CS39002: Operating Systems Lab
Spring 2016
Assignment 3
Due on: 1st February 2016

Implementing Terminal Mirroring using message queue

In this assignment you will develop a system for multiple terminals which will be interacting with each other using message queues. The commands on a given terminal (shell), will be mirrored in the coupled terminal along with output.

Server: The system will have a server, which will start a message queue. It will also maintain the list of client terminals which joins the coupling. The server will be started first which will maintain the message queue and the coupling list. After receiving message from a client for the group, it sends the message to all the members of the group.

Client: The client program will have an interface similar to that of the shell assignments (as in Assignment 1&2). The programs will share a message queue, through which you can communicate among with the other terminals (via the server). There is a joining protocol to the terminal mirroring. This protocol will add the members to the list of terminals sharing their screen (like Team Viewer for GUI). After a terminal joins the group all the commands and the relevant outputs will be broadcasted to all the members which has joined the group.

Joining Protocol:

1. When a client types "couple" in the terminal, it is added to the broadcast group and an ID is sent to it from the server.

2. For leaving the group a client has to type "uncouple", which removes it from the broadcast group.

Broadcasting Protocol:

While broadcasting, when a member of the group types a command, it is executed in the terminal. The command along with the relevant output of the command is then forwarded to the server for broadcasting.

Interface

Server: The server will be started first and operate in the background and maintain the necessary components. It will print the updated list of clients on the screen.

Client:

1. It will have an interface similar to the shell as in (Assignment 2). It will run in an infinite loop and exit when "exit" command is given.
2. Upon joining the coupling group the server will send a ID to the client, and it will be displayed in the shell, to tell the user it's ID in the group.

3. Any command entered at a client is executed and the command along with the relevant output is sent to all the members of the group. The commands can be executed using a bash in forked subprocess. Only single line commands need to be supported.
4. The command entered by another terminal in the group will be displayed in the terminal of all the members (to be sent by server).
5. In addition to standard shell commands, “couple” and “uncouple” should be supported. Couple should join the server and mirror all other coupled clients as well as the current client to the other clients. Uncouple should “unjoin” from the server. Initial state when starting the client can be “uncoupled”.
6. The ID of the origin terminal is also displayed, to the member terminal.

Below example shows sample behaviour.

Example:

Let’s say we have two coupled terminals (command typed by the terminal are shown in bold)

Terminal 1	Terminal 2
/usr/home: **ls** File1.c File2.txt Terminal 2: mkdir test Terminal 2 : cd test Terminal 2: ls /usr/home: **uncouple** /usr/home: **cd ..** /usr: /usr: **couple** ID : 1 Terminal 2 : rmdir test	Terminal 1 : ls File1.c File2.txt /usr/home: **mkdir test** /usr/home : **cd test** /usr/home/test: **ls** Terminal 1 : uncouple /usr/home/test : **ifconfig** /* Output of the command */ /usr/home/test: cd /usr/home Terminal 1 : couple /usr/home : **rmdir test**

Deliverable

- Server.c (cpp), which will create the message queue and maintain the list of clients.
- Client.c (cpp), which will act as the shell and run in each terminal and execute commands.

Implementation Strategy (hint)

The client sends the message to the server which sends the message to the members.

For executing a command in a terminal bash in a subprocess can be used.

Only one message queue can be used by all the programs using priority feature.