XcodeによるiPhone/Mac アプリの制作 (2)

Objective-C++ programming

IPv4/v6アドレス範囲の算出ツールを制作する

create Tab Bar Application

株式会社あいはら 山田 泰司 <taiji@aihara.co.jp>

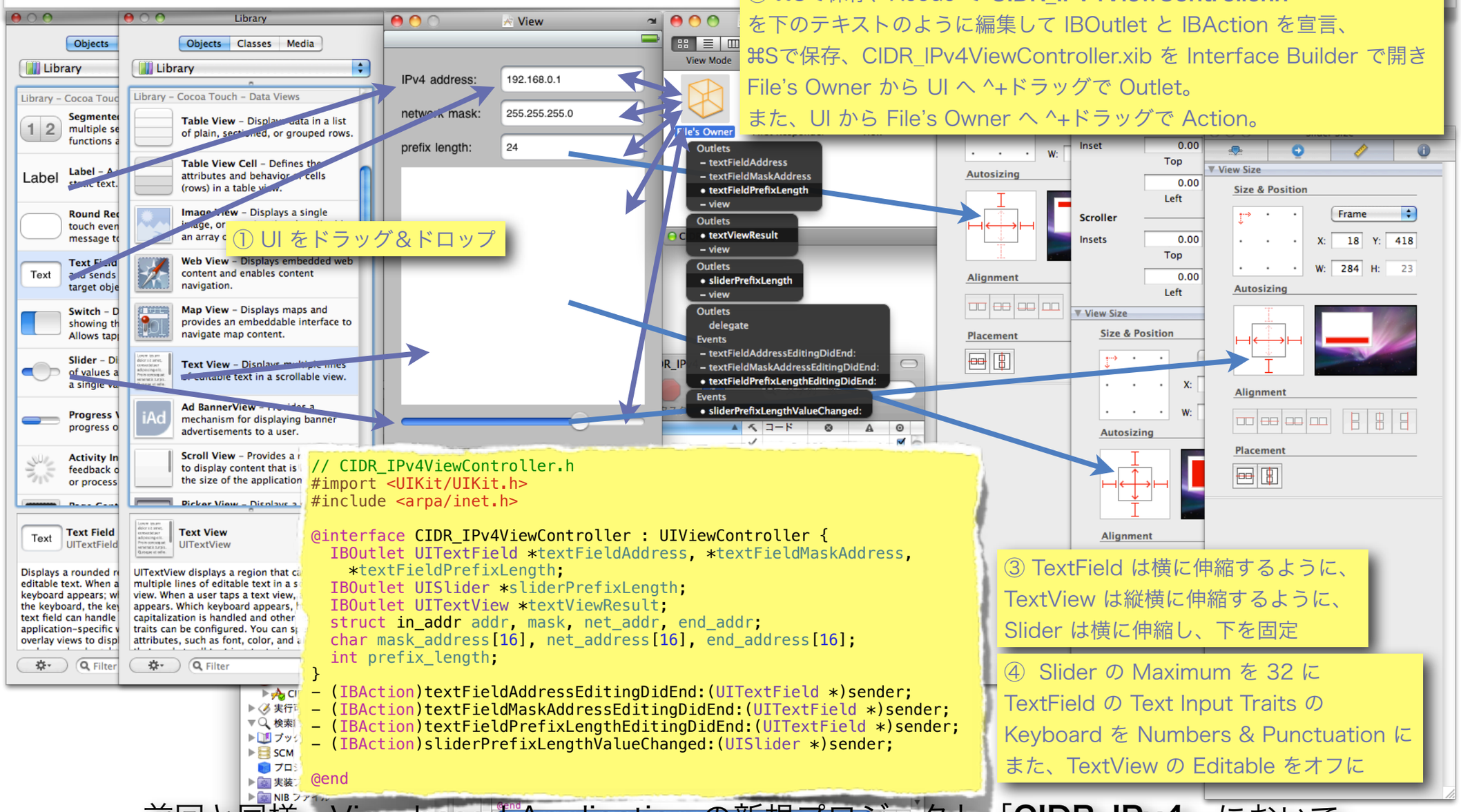
IPv4/v6アドレス範囲の算出ツールの設計

- ネットワーク管理者向けツール
- IPアドレスとネットワークマスクまたはマスク長(prefix length)からアドレスレンジを算出する
- IPv4、IPv6 対応
  IPv4 は32ビットでエンディアンに気を付ければ簡単
  IPv6 は128ビットのレンジ計算を要するが、どうすれば簡単か

とにかくこんなのを作りたい ☞
簡単なところから手を着けよう

1. IPv4 のアドレスレンジの算出
2. IPv6 のアドレスレンジの算出
   - 128ビット演算に C++ を使う
3. IPv4 と IPv6 のアドレスレンジ算出ツールの統合
   - タブバーアプリケーションの制作

IPv4アドレス範囲の算出ツールの作成(1)
create View-based Application




```
// CIDR_IPv4ViewController.h
#import <UIKit/UIKit.h>
#include <arpa/inet.h>

@interface CIDR_IPv4ViewController : UIViewController {
  IBOutlet UITextField *textFieldAddress, *textFieldMaskAddress,
    *textFieldPrefixLength;
  IBOutlet UISlider *sliderPrefixLength;
  IBOutlet UITextView *textViewResult;
  struct in_addr addr, mask, net_addr, end_addr;
  char mask_address[16], net_address[16], end_address[16];
  int prefix_length;
}
- (IBAction)textFieldAddressEditingDidEnd:(UITextField *)sender;
- (IBAction)textFieldMaskAddressEditingDidEnd:(UITextField *)sender;
- (IBAction)textFieldPrefixLengthEditingDidEnd:(UITextField *)sender;
- (IBAction)sliderPrefixLengthValueChanged:(UISlider *)sender;

@end
```

- 前回と同様、View-based Application の新規プロジェクト「**CIDR_IPv4**」において、UI の配置とOutlet、Action の定義を行う。細かな設定は上記コメント参照。

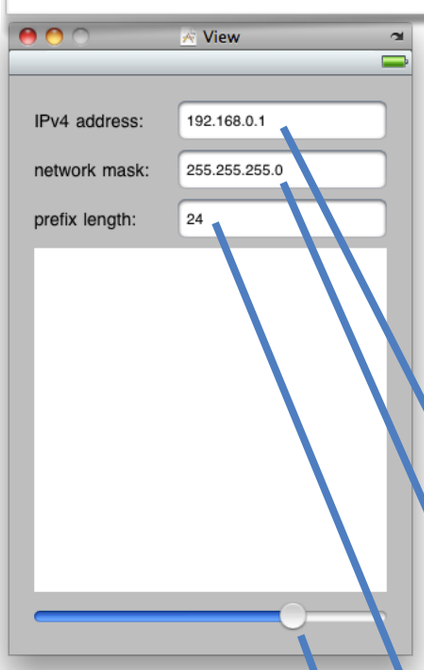
IPv4アドレス範囲の算出ツール(2)

