## 实验十：文件系统编程

### 实验内容:

编写一个程序，实现将一个目录的所有内容复制到另一个目录的功能。要求：

源文件（目录）和目标文件（目录）的属主、权限等信息保持一致；

每复制一个文件（目录），在屏幕提示相应信息；

当遇到符号链接文件时，显示该文件为链接文件，不复制。

答：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  14  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157 | #include <stdio.h>  #include <string.h>  #include <sys/types.h>  #include <sys/stat.h>  #include <fcntl.h>  #include <unistd.h>  #include <dirent.h>  #define OK 0 // 成功  #define INPUT\_ERROR 1 // 输入参数有误  #define FILE\_EXIST 2 // 文件已存在  #define COPY\_ERROR 3 // 复制失败  #define NOT\_A\_DIRECTORY 2 // 源文件不是目录  #define BUFF\_SIZE 1024 // 复制文件的缓冲区大小  #define PATH\_LEN 100 // 路径占的字节数最大值  #define OWN\_CHANG\_ERROR 01 // 属主更改失败  #define MOD\_CHANG\_ERROR 010 // 权限更改失败  #define FOLDER\_CREAT\_ERROR 1 // 文件夹创建失败  char\* getFileName(char\* path);  int copyFile(char\* dst, char\* src);  int changeAttr(char\* filename, struct stat\* stat);  int copyFolder(char\* dst, char\* src);  int copy(char\* dst, char\* src);  int main(int argc, char \*argv[]) {  int fd; // open函数的返回值  char\* path[2]; // path[0]-srcfile, path[1]-dstfile  if (argc < 3) {  printf(" - Usage: %s srcfile dstfile\n", getFileName(argv[0]));  return INPUT\_ERROR;  }  path[0] = argv[1];  path[1] = argv[2];  // 检测源文件是否目录  fd = open(path[0], O\_DIRECTORY);  if (fd < 0) { // 源文件不是目录  printf("ERROR: Source file is not a directory!");  return NOT\_A\_DIRECTORY;  }  // 检测目标文件  if (access(path[1], F\_OK) != -1) { // 目标文件已存在  printf("ERROR: File already exist!");  return FILE\_EXIST;  }  copyFolder(path[1], path[0]); // 创建目录  copy(path[1], path[0]);  return OK;  }  // 复制完整目录(包含里面的文件)  int copy(char\* dst, char\* src) {  int fd1, fd2;  struct dirent\* ptr;  DIR\* dir = opendir(src);  char srcpath[PATH\_LEN], dstpath[PATH\_LEN];  while ((ptr = readdir(dir)) != NULL)  {  // 跳过.和..目录  if (strcmp(ptr->d\_name, ".") == 0 || strcmp(ptr->d\_name, "..") == 0)  continue;  // 生成完整路径  strcpy(srcpath, src);  strcpy(dstpath, dst);  if (\*(srcpath + strlen(srcpath) - 1) != '/')  strcat(srcpath, "/");  strcat(srcpath, ptr->d\_name);  if (\*(dstpath + strlen(dstpath) - 1) != '/')  strcat(dstpath, "/");  strcat(dstpath, ptr->d\_name);  // 判断文件类型  fd1 = open(srcpath, O\_DIRECTORY);  if (fd1 < 0) { // 源文件不是目录  fd2 = open(srcpath, O\_NOFOLLOW);  if (fd2 < 0) // 源文件是符号链接  printf("Message: File '%s' is a soft link!");  else // 源文件是普通文件  if (copyFile(dstpath, srcpath) != OK) {  printf("ERROR: File '%s' copy error!", srcpath);  return COPY\_ERROR;  }  close(fd2);  } else { // 源文件是目录  if (copyFolder(dstpath, srcpath) != OK) {  printf("ERROR: Directory '%s' copy error!", srcpath);  return COPY\_ERROR;  }  if (copy(dstpath, srcpath) != OK) { // 递归复制  return COPY\_ERROR;  }  }  close(fd1);  }  return OK;  }  // 复制文件夹(不包含里面的文件)  int copyFolder(char\* dst, char\* src) {  int fd;  struct stat srcstat;  fd = open(src, O\_NOFOLLOW | O\_DIRECTORY);  fstat(fd, &srcstat);  close(fd);  // 创建文件夹  if (mkdir(dst, srcstat.st\_mode) == -1) {  printf("> %s:\n", getFileName(src));  printf("\tERROR: Folder '%s' copy failed!\n", src);  return FOLDER\_CREAT\_ERROR;  }  // 设置文件夹权限  printf("> %s:\n", src);  printf("\tFolder '%s' copied succeeded!\n", src);  return changeAttr(dst, &srcstat);  }  // 复制文件  int copyFile(char \*dst, char \*src)  {  struct stat srcstat;  char buff[BUFF\_SIZE];  int len, fd1, fd2;  fd1 = open(src, O\_RDONLY);  fd2 = open(dst, O\_WRONLY | O\_CREAT);  fstat(fd1, &srcstat); // 记录源文件的属性  while ((len = read(fd1, buff, BUFF\_SIZE)))  write(fd2, buff, len);  close(fd1);  close(fd2);  printf("> %s:\n", src);  printf("\tFile '%s' copied succeeded!\n", src);  changeAttr(dst, &srcstat); // 修改复制后的文件属性  return OK;  }  // 更改属主、权限等属性，返回值为OK、MOD\_CHANG\_ERROR、OWN\_CHANG\_ERROR的组合  int changeAttr(char \*filename, struct stat\* stat) {  int own, mod, ret = OK;  own = chown(filename, stat->st\_uid, stat->st\_gid); // 更改文件属主  mod = chmod(filename, stat->st\_mode); // 更改文件权限  if (own == -1) { // 属主更改失败  printf("\tERROR: File '%s' owner failed to change!\n", filename);  ret |= OWN\_CHANG\_ERROR;  }  if (mod == -1) { // 权限更改失败  printf("\tERROR: File '%s' mode failed to change!\n", filename);  ret |= MOD\_CHANG\_ERROR;  }  if (mod == OK) { // 更改成功  printf("\tFile '%s' attribute changed succeeded!\n", filename);  }  return ret;  }  // 获取路径中的文件名  char\* getFileName(char\* path) {  char\* p = path + strlen(path) - 1;  if (p > path && \*p == '/')  \*p = '\0'; // 除根目录外，路径不以'/'结尾  while (p > path && \*(p - 1) != '/')  p--;  return p;  } |



