

CS 4700 / CS 5700 - Network Fundamentals

Project 1: Simple Client

This project is due at 11:59pm on February 1, 2022 and is for 5700 students only (i.e., those who did not do it in CS 3700). It is highly recommended that any 4700 students who have not taken 3700 do it, but it will not be graded.

Description

This assignment is intended to familiarize you with writing simple network code. You will implement a client program which communicates with a server using sockets. The server will ask your program to solve hundreds of simple mathematical expressions. If your program successfully solves all of the expressions, then the server will return a *secret flag* that is unique for each student. If you receive the *secret flag*, then you know that your program has run successfully, and you will receive full credit for the assignment.

Language

You can write your code in whatever language you choose, as long as your code compiles and runs on **unmodified** Khoury Linux machines **on the command line**. Do not use libraries that are not installed by default on the Khoury Linux machines. Similarly, your code must compile and run on the command line. You may use IDEs (e.g. Eclipse) during development, but do not turn in your IDE project without a Makefile. Make sure your code has **no dependencies** on your IDE.

Protocol

The server runs on the machine *project1.5700.network* and listens for requests on a TLS socket bound to port 27995. This exercise has four types of messages: HELLO, STATUS, SOLUTION, and BYE. Each message is an ASCII string consisting of multiple fields separated by spaces (0x20) and terminated with a line feed (0x0A, \n). The maximum length of each message is 16384 bytes. Messages are case sensitive.

The protocol works as follows. The client initiates the protocol by creating a TLS socket connection to the server. Once the socket is connected, the client sends a HELLO message to the server. The format of the HELLO message is:

```
cs5700spring2022 HELLO [your NEU ID]\n
```

In your program you should replace [your Northeastern ID] with the username of your northeastern.edu email address. You must supply your current email username so the server can look up the appropriate *secret flag* for you. The server will reply with a EVAL message. The format of the EVAL message is:

```
cs5700spring2022 EVAL [txpr]\n
```

The txpr (tree expression) will be a large, complex arithmetic expression, that follows the following grammar:

```
txpr := [int] | '(' [txpr] [op] [txpr] ')'  
op   := '+' | '-' | '*' | '/' | '<<^'
```

All integers in the tree are 32-bit signed integers, however, the result of the calculation must have infinite precision. The add, subtract, and multiply operators have their usual definitions. Division is defined as follows:

```
'a // b' := floor[a / b]
```

And

```
'a <<^ b' := (a << 13) (xor) b
```

Your program must solve the expression and return the answer to the server in a STATUS message. The STATUS message has the following format:

```
cs5700spring2022 STATUS [result]\n
```

It is okay for the solution to be negative. In the case of division, round the answer down to the nearest integer (do not send floating point numbers to the server).

If evaluating the expression would result in an arithmetic error, your code must send back an error message:

```
cs5700spring2022 ERR [msg]\n
```

Per the definition of the grammar, the only error that can happen is divide-by-zero. The associated error code is

```
#DIV/0
```

The server will respond to the STATUS message with either another EVAL message, or a BYE message. If the server terminates the connection, that means your solution was incorrect. If the server sends another EVAL message, your program must solve the expression and return another STATUS message. The server will ask your program to solve hundreds of expressions; the exact number of expressions is chosen at random. Eventually, the server will return a BYE message. The BYE message has the following format:

```
cs5700spring2022 BYE [a 64 byte secret flag]\n
```

Once your program has received the BYE message, it can close the connection to the server. If the server returns "Unknown_Husky_ID" in the BYE message, that means it did not recognize the NEU ID that you supplied in the HELLO message. Otherwise, the 20-character ASCII string is your *secret flag*: write this value down, since you need to turn it in along with your code.

Your client program

Your client program must execute on the command line using the following command.

```
$ ./client <-p port> <-s> [hostname] [NEU ID]
```

Your program must follow this command line syntax **exactly**, i.e. your program must be called *client* and it must accept these exact parameters in exactly this order. If you cannot name your program *client* (i.e. your program is in Java and you can only generate *client.class*) then you **must** include a script called *client* in your submission that accepts these parameters and then executes your actual program. Keep in mind that all of your submissions will be evaluated by grading scripts; if your program does not conform **exactly** to the specification then the grading scripts may fail, which will result in loss of points.

The *-p port* parameter is optional; it specifies the TCP port that the server is listening on. If this parameter is not supplied on the command line, assume the port is 27995. The *-s* flag is mandatory; it indicates that the client should use an encrypted socket connection. The *[hostname]* parameter is required, and specifies the name of the

server (either a DNS name or an IP address in dotted notation). The [NEU ID] parameter is required. Your code must support NEU IDs that have leading zeroes (do not strip them!).

