lab11-irc-gui

CS240 Lab 11

Implementing an IRC (Internet Relay Chat) Client

Demo Grading Form

Before the Lab

Study in the slides on C++.

Goal

In this lab you will build a IRC client GUI (Graphical User Interface) that will communicate with your IRC server. For this you will be using the GTK 2.0 library in the Linux environment. Make sure that your program compiles and runs in data or the lab machines.

Using Your Laptop/Computer to Work on This Lab

To be able to see the IRC Client Window you either will need to work in one of the lab machines or you will need to use a Remote Desktop Program (RDP) and connect to **mc18.cs.purdue.edu**

Windows
For windows go to Start->All Programs->Accessories->Remote Desktop Connection and run it.

Macintosh
For MacOS download the Remote Desktop Program from here and run it.

Run the Remote Desktop program and then connect to **mc18.cs.purdue.edu** and type your login and password. You will get a Linux environment similar to what you will get in a computer in any of the Linux labs.

Purdue VPN
If you are in a network outside Purdue, you will need to connect first using VPN. See how to install VPN in Windows or in MacOS to connect to the Purdue network. The VPN server is webvpn.purdue.edu.

Thanks to the School of Science Systems support for making this possible.

Step 1. Setting up Your Environment

Remote login to data by typing:

```
ssh <your-user-name>@data.cs.purdue.edu
```

Change to the directory ~/cs240 that you created previously.

```
cd
cd cs240
```

Now copy the initial lab11 files by typing:

```
tar -xvf  /homes/cs240/2017Spring/lab11-irc-gui/lab11-src.tar
```

```
cd lab11-src
```

Step 2. Learning Client Sockets

In the initial sources there is a program lab11-src/TestIRCServer.c that shows you how to implement a client program that communicates to your IRC server. This program is used in the server test scripts and you will use it as a base in your IRC gui to communicate to your IRC server.

```
#include <time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>

char * user;
char * password;
char * host;
char * sport;
int port;

int open_client_socket(char * host, int port) {
        // Initialize socket address structure
        struct  sockaddr_in socketAddress;

        // Clear sockaddr structure
        memset((char *)&socketAddress,0,sizeof(socketAddress));

        // Set family to Internet
        socketAddress.sin_family = AF_INET;

        // Set port
        socketAddress.sin_port = htons((u_short)port);

        // Get host table entry for this host
        struct  hostent  *ptrh = gethostbyname(host);
        if ( ptrh == NULL ) {
                perror("gethostbyname");
                exit(1);
        }

        // Copy the host ip address to socket address structure
        memcpy(&socketAddress.sin_addr, ptrh->h_addr, ptrh->h_length);

        // Get TCP transport protocol entry
        struct  protoent *ptrp = getprotobyname("tcp");
        if ( ptrp == NULL ) {
                perror("getprotobyname");
                exit(1);
        }

        // Create a tcp socket
        int sock = socket(PF_INET, SOCK_STREAM, ptrp->p_proto);
if (sock < 0) {
                perror("socket");
                exit(1);
        }
        // Connect the socket to the specified server
        if (connect(sock, (struct sockaddr *)&socketAddress,
                    sizeof(socketAddress)) < 0) {
                perror("connect");
                exit(1);
        }

        return sock;
}

#define MAX_RESPONSE (10 * 1024)
int sendCommand(char *  host, int port, char * command, char * response) {

        int sock = open_client_socket( host, port);

        if (sock<0) {
                return 0;
        }

        // Send command
        write(sock, command, strlen(command));
        write(sock, "\r\n",2);

        //Print copy to stdout
        write(1, command, strlen(command));
        write(1, "\r\n",2);

        // Keep reading until connection is closed or MAX_REPONSE
        int n = 0;
        int len = 0;
        while ((n=read(sock, response+len, MAX_RESPONSE - len))>0) {
                len += n;
        }
        response[len]=0;

        printf("response:\n%s\n", response);

        close(sock);

        return 1;
}

void
printUsage()
{
printf("Usage: test-talk-server host port command\n");
        exit(1);
}

int
main(int argc, char **argv) {

        char * command;

        if (argc < 4) {
                printUsage();
        }

        host = argv[1];
        sport = argv[2];
        command = argv[3];

        sscanf(sport, "%d", &port);

        char response[MAX_RESPONSE];
        sendCommand(host, port, command, response);

        return 0;
}
```

To run this program type “make” and then open two windows. In one window run your

IRCServer <port>

and in other window try running the TestIRCServer

./TestIRCServer localhost <port> "ADD-USER superman clarkkent"

Try all different commands.

The **`sendCommand(char *  host, int port, char * command, char * response)`** function calls **`open_client_socket(host, port)`** to create a socket that connects to the server in the corresponding port. Then **`write(sock, command, strlen(command))`** sends the command to the server. the while loop reads the response and stores it in the string response.

