# Part IV Practices

The first 3 parts of this book have introduced the concepts, tools and theories of iOS reverse engineering, along with examples to give you a better understanding of them. I believe you have the same feeling that only if concepts, tools and theories are combined together can we get the best out reverse engineering.

So far, you may still feel unsatisfied with the fragmented and conservative examples. So in this part, we’ve prepared 4 original and systematic examples to show you the combination of concepts, tools and theories. They are:

* Characount for Notes 8
* Mark user specific emails as read automatically
* Save and share Sight in WeChat
* Detect and send iMessage

Now, welcome to the most splendid part of this book. Let’s enjoy the art of iOS reverse engineering!

# Chapter 7 Practice 1: Characount for Notes 8

## 7.1 Notes

I bet Notes App (hereafter referred to as Notes) is one of your most familiar iOS Apps. Its userinterface and functionality have experienced very few changes since iOS came out. The simplicity and convenience of Notes win my heart, all my secrets are sealed in it, as shown in figure 7-1.

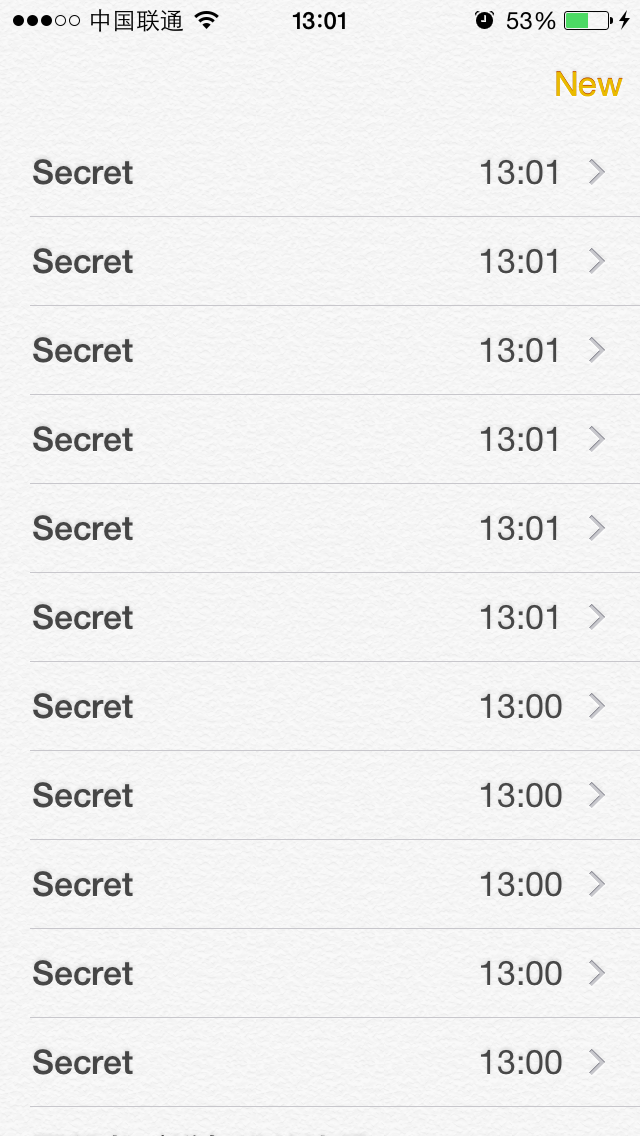


Figure 7- 1 Notes

Being a power user of Notes, not only do I save secrets in it, but also compose SMS or tweets in it. Since there is word limit on SMS and tweets, I really wish Notes can display each note’s character count as a reminder. DIY is a born spirit of reverse engineers, so I’ve developed Characount for Notes, which is one of my daily necessities on iOS 6. It’s not a difficult tweak, hence can be a stepping-stone for beginners like you. Our goal in this chapter is to rewrite Characount for Notes on iOS 8, and all the following operations are performed on iPhone 5, iOS 8.1.

## 7.2 Tweak prototyping

On iOS 8, the original note browsing view looks like figure 7-2.

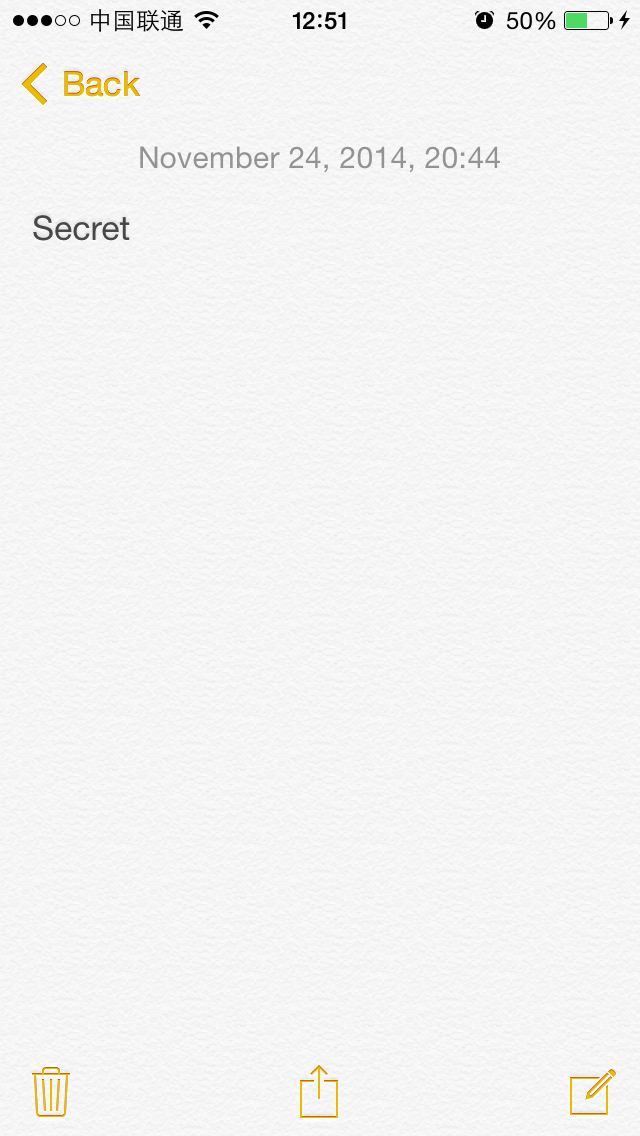


Figure 7- 2 Note browsing view on iOS 8

If we’re to choose a place to display the character count of this note, where do you think looks better? If you used to be an iOS 6 user, do you remember that each note has a centered title as shown in figure 7-3?

