

Objectives - Thu 2/3/2022 *

- Strings
- Memory
- Variadic functions

* This video lecture was posted Sat 2/5/2022 in lieu of the in-person lecture that was originally scheduled for Thu 2/3/2022.

Strings

A string is a sequence of characters.

In C:

- String is an array of char.

- In memory, it is followed by a null terminator byte.

'\0' value 0 in ASCII

Declare (and initialize) a string

There are two ways to declare.

String
on
stack
segment

```
char s_on_stack[] = "abc";
```

String
on
data
segment

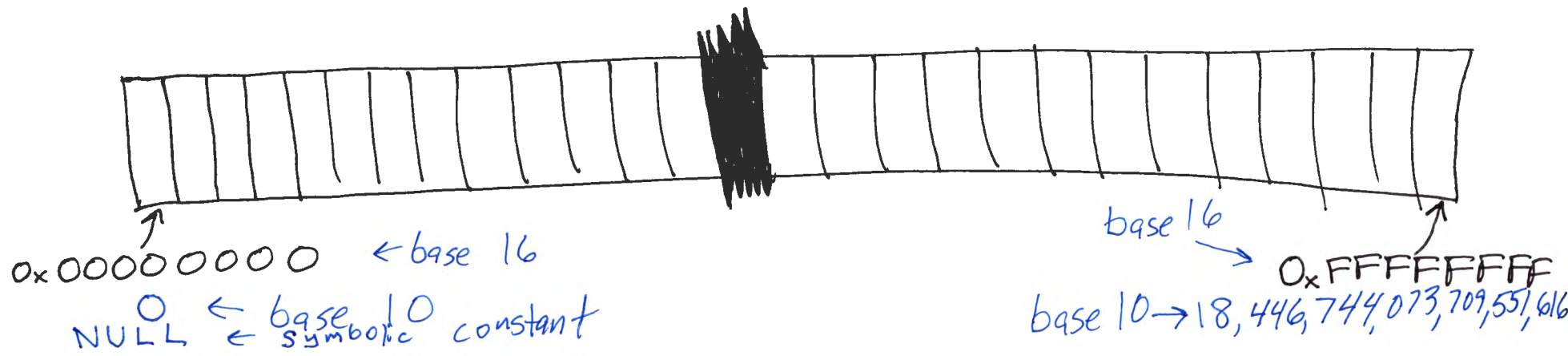
```
char* s_on_data_segment = "abc"
```

Memory

- Just a sequence of bytes.

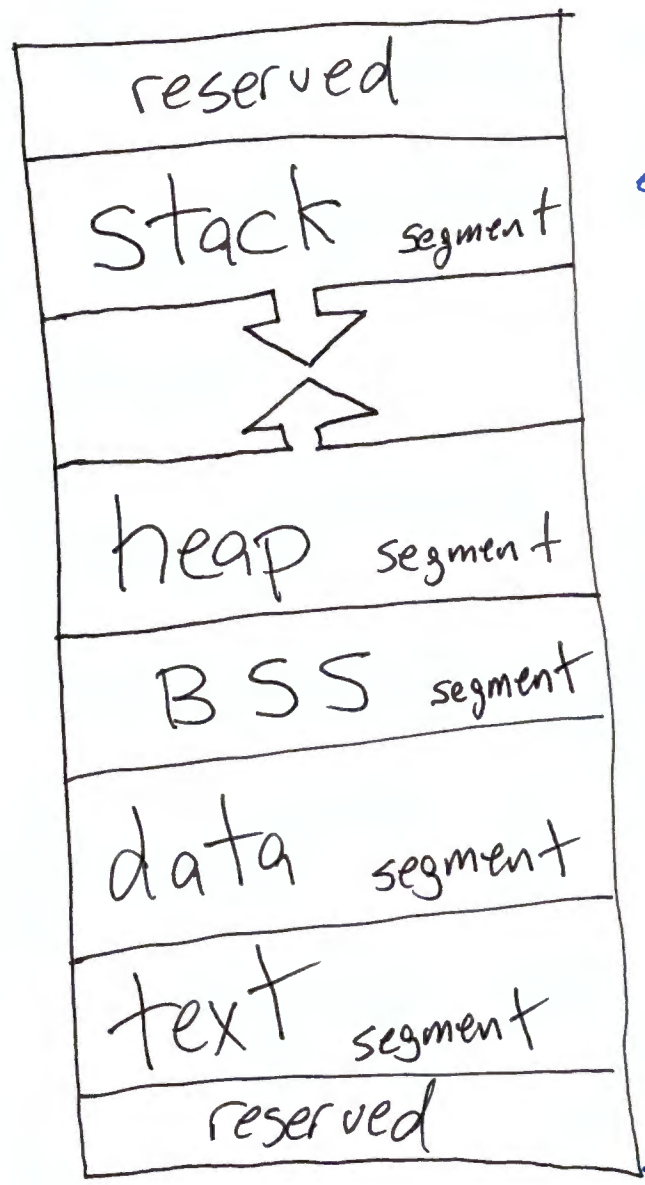
It's all just bytes.

- Every location in memory has an address.



Memory segments

See reference sheet page 2 under "memory".



← off limits

← any local variable or parameter ^{with a name} in a function

← Arrays and other data structures the size of which is determined at runtime (i.e. as the program runs). (See HW09 smin+fc.)

