

Linux 下链表与树实验报告

191220008 计算机科学与技术系 陈南瞳

一、Linux 安装

- 1、一开始完全不知道 VMware Workstation、Ubuntu、WinSCP 是什么东西？和 Linux 有什么关系？在 Linux 虚拟机的安装中和在使用 Linux 时起的作用？在一步步地操作后才逐渐明白这些东西的作用和之间的联系。
- 2、因为自己使用的是 mac 下装的双系统 Windows，本以为需要回到 MacOS 系统下才能装虚拟机，但其实发现不需要，并没有关系。
- 3、一开始不明白换源是什么操作，结果在南大镜像源上多下了一次 Ubuntu……

二、代码编辑

- 1、发现助教给的 main.cpp 里有很多行的末尾都有“^M”这个符号，在 CSDN 查到是和行末尾的回车换行有关系。

linux文件中的^M

原创 [vivian_wanjin](#) 最后发布于2018-09-24 23:21:39 阅读数 1549 ☆ 收藏

[展开](#)

基于 DOS/Windows 的文本文件在每一行末尾有一个 CR（回车）和 LF（换行），而 UNIX 文本只有一个换行，即 win 每行结尾为 \r\n，而 linux 只有一个 \n

如果 win 下的文档上传到 linux，每行的结尾都会出现一个 ^M，（M 是 ctrl+v, ctrl+m）

如果是单个文档的话，可以用 vi 打开，删除行尾的 ^M

```
1 | :%s/\r//g
```

但如果批量去除的话就不能用 vi 了，

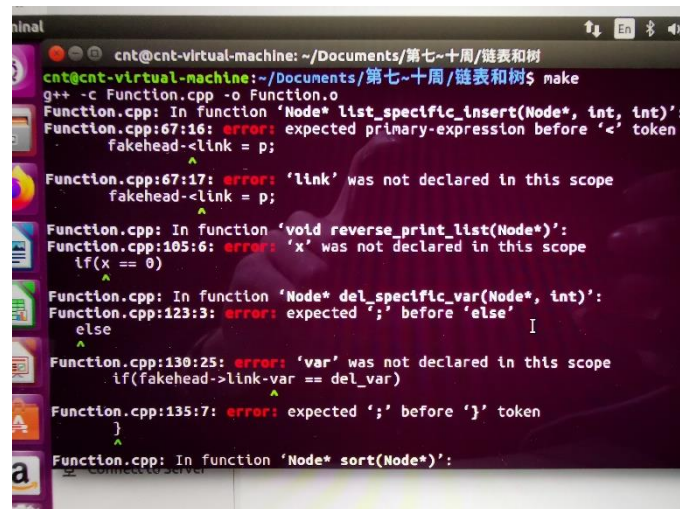
方法 1：用 dos2unix 工具，把 win 文档转换成 linux 下文档命令：`find ./ -type f -print0 | xargs -0 dos2unix` 如果想把 linux 下的文档转换成 win 下的：`find ./ -type f -print0 | xargs -0 unix2dos`

方法 2：用 sed 命令把 win 文档转换成 linux 下文档：`find ./ -type f -print0 | xargs -0 sed -i 's/^M//'` 把 linux 下的文档转换成 win 下的：`find ./ -type f -print0 | xargs -0 sed -i 's//^M'`