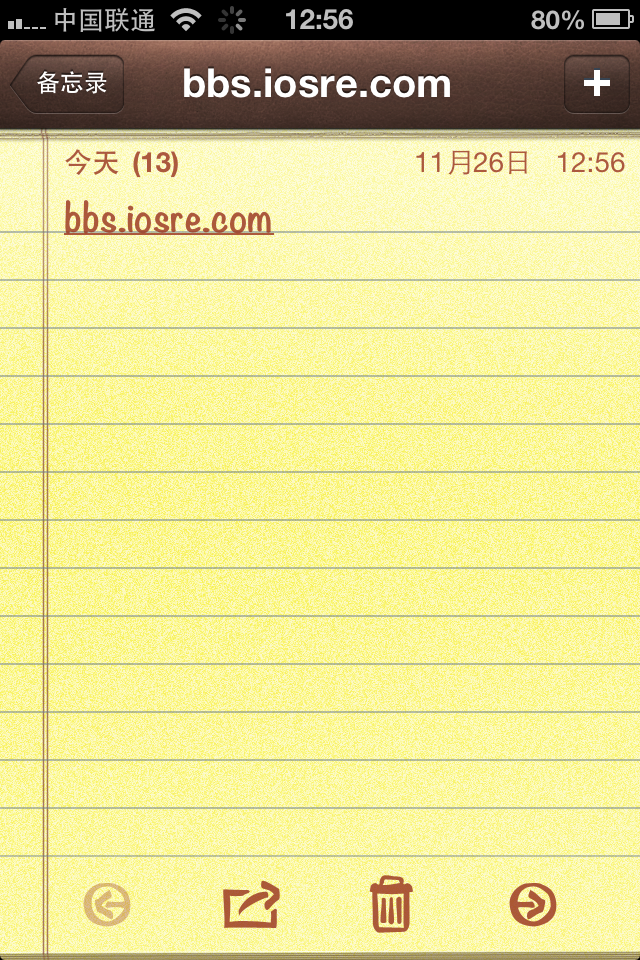


Figure 7- 3 Note browsing view on iOS 6

However, Notes on iOS 8 has removed the title, leaving a blank navigation bar. Why don’t we just display the character count here, as shown in figure 7-4?



Figure 7- 4 Note browsing view with a title

It looks good! So, what exactly should we do to make Notes look like this? Hope you remember the saying in chapter 5 that everything you see on iOS is an object. Keep that in mind and think with me:

(1) Every note is an object, and note browsing view contains the content and modification time of a note object. Since note browsing view is a subclass of UIView, we can trace back to its view controller via nextResponder, and further access all note concerned data via its view controller according to MVC design pattern. With the note data, we can initialize the character count when this view appears.

(2) While we are editing a note, a “Done” button will appear on the right side of the navigation bar, as shown in figure 7-5.

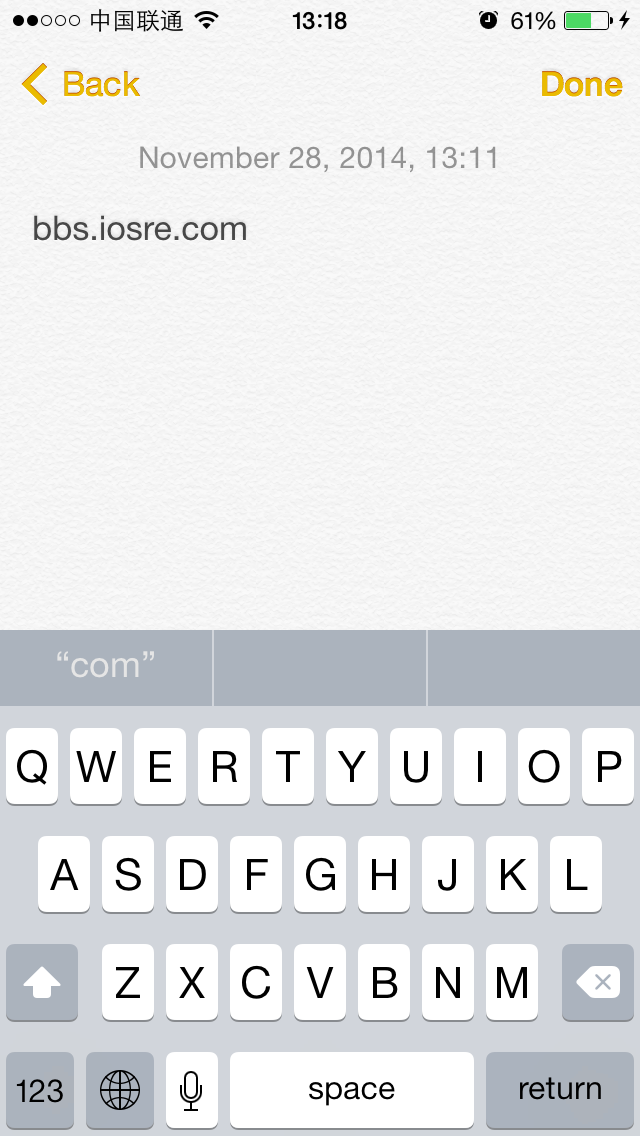


Figure 7- 5 “Done” button

After tapping “Done”, the current note is saved. This phenomenon indicates that a note is not saved in real time during editing, or we just don’t need this button at all. Of course, character count changes instantly with the editing content would be the ideal visual effect, so to accomplish this goal, we need to find a specific method which monitors the changes of the current note. In addition, we should be able to get the character count of this note and update the title just in time within this method. Because this kind of methods are usually defined in protocols, we should keep an eye on protocols in Notes.

(3) Suppose we can get the current note’s character count, how do we put it on the navigation bar? Commonly, the note browsing view controller should inherits from UIViewController, which possesses a “title” property. So, “setTitle:” is the answer.

If we managed to solve these 3 problems, there’ll be no more technical difficulties for Characount for Notes. Code speaks louder than words, let’s move it!

### 7.2.1 Locate Notes’ executable

There’s no Notes.app under /Applications/ at all. Besides searching blindly, what else can we do to locate its executable? Do you still remember the trick of getting an App’s path in dumpdecrypted section? Yeah, it’s ps command again: first close all Apps, then open Notes and ssh to iOS to list all system processes with ps:

FunMaker-5:~ root# ps -e | grep /Applications/

592 ?? 0:37.70 /Applications/MobileMail.app/MobileMail

761 ?? 0:02.78 /Applications/MessagesNotificationViewService.app/MessagesNotificationViewService

1807 ?? 0:00.55 /private/var/db/stash/\_.29LMeZ/Applications/MobileSafari.app/webbookmarksd

2016 ?? 0:05.23 /Applications/InCallService.app/InCallService

2619 ?? 0:02.66 /Applications/MobileSMS.app/MobileSMS

2672 ?? 0:01.20 /Applications/MobileNotes.app/MobileNotes

2678 ttys000 0:00.01 grep /Applications/

Among those processes, MobileNotes attracts us most. How to verify our guess? We can simply kill it and see whether Notes quit.

FunMaker-5:~ root# killall MobileNotes

Notes has quit as we expected, which clearly means that “/Applications/MobileNotes.app/MobileNotes” is Notes’ executable. Meanwhile, we’ve discovered some Apps that’re running in the background. Copy MobileNotes to OSX and get ready to class-dump it.

### 7.2.2 class-dump MobileNotes’ headers

Because Notes is a stock App, its executable is not encrypted, enabling us to class-dump it directly:

snakeninnys-MacBook:~ snakeninny$ class-dump -S -s -H /Users/snakeninny/Code/iOSSystemBinaries/8.1\_iPhone5/MobileNotes.app/MobileNotes -o /Users/snakeninny/Code/iOSPrivateHeaders/8.1/MobileNotes

We’ve got 88 headers in total. Let’s take a brief look to see what we can discover, as shown in figure 7-6.

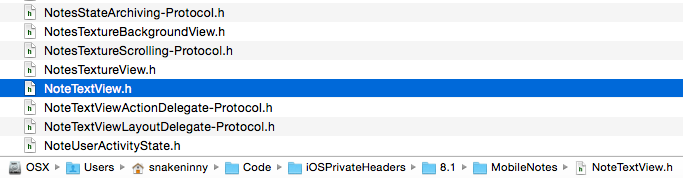


Figure 7- 6 Headers of Notes

Do you see the selected file in figure 7-6? I am not sure if it is a key clue of this chapter for now, but we’ll see.

