

Ownership

ECS 189C Lecture 8 – Spring 2024

A Word of Warning

```
error[E0502]: cannot borrow `v` as mutable  
because it is also borrowed as immutable
```



- Some learning curve!
- The compiler will complain about your code that would work in any other language
- Harder to appreciate if you have not spent time debugging C/C++

Ownership

Memory management errors in languages like C and C++ can often be attributed to **ownership mistakes**



Ownership Mistakes: Quiz 1

```
void ex1() {  
    vector<int> vec = { 1, 2, 3 };  
  
    int j = 0;  
    for (auto it = vec.begin(); it != vec.end(); ++it) {  
        vec.push_back(j);  
        j++;  
        cout << j << ", ";  
    }  
}
```

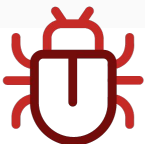
Ownership Mistakes: Quiz 1



```
void ex1() {  
    vector<int> vec = { 1, 2, 3 };  
  
    int j = 0;  
    for (auto it = vec.begin(); it != vec.end(); ++it) {  
        vec.push_back(j);  
        j++;  
        cout << j << ", ";  
    }  
}
```

it expects
temporary
“ownership”
over vec

vec retains
ownership



Fails at runtime if vec is resized!



Ownership Mistakes: Quiz 2

```
void ex2() {  
    vector<int> vec = { 1, 2, 3 };  
  
    // Find location of 2  
    std::vector<int>::iterator item;  
    item = std::find(vec.begin(), vec.end(), 2);  
  
    vec.clear();  
    vec.push_back(4);  
    vec.push_back(5);  
    vec.push_back(6);  
  
    std::cout << "Found:" << *item << std::endl;  
}
```



Ownership Mistakes: Quiz 2

```
void ex2() {  
    vector<int> vec = { 1, 2, 3 };  
  
    // Find location of 2  
    std::vector<int>::iterator item;  
    item = std::find(vec.begin(), vec.end(), 2);  
  
    vec.clear();  
    vec.push_back(4);  
    vec.push_back(5);  
    vec.push_back(6);  
  
    std::cout << "Found:" << *item << std::endl;  
}
```

item expects temporary
ownership

Original owner deletes
element



Output: 5



Ownership Mistakes: Quiz 3

```
void ex3() {  
    vector<vector<int>*> grid;  
  
    grid.push_back(new vector<int>({ 1, 2, 3 }));  
    grid.push_back(new vector<int>({ 4, 5, 6 }));  
    grid.push_back(new vector<int>({ 7, 8, 9 }));  
  
    for (auto row: grid) {  
        for (auto col: *row) {  
            std::cout << col << " ";  
        }  
        std::cout << std::endl;  
    }  
  
    grid.clear();  
}
```




Ownership Mistakes: Quiz 3

```
void ex3() {  
    vector<vector<int>*> grid;  
  
    grid.push_back(new vector<int>({ 1, 2, 3 }));  
    grid.push_back(new vector<int>({ 4, 5, 6 }));  
    grid.push_back(new vector<int>({ 7, 8, 9 }));  
  
    for (auto row: grid) {  
        for (auto col: *row) {  
            std::cout << col << " ";  
        }  
        std::cout << std::endl;  
    }  
  
    grid.clear();  
}
```

Allocated vector expects
grid to take ownership

grid doesn't assume ownership,
releases resources



Memory leak

valgrind --leak-check=full ./example

LEAK SUMMARY:

definitely lost: 72 bytes in 3 blocks
indirectly lost: 36 bytes in 3 blocks

How can languages prevent memory management bugs?

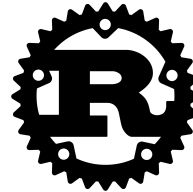
- Garbage collection



- Recommend specific programming patterns

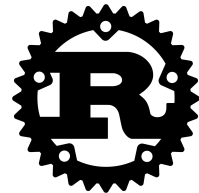


- Analyze code to catch bugs at compile time



How can languages prevent memory management bugs?

- In Rust, ownership is a static, syntactic property (like a type).
- Rust uses ownership to analyze code to catch bugs at compile time.



What is Ownership?

1. Each value in Rust has a variable that's called its owner.
2. There can only be one owner at a time.
3. When the owner goes out of scope, the value will be dropped.

```
fn ex1() {  
    let mut v = vec![1, 2, 3];  
  
    let mut j = 0;  
    for x in &v {  
        v.push(j);  
        j += 1;  
        println!("{}", j, x);  
    }  
    v.push(3);  
}
```

What is Ownership?

1. Each value in Rust has a variable that's called its owner.
2. There can only be one owner at a time.
3. When the owner goes out of scope, the value will be dropped.

```
fn ex1() {  
    let mut v = vec![1, 2, 3];  
  
    let mut j = 0;  
    for x in &v {  
        v.push(j);  
        j += 1;  
        println!("{}", j, x);  
    }  
    v.push(3);  
}
```

What is Ownership?

1. Each value in Rust has a variable that's called its owner.
2. There can only be one owner at a time.
3. When the owner goes out of scope, the value will be dropped.

```
fn ex1() {  
    let mut v = vec![1, 2, 3];  
  
    let mut j = 0;  
    for x in &v {  
        v.push(j);  
        j += 1;  
        println!("{}", j, x);  
    }  
    v.push(3);  
}
```

Type Error

What is Ownership?

1. Each value in Rust has a variable that's called its owner.
2. There can only be one owner at a time.
3. When the owner goes out of scope, the value will be dropped.

```
fn ex1() {  
    let mut v = vec![1, 2, 3];  
  
    let mut j = 0;  
    for x in &v {  
        v.push(j);  
        j += 1;  
        println!("{}", j, x);  
    }  
    v.push(3);  
}
```

_it dropped

v dropped

What is Ownership?


1. Each value in Rust has a variable that's called its owner.
2. There can only be one owner at a time.
3. When the owner goes out of scope, the value will be dropped.

```
fn ex1() {  
    let mut v = vec![1, 2, 3];  
  
    let mut j = 0;  
    for x in &v {  
        v.push(j);  
        j += 1;  
        println!("{}", j, x);  
    }  
    v.push(3);  
}
```

error[E0502]: cannot borrow `v` as mutable because it is also borrowed as immutable

Final notes: Ownership

- **Safety**
- **Errors at compile time**
- **Zero overhead abstraction**
- **Opt-out**
 - Rust interfaces with C code
 - Unsafe code
 - Abstractions to hide ownership



More on this in
later lectures

Demo / Coding

We can use ownership to:

- Understand references and borrowing
- Fix the quiz examples

References

<https://stackoverflow.com/questions/5638323>

<https://doc.rust-lang.org/book/ch04-01-what-is-ownership.html>

<https://jaxenter.com/most-difficult-programming-languages-152590.html>