Go Slice with Cap

Brian Kidd, Byron Samaripa, Nir Boneh

December 11, 2014

Overview

- Google Go
- Difference between Arrays and Slices
- Slice Syntax Proposal
- Impact and Uses

Google Go

- Introduced in 2009 BY GOOGLE
- Created by Rob Pike
- Statically-typed, Garbage Collected, Compiled
- Goal to Create a Better C

Arrays in Go

- Arrays are Primitive Values

```
b := [2]int{42, 47}
```

```
b := [...]string{"fourtytwo", "fourtyseven"}
```

Introduction to Slices

- First introduced in Fortran 57
- A way to interact with partitions of arrays
- Less common these days

Slices

- Common Use: Extracting a Substring from a String
- e.g. “ell” from “hello”

Slices in Go

Brian Kidd, Byron Samaripa, Nir Boneh

- Slice Syntax Declaration

```
letters := []string{"a", "b", "c", "d"}
```

```
func make([]T, len, cap) []T
```

Slices in Go

Brian Kidd, Byron Samaripa, Nir Boneh

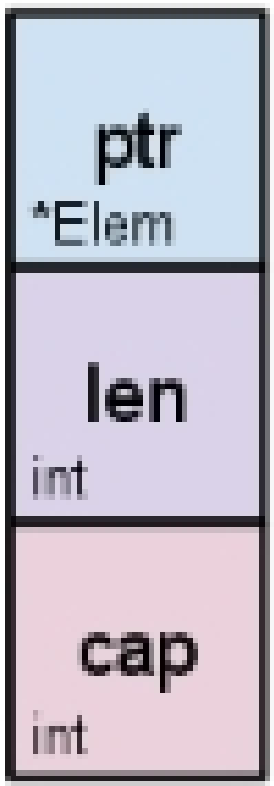

Figure: Slice Structure

Slices in Go

Brian Kidd, Byron Samaripa, Nir Boneh

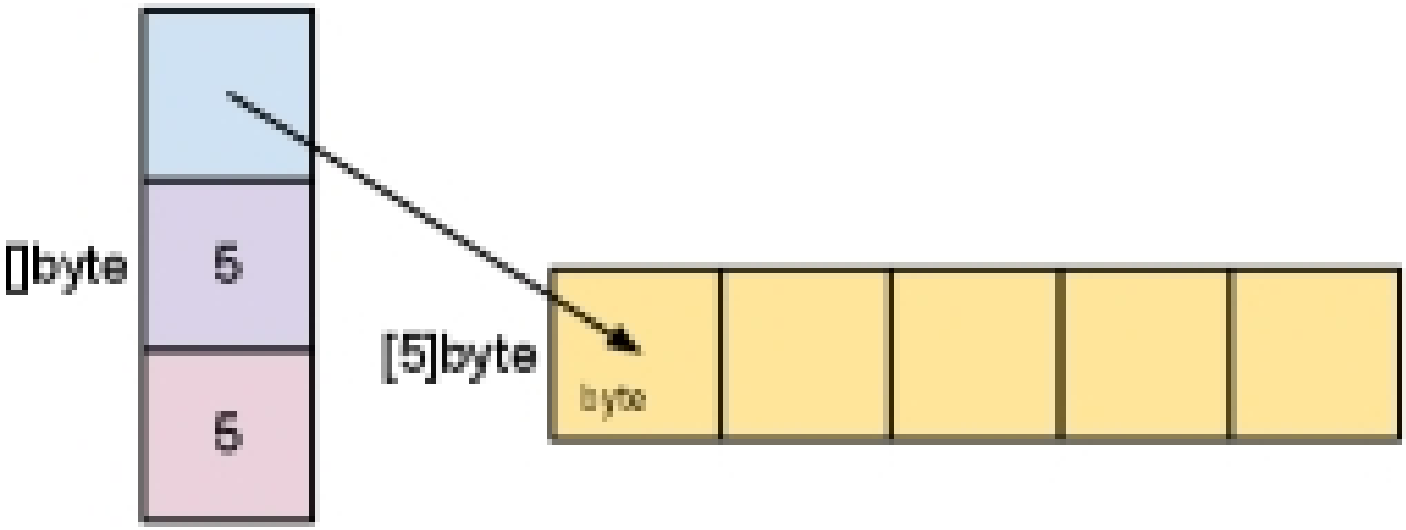

Figure: Slice pointing to array

Slices in Go

- To Construct a Slice from an Existing Slice
- a[i:j]
- The index of the Slice is i
- The length is j - i

