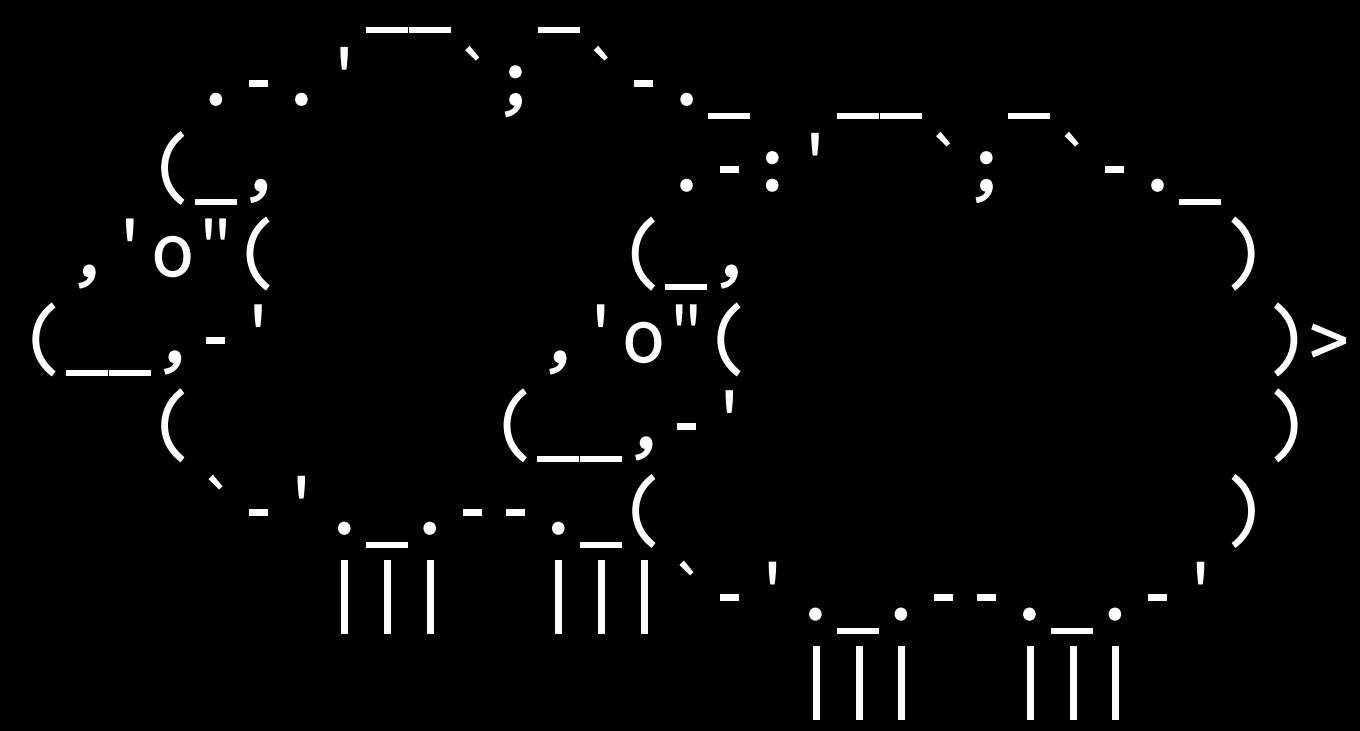
--author by Vincent Moscatello
--title C bugs
--Sun Feb 16 19:00:00 EST 2016



TODAY we are covering...

* C basic overview (how does the thing work?)
* 3 common C bug classes

Basic C overview

* A lot like C++ but without classes
* Pretty close to assembly

* This thing --> * <--- is a pointer
* Doesn't do garbage collection like java

Basic C overview 2

```
void fuzzy(){
      int foo = 1337;
      int * bar = &foo;

      foo = 69;
      printf("bar: %d", *bar);
}
```