CIDR_v4ViewController.h の編集

```
// CIDR_IPv4ViewController.h
#import <UIKit/UIKit.h>
#include <arpa/inet.h>

@interface CIDR_IPv4ViewController : UIViewController {
  IBOutlet UITextField *textFieldAddress, *textFieldMaskAddress,
    *textFieldPrefixLength;
  IBOutlet UISlider *sliderPrefixLength;
  IBOutlet UITextView *textViewResult;
  struct in_addr addr, mask, net_addr, end_addr;
  char mask_address[16], net_address[16], end_address[16];
  int prefix_length;
}
- (IBAction)textFieldAddressEditingDidEnd:(UITextField *)sender;
- (IBAction)textFieldMaskAddressEditingDidEnd:(UITextField *)sender;
- (IBAction)textFieldPrefixLengthEditingDidEnd:(UITextField *)sender;
- (IBAction)sliderPrefixLengthValueChanged:(UISlider *)sender;

@end
```



Action 先

```
#import "CIDR_IPv4ViewController.h"
long uint32ntz(uint32_t x)
{ /* number of trailing zeros */
  long n = 0;
  x = ~x & (x - 1);
  while (x != 0) { n++; x >>= 1; }
  return n;
}

@implementation CIDR_IPv4ViewController

- (void)showResult
{
  net_addr.s_addr = addr.s_addr & mask.s_addr;
  end_addr.s_addr = addr.s_addr | ~mask.s_addr;
  inet_ntop(AF_INET, &net_addr, net_address, sizeof(net_address));
  inet_ntop(AF_INET, &end_addr, end_address, sizeof(end_address));
  textViewResult.text = [NSString stringWithFormat:@"net address:\t%s\nend address:\t%s\n", net_address, end_address];
}
- (IBAction)textFieldAddressEditingDidEnd:(UITextField *)sender
{
  if (inet_pton(AF_INET, [textFieldAddress.text cStringUsingEncoding:NSUTF8StringEncoding], &addr) != 1) return;
  if (sender) [self showResult];
}
- (IBAction)textFieldMaskAddressEditingDidEnd:(UITextField *)sender
{
  if (inet_pton(AF_INET, [textFieldMaskAddress.text cStringUsingEncoding:NSUTF8StringEncoding], &mask) != 1) return;
  if (~ntohl(mask.s_addr) & (~ntohl(mask.s_addr) + 1)) return;
  prefix_length = 32 - uint32ntz(ntohl(mask.s_addr));
  sliderPrefixLength.value = prefix_length;
  textFieldPrefixLength.text = [NSString stringWithFormat:@"%d", prefix_length];
  if (sender) [self showResult];
}
- (IBAction)textFieldPrefixLengthEditingDidEnd:(UITextField *)sender
{
  if (!([[NSScanner scannerWithString:textFieldPrefixLength.text] scanInt:&prefix_length] &&
        0 <= prefix_length && prefix_length <= 32)) return;
  sliderPrefixLength.value = prefix_length;
  mask.s_addr = htonl(~0ULL << (32 - prefix_length));
  inet_ntop(AF_INET, &mask, mask_address, sizeof(mask_address));
  textFieldMaskAddress.text = [NSString stringWithFormat:@"%s", mask_address];
  [self textFieldMaskAddressEditingDidEnd:sender];
}
- (IBAction)sliderPrefixLengthValueChanged:(UISlider *)sender
{
  textFieldPrefixLength.text = [NSString stringWithFormat:@"%d", (int)sliderPrefixLength.value];
  [self textFieldPrefixLengthEditingDidEnd:(id)sender];
}
- (void)viewDidLoad {
  [super viewDidLoad];
  textViewResult.font = [UIFont fontWithName:@"Helvetica" size:14];
  [self textFieldAddressEditingDidEnd:nil];
  [self sliderPrefixLengthValueChanged:nil];
  [self showResult];
}
```

- ⌘Rで実行！

IPv4アドレス範囲の算出ツールの作成(3)

CIDR_v4ViewController.m でのビット演算

```
#import "CIDR_IPv4ViewController.h"
long uint32ntz(uint32_t x)
{ /* number of trailing zeros */
  long n = 0;
  x = ~x & (x - 1);
  while (x != 0) { n++; x >>= 1; }
  return n;
}

@implementation CIDR_IPv4ViewController

- (void)showResult
{
  net_addr.s_addr = addr.s_addr & mask.s_addr;
  end_addr.s_addr = addr.s_addr | ~mask.s_addr;
  inet_ntop(AF_INET, &net_addr, net_address, sizeof(net_address));
  inet_ntop(AF_INET, &end_addr, end_address, sizeof(end_address));
  textViewResult.text = [NSString stringWithFormat:@"net address:\t%s\nend address:\t%s
}
- (IBAction)textFieldAddressEditingDidEnd:(UITextField *)sender
{
  if (inet_pton(AF_INET, [textFieldAddress.text cStringUsingEncoding:NSUTF8StringEncodi
  if (sender) [self showResult];
}
- (IBAction)textFieldMaskAddressEditingDidEnd:(UITextField *)sender
{
  if (inet_pton(AF_INET, [textFieldMaskAddress.text cStringUsingEncoding:NSUTF8StringEr
  if (~ntohl(mask.s_addr) & (~ntohl(mask.s_addr) + 1)) return;
  prefix_length = 32 - uint32ntz(ntohl(mask.s_addr));
  sliderPrefixLength.value = prefix_length;
  textFieldPrefixLength.text = [NSString stringWithFormat:@"%d", prefix_length];
  if (sender) [self showResult];
}
- (IBAction)textFieldPrefixLengthEditingDidEnd:(UITextField *)sender
{
  if (!([[NSScanner scannerWithString:textFieldPrefixLength.text] scanInt:&prefix_length] &&
        0 <= prefix_length && prefix_length <= 32)) return;
  sliderPrefixLength.value = prefix_length;
  mask.s_addr = htonl(~0ULL << (32 - prefix_length));
  inet_ntop(AF_INET, &mask, mask_address, sizeof(mask_address));
  textFieldMaskAddress.text = [NSString stringWithFormat:@"%s", mask_address];
  [self textFieldMaskAddressEditingDidEnd:sender];
}
- (IBAction)sliderPrefixLengthValueChanged:(UISlider *)sender
{
  textFieldPrefixLength.text = [NSString stringWithFormat:@"%d", (int)sliderPrefixLength.value];
  [self textFieldPrefixLengthEditingDidEnd:(id)sender];
}
- (void)viewDidLoad {
  [super viewDidLoad];
  textViewResult.font = [UIFont fontWithName:@"Helvetica" size:14];
  [self textFieldAddressEditingDidEnd:nil];
  [self sliderPrefixLengthValueChanged:nil];
  [self showResult];
}
```

