

# 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.**

pts in  
high-level  
program  
particular, you  
will understand **theory and practice of lexing, parsing, semantic analysis, and code interpretation.** You will also have gained practical experience programming in multiple **different languages.**

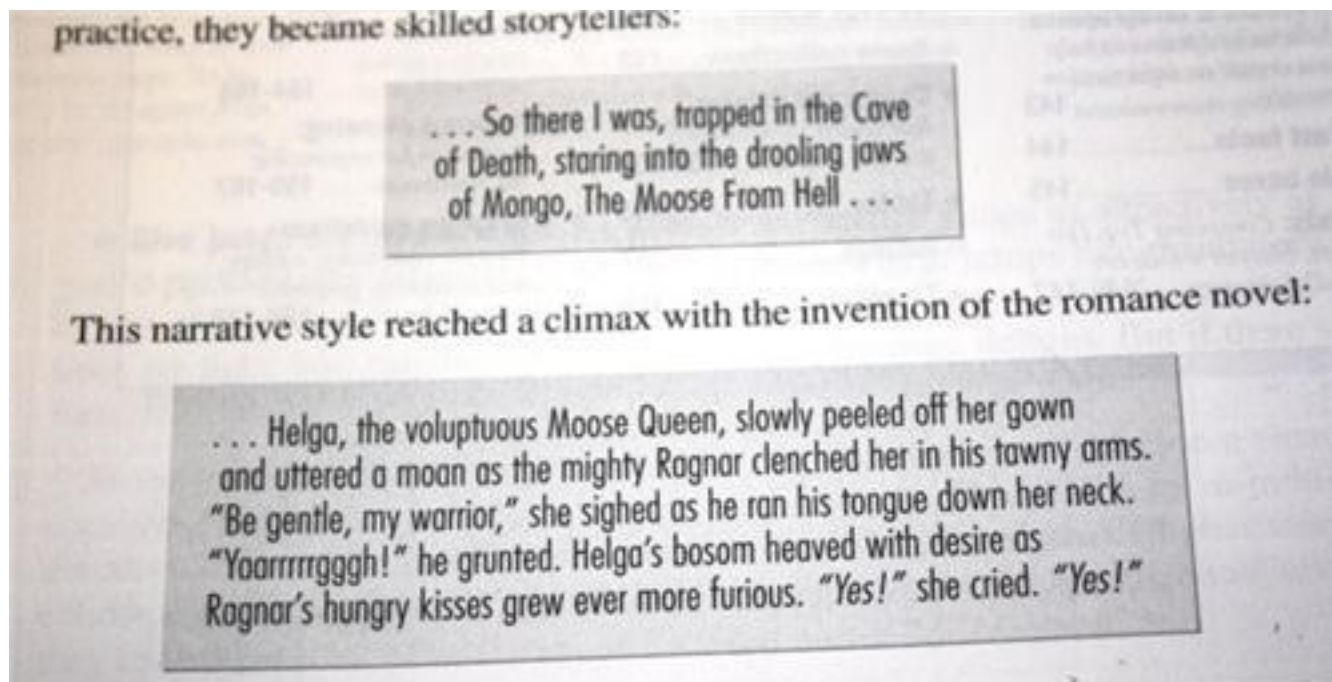
# Lecture Outline

- Motivating Example
  - XOR (String Cryptography)
- Ocaml + C
  - Object Layout, Type Tags
  - Interfacing
- Python + C
  - Interfacing
- Java + C
  - Interfacing



