

# 编程作业：遍历文件目录

## 一、实验目的

- ◆ 编程实现程序 `list.c`，列表普通磁盘文件，包括文件名和文件大小。
- ◆ 使用 `vi` 编辑文件，熟悉工具 `vi`。
- ◆ 使用 Linux 的系统调用和库函数。
- ◆ 体会 Shell 文件通配符的处理方式以及命令对选项的处理方式。

## 二、实验要求

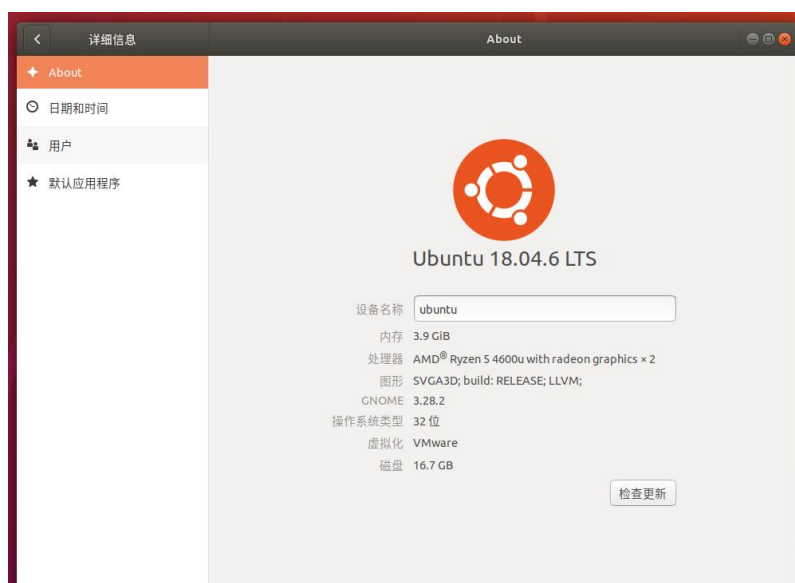
1. 与 `ls` 命令类似，处理对象可以有 0 到多个

- ◆ 0 个：列出当前目录下所有文件
- ◆ 普通文件：列出文件
- ◆ 目录：列出目录下所有文件

2. 实现自定义选项 `r,a,l,h,m` 以及 `--`

- ◆ `r` 递归方式列出子目录（每项要含路径，类似 `find` 的 `-print` 输出风格，需要设计递归程序）
- ◆ `a` 列出文件名第一个字符为圆点的普通文件（默认情况下不列出文件名首字符为圆点的文件）
- ◆ `l` 后跟一整数，限定文件大小的最小值（字节）
- ◆ `h` 后跟一整数，限定文件大小的最大值（字节）
- ◆ `m` 后跟一整数 `n`，限定文件的最近修改时间必须在 `n` 天内
- ◆ `--` 显式地终止命令选项分析

## 三、运行环境



## 四、实验步骤

1. 编写 list.c 源程序，其中主要函数如下：

① 初始化 Stu 结构体，返回初始化后的结构体变量 stu

```
struct Stu initStu()
```

② 根据用户输入，从数组 argv[] 中读出字符串，给 stu 赋值

```
void get_parameter(int argc, char *argv[])
```

③ 打印 path 路径的文件信息，根据 stu 的内容，进行筛选和处理后，打印文件信息。格式为“文件大小 文件路径/文件名”

