Multi-Language Projects
One-Slide Summary

• Many modern software projects involve code written in multiple languages. This can involve a common bytecode or C native method interfaces.

• Native code interfaces can be understood in terms of (1) data layout and (2) special common functions to manipulate managed data.

• Performance modeling and debugging are complicated in multi-language projects.
Course Goals

Includes platform- and language-independent code, as well as multi-language projects.

At the end of this course, you will be acquainted with the fundamental concepts in the design and implementation of high-level programming languages. In particular, you will understand the theory and practice of lexing, parsing, semantic analysis, and code interpretation. You will also have gained practical experience programming in multiple different languages.
Lecture Outline

• Motivating Example
  - XOR (String Cryptography)

• Ocaml + C
  - Object Layout, Type Tags
  - Interfacing

• Python + C
  - Interfacing

• Java + C
  - Interfacing
Motivating Example

- Take out a piece of paper
- First: record every word you heard.
  - This will be hard.
- Second: translate.

practice, they became skilled storytellers:

... So there I was, trapped in the Cave of Death, staring into the drooling jaws of Mongo, The Moose From Hell . . .

This narrative style reached a climax with the invention of the romance novel:

... Helga, the voluptuous Moose Queen, slowly peeled off her gown and uttered a moan as the mighty Ragnar clenched her in his tawny arms. “Be gentle, my warrior,” she sighed as he ran his tongue down her neck. “Yaarrrrgggh!” he grunted. Helga’s bosom heaved with desire as Ragnar’s hungry kisses grew ever more furious. “Yes!” she cried. “Yes!”
Speech Perception, Segmentation

- The spectrogram is for the phrase “I owe you”
  - cf. “Raw Data Layout”
  - Note: no obvious boundaries
Motivating Example

In un mondo splendido, colorato e magico
Little ponies vivono, in pace sempre in armonia
Timidi e simpatici, burberi e romantici
Sono i caratteri, degli amici che troverai
Ed ogni giorno crescerai, quanti problemi risolverai
Insieme agli altri pony, lo sai, ti divertirai!

Vola e vai, my little pony, se nuovi amici vorrai incontrare
Prendi il volo, ascolta il cuore, ed ogni avventura potrai affrontare!
Vola e vai, my little pony, realizza i tuoi sogni e non ti fermare!
In un mondo splendido, colorato e magico
Little ponies vivono, in pace sempre in armonia
Timidi e simpatici, burberi e romantici
Sono i caratteri, degli amici che troverai
Ed ogni giorno crescerai, quanti problemi risolverai
Insieme agli altri pony, lo sai, ti divertirai!

Vola e vai, my little pony, se nuovi amici vorrai incontrare
Prendi il volo, ascolta il cuore, ed ogni avventura potrai affrontare!
Vola e vai, my little pony, realizza i tuoi sogni e non ti fermare!
Motivating Example

In un mondo splendido, colorato e magico
Little ponies vivono, in pace sempre in armonia
Timidi e simpatici, burberi e romantici
Sono i caratteri, degli amici che troverai
Ed ogni giorno crescerai, quanti problemi risolverai
Insieme agli altri pony, lo sai, ti divertirai!
Vola e vai, my little pony, se nuovi amici vorrai incontrare
Prendi il volo, ascolta il cuore, ed ogni avventura potrai affrontare!
Vola e vai, my little pony, realizza i tuoi sogni e non ti fermare!
Multi-Language Projects In Two Stages

- First, reason about the **raw data layout**
- Second, translate **concepts** you already know

- We will reason about the raw data layout using C and Assembly
  - Projects almost always use C for performance-critical kernels and low-level OS/hardware interfacing.
  - C is the Lingua Franca of multi-language projects.
Traditional Multi-Language Projects

• **Application Kernel**
  - Statically Typed, Optimized, Compiled, interfaces with OS and libraries.

• **Scripts**
  - Dynamically Typed, Interpreted, Glue Components, Business Logic.

• **Examples:** Emacs (C / Lisp), Adobe Lightroom (C++ / Lua), NRAO Telescope (C / Python), Google Android (C / Java), most games (C++ / Lua),
Bytecode
Multi-Language Projects

- **Microsoft's Common Language Runtime** of Managed Code in the .NET Framework
  - C++, C#, J#, F#, Visual Basic, ASP, etc.
  - Common Language Infrastructure

- **Java Bytecode**, Java Virtual Machine, Java Runtime Environment
  - Java, Scala, JRuby, JScheme, Jython, Fortress, etc.
Why Cover “Multi-Language”?

