CS 140e

Operating Systems from the Ground Up

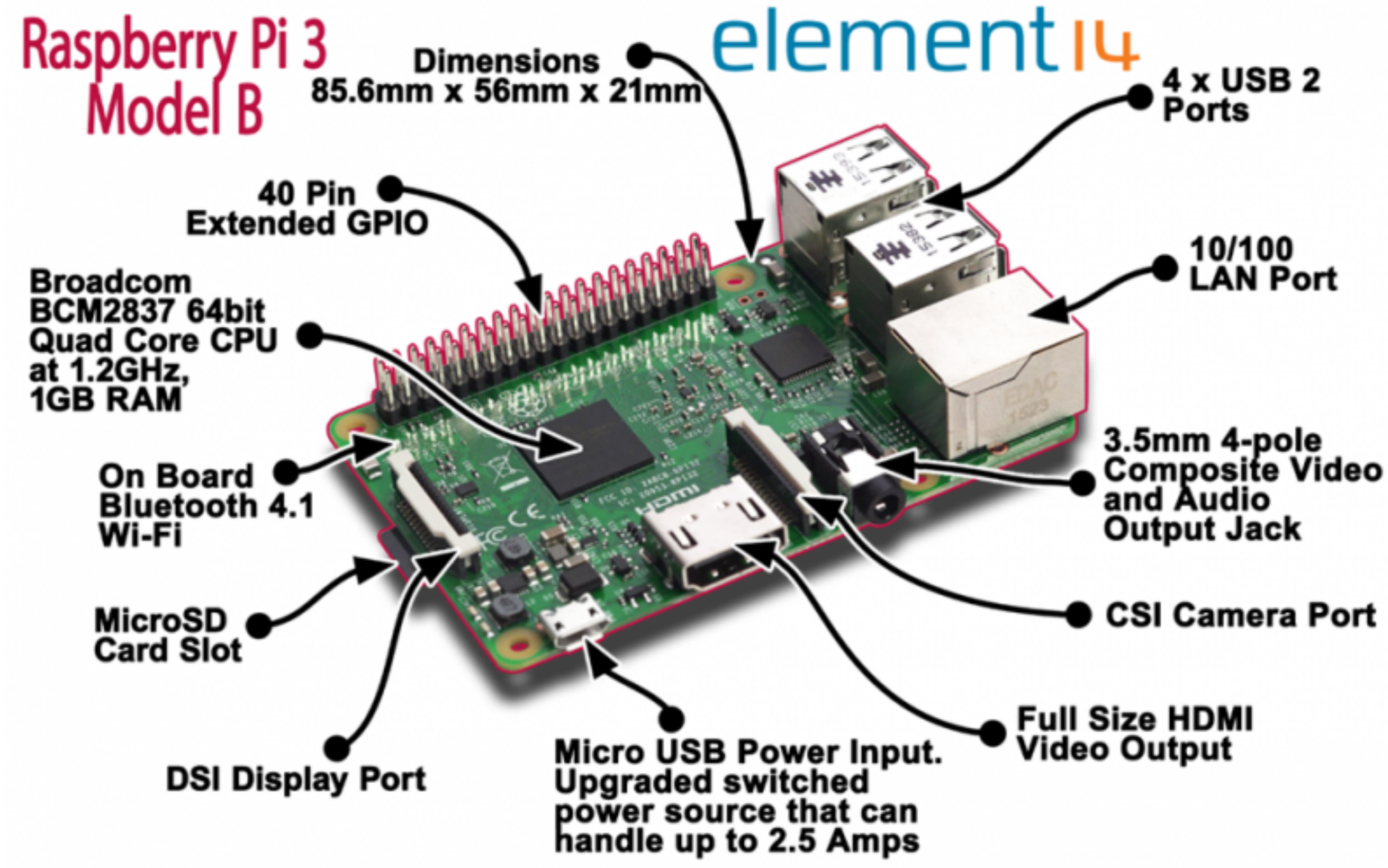
CS 140e Goals

- **Learn OS fundamentals.**
  - Disks, File systems, I/O
  - Threads & Processes
  - Scheduling
  - Virtual Memory
  - Protection & Security
  - Interrupts
  - Concurrency & Synchronization
- **Build an OS from scratch.**
- **Learn embedded development fundamentals.**
- **Write low-level embedded software and drivers.**

About CS 140e: Why?

- **Brand new Operating Systems course!**
  - *All* course material is new.
  - Not based off any existing course.
  - …expect rough edges.
- **Everything is *real*.**
  - No virtual machines. All software runs on Raspberry Pi 3.
- **A bottom-up, *from scratch* approach.**
  - *You* write vast majority of software.
- **A *modern* approach.**
  - Multicore, 64-bit ARMv8 platform.
  - Modern programming languages and tools.

About CS 140e: Why?



Who *We* Are

Jennifer Lin

Your TA!