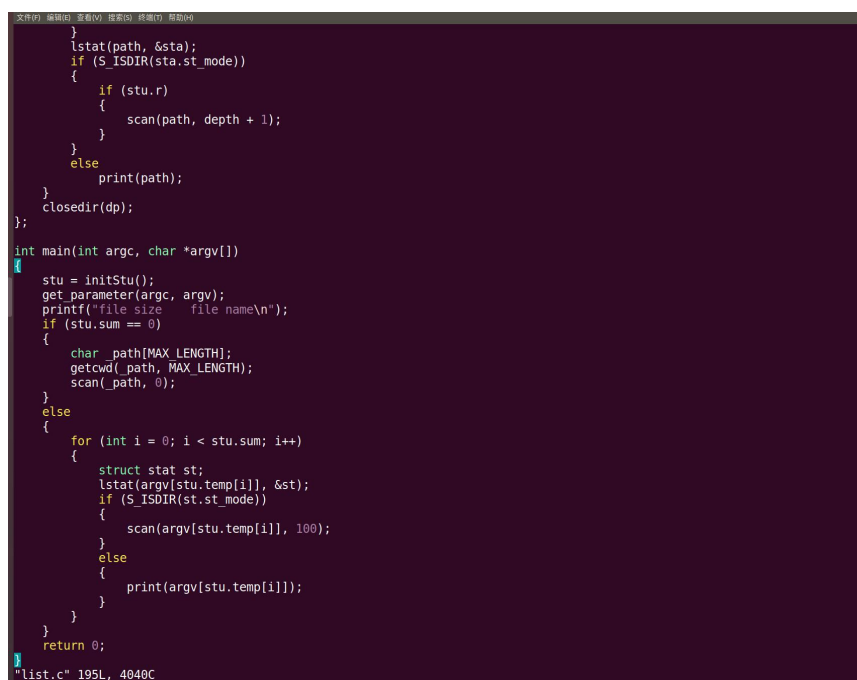
```
void print(char *path)
```

④ 用于扫描当前路径\*dir 目录下的所有文件和目录，并根据深度 depth 的值做不同的处理，scan() 是主要的递归函数，可以用于递归遍历整个目录及子目录

```
void scan(char *dir, int depth)
```

所定义的结构体如下：

```
struct Stu
{
    bool r;//判断是否有 r 命令
    bool a;//判断是否有 a 命令
    bool l;//判断是否有 l 命令
    bool h;//判断是否有 h 命令
    bool m;//判断是否有 m 命令
    int min;//l 命令指定文件大小的最小值
    int max;//h 命令指定文件大小的最大值
    int day;//m 命令指定时间
    int sum;//包含的 目录/文件名 的个数
    int temp[MAX_LENGTH]; //将选项中的目录/文件名储存在该数组
};
```



```
};
    lstat(path, &sta);
    if (S_ISDIR(sta.st_mode))
    {
        if (stu.r)
        {
            scan(path, depth + 1);
        }
        else
        {
            print(path);
        }
    }
    closedir(dp);
};

int main(int argc, char *argv[])
{
    stu = initStu();
    get_parameter(argc, argv);
    printf("file size    file name\n");
    if (stu.sum == 0)
    {
        char _path[MAX_LENGTH];
        getcwd(_path, MAX_LENGTH);
        scan(_path, 0);
    }
    else
    {
        for (int i = 0; i < stu.sum; i++)
        {
            struct stat st;
            lstat(argv[stu.temp[i]], &st);
            if (S_ISDIR(st.st_mode))
            {
                scan(argv[stu.temp[i]], 100);
            }
            else
            {
                print(argv[stu.temp[i]]);
            }
        }
    }
    return 0;
}
```

"list.c" 195L, 4040C

2. 使用 `gcc list.c -o list` 命令编译该代码文件

```
lhfh1@ubuntu:~$ vi list.c
lhfh1@ubuntu:~$ gcc list.c -o list
lhfh1@ubuntu:~$
```

3. 使用 `./list` 命令在当前目录运行该文件:

```
lhfh1@ubuntu:~$ vi list.c
lhfh1@ubuntu:~$ gcc list.c -o list
lhfh1@ubuntu:~$ ./list
file size  file name
  22209    beijing.html
 494314    gcc-3.4_3.4.6-6ubuntu3_i386.deb
 164846    gcc-3.4-base_3.4.6-6ubuntu3_i386.deb.1
 164846    gcc-3.4-base_3.4.6-6ubuntu3_i386.deb
    88     flow.awk
  12112    list
 1687592   cpp-3.4_3.4.6-6ubuntu3_i386.deb
 1921464   g++-3.4_3.4.6-6ubuntu3_i386.deb
   4040    list.c
   8980    examples.desktop
22210998   lab4.tar.gz
 1266146   libstdc++6-dev_3.4.6-6ubuntu3_i386.deb
lhfh1@ubuntu:~$
```

4. 查看指定目录下的文件:

```
lhfh1@ubuntu:~$ ./list test
file size  file name
   169     test/test1.c
   7636    test/test
   2805    test/t2.c
   1735    test/t4.c
  12588    test/test1
   7792    test/t4
   1260    test/test.c
   1332    test/t3.c
   7388    test/a.out
   7740    test/t2
   7704    test/t3
   1272    test/t1.c
lhfh1@ubuntu:~$
```