- Increasingly common. 2009 developer quote:
  - “My last 4 jobs have been apps that called: Java from C#, and C# from F#; Java from Ruby; Python from Tcl, C++ from Python, and C from Tcl; Java from Python, and Java from Scheme (And that's not even counting SQL, JS, OQL, etc.)”

- Use the best tool for the job (Course Goal!)
  - Example: concurrency might be better handled in OCaml (immutable functional) or Ruby (designed to hide such details), while low-level OS or hardware access is much easier in C or C++, while rapid prototyping is much easier in Python or Lua
Disadvantages of Multi-Language Projects

• Integrating data and control flow across languages can be difficult

• Debugging can be harder
  - Especially as values flow and control flow from language A to language B

• Build process becomes more complicated

• Developer expertise is required in multiple languages
  - Must understand type safety (etc.) in all languages
How Will We Do It?
In practice, interoperating between F# and C# (or any other CLR language) is relatively straightforward, once the "shape" of the code (what the language turns into at the IL level) in both languages is well understood.
- Ted Neward, Microsoft Developer Network
Worked Examples

- We are going to write a fast C-and-assembly routine for low-level processing.
- Then we will call that C code from
  - Python
  - Java
  - OCaml
- This will involve
  - Data Layout and Run-Time Organizations
  - Translating Familiar Concepts
Native Kernel: One-Time Pad

• One of the building blocks of modern cryptography is the one-time pad.
  - When used correctly it has a number of very desirable properties.

• To encrypt plaintext $P$ with a key $K$ (the one-time pad) you produce cyphertext $C$ as follows:
  - cyphertext$[i] = $plaintext$[i] \ XOR \ keytext[i]$
  - A constant key mask may be also used for testing.

• Decryption also just xors with the key.
Basic Ocaml Implementation

```ocaml
type char_or_string =
    | MyChar of char (* constant bit pattern *)
    | MyString of string (* one-time pad *)

let ocaml_xor_function plain key =
    let cypher = String.create (String.length plain) in
    let match key with
        | MyChar(mask) ->
            for i = 0 to pred (String.length plain) do
                cypher.[i] <- Char.chr
                ((Char.code plain.[i]) lxor (Char.code mask))
            done
        | MyString(keyt) ->
            for i = 0 to pred (String.length plain) do
                cypher.[i] <- Char.chr
                ((Char.code plain.[i]) lxor (Char.code keyt.[i]))
            done
    in cypher
```

#18
Telling Ocaml about C

```ocaml
external
ocaml_name_for_c_xor_function : string -> char_or_string -> string
= "c_string_xor"
```

- We are promising to provide a Native C function called “c_string_xor” that takes a “string”, a “char_or_string”, and returns a “string”.

Native C Implementation

• Basic idea:
  - accept “string” and “char_or_string” as args
  - extract contents of “string” (plaintext)
  - examine “char_or_string”
    • If “char” (mask), extract character code value
    • If “string” (keytext), extract contents of string
  - create a new string (return value, cyphertext)
  - for loop (over length of string)
    • cyphertext = plaintext xor key
  - return cyphertext
The Problem

• int x = 127;
  
  00 00 00 7f
  
• char * p = “hi”;
  
  8d 50 00 62
  
  68 69 00
  
• let cos = MyChar(\127) in
  
  ???
The Problem

• let cos = MyChar('\127') in

```
cos
ff 00 00 00 00 00 00 00 fc 08 00 00 00 00 00 00 ..
```

• let cos2 = MyString(“hi”) in

```
cos2
60 8d 62 00 00 00 00 00 fc 04 00 00 00 00 00 00 ..
```
The Problem

- let cos = MyChar(127) in
- let cos2 = MyString("hi") in

```plaintext
ff 00 00 00 00 00 00 00 fc 08 00 00 00 00 00 00 ..
```

```plaintext
cos2 60 8d 62 00 00 00 00 00 fc 04 00 00 00 00 00 00 ..
```

```plaintext
ff 00 00 00 00 00 00 00 fc 08 00 00 00 00 00 00 ..
```

```plaintext
cos
```