Who *We* Are

Dawson Engler

Professor, Instructor

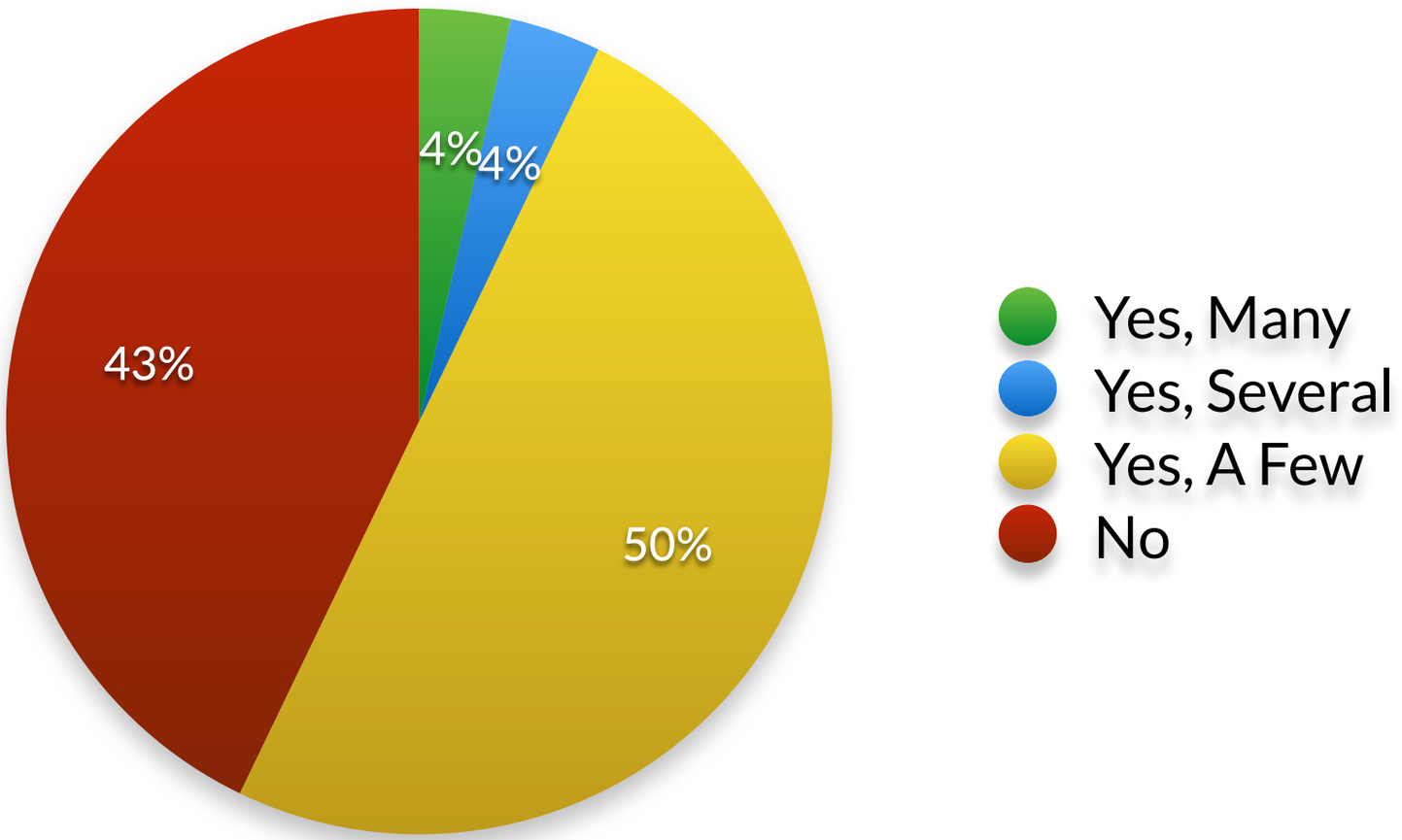
Who *We* Are

Sergio Benitez

Ph.D., Instructor

