

**SCHOOL OF DIGITAL MEDIA AND INFOCOMM TECHNOLOGY
DIPLOMA IN INFOCOMM SECURITY MANAGEMENT**

Year 2

ST2614 PROGRAMMING USING PERL AND C

~ ASSIGNMENT 2 ~

Learning Outcomes

By completing this assignment, you will be able to

- Design the program flow and function prototypes
- Implement a sorting algorithm of your choice, or use the quicksort library function
- Design struct to contain required fields
- Experience linked-list implementation, including adding nodes and traversal
- Handle file input and output operations
- Process formatted input and output data

Instructions

1. This is a group assignment. The acceptable group size is three or two.

State the class, names and student IDs of your group members as comments at the beginning of your source codes.

2. Submit your source codes and a report of up to 4 pages via blackboard by **4th Aug 2013 11:59PM**.

5 marks will be deducted for each working day of overdue. Submissions will NOT be accepted for marking after 5 working days.

Outline the key ideas and give reasons of your program design (e.g. functions, data structure, and algorithm), describe any extra features your program has in the report.

3. You need to demonstrate your program and will be asked to explain your program design and code implementation. Your program may be tested with fresh input data.

Your source codes must be able to be compiled in the **Cygwin** environment.

4. To ensure a higher degree of readability, you may need to add in relevant inline comments to explain your program (usage, logic, assumptions, etc).

5. If you are applying some codes / techniques that you have learned from external sources, you must state the references clearly in your source codes, e.g. a C sorting routine from a web page, a string manipulation algorithm from a text book.

Plagiarism Warning Statement

Warning: Plagiarism means passing off as one's own the ideas, works, writings, etc., which belong to another person. In accordance with this definition, you are committing plagiarism if you copy the work of another person and turning it in as your own, even if you would have the permission of that person. Plagiarism is a serious offence and disciplinary action will be taken against you. If you are guilty of plagiarism, you may fail all modules in the semester, or even be liable for expulsion.

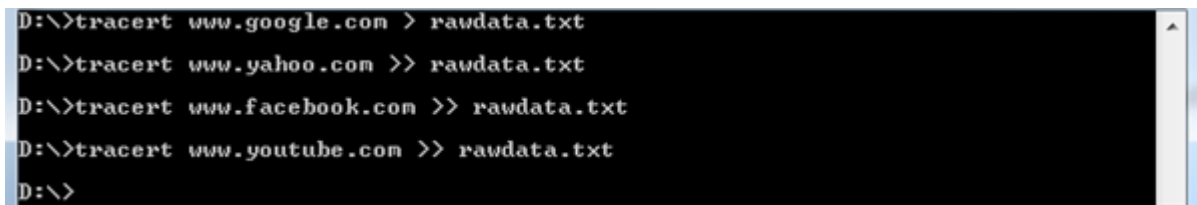
Requirements

In this assignment you will implement a C program that processes several pieces of trace route data and combines them into the source data for querying the network topology.

Program Input

tracert (Windows) / **traceroute** (Linux) determines the path taken to a destination by sending Internet Control Message Protocol (ICMP) Echo Request messages to the destination with incrementally increasing Time to Live (TTL) field values. The path displayed is the list of routers in the path between a source host and a destination. [1]

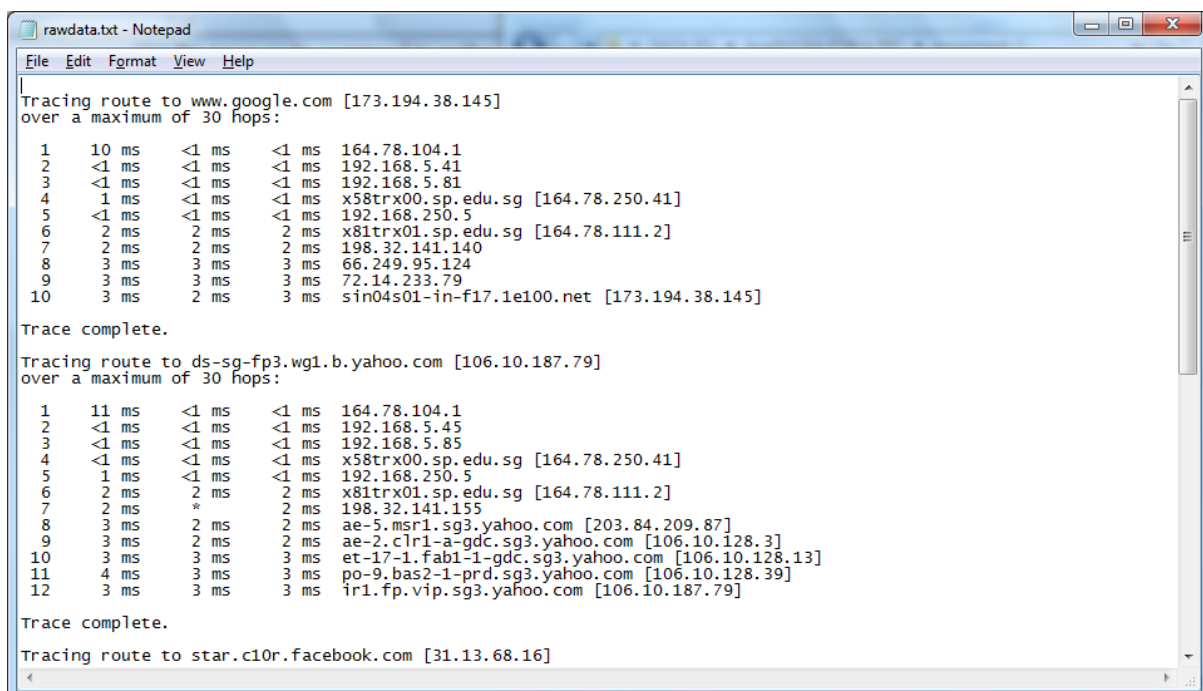
Use the trace route command to gather raw data on network paths to **FOUR** popular websites. Store the raw data into one file.



```
D:\>tracert www.google.com > rawdata.txt
D:\>tracert www.yahoo.com >> rawdata.txt
D:\>tracert www.facebook.com >> rawdata.txt
D:\>tracert www.youtube.com >> rawdata.txt
D:\>
```

Figure 1 trace route command

Sample rawdata.txt is partially shown in Figure 2.



```
rawdata.txt - Notepad
File Edit Format View Help

Tracing route to www.google.com [173.194.38.145]
over a maximum of 30 hops:
  1  10 ms  <1 ms  <1 ms  164.78.104.1
  2  <1 ms  <1 ms  <1 ms  192.168.5.41
  3  <1 ms  <1 ms  <1 ms  192.168.5.81
  4  1 ms   <1 ms  <1 ms  x58trx00.sp.edu.sg [164.78.250.41]
  5  <1 ms  <1 ms  <1 ms  192.168.250.5
  6  2 ms   2 ms   2 ms   x81trx01.sp.edu.sg [164.78.111.2]
  7  2 ms   2 ms   2 ms   198.32.141.140
  8  3 ms   3 ms   3 ms   66.249.95.124
  9  3 ms   3 ms   3 ms   72.14.233.79
 10  3 ms   2 ms   3 ms   sin04s01-in-f17.1e100.net [173.194.38.145]

Trace complete.

Tracing route to ds-sg-fp3.wg1.b.yahoo.com [106.10.187.79]
over a maximum of 30 hops:
  1  11 ms  <1 ms  <1 ms  164.78.104.1
  2  <1 ms  <1 ms  <1 ms  192.168.5.45
  3  <1 ms  <1 ms  <1 ms  192.168.5.85
  4  <1 ms  <1 ms  <1 ms  x58trx00.sp.edu.sg [164.78.250.41]
  5  1 ms   <1 ms  <1 ms  192.168.250.5
  6  2 ms   2 ms   2 ms   x81trx01.sp.edu.sg [164.78.111.2]
  7  2 ms   *      2 ms   198.32.141.155
  8  3 ms   2 ms   2 ms   ae-5.msr1.sg3.yahoo.com [203.84.209.87]
  9  3 ms   2 ms   2 ms   ae-2.clr1-a-gdc.sg3.yahoo.com [106.10.128.3]
 10  3 ms   3 ms   3 ms   et-17-1.fab1-1-gdc.sg3.yahoo.com [106.10.128.13]
 11  4 ms   3 ms   3 ms   po-9.bas2-1-prd.sg3.yahoo.com [106.10.128.39]
 12  3 ms   3 ms   3 ms   ir1.fp.vip.sg3.yahoo.com [106.10.187.79]

Trace complete.

Tracing route to star.c10r.facebook.com [31.13.68.16]
```

Figure 2 trace route data

The file containing the raw data from trace route will be the input to your program.

Program Output

Your program reads in the formatted data from the raw data file, and constructs internally the network topology. **Network topology** is the arrangement of links and nodes of a network. [1]

(I) Display the list of network nodes

Your program displays a sorted list of the network nodes. The nodes should be sorted in ascending order by IP addresses. Format of the output is

ID: IP <-- ID of last hops

```
$ ./topo rawdata.txt topology.txt
27: 31. 13. 28.121 <-- 26
28: 31. 13. 68. 16 <-- 27
8: 66.249. 95.124 <-- 7
9: 72. 14.233. 79 <-- 8
15: 106. 10.128. 3 <-- 14
16: 106. 10.128. 13 <-- 15
17: 106. 10.128. 39 <-- 16
18: 106. 10.187. 79 <-- 17
1: 164. 78.104. 1 <--
6: 164. 78.111. 2 <-- 5
4: 164. 78.250. 41 <-- 3, 12
20: 165. 21. 12. 68 <-- 19
19: 165. 21.240.117 <-- 4
29: 173.194. 38.132 <-- 9
10: 173.194. 38.145 <-- 9
2: 192.168. 5. 41 <-- 1
11: 192.168. 5. 45 <-- 1
3: 192.168. 5. 81 <-- 2
12: 192.168. 5. 85 <-- 11
5: 192.168.250. 5 <-- 4
7: 198. 32.141.140 <-- 6
13: 198. 32.141.155 <-- 6
14: 203. 84.209. 87 <-- 13
23: 203.208.151.253 <-- 22
24: 203.208.152. 74 <-- 23
22: 203.208.152.161 <-- 21
25: 203.208.153. 19 <-- 24
26: 203.208.153.193 <-- 25
21: 203.208.190. 21 <-- 20
```

Figure 3 Sample output of sorted node list

The first column is the unique node ID that your program assigns to identify each node, the second column shows the IP address of the node, and the last column shows the link information, i.e. the immediate last hops linking to this node. Please pay attention to the output alignment.

Note that a network node should not be duplicated even if it exists on paths to different webservers. A node may have more than one last hop. It may also have more than one next hop.

(II) Print menu for user query

Your program then prints the below menu for user to query on the network topology:

“Please select an option from the menu

- 1 Report ID of previous hops of a network node
- 2 Identify the next hops of a network node
- 3 Quit”

If user chooses option 1, prompt the user to enter the ID of the network node to query on, and respond with the **IDs** of the immediate previous hops of this network node. Reprint the menu afterwards.

If user chooses option 2, prompt the user to enter the ID of the network node to query on, and respond with the **ID and IP addresses** of the next hops of the network node. Reprint the menu afterwards.

If user chooses option 3, program terminates.

You are free to design the data structures for modelling the network nodes and their linkage information, e.g. linked-list, array. However, your program **MUST** contain a part that demonstrates the usage of linked list.

Sample Data Files

Sample input, output files

rawdata.txt, nodelist.txt

are available for download from Blackboard. Submit your program after you have tested it with rawdata.txt

Bonus Parts

Get bonus marks if your program has the following features:

- a. It supports user query by node IP.
- b. It handles a large number of websites, where the number is unknown in advance.
- c. It has an extra menu option:
 “4 List network nodes at n hops away”
 If this option is chosen, prompt the user to provide a number n, and display all network nodes at n hops away.

Marking Scheme

Category	Weight
Syntax and Functionality	60 %
Program Design	25 %
Good Programming Practices	10 %
Team contribution	5 %

References

[1] Windows command line reference.
<http://www.microsoft.com/resources/documentation/windows>

~The End~