Lorenzo Zafra - 1395521
CMPUT 379 Assignment 2 Project Report

Objectives:

The objective of the assignment is to learn how to develop a client-server program and the quirks that comes with such programs. This assignment specifically focuses on FIFOs for communication and file locking for controlling access to the said FIFOs. This assignment also taught us how to do various tests and experiments that deals with a client-server program as it can be quite tedious.

Design Overview:

- .c files - a2chat.c, server.c, client.c
    - server.c contains all the functions and functionality that deals with the server.
    - client.c contains all the functions and functionality that is needed by the client.
    - a2chat.c is the main program which initializes either the server or the client.
- Created a struct "conn" which holds various information about the clients and its connection. The server looks at this struct to find the fd, outFIFO name, username and receipients of a specific client.
- Created an enum for all the error codes that a user may encounter.
- To pass messages between clients and servers, the client is passing a message in a custom format which is then parsed by the server.

Assumptions:

- Each user can only have 10 recipients (can be changed in server.h) per session. To refresh this list, they would need to close the session and re-open a new one.
- The max username length is 30 characters
- Maximum server message length is 240 characters
- Maximum user input is 512 characters
- Maximum number of clients is 5.
- The server keeps all 5 inFIFOS open until it is terminated

Project Status:

At time of submission, the assignment is working as specified. In the beginning, I had difficulties with being able to read the messages sent by the server and be able to get input from the client without blocking. I later found out that this can be done easily by using poll().

Another difficulty I was having is that a printf() to the stdout will not be printed unless it is printed with a newline. I found out from stackoverflow that the stdout buffer needs to be flushed. The referenced link can be found below.

Also, there is an issue with the client where it will always have ‘a2chat_client: ’ before a message from the server. This is because the ‘a2chat_client:’ string is already printed to stdout, and any further printing (of the server’s messages) will keep that line. A TA I spoke to during the lab section said that that is okay and I won’t lose marks for it.

The last difficulty I had with the project is letting the client know that the username is taken if so. I fixed this by having the client open and lock both inFIFO and outFIFO, then the server checks if the username is taken. If it’s taken, it relays a message back to the client and the client closes the FIFOs.

Testing and Results:

With the help of ‘tmux’, I was able to multiplex multiple terminals and open multiple clients to test the functionalities of the program. From there, I would do various tests like messaging, closing, exiting, re-opening and adding recipients etc. Testing was successful and at time of submission I believe that there are no bugs.

A lot of unit testing was also done to check if various functions work by printing the output values to the STDOUT and seeing if they are correct.

Acknowledgements:

- APUE 3rd Edition
- https://linux.die.net/
- http://man7.org/linux/man-pages/man3/lockf.3.html
- http://beej.us/guide/bgnet/output/html/multipage/pollman.html
- http://www.cs.cmu.edu/afs/cs/academic/class/15213-f99/www/lab5-fifo.c
- http://pubs.opengroup.org/onlinepubs/7908799/xsh/open.html
- http://www.cs.rpi.edu/courses/fall96/netprog/lectures/html/ipc.html
- http://stackoverflow.com/questions/1716296/why-does-printf-not-flush-after-the-call-unless-a-newline-is-in-the-format-strin

How to Run:

To run the server:
`./a2rchat -s [fifo name] [num clients]`

To run the client
`./a2rchat -c [fifo name]`