COMPUTER SCIENCES AND SOFTWARE ENGINEERING
AUBURN UNVERSITY

COMP 2710
Software Construction

Fall 2015

Lab 2
Distributed WeagleBook

Due: October 14, 2015

Points Possible: 100
Due: Written Portion: October 7th, 2015 by 11:55 pm. Submission via Canvas (25 points)
Program: October 14th, 2015 by 11:55 pm via Canvas (75 points)

No late assignments will be accepted.
No collaboration between students. Students should NOT share any project code with each other. Collaborations in any form will be treated as a serious violation of the University's academic integrity code.

Goals:

- To perform Object-Oriented Analysis, Design, and Testing
- To develop a non-trivial application using classes and constructors
- To learn distributed communication methods and develop a simple but useful distributed WeagleBook system
- To learn to use file I/O and take advantage of the Network File System (NFS)

Process – 25 points:
Create a text, doc, or .pdf file named "<username>_2p" (for example, mine might read "lim_2p.txt") and provide each of the following. Please submit a text file, a .doc file or .pdf file (if you want to use diagrams or pictures, use .doc or .pdf). You are free to use tools like Visio to help you, but the final output needs to be .txt, .doc, or .pdf.

1. **Analysis:** Prepare use cases. Remember, these use cases describe how the user interacts with a messaging system (what they do, what the systems does in response, etc.). Your use cases should have enough basic details such that someone unfamiliar with the system can have an understanding of what is happening in the messaging system interface. They should not include internal technical details that the user is not (and should not) be aware of.

2. **Design**:
   a. Create a Class Diagram (as in Lab 1). Be sure to include:
      1) The name and purpose of the classes
      2) The member variables and the functions of the class
      3) Show the interactions between classes (for example, ownership or dependency)
      4) Any relevant notes that don't fit into the previous categories can be added
   b. Create the data flow diagrams. Show all the entities and processes that comprise the overall system and show how information flows from and to each process.

3. **Testing**: Develop lists of *specific* test cases OR a driver will substitute for this phase:
   1) For the system at large. In other words, describe inputs for "nominal" usage. You may need several scenarios. In addition, suggest scenarios for abnormal usage and show what your program should do (for example, entering a negative number for a menu might ask the user to try again).
   2) For each object. (Later, these tests can be automated easily using a simple driver function in the object)

4. Implement the Distributed WeagleBook System

5. **Test Results**: ***After developing the Distributed WeagleBook System***, actually try all of your test cases (both system and unit

testing). Actually show the results of your testing (a copy and paste from your program output is fine – don't stress too much about formatting as long as the results are clear). You should have test results for every test case you described. If your system doesn't behave as intended, you should note this. Note: Driver output will substitute for this phase.

Program Portion:

The proliferation of Internet applications in smartphones, laptops, ipads, and desktops shows the importance of communications among computing devices. One of the key functionalities is the ability to *send and receive messages from one computer to another over the network*. It is interesting to learn how a simple messaging system can provide this functionality. In this Lab 2, you will analyze, design, implement and test a simple distributed WeagleBook system. It will have a simple text-based user interface that allows multiple users to send and receive messages to each other. In this lab project all the users must be implemented in *distributed address space*, i.e. *each user must execute on a different computer and message of one user can be read by another through shared common files*. The *common files are shared through the Network File Server (NFS).*

What is a Distributed WeagleBook Message System?

You probably have used a distributed messaging system over the Internet. Although there are many different types of messaging systems, you will implement the Distributed WeagleBook System that *allows users to post and read messages from another user executing on a different computer*. A user U can read the messages from another user *V* in two ways. The first method is by having U to add V as his friend and the system will retrieve messages posted by *V*. The second method is for V to tweet a message which will then be seen by all users, including U. For the purpose of this lab project, the Distributed WeagleBook System is a program (or collection of programs) that does the following:

1) There are many *distributed and concurrent* users that use the WeagleBook for posting their own messages and sharing them with their friends, i.e. all users may use the system simultaneously on different computers to post and read messages. In this assignment you must implement **at least four users**.
2) Each user can ***posts*** their message in their own **wall page**. *The program must allow multiple lines of a message to be posted. To end a message, the user will enter a new line with a string "!!"followed by the enter key. The final string "!!"must not be stored in the message buffer, neither should it be displayed in the wall or home pages.* The messages of each user are stored in a file with the user's name, e.g. Jane's wall page messages will be stored in the file called "*Jane"*. Since the file is maintained by NFS, another user Bill will also see the same file "*Jane"*, although it is executing on a different computer. The Distributed WeagleBook System can then read the messages posted by Jane if Jane is Bill's friend.
3) Each user can ***tweet*** a message which will be seen by all users. *The program must allow multiple lines of a message to be tweeted. To end a message, the user will enter a new line with a string "!!"followed by the enter key. The final string "!!"must not be stored in the message buffer, neither should it be displayed in the wall or home pages.* Messages tweeted by any user may be stored in a file called "*tweet*". The format of the tweet messages will contain the user's name, i.e. if Bill tweets a message "`Greetings, everybody!`", then the message is stored in the file "tweet" as "{<164>}{[Bill]}`Greetings, everybody!`", where 164 is the timestamp. All users can then read the file "tweet" and display the user name and tweeted messages in reverse chronological order together with other messages.
4) Each user has a **friend list** which permits them to read the posted messages from the wall page file of each friend in the list. The user must be allowed to enter any name in the friend list only if the name is a valid user.
5) Each user has a home page. The **home page** of a user displays all the messages of all their friends, *all tweeted messages from all users* and all their own messages in reverse chronological order in which they are created, i.e. the most recent messages are displayed before the less recent messages.
6) In the home or wall pages, the program will first display only the two most recent messages from each friend, tweeted messages and messages from themselves, and prompt if the user wants to display more messages. If the response is "yes", then all messages from themselves, tweeted messages and all their friends' messages will be displayed, otherwise, the program will stop the display.
7) Since the users are each running on different computer, you must keep track of the time ordering of the messages. There are at least two methods for keeping track of the time ordering of the messages. The first method is to use the time() function to obtain the time and storing the time with the message. The second method is to maintain a file called "time", shared by all the users. The file "time" contains an integer value which is incremented by one every time a user read the number to time-stamp their message. For example the number in the time file is initially 0. When Jane posts a message, the system will read the time file and time-stamp Jane's message with 0. It then increments the time value and writes back 1 into the time file, over-writing the previous 0 value.

8) Each message stored in the wall page file of a user must NOT contain the name of the user, only the text message and the time-stamp. The name of the user is identified by the filename.

Message format

For this assignment, the messages must be stored in the file for each user using the following format. There must be exactly *one message file* for each user. In each of the user message file, the messages must be stored in reverse chronological order from which they were created. Each message must contain the timestamp and the user message using the following format "{<Timestamp>} message text". The message must *not* contain the user's name. However, each message must be of variable length. Different lines of a message must be separated by a '`\n`' character, i.e. each line of the message must be terminated by a newline character. For examples, if `John` first entered the message "`Welcome to the group!\nGreat to hear from you all!`" And later `John` entered "`I'll update you all soon`" then John's message file must contain "`{<253>}I'll update you all soon{<216>}Welcome to the group!\nGreat to hear from you all!`". The string "`{<253>}`" is the timestamp enclosed in the "`{<`" and "`>}`" strings. Assume that messages will not contain these markers. The timestamp allows multiple messages sent by different users to be ordered correctly in reverse chronological order from which they were posted. The purpose for this standard message format is that everyone's wall message page should end up being compatible. That is, every user should be able to work with every friend's wall messages.

Tweeted messages must use the following format "{<Timestamp>}{[Username]} message text". For example, if Bill tweets a message "`Greetings, everybody!`", then the message is stored in the file "tweet" as "{<164>}{[Bill]}`Greetings, everybody!`", where 164 is the timestamp.

Your program must meet this requirement for message format, otherwise significant points will be deducted.

The user interface

Write a menu-based and text-based user interface for interacting with the Distributed WeagleBook System. When the WeagleBook system is first started, it will first display the WeagleBook banner and *prompts for the user's name*. After the user's name is entered, the system will print a banner welcoming the user.

```
        ============================================================
        |            Distributed WeagleBook System!                |
        ============================================================

Please enter user name: Bill

        ============================================================
        |      Welcome to Distributed WeagleBook System, Bill!     |
        ============================================================
```

The program will then make Bill the current user. If the file "Bill" exists, it will then read and update that file. If the file "Bill" *does not* exist, then it will first create the file "Bill" and then update and read "Bill" as the user posts messages and reads home or wall pages.

The main menu has (at least) 6 options to choose from as shown below. *Abbreviate the menu so that all the 6 options can be shown in only one or two lines.* The user interface will accept the option and all related inputs from the user, perform the operation and then repeatedly shows this menu of options again and prompts for the next option.

