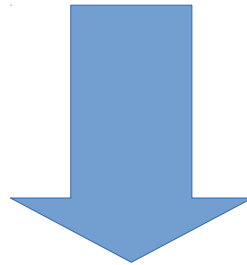


Lowcode

Redoing NativeBoost Portably

Ronie Salgado

```
nbStr: largeString str: smallerString  
<primitive: #primitiveNativeCall module: #NativeBoostPlugin>  
^ self nbCall: #(String strstr(String largeString, String smallerString)) module: self cLibraryName
```



```
str: largeString str: smallerString  
^ self ffiCall: #(String strstr(String largeString, String smallerString)) module: self cLibraryName
```

The Current FFI Problem

- SqueakFFI/ThreadedFFI, Alien, NativeBoost.
- The callbacks
- Performance
- Raw memory manipulation?

Ways of interfacing with C

- Extend the Interpreter. Done in Java(JNI), Lua, Python
- Just call. Done in .Net, D

Extending the Interpreter

```
static PyObject *
spam_system(PyObject *self, PyObject *args)
{
    const char *command;
    int sts;

    if (!PyArg_ParseTuple(args, "s", &command))
        return NULL;
    sts = system(command);
    return Py_BuildValue("i", sts);
}
```

Python Example. Source: <https://docs.python.org/2/extending/extending.html>

Just Calling

```
using System;
using System.Runtime.InteropServices;

namespace Sample
{
    public unsafe class Sample
    {
        [DllImport("libc-2.17.so")]
        private static extern void puts(byte *str);

        [DllImport("libc-2.17.so")]
        private static extern void* malloc(long size);

        [DllImport("libc-2.17.so")]
        private static extern void free(void *p);

        public static void Main(string[] args)
        {
            byte *str = (byte*)malloc(2);
            str[0] = (byte)'a';
            str[1] = 0;
            puts(str);
            free(str);
        }
    }
}
```

Why I do not like NativeBoost

```
nbFloat32AtOffset: zeroBasedOffset put: value
```

```
"Store 32-bit float at ZERO-based index.
```

```
Note, there is no range checking "
```

```
<primitive: #primitiveNativeCall module: #NativeBoostPlugin>
```

```
^ self nbCallout function: #(void (self, ulong zeroBasedOffset, float32 value)) emit: [:gen |  
  | asm |
```

```
  asm := gen asm.
```

```
  asm
```

```
    pop: asm EAX; "pointer to receiver's first byte"
```

```
    pop: asm ECX; "offset"
```

```
    pop: asm EDX;
```

```
    mov: asm EDX to: asm EAX ptr + asm ECX
```

```
]
```

More Portable

```
ffiFloat32AtOffset: zeroBasedOffset put: value
```

```
  ^ Lowcode here: [ :gen |
```

```
    gen pushReceiver;
```

```
      firstIndexableFieldPointer;
```

```
      pushTemp: 0;
```

```
      oopToInt32;
```

```
      pointerOffset32;
```

```
      pushTemp: 1;
```

```
      oopToFloat32;
```

```
      storeFloat32ToMemory;
```

```
      returnReceiver
```

```
  ]
```

What I would like to do

```
ffiFloat32AtOffsetPut: zeroBasedOffset put: value  
(float*)self atOffset: zeroBasedOffset put: value
```

The actual syntax has to be discussed.

What I also want to do

```
ffiFloat32AtOffsetPut: zeroBasedOffset put: value {  
    ((float*)self)[0] = value;  
}
```

How it is done

- Extend the Sista Extended Bytecode set.
- Expose basic CPU primitive type.
- Implemented in the VM.
- Implemented in plugins.

Many Backends

- C Interpreter plugin. (Done)
- C++ LLVM plugin. (Done)
- VirtualCPU. (Incomplete)
- StackInterpreter. (ToDo)
- Cogit. (ToDo)