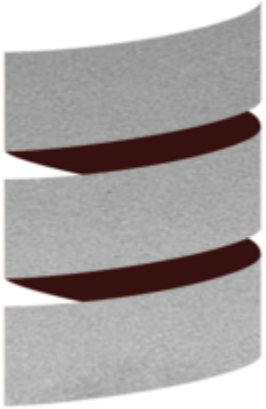
Introduction to Scala Native

@ Haxogreen 2016

Dudelange, Luxembourg

Stefan Ollinger



What?

- Scala
  - Language on the JVM
- Scala Native
  - Compiles Scala source code to LLVM intermediate representation
- LLVM: collection of compiler technologies
  - .ll intermediate representation
  - llc: compiler for .ll to machine code
  - Clang: LLVM compiler for C/C++
  - Many others

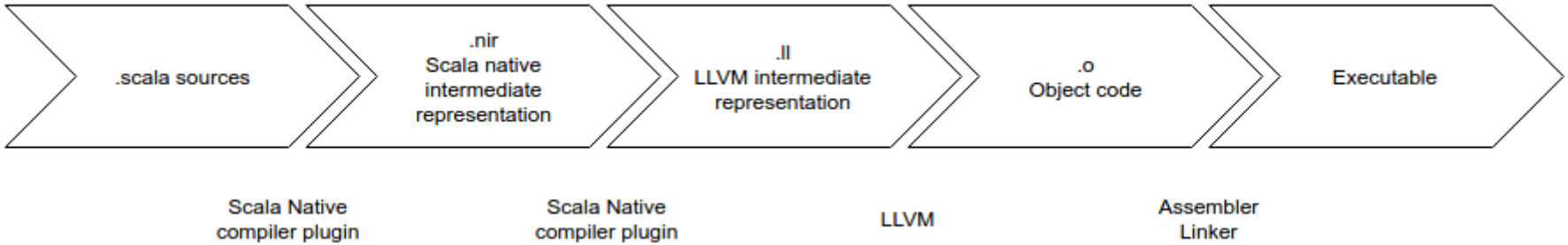
Current state

- Project at EPFL in Switzerland
- Pre-release stage in May 2016
- Partial implementation of JDK
  - java.util.*
  - java.lang.*
  - and a few others

Current state

- Scala Native is a collection of modules
- Libraries to write and link programs against
  - **nativelib**: Native data types and bindings
  - **scalalib**: Implementation of scala std library
  - **javalib**: (Partial) implementation of JDK
- Compiler plugins and tools
  - **nir**: Native intermediate representation
  - **nscplugin**: Compiler plugin for scalac
  - **sbtplugin**: Integration into SBT
  - **tools**: Tools

Compilation phases



A Scala Native “hello world”

```scala
package hello

import scala.scalanative._
import native._, stdio._, stdlib._

object Main {

  def main(args: Array[String]): Unit = {

    fprintf(stderr, c"Hello, World!\n")

  }
}
```

- Object with main method
- Bindings to stdio, stdlib libs
  - e.g. `fprintf`, `stderr`
- C strings with string interpolator
  - e.g. `c”Hello, World\n”`

Demo

```
root@b4f276eef805: /home/sources/1-hello
 /usr/bin/../lib/gcc/x86_64-linux-gnu/5.4.0/../../../../include/x86_64-linux-gnu/c++/5.4.0
 /usr/bin/../lib/gcc/x86_64-linux-gnu/5.4.0/../../../../include/c++/5.4.0/backward
 /usr/local/include
 /usr/lib/llvm-3.7/bin/../lib/clang/3.7.1/include
 /usr/include/x86_64-linux-gnu
 /usr/include
End of search list.
 "/usr/bin/ld" --eh-frame-hdr -m elf_x86_64 -dynamic-linker /lib64/ld-linux-x86-64.so.2 -o
 /home/sources/1-hello/target/scala-2.11/samplehello-out /usr/bin/../lib/gcc/x86_64-linux-
gnu/5.4.0/../../../x86_64-linux-gnu/crt1.o /usr/bin/../lib/gcc/x86_64-linux-gnu/5.4.0/../.
./../x86_64-linux-gnu/crti.o /usr/bin/../lib/gcc/x86_64-linux-gnu/5.4.0/crtbegin.o -L/usr/
bin/../lib/gcc/x86_64-linux-gnu/5.4.0 -L/usr/bin/../lib/gcc/x86_64-linux-gnu/5.4.0/../../.
./x86_64-linux-gnu -L/lib/x86_64-linux-gnu -L/lib/../lib64 -L/usr/lib/x86_64-linux-gnu -L/
usr/bin/../lib/gcc/x86_64-linux-gnu/5.4.0/../../.. -L/usr/lib/llvm-3.7/bin/../lib -L/lib -
L/usr/lib -lgc /tmp/samplehello-out-757a4b.o /tmp/rt-afbaa7.o -lstdc++ -lm -lgcc_s -lgcc -
lc -lgcc_s -lgcc /usr/bin/../lib/gcc/x86_64-linux-gnu/5.4.0/crtend.o /usr/bin/../lib/gcc/x
86_64-linux-gnu/5.4.0/../../../x86_64-linux-gnu/crtn.o
[info] Running /home/sources/1-hello/target/scala-2.11/samplehello-out
Hello, World!
[success] Total time: 13 s, completed Jul 30, 2016 2:20:22 PM
>
```