- **Add a friend (f)**: When the user enters option 'f', the program will then prompt for the name of the friend. It then checks if the friend name is a currently valid user by checking if the wall page file with that user name exists. If so, it adds the friend in the current user's friend list. If not, it'll display an error message.
- **Post a message (p)**: When the user enters option 'p', the program will prompt for the message and stores the message in the user's wall page file with the user's name. *Your program must allow multiple lines of messages. To end a message, the user will enter a new line and a string "!!"followed by hitting the enter key. The final string "!!" must not be stored in the message buffer; neither should it be displayed in the wall or home pages.*
- **Tweet a message (t)**: When the user enters option 't', the program will prompt for the message and stores the message in the file "tweet". As in posted messages, *your program must allow multiple lines of messages. To end a message, the user will enter a new line and a string "!!" followed by hitting the enter key.*

- **Display wall page (w)**: When the user enters option 'w', the program will first display a title indicating that it is displaying the current user's wall page, e.g. "Jane's Wall Page". It then displays the two latest messages in the current user's wall page *and tweeted messages from that user only*, in reverse chronological order. When displaying the wall page messages, it must *NOT* display the user's name. After display the two latest messages (including tweet messages), it will then prompt the user if they want more messages. If the response is "no", then it will stop displaying messages, but if the response is "yes", it will display all the remaining messages from that user. If there are two or fewer messages then the program must not prompt for more messages.
- **Display home page (h)**: When the user enters option 'h', the program will first display the current user home page title, e.g. "Joe's Home Page", and then it displays only the two latest messages either from the current user's wall page, wall pages of all the friends of the current user, *or tweeted messages from all users*, whichever are the two latest messages. When displaying the message from each user, it must display the user name following by a ':' and a newline and then followed by the message. It will then prompt the user if they want more. If the response is "no", then it will stop displaying messages, but if the response is "yes", it will display all the remaining messages from that user, wall pages of all the friends of the current user, and all tweeted messages. If there are two or fewer messages then the program must not prompt for more messages. In both the cases above, the messages are all displayed in reverse chronological order, i.e. the most recently posted messages will be displayed before the earlier messages.
- **Quit WeagleBook System (q)**: When the user enters the option 'q', the program will exit normally.

The name of your program must be called <username>_2.cpp (for example, mine would read "lim_2.cpp"

Use comments to provide a heading at the top of your code containing your name, Auburn Userid, and filename. You will lose points if you do not use the specific program file name, or do not have a comment block on **EVERY** program you hand in.

Your program's output need not exactly match the style of the sample output (see the end of this file for one example of sample output).

Important Notes:
You must use an object-oriented programming strategy in order to design and implement this distributed messaging system (in other words, you will need write class definitions and use those classes, you can't just throw everything in main()). A well-done implementation will produce a number of robust classes, many of which may be useful for future programs in this course and beyond. Some of the classes in your previous lab project may also be re-used here, possibly with some modifications. Remember good design practices discussed in class:

a) A class should do one thing, and do it well
b) Classes should NOT be highly coupled
c) Classes should be highly cohesive

- You should follow standard commenting guidelines.

- You DO NOT need any graphical user interface for this simple, text-based application. If you want to implement a visualization of some sort, then that is extra credit.

Error-Checking:

You should provide enough error-checking that a moderately informed user will not crash your program. This should be discovered through your unit-testing. Your prompts should still inform the user of what is expected of them, even if you have error-checking in place.

Please submit your program through the Canvas system online. If for some disastrous reason Canvas goes down, instead e-mail your submission to the TA – Liang Tang – at lzt0006@tigermail.auburn.edu Canvas going down is not an excuse for turning in your work late.

You should submit the two files in digital format. No hardcopy is required for the final submission:

<username>_2.cpp
<username>_2p.txt (script of sample normal execution and a script of the results of testing)

Sample Usage:

The following is executed in tux187:

> lim_2

```
          ============================================================
          |              Distributed WeagleBook System!              |
          ============================================================

          Please enter user name: Bill

          ============================================================
          |     Welcome to Distributed WeagleBook System, Bill!      |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: p

Enter message: Hello there, everyone!
Welcome to our new chat group.
Feel free to post messages any time.
!!

          ============================================================
          |                     New message added                    |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: t

Enter message: Amazing WeagleBook!
Let's get this out quick.
!!

          ============================================================
          |                    New message tweeted                   |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: p

Enter message: Taking a slow cruise to the Bahamas, y'll!
Will be back to civilization in 7 days.
Keep you posted soon.
!!

          ============================================================
          |                     New message added                    |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: q

          ============================================================
          |       Thank you for using the WeagleBook System          |
          ============================================================
```