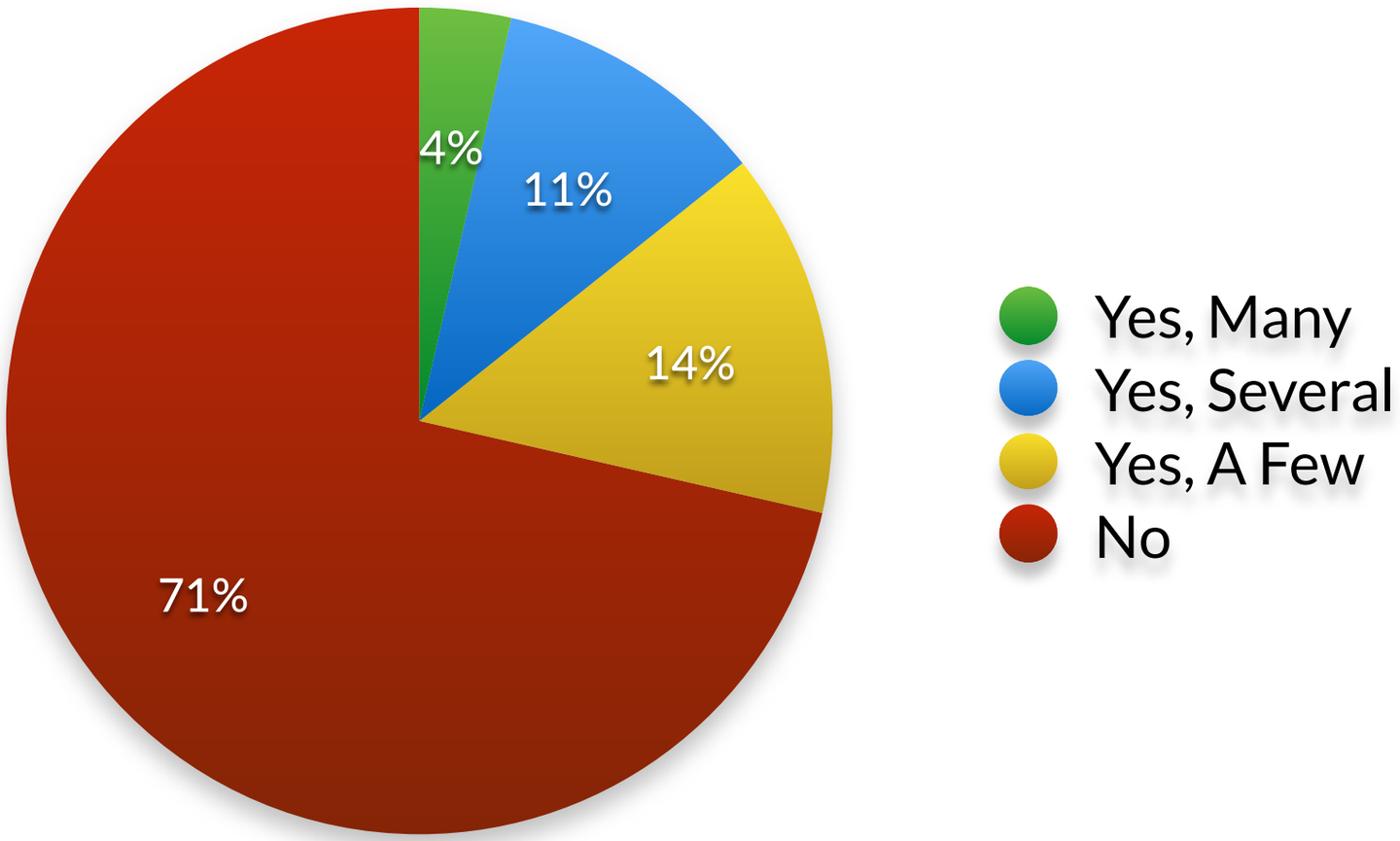
Who *You* Are

Have you implemented any operating system mechanisms such as virtual memory support, processes, or threads?