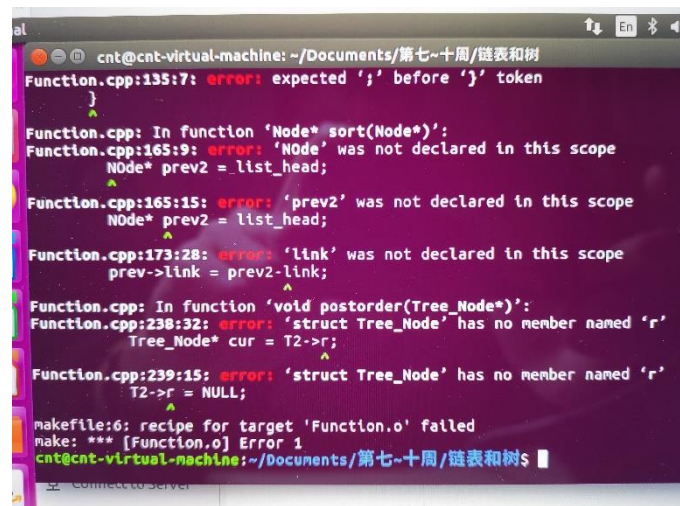
- 2、没有了 VS 里的自动对齐，写代码时需要自己对齐，很不方便，用 tab 又发现分隔太大。
- 3、由于没有自动对齐，两个配对大括号有时没有写在同一列，就容易造成大括号写多或写少。
- 4、有时自己不小心把正确的代码删除了，却发现无法撤销上一步操作，在网上查找了一下也没有找到（也可能是我没找到？）。
- 5、一开始不知道如何创建 makefile 文件，后来发现可以在终端 touch makefile，并在终端编辑；也可以在相应目录内创建一个 New Document 命名为 makefile，并在里面直接编辑。
- 6、由于自己的文件是中文，无法直接通过在终端输入路径来找到相应目录，需要直接在文件内打开终端来进行操作。

三、 Debug

1、 语法错误

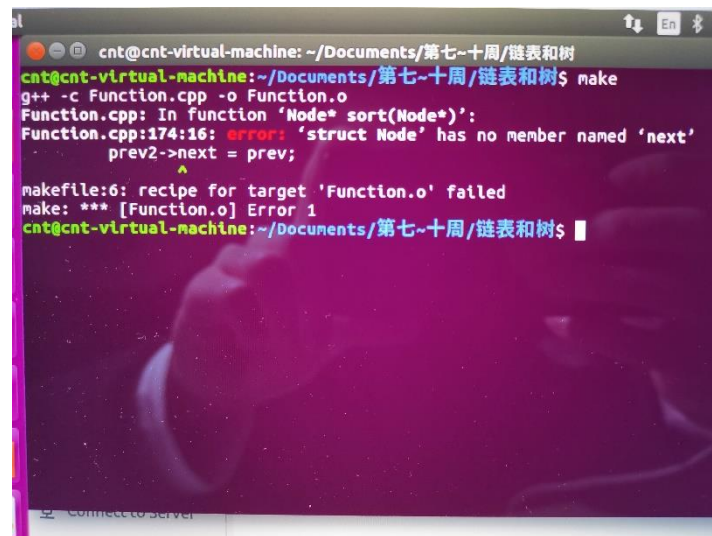


```
cnt@cnt-virtual-machine: ~/Documents/第七~十周/链表和树
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$ make
g++ -c Function.cpp -o Function.o
Function.cpp: In function 'Node* list_specific_insert(Node*, int, int)':
Function.cpp:67:16: error: expected primary-expression before '<' token
    fakehead-<link = p;
               ^
Function.cpp:67:17: error: 'link' was not declared in this scope
    fakehead-<link = p;
               ^
Function.cpp: In function 'void reverse_print_list(Node*)':
Function.cpp:105:6: error: 'x' was not declared in this scope
    if(x == 0)
       ^
Function.cpp: In function 'Node* del_specific_var(Node*, int)':
Function.cpp:123:3: error: expected ';' before 'else'
    else
    ^
Function.cpp:130:25: error: 'var' was not declared in this scope
    if(fakehead->link-var == del_var)
                        ^
Function.cpp:135:7: error: expected ';' before '}' token
    }
    ^
Function.cpp: In function 'Node* sort(Node*)':
```



```
Function.cpp:135:7: error: expected ';' before '}' token
    }
    ^
Function.cpp: In function 'Node* sort(Node*)':
Function.cpp:165:9: error: 'Node' was not declared in this scope
    Node* prev2 = list_head;
    ^
Function.cpp:165:15: error: 'prev2' was not declared in this scope
    Node* prev2 = list_head;
    ^
Function.cpp:173:28: error: 'link' was not declared in this scope
    prev->link = prev2->link;
                        ^
Function.cpp: In function 'void postorder(Tree_Node*)':
Function.cpp:238:32: error: 'struct Tree_Node' has no member named 'r'
    Tree_Node* cur = T2->r;
                           ^
Function.cpp:239:15: error: 'struct Tree_Node' has no member named 'r'
    T2->r = NULL;
    ^
makefile:6: recipe for target 'Function.o' failed
make: *** [Function.o] Error 1
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$
```

