CSC358H5: Principles of Computer Networking — Winter 2025

Programming Assignment 0: Warm-up

Due Date: Sunday, Jan 26, 11:59:59 PM

Welcome to CSC358H5. Apart from lectures, problem sets, and Wireshark labs, we'll be delving into a series of Programming Assignments. The primary aim of these tasks is to provide students with an opportunity to acquire hands-on experience in crafting networking code. Throughout the course, there will be four programming assignments. The first three assignments build upon each other, such that you only need to download the starter code **once** and simply work on the files for the respective assignment. It's noteworthy that the first three assignments are modeled after Stanford's CS 144 course lab assignments.

In this programming assignment, *i.e.* PA0, you will establish the environment necessary for the subsequent two programming tasks. Furthermore, you will complete a relatively short program in C++ to familiarize yourself with the coding environment to be employed in these assignments.

PA0 is graded with a total of 10 marks and it contributes 2% towards your course grade.

Collaboration

All programming assignments should be done individually. All codes that you use in your submissions should be written by you. Do NOT simply copy-and-paste code from websites such as Stack Overflow, GitHub, or even AI generating agents. It is OK to see the sample code snippets on such places, but you should write the code yourself and give the credit to them by citing the source that you used (*e.g.*, by mentioning the page URL).

You should not share your code with others or see other student codes. It is OK to discuss the assignment or potential solutions for them with another student. However, in addition to writing the whole code yourself, you should also mention this (the discussion with other students in finding the answer to the assignments) in the submissions.

You can also ask your questions on the course discussion board publicly, as long as you do not reveal your code. If there is a specific question related to a part of your code that needs revealing it, either ask it in a tutorial session or an office hour from a TA or via a private question on the discussion website.

Using Git

You are welcome to store your code in a private repository on GitHub, GitLab, Bitbucket, etc., but please make sure your code is not publicly accessible.

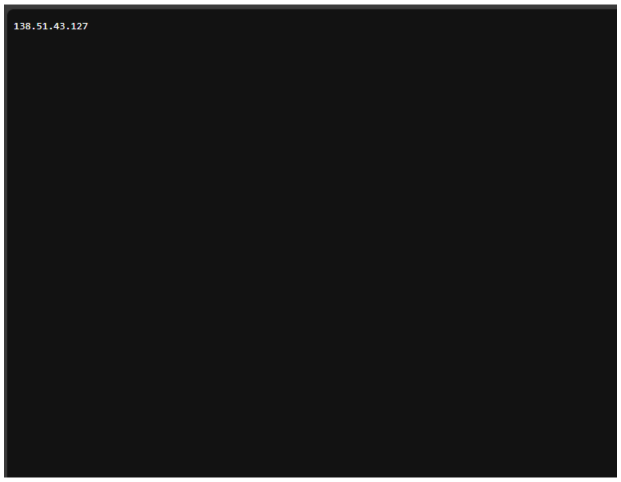
0 Environment Setup

The programming assignments require Linux operating system and a recent C++ compiler. All the required modules have already been installed on the lab machines so you can easily work on the assignments on the lab machines, either on a machine physically, or through ssh.
[**NOTE: It is crucial to confirm that your submissions for PA0, PA1, and PA2 align with the lab environment, as these submissions will undergo compilation and autotesting within that specific environment.**]

1 Simple HTTP Get Request

Before describing the main task that your program should perform for this assignment, we will give you a brief summary of how a client can use the HTTP protocol to send a request to a server. This is the primary task that web browsers use to fetch information from the internet servers and display it to you. Here, we try to send a request to a server that will return the public IP address of our computer to us.

i. Visit the website `https://api.ipify.org/` in a web browser. You should see a page similar to Figure 1 that contains the public IP address of your computer. Take a note of this IP address. In the next steps, we will try to send the same request directly to the server and get the answer.



```
138.51.43.127
```

Figure 1: Visit `https://api.ipify.org/` in your browser to check your public IP address.

ii. On the lab machine, run the following command:

Linux Command: Establishing a TCP connection with nc

```
echo -e "GET / HTTP/1.1\r\nHost: api.ipify.org\r\nConnection: close\r\n\r\n" | nc api.ipify.org 80
```

This command uses the nc program to establish a connection with the host "api.ipify.org" and engage with the program operating on the port linked to the "http" protocol (specifically port 80). If your computer is connected to the internet, you should be able to connect to it.