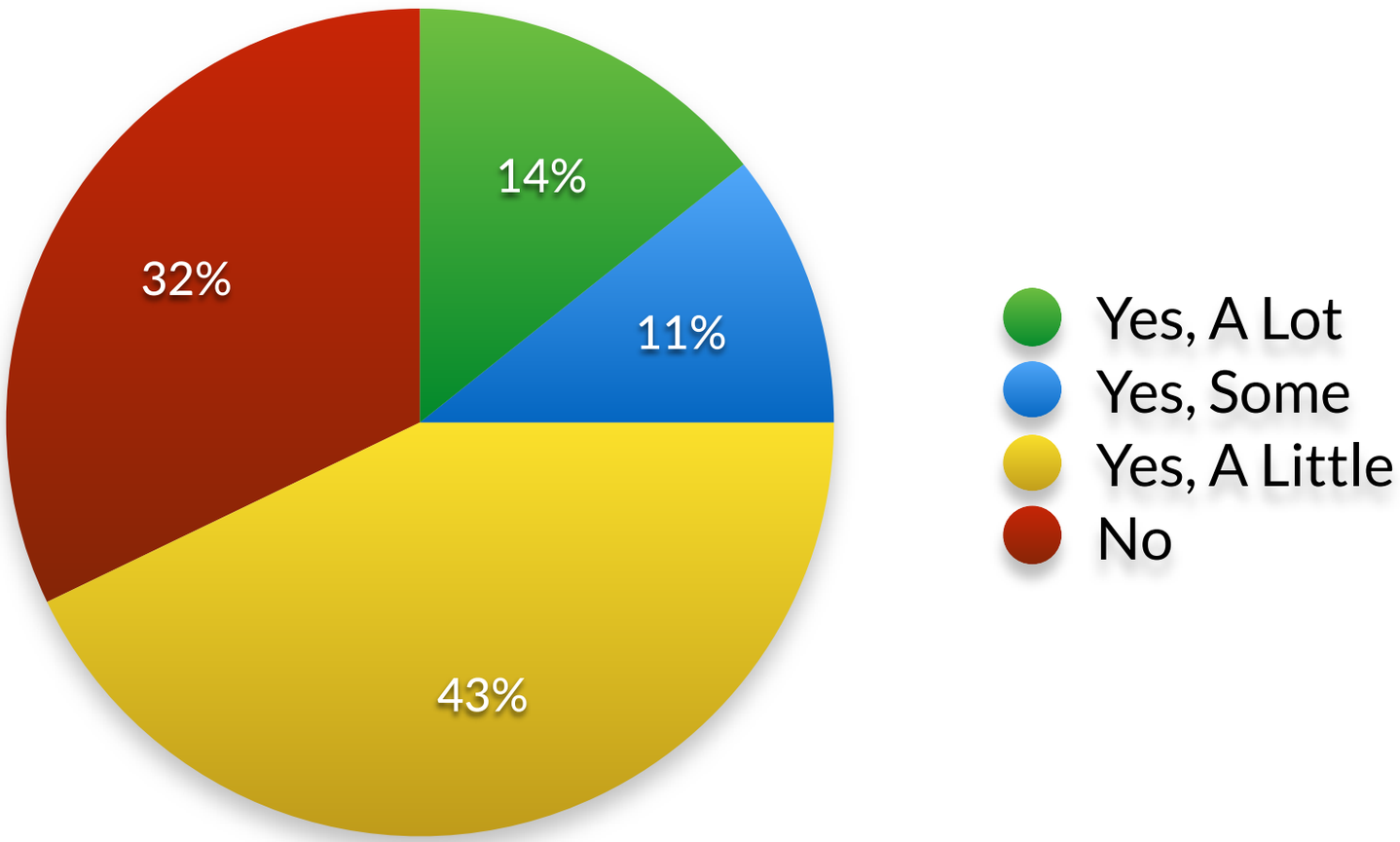
Who *You* Are

Have you implemented any bare metal software such as bootloaders or device drivers?



Who *You* Are

Do you have any experience with embedded software development?



Administrivia

- **Lectures**
  - Mondays and Wednesday, 3:00pm - 4:20pm (160-124)
  - Focus on theoretical/conceptual OS concepts
- **Labs (optional but encouraged)**
  - Mondays and Wednesday, 5:30pm - 6:30pm (Gates 463a)
  - Focus on assignments.
- **Course Website: https://cs140e.stanford.edu**
  - News, assignments, policies, etc.
  - Still a work in progress.
- **Q&A on Piazza (see course website for link)**
- **Office Hours TBA.**

Assignments and Exams

- **Midterm and Final Exam**
  - Focus primarily on lecture material.
  - Open notes. Anything written/printed is allowed.
- **5 Heavy (!!) Programming Assignments**
  - Budget about 10 - 15 hours per week per assignment.
  - First assignment out on Wednesday.
  - Writing component: answer questions.
  - Individual, but collaboration on ideas is encouraged.
  - All code must be *your own*. *Please* don't cheat.
  - Start early. Don't procrastinate!
- **Late Policy: 100 penalized hours per assignment**
  - Does not apply to last assignment.
  - Score *capped* at max(0, 100 - n)% for n hours late

Assignment Schedule

- **Assignment 0: Getting Started**
  - Set-up Raspberry Pi. Write first program.
  - Out on Wednesday. Due Monday.
- **Assignment 1: The Shell**
  - Write a bash-like shell that runs on your machine *and* Pi.
  - Write a shell command that loads program over UART: “bootloader”.
- **Assignment 2: File System**
  - Write SD card driver and FAT32 file system.
- **Assignment 3: Spawn**
  - Load and execute programs from SD card in separate processes.
- **Assignment 4: Preemptive Multicore Multitasking**
  - Run multiple programs at once on multiple cores.