- let cos2 = MyString(60 8d 62)

```plaintext
60 8d 62 00 00 00 00 00 fc 04 00 00 00 00 00 00 ..
```

```plaintext
cos2
```

```plaintext
60 8d 62 00 00 00 00 00 fc 04 00 00 00 00 00 00 ..
```
The Problem

- let \( \text{cos} = \text{MyChar('\127')} \) in
  
  \[
  \text{cos}
  = \begin{array}{c}
  \text{ff 00 00 00 00 00 00 00 fc 08 00 00 00 00 00 00 ..}
  \end{array}
  \]

- let \( \text{cos2} = \text{MyString('hi')} \) in
  
  \[
  \text{cos2}
  = \begin{array}{c}
  \text{60 8d 62 00 00 00 00 00 fc 04 00 00 00 00 00 00 ..}
  \end{array}
  \]

\[
0x628d60
= \begin{array}{c}
68 69 00 00 ..
\end{array}
\]
Run-Time Type Tags

- let cos = MyChar('\127') in
  
  ```
  \00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff 00 00 00 00 00 00 00 fc 08 00 00 00 00 00 00 ..
  ```

- let cos2 = MyString("hi") in
  
  ```
  \01 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 60 8d 62 00 00 00 00 00 00 00 00 fc 04 00 00 00 00 ..
  ```

0x628d60

```
68 69 00 00 ..
```
Run-Time Types

- `let cos = MyChar('\127') in`  
  ```
  00 04 00 00 00 00 00 00 00
  01 04 00 00 00 00 00 00 00
  60 8d 62 00 00 00 00 00 00 fc 04 00 00 00 00 00 00
  fc 04 00 00 00 00 00 00 68 69 00 00 ..
  ```
  - Type Tag 0
  - `C(127) == Ocaml(255)` (garbage collection)
  - Pointer To String (little endian)
  - “Color” (2 bits) and Size (54 bits)
  - Type Tag 252 = String

- `let cos2 = MyString("hi") in`

```
0x628d60
ff 00 00 00 00 00 00 00 fc 08 00 00 00 00 00 00 ..
```

```
0x628d60
```

```
"hi"
```
Medieval History

• This Greek-speaking descendant of the Roman Empire centered around Istanbul (was Constantinople) by conquered much of the Mediterranean coast. Greek fire, mosaics, orthodox Christianity, the crusades, and the Hagia Sophia are all associated with this empire.
Trivia: 1980-1989 Science Fiction
(mra9cg memorial)

• This 1986 James Cameron film is, among other things, the first movie to pass the Bechdel Test (two named female characters talk to each other about something other than a man).

• Name each 1980-89 movie from its brief hint:
  - Spock dies on the Genesis planet.
  - Elliot offers Reese's Pieces.
  - Arnie vs. the invisible alien.
  - Who you gonna call?
  - “I love you.” / “I know.”
Modern Languages

• These mutually-intelligible Central Semitic languages are closely related to Hebric, Phoenician and Aramaic. Used as a liturgical language for 1.6 billion Muslims as well as a natural language for 422M speakers, it features a right-to-left script, open and closed syllables, elided vowels, and a rich literary tradition.

Example: العربية
CAMLprim value c_string_xor(value o_plain, value o_key){
    CAMLparam2 (o_plain, o_key);
    CAMLlocal1 (o_cypher);
    int len = caml_string_length(o_plain);
    int i;
    char * n_plain = String_val(o_plain);
    char * n_cypher;
    o_cypher = caml_alloc_string(len);
    n_cypher = String_val(o_cypher);
    if (Tag_val(o_key) == 0) { /* MyChar:Mask */
        char n_mask = Int_val(Field(v2, 0));
        for (i=0; i<len; i++) n_cypher[i] = n_plain[i] ^ n_mask;
    } else if (Tag_val(o_key) == 1) { /* MyString:Key */
        char * n_keytext = String_val(Field(v2, 0));
        for (i=0; i<len; i++) n_cypher[i] = n_plain[i] ^ n_keytext[i];
    }
    CAMLreturn(o_cypher);
}
If you choose an answer to this question at random, what is the chance you will be correct?

