C TYPES

C TYPES

TYPE = SYNTAX + SEMANTIC

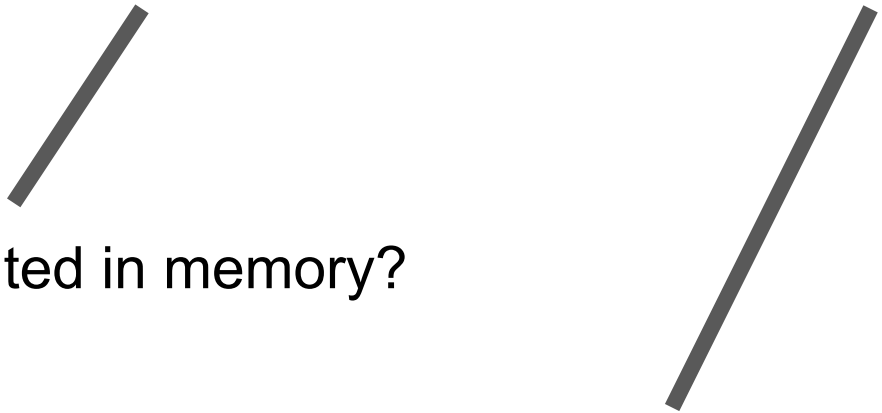
C TYPES

TYPE = SYNTAX + SEMANTIC

How is data represented in memory?

C TYPES

TYPE = SYNTAX + SEMANTIC



How is data represented in memory?

What does the data mean?

C TYPES - SYNTAX

TYPE = SYNTAX + SEMANTIC

How is data represented in memory?

C TYPES - SYNTAX

TYPE = SYNTAX + SEMANTIC

How is data represented in memory?

By n bytes!! **(1 byte = 8 bits)**

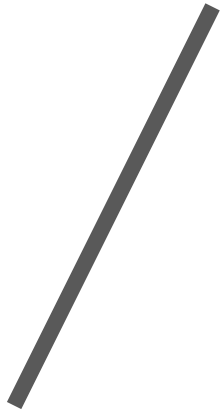
C TYPES - SYNTAX

TYPE	# BYTES
char	1 (8 bits)
short	2 (16 bits)
int	4 (32 bits)
long	8 (64 bits)

Warning: these numbers depends on your computer
see https://en.wikipedia.org/wiki/C_data_types

C TYPES - SEMANTIC

TYPE = SYNTAX + SEMANTIC



What does the data mean?

C TYPES - SEMANTIC

- On 8 bits I can represent 2^8 different *things*
- On n bits I can represent 2^n different *things*

But what *things* will I represent??

C TYPES - SEMANTIC

UNSIGNED INTEGERS

TYPE	SYNTAX: # BYTES	SEMANTIC
unsigned char	1 (8 bits)	0 to 2^8 - 1
unsigned short	2 (16 bits)	0 to 2^16 - 1
unsigned int	4 (32 bits)	0 to 2^32 - 1
unsigned long	8 (64 bits)	0 to 2^64 - 1

C TYPES - SEMANTIC

UNSIGNED INTEGERS

TYPE	SYNTAX: # BYTES	SEMANTIC
unsigned char	1 (8 bits)	0 to 255
unsigned short	2 (16 bits)	0 to 65535
unsigned int	4 (32 bits)	0 to ~4.3 billion
unsigned long	8 (64 bits)	0 to ~1.8 x 10^19

OVERFLOW!!!!!

