lab04Copy One Directory

软件工程2018级 1813075 刘茵

## Target

1. Write a c/c++ program
2. To implement copy one diretory and it's subdiretories with multi-threads
3. Gcc
4. Install GCC Software Colletion

>> sudo apt-get install build-essential

1. How to use GCC

* [gcc and make](https://www3.ntu.edu.sg/home/ehchua/programming/cpp/gcc_make.html)

1. Struct of directory

**struct dirent**

**{**

**ino\_t d\_ino; //d\_ino 此目录进入点的inode**

**ff\_t d\_off; //d\_off 目录文件开头至此目录进入点的位移**

**signed short int d\_reclen; //d\_reclen \_name 的长度, 不包含NULL 字符**

**unsigned char d\_type; //d\_type d\_name 所指的文件类型 d\_name 文件名**

**har d\_name[256];**

**};**

**opendir()**

**readdir()**

**closedir()**

1. Write a c program to implement copy one diretory and it's subdiretories, and the program also verifies the result

* 编写main.cpp文件：

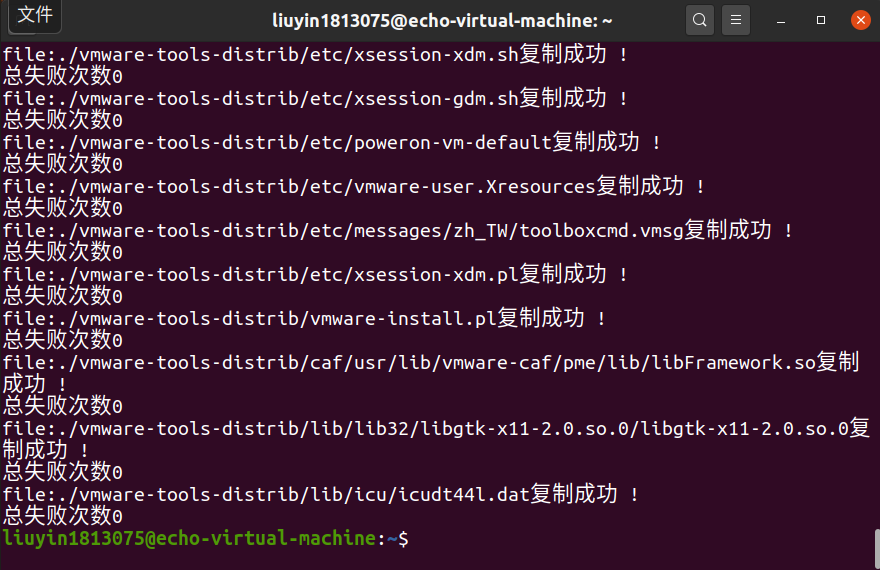
1. #include <iostream>
2. #include <string>
3. #include <unistd.h>
4. #include <sys/stat.h>
5. #include <dirent.h>
6. #include <pthread.h>
7. #include <semaphore.h>
8. #include <stack>
9. #include <cstring>
10. #include <fcntl.h>
11. #include <cstdlib>
12. #include <vector>
13. #include <sys/wait.h>
15. #define NUM\_OF\_THREADS 10
16. **using** **namespace** std;
17. sem\_t my\_sem;
19. **struct** FileOperation
20. {
21. string src\_file;
22. string dst\_file;
23. };
24. **int** times = 0;
25. stack<FileOperation> homework;
27. //复制文件
28. **void** cp(string src\_file, string dst\_file)
29. {
30. **int** fd\_read = open(src\_file.c\_str(), O\_RDONLY, S\_IREAD | S\_IWRITE | S\_IRGRP | S\_IROTH);
31. **int** fd\_write = open(dst\_file.c\_str(), O\_WRONLY | O\_CREAT, S\_IREAD | S\_IWRITE | S\_IRGRP | S\_IWGRP | S\_IROTH);
32. cout << dst\_file.c\_str() << endl;
33. cout << src\_file.c\_str() << endl;
34. cout << fd\_read << ":" << fd\_write << endl;
35. cout << src\_file << ":" << dst\_file << endl;
36. **if** (fd\_read == -1 || fd\_write == -1)
37. {
38. cout << "when copy " << src\_file << "复制失败 !" << endl;
39. cout << "failed:" << times << endl;
40. times++;
41. }
42. **else**
43. {
44. **char** buf[1024];
45. **int** size = 0;
46. **while** ((size = read(fd\_read, buf, 1024)) > 0)
47. {
48. write(fd\_write, buf, size);
49. }
50. cout << "file:" << src\_file << "复制成功 !" << endl;
51. }
52. close(fd\_write);
53. close(fd\_read);
54. cout << "总失败次数" << times << endl;
55. }
57. //遍历目录
58. **void** walk\_dir(**const** **char** \*src\_dir, **const** **char** \*dst\_dir)
59. {
60. **struct** dirent \*filename;
61. DIR \*dir;
62. dir = opendir(src\_dir); //获得目录信息
63. **if** (dir == NULL)
64. {
65. cout << "打开 src\_dir 失败" << endl;
66. exit(0); //结束当前进程
67. }