Two new symbols * and &

Basic C overview 3

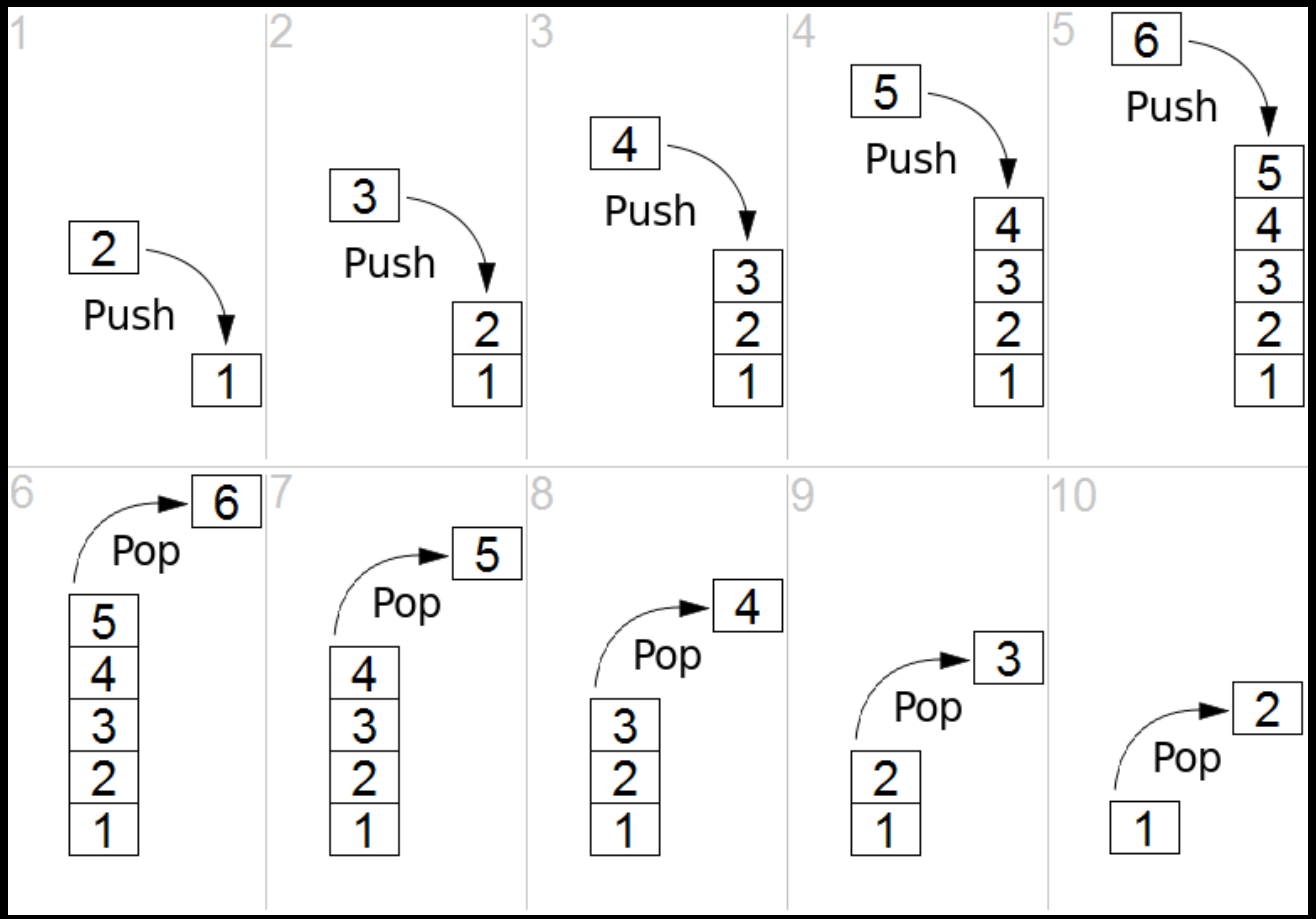
* C/C++ Makes memory allocation in two areas easy