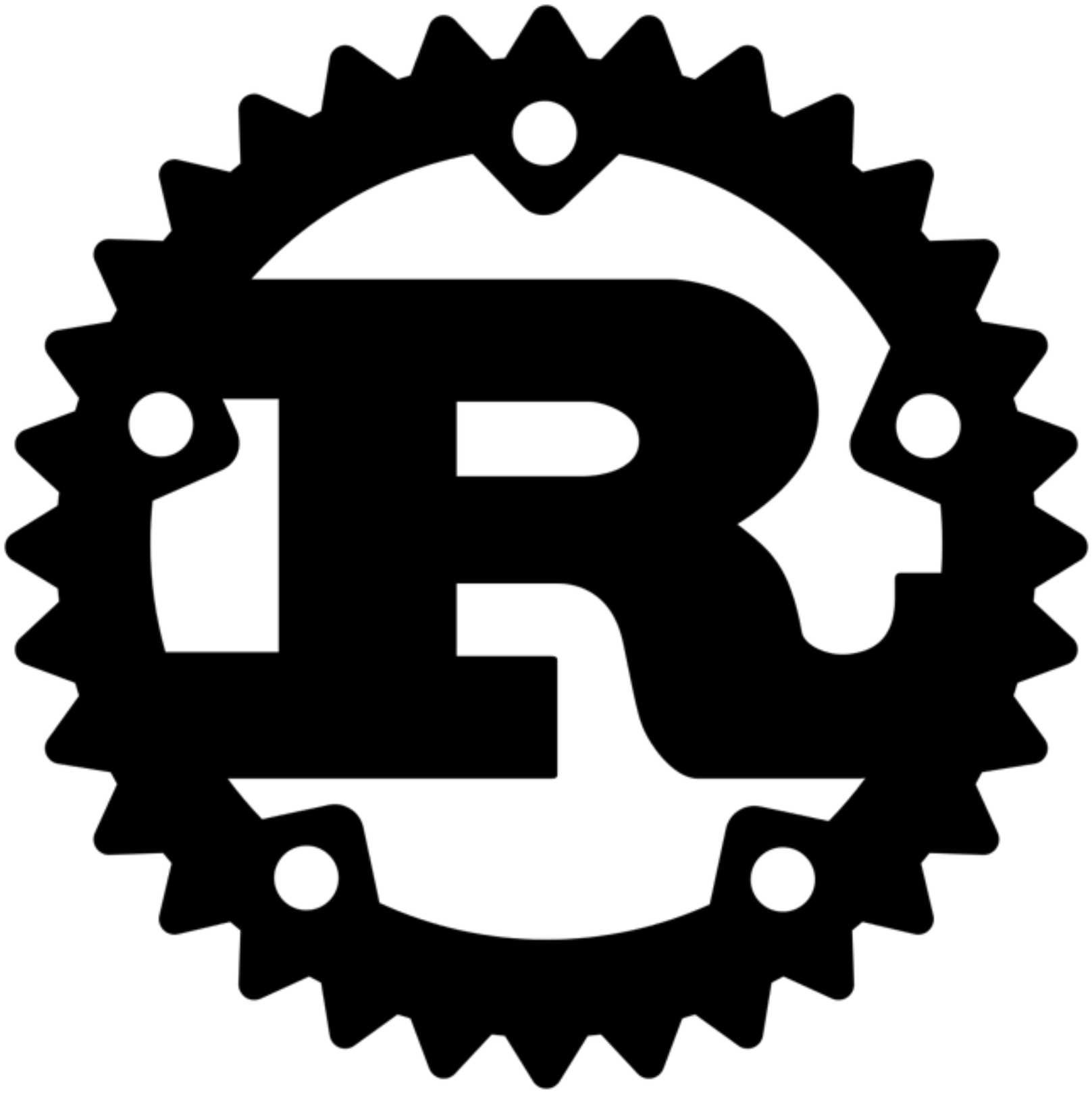
Grading

- **Grading Breakdown**
  - 55% assignments
  - 20% midterm
  - 25% final exam
  - (+/-) 5% participation
- **We hope to gives lots of As!**

Who *You* Are

In which programming languages do you feel comfortable and confident writing software?

C, C++, Java, Python, JavaScript

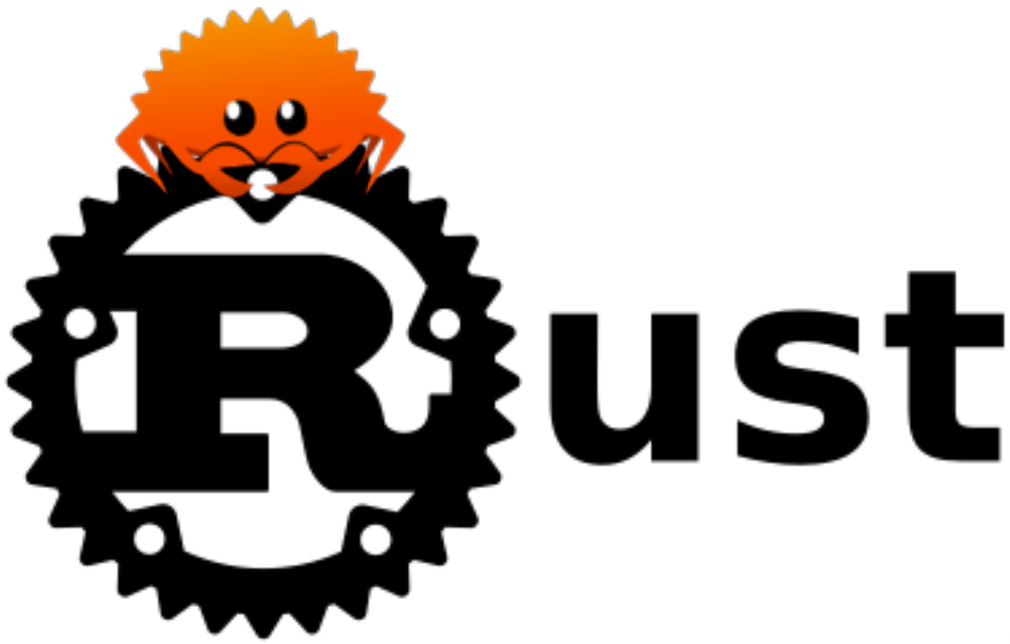


Rust!

In which programming languages do you feel comfortable and confident writing software?

C, C++, Java, Python, JavaScript

Rust!



