소유권

2020.3

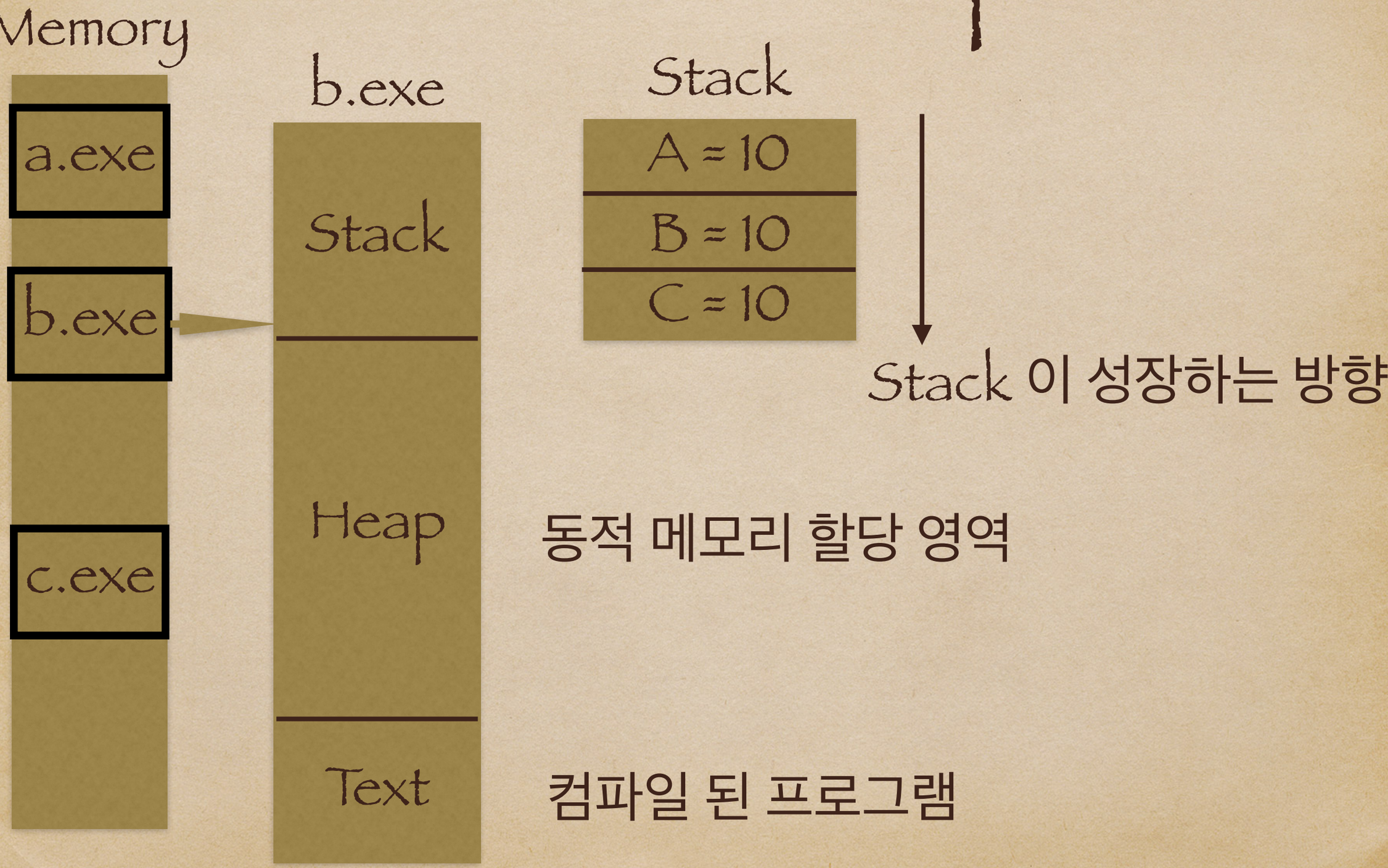
소유권~1

- 가비지 콜렉터
- 메모리 안정성

소유권-2

- 메모리의 명시적 해제, 가비지 콜렉터
- 스택과 힙
- 오너와 스코프
- drop
- Move(얕은 복사), clone(깊은 복사)
- Copy trait
- 함수 파라메터와 반환값

Stack & heap


Memory
a.exe
b.exe
c.exe
b.exe
Stack
Heap
Text
Stack
A = 10
B = 10
C = 10
Stack 이 성장하는 방향
동적 메모리 할당 영역
컴파일 된 프로그램

오너와 스코프

- 오너는 오브젝트의 소유권 자
- Let a = 10; // 변수 a는 오브젝트 10의 소유권자
- 스코프

```
Fn foo() {
    Let a = 10;
    {
        Let a = 20;
            {
                Let a = 30;
            }
        }
    }
}
```

