Visual Explanation of Memory Management in Different Languages

Carol (Nichols || Goulding)
@carols10cents

DISCLAIMER

"The metaphor ... explains what is happening in the low-level language so programmers can understand the consequences of the code they write."

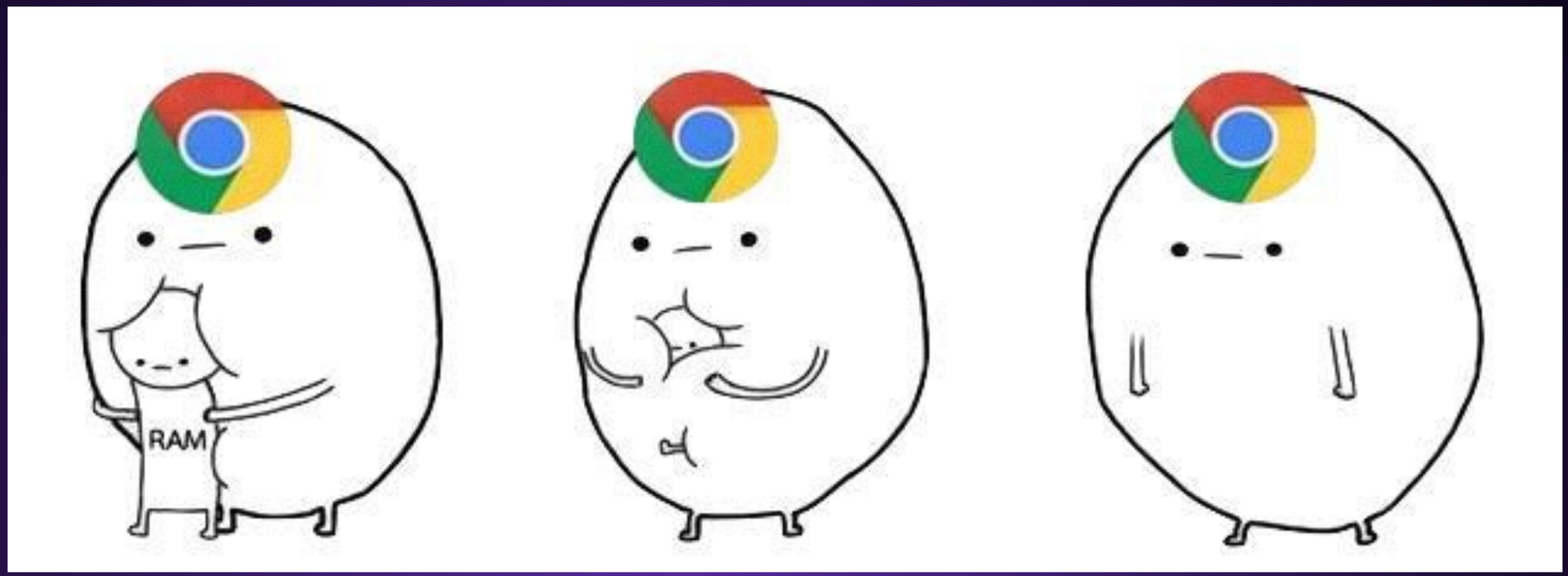
–*Metaphor in Computer Science, Colburn & Shute, 2008*
http://is.gd/metaphor_in_cs

Memory


CARLINI
100%
EXTRA
VIRGIN
OLIVE OIL
SALT
EARTH
Market District
Sweet Basil

Mehmory


EXTRA
VIRGIN


RAM

C

```
pot * p;

p = (pot *) malloc (sizeof(pot));

// plant, grow, then return the pot

free(p);
```

DVD = your data

DVD Player = chunk of memory

Happy path

```
// Ask for a place to put the DVD
dvd* player = malloc(sizeof(dvd));
// Use the DVD
*player = say_anything;
play(player);
// Put the DVD back
free(player);
```

Use after Free

Use after Free

```
dvd* player = malloc(sizeof(dvd));

*player = say_anything;

free(player);

play(player); // UNDEFINED!
```

Double Free

Double Free

```
dvd* player = malloc(sizeof(dvd));
free(player);

free(player); // ERROR!!
```

Memory leak

Memory leak

```
dvd* player = malloc(sizeof(dvd));

*player = say_anything;

play(player);

// No one ever puts the DVD away :(

// No one can use this DVD player :(
```

Not enough memory

Not enough memory

```
// Might not have enough space!
dvd* player = malloc(sizeof(dvd));
// Using it might fail!
*player = say_anything;
```

Ruby

Differences

- Automatic
- Less control
- Lots bigger!