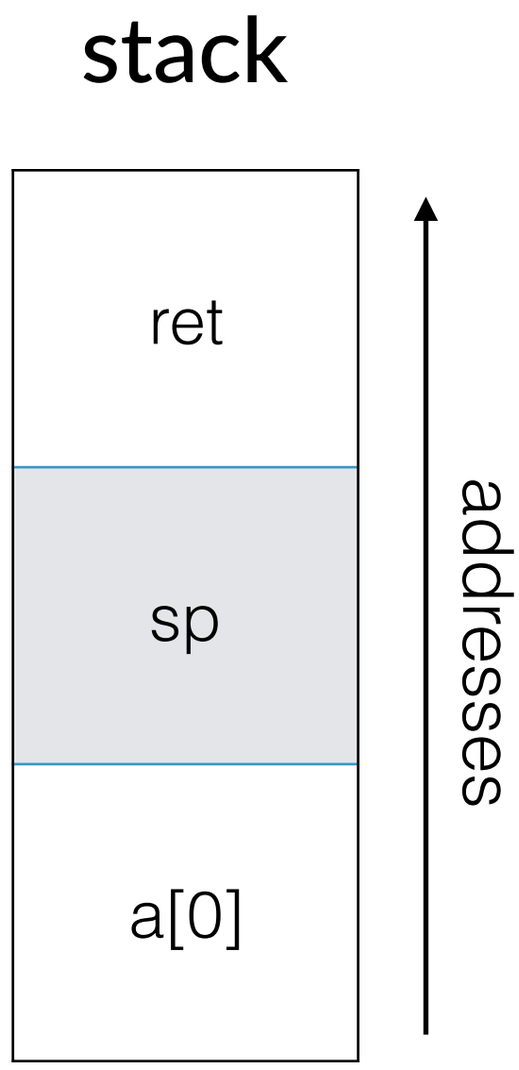
- Memory-safe without a GC.
- No data races, guaranteed!
- Minimal runtime.
- Static, strong types.
- Package manager.

C: Undefined Behavior

```
int main(int argc, char **argv) {
    unsigned long a[1];
    a[2] = 0x7ffff7b36cebUL;
    return 0;
}
```

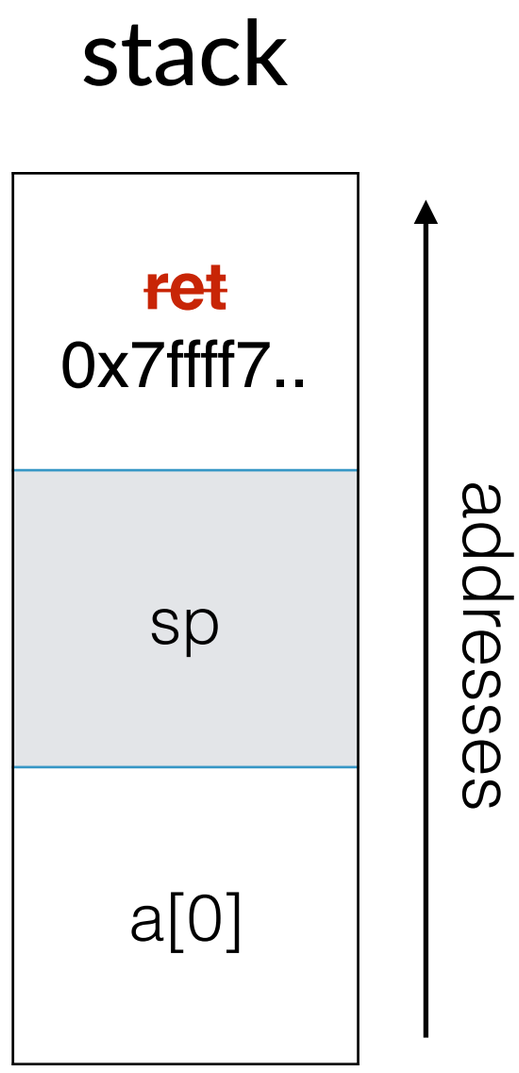
C: Undefined Behavior

```
int main(int argc, char **argv) {
    unsigned long a[1];
    a[2] = 0x7ffff7b36cebUL;
    return 0;
}
```



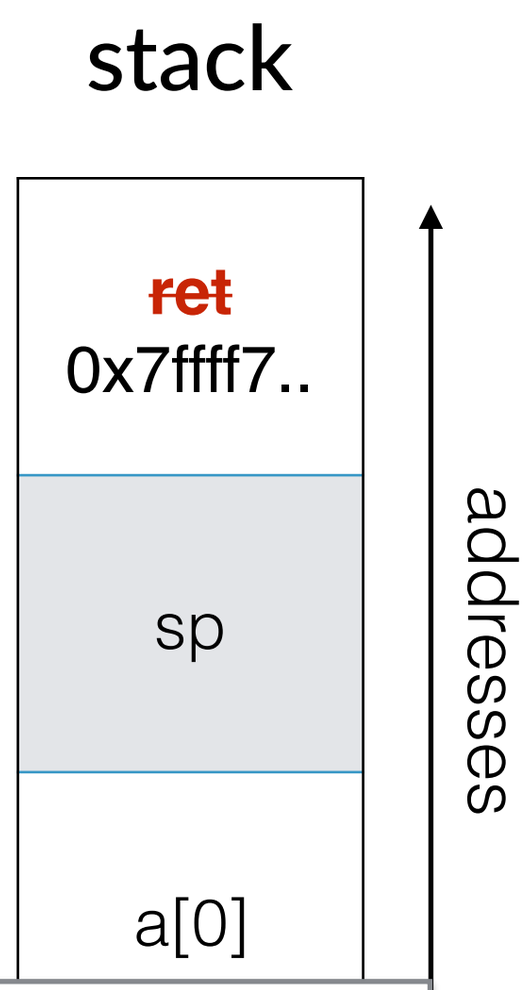
C: Undefined Behavior

```c
int main(int argc, char **argv) {
    unsigned long a[1];
    a[2] = 0x7ffff7b36cebUL;
    return 0;
}
```



