

# Multi-Language Projects

Prof. Kochunas  
EECS 481 (W23)



# The Story so far...

- We are at the end of the semester and I hope you learned something.

# 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.
- Almost **all aspects of software engineering** are impacted in multi-language projects.

# Outline

- Motivating Example
  - XOR (String Cryptography)
- Python + C
- Java + C
  - Interfacing
- SE Implications
- (Bonus Material:
  - Ocaml/F# + C
  - Object Layout, Type Tags)



**Learning Objectives: by the end of today's lecture you should be able to...**

1. (*knowledge*) all you need is a C interface

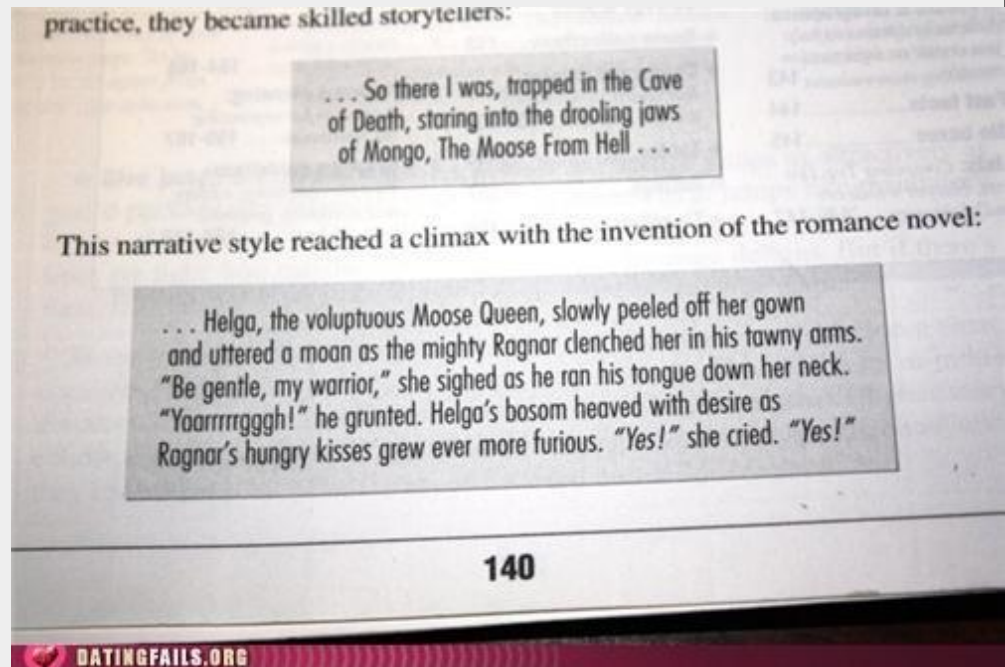


# Motivation



# Motivating Example

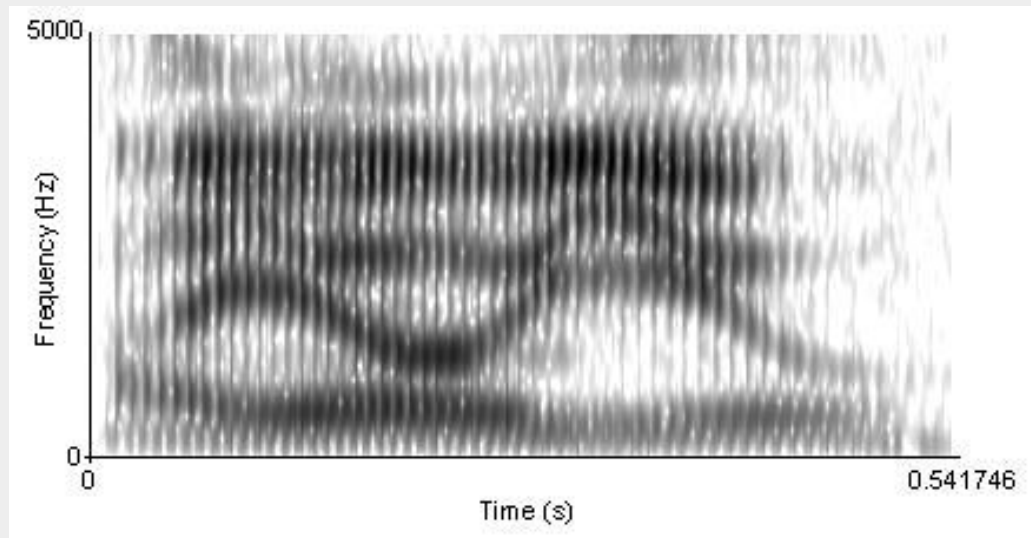
- Use a participation notecard (pairs or solo)
- First: record every word you heard
  - This will be hard
- Second: translate





