从C++到Rust，直接读写内存，vec内存布局

原创 Ajonbin AJonbin的杂货铺 2024年02月20日 23:05 美国

对一个C++程序员来说，直接读写内存是很常见的，这也是C++特别灵活和强大的地方。

Rust将自己定位是一门系统语言，也必须能直接读写内存。

之前讲了Rust的原始指针，今天就来看看怎么用Raw Pointer和unsafe代码块来按地址读写内存。

今天就用std::vec作为一个例子

- 先来看看Rust中vec对象的内存布局
- 再通过内存地址直接修改vec中的值

先看段代码

```rust
fn main(){
    let mut v = Vec::new();
    v.push(11);
    v.push(22);
    println!("{:?}",v)
}
```


这段代码很直白，先创建了一个Vector，加入两个元素，11和22，最后打印这个vector。

输出的结果是

Standard Output

```
[11, 22]
```

现在我们就来看看这个变量v在内存中是怎么表示的。

老样子先上完整代码

```
1  use std::mem;
2
3  fn main(){
4      let mut v = Vec::new();
5      v.push(11);
6      v.push(22);
7      println!("{:?}",v);
8
9      let v_size = mem::size_of::<Vec<u32>>();
10     println!("v_size = {} bytes",v_size);
11
12     println!("address of v = {:p}", &v);
13     println!("address of v[0] = {:p}", &v[0]);
14
15     let p:[u64;3];
16     p = unsafe{
17         mem::transmute(v)
18     };
19
20     println!("p ==> {:x},{:x},{:x}", p[0],p[1],p[2]);
21 }
```


再贴结果

```
Standard Output

[11, 22]
v_size = 24 bytes
address of v = 0x7ffd07edc208
address of v[0] = 0x55e3a30699d0
p ==> 4,55e3a30699d0,2
```


第9行，标记1

```
let v_size = mem::size_of::<Vec<u32>>();
```

先调用std::mem::size_of<T>函数来看看vector对象v的大小，得到

```
v_size = 24 bytes
```

v的大小是24个字节，由于是64位系统，这样我们就知道vector对象v的大小是3个64位变量。

第12和13行，标记2和3

```
println!("address of v = {:p}", &v);
println!("address of v[0] = {:p}", &v[0]);
```

这两行分别打印v和v第一个元素v[0]的地址。

```
address of v = 0x7ffd07edc208
address of v[0] = 0x55e3a30699d0
```

可以看到v的地址是0x7fxxx，这是一个栈stack上的地址，v[0]的地址是0x55xxx，这是在堆heap上的地址。

可见v是一个栈上变量，而实际的元素是在堆上创建的。

第15行，标记4

```
let p:[u64;3];
```

这里申明了一个变量p，是一个包含3个u64元素的数组。由于变量v是一个包含3个64位的变量，我们就把这3个值取出来，赋值给p。

第16-17行，标记5和6

```
p = unsafe{
    mem::transmute(v)
};
```


我们调用std::mem::stransmute函数，将v的值按bit拷贝给p。由于这个操作是违背Rust的安全原则的，所以必须将它包含在unsafe代码段中执行。

第20行，标记7

```
println!("p ==> {:x},{:x},{:x}", p[0],p[1],p[2]);
```

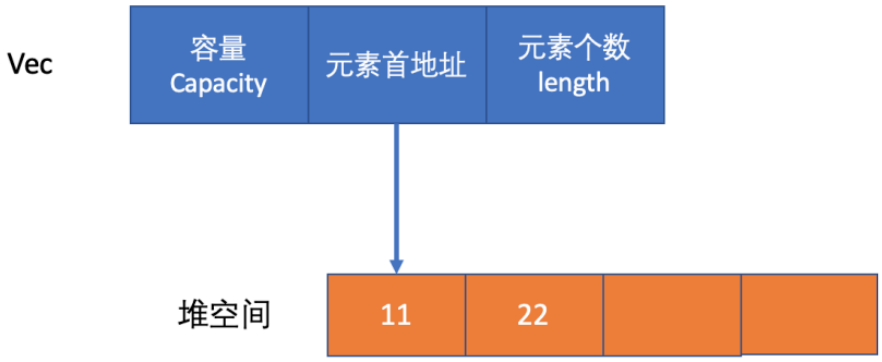
我们将p的元素打印出来

```
p ==> 4,55e3a30699d0,2
```

得到3个u64，结合Vec的定义，我们可以得到：

- 4 --> v的容量capacity
- 55e3a30699d0 --> 元素首地址
- 2 --> v的长度，也就是元素的个数

这样，我们就知道vector变量的内存布局大致是这样的



明天讲讲如何通过指针修改v的元素的值。