  -Stack
  -Heap

* A lot of bugs occur because programmers don't Understand this distinction

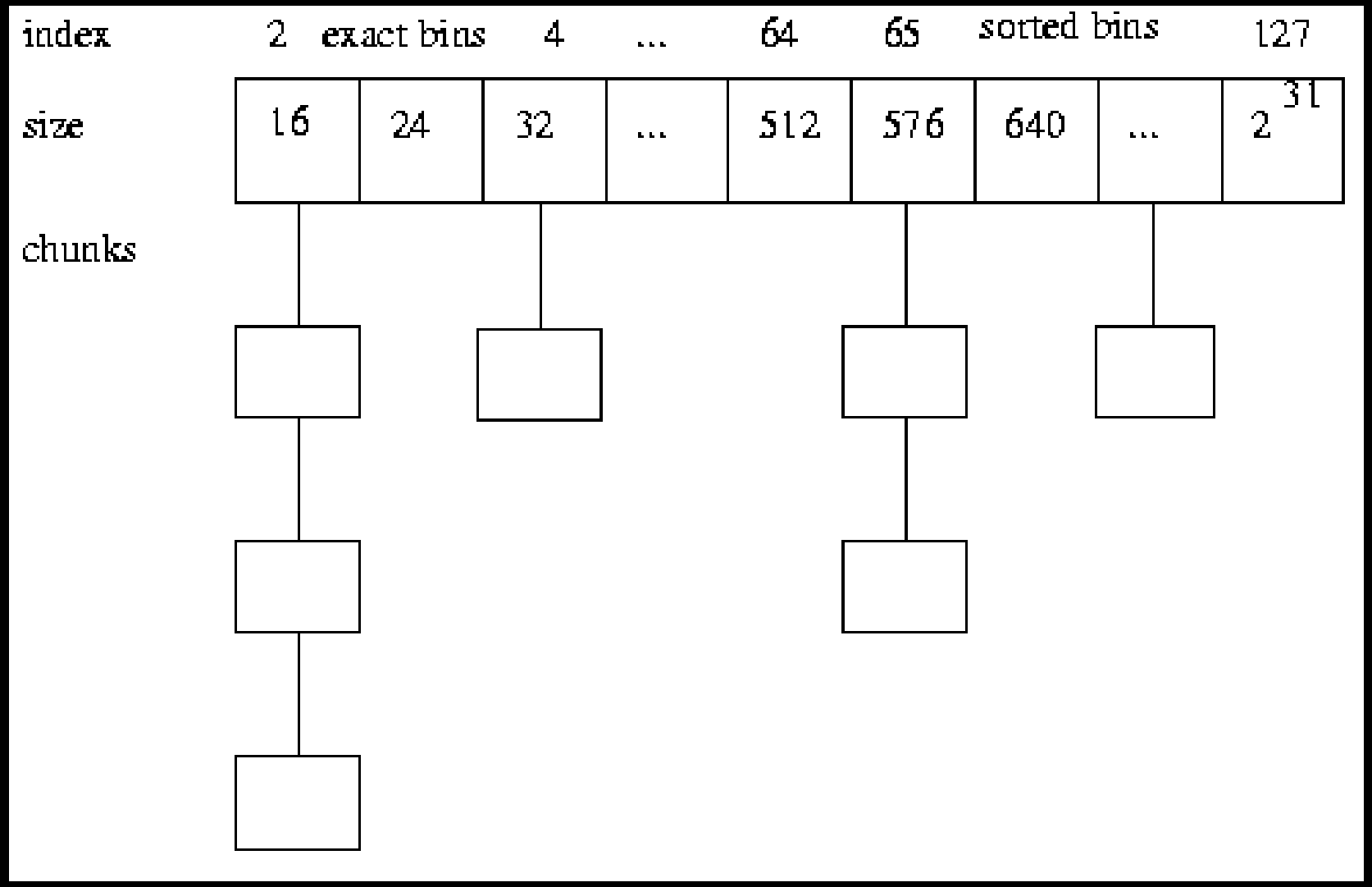
Basic C overview 4

* Stack = first in last out structure



Basic C overview 5

Heap = non local variables malloc/new


index
2
exact bins
4
...
64
65
sorted bins
127
size
16
24
32
...
512
576
640
...
2^31
chunks

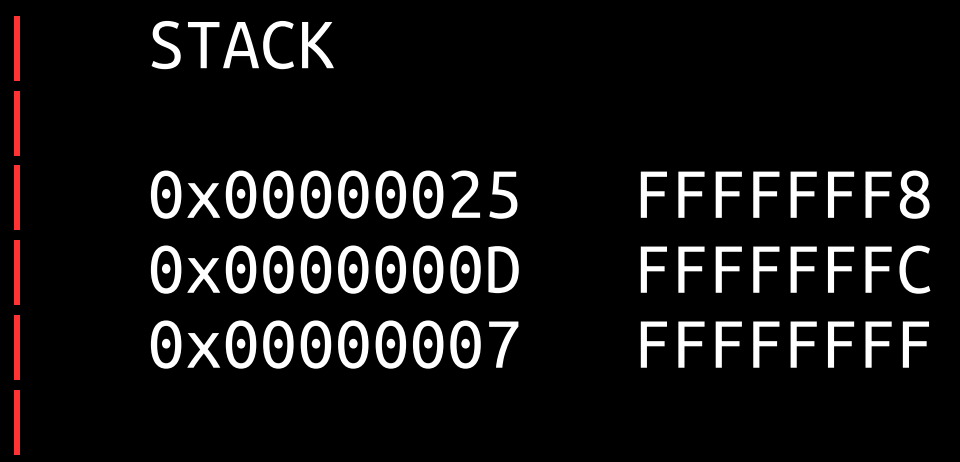
Basic C overview 6

/proc/pid/maps

```
7fff57d43000-7fff57d64000 rw-p 00000000 00:00 0 [stack]
008eb000-00b55000 rw-p 00000000 00:00 0         [heap]
```