# Speech Perception, Segmentation

- The spectrogram is for the phrase “I owe you”
  - cf. “Raw Data Layout”
  - Note: no obvious boundaries (cf. neural net)





# 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

# Motivating Example

In un mondo splendido, colorato e magico

Little ponies vivono, in pace sempre in armonia

Timidi e simpatici, burberi e romantici

vivacious = living

semper fi =  
always

harmony

La tua vita crescerà, quanti problemi risolverai

Insieme agli amici ti divertirai!

Requiescat in pace = RIP  
peace

Vola e vai, my little pony, i tuoi 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

timid

sympathetic

brusque

romantic

Little ponies vivono, in pace sempre in ar

Timidi e simpatici, burberi e romantici

Sono i caratteri, degli amici che troverai

Ed ogni giorno crescerai quanti problemi risolvi

Ma con altri pony, lo sai, ti divertirai!

characters

treasure trove =  
found

amicable =  
friends

Vola e vai, my little pony, troverai incontrare

Prendi il volo, ascolta il vento, un'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), etc.

# Entertainment Example

- The Retro64 Mod for Minecraft combines Super Mario 64 with Minecraft
- It uses JNI and DLL files to connect “C” code associated with Super Mario 64 and Java code from Minecraft



# 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.
- Others: LLVM Bitcode, Python Bytecode, etc.



# Why Cover “Multi-Language”?

- Increasingly **common**. 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.)”
- SE process: choose the **best tool** for the job
  - Example: concurrency might be better handled in F#/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, etc.

# 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 types (etc.) in all languages

# How Will We Do It?

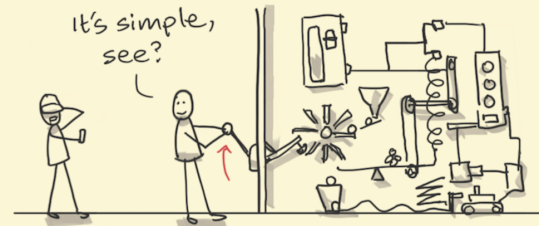
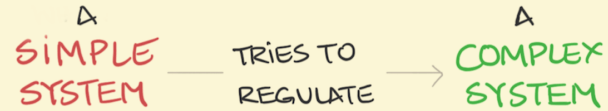
- “In practice, interoperating between F# and C# (or any other CLR language) is relatively straightforward, once the 'shape' of the code (what the language turns into at the IL level) in both languages is well understood.”
  - Ted Neward, Microsoft Developer Network



# Python + C Example

## LAW OF UNINTENDED CONSEQUENCES

OFTEN SEEN WHEN :



ACTION → ?

sketchplanations

# Worked Examples

- We are going to write a fast C-and-assembly routine for low-level processing
  - Assume you know C or C++ (e.g., libpng, afl, etc.)
- Then we will call that C code from
  - Python (e.g., avl.py, mutate.py, delta.py)
  - Java (e.g., JFreeChart, JSoup, EvoSuite)
  - (Bonus Material: OCaml/F# (e.g., Infer))
- This will involve
  - Understanding Data
  - Translating Familiar Concepts

# Python Data Representation

- In our Dynamic Analysis lecture, we saw that Python supports **Dynamic Mocking**
  - replacing one method with another at runtime
- How does Python **represent data**?
- Contrast to C++

