Java Native Runtime

The Missing Link

http://www.talkorigins.org/faqs/homs/toumai.jpg

Me

- Charles Oliver Nutter
- headius@headius.com
- @headius
- http://blog.headius.com
- Languages, indy, optimization, all that jazz

Java Native Runtime

- **Java** API
- for calling **Native** code
- supported by a rich **Runtime** library
- You may be familiar with **JNA**
- https://github.com/jnr

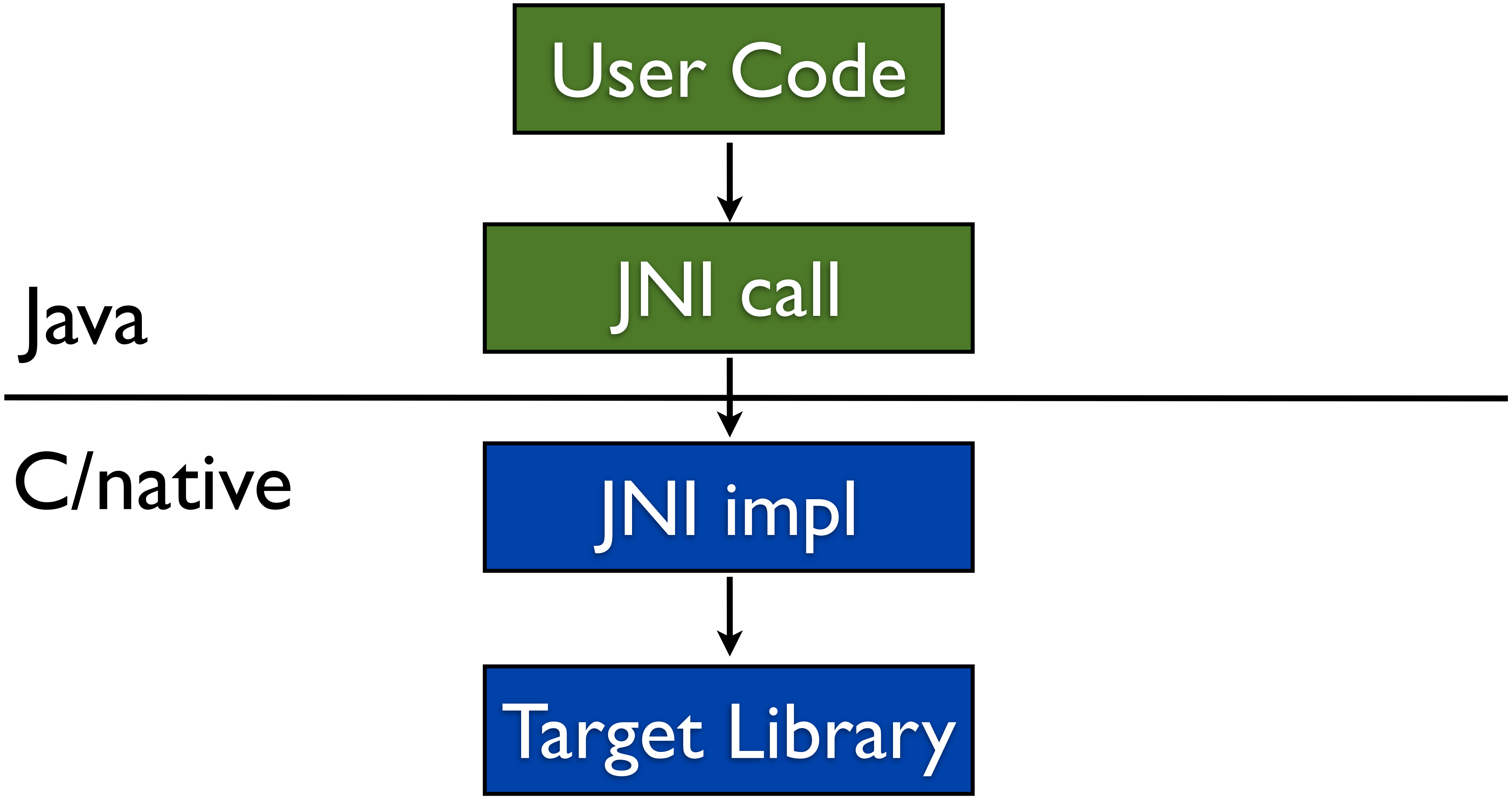
A Java API for binding native libraries and native memory

Justifications

- NIO, NIO.2 could have been FFI
  - Native IO, symlinks, FS-walking,
- Unmanaged memory
- Selectable stdio, process IO
- Low-level or other sockets (UNIX, ICMP, ...)
- New APIs (graphics, crypto, OS, ...)

Fear

- Crashing
- Security
- Platform-dependence



User Code

JNI call

Java

C/native

JNI impl

Target Library

JNI

```java
public class GetPidJNI {
    public static native long getpid();

    public static void main( String[] args ) {
        getpid();
    }

    static {
        System.load(System.getProperty("user.dir") + "/getpidjni.dylib");
    }
}
```

JNI

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class com_headius_jnr_presentation_GetPidJNI */

#ifndef _Included_com_headius_jnr_presentation_GetPidJNI
#define _Included_com_headius_jnr_presentation_GetPidJNI
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:     com_headius_jnr_presentation_GetPidJNI
 * Method:    getpid
 * Signature: ()J
 */
JNIEXPORT jlong JNICALL Java_com_headius_jnr_1presentation_GetPidJNI_getpid
  (JNIEnv *, jclass);