### 7.2.3 Find the controller of note browsing view using Cycript

Again, recursiveDescription makes our days:

FunMaker-5:~ root# cycript -p MobileNotes

cy# ?expand

expand == true

cy# [[UIApp keyWindow] recursiveDescription]

@"<UIWindow: 0x17688db0; frame = (0 0; 320 568); gestureRecognizers = <NSArray: 0x17689620>; layer = <UIWindowLayer: 0x17688fc0>>

| <UILayoutContainerView: 0x175bb880; frame = (0 0; 320 568); autoresize = W+H; layer = <CALayer: 0x175bb900>>

| | <UILayoutContainerView: 0x17699350; frame = (0 0; 320 568); clipsToBounds = YES; gestureRecognizers = <NSArray: 0x1769cf60>; layer = <CALayer: 0x17699530>>

| | | <UINavigationTransitionView: 0x176564c0; frame = (0 0; 320 568); clipsToBounds = YES; autoresize = W+H; layer = <CALayer: 0x17658ec0>>

| | | | <UIViewControllerWrapperView: 0x176d13b0; frame = (0 0; 320 568); layer = <CALayer: 0x176d1530>>

| | | | | <UILayoutContainerView: 0x1769dd80; frame = (0 0; 320 568); clipsToBounds = YES; gestureRecognizers = <NSArray: 0x176a16f0>; layer = <CALayer: 0x1769de00>>

| | | | | | <UINavigationTransitionView: 0x1769ebb0; frame = (0 0; 320 568); clipsToBounds = YES; autoresize = W+H; layer = <CALayer: 0x1769ec40>>

| | | | | | | <UIViewControllerWrapperView: 0x175109e0; frame = (0 0; 320 568); layer = <CALayer: 0x175109b0>>

| | | | | | | | <NotesBackgroundView: 0x175ee3e0; frame = (0 0; 320 568); gestureRecognizers = <NSArray: 0x17510a70>; layer = <CALayer: 0x175ee580>>

| | | | | | | | | <NotesTextureBackgroundView: 0x175ee5b0; frame = (0 0; 320 568); clipsToBounds = YES; layer = <CALayer: 0x175ee630>>

| | | | | | | | | | <NotesTextureView: 0x175ee940; frame = (0 -64; 320 640); layer = <CALayer: 0x175ee9c0>>

| | | | | | | | | <NoteContentLayer: 0x176c5110; frame = (0 0; 320 568); layer = <CALayer: 0x176ca850>>

| | | | | | | | | | <UIView: 0x175f2130; frame = (16 0; 288 0); hidden = YES; layer = <CALayer: 0x175dd2b0>>

| | | | | | | | | | <NotesScrollView: 0x175f2a10; baseClass = UIScrollView; frame = (0 0; 320 568); clipsToBounds = YES; gestureRecognizers = <NSArray: 0x175f1b70>; layer = <CALayer: 0x175f28d0>; contentOffset: {0, -64}; contentSize: {320, 460}>

| | | | | | | | | | | <UIView: 0x175f09a0; frame = (0 0; 320 0); layer = <CALayer: 0x175f2790>>

| | | | | | | | | | | <UIView: 0x175f27e0; frame = (0 0; 0 460); layer = <CALayer: 0x175f2850>>

| | | | | | | | | | | <NoteDateLabel: 0x175f3400; baseClass = UILabel; frame = (69 5.5; 182 18); text = 'November 24, 2014, 20:44'; userInteractionEnabled = NO; layer = <\_UILabelLayer: 0x175f3560>>

| | | | | | | | | | | <NoteTextView: 0x175ee3e0; baseClass = \_UICompatibilityTextView; frame = (6 28; 308 418); text = 'Secret'; clipsToBounds = YES; gestureRecognizers = <NSArray: 0x176c7ed0>; layer = <CALayer: 0x176d88e0>; contentOffset: {0, 0}; contentSize: {308, 52}>

……

Look! There is a NoteTextView with the keyword “Secret”. Call nextResponder continuously until we get its controller:

cy# [#0x175ee3e0 nextResponder]

#"<NotesScrollView: 0x17d307c0; baseClass = UIScrollView; frame = (0 0; 320 568); clipsToBounds = YES; gestureRecognizers = <NSArray: 0x17e502a0>; layer = <CALayer: 0x17d30b60>; contentOffset: {0, -64}; contentSize: {320, 251}>"

cy# [#0x17d307c0 nextResponder]

#"<NoteContentLayer: 0x17e505b0; frame = (0 0; 320 568); layer = <CALayer: 0x17e50470>>"

cy# [#0x17e505b0 nextResponder]

#"<NotesBackgroundView: 0x17e52320; frame = (0 0; 320 568); gestureRecognizers = <NSArray: 0x17d0c940>; layer = <CALayer: 0x17e522f0>>"

cy# [#0x17e52320 nextResponder]

#"<NotesDisplayController: 0x17edc340>"

Okay, NoteDisplayController is the one. Let’s see if setTitle: really changes the title of note browsing view:

cy# [#0x17edc340 setTitle:@"Characount = Character count"]

The UI after setTitle: is shown in figure 7-7.

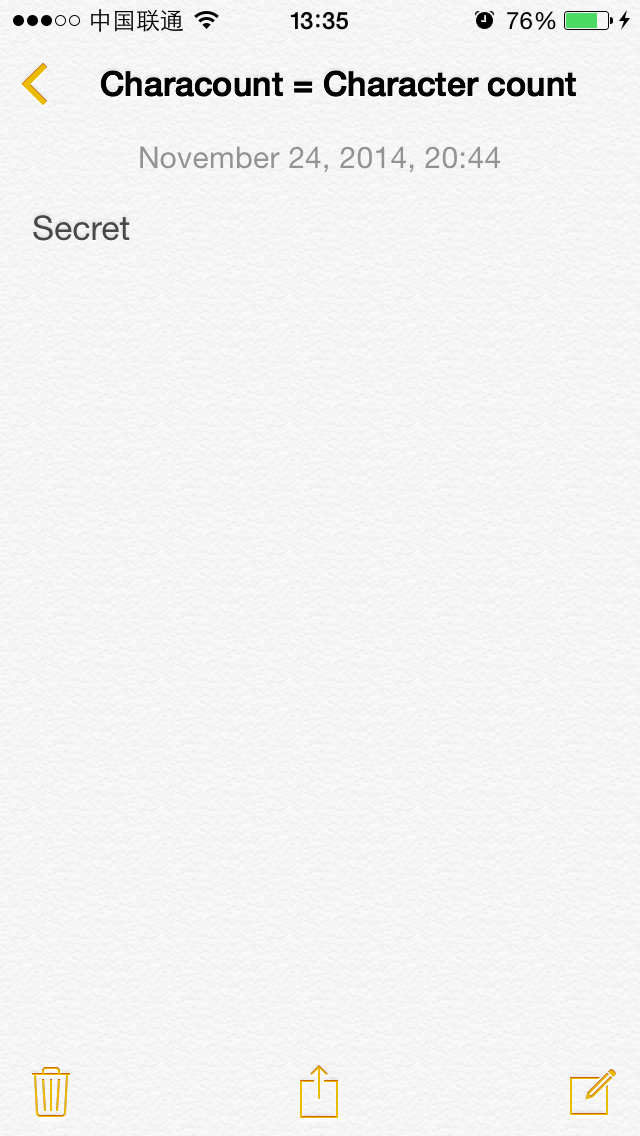


Figure 7- 7 UI After setTitle:

Neet! Mission 1, completed!