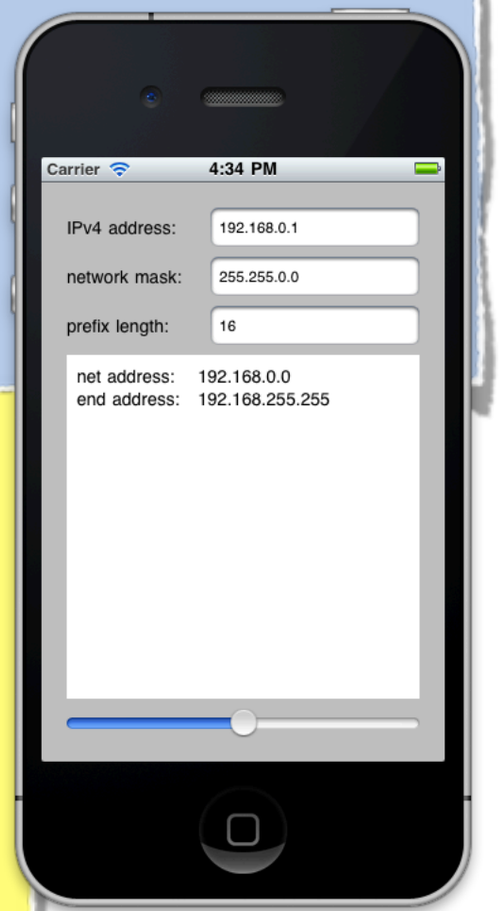
A_a: アドレス
M: サブネットマスク
N_a: 始点アドレス (ネットワークアドレス)
- $N_a = A_a \,\&\, M$

E_a: 終点アドレス (IPv4 ブロードキャストアドレス)
- $E_a = A_a \mid \tilde{}M$

最右 $01\ldots$ を 0 にする。$2^n - 1$ なら 0 になる
$m \,\&\, (m+1)$
サブネットマスク M がビット列 $1\ldots0\ldots$ か判定
- $\tilde{}M \,\&\, (\tilde{}M + 1) \equiv 0$

末尾まで続く 0 を 1 ビット列として取り出す
$\tilde{}m \,\&\, (m-1)$
サブネットマスク M のビット列 $1\ldots0\ldots$ の 0 の個数を数える ntz
- ntz(M) {
  $N = 0$;
  $M = \tilde{}M \,\&\, (M-1)$;
  while ($M \neq 0$) {
    $N++$;
    $M \gg= 1$;
  }
  return N;

L_M: M のプレフィックス長
- $L_M = 32 - \mathrm{ntz}(M)$
- $M = \tilde{}0 \ll (32 - L_M)$



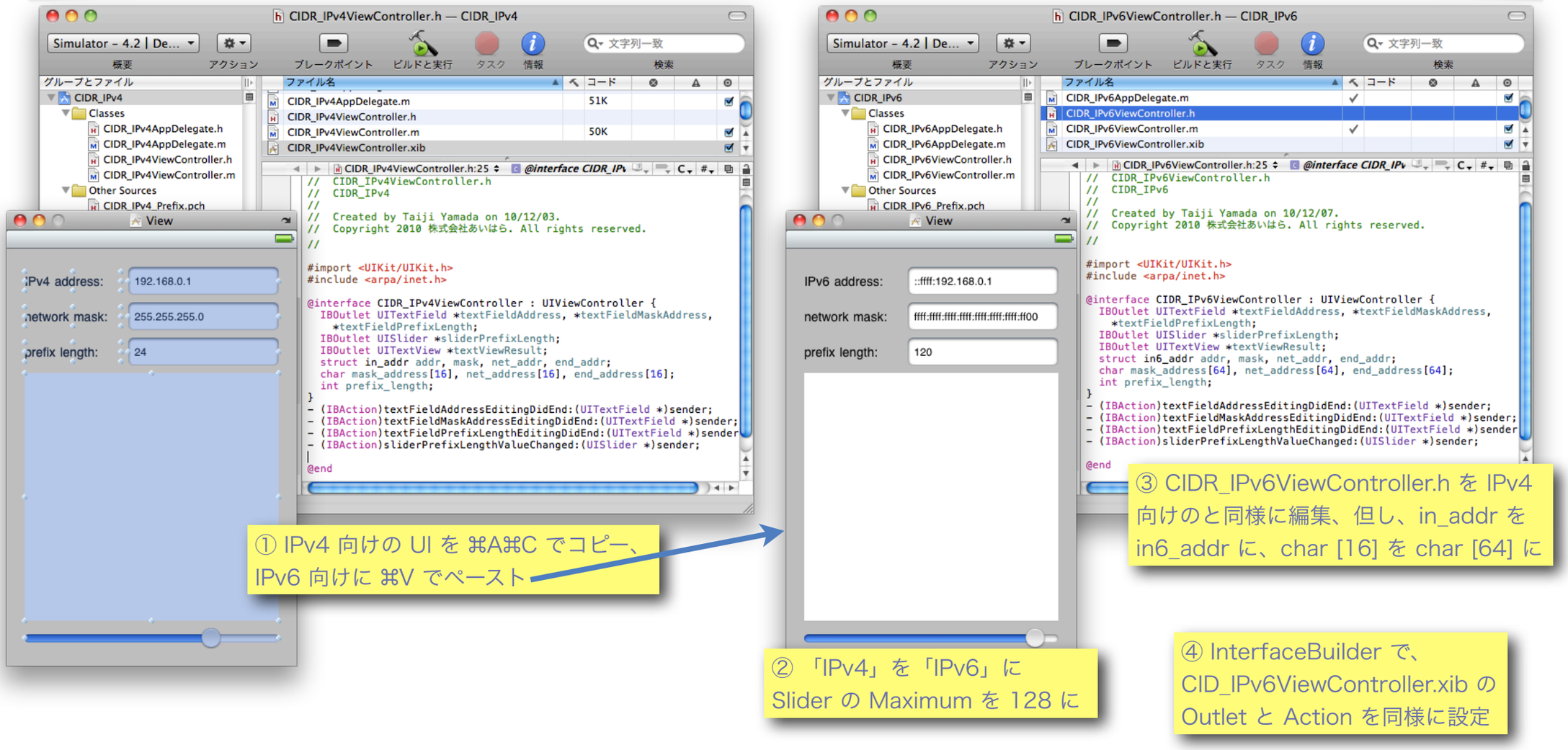
IPv4アドレス範囲の算出ツールの作成(4)

