ECEN 5013

Assignment 5: Socket Server

Github Classroom Link

https://classroom.github.com/a/P5lSocwJ

Suggested Reading:

1. Lecture 5 (daemons)
2. Lecture 10 (sockets)
    a. https://beej.us/guide/bgnet/html/single/bgnet.html
3. QEMU Documentation and network options:
    a. https://qemu.weilnetz.de/doc/qemu-doc.html#Network-options
4. Buildroot Documentation:
    a. https://buildroot.org/downloads/manual/manual.html#configure
5. Mastering Embedded Linux Programming Chapter 10: Starting Up
6. Init scripts documentation:
    a. http://man7.org/linux/man-pages/man8/start-stop-daemon.8.html

Github Classroom Start Instructions

You will start with an empty project submission repository, and fill it with the content of your assignment 4 local repository using these commands on your local buildroot-assignments repository

1. git remote rename origin assignment-4-remote
2. git remote add origin <your github classroom submission repo created with the link above>
3. git push origin master

Note that this assignment will also require changes to your aesd-assignments repository, and will also require some additional code for this repository from the aesd-assignments base repository assignment 5 branch. So you will be turning in changes to two repositories, with the buildroot repository referencing the correct commit in your aesd-assignments assignment-3-and-later repository.

Execute the following instructions in your aesd-assignments local repository folder:

1. git fetch assignments-base

2. git merge assignments-base/assignment5

Also make sure your aesd-assignments repository origin still points to your assignment-3-and-later repository using git remote get-url origin

Implementation:

1. Modify your aesd-assignments repository to add a new directory “server”.
2. Create a socket based program with name “**aesdsocket**” in the “server” directory which:
    a. Is compiled by the “all” and “default” target of your Makefile (along with your existing tester application) and supports cross compilation.
    b. Opens a stream socket bound to port 9000, failing and returning -1 if any of the socket connection steps fail.
    c. Listens for and accepts a connection
    d. Logs message to the syslog “Accepted connection from xxx” where XXXX is the IP address of the connected client.
    e. Receives data over the connection and appends to file /var/tmp/aesdsocketdata, creating this file if it doesn’t exist.
        i. Each data packet received should be separated by a newline character.
    f. Returns the full content of /var/tmp/aesdsocketdata to the caller.
    g. Logs message to the syslog “Closed connection from xxx” where XXXX is the IP address of the connected client.
    h. Restarts accepting connections.
    i. Gracefully exits when SIGINT or SIGTERM is received, completing any open connection operations, closing any open sockets, and **deleting the file /var/tmp/aesdsocketdata**.
        i. Logs message to the syslog “Caught signal, exiting” when SIGINT or SIGTERM is received.
3. Verify the sample test script “sockettest.sh” successfully completes against your native compiled application each time your application is closed and restarted. Add a commit with your working code.
4. Modify your program to support a -d argument which runs as a daemon. When in daemon mode the program should fork **after** ensuring it can bind to port 9000.
5. Add a System V startup script aesdsocket-start-stop to your assignment repository which uses start-stop-daemon to start your daemon with the -d option i.e the aesdsocket code should execute as a daemon if a -d argument is provided.
    a. On stop it should send SIGTERM to gracefully exit your daemon and perform associated cleanup steps.
6. Modify your buildroot aesd-assignments package to:
    a. Use your latest code.

   b. Install your **aesdsocket** executable to /usr/bin
   c. Install your aesd-socket-start-stop script to /etc/init.d/S99aesdsocket
7. Modify your runqemu.sh script to forward host port 9000 to your qemu instance port 9000.
8. Verify your socket is started automatically when starting qemu on the target. Verify your socket test scripts work as expected when running against the qemu target.

Validation:

1. You should be able to clone your final buildroot assignment repository to a new directory, run ./build.sh to build the system image and ./runqemu.sh to start the image.
2. When the QEMU image is started, it should pass port 9000 from the host into the guest image, the aesdsocket utility should be running in daemon mode, and all functionality should match the implementation section.
3. When the QEMU image is shutdown gracefully and restarted, the data returned over the connection (and content of /var/tmp/aesdsocketdata) should not contain content from a previous run (the shutdown script should have gracefully terminated the process during normal shutdown).
4. Ensure all error handling has been implemented wherever required.

Submission:

1. Your assignment submission repository should contain the buildroot setup used to generate and run the qemu image described above.
2. Your buildroot submission repository should reference your aesd-assignments repository, which will contain the assignment content and tags referenced above.

Extra Credit:

1. See the binary copy of the student assistant's socket server implementation at this link. You should find that this implementation passes with the sockettest script provided with the assignment (by replacing your implementation with this binary). Anyone who is able to find unique changes to the sockettest script which will cause the student assistant's implementation to crash or otherwise fail to meet requirements will be awarded 10 extra credit points for the assignment (subject to verification by the student assistants.)