## Dynamic Mocking Example

```
import unittest
from unittest import mock

def lowLevelOp():
    # might fail for users
    # example: no memory
    pass

def highLevelTask():
    try:
        lowLevelOp()
        return True
    except MemoryError:
        return False
```

```
class HLTTestCase(unittest.TestCase):
    def test_LLO_no_memory(self):
        def mocked_memory_error():
            raise MemoryError('test :-(')

        with mock.patch( # look here!
            '__main__.lowLevelOp',
            mocked_memory_error ):
            self.assertFalse(highLevelTask())

if __name__ == '__main__':
    unittest.main()
```

# Python Object Implementation

- In Python, everything is an object
  - Type, reference count, data values
- Eliding some details:

```
struct PyObject {  
    size_t          reference_count;  
    PyTypeObject * python_type;  
    size_t          data_size;  
    void []         data;  
}
```



# Size “Paradox”

- In this example, two tuples of numbers are constructed

```
>>> tup1 = (1,2,3)
>>> tup2 = (4,5,6)
>>> tup3 = (tup1,tup2)
>>> sys.getsizeof(tup1)
64
>>> sys.getsizeof(tup2)
64
>>> sys.getsizeof(tup3)
56
```

- How can  $((1,2,3),(4,5,6))$  be *smaller than*  $(1,2,3)$  ?

# Objects as Dictionaries

- In Python, most objects are just dictionaries
  - Mappings from “field names” → “field values”
  - Mappings from “method names” → “func. Ptrs”

```
>>> class CustomClass:
...     def __init__(self, x, y):
...         self.x = x
...         self.y = y
...     def get_sum(self):
...         return self.x + self.y
...
>>> my_custom_object = CustomClass(1234,5678)
>>> my_custom_object.get_sum()
6912
```

```
>>> sys.getsizeof(my_custom_object)
48
>>> vars(my_custom_object)
{'x': 1234, 'y': 5678}
>>> dir(my_custom_object)
['__class__', '__delattr__', '__dict__', '__dir__',
'__gt__', '__hash__', '__init__', '__init_subclass__',
'__reduce__', '__reduce_ex__', '__repr__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__', 'get_sum', 'x', 'y']
```

- C++ and Java are “functionally” similar, but use optimizations (e.g., static orderings for fields and methods rather than dynamic hash tables), see EECS 483 Compilers

# Dynamic Type Checking

- We can use the “type” information in a PyObject at run-time
- Because most objects are just dynamic collections of fields ...
- We may need type details
  - e.g., `(int, str) != (str, str)`
    - Even if both are tuples

```
>>> x = (123, "hello")
>>> y = (123, True)
>>> isinstance(x, tuple)
True
>>> isinstance(x[0], int)
True
>>> isinstance(x[1], str)
True
>>> isinstance(y, tuple)
True
>>> isinstance(y[0], int)
True
>>> isinstance(y[1], str)
False
```

# 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.

# 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
```

# Idealized C

- Accept as input the plaintext and keytext
  - Since this is C, use “char \*” for strings
  - Also: since this is C, pass in the length :-(
  - Also also: since this is C, allocate space for the encrypted cyphertext we return

