Linux 下的目录是依照标准来实作的，因此，您可以毫无问题地移殖到任何其它 UNIX 平台。

---

getcwd/getwd： 取得目前所在目录

---

#include
char * getcwd(char *buf,size_t size);
buf 将会返回目前路径名称。

任何的错误发生，将会返回 NULL。如果路径长度超过 size，errno 为 ERANGE。getcwd 返回的值永远是没有 symbol link 的。

---

#include
char *getwd(char *buf);
getwd 是个危险的函数，一般都会强烈建议不要用，因为您无法确定最长的目录长度为多少。PATH_MAX 定义了最长的路径长度。在 Linux 下所以提供这个函数主要是因为「传统」。

---

//获取系统目录最大长度
long pathconf(char* path, int flag);
---

chdir/fchdir/chroot： 改变目前所在目录

---

#include
int chdir(const char * pathname);
int fchdir(int fd);
chdir 根据 pathname 变更目前的所在目录，它只改变该程式的所在目录。
fchdir 根据已开启的 fd(file descriptor)目录来变更。

```
//sample
    /*更改当前工作目录到上级目录*/
   if(chdir("..")==-1){
        perror("Couldn't change current working directory.\n");
        return 1;
    }
```

---

```
#include
int chroot(const char * path);
```

chroot 改变该程式的根目录所在。例如 chroot("/home/ftp")会将根目录换到/home/ftp 下，而所有档案操作都不会超出这个围内。为保障安全性，当 chdir("/..")时，将会仅切换到 chdir("/")，如此便不会有档案安全问题。

---

mkdir/rmdir ： 造/移除目录

---

```
#include <sys/stat.h>
#include <sys/types.h>
int mkdir(const char * dirname,mode_t mode);
```

mkdir 会造一个新目录出来，例如 mkdir("/home/foxman",0755);。
如果该目录或档案已经存在，则操作失败。

```
/*mode 设置为 0700，开始的 0 表示八进制*/
if(mkdir("/home/zxc/z", 0700) == -1){
    perror("Couldn't create the directory.\n");
    return 1;
}
```

---

```
#include <unistd.h>
int rmdir(char * pathname);
```

这个函数移除 pathname 目录。

---

//获得文件信息

```
#include <sys/types.h> <sys/stat.h> <unistd.h>
int stat(const char* path, struct stat* buf);
int fstat(int filedes, struct stat* buf);
int lstat(const char* path, struct stat* buf);
```

opendir/readdir/closedir/rewinddir ： 读取目录资讯

---

```
#include
DIR * opendir(const char * pathname);
int closedir(DIR *dir);
struct dirent * readdir(DIR *dir);
int rewinddir(DIR *dir);
struct dirent {
    long d_ino;              /* inode number */
    off_t d_off;             /* offset to this dirent */
    unsigned short d_reclen;     /* length of this d_name */
    char d_name [NAME_MAX+1];    /* file name (null-terminated) */
};
```

opendir 开启一个目录操作 DIR，closedir 关闭之。

readdir 则循序读取目录中的资讯，rewinddir 则可重新读取目录资讯。

以下是个标准例。

---

```
#include <sys/types.h>
#include <direct.h>
char ** dirGetInfo(const char *pathname)
{
  char ** filenames;
  DIR * dir;
  struct dirent * ent;
  int n = 0;
```

```
  filenames = (char **)malloc(sizeof(char*));
  filenames[0]=NULL;

  dir = opendir(pathname);
  if (!dir) return filenames;

  while ((ent = readdir(dir))) {
    filenames = (char**)realloc(filenames,sizeof(char*)*(n+1));
    filenames[n] = strdup(ent->d_name);
    n++;
  }

  closedir(dir);

  filenames = (char **)realloc(filenames,sizeof(char*)*(n+1));
  filenames[n] = NULL;

  return filenames;
}
```

c 语言实现目录遍历

思路：

采用深度遍历算法，进行目录遍历.

```
#include <direct.h> //use _chdir() and _getcwd() function
#include <stdio.h>
#include <io.h> //use _findfirst() and findnext() function
#define _MAXPATH 256
void view();
int main()
{
  char filename[128];
  printf("input path\n");
   gets(filename);
  _chdir(filename); //进入目录
  view();
  return 0;
}
void view()
{
  struct _finddata_t file; //定义结构体变量
```

```
long handle;
char path[_MAXPATH];   //路径
handle=_findfirst("*",&file);//查找所有文件
if(handle==-1)/*如果 handle 为－1，表示当前目录为空，则结束查找而返回
              如果 handle 为－1，表示当前目录为空，则结束查找而返回 */
    return ;
else
{
  if(file.attrib &_A_SUBDIR ) //是目录
  {
      if(file.name[0]!='.') //文件名不是'.'或'..'时
           {
        _chdir(file.name); //进入该目录
        _getcwd(path,_MAXPATH) //获得目录路径
              puts(path); //输出目录路径
         view();//继续遍历
         _chdir("..");/*查查找完毕之后，返回上一级目录找完毕之后，返回
上一级目录*/

      }
   }
  else    //// 如果第一个实体不是目录,显示该文件
  {
   _getcwd(path,_MAXPATH);
     // 再获得文件的完整的路径名(包含文件的名称)
     strcat(path,"\\");
     strcat(path,file.name);
     printf("%-20s  9ld\n",path,file.size);
   }
   // 继续对当前目录中的下一个子目录或文件进行与上面同样的查找
   while(!(_findnext(handle,&file))
   {
       if(file.attrib &_A_SUBDIR) //是目录
  {
      if(file.name[0]!='.') //文件名不是'.'或'..'时
           {
        _chdir(file.name); //进入该目录
        _getcwd(path,_MAXPATH) //获得目录路径
              puts(path); //输出目录路径
         view();//继续遍历
         _chdir("..");/*查查找完毕之后，返回上一级目录找完毕之后，返回
上一级目录*/

      }
```

```
    }
   else    //// 如果第一个实体不是目录,显示该文件
   {
    _getcwd(path,_MAXPATH);
      // 再获得文件的完整的路径名(包含文件的名称)
      strcat(path,"\\");
      strcat(path,file.name);
      printf("%-20s  9ld\n",path,file.size);
    }
     }
  _findclose(handle);
 }
```

这样我们就可以对整个目录进行遍历搜索，并输出显示其完整的文件路径。以上的程序在 Visual C＋＋ 6.0 中已调试通过。

```
#include <dirent.h>
#include <stdio.h>

int main()
{
DIR *dir;
struct dirent *subfile;

dir = opendir(".");
while ((subfile = readdir(dir)) != NULL) {
printf("%s\n", subfile->d_name);
}
return 0;
}
```

UNIX: lstat,readdir,opendir
win32: findfirst,findnext

这个头文件中的函数只不过提供了简单的文件目录服务而已
使用文件目录服务和学操作系统编程是两码事
你要使用文件目录服务根本不需要从新编写文件系统, 也不必要换操作系统
因为无论哪个操作系统都会提供文件目录编程接口

windows 下需要文件目录服务时一般是使用 Win API ( 当然,也可以使用 文件系统 API 的某个封装 ). 简单的文件目录服务可以用 C 运行时库函数
windows 下和 文件目录服务 有关的 C 运行时库函数 的声明 一般 在 direct.h 和 io.h 中.比如 linux 平台下的 opendir 就可以用 dev c++ 的头文件 io.h 中的函数 chdir 与 direct.h 中的 _chdrive 替代.

如果想使用 Win API 来访问文件目录服务, 你应该先去找本 Windows 编程基础的书来看.