#ifdef __cplusplus
}
#endif
#endif
```

JNI

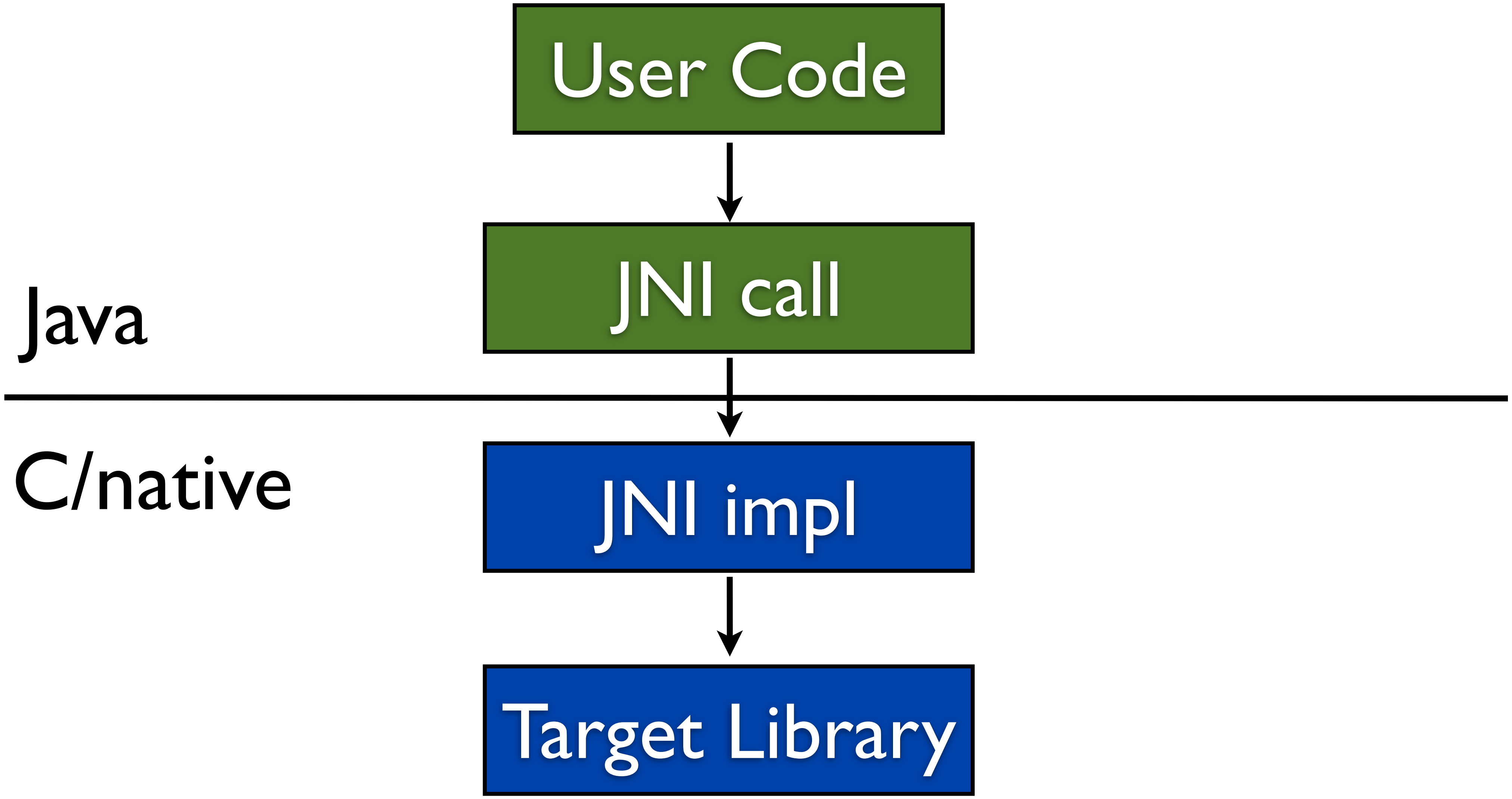
```
#include "com_headius_jnr_presentation_GetPidJNI.h"

jlong JNICALL Java_com_headius_jnr_1presentation_GetPidJNI_getpid
  (JNIEnv *env, jclass c) {

  return getpid();
}
```

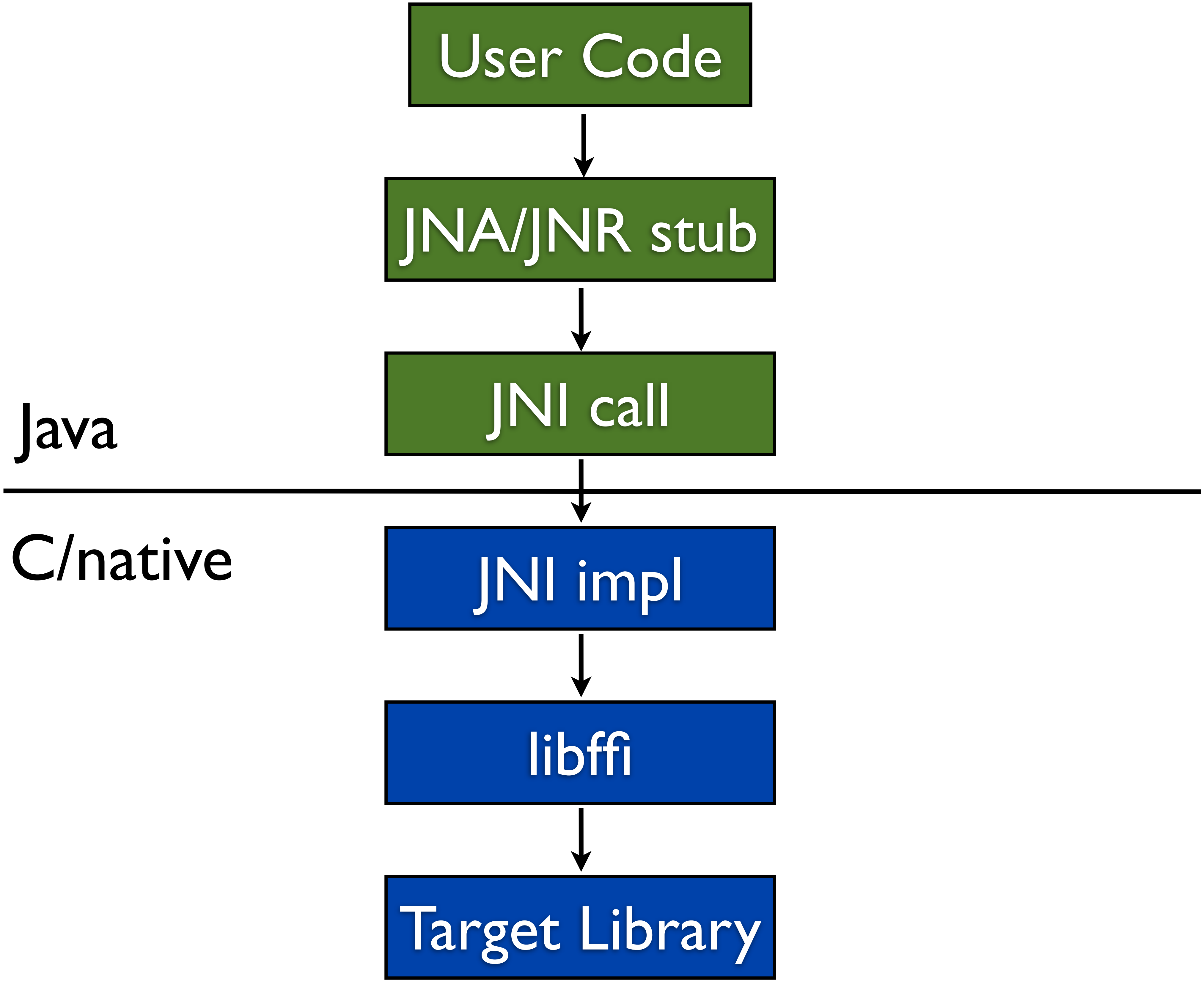
JNI

```
$ gcc -I $JAVA_HOME/include -I $JAVA_HOME/include/darwin -L
$JAVA_HOME/jre/lib/ -dynamiclib -ljava -o getpidjni.dylib
com_headius_jnr_presentation_GetPidJNI.c

