

**Linux开发环境及应用**



**遍历文件目录**

**姓 名 鄭毓恒**

**学 院 计算机学院**

**专 业 计算机科学与技术**

**班 级 2020211302**

**学 号 2020211262**

**任课教师 张雪松**

**2023年 4 月**

**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。
* 对选项的处理，自行编程逐个分析命令行参数。不考虑多选项挤在一个命令行参数内的情况。

**作业要求**

1. 与ls命令类似，处理对象可以有0到多个
   1. 0个：列出当前目录下所有文件（不含点打头的文件，如.profile）
   2. 普通文件：列出文件
   3. 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—
   1. 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）

* 1. 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）

* 1. l 后跟一整数，限定文件大小的最小值（字节）
  2. h 后跟一整数，限定文件大小的最大值（字节）
  3. m 后跟一整数n，限定文件的最近修改时间必须在n天内
  4. -- 显式地终止命令选项分析

**高层数据信息定义**

|  |  |
| --- | --- |
| 数据结构定义 | 功能 |
| typedef struct  {  int \_r;  int \_a;  int \_l;  int minbyte;  int \_h;  int maxbyte;  int \_m;  int nday;  int path\_count;  int path\_loc[MAX\_NUM];  }option; | 用于表示各命令选项是否存在，以及存储一些参数所需的数据，例如minbyte是-l选项所需的限定文件大小的最小值。  path\_count是指令中的路径数量。path\_loc存储各路径在argv的下标。 |

|  |  |
| --- | --- |
| 全局常量定义 | 功能 |
| #define MAX\_NUM 256 | 同时用于表示最长路径长度和命令可处理路径的最大数量。 |

|  |  |
| --- | --- |
| 全局变量定义 | 功能 |
| option opt; | 表示各命令选项的状态 |

**函数定义**

|  |  |
| --- | --- |
| 函数定义 | 功能 |
| void readOpt(int argc, char\* argv[]); | 初始化全局变量opt，然后根据输入，存储各命令选项的状态和路径信息。 |
| void do\_ls(char\* path); | 扫描路径为path的目录，根据opt的信息，决定哪些文件要经过output函数输出。该函数同时也是-r选项下的递归函数，递归遍历子目录。 |
| void output(char\* path); | 打印路径为path的文件名称和大小 |

**总体设计**

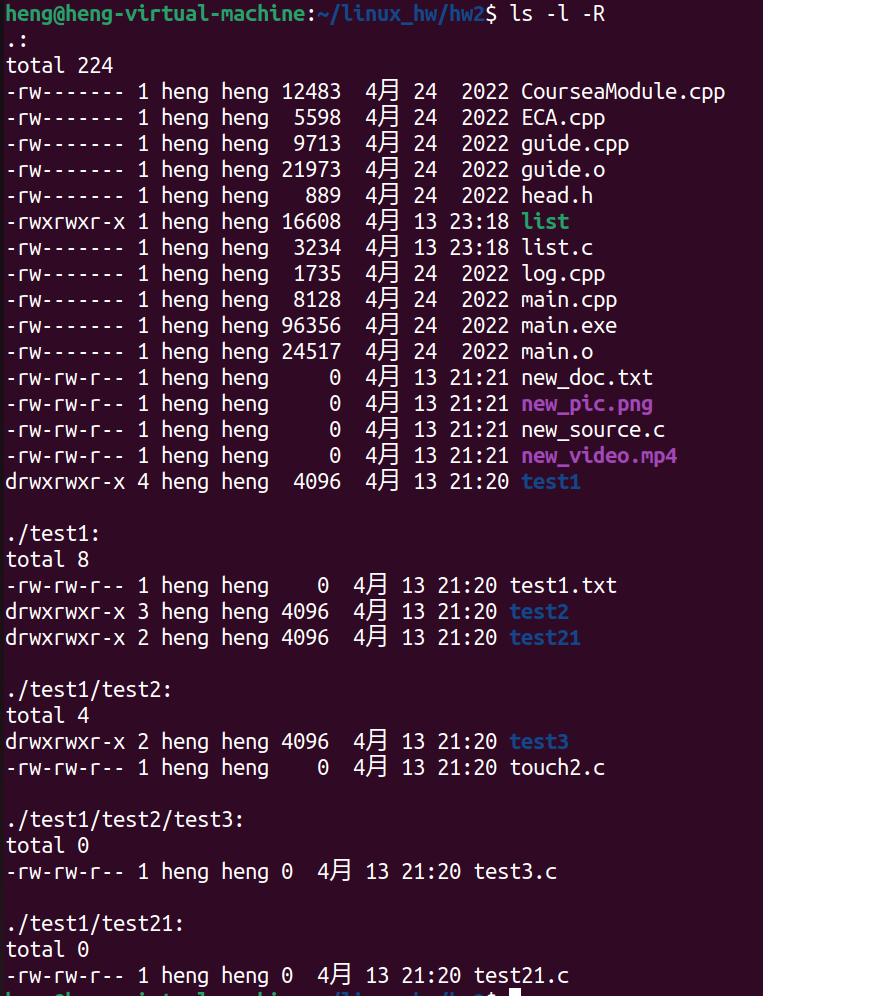
主函数首先调用readOpt函数，根据argc和argv两个参数，初始化全局变量opt，存储输入中的选项和路径信息。然后，根据opt中的路径信息，遍历所有输入的遍历。假如路径是文件，直接调用output函数进行输出。假如路径是文件夹，调用do\_ls函数进行遍历。假如没有输入路径，调用do\_ls函数遍历当前目录。

