Week 3 Tuesday

Strings I

Fill in the check-in form on cs107a.stanford.edu!
Announcements

- Please make sure you’re on Slack and can receive my communications! Some of y’all are ghosting me and I’m sad
- Sign up for a 1:1! https://calendly.com/adbenso/ct107a-1-1 (Starts this weekend, else probably Mon/Wed next week)
- Pls enroll, you can always drop later
  - Add/drop deadline is Friday
  - Before Friday (and before 1:1s) I’m happy to chat if there’s anything you want to discuss ahead of the add/drop deadline
Unix Tip Spotlight

- How do you clear the screen?
  - clear
    - Does what it says on the tin
  - <CTRL+L>
    - Does the same thing as clear
    - Doesn’t involve an actual command, so it’s more...aesthetic?
  - Neither of these necessarily deletes the screen output – your terminal probably allows you to scroll up to see the cleared output!
Agenda

- Chars and Strings as Arrays of Chars
- String Functions
- Debugging `buggy_strstr`
- Practice: mini-exercises with string functions
- Practice: `filter_pollution`
Chars and Strings as Arrays of Chars
Chars

- 1 value, 2 interchangeable interpretations
- Interpretation 1: 8-bit integer value (-128 to 127, or 0 to 255)
- Interpretation 2: A particular character in the ASCII table
  - `man ascii` shows you an ASCII table
- `'e' - 'a' == 4` // true!
Important Chars

- The NUL character `\0` (value: 0)
- The newline character `\n` (value: 10)
- The space character ` ` (value: 32)
- The digit `0` (value: 48)
- Uppercase `A` (value: 65)
- Lowercase `a` (value: 97)
Strings [of characters]

- Strings are just contiguous sequences ("arrays") of characters in memory
- Valid strings need to be terminated by a NUL character
  - Without it, they're still char arrays, just not strings
- Num bytes used = string length + 1
- We'll discuss more about this (and how to create strings) on Thursday
Strings

- What does “” look like in memory?
- What does “he
  ll” look like in memory?
String Functions
<table>
<thead>
<tr>
<th>Operation</th>
<th>How to do it</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access some character in the middle</td>
<td>Array indexing</td>
</tr>
<tr>
<td>Get length of string</td>
<td>strlen</td>
</tr>
<tr>
<td>Check [sub]string equality</td>
<td>strcmp, strncmp</td>
</tr>
<tr>
<td>Compare strings</td>
<td>strcmp, strncmp</td>
</tr>
<tr>
<td>Copy strings into existing buffer</td>
<td>strcpy, strncpy</td>
</tr>
<tr>
<td>Find index of first instance of character</td>
<td>strchr</td>
</tr>
<tr>
<td>Find index of first instance of substring</td>
<td>strstr</td>
</tr>
<tr>
<td>Append string to another, if room</td>
<td>strcat, strncat</td>
</tr>
<tr>
<td>Split a string based on delimiters</td>
<td>strtok</td>
</tr>
<tr>
<td>Parse a string based on accept/reject lists</td>
<td>strspn, strncspn</td>
</tr>
</tbody>
</table>
Examples

- Array indexing
  - `s[5] = 'a'; char c = s[0];`

- Get length of string
  - `for (int i = 0; i < strlen(s); i++) {`

- String equality
  - `if (strcmp(s, t) == 0) {`

- String comparison
  - `if (strcmp(s, t) < 0) {`
    - `printf("s is alphabetically before t");`
  }

- String copying
  - `char buf[6]; strcpy(buf, "hello");`
Examples

- **Find index of first instance of character**
  - char *first_r = strchr("arrillaga", 'r');

- **Find index of first instance of substring**
  - char *where_llaga = strstr("arrillaga", "llaga");

- **Append strings**
  - char buf[13]; buf[0] = '\0';
    - strcat(buf, "hello "); strcat(buf, "world!");

- **Split a string based on delimiters**
  - Just don’t use `strtok` ever please

- **Parse a string based on accept/reject lists**
  - char *ptr = " ....and then i oop...";
    - ptr += strspn(ptr, " . ");
    - ptr += strcspn(ptr, " . ");
git clone /afs/ir/class/archive/cs/cs107a/cs107a.1226/WWW/exercises/strings1
Debugging buggy_strstr
Practice: mini-exercises with string functions
Write a 1-line boolean expression involving a valid string (char*) \( s \) for the following:

- \( s \) contains the substring "107"
- \( s \) is the empty string
- \( s \) is equal to the string \( t \)
- \( s \) does not contain ' # '
- \( s \) is the same length as the string \( t \)
- \( s \)'s second character is the same as the string \( t \)'s third (assume both strings have enough characters)
- \( s \) is a prefix of \( t \) (assume \( t \) is much, much longer)
- (hard) the first 5 characters of \( s \) are digits (0-9)
Write a 1-line boolean expression involving a valid string (char*) `s` for the following:

- `s` contains the substring "107": `strstr(s, "107") != NULL`
- `s` is the empty string
- `s` is equal to the string `t`
- `s` does not contain ' # '
- `s` is the same length as the string `t`
- `s`'s second character is the same as the string `t`'s third (assume both strings have enough characters)
- `s` is a prefix of `t` (assume `t` is much, much longer)
- (hard) the first 5 characters of `s` are digits (0-9)
Write a 1-line boolean expression involving a valid string (char*) s for the following:

- s contains the substring "107"  `strstr(s, "107") != NULL`
- s is the empty string  `s[0] == '\0'`
- s is equal to the string t
- s does not contain '#'`
- s is the same length as the string t
- s's second character is the same as the string t's third (assume both strings have enough characters)
- s is a prefix of t (assume t is much, much longer)
- (hard) the first 5 characters of s are digits (0-9)
Write a 1-line boolean expression involving a valid string (char*) `s` for the following:

- `s` contains the substring "107" `strstr(s, "107") != NULL`
- `s` is the empty string `s[0] == '\0'`
- `s` is equal to the string `t` `strcmp(s, t) == 0`
- `s` does not contain ' #'`
- `s` is the same length as the string `t`
- `s`'s second character is the same as the string `t`'s third (assume both strings have enough characters)
- `s` is a prefix of `t` (assume `t` is much, much longer)
- (hard) the first 5 characters of `s` are digits (0-9)
Write a 1-line boolean expression involving a valid string (char*) s for the following:

- s contains the substring "107"  `strstr(s, "107") != NULL`
- s is the empty string  `s[0] == '\0'`
- s is equal to the string t  `strcmp(s, t) == 0`
- s does not contain '#'  `strchr(s, '#') == NULL`
- s is the same length as the string t
- s's second character is the same as the string t's third (assume both strings have enough characters)
- s is a prefix of t (assume t is much, much longer)
- (hard) the first 5 characters of s are digits (0-9)
Write a 1-line boolean expression involving a valid string (char*) s for the following:

- s contains the substring "107"  `strstr(s, "107") != NULL`
- s is the empty string  `s[0] == '\0'`
- s is equal to the string t  `strcmp(s, t) == 0`
- s does not contain '#'  `strchr(s, '#') == NULL`
- s is the same length as the string t  `strlen(s) == strlen(t)`
- s's second character is the same as the string t's third (assume both strings have enough characters)
- s is a prefix of t (assume t is much, much longer)
- (hard) the first 5 characters of s are digits (0-9)
Write a 1-line boolean expression involving a valid string (char*) s for the following:

- s contains the substring "107"  \( \text{strstr}(s, \text{"107"}) \neq \text{NULL} \)
- s is the empty string  \( s[0] == '\\0' \)
- s is equal to the string t  \( \text{strcmp}(s, t) == 0 \)
- s does not contain '#'  \( \text{strchr}(s, \text{"#'}) == \text{NULL} \)
- s is the same length as the string t  \( \text{strlen}(s) == \text{strlen}(t) \)
- s's second character is the same as the string t's third (assume both strings have enough characters)  \( s[1] == t[2] \)
- s is a prefix of t (assume t is much, much longer)
- (hard) the first 5 characters of s are digits (0-9)
Write a 1-line boolean expression involving a valid string (char*) s for the following:

- s contains the substring "107" `strstr(s, "107") != NULL`
- s is the empty string `s[0] == '\0'`
- s is equal to the string t `strcmp(s, t) == 0`
- s does not contain '#' `strchr(s, '#') == NULL`
- s is the same length as the string t `strlen(s) == strlen(t)`
- s's second character is the same as the string t's third (assume both strings have enough characters) `s[1] == t[2]`
- s is a prefix of t (assume t is much, much longer) `strncmp(s, t, strlen(s))`
- (hard) the first 5 characters of s are digits (0-9)
Write a 1-line boolean expression involving a valid string (char*) `s` for the following:

- `s` contains the substring "107" 
  \[\text{strstr}(s, "107") \neq \text{NULL}\]
- `s` is the empty string 
  \[s[0] == '\0'\]
- `s` is equal to the string `t` 
  \[\text{strcmp}(s, t) == 0\]
- `s` does not contain '#' 
  \[\text{strchr}(s, '\#') == \text{NULL}\]
- `s` is the same length as the string `t` 
  \[\text{strlen}(s) == \text{strlen}(t)\]
- `s`'s second character is the same as the string `t`'s third (assume both strings have enough characters) 
  \[s[1] == t[2]\]
- `s` is a prefix of `t` (assume `t` is much, much longer) 
  \[\text{strncmp}(s, t, \text{strlen}(s))\]
- (hard) the first 5 characters of `s` are digits (0-9) 
  \[\text{strspn}(s, "0123456789") \geq 5\]
Practice: filter_pollution