### 7.2.4 Get the current note object from NoteDisplayController

Strike while the iron is hot, let’s overview NoteDisplayController.h.

@interface NotesDisplayController : UIViewController <NoteContentLayerDelegate, UIActionSheetDelegate, AFContextProvider, UIPopoverPresentationControllerDelegate, UINavigationControllerDelegate, UIImagePickerControllerDelegate, NotesQuickLookActivityItemDelegate, ScrollViewKeyboardResizerDelegate, NSUserActivityDelegate, NotesStateArchiving>

{

……

@property(nonatomic, getter=isVisible) BOOL visible; // @synthesize visible=\_visible;

- (void)loadView;

@property(retain, nonatomic) NoteObject \*note; // @synthesize note=\_note;

……

}

After going over this large header, we’ve found a property of NoteObject type. Since a note is exactly an object, NoteObject seems to be too obvious to believe… Hehe, let’s print it in Cycript:

cy# [#0x17edc340 note]

#'<NoteObject: 0x176aa170> (entity: Note; id: 0x176a9040 <x-coredata://4B88CC7C-7A5F-4F15-9275-53C6D0ABE0C3/Note/p15> ; data: {\n attachments = (\n );\n author = nil;\n body = "0x176a8b20 <x-coredata://4B88CC7C-7A5F-4F15-9275-53C6D0ABE0C3/NoteBody/p15>";\n containsCJK = 0;\n contentType = 0;\n creationDate = "2014-11-24 05:00:59 +0000";\n deletedFlag = 0;\n externalFlags = 0;\n externalSequenceNumber = 0;\n externalServerIntId = "-4294967296";\n guid = "781B6C87-2855-4512-8864-50618754333A";\n integerId = 3865;\n isBookkeepingEntry = 0;\n modificationDate = "2014-11-24 12:44:08 +0000";\n serverId = nil;\n store = "0x175a2b60 <x-coredata://4B88CC7C-7A5F-4F15-9275-53C6D0ABE0C3/Store/p1>";\n summary = nil;\n title = Secret;\n})'

Needless to say, NoteObject is exactly the current note. Each field in the description is explicit, let’s take a look at its header:

@interface NoteObject : NSManagedObject

{

}

- (BOOL)belongsToCollection:(id)arg1;

@property(nonatomic) unsigned long long sequenceNumber;

- (BOOL)containsAttachments;

@property(retain, nonatomic) NSString \*externalContentRef;

@property(retain, nonatomic) NSData \*externalRepresentation;

@property(readonly, nonatomic) BOOL hasValidServerIntId;

@property(nonatomic) long long serverIntId;

@property(nonatomic) unsigned long long flags;

@property(readonly, nonatomic) NSURL \*noteId;

@property(readonly, nonatomic) BOOL isBeingMarkedForDeletion;

@property(readonly, nonatomic) BOOL isMarkedForDeletion;

- (void)markForDeletion;

@property(nonatomic) BOOL isPlainText;

- (id)contentAsPlainTextPreservingNewlines;

@property(readonly, nonatomic) NSString \*contentAsPlainText;

@property(retain, nonatomic) NSString \*content;

// Remaining properties

@property(retain, nonatomic) NSSet \*attachments; // @dynamic attachments;

@property(retain, nonatomic) NSString \*author; // @dynamic author;

@property(retain, nonatomic) NoteBodyObject \*body; // @dynamic body;

@property(retain, nonatomic) NSNumber \*containsCJK; // @dynamic containsCJK;

@property(retain, nonatomic) NSNumber \*contentType; // @dynamic contentType;

@property(retain, nonatomic) NSDate \*creationDate; // @dynamic creationDate;

@property(retain, nonatomic) NSNumber \*deletedFlag; // @dynamic deletedFlag;

@property(retain, nonatomic) NSNumber \*externalFlags; // @dynamic externalFlags;

@property(retain, nonatomic) NSNumber \*externalSequenceNumber; // @dynamic externalSequenceNumber;

@property(retain, nonatomic) NSNumber \*externalServerIntId; // @dynamic externalServerIntId;

@property(readonly, retain, nonatomic) NSString \*guid; // @dynamic guid;

@property(retain, nonatomic) NSNumber \*integerId; // @dynamic integerId;

@property(retain, nonatomic) NSNumber \*isBookkeepingEntry; // @dynamic isBookkeepingEntry;

@property(retain, nonatomic) NSDate \*modificationDate; // @dynamic modificationDate;

@property(retain, nonatomic) NSString \*serverId; // @dynamic serverId;

@property(retain, nonatomic) NoteStoreObject \*store; // @dynamic store;

@property(retain, nonatomic) NSString \*summary; // @dynamic summary;

@property(retain, nonatomic) NSString \*title; // @dynamic title;

@end

Great! Lots of properties indicate that NoteObject is a very standard model. How do we get its text? Among its properties, we can see a possible property named contentAsPlainText. Let’s see what it is:

cy# [#0x176aa170 contentAsPlainText]

@"Secret"

For further confirmation, let’s change the text of this note and add a picture, as shown in figure 7-8.



Figure 7- 8 Change this note

Then call contentAsPlainText again:

cy# [#0x176aa170 contentAsPlainText]

@"bbs.iosre.com"

Now we’re certain that this method can correctly return the text of the current note. With a further length method, we’re able to get the character count of this note:

cy# [[#0x176aa170 contentAsPlainText] length]

13

We’re almost done.

### 7.2.5 Find a method to monitor note text changes in real time

At the beginning of this chapter we’ve mentioned that “this kind of methods are usually defined in protocols”. Because both setTitle: and NoteObject are found in NotesDisplayController.h, if we can find the “monitor” method inside this header too, our code will be greatly simplified. Open NotesDisplayController.h and look what protocols it has implemented.

@interface NotesDisplayController : UIViewController <NoteContentLayerDelegate, UIActionSheetDelegate, AFContextProvider, UIPopoverPresentationControllerDelegate, UINavigationControllerDelegate, UIImagePickerControllerDelegate, NotesQuickLookActivityItemDelegate, ScrollViewKeyboardResizerDelegate, NSUserActivityDelegate, NotesStateArchiving>

……

@end

Among those protocols, UIActionSheetDelegate, UIPopoverPresentationControllerDelegate, UINavigationControllerDelegate and UIImagePickerControllerDelegate are all documented, they have nothing to do with the changes of the current note, hence can be ignored. The remaining ones, i.e. NoteContentLayerDelegate, AFContextProvider, NotesQuickLookActivityItemDelegate, ScrollViewKeyboardResizerDelegate, NSUserActivityDelegate and NotesStateArchiving are worth attention, we should inspect them one by one. Let’s start with NoteContentLayerDelegate-Protocol.h:

@protocol NoteContentLayerDelegate <NSObject>

- (BOOL)allowsAttachmentsInNoteContentLayer:(id)arg1;

- (BOOL)canInsertImagesInNoteContentLayer:(id)arg1;

- (void)insertImageInNoteContentLayer:(id)arg1;

- (BOOL)isNoteContentLayerVisible:(id)arg1;

- (BOOL)noteContentLayer:(id)arg1 acceptContentsFromPasteboard:(id)arg2;

- (BOOL)noteContentLayer:(id)arg1 acceptStringIncreasingContentLength:(id)arg2;

- (BOOL)noteContentLayer:(id)arg1 canHandleLongPressOnElement:(id)arg2;