Demo

root@b4f276eef805: /home/sources

```
root@b4f276eef805:/home/sources# readelf -h 1-hello/target/scala-2.11/samplehello-out
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x400c60
  Start of program headers:          64 (bytes into file)
  Start of section headers:          25344 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         8
  Size of section headers:           64 (bytes)
  Number of section headers:         39
  Section header string table index: 36
root@b4f276eef805:/home/sources#
```

Low-level primitives

- CStrings

```
val hello: CString = c"hello haxogreen"
fprintf(stdout, c"hello 5th %s\n", hello.+(6))
```

- Allocate memory

```
val ints: Ptr[CInt] = malloc(sizeof[CInt] * 3).cast[Ptr[CInt]]

ints.update(0, 100)
ints.update(1, 200)
ints.update(2, 300)

fprintf(stdout, c"%d\n", ints(0))
fprintf(stdout, c"%d\n", ints(1))
fprintf(stdout, c"%d\n", ints(2))
```

Low-level primitives

Structs

```
@struct
class Pi(n: Int = 0) {
  @inline def `++`: Pi = {
    new Pi(n+1)
  }

  // https://en.wikipedia.org/wiki/Leibniz_formula_for_%CF%80
  @inline def value: Double = {
    4 * (0 to n).foldLeft(0.0) { case (acc, i) =>
      val sig = if (i % 2 == 0) 1.0 else -1.0
      acc + sig / (i*2 + 1)
    }
  }
}
```

Allocate on stack

```
val π = stackalloc[Pi]

!π = new Pi(42)

def showPi(π: Ptr[Pi]): Unit = {
  fprintf(stdout, c"%f\n", (!π ++).value)
}

showPi(π)
```

Let’s download a file

- Using scala.io.Source.fromURL
  - should be simple

```
val content = io.Source.fromURL("https://www.google.de", "ISO-8859-1").mkString
println(content)
```

- Compiling, oops...

```
> run
Unresolved dependencies:
  `@java.io.PushbackReader::init_class.java.io.Reader`
  `@java.io.PushbackReader::unread_i32_unit`
  `@java.io.PushbackReader`
  `@java.net.URL::init_class.java.lang.String`
  `@java.net.URL::openStream_class.java.io.InputStream`
  `@java.net.URL`
```

Let’s download a file

- The **javalib** module is a partial implementation of JDK
  - e.g. java.net.URL is missing
  - Work-in-progress
  - Contribute!

scala-native / **scala-native**

Watch 140 | Star 1,924 | Fork 77

Code | Issues 76 | Pull requests 4 | Pulse | Graphs

Implement java.util.Random #250

Merged **densh** merged 1 commit into `scala-native:master` from `andreaTP:jurandom` 9 days ago

Conversation 5 | Commits 1 | Files changed 6

Showing changes from **all commits** | **6 changed files** | +379 −16 | Options

131 javalib/src/main/scala/java/util/Random.scala | View

```
@@ -1,18 +1,121 @@
 package java.util

-class Random {
-  def this(seed: Long) = this()
-
-  def next(bits: scala.Int): scala.Int     = ???
-  def nextBoolean(): scala.Boolean         = ???
-  def nextBytes(bytes: Array[Byte]): Unit  = ???
-  def nextDouble(): scala.Double           = ???

-  def nextFloat(): scala.Float             = ???
-  def nextGaussian(): scala.Double         = ???
-  def nextInt(): scala.Int                 = ???
-  def nextInt(bound: scala.Int): scala.Int = ???
-  def nextLong(): scala.Long               = ???
-  def setSeed(seed: scala.Long): Unit      = ???
```

```
 package java.util

+import scala.annotation.tailrec
+
+import java.lang.Math
+
+import scala.scalanative.native.stdlib
+
+class Random(seed_in: Long) extends AnyRef with
java.io.Serializable {
+
+  private var seed: Long = _
+
+  // see nextGaussian()
+  private var nextNextGaussian: Double      = _
+  private var haveNextNextGaussian: Boolean = false
+
+  setSeed(seed_in)
```