Slices in Go

Brian Kidd, Byron Samaripa, Nir Boneh

```
b := []byte{'g', 'o', 'l', 'a', 'n', 'g'}

// b[:2] == []byte{'g', 'o'}
// b[2:] == []byte{'l', 'a', 'n', 'g'}
// b[:] == b
```

Slices in Go


```
s = s[2:4]
```

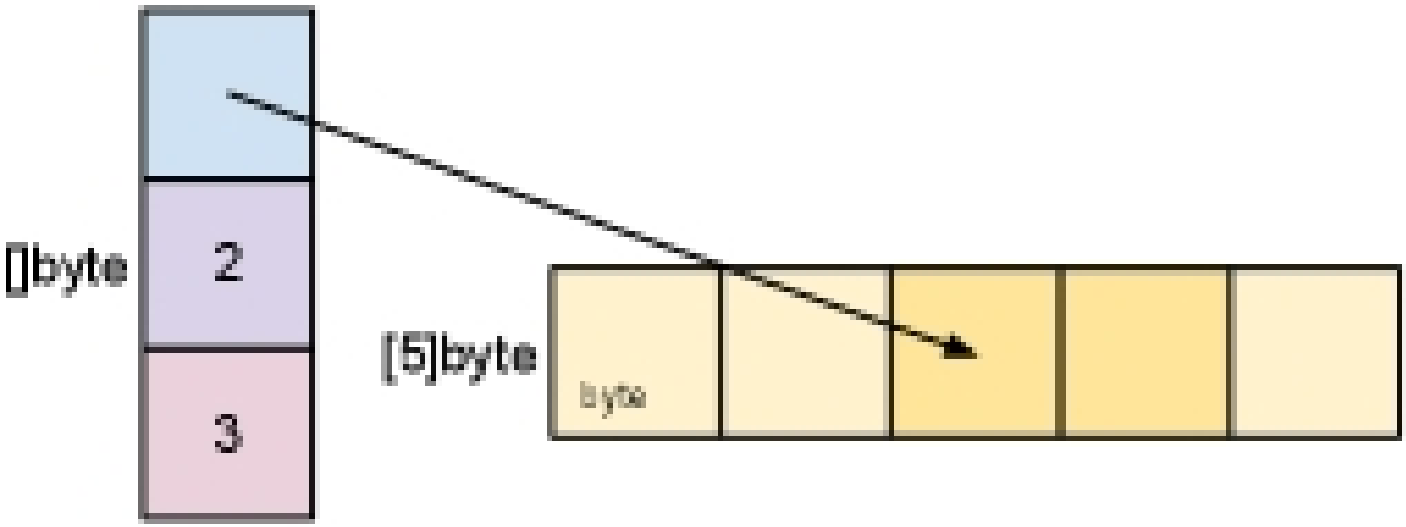

Figure: Sub-slice

Capacity of Slices


```
s = s[:cap(s)]
```

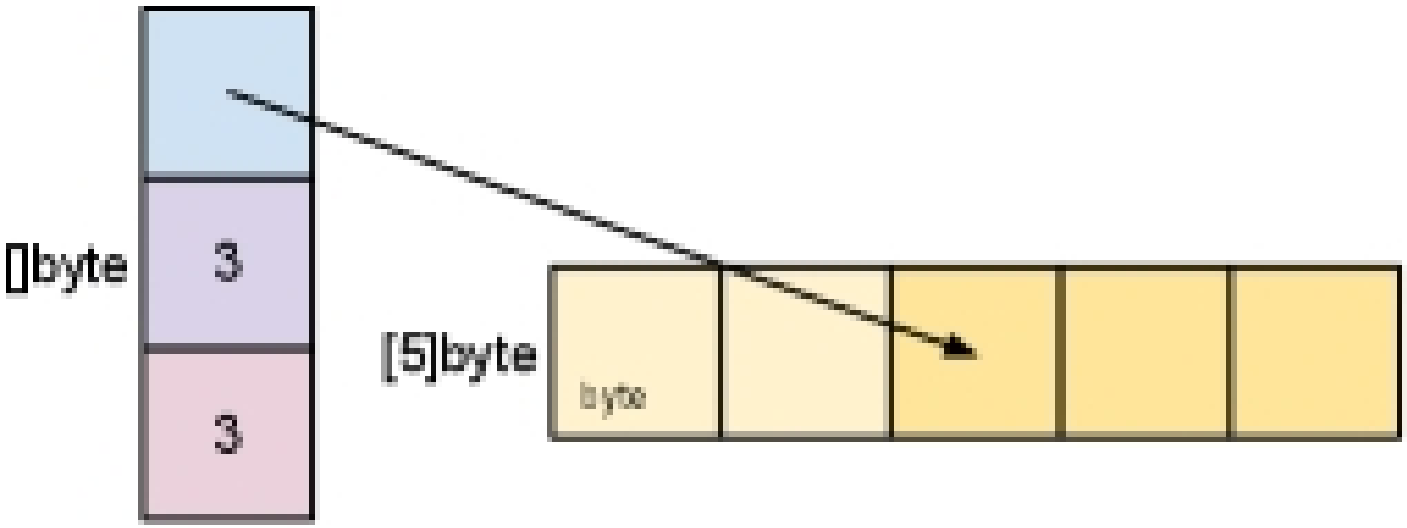

Figure: Length Expansion

Proposal

- Proposed by Russ Cox
- a[i : j : k]
- The index of the Slice is i
- The length is j - i
- The capacity is k - i

Proposal Explained

- Allows Construction of a new Slice from an Existing Slice with a Lower Capacity

```
x = make([]int, 10, 20)
// x[0:10:15] == make([]int, 10, 15)
```

Implications

- Gives the Programmer more Control over Slices
- Language does not feel complete without it

Syntax Concerns

- Confusing Syntax

"The syntax s[a:b:c] seems a bit confusing to me. [a:b] can easily be translated to"from a to b" while the semantics of [a:b:c] wouldn't be that obvious. I think even more problematic is [a::b]. For one thing it could easily be missed, like gri pointed out, for another thing :: has a special meaning in many other languages." - Julien Schmidt

Syntax Addressed

- There is no real solution for the confusing syntax
- Although it is not necessary to include the third element of the slice
- In other words, backwords compatible.

Alias Concerns

- Lose the ability to compare two slices and see if they share memory
- Previously slices would compare the last element of two slices

"People keep talking about"alias" but I'm not convinced that's the concept we're losing. What we're losing is the ability to tell if two slices reside in the same underlying array. The ability to change the capacity of a slice makes aliasing a dynamic property, but sharing an array remains a static one, and perhaps the more important and, after this change, harder to compute. I can always compute 'aliasing' by comparing addresses, but after a capacity change there is no unsafe-free way to discover if two slices could share memory." - Rob Pike

Alias Addressed

- This was addressed this on the mailing list
- The test was rarely used practice
- it is not as important as the feature
- Rob Pike addresses the issue by stating that in practice, this test is not widely used

"I don't see a compelling argument that overlap testing is important enough to warrant language support beyond package unsafe. If that argument is made, we could consider it then." -Rob Pike

Overview

- In the grand scheme of things, this change was not very important to language
- The entire reason as to why this was implemented in the first place was to satisfy a specific instance where you would need to change the capacity
- There is no reason to leave out functionality in a programming language