# 编程作业：遍历文件目录

1. 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。

1. 实验要求

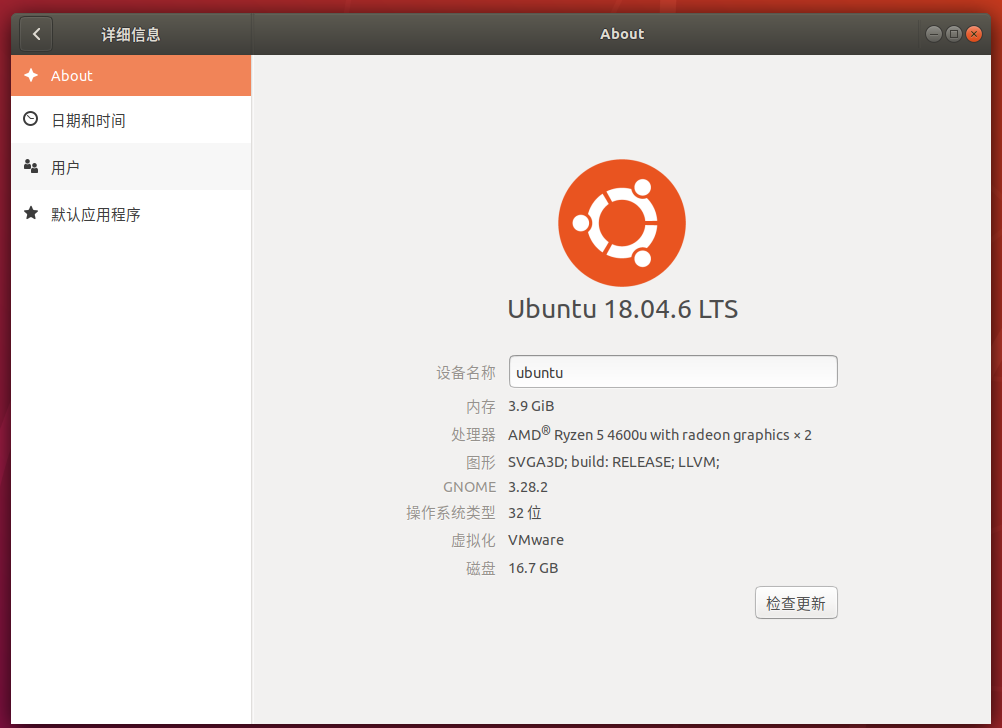
1.与ls命令类似，处理对象可以有0到多个

* 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()

1. 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[])

1. 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)
2. 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