- (void)noteContentLayer:(id)arg1 containsCJK:(BOOL)arg2;

- (void)noteContentLayer:(id)arg1 contentScrollViewWillBeginDragging:(id)arg2;

- (void)noteContentLayer:(id)arg1 didChangeContentSize:(struct CGSize)arg2;

- (void)noteContentLayer:(id)arg1 handleLongPressOnElement:(id)arg2 atPoint:(struct CGPoint)arg3;

- (void)noteContentLayer:(id)arg1 setEditing:(BOOL)arg2 animated:(BOOL)arg3;

- (void)noteContentLayerContentDidChange:(id)arg1 updatedTitle:(BOOL)arg2;

- (BOOL)noteContentLayerShouldBeginEditing:(id)arg1;

@optional

- (void)noteContentLayerKeyboardDidHide:(id)arg1;

@end

2 methods are quite suspecious, they’re noteContentLayer:didChangeContentSize: and noteContentLayerContentDidChange:updatedTitle:. While we are editing a note, the content and size of it are indeed changing, thus those 2 methods may be called when changes occur, and actually they’re implemented in NotesDisplayController.h. Let’s use LLDB to make sure they’re called when a note changes.

Attach to MobileNotes with LLDB, and check its ASLR offset:

(lldb) image list -o -f

[ 0] 0x00035000 /private/var/db/stash/\_.29LMeZ/Applications/MobileNotes.app/MobileNotes(0x0000000000039000)

[ 1] 0x00197000 /Library/MobileSubstrate/MobileSubstrate.dylib(0x0000000000197000)

[ 2] 0x06db3000 /Users/snakeninny/Library/Developer/Xcode/iOS DeviceSupport/8.1 (12B411)/Symbols/System/Library/Frameworks/QuickLook.framework/QuickLook

……

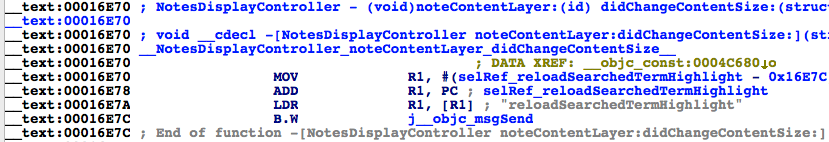
The ASLR offset is 0x35000. Drag and drop MobileNotes into IDA, then check the base addresses of [NotesDisplayController noteContentLayer:didChangeContentSize:] and [NotesDisplayController noteContentLayerContentDidChange:updatedTitle:] after the initial analysis, as shown in figure 7-9 and figure 7-10.

Figure7- 9 [NotesDisplayController noteContentLayer:didChangeContentSize:]

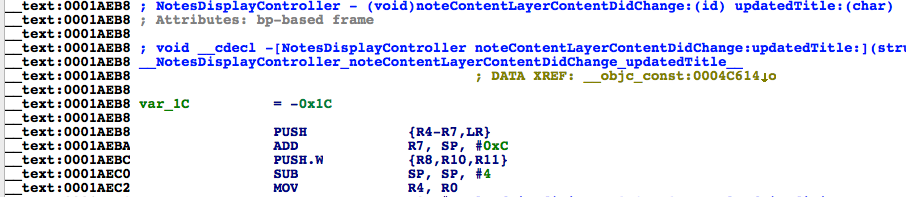


Figure7- 10 [NotesDisplayController noteContentLayerContentDidChange:updatedTitle:]

The base addresses are 0x16E70 and 0x1AEB8 respectively, so breakpoints should be set at 0x4BE70 and 0x4FEB8. Then try to edit a note and see whether these breakpoints are triggered:

(lldb) br s -a 0x4BE70

Breakpoint 1: where = MobileNotes`\_\_\_lldb\_unnamed\_function382$$MobileNotes, address = 0x0004be70

(lldb) br s -a 0x4FEB8

Breakpoint 2: where = MobileNotes`\_\_\_lldb\_unnamed\_function458$$MobileNotes, address = 0x0004feb8

Great eyes see alike: These two breakpoints are hit a lot! The reason a protocol method gets called is generally that the corresponding event mentioned in the method name happened. And the thing that triggers the event is usually the method’s arguments. In this case, [NotesDisplayController noteContentLayer:didChangeContentSize:] and [NotesDisplayController noteContentLayerContentDidChange:updatedTitle:] are called because didChangeContentSize and ContentDidChange events happened, and content itself is probably the arguments of both methods. Let’s verify our guess in LLDB.

(lldb) br com add 1

Enter your debugger command(s). Type 'DONE' to end.

> po $r2

> c

> DONE

(lldb) br com add 2

Enter your debugger command(s). Type 'DONE' to end.

> po $r2

> c

> DONE

(lldb) c

We can see quite a few occurrences of NoteContentLayer:

Process 24577 resuming

Command #2 'c' continued the target.

<NoteContentLayer: 0x14ecdf50; frame = (0 0; 320 568); animations = { bounds.origin=<CABasicAnimation: 0x16fee090>; bounds.size=<CABasicAnimation: 0x16fee4a0>; position=<CABasicAnimation: 0x16fee500>; }; layer = <CALayer: 0x14eca900>>

Process 24577 resuming

Command #2 'c' continued the target.

<NoteContentLayer: 0x14ecdf50; frame = (0 0; 320 568); animations = { bounds.origin=<CABasicAnimation: 0x16fee090>; bounds.size=<CABasicAnimation: 0x16fee4a0>; position=<CABasicAnimation: 0x16fee500>; }; layer = <CALayer: 0x14eca900>>

Process 24577 resuming

Command #2 'c' continued the target.

<NoteContentLayer: 0x14ecdf50; frame = (0 0; 320 568); layer = <CALayer: 0x14eca900>>

Process 24577 resuming

Command #2 'c' continued the target.

If NoteContentLayer comes, can NoteContent be far behind? Let’s search in NoteContentLayer.h for NoteContent:

@interface NoteContentLayer : UIView <NoteTextViewActionDelegate, NoteTextViewLayoutDelegate, UITextViewDelegate>

……

@property(retain, nonatomic) NoteTextView \*textView; // @synthesize textView=\_textView;

……

@end

There’s a property of NoteTextView type in NoteContentLayer. In the beginning of this chapter, we have printed the view hierarchy of note browsing view in Cycript, and found the note text was displayed right on a NoteTextView. So, let’s change the commands on the breakpoints and print NoteTextView:

(lldb) br com add 1

Enter your debugger command(s). Type 'DONE' to end.

> po [$r2 textView]

> c

> DONE

(lldb) br com add 2

Enter your debugger command(s). Type 'DONE' to end.

> po [$r2 textView]

> c

> DONE

Continue editing this note and keep watching the output. Our editing shows in the output in real time:

Process 24577 resuming

Command #2 'c' continued the target.

<NoteTextView: 0x15aace00; baseClass = \_UICompatibilityTextView; frame = (6 28; 308 209); text = 'Secre'; clipsToBounds = YES; gestureRecognizers = <NSArray: 0x14eddfc0>; layer = <CALayer: 0x14ee7da0>; contentOffset: {0, 0}; contentSize: {308, 52}>

Process 24577 resuming

Command #2 'c' continued the target.