레퍼런스와 빌림

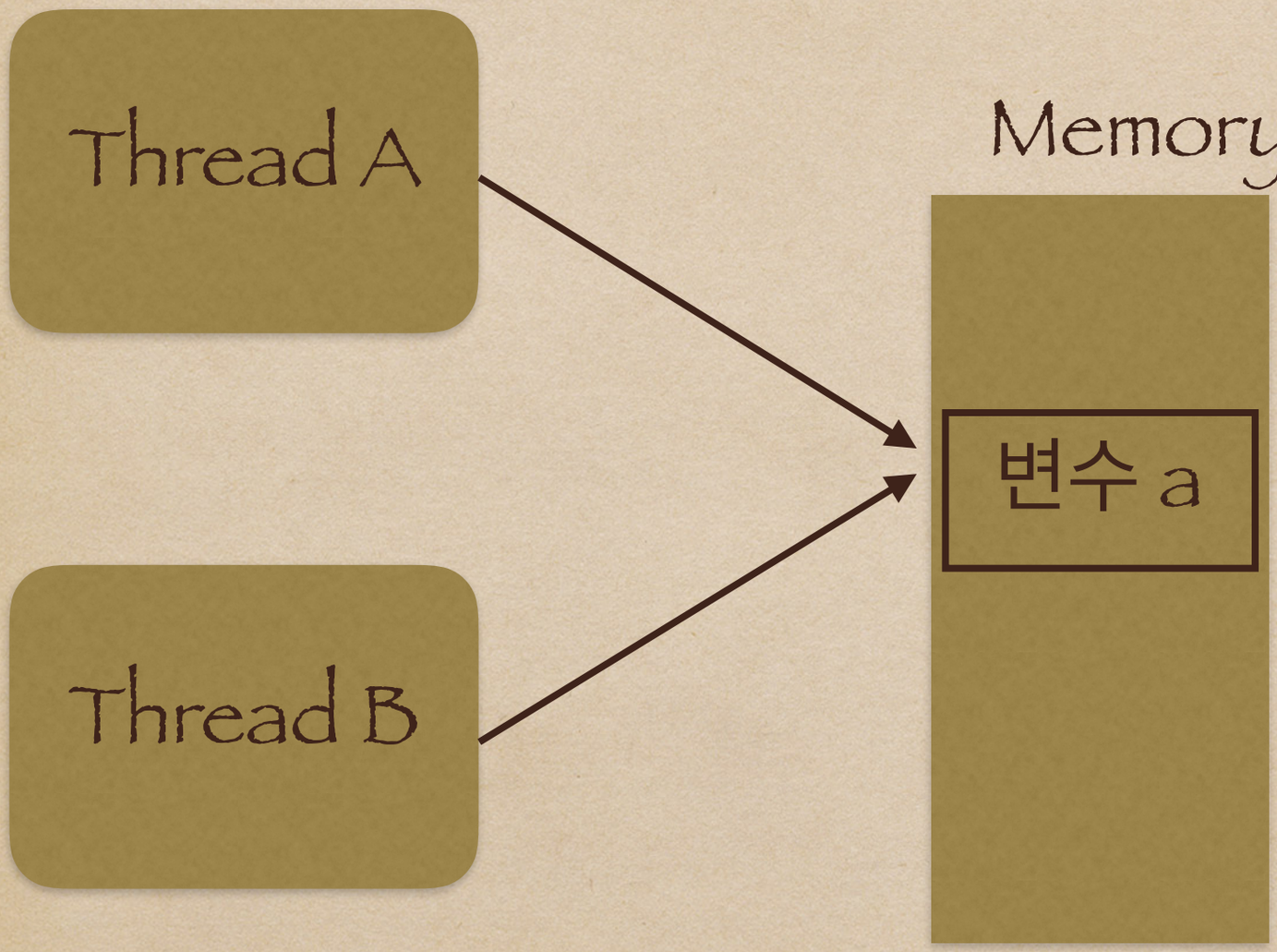
- 포인터
- 가변 참조자
- Data race
- Dangling Reference or pointer
- 참조 규칙(오직 하나의 가변참조, 다수의 불변 참조)

레퍼런스

- Let a = 10;
- Let b = &a;
- *b = 10 // compile error

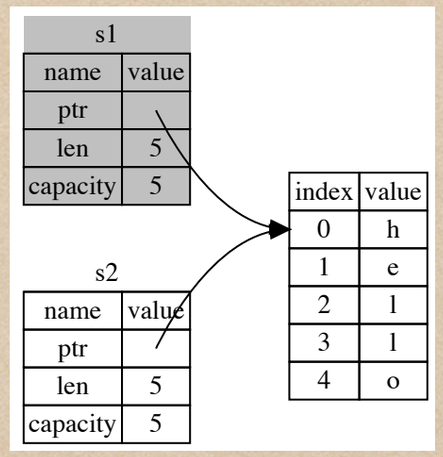
- Let mut a = 10;
- Let b = &mut a;
- *b = 20; // no error

