Cycript Tricks

Contents

Getting objects

Objective-C objects using choose()

The function `choose()`, introduced in version 0.9.502[citation needed] and documented here (http://www.cycript.org/manual/#7061c058-5485-4c00-be7e-b67accc55796), allows us to get an array of existing objects of a certain class.

Objective-C objects from addresses

Use `#0xdeadbabe`.

```
cy# var p = #0x8614390
cy# p
["<SKPaymentTransaction: 0x8613d80>"]
```

Javascript variables

Requires testing.

```
cy# typedef int a;
cy# for (x in this) if (x == 'a') system.print('yay');
```

Getting ivars

Often just typing `*varName` works:

```
cy# *controller
{isa:"PrefsRootController",_contentView:"<UIView: 0x10bd70; frame = (0 0; 320 460); autoresize = W+H; layer =
<CALayer: 0x150120>>",_navBar:...
cy#
```

Sometimes it does not...

```
cy# *UIApp
{message:"hasProperty callback returned true for a property that doesn't exist.",name:"ReferenceError"}
```

Then you can do:

```
cy# [i for (i in *UIApp)]
["isa","_delegate","_touchMap","_exclusiveTouchWindows","_event",...
```

You may use this function to get as much ivar values as possible:

```
function tryPrintIvars(a){ var x={}; for(i in *a){ try{ x[i] = (*a)[i]; } catch(e){} } return x; }
```

To use:

```
cy# *a
{message:"hasProperty callback returned true for a property that doesn't exist.",name:"ReferenceError"}
cy# tryPrintIvars(a)
{isa:"SBWaveView",_layer:"<CALayer: 0x2a5160>",_tapInfo:null,_gestureInfo:null,_gestureRecognizers:...
```

Getting bundle identifier

```
NSBundle.mainBundle.bundleIdentifier
```

Getting methods

Function to get the methods:

```
function printMethods(className, isa) {
  var count = new new Type("I");
  var classObj = (isa != undefined) ? objc_getClass(className).constructor : objc_getClass(className);
  var methods = class_copyMethodList(classObj, count);
  var methodsArray = [];
  for(var i = 0; i < *count; i++) {
    var method = methods[i];
    methodsArray.push({selector:method_getName(method), implementation:method_getImplementation(method)});
  }
  free(methods);
  return methodsArray;
}
```

Usage:

```
cy# printMethods("MailboxPrefsTableCell")
[{selector:@selector(layoutSubviews),implementation:0x302bf2e9},
{selector:@selector(setCurrentMailbox:),implementation:0x302bee0d},...
cy#
```

You can also just look at the prototype property of the isa, e.g. to get rootViewControllers methods: `UIApp.keyWindow.rootViewController.isa.prototype`

Get methods matching particular RegExp

```
function methodsMatching(cls, regexp) { return [[new Selector(m).type(cls), m] for (m in cls.prototype) if (!regexp ||
regexp.test(m))]; }
```

Usage:

```
cy# methodsMatching(NSRunLoop, /forKey:$/)
[["v20@0:4I8@12@16","didChange:valuesAtIndexes:forKey:"],["v20@0:4I8@12@16","willChange:valuesAtIndexes:forKey:"],
["v16@0:4@8@12","setValue:forKey:"]]
```

Getting class methods

class`.prototype` only contains instance methods. To hook class methods, you need to get to its *metaclass*. A simple way would be

```
cy# NSRunLoop.constructor.prototype['currentRunLoop'] = ...
```

Alternatively, set the optional second parameter in `printMethods()` to `true`, e.g. `printMethods("NSRunLoop", true)`

```
cy# printMethods("NSRunLoop", true)
[{selector:@selector(currentRunLoop),implementation:&(extern "C" id 674681217(id, SEL, ...))}...
```

Replacing existing Objective-C methods

You can simulate MSHookMessage by replacing contents in the `prototype` array, e.g.

```
cy# original_NSRunLoop_description = NSRunLoop.prototype['description'];
(extern "C" id ":description"(id, SEL))
cy# NSRunLoop.prototype['description'] = function() { return
original_NSRunLoop_description.call(this).toString().substr(0, 80)+", etc."; }
function (){var e;e=this;return original_NSRunLoop_description.call(e).toString().substr(0,80)+", etc."}
cy# [NSRunLoop currentRunLoop]
#"<CFRunLoop 0x13750a630 [0x1a103e150]>{wakeup port = 0x1003, stopped = false, ign, etc."
```

Note the `func.call(this)` construct. This binds the `this` in the original function to the user-specified one. If more than one variable is needed, use `function(arg1, arg2, arg3, ...) {...func.call(self, arg1, arg2, arg3, ...);}`, e.g.

```
cy# original_SpringBoard_menuButtonDown = SpringBoard.prototype['menuButtonDown:']
0x17dbab1
cy# SpringBoard.prototype['menuButtonDown:'] = function(arg1) {original_SpringBoard_menuButtonDown.call(this, arg1);}
function (e) {var e;var $cy0=this;original_SpringBoard_menuButtonDown.call($cy0,e);}
```

Note that the subsequent arguments will not be automatically mapped to the corresponding Objective-C types, so instead of "foo" you will need to use `[NSString stringWithString:"foo"]`.

