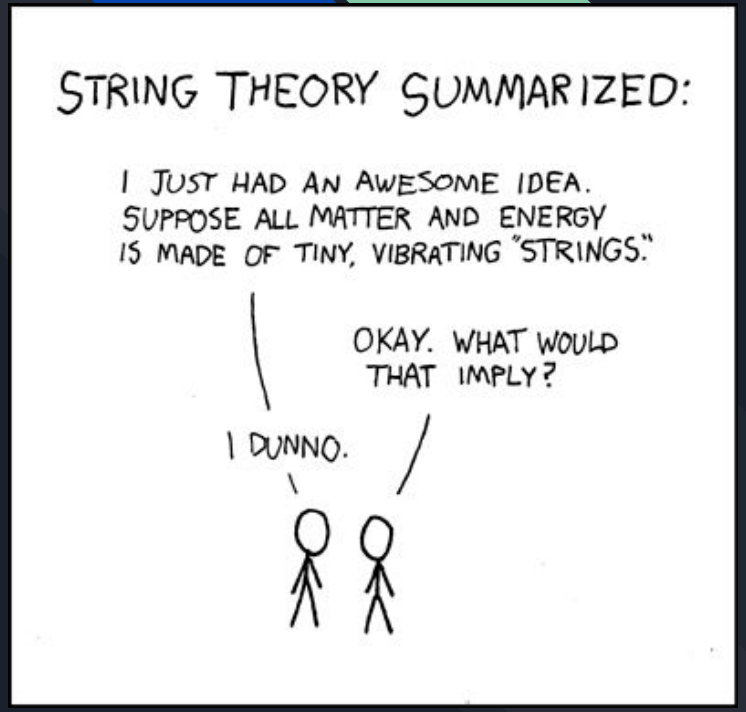


https://xkcd.com/171/

Section 6
More C-Strings

CS 107A, Autumn 2021
Andrew Benson (adbenson@)

Don’t forget to start recording

Unix Tip Spotlight

- Tab completion
  - Use it for programs you're running!
  - Use it for filenames!
  - Use it in gdb!
  - Just mash <TAB>, what's the worst that could happen

Announcements

- Friday 5pm: Add / Drop deadline
  - Please reach out if you have any questions
- 1:1s (Thursday, Friday, Monday): https://calendly.com/adbenson/cs107a-1-1
- assign2 walkthrough out

Bird's Eye View

Day	Week 3 Wednesday	Thursday	Friday	Week 4 Monday	Tuesday	Wednesday	Thursday	Friday
CS 107A		Section: More C-Strings			Section: Pointers and Memory		Section: Stack and Heap	
CS 107	Lab 2: C-Strings		Lecture: Arrays and Pointers	Lecture: Stack and Heap		Lab 3: Arrays / Pointers		Lecture: `void*`, Generics
CS 107 assignments	assign1 due, assign2 released					assign2 due, assign3 released		

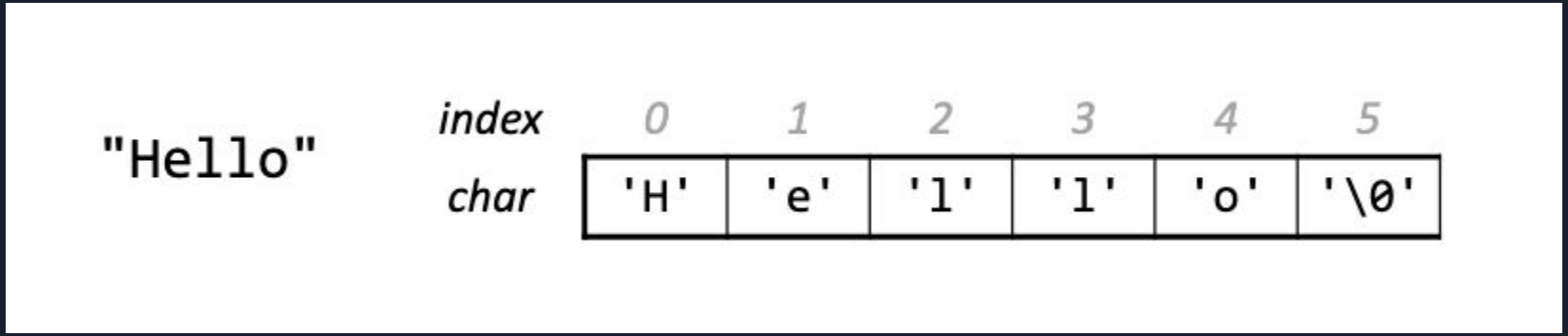
Agenda

- Creating Strings
- String Practice 1
- Strings in Memory
- String Practice 2
- Coding Practice

Creating Strings

Strings are Arrays

- The array can be in the stack, heap (Monday), or in the data segment
- All valid C strings have NUL terminators, whether you see them explicitly or not
- strlen(my_string) returns the length NOT including the NUL terminator



Strings in the data segment

- Read-only
- The initial string involved a literal string (double-quoted string) not assigned to an array variable

```
char *str = “hello”;
char *str2 = str;
char c = str[1];
char c2 = ‘e’;
```

- What is the value of each variable?
- What in the code is read-only?
- What is `strlen(str2)`?
- How many bytes does str take up in memory?