CIDR_v4ViewController.m の編集


```
$ diff -u CIDR_IPv4ViewController.m~ CIDR_IPv4ViewController.m
@@ -31,13 +31,17 @@
 }
 - (IBAction)textFieldAddressEditingDidEnd:(UITextField *)sender
 {
+  textFieldAddress.clearButtonMode = UITextFieldViewModeAlways;
   if (inet_pton(AF_INET, [textFieldAddress.text cStringUsingEncoding:NSUTF8StringEncoding], &addr) != 1) return;
+  textFieldAddress.clearButtonMode = UITextFieldViewModeNever;
   if (sender) [self showResult];
 }
 - (IBAction)textFieldMaskAddressEditingDidEnd:(UITextField *)sender
 {
+  textFieldMaskAddress.clearButtonMode = UITextFieldViewModeAlways;
   if (inet_pton(AF_INET, [textFieldMaskAddress.text cStringUsingEncoding:NSUTF8StringEncoding], &mask) != 1) return;
   if (~ntohl(mask.s_addr) & (~ntohl(mask.s_addr) + 1)) return;
+  textFieldMaskAddress.clearButtonMode = UITextFieldViewModeNever;
   prefix_length = 32 - uint32ntz(ntohl(mask.s_addr));
   sliderPrefixLength.value = prefix_length;
   textFieldPrefixLength.text = [NSString stringWithFormat:@"%d", prefix_length];
@@ -45,8 +49,10 @@
 }
 - (IBAction)textFieldPrefixLengthEditingDidEnd:(UITextField *)sender
 {
+  textFieldPrefixLength.clearButtonMode = UITextFieldViewModeAlways;
   if (!([[NSScanner scannerWithString:textFieldPrefixLength.text] scanInt:&prefix_length] &&
         0 <= prefix_length && prefix_length <= 32)) return;
+  textFieldPrefixLength.clearButtonMode = UITextFieldViewModeNever;
   sliderPrefixLength.value = prefix_length;
   mask.s_addr = htonl(~0ULL << (32 - prefix_length));
   inet_ntop(AF_INET, &mask, mask_address, sizeof(mask_address));
```

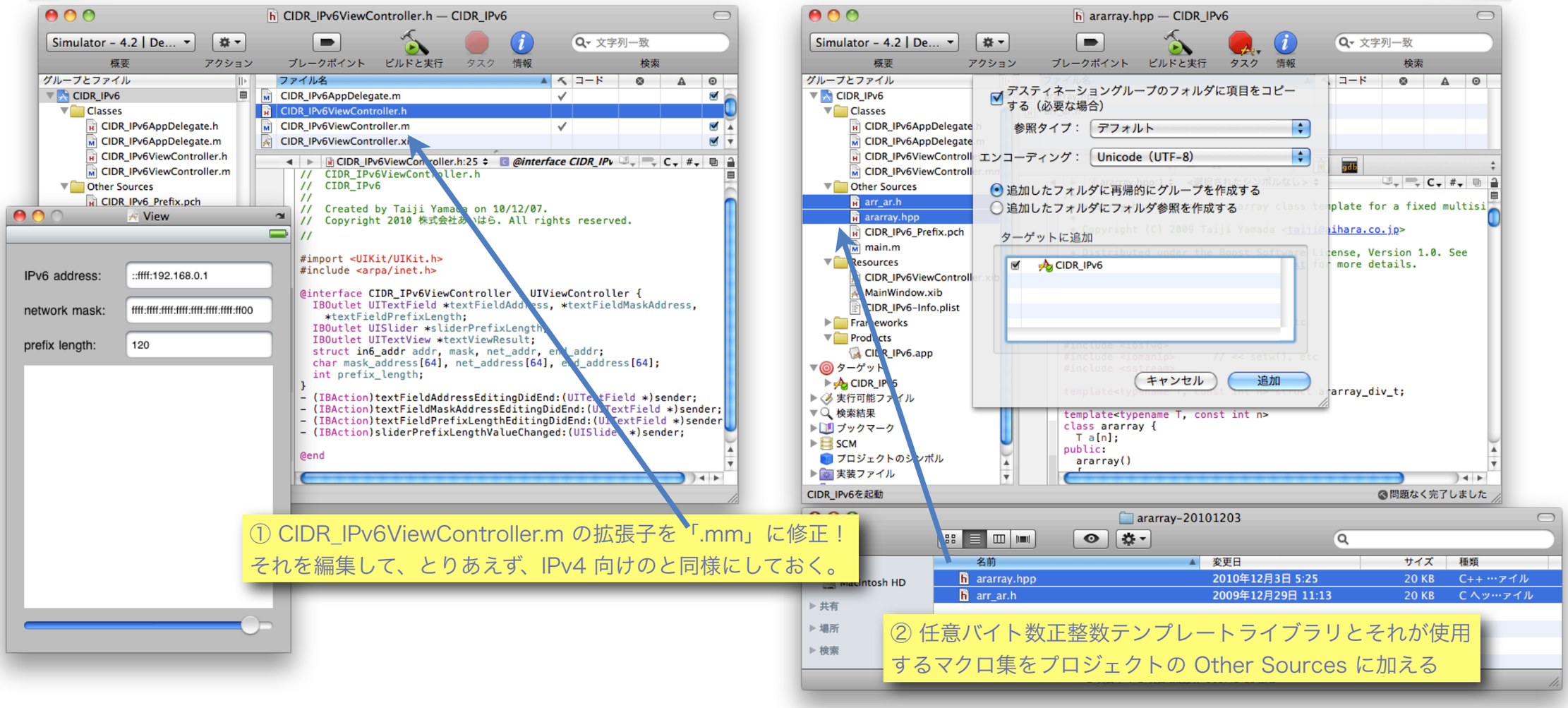
- 不正な入力をユーザに知らしめるよう、上記のように修正
- 左図のように入力が不正な場合のみ UITextField にクリアボタンを表示

IPv6アドレス範囲の算出ツールの作成(1)
create View-based Application



- 先程と同様、 View-based Application の新規プロジェクト「**CIDR_IPv6**」において、UI の配置と Outlet、Action の定義を行う。細かな設定は上記コメント参照。

IPv6アドレス範囲の算出ツールの作成(2)
C++ programming in Objective-C



- CIDR_IPv6ViewController.m の拡張子を「.mm」にして、C++ としてコーディング可能に
- C++ で書かれた、任意バイト数正整数テンプレートライブラリ「**ararray.hpp**」とそれが使用するマクロ集「**arr_ar.h**」をこのプロジェクトの Other Sources に加える