$ java -Djava.library.path=`pwd` -cp target/jnr_presentation-1.0-
SNAPSHOT.jar com.headius.jnr_presentation.GetPidJNI
```

Nobody enjoys calling native libraries...

...but if you have to call native libraries, you might as well enjoy it.


User Code
JNI call
Java
C/native
JNI impl
Target Library


User Code
JNA/JNR stub
JNI call
Java
C/native
JNI impl
libffi
Target Library

JNA

```java
import com.sun.jna.Library;
import com.sun.jna.Native;

public class GetPidJNAExample {
    public interface GetPid extends Library {
        long getpid();
    }

    public static void main(String[] args) {
        GetPid getpid = (GetPid)Native.loadLibrary(GetPid.class);

        getpid.getpid();
    }
}
```

JNR

```java
import jnr.ffi.LibraryLoader;
import jnr.ffi.annotations.IgnoreError;
import jnr.ffi.provider.FFIProvider;

public class GetPidJNRExample {
    public interface GetPid {
        @IgnoreError
        long getpid();
    }

    public static void main( String[] args ) {
        LibraryLoader<GetPid> loader =
                FFIProvider
                    .getSystemProvider()
                    .createLibraryLoader(GetPid.class);

        GetPid getpid = loader.load("c");

        getpid.getpid();
    }
}
```

Who To Blame

- Wayne Meissner (@wmeissner)
  - Author, maintainer, expert
- JRuby Team (@jruby)
  - Primary users, drivers, promoters
- Ruby Community
  - For stubbornly insisting on native APIs

Use in JRuby

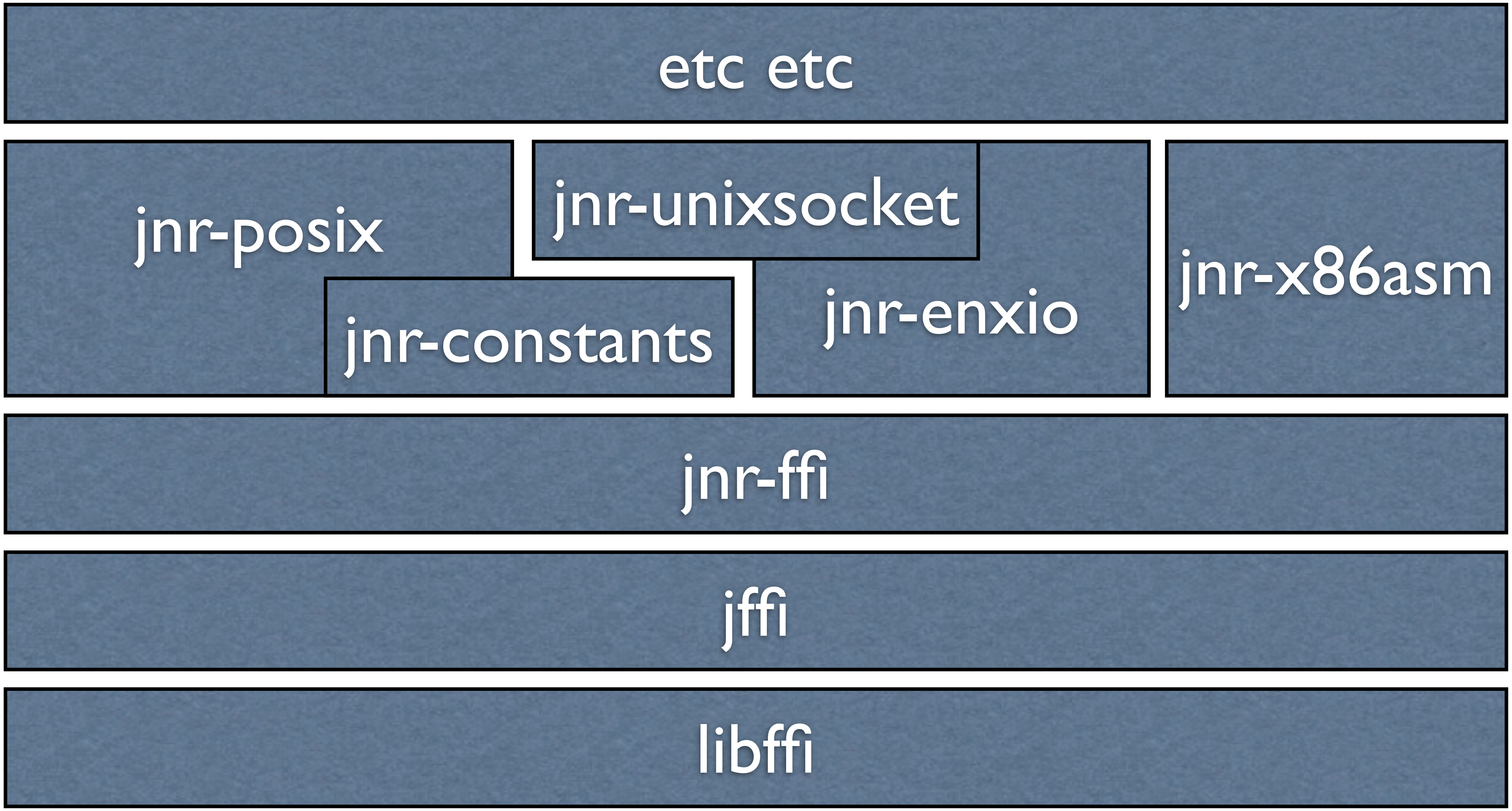
- OS-level: filesystem, spawn, stat, tty/pty/fcntl
- Libraries: graphics, crypto, nosql, mq, db
- C ext replacement/transition
- Langs: llvm, clang, V8

We Had No Choice.

Java Native Runtime

https://github.com/jnr

Layered Runtime



etc etc

jnr-posix

jnr-unixsocket

jnr-constants

jnr-enxio

jnr-x86asm

jnr-ffi

jffi

libffi

JFFI

- Java Foreign Function Interface
- libffi-based
- Low-level...not the API you're looking for
- Broad platform support
- https://github.com/jnr/jffi

JFFI Platforms

- Darwin (OS X): universal (+ppc?)
- Linux: i386, x86_64, arm, ppc, ppc64, s390x
- Windows: i386, x86_64
- FreeBSD, OpenBSD: i386, x86_64
- SunOS: i386, x86_64, sparc, sparcv9
- AIX: ppc
- OpenVMS, AS/400: builds out there somewhere
- If your platform isn't here, contribute a build

JRuby User Platforms

- Darwin (OS X): universal (+ppc?)
- Linux: i386, x86_64, arm, ppc, ppc64, s390x
- Windows: i386, x86_64
- FreeBSD, OpenBSD: i386, x86_64
- SunOS: i386, x86_64, sparc, sparcv9
- AIX: ppc
- OpenVMS, AS/400: builds out there somewhere
- If your platform isn't here, contribute a build

JNR-FFI

- User-oriented API
- Roughly equivalent to what JNA gives you
- Functions, structs, callbacks, memory
- https://github.com/jnr/jnr-ffi

Why Not JNA?

- Preprocessor constants?
- Standard API sets out of the box
- C callbacks?
- Performance?!?

Constants

jnr-constants

- Preprocessor constants (#define)
- Generation tools
- Several platforms, families built in
- https://github.com/jnr/jnr-constants

Provided Constants

- AddressFamily.java
- ConstantResolver.java
- Errno.java
- Fcntl.java
- INAddr.java
- IPProto.java
- NameInfo.java
- OpenFlags.java
- PRIO.java
- ProtocolFamily.java
- RLIM.java
- RLIMIT.java
- Shutdown.java
- Signal.java
- Sock.java
- SocketLevel.java
- SocketOption.java
- Sysconf.java
- TCP.java
- WaitFlags.java

