1 Problem Description

Implement the standard library function `strtok` using C++/C basic language constructs. In concept, this assignment is similar to that of Lab two where we have implemented two simpler library functions `strlen` and `strcat`. To develop a solution for a problem, we must first understand the problem. To implement `strtok`, we must first understand `strtok`. In Lab two you have learned a little about how `strtok` works. You are welcome to experiment with it more if there is anything that you are not sure about.

Typing "man strtok" on the prompt in a linux machine such as turing, we get the following. Try to read the documentation to see if it makes sense to you.

```
wingning@turing:~/csce3143$ man strtok
STRTOK(3) Linux Programmers Manual STRTK(3)

NAME
strtok, strtok_r — extract tokens from strings

SYNOPSIS
#include <string.h>
char *strtok(char *str, const char *delim);
char *strtok_r(char *str, const char *delim, char **saveptr);

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):
strtok_r(): _SVID_SOURCE || _BSD_SOURCE || _POSIX_C_SOURCE || _XOPEN_SOURCE

DESCRIPTION
The `strtok()` function parses a string into a sequence of tokens. On the first call to `strtok()` the string to be parsed should be specified in `str`. In each subsequent call that should parse the same string, `str` should be NULL.

The `delim` argument specifies a set of characters that delimit the tokens in the parsed string. The caller may specify different strings in `delim` in successive calls that parse the same string.

Each call to `strtok()` returns a pointer to a null-terminated string containing the next token. This string does not include the delimiting character. If no more tokens are found, `strtok()` returns NULL.

A sequence of two or more contiguous delimiter characters in the parsed string is considered to be a single delimiter. Delimiter characters at the start or end of the string are ignored. Put another way: the tokens returned by `strtok()` are always nonempty strings.

The `strtok_r()` function is a reentrant version `strtok()`. The saveptr argument is a pointer to a char * variable that is used internally by `strtok_r()` in order to maintain context between successive calls that parse the same string.
On the first call to `strtok_r()`, `str` should point to the string to be parsed, and the value of `saveptr` is ignored. In subsequent calls, `str` should be `NULL`, and `saveptr` should be unchanged since the previous call.

Different strings may be parsed concurrently using sequences of calls to `strtok_r()` that specify different `saveptr` arguments.

**RETURN VALUE**
The `strtok()` and `strtok_r()` functions return a pointer to the next token, or `NULL` if there are no more tokens.

**CONFORMING TO**

- `strtok()`
  - SVr4, POSIX.1-2001, 4.3BSD, C89, C99.
- `strtok_r()`
  - POSIX.1-2001

**BUGS**
Avoid using these functions. If you do use them, note that:

- These functions modify their first argument.
- These functions cannot be used on constant strings.
- The identity of the delimiting character is lost.

The `strtok()` function uses a static buffer while parsing, so it's not thread safe. Use `strtok_r()` if this matters to you.

**EXAMPLE**
The following program uses nested loops that employ `strtok_r()` to break a string into a two-level hierarchy of tokens. The first command-line argument specifies the string to be parsed. The second argument specifies the delimiter character(s) to be used to separate that string into "major" tokens. The third argument specifies the delimiter character(s) to be used to separate the "major" tokens into subtokens.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[])
{
    char *str1, *str2, *token, *subtoken;
    char *saveptr1, *saveptr2;
    int j;

    if (argc != 4) {
        fprintf(stderr, "Usage: %s string delim subdelim\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    for (j = 1, str1 = argv[1]; ; j++, str1 = NULL) {
        token = strtok_r(str1, argv[2], &saveptr1);
        for (; token; token = strtok_r(token, argv[3], &saveptr2))
            fprintf(stdout, "%s\n", token);
    }
}
```
if (token == NULL)
    break;
printf("%d: %s\n", j, token);

for (str2 = token; ; str2 = NULL) {
    subtoken = strtok_r(str2, argv[3], &saveptr2);
    if (subtoken == NULL)
        break;
    printf(" --> %s\n", subtoken);
}
}
exit(EXIT_SUCCESS);
} /* main */

An example of the output produced by this program is the following:

$ ./a.out a/bbb///cc;xxx:yyy: :; /
1: a/bbb///cc
   --> a
   --> bbb
   --> cc
2: xxx
   --> x
3: yyy
   --> y

SEE ALSO
    index(3), memchr(3), rindex(3), strchr(3), strpbrk(3), strsep(3), strspn(3), strstr(3), wcstok(3)

COLOPHON
    This page is part of release 2.77 of the Linux man-pages project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages.

GNU 2007-07-26 STRTOK(3)
Manual page strtok(3) line 87/134 (END)

Please pay attention to the function prototype. This is a complicated function for us so a discussion has been provided in the lecture. The example code in the document page is also quite complicated for us perhaps for now. So it is fine if you are not able to understand the example code or want to skip it, but we hope we can understand it at the end of the semester when we complete the course. We will come back to it.

2 Purpose

Understand pointer and array, and in particular pointer to char, array of char, and C-style string. Practice writing functions that are similar to standard library functions to show our ability and command of basic data structures and programming in C/C++ and to build our confidence. Learn the function strtok.
Learn static storage class.

3 Design

For this part, we will follow what is provided in the documentation except for the name. We could use the identical name as used in the standard library if we create our own namespace, otherwise name conflict or clash may result. We will prefix the name with my.

\[
\text{char *mystrtok(char *str, const char *delim);}
\]

4 Implementation

We assume that someone (you or other folks) will use our mystrtok in their main program. We will create two files: mystrtok.h and mystrtok.cpp. Conceptually, mystrtok.h is the header file for any code that uses our function to include just as we have to include other headers in order to use the library functions or types. The mystrtok.cpp file is used to create the object file, say mystrtok.o, to be linked together by the compiler to create the executable file. We will have another file called homework1.cpp that contains the main function for testing our implementation of strtok, that is mystrtok function. Here is the idea or hint about how to implement strtok.

1. introduce a static pointer variable to remember where we left off.

2. decide whether to use the first augment value or the value remembered to search for the next token.

3. find the next token (from the current beginning position search for a character that is not a delimiter, which is where the token starts; from that place search for a character that is a delimiter or null character, which is where the token ends after changing the character to null character)

4. make sure to update the static variable, and return the address (which is determined in the previous step) of the first character of the token.

5 Test and evaluation

Our main program should have several test cases for different conditions of using strtok or mystrtok function. We will use the main program of lab two as our first test case. Then We need to think about other test cases, for example, what if the first parameter of strtok does not contain any delimiter at all. Or the delimiter string contains more than one delimiters. We hope to design the test cases such that each test case covers a new situation yet considered.
6 Report and documentation

A short report about things observed and things learned and understood. The report should also describe the test cases used in the main program and the reasons for each test case selected. Properly document and indent the source code. The source code must include the author name and as well as a synopsis of the file.

7 Lab submission

Use the same website (http://csce.uark.edu/mmoccaro/2014/F13/) and procedure as you have done to submit your lab.