Introduction to Swift

These extra chapters are intended to teach you something extra beyond the core material that you may otherwise not come into contact with. One of those things is Swift, Apple's hot new programming language intended to replace Objective-C. In fact, Swift can now also be used on other platforms besides Apple's OS X and iOS. Since December 2015 Apple has made Swift 3.0 available for Linux. There are still limits. To create apps for Apple products there will need to be a development framework such as Xcode. Sadly Apple has not released that. At this moment we can only wait for an open source port. However, you can create applications such as games, text editors and webservers on Linux. So let's give swift a try on Ubuntu!

Installing Swift

The first step is to download the Swift Development Snapshot from swift.org/download/, choose the latest Ubuntu version. Next unpack the file you downloaded. Put the file in your workspace. This will then become the location of your Swift installation. You may also optionally choose to shorten the folder name. I have chosen

```
Swift_3.0
```

Next we need to install some dependencies for Swift in order to develop code.

```
sudo apt-get install clang libicu-dev
```

Clang is a compiler for C, C++ and Objective-C. A lot of low level Swift is actually still Objective-C. Next we need to adjust the environment variables. First check your current PATH settings with

```
$PATH
```

Now let’s change this path with the command

```
export PATH=/home/username/Swift_3.0/usr/bin:$PATH
```

Note the location of the directory where you placed you’re swift repo. Of course you should also change *username*. If you use $PATH again you will notice it has been changed. Now you are set to Swift! With

```
swift –version
```

we confirm everything is properly installed.

```
Swift version 3.0-dev (LLVM b361b0fc05, Clang 11493b0f62, Swift 24a0c3de75)

Target: x86_64-unknown-linux-gnu
```

Testing REPL

Swift comes with its own interactive shell like Python, it’s called REPL. By just typing *swift* into Terminal you gain access to it. In REPL you can quickly run some code when you need to. Let’s do a test. Type the following into REPL.

```
1>   var welcome = "Hello, world!"
```

REPL tells us the String value is assigned to welcome.

```
2>   print(welcome)
```

With the print command we simply print out the string to the REPL interface. Note how we assigned the value to a var. We could also have used let, but then we could not change the value afterwards. Let is a constant, it is immutable. That’s enough for REPL. Let’s write a proper test application.

Creating an application

For our Swift test application we will set up a proper folder.

```
mkdir HelloSwift
```

Inside this directory we create a file called Package.swift which every Swift application must have.

```
touch Package.swift
```

Next we create a *Sources* folder

```
mkdir Sources
```

Every Swift application must have a file called main.swift inside Sources. This will start the application.

```
touch Sources/main.swift
```

The only thing left to do is alter main.swift with your favorite text editor. Add the following lines to read input from Terminal.

```
let response = readLine(stripNewline: true)
print(response)
```

Now we only need to compile the code with the command

```
swift build
```

Now we can run the program. Inside the HelloSwift folder run the following command

```
.build/debug/HelloSwift
```

With the readLine function the program will stop until the user has pressed Enter. If you input Hello, World! and press Enter you will see the string printed to the console. readLine is from the Swift Standard Library and as such does not require any specific import. The list of standard functions are those you would expect such as swap, sizeof, min, max, assert and of course print.

Below is a more lengthy coding examples. If you can fully understand it than you both know the basics of programming and of Swift. But first copy paste the code, then compile and run the code to follow the tutorial.

```
import Foundation

print("Hello to this basic Swift script")
print("Would you like to proceed with a tutorial on Swift? Yes or No?")

let response = readLine(stripNewline: true)
var runTutorial: Bool
repeat {
        if response == "Yes" || response == "yes" || response == "Y" || response == "y" {
                print("Conditional statements are similar to Java")
                print("You can use the || and && to chain conditions to together")
                runTutorial = false
        } else if response == "No" || response == "no" || response == "N" || response == "n" {
                runTutorial = false
                print("This tutorial will now end")
                exit(0)
        } else {
                runTutorial = true
                print("Incorrect input! Choose [Y]es or [N]o")
                print(response)
        }
} while runTutorial
print("For loops are simple. They follow the initialization; condition and increment structure")
print(“Note: variables are incremented with double plus sign on the left”)
for var i = 1; i <= 10; ++i {
   print(“I love Swift”)
}

print("Classes are also similar to Java. You initialize a class object calling the init function")
let s = Rectangle(base:5, height:6)
print("And you can use class functionality using the object.function() call format.")
print("Example: the area of the rectangle is \(s.area()).")
```

As Swift focuses heavily on Object-Oriented programming there is also a small class defines that calculates the area of a rectangle. Again, if you’re familiar with Java, or Python all this will look familiar.

```
/*
 * A simple Swift class for Linux!
 */
class Rectangle {
   var base:Int
   var height:Int

   init(base:Int, height:Int) {
      self.base = base
      self.height = height
   }

   func area() -> Int {
      return self.base * self.height
```

```
        }
    }
```

Essentially Rectangle is a user defined type, like Int and Bool. It has two variables, an initializer used to first set it up and user defined functionality. This one function, area(), calculates the area of the rectangle.

Swift in Linux is still under development, the Foundation package still needs a lot of work, but it is getting there. Swift is an exciting new programming language. If you want to develop a new skill that is bound to be relevant you can do worse than picking Swift. Enjoy!

Source; http://itsfoss.com/use-swift-linux/