HANDS-ON SCALA-NATIVE

GUILLAUME MASSÉ

MARTIN DUHEM

PRESENTATION PLAN

- What is Scala-Native ?
- Demo: Ncurse Bandwidth Monitor
- Setup (System/Sbt)
- Implementation
  - Draw Loop
  - Find Network Interface Name
  - Get Bitrate
  - Draw Bitrate Graph

WHAT IS SCALA-NATIVE ?

WHAT IS SCALA-NATIVE ?

- Created by Denys Shabalin (quasiquote)
- Open source: github.com/scala-native/scala-native
- Started August 2015

HOW DOES IT WORK?

- JVM: `*.scala -> *.class -> jvm`
- Native: `*.scala -> *.ll -> app.exe`

FEATURES 0.1.0

- Amazing C Interrop
- Garbage Collection
- Fast startup time
- IDE support (100% Scala)
- seamless sbt integration (sbt compile, crossProject)

FEATURES 0.2.0

- java.util.regex.*, java.io.*
- scala.concurrent.future
- Better support for strings and characters
- System properties
- ...

FEATURES 0.3.0

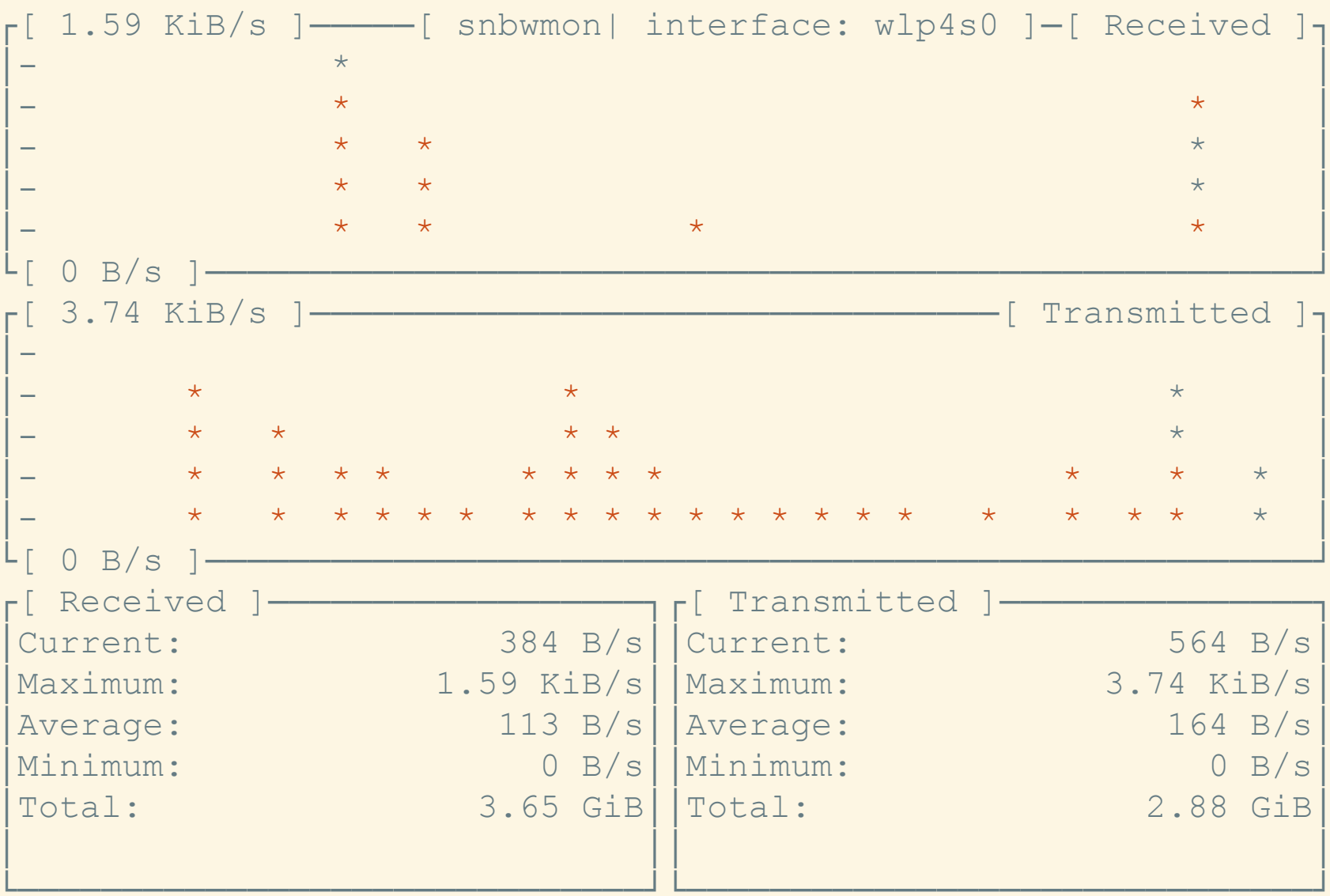
- Improved garbage collector
- java.nio.*, java.util.jar.*, java.util.zip.*
- sbt test