List all subclasses

```
[c for each (c in ObjectiveC.classes) if (class_getSuperclass(c) && [c isSubclassOfClass:UIView])]
```

(The `class_getSuperclass` is needed to prevent crash due to the "Object" class not inheriting from NSObject)

Load frameworks

```
function loadFramework(fw) {
  var h="/System/Library/",t="Frameworks/"+fw+".framework";
  [[NSBundle bundleWithPath:h+t]||[NSBundle bundleWithPath:h+"Private"+t] load];
}
```

Include other Cycript files

As of 0.9.274-1, there isn't a native file import feature. If cycript will be hooked into another process, since the data will be retained there, you can first load the other .cy file with this:

```
localhost:~ mobile$ cycript -p SpringBoard main.cy
0x12345678
localhost:~ mobile$ cycript -p SpringBoard
cy# ...
```

If cycript is launched standalone, inclusion can still be faked with a combination of cycript compiler and Javascript's `eval` function:

```
// include other .cy files
function include(fn) {
  var t = [new NSTask init]; [t setLaunchPath:@"/usr/bin/cycript"]; [t setArguments:["-c", fn]];
  var p = [NSPipe pipe]; [t setStandardOutput:p]; [t launch]; [t waitUntilExit];
  var s = [new NSString initWithData:[[p fileHandleForReading] readDataToEndOfFile] encoding:4];
  return this.eval(s.toString());
}
```

As of version 0.9.502[citation needed], there is. See @import's documentation (http://www.cycript.org/manual/#754b6781-f13f-4a0a-ba2f-594d5778e97f).

Using NSLog

In recent versions of cycript, NSLog should just work. If not, using the following:

Type in the console:

```
NSLog_ = dlsym(RTLD_DEFAULT, "NSLog")
NSLog = function() { var types = 'v', args = [], count = arguments.length; for (var i = 0; i != count; ++i) { types += '@'; args.push(arguments[i]); } new Functor(NSLog_, types).apply(null, args); }
```

And then you can use NSLog as usual:

```
cy# NSLog_ = dlsym(RTLD_DEFAULT, "NSLog")
0x31451329
cy# NSLog = function() { var types = 'v', args = [], count = arguments.length; for (var i = 0; i != count; ++i) {
types += '@'; args.push(arguments[i]); } new Functor(NSLog_, types).apply(null, args); }
```

```
{}
cy# NSLog("w ivars: %@", tryPrintIvars(w))
```

If you are attached to a process, the output is going to be in the syslog:

```
Nov 17 20:26:01 iPhone3GS Foobar[551]: w ivars: {\n    contentView = <UIView: 0x233ea0; ....}
```

Using CGGeometry functions

CGPoint, CGSize, and CGRect are structures of numbers (floats or doubles), and can be represented in Cycript with simple arrays:

```
cy# view.frame = [[10, 10], [100, 100]];
[[10,10],[100,100]]
```

If you'd prefer to use the CG...Make() functions, you can construct them yourself:

```
function CGPointMake(x, y) { return [x, y]; }
function CGSizeMake(w, h) { return [w, h]; }
function CGRectMake(x, y, w, h) { return [[x, y], [w, h]]; }
```

Using NSError

```
cy# var error = new @encode(NSError *)
&null
cy# var thing; [[NSFileManager defaultManager] copyItemAtPath:@"aaadsdsds" toPath:@"bbbdsdsdsds" error:error]; thing =
*error
cy# thing
#'Error Domain=NSCocoaErrorDomain Code=260 "The file \xe2\x80\x9caaadsdsds\xe2\x80\x9d couldn\xe2\x80\x99t be opened
because there is no such file." UserInfo=0x100310af0 {NSFilePath=aaadsdsds, NSUnderlyingError=0x1003108e0 "The
operation couldn\xe2\x80\x99t be completed. No such file or directory"}'
```

Writing Cycript output to file

Cycript output is an NSString, so it is possible to call writeToFile and save it somewhere. Example:

```
[[someObject someFunction] writeToFile:"/var/mobile/cycriptoutput.txt" atomically:NO encoding:4 error:NULL]
```