Strings on the stack

- Modifiable, like any stack variable
- Involves a stack array

- Why not `char s11[];` ?
- Why might you want more space than your string takes up?

```
char s[] = “ab”;
char s2[3] = “ab”;
char s3[3409] = “ab”;
char s4[] = {‘a’, ‘b’, ‘\0’};
char s5[3] = {‘a’, ‘b’, ‘\0’};
char s6[3409] = {‘a’, ‘b’, ‘\0’};
char s7[3]; strcpy(s7, “ab”);
char s8[3409]; strcpy(s8, “ab”);
char s9[3]; strncpy(s9, “ab”, 3);
char s10[3409]; strncpy(s10, “ab”, 3);
```

String Practice 1

What's wrong with these code snippets?

```
char *s = "stocks";
s[3] = 'n';
printf(s);
```

```
char s[] = {'a', 'b'};
s[0] = 'A';
printf("%s\n", s);
```

Based on practice problems I originally wrote for CMU 15-122 Fall 2015 Recitation 12

What's wrong with these code snippets?

```
char *s = "stocks";
s[3] = 'n';
printf(s);
```

This modifies a read-only string! Undefined behavior.

```
char s[] = {'a', 'b'};
s[0] = 'A';
printf("%s\n", s);
```

s is not a valid string – there's no NUL terminator! Undefined behavior.

Based on practice problems I originally wrote for CMU 15-122 Fall 2015 Recitation 12

What's wrong with these code snippets?

```
char s[4];
strcpy(s, "stanford cs107a");
printf("%s\n", s);
```

```
char s[] = 'work';
s[1] = "e";
printf("%s\n", s);
```

What's wrong with these code snippets?

```
char s[4];
strcpy(s, "stanford cs107a");
printf("%s\n", s);
```

`s` is not big enough for the string copied in! Overwrites unallocated memory - undefined behavior.

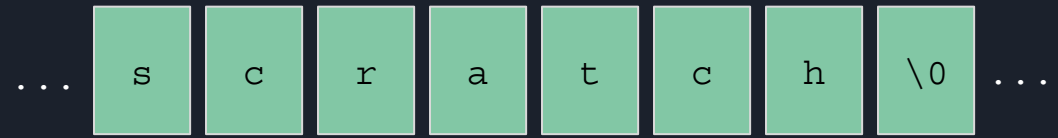
```
char s[] = 'work';
s[1] = "e";
printf("%s\n", s);
```

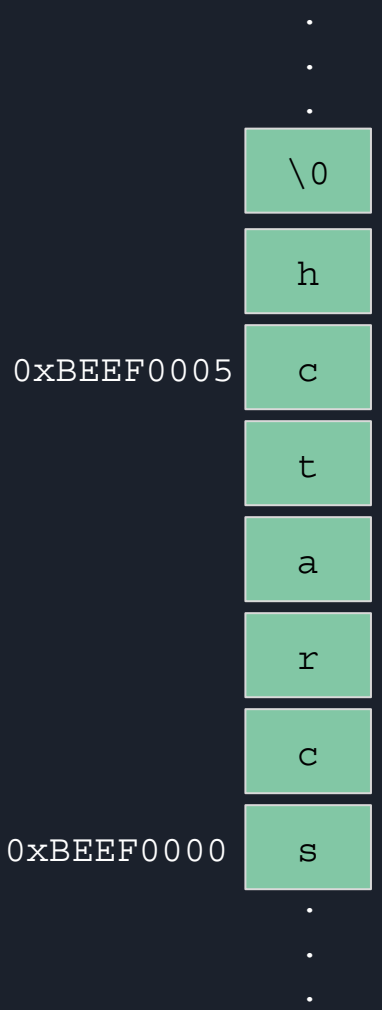
Doesn't compile! Strings use double quotes, chars use single quotes.

Strings in Memory

Memory Diagrams

```
char *t = “scratch”;
char *t2 = t+2;
char *t3 = t+6;
char *t4 = t+7;
char *t5 = t+8;
```





Using GDB to examine memory

String Practice 2

String Pointers

```
char *foo(char *s, char *t) {
  s[0] = t[2];
  t[2] = 'g';
  s = t;
  t = NULL;
  return s+1;
}
```

```
int main() {
  char s[] = "abc";
  char t[] = "def";
  printf("%s\n", foo(s, t));
  printf("%s\n", s);
  return 0;
}
```

Coding Practice

Section 6 Worksheet

```
git clone /afs/ir/class/cs107a/WWW/git/section6
```