<NoteTextView: 0x15aace00; baseClass = \_UICompatibilityTextView; frame = (6 28; 308 209); text = 'Secret'; clipsToBounds = YES; gestureRecognizers = <NSArray: 0x14eddfc0>; layer = <CALayer: 0x14ee7da0>; contentOffset: {0, 0}; contentSize: {308, 52}>

One last step is to get “text” from NoteTextView. Open NoteTextView.h:

@interface NoteTextView : \_UICompatibilityTextView <UIGestureRecognizerDelegate>

{

id <NoteTextViewActionDelegate> \_actionDelegate;

id <NoteTextViewLayoutDelegate> \_layoutDelegate;

……

}

……

@property(nonatomic) \_\_weak id <NoteTextViewActionDelegate> actionDelegate; // @synthesize actionDelegate=\_actionDelegate;

……

@property(nonatomic) \_\_weak id <NoteTextViewLayoutDelegate> layoutDelegate; // @synthesize layoutDelegate=\_layoutDelegate;

……

@end

There’s not much content in this header, and there’re only 2 delegates with the keyword “text”. Obviously, delegates don’t return NSString objects. If we cannot get text in NoteTextView, it gets to be in NoteTextView’s super class. Open \_UICompatibilityTextView then:

@interface \_UICompatibilityTextView : UIScrollView <UITextLinkInteraction, UITextInput>

……

@property(nonatomic) int textAlignment;

@property(copy, nonatomic) NSString \*text;

- (BOOL)hasText;

@property(retain, nonatomic) UIColor \*textColor;

@property(retain, nonatomic) UIFont \*font;

@property(copy, nonatomic) NSAttributedString \*attributedText;

……

OK, here comes NSString \*text. Let’s use LLDB for a final confirmation:

(lldb) br com add 1

Enter your debugger command(s). Type 'DONE' to end.

> po [[$r2 textView] text]

> c

> DONE

(lldb) br com add 2

Enter your debugger command(s). Type 'DONE' to end.

> po [[$r2 textView] text]

> c

> DONE

Secret

Process 24577 resuming

Command #2 'c' continued the target.

Secret i

Process 24577 resuming

Command #2 'c' continued the target.

By now, we’ve successfully found 2 methods to monitor note text changes in real time, you can choose either of them, and [NotesDisplayController noteContentLayerContentDidChange:updatedTitle:] is my choice. All 3 previous problems are solved, iOS reverse engineering is way easier than you originally thought, isn’t it?

## 7.3 Result interpretation

The mission of this chapter is to reverse a stock App, Notes. We’ve successfully prototyped the tweak with only Cycript and LLDB, and actually we can replace LLDB with Theos too. You may call it luck and it’s true that reverse engineering depends on fortune. To rewrite Characount for Notes 8, the general thoughts are as follows.

#### 1. Find a proper location on UI and a method to display the character count

Upgrading from iOS 6 to iOS 8 eliminates Notes’ title, where is a good place to display the character count. In this chapter, we’ve cut into the code from the note browsing view and got NoteDisplayController with Cycript, therefore managed to solve the 1st problem.

#### 2. Browse the class-dump headers and find methods in controller to access model

Accessing model via controller conforms to MVC design pattern, which Apple made Apps should apply. Therefore, NoteDisplayController should be able to access note objects. By just looking through headers and examine some suspicious properties with Cycript, we’ve got NoteObject, thus got the character count of a note.

#### 3. Find protocol methods to monitor note text changes in real time

Event related methods with keywords like "did" or "will" are often defined in protocols. Due to the high readability of Objective-C methods’ names, we didn’t use IDA or LLDB to find methods that meet our needs, but instead went over all headers with the keyword “protocol”. With a 1st round filtering by header names and a 2nd round filtering by LLDB, we’ve found our target methods. This is the charm of reverse engineering, regardless of fortune or guess.

## 7.4 Tweak writing

This example is relatively easy, all operations can be done inside the class NotesDisplayController.

### 7.4.1 Create tweak project "CharacountforNotes8" using Theos

The Theos commands are as follows:

snakeninnys-MacBook:Code snakeninny$ /opt/theos/bin/nic.pl

NIC 2.0 - New Instance Creator

------------------------------

[1.] iphone/application

[2.] iphone/cydget

[3.] iphone/framework

[4.] iphone/library

[5.] iphone/notification\_center\_widget

[6.] iphone/preference\_bundle

[7.] iphone/sbsettingstoggle

[8.] iphone/tool

[9.] iphone/tweak

[10.] iphone/xpc\_service

Choose a Template (required): 9

Project Name (required): CharacountForNotes8

Package Name [com.yourcompany.characountfornotes8]: com.naken.characountfornotes8

Author/Maintainer Name [snakeninny]: snakeninny

[iphone/tweak] MobileSubstrate Bundle filter [com.apple.springboard]: com.apple.mobilenotes

[iphone/tweak] List of applications to terminate upon installation (space-separated, '-' for none) [SpringBoard]: MobileNotes

Instantiating iphone/tweak in characountfornotes8/...

Done.

### 7.4.2 Compose CharacountForNotes8.h

The finalized CharacountForNotes8.h looks like this:

@interface NoteObject : NSObject

@property (readonly, nonatomic) NSString \*contentAsPlainText;

@end

@interface NoteTextView : UIView

@property (copy, nonatomic) NSString \*text;

@end

@interface NoteContentLayer : UIView

@property (retain, nonatomic) NoteTextView \*textView;

@end

@interface NotesDisplayController : UIViewController

@property (retain, nonatomic) NoteContentLayer \*contentLayer;

@property (retain, nonatomic) NoteObject \*note;

@end

This header is composed by picking snippets from other class-dump headers. The existence of this header is simply for avoiding any warnings or errors when compiling the tweak.

### 7.4.3 Edit Tweak.xm

The finalized Tweak.xm looks like this:

#import "CharacountForNotes8.h"

%hook NotesDisplayController

- (void)viewWillAppear:(BOOL)arg1 // Initialze title

{

%orig;

NSString \*content = self.note.contentAsPlainText;

NSString \*contentLength = [NSString stringWithFormat:@"%lu", (unsigned long)[content length]];

self.title = contentLength;

}

- (void)viewDidDisappear:(BOOL)arg1 // Reset title

{

%orig;

self.title = nil;

}

- (void)noteContentLayerContentDidChange:(NoteContentLayer \*)arg1 updatedTitle:(BOOL)arg2 // Update title

{

%orig;

NSString \*content = self.contentLayer.textView.text;

NSString \*contentLength = [NSString stringWithFormat:@"%lu", (unsigned long)[content length]];

self.title = contentLength;

}

%end

### 7.4.4 Edit Makefile and control files

The finalized Makefile looks like this:

THEOS\_DEVICE\_IP = iOSIP

ARCHS = armv7 arm64

TARGET = iphone:latest:8.0

include theos/makefiles/common.mk

TWEAK\_NAME = CharacountForNotes8

CharacountForNotes8\_FILES = Tweak.xm

include $(THEOS\_MAKE\_PATH)/tweak.mk

after-install::

install.exec "killall -9 MobileNotes"

The finalized control looks like this:

Package: com.naken.characountfornotes8

Name: CharacountForNotes8

Depends: mobilesubstrate, firmware (>= 8.0)

Version: 1.0

Architecture: iphoneos-arm

Description: Add a character count to Notes

Maintainer: snakeninny

Author: snakeninny

Section: Tweaks

Homepage: http://bbs.iosre.com

### 7.4.5 Test

After packaging and installing Characount for Notes 8, let’s test it by editing a random note and see if the character count changes in real time, as shown in figure 7-11 to figure 7-17.