```
char * encrypt_xor(char * plaintext, char * keytext, int size) {  
    char * cyphertext = malloc(size);  
    for (int i=0; i<size; i++)  
        cyphertext[i] = plaintext[i] ^ keytext[i];  
    return cyphertext;  
}
```

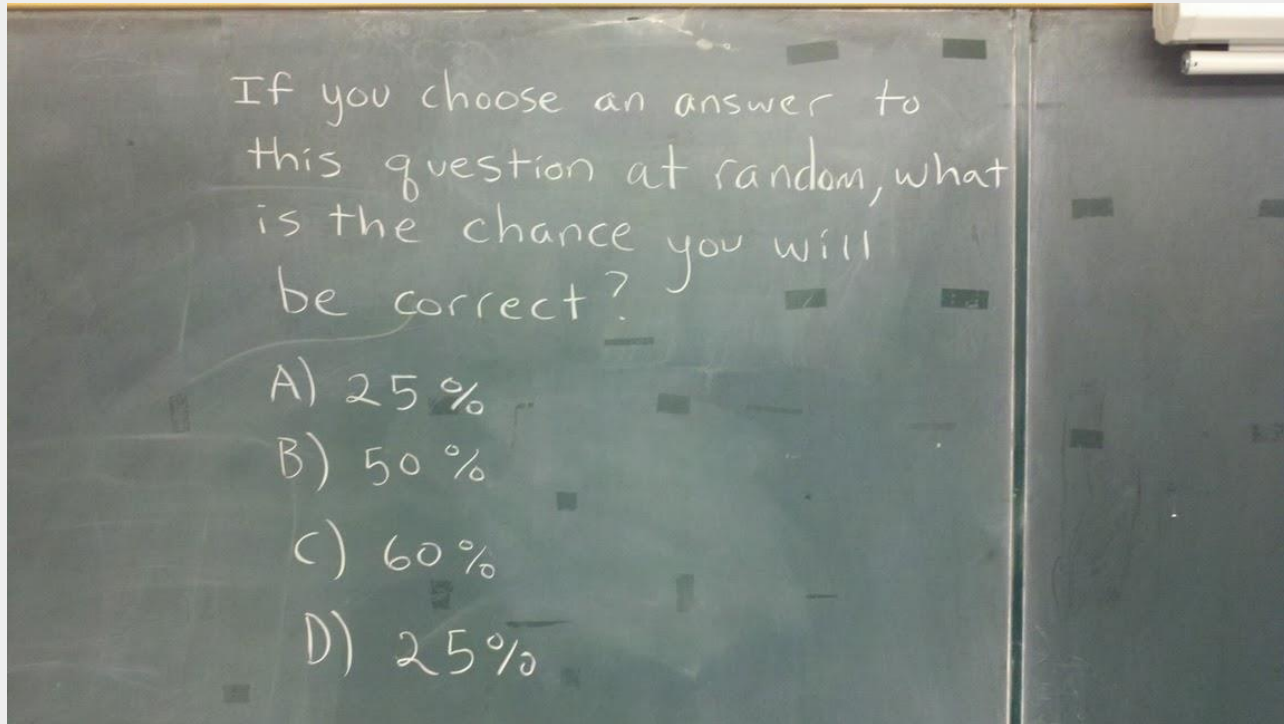
- Now let's interface that with Python!

# 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;
}
```



# “Readability”



# The Heart of the Matter

```
char * n_cypher = malloc(plain_size);  
for (i=0;i<plain_size;i++)  
    n_cypher[i] = n_plain[i] ^ n_keytext[i];
```

```
}
```

# 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;
}
```

# Interface with C

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

```
static PyObject * cpy... (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;
}
```

**All functions are  
"variable argument".**

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

# 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 initemptypython(void)  
{  
    (void) Py_InitModule("cpython", CpythonMethods);  
}
```

# Interfacing Python3 with C, cont'd

```
static PyObject* CPythonMethods[] = {  
    PyString_FromString("xor"), METH_VARARGS,  
    PyString_FromString("or-character"),  
};
```

Python3 requires definition of module as struct

Same as Python2 before

```
static struct PyModuleDef cpython = {  
    PyModuleDef_HEAD_INIT, "cpython", "", -1, CpythonMethods  
};
```

```
PyMODINIT_FUNC PyInit_cpython(void)  
{  
    return PyModule_Create(&cpython);  
}
```

Python3 requires different declaration of init function



# 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,-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

- gcc -pthread -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -Wstrict-prototypes -fPIC I/usr/include/python2.7 build/temp.linux-x86\_64-2.7/cpython.o -o build/lib.linux-x86\_64-2.7/cpython.so
- 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