IPv6アドレス範囲の算出ツールの作成(3)
「CIDR_v4ViewController.mm」の編集

- ⌘Rで実行！

```
$ diff -u CIDR_IPv6ViewController.mm~ CIDR_IPv6ViewController.mm
 #import "CIDR_IPv6ViewController.h"
+#include "ararray.hpp"
+typedef ararray<uint8_t, 16> uint128_t;

 @implementation CIDR_IPv6ViewController

 - (void)showResult
 {
-  net_addr.s_addr = addr.s_addr & mask.s_addr;
-  end_addr.s_addr = addr.s_addr | ~mask.s_addr;
-  inet_ntop(AF_INET, &net_addr, net_address, sizeof(net_address));
-  inet_ntop(AF_INET, &end_addr, end_address, sizeof(end_address));
+  (uint128_t(addr.s6_addr) & uint128_t(mask.s6_addr)).get_a(net_addr.s6_addr);
+  (uint128_t(addr.s6_addr) | ~uint128_t(mask.s6_addr)).get_a(end_addr.s6_addr);
+  inet_ntop(AF_INET6, &net_addr, net_address, sizeof(net_address));
+  inet_ntop(AF_INET6, &end_addr, end_address, sizeof(end_address));
   textViewResult.text = [NSString stringWithFormat:@"net address:\t%s\nend address:\t%s\n", net_address, end_address];
 }
 - (IBAction)textFieldAddressEditingDidEnd:(UITextField *)sender
 {
   textFieldAddress.clearButtonMode = UITextFieldViewModeAlways;
-  if (inet_pton(AF_INET, [textFieldAddress.text cStringUsingEncoding:NSUTF8StringEncoding], &addr) != 1) return;
+  if (inet_pton(AF_INET6, [textFieldAddress.text cStringUsingEncoding:NSUTF8StringEncoding], &addr) != 1) return;
   textFieldAddress.clearButtonMode = UITextFieldViewModeNever;
   if (sender) [self showResult];
 }
 - (IBAction)textFieldMaskAddressEditingDidEnd:(UITextField *)sender
 {
   textFieldMaskAddress.clearButtonMode = UITextFieldViewModeAlways;
-  if (inet_pton(AF_INET, [textFieldMaskAddress.text cStringUsingEncoding:NSUTF8StringEncoding], &mask) != 1) return;
-  if (~ntohl(mask.s_addr) & (~ntohl(mask.s_addr) + 1)) return;
+  if (inet_pton(AF_INET6, [textFieldMaskAddress.text cStringUsingEncoding:NSUTF8StringEncoding], &mask) != 1) return;
+  if ((~uint128_t(mask.s6_addr) & (~uint128_t(mask.s6_addr) + uint128_t(1ULL))) != 0) return;
   textFieldMaskAddress.clearButtonMode = UITextFieldViewModeNever;
-  prefix_length = 32 - uint32ntz(ntohl(mask.s_addr));
+  prefix_length = 128 - ntz(uint128_t(mask.s6_addr));
   sliderPrefixLength.value = prefix_length;
   textFieldPrefixLength.text = [NSString stringWithFormat:@"%d", prefix_length];
   if (sender) [self showResult];
@@ -40,11 +42,11 @@
 {
   textFieldPrefixLength.clearButtonMode = UITextFieldViewModeAlways;
   if (!([[NSScanner scannerWithString:textFieldPrefixLength.text] scanInt:&prefix_length] &&
-        0 <= prefix_length && prefix_length <= 32)) return;
+        0 <= prefix_length && prefix_length <= 128)) return;
   textFieldPrefixLength.clearButtonMode = UITextFieldViewModeNever;
   sliderPrefixLength.value = prefix_length;
-  mask.s_addr = htonl(~0ULL << (32 - prefix_length));
-  inet_ntop(AF_INET, &mask, mask_address, sizeof(mask_address));
+  (~uint128_t(0ULL) << (128 - prefix_length)).get_a(mask.s6_addr);
+  inet_ntop(AF_INET6, &mask, mask_address, sizeof(mask_address));
   textFieldMaskAddress.text = [NSString stringWithFormat:@"%s", mask_address];
   [self textFieldMaskAddressEditingDidEnd:sender];
 }
```

(補足) ararray.hpp の使い方

```
#include ”ararray.hpp”
uint8_t s6_addr[16] = { 0, };
typedef ararray<uint8_t, 16> uint128_t;
uint128_t A(1ULL);     // 符号無し整数の値からインスタンスを生成
uint128_t B(s6_addr); // 配列の値からインスタンスを生成
A.get_a(s6_addr);      // 配列へインスタンスの値をコピー
```

その他の演算などは**組み込みの符号無し整数と同じ**

IPv6アドレス範囲の算出ツールの作成(4)

CIDR_v6ViewController.mm でのビット演算

```
$ diff -u CIDR_IPv6ViewController.mm~ CIDR_IPv6ViewController.mm
 #import "CIDR_IPv6ViewController.h"
+#include "ararray.hpp"
+typedef ararray<uint8_t, 16> uint128_t;

 @implementation CIDR_IPv6ViewController

 - (void)showResult
 {
-  net_addr.s_addr = addr.s_addr & mask.s_addr;
-  end_addr.s_addr = addr.s_addr | ~mask.s_addr;
-  inet_ntop(AF_INET, &net_addr, net_address, sizeof(net_address));
-  inet_ntop(AF_INET, &end_addr, end_address, sizeof(end_address));
+  (uint128_t(addr.s6_addr) & uint128_t(mask.s6_addr)).get_a(net_addr.s6_addr);
+  (uint128_t(addr.s6_addr) | ~uint128_t(mask.s6_addr)).get_a(end_addr.s6_addr);
+  inet_ntop(AF_INET6, &net_addr, net_address, sizeof(net_address));
+  inet_ntop(AF_INET6, &end_addr, end_address, sizeof(end_address));
   textViewResult.text = [NSString stringWithFormat:@"net address:\t%s\nend address:\t%
 }
 - (IBAction)textFieldAddressEditingDidEnd:(UITextField *)sender
 {
   textFieldAddress.clearButtonMode = UITextFieldViewModeAlways;
-  if (inet_pton(AF_INET, [textFieldAddress.text cStringUsingEncoding:NSUTF8StringEncod
+  if (inet_pton(AF_INET6, [textFieldAddress.text cStringUsingEncoding:NSUTF8StringEnco
   textFieldAddress.clearButtonMode = UITextFieldViewModeNever;
   if (sender) [self showResult];
 }
 - (IBAction)textFieldMaskAddressEditingDidEnd:(UITextField *)sender
 {
   textFieldMaskAddress.clearButtonMode = UITextFieldViewModeAlways;
-  if (inet_pton(AF_INET, [textFieldMaskAddress.text cStringUsingEncoding:NSUTF8String
-  if (~ntohl(mask.s_addr) & (~ntohl(mask.s_addr) + 1)) return;
+  if (inet_pton(AF_INET6, [textFieldMaskAddress.text cStringUsingEncoding:NSUTF8StringEncoding], &mask) != 1) return;
+  if ((~uint128_t(mask.s6_addr) & (~uint128_t(mask.s6_addr) + uint128_t(1ULL))) != 0) return;
   textFieldMaskAddress.clearButtonMode = UITextFieldViewModeNever;
-  prefix_length = 32 - uint32ntz(ntohl(mask.s_addr));
+  prefix_length = 128 - ntz(uint128_t(mask.s6_addr));
   sliderPrefixLength.value = prefix_length;
   textFieldPrefixLength.text = [NSString stringWithFormat:@"%d", prefix_length];
   if (sender) [self showResult];
@@ -40,11 +42,11 @@
 {
   textFieldPrefixLength.clearButtonMode = UITextFieldViewModeAlways;
   if (!([[NSScanner scannerWithString:textFieldPrefixLength.text] scanInt:&prefix_length] &&
-        0 <= prefix_length && prefix_length <= 32)) return;
+        0 <= prefix_length && prefix_length <= 128)) return;
   textFieldPrefixLength.clearButtonMode = UITextFieldViewModeNever;
   sliderPrefixLength.value = prefix_length;
-  mask.s_addr = htonl(~0ULL << (32 - prefix_length));
-  inet_ntop(AF_INET, &mask, mask_address, sizeof(mask_address));
+  (~uint128_t(0ULL) << (128 - prefix_length)).get_a(mask.s6_addr);
+  inet_ntop(AF_INET6, &mask, mask_address, sizeof(mask_address));
   textFieldMaskAddress.text = [NSString stringWithFormat:@"%s", mask_address];
   [self textFieldMaskAddressEditingDidEnd:sender];
 }
```

