实验二 目录树的遍历

姓名：刘梦杰 学号：22920212204168 完成时间：2023.10.29

一、实验内容

编写程序 myfind

命令语法:

myfind <pathname> [-comp <filename> | -name <str>…]

命令语义:

（1）myfind <pathname> 的功能：

除了具有与程序 4-7 相同的功能外，还要输出在<pathname>目录子树之下，文件长度不大于 4096 字节的常规文件，在所有允许访问的普通文件中所占的百分比。程序不允许打印出任何路径名。

（2）myfind <pathname> -comp <filename>的功能：

<filename>是常规文件的路径名（非目录名，但是其路径可以包含目录）。命令仅仅输出在<pathname>目录子树之下，所有与<filename>文件内容一致的文件的绝对路径名。不允许输出任何其它的路径名，包括不可访问的路径名。

（3）myfind <pathname> -name <str>…的功能：

<str>…是一个以空格分隔的文件名序列(不带路径)。命令输出在<pathname>目录子树之下，所有与<str>…序列中文件名相同的文件的绝对路径名。不允许输出不可访问的或无关的路径名。

<pathname>和<filename>均既可以是绝对路径名，也可以是相对路径名。<pathname>既可以是目录，也可以是文件，此时，目录为当前工作目录。

二、 实验原理及设计思路

1. 实验原理

实验基于教材图 4-22 的程序 ftw8.c，该程序通过调用文件系统相关函数及宏，如 opendir、lstat、S_ISDIR 等，来实现对文件系统遍历，借助基本系统数据类型 stat 等来获取文件信息。

本次实验才此框架基础上进行编写扩展。

2. 设计思路

首先说明教材图 4-22 的程序 ftw8.c:

main 函数：读入命令行参数，调用 myftw 函数。

myftw 函数：为目录名的存储分配空间，调用 dopath 函数。

dopath 函数：深度优先遍历目录，并对每个文件调用 myfunc 函数。

myfunc 函数：使用 lstat 统计各类文件信息，并进行计数。

可见，目录遍历功能已经实现，无需改动。所以我们只需按照要求编写不同的 myfunc 函数来实现所需功能即可。

```
typedef int Myfunc(const char *, const struct stat *, int);
```

代码中已经定义了 Myfunc 型数据，根据要实现的功能，我们声明如下三个函数

```
static Myfunc myfuncCount, myfuncContentCompare,
myfuncNameCompare;
```

整体思路就是在 main 函数中根据参数判断所要使用的功能，然后调用相应的函数即可。

3. **具体实现**

'//'的处理:

在 dopath 函数中加上 if 判断，如果 fullpath 已经以'/'结尾，则不执行操作。

```
    if (fullpath[n - 1] != '/')
    {
        fullpath[n++] = '/';
        fullpath[n] = 0;
    }
```

(1) myfind <pathname> 的功能:

在原来计数的基础上加上判断文件大小的 if 语句。

```
static int
myfuncCount(const char *pathname, const struct stat *statptr,
int type)
{
    switch (type)
    {
    case FTW_F:
        switch (statptr->st_mode & S_IFMT)
        {
        case S_IFREG:
            nreg++;
            if (statptr->st_size <= 4096)
                nsize4096++;
            break;
```

(2) myfind <pathname> -comp <filename>的功能:

预处理:

打开<filename>，获取文件大小，用于筛选掉大小不同的文件，提高程序效率。

```
    int fd;
    if ((fd = open(argv[3], O_RDONLY, FILE_MODE)) == -1){
            err_sys("open error");
    }
    filesize = buf.st_size;
```

获取绝对路径:

```
static char *
getAbsolutePath(const char *pathname, char *absolutepath)
{
    static size_t length;
```

```
    char *currentpath; //current workd dir
    currentpath = path_alloc(&length);
    if (getcwd(currentpath, length) == NULL){
        err_sys("getcwd error");
    }
    if (chdir(pathname) < 0){
        err_sys("chdir error");
    }
    if (getcwd(absolutepath, length) == NULL){
        err_sys("getcwd error");
    }
    if (chdir(currentpath) < 0){
        err_sys("chdir error");
    }
    return absolutepath;
}
```

myfuncContentCompare 函数具体实现：

首先筛选文件，只有是普通文件且大小和待比较文件相同的文件才继续执行。

然后将当前遍历的文件读入缓冲区，并使用 strcmp 函数进行比较即可。

```
static int
myfuncContentCompare(const char *pathname, const struct stat
*statptr, int type)
{
    if (type == FTW_F && S_ISREG(statptr->st_mode) &&
statptr->st_size == filesize){
        int fd;
        if ((fd = open(pathname, O_RDONLY,FILE_MODE)) == -1){
            return 0;
        }
        if (read(fd, comparedfilebuffsize, statptr->st_size) !=
statptr->st_size){
            err_sys("read error");
        }
        close(fd);
        if (strcmp(filebuffsize, comparedfilebuffsize) == 0){
            printf("%s\n", pathname);
```

```
            filecnt++;
        }
    }
    return 0;
}
```