C: Undefined Behavior

```c
int main(int argc, char **argv) {
    unsigned long a[1];
    a[2] = 0x7ffff7b36cebUL;
    return 0;
}
```




```
undef: Error: .netrc file is readable by others.
undef: Remove password or make file unreadable by others.
```

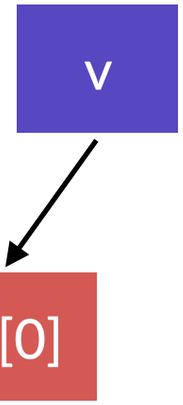
Aliasing is Hard!

```
int main() {
    std::vector<std::string> v;

    v.push_back("Hello, ");

    std::string &x = v[0];

    v.push_back(" world!");

    std::cout << x;
}
```

Rusty Types for Solid Safety (*PLAS'16*)

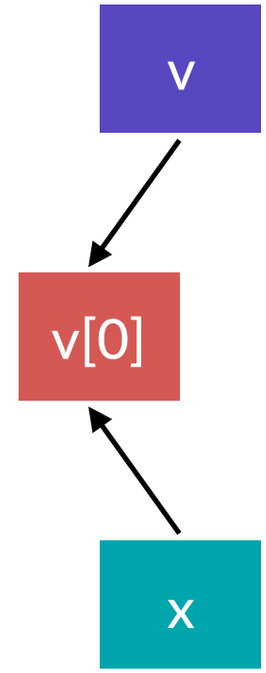
Aliasing is Hard!

```
int main() {
    std::vector<std::string> v;

    v.push_back("Hello, ");

    std::string &x = v[0];

    v.push_back(" world!");

    std::cout << x;
}
```



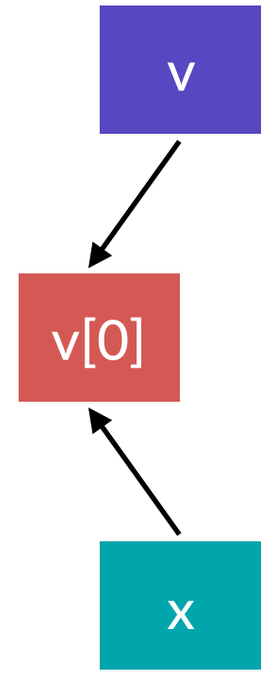
Aliasing is Hard!

```
int main() {
    std::vector<std::string> v;

    v.push_back("Hello, ");

    std::string &x = v[0];

    v.push_back(" world!");

    std::cout << x;
}
```



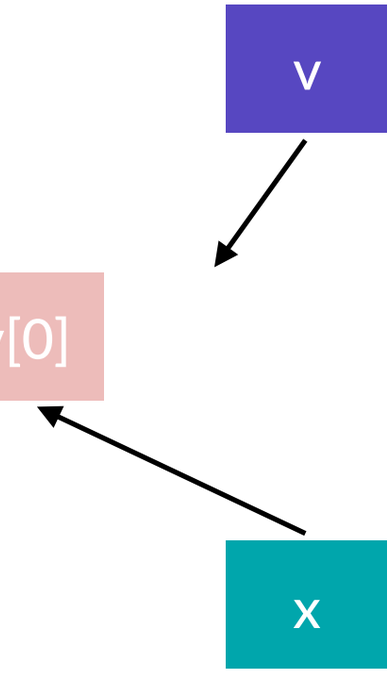
Aliasing is Hard!

```
int main() {
    std::vector<std::string> v;

    v.push_back("Hello, ");

    std::string &x = v[0];

    v.push_back(" world!");

    std::cout << x;
}
```



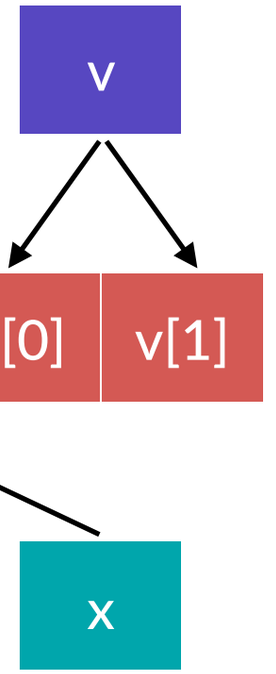
Aliasing is Hard!

```
int main() {
    std::vector<std::string> v;

    v.push_back("Hello, ");

    std::string &x = v[0];

    v.push_back(" world!");

    std::cout << x;
}
```