FEATURES 0.4.0?

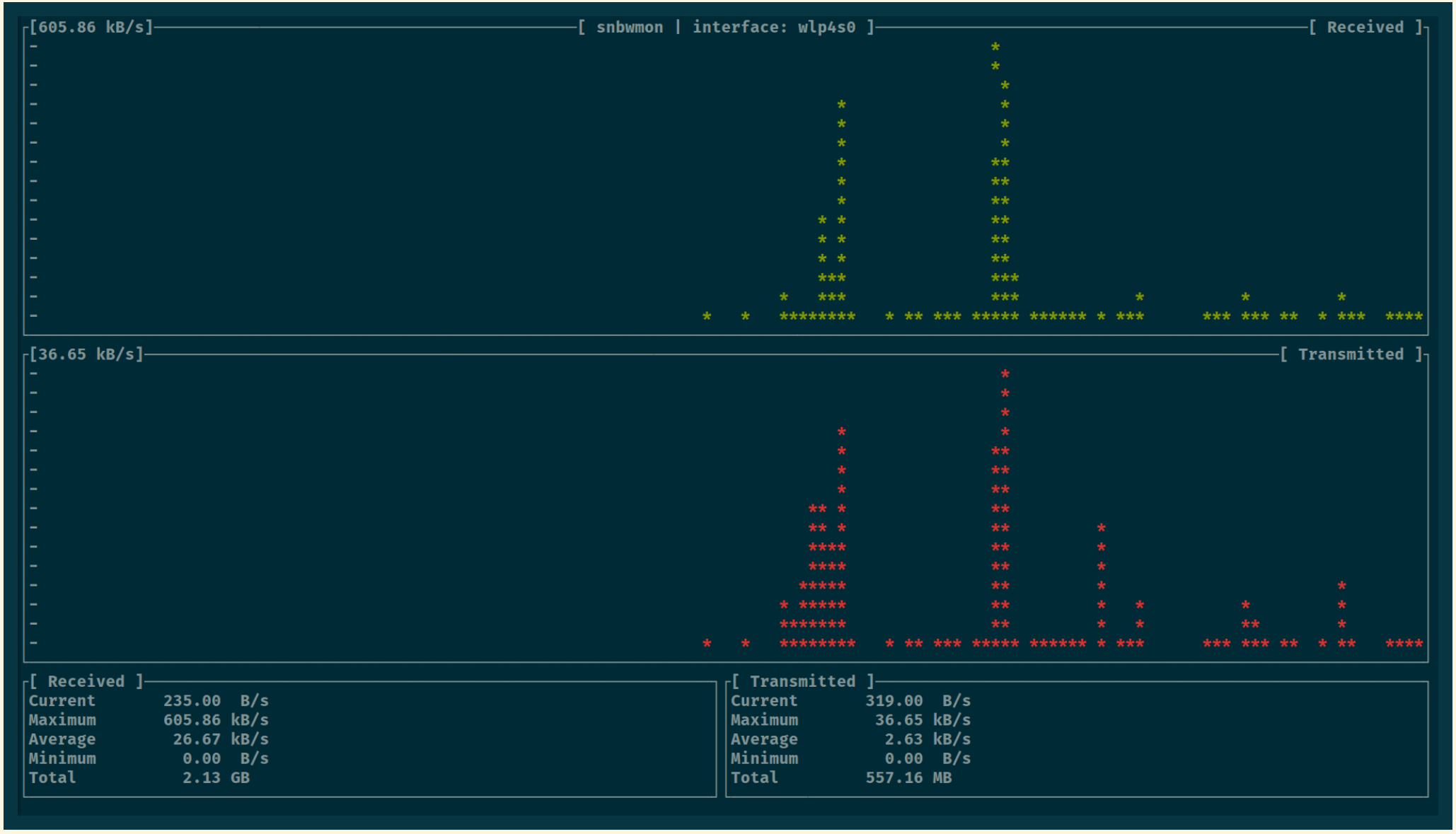
- Improved Interop (@densh)
- Windows support (@muxanick)
- Automatic binding generation (@jonas)