函数readOpt首先将全局结构体opt中的成员全部初始化为0，然后读取argv数组，判断是否为选项，然后将对应的成员置为1，直到遇到--选项。如果是路径，则存储路径在argv中的下标。

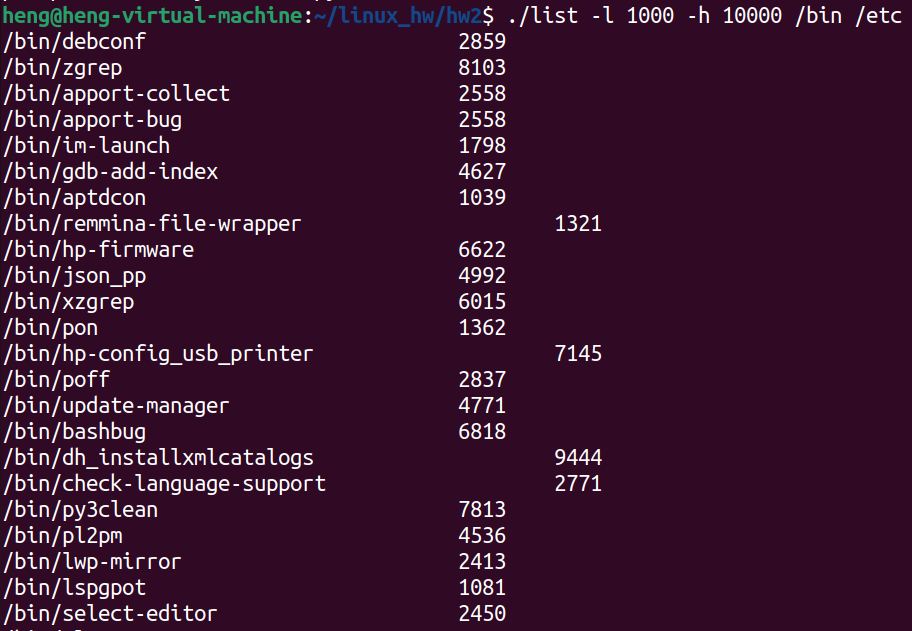
函数do\_ls打开路径为参数path的文件夹，然后遍历文件夹中的内容。假如为文件，根据opt的信息，决定是否调用output函数进行输出。遍历完毕后，假如输入有-r选项，也就是需要递归遍历子目录。重新开始遍历当前目录，遍历到子目录后，再次调用do\_ls进行遍历。