A) 25%  
B) 50%  
C) 60%  
D) 25%
CAMLprim value c_string_xor(value o_plain, value o_key){
    CAMLparam2 (o_plain, o_key);
    CAMLlocal1 (o_cypher);
    int len = caml_string_length(o_plain);
    char* n_plain = String_val(o_plain);
    char* n_cypher = caml_alloc_string(len);
    if (Tag_val(o_key) == 0) { /* MyChar:Mask */
        char n_mask = Int_val(Field(v2, 0));
        for (i=0;i<len;i++) n_cypher[i] = n_plain[i] ^ n_mask;
    } else if (Tag_val(o_key) == 1) { /* MyString:Key */
        char* n_keytext = String_val(Field(v2, 0));
        for (i=0;i<len;i++) n_cypher[i] = n_plain[i] ^ n_keytext[i];
    }
    CAMLreturn(o_cypher);
}

Macro:
This C function will be called from Ocaml.

Typedef:
Opaque Type for Ocaml-managed Data Values
CAMLprim value c_string_xor(value o_plain, value o_key) {
    CAMLparam2 (o_plain, o_key);
    CAMLlocal1 (o_cypher);
    int len = caml_string_length(o_plain);
    int i;
    char * n_plain = String_val(o_plain);
    char * n_cypher;
    o_cypher = caml_alloc_string(len);
    n_cypher = String_val(o_cypher);
    if (Tag_val(o_key) == 0) { /* MyChar:Mask */
        char n_mask = Int_val(Field(v2, 0));
        for (i=0; i<len; i++) n_cypher[i] = n_plain[i] ^ n_mask;
    } else if (Tag_val(o_key) == 1) { /* MyString:Key */
        char * n_keytext = String_val(Field(v2, 0));
        for (i=0; i<len; i++) n_cypher[i] = n_plain[i] ^ n_keytext[i];
    }
    CAMLreturn(o_cypher);
}
Special C File

CAMLprim value c_string_xor(value o_plain, value o_key) {
    CAMLparam2 (o_plain, o_key);
    CAMLlocal (o_cypher);
    int len = caml_string_length(o_plain);
    char * n_plain = String_val(o_plain);
    char * n_cypher;
    o_cypher = caml_alloc_string(len);
    n_cypher = String_val(o_cypher);
    if (Tag_val(o_key) == 0) {
        /* MyChar:Mask */
        char n_mask = Int_val(Field(v2, 0));
        for (i=0;i<len;i++) n_cypher[i] = n_plain[i] ^ n_mask;
    } else if (Tag_val(o_key) == 1) {
        /* MyString:Key */
        char * n_keytext = String_val(Field(v2, 0));
        for (i=0;i<len;i++) n_cypher[i] = n_plain[i] ^ n_keytext[i];
    }
    CAMLreturn (o_cypher);
}
Linking C and OCaml

$ ocamlopt -verbose -o odemo ocaml.ml cocaml.c
+ as -o 'ocaml.o' '/tmp/camlasmb117d1.s'
+ gcc -D_FILE_OFFSET_BITS=64 -D_REENTRANT -c
   -I'/usr/lib/ocaml' 'cocaml.c'
+ as -o '/tmp/camlstartupf4cd24.o'
   '/tmp/camlstartup31ba44.s'
+ gcc -o 'odemo' '-L/usr/lib/ocaml'
   '/tmp/camlstartupf4cd24.o'
   '/usr/lib/ocaml/std_exit.o' 'ocaml.o'
   '/usr/lib/ocaml/stdlib.a' 'cocaml.o'
   '/usr/lib/ocaml/libasmrun.a' -lm -ldl

• Just pass C files on the end of ocamlopt command line.
Linking C and OCaml

```
$ ocamlopt -verbose -o odemo ocaml.ml cocaml.c
+ as -o 'ocaml.o' '/tmp/camlasmb117d1.s'
+ gcc -D_FILE_OFFSET_BITS=64 -D_REENTRANT -c
  -I'/usr/lib/ocaml' 'cocaml.c'
+ as -o '/tmp/camlstartupf4cd24
  '/tmp/camlstartup31ba44.s'
+ gcc -o 'odemo'   '-L/usr/lib/ocaml'
  '/tmp/camlstartupf4cd24.o'
  '/usr/lib/ocaml/std_exit.o' 'ocaml.o'
  '/usr/lib/ocaml/stdlib.a' 'cocaml.o'
  '/usr/lib/ocaml/libasmrun.a' -lm  -ldl
```

- Just pass C files on the end of ocamlopt command line.