```
// WARNING: This file is autogenerated. DO NOT EDIT!
// Generated Tue Feb 24 09:44:06 +1000 2009
package jnr.constants.platform.linux;
public enum Sock implements jnr.constants.Constant {
SOCK_STREAM(1),
SOCK_DGRAM(2),
SOCK_RAW(3),
SOCK_RDM(4),
SOCK_SEQPACKET(5);
// SOCK_MAXADDRLEN not defined
private final int value;
private Sock(int value) { this.value = value; }
public static final long MIN_VALUE = 1;
public static final long MAX_VALUE = 5;

public final int value() { return value; }
public final int intValue() { return value; }
public final long longValue() { return value; }
}
```

```
require 'gen/ConstGenerator'
def gen_sock_java(options)
  ConstGenerator.new 'platform.sock', options do |cg|
    cg.include "sys/socket.h"
    %w[
      SOCK_STREAM
      SOCK_DGRAM
      SOCK_RAW
      SOCK_RDM
      SOCK_SEQPACKET
      SOCK_MAXADDRLEN
    ].each {|c| cg.const c}
  end
end
```

Generation Tools

Ruby FFI

- Ruby DSL for binding native code
- Escaping from MRI's invasive C API
- Slowly taking over the Ruby world
- Built atop JNR in JRuby (of course)

Ruby FFI example

```ruby
require 'ffi'

module GetPid
  extend FFI::Library
  ffi_lib 'c'
  attach_function :getpid, [], :uint
end

GetPid.getpid
```

Ruby FFI example

```
class Timeval < FFI::Struct
  layout :tv_sec => :ulong,
         :tv_usec => :ulong
end

module LibC
  extend FFI::Library
  ffi_lib FFI::Library::LIBC
  attach_function :gettimeofday,
                  [ :pointer, :pointer ],
                  :int
end

t = Timeval.new
LibC.gettimeofday(t.pointer, nil)
```

C Sucks

- Inter and intra-platform oddities
- Preprocessor macros
- No binary metadata
- Struct layout
- We **will** need to generate FFI bindings

Ruby FFI Generator

- https://github.com/neelance/ffi-gen
- Clang-based Ruby FFI generator
- Used to generate clang binding it uses
  - It's meta!
- Could be trivially made to generate Java

```ruby
require "ffi/gen"

FFI::Gen.generate(
  module_name: "Clang",
  ffi_lib:     "clang",
  headers:     ["clang-c/Index.h"],
  cflags:      `llvm-config --cflags`.split(" "),
  prefixes:    ["clang_", "CX"],
  output:      "clang-c/index.rb"
)
```

```
# A single translation unit, which resides in an index.
class TranslationUnitImpl < FFI::Struct
  layout :dummy, :char
end

# Identifies a specific source location within a translation
# unit.
#
# Use clang_getExpansionLocation() or clang_getSpellingLocation()
# to map a source location to a particular file, line, and column.
#
# = Fields:
# :ptr_data ::
#   (Array<FFI::Pointer(*Void)>)
# :int_data ::
#   (Integer)
class SourceLocation < FFI::Struct
  layout :ptr_data, [:pointer, 2],
         :int_data, :uint
end
```

```ruby
# Retrieves the source location associated with a given file/line/column
# in a particular translation unit.
#
# @method get_location(tu, file, line, column)
# @param [TranslationUnitImpl] tu
# @param [FFI::Pointer(File)] file
# @param [Integer] line
# @param [Integer] column
# @return [SourceLocation]
# @scope class
attach_function :get_location, :clang_getLocation,
                [TranslationUnitImpl, :pointer, :uint, :uint],
                SourceLocation.by_value
```

Support Libraries

jnr-posix

- Pre-bound set of POSIX functions
- Mostly driven by what JRuby, Jython use
- Goal: 100% of POSIX bound to Java
- Bonus: partial pure-Java backend

```
public int chmod(String string, int i);
public int chown(String string, int i, int i1);
public int execv(String string, String[] strings);
public int execve(String string, String[] strings, String[] strings1);
public int fork();
public int seteuid(int i);
public int getgid();
public String getlogin();
public int getpgid();
public int getpgid(int i);
public int getpgrp();
public int getpid();
public int getppid();
public Passwd getpwent();
public Passwd getpwuid(int i);
public Passwd getpwnam(String string);
public Group getgrgid(int i);
public Group getgrnam(String string);
public int getuid();
public boolean isatty(FileDescriptor fd);
public int kill(int i, int i1);
public int symlink(String string, String string1);
public int link(String string, String string1);
public String readlink(String string) throws IOException;
public String getenv(String string);
public int setenv(String string, String string1, int i);
public int unsetenv(String string);
public int getpriority(int i, int i1);
public int setpriority(int i, int i1, int i2);
public int setuid(int i);
public FileStat stat(String string);
public int stat(String string, FileStat fs);
public int umask(int i);
public Times times();
public int utimes(String string, long[] longs, long[] longs1);
public int waitpid(int i, int[] ints, int i1);
public int wait(int[] ints);
public int errno();
public void errno(int i);
public int posix_spawnp(String string, List<? extends SpawnFileAction> list,
List<? extends CharSequence> list1, List<? extends CharSequence> list2);
```

```
POSIX posix = POSIXFactory.getPOSIX(
        new JRubyPOSIXHandler(this),
        isNativeEnabled);
```