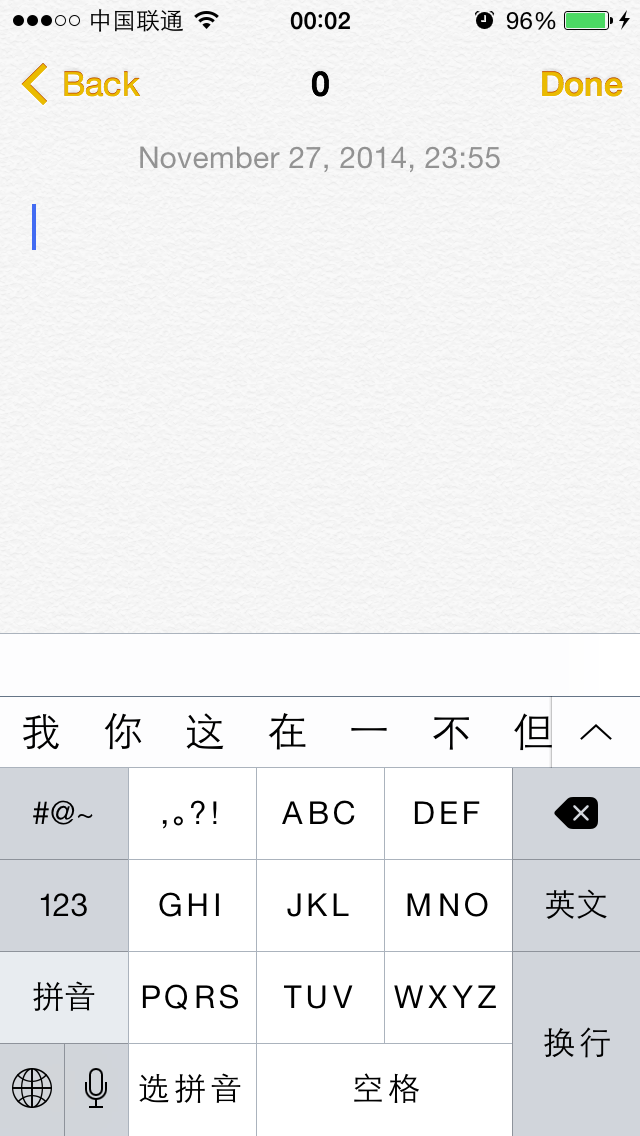


Figure 7- 11 Characount for Notes 8

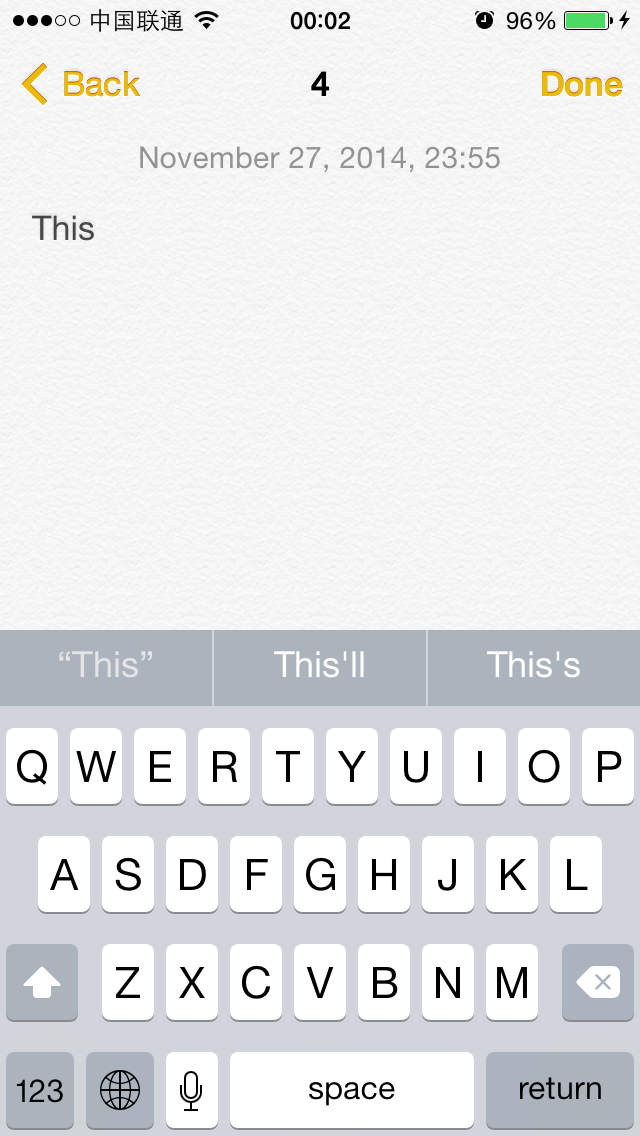


Figure 7- 12 Characount for Notes 8

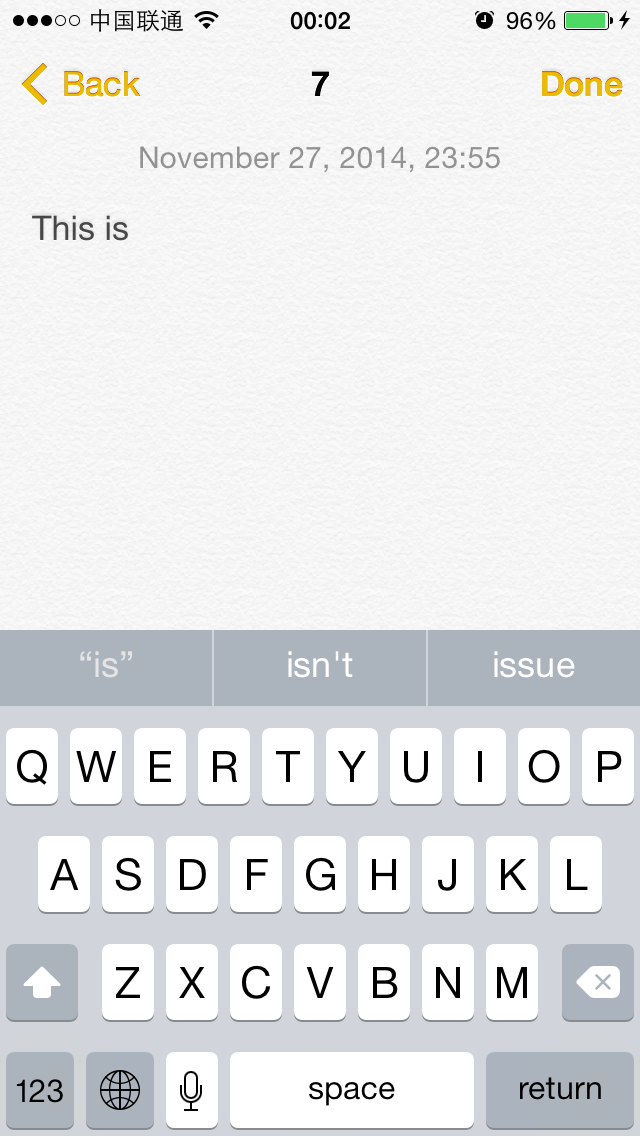


Figure 7- 13 Characount for Notes 8

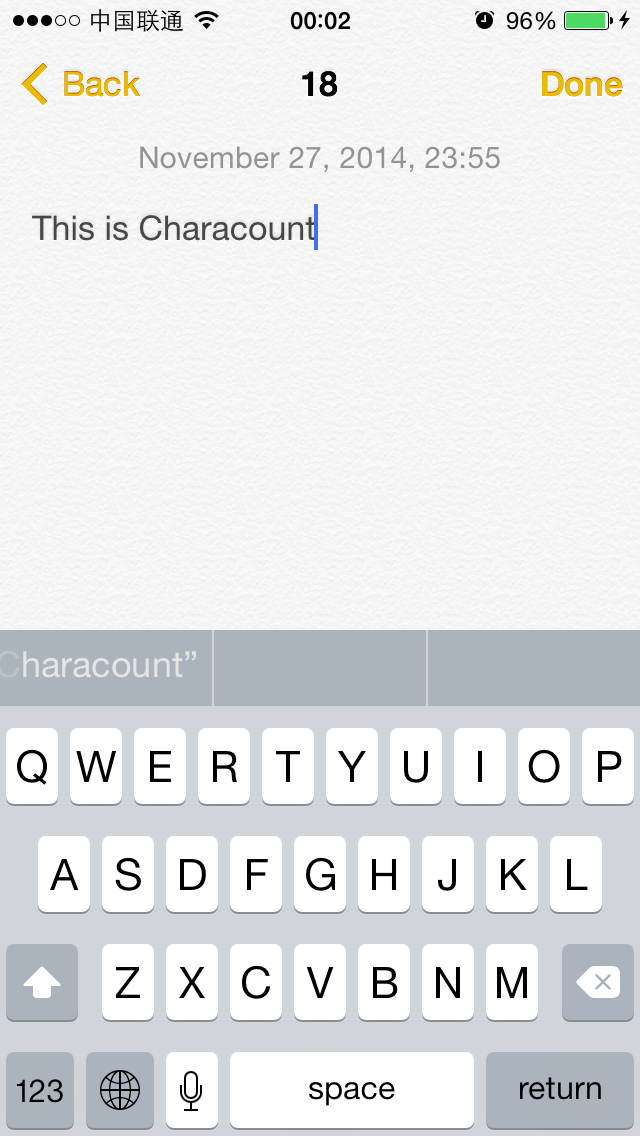


Figure 7- 14 Characount for Notes 8

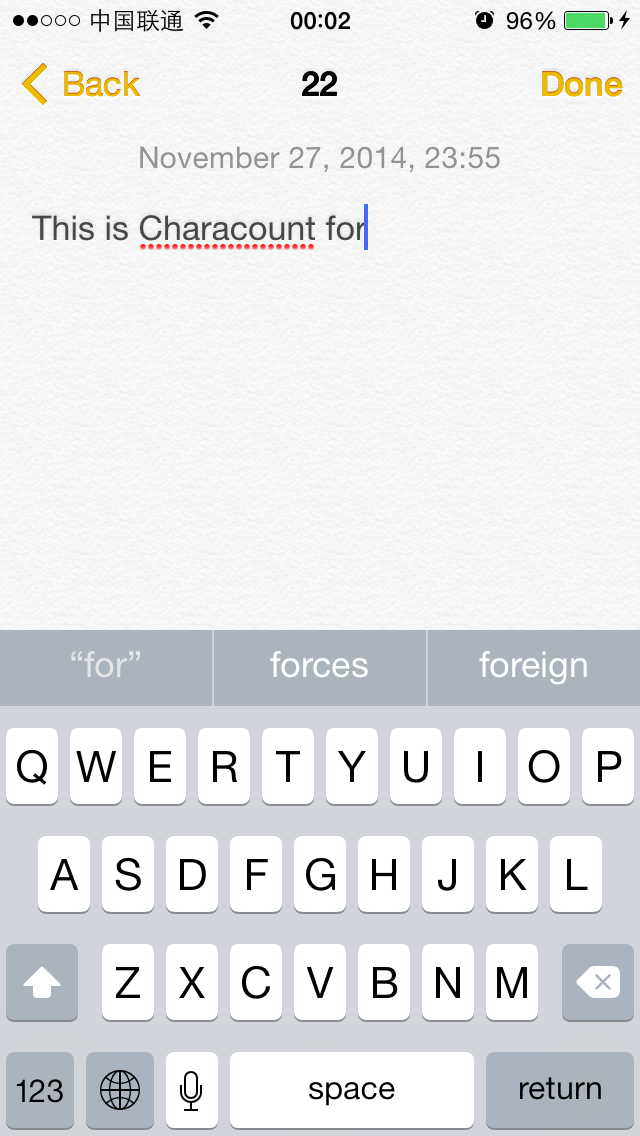


Figure 7- 15 Characount for Notes 8



Figure 7- 16 Characount for Notes 8

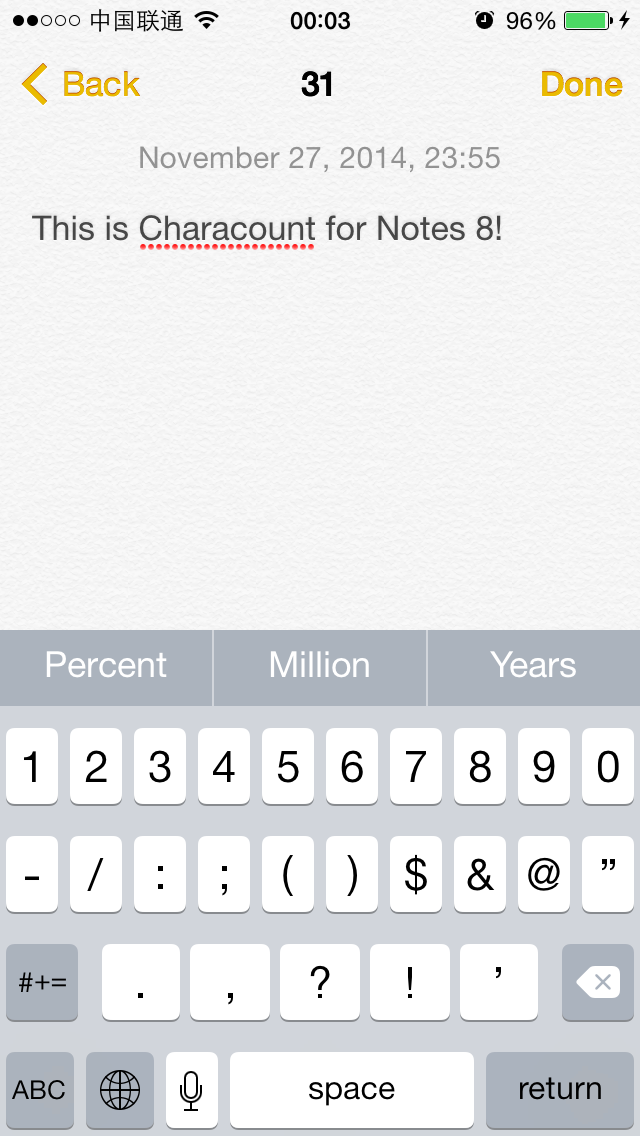


Figure 7- 17 Characount for Notes 8

It works as we expected.

## 7.5 Conclusion

As a veteran on iOS, Notes is simple yet popular, a great number of people use this App frequently in their daily lives. Characount for Notes 8 is so simple that we don’t even need advanced reverse engineering tools to finish the whole project, I hope you don’t have difficulty reading this chapter. It’s energy-and-time-consuming to learn assembly-level reverse engineering when you are not familiar with IDA and LLDB, I suggest you beginners carry out some simple reverse engineering projects just like the example in this chapter. In this way, not only can you form a thinking pattern of reverse engineering, but also gain a sense of achievement, so why not get your hands dirty right now?