- But we can just use an existing system library, e.g. **libcurl**

External bindings

```
enablePlugins(ScalaNativePlugin)

name := "hello"
description := "hello scala.native"

scalaVersion := "2.11.8"

nativeVerbose := true

nativeClangOptions := Seq(
  "-O2",
  "-lcurl",
  "-g",
  "-v"
)
```

- libcurl-dev installed
- Link
- “-lcurl” in build.sbt
  - nativeClangOptions
- Define bindings
- Invoke bindings

External bindings

- Define bindings to easy.h

```scala
package hello

import scala.scalanative._, native._, stdio._, stdlib._

@extern
object curl {

  type CURLcode = CInt
  type CURLoption = CInt

  def curl_easy_init(): Ptr[_] = extern
  def curl_easy_cleanup(curl: Ptr[_]): Unit = extern
  def curl_easy_setopt(curl: Ptr[_], option: CURLoption, value: Ptr[_]): CURLcode = extern
  def curl_easy_perform(curl: Ptr[_]): CURLcode = extern

}

object CURLOpts {

  final val CURLOPT_URL: CInt                 = 10002
  final val CURLOPT_WRITEFUNCTION: CInt       = 20011
  final val CURLOPT_WRITEDATA: CInt           = 10001

}
```

External bindings

- Invoke the bindings

```
def downloadFile(url: CString, outfilename: CString, outfilename1: String): Unit = {

  val c = curl.curl_easy_init()

  val fp: Ptr[FILE] = fopen(outfilename, c"wb")

  curl.curl_easy_setopt(c, CURLOpts.CURLOPT_URL, url)
  curl.curl_easy_setopt(c, CURLOpts.CURLOPT_WRITEDATA, fp)

  val res = curl.curl_easy_perform(c)

  fclose(fp)

  curl.curl_easy_cleanup(c)

}
```

- Ok, now we can write the file to the disk!

Using a Scala library

- Let’s choose an example
  - Managing an imaginary device
  - Turn device power on/off
  - Set GPIO on/off

- Device is represented as a data structure
- Operations are state modifications

Device is represented as a data structure

```scala
case class HW(
  vendor: String,
  powerState: PowerState,
  gpio: GPIO
)

sealed trait PowerState
object PowerState {
  case object Powered extends PowerState
  case object Unpowered extends PowerState
}

sealed trait GPIOState
object GPIOState {
  case object On extends GPIOState
  case object Off extends GPIOState
}

case class GPIO(
  pin1: GPIOState,
  pin2: GPIOState,
  pin3: GPIOState,
  pin4: GPIOState
)
```

Types:

- `HW`: Hardware device
- `GPIO`: State of IO module
- `PowerState`: (un-)powered
- `GPIOState`: on/off

Example value:

```scala
val hw = HW(
  vendor = "Raspberry Pi Foundation",
  powerState = PowerState.Unpowered,
  gpio = GPIO(
    pin1 = GPIOState.Off,
    pin2 = GPIOState.Off,
    pin3 = GPIOState.Off,
    pin4 = GPIOState.Off
  )
)
```

Operations are state modifications

- A state modification
  - Is a function, which takes a `HW`
  - And returns the possibly modified `HW`
  - With a return value `A`, which often is just the empty value `()`

```
val hw = HW(
  vendor = "Raspberry Pi Foundation",
  powerState = PowerState.Unpowered,
  gpio = GPIO(
    pin1 = GPIOState.Off,
    pin2 = GPIOState.Off,
    pin3 = GPIOState.Off,
    pin4 = GPIOState.Off
  )
)
```

→ (

```
val hw2 = HW(
  vendor = "Raspberry Pi Foundation",
  powerState = PowerState.Powered,
  gpio = GPIO(
    pin1 = GPIOState.Off,
    pin2 = GPIOState.On,
    pin3 = GPIOState.Off,
    pin4 = GPIOState.On
  )
)
```

,())

Operations are state modifications

- A state modification
  - Can be represented as value/function of type: `HW => (HW, A)`
  - The `A` turns out to be useful, e.g. if the state is only read and a value is computed from the state

```
val hw = HW(
  vendor = "Raspberry Pi Foundation",
  powerState = PowerState.Unpowered,
  gpio = GPIO(
    pin1 = GPIOState.Off,
    pin2 = GPIOState.Off,
    pin3 = GPIOState.Off,
    pin4 = GPIOState.Off
  )
)
```