```java
public interface POSIXHandler {
    public void error(Errno errno, String string);
    public void unimplementedError(String string);
    public void warn(WARNING_ID wrngd, String string, Object[] os);
    public boolean isVerbose();
    public File getCurrentWorkingDirectory();
    public String[] getEnv();
    public InputStream getInputStream();
    public PrintStream getOutputStream();
    public int getPID();
    public PrintStream getErrorStream();
}
```

jnr-x86asm

- Generate and link ASM via JNI
- Used internally by jnr-ffi
- https://github.com/jnr/jnr-x86asm

jnr-enxio

- Extended Native X-platform IO
- NIO-compatible JNR-backed IO library
  - Read, write, select (kqueue, epoll, etc)
  - Low-level fcntl control
- https://github.com/jnr/jnr-enxio

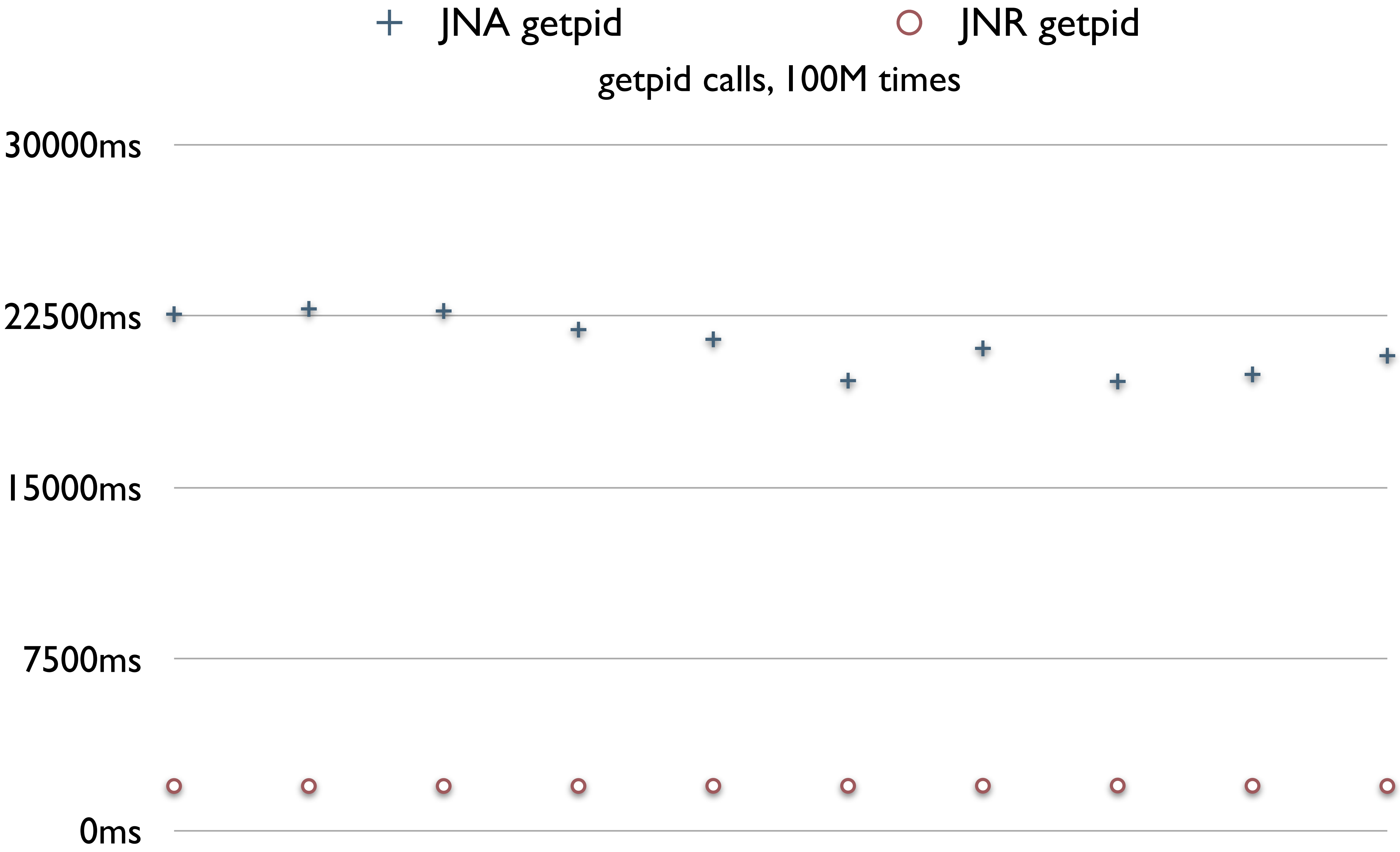
```
public class NativeSocketChannel
        extends AbstractSelectableChannel
        implements ByteChannel, NativeSelectableChannel {