OCaml created this ASM from my “ocaml.ml”

OCaml invokes GCC To compile my Special “C” file

OCaml invokes GCC to link all object and library files
def python_string_xor(plain, key):
    cypher = bytearray(' ' * len(plain))
    if type(key) is str:
        for i in range(len(plain)):
            cypher[i] = ord(plain[i]) ^ ord(key[i])
    else:  # is char
        for i in range(len(plain)):
            cypher[i] = ord(plain[i]) ^ key
    return cypher
Interfacing Python with C

```c
static PyObject * cpython_string_xor(PyObject *self, PyObject *args)
{
    const char *n_plain, *n_keytext;
    int plain_size, i, n_mask;
    if (PyArg_ParseTuple(args, "s#s", &n_plain, &plain_size, &n_keytext)) {
        char * n_cypher = malloc(plain_size);
        for (i=0; i<plain_size; i++)
            n_cypher[i] = n_plain[i] ^ n_keytext[i];
        return Py_BuildValue("s#", n_cypher, plain_size);
    } else if (PyArg_ParseTuple(args, "s#i", &n_plain, &plain_size, &n_mask)) {
        char * n_cypher = malloc(plain_size);
        for (i=0; i<plain_size; i++)
            n_cypher[i] = n_plain[i] ^ n_mask;
        return Py_BuildValue("s#", n_cypher, plain_size);
    }
    return NULL;
}
```
Interfacing Python with C

```c
static PyObject *
cython_string_xor(PyObject *self, PyObject *args)
{
    const char *n_plain, *n_keytext;
    int plain_size, i, n_mask;
    if (PyArg_ParseTuple(args, "s#s", &n_plain, &plain_size, &n_keytext)) {
        char * n_cypher = malloc(plain_size);
        for (i=0; i<plain_size; i++)
            n_cypher[i] = n_plain[i] ^ n_keytext[i];
        return Py_BuildValue("s#", n_cypher, plain_size);
    } else if (PyArg_ParseTuple(args, "s#i", &n_plain, &plain_size, &n_mask)) {
        char * n_cypher = malloc(plain_size);
        for (i=0; i<plain_size; i++)
            n_cypher[i] = n_plain[i] ^ n_mask;
        return Py_BuildValue("s#", n_cypher, plain_size);
    }
    return NULL;
}
```

Typedef:
Opaque type for Python-controlled Values.

All functions are "variable argument".

Duck typing:
Can we interpret the arguments as two strings?
static PyObject * cpython_string_xor(PyObject *self, PyObject *args)
{
    const char *n_plain, *n_keytext;
    int plain_size, i, n_mask;
    if (PyArg_ParseTuple(args, "s#s", &n_plain, &plain_size, &n_keytext)) {
        char * n_cypher = malloc(plain_size);
        for (i=0; i<plain_size; i++)
            n_cypher[i] = n_plain[i] ^ n_keytext[i];
        return Py_BuildValue("s#", n_cypher, plain_size);
    } else if (PyArg_ParseTuple(args, "s#i", &n_plain, &plain_size, &n_mask)) {
        char * n_cypher = malloc(plain_size);
        for (i=0; i<plain_size; i++)
            n_cypher[i] = n_plain[i] ^ n_mask;
        return Py_BuildValue("s#", n_cypher, plain_size);
    }
    return NULL;
}
static PyMethodDef CpythonMethods[] = {
    {"string_xor", cpython_string_xor, METH_VARARGS, "XOR a string with a string-or-character"},
    {NULL, NULL, 0, NULL}
};

PyMODINIT_FUNC initcpython( void )
{
    (void) Py_InitModule("cpython", CpythonMethods);
}
Linking Our Native Python Code

- gcc -pthread -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -Wstrict-prototypes -fPIC -I/usr/include/python2.7 -c cpython.c -o build/temp.linux-x86_64-2.7/cpython.o
- gcc -pthread -shared -Wl,-O1 -Wl,-Bsymbolic-functions -Wl,-Bsymbolic-functions -Wl,-z,relro build/temp.linux-x86_64-2.7/cpython.o -o build/lib.linux-x86_64-2.7/cpython.so
Linking Our Native Python Code

• gcc -pthread -g -fwrapv -O2 -Wall -Wstrict-prototypes -fPIC -I/usr/include/python2.7 -c cpython.c -o build/temp.linux-x86_64-2.7/cpython.o

• gcc -pthread -shared -Wl,-O1 -Wl,-Bsymbolic-functions -Wl,-Bsymbolic-functions -Wl,-z,relro build/temp.linux-x86_64-2.7/cpython.o -o build/lib.linux-x86_64-2.7/cpython.so