ollie
scalafmt-cli-native
target
scala-2.11

NCURSE BANDWIDTH MONITOR



https://github.com/causes-/nbwmon


[605.86 kB/s]
[snbwmon | interface: wlp4s0]
[Received]
[36.65 kB/s]
[Transmitted]
[Received]
Current 235.00 B/s
Maximum 605.86 kB/s
Average 26.67 kB/s
Minimum 0.00 B/s
Total 2.13 GB
[Transmitted]
Current 319.00 B/s
Maximum 36.65 kB/s
Average 2.63 kB/s
Minimum 0.00 B/s
Total 557.16 MB

SYSTEM SETUP

macOS:

```
$ brew install llvm \
               bdw-gc \
               re2 \
               ncurses
```

Ubuntu:

```
$ sudo apt-get install clang \
                       libgc-dev \
                       libunwind-dev \
                       libre2-dev \
                       libncurses-dev
```

Nix:

```
$ nix-shell .
```

SBT SETUP

project/plugins.sbt

```
addSbtPlugin("org.scala-native" % "sbt-scala-native" % "0.2.1")
```

build.sbt

```
enablePlugins(ScalaNativePlugin)

scalaVersion := "2.11.11"
```

IMPLEMENTATION PLAN

- Draw Loop
- Find Network Interface Name
- Get Bitrate
- Draw Bitrate Graph

DRAW LOOP

```
object LoopMain {
  def main(args: Array[String]): Unit = {
    waitLoop {
      println("tick")
    }
  }
}
```

DRAW LOOP (C VS SCALA)

```c
// In C
void waitLoop() {
  long timer  = 0;
  bool redraw = true;
  struct timeval tv;


  while(true){
    gettimeofday(&tv, NULL);

    if (timer < tv.tv_sec) {
      timer = tv.tv_sec;
      redraw = true;
    }

    if(redraw) {
      draw()
      redraw = false
    }
  }
}
```

```scala
// IN Scala
def waitLoop(body: => Unit): Unit = {
  var timer  = 0L
  var redraw = false
  val tv: Ptr[timeval] =
    stackalloc[timeval]

  while (true) {
    gettimeofday(tv, null)

    if (timer < tv.tv_sec) {
      timer = tv.tv_sec
      redraw = true
    }

    if (redraw) {
      body
      redraw = false
    }
  }
}
```

POSIX API

```c
//in C


int gettimeofday(
  struct timeval *tv,
  struct timezone *tz
);

struct timeval {
  time_t      tv_sec;
  suseconds_t tv_usec;
};
```

```
@extern
object posix {

  def gettimeofday(
    tv: Ptr[timeval],
    tz: Ptr[timezone]
  ): Unit = extern

  type timeval = CStruct2[
    time_t,
    suseconds_t
  ]
}
```

POSIX API (CONT.)

```
object posixops {
  implicit class timevalOps(val ptr: Ptr[timeval]) {
    def tv_sec: time_t      = !(ptr._1)
    def tv_usec: suseconds_t = !(ptr._2)
  }
}
```

IMPLEMENTATION PLAN

- Draw Loop
- Find Network Interface Name
- Get Bitrate
- Draw Bitrate Graph

IFADDRS API

```
int getifaddrs(
  struct ifaddrs **ifap);


void freeifaddrs(
  struct ifaddrs *ifa);


struct ifaddrs {
  struct ifaddrs  *ifa_next;
  char            *ifa_name;
  unsigned int     ifa_flags;
  struct sockaddr *ifa_addr;
  struct sockaddr *ifa_netmask;
  struct sockaddr *ifa_ifu;
  void            *ifa_data;
};
```

```
@extern
object Ifaddr {
  def getifaddrs(
    ifap: Ptr[Ptr[Ifaddrs]]
  ): CInt = extern

  def freeifaddrs(
    ifa: Ptr[Ifaddrs]
  ): Unit = extern

  type Ifaddrs = CStruct7[
    Ptr[CByte], // scala-native#634
    Ptr[CChar],
    CInt,
    Ptr[SockAddr],
    Ptr[SockAddr],
    Ptr[SockAddr],
    Ptr[CByte]
  ]
}
```

