Google

Code

Archive

networkpx - class_dump_z.wiki

Export to GitHub

Download: * **0.2a**: http://networkpx.googlecode.com/files/class-dump-z_0.2a.tar.gz (Windows not yet supported). * **0.2-0**: http://networkpx.googlecode.com/files/class-dump-z_0.2-0.tar.gz .

``` Usage: class-dump-z []

where options are:

Analysis: -p Convert undeclared getters and setters into properties (propertize). -h proto Hide methods which already appears in an adopted protocol. -h super Hide inherited methods. -y Choose the sysroot. Default to the path of latest iPhoneOS SDK, or /. -u Choose a specific architecture in a fat binary (e.g. armv6, armv7, etc.)

Formatting: -a Print ivar offsets -A Print implementation VM addresses. -k Show additional comments. -k -k Show even more comments. -R Show pointer declarations as int *a instead of int* a. -N Keep the raw struct names (e.g. do no replace __CFArray* with CFArrayRef). -b Put a space after the +/- sign (i.e. + (void)... instead of +(void)...). -i Read and update signature hints file.

Filtering: -C Only display types with (original) name matching the RegExp (in PCRE syntax). -f Only display methods with (original) name matching the RegExp. -g Display exported classes only. -X Ignore all types (except categories) with a prefix in the comma-separated list. -h cats Hide categories. -h dogs Hide protocols.

Sorting: -S Sort types in alphabetical order. -s Sort methods in alphabetical order. -z Sort methods alphabetically but put class methods and -init... first.

Output: -H Separate into header files -o Put header files into this directory instead of current directory. ```

## Why a yet another class-dump?

class-dump is a command-line tool to extract Objective-C class interfaces, written by Steve Nygard 17 years ago. The development however paused in 2007 at version 3.1.2, without support for the newest ABI.
```

This caused the birth of class-dump-x by Holly Lee in 2008. But being a straight modified version of class-dump, it inherited some of the problems, e.g. the ivar offsets are calculated wrongly, properties are not supported, etc.

I, using class-dump-x a lot for reverse engineering, finds that the ivar offset problem is hard to get over with. Having no answers a few months after a bug report, I decided to just fix the problem myself — hence class-dump-z is started.

(While I was creating class-dump-z, the original class-dump development suddenly reactivated at July 1st. The newest version 3.2 does support 2.0 ABI now, but the ivar offset info is even worse than class-dump-x.)

Why you don't want to use class-dump-z

Instead of a generic-purpose class dumper, class-dump-z was written with iPhoneOS development in mind. Therefore, the following features will probably never be implemented: * 64-bit support. (Unless there is a 4 GiB RAM iPhone.) * Objective-C 1.0 ABI. In general, if you want to dump Mac OS X, you should use the original class-dump. But if you're dumping iPhoneOS binaries, class-dump-z is definitely the better choice.

Features

10x the speed

| **Time to dump...** | class-dump-x 3.1.2 | class-dump 3.3.1 | class-dump-z 0.2-0 | |:-------------------|:-----------------|:----------------|:-----------------| | **UIKit 2.0** (run in Mac OS X) | 2.2s | 1.8s | **0.13s** | | **UIKit 3.1** (run in Mac OS X) | 2.9s | 2.1s | **0.19s** | | **CoreData 3.0** (run in iPhoneOS) | 21s | - | **2.6s** |

- `time class-dump-? -a -A UIKit > /dev/null`

class-dump-z is written from scratch using C++ avoiding using dynamic calls, unlike class-dump and class-dump-x which are written in Objective-C. Removing these unnecessary calls makes class-dump-z near 10 times faster than the precedences.

Portable

| **Platforms** | class-dump-x 3.1.2 | class-dump 3.3.1 | class-dump-z 0.2-0 | |:--------------|:-----------------|:----------------|:-----------------| | Mac OS X 10.6 | ✓ | ✓ | ✓ | | iPhoneOS 3.1 | ✓ | ✗ | ✓ | | Linux | ✗ | ✗ | ✓ | | Windows | ✗ | ✗ | ✓ |

Since class-dump-z is written in C++, it is very easy to port to other platforms. Currently Mac OS X 10.6, iPhoneOS 3.1, Linux (x86 and amd64) and Windows (≥XP) are officially supported.

