Homework 9 C Programming - Parsing URL Strings

Due Wednesday 12/4/2019 11pm

In this assignment you will define a parse_url function that parses a given URL into three components: host, port and path. This homework is designed to help you get started with the proxylab since this will be the first major function you have to implement for the proxy lab to interpret a url to figure out what is the actual web server the proxy is supposed to connect to. Doing this homework will not be a waste of time, it will help you out with project 5:proxylab.

Download hw9handout.tar

Use instructions similar to homework 2 to download hw9handout.tar file, copy it to w204 machine's 311 folder, use `tar xvf` command to unpack it to hw9handout folder.

Function to implement

You are to implement a single function, see comments in file to find out what you are supposed to do.

```
// returns 1 if the url starts with http:// (case insensitive)
// and returns 0 otherwise
// If the url is a proper http scheme url, parse out the three
// components of the url: host, port and path and copy
// the parsed strings to the supplied destinations.
// Remember to terminate your strings with 0, the null terminator.
// If the port is missing in the url, copy "80" to port.
// If the path is missing in the url, copy "/" to path.
int parse_url(const char *url, char *host, char *port, char *path) {
  return 0;
}
```

Useful library functions

The following library functions could be helpful during your implementation. use the command

%**man functionName**

to learn more about how to use these library functions.

```
int strncasecmp(const char *s1, const char *s2, size_t n);
```

```
char *strncpy(char *dest, const char *src, size_t n);
```

Keep in mind that strncpy copies n characters from src to dest, but if that n characters don't have the null terminator, your destination string will not have the null terminator either, and you will have to manually put the null terminator in the dest string.

You can feel free to use any other string library functions too. Just make sure you read the man page carefully and understand what the functions do. And if your function also allocates data on the heap (such as strdup() that uses malloc()), you should also be responsible for cleaning up and freeing the memory when you are done. You may also choose to parse the string by reading one byte (one character) at a time.

Edit/Compile/Test your C code

1. Compile your code

Assuming you successfully copied and unpacked hw3handout.tar file in ~/311 folder on your W204 account, the following command will help you compile the given C program. The original tar file contains a complete C program that compiles and works.

```
cse-p204inst11.cse.psu.edu 160% cd ~/311/hw9handout
```

You use **make** command in the hw9handout folder to compile all the .c files to create an executable called **parse_url**.

Make sure that make does not generate any error/warning messages while compiling your C code. Any warning/error messages while compiling your code will result in a zero for this assignment. You must address these warnings/errors and remove all compiler warnings/error messages before you submit. To be safe, do the following before you submit to ensure no warnings/errormessages were produced.
%make clean
%make

2. Test your code

2.1 Testing with test functions provided in main.c

You start with code that contains functions that don't really do anything. If you test your code by executing ./parse_url before implementing anything, it will fail assertion of the first test function. When you successfully implemented all functions, your output should look like this:

```
[vagrant@localhost hw9]$ make
gcc -Wall -Wuninitialized -Og -g   -c -o parse_url.o parse_url.c
gcc -Wall -Wuninitialized -Og -g -o parse_url main.o parse_url.o
[vagrant@localhost hw9]$ ./parse_url
```

```
passed scheme test
passed parse_with_port test
passed parse_without_port test
```

You should look at the sample test in main.c to understand what you are expected to do.

- **host part** of the url is from after http:// to the first occurrence of either ':' or '/' or end of string.
- If the host part ended with a ':', then **port part** of url is from the character after that ':' to first occurrence of '/' or end of string; if the host part didn't end with a ':', then there is no port provided, you should use default port "80".
- If the host part or port part ended with a '/' then that till the end of the url is the **path part**. Otherwise, path part was missing and you should use default path "/".

3. Edit your code

Before you start, please make sure to write your name and email at the beginning of dict.c code to replace Prof. Wang's name and email.

```
// Author: Yanling Wang
// Email: yuw17@psu.edu
```

Edit your code in `parse_url.c` with vim and save it and repeat step 1/2 to compile and test your code.

Always start with small changes before your compile and test your code.

Keep your code properly indented.

Add comment in your function to explain your algorithm for this assignment. Use block comments that are on its own lines. Do not use tail comments at the end of each line of the code.

Submit your C code

You are to submit only the **parse_url.c** file to gradescope.