Lab Assignment 2

(Due: April 2 (Mon) 3:30 pm, 2018)

1. Goal

Learn how to develop an application protocol as well as build an application over UNIX (Linux) socket.

2. Project Description: Build an on-line personal calendar

In this programming assignment, you are going to build a simple on-line personal calendar. You need to implement the protocol, and write a client and a server that communicate using your protocol through TCP sockets. Your calendar server should be a **concurrent application server** that handles different calendar clients at the same time. Your calendar protocol should be able to allow the following functionalities:

- Add a user
- Delete a user
- Modify user information: Name, password, contact phone number, and email address
- User login authentication with user name and password
- Add a new appointment
- Remove an appointment
- Update an existing appointment
- Display a user's the appointment for a specific time or a time range [e.g., year, month, week, day, and hours].
- Help : display syntax of each commend
- Proper exception handling
  - Check a time conflict (duplicated appointment) when a new schedule is added.
  - Check duplicated users.
  - Check incomplete schedule.
  - Check invalid time and character.
- NOTE:
  - Each user name is one single word (only alpha numeric, hyphen, only dash)
  - Each user password should be longer than 8 and shorter than 16 alpha-numeric character)
  - Each user password has to be encrypted.
  - Each appointment has following attribute:
    - Appointment begin Format – HH:MM MM/DD/YEAR
      - e. g.: 15:20 03/14/2018 (24 hour format)
    - Appointment end Format– HH:MM MM/DD/YEAR
      - e.g.: 14:00 03/14/2018 (24 hour format)
    - Appointment place (or room) – Ascii text upto 64 characters

        - Appointment contents – Ascii text upto 128 characters
    - Assume your calendar server does not allow multiple logins with same user id (user name) at simultaneously.
    - Assume your calendar

You can use different techniques to store and manage your schedule at the server side: using simple text files, or databases, etc. It is up to you to choose the appropriate way to implement your personal schedule manager protocol.

2.1 Programming environment

- All programs have to be written C or C++ and run on UNIX like platform.
    - **<u>Please use CSEgrid machines</u>**
- All connections between a server and clients should be TCP/IP socket.
- A server and client should run on different machines.

2.2 Required Skills

Everyone is expected to know following skills and knowledge in order to complete this programming assignment.

- TCP/IP Socket programming
- Understanding UNIX like Operating System
- Creating/invoking processes in UNIX like environment
- Concurrent programming (fork() or multithread)
- Makefile
- C or C++

3. Deliverables

The deliverables for this assignment include the following files:

- Design document [in a word file] – describe your program
  ①. Describe overall software architecture of your program
  ②. Describe main modules and data structures
- Written Code for Server and Client program
- Makefile
- Readme: A short description of your programs includes: the names of created executable images that will be created after run your makefile, how to run your programs. In addition, you should specify your execution environments such as type of OS with version number, compiler, etc.

4. Submission

Please do the followings when you submit your programming assignment.

- Create a tar file that contains your design document, written source code, makefile and readme. <u>DO NOT INCLUDE EXECUTABLES AND OBJECT FILES.</u>
- Please use the following convention when you create a tar file
  - First 3 letters of your last name + last 4 digits of your student ID
  - e.g.: If a student name is "Bill Clinton" and his ID is 999-34-5678, then his tar file name is "cli5678.tar".
- Once you create the tar file, and compress the tar file using 'gzip' or "compress".
- If the compressed tar file is ready, **<u>Upload it to Class Canvas by 3:30 pm April 3 (Mon), 2017.</u>**

5. Grading

The maximum point for the assignment is 40. This programming assignment will be graded by following criteria:

1. Completeness: Max 30
   (a) If your server and clients are connected via TCP and then,
   - If a sever support all required functions: 13
   - If a client supports all required functions: 12
   - If only a sever program is working partially: 5
   - If only a client program is working partially: 5
   - Proper error handling: 3
     - Handling wrong input
     - Providing usage message
   - Display appropriate result messages with respect to a user's input: 2

   (b) If a server and clients do not work over TCP, but they work within machine: 10

   (c) If either server or clients doesn't work at all: 0

2. Quality of Design Document : 5
3. Deliverables: 5
   - If you submit all deliverables as I suggested in section 4: 5
   - Otherwise: 0