CS 521: Systems Programming

String Tokenization

Lecture 7

Tokenization

- Looking at Lab 4, you might be wondering if there are other ways to deal with strings in C
  - In particular, splitting them up (***tokenizing*** them)
- Consider the string: "Hello, how are you today?"
  - How can we retrieve each word individually?
  - [Hello,] [how] [are] [you] [today?]
- In Java/Python, we have split()… Does C have an equivalent?
- The answer: **yes**
  - The better answer: **yes**, *but…*

Approaches

- `strtok`
- `strtok_r`
- `strsep`
- `next_token`

strtok

- One of the classic approaches to string tokenization is called `strtok`
- `strtok` works. But it does have some pretty major issues.

Using strtok

```c
/* Tokenize based on space and newline characters: */

char line[] = "Here is my amazing line of text!";
char *token = strtok(line, " \n");
while (token != NULL) {
        /* do something with token */
        /* then grab the next token: */
    printf("-> %s\n", token);
        token = strtok(NULL, " \n");
}
```

How strtok works [1/2]

- When it comes to C functions, strtok is one of the stranger ones
- First, we pass in the string we want to tokenize
- After that, we pass in `NULL` and it gives us the next token
- How does it even know what string to operate on?
  - `strtok` maintains a global pointer to the start of the most recent token

Global vs. Local State

- In C, we have global and local variables
- Globals are defined outside of any function
  - For example, up above your main function
- Some C library functions even do this
  - When you #include them, they get added to your code
  - C provides the static keyword to restrict global variables' scope to their compilation unit (file)
- This way we don't pollute the global namespace

How strtok works [2/2]

- Beyond the strange pointer magic, we also need to know how strtok splits strings up
- It scans through the string until it comes across one of the defined delimiters
- The delimiter is replaced with `\0`
- Now printing the string only prints up to the `NUL` byte
- To move to the next token, `strtok` simply changes the pointer to come after the `NUL` byte!
  - We should sketch this out. The other approaches are roughly the same

Why strtok is bad

- The global state means that `strtok` is **NOT** reentrant
  - Can only be used in one place at a time

```
while ( ... ) {
    strtok( ... );
    while ( ... ) {
        strtok( ... ); /* No!!!!!! */
    }
}
```

- Beyond not being reentrant, it's also not thread safe.
- Use it in a library you're writing? You have to warn EVERYONE not to use `strtok` while you are 🤮

strtok_r

- There **is** a reentrant version of `strtok` available, though not in all C libraries
- It's a good choice to use by default (assuming it's available)
  - Even if you don't think you'll need reentrancy or thread safety

strsep

- Ding ding! Next contender: `strsep`
- `strsep` is non-standard, but still usually available in most C libraries
- Let's take a look at an example…

Using strsep

```c
/* Tokenize based on space and newline characters: */
char *line = malloc(128 * sizeof(char));
strcpy(line, "Here is my amazing line of text!");

char *token;
while ((token = strsep(&line, " \n")) != NULL) {
    printf("-> %s\n", token);
}
/* DANGER: MEMORY LEAK! (and a sneaky one at that) */
```

- Huh, what's different about this compared to `strtok`?
  - Also: `strsep` will produce empty strings for delimiters that are located next to one another in the string, whereas `strtok` skips over all the delimiters

next_token [1/2]

- Our next option is `next_token`. It is **NOT** part of any C library, but quite commonly used:
  - (there's a copy on the schedule page)

```
/* Tokenize based on space and newline characters: */

char str[] = "Here is my amazing line of text!";
char *next_tok = str;
char *curr_tok;
while ((curr_tok = next_token(&next_tok, " \n")) != NULL) {
    printf("-> %s\n", curr_tok);
}
```

next_token [2/2]

- Replicates `strtok` using `strspn` and `strcspn`
  - Will skip over several delimiters in sequence rather than returning empty strings like `strsep`
- Thread safe, reentrant
- You have to include it manually in your project 😢