    public NativeSocketChannel(int fd);
    public NativeSocketChannel(int fd, int ops);
    public final int validOps();
    public final int getFD();
    public int read(ByteBuffer dst) throws IOException;
    public int write(ByteBuffer src) throws IOException
    public void shutdownInput() throws IOException;
    public void shutdownOutput() throws IOException;
}
```

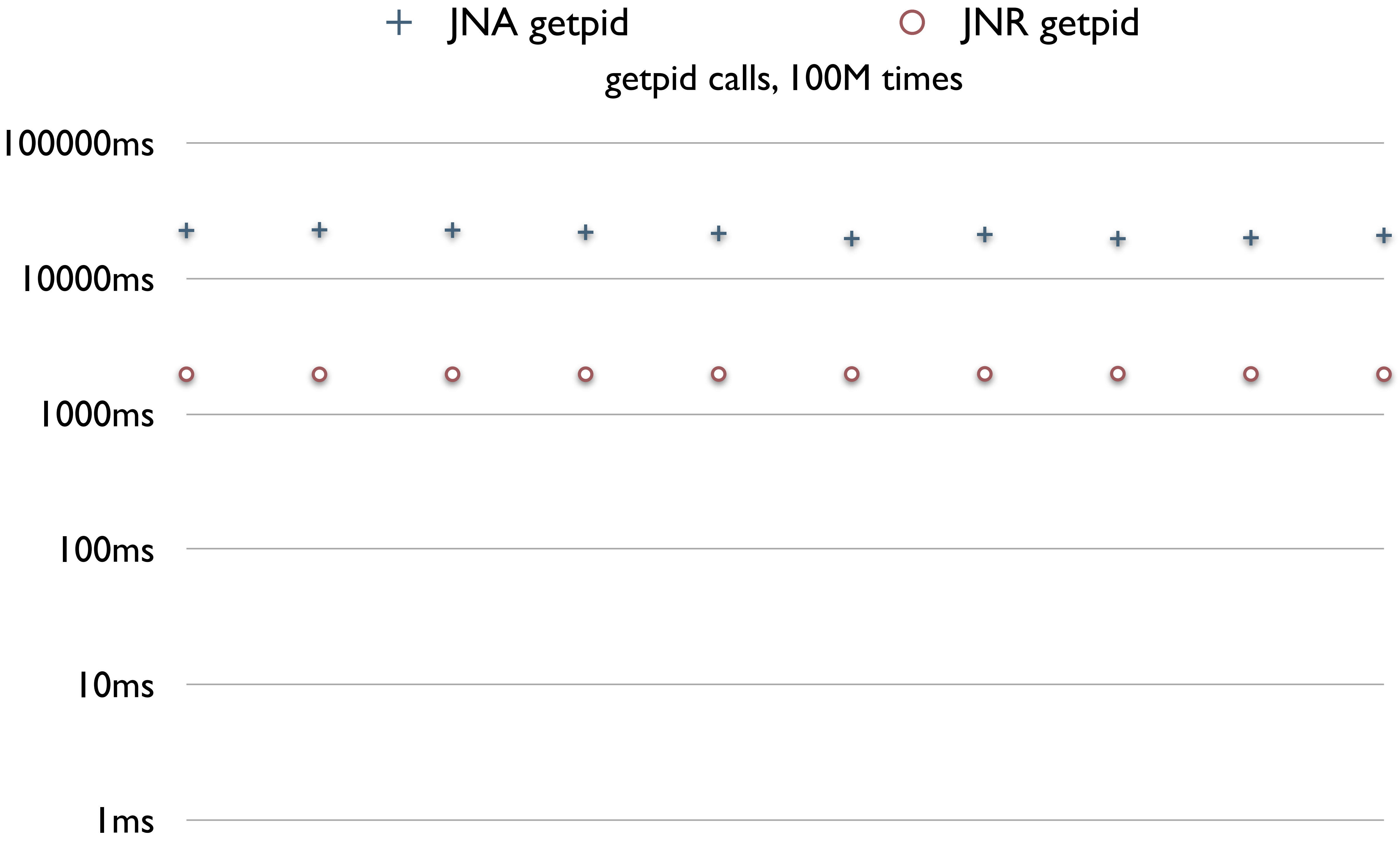
jnr-unixsocket

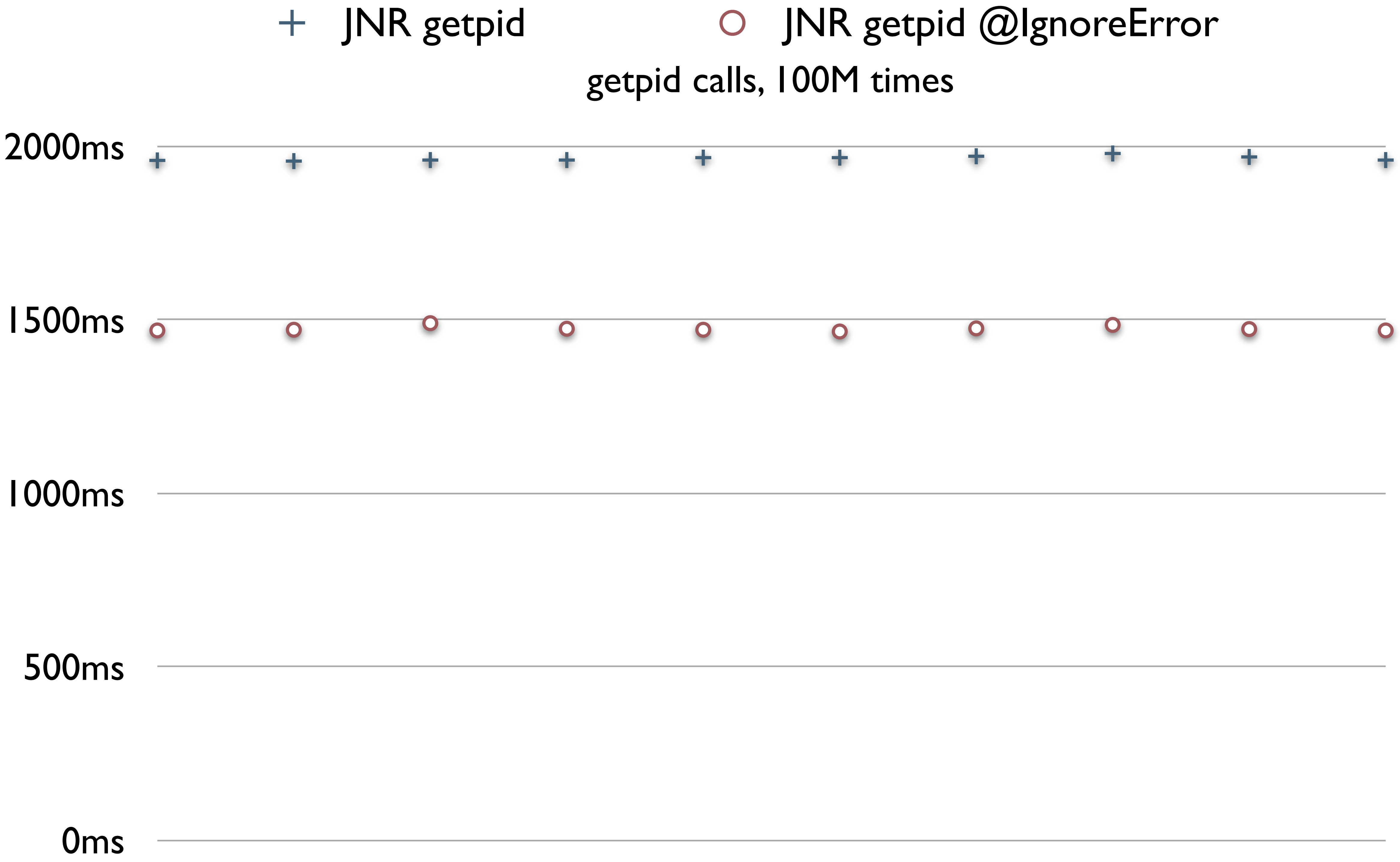
- UNIX sockets for NIO
- Built atop jnr-enxio
- Fully selectable, etc
- https://github.com/jnr/jnr-unixsocket

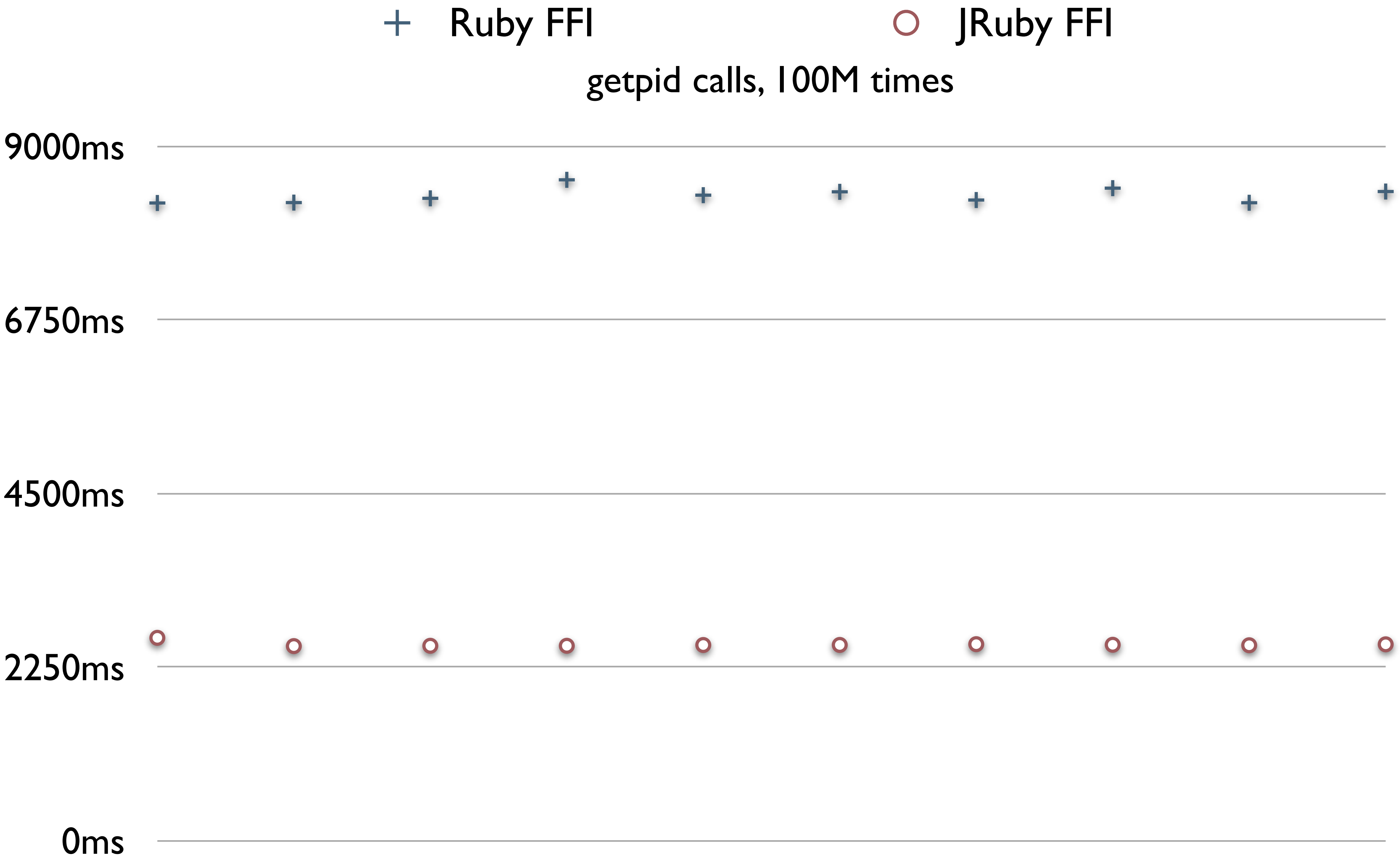
```java
public class UnixSocketChannel extends NativeSocketChannel {
    public static final UnixSocketChannel open(UnixSocketAddress remote) throws
IOException {
        UnixSocketChannel channel = new UnixSocketChannel();
        channel.connect(remote);
        return channel;
    }

