COMS W3157 Advanced Programming, Lab #5

Please refer to our instructions for lab retrieval and submission. You must include the required information in your README.txt. This lab consists of multiple parts, so your code for each part must be in a subdirectory (named "part1", "part2", etc.). Like before, you are required to have 5 Git commits with meaningful commit messages, a Makefile for each part where you write code, and Valgrind output in your README.

Part 1: Web page downloader

There is a very useful program called "wget". It’s a command line tool that you can use to download a web page like this:

```
    wget http://www.gnu.org/software/make/manual/make.html
```

That will download the make manual page, make.html, and save it in the current directory. wget can do much more (downloading a whole web site, for example); see man wget for more info.

Your job is to write a limited version of wget, which we will call http-client, that can download a single file. You use it like this:

```
    ./http-client www.gnu.org 80 /software/make/manual/make.html
```

So you give the components of the URL separately in the command line: the host, the port number, and the URI. The program will download the given file and save it in the current directory. So in the case above, it should produce make.html in the current directory. It should overwrite an existing file named make.html.

Here are some hints and requirements:

- Your program must be implemented in a file named http-client.c.

- You will need to convert the host name into an IP address; you must use getaddrinfo() to do so. And don’t forget to use freeaddrinfo() to free the result returned by getaddrinfo().

- We forbid the use of memset() in this class because it can hide memory bugs that arise from logical errors. However getaddrinfo() expects all of the fields of the parameter struct addrinfo to be initialized. So for this lab only, we’re making an exception: you may use memset() to zero-initialize the struct addrinfo hint that you pass to getaddrinfo().

- The program should open a socket connection to the host and port number specified in the command line, and then request the given file using the HTTP/1.0 protocol.

- Make sure to include the following HTTP header in your request:

  ```
      Host: the.host.name.you.are.connecting.to:<port_number>
  ```

  Some web sites require it.

- Use "\r\n" rather than "\n" to terminate lines in your request, as required by HTTP.

- The first line of the server's HTTP response is the status line; if the status code is not 200, http-client should print the status line and exit.

- The status line will be followed by some HTTP response headers, a blank line, and then the actual file contents. http-client should skip over all headers and the blank line, and only save the file contents.

- Be aware that lines in the response will also be terminated by "\r\n" rather than "\n", including the blank line.

- You can use fdopen() to wrap the socket file descriptor with a FILE *, which will make it easier to read the response headers line by line.

  Make sure to create two separate FILE *s if you fdopen() the file descriptor for reading as well as writing, and don't forget to use dup().

- http-client must be able to download any type of file content from the server, not just HTML files.

  To check the integrity of downloaded images, you can use the cacaview tool to view images in the command line.

- The server will terminate the socket connection when it is done sending the file.

- You will need to obtain the file name from the URI (for example, make.html from /software/make/manual/make.html). Check out strrchr().

Part 2: Domain name resolver (optional)

(a)

Domain names are strings that help us identify hosts on the internet, and are resolved to IP addresses using DNS. Some domain names resolve to multiple IP addresses. For instance, CloudFlare is a widely used content delivery network that helps users access web content faster and more reliably; using the "host" tool, we can query its IP addresses:

```
$ host cloudflare.com
cloudflare.com has address 104.16.133.229
cloudflare.com has address 104.16.132.229
cloudflare.com has IPv6 address 2606:4700::6810:85e5
cloudflare.com has IPv6 address 2606:4700::6810:84e5

(( irrelevant output omitted ))
```

DNS servers will rotate between IP addresses when resolving domain names, providing a simple form of load balancing that mitigates any one host getting overwhelmed with queries. (You can even watch this rotation happen in real-time with "watch host reddit.com"! This command runs "host reddit.com" every two seconds.) DNS servers also provide IPv6 addresses for clients that support it.

Use the host tool to investigate the IP addresses of various domains, and record your observations in your README.

- See if you can find any other domains that resolve to multiple IP addresses. Also find popular domains that you would have expected to resolve to multiple IP addresses, but only resolve to one.

  - Note that some domains resolve to different IP addresses depending on where
    you perform the lookup (e.g., from your computer vs on CLAC). Can you find
    examples of these domains?

(b)

For this second part, you will build a basic version of the host tool that
resolves domain names, using the getaddrinfo() function:

```
    $ ./host cloudflare.com
    cloudflare.com has address 104.16.132.229
    cloudflare.com has address 104.16.133.229
    cloudflare.com has IPv6 address 2606:4700::6810:84e5
    cloudflare.com has IPv6 address 2606:4700::6810:85e5
```

Here are some hints:

  - The struct addrinfo that getaddrinfo() returns is actually a linked list of
    address information structures. You can access the next node of this list
    using the .ai_next field. Take a look at the man pages for getaddrinfo()
    for detailed examples of how to iterate through this linked list.

  - The IP addresses that getaddrinfo() returns are embedded in socket address
    structures as 4-byte (IPv4) and 16-byte (IPv6) integers, in network byte
    order. You can convert IPv4 addresses to dotted-quad notation using
    inet_ntoa(), but that function does not work for IPv6 addresses; instead,
    you should use inet_ntop().

    Make sure to check the .sa_family field of the socket address structure to
    determine whether it holds an IPv4 or IPv6 address.

--

Good luck!