Exercise 1:UNIX system programming

Mr. Roee Leon
Shenkar 2017

Bonus date: August 14th 2017
Due date: August 18th 2017

Instructions

1. Please grok Beej's guide for network programming before trying to do this ex.
2. Please send the compiling project in 1 ZIP or gzip file containing only. The zip file name should be your "your ID number" i.e. a filename should be 0123456789.tgz or 9876543210.zip
Do not submit RAR or 7z etc.
3. Please ensure your project compiles with make all
4. Please ensure make clean deletes your binary and any cruft.
5. Please add any additional instructions and comments to README.TXT file.
6. All dates refer to 23:59:59 Israel standard time on the date.
7. Submission by or before the bonus date will result in 10% grade BONUS
8. Submission by the due date is allowed.
9. Submission after the due date will be penalized by 25% for late submission (automatic) + 10% per day after the second day. (ie. Late submission by 1 minute will be penalized by 25%. Late submission by 4 days will result by 55% penalty) – PLEASE you are responsible adults. Manage your time and meet the deadline.
10. You will receive an "acknowledged mail" for your exercise within 6 hours.
If for any reason you did not receive mail please send another email as I may have missed your first email.
11. You need to implement this exercise on your own. You are allowed idea exchange ideas with your fellow students but not to exchange code. You are allowed to use web resources though. If you use any code you found on the Internet SPECIFY that in the README.TXT file.
If two students submit identical work a disciplinary action will be taken. (Even if both copied the same source from the net)
12. It is your responsibility to ensure your work compiles and run on the first submission. 2nd submission (and 3rd and 4th etc.) will be penalized by 10% penalty per "re---submission." exercises that do not compile will receive 0%.
13. Please follow KNF, 1TBS or BSD code style. Exceptionally good code will receive 10% BONUS. Other coding standards (Allman, GNU etc.) are allowed as long as they are consistent but will not result in bonus. Kludge will receive 10% penalty. 14. The code should be self---documented. You may comment critical parts of the code. Excessive commenting does not consist of good code and will be penalized.

Learning objectives

1. Grok TCP/IP socket programming and IPC
2. Grok unbuffered I/O API
2. Some C canapé before we go to something really interesting.
3. Simple system administration tasks on UNIX

Cover Story

The system that you are required to build in this exercise has 3 components.

- Chat Server – A simple live-chat server that also provides file-sharing services
- Chat Client – A simple TCP/IP chat client
- Admin Service – Chat management console to control the chat server

Chat Server

A live-chat server that "talks" to the connected clients using TCP/IP.
The server provides file-sharing services by storing the clients uploaded files within a pre-created directory. The server also provides Admin Service information using UNIX domain sockets.

Chat Client

A simple TCP/IP chat client.

Admin Service

A chat management console that communicates with the server using UNIX domain sockets. The management console allows sending simple queries to the server (*kickall*, *kick* (*id*), *chatmaster*, *list*).

The commands listed in red are bonuses (10% each)

Exercise 1 – select based Server (50%)

A – simultaneous server + client (30%) – select(2) + socket(2)/listen(2)/accept(2)/connect(2)/close(2)/read(2) /write(2)/lseek(2)

1. Start with the select---based chat server from beej.
    a. We will have TWO listening sockets. One for network elements and one for admin queries
    b. You may use the standard TCP/IP API for both sockets
2. Allow any number of clients to connect. Each client is given an ID upon connecting (may be used by the Admin Service)
3. The server allows uploading/downloading files by exporting the following commands to TCP/IP clients **only**: *download [file], listfiles*, *upload [file]*. You may store these files within a pre-created directory (e.g. within the server's root directory).
4. The server provides Admin Services by exporting the following commands to UNIX domain socket clients **only**: *kickall, chatmaster, list, kick [id]*.

Exercise 2 – Chat Client (25%)

The client has its own status to report to the server.

1. Start with Beej standard client.
2. Support the following commands: *download*, *listfiles, upload*. You may assume that the *download* command works only on files located in the CWD (same applies for *upload*).

    Usage example (assuming CWD is */home/someone/client*):
    ./client
    listfiles
    x.txt
    y.txt
    download x.txt

The client should now have the file *x.txt* within */home/someone/client*

Exercise 3 – UDS client (25%)

1. Start with Beej IPC client
2. Support the following commands:
    a. *chatmaster* – prints to stdout the online chat client that sent the most messages (just calculate the number of characters sent by each client)
    b. *kickall* – kicks all the currently connected clients from the server
    c. *list* – view a list of currently connected chat clients in the following form:
    [NAME] [ID]
    [NAME] [ID]
    …
    d. *kick [id]* – kicks the client with the given id from the server

More Bonuses …

1) FORK based server (10%)

1. Start with the server in section 1 and beej fork based server
2. Implement the same mechanism using processes instead of I/O multiplexing

2) MMAP/MUNMAP API (10%)

1. Modify your code (both client and server sides) to support file download/upload using *mmap*