实验四 Linux文件目录

一、实验目的

1、了解Linux文件系统与目录操作；

2、了解Linux文件系统目录结构；

3、掌握文件和目录的程序设计方法。

二、实验内容

编程实现目录查询功能：

- 功能类似ls -lR；
- 查询指定目录下的文件及子目录信息；

显示文件的类型、大小、时间等信息；

- 递归显示子目录中的所有文件信息。

/* 也可非递归实现 */

三、预备知识

1、Linux文件属性接口

```
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
int fstat(int fildes,struct stat *buf);
```

返回文件描述符相关的文件的状态信息

```
int stat(const char *path, struct stat *buf);
```

通过文件名获取文件信息，并保存在buf所指的结构体stat中

```
int lstat(const char *path, struct stat *buf);
```

如读取到了符号连接，lstat读取符号连接本身的状态信息，而stat读取的是符号连接指向文件的信息。

```
struct stat {
    unsigned long  st_dev;    // 文件所属的设备
    unsigned long  st_ino;// 文件相关的inode
    unsigned short st_mode;  // 文件的权限信息和类型信息:
                S_IFDIR, S_IFBLK, S_IFIFO, S_IFLINK
    unsigned short st_nlink;  //硬连接的数目
    unsigned short st_uid; // 文件所有者的ID
    unsigned short st_gid; //文件所有者的组ID
    unsigned long  st_rdev;   //设备类型
    unsigned long  st_size;    //文件大小
    unsigned long  st_blksize;//块大小
    unsigned long  st_blocks; //块数
    unsigned long  st_atime;  // 文件最后访问时间
    unsigned long  st_atime_nsec;
    unsigned long  st_mtime; // 最后修改内容的时间
    unsigned long  st_mtime_nsec;
    unsigned long  st_ctime;  // 文件最后修改属性的时间
    unsigned long  st_ctime_nsec;
    unsigned long  __unused4;
    unsigned long  __unused5;
};
```

stat结构体几乎保存了所有的文件状态信息

stat结构体几乎保存了所有的文件状态信息：

- st_mode 文件的权限信息和类型信息

S_ISDIR // 是不是目录

S_IFBLK // 文件是不是块设备

S_IFIFO // 文件是不是FIFO(命名管道)

S_IFLNK // 文件是不是符号连接

- st_ino 文件相关的inode
- st_dev 文件所属的设备
- st_uid 文件所有者的ID
- st_gid 文件所有者的组ID
- st_atime 文件最后访问时间
- st_ctime 文件最后修改时间(修改权限，用户，组或者内容)
- st_mtime 最后修改内容的时间
- st_nlink 硬连接的数目

st_mode成员定义了一些操作st_mode的宏：

2、Linux目录结构接口

```
#include <sys/types.h>
#include <dirent.h>
#include <unistd.h>
```

- opendir()

  DIR *opendir(const char *name);

  通过路径打开一个目录，返回一个DIR结构体指针（目录流），失败返回NULL；

- readdir()

  struct dirent *readdir(DIR *)

  读取目录中的下一个目录项，没有目录项可以读取时，返回为NULL；

目录项结构：

```
struct dirent {
        #ifndef __USE_FILE_OFFSET64
    __ino_t  d_ino; //索引节点号
    __off_t  d_off; //在目录文件中的偏移
         #else
             __ino64_t  d_ino;
            __off64_t   d_off;
        #endif
    unsigned  short  int  d_reclent;    //文件名的长度
    unsigned  char  d_type;    //d_name所指的文件类型
       char  d_name[256];    //文件名
 };
```

注：需跳过两个目录项“.”和“..”

