Scripting in Swift

Francisco Díaz

franciscodiaz.cl - @fco_diaz

Hello World

Shell scripting

```
#!/usr/bin/env bash
echo "hello, world!"
```

```
#!/some/other/thing
echo "hello, world!"
```

```
echo "hello, world!"
```

Swift

```
#!/usr/bin/env swift
print("hello, world!")
```

STDOUT

Stream of data **produced** by a command line program to output data

STDERR

Strea of data **produced** by a command line program to output error messages

STDIN

Stream from which a command line program **reads** data

STDOUT

```
#!/usr/bin/env swift
print("hello, world!")
```

```
func print<Target>(
  _ items: Any...,
  separator: String = " ",
  terminator: String = "\n",
  to output: inout Target)
    where Target : TextOutputStream
```

```
func print<Target>(
  _ items: Any...,
  separator: String = " ",
  terminator: String = "\n",
  to output: inout Target)
    where Target : TextOutputStream
```

Common use cases

- Display to the user
- Used by another program to do further processing

STDERR

```
#!/usr/bin/env swift

import Darwin

struct StderrOutputStream: TextOutputStream {
  mutating func write(_ string: String) { fputs(string, stderr) }
}

var standardError = StderrOutputStream()

print("Some error", to: &standardError)

exit(1)
```

Common use case

- Show an error to the user

STDIN

```
#!/usr/bin/env swift

import Foundation

let arguments = CommandLine.arguments

print("Arguments: \(arguments)")
```

Common use case

- Take information from the user

Example:

Count the number of lines of Swift in a folder

ls

```
→ ls *.swift

DefaultNetworkConditioner.swift DefaultPathMonitor.swift DelayedRequestHandler.swift
```

Count the number of lines of Swift in a folder

wc

```
→ wc -l DefaultNetworkConditioner.swift

137 DefaultNetworkConditioner.swift
```

Count the number of lines of Swift in a folder

xargs

It converts input from standard input into arguments to a command

Count the number of lines of Swift in a folder

```
ls *.swift | xargs wc -l

137 DefaultNetworkConditioner.swift
136 DefaultPathMonitor.swift
 44 DelayedRequestHandler.swift
317 total
```

`wc` uses the output of `ls` as input

Software Tools Principles [1]

Don't be chatty

No starting processing, almost done, or finished processing kind of messages should be mixed in with the regular output of a program (or at least, not by default).

[1] Robbins, Arnold, and Nelson H. F. Beebe. Classic Shell Scripting. O'Reilly, 2005.

```
wc -l shell/example.swift
```

Problem?

Let's not count non-swift lines

```
swift package init --type executable
swift package generate-xcodeproj
```

SwiftCount
SwiftCountCore

SwiftCount

- Parses input from user (Handles STDIN)
- Outputs to user (Handles STDOUT / STDERR)

SwiftCountCore

- Does the business logic of the tool

```
let package = Package(
  name: "SwiftCount",
  products: [
    .executable(name: "SwiftCount", targets: ["SwiftCount"]),
    .library(name: "SwiftCountCore", targets: ["SwiftCountCore"]),
  ],
  targets: [
    .target(
      name: "SwiftCount",
      dependencies: ["SwiftCountCore"]),
    .target(
      name: "SwiftCountCore"),
    .testTarget(
      name: "SwiftCountCoreTests",
      dependencies: ["SwiftCountCore"]),
  ]
)
```

SwiftSyntax

It allows for Swift tools to parse, inspect, generate, and transform Swift source code.

```
import PackageDescription

let package = Package(
  name: "SwiftCount",
  products: [
    .executable(name: "swiftcount", targets: ["SwiftCount"]),
  ],
  dependencies: [
    .package(name: "SwiftSyntax",
             url: "https://github.com/apple/swift-syntax.git", .exact("0.50200.0")),
  ],
  targets: [
    .target(
      name: "SwiftCount",
      dependencies: ["SwiftCountCore"]),
    .target(
      name: "SwiftCountCore",
      dependencies: ["SwiftSyntax"]),
    .testTarget(
      name: "SwiftCountCoreTests",
      dependencies: ["SwiftCountCore"]),
  ]
)
```

[Trivia] | [Token] | [Trivia]

```
// Some comment
struct Some {
}
```

Token 1

Leading Trivia:

1) newline
2) comment
3) newline

Token Kind:

1) struct

Trailing Trivia:

1) space

main.swift

```
var arguments = CommandLine.arguments

guard arguments.count > 1 else {
  var standardError = StderrOutputStream()

  print("swiftcount Error: Need to pass at least one file path", to: &standardError)

  exit(1)
}

let filePaths: [String] = Array(arguments.dropFirst())

let swiftReader = SwiftFileReader(filePaths: filePaths)
let lineCount = swiftReader.run()

lineCount.forEach {
  print("\($0.numberOfLines) \($0.relativePath)")
}
```

SwiftFileReader

```swift
public struct SwiftFileReader {

  /// - Parameter filePaths: A list of relative paths to Swift files
  public init(filePaths: [String]) {
    self.filePaths = filePaths
  }

  private let filePaths: [String]

  public func run() -> [LineCount] {
    filePaths.map { LineCount(numberOfLines: 1, relativePath: $0) }
  }

}

public struct LineCount: Equatable {
  public let numberOfLines: Int
  public let relativePath: String
}
```

Demo

Binary

```
swift build -c release
```

Possible improvements

- Ignore classes conforming to XCTestCase
- Ignore whitespace only lines

a.k.a. this is not production ready

github.com/fdiaz/ioslove-2020

github.com/fdiaz/swiftinspector