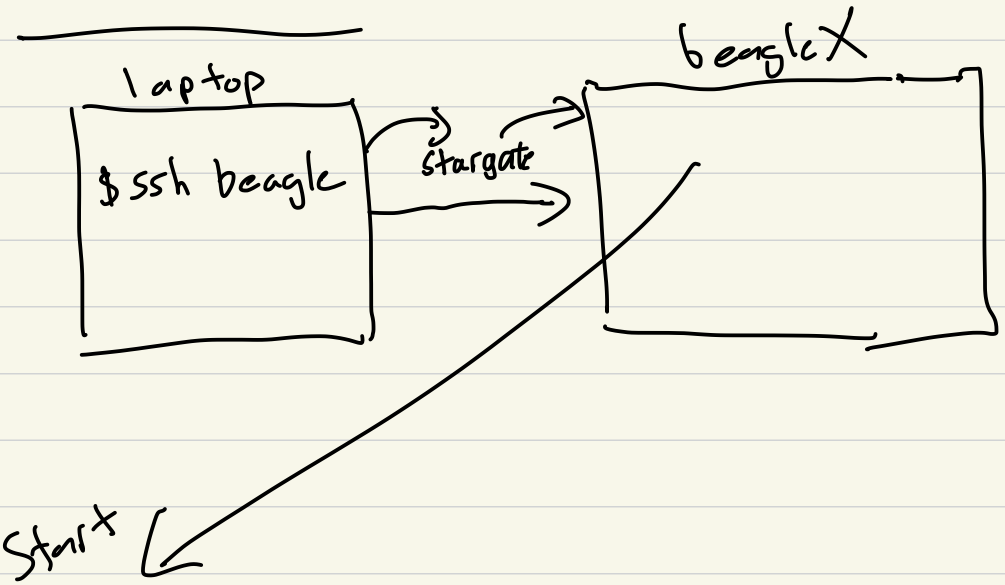
CS 315-01 C Numbers Conversions

Dev workflow
Number representation
Bases
Conversion
 numconv flow
Structs for options

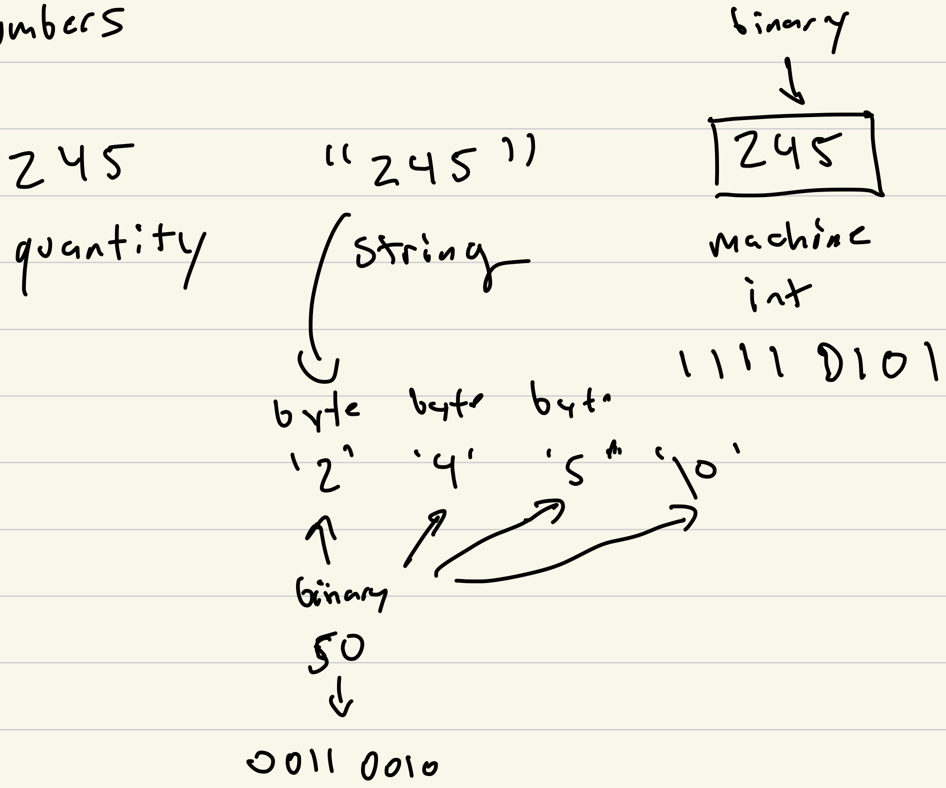


```
git clone <your-repo>

cd your-repo
```

edit | compiling / testing
can run code directly

```
git add newfile

git commit -a -m "message"
git push
```