- The first command sends a "GET" request to get the service at endpoint (or path) "/". It will also specify to use the version 1.1 of HTTP protocol.
- The second command tells the server the host (*i.e.*, "api.ipify.org") that should handle this request.
- The third command tells the server that after processing our request and returning back the answer to us, we do not have any more request and it should terminate the connection.
- Finally, the extra enter at the end (*i.e.*, empty line command) indicates that our request is complete.

Once the TCP connection is established to the server, you should see something like this:

Output

```
HTTP/1.1 200 OK
Date: Fri, 08 Sep 2023 18:18:34 GMT
Content-Type: text/plain
Content-Length: 13
Connection: close
Vary: Origin
cf-cache-status: DYNAMIC
Server: cloudflare
```

```
138.51.8.116
```

The first few lines are the **header** that the server sends back. For example,

- the first line indicates that the request was handled successfully by returning the status code "200".
- The next one provides information about the server (*e.g.*, it is running "cloudflare").
- The last line is the body of the response, which will be your public IP address.

After running this, you should see the same IP address as in part **i.**

2 Writing a Program to Fetch a Webpage

The main part of this assignment is to complete a simple program that performs similar action. It will connect to the web server, send a request to get the page, and display the result that it receives.

In this assignment, your program will ask the operating system to provide a TCP socket to it. Then it connects that socket to the server that it wants to fetch page from (and port 80 of that server). After successfully connecting to the server, it will send the GET request similar to what you did in the previous section to get the page that it needs (usually referred to as path).

2.1 Getting the starter code

Download the assignment starter code (*i.e.*, pa_networking_stack.tar.gz) from the course Quercus webpage. It is available under "Files/Programming Assignments" section.
The easiest ways to move the starter code from your local machine to the lab machine is by SSHing to the lab machine through VSCode and simply dragging and dropping the folder wherever you wish to place it in the machine. You can also use "scp" if you like.

On the lab machine, extract the compressed ".tar.gz" file and enter the folder. Then, run the following command to create directory for compiling the code.

Linux Command: Configuring the build system and generating the build files in the "build" directory

```
cmake -S . -B build
```

Run the following command every time you make a new session to add a GNU dependency required to run the pa0 tests.

Linux Command: Adding the required GNU dependency

```
export LD_LIBRARY_PATH="/opt/gcc-14/lib64:$LD_LIBRARY_PATH"
```

Enter the build directory. You can build the source code by running the following command **from the build directory**:

Linux Command: Building the source code

```
cmake --build .
```

2.2 Provided Support Code

To support this style of programming, utility classes in the starter code wrap operating-system functions, which can be called from C, in "modern" C++. We have provided you with C++ wrappers for concepts such as sockets and file descriptors that you may be familiar from previous programming courses.

Please read over the public interfaces (*i.e.*, the part that comes after "public:" in the files util/socket.hh and util/file_descriptor.hh). Note that a Socket is a type of FileDescriptor, and a TCPSocket is a type of Socket.

2.3 Completing webget.cc

It's time to implement webget, a program to fetch Web pages over the Internet using the operating system's TCP support and stream-socket abstraction – just like you did by hand earlier in this assignment.

i. Open the webget.cc file located in the "src" directory of the starter code.
[**NOTE:** This file is different from the webget file in the "src" directory of the "build" directory].

ii. In the get_URL function, find the comment starting "// Update here with your code!."

iii. Implement the simple Web client as described in this file, using the format of an HTTP (Web) request that you used earlier. Use TCPSocket and Address classes. Our solution is no more than ten lines of code. However, feel free to deviate from that.

iv. Enter the build directory, if you have not done so already. Compile your program by running "make" **from the build directory**. If you see an error message, you will need to fix it before continuing.

v. You can test your program by running the following command **from the build directory**:

Linux Command: Testing your implementation

```
./src/webget api.ipify.org /
```

How does this compare to what you see when visiting `http://api.ipify.org/` in a Web browser? How does it compare to the results from using telnet earlier in this assignment? Feel free to experiment and test it with any http URL you like!

vi. When it seems to be working properly, run the following command **from the build directory** to run the provided automated tests:

Linux Command: PA0 automated tests

```
cmake --build . --target pa0
```

Before implementing the get_URL function, you should expect to see something similar to the following after running the automated tests.

