Mono C++ Interop

What is it?

- Project started by Alex Corrado on GSOC 2010
  - Development ongoing
- Using c++ libraries without having to go through C

- Extending by subclassing
- Opening new areas
  - QT comes to mind

What Mono supports now

- For C, Pinvoke
  - Simple
  - Not object oriented
- For C++, COM
  - COM support is all about invoking virtual methods
  - Vtable layout is easy, well known, and doesn't require mangling
  - Pointers created in native

What do we want to do?

- Calling C++ non-virtual and static methods
- Instantiating classes in C#
- Subclassing
- overriding

The problems

- Virtual calls are easy
  - Vtable layout is known
  - Can be mapped with a C# interface
  - Doesn't require mangling
- Everything else is hard
  - Name mangling
  - Class instantiating on the C# side
  - Subclassing
  - Overriding methods

Name mangling

- _ZN12QApplication4execEv = Qapplication::exec()
- Each compiler does mangling differently
  - GCC has two versions
  - MSVC
- Fortunately, there's a paper detailing the different mangling strategies
- GCC and c++filt source
- This part is mostly done for GCC and MSVC
- Need to know the exact signature for every method
  - C# types are not enough

Class instantiating

- The easy way is to use a C++-created pointer
- Doing it in C# requires
  - Knowing the size native expects it to be
  - A way to support callbacks from C++
- Size requires knowing not only how big the class is, but also how big all the base classes are
  - Requires an exact map of all the C++ classes
- Callbacks also require deep knowledge of the class
- Same for subclassing, overrides, etc


Im in ur
computer
generating ur code

Data is what we need...

- C++ interop requires defining classes, structs and interfaces in C# with full type information
- Parsing C++ headers is a pain!
- GCC-XML + Binding generator to the rescue!

```
<Constructor id="_6455" name="QPushButton" explicit="1"
             context="_1718" access="public"
             mangled="_ZN11QPushButtonC1EP7QWidget *INTERNAL* "
             demangled="QPushButton::QPushButton(QWidget*)"
             location="f29:66" file="f29" line="66" extern="1">

      <Argument name="parent" type="_2824" location="f29:66"
                file="f29" line="66" default="0"/>

</Constructor>
```

... to generate C# bindings

```
QtGui  = new CppLibrary ("QtGui", new ItaniumAbi ());

public class QPushButton : QabstractButton {

  static IQPushButton impl = Qt.Libs.QtGui.GetClass<IQPushButton,
     _QPushButton, QPushButton> ("QpushButton");

  public QPushButton (QWidget parent) : base(impl.TypeInfo) {

     Native = impl.Alloc (this);

     impl.QPushButton (Native, parent);
  }


[...]


public interface IQPushButton : ICppClassOverridable<QPushButton> {

   [Constructor]
   void QPushButton (CppInstancePtr @this, QWidget parent);
```


place cat here.
See? The system works
ICANHASCHEEZBURGER.COM

How does it work

- Library created at runtime via Reflection.Emit from the binding declarations
- An interface represents the actual C++ class
- It inherits from one of
  - ICppClass
    - native pointer, static or instance methods
  - ICppClassInstantiatable
    - C# instantiated object,
  - ICppClassOverridable<T>
    - C# instantiated
    - Virtual methods can be overriden

How does it work

- Types are tagged with [MangleAs]
  - C# type information is not enough to figure out the mangled name
- Methods are tagged with [Constructor], [Destructor], [Virtual]
- DllImport == CppLibrary via GetClass call
- ItaniumAbi for GCC, MsVcAbi for Windows
  - Detect at runtime from environment
    - Or even checking the library exports

How does it work

- CppType
  - Type abstraction
  - Primitives, classes, structs, enums, unions, modifiers
- CppTypeInfo
  - Type memory layout
  - Constructed at runtime, passed to base classes to layout everything correctly
  - Information about vtable slots, offsets for fields

From header to heaven

- Generate a complete description with gcc-xml
- Parse the xml and generate C# bindings
- When you run:
  - bindings are processed, assembly is emitted
  - contains DllImports with compiler-specific mangling
  - Information about memory layouts for instances
  - You get an instance that C++ likes
- Our objective is to reduce manual code to 0
  - And we're almost there! But not quite...

Current status

Current status

- Generator is working
- Framework is in place
  - Currently blowing up, sorry!
    - Native size calculations are probably a bit off
- Polymorphism is... complicated
- Some support for templates... but they're a pain
- More mangling support is needed
- Generated bindings not quite right yet, need tweaks

That's it!

- Come help us!
- github.com/andreiagaita/cppinterop
- github.com/nirvanai/cppinterop
- shana@spoiledcat.net