(

```
val hw2 = HW(
  vendor = "Raspberry Pi Foundation",
  powerState = PowerState.Powered,
  gpio = GPIO(
    pin1 = GPIOState.Off,
    pin2 = GPIOState.On,
    pin3 = GPIOState.Off,
    pin4 = GPIOState.On
  )
)
```

,())

Using a Scala library

- A function `S => (S, A)` can be encoded in the type `State[S, A]`
- Let's define a few basic state modification operations
- Modify existing state, by creating a modified copy

```scala
val switchOn: State[HW, Unit] =
  State.modify[HW](hw => hw.copy(powerState = PowerState.Powered))
```

- Read a value from state, and return it

```scala
def getGPIO1: State[HW, GPIOState] = State.gets(_.gpio.pin1)
```

- Dont do anything, return the unit value (), nop

```scala
def delay(ms: Int): State[HW, Unit] = State.state(())
```

Using a Scala library

- Compose basic modifications to a larger program

```scala
val togglePins42: State[HW, (GPIOState, GPIOState)] = for {
  _ <- setGPIO(2, GPIOState.On)
  _ <- setGPIO(4, GPIOState.On)
  _ <- delay(1000)
  pin2 <- getGPIO2
  pin4 <- getGPIO4
} yield (pin2, pin4)
```

- “Did you try turning it off and on again?”

```scala
val prog1: State[HW, (GPIOState, GPIOState)] = for {
  _ <- switchOn
  x <- togglePins42
} yield x
```

- Run the program
  - Note that `prog1.run` is of type `HW => (HW, (GPIOState, GPIOState))`

```scala
val (hw2, (pin2, pin4)) = prog1.run(hw)

println(pin2)            // GPIOState.On
println(pin4)            // GPIOState.On
println(hw.powerState)   // PowerState.Unpowered
println(hw2.powerState)  // PowerState.Powered
```

Demo

```
root@b4f276eef805: /home/sources/3-library
 /usr/lib/llvm-3.7/bin/../lib/clang/3.7.1/include
 /usr/include/x86_64-linux-gnu
 /usr/include
End of search list.
 "/usr/bin/ld" --eh-frame-hdr -m elf_x86_64 -dynamic-linker /lib64/ld-linux-x86-64.so.2 -o
 /home/sources/3-library/target/scala-2.11/samplelibrary-out /usr/bin/../lib/gcc/x86_64-li
nux-gnu/5.4.0/../../../x86_64-linux-gnu/crt1.o /usr/bin/../lib/gcc/x86_64-linux-gnu/5.4.0/
../../../x86_64-linux-gnu/crti.o /usr/bin/../lib/gcc/x86_64-linux-gnu/5.4.0/crtbegin.o -L/
usr/bin/../lib/gcc/x86_64-linux-gnu/5.4.0 -L/usr/bin/../lib/gcc/x86_64-linux-gnu/5.4.0/../
../../x86_64-linux-gnu -L/lib/x86_64-linux-gnu -L/lib/../lib64 -L/usr/lib/x86_64-linux-gnu
 -L/usr/bin/../lib/gcc/x86_64-linux-gnu/5.4.0/../../.. -L/usr/lib/llvm-3.7/bin/../lib -L/l
ib -L/usr/lib -lgc /tmp/samplelibrary-out-5125ae.o /tmp/rt-8c8b39.o -lstdc++ -lm -lgcc_s -
lgcc -lc -lgcc_s -lgcc /usr/bin/../lib/gcc/x86_64-linux-gnu/5.4.0/crtend.o /usr/bin/../lib
/gcc/x86_64-linux-gnu/5.4.0/../../../x86_64-linux-gnu/crtn.o
[info] Running /home/sources/3-library/target/scala-2.11/samplelibrary-out
On
On
Unpowered
Powered
[success] Total time: 29 s, completed Jul 30, 2016 2:32:33 PM
>
```

Future

- Implement missing JDK parts
  - java.io
  - java.util.regex
  - java.util.concurrent
- Run stable on target architectures x86, ARM and iOS
- Scala libraries should be cross-compiled and published with .nir
  - Similarily how it works for Scala.JS now (.sjsir)
- Transparently compile Scala language to
  - JVM (via Scalac / Dotty)
  - JavaScript (via Scala.JS)
  - Native Code (via Scala Native)

Thanks! Questions?