Position Independent Code (see lecture on Libraries)

Build Shared Library Code (see lecture on Libraries)

.so = .dll = shared library
import cpython  # loads cpython.so

...  

if do_native:
    result = cpython.string_xor(plaintext, \  
        char_or_string_key)
else:
    result = python_string_xor(plaintext, \  
        char_or_string_key)
Programming Paradigms

• This “pass a string or an integer as the second argument” plan ...
  - Works well for Functional (algebraic datatypes)
  - Works well for Dynamic (duck typing)
  - Is not a natural fit for Object-Oriented
    • More natural: dynamic dispatch on “string-or-int”

• abstract class StringOrInt
• class StringOrInt_IsInt extends StringOrInt
• class StringOrInt_IsString extends StringOrInt
abstract class StringOrInt {
    abstract public byte[] java_string_xor (byte[] str1);
}

class StringOrInt_IsInt extends StringOrInt {
    public int my_int;
    public StringOrInt_IsInt (int i) { my_int = i; }
    public byte[] java_string_xor (byte[] plain) {
        byte [] cypher = new byte[plain.length];
        for (int i = 0; i < plain.length; i++)
            cypher[i] = (byte) ((int)plain[i] ^ my_int);
        return cypher;
    }
}
abstract class StringOrInt {
    abstract public byte[] java_string_xor (byte[] str1);
}

class StringOrInt_IsInt extends StringOrInt {
    public int my_int;
    public StringOrInt_IsInt (int i) { my_int = i; }
    public byte[] java_string_xor (byte[] plain) {
        byte[] cypher = new byte[plain.length];
        for (int i = 0; i < plain.length; i++)
            cypher[i] = (byte) ((int)plain[i] ^ my_int);
        return cypher;
    }
}
abstract class StringOrInt {
    abstract public byte[] java_string_xor (byte[] str1);
}

class StringOrInt_IsString extends StringOrInt {
    public byte[] my_string;
    public StringOrInt_IsString (byte[] s) { my_string = s; }
    public byte[] java_string_xor (byte[] plain) {
        byte [] cypher = new byte[plain.length];
        for (int i = 0; i < plain.length; i++)
            cypher[i] = (byte) (plain[i] ^ my_string[i]);
        return cypher;
    }
}
Tell Java about the Native Method

```java
static {
    /* load native library */
    System.loadLibrary("cjava");
}

private static native byte[] c_string_xor(byte[] plain, StringOrInt key);
```
C Code using JNI (1/2)

```c
JNICALL Java_StringXOR_c_string_xor(JNIEnv * env, jclass self, jbyteArray jplain, jobject jkey) {
    jbyte * n_plain = (*env)->GetByteArrayElements(env, jplain, NULL);
    size_t plainsize = (*env)->GetArrayLength(env, j_plain);
    jclass key_cls = (*env)->GetObjectClass(env, jkey);
    jfieldID fid;
    int i;
    jbyteArray jcypher = (*env)->NewByteArray(env, plainsize);
    jbyte * n_cypher = (*env)->GetByteArrayElements(env, jcypher, NULL);

    fid = (*env)->GetFieldID(env, key_cls, "my_int", "I");
    if (fid != NULL) {
        /* key has "int my_int;" field */
        jint n_mask = (*env)->GetIntField(env, jkey, fid);
        for (i=0; i<plainsize; i++) {
            n_cypher[i] = n_plain[i] ^ n_mask;
        }
    } else {
```

#50
>Name: StringXOR

Code:
```c
JNIEXPORT jbyteArray JNICALL Java_StringXOR_c_1string_1xor
(JNIEnv * env, jclass self, jbyteArray jplain, jobject jkey) {
  jbyte * n_plain = (*env)->GetByteArrayElements(env, jplain, NULL);
  size_t plainsize = (*env)->GetArrayLength(env, j_plain);
  jclass key_cls = (*env)->GetObjectClass(env, jkey);
  jfieldID fid;
  int i;
  jbyteArray jcypher = (*env)->NewByteArray(env, plainsize);
  jbyte * n_cypher = (*env)->GetByteArrayElements(env, jcypher, NULL);
  fid = (*env)->GetFieldID(env, key_cls, "my_int", "I");
  if (fid != NULL) {
    /* key has "int my_int;" field */
    jint n_mask = (*env)->GetIntField(env, jkey, fid);
    for (i=0; i<plainsize; i++) {
      n_cypher[i] = n_plain[i] ^ n_mask;
    }
  } else {
  }
```

Macro:
This function is visible to Java.

