Rust: ownership, references, slicing

Madhavan Mukund, S P Suresh

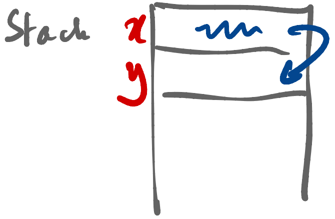
Programming Language Concepts

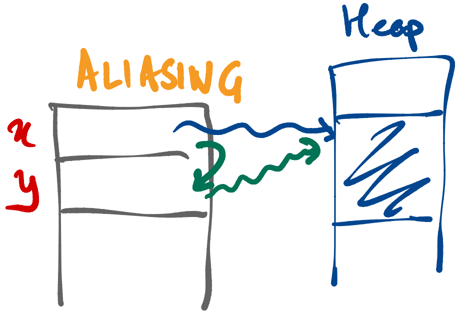
Lecture 10, 13 February 2024

Ownership, references, slicing

Stack & Heap

$y = x$ – "Copy" in stack





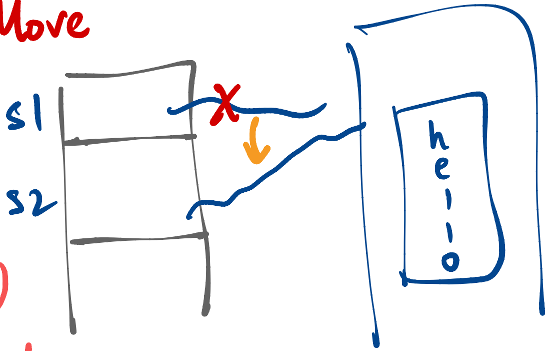
Ownership, references, slicing

Every value has an owner – unique!

– No "garbage"

"Move"

s2 = s1



A value is deallocated (restored to free space) when owner's scope ends

Ownership, references, slicing

Trait = Rust equivalent of Java Interface
Haskell type class

Copy trait — value is copied, not moved

└ all scalar types : i32, u64, f32, bool, char

Ownership, references, slicing

Call a function with a parameter on the heap

- Value "moves" to the function
- Function needs to return it back

... $\underset{\longleftrightarrow}{\text{Tedious}}$

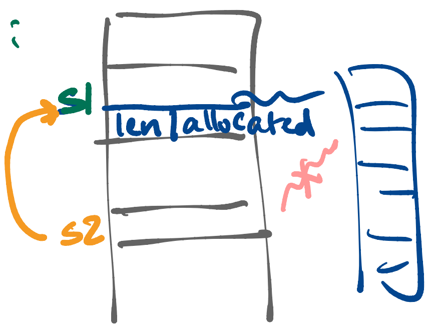
Ownership, references, slicing

References

&v is a reference to v

```
let s1 = String::from("hello");
let s2 = &s1;
```

"Borrow"


s1
len | allocated
s2