Program 1
HTML Tag and Byte Counter

Designed by Prof. Kredo

Due: By 23:59 Friday, September 18

Introduction

In this program you will accomplish several goals:

- Study the socket API client interface
- Develop a simple network client program

All programming assignments must be done in pairs.

Requirements

You may use the sample client socket program from Blackboard Learn course module as a base for this program. Even though I allow you to use the program, it is good practice to indicate you started with this code in your comments.

Your objective for this program is to write a program that requests a file from a web server and parses the data received to count the instances of the string '`<p>`' and the total number of bytes you receive from the server. In HTML files, values within angled brackets are called tags, so `<p>` is a `p` (paragraph) tag. You should ignore any tag that is not the exact three character sequence '`<p>`'. Your program must then display the total number of `p` tags and total number of bytes to the user. Your program requests the file by sending the string `"GET /~kkredo/file.html HTTP/1.0\r\n\r\n"` to port 80 on the host `www.ecst.csuchico.edu`. The HTML file for this program contains only text characters.

While processing the file, you must do so in chunks whose size in Bytes is determined by the user through a command line argument. The chunk size will always be a positive integer larger than the tag and no greater than 1000. **Points will be deducted for not processing data in chunks.** Importantly, `recv` may not give you all the data you requested, so always check return values. See 'Handling Partial send()s'[1] and other sections of Beej's Guide for details about `send` that also apply to `recv`.

Do not count `p` tags that span chunks. If `<` is at the end of one chunk and `p>` is at the start of the next chunk, then you do not count that tag. Thus, different chunk sizes will yield a different count of `p` tags.

Every student must complete and independent Group Evaluation form on Learn for each program, which may be used to adjust individual grades.

These are the general steps required for your program:

1. Retrieve the chunk size from the command line arguments
2. Connect to `www.ecst.csuchico.edu` on port 80
3. Send the string `"GET /~kkredo/file.html HTTP/1.0\r\n\r\n"`
4. Receive and process the file
   (a) Receive a chunk of data
   (b) Count the `p` tags in the chunk
   (c) Count the bytes received
   (d) Repeat until the file is done

[1] https://beej.us/guide/bgnet/html/#sendall

5. Print the number of `p` tags and total bytes

Additional requirements for this program include:

- All submissions must pass compilation checks before they will be graded.
- Submit your program files through Learn. Do not submit archive (zip/tar) files.
- You must include a Makefile with your submission, which will build the program by default and remove all generated files with the target `clean`. For Program 1, an example Makefile is provided on Learn.
- The executable must be called `p-counter` and accept one command line argument, the size in bytes, of the chunk used for counting tags.
- Your program must compile cleanly on Ubuntu 18.04 or `jaguar` and you must use the gcc/g++ argument `-Wall`.
- Check all function calls for errors. Points will be deducted for poor code.
- Put both group member names, the class, and the semester in the comments of all files you submit.

Input/Output

Below is an input and output example (with incorrect values). Make your programs match the style and formatting of these examples, but you are free to write your own error messages.

```
$ ./p-counter 10
Number of <p> tags: 20
Number of bytes: 511
$ ./p-counter 20
Number of <p> tags: 22
Number of bytes: 511
```

Evaluation

Your program will be evaluated based on:

- 10 points: Command line argument used for chunk size
- 10 points: Connection made to server
- 20 points: Request sent correctly
- 40 points: Correct count of `<p>` tags
- 20 points: Correct count of bytes received

Hints

- Beej's Guide to Network Programming[2] is a valuable resource.
- Test and debug in steps. Start with a subset of the program requirements, implement it, test it, and then add other requirements. Performing the testing and debugging in steps will ease your efforts. In general, the Grading requirements are ordered in a way to ease this implementation process.
- Your program should close after processing the entire file. The server closes the connection when the file is transferred, so `recv(2)` may be helpful. (HINT:"orderly shutdown" means to close a connection).
- You may have an optional command line argument that enables debugging output. The debug argument must be optional (meaning that your program must run when it is not specified) and the debug output in your code must be cleanly organized.
- For the declarations `char buff[256]; char *bp = buff;`, ensure you know the difference between `sizeof(buff)` and `sizeof(bp)`. **They are not equal!**

[2] https://beej.us/guide/bgnet/