函数output将输出文件的名称和大小。输出之前，会先检查是不是一个有效文件，是再输出。

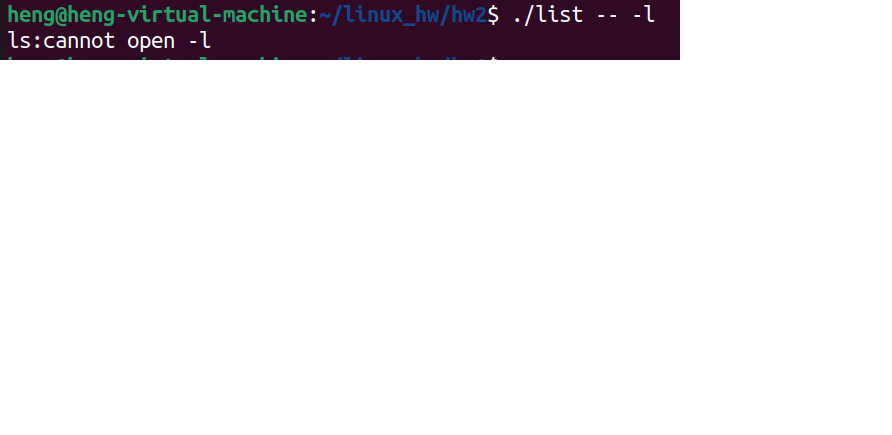
**样例测试和输出**



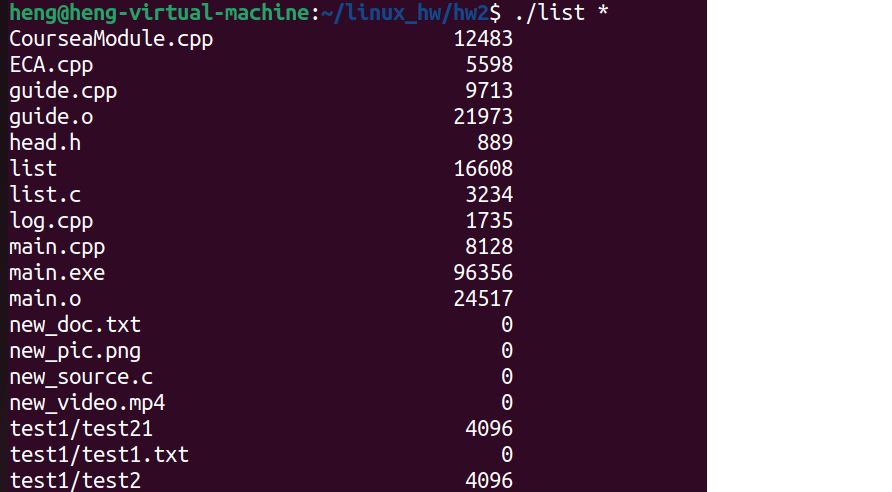
用于测试的自建文件夹内容如上。程序list是我写的目录遍历程序。



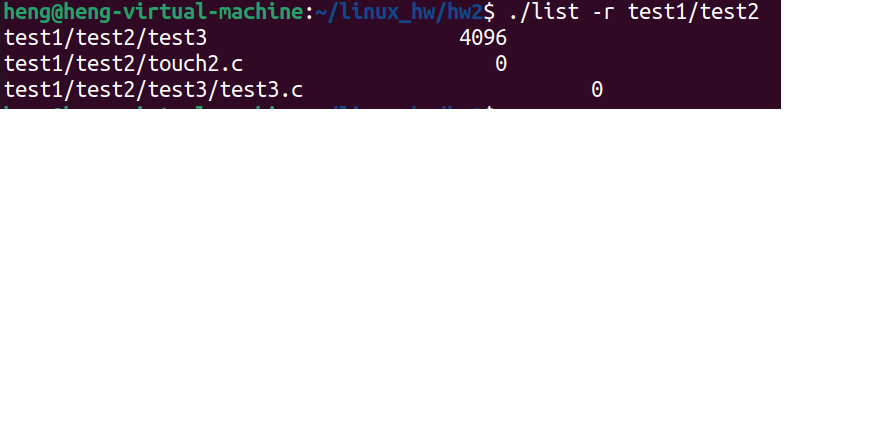
指令./list -l 1000 -h 10000 /bin /etc。列出/bin和/etc文件夹下，大小为1000至10000的文件。