定义见/usr/include/dirent.h

· chdir()

int chdir(const char *path);

改变目录,与用户通过cd命令改变目录一样，程序也可以通过chdir来改变目录，这样使得 fopen(),opendir(),这里需要路径的系统调用，可以使用相对于当前目录的相对路径打开文件(目录)。

· closedir()

int closedir(DIR*)

关闭目录流

四、程序结构

```
#include <unistd.h>
#include <sys/stat.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <dirent.h>

void printdir(char *dir, int depth){
    DIR *dp;
    struct dirent *entry;
    struct stat statbuf;
    if ((dp = 打开dir目录) 不成功){ //opendir
        打印出错信息;
        返回;
    }
    将dir设置为当前目录; //chdir
```

```
while(读到一个目录项){
        以该目录项的名字为参数,调用lstat得到该目录项的相关信息;
        if(是目录){
            if(目录项的名字是”..”或”.”)
                跳过该目录项;
            打印目录项的深度、目录名等信息//printf(%*depth)
            递归调用printdir,打印子目录的信息,其中的depth+4;
        }
    else   打印文件的深度、文件名等信息
    }
    返回父目录; //chdir
    关闭目录项; //closedir
}

int main(…){
    ……
}
```

输入ls -l 可以看到如下信息：

```
drwxr-xr-x 3 killercat killercat 4096 2007-01-11 16:27 Desktop
drwx------  8 killercat killercat 4096 2007-01-09 14:33 Documents
drwxr-xr-x 2 killercat killercat 4096 2006-11-30 19:27 Downloads
drwx------  4 killercat killercat 4096 2006-12-16 20:20 References
drwx------  9 killercat killercat 4096 2007-01-11 13:34 Software
drwxr-xr-x 3 killercat killercat 4096 2006-12-11 16:39 vmware
drwx------  6 killercat killercat 4096 2007-01-11 13:34 Workspace
```

输入ls -lR /可以看到如下信息：

```
.:
总用量 92
drwxr-xr-x   2 root root  4096  8月 24  2016 bin
drwxr-xr-x   3 root root  4096  2月 11  2017 boot
drwxrwxr-x   2 root root  4096  8月 24  2016 cdrom
drwxr-xr-x  16 root root  4280 12月 15 22:08 dev
drwxr-xr-x 128 root root 12288 12月 15 22:10 etc
drwxr-xr-x   3 root root  4096  8月 24  2016 home
lrwxrwxrwx   1 root root    32  8月 24  2016 initrd.img -> boot/initrd.img-4.2.0-27-generic
drwxr-xr-x  23 root root  4096  8月 24  2016 lib
drwx------   2 root root 16384  8月 24  2016 lost+found
drwxr-xr-x   4 root root  4096  8月 27  2016 media
drwxr-xr-x   2 root root  4096  4月 11  2014 mnt
drwxr-xr-x   3 root root  4096  2月 11  2017 opt
dr-xr-xr-x 174 root root     0 12月 15 22:08 proc
drwx------   3 root root  4096  3月  1  2018 root
drwxr-xr-x  23 root root   740 12月 15 22:10 run
drwxr-xr-x   2 root root 12288  2月 11  2017 sbin
drwxr-xr-x   2 root root  4096  2月 18  2016 srv
dr-xr-xr-x  13 root root     0 12月 15 22:08 sys
drwxrwxrwt   4 root root  4096 12月 15 22:11 tmp
drwxr-xr-x  10 root root  4096  2月 18  2016 usr
drwxr-xr-x  13 root root  4096  2月 18  2016 var
lrwxrwxrwx   1 root root    29  8月 24  2016 vmlinuz -> boot/vmlinuz-4.2.0-27-generic

./bin:
总用量 9472
-rwxr-xr-x 1 root root  986672 10月  8  2014 bash
-rwxr-xr-x 1 root root   30240 10月 21  2013 bunzip2
-rwxr-xr-x 1 root root 1713424 11月 15  2013 busybox
-rwxr-xr-x 1 root root   30240 10月 21  2013 bzcat
lrwxrwxrwx 1 root root       6  8月 24  2016 bzcmp -> bzdiff
-rwxr-xr-x 1 root root    2140 10月 21  2013 bzdiff
lrwxrwxrwx 1 root root       6  8月 24  2016 bzegrep -> bzgrep
-rwxr-xr-x 1 root root    4877 10月 21  2013 bzexe
lrwxrwxrwx 1 root root       6  8月 24  2016 bzfgrep -> bzgrep
-rwxr-xr-x 1 root root    3642 10月 21  2013 bzgrep
-rwxr-xr-x 1 root root   30240 10月 21  2013 bzip2
-rwxr-xr-x 1 root root    9624 10月 21  2013 bzip2recover
```