Computer Systems

CS107

Cynthia Lee

Today's Topics

- Wrap-up of C programming topics:
  - Function pointers, callbacks
- Number representation
  - Integer representation
  - Signed numbers with two's complement

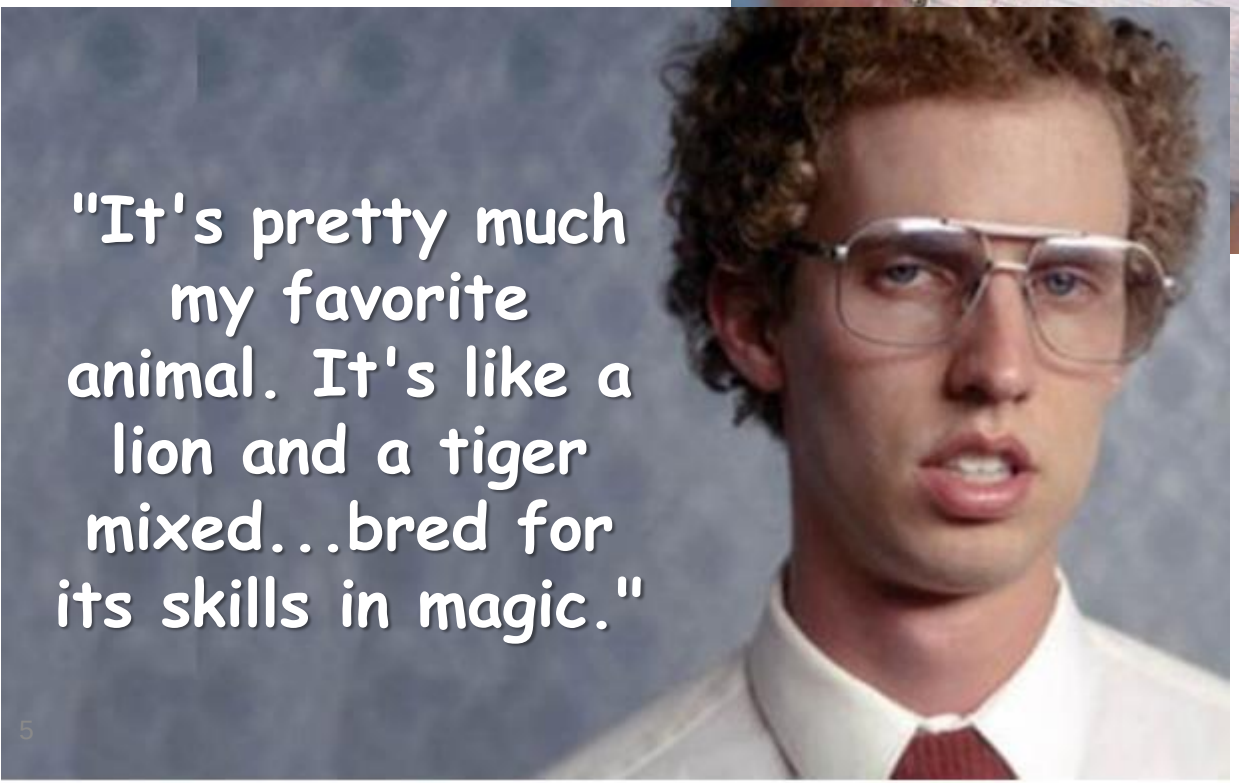
NEXT TIME:

- Monday is last day of topics that will be included on next Friday's midterm
  - Reasoning about special conditions with signed and unsigned
    - Overflow and underflow conditions
    - Comparison operators (< >) with signed and unsigned
  - Bytes, bits, bitwise operators

More on building generics with void* and function pointers

How NOT to use void pointers

Let's make a Liger!

