CS3841 Operating Systems

Lets Chat: *"A lab where two partners can talk to each other."*

Due by 23:00 PM on October 24, 2011

1. Objectives

- Construct code which uses POSIX threads.
- Construct software which protects against race conditions using semaphores and other protection mechanisms.
- Construct code which uses sockets to communicate.

2. Reference Reading

Topic	Book	
getc	C: A Reference Manual	374-375
C input / output	C: A Reference Manual	363-369, 387-400[1]
Ncurses Library	Discovering ncurses, the GUI for the Linux Console http://www.linuxplanet.com/linuxplanet/reviews/6964/1/	
NCurses Library	Writing Programs with NCURSES http://invisible-island.net/ncurses/ncurses-intro.html	
Beep	Man beep	

3. Prelab

Run the following commands from within your virtual machine while connected to the network

1. sudo apt-get install beep
2. beep
3. sudo apt-get install libncurses5-dev
4. man ncurses

These commands will install a few libraries that were not installed when the images were created.

4. Assignment

A chat program used to be the hippest of tools to waste time with. College students galore would spend hours using internet relay chat (IRC) discussing "useful" things. With the invention of texting, that has gone away. But there is still a use. After all, even Facebook has a chat program built into it.

In this lab, you will work with a partner to develop a UNIX command line chat program. The program will allow the user to enter the command line:

Chat <username> <server> <port>

[1] Note: Pay attention to fprintf…

and it will open a socket connection to a machine running on that server and port. If a connection cannot be made, it will wait, trying again at some point in the future.

When a connection is made, the user will be able to chat on a screen resembling that shown below. (Note: This screen would show Paul's screen.) In essence, the text that is entered would go in the bottom middle segment, and the messages that are received and transmitted would go up to, and the bottom would keep track of how long the system has been connected.

<Barb>Hi Paul. How's it going. <Paul>Going great. <Paul>I got the horse right here. His name is Paul Revere and here's a guy who says if the weather's clear <Paul>Can do, can do This guy says the horse can do <Paul>If he says the horse can do, can do, can do <Barb>I'm picking Valentine Cause on the morning line The guy has got him figured at 5 to 9 <Barb>Has chance, has chance <Barb>This guy says the horse has chance
<Paul> No way for Paul Revere I'll bite I hear his foot's all right Of course it all depends If it rained last night
Paul: Connected to Barb for 2:05.

5. Program requirements

1. The program shall allow a user to chat with another user who is using the same client on a second machine.
2. Communications between machines shall be carried out using sockets.
3. Upon reception of a new message, a sound shall be made.(Note: Requirement currently suspected. Beep is not working in the current virtual machines.)
4. The program shall log each message as it comes in a linked list.
5. If the user simply types "savelog", the linked list will be printed out to the file "textlog.txt" and stored.
6. The program shall use multiple threads. At a minimum, there shall be the following threads of control:
    a. One thread for the user to type on the keyboard (hint: keys will be read individually using getc / getchar
    b. One thread which is monitoring the socket connection from the remote machine and adds new messages to the display as they come in
    c. One thread which sends a message when the user finishes typing a line and presses return
    d. One thread which keeps track of how long the user has been online, ticking once a second
    e. One thread which manages the screen.
7. When necessary, the program shall use mutexes or semaphores to protect critical sections.
8. The program shall use the ncurses5 library (or other alternative) to create the screen.

9. Your program must do something else beyond these which is creative.

6. Lab Deliverables

6.1. Interim deliverable (Due in lab on October 12, 2011)

1. Handrawn UML diagram showing the basic classes which are part of your project.
2. Explanation of how the UML will be divided amongst lab partners

6.2. Final deliverable

1. Tar file including all of your source code, make file, etc. Your source code must compile without any compiler warnings or errors.
2. Lab report, submitted in pdf format, with the following
    a. Name, partner's name (if one), date, and lab title
    b. Introduction. Summarize how you approached this problem. This should be one paragraph describing your approach.
    c. What things went right and what things went wrong with this lab.
    d. What did you learn from this lab.
    e. An analysis of how long you spent on this lab. This does not need to be formal, but how much time did each person put into this lab, and what areas required the most effort.
    f. Conclusions
    g. Source code as an appendix (In order that it can be commented)

Deliverables are to be uploaded to the course website.

If you have any questions, consult your instructor.