← Global variables that are not initialized
`int g_num_rabbits;`
`int main() {`
`3`
⚠ Do not use in ECE 264.

← String constants in double quotes ("")
Global variables that are initialized.
⚠ Do not use global variables in ECE 264, except constants following Codequality standards.

← your compiled code


← off limits

← NULL (memory address 0x00000000)

Stack

addr	type*	name*	value	part	fn
200	int	argc	1	args	main(...)
204	char**	argv	→ {"/foo"}		
212	void*			ret addr	
220				locals	

Heap

addr	value	
400		

Data segment

addr	type*	value
600		

Type and name are not actually stored in memory or executable. Addresses shown are fictional. Assume `sizeof(int)==4`, `sizeof(char)==1`, `sizeof(void*)==8`.

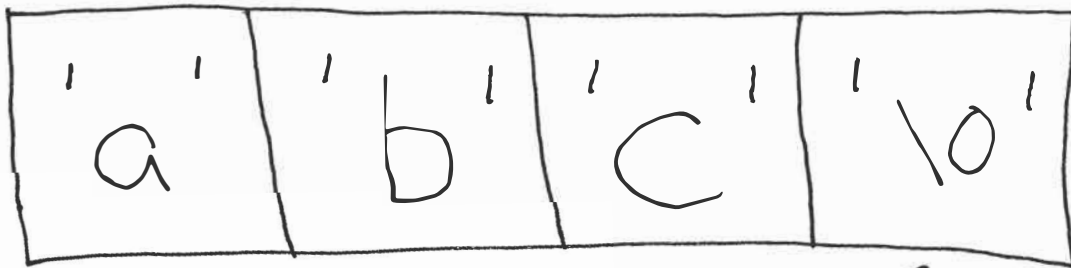
NULL (memory address constant)

- Symbolic constant equal to 0.
- Memory address 0
- This location may not be accessed by your code
 - You would get a "Segmentation Fault" (aka "segfault")
- Not needed in HW05 (mintf(-))
- Used to mean "nothing" or "empty" in code that uses dynamic data structures.
- You will use NULL starting in HW09 (smintf(-))

String on stack segment

```
char s_on_stack[] = "abc";
```

This is literally just an array of type char.



See reference sheet page 2 under "strings".

↑
null terminator byte
(value 0 in ASCII)

string on stack segment

- Use when you might want to change contents of string.

```
char s[] = "tart"
```

```
s[0] = 'd';
```


```
printf("%s", s); // → dart
```

⚠ Can be tricky to assign to new variables or pass to functions.

Stack

addr	type*	name*	value	part	fn
200	int	argc	1	args	main(...)
204	char**	argv	→ {"/foo"}		
212	void*			ret addr	
220				locals	

Heap

addr	value	
400		

Data segment


addr	type*	value
600		

Type and name are not actually stored in memory or executable. Addresses shown are fictional. Assume sizeof(int)==4, sizeof(char)==1, sizeof(void*)==8.

Stack

addr	type*	name*	value	part	fn
200	int	argc	1	args	main(...)
204	char**	argv	→ {"/foo"}		
212	void*			ret addr	
220	char [4]	s_on_stack	'a' 'b' 'c' 'o'	locals	
224					

Heap

addr	value	
400		

Data segment

addr	type*	value
600		

Type and name are not actually stored in memory or executable. Addresses shown are fictional. Assume sizeof(int)==4, sizeof(char)==1, sizeof(void*)==8.

string on data segment

- Use when the string contents are constant.

- Easy to assign to new variables.

```
char * s = "abc";
```

```
char * t = s;
```

TIP: Use strings on data segment for HW 05 to keep life easy.

- char * means "address of a char".


It is the address of the first character in the string in memory.

⚠ You cannot change contents of a string on the data segment.

Stack

addr	type*	name*	value	part	fn
200	int	argc	1	args	main(...)
204	char**	argv	→ {"/./foo"}		
212	void*			ret addr	
220				locals	

Heap

addr	value	
400		

Data segment


addr	type*	value
600		

Type and name are not actually stored in memory or executable. Addresses shown are fictional. Assume `sizeof(int)==4`, `sizeof(char)==1`, `sizeof(void*)==8`.

Stack

addr	type*	name*	value	part	fn
200	int	argc	1	args	main(...)
204	char**	argv	→ {"/foo"}		
212	void*	 	 	ret addr	
220	<i>char *</i>	<i>s-on-data-segment</i>	<i>600</i>	locals	
<i>228</i>					

Heap

addr	value	
400		

Data segment

addr	type*	value
600	<i>char[4]</i>	<i>'a' 'b' 'c' 'v'</i>

Type and name are not actually stored in memory or executable. Addresses shown are fictional. Assume sizeof(int)==4, sizeof(char)==1, sizeof(void*)==8.

HW05

TIP: Use strings on data segment.

```
char* s = "abc";
```

Strings on stack segment will probably work, but you might run into issues, depending on your code. There is no benefit to fighting those battles at this point.

Null terminator byte ('\0')

- Character literal: '\0'
- Value: 0
- Always use the character literal ('\0') in your code, not 0.
- Just a marker in memory to tell your code (and library functions, such as print(...)) where the string ends, so that you do not need to keep track of the string size (i.e. array length) separately.



Do not print the
null terminator ('\0')
to the console.
(screen)

~~X fputc('\0', stdout); //BAD~~