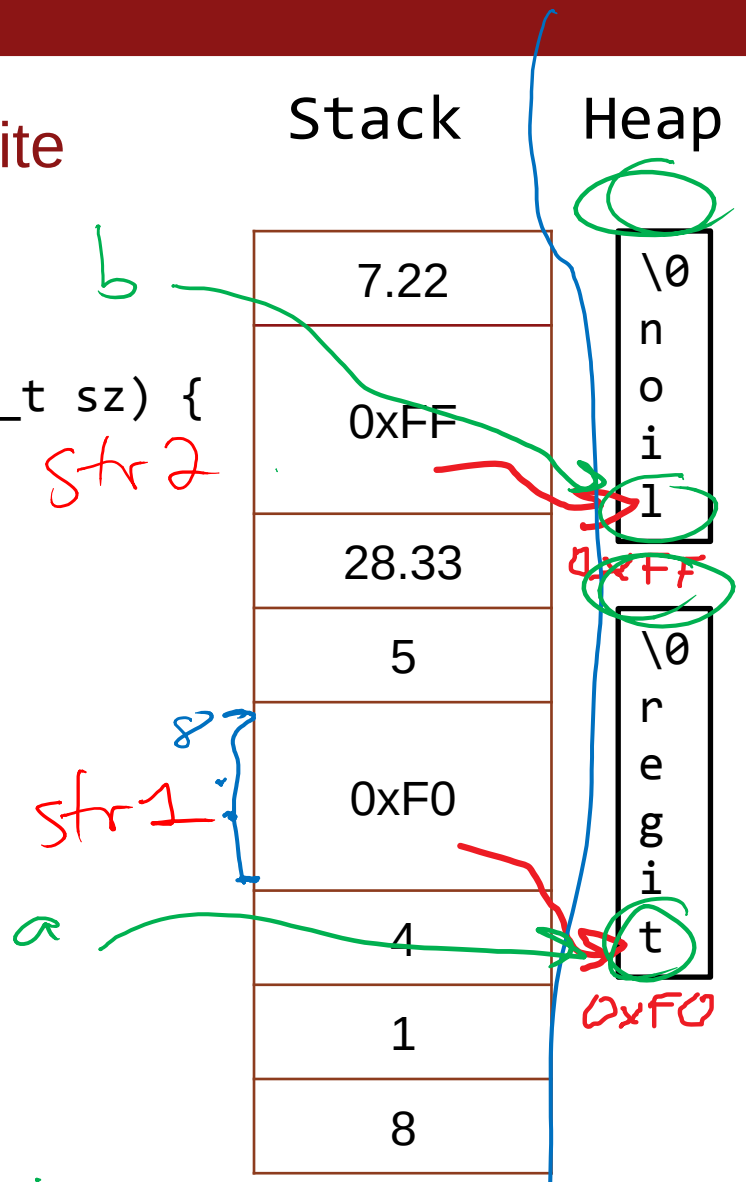


How can we (mis-)use our swap to write the string "liger" at mem addr 0xF0?

- Generic swap function:

```
void swap_any(void *a, void *b, size_t sz) {
    char tmp[sz];
    memcpy(tmp, a, sz);
    memcpy(a, b, sz);
    memcpy(b, tmp, sz);
}
```

```
char *str1 = strdup("tiger");
char *str2 = strdup("lion");
A. swap_any(&str1, &str2, sizeof(char*));
B. swap_any(&str1, &str2, sizeof(char));
C. swap_any(str1, str2, sizeof(char*));
D. swap_any(str1, str2, sizeof(char));
E. Other
```



Even more horrifying hybrids!

- Generic swap function:

```
void swap_any(void *a, void *b, size_t sz) {
    char tmp[sz];
    memcpy(tmp, a, sz);
    memcpy(a, b, sz);
    memcpy(b, tmp, sz);
}
```

- See if you can draw memory diagrams showing what happens in these examples of **mis-**usage:

```
int x = 8, y = 4;
swap_any(&x, &y, sizeof(char));
double dx = 3.75, dy = 9.21;
swap_any(&dx, &dy, sizeof(int));
int a = 8, b = 1;
swap_any(&a, &b, sizeof(double));
```

7.22
9.21
28.33
5
3.75
4
1
8

Function pointers

Returning to our int max example

```
void *find_max_any(void *arr, int n, size_t sz,
                   int (*cmp)(const void *, const void *)) {
    void *max = arr;
    for (int i = 1; i < n; i++) {
        void *ith = (char *)arr + i * sz;
        if (cmp(ith, max) > 0)
            max = ith;
    }
    return max;
}

int cmp_int(const void *a, const void *b) {
    int one = *(int *)a, two = *(int *)b;
    return one - two;
}

int main(int argc, char *argv[]) {
   int nums[] = {40, 99, 23, 45, 12, 45, 23, 59, 33, 92};
   printf("%d\n", *(int*)find_max_any(nums, 10, sizeof(int), cmp_int));
   return 0;
}
```