Your program should print **exactly one line of output**: the *secret flag* from the server's BYE message. If your program encounters an error, it may print an error message before terminating. Your program **should not write any files to disk**, including writing to the *secret_flags* file.

Other Considerations

You may test your client code with our server as many times as you like. Your client should conform to the protocol described above, otherwise the server will terminate the connection silently. Your client program must verify the validity of messages by strictly checking their format, i.e. the server may send corrupted messages just to try and crash your software. If a received message is not as expected, such as an incorrect field or wrong message type, you must assert an error and terminate your program. You should be strict; if the returned message does not exactly conform to the specification above, you should assert an error. Remember that network-facing code should be written defensively.

Submitting Your Project

Before turning in your project, you must register. To register yourself in the class, execute the following script:

```
$ /course/cs5700sp22/bin/register-student [your-NUID-number]
```

Next, you must register yourself for the project by executing the following script:

```
$ /course/cs5700sp22/bin/register project1 [your-ccis-username]
```

This will either report back success or will give you an error message. If you have trouble registering, please contact the course staff.

To turn-in your project, you should submit your (thoroughly documented) code along with three other files:

- A Makefile that compiles your code.
- A plain-text (no Word or PDF) README file. In this file, you should briefly describe your high-level approach, any challenges you faced, and an overview of how you tested your code.
- A file called *secret_flags*. This file should contain your *secret flag*, in plain ASCII.

Your README, Makefile, *secret_flags* file, source code, etc. should all be placed in a directory. You submit your project by running the turn-in script as follows:

```
$ /course/cs5700sp22/bin/turnin project1 [project directory]
```

[project directory] is the name of the directory with your submission. The script will print out every file that you are submitting, so make sure that it prints out all of the files you wish to submit! You may submit as many times as you wish; only the last submission will be graded, and the time of the last submission will determine whether your assignment is late.

Grading

This project is worth 8 points. If your program compiles, and you successfully submit the *secret flag*, then you will receive full credit. We will randomly check student's code to make sure that it works correctly. All student code will be scanned by plagiarism detection software to ensure that students are not copying code from the Internet or each other.

Using openssl s_client

You may find it useful to interrogate the server at the command line to see examples of EVAL messages. OpenSSL contains a generic SSL/TLS client, *s_client*, which connects to remote hosts using SSL/TLS. NOTE: The server will time out and close the connection if ~20 seconds elapse and no input is given.