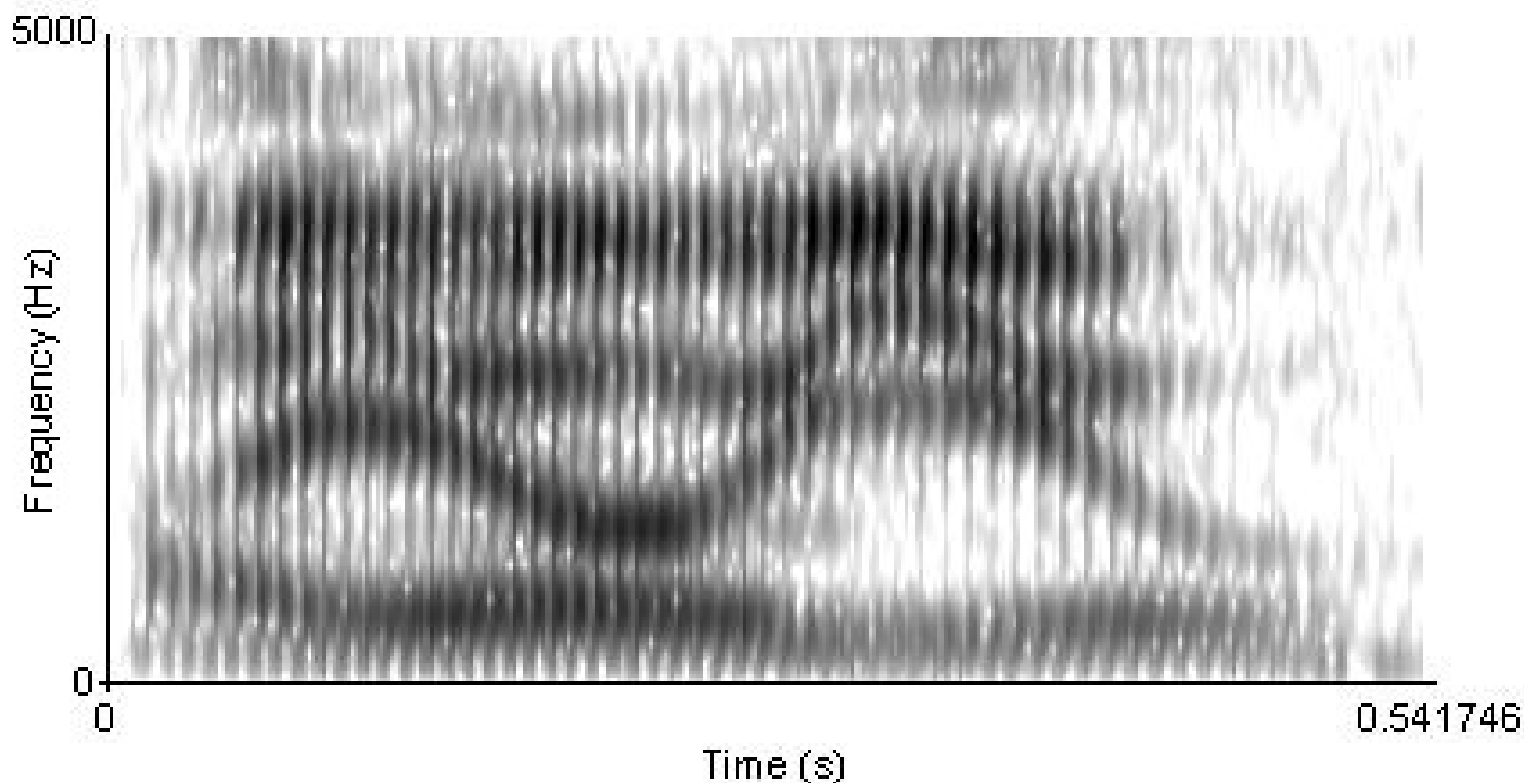
# Motivating Example

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



# 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 a world

splendid

colorful

magical

va sempre in armonia

In un mondo splendido, colorato e magico

Little ponies vivono, in pace sempre in armonia

Tutti i pony hanno caratteri diversi: vivaci, burrascosi e romantici.

Sono i caratteri, degli amici che ti aiuteranno.

Ed ogni giorno, con i tuoi amici, tutti i problemi risolverai.