5. 使用 `./list -a` 查看文件目录下所有的文件, 其中也包含以 `.` 开头的文件:

```
lhfh1@ubuntu:~$ ./list -a
file size  file name
   655     .profile
  10952    .bash_history
  22209    beijing.html
   5406    .ICEauthority
 494314    gcc-3.4_3.4.6-6ubuntu3_i386.deb
 164846    gcc-3.4-base_3.4.6-6ubuntu3_i386.deb.1
  11008    .viminfo
   1237    .xsession-errors.old
    132    .xinputrc
 164846    gcc-3.4-base_3.4.6-6ubuntu3_i386.deb
    88     flow.awk
   220     .bash_logout
    51     .Xauthority
   267     .pam_environment
    25     .dmrc
  12112    list
 1687592   cpp-3.4_3.4.6-6ubuntu3_i386.deb
   1392    .xsession-errors
 1921464   g++-3.4_3.4.6-6ubuntu3_i386.deb
   3771    .bashrc
   4040    list.c
   8980    examples.desktop
    0      .sudo_as_admin_successful
22210998   lab4.tar.gz
 1266146   libstdc++6-dev_3.4.6-6ubuntu3_i386.deb
lhfh1@ubuntu:~$
```

6. 使用命令: `./list -r` 递归的查看该目录下所有文件

```
lhfh1@ubuntu:~$ ./list -r
file size    file name
    169    test/test1.c
    7636    test/test
   2805    test/t2.c
    1735    test/t4.c
  12588    test/test1
    7792    test/t4
    3798    test/test_linux_lab2/test1.c
  12588    test/test_linux_lab2/test1
    1260    test/test.c
    1332    test/t3.c
    7388    test/a.out
    7740    test/t2
    7704    test/t3
    1272    test/t1.c
  22209    beijing.html
494314    gcc-3.4_3.4.6-6ubuntu3_i386.deb
    268    lab4/rungdb
    762    lab4/files/testlab2.sh
    1395    lab4/files/process.c
  13628    lab4/files/memtest
    5009    lab4/files/testlab2.c
    8502    lab4/files/stat_log.py
  119902    lab4/linux-0.11.tar.gz
     0     lab4/hdc/umounted
    131    lab4/mount-hdc
    701    lab4/run
    115    lab4/dbg-asm
    119    lab4/dbg-c
63504384    lab4/hdc-0.11.img
  14896    lab4/bochsout.txt
     75    lab4/gdb-cmd.txt
    6960    lab4/linux-0.11/kernel/math/math_emulate.o
    7124    lab4/linux-0.11/kernel/math/math.a
```

7. 使用命令 `./list -l 10 -h 1000` 列出文件大小在 10~1000 之间的文件

```
lhfh1@ubuntu:~$ ./list -l 10 -h 1000
file size    file name
     88    flow.awk
lhfh1@ubuntu:~$
```

8. 使用命令 `./list -a -r -l 50000 -m 2` 递归式列出当前目录树下大小超 50KB 且 2 天内修改过的文件（包括文件名首字符为圆点的文件）

```
lhfh1@ubuntu:~$ ./list -a -r -l 50000 -m 2
file size    file name
.mozilla/firefox/89hxivcb.default/lock: No such file or directory
 65536    .local/share/zeitgeist/fts.index/term1ist.DB
 65536    .local/share/zeitgeist/fts.index/postlist.DB
 74440    .local/share/zeitgeist/activity.sqlite-wal
 66560    .local/share/zeitgeist/activity.sqlite
 76871    .cache/fcitx-qimpanel/qmlcache/21b49ce66dfff905e673d82c1a05bb4b0c772ff7.qmlc
327036    .cache/gnome-software/cssresource/f6382ec312737428383b9771f0adb4d9be7bc434-jami-qt-gnulinux-banner-20220504.png
1421509    .cache/gnome-software/odrs/ratings.json
 829148    .cache/gnome-software/fwupd/remotes.d/lvfs/metadata.xml.gz
 536193    .cache/wallpaper/0_5_2560_1408_792beab7550410d531e55f95b449f135
lhfh1@ubuntu:~$
```

9. 使用命令./list -- -l 显示取消-l 选项，将-l 当前文件/目录名处理

```
lhfh1@ubuntu:~$ ./list -- -l
file size  file name
-l: No such file or directory
lhfh1@ubuntu:~$
```

10. 使用命令./list \*

```
lhfh1@ubuntu:~$ ./list *
file size  file name
  22209  beijing.html
1687592  cpp-3.4_3.4.6-6ubuntu3_i386.deb
   8980  examples.desktop
    88   flow.awk
1921464  g++-3.4_3.4.6-6ubuntu3_i386.deb
494314  gcc-3.4_3.4.6-6ubuntu3_i386.deb
164846  gcc-3.4-base_3.4.6-6ubuntu3_i386.deb
164846  gcc-3.4-base_3.4.6-6ubuntu3_i386.deb.1
   268  lab4/rungdb
119902  lab4/linux-0.11.tar.gz
   131  lab4/mount-hdc
   701  lab4/run
   115  lab4/dbg-asm
   119  lab4/dbg-c
63504384  lab4/hdc-0.11.img
  14896  lab4/bochsout.txt
    75  lab4/gdb-cmd.txt
```