You can use this, for example, to get a dump of SpringBoard's view tree.

```
iPhone:~$ cycript -p SpringBoard
cy# [[UIApp->_uiController.window recursiveDescription] writeToFile:"/var/mobile/viewdump.txt" atomically:NO
encoding:4 error:NULL]
```

Printing view hierarchy

```
cy# UIApp.keyWindow.recursiveDescription().toString()
"<UIWindow: 0x13a900; frame = (0 0; 320 480); opaque = NO; autoresize = RM+BM; layer = <CALayer: 0x13a9d0>>
   <UITextField: 0x13abf0; frame = (20 40; 280 31); text = ''; opaque = NO; autoresize = RM+BM; layer = <CALayer:
0x13ad10>>
      <UITextFieldRoundedRectBackgroundView: 0x143d10; frame = (0 0; 280 31); userInteractionEnabled = NO; layer =
<CALayer: 0x143dc0>>
         <UIImageView: 0x144030; frame = (0 0; 8 15); opaque = NO; userInteractionEnabled = NO; layer = <CALayer:
0x1440b0>>
         <UIImageView: 0x144400; frame = (8 0; 264 15); opaque = NO; userInteractionEnabled = NO; layer = <CALayer:
0x144430>>
         <UIImageView: 0x144460; frame = (272 0; 8 15); opaque = NO; userInteractionEnabled = NO; layer = <CALayer:
```

```
0x144490>>
            <UIImageView: 0x1444c0; frame = (8 0; 0 15); userInteractionEnabled = NO; layer = <CALayer: 0x1444f0>>
            <UIImageView: 0x144520; frame = (0 15; 8 1); opaque = NO; userInteractionEnabled = NO; layer = <CALayer:
0x144550>>
            <UIImageView: 0x144580; frame = (8 15; 264 1); opaque = NO; userInteractionEnabled = NO; layer = <CALayer:
0x1445b0>>
            <UIImageView: 0x1445e0; frame = (272 15; 8 1); opaque = NO; userInteractionEnabled = NO; layer = <CALayer:
0x144610>>
            <UIImageView: 0x144640; frame = (8 15; 0 1); userInteractionEnabled = NO; layer = <CALayer: 0x144670>>
            <UIImageView: 0x1446a0; frame = (0 16; 8 15); opaque = NO; userInteractionEnabled = NO; layer = <CALayer:
0x1446d0>>
            <UIImageView: 0x144700; frame = (8 16; 264 15); opaque = NO; userInteractionEnabled = NO; layer =
<CALayer: 0x144730>>
            <UIImageView: 0x144760; frame = (272 16; 8 15); opaque = NO; userInteractionEnabled = NO; layer =
<CALayer: 0x144790>>
            <UIImageView: 0x1447c0; frame = (8 16; 0 15); userInteractionEnabled = NO; layer = <CALayer: 0x1447f0>>
        <UILabel: 0x13aaf0; frame = (9 8; 266 15); text = 'Test'; clipsToBounds = YES; opaque = NO;
userInteractionEnabled = NO; layer = <CALayer: 0x1399f0>>"
```

Cycript scripts

Custom shell function that loads a cycript file:

```
cyc () { cycript -p $1 /var/root/common.cy > /dev/null; cycript -p $1; }
```

```
Usage: cyc ProcessName
```

Add this to /etc/profile.d/cycript.sh to make it available in all sessions.

> **Warning:** If you run this command multiple times against a process, the scripts will be loaded into the script multiple times. This could potentially have unexpected consequences depending on the scripts you are loading. It is not a proper way of doing this and saurik recommends against it.

Weak Classdump (Cycript based class-dump)

Link: https://github.com/limneos/weak_classdump

Usage:

```
root# cycript -p Skype weak_classdump.cy; cycript -p Skype
'Added weak_classdump to "Skype" (1685)'
cy# UIApp
"<HellcatApplication: 0x1734e0>"
cy# weak_classdump(HellcatApplication);
"Wrote file to /tmp/HellcatApplication.h"
cy# UIApp.delegate
"<SkypeAppDelegate: 0x194db0>"
cy# weak_classdump(SkypeAppDelegate,"/someDirWithWriteAccess/");
"Wrote file to /someDirWithWriteAccess/SkypeAppDelegate.h"
root# cycript -p iapd weak_classdump.cy; cycript -p iapd
'Added weak_classdump to "iapd" (1127)'
cy# weak_classdump(IAPPortManager)
"Wrote file to /tmp/IAPPortManager.h"
root# cycript -p MobilePhone weak_classdump.cy; cycript -p MobilePhone
'Added weak_classdump to "MobilePhone" (385)'
#cy weak_classdump_bundle([NSBundle mainBundle],"/tmp/MobilePhone")
"Dumping bundle... Check syslog. Will play lock sound when done."
```

Utils

Link: https://github.com/Tyilo/cycript-utils