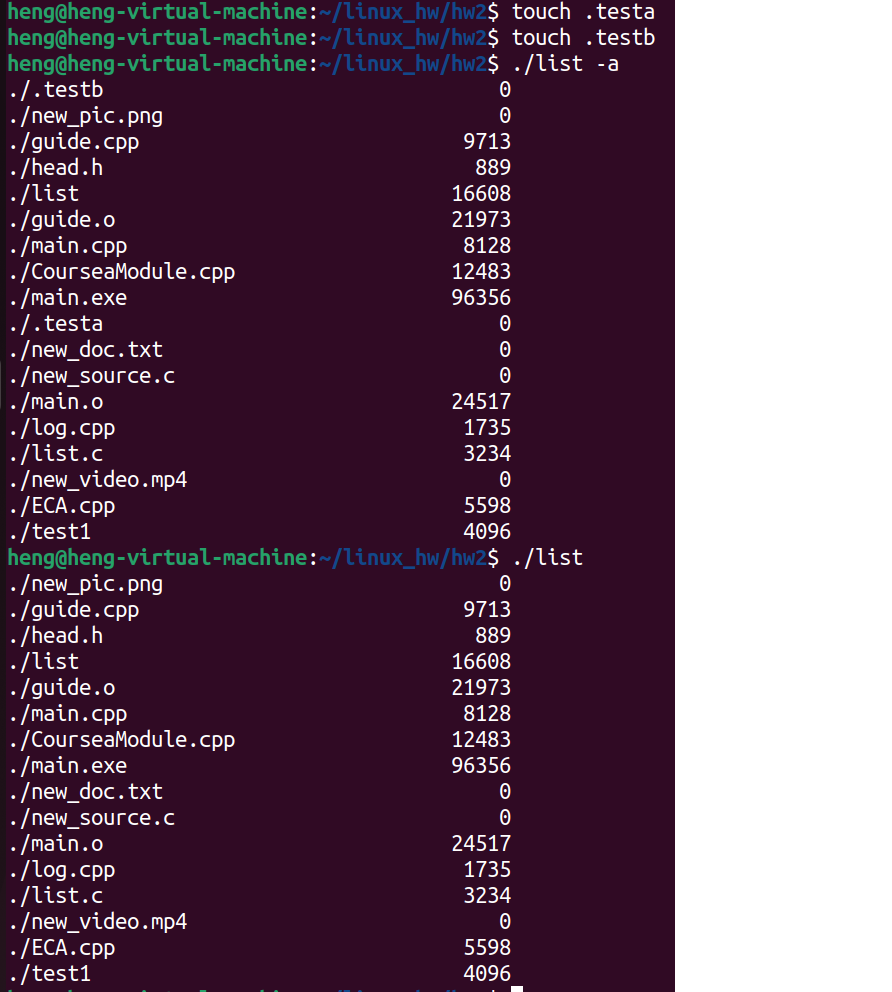
指令./list -- -l。程序取消了-l，将-l当文件或目录进行处理。



指令./list \*。使用\*，输出了当前目录下的所有文件。



指令./list -r test1/test2。递归遍历当前目录下的test1目录的test2目录。



文件.testa和.testb以.开头，只有-a选项下才会输出。经过上图对比，确实只有在-a选项下，才输出.testa和.testb的信息。

经过以上多个测试，程序能正确输出内容，达到作业要求。

**源代码**

|  |
| --- |
| #include<stdio.h>  #include<dirent.h>  #include<sys/stat.h>  #include<stdlib.h>  #include<string.h>  #include<time.h>  #include<unistd.h>  #define MAX\_NUM 256  typedef struct  {  int \_r;  int \_a;  int \_l;  int minbyte;  int \_h;  int maxbyte;  int \_m;  int nday;  int path\_count;  int path\_loc[MAX\_NUM];  }option;  option opt;  void readOpt(int argc, char\* argv[]);  void do\_ls(char\* path);  void output(char\* path);  int main(int argc, char\* argv[])  {  readOpt(argc, argv);  if (opt.path\_count == 0)  {  do\_ls(".");  }  else  {  for (int i = 0; i < opt.path\_count; i++)  {  struct stat st;  stat(argv[opt.path\_loc[i]], &st);  if (S\_ISDIR(st.st\_mode))  {  do\_ls(argv[opt.path\_loc[i]]);  }  else  {  output(argv[opt.path\_loc[i]]);  }  }  }  return 0;  }  void readOpt(int argc, char\* argv[])  {  opt.\_r = 0;  opt.\_a = 0;  opt.\_l = 0;  opt.minbyte = 0;  opt.\_h = 0;  opt.maxbyte = 0;  opt.\_m = 0;  opt.nday = 0;  opt.path\_count = 0;  int opt\_stop = 0;  for (int i = 1; i < argc; i++)  {  if (argv[i][0] == '-' && !opt\_stop)  {  switch (argv[i][1])  {  case 'r':  opt.\_r = 1;  break;  case 'a':  opt.\_a = 1;  break;  case 'l':  opt.\_l = 1;  opt.minbyte = atoi(argv[++i]);  break;  case 'h':  opt.\_h = 1;  opt.maxbyte = atoi(argv[++i]);  break;  case 'm':  opt.