Programming Assignment Zero

CMSC 417 Spring 2014

1 Deadline

Feb 7, 2014. Please complete by Feb 1 unless you have technical trouble. You may want to start the next assignment before this one is "due." This assignment is intended to make sure you're able to write and run a little socket code. If you find that you have trouble with submit/grades/linuxlab after Feb 3, I will grant no extensions.

Post general questions to piazza.

2 Objective

The goals of this mini-project are:

- to de-mystify the process of generating code for networks.
- to test out the submit server, linuxlab and class accounts on grades.cs.umd.edu (if needed).
- to welcome you back from break!

3 Specification

Write a program, in any language supported by the submit servers[1] that has the following behavior:

1. Connect using TCP to argv[1] with port argv[2].
2. Read stdin and write that stuff to the TCP socket until stdin ends (end of file).
3. Shutdown the socket for writing. See shutdown(2). (i.e., man shutdown)
4. Read what the TCP socket has and write that stuff to stdout until the socket ends (end of file because the other side closed).
5. Exit. Important! If you don't exit, the tests will fail.

[1] Or any language at all with advance permission of the instructor or TA to demo the code for credit using an esoteric language. Supported languages include at least perl, python, ruby, C, java, and probably fortran.

4 Example Tests

```
echo "GET /" | pa0 www.google.com 80
```

If you ask for a web page, expect to see `</html>` at the end.

5 Overspecification

Don't make this harder on yourself than it has to be. You don't have to alternate reading stdin and writing stdout. You don't have to call select() or poll() to figure out what's available. Non-blocking I/O is not needed. Ruby's ARGV[0], for example, is the same as C's argv[1]; I specify as if implemented in C and you knew what I meant from the example anyway. Where I wrote "socket" above, you can substitute "stream" or whatever your language really likes.

You may not use any glorious library that makes everything easy that does not come with the language. (For example, glib and the apache portable runtime are forbidden.) If you wrote it yourself, you're welcome to turn it in, though.

Stdin may be arbitrarily large. What comes from the server may be arbitrarily large. (That is, there's a reason step 2 is a composite of two things.) In general, don't ask me how big a buffer needs to be.

6 Hints

I suggest you use C, just to dust off skills that you will need later in the semester... but it will take a bit more effort to complete this assignment in C because of diminished support.

If you use C, try to use getaddrinfo() instead of gethostbyname. In C, read(socket, ..., ...) will return 0 when there is no more. The send() and recv() functions are not necessary; read() and write() are sufficient.

7 To Turnin

Provide to the submit server your source and a Makefile that will permit pa0 to be the executable file to run. The makefile WILL NEED TO "chmod a+x pa0" IF pa0 is not compiled by gcc (e.g., is any sort of script). The pa0 may be a shell script whose contents are "java -jar icky-language.jar $1 $2"

Your code MUST RUN on the submit server. You may have to test the compilation on linuxlab (which has a similar setup). Errors can occur. You may need more #include lines than you expect or they must be in the order on the man page. It is not sufficient that your code work on some machine you happen to know about. IF your code runs on linuxlab but not on submit, we'll grant you a pass on this requirement... at least until we can figure out the mismatch.

8 Finding Code On-line

If you find precisely this online, don't copy it. Cite sources in the top of your file. Basic documentation should be 75% of the code here.