CS 3432 Computer Architecture I

Lab Assignment 1 – String Tokenizer

Teaching Assistant: Steven Ibarra

Instructional Assistant: Salvador Herrera Robles

Assigned: January 31, 2022

Due: February 13, 2022 @ 11:59pm

Introduction

String tokenizers are useful tools for breaking up strings into meaningful or digestible chunks (tokens). They can be used to separate .csv strings by comma to quickly gather comma-delimited data from a string. C has a built-in string tokenizer, but in this lab, you will be building your own.

Labs must be completed alone. You may discuss ideas and concepts with your peers but must be able to explain the functionality of your code. Questions will be asked to make sure the code was not plagiarized be prepared to explain your ideas.

Objectives

By the end of this lab, you will:

1. Have a basic understanding of strings and pointers in C
2. Have a basic understanding of memory addressing and allocation in C
3. Effectively explain your process and algorithm at a high and low level
4. Basic understanding of using bash commands
5. Basic understanding of using a Linux terminal
6. Basic understanding of using git and GitHub

Your Task

You are to design a function:

char** tokenize(char* str, char delim)

```
char** tokenize(char* str, char delim)
```

That will break up an input string into chunks according to the delimiter and return a char** that points to the series of tokens, or substrings, each containing a copy of a segment of the string. See the example below: 2 You are merely copying segments of the string into separate char pointers and not damaging or changing the original string in any way. In the example above, there are 2 spaces in the string, “Hello pretty world!” which forms 3 tokens, “Hello” “pretty” and “world!” which were copied and stored separately in memory with their char pointers being stored in succession starting at memory address 0x400c. Your **tokenize** function will a return a **char**** like seen above in **result**.

The major goal is to build the string tokenizer, but your code should have some way to showcase the tokenize function by asking for user input (the input string) and displaying the results of the tokenizer by iterating through the char** and printing each token line-by-line along with the total amount. A recommendation would build a pre-shell that would take a string input and then tokenize it into tokens.

```
[steven@Stevens-MBP Lab1 % gcc -o proTokenizer proTokenizer.c
[steven@Stevens-MBP Lab1 % ./proTokenizer
 Please enter the delimiter char:
 $$ ,
 Please enter the input string:
 $$ I,Attend,Utep,!
 Tokens[0]: I
 Tokens[1]: Attend
 Tokens[2]: Utep
 Tokens[3]: !
```

Hints

To build your tokenizer you will need to determine how long the string is and where the delimiters are. This will likely require multiple iterations through the string. A couple things you will need to figure out:

- How long is the string?
- How many tokens are there? (you will need to create a char* for each token)
- How many passes of the string do you need?
- Where is the start and end of a word?

You will need **malloc()** to allocate the precise amount of space for copying substrings over. Remember, **malloc** can be called like so:

```
char* new_string = (char*) malloc(str_size * sizeof(char))
```

; where you request a pointer to a location in memory with the specific number of bytes you need. This will be needed for storing the new strings and the sequence of **char*** that form the **char****.

Grading (50 total points)

1. Fully Implemented tokenize function (25 pts)
2. Demonstrates knowledge of the terminal (5 pts)
3. Demonstrate use of the Git commands (5 pts)
4. Submit clean code that is well documented (5 pts)
5. Demo code with great understanding within a week (10 pts)

Hand-In Procedures

This lab will be pushed onto your GitHub repository by the due date of February 13, 2022 @ 11:59pm**. Late Penalty**: Project is due on Sunday, February 13 @ 11:59pm. Once late, 0.5 point will be taken off every hour up to the first 10 hours on the FIRST DAY ONLY . After this, 5 points will be taken off per day late

Demos

Your code will need to be demoed within 1 weeks of your submission. In the demo, you will walk through your solution and may need to answer questions from Steven, and Salvador about aspects of your code. Please mention any troubles you encountered during the lab and resources you used (online tutorials, the book, etc.). We anticipate most demos will take roughly 10-15 minutes but may take longer if there are numerous questions about the assignment or your solution.