Network Programming Project 3 (Part 1)
Remote Batch System

NP TA

Deadline: Sunday, 2018/12/16 23:59

1 Introduction

In this project, you are asked to write a remote batch system that runs over HTTP.

2 Requirements

1. All programs in this project **MUST** be implemented using **Boost.Asio**. Directly using low-level network related system calls (e.g. read, write, listen, accept, select ...) is **NOT** allowed.
2. You are asked to implement two programs in this project: **console.cgi** and **http_server**
3. All network operations (e.g. DNS query, connect, accept, send, receive ...) **MUST** be implemented with non-blocking (asynchronous) approaches.

3 Specification

3.1 console.cgi

1. You are highly recommended to inspect and run the CGI examples provided by TA before you start this part.
2. The **console.cgi** should parse the connection information (e.g. host, port, file) from environment variable **QUERY_STRING** set by HTTP server. The connection information stored in the **QUERY_STRING** is in the form of URL encoded HTTP GET parameters. For example:

   ```
   h0=nplinux1&p0=1234&f0=t1.txt&h1=npbsd2&p1=5678&f1=t2.txt&h2=&p2=&f2=&h3=&p3=&f3=&
   h4=&p4=&f4=
   ```

   Which means:

   ```
   h0=nplinux1      # the hostname or IP of the first server.
   p0=1234          # the port of the first server.
   f0=t1.txt        # the file that sent to the first server.
   h1=npbsd2        # the hostname or IP of the second server.
   p1=5678          # the port of the second server.
   f1=t2.txt        # the file that sent to the second server.
   h2=              # no third server, so this field is empty.
   p2=              # ...
   ```

```
f2=
h3=
p3=
f3=
h4=
p4=
f4=
```

3. Then, the **console.cgi** should connect to the servers (There are two servers in the example above) specified in the GET query parameters. (The maximum number of the shell sessions will not exceed 5)

4. In this project, the remote servers we connect to are shells with shell prompt "% ", and the files we want to send actually contain commands for the remote shells.
   You should not send the entire file to the remote server at once. Instead, send one command (line) from the file when you receive a shell prompt "% " from remote.

5. Your CGI program should display the results on remote servers in real time.
   i.e. Everything you send to remote or receive from remote should be displayed on the web page
   For example:

   ```
   % ls
   bin
   test.html
   ```

   The red part is the content (output) you received from the remote shell, and the blue part is the content (command) you sent to the remote.

6. Regarding how to beautifully display the shell, please refer to the CGI examples provided by TA. Since we will not judge your answers with **diff** for this project, feel free to style up the web page as you prefer. Just make sure the commands are displayed in the right order at the right time. And please make it easy to distinguish the **commands** from the **output of the shell**. (we use bold <b> for commands in the example)

3.2 http_server

1. The **http_server** accepts TCP connections and parse the HTTP requests.

2. In this project, the URI of HTTP requests will always only be in the form of /XXXXXX.cgi (e.g. /panel.cgi), and we will only test the HTTP GET method

3. The **http_server** should parse the HTTP headers and run XXXXXX.cgi with the CGI procedure (set environment variable, fork, dup, exec ...)

4. The following environment variables are required to set:

   (a) REQUEST_METHOD
   (b) REQUEST_URI
   (c) QUERY_STRING
   (d) SERVER_PROTOCOL
   (e) HTTP_HOST

(f) SERVER_ADDR

(g) SERVER_PORT

(h) REMOTE_ADDR

(i) REMOTE_PORT

3.3 panel.cgi (Provided by TA)

1. Written in Python 3. But sorry, you can not use Python in this project :(
2. This CGI generate the HTML form dynamically. (It detects all files in the directory **test_case** and list them in the HTML <select> menu)

3.4 test_case/

1. You can put all the test cases into this directory

3.5 Execution Flow

3.5.1 Initial Setup

The structure of your working directory:

```
working_dir
    |-----http_server       # HTTP server.
    |-----console.cgi       # Remote Batch System.
    |-----panel.cgi         # Provided by TA.
    |-----test_case/        # Put all test cases into this folder.
```

3.5.2 Execution

1. Run your **http_server** by **./http_server [port]**
2. Open a browser and visit `http://[NP_server_host]:[port]/panel.cgi`
3. Fill the form with the servers your want to connect to and select the input file then click **Run**.
4. The web page will be automatically redirected to `http://[NP_server_host]:[port]/console.cgi` and your Remote Batch System (console.cgi) should start now.

3.6 About Submission

1. E3

   (a) Create a directory named your student ID, put your files in the **same directory layer**.

   (b) You must provide a **Makefile**, which compiles your source code into two **executable** named **http_server** and **console.cgi**. The executables should be in the same directory as the source codes. We will use this executable for demo.

   (c) Upload **ONLY** your code and Makefile.
   Do **NOT** upload anything else (e.g. **.git**, **__MACOSX**, **panel.cgi**, **test_case/**...)

   (d) **zip** the directory and upload the .zip file to the E3 platform
   Attention!! we only accept .zip format

```
e.g.
Create a directory 0756000, the directory structure may be:
0756000
    |-- Makefile
    |-- http_server.cpp
    |-- console.cpp
    |...
zip the folder 0756000 into 0756000.zip, and upload 0756000.zip onto E3
```

2. Bitbucket:

   (a) Create a **private** repository: ${your_student_ID}_np_project3 inside the **nctu_np_2018** team, under **np_project3**.
   Set the ownership to **nctu_np_2018**

   ```
   e.g. 0756000_np_project3
   ```

   (b) For this project, you need to commit on Bitbucket for **at least 5 times**.

   (c) You can push anything you need onto Bitbucket as long as the size of the file is reasonable.

3. **We take plagiarism seriously.**

   ```
   All projects will be checked by a cutting-edge plagiarism detector.
   You will get zero points on this project for plagiarism.
   Please don't copy-paste any code from the internet, this may be
   considered plagiarism as well.
   Protect your code from being stolen.
   ```

4 Notes

1. NP project should be run on NP servers, otherwise, your account may be locked.
2. Any abuse of NP server will be recorded.
3. Don't leave any zombie processes in the system.
4. You will lose points for violating any of the rules mentioned in this spec.
5. Enjoy the project!

5 Hints and Reminders

1. If you want to test your CGI program before you finish your own HTTP server, you can use the HTTP server hosted on NP servers. Just place all CGI programs (the file name of the executables MUST end with .cgi) and **test_case/** into ~/public_html (A directory named "public_html" in your home directory, create it by yourself if it does not exist yet)
   Then, visit `http://[NP_server_host]/~[your_user_name]/XXXXXX.cgi`
   For example: `http://npbsd1.cs.nctu.edu.tw/~yhchen0906/panel.cgi`

2. You can use the command line tool **nc** to inspect the HTTP request send from the browser

   (a) execute the command **nc -l [port]** on one of the NP servers (ex. run **nc -l 8888** on nplinux3)

(b) open a browser and type in http://[host]:[port] in the URL, you can add some query parameters if you want.
For example:
http://nplinux1.cs.nctu.edu.tw:8888/test.cgi?a=b&c=d

3. You can try using C++11 <regex> library for string parsing.

4. Remember to recompile your code when you switch to another operating system (Linux ↔ BSD).