Step 3. Learning GTK

Inside lab11-src type make and run the different GTK example programs.

```
make
./hello
<ctrl-c>
./entry
<ctrl-c>
./radio
<ctrl-c>
./panned
<ctrl-c>
./timer
```

These C programs show you how to use GTK. There is a tutorial; on how to use GTK in here: https://developer.gnome.org/gtk-tutorial/stable/ Go over the examples. the most useful one will be panned.c that shows how to make a split panel.

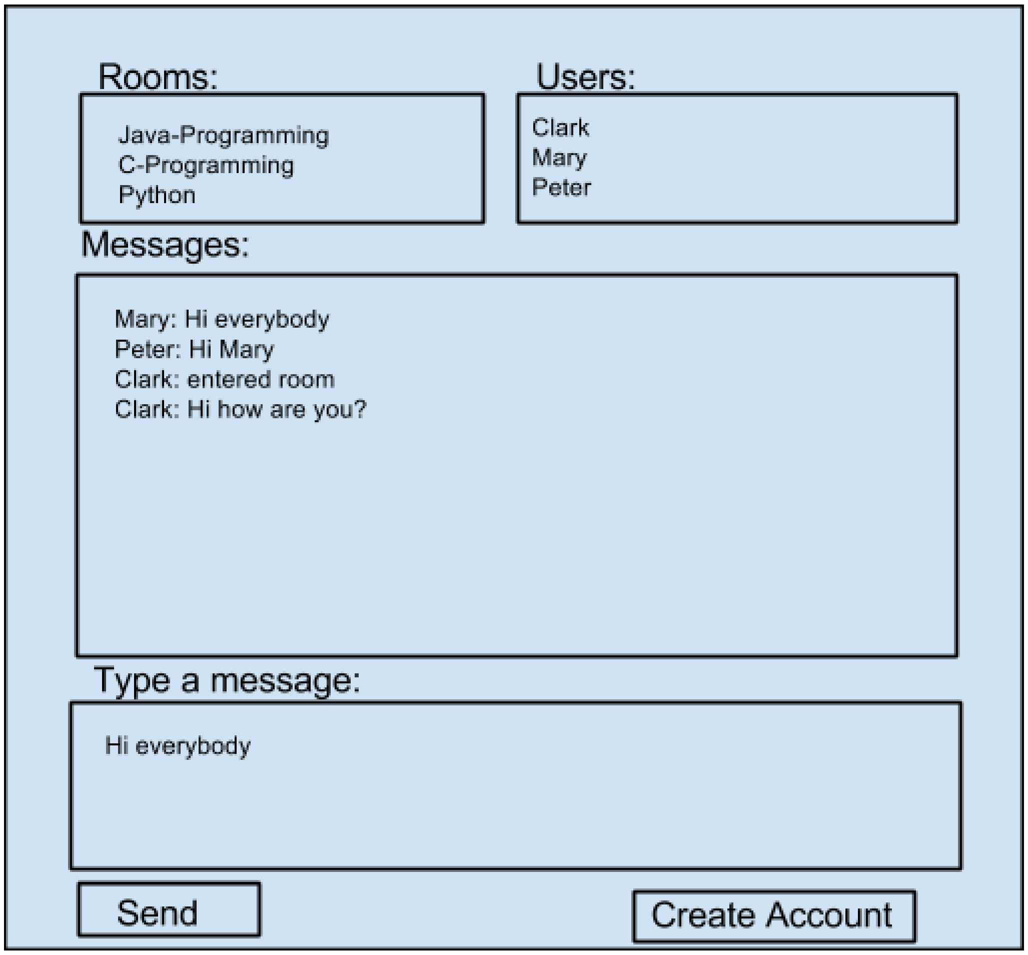
The GTK+ 2 reference manual is in https://developer.gnome.org/gtk2/stable/

See the widget gallery in https://developer.gnome.org/gtk2/stable/ch02.html

Also timer.c shows you how to start a timer that will be executed periodically You will use this timer to pull the messages and update the message window.

Step 4. Building the IRC client GUI

The following drawing shows a mockup of the IRC client. The GUI is just an example. You can improve it in any way you want.



Java-Programming
C-Programming
Python

Clark
Mary
Peter

Mary: Hi everybody
Peter: Hi Mary
Clark: entered room
Clark: Hi how are you?

Hi everybody

The user will enter a room by selecting any of the rooms. The users frame will be updated with the users in the room. The user will type a message and will be sent to the server.

Once every 5 secs, the GUI client will contact the server to get the new messages from the server. The client will get the messages from the server with a number larger than the last message number received.

When selecting Create Account, a dialog will appear that will ask for a user and password and a button OK. When pressing the button the account will be created in the server.

Step 5. Turning In your Project

Follow these instructions to turn in lab11:

Login to data.cs.purdue.edu and type

```
cd cs240
turnin -c cs240 -v -p lab11 lab11-src
```

The deadline for this lab is your lab time the week of Monday April 24th. You will demo your client that week.

Published by Google Drive – Report Abuse – Updated automatically every 5 minutes