

CS152 – Programming Language Paradigms
Prof. Tom Austin

Closures & Scoping



Variables

- Parameters
- Local variables
- Free variables
 - Variables not defined in the current scope
 - e.g. global variables

```
#!/bin/bash
```

Is x 42?

```
x=42
```

```
function foo {
```

```
    echo $x
```

x is a *free*
variable

```
}
```

```
function bar {
```

```
    local x=666
```

Or is x
666?

```
    foo
```

```
}
```

```
bar
```

Lab part 1

- Guess what the bash script should print
- Run the script
- Rewrite the script into a Java program as faithfully as you can. What does it return?

```
#!/bin/bash
```

```
x=42
```

```
function foo {
```

```
    echo $x
```

```
}
```

```
function bar {
```

```
    local x=666
```

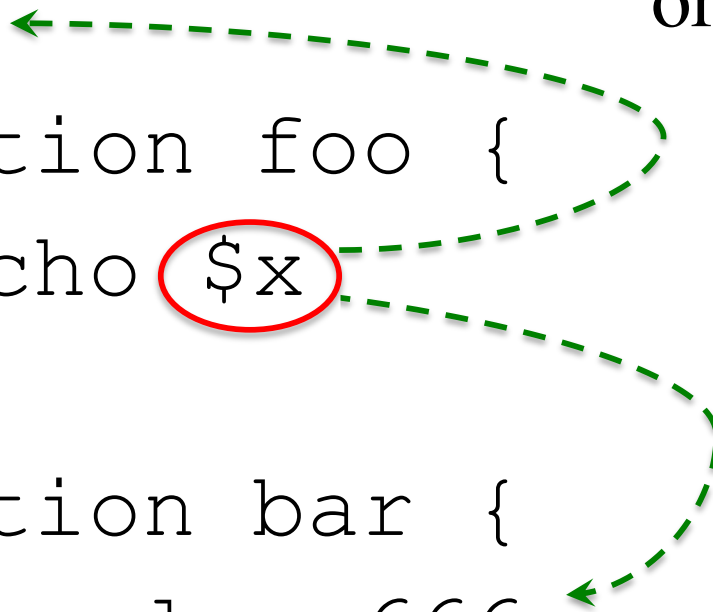
```
    foo
```

```
}
```

```
bar
```

Most languages uses *static*
or *lexical* scoping, so `x`
would be 42.

But Bash uses dynamic
scoping, so `x` is 666?



Scoping definitions

- In static or lexical scoping, name resolution depends on where the named variable is defined.
- In dynamic scoping, name resolution depends on the execution path of the code (the calling context).

Why do some languages use
dynamic scoping?

Closures and Environments

- A closure is a pair of
 - a function, and
 - its environment
- An environment is a mapping of free variables to their values defined outside the function.

Simple example of closures

```
(define (make-adder x)
  (lambda (y) (+ x y)))
```

```
(let ([add-two (make-adder 2)])
  (add-two 3))
```

```
(define (make-counter)
  (let ([count 0])
    (lambda ()
      (set! count (+ count 1))
      count)))

(define my-count (make-counter))

(my-count)

(my-count)

(define ctr2 (make-counter))

(ctr2)

(my-count)
```

```
(define (box x)
  (cons
    (λ () x)
    (λ (y) (set! x y))))
(define (get-val bx)
  ((car bx)))
(define (set-val! bx new-val)
  ((cdr bx) new-val))
```

Using box

```
(let ([my-box (box 3)])  
  (displayln (get-val my-box))  
  (set-val! my-box 4)  
  (displayln (get-val my-box)))
```

Lab, part 2

- Use `box` to create an `Employee` object.
- Details in Canvas.