COMP1511 17s2

— Lecture 9 —

More Pointers!

~~Andrew Bennett~~
~~<andrew.bennett@unsw.edu.au>~~
Jashank Jeremy
<jashank.jeremy@unsw.edu.au>

memory layout
pointers and arrays
strings

1

Don’t panic!

Style in Assignment 0

"how do I make sure my assignment code has good style?"
code should be clear and readable

mechanically
style guide;
consistent indentation
consistent + sufficient whitespace

readable
consistent naming of variables, functions
proper use of `#defines`

semantically
good, clear flow of logic and control
clear construction of `if` statements
using functions well

Style in Assignment 0

“can I have too many functions?”

no.
well, probably not.

functions should describe what they’re doing,
rather than do it all themselves

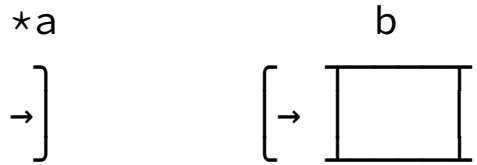
Style in Assignment 0

"what if my `if` statements are too long?"

https://webcms3.cse.unsw.edu.au/COMP1511/17s2/forums/2680545

Review: Pointers


```
a = &b;   *a == b;
```



pointers are like worm-holes
you reach through one to get a value

[[demo: wormhole.c]]

Pointer Operations: &

In C, we have two pointer operations:
& and *

&
takes an object, gives us
a reference to that object
the memory location of that object
the address of that object
a pointer to that object
a level of indirection

Pointer Operations: *

In C, we have two pointer operations:
& and *

*
takes
a reference to that object
the memory location of that object
the address of that object
a pointer to that object
a level of indirection
gives us the object's value

Now you’re thinking with Pointers!

[[demo: portal.c]]

Strings!

```
printf("Hello, world!\n");
```

text between double quotes is a string,
interpreted as ASCII.

(single quotes are a character literal!)

ASCII Silly Question…

when the world went computers,
we needed ways to represent characters
and everyone invented their own

ASCII, EBCDIC, KOI-8R, Shift-JIS, …

they all worked in weird and wonderful ways

we've mostly eradicated them all,
in favour of Unicode/UTF-8
… which is ASCII compatible

you need to use ASCII single- and double- quotes;
quotes from other character sets,
e.g., zh, jp, ko, etc., won't work

Arrays of `char`

a string is just an array of characters

nominally, `char []`, but often we just talk about `char *`

how long is a string? where does it end?

[[demo: `showCharArray` but display as `*%c` and use `char`]]

Strings are Arrays!

strings are arrays of characters…
but where the last character is `\0`

we call this the null terminator
makes it *much* easier to write functions

[[demo: `showCharArray`, but the array only, no length, and null-terminating]]

String literals, again

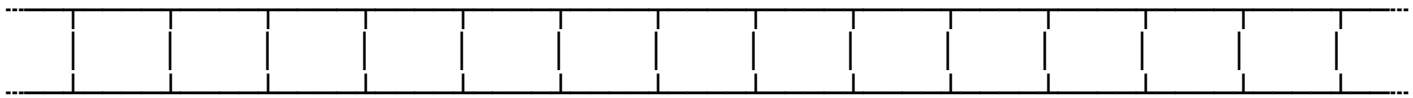
double-quoted string literals are arrays, too!

```
{ 'S', 'e', 'v', 'e', 'n', 't', 'e', 'e', 'n', '\0' }
```

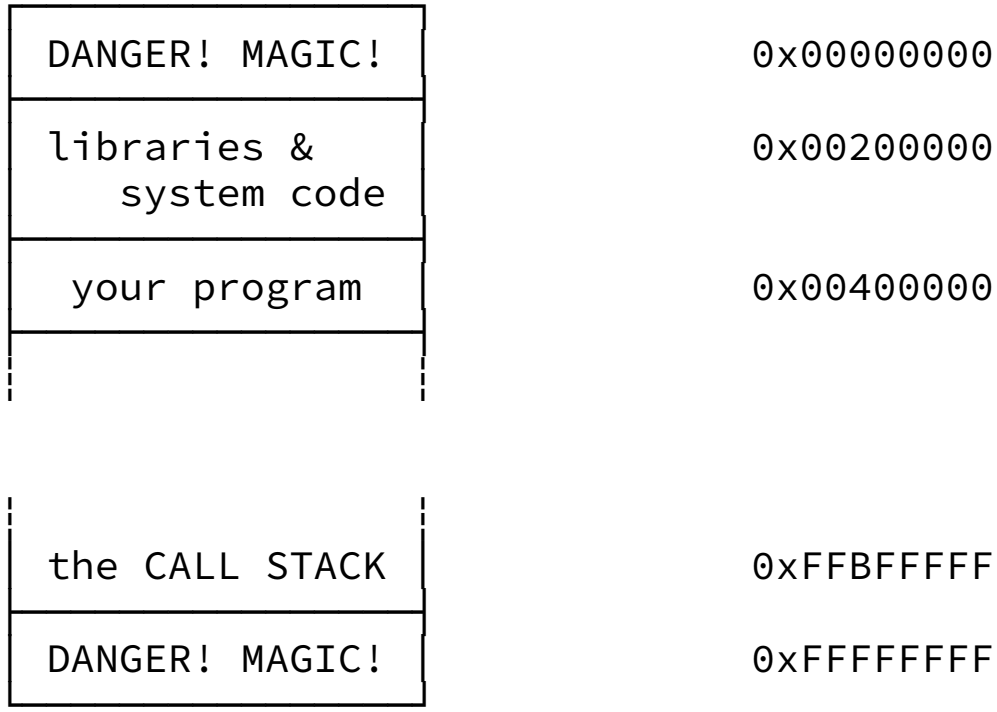
[[demo: `showCharArray`, but on a string literal]]
[[demo: strmem.c]]

Recap: Memory

memory is just a series of boxes



there's too many boxes
to draw them all horizontally...
what would it look like
if we drew them vertically?
(and zoomed out a long way?)


DANGER! MAGIC!
0x00000000
libraries &
system code
0x00200000
your program
0x00400000
the CALL STACK
0xFFBFFFFF
DANGER! MAGIC!
0xFFFFFFFF