ECEN 5013

Assignment 6: Yocto Environment Bringup and Threading

Github Classroom Link

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

Suggested Reading:

1. Lecture 13
2. Yocto documentation, including:
    a. https://www.yoctoproject.org/docs/latest/brief-yoctoprojectqs/brief-yoctoprojectqs.html
    b. https://www.yoctoproject.org/docs/latest/sdk-manual/sdk-manual.html#sdk-a-closer-look-at-devtool-add
    c. https://www.yoctoproject.org/docs/latest/dev-manual/dev-manual.html#usingpoky-extend-customimage-imagefeatures
    d. https://wiki.yoctoproject.org/wiki/Building_your_own_recipes_from_first_principles#The_bbexample_recipe
    e. https://www.yoctoproject.org/docs/latest/dev-manual/dev-manual.html#understanding-and-creating-layers
3. Linked List Documentation:
    a. https://blog.taborkelly.net/programming/c/2016/01/09/sys-queue-example.html
    b. http://cslibrary.stanford.edu/103/LinkedListBasics.pdf
4. Mastering Embedded Linux Programming Chapter 6

Implementation:

Continuation of Assignment 5:

1. Modify your socket based program to accept multiple simultaneous connections, with each connection spawning a new **thread** to handle the connection.
    a. Writes to /tmp/aesdsocketdata should be synchronized between threads using a mutex, to ensure data written by synchronous connections is not intermixed.
        i. For instance, if one connection writes “12345678” and another connection writes “abcdefg” it should not be possible for the resulting /var/tmp/aesdsocketdata file to contain a mix like “123abcdefg456”, the content should always be “12345678”, followed by “abcdefg”
    b. The thread should exit when the connection is closed by the client or when an error occurs in the send or receive steps.

    c. Your program should continue to gracefully exit when SIGTERM/SIGINT is received, after requesting an exit from each thread and waiting for threads to complete execution.
    d. Use the singly linked list API's discussed in class to manage threads.
        i. Use pthread_join() to join completed threads, do not use detached threads for this assignment.
2. Use the updated sockettest.sh script to test your modified implementation. You can obtain the file from https://github.com/cu-ecen-5013/aesd-assignments/blob/assignment6/sockettest.sh
3. Modify your aesdsocket source code repository to:
    a. Append a timestamp in the form "timestamp:time" where time is specified by the asctime format, followed by newline. This string should be appended to the /var/tmp/aesdsocketdata file every 10 seconds, where the string includes the year, month, day, hour (in 24 hour format) minute and second representing the system wall clock time.
    b. Use appropriate locking to ensure the timestamp is written atomically with respect to socket data
    Hint:
    Think where should the timer be initialized. Should it be initialized in parent or child?

Yocto Installation:

4. Install essential build packages on your VM as described in the "Build Host Packages" section of the yocto quick start guide.
5. Add yocto poky as a git submodule in the root of your project repository.
    a. Use https://git.yoctoproject.org/cgit/cgit.cgi/poky/ as the source
    b. Use the branch "warrior"
6. Update the recipe "aesd-assignments" in the meta-aesd/recipes-aesd folder of your project as discussed in class. This recipe should build your assignment 5 aesdsocket source using the git repository with appropriate commit specified via SRCREV and install associated files in the rootfs of the system.
7. Add your package to core-image-aesd as described in class
    a. Remember to add your start script as well using the instructions in class
8. Use the **build.sh** script to run your combined image
9. Add a **runqemu.sh** script in the root folder which:
    a. Runs *source oe-init-build-env* from the poky directory to setup your yocto build environment
    b. Uses *runqemu nographic* plus appropriate arguments to start QEMU with host port 9000 forwarded to your QEMU instance.
10. The updated sockettest.sh script linked above should pass when run against your running qemu image.
    a. Remember to use ./sockettest.sh -t 192.168.7.2 to run against the target when using runqemu yocto tunneling support

Validation:

1. You should be able to clone your final yocto 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. Timestamps should be written as described in the assignment instructions.
5. Threads should be implemented to handle connections as described in the assignment instructions.
    a. You should use a linked list to manage threads.
6. The sockettest.sh script should pass when run against your running qemu image.
7. You should not have any memory leaks or errors identified when running valgrind.

Submission:

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