**Position Independent Code**  
(see EECS 483)

**Build Shared Library Code**  
(see EECS 483)

**.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)
```

# Trivia Break

**AND NOW FOR SOMETHING  
COMPLETELY DIFFERENT...**



# Trivia: Medieval History

- This Greek-speaking descendant of the Roman Empire centered around Istanbul (was Constantinople) and conquered much of the Mediterranean coast. Greek fire, mosaics, orthodox Christianity, the crusades, and the Hagia Sophia are all associated with this empire.

# Politics of India

- The first female Prime Minister of India was known for her centralization of power, the Indo-Pakistani war of 1971, the conflict over the creation of Bangladesh, authorizing the development of nuclear weapons, and a suspension of civil liberties. She stood firm in the face of American pressure, but is also associated with a culture of nepotism.

# Trivia: Zoology

- These ray-finned fish hatch in fresh water, migrate to the ocean, and then return to fresh water to reproduce. Tracking studies have shown that they often return to the same spot they hatched from to spawn. Commercial production of them is currently over three million tonnes. They are often a keystone species, supporting bears, birds and otters.



# Modern Languages

- This Central Semitic language is closely related to Hebrew, Phoenician and Aramaic. Used as a liturgical language for ~1.8 billion Muslims as well as a native language for ~300M speakers, it features a right-to-left script, open and closed syllables, elided vowels, and a rich literary tradition.
  - العَرَبِيَّة : Example



# Psychology: Memory?

- 54 students and 108 community members were posed questions like:

Imagine that you are single and do not have the opportunity to meet many other single people. A friend of yours would like to set you up on a blind date. She has two people in mind that she would like to set you up with. However, those two people are friends with each other and your friend doesn't want to cause problems between them. Thus, she says you should pick just one that you would be interested in dating. She gives you a description of each of them. Who would you choose for a blind date?<sup>1</sup>

- Days later, they were given a memory task related to features in the questions (e.g., was it a “red brick house”, a “white house built of wood”, or “neither”).

# Psychology: Value Judgement

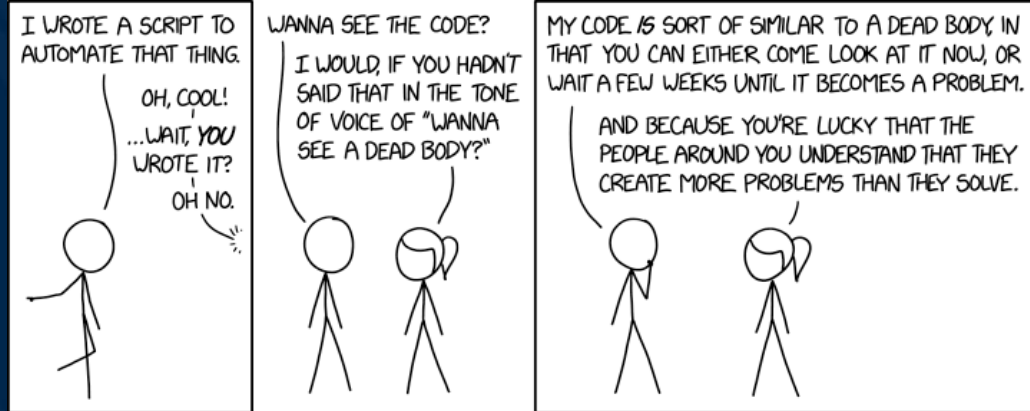
- Finally, they were asked to rate how positive or negative the feature would be in the context of making the decision

Red brick house	White house built of wood
<p>More expensive than you would like</p> <p>Beautiful architectural details in the house</p> <p>Cathedral ceilings</p> <p>Large living room</p> <p>Basement leaks</p> <p>Within walking distance to stores</p> <p>Driveway is shared with neighbors</p> <p>Many neighbors have children</p> <p>Newly renovated and fully equipped kitchen</p> <p>Floor visibly uneven in some places</p> <p>Cracks in the walls</p>	<p>Asking price is within your range</p> <p>Smaller than you would like</p> <p>Lots of sunlight</p> <p>Poor insulation</p> <p>Beautifully landscaped yard</p> <p>Safe neighborhood</p> <p>Has a roach problem</p> <p>Has an old oil furnace</p> <p>Water stains on the ceiling on the top floor</p> <p>Some shingles missing from the roof</p> <p>Bedrooms are very small</p> <p>Newly refinished wood floors</p>

# Choice-Supportive Bias

- Humans attributed significantly more positive and fewer negative features to their chosen options than to foregone options.
  - “Remembering that the option we chose was the better one is more emotionally gratifying than remembering that the foregone option was better.”
  - [ Mara Mather and Marcia Johnson. *Choice-Supportive Source Monitoring: Do our decisions seem better to us as we age?* J. Psychology and Aging. ]
- Example SE Implication: Once you have chosen a language or tool for Project 1, you are likely to remember positives about that when choosing for Project 2.

# Java + C



# Programming Paradigms

- This “pass a string or an integer as the second argument” plan ...
  - Works well for Dynamic (e.g., Python duck typing)
  - Works well for Functional (algebraic datatypes)
    - See EECS 490
  - 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)

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

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  
                      (env, jplain,  
                      JNI_ABORT);  
    jint n_plainsize = (*env)->GetByteArrayLength(env, jplain);  
    jclass key_cls = (*env)->GetObjectClass(env, jkey);
```

