻辑实现时，步骤如下:
- - step 1 : 获取文件类型, 并转换成枚举
  - ```
    enum file_type get_file_type(const char *path)
    {
        int ret;
        struct stat stat_info;

        if (path == NULL)
            return FT_ERROR;

        ret = stat(path,&stat_info);
        if(ret == -1){
            perror("stat(): ");
            return FT_ERROR;
        }

        if(S_ISDIR(stat_info.st_mode))
            return FT_DIR;
        else if(S_ISREG(stat_info.st_mode))
            return FT_FILE;

        return FT_UNKNOWN;
    }
    ```
  - step 2 ： 将获取的文件类型存储自定义存储结构中
  - ```
    int cmd_cp_parse_type(cp_file_info_t *pfileinfo)
    {
    ```

```
    int ret;
    enum file_type ftype;
    if (pfileinfo == NULL)
        return -1;


    ftype = get_file_type(pfileinfo->src_path);

    if (ftype == FT_ERROR || ftype == FT_UNKNOWN)
        return -1;
    else
        pfileinfo->src_ftype  = ftype;

    return 0;
}
```

- step 3：在主逻辑函数 cmd_cp_execute 中进行调用 cmd_cp_parse_type

```
int cmd_cp_execute(cmd_t *pcmd)
{

#ifdef DEBUG
    print_command_info(pcmd);
#endif
    int ret;
    struct cp_file_info fileinfo;

    if (pcmd->cmd_arg_count != 2)
        return -1;

    ret = cmd_cp_parse_path(&fileinfo,pcmd);
    if (ret == -1)
        return -1;

    ret = cmd_cp_parse_type(&fileinfo);
    if (ret == -1)
        return -1;
     return 0;
}
```