    private UnixSocketChannel() throws IOException {
        super(Native.socket(ProtocolFamily.PF_UNIX, Sock.SOCK_STREAM, 0),
                SelectionKey.OP_CONNECT | SelectionKey.OP_READ |
                SelectionKey.OP_WRITE);
        state = State.IDLE;
    }
...
   private final boolean doConnect(SockAddrUnix remote) throws IOException {
        if (Native.connect(getFD(), remote, remote.length()) != 0) {
            Errno error =
                    Errno.valueOf(

LastError.getLastError(jnr.ffi.Runtime.getSystemRuntime()));

            switch (error) {
                case EAGAIN:
                case EWOULDBLOCK:
                    return false;
...
```

Performance


+ JNA getpid
○ JNR getpid
getpid calls, 100M times
30000ms
22500ms
15000ms
7500ms
0ms


+ JNA getpid
○ JNR getpid
getpid calls, 100M times
100000ms
10000ms
1000ms
100ms
10ms
1ms


+ JNR getpid
○ JNR getpid @IgnoreError
getpid calls, 100M times
2000ms
1500ms
1000ms
500ms
0ms


+ Ruby FFI
○ JRuby FFI
getpid calls, 100M times
9000ms
6750ms
4500ms
2250ms
0ms

Trying Really Hard...

Bytecode Stub to JNI

```
public final class GetPidJNRExample$GetPid$jnr$ffi$0
        extends jnr/ffi/provider/jffi/AbstractAsmLibraryInterface
        implements GetPidJNRExample$GetPid  {

  // access flags 0x11
  public final getpid()J
    GETSTATIC AbstractAsmLibraryInterface.ffi : Lcom/kenai/jffi/Invoker;
    ALOAD 0
    GETFIELD GetPidJNRExample$GetPid$jnr$ffi$0.callContext_1 : Lcom/
kenai/jffi/CallContext;
    ALOAD 0
    GETFIELD GetPidJNRExample$GetPid$jnr$ffi$0.functionAddress_1 : J
    INVOKEVIRTUAL Invoker.invokeL0 (Lcom/kenai/jffi/CallContext;J)J
    LRETURN
```

Bytecode Stub to JNI

