Exercise 1:UNIX system programming

Mr. Roee Leon
Shenkar 2016

Bonus date: December 5th 2016
Due date: December 12th 2016

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 *fork/exec*
2. Some C canapé before we go to something really interesting.
3. Simple system administration tasks on UNIX
4. Bonuses – *fork*/*mmap* based server using record locking & *sendto*/*recvfrom* API

Cover Story

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

- Network element – connecting to a server and reporting status (a number between 1 and 100) every 5 seconds
- Control server – receiving info from all clients
- Monitor service – system management (C&C) console to provide info gathered.

The Network Element

TCP/IP clients that connects to a server and periodically (Every 3 seconds) report their status.

Control Server

A dummy server that "talks" to the connected clients using TCP/IP (Monitoring their status/info in a TEXT file and memory). The server checks whether the TEXT file needs to be updated every 10 minutes.
The server also "talks" to a Management Console (To provide it information) using UNIX domain sockets

Monitor Service

A management console that communicates with the server using UNIX domain sockets. The management console allows sending simple queries to the server (*whois* and *crit*) along with sending more complicated queries using *fork/execve/grep*.

Exercise 1 – select based Server (50%)

A – simultaneous server + client (30%) – select(2) + socket(2)/listen(2)/accept(2)/connect(2)/close(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 management queries
    b. You may use the standard TCP/IP API for both sockets
2. Allow any number of clients to connect
3. When connecting a new client his id and status should be traced (in a text based DB and memory)
4. The server should check whether the DB need to be updated every 10 minutes
5. The management console allows the following queries:
    a. *whois* – lists all of the connected clients along with their status
    b. *crit* – lists only the connected clients which are in “Critical State” (lowest 5%)
    c. *grep* ARGS – performs a *grep* based query on the text based DB

Exercise 2 – TCP/IP client (25%)

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

1. Start with Beej standard client.
2. Randomize a unique ID (mktemp is nice way) to be reported to the server
3. Randomize a number (this is your current status)
4. Once a client reported the status randomize a new status and report it again in 3 seconds
5. Repeat state 4 as long as you are alive

Exercise 3 – UDS client (25%)

1. The management console should send commands to the server and receive responses.
2. Three queries can be sent – display all connected clients, display all clients with status in the lowest 5% and *grep* based queries
3. Response should be calculated by the server and reported back to the client

Bonus – UDP (DGRAM) API (10%)

1. Modify your code to use *sendto*/*recvfrom* for the Management Console listening socket

Bonus – fork/mmap based server (20%)

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 (10%)
3. Implement shared memory using mmap(2).
4. Implement locking using fcntl(2) or flock(2)