Data race



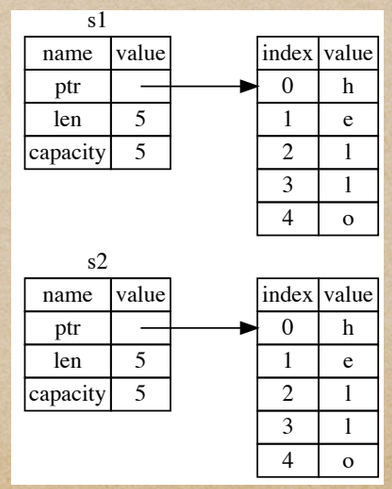
얕은 복사 vs. 깊은 복사

- Let s1 = “hello”.to_string();
- Let s2 = s1;



- Let a = 10; // compile time에 크기를 알 수 있다.
- Let b = a;

- Let s1 = “hello”.to_string();
- Let s2 = s1.clone();



Copy trait

```
fn main() {
    let s = String::from("hello");  // s가 스코프 안으로 들어왔습니다.

    takes_ownership(s);             // s의 값이 함수 안으로 이동했습니다...
                                    // ... 그리고 이제 더이상 유효하지 않습니다.
    let x = 5;                      // x가 스코프 안으로 들어왔습니다.

    makes_copy(x);                  // x가 함수 안으로 이동했습니다만,
                                    // i32는 Copy가 되므로, x를 이후에 계속
                                    // 사용해도 됩니다.

} // 여기서 x는 스코프 밖으로 나가고, s도 그 후 나갑니다. 하지만 s는 이미 이동되었으므로,
  // 별다른 일이 발생하지 않습니다.

fn takes_ownership(some_string: String) { // some_string이 스코프 안으로 들어왔습니다.
    println!("{}", some_string);
} // 여기서 some_string이 스코프 밖으로 벗어났고 `drop`이 호출됩니다. 메모리는
  // 해제되었습니다.

fn makes_copy(some_integer: i32) { // some_integer이 스코프 안으로 들어왔습니다.
    println!("{}", some_integer);
} // 여기서 some_integer가 스코프 밖으로 벗어났습니다. 별다른 일은 발생하지 않습니다.
```

함수의 호출

- Fn foo() {
  Let a = “hello”.to_string();
  Let b = bar(a);
  bar(a) // a는 이미 이동했으므로 compile error
}

Fn bar(a: String) -> String{
  A
}

슬라이스

```
let s = String::from("hello world");

let hello = &s[0..5];
let world = &s[6..11];
```

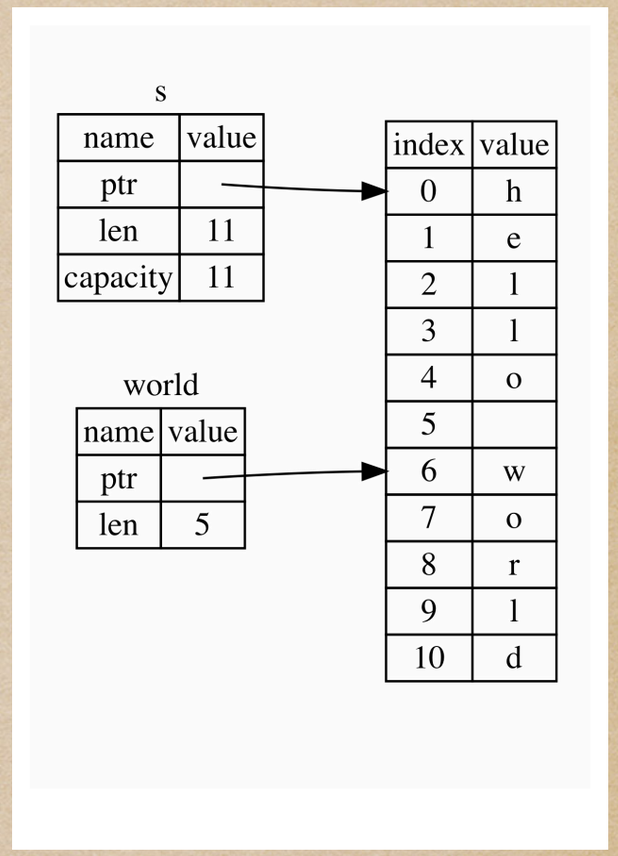
스트링 리터럴

```
let s = "Hello, world!”;
```

그밖의 슬라이스

```
let a = [1, 2, 3, 4, 5];

let slice = &a[1..3];
```



Q & A