```
public final long invokeL0(CallContext context, long function) {
    // <editor-fold defaultstate="collapsed" desc="Compiled Code">
    /* 0: aload_1
     * 1: getfield      #9  // Field com/kenai/jffi/CallContext.contextAddress:J
     * 4: lload_2
     * 5: invokestatic  #26 // Method com/kenai/jffi/Foreign.invokeL0:(JJ)J
     * 8: lreturn
     *  */
    // </editor-fold>
}
```

Indy from Ruby to JNI

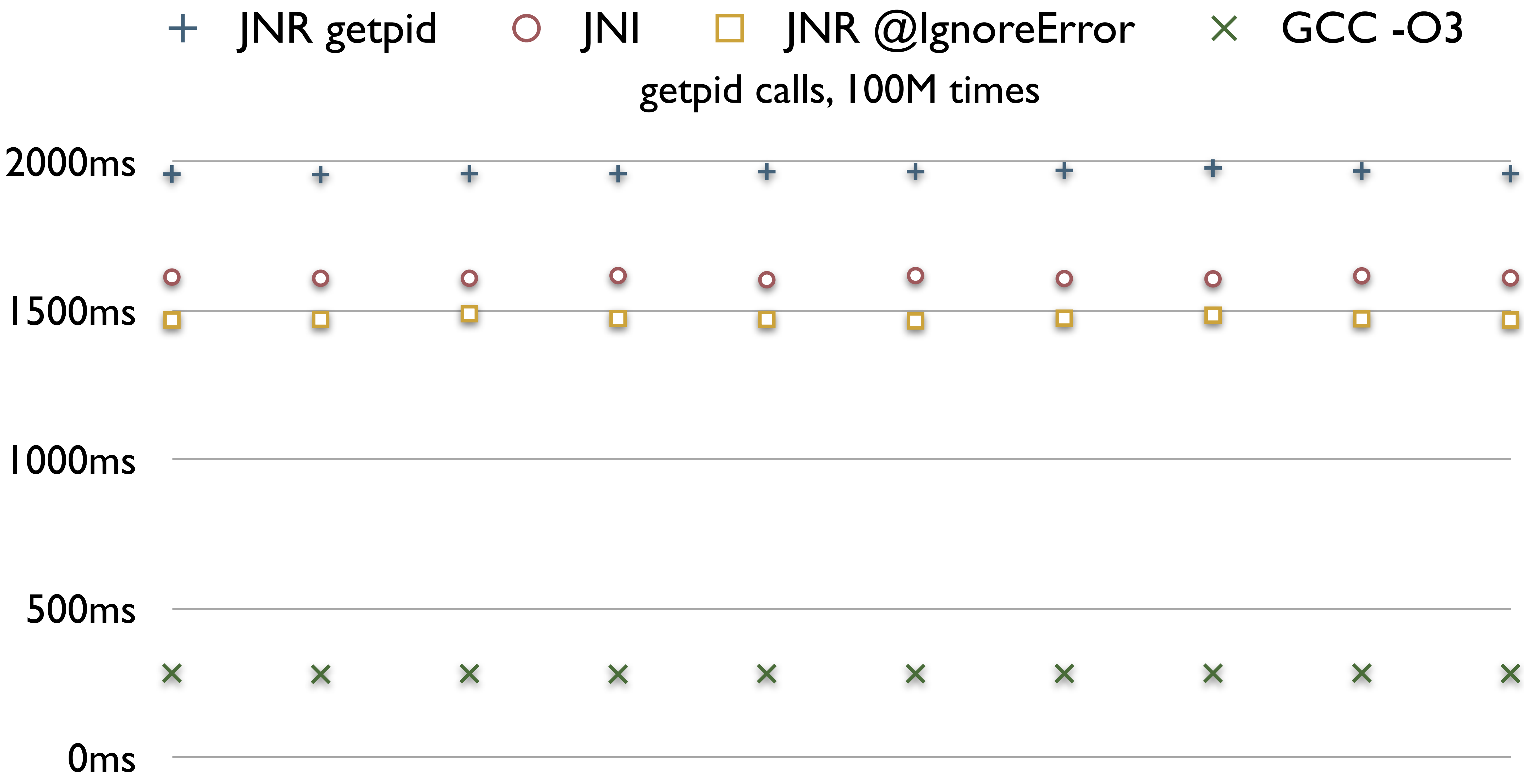
```ruby
require 'ffi'

module GetPid
  extend FFI::Library
  ffi_lib 'c'
  attach_function :getpid, [], :uint
end

def go; GetPid.getpid; end
```

```
{0x000000014ba85810} 'invokeI0' '(JJ)I' in 'com/kenai/jffi/Foreign')}
 0x000000011192233f: mov    %rsp,%rbp
                     ;*invokestatic linkToStatic
                     ; - java.lang.invoke.LambdaForm$DMH/731395981::invokeStatic_JJ_I@13
                     ; - java.lang.invoke.LambdaForm$BMH/1757676444::reinvoke@32
                     ; - java.lang.invoke.LambdaForm$MH/892529689::collect@5
                     ; - java.lang.invoke.LambdaForm$DMH/1058025095::invokeSpecial_LLLL_L@16
                     ; - java.lang.invoke.LambdaForm$MH/152134087::guard@50
                     ; - java.lang.invoke.LambdaForm$DMH/1058025095::invokeSpecial_LLLL_L@16
                     ; - java.lang.invoke.LambdaForm$MH/1580893732::guard@50
                     ; - java.lang.invoke.LambdaForm$MH/1781256139::linkToCallSite@14
                     ; - getpid_bench::method__1$RUBY$go@9 (line 13)
```

Generated JNI Stubs

```
public final class GetPidJNRExample$GetPid$jnr$ffi$0
        extends jnr/ffi/provider/jffi/AbstractAsmLibraryInterface
        implements GetPidJNRExample$GetPid  {

  // access flags 0x111
  public final native getpid()J
...
}

GetPidJNRExample$GetPid$jnr$ffi$0.getpid ()J
       0: sub rsp, 0x8
       4: mov rax, 0x0
       b: call 0x22
      10: mov [rsp], rax
      14: call 0xfffffffffe2b740
      19: mov rax, [rsp]
      1d: add rsp, 0x8
      21: ret
      22: <indirect call trampolines>
```

But There's More to Do



```
mov    0x8(%rbx),%r11d    ; implicit exception: dispatches to 0x000000010e31f0f8
cmp    $0x2232118f,%r11d  ;   {oop('.../GetPidJNRExample$GetPid$jnr$ffi$0')}
jne    0x000000010e31f0f8 ;*aload_0
                          ; - GetPidJNRExample::benchGetPid@12 (line 26)
mov    %rbx,%r10          ;*invokeinterface getpid
                          ; - GetPidJNRExample::benchGetPid@13 (line 26)
jmp    0x000000010e31f049
...
mov    %r10,0x8(%rsp)
mov    %r13,(%rsp)        ;*aload_0
                          ; - GetPidJNRExample::benchGetPid@12 (line 26)
mov    %r10,%rsi
xchg   %ax,%ax
callq  0x000000010e2cfc60 ; OopMap{[8]=Oop off=156}
                          ;*invokeinterface getpid
                          ; - GetPidJNRExample::benchGetPid@13 (line 26)
                          ;   {optimized virtual_call}
```

```
callq  <getpid address>   ; - libSystem.B.dylib
                          ;*invokeinterface getpid
                          ; - GetPidJNRExample::benchGetPid@13 (line 26)
                          ;   {optimized virtual_call}
```

We Need JVM Help

- Standard FFI API in JDK
- JIT intelligence
  - Drop JNI overhead where possible
  - Bind native call directly at call site
- Security policies, segv protection, etc

It's Time for an FFI JSR

Thank You!

- Charles Oliver Nutter
- headius@headius.com
- @headius
- http://blog.headius.com