Generic functions/callbacks: USE WISELY!!

```
int cmp_int(const void *a, const void *b) {
    int one = *(int *)a, two = *(int *)b;
    return one - two;
}

int main(int argc, char *argv[]) {
    int nums[] = {40, 99, 23, 45, 12, 45, 23, 59, 33, 92};
    printf("%d\n", *(int*)find_max_any(nums, 5, sizeof(double), cmp_int));
}

void *find_max_any(void *arr, int n, size_t sz,
                   int (*cmp)(const void *, const void *)) {
    void *max = arr;
    for (int i = 1; i < n; i++) {
        void *ith = (char *)arr + i*sz;
        if (cmp(ith, max) > 0)
            max = ith;
    }
    return max;
}
```

shark

bear

octopus

Generic functions/callbacks: USE WISELY!!

```
int cmp_int(const void *a, const void *b) {
    int one = *(int *)a, two = *(int *)b;
    return one - two;
}

int main(int argc, char *argv[]) {
   int nums[] = {40, 99, 23, 45, 12, 45, 23, 59, 33, 92};
   printf("%d\n", *(int*)find_max_any(nums, 5, sizeof(double), cmp_int));
}

void *find_max_any(void *arr, int n, size_t sz,
                   int (*cmp)(const void *, const void *)) {
    void *max = arr;
    for (int i = 1; i < n; i++) {
        void *ith = (char *)arr + i*sz;
        if (cmp(ith, max) > 0)
            max = ith;
    }
    return max;
}
```

shark bear octopus

PREDICT: What is printed?

(a) Nothing—compiler error

(b) Nothing—crashes

(c) 99

(d) 40

(e) Unpredictable /other

Binary


Solo hay 10 tipos
de personas en el mundo:
los que entienden binario
y los que no

Bits and Bytes

THE BUILDING BLOCKS OF EVERYTHING IN THE COMPUTER

Stanford University

Bits and Bytes: essential facts

- "Bit" is a **b**inary dig**it**, 0 or 1
- "Byte" is 8 bits (one `char`)
- Our system is "byte-addressable," meaning each address refers to storage space for 1 byte

- The `char, short, int, long` family of types:
  - `char` is 1 byte = 8 bits
    - 2^8 = 256 possible `char` values
  - `short` is 2 bytes = 16 bits
    - 2^{16} = 65,536 possible `short` values
  - `int` is 4 bytes = 32 bits
    - 2^{32} = 4,294,967,296 possible `int` values (~4 billion)
  - `long` is 8 bytes = 64 bits
    - 2^{64} = 18,446,744,073,709,551,616 possible `long` values (~18 quintillion)

Bits as *Unsigned* Base 2 Numbers

Self-test: Integer representation in binary

What is the unsigned 4-bit binary representation of 14?

a) 1111 $= 8+4+2+1 = 15$

b) 1110 $= 8+4+2 = 14$

c) 1010 $= 8+2 = 10$

8421

d) Other

$101_2 = 5_{10}$

Self-test: Integer representation in binary

What is the base-10 equivalent of the unsigned 4-bit binary number 1010?

a) 20

b) 101

c) 10

d) 5

e) Other

Hexadecimal (base 16)