Below is an example using *s_client*:

```
$ openssl s_client -quiet -connect project1.5700.network:27995
depth=3 0 = Digital Signature Trust Co., CN = DST Root CA X3
verify return:1
depth=2 C = US, 0 = Internet Security Research Group, CN = ISRG Root X1
verify return:1
depth=1 C = US, 0 = Let's Encrypt, CN = R3
verify return:1
depth=0 CN = project1.5700.network
verify return:1
cs5700spring2022 HELLO [your ID]
cs5700spring2022 EVAL ( ( ( ( ( ( ( 330938623 * ( -44720962 // 12312 ) ) //
-14069 ) // ( ( ( ( 21486 + -3797 ) - ( 6539 - -4354 ) ) // -92 ) // ( (
-76 + ( ( -62 // 3 ) - ( -80 + -111 ) ) ) - 84 ) ) ) <<^ ( ( -1662702836
// ( 24247 - -6619 ) ) + ( ( ( ( -1498156034 <<^ -708311476 ) - (
1541239628 // -24158 ) ) * ( ( 206140450 + ( 828742585 + -1005239800 ) ) *
( ( -463178467 // 23507 ) + ( ( ( -2090027431 - 2036864129 ) * 1806232291
) * ( 158179264 * -887639797 ) ) ) ) // 29362 ) ) ) - ( ( ( -1285730016
// ( 27254 * -26475 ) ) <<^ ( ( -595204296 * 1993902957 ) + ( -1967172409
+ -372520500 ) ) ) // ( ( 31441 + ( -23735 <<^ ( 10745 * 26428 ) ) ) +
17296 ) ) ) - ( ( -1321704357 <<^ ( ( ( ( -1198070549 + -1614881430 ) +
-1814505099 ) // -1157 ) - 1251931495 ) ) * ( ( -545687743 <<^ 551487888 )
- 1081017645 ) ) ) <<^ -1846872194 ) + ( ( ( ( ( ( -1665375506 +
-233210943 ) <<^ ( 1177854063 + -1602657823 ) ) * ( -2080846166 * (
296865374 <<^ -1261409301 ) ) ) * ( ( ( 2064756327 <<^ -206315412 ) // ( (
-24974 // 10 ) + ( ( -24220 - -24063 ) + -31439 ) ) ) + 40477143 ) ) // (
( ( -18376 // ( ( ( -127 - 61 ) + ( 50 * 74 ) ) // ( 7 // -1 ) ) ) ) // ( (
-92 * ( 87 - -30 ) ) <<^ 111 ) ) // ( ( 7 // ( -7 + -2 ) ) * ( -75 * (
-106 <<^ -7 ) ) ) ) ) + ( ( 1057863386 * 380984409 ) // ( ( 10347 * 24704
) + ( -25579 + -10966 ) ) ) ) )
cs5700spring2022 STATUS 106239235818368919843985044682453135716255749122013737872
cs5700spring2022 EVAL ( ( ( ( ( 2121034265 - ( -1362320398 // -1598327332 ) ) <<^ (
-1084742079 <<^ 2025695981 ) ) + ( ( ( -1499725523 * 1226180197 ) <<^ (
-1458962115 // 1679243938 ) ) ) <<^ ( ( ( ( -37467154 * 868508523 ) * ( ( (
-1919520462 - 1283811728 ) * ( ( -886841718 // 194114367 ) * ( -510612122 - (
241187362 // 1211702715 ) ) ) ) // ( ( ( -793873958 + -549241529 ) + -1520294578
) + ( -1969166079 + 286459704 ) ) ) ) - 1780140246 ) <<^ ( ( -1451648835 *
-1688375010 ) + ( ( -453990213 // -653825190 ) + ( ( ( -1534980779 * 1008208048
) + ( 228730444 - 2011470872 ) ) * -446146030 ) ) ) ) ) <<^ ( ( ( ( (
-1898346654 <<^ -1301453199 ) * 768560566 ) // ( -114228069 // 1947249057 ) ) -
( ( ( ( -979709711 - ( 250343953 * -617359272 ) ) + 64459754 ) <<^ ( ( ( (
1043124425 + ( 1192091003 <<^ 561849741 ) ) + 1204338143 ) + 1341709223 ) <<^ (
( 1853584882 + ( ( -2087831407 // ( ( -1859587024 - -1449799538 ) * 706731343 )
) // 1434406504 ) ) + ( 265524022 * 2129947997 ) ) ) ) * ( -1561046929 // (
2071735449 <<^ 1598417605 ) ) ) ) <<^ ( ( -1938880905 // 1732631279 ) -
1063094031 ) ) ) * ( ( ( ( 208429724 + ( -1695507684 <<^ 1983222085 ) ) <<^ (
1623455938 + ( ( -34937652 <<^ ( -569452094 <<^ -916871833 ) ) // ( ( 1059251101
+ 1845071692 ) - -1672536338 ) ) ) ) - ( ( ( ( 1572948000 + ( -936620457 //
1476986110 ) ) <<^ ( ( 1141664059 - -1693307541 ) * ( ( -255634292 + -1395580150
) * ( 1084630309 <<^ -1169790078 ) ) ) ) ) - ( ( -547459515 // 482297501 ) * ( (
461828879 * 298466258 ) - ( ( -794306182 <<^ -1700439312 ) * ( -2039336747 +
-1163422135 ) ) ) ) ) + -1869021759 ) ) <<^ ( ( ( ( 105164505 * -604781850 ) *
-812336002 ) + ( ( ( ( ( -1906538964 * 886189226 ) + 1311829189 ) <<^ (
1599178428 - -1427793326 ) ) // ( ( ( ( -22481901 <<^ 2053383808 ) - 1699342568
```

```
) + ( ( -382764021 <<^ 1820180422 ) <<^ 1571849620 ) ) + ( -283543375 -  
-55971170 ) ) ) - ( ( ( -1290351086 <<^ ( -1132344796 * 37006194 ) ) // (   
226237214 * ( ( -124145501 * -645277570 ) * 130204634 ) ) ) // 1921549170 ) ) )  
// ( 525715353 // ( ( ( 719754363 - -1754491217 ) + ( -131460798 * ( 2109901059  
* 1817484234 ) ) ) <<^ ( ( -1308336071 + -1214320267 ) * 1342918677 ) ) ) ) )  
cs5700spring2022 ERR #DIV/0  
.  
.  
.  
cs5700spring2022 BYE SIYZ5B1/PG7529RSBYIJ
```