**Java Native Interface  
environment  
provides services for  
manipulating Java values.**

**The self object  
is passed in as a  
'hidden' first  
parameter.**

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

# C Code using

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

```
JNIEXPORT jbyteArray JNICALL Java_String_002Dto_002FCString(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(key_cls, "tag", "I");
    if (fid != NULL) {
        /* key has "int tag" field */
        jint n_mask = (*env)->GetIntField(key_cls, 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

```
JNIEXPORT  
(JNIEXPORT JNICALL  
{  
    jbyte  
  
    size_  
    jclass  
    jfieldID  
    int i;  
    jbyteArray jcypher =  
    jbyte * n_cypher = (*  
  
    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 {
```

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 (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)

Field lookup again.  
"[B" == "[[] Byte"

```
    } 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 (int i = 0; i < n_keytext; i++) {  
                jbyte jbyte = jkey[i];  
                if (jbyte == 'A') {  
                    jplain[i] = jbyte; n_plain++;  
                } else if (jbyte == 'a') {  
                    jcypher[i] = jbyte; n_cypher++;  
                }  
            }  
            (*env)->ReleaseByteArrayElements(env, jplain, n_plain, 0);  
            (*env)->ReleaseByteArrayElements(env, jcypher, n_cypher, 0);  
            return jcypher;  
        }  
    }  
}
```

Can indicate whether  
elements were copied or shared.

Playing nice with  
the garbage collector.

# Compiling, Linking and Running JNI

```
gcc -I $(JAVA)/include \
```

```
    -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.





# SE Implications

LATEST: 10.17

UPDATE

CHANGES IN VERSION 10.17:  
THE CPU NO LONGER OVERHEATS  
WHEN YOU HOLD DOWN SPACEBAR.

COMMENTS:

LONGTIMEUSER4 WRITES:

THIS UPDATE BROKE MY WORKFLOW!  
MY CONTROL KEY IS HARD TO REACH,  
SO I HOLD SPACEBAR INSTEAD, AND I  
CONFIGURED EMACS TO INTERPRET A  
RAPID TEMPERATURE RISE AS "CONTROL".

ADMIN WRITES:

THAT'S HORRIFYING.

LONGTIMEUSER4 WRITES:

LOOK, MY SETUP WORKS FOR ME.  
JUST ADD AN OPTION TO REENABLE  
SPACEBAR HEATING.

EVERY CHANGE BREAKS SOMEONE'S WORKFLOW.

# Cross-Cutting Implications for SE

- **Hiring** and **Expertise**

- You need developers experienced with “both” languages
- Per-language experience may not be equal

- **Code Inspection** and **Review**

- Recall Google's per-language “badge” policy
  - Need badges in all relevant languages
- How would you evaluate a pull request if you do not know all of the languages?

# Cross-Cutting Implications for SE

## • Design

- Because cross-language coding is so difficult and error-prone, you must design those interfaces very carefully in advance
  - cf. native method interface ← key word
- Think carefully about relevant metrics (e.g., coupling, cohesion, etc.)
- Design patterns can help, but you typically want to encapsulate any cross-language code inside one
  - e.g., don't have some native code in the Model and some in the View and have them share: backdoor?

# Cross-Cutting Implications for SE

- **Readability**

- “Glue” code is typically incomprehensible without training
- Recall: look for familiar motifs
  - All of our examples have parts that “do the same thing” (e.g., convert value from X to C)
- But comprehension may also require knowing about both languages
  - Python and Java field queries
  - Ocaml integer conversions

# Cross-Cutting Implications for SE

## • Test Input Generation

- Most tools do not support test input generation across multiple language layers (it is an open research problem)
- AFL is popular because it works on binaries (and thus any compiled language)
- Microsoft's PEX works for any .NET / common language runtime program
- But do not assume tools will work for multi-language projects: plan in advance to mitigate risk!

