



Unix 实验报告

实验：实验 4 目录树的遍历
专业：计算机科学与技术
班级：1 班
姓名：姚怀聿
学号：22920202204632

2022 年 11 月 18 日

目 录

一、	实验内容描述	2
二、	设计、实验构思	2
	构思:	2
三、	实验结果	3
四、	体会和建议.....	4
五、	完成人姓名及完成时间.....	4

一、 实验内容描述

本实验的目的是掌握与文件和目录树有关的系统调用和库函数。。

实验要求编写程序 `myfind`;

命令语法为:

```
./myfind <pathname> [-comp <filename> | -name <str>...]
```

具体要实现的功能如下:

`myfind <pathname>` 的功能:

输出在<pathname>目录子树下, 文件长度大于 4096 字节的所有文件的数目。程序不允许打印出任何路径名或者其他任何无关文字。

程序在实现时的要求如下:

遍历目录树时, 访问结点(目录项)的具体操作应当由遍历函数 `dopath` 携带的函数指针参数决定。

二、 设计、实验构思

构思:

需要在程序 4-22 的基础上增加一个统计<pathname>目录子树下文件长度大于 4096 字节的所有文件的数目, 具体实现时, 我在全局增加了一个 `int` 类型的变量 `nlrg`, 用于记录遍历到的文件长度大于 4096 字节的常规文件的个数。然后在

