The Crack Scripting Language

Because Life is too Short to Wait for Software

http://www.mindhog.net/pub/CrackScriptingLanguage
What is It?

Crack

- A C/C++/Java-like Scripting Language.
- Speed of a compiled language, ease of use of a scripting language.
History

- First conception of "a scripting language that compiles to machine code" in the mid nineties.
- Experimental language in 2001ish (built on GNU Lightning, called it "thunder" -> "crack")
- Work on the current language began in September 2009.
- Released 0.1 in mid-July 2010
- Released 0.2 at the beginning of October
Other Scripting Languages that Compile to Machine Code

- Python has "Unladen Sparrow" (Built on LLVM)
- Ruby has "Rubinius" (Also built on LLVM)
- JavaScript has V8 and TraceMonkey (and probably others)
- PHP has "HipHop" and RoadSend.
Dynamic Attributes and Typing

Pros:

- Users don't have to specify types (decreased verbosity).
- Duck-typing (types must conform to well defined interfaces)

Cons:

- Complicates the compiler.
- Less protection, problems are discovered at runtime.
- Interfaces are less obvious.
Crack - Guiding Principals

- Everything should be fast.
- Common things should be terse.
- "Use the existing wiring" for C/C++/Java programmers.
The Language

Hello World

#!/usr/bin/crack
import crack.io cout;
cout `hello world
`;
Comments

C, C++ and Shell style comments are all supported:

    /* C-Style */
    // C++ style
    # shell style
Primitive Types

byte b;
int32 i;
int64 j;
uint32 u;
uint64 v;
int k;
uint w;
bool x;
void f() {}
Primitive Pointer Types

byteptr ptr;  // an array of bytes
voidptr v;    // for comparison with null
Primitive Arrays

array[int] arr = array[int](100);
arr[0] = 10;
arr[1] = 20;
Aggregate Types

String s = "this is a string"; // a string of bytes

class Soldier {
    String name;
    int rank;
};
Avoiding Verbosity
Simplifying Construction and Definition

In Java:

    BigClassName variable = new BigClassName();

In Crack:

    variable := BigClassName(); // ... or
    BigClassName variable = {}; // ... or
    BigClassName variable; // hmmm... maybe not.
Avoiding Verbosity

Efficient Construction of Collections:

    List[Int] list = [1, 2, 3, 4];

Type Inferencing:

    class A { oper init(int x, int y) { ... } }
    void foo(A a) { ... }
    foo({1, 2});  // same as foo(A(1, 2));

Iteration:

    for (x :in list)
        doSomethingWith(x);
String Interpolation

```
a := 1;
b := 2;
cout `\$a + \$b = $(a + b)`;
// prints "1 + 2 = 3"
```
String Interpolation

```cpp
    cout `$a + $b = $(a + b)`;
```

Is syntactic sugar for:

```cpp
    if (cout) {
        cout.format(a);
        cout.format(" + ");
        cout.format(b);
        cout.format(" = ");
        cout.format(a + b);
        cout.format("\n");
    }
```
Like Python or Perl, Crack lets you load common code from modules:

```javascript
import crack.io cout, StringFormatter;
fmt := StringFormatter();
fmt `hello world\n`;
cout.write(fmt.createString());
```
Shared Libraries

Crack lets you import functions from shared libraries, declare and call them:

```c
import "lib.so.6" abort;
void abort();
abort();
```
Generics

Crack generic syntax will be similar to Java's:

```java
class List<T : Object> {
    void add(T element) { ... }
    T oper [](uint index) { ... }
}
```
Annotations

Crack annotations will be like compiler plugins:

```c
    # annotation to trace when we enter and leave a function
    @myAnnotation
    void func() { ... }
```

During parsing:

```c
    myAnnotation(context);
```
Challenges with LLVM

- Placeholder instructions.
- Single Module vs. Multiple Modules.
The Future of Crack

- To become a major (or even a minor) language, Crack needs mindshare.
- If you think it's a good idea - we'd love more developers!
References