Typedef:
Opaque types for Java objects.

Java Native Interface environment provides services for Manipulating Java values.

Remember when we said the receiver object was passed as a hidden first 'self' parameter?
C Code using JNI (1/2)

JNIEXPORT jbyteArray JNICALL Java_StringXOR_c_1string_1xor
(JNIEnv * env, jclass self, jbyteArray jplain, jobject jkey)
{
    jbyte * n_plain = (*env)->GetByteArrayElements(env, jplain, NULL);
    size_t plainSize = (*env)->GetArrayLength(env, jplain);
    jclass key_cls = (*env)->GetObjectClass(env, jkey);
    jfieldID fid;
    int i;
    jbyteArray jcypher = (*env)->NewByteArray(env, plainSize);
    jbyte * n_cypher = (*env)->GetByteArrayElements(env, jcypher, NULL);

    fid = (*env)->GetFieldID(env, key_cls, "my_int", "I");
    if (fid != NULL) {
        /* key has "int my_int;" field */
        jint n_mask = (*env)->GetIntField(env, jkey, fid);
        for (i=0; i<plainSize; i++) {
            n_cypher[i] = n_plain[i] ^ n_mask;
        }
    } else {

Function: extract C string from Java byte[]. “Drop tags”, etc.

Function: Extract type tag from Object. Each object is an instance of a class.
C Code using JNI (1/2)

Function:
This is the WHAT from WHAT: is there an int field named “my_int” in this class (or inherited from its parents)? If so, at what position/offset does it live?

JNIEXPORT jbyteArray JNICALL Java_StringXOR_c_1string_1xor(JNIEnv * env, jclass self, jbyteArray jplain, jobject jkey) {
    jbyte * n_plain = (*env)->GetByteArrayElements(env, jplain, NULL);
    size_t plainsize = (*env)->GetArrayLength(env, j_plain);
    jclass key_cls = (*env)->GetObjectClass(env, jkey);
    jfieldID fid;
    jint i;
    jbyteArray jcypher = (*env)->NewByteArray(env, plainsize);
    jbyte * n_cypher = (*env)->GetByteArrayElements(env, jcypher, NULL);
    fid = (*env)->GetFieldID(env, key_cls, "my_int", "I");
    if (fid != NULL) {
        /* key has "int my_int;" field */
        jint n_mask = (*env)->GetIntField(env, jkey, fid);
        for (i=0; i<plainsize; i++) {
            n_cypher[i] = n_plain[i] ^ n_mask;
        }
    } else {

C Code using JNI (1/2)

JNIEXPORT jbyteArray JNICALL Java_StringXOR_c_1string_1xor
(JNIEnv * env, jclass self, jbyteArray jplain, jobject jkey)
{
    jbyte * n_plain = (*env)->GetByteArrayElements(env, jplain, NULL);
    size_t plainsize = (*env)->GetArrayLength(env, jplain);
    jclass key_cls = (*env)->GetObjectClass(env, jkey);
    jfieldID fid;

    fid = (*env)->GetFieldID(env, key_cls, "my_int", "I");
    if (fid != NULL) {
        /* key has "int my_int;" field */
        jint n_mask = (*env)->GetIntField(env, jkey, fid);
        for (i=0;i<plainsize;i++) {
            n_cypher[i] = n_plain[i] ^ n_mask;
        }
    } else {

else {
    fid = (*env)->GetFieldID(env, key_cls, "my_string", "[B");
    if (fid != NULL) {
        /* key has "byte[] my_string;" field */
        jbyteArray jkeyt = (*env)->GetObjectField(env, jkey, fid);
        jbyte * n_keytext = (*env)->GetByteArrayElements
        (env, jkeyt, NULL);
        for (i=0;i<plainsize;i++)
            cypher[i] = n_plain[i] ^ n_keytext[i];
        (*env)->ReleaseByteArrayElements(env, jkeyt, n_keytext, 0);
    }
}
(*env)->ReleaseByteArrayElements(env, jplain, n_plain, 0);
(*env)->ReleaseByteArrayElements(env, jcypher, n_cypher, 0);
return jcypher;
else {
    fid = (*env)->GetFieldID(env, key_cls, "my_string", "[B");
    if (fid != NULL) {
        /* key has "byte[] my_string;" field */
        jbyteArray jkeyt = (*env)->GetObjectField(env, jkey, fid);
        jbyte * n_keytext = (*env)->GetByteArrayElements(env, jkeyt, NULL);
        for (i = 0; i < plainsize; i++)
            cypher[i] = n_plain[i] ^ n_keytext[i];
        (*env)->ReleaseByteArrayElements(env, jkeyt, n_keytext, 0);
    }
}
(*env)->ReleaseByteArrayElements(env, jplain, n_plain, 0);
(*env)->ReleaseByteArrayElements(env, jcypher, n_cypher, 0);
return jcypher;