C FLAGS

```
class SiocgifFlags(val value: CInt) extends AnyVal {
  def &(other: SiocgifFlags): Boolean =
    (value & other.value) == other.value
}
object SiocgifFlags {
  val Up       = new SiocgifFlags(1 << 1)
  val Loopback = new SiocgifFlags(1 << 4)
  val Running  = new SiocgifFlags(1 << 6)
  // ...
}
```

$$$ RICHER TYPES $$$

```
implicit class IfaddrsOps(val ptr: Ptr[Ifaddrs]){
  // scala-native#634
  def next: Ptr[Ifaddrs]  = (!(ptr._1)).cast[Ptr[Ifaddrs]]

  def name: Ptr[CChar]        = !(ptr._2)
  def flags: SiocgifFlags     = new SiocgifFlags(!(ptr._3))
  def addr: Option[SockAddr] = Option(new SockAddrOps(!(ptr._4)))
  def data: Ptr[Byte]         = !(ptr._7)

  // scala-native#367 we need to manually box Ptr[T]
  def iterator: Iterator[IfaddrsOps] = new Iterator[IfaddrsOps]{
    private var current = new IfaddrsOps(ptr)

    def hasNext: Boolean = current.ptr.next != null

    def next(): IfaddrsOps = {
      current = new IfaddrsOps(current.next)
      current
    }
  }
}
```

GET NETWORK INTERFACES

```
def withIfaddrs[A](f: Iterator[IfaddrsOps] => A): A = {
  val ifaddrs = stackalloc[Ptr[Ifaddrs]]

  if (getifaddrs(ifaddrs) != -1) {
    val ret = f((!ifaddrs).iterator)
    freeifaddrs(!ifaddrs)
    ret
  } else {
    println("failed to getifaddrs")
    sys.exit(1)
  }
}
```

FIND NETWORK INTERFACE NAME

```
def findInterface: Option[String] = {
  withIfaddrs(_.find(interface =>
     (interface.flags & Up) &&
     (interface.flags & Running) &&
    !(interface.flags & Loopback)
  ).map(interface => fromCString(interface.name)))
}
```

IMPLEMENTATION PLAN

- Draw Loop
- Find Network Interface Name
- Get Bitrate
- Draw Bitrate Graph

TOTAL RECEIVED/TRANSMITTED

```
case class Counters(val rx: CUnsignedLong, val tx: CUnsignedLong)
object Counters {
  def apply(stats: Ptr[RtnlLinkStats]): Counters =
    Counters(stats.rxBytes, stats.txBytes)
}

def getCounter(interfaceName: String): Option[Counters] = {
  withIfaddrs(
    _.find(interface =>
      fromCString(interface.name) == interfaceName &&
      ifa.addr.map(_.family == Packet).getOrElse(false)
    )
    .map(interface =>
      Counters(ifa.data.cast[Ptr[RtnlLinkStats]])
    )
  )
}
```

TOTAL RECEIVED/TRANSMITTED (IN C)

```
int getData(char* ifname, unsigned long *tx_bytes,
                          unsigned long *rx_bytes) {
  struct ifaddrs *ifaddr, *ifa;
  int ret = 0;
  if (getifaddrs(&ifaddr) == -1) return;
  for (ifa = ifaddr; ifa != NULL; ifa = ifa->ifa_next) {
     if (ifa->ifa_addr == NULL)
       continue;

     if (!strncmp(ifname, ifa->ifa_name, IFNAMSIZ))
       continue;

     if (ifa->ifa_addr->sa_family == AF_PACKET &&
         ifa->ifa_data != NULL) {
       struct rtnl_link_stats *stats = ifa->ifa_data;
       *tx_bytes = stats->tx_bytes;
       *rx_bytes = stats->rx_bytes;
       ret = 1;
       break;
     }
  }
  freeifaddrs(ifaddr);
  return ret;
`
```