A_a: アドレス
M: サブネットマスク
N_a: 始点アドレス (ネットワークアドレス)
$N_a = A_a \ \& \ M$
E_a: 終点アドレス
$E_a = A_a \mid \tilde{}M$
最右 $01\ldots$ を 0 にする。$2^n - 1$ なら 0 になる
$m \ \& \ (m+1)$
サブネットマスク M がビット列 $1\ldots0\ldots$ か判定
$\tilde{}M \ \& \ (\tilde{}M + 1) \equiv 0$
末尾まで続く 0 を 1 ビット列として取り出す
$\tilde{}m \ \& \ (m-1)$
サブネットマスク M のビット列 $1\ldots0\ldots$ の 0 の個数を数える ntz

```
ntz(M) {
  N = 0;
  M = ~M & (M - 1);
  while (M != 0) {
    N++;
    M >>= 1;
  }
  return N;
```

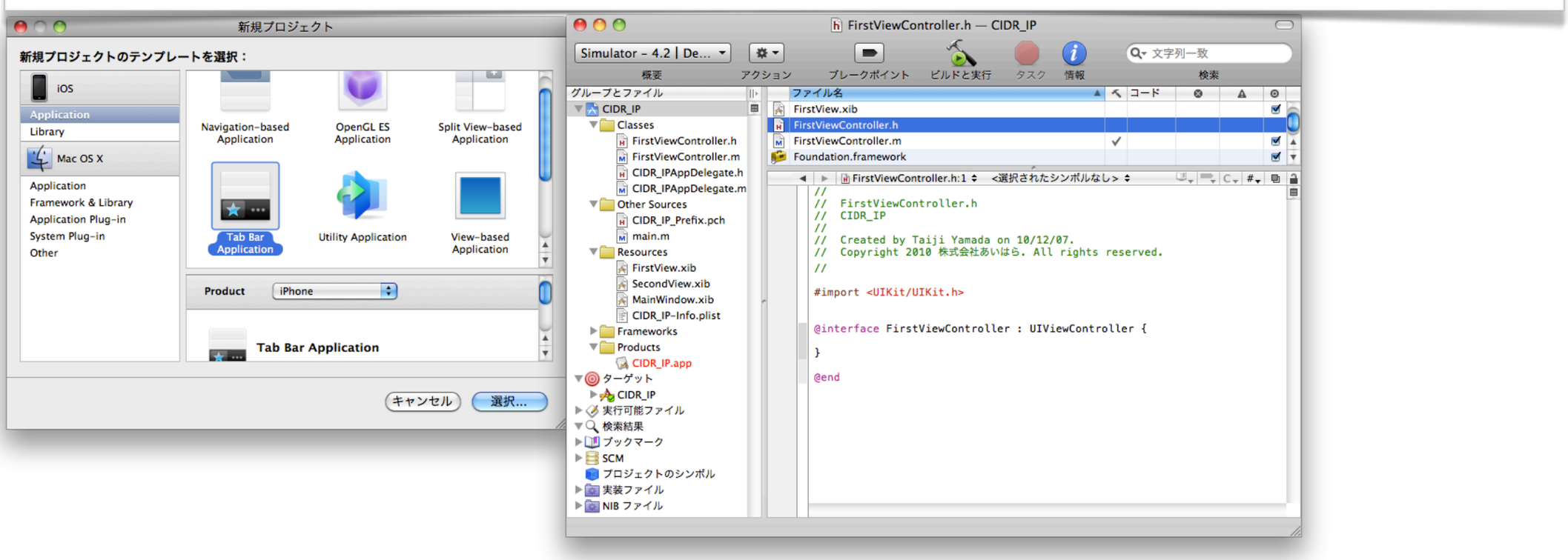
L_M: M のプレフィックス長
$L_M = 128 - \mathrm{ntz}(M)$
$M = \tilde{}0 \ll (128 - L_M)$

(補足) ararray.hpp の使い方

```
#include ”ararray.hpp”
uint8_t s6_addr[16] = { 0, };
typedef ararray<uint8_t, 16> uint128_t;
uint128_t A(1ULL);     // 符号無し整数の値からインスタンスを生成
uint128_t B(s6_addr); // 配列の値からインスタンスを生成
A.get_a(s6_addr);      // 配列へインスタンスの値をコピー
```