Correct ivar offsets

`UIRemoveControlMultiSelectButton` 3.1`UIMoreListController` 3.1 **class-dump-x 3.1.2**

```
unsigned int _isHighlighted:1; // 68 = 0x44

unsigned int _isSelected:1;    // 69 = 0x45


UITableView *_table;                          // 92 = 0x5c

BOOL _allowsCustomizing;                      // 96 = 0x60

NSArray *_moreViewControllers;                // 97 = 0x61

UIMoreListCellLayoutManager *_layoutManager; // 101 = 0x65
```

class-dump 3.3.1

```
unsigned int _isHighlighted:1;

unsigned int _isSelected:1;
```

(No offsets shown. Don't know why.)

```
UITableView *_table;

BOOL _allowsCustomizing;

NSArray *_moreViewControllers;

UIMoreListCellLayoutManager *_layoutManager;
```

class-dump-z 0.2-0

```
unsigned _isHighlighted : 1; // 68 = 0x44

unsigned _isSelected : 1;    // 68 = 0x44


UITableView* _table;                          // 92 = 0x5c

BOOL _allowsCustomizing;                      // 96 = 0x60

NSArray* _moreViewControllers;                // 100 = 0x64

UIMoreListCellLayoutManager* _layoutManager; // 104 = 0x68
```

- `class-dump-? -a -C UIMoreListController UIKit`

Generating ivar offsets by accumulation is a tricky business, due to alignments and bitfield packing. It is so tricky that the compiler will generate this info into the file. class-dump-z will read from that part of memory and give the most correct result.

Struct name prettifying

| | `UIScrollView` 3.0 | |:|:----------| | class-dump-x 3.1.2 | `-(id)hitTest:(struct CGPoint)fp8 forEvent:(struct __GSEvent *)fp16;` | | class-dump 3.3.1 | `-(id)hitTest:(struct CGPoint)arg1 forEvent:(struct __GSEvent *)arg2;` | | class-dump-z 0.2-0 | `-(id)hitTest:(CGPoint)test forEvent:(GSEventRef)event;` |

- `class-dump-? -C UIScrollView -f hitTest UIKit`

class-dump-z will `typedef` structs and unions to the most presentable name with heuristics.

> *This feature can be explicitly turned off with the -N switch.*

Stable name generation for anonymous structs

| | `UIThreePartImageView` 2.0 | `UIThreePartImageView` 3.0 | |:|:----------|:----------| | class-dump-x 3.1.2 | `-(void)setSlices:(CDAnonymousStruct8)fp8;` | `-(void)setSlices:(CDAnonymousStruct10)fp8;` | | class-dump 3.3.1 | `-(void)setSlices:(CDStruct_75b8db5d)arg1;` | `-(void)setSlices:(CDStruct_75b8db5d)arg1;` | | class-dump-z 0.2-0 | `-(void)setSlices:(XXStruct_UUz0SD)slices;` | `-(void)setSlices:(XXStruct_UUz0SD)slices;` |

- `class-dump-? -C UIThreePartImageView UIKit`

Ever tried to diff a library between two versions? You'll be frustrated by so many differences that are caused by a change of indices in anonymous structs. No more problem in class-dump-z — as long as the struct is having the same members, the generated name will be fixed.

> *The name is computed by the CRC-32 checksum of the Objective-C type encoding of the struct.*

Properties

| | `UIScrollView` 3.0 | |:|:----------| | class-dump-x 3.1.2 | N/A | | class-dump 3.3.1 | `@property(nonatomic) struct CGPoint contentOffset;` | | class-dump-z 0.1-11o | `@property(assign, nonatomic) CGPoint contentOffset;` |

- `class-dump-? -C UIScrollView UIKit`

class-dump-z supports declared properties. Not only that, it supports *every property attributes*, including the undocumented ones. Moreover, it will hide the extra copy of getters/setters if a property is present.

Propertization

Some libraries are written before the dot syntax was introduced or by some dot-syntax-haters, so you'll see a long list of getters/setters like `-(void)setTitle:(id)title; -(id)title; -(void)setSubtitle:(id)subtitle; -(id)subtitle; ...` I found it pretty annoying. In class-dump-z you can supply the `-p` switch to automatically convert them into `@property(retain) id title; @property(retain) id subtitle; ...`