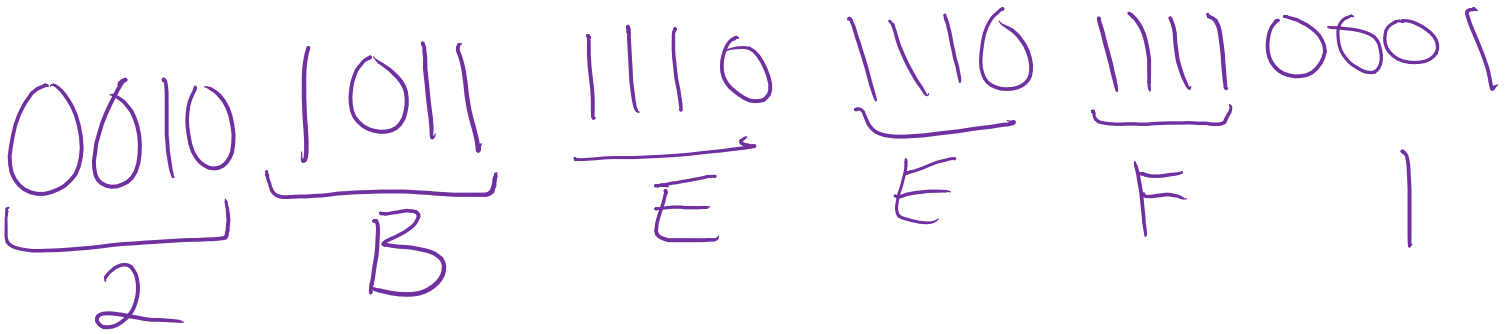
Base 10	Base 2 (4-bit)	Base 16
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Base 16: 0xF0

Base 2: 1111 0000

Self-test: Integer representation in hexadecimal

What is the unsigned binary equivalent of the unsigned hexadecimal number 0x2BEEF1?



Bits as *Signed* Base 2 Numbers

Self-test: Two's complement

What is the base-10 equivalent of the signed (two's complement) 4-bit binary number 1010?