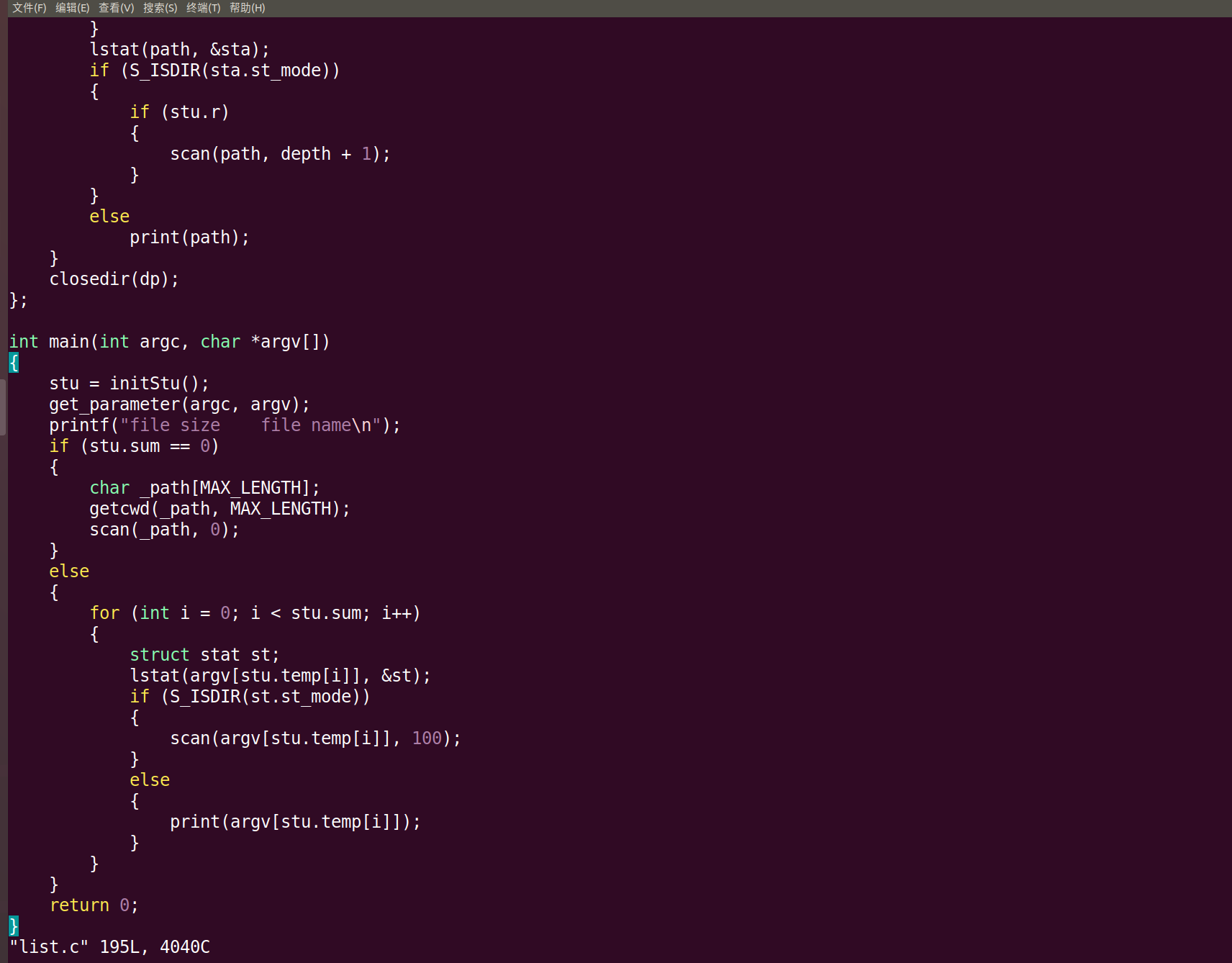
int max;//h命令指定文件大小的最大值

int day;//m命令指定时间

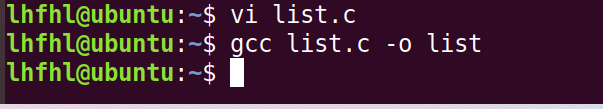
int sum;//包含的 目录/文件名 的个数

int temp[MAX\_LENGTH];//将选项中的目录/文件名储存在该数组

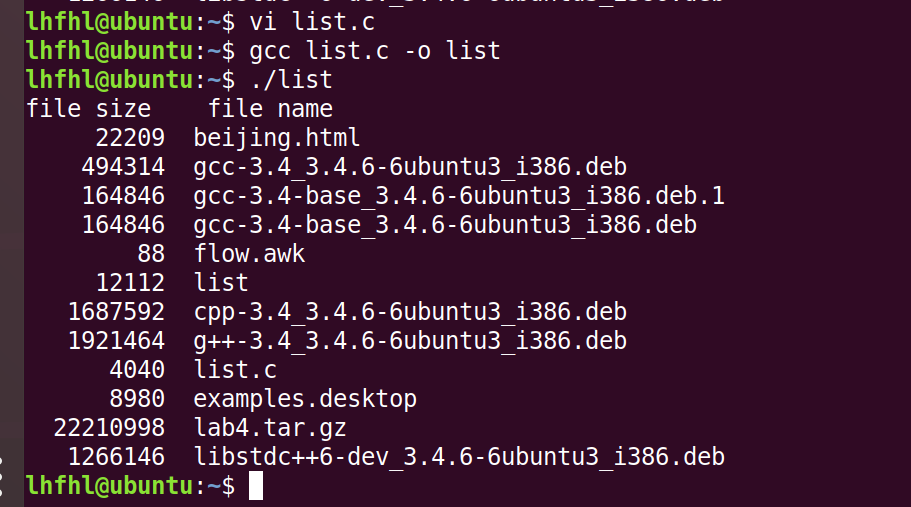
};



