CS 315-02 C Numbers Conversions

Dev workflow

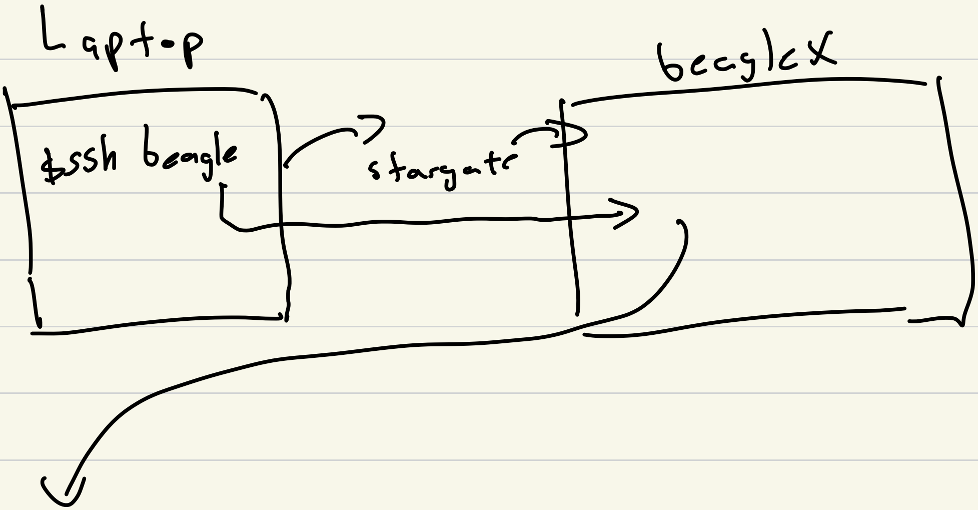
Number representation

Bases

Conversion

numconv structure

structs options



Start

```
git clone <your_repo>

cd your_repo
```

edit / compile / run / test

```
git add newfile

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

Numbers

245
quantity

"245"
string
↓
byte byte byte
'2' '4' '5' '\0'
↓
50 (dec)
↓
0011 0010

245
machine
Int 32 bit
↓
binary
↓
1111 0101

Decimal Base 10

245

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

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

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

Binary Base 2

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

$$0b\,\overset{3}{\underset{8}{1}}\,\overset{2}{\underset{4}{1}}\,\overset{1}{\underset{2}{0}}\,\overset{0}{\underset{1}{1}}$$

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

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

$$= 13$$

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

$\overset{2}{}\,\overset{1}{}\,\overset{0}{}$

0x1AF

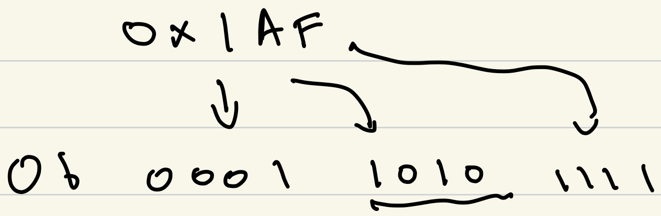
$$(1 \times 16^2) + (A \times 16^1) + (F \times 16^0)$$

$$256 + 10 \times 16 + 15 \times 1$$

$$256 + 160 + 15$$

$$\begin{array}{r} {}^{11} \\ 256 \\ 175 \\ \hline \boxed{431} \end{array}$$

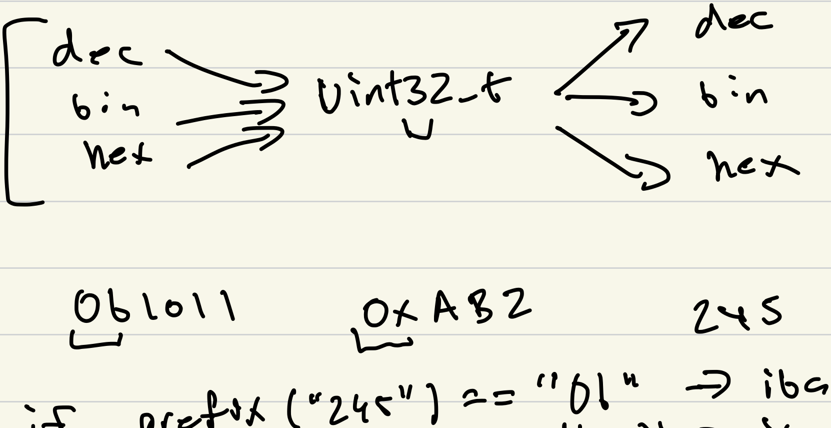
Hex ⇒ Bin



Project01

```
                 base
numconv 431 -o 16
0x1AF
numconv 0x1AF -o 10
431
```

intstr "245" → machine int uint32_t → num str (base)



0b1011 0xAB2 245

```
if prefix("245") == "0b" → ibase=2
else if prefix("245")=="0x" → ibase = 16
else ibase = 10
```

char *s = "$\overline{245}$" ;

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'

$\boxed{v_2}$ = 50 - 48

= 2

v_1 = s[1] - '0'

$\boxed{v_1}$ = 52 - 48

= 4

v_2 = s[2] - '0'

$\boxed{v_0}$ = 53 - 48

= 5

int v = (v_0 * 100) + (v_1 * 10) + (v_2 * 1)

```
                                             int base
uint32_t intstr_to_int(char *s) {
    uint32_t v = 0;
    uint32_t d;
    int i = 0;

    while (s[i] != '\0') {
        v = v * 10;    base
        d = s[i] - '0';
        v = v + d
        i = i + 1;
    }
    return v;
}
```

"245"

i	

0	v = 0
	v = 2

1	v = 20
	v = 24

2	v = 240
	v = 245

~~printf("%x", v);~~

printf("%s", strnum)

int to strnum

int v = 245;

$245/10 = 24$

$245 \% 10 = 5$

int d_0, d_1, d_2

$d_0 = v \% 10$

$= 245 \% 10$

$= \boxed{5}$ —

ascii

$v = v / 10$

$= 245/10$

$= 24$

char c = d_0 + '0'

$d_1 = v \% 10$

$= 24 \% 10$

$= \boxed{4}$ —

$v = v / 10$

$= 24 / 10$

$= 2$

$d_2 = v \% 10$

$= \boxed{2}$ —

$v = v / 10 = 2/10 = \boxed{0}$ done!

Binary output

uint32_t v = 0b1010 10

int d_0, d_1, d_2, d_3

$d_0 = v \% \overset{\text{base}}{\underbrace{2}}$ 10 % 5 = 0

$= 101\underbrace{0} \% 2$

$= 0$

$v = v / \overset{\text{base}}{\underbrace{2}}$ 10 / 2 = 5

$= 1010 / 2$ 010101

$= 0101$

$d_1 = v \% 2$

$= 010\boxed{1} \% 2$

$= 1$

```
Main() {
        // steps
        parse_args()
        x = conv_to_int()
        output_bases()
}
```

```
Main() {
        bool base2;
        bool base10;
        bool base16;
        parse_args(argc, argv, &base2, &base10
                                &base16);
}
```

```
struct config_st {
        char input[MAX];
        int ibase;
        bool base2;
        bool base10;
        bool base16;

 }


main() {
      struct config_st config;
       :
     →config_init(&config)

       parse_args(argc, argv, &config)
       uint32_t v;
       v = conv_str_to_int(config.input, config.ibase)
     :
}

int parse_args(int argc, char *argv[],
                 struct config_st *cp)
  →    cp->base2 = false
```