sizeof

```
int main(int argc, char *argv[]) {
    printf("%u\n", sizeof(char));
    printf("%u\n", sizeof(short));
    printf("%u\n", sizeof(int));
    printf("%u\n", sizeof(long));
    printf("%u\n", sizeof(long long));
    printf("%u\n", sizeof(float));
    printf("%u\n", sizeof(double));
    printf("%u\n", sizeof(long double));
    printf("%u\n", sizeof(int *));
    int x = 0;
    printf("%u\n", sizeof(x));
    return x;
}
```

endian

```
unsigned int x = 0x12345678;

void dump(void *in, int len) {
    unsigned char *p = (unsigned char *) in;
    for (i = 0; i < len; i++) {
        printf("addr: %p   value: %x\n", p, *p);
        p++;
    }
}
int main(int argc, char *argv[]) {
    printf("&x = %p\n", &x);
    dump(&x, sizeof(int));
    return 0;
}
```

bits

```
int main(int argc, char *argv[]) {
    unsigned char x = 0x69;
    unsigned char y = 0x55;

    printf("%hhx\n", x & y);
    printf("%hhx\n", x | y);
    printf("%hhx\n", x ^ y);
    printf("%hhx\n", ~x);
    printf("%hhx\n", ~y);

    return 0;
}
```

logic

```
int main(int argc, char *argv[]) {
    unsigned char x = 0x69;
    unsigned char y = 0x55;

    printf("%hhx\n", x && y);
    printf("%hhx\n", x || y);
    printf("%hhx\n", !x);

    return 0;
}
```

shift

```
int main(int argc, char *argv[]) {
    unsigned int x = 0x0;
    printf("%.8x\n", x);
    x = 0x1;
    printf("%.8x\n", x);
    x = x << 1;
    printf("%.8x\n", x);
    x = x << 32;
    printf("%.8x\n", x);
    x = x << 8;
    printf("%.8x\n", x);
    x = x >> 8;
    printf("%.8x\n", x);
    return 0;
}
```

set

```
int main(int argc, char *argv[]) {
    unsigned int x;
    x = 0x0;
    x = x | (0x1 << 12);  //12 is member
    printf("set: %.8x\n", x);
    x = x | (0x1 << 6); // 6 is member
    printf("set: %.8x\n", x);
    // test for membership
    printf("6 in set? %d\n", (x>>6)& 0x1);
    printf("7 in set? %d\n", (x>>7)& 0x1);
    // & is intersection, | is union
    // ~ is complement, ^ is sym diff
    unsigned int y = 0x01010101;
    printf("set: %.8x\n", y);
    printf("set: %.8x\n", x | y);
    printf("set: %.8x\n", x & y);
}
```

twocomp

```
int main(int argc, char *argv[]) {
    short int x = 12;
    short int y = -12;

    printf("%hhx\n", x);
    printf("%hhx\n", y);

    // just the same bits ...
    unsigned int a = 0x1 << 31;
    printf(" %u\n", a);

    int b = (int) a;
    printf("%d\n", b);
    return 0;
}
```

Binary	**Unsigned**	**2's comp**
0000b	0	
0001b	1	
0010b	2	
0011b	3	
0100b	4	
0101b	5	
0110b	6	
0111b	7	
1000b	8	
1001b	9	
1010b	10	
1011b	11	
1100b	12	
1101b	13	
1110b	14	
1111b	15	

examples

Assume 4-bits:

*** 3 + -2 ?**

*** 7 + 7 ?**

*** -8 + -8 ?**

cast

```
int main(int argc, char *argv[])
{
    printf("%d\n", 0 == 0U);
    printf("%d\n", -1 < 0);
    printf("%d\n", -1 < 0U);
    return 0;
}
```

security

```
#define KSIZE (1024)
unsigned char kbuf[KSIZE];

void mcopy(void *dst, void *src, unsigned int n) {
    printf("copying %u bytes to dst\n", n);
    // does the copy of n bytes ...
}

void copy_from_kernel(void *ubuf, int maxlen) {
    int len = maxlen;
    if (len > KSIZE) {
        len = KSIZE;
    }
    printf("len: %d\n", len);
    mcopy(ubuf, kbuf, len);
}

int main(int argc, char *argv[]) {
    char mybuf[512];
    copy_from_kernel(mybuf, 512);
    return 0;
}
```

signextend

```
int main(int argc, char *argv[]) {
    short int x = 15; // also try: -15
    int ex = (int) x;

    dump("short x::", &x, sizeof(short int));
    dump("just x ::", &ex, sizeof(int));

    return 0;
}
```

fastmul

```
int main(int argc, char *argv[]) {
    int x = 3;
    printf("%d\n", x << 3);
    printf("%d\n", x * 8);
    printf("%d\n", (x << 5) - (x << 3));
    // printf("%d\n", x * ??);
    x = 24;
    printf("%d\n", x >> 3);
    // printf("%d\n", x / ??);
    return 0;
}
```