# Cross-Cutting Implications for SE

- **Test Coverage**

- Outside of giant ecosystems (e.g., Java Bytecode, Microsoft Common Language Runtime), coverage tools do *not* span languages
  - Pick one or run them separately

- **Mutation Analysis**

- Similarly, mutation tools are typically language specific
- Exam-style thought question: should you mutate the glue code when doing mutation testing?

# Cross-Cutting Implications for SE

## • Debugging

- Outside of some bytecode/CLR instances, debuggers almost **never** help with multi-language projects
- You “can” run GDB on an Ocaml-produced (etc.) executable, but it won't see any of your function or variable names
  - Basically just a raw assembly view
  - cf. C++ name mangling

# Cross-Cutting Implications for SE

## • Debugging

- Typically you pick one language's debugger
- Augment that with print-statement debugging at interface boundaries
- Debugging multi-language code is merely “annoying” if the bug is isolated to code in just one language
- It is “very, very difficult” if the bug actually involves crossing the boundary



# Cross-Cutting Implications for SE

- **Static Analysis** and **Refactoring**

- Unless the tool happens to support all relevant languages it will only report defects in some of the code
  - And it will make conservative assumptions about what happens at the cross-language interface
    - Result: more false positives and/or false negatives
- Multi-language refactoring is an open research problem

# Cross-Cutting Implications for SE

- **Dynamic Analyses** and **Profiling**

- Similar story: unless the tool happens to support multiple languages (and most do not), you will have to pick one language and just use that language's tool
- Example: you can run gprof on a non-C-produced binary, but it probably will not be able to give recognizable function names or useful call graphs
- Thought question: would CHESSE or Eraser work on multi-language projects?

# Cross-Cutting Implications for SE

- **Process, Planning** and **Metrics**

- Will developers be as precise at effort estimation for coding in multi-language projects?
- How will you make high-level QA decisions (e.g., “is it good enough to ship?”) if coverage metrics only apply to part of the code?
- What additional risks do you take on by choosing to carry out a multi-language project?
  - How would you mitigate those risks?
- Do the benefits outweigh the costs?

# Cross-Cutting Implications for SE

- **Requirements** and **Quality Properties**
  - The dominant reason to use multiple languages is to gain the ease and safety of a high-level language for most of your program and the speed of a low-level one for critical kernels
    - This is a quality (non-functional) requirement
  - Another common reason is to make use of an already-written library (COTS)
    - This is usually a functional requirement
  - Elicitation: how critical are those to stakeholders?

# Homework

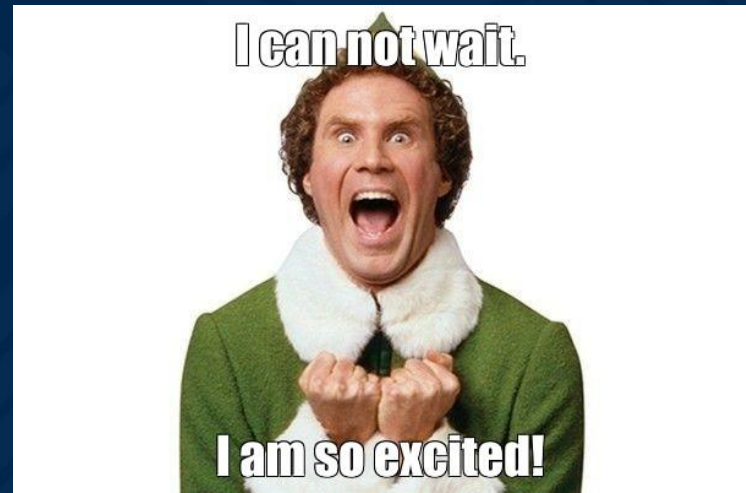
- Exam 2 Is A Thing
  - Cumulative
  - Don't neglect studying while working on HW6
  - Exam and HW6b are "due" one day apart
  
- Please attend the next two lectures so that we may convince you to join the dark side





# Bonus Material

(for BONUS LEARNING!)