Insieme agiterai la tua vita, e 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!

vivacious = living

semper fi =  
always

harmony

Requiescat in pace = RIP  
peace



timid

sympathetic

brusque

romantic

Volare e andare in volo, in pace sempre in armonia

Timidi e simpatici, burberi e romantici

Sono i caratteri, degli amici che troverai

characters

treasure trove =  
found

amicable =  
friends

Ma non crescerai quanti ne troverai  
Insieme agli altri non lo sai

Volare e vai, nuovi amici vorrai incontrare

Prendi il volo, ascolta il cuore, ed ogni avventura potrai affrontare!

Volare 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:
  - $\text{cyphertext}[i] = \text{plaintext}[i] \text{ XOR } \text{keytext}[i]$
  - A constant key **mask** may be also used for testing.
- Decryption also just xors with the key.

# Basic Ocaml Implementation

```
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
  ( 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
  ) ; cypher
```

# Telling Ocaml about C

```
external
```

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

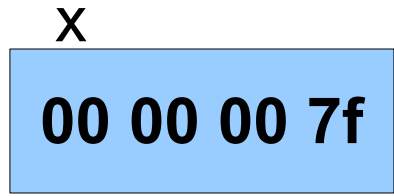
- We are promising to provide a Native C function called “c\_string\_xor” that takes a “string”, a “char\_or\_string”, and returns a “string”.

# Native C Implementation

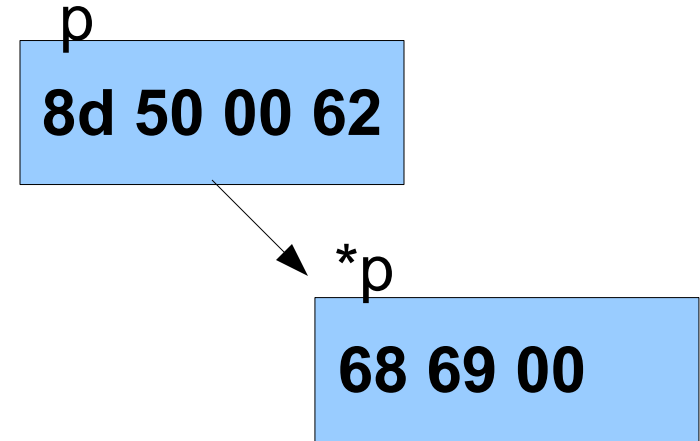
- Basic idea:
  - accept “string” and “char\_or\_string” as args
  - extract contents of “string” (plaintext)
  - examine “char\_or\_string”
    - If “char” (mask), extract character code value
    - If “string” (keytext), extract contents of string
  - create a new string (return value, cyphertext)
  - for loop - over length of string
    - $\text{cyphertext} = \text{plaintext} \text{ xor } \text{key}$
  - return cyphertext

# The Problem

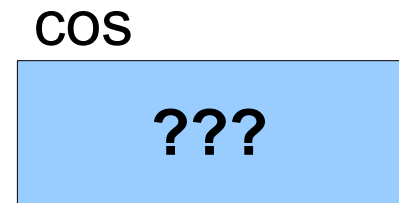
- `int x = 127;`



- `char * p = "hi";`



- `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 ..
```

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

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



- let cos = MyCh

COS

ff 00 00 00 0

- let cos2 = MyS

cos2

60 8d 62 00 0



# 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 ..
```