C TYPES - SEMANTIC

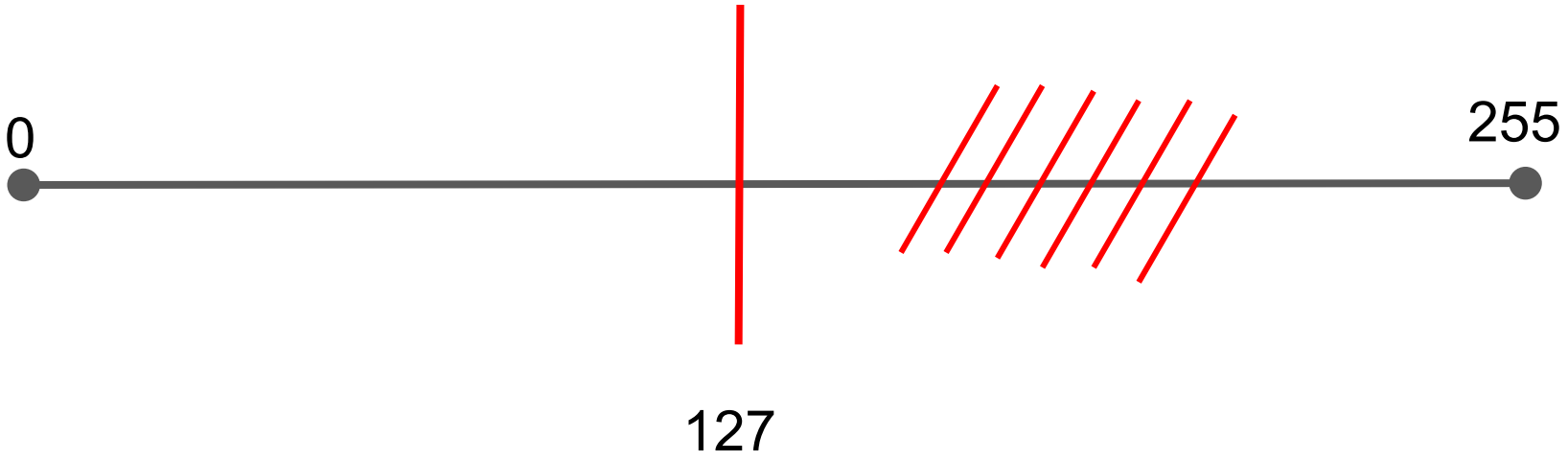
SIGNED INTEGERS

On 8 bits we can represent 256 different *things*:

C TYPES - SEMANTIC

SIGNED INTEGERS

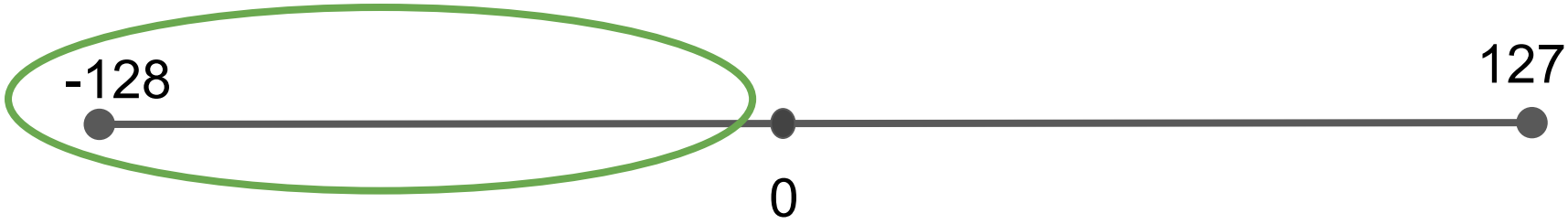
On 8 bits we can represent 256 different *things*:



C TYPES - SEMANTIC

SIGNED INTEGERS

On 8 bits we can represent 256 different *things*:



C TYPES - SEMANTIC

SIGNED INTEGERS

TYPE	SYNTAX: # BYTES	SEMANTIC
char	1 (8 bits)	-2^8 to 2^8-1
short	2 (16 bits)	-2^16 to 2^16-1
int	4 (32 bits)	-2^4 to 2^32-1
long	8 (64 bits)	-2^64 to 2^64-1

C TYPES - SEMANTIC

SIGNED INTEGERS

Example: how to represent -1 on 8 bits?

C TYPES - SEMANTIC

SIGNED INTEGERS

Example: how to represent -1 on 8 bits?

By 11111111 !!
Because 11111111 + 1 = (1)00000000

Its called “Two’s complement”
The first bit gives the sign! 0: positive, 1: negative

OVERFLOW!!!!!

C TYPES - SEMANTIC

FLOAT NUMBERS

TYPE	SYNTAX: # BYTES	SEMANTIC
float	4 (32 bits)	IEEE 754
double	8 (64 bits)	IEEE 754
long double	16 (128 bits)	IEEE 754

In practice: **use double! (when you can)**

C TYPES - SEMANTIC

BOOLEANS

TYPE	SYNTAX: # BYTES	SEMANTIC
bool	1	true or false

Warning: not always available. Needs #include <stdbool.h>