# linux系統編程之文件與IO（四）：目錄訪問相關係統調用



## 1. 目錄操作相關的系統調用

### 1.1 mkdir和rmdir系統調用

[code]
filename: mk_rm_dir.c
#include <sys/stat.h>
int mkdir(const char *path, mode_t mode);
return:
S 0
F -1
note:
mode權限至少要有執行權限。
[/code]
[code]
#include <unistd.h>
int rmdir(const char *pathname);
return:
S 0
F -1
note:
pathname目錄必須是空目錄。

#### 1.1.1 實例

```
#include <unistd.h>
#include <sys/stat.h>
#include <stdio.h>
#include <assert.h>
 #define MODE (S_IRUSR | S_IWUSR | S_IXUSR | S_IXGRP | S_IXOTH)
 int main( int argc, char * argv[])
{
    char     * pname;
    assert(argc == 2 );
    pname = argv[ 1 ];
    assert(mkdir(pname, MODE) == 0 );
    printf( " create %s successful!\n " , pname);
```

```
    assert(rmdir(pname) == 0 );
    printf( " rm %s\n " , pname);
     return  0 ;
}
```

測試： [qtlldr@qtldr editing]$ ./mk_rm_dir testdir create testdir successful! rm testdir [qtlldr@qtldr editing]$

## 1.2 chdir, getcwd系統調用

#include <unistd.h>
int chdir(const char *pathname);
return:
S 0
F -1 #include <unistd.h> char *getpwd(char *buf, size_t size); return: S buf F NULL buf是緩沖地址，size是buf的長度。該緩衝必須有足夠的長度以容納絕對路徑名加上一個null終止符。

### 1.2.1 實例

[code]
filename:ch_get_dir.c

```
#include <unistd.h>
#include <stdio.h>
#include < string .h>
#include <assert.h>
 #define BUFSIZE (50)
 int main( void )
{
    char buf[BUFSIZE];
    memset(( void *)buf, ' \0 ' , sizeof buf);
    assert(chdir( " /tmp " ) == 0 );
    printf( " chdir to /tmp successful\n " );
    assert(getcwd(buf, BUFSIZE) != NULL);
    printf( " now the directory is %s\n " , buf);
```

```
    return  0 ;
}
```

測試： [qtlldr@qtldr editing]$ ./ch_get_dir chdir to /tmp successful now the directory is /tmp [qtlldr@qtldr editing]$

## 1.3 opendir, closedir, readdir,

#include <sys/type.s>
#include <dirent.h>
DIR *opendir(const char *dirname);
return:
S DIR指針
F NULL
note:
DIR是一種目錄結構，類似FILE。 #include <sys/types.h> #include <dirent.h> struct dirent *readir(DIR *dirp); return: S一個指向保存目錄流下一個目錄項的dirent指針 F NULL note: struct dirent { char d_name[ NAME + 1]; /* \0結尾的文件名*/ } 到達目錄尾或出錯返回NULL，但是到達目錄尾不會設置errno，出錯則設置。 如果在readir的同時有其他進程在目錄中創建或者刪除文件愛你，readdir不保證能列處該目錄中所有文件。

#include <sys/types.h>
#include <dirent.h>
int closedir(DIR *dirp);
return:
S 0
F -1