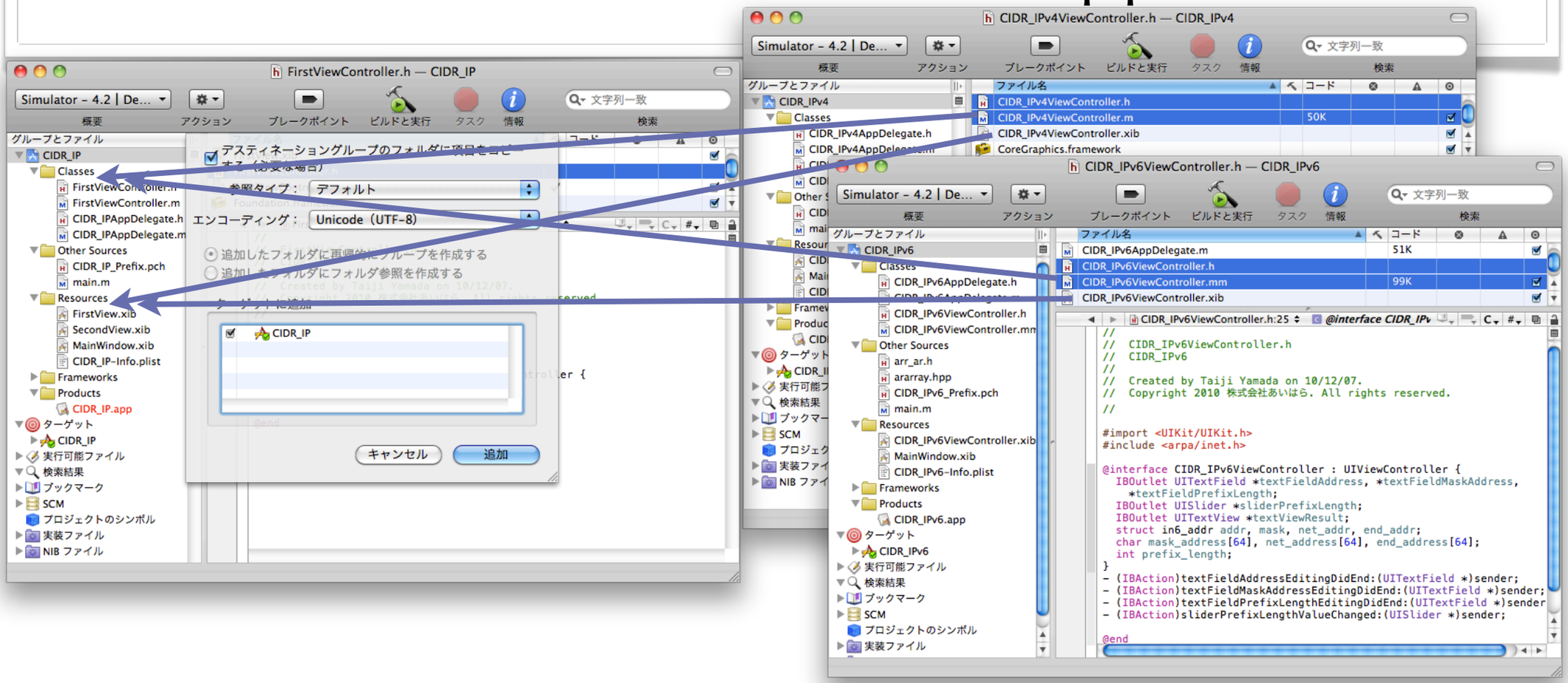
その他の演算などは**組み込みの符号無し整数と同じ**

IPv4/v6アドレス範囲の算出ツールの作成(1)
create Tab Bar Application



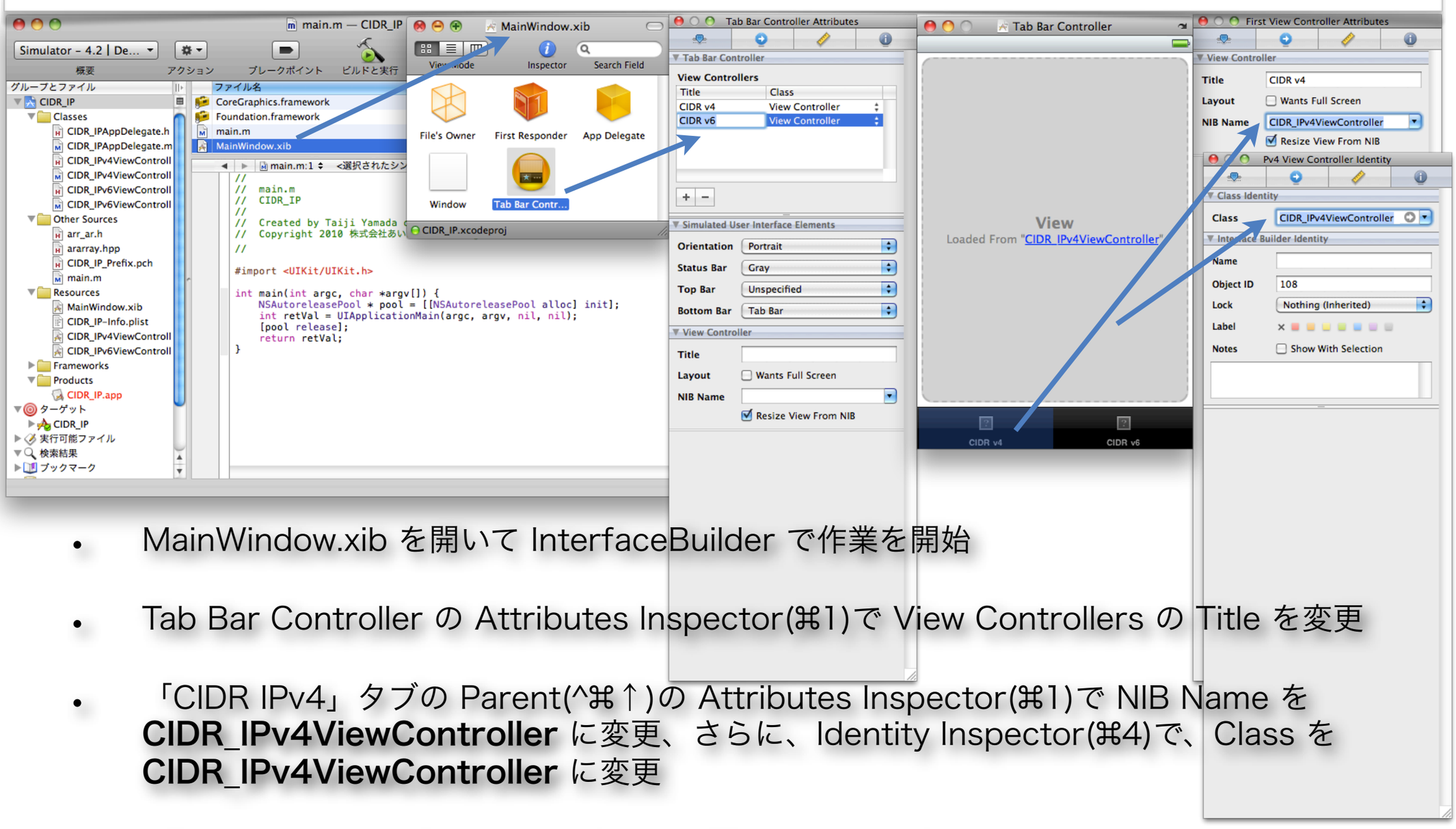
- 新規プロジェクトで Tab Bar Application のテンプレートから「**CIDR_IP**」を作成する
- FirstView.xib, FirstViewController.{h,m}, SecondView.xib は不要なので、削除する
- Xcode の「ファイル」「最近使ったプロジェクトを開く」で **CIDR_IPv4** と **CIDR_IPv6** を開いておく