Garbage collection <=2.0

- Stop-the-world
- Mark-and-sweep

Garbage collection

- Generational (2.1)
- Incremental marking (2.2)

Tuning GC

- RUBY_GC_HEAP_INIT_SLOTS
- RUBY_GC_HEAP_FREE_SLOTS
- RUBY_GC_HEAP_GROWTH_FACTOR
- RUBY_GC_HEAP_GROWTH_MAX_SLOTS
- RUBY_GC_HEAP_OLDOBJECT_LIMIT_FACTOR
- RUBY_GC_MALLOC_LIMIT
- RUBY_GC_MALLOC_LIMIT_MAX
- RUBY_GC_MALLOC_LIMIT_GROWTH_FACTOR
- RUBY_GC_OLDMALLOC_LIMIT
- RUBY_GC_OLDMALLOC_LIMIT_MAX
- RUBY_GC_OLDMALLOC_LIMIT_GROWTH_FACTOR

Rust

Features

- Still manual
- Safety guarantees
- Done at compile-time
- Ownership
- Does not prevent memory leaks

Automatically Drop

```
fn main() {

    let player = DvdPlayer {
        dvd: "Say Anything”
    };

    player.play();

} // player out of scope; dropped here
```

Moving

```
fn youplay(p: DvdPlayer) {
    p.play();
}
```

Moving

```
fn main() {
    let player = DvdPlayer {
        dvd: "Say Anything”
    };
    player.play();
    youplay(player); // Works!
}
```

Moving

```
fn main() {
    let player = DvdPlayer {
        dvd: "Say Anything”
    };
    youplay(player);
    player.play(); // error: use of moved value!
}
```

Moving

```
fn main() {
    let player = DvdPlayer {
        dvd: "Say Anything”
    };
    youplay(player);
    youplay(player); // error: use of moved value!
}
```

Borrowing

```
fn youplay(p: &DvdPlayer) {
    p.play();
}
```

Borrowing

```
fn youplay(p: &DvdPlayer) {
    p.play();
}
```

Borrowing

```
fn main() {
    let player = DvdPlayer {
        dvd: "Say Anything”
    };
    youplay(&player);
    youplay(&player); // works!!
    player.play(); // also works!!
} // now player is dropped.
```

Clone

```
fn main() {
    let player = DvdPlayer {
        dvd: "Say Anything”
    };
    play(player.clone());
    player.play();
}
```

But wait, there's more!

- Reference counting (python)
- Automatic Reference Counting (Objective-C)

Homework

- Stack vs Heap
- memorymanagement.org
- Leak memory in Rust or Ruby
- Try implementing a garbage collector!

References - C

- Memory Management in C by David Evans (video)
- Secure Coding in C and C++ by Robert C. Seacord, CMU SEI (book)

References - Ruby

- Presentation on Ruby Garbage Collection by Joe Damato and Aman Gupta, 2010
- Object Management on Ruby 2.1 by Koichi Sasada, 2013
- Ruby GC: Still Not Ready for Production by Tim Robertson, 2014
- Ruby 2.1 GC: Ready for Production by Sam Saffron, 2014
- Incremental GC in Ruby 2.2 by Koichi Sasada, 2015

References - Rust

- The Rust Programming Language Book chapter on Ownership
- Rust Ownership, The Hard Way by Chris Morgan
- Explore the Ownership System in Rust by Nercury
- Understanding Pointers, Ownership, and Lifetimes in Rust by Paul Körbitz
- Rust's Ownership model for JavaScript developers by Christoph Burgdorf

Thanks! <3

@carols10cents