### 1.3.1 實例:遞歸便利目錄

filename:help.txt幫助文檔
本程序只為學習linux目錄操作而寫

printdir
　輸出目錄文件或者統計目錄中的文件數目
語法：
　printdir [option] <files...>
選項：
　-l
　　輸出目錄下的文件名
　-c
　　統計目錄下的文件
　-dn
　　指定最大層次，最大為30
默認行為：
　如果沒有指定選項，那麼只輸出該目錄下的文件名
BUG:
　-l與-c選項不能同時使用，如果同時使用統計出錯。(以後會修正）　　本程序只為學習linux目錄操作而寫 filename:printdir.c

```c
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <dirent.h>
#include <stdio.h>
#include < string .h>
#include <stdlib.h>
 #define INDENT_DEPTH (4) /*列舉文件時的縮進數*/
 #define DEPTH_MAX (30) /*遞歸便利的最大層次*/
 #define HELPFILE ("help.txt ")
typedef int count_t;
 struct nfiletype {
    count_t ndir;
    count_t nreg;
    count_t nchr;
    count_t nfifo;
    count_t nsock;
    count_t nchar;
    count_t nblock;
    count_t nlink;
    count_t ntotol;
    count_t nunknow;
}; /* 記錄各個類型文件的數目*/
int     DEPTH = 20 ; /* 遞歸層級限制*/
int idepth_count = 1 ;
 int idepth_print = 1 ;

static  struct nfiletype *count_files( const  char * pathname,
             struct nfiletype * nfile);

static  void   printdir( const  char *pathname, int indent);
```

```
 int main( int argc, char ** argv)
{
    int opt;
     int     depth_opt;
     int     count_flag = 0 ;
     int     print_flag = 0 ;
     char *parg = NULL;
     struct nfiletype nfiles = { 0 };
     int     fd_help;
     char buf_help[BUFSIZ];
     int nread_help;
     char *filename_help = HELPFILE;
     while ( (opt = getopt(argc, argv, " lhd:c " )) != - 1 ) {
         switch (opt) {
             case  ' l ' :
                print_flag = 1 ;
                 break ;
             case  ' c ' :
                count_flag = 1 ;
                 break ;
             case  ' d ' :
                depth_opt = strtol(optarg, NULL, 10 );
                DEPTH = depth_opt <= DEPTH_MAX ? depth_opt: DEPTH;
                 break ;
             case  ' : ' :
                printf( " option needs a value\n " );
                 break ;
             case  ' ? ' :
                printf( " unknown option :%c\n " , optopt);
                 break ;
             case  ' h ' :
                fd_help = open(filename_help, O_RDONLY);
                 if (fd_help != - 1 ) {
                     while ((nread_help = read(fd_help, buf_help, BUFSIZ)) > 0 ) {
                        write( 1 , buf_help, nread_help);
                    }
                    close(fd_help);
                } else {
                    fprintf(stderr, " open %s failed!\n " , filename_help);
                }
                return  0 ;
         }
    }
    /* 如果沒有選項，那麼默認是打印目錄*/
    if (!print_flag && ! count_flag)
        print_flag = 1 ;
     for ( ; optind < argc; optind++ ) {
        parg = argv[optind];
         if (print_flag) {
             // printf("DEBUG-- printdir --%s\n", parg);
            printdir(parg, 4 );
         }
        if (count_flag) {
            memset(( void *)&nfiles, ' \0 ' , sizeof nfiles);
```

```
            // printf("DEBUG-- count_files--%s\n", parg);
            count_files(parg, & nfiles);
            printf( " In the %s there are :\n " , parg);
            printf( "      directory %d\n " , nfiles.ndir);
            printf( "      regular file %d\n " , nfiles.nreg);
            printf( "      specal character file %d\n " , nfiles.nchr);
            printf( "      special block file %d\n " , nfiles.nblock);
            printf( "      fifo file %d\n " , nfiles.nfifo);
            printf( "      sock file %d\n " , nfiles.nsock);
            printf( "      link file %d\n " , nfiles.nlink);
            printf( "      unknown file %d\n " , nfiles.nunknow);
            printf( " Total %d\n " , nfiles.ntotol);
        }
    }
    return  0 ;
}
/*
 *function: 對該目錄下的文件類型進行統計
 * input arg:
 * pathname:目錄名指針
 * nfile:記錄文件類型數目的結構體指針
 * return：
 * 記錄文件類型數目的結構體指針
 */

static  struct nfiletype *count_files( const  char * pathname,
             struct nfiletype * nfile)
{
    DIR * dp;
     struct dirent * entry;
     struct stat statbuf;
     // printf("DEBUG-- in count_files -- %s\n", pathname);
    /* 層次控制*/
    if (idepth_count > DEPTH)
         return NULL;
    idepth_count ++ ;
     if ((dp = opendir(pathname)) == NULL) {
        fprintf(stderr, " can not open %s\n " , pathname);
         return NULL;
    }
    chdir(pathname);
    while ((entry = readdir(dp)) != NULL) {
         /* 跳過.和.. */
        if (strcmp(entry->d_name, " . " ) == 0 || strcmp(entry->d_name, " .. " ) == 0 )
             continue ;
         /* 取得文件信息*/
        if (lstat(entry->d_name, &statbuf) == - 1 ) {
            fprintf(stderr, " can not test the %s's type\n " , entry-> d_name);
             return NULL;
        }
        /* 統計文件數目*/
        if (S_ISDIR(statbuf.st_mode)) { /* 是目錄就遞歸吧*/
            // printf("DEBUG -- directory %s\n", entry->d_name);
            count_files(entry -> d_name, nfile);
```

```
            nfile ->ndir++ ;
        }
        else  if (S_ISREG(statbuf.st_mode)) {
             // printf("DEBUG -- regular file %s\n", entry->d_name);
            nfile->nreg++ ;
        }
        else  if (S_ISCHR(statbuf.st_mode))
            nfile ->nchr++ ;
         else  if (S_ISBLK(statbuf.st_mode))
            nfile ->nblock++ ;
         else  if (S_ISLNK(statbuf.st_mode))
            nfile ->nlink++ ;
         else  if (S_ISFIFO(statbuf.st_mode))
            nfile ->nfifo++ ;
         else  if (S_ISSOCK(statbuf.st_mode))
            nfile ->nsock++ ;
         else nfile->nunknow++ ;
        nfile ->ntotol++ ;
    }
    chdir( " .. " );
    closedir(dp);
    return nfile;
 }
 /*
nblock; *function:列出目錄中的文件
nlink; *input arg:
ntotol; * pathname: 目錄名
 *return:
 * void
 */

static  void    printdir( const  char *pathname, int indent)
{
    DIR * dp;
     struct dirent * entry;
     struct stat statbuf;
     /* 層次控制*/
    if (idepth_print > DEPTH)
         return ;
    idepth_print ++ ;
     if ((dp = opendir(pathname)) == NULL) {
        fprintf(stderr, " can not open %s\n " , pathname);
         return ;
    }
    chdir(pathname);
    while ((entry = readdir(dp)) != NULL) {
         /* 跳過.和.. */
        if (strcmp(entry->d_name, " . " ) == 0 || strcmp(entry->d_name, " .. " ) == 0 )
             continue ;
         if (lstat(entry->d_name, &statbuf) == - 1 ) {
            fprintf(stderr, " can not test the %s's type\n " , entry-> d_name);
             return ;
        }
        if (S_ISDIR(statbuf.st_mode)) { /* 是目錄就遞歸吧*/
```

```
            printf( " %*s%s/\n " , indent, "  " , entry-> d_name);
            printdir(entry ->d_name, indent + INDENT_DEPTH);
        }
        else {
            printf( " %*s%s\n " , indent, "  " , entry-> d_name);
            }
    }
    chdir( " .. " );
    closedir(dp);
}
```