0x628d60

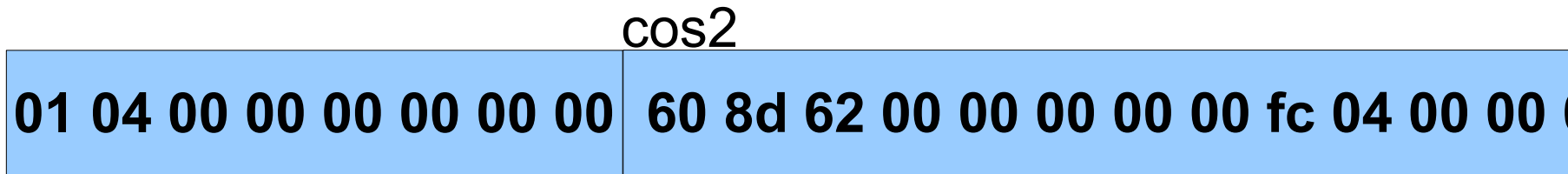
```
68 69 00 00 ..
```

# Run-Time Type Tags

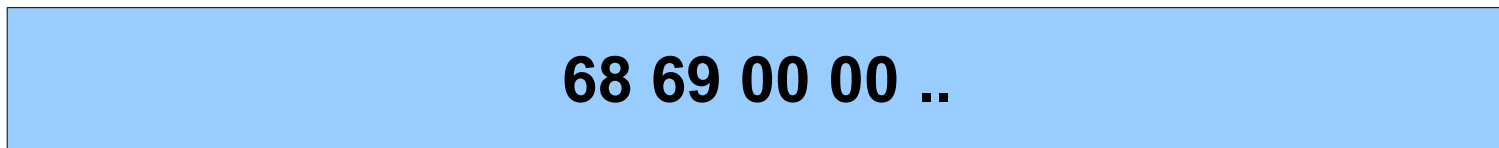
- let `cos = MyChar('\127')` in



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



0x628d60



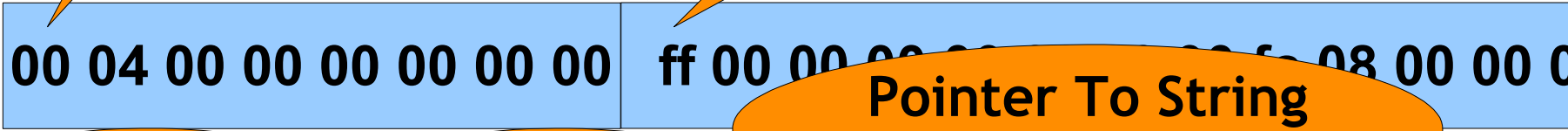
# Run-Time Type

Type Tag 0

C(127) == Ocaml(255)  
(garbage collection)

```
let cos = MyChar('\127') in
```

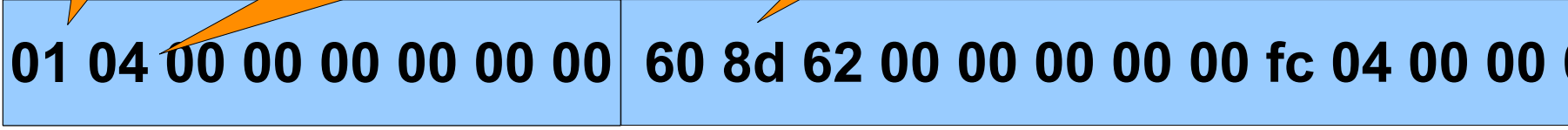
COS



Pointer To String  
(little endian)

```
let cos2 = MyChar('\127') in
```

COS2



“Color” (2 bits)  
and Size (54 bits)

0x628d60



Type Tag 252 = String

“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.

# Modern Languages

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

Example: العربية

# Special C File

```
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

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%

# Special C File

```
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);
  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);
}
```

**Macro:**  
This C function will be called from Ocaml.

**Typedef:**  
Opaque Type for Ocaml-managed Data Values

# Special C File

```
CAMLprim value c_string_xor(value  
  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);  
}
```

**Macros:**  
Play nice with Ocaml's garbage collector.

**Functions:**  
Extract C-string From Ocaml-string (drop header)

**Functions:**  
Make Ocaml-string (create header)

# Special C File

```
CAMLprim value c_string_xor(value o_plain, value o_key) {
  CAMLparam2 (o_plain, o_key);
  CAMLlocal1 (n_cypher);
  int len = Int_val(Field(v1, 0));
  char * n_plain = String_val(o_plain);
  char * n_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) { /* MyChar:Key */
    char * n_keytext = String_val(o_key);
    for (i=0; i<len; i++) n_cypher[i] = n_plain[i]^n_keytext[i];
  }
  CAMLreturn (o_cypher);
}
```

Macros, Functions:  
Check Type Tag  
(from Ocaml Header)

Macros, Functions:  
Extract Fields of  
Ocaml Tuple (Block)

Macros:  
Convert Ocaml-Int  
To C-Int  
(bit shift/mask)

# Linking C and OCaml

```
$ ocamlc -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 ocamlc command line.