The following is executed in tux192:

> lim_2

```
          ============================================================
          |              Distributed WeagleBook System!              |
          ============================================================

          Please enter user name: Tom

          ============================================================
          |     Welcome to Distributed WeagleBook System, Tom!       |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: f

Please enter friend's name: Bill

          ============================================================
          |            Added Bill to Friend's List                   |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: p

Enter message: Welcome to my new msg page!
I'm new to this, so be gentle, no flame please.
!!

          ============================================================
          |                      New message added                   |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: t

Enter message: Let me be your friend.
It's fun.
!!

          ============================================================
          |                     New message tweeted                  |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: p

Enter message: Hiking the Rockies in 5 days!
Can't wait to head for the mountains.
!!

          ============================================================
          |                      New message added                   |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: w

          ============================================================
          |                       Tom's Wall Page                    |
          ============================================================


Hiking the Rockies in 5 days!
Can't wait to head for the mountains.

Let me be your friend.
It's fun.

                      More message? (yes/no): yes

Welcome to my new msg page!
I'm new to this, so be gentle, no flame please.
```

```
          ============================================================
          |                       End of Tom's Wall Page              |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: q

          ============================================================
          |         Thank you for using the WeagleBook System        |
          ============================================================
```

The following is executed in tux189:

> lim_2

```
          ============================================================
          |              Distributed WeagleBook System!              |
          ============================================================

          Please enter user name: Jane

          ============================================================
          |     Welcome to Distributed WeagleBook System, Jane!      |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: f

Please enter friend's name: Tom

          ============================================================
          |                 Added Tom to Friend's List               |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: p

Enter message: Summer is cool.
Still trying to adjust to the semester drill.
!!
          ============================================================
          |                       New message added                  |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: t

Enter message: Great stuff.
You need to try this.
!!
          ============================================================
          |                       New message tweeted                |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: p

Enter message: We had a great time at the beach!
Not the mention scuba diving and sailing.
!!

          ============================================================
          |                       New message added                  |
          ============================================================
```

```
Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: h

          ============================================================
          |                      Jane's Home Page                    |
          ============================================================

Jane:
We had a great time at the beach!
Not the mention scuba diving and sailing.

Jane:
Great stuff.
You need to try this.

                      More message? (yes/no): yes

Jane:
Summer was cool.
Still trying to adjust to the semester drill.

Tom:
Hiking the Rockies in 5 days!
Can't wait to head for the mountains.

Tom:
Let me be your friend.
It's fun.

Tom:
Welcome to my new msg page!
I'm new to this, so be gentle, no flame please.

Bill:
Amazing WeagleBook!
Let's get this out quick.

          ============================================================
          |                    End of Jane's Home Page                 |
          ============================================================


Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: q

          ============================================================
          |         Thank you for using the WeagleBook System        |
          ============================================================
```

The following is executed in tux194:

> lim_2

```
          ============================================================
          |               Distributed WeagleBook System!             |
          ============================================================

          Please enter user name: Bill

          ============================================================
          |     Welcome to Distributed WeagleBook System, Bill!      |
          ============================================================
Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: f
```

```
Please enter friend's name: Jane

          ============================================================
          |                Added Jane to Friend's List               |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: p

Enter message: I'm back to the daily grind.
What's new here?
!!

          ============================================================
          |                      New message added                     |
          ============================================================


Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: h

All messages (a), Latest messages (l), Enter option: l

          ============================================================
          |                      Bill's Home Page                      |
          ============================================================

Bill:
I'm back to the daily grind.
What's new here?

Jane:
We had a great time at the beach!
Not the mention scuba diving and sailing.

                             More message? (yes/no): yes
Jane:
Great stuff.
You need to try this.
Jane:
Summer was cool.
Still trying to adjust to the semester drill.

Tom:
Let me be your friend.
It's fun.

Bill:
Taking a slow cruise to the Bahamas, y'll!
Will be back to civilization in 7 days.
Keep you posted soon.

Bill:
Amazing WeagleBook!
Let's get this out quick.

Bill:
Hello there, everyone!
Welcome to our new chat group.
Feel free to post messages any time.

          ============================================================
          |                    End of Bill's Home Page                 |
          ============================================================

Add Friend (f), Post (p), Tweet(t), Wall (w), Home (h), Quit (q)
Enter option: q

          ============================================================
          |       Thank you for using the WeagleBook System          |
          ============================================================
```