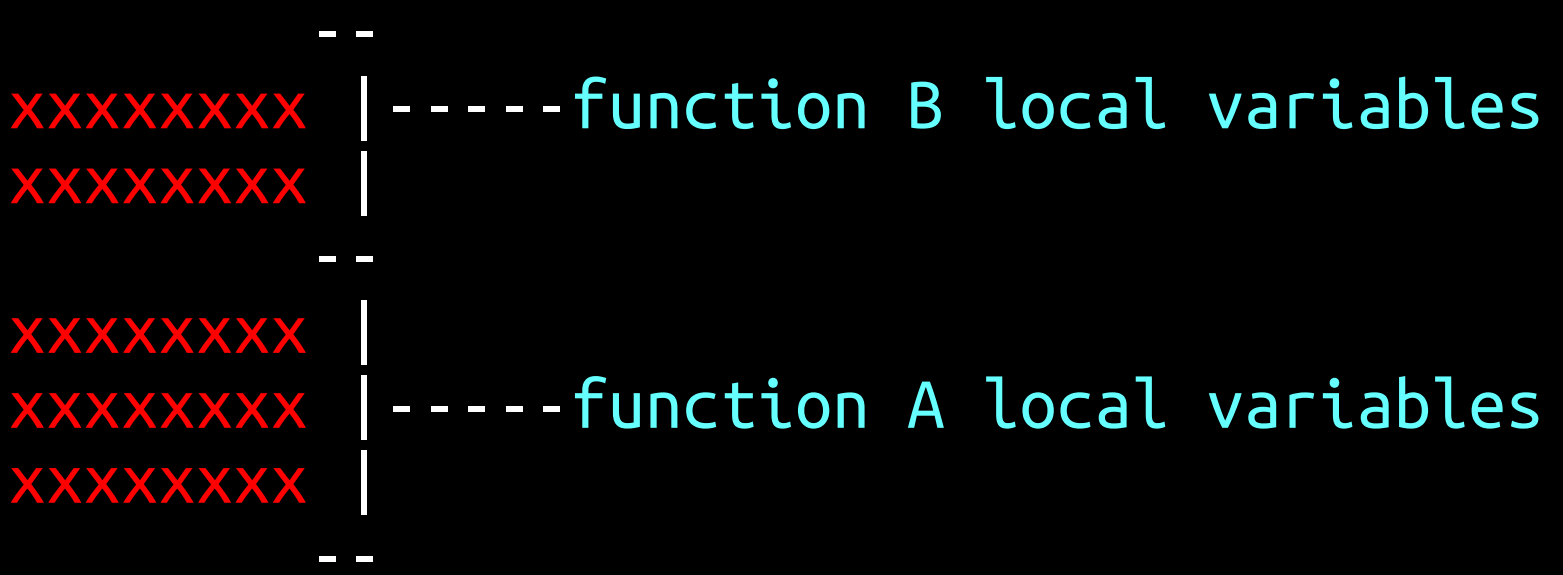
Basic C overview 7

```
void counting(){
    int foo = 7;
    int bar = 13;
    int baz = 37;
}
```



Basic C overview 8

Each function gets its own little stack aka (stack frame)