main 函数中用该个数除以所有允许访问的常规文件的总数量，得到功能要求的百分比，最后在 main 函数中输出。程序具体实现如下：

```

34
35 typedef int Myfunc(const char*, const struct stat*, int);
36 static long nreg, ndir, nblk, nchr, nfifo, nsock, ntot, nlrg; // 分别表示常规文件、目录文件...
37
38 static int path_noloop(const char* pathname) {
39     char* pos;
40
41     if (stat(pathname, &st) == 0) {
42         if (S_ISREG(st.st_mode)) nreg++;
43         if (S_ISDIR(st.st_mode)) ndir++;
44         if (S_ISBLK(st.st_mode)) nblk++;
45         if (S_ISCHR(st.st_mode)) nchr++;
46         if (S_ISFIFO(st.st_mode)) nfifo++;
47         if (S_ISLNK(st.st_mode)) nsock++;
48         if (S_ISOCK(st.st_mode)) nsock++;
49         if (S_ISDIR(st.st_mode) & S_IFMT) {
50             if (st.st_size > 4096) nlrg++;
51         }
52     }
53     return 0;
54 }
55
56 static int myfunc1(const char* pathname, const struct stat* statptr, int type) {
57     switch(type) {
58     case FIND_F: // 文件
59         switch(statptr->st_mode & S_IFMT) {
60             case S_IFREG: nreg++; break;
61             case S_IFBLK: nblk++; break;
62             case S_IFCHR: nchr++; break;
63             case S_IFIFO: nfifo++; break;
64             case S_IFLNK: nsock++; break;
65             case S_IFSOCK: nsock++; break;
66             case S_IFDIR: fprintf(stderr, "for S_IFDIR for %s", pathname);
67                 if (statptr->st_size > 4096) nlrg++;
68                 break;
69             default:
70                 fprintf(stderr, "unknown type %d for pathname %s\n", type, pathname);
71         }
72     }
73     return 0;
74 }
75
76 int main(int argc, char** argv) {
77     int ret;
78     if (argc < 2 || argc == 3) { // 参数数量不符 报错
79         fprintf(stderr, "Usage: ./myfind <pathname> [-comp filename] [-name str]\n");
80         exit(1);
81     }
82     if (argc == 2) {
83         ret = myfunc(argv[1], myfunc1, TYPE_P);
84         ntot = nreg + ndir + nblk + nchr + nfifo + nsock;
85         if (ntot == 0) ntot = 1; // 避免除数为0 所有文件所占的百分比都为0
86         //printf(stdout, "regular files = %ld %2f%%\n", nreg, nreg * 100.0 / ntot);
87         //printf(stdout, "directories = %ld %2f%%\n", ndir, ndir * 100.0 / ntot);
88         //printf(stdout, "block special = %ld %2f%%\n", nblk, nblk * 100.0 / ntot);
89         //printf(stdout, "char special = %ld %2f%%\n", nchr, nchr * 100.0 / ntot);
90         //printf(stdout, "FIFOs = %ld %2f%%\n", nfifo, nfifo * 100.0 / ntot);
91         //printf(stdout, "symbolic links = %ld %2f%%\n", nsock, nsock * 100.0 / ntot);
92         //printf(stdout, "sockets = %ld %2f%%\n", nsock, nsock * 100.0 / ntot);
93         printf(stdout, "There are %ld files with a length greater than 4096 bytes and the ratio is %2f%%\n", nlrg, nlrg * 100.0 / ntot);
94     } else {
95         if (strcmp(argv[2], "-comp") == 0) {
96             if (argc > 4) {
97                 fprintf(stderr, "Usage: ./myfind <pathname> [-comp filename]\n");
98                 exit(1);
99             }
100             struct stat statres;
101             if (stat(argv[1], &statres) < 0) {
102                 fprintf(stderr, "stat %s failed\n", argv[1]);
103                 exit(1);
104             }
105             if (S_ISREG(statres.st_mode)) nreg++;
106             if (S_ISDIR(statres.st_mode)) ndir++;
107             if (S_ISBLK(statres.st_mode)) nblk++;
108             if (S_ISCHR(statres.st_mode)) nchr++;
109             if (S_ISFIFO(statres.st_mode)) nfifo++;
110             if (S_ISLNK(statres.st_mode)) nsock++;
111             if (S_ISOCK(statres.st_mode)) nsock++;
112             if (S_ISDIR(statres.st_mode) & S_IFMT) {
113                 if (statres.st_size > 4096) nlrg++;
114             }
115         }
116     }
117     ntot = nreg + ndir + nblk + nchr + nfifo + nsock;
118     if (ntot == 0) ntot = 1;
119     printf(stdout, "regular files = %ld %2f%%\n", nreg, nreg * 100.0 / ntot);
120     printf(stdout, "directories = %ld %2f%%\n", ndir, ndir * 100.0 / ntot);
121     printf(stdout, "block special = %ld %2f%%\n", nblk, nblk * 100.0 / ntot);
122     printf(stdout, "char special = %ld %2f%%\n", nchr, nchr * 100.0 / ntot);
123     printf(stdout, "FIFOs = %ld %2f%%\n", nfifo, nfifo * 100.0 / ntot);
124     printf(stdout, "symbolic links = %ld %2f%%\n", nsock, nsock * 100.0 / ntot);
125     printf(stdout, "sockets = %ld %2f%%\n", nsock, nsock * 100.0 / ntot);
126     printf(stdout, "There are %ld files with a length greater than 4096 bytes and the ratio is %2f%%\n", nlrg, nlrg * 100.0 / ntot);
127     return 0;
128 }

```

记录长度大于4096字节的文件个数

如果该文件的文件长度大于4096字节，就将nlrg++

在main函数中打印

三、实验结果

源程序名	可执行程序名
myfind.c	myfind

编译生成可执行文件：

使用如下指令：

`make myfind`，即可得到可执行文件`myfind`

```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/unix/homework/homework_4$ ls
myfind.c
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/unix/homework/homework_4$ make myfind
cc      myfind.c      -o myfind
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/unix/homework/homework_4$ ls
myfind  myfind.c
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/unix/homework/homework_4$
```

运行程序：

```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/unix/homework/homework_4$ ./myfind /home/rongrong/
There are 11292 files with a length greater than 4096 bytes and the ratio is 59.38%.
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/unix/homework/homework_4$
```

经检验，结果正确。

四、 体会和建议

体会：本次实验基本是第二次实验的重复，且要求更简单了，只修改了一个大于小于号和 main 函数中的输出。

建议：希望以后上机课之前，老师能提前说明实验课需要做什么，比如要用到哪些知识点，以及书上哪些程序对本次实验是比较有帮助的，我们可以通过看书本上的例程去更好的实现实验要求。以及可能要用到的不熟悉的函数能为我们做详细的解释。

五、 完成人姓名及完成时间

完成人姓名	完成时间
姚怀聿	2022 年 11 月 18 日