a) -10

b) 10

c) 11

d) -11

e) 5

f) -5

g) 6

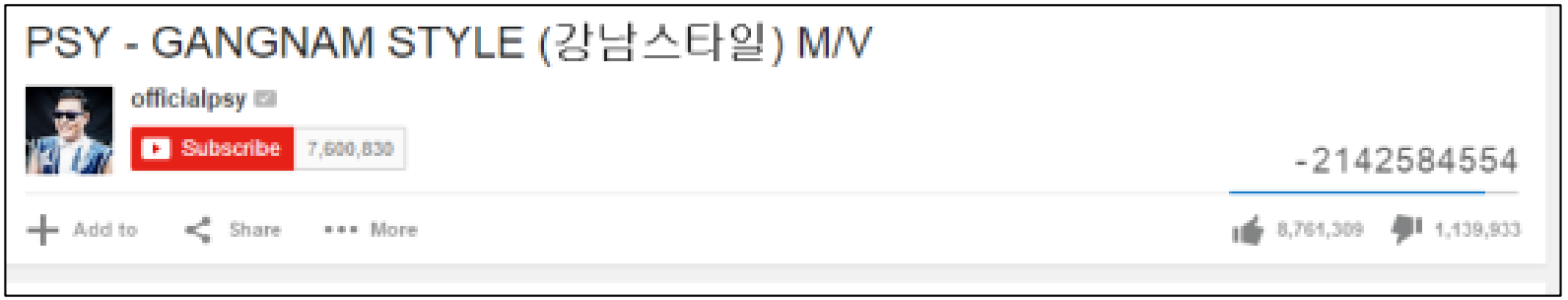
h) -6

i) Other

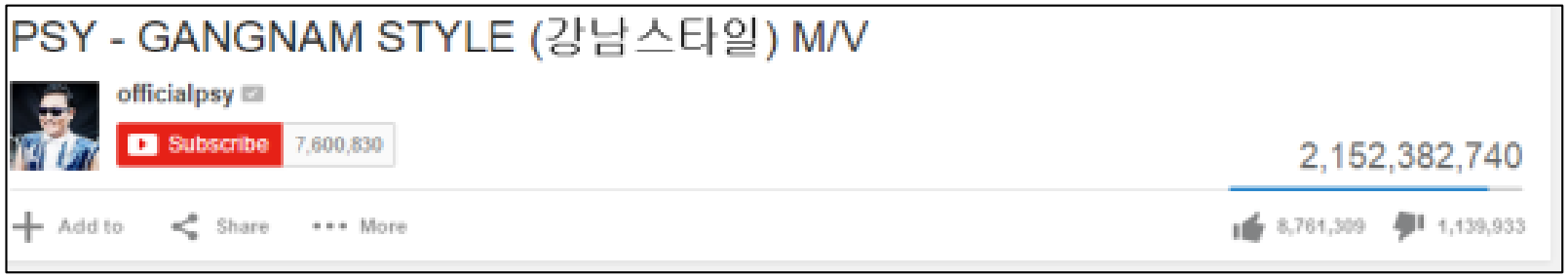

Inbox (15) - cynthia.b
Stanford Center for P
integer - What is the
stackoverflow.com/questions/94591/what-is-the-maximum-value-for-a-int32
What is the maximum value for a int32?
I can never remember that number. I need a memory rule.
606
integer
92
share improve this question
edited May 28 '14 at 14:09
Ben Hoffstein
49.5k • 5 • 66 • 101
asked Sep 18 '08 at 17:18
Flinkman
5,181 • 4 • 18 • 48
107 Why would you need the exact number? I remember "(2^31)-1" or "+/- 2 billion" and that's good enough for everything I ever needed. – Joachim Sauer Mar 3 '09 at 11:21
27 unsigned: 2^32-1 = 4·1024^3-1; signed: -2^31 .. +2^31-1, because the sign-bit is the highest bit. Just learn 2^0=1 to 2^10=1024 and combine. 1024=1k, 1024^2=1M, 1024^3=1G – comonad Mar 28 '11 at 20:01
6 I generally remember that every 3 bits is about a decimal digit. This gets me to the right order of magnitude: 32 bits is 10 digits. – Barmar Oct 2 '13 at 15:11
30 Answers
active
oldest
votes
It's 2,147,483,647. Easiest way to memorize it is via a tattoo.
2397
share improve this answer
edited Oct 20 '14 at 16:30
Allbite
1,415 • 1 • 13 • 15
answered Sep 18 '08 at 17:20
Ben Hoffstein
49.5k • 5 • 66 • 101

Overflow in two's complement

- In two's complement, when you exceed the maximum value of int (2,147,483,647), you "wrap around" to negative numbers:

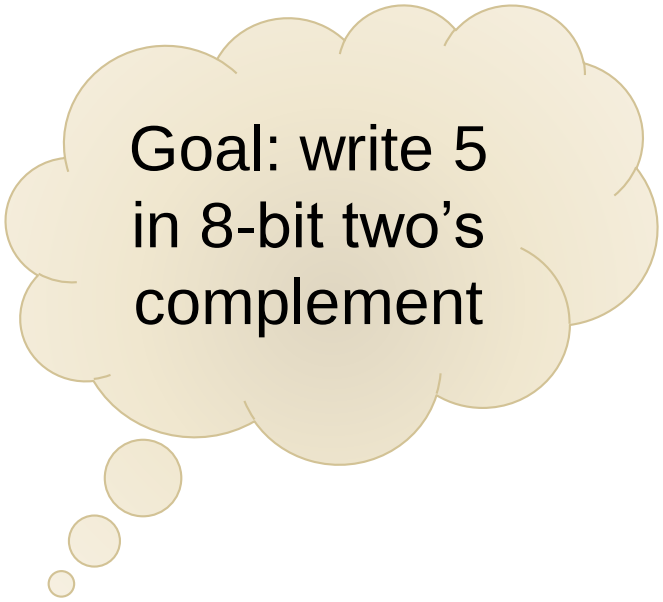


- Here is the link after Google upgraded to 64-bit integers:



Signed integers with two's complement representation

Signed integers with two's complement



Steps to write a positive (or zero) number in two's complement:

1. Write the number in usual unsigned binary representation
2. Make sure that the number will "fit" in the number of bits you have
   - For positive numbers, there needs to be at least one zero in the most significant (leftmost) bit
   - 00000101 (no problem for 5 in 8 bits)
3. Done!
   - Answer: 00000101

Signed integers with two's complement

Goal: write -5 in 8-bit two's complement

Steps to write a negative number in two's complement:

1. Write the *absolute value* of the number in usual unsigned binary representation
2. Make sure that the number will "fit" in the number of bits you have
   - Since we are writing the absolute value, a positive number, there needs to be at least one zero in the most significant (leftmost) bit*
   - 00000101 (no problem for 5 in 8 bits)
3. "Flip" each bit (0 → 1, 1 → 0)
   - 00000101 → 11111010
4. Add one
   - 11111010 → 11111011
5. Done!
   - Answer: 11111011

* There is one negative number whose positive number won't "fit"—more on this Friday