# Linking C and OCaml

Ocaml  
created this ASM from  
my "ocaml.ml"

```
$ ocamlc -o ocaml.o ocaml.ml
+ 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
```

Ocaml invokes GCC  
To compile my  
Special "C" file

- Just pass C files on the end of o

Ocaml invokes GCC  
to link all  
object and library  
files

# XOR In Python

```
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

```
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

**Typedef:  
Opaque type for  
Python-controlled  
Values.**

**All functions are  
“variable argument”.**

**Duck typing:  
Can we interpret  
The arguments as two strings?**

```
static PyObject * encrypt(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

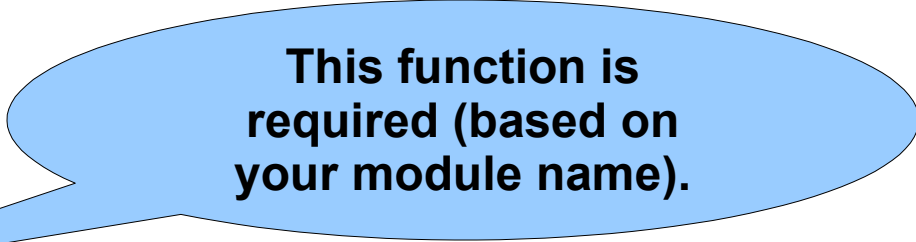
```
static PyObject * cpython_string_xor(PyObject *self, PyObject *args)
{
    const char *n_plain, *n_keytext;
    int plain_size, i, n_mask;
    if (PyArg_ParseTuple(args, "s#s", &n_plain, &plain_size, &n_keytext)) {
        char * n_cypher = malloc(plain_size);
        for (i=0;i<plain_size;i++)
            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;
}
```

**Function:**  
Build a Python String  
from a C string.

**Duck Typing:**  
Can we interpret the  
arguments as a string  
followed by an int?

# Interfacing Python with C, cont'd

```
static PyMethodDef CpythonMethods[] = {  
    {"string_xor", cpython_string_xor, METH_VARARGS,  
     "XOR a string with a string-or-character"},  
    {NULL, NULL, 0, NULL}  
};
```



This function is required (based on your module name).

```
PyMODINIT_FUNC initspython(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 -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`

Position Independent Code  
(see lecture on Libraries)
- `gcc -pthread -shared -Wl,-O1 -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`

Build Shared Library Code  
(see lecture on Libraries)

`.so = .dll = shared library`

# Interfacing C with Python

```
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

# Java Code (1 / 2)

```
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;  
    }  
}
```



# Java Code (1/2)

Java's String is so tied up in encodings That it's not raw-content-preserving.

```
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;  
    }  
}
```

Cutely, Java warns about a lack of precision here (int/byte) unless you cast.

# Java Code (2/2)

```
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

```
static {
```

```
    /* load native library */
```

```
    System.loadLibrary("cjava");
```

```
}
```

```
private static native byte[]
```

```
    c_string_xor(byte[] plain, StringOrInt key);
```

# 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, 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.**

```
JNIEXPORT jbyteArray JNICALL Java_StringXOR_c_1string_1xor
(JNIEnv * env, jclass self, jbyteArray jplain, jobject jkey)
{
    jbyte * n_plain = (*env)->GetByteArrayElements(jplain, NULL);
    jint n_plain_size = (*env)->GetIntField(env, self, "n");
    jint n_key_size = (*env)->GetIntField(env, jkey, "n");
    jint n_cypher_size = (*env)->GetIntField(env, self, "n");
    jbyteArray jcypher = (*env)->NewByteArray(env, n_cypher_size);
    jbyte * n_cypher = (*env)->GetByteArrayElements(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<n_plain_size; i++) {
            n_cypher[i] = n_plain[i] ^ n_mask;
        }
    } else {
```

**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 for JNI (1/2)

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

```
JNIEXPORT jbyteArray JNICALL Java_com_example_Main_c1string_1xor  
(JNIEnv * env, jobject 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 {
```

Function:  
Extract type tag from  
Object. Each object  
is an instance of a class.