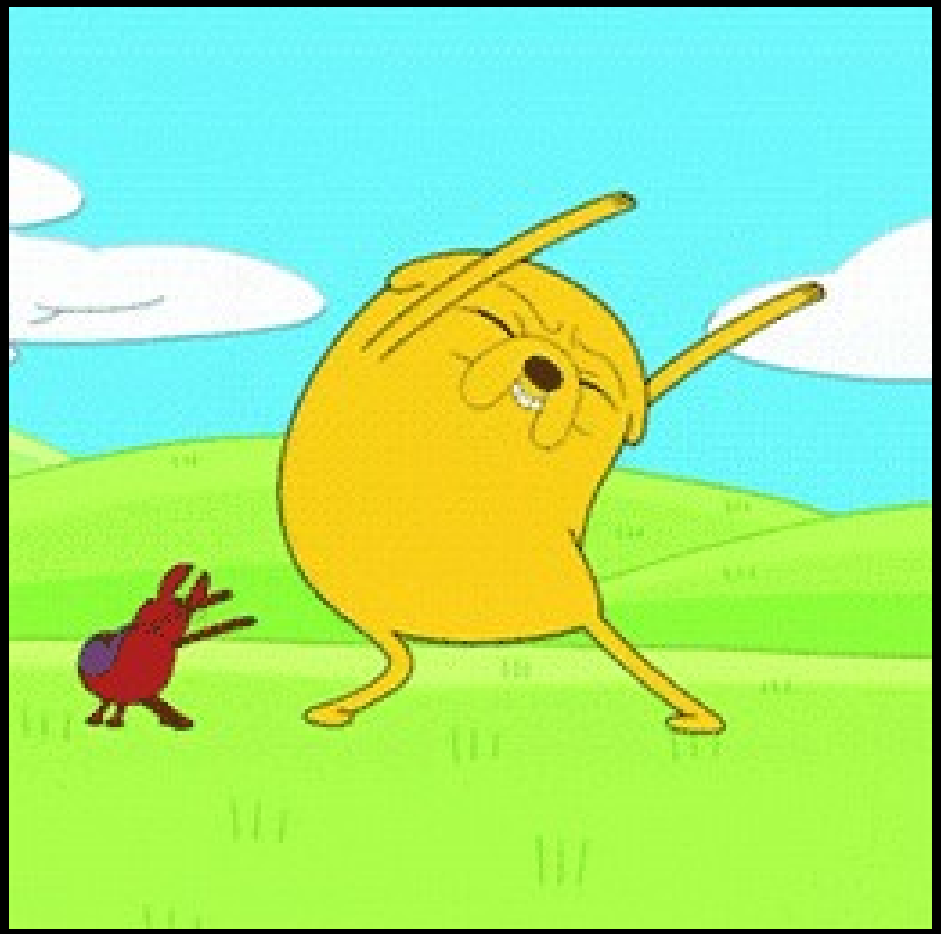


Basic C overview 8

Little endian BYTE order.

```
07 00 00 00 = int foo = 7;
41 42 43 00 = char bar[] = "ABC"
```

CBUGS



Why bother learning c bugs?

* Linux kernel exploitation
* VLC media player exploitation
* strace <--- (We will crash this peace of crap later)

* We can pwn embeded devices!

USE AFTER FREE (stack)

Vulnerable code

```
char * sheep(){
    char wool[128];
    fgets(wool, sizeof(wool), stdin);
    return wool;
}

int main(){
    char * woops =  sheep();
    printf("I'm a %s sheep", woops);
    return 0;
}
```

Use after free fix

```
char * sheep(){
        char * wool = malloc(128);
        fgets(wool, 128, stdin);
        return wool;
}
```

ARRAYS VS POINTERS

Vulnerable function

```
void easy_strcpy(char * foo, char * bar){
    strncpy(foo, bar, sizeof(foo));
 }
```

Proof of concept

```
void wool_1(){
    char * foo = "hello world"
    printf("%d", sizeof(foo));
}

void wool_2(){
    char foo[] = "hello world"
    printf("%d", sizeof(foo));
}
```

The results?

4

12

Fix

```
//don't make a function like this...

 void easy_strcpy(char * foo,
                  int foo_size,
                  char * bar)
 {
    snprintf( foo, foo_size, “%s”, bar);
 }
```

Use strlen() when appropriate instead of sizeof()

More info...

Two different allocation

```
//allocates hello world on the data segment
char * foo = "hello world"

//allocates hello world on the stack
char foo[] = "hello world”
```

OVERFLOWS

OVERFLOWS

```
void vulnerable(int foo, char * bar){
    char foo[64];
    char bar[32];
    gets(bar);
}
```

FIX

```
void vulnerable(int foo, char * bar){
    char foo[64];
    char bar[32];
    fgets(bar, sizeof(bar), stdin);
}
```

If we can corrupt memory in FOO! We may be able to control the IP!

MAN PAGES

Not sure how a function works?

Read the man pages. Stackoverflow is sometimes wrong.

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    char ch, source_file[20], target_file[20];
    FILE *source, *target;
    printf("Enter name of file to copy\n");
    gets(source_file);
```

BUGS REALLY HAPPEN

[demo strace]

What I need to know tonight?

Make sure you understand OVERFLOWS
-If not ask an officer c: !

ncat will help you solve the challenge!

Getting started...

Log onto ndg machines...

Exploit c bug in program found at:
https://github.com/ufsit/cbugs-2-15-2016

Your objective is to get flag.txt