## 五、实验总结

通过本次实验，熟练了使用 vi 编辑文件，熟悉了 Linux 的一些系统调用和库函数，体会了 Shell 文件通配符的处理方式以及命令对选项的处理方式，对于 ls 命令有了更深的理解。

## 六、实验代码

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

```
#define MAX_LENGTH 255
```

```

struct Stu
{
    bool r;
    bool a;
    bool l;
    bool h;
    bool m;
    int min;
    int max;
    int day;
    int sum;
    int temp[MAX_LENGTH];
};

struct Stu stu;

struct Stu initStu()
{
    struct Stu stu;
    stu.r = false;
    stu.a = false;
    stu.l = false;
    stu.h = false;
    stu.m = false;
    stu.min = 0;
    stu.max = 0;
    stu.day = 0;
    stu.sum = 0;
    return stu;
};

void get_parameter(int argc, char *argv[])
{
    int i = 0;
    bool para__ = false;
    for (i = 1; i < argc; i++)
    {
        if (argv[i][0] == '-' && argv[i][1] == '-')
        {
            para__ = true;
            continue;
        }
        /*如果有一选项 则所有自定义选项命令终止分析*/
        if (argv[i][0] == '-' && !para__)

```

```

        {
            switch (argv[i][1])
            {
                case 'r':
                    stu.r = true;
                    break;
                case 'a':
                    stu.a = true;
                    break;
                case 'l':
                    stu.l = true;
                    i++;
                    stu.min = atoi(argv[i]);
                    break;
                case 'h':
                    stu.h = true;
                    i++;
                    stu.max = atoi(argv[i]);
                    break;
                case 'm':
                    stu.m = true;
                    i++;
                    stu.day = atoi(argv[i]);
                    break;
                default:
                    break;
            }
        }
    else
    {
        stu.temp[stu.sum] = i;//将目录路径存入数组
        stu.sum++;
    }
}

};

```

```

void print(char *path)
{
    struct stat st;
    time_t t_now;
    char *fileName;
    time(&t_now);
    int ret = stat(path, &st);
    if (ret == -1)

```

```

    {
        printf("%s: No such file or directory\n", path);
    }
    else
    {
        if (stu.l)
        {
            if (st.st_size < stu.min)
                return;
        }
        if (stu.h)
        {
            if (st.st_size > stu.max)
                return;
        }
        if (stu.m)
        {
            if (t_now - st.st_mtime > stu.day * 24 * 60 * 60)
                return;
        }
        printf("%10ld  %s\n", st.st_size, path);
    }
};

void scan(char *dir, int depth)
{
    DIR *dp;
    struct dirent *entry;
    struct stat sta;
    char path[512] = {0};
    if ((dp = opendir(dir)) == NULL)
    {
        printf("%s\n:No such file or directory\n", dir);
        return;
    }
    while ((entry = readdir(dp)) != NULL)
    {
        if (strcmp(".", entry->d_name) == 0 || strcmp("..",
entry->d_name) == 0)
            continue;
        if (entry->d_name[0] == '.' && stu.a == false)
            continue;
        if (depth == 0)
        {

```



```

        sprintf(path, "%s", entry->d_name);
    }
    else
    {
        sprintf(path, "%s/%s", dir, entry->d_name);
    }
    lstat(path, &sta);
    if (S_ISDIR(sta.st_mode))
    {
        if (stu.r)
        {
            scan(path, depth + 1);
        }
    }
    else
        print(path);
}
closedir(dp);
};

```

```

int main(int argc, char *argv[])
{
    stu = initStu();
    get_parameter(argc, argv);
    printf("file size    file name\n");
    if (stu.sum == 0)
    {
        char _path[MAX_LENGTH];
        getcwd(_path, MAX_LENGTH);
        scan(_path, 0);
    }
    else
    {
        for (int i = 0; i < stu.sum; i++)
        {
            struct stat st;
            lstat(argv[stu.temp[i]], &st);
            if (S_ISDIR(st.st_mode))
            {
                scan(argv[stu.temp[i]], 100);
            }
            else
            {
                print(argv[stu.temp[i]]);
            }
        }
    }
}

```

```
        }  
    }  
}  
return 0;  
}
```