Today

C: Wrap up?
- [More Types]
- [Tools]
- [A few more things in C]
Types

int  char

signed

int

- → +

→ unsigned

operators:

bitwise and, or, not, xor

| 3 | 1 | ~ | ^ |

unsigned int +

\[
\begin{array}{c}
11111111 \\
10100101 \\
\end{array}
\]

unsigned int = 2

\[
\begin{array}{c}
0 \ldots 0 \quad 1 \quad 1 \\
0 \ldots 0 \quad 1 \quad 0 \\
\end{array}
\]

2 << 2 = 8

... 1000

shift: << >>
Types:

purpose of type systems?

⇒ check "consistency"

⇒ increase chances of correctness
Undefined Behavior

Performance / Speed

$\Rightarrow$ expose hardware

$\Rightarrow$ Compiler optimization

$\Rightarrow$ -O (opt flag) matters
Summary:

=> be careful when mixing types

(float / int)

always use double (if can)
not float

=> corner case =>
undefined

=> be wary

int
Type: void *

```
malloc -> returns this

int * x = malloc (sizeof (int));
```

```
char * x = malloc (10 * sizeof (char));
```

Generic data structures:

```
list_t * list_insert (int key, void * value)
```

```
node_t * n = malloc(...)
n->key = key;
n->value = value;
n->next = NULL;
```

```
void * lookup (list_t ** l, int key);
```
Tools

gcc          compiler driver
makefile     build code
valgrind     memory bugs
gdb          debugger
gcc: compiler driver

Steps:

$(C + \text{CPP}) \Rightarrow \text{CPP} \Rightarrow C$

$\text{CPP} + C \Rightarrow C$

Compiler (ecl, clang)

C \Rightarrow .S (assembly)

Assembler

.S \Rightarrow .O (object)

Binary

.O \Rightarrow \text{executable}

1001 0010 0110 1110 0101
Makefile

- Often: file, abstract

- target

- `(cmd1)` one or more commands

- prereq1 - prereqN

- files
valgrind

⇒ binary instrumentation

free(p)

// it knows how big alloc was
Debugger

Debugging: Art / Science

Logic:

Be Methodical
Have

⇒ Expectation

⇒ Guess: What went wrong
(Gather information)

⇒ Make change

⇒ Check