Chapter 54: (Unsafe) Buffer Pointers

“A buffer pointer is used for low-level access to a region of memory. For example, you can use a buffer pointer for efficent processing and communication of data between apps and services.”

Excerpt From: Apple Inc. “Using Swift with Cocoa and Objective-C (Swift 3.1 Edition).” iBooks.
https://itun.es/us/utTW7.l

You are responsible for handling the life cycle of any memory you work with through buffer pointers, to avoid leaks or undefined behavior.

Section 54.1: UnsafeMutablePointer

```
struct UnsafeMutablePointer<Pointee>
```

> A pointer for accessing and manipulating data of a specific type.

You use instances of the UnsafeMutablePointer type to access data of a specific type in memory. The type of data that a pointer can access is the pointer's Pointee type. UnsafeMutablePointer provides no automated memory management or alignment guarantees. You are responsible for handling the life cycle of any memory you work with through unsafe pointers to avoid leaks or undefined behavior.

Memory that you manually manage can be either untyped or bound to a specific type. You use the UnsafeMutablePointer type to access and manage memory that has been bound to a specific type. (Source)

```
import Foundation

let arr = [1,5,7,8]

let pointer = UnsafeMutablePointer<[Int]>.allocate(capacity: 4)
pointer.initialize(to: arr)

let x = pointer.pointee[3]

print(x)

pointer.deinitialize()
pointer.deallocate(capacity: 4)

class A {
  var x: String?

  convenience init (_ x: String) {
    self.init()
    self.x = x
  }

  func description() -> String {
    return x ?? ""
  }
}


let arr2 = [A("OK"), A("OK 2")]
let pointer2 = UnsafeMutablePointer<[A]>.allocate(capacity: 2)
```

```
pointer2.initialize(to: arr2)

pointer2.pointee
let y = pointer2.pointee[1]

print(y)

pointer2.deinitialize()
pointer2.deallocate(capacity: 2)
```

Converted to Swift 3.0 from original source

Section 54.2: Practical Use-Case for Buffer Pointers

Deconstructing the use of an unsafe pointer in the Swift library method;

```
public init?(validatingUTF8 cString: UnsafePointer<CChar>)
```

Purpose:

Creates a new string by copying and validating the null-terminated UTF-8 data referenced by the given pointer.

This initializer does not try to repair ill-formed UTF-8 code unit sequences. If any are found, the result of the initializer is `nil`. The following example calls this initializer with pointers to the contents of two different `CChar` arrays---the first with well-formed UTF-8 code unit sequences and the second with an ill-formed sequence at the end.

> **Source**, **Apple Inc., Swift 3 header file** (For header access: In Playground, Cmd+Click on the word Swift) in the line of code:

> `import Swift`

```
let validUTF8: [CChar] = [67, 97, 102, -61, -87, 0]
     validUTF8.withUnsafeBufferPointer { ptr in
         let s = String(validatingUTF8: ptr.baseAddress!)
         print(s as Any)
     }
     // Prints "Optional(Café)"

     let invalidUTF8: [CChar] = [67, 97, 102, -61, 0]
     invalidUTF8.withUnsafeBufferPointer { ptr in
      let s = String(validatingUTF8: ptr.baseAddress!)
      print(s as Any)
     }
// Prints "nil"
```

> (Source, Apple Inc., Swift Header File Example)