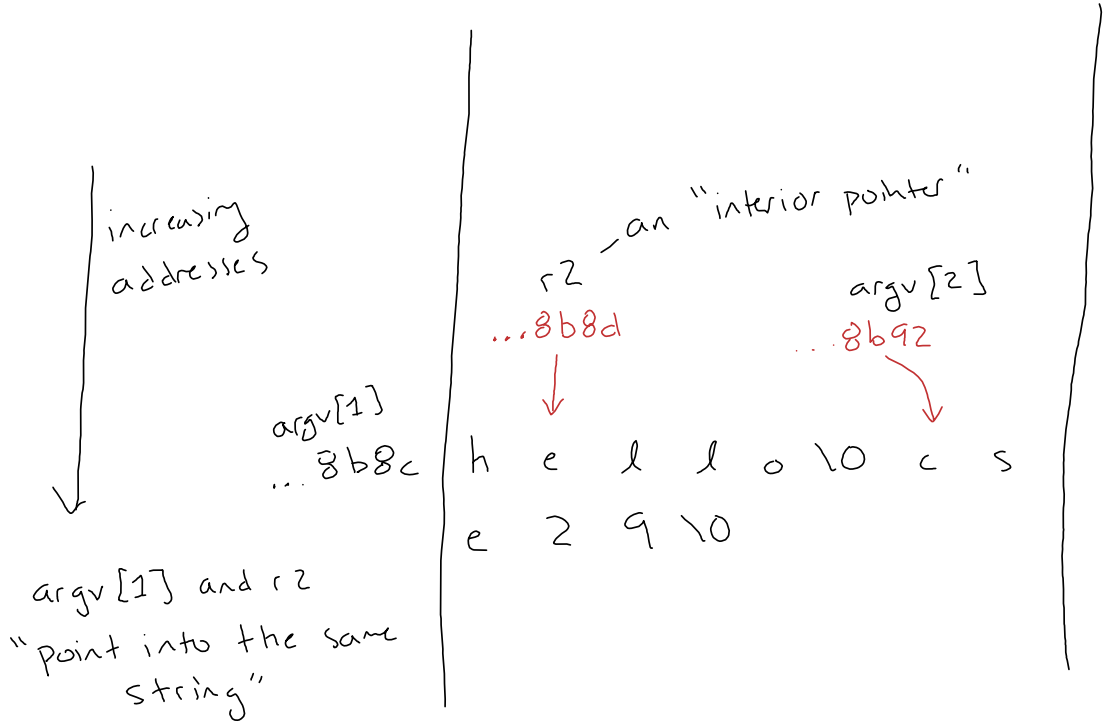


"slices" in Go
Rust

./strfunc hello csc29



./strfunc ¹²³⁴⁵⁶⁷⁸ joe-had-surgery surge

argv[1] 0x... 6b82
r1 0x... 6b8a ↗ 8 bytes later

`uint32_t hello32[] = {0x48`
~~~~~  
gets expanded  
to 32 bits

`0x00000048` is stored in memory

1 byte each



hello8 | 0x48 | 0x65 | . . . . .

hello32 | 0x00 | 0x00 | 0x00 | 0x48 | ...  
or is it  
0x48 | 0x00 | 0x00 | 0x00

← if this is how it is stored, why does it print?

Big-endian

`0x48000000`

$4 * 16^7 + 8 * 16^6 \dots$

Little-endian

`0x00000048`

$4 * 16 + 8 = 72$

"Endian-ness" how to interpret multi-byte numbers

Big-endian first byte in order is most significant

Little-endian first byte in order is least significant