Playing nice with the garbage collector.

Can indicate whether elements were copied or shared.
Compiling, Linking and Running JNI

gcc -I $(JAVA)/include \n    -o libcjava.so -shared -fPIC cjava.c
javac StringXOR.java
java -Djava.library.path=. StringXOR

• That's it!
• “javap” also exists to automatically generate header files for C JNI implementations.
Actual Numbers

(20 trials, best wall-clock ms time reported)

Ocaml - Ocaml 143
Ocaml - Native 103
Python - Python 598
Python - Native 29
Java - Java 165
Java - Native 183
C 22
<table>
<thead>
<tr>
<th>Language</th>
<th>Environment</th>
<th>Wall-Clock Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ocaml</td>
<td>Native</td>
<td>103</td>
</tr>
<tr>
<td>Ocaml</td>
<td>Ocaml</td>
<td>143</td>
</tr>
<tr>
<td>Python</td>
<td>Native</td>
<td>29</td>
</tr>
<tr>
<td>Python</td>
<td>Python</td>
<td>598</td>
</tr>
<tr>
<td>Java</td>
<td>Native</td>
<td>183</td>
</tr>
<tr>
<td>Java</td>
<td>Java</td>
<td>165</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>22</td>
</tr>
</tbody>
</table>
# Actual Numbers (You Explain)

(20 trials, best wall-clock ms time reported)

<table>
<thead>
<tr>
<th>Language 1</th>
<th>Language 2</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ocaml</td>
<td>Ocaml</td>
<td>143</td>
</tr>
<tr>
<td>Ocaml</td>
<td>Native</td>
<td>103</td>
</tr>
<tr>
<td>Python</td>
<td>Python</td>
<td>598</td>
</tr>
<tr>
<td>Python</td>
<td>Native</td>
<td>29</td>
</tr>
<tr>
<td>Java</td>
<td>Java</td>
<td>165</td>
</tr>
<tr>
<td>Java</td>
<td>Native</td>
<td>183</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>22</td>
</tr>
</tbody>
</table>
Ocaml Native Interface
Problem Solving Example

• Input:
  - 4b50 0403 0014 0000 0008 59b7 42cd 0ed7

• Expected Output, XOR with \127:
  - 342f 7b7c 7f6b 7f7f 7f77 26c8 3db2 71a8

• Actual Output, Deterministic:
  - b4af fbfc ffeb ffff ffff7 a648 bd32 f128

• What's the bug?
Ocaml Native Interface

Problem Solving Example

- Input:
  - 4b50 0403 0014 0000 0008 59b7 42cd 0ed7

- Expected Output, XOR with '127':
  - 342f 7b7c 7f6b 7f7f 7f77 26c8 3db2 71a8

- Actual Output, Deterministic:
  - b4af fbfc ffeb ffff fff7 a648 bd32 f128

- What’s the bug?

\[
\begin{align*}
\mathcal{E}_y &= \mathcal{D}_x \phi \mathcal{V} \mathcal{T} \overline{A}_{xy} \\
\mathcal{H}_y &= \mathcal{D}_x \phi \mathcal{V} \mathcal{T} \overline{B}_{xy}
\end{align*}
\]

“An expert is a person who has made all the mistakes that can be made in a very narrow field.”
— Niels Bohr
Native Interface
Problem Solving Example

• Input:
  - 4b50 0403 0014 0000 0008 59b7 42cd 0ed7

• Expected Output, XOR with \127':
  - 342f 7b7c 7f6b 7f7f 7f7f 7f7f 26c8 3db2 71a8

• Actual Output, Deterministic:
  - 342f 7b7c 7f6b

• What's the bug?
Homework

- PA4 Due Today
- PA5t Due Tue March 29 (7 days)