CATCH 22

```
CUnsignedInt, //  [6] tx_errors
CUnsignedInt, //  [7] rx_dropped
CUnsignedInt, //  [8] tx_dropped
CUnsignedInt, //  [9] multicast
CUnsignedInt, // [10] collisions
CUnsignedInt, // [11] rx_length_errors
CUnsignedInt, // [12] rx_over_errors
CUnsignedInt, // [13] rx_crc_errors
CUnsignedInt, // [14] rx_frame_errors
CUnsignedInt, // [15] rx_fifo_errors
CUnsignedInt, // [16] rx_missed_errors
CUnsignedInt, // [17] tx_aborted_errors
CUnsignedInt, // [18] tx_carrier_errors
CUnsignedInt, // [19] tx_fifo_errors
CUnsignedInt, // [20] tx_heartbeat_errors
CUnsignedInt, // [21] tx_window_errors
CUnsignedInt  // [22] rx_compressed

// we are limited to 22 fields scala-native#637
// it's ok to ignore those since we don't allocate RtnlLinkStats64
// CUnsignedInt, // [23] tx_compressed


// CUnsignedInt  // [24] rx_nohandler
```

< 22 WE ARE SAFE

```
ifa.data.cast[Ptr[RtnlLinkStats]]

implicit class RtnlLinkStatsOps(val ptr: Ptr[RtnlLinkStats]){
  def rxBytes: CUnsignedInt = !(ptr._3)
  def txBytes: CUnsignedInt = !(ptr._4)
}
```

IMPLEMENTATION PLAN

- Redraw Loop
- Find Network Interface Name
- Get Bitrate
- Draw Bitrate Graph

```
val size = windowSize(stdscr)
waitLoop {
  getCounter(interfaceName).foreach(data =>
    history += data
  )
  graphWindow(rxGraph, history, RX, Some(interfaceName), green)
  graphWindow(txGraph, history, TX, None, red)
  statsWindow(rxStats, history, RX)
  statsWindow(txStats, history, TX)
  doupdate()
}
```

```
@link("ncurses")
@extern
object ncurses {
  import ncursesh._
  @name("newwin")
  def newWindow(
    height: Int, width: Int,
    y: Int, x: Int): Ptr[Window] = extern

  // print formatted string
  @name("mvwprintw")
  def printFormatted(
    window: Ptr[Window],
    y: CInt, x: CInt,
    fmt: CString, args: CVararg*
  ): CInt = extern

  // print one char
  @name("mvwaddch")
  def printChar(
    window: Ptr[Window],
    y: CInt, x: CInt,
    ch: ChType
```

PLAIN SCALA

```
var lastTotal: Option[Counters],
txQueue: Queue[CUnsignedLong],
rxQueue: Queue[CUnsignedLong]){

def maximum(way: Way): Option[Double] =
  stats(way, _.max.toDouble)

def average(way: Way): Option[Double] =
  stats(way, q => q.sum.toDouble / q.size.toDouble)

def +=(v: Counters): CountersHistory = {
  lastTotal.foreach{ lv =>
    if(txQueue.size > maxSize) {
      txQueue.dequeue()
    }

    if(rxQueue.size > maxSize) {
      rxQueue.dequeue()
    }

    val t = v - lv
    rxQueue += t.rx
    txQueue += t.tx
  }
```

PLAIN SCALA

```
    printFormatted(window, 0, center, toCString(text));
  }

  // For each column...
  history.maximum(way).foreach{ max =>
    val (rate, unit) = showBytes(max)
    printFormatted(window, 0, 1, c"[%.2f %s/s]", rate, toCString(unit))
    // ... starting from the right ...
    history.getQueue(way).reverse.zipWithIndex.foreach{ case (value, i)
      val col = size.width - i - 2
      val border = 2
      val h = Math.ceil(value.toDouble / max.toDouble * (size.height - b
      var j = size.height - 2
      var jj = 0
      // ... print each row
      while(j > 0 && jj < h) {
        printChar(window, j, col, '*')
        j -= 1
        jj += 1
      }

    }
  }
}
```

DEMO

WHAT IS NEXT ?

TRY IT OUT: git.io/vSMS7

SBT NEW DUHEMM/CMDLINE.G8

- Simple example of command line app with Native
- Everybody has lots of small tools they wrote
- Port them to Native, this can serve as inspiration!
- ... Or simply complete this app :)

HACKING IDEAS

- use crossProject on existing libraries
- port cli tools to native (scalafmt, coursier, ...)
- Hack on your Raspberry PI 3 (IOT)
- ...

QUESTIONS ?

PLEASE VOTE (:--))