Output: Test results before implementing the get_URL function

```
Start 1: compile with bug-checkers
1/2 Test #1: compile with bug-checkers ........ Passed 3.34 sec
Start 2: t_webget
2/2 Test #2: t_webget ........................***Failed 0.01 sec
Function called: get_URL(httpbin.org, /base64/SGVsbG93IHdvcmxk)
Warning: get_URL() has not been implemented yet.
ERROR: webget returned output that did not match the test's expectations

50% tests passed, 1 tests failed out of 2
```

```
Total Test time (real) = 3.35 sec

The following tests FAILED:
2 - t_webget (Failed)
Errors while running CTest
make[3]: *** [CMakeFiles/pa0.dir/build.make:70: CMakeFiles/pa0] Error 8
make[2]: *** [CMakeFiles/Makefile2:1315: CMakeFiles/pa0.dir/all] Error 2
make[1]: *** [CMakeFiles/Makefile2:1322: CMakeFiles/pa0.dir/rule] Error 2
make: *** [Makefile:693: pa0] Error 2
```

After completing the assignment, you will see:

Output: Test results after correctly implementing the get_URL function

```
Start 1: compile with bug-checkers
1/2 Test #1: compile with bug-checkers ........ Passed 0.14 sec
Start 2: t_webget
2/2 Test #2: t_webget ......................... Passed 1.13 sec

100% tests passed, 0 tests failed out of 2

Total Test time (real) = 1.27 sec
```

3 Hints

You may find the following hints helpful.

i. Please note that in HTTP, each line must be ended with "\r\n" (it's not sufficient to use just "\n" or endl).

ii. Don't forget to include the "Connection: close" line in your client's request. This tells the server that it shouldn't wait around for your client to send any more requests after this one. Instead, the server will send one reply and then will immediately end its outgoing bytestream (the one from the server's socket to your socket). You'll discover that your incoming byte stream has ended because your socket will reach "EOF" (end of file) when you have read the entire byte stream coming from the server. That's how your client will know that the server has finished its reply.

iii. Make sure to read and print all the output from the server until the socket reaches "EOF" (end of file) – a single call to read is not enough.

4 Submitting The Assignment and Mark Breakdown

You can access the MarkUs submission website of the course at:

`https://markus.teach.cs.toronto.edu/utm-2025-01/main/login_remote_auth`

We prepared the submission assignment for Programming Assignment 0 there (PA0). You should be able to login to the MarkUs with your UTORid credential. For this assignment, you should submit two files:

- webget.cc
- pa0.md

The first one is the client code that you should update to send the request and print the response. The second file includes a few questions that you should answer. Please submit both files.

The graders will run your webget program with the same automated test that is provided to you. However, we will read through your code to make sure that it is implemented properly and it doesn't only work with the hostname and path used by the unit tests.

The following is the mark breakdown for this assignment:

(i) Test #1: 1 point

(ii) Test #2: 8 point

(iii) Completing pa0.md and answering the mandatory questions there: 1 point

5 A Few Notes About Modern C++ and Starter Code

The programming assignments will be done in a contemporary C++ style that uses recent (2011) features to program as safely as possible. This might be different from how you have been asked to write C++ in the past. For references to this style, please see the C++ CoreGuidelines (`http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines`).

The basic idea is to make sure that every object is designed to have the smallest possible public interface, has a lot of internal safety checks and is hard to use improperly, and knows how to clean up after itself. We want to avoid "paired" operations (*e.g.*, malloc/free, or new/delete), where it might be possible for the second half of the pair not to happen (*e.g.*, if a function returns early or throws an exception). Instead, operations happen in the constructor to an object, and the opposite operation happens in the destructor. This style is called "Resource acquisition is initialization," or RAII.

In particular, we would like you to:

- Use the language documentation at `https://en.cppreference.com` as a resource. (We'd recommend you avoid cplusplus.com which is more likely to be out-of-date.)
- Never use malloc() or free().
- Never use new or delete.
- Essentially never use raw pointers (*), and use "smart" pointers (unique ptr or shared ptr) only when necessary. (You will not need to use these in CSC358.)
- Avoid templates, threads, locks, and virtual functions. (You will not need to use these in CSC358.)
- Avoid C-style strings (char *str) or string functions (strlen(), strcpy()). These are pretty error-prone. Use a std::string instead.
- Never use C-style casts (*e.g.*, (FILE *)x). Use a C++ static cast if you have to (you generally will not need this in CSC358).
- Prefer passing function arguments by const reference (*e.g.*: const Address & address).
- Make every variable const unless it needs to be mutated.
- Make every method const unless it needs to mutate the object.
- Avoid global variables, and give every variable the smallest scope possible.
- Before handing in an assignment, it is recommended to run the following, **from the build directory**, for suggestions on how to improve the code related to C++ programming practices

  Linux Command: tidy

  ```
  cmake --build . --target tidy
  ```

  Then run the following command to format the code consistently:

  Linux Command: format

  ```
  cmake --build . --target format
  ```