# Bonus Learning

- The remaining slides have optional material presenting the same native method interface (“string XOR”) with a third language
- This material is *not required* for the Exam
- But seeing it a third way may help students who do not have as much systems programming background

# Exotic Language Example

- How do you maintain code in a language you don't really know?
- First, look for common patterns or markers!
  - cf. “song” exercise



“NOBODY UNDERSTANDS ME.”



# Basic Ocaml Implementation

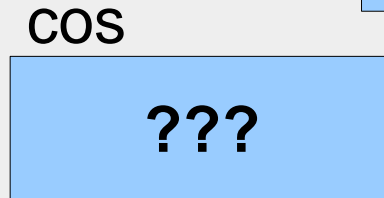
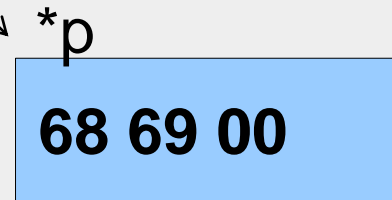
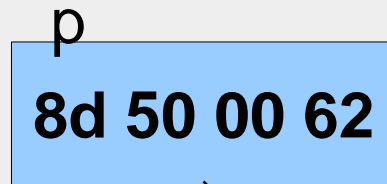
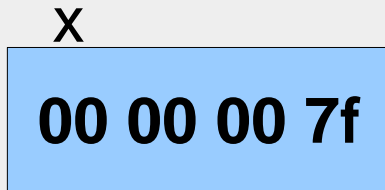
```
type char_or_string =  
  | MyChar of char      (* constant bit pattern *)  
  | MyString of string  (* one-time pad *)  
  
let ocaml_xor function plain key =  
  let cypher = String.create (String.length plain) in  
  ( match key with  
  | MyChar(mask) ->  
    for i = 0 to pred (String.length 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
```

# 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;`
- `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 ..**

# The Problem

- let cos = MyChar("\127") in  
cos

ff 00 00 00 00 00 00 00 fc 08 0

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

60 8d 62 00 00 00 00 00 fc 04



# 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 ..**

**0x628d60**

**68 69 00 00 ..**

# The Problem

- let cos = MyChar("\127") in  
COS

00 04 00 00 00 00 00 00 00 ff 00 00 00 00 00 00 00 00 fc 08 00 00 00 00 00 00 ..

- let cos2 = MyString("hi") in  
COS2

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 ..

0x628d60

68 69 00 00 ..

# The Problem

```
• let cos = MyChar("\127") in
  COS
```



Type Tag 0

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

Type Tag 0

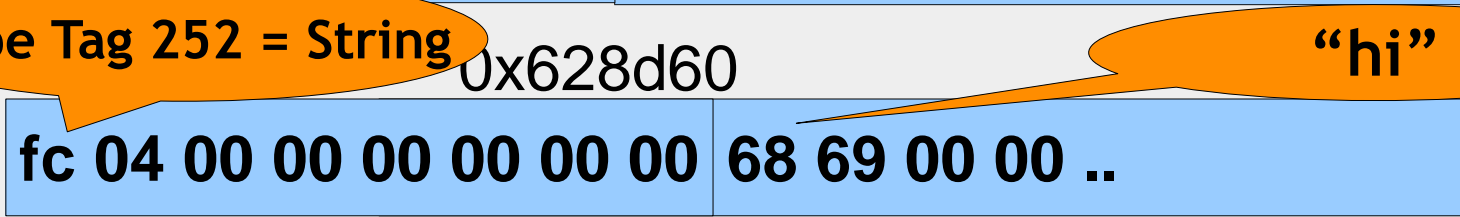
“Color” (2 bits)  
and Size (54 bits)

Pointer To String  
(little endian)



Type Tag 252 = String

“hi”





# 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

```
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_keytext ;  
  o_cypher = String_val(String(len);  
  MyChar:Ma  
  (v2, 0))  
  er[i] = i  
  == 1) { /*  
  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 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);
}
```

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

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

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

# Linking C and OCaml

- `$ ocamlc -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 ocamlc command line.**

# Ocaml Native Interface Debugging Example

- You try to write this C/OCaml code, but ...

- 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 in your code?

# Actual Numbers (Quality)

(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 (Quality)

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

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