68. cout << "open src\_dir success!" << endl;
69. **char** path[256];
70. **while** ((filename = readdir(dir)) != NULL)
71. { //读取文件夹下文件
72. **if** ((!strcmp(filename->d\_name, ".")) || (!strcmp(filename->d\_name, "..")))
73. {
74. **continue**; //遇到. ..就跳过
75. }
76. snprintf(path, 256, "%s/%s", src\_dir, filename->d\_name); //按照"%S"格式储存到path中
77. **struct** stat s;
78. lstat(path, &s); //获取path详细信息储存进s
79. **if** (S\_ISDIR(s.st\_mode))
80. { // S\_ISDIR()函数 判断一个路径是否为目录
81. **char** sub\_src\_dir[256];
82. **char** sub\_dst\_dir[256];
83. snprintf(sub\_dst\_dir, 256, "%s/%s", dst\_dir, filename->d\_name);
84. snprintf(sub\_src\_dir, 256, "%s/%s", src\_dir, filename->d\_name);
85. cout << sub\_dst\_dir << endl;
86. mkdir(sub\_dst\_dir, S\_IWUSR | S\_IRUSR | S\_IXUSR | S\_IRGRP | S\_IXGRP | S\_IROTH | S\_IXOTH); //!!竟然把dst写成了src
87. walk\_dir(sub\_src\_dir, sub\_dst\_dir);
88. }
89. **else**
90. { //是文件
91. **char** dst\_file[256];
92. **char** src\_file[256];
93. snprintf(dst\_file, 256, "%s/%s", dst\_dir, filename->d\_name);
94. snprintf(src\_file, 256, "%s/%s", src\_dir, filename->d\_name);
95. **struct** FileOperation new\_operation;
96. new\_operation.src\_file = src\_file;
97. new\_operation.dst\_file = dst\_file;
98. homework.push(new\_operation);
99. }
100. }
101. closedir(dir);
102. }
104. **void** \*run(**void** \*)
105. {
106. **struct** FileOperation opreation;
107. **while** (!homework.empty())
108. {
109. sem\_wait(&my\_sem);
110. opreation.dst\_file = homework.top().dst\_file;
111. opreation.src\_file = homework.top().src\_file;
112. homework.pop();
113. sem\_post(&my\_sem); //释放锁
114. cp(opreation.src\_file, opreation.dst\_file);
115. }
116. **return** NULL;
117. }
118. **int** main(**int** argc, **char** \*argv[])
119. {
120. sem\_init(&my\_sem, 0, 1);
121. **if** (argc < 3)
122. {
123. cout << "please give right path" << endl;
124. exit(0);
125. }
126. **struct** stat s;
127. //检查文件夹是否有校
128. lstat(argv[1], &s);
129. **if** (!S\_ISDIR(s.st\_mode))
130. {
131. cout << "the source path is wrong" << endl;
132. exit(0);
133. }
134. //检查目标文件夹是否有效;
135. lstat(argv[2], &s);
136. **if** (!S\_ISDIR(s.st\_mode))
137. {
138. cout << "the dest path is wrong" << endl;
139. }
140. walk\_dir(argv[1], argv[2]);
141. vector<pthread\_t> threads;
142. threads.resize(NUM\_OF\_THREADS); //设置线程数目
143. **for** (**int** i = 0; i < threads.size(); i++)
144. {
145. pthread\_create(&threads[i], NULL, run, NULL);
146. }
147. **for** (**int** i = 0; i < threads.size(); i++)
148. {
149. pthread\_join(threads[i], NULL);
150. }
151. **return** 0;
152. }

* 编译并运行多线程main.cpp



* 将vmware-tools-distrib文件夹及其子文件夹下内容拷贝





(输出成功提示)