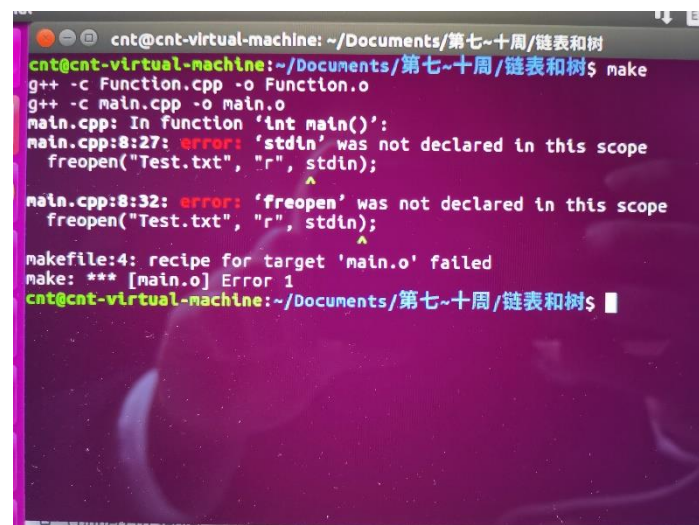
由于没有在 VS 下自动显示语法错误的功能，常常会出现符号打错、字母大小写颠倒、漏打字符等错误。再加上被限制的变量名与自己之前的代码中变量名不同，时常有打错的情况。



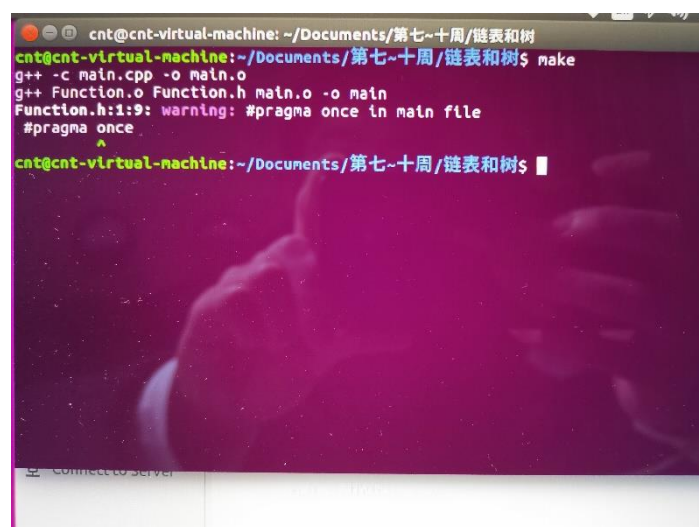
```
cnt@cnt-virtual-machine: ~/Documents/第七~十周/链表和树
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$ make
g++ -c Function.cpp -o Function.o
Function.cpp: In function 'Node* sort(Node*)':
Function.cpp:174:16: error: 'struct Node' has no member named 'next'
    prev2->next = prev;
                   ^
makefile:6: recipe for target 'Function.o' failed
make: *** [Function.o] Error 1
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$
```

2、代码移植

在 VS 下特有的 `_s` (如: `freopen_s`) 和仅有部分编译器支持的 `#pragma once` 在 Linux 内不支持，需要换成 `freopen` 和 `#ifndef`, `#define`, `#endif`。



```
cnt@cnt-virtual-machine: ~/Documents/第七~十周/链表和树
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$ make
g++ -c Function.cpp -o Function.o
g++ -c main.cpp -o main.o
main.cpp: In function 'int main()':
main.cpp:8:27: error: 'stdin' was not declared in this scope
    freopen("Test.txt", "r", stdin);
                          ^
main.cpp:8:32: error: 'freopen' was not declared in this scope
    freopen("Test.txt", "r", stdin);
                          ^
makefile:4: recipe for target 'main.o' failed
make: *** [main.o] Error 1
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$
```



```
cnt@cnt-virtual-machine: ~/Documents/第七~十周/链表和树
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$ make
g++ -c main.cpp -o main.o
g++ Function.o Function.h main.o -o main
Function.h:1:9: warning: #pragma once in main file
#pragma once
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$
```