# 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);
    jint n_mask = (*env)->GetIntField(env, jkey);
    for (i=0; i<plainsize; i++) {
        n_cypher[i] = n_plain[i] ^ n_mask;
    }
} else {
```

**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?**

# 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);
    jint n_mask = (*env)->GetIntField(env, jkey);
    for (i=0; i<n_plain->length; i++) {
        n_cypher[i] = n_plain[i] ^ n_mask;
    }
} else {
```

**Function:**  
**This is the CLASS MAP from PA4.**  
**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?**

```
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<n_plain->length; i++) {
        n_cypher[i] = n_plain[i] ^ n_mask;
    }
} else {
```



# C Code using JNI (2/2)

```
else {
    fid = (*env)->GetFieldID(env, key_cls, "my_string", "[B"]);
    if (fid != NULL) {
        /* key has "byte[] my_string;" field */
        jbyteArray jkey = (*env)->GetObjectField(env, jkey, fid);
        jbyte * n_keytext = (*env)->GetByteArrayElements
                               (env, jkey, NULL);

        for (i=0; i<plainsize; i++)
            cypher[i] = n_plain[i] ^ n_keytext[i];
        (*env)->ReleaseByteArrayElements(env, jkey, n_keytext, 0);
    }
}

(*env)->ReleaseByteArrayElements(env, jplain, n_plain, 0);
(*env)->ReleaseByteArrayElements(env, jcypher, n_cypher, 0);
return jcypher;
```

# C Code using JNI (2/2)

```
else {
    fid = (*env)->GetFieldID(env, key_cls, "my_string", "[B");
    if (fid != NULL) {
        /* key has "byte[] my_str
        jclass jkey = (*env)->GetObjectField(env, jkey, fid);
        jbyte * n_keytext = (*env)->GetByteArrayElements
                                (env, jkey, NULL);
        for (int i = 0; i < n_keytext; i++)
            (*env)->ReleaseByteArrayElements(env, n_keytext[i], 0);
    }
}

(*env)->ReleaseByteArrayElements(env, jplain, n_plain, 0);
(*env)->ReleaseByteArrayElements(env, jcypher, n_cypher, 0);
return jcypher;
}
```

CLASS MAP again.  
"[B" == "[Byte"

Can indicate whether  
elements were copied or shared.

Playing nice with  
the garbage collector.