IPv4/v6アドレス範囲の算出ツールの作成(2)
ViewController in Tab Bar Application



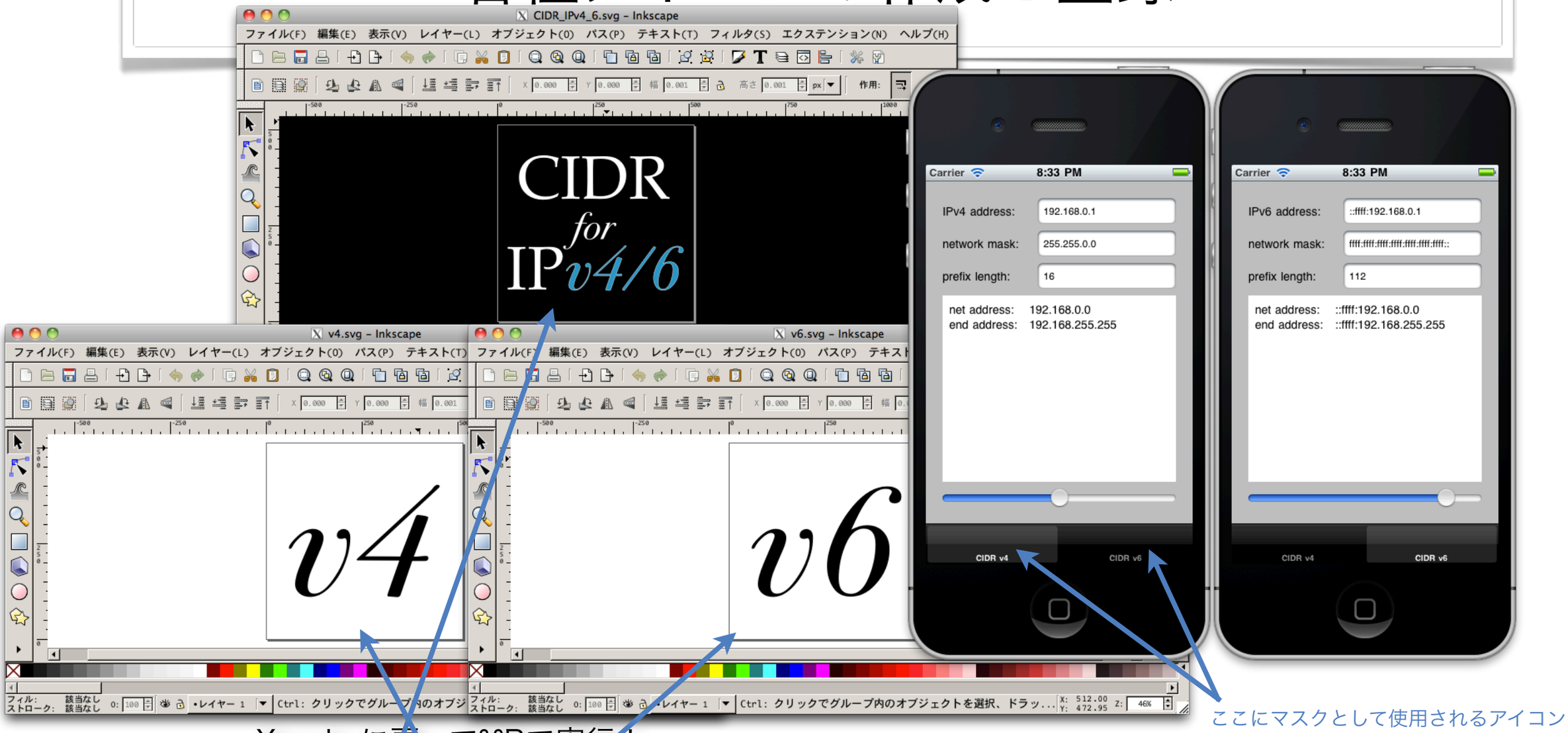
- **CIDR_IPv4ViewController.{h,m}** を CID_IP プロジェクトの Classes へコピー
- **CIDR_IPv4ViewController.xib** を CID_IP プロジェクトの Resources へコピー
- **CIDR_IPv6ViewController.{h,mm}** を CID_IP プロジェクトの Classes へコピー
- **CIDR_IPv6ViewController.xib** を CID_IP プロジェクトの Resources へコピー
- **arr_ar.h, ararray.hpp** を CID_IP プロジェクトの Other Sources へコピー

IPv4/v6アドレス範囲の算出ツールの作成(3)
create Tab Bar Application



- MainWindow.xib を開いて InterfaceBuilder で作業を開始
- Tab Bar Controller の Attributes Inspector(⌘1)で View Controllers の Title を変更
- 「CIDR IPv4」タブの Parent(^⌘↑)の Attributes Inspector(⌘1)で NIB Name を **CIDR_IPv4ViewController** に変更、さらに、Identity Inspector(⌘4)で、Class を **CIDR_IPv4ViewController** に変更
- 「CIDR_IPv6」タブについても同様に変更して、⌘S で保存。

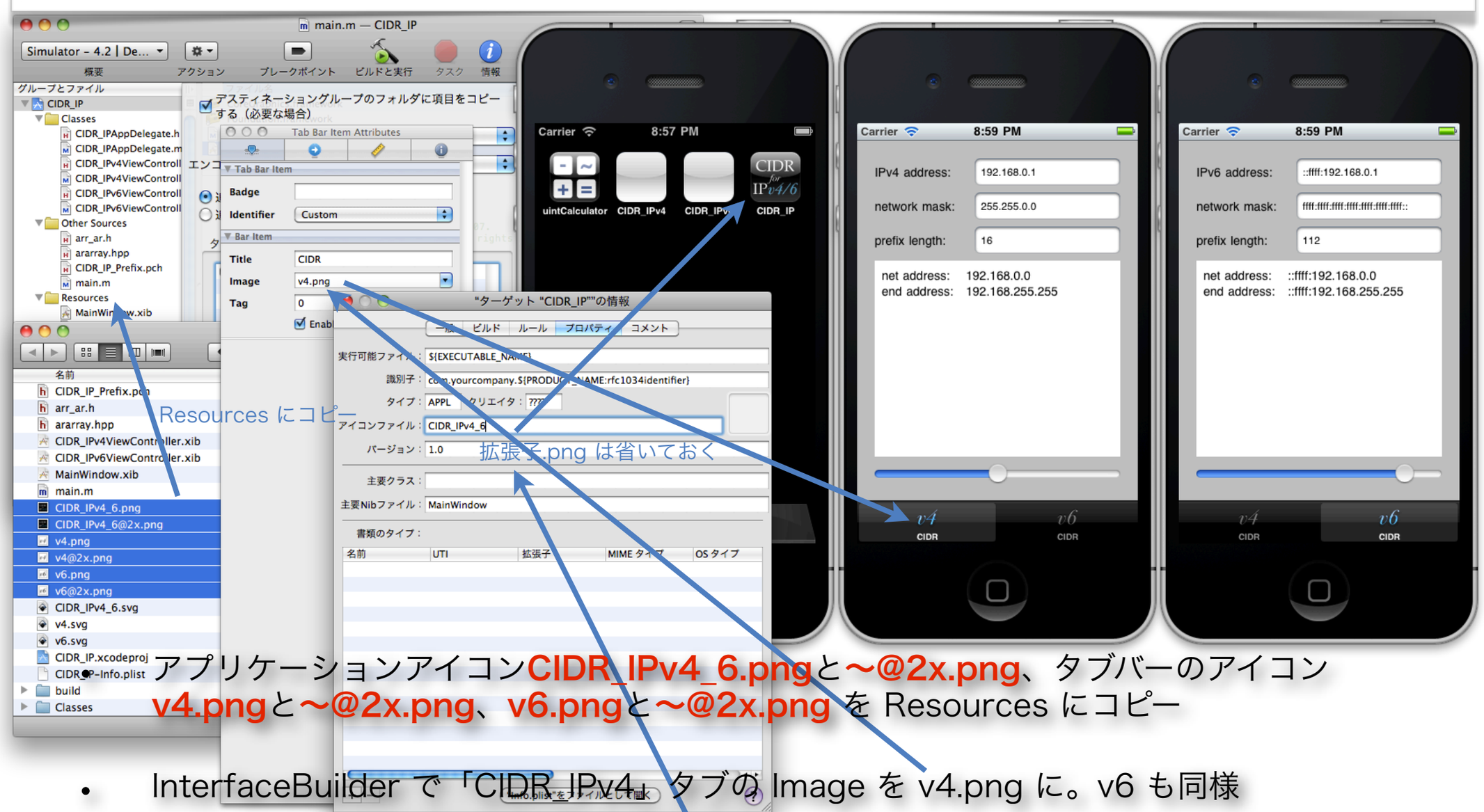
IPv4/v6アドレス範囲の算出ツールの作成(4)
各種アイコンの作成と登録



- Xