Assignment 1 (CS558)

Due: 11:59pm Feb. 18th (Sunday)

This assignment is done by a group of 2 students.

Goal :

1.Learn how to write and use makefile:
http://www.delorie.com/djgpp/doc/ug/larger/makefiles.html (C)
http://www.cs.swarthmore.edu/~newhall/unixhelp/javamakefiles.html (Java)

2. Implement a simple **iterative** telnet client and a telnet server using TCP sockets. Following commands should be supported: ls, mkdir, rmdir, cd, pwd, and exit.

3. Implement the monoalphabetic cipher.

PART I (Socket Programming)

Implement a simple **iterative** telnet client and a telnet server using TCP sockets. Following commands should be supported: ls, mkdir, rmdir, cd, pwd, and exit.

Specifications

The telnet server is invoked as

telnetserv <*port_number*>

<*port_number*> specifies the port at which the telnet server accepts connection requests.

The telnet client is invoked as

telnetcli <server_machine> <server_port>

<*server_machine*> is the domain name of the server **(bingsuns.binghamton.edu)**. If you use C/C++, your program needs to convert it to corresponding 32 bit IP address using the **gethostbyname().** http://retran.com/beej/gethostbynameman.html

<***server_port***> refers to the port number on which the server listens for connection.

Upon connecting to the server, the client prints telnet >, which allows the user to execute the following commands.

telnet > ls //lists contents of the directory on the server side

telnet > mkdir <dir> //create a directory <**dir**> on the server side

telnet > rmdir <dir> //delete directory <**dir**> on the server side

telnet > cd <dir-relative-path> //change the current working directory to a directory that is relative to the //current directory. E.g. cd path. <dir-relative-path> does not contain "/"

telnet > cd .. //move up one folder

telnet > pwd //print the working directory of the server

telnet > exit // ends the telnet session with the server and exits telnet

Error handling:

1. Invalid commands, i.e., commands other than ls, cd, pwd, mkdir, rmdir, and exit specified above. Note: you should not implement commands that are supported in Linux/Unix, but are not specified in the above. For example, commands "ls –a", "vi" should be considered as invalid commands.
2. telnet > mkdir <dir>: <dir> exists
3. telnet > rmdir <dir>: <dir> does not exist
4. telnet > cd <dir>: <dir> does not exist
   Note: your implementation should NOT support absolute path in cd command.

Note:

1. If you use C, you can use cli.c and ser.c provided on mycourses for this assignment. If you use Java, you can use TCPServer.java and TCPClient.java provided on mycourses for this assignment.

2. Since multiple teams will be testing their servers, it is possible that multiple teams end up using the same port number. To prevent this, use a unique port number such as last 4 digits of your ID.

PART II (Monoalphabetic Cipher)

Implement the monoalphabetic cipher to encrypt/decrypt files containing only lower-case letters a-z. **The file does not contain spaces, tabs, and newlines.** You can assume that the file contains less than 10000 characters. For each plaintext letter, your program should randomly generate a corresponding ciphertext letter based on a seed. Different plaintext letters should be mapped to different ciphertext letters. Different mappings will be generated with different seed. Your code should result in an executable of the following form:

./mono <inputfile> <outputfile> <seed> 1/0
- inputfile: input file name
- outputfile: output file name
- seed: the seed used to generate the mapping
- 1/0: encryption/decryption

E.g. mono in out 1: encrypt file in and store the result in file out:

mono out in1 0: decrypt file out and store the result in file in1

Your program will print the mappings generated using the following format, E.g.

a-b, b-d, e-h, ……

Submission guideline

- **Only one** of the group members should submit the assignment

- You need to hand in your **source code** and **two Makefiles (one for Part I and one for Part II)** electronically (please **do not submit .o or executable code**). Your code should compile and run correctly on bingsuns.binghamton.edu. In PART I, the Makefile **must** give the executable server code the name **telnetserv** and the executable client code the name **telnetcli**. In PART II, the Makefile must give the executable code the name **mono**.
-
- Submit one **README** file (text file, do not submit a .doc file) which contains
  - You name and email address.
  - Whether your code was tested on bingsuns.
  - How to execute your program.
  - (Optional) Briefly describe your algorithm or anything special about your submission that the TA should take note of.

- Create two folders: part1 (containing the source code of PART I and a Makefile for compiling the code) and part2 (containing the source code of PART II and a Makefile for compiling the code). Place both folders under one directory with a unique name (such as p1-[userid] for assignment 1, e.g. p1-pyang).

- Tar the contents of the above directory using the following command.
  tar –cvf [directory_name].tar [directory_name]
  E.g. tar -cvf p1-pyang.tar p1-pyang/

  Submit the tared file created through mycourses.binghamton.edu.

Grading guideline

- PART I: 50'
  - ls, exit : 12’
  - cd, mkdir, rmdir, pwd : 26’
  - error handling : 12’
- PART II: 38’
- Readme, correct format of execution: 4’
- Makefile: 8’

Academic Honesty

All students should follow Student Academic Honesty Code (**if you have not already read it, please read it carefully**). All forms of cheating will be treated with utmost seriousness. You may discuss the problems with other students, however, you must write your OWN codes and solutions. Discussing algorithms and code is NOT acceptable. Copying an assignment from another student or allowing another student to copy your work may lead to the following:

- **Report to the department and school**
- **0 in the assignment or F in this course.**

Moss will be used to detect plagiarism in programming assignments. You need ensure that your code and documentation are protected and not accessible to other students. Use **chmod 700** command to change the permissions of your working directories before you start working on the assignments. If you have any questions about whether an act of collaboration may be treated as academic dishonesty, please consult the instructor before you collaborate.