# Compiling, Linking and Running JNI

```
gcc -I $(JAVA)/include \  
    -o libcjvava.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

# Actual Numbers

(  
C  
C  
C  
P  
P  
J  
J  
C



# Actual Numbers (You Explain)

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

Ocaml - Ocaml	143	
Ocaml - Native	103	} What?
Python - Python	598	
Python - Native	29	
Java - Java	165	} What?
Java - Native	183	
C	22	

# Ocaml Native Interface Problem Solving Example

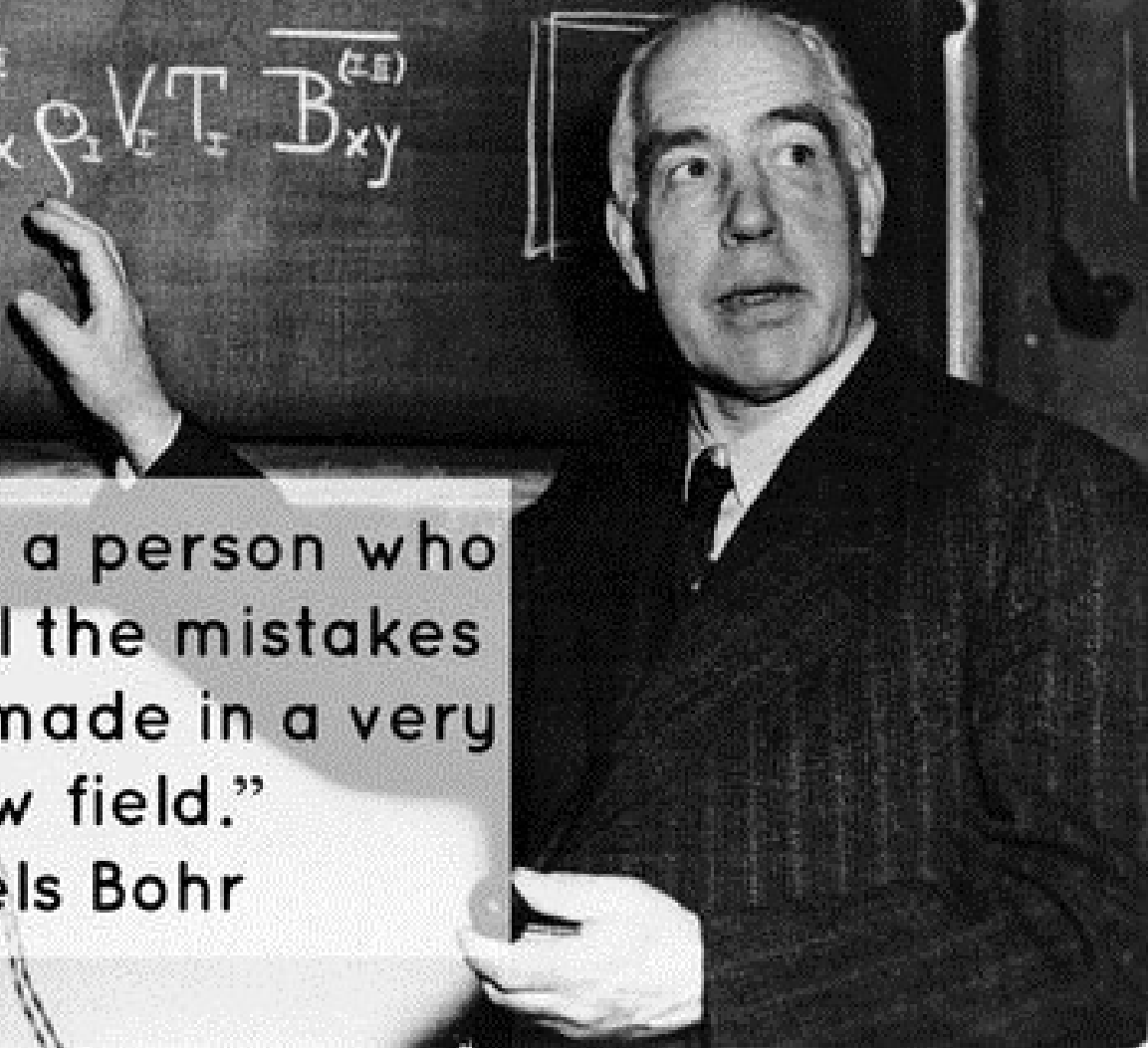
- Input:
  - 4b50 0403 0014 0000 0008 59b7 42cd 0ed7
- Expected Output, XOR with '\127':
  - 342f 7b7c 7f6b 7f7f 7f77 26c8 3db2 71a8
- Actual Output, Deterministic:
  - b4af fbfc ffeb ffff fff7 a648 bd32 f128
- What's the bug?

# General Nonsingular Interface

$$\overline{G}_y^{(I)} = \overline{D}_x^I \overline{\rho}_I \overline{V}_I \overline{T}_I \overline{A}_{xy}^{(IG)}$$

$$\overline{H}_y^{(II)} = \overline{D}_x^I \overline{\rho}_I \overline{V}_I \overline{T}_I \overline{B}_{xy}^{(IE)}$$

- I
- E
- A
- V



“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 7f77 26c8 3db2 71a8
- Actual Output, Deterministic:
  - 342f 7b7c 7f6b
- What's the bug?

# Homework

- WA5t due in 7 days
- PA5t due in 7 days
- Make-Up Lecture on Sunday, March 30<sup>th</sup>