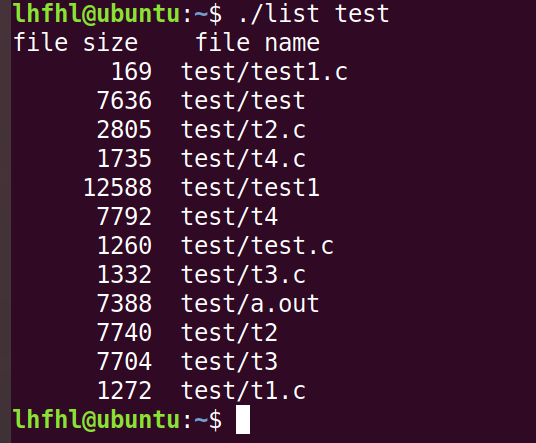
2.使用gcc list.c -o list命令编译该代码文件

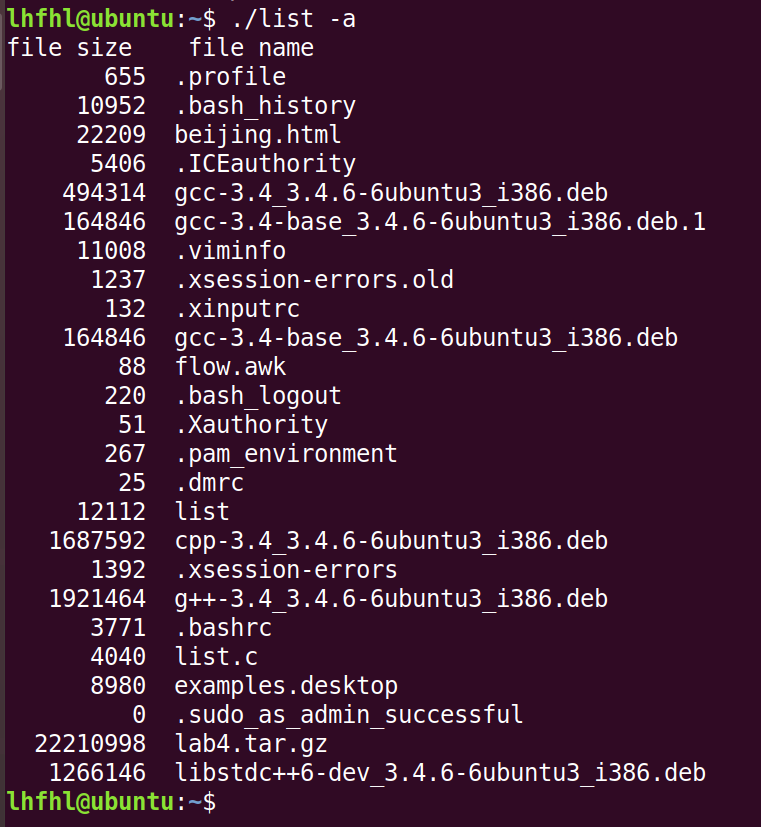


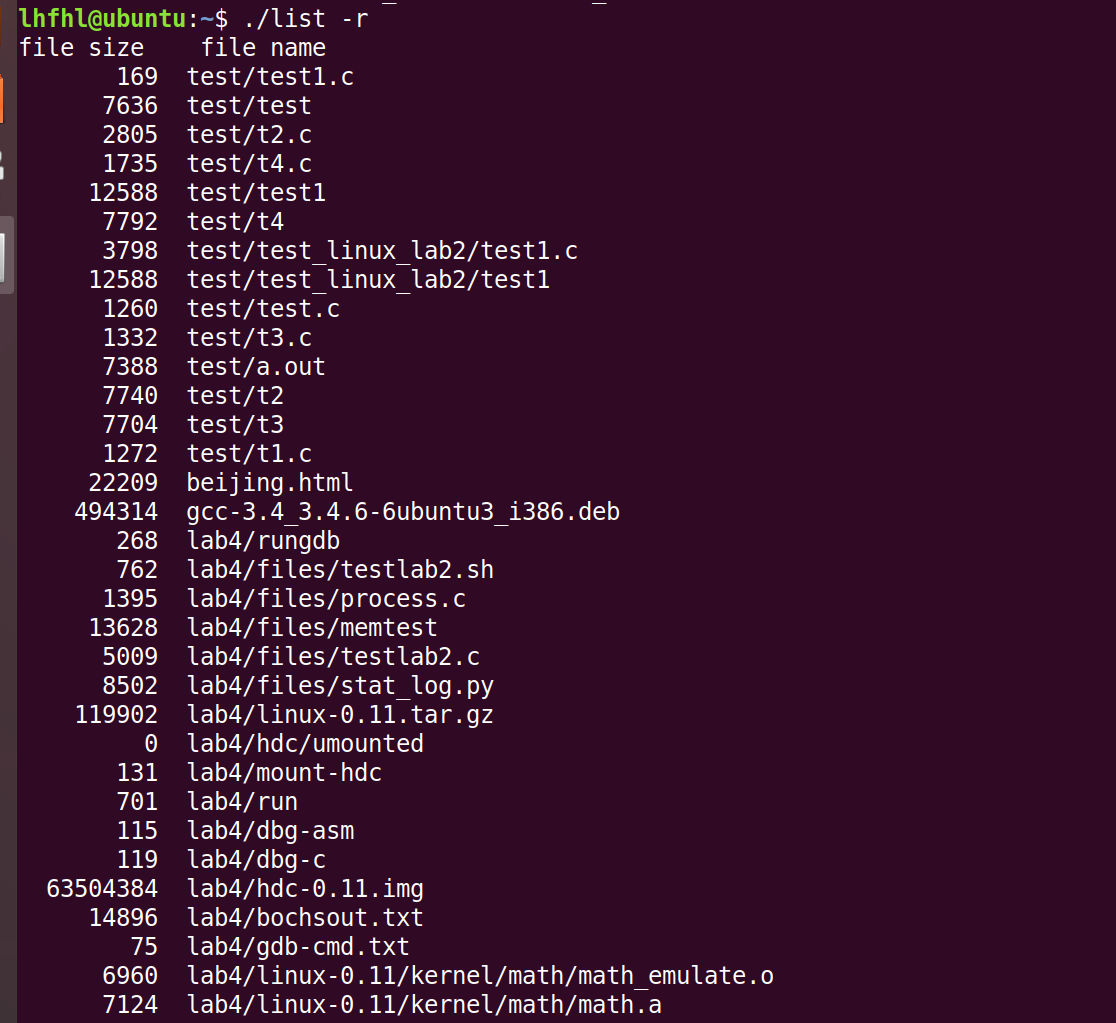
3.使用./list 命令在当前目录运行该文件：

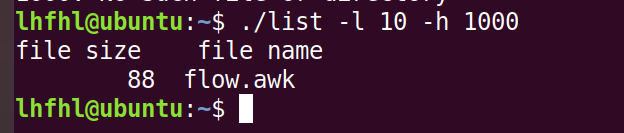


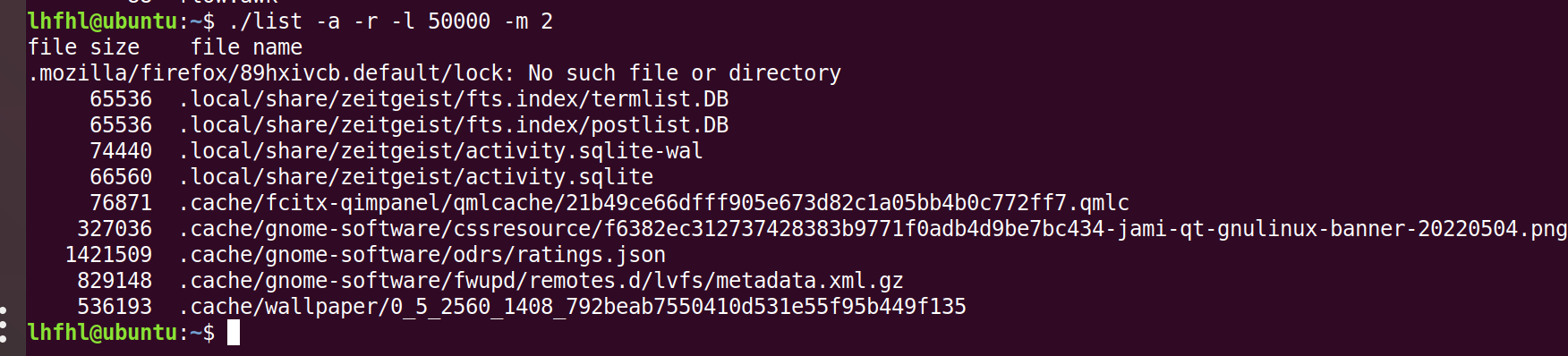