3、gdb

当编译成功后，运行时却发现链表的输出结果不正确。

```
cnt@cnt-virtual-machine: ~/Documents/第七~十周/链表和树
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$ make
g++ -c Function.cpp -o Function.o
g++ Function.o Function.h main.o -o main
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$ ./main
1 4 5 6
8 1 4 5 6
9
9
9
9
----
9
----
5 8 9 6 1
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$
```

从上图可以发现，在第一次的输出前就出现了问题，于是使用 gdb 设置了断点。因为发现 gdb 不像 VS 里一样会自动显示相关变量的值，以及是否变化，因此我决定观察一个关键的数据来判断 bug 的区间。于是使用 display 监控头节点数据 head->var 的变化。

```
cnt@cnt-virtual-machine: ~/Documents/第七~十周/链表和树
Breakpoint 1, main () at main.cpp:10
10      Node* head = NULL;
(gdb) n
11      head=list_head_insert(head, 1);
(gdb)
12      head = list_head_insert(head, 2);
(gdb) display head->var
1: head->var = 1
(gdb) n
13      head = list_head_insert(head, 2);
1: head->var = 2
(gdb)
14      list_tail_insert(head, 5);
1: head->var = 2
(gdb)
15      list_tail_insert(head, 6);
1: head->var = 2
(gdb)
16      head = list_specific_insert(head, 4, 4);
1: head->var = 2
(gdb)
17      print_list(head);
1: head->var = 1
(gdb)

22      head = list_specific_insert(head, 8, 9);
1: head->var = 8
(gdb)
23      print_list(head);
1: head->var = 9
(gdb)
```

在上面的调试中，可以发现，头节点数据 head->var 在两次的 list_specific_insert 函数后，都发生了不该出现的变化。于是将 bug 锁定在了 list_specific_insert 这个函数内。

最后经过简单的检查，发现自己最后忘记 return list_head 了。

经过修改后，发现大部分输出已经恢复正常，但是发现 reverse_print_list 函数输出发生问题。

```
cnt@cnt-virtual-machine: ~/Documents/第七~十周/链表和树
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$ make
g++ -c Function.cpp -o Function.o
g++ -c main.cpp -o main.o
g++ Function.o Function.h main.o -o main
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$ ./
bash: ./: Is a directory
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$ ./main
2 2 1 4 5 6

8 2 2 1 4 5 6

8 2 2 1 4 5 6 9

1 2 2 4 5 6 8 9

9

1 3 3 4 5 6 8 9
----
1 4 5 6 8 9
----
5 8 9 6 1

cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$
```

于是锁定 bug 出在 reverse_print_list 函数中。

经过简单的查找，发现输出的指令有问题，导致除最后一个节点外没有输出。

后来还发现了几个 bug，如：

发现 T2 中的 2 打漏了，又恰巧有 T 这个变量，所以没有被编译器检查出来。

```
{
    Tree_Node* T2 = T;
    while((x < T2->var && T2->lchild) || (x > T->var && T2->rchild))
    {
        if(x < T2->var)
            T2 = T2->lchild;
        else
            T2 = T2->rchild;
    }
}
```

182,1 79%

经过修改后，最后成功输出。

```
cnt@cnt-virtual-machine: ~/Documents/第七~十周/链表和树
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$ make
g++ -c Function.cpp -o Function.o
g++ -c main.cpp -o main.o
g++ Function.o Function.h main.o -o main
cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$ ./main
2 2 1 4 5 6

8 2 2 1 4 5 6

8 2 2 1 4 5 6 9

1 2 2 4 5 6 8 9

9 8 6 5 4 2 2 1

1 3 3 4 5 6 8 9
----
1 4 5 6 8 9
----
5 8 9 6 1

cnt@cnt-virtual-machine:~/Documents/第七~十周/链表和树$
```