Numbers



Decimal Base 10

245

$$(2 \times 10^2) + (4 \times 10^1) + (5 \times 10^0)$$

$$2 \times 100 \quad + \quad 4 \times 10 \quad + \quad 5 \times 1$$

$$200 \quad + \quad 40 \quad + \quad 5 = 245$$

Binary Base 2

$$\begin{array}{cccccc} & & 3 & 2 & 1 & 0 \\ 0 & b & 1 & 1 & 0 & 1 \\ & & 8 & 4 & 2 & 1 \end{array} \longrightarrow$$

```
int x = 13
int x = 0b1101
int x = 0xD
```

$$(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$$

$$8 \quad + \quad 4 \quad + \quad 0 \quad + \quad 1$$

$$= 13$$

4 bit binary number

0b1101

msb

most significant bit

lsb

least significant bit

n - bit binary numbers

2^n possible values

range 0 to $2^n - 1$

Hexadecimal Base 16

Dec(10)	Bin(2)	Hex(16)	
0	0000	0	
1	0001	1	
2	0010	2	
3	0011	3	
4	0100	4	
5	0101	5	
6	0110	6	
7	0111	7	
8	1000	8	
9	1001	9	
10	1010	A	a
11	1011	B	b
12	1100	C	c
13	1101	D	d
14	1110	E	e
15	1111	F	f

$0x\overset{2}{1}\overset{1}{A}\overset{0}{F}$

$(1 \times 16^2) + (A \times 16^1) + (F \times 16^0)$

$256 + \underset{160}{10 \times 16} + \underset{+15}{15 \times 1}$

$$\begin{array}{r} 256 \\ 175 \\ \hline \boxed{431} \end{array}$$

Project01

$$\text{numconv } \underbrace{431} \; \text{-o } \overset{\overset{\text{base}}{\downarrow}}{16}$$

0x1AF

numconv 0x1AF -o 10

431

$$\begin{tikzcd}
\text{intstr} \arrow[r] & \text{machine int} \arrow[r] & \text{numstr (base)} \\
\text{"245"} & \text{uint32\_t} &
\end{tikzcd}$$

$$\begin{tikzcd}
\text{dec} \arrow[dr] & & \text{dec} \\
\text{bin} \arrow[r] & \text{uint32\_t} \arrow[ur] \arrow[r] \arrow[dr] & \text{bin} \\
\text{hex} \arrow[ur] & & \text{hex}
\end{tikzcd}$$

$$\text{numconv } \underset{\underset{\text{argv[1]}}{\uparrow}}{\text{"245"}}$$

```
if prefix("245") == "0b" => bin
if prefix("245") == "0x" => hex
else => dec
```

char *s = "245";

s[0] = '2'
s[1] = '4'
s[2] = '5'
s[3] = '\0'

ASCII

'0'	= 48
'1'	= 49
'2'	= 50
'3'	= 51
'4'	= 52
'5'	= 53

int v_0 = s[0] - '0';
= 50 - 48
= 2

v_1 = s[1] - '0'
= 52 - 48
= 4

v_2 = s[2] - '0'
= 53 - 48
= 5

v = v_0 + v_1 + v_2
= v_0 * 100 + v_1 * 10 + v_2

```
                                         , int base
uint32_t intstr_to_int ( char *s ) {
        int v = 0;                       "245"
        int d;
        int i;
        while ( s[i] != '\0' ) {         v = 0
             v = v * 10;  - base         v = 2
             d = s[i] - '0';             v = 20
             v = v + d;  <-              v = 24
             i = i + 1                   v = 240
        }                                v = 245
        return v;
}
```

---

int to string   printf("%d", v);

```
  int v = 245;                   245 / 10 = 24
  int d0, d1, d2                 245 % 10 = 5
     d0 = v % 10      base
        = 245 % 10
        = 5  ->           ascii
                          char c = d0 + '0'
     v = v / 10
       = 24

     d1 = v % 10          ascii
        = 24 % 10
        = 4  ->
```

$v = v / 10$

$= 2$

$d_2 = v \% 10$

$= 2 \% 10$

$= 2 \longrightarrow$ ascii

$v = v / 10$

$= \boxed{0}$ <u>done!</u>

Binary

int $v = 0b\underbrace{101}0\circled{0}$ $\rightarrow$ 10

10/2

5

0b0101

$\uparrow \; \uparrow$

int d

$d_0 = v \% 2$

$= 1010 \% 2$

$= 0$

$v = v / 2$

$= 0101$

$d_1 = v \% 2$

$= 010\textcircled{1}$

$= 1$

```
main() {                                   numconv 245 -o2 -o16

                                                 info?
    // steps                                   [ input string
                                                 bool base2
? = parse_args()                                 bool bas16
                                                 bool base10
    v = conv_to_int()                          ]

    output_bases()

}


main() {
    bool base2;
    bool base10;
    bool base16;

    parse_args(argc, argv, &base2, &base10
                           &base16);

}
```

```
struct config_st {  <-  input_base =
        char input[MAX];
        bool base2;
        bool base10;
        bool base16;
  }


main() {
        struct config_st config;
        cp.base2 = false.

        parse_args(argc, argv, &config);
        uint32_t v;
        v = conv_str_to_int(cp.input, cp.input_base)

}


int parse_args( int argc, char *argv[],
                        struct config_st *cp )

        cp -> base2 = false;
        cp -> base10 = false;
```