\_m = 1;  opt.nday = atoi(argv[++i]);  break;  case '-':  opt\_stop = 1;  break;  default:  break;  }  }  else  {  opt.path\_loc[opt.path\_count++] = i;  }  }  }  void do\_ls(char\* path)  {  DIR\* dir\_ptr;  struct dirent \*direntp;  if ((dir\_ptr = opendir(path)) == NULL)  {  fprintf(stderr, "ls:cannot open %s\n", path);  return;  }  while ((direntp = readdir(dir\_ptr)) != NULL)  {  if (strcmp(".", direntp->d\_name) == 0 || strcmp("..", direntp->d\_name) == 0)  continue;  if (direntp->d\_name[0] == '.' && !opt.\_a)  continue;  char doc[MAX\_NUM \* 2];  sprintf(doc, "%s/%s", path, direntp->d\_name);  struct stat st;  stat(doc, &st);  if (opt.\_l)  {  if (st.st\_size < opt.minbyte)  continue;  }  if (opt.\_h)  {  if (st.st\_size > opt.maxbyte)  continue;  }  if (opt.\_m)  {  time\_t cur\_time;  time(&cur\_time);  if (cur\_time - st.st\_mtime > opt.nday \* 24 \* 60 \* 60)  continue;  }  output(doc);  }  closedir(dir\_ptr);  if (opt.\_r)  {  if ((dir\_ptr = opendir(path)) == NULL)  {  fprintf(stderr, "ls:cannot open %s\n", path);  return;  }  while ((direntp = readdir(dir\_ptr)) != NULL)  {  if (strcmp(".", direntp->d\_name) == 0 || strcmp("..", direntp->d\_name) == 0)  continue;  if (direntp->d\_name[0] == '.' && !opt.\_a)  continue;  struct stat st;  char next[MAX\_NUM \* 2];  sprintf(next, "%s/%s", path, direntp->d\_name);  if (lstat(next, &st) == -1)  perror(next);  if (S\_ISDIR(st.st\_mode))  do\_ls(next);  }  closedir(dir\_ptr);  }  }  void output(char\* path)  {  struct stat st;  if (stat(path, &st) == -1)  {  fprintf(stderr, "ls:cannot open %s\n", path);  return;  }  printf("%-20s\t\t%10ld\n", path, st.st\_size);  } |