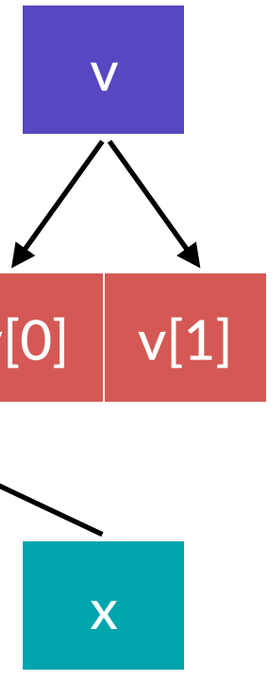
Aliasing is Hard!

```
int main() {
    std::vector<std::string> v;

    v.push_back("Hello, ");

    std::string &x = v[0];

    v.push_back(" world!");

    std::cout << x;
}
```



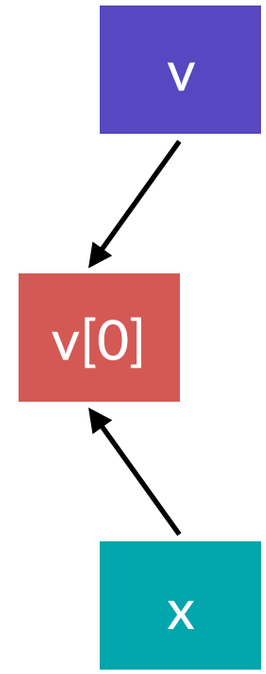
Aliasing is Hard!

```
int main() {
    std::vector<std::string> v;

    v.push_back("Hello, ");

    std::string &x = v[0];

    v.push_back(" world!");

    std::cout << x;
}
```



Aliasing is Hard!

```
int main() {
    std::vector<std::string> v;

    v.push_back("Hello, ");

    std::string &x = v[0];

    v.push_back(" world!");

    std::cout << x;
}
```



Restricted Aliasing

C++

```
int main() {
    vector<string> v;

    v.push_back("Hello, ");

    string &x = v[0];

    v.push_back(" world!");

    cout << x;
}
```

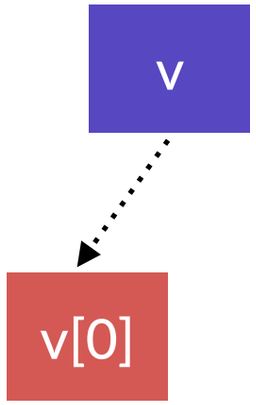
Rust

```
fn main() {
    let mut v = Vec::new();

    v.push("Hello, ");

    let x = &v[0];

    v.push(" world!");

    println!("{}", x);
}
```

Rusty Types for Solid Safety (*PLAS'16*)

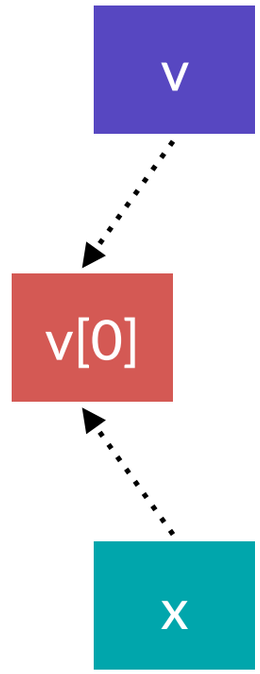
Restricted Aliasing

```
fn main() {
    let mut v = Vec::new();

    v.push("Hello, ");

    let x = &v[0];

    v.push(" world!");

    println!("{}", x);
}
```



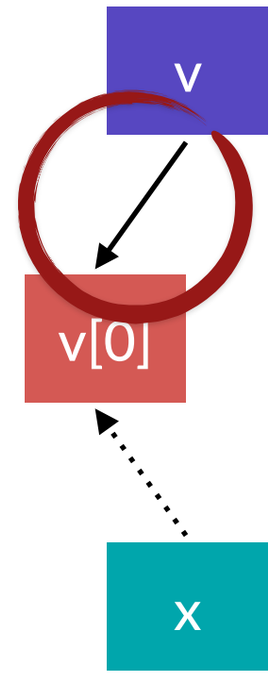
Restricted Aliasing

```
fn main() {
    let mut v = Vec::new();

    v.push("Hello, ");

    let x = &v[0];

    v.push(" world!");

    println!("{}", x);
}
```



Restricted Aliasing

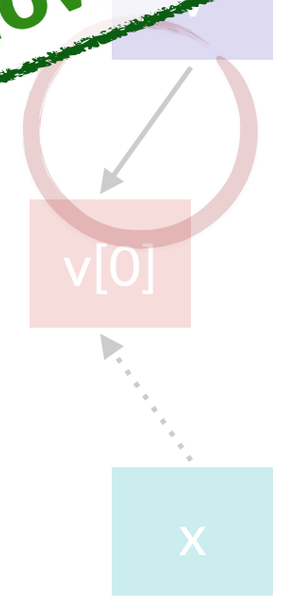
```
fn main() {
    let mut v = Vec::new();

    v.push("Hello, ");

    let x = &v[0];

    v.push(" world!");

    println!("{}", x);
}
```



Restricted Aliasing

```
fn main() {
    let mut v = Vec::new();

    v.push("Hello, ");

    let x = &v[0];

    v.push("world!");

    println!("{}", x);
}
```

breaks invariant => statically disallowed



Rusty Types for Solid Safety (*PLAS'16*)