Hide inherited and delegate methods

`@interface UITableViewController : UIViewController <UITableViewDelegate, UITableViewDataSource> { int _tableViewStyle; id _keyboardSupport; } @property(retain, nonatomic) UITableView* tableView; // inherited: -(id)init; -(id)initWithStyle:(int)style; // inherited: -(void)dealloc; -(id)existingTableView; // declared property getter: -(id)tableView; // declared property setter: -(void)setTableView:(id)view; // inherited: -(void)loadView; // inherited: -(void)viewWillAppear:(BOOL)view; // inherited: -(void)viewWillDisappear:(BOOL)view; // inherited: -(void)viewDidAppear:(BOOL)view; -(void)setEditing:(BOOL)editing animated:(BOOL)animated; -(void)_adjustTableForKeyboardInfo:(id)keyboardInfo; // in a protocol: -(int)tableView:(id)view numberOfRowsInSection:(int)section; // in a protocol: -(id)tableView:(id)view cellForRowAtIndexPath:(id)indexPath; // in a protocol: -(void)tableView:(id)view willBeginEditingRowAtIndexPath:(id)indexPath; // in a protocol: -(void)tableView:(id)view didEndEditingRowAtIndexPath:(id)indexPath; @end` Method overloading in subclass is very common in OO design, but such information is useless when generating headers. Yet, by default a class dumper wouldn't know if an implementation is overloaded or a new method. If it's not useful, why not hide them? class-dump-z does that.

Another class of useless information are those being implemented to adopt a protocol. class-dump-z can also filter them out.

> *If you are running class-dump-z outside of the iPhoneOS, or you haven't installed the SDK, you have to tell class-dump-z where can the libraries be found by the -y switch. If you using using firmware 3.1, because all framework binaries are removed, hiding inherited methods from external frameworks won't work properly.*

Readable argument names

| | `UIImage(UIImageDeprecated)` 3.0 | |:|:----------| | class-dump-x 3.1.2 | `-(void)draw9PartImageWithSliceRects:(CDAnonymousStruct13)fp8 inRect:(struct CGRect)fp152;` | | class-dump 3.3.1 | `-(void)draw9PartImageWithSliceRects:(CDStruct_c8cd2c5d)arg1 inRect:(struct CGRect)arg2;` | | class-dump-z 0.2-0 | `-(void)draw9PartImageWithSliceRects:(XXStruct_4cr1oD)sliceRects inRect:(CGRect)rect;` |

- `class-dump-? -C UIImageDeprecated UIKit`

No more meaningless fpXX. class-dump-z will give a suitable name to each argument using Apple's coding style guide.

Correct header generation

| | `WebThreadSafeUndoManager` 3.1 | |:|:----------------------------| | class-dump-x 3.1.2 | `#import "NSUndoManager.h"` | | class-dump 3.3.1 | `#import "NSUndoManager.h"` | | class-dump-z 0.2-0 | `#import <Foundation/NSUndoManager.h>` |

The headers generated by `-H` are not immediately usable because the imports usually points to non-existing .h files. There is even a page dedicated to fixing this problem.

class-dump-z now will see which library the external class comes from, and tries to make up a better .h file location to import.

> *Even with this change, headers generated by class-dump-z is generally not immediately usable either. This is usually because of name clash with Foundation and CoreFoundation objects. Most of the cases supplying the `-X NS,CF` flag to filter them out is enough to fix it.*

Hints file

When an Objective-C source is compiled, the class name in a method will be stripped, so in the dump you'll get `-(void)touchesBegan:(id)began withEvent:(id)event;` instead of `-(void)touchesBegan:(NSSet*)began withEvent:(UIEvent*)event;` Hints file is created to address this problem. When you pass the `-i` flag, a tab-delimited file will be created, which contains the type signature e.g. `-[UIGestureRecognizer touchesBegan:withEvent:] void id id` You can edit this file and change to the more precise signature, i.e. `-[UIGestureRecognizer touchesBegan:withEvent:] void NSSet* UIEvent*` then feed the same file with the `-i` flag. The updated signature will be reflected in the dump.

Miscellaneous features

- **Class attributes**, e.g. `__attribute__((visibility("hidden")))`.
- **Supports obscure type encodings**, e.g. the C99 `_Complex` types, Objective-C class ivars with ≥2 protocols, GC-invisible pointers, etc.
- **Put that star on the left or right**, I like `int* x` more, some people like `int *x` more. With the `-R` switch one can switch between the two styles easily.
- **Put a space or not after the +/-**. With the `-b` switch one can choose to use `- (void)hello` or `-(void)hello`.

What's missing

There are some features I don't find them immediately useful, so they are not supported yet.

Sort by inheritance (-I flag), Recursive dumping (-r)

These are not easy to implement and I find them not so useful so I left them out.