4.查看指定目录下的文件：



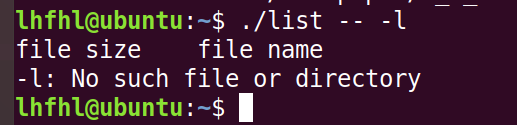
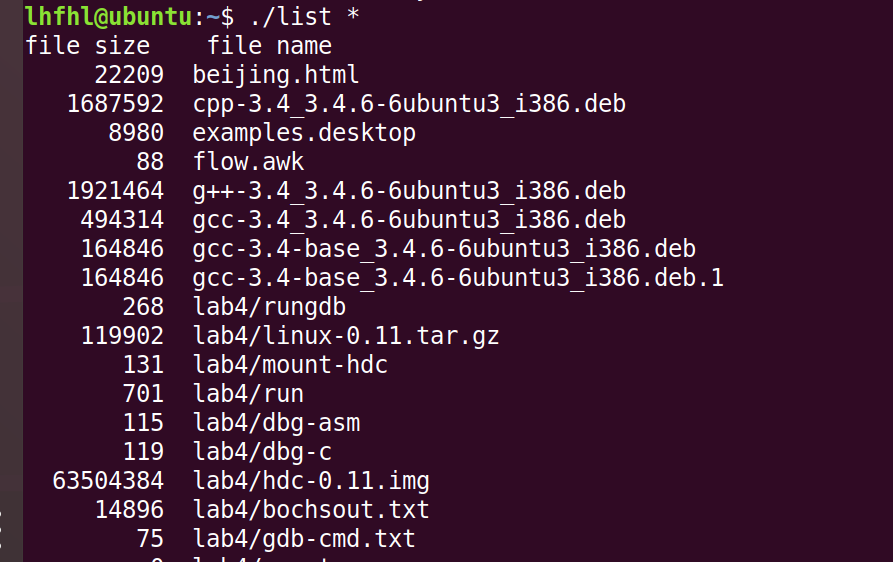
1. 使用 ./list -a查看文件目录下所有的文件，其中也包含以.开头的文件：  
   
2. 使用命令： ./list -r 递归的查看该目录下所有文件



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


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）  


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

  
10.使用命令./list \*  


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--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;

}