* 将源文件夹和拷贝文件夹下的内容进行md5sum验证





* 将生成的origin.txt 和 newone.txt文件导出到本地windows操作系统下
* 在windows下编写python代码比较md5值

Python验证代码：

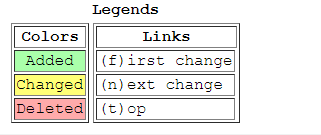
1. **import** difflib
2. **import** sys

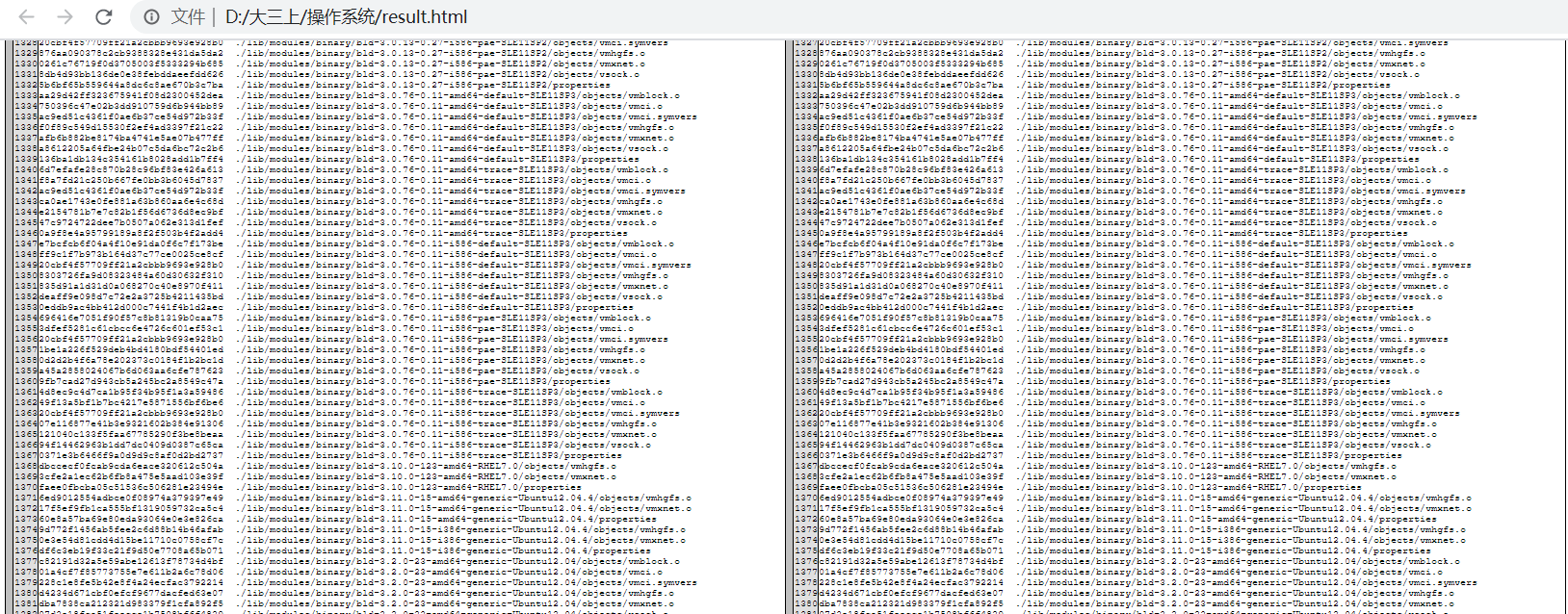
5. # 读取配置文件函数
6. **def** read\_file(file\_name):
7. **try**:
8. file\_handle = open(file\_name, 'r')
9. text = file\_handle.read().splitlines()  # 读取后以行进行分割
10. file\_handle.close()
11. **return** text
12. **except** IOError as error:
13. **print**('Read file Error: {0}'.format(error))
14. sys.exit()

17. # 比较两个文件并输出html格式的结果
18. **def** compare\_file(file1\_name, file2\_name):
19. **if** file1\_name == "" or file2\_name == "":
20. **print**('文件路径不能为空：file1\_name的路径为：{0}, file2\_name的路径为：{1} .'.format(file1\_name, file2\_name))
21. sys.exit()
22. text1\_lines = read\_file(file1\_name)
23. text2\_lines = read\_file(file2\_name)
24. diff = difflib.HtmlDiff()  # 创建htmldiff 对象
25. result = diff.make\_file(text1\_lines, text2\_lines)  # 通过make\_file 方法输出 html 格式的对比结果
26. #  将结果保存到result.html文件中并打开
27. **try**:
28. with open('result.html', 'w') as result\_file:      #同 f = open('result.html', 'w') 打开或创建一个result.html文件
29. result\_file.write(result)                      #同 f.write(result)
30. **except** IOError as error:
31. **print**('写入html文件错误：{0}'.format(error))

34. **if** \_\_name\_\_ == "\_\_main\_\_":
35. compare\_file('./newone.txt', './origin.txt')

* 在生成的html文件中进行查验。





发现没有颜色出现，全部对应相同。=>复制成功。