（3）myfind <pathname> -name <str>…的功能：

首先来看 main 函数

由于 Myfunc 函数中需要将当前遍历到的文件与命令行输入的文件逐一比较，所以定义了两个全局变量 global_argc 和 global_argv，将 argc 和 argv 扩展到全局。

同样地，要输入绝对路径名，所以这里也将输入的<pathname>转为绝对路径。

```
if (argc >= 4 && strcmp(argv[2], "-name") == 0){
        global_argc = argc;
        global_argv = argv;
        static size_t len;
        absolutepath = path_alloc(&len);
        absolutepath = getAbsolutePath(argv[1], absolutepath);
        printf("Matching files:\n");
        ret = myftw(absolutepath, myfuncNameCompare);
        if (filecnt == 0){
            printf("No matching files found\n");
        }
    }
```

myfuncNameCompare 函数具体实现

由于传入的参数 pathname 为绝对路径，需要先将文件名从中分离出来，之后与 argv 中的文件名逐一比较即可。

```
static int
myfuncNameCompare(const char *pathname, const struct stat
*statptr, int type)
{
    if (type == FTW_F){
        int pos = 0;
        for (int i = strlen(pathname) - 1; i >= 0; i--){
            if (pathname[i] == '/'){
                pos = i;
                break;
            }
        }
```

```
        pos++;
        const char *curfilename = pathname + pos;
        for (int i = 3; i < global_argc; i++){
            if (strcmp(curfilename, global_argv[i]) == 0){
                printf("%s\n", pathname);
                filecnt++;
            }
        }
    }
    return 0;
}
```

三、 实验结果

源程序为 myfind.c
可执行程序为 myfind
程序编译:

需要使用到 apue 源码 make 生成的 libapue.a 静态链接库。

```
[cs214168@mcore 2]$ gcc myfind.c -o myfind libapue.a
[cs214168@mcore 2]$
```

功能一：

```
[cs214168@mcore 2]$ ./myfind ~
regular files  =     394, 60.99 %,
whose length is not greater than 4096 =     329, 83.50 %
directories    =      44,  6.81 %
block special  =       0,  0.00 %
char special   =       0,  0.00 %
FIFOs          =       0,  0.00 %
symbolic links =     208, 32.20 %
sockets        =       0,  0.00 %
[cs214168@mcore 2]$
```

测试文件说明:

在/home/cs21/cs214168/homework/2 下创建 test0.txt，写入 123。
在/home/cs21/cs214168/homework/2 下创建 test1.txt，写入 456。
在/home/cs21/cs214168 下创建 test0.txt，写入 456。
在/home/cs21/cs214168 下创建 test1.txt，写入 123。
在/home/cs21/cs214168 下创建 test2.txt，写入 456。
在/home/cs21/cs214168/temp 下创建 test0.txt，写入 123。
在/home/cs21/cs214168/temp 下创建 test1.txt，写入 789。
在/home/cs21/cs214168/temp 下创建 test2.txt，写入 123。

功能二：

执行命令，结果符合预期：

```
[cs214168@mcore 2]$ ./myfind ~ -comp test0.txt
The files same as test0.txt are:
/home/cs21/cs214168/homework/2/test0.txt
/home/cs21/cs214168/test1.txt
/home/cs21/cs214168/temp/test2.txt
/home/cs21/cs214168/temp/test0.txt
[cs214168@mcore 2]$
```

```
[cs214168@mcore 2]$ ./myfind ~ -comp test1.txt
The files same as test1.txt are:
/home/cs21/cs214168/homework/2/test1.txt
/home/cs21/cs214168/test2.txt
/home/cs21/cs214168/test0.txt
[cs214168@mcore 2]$
```

功能三：

```
[cs214168@mcore 2]$ ./myfind ~ -name test0.txt test1.txt
Matching files:
/home/cs21/cs214168/homework/2/test1.txt
/home/cs21/cs214168/homework/2/test0.txt
/home/cs21/cs214168/test1.txt
/home/cs21/cs214168/temp/test1.txt
/home/cs21/cs214168/temp/test0.txt
/home/cs21/cs214168/test0.txt
[cs214168@mcore 2]$
```

四、体会和建议

初次接触到这个实验有点不知所措，不知从何下手，在搞懂教材源程序后难度就下来了，所以关键在于理解原框架，也就是 dopath 函数会对每个文件（包括目录，目录也是文件）都执行一次 myfunc 函数，明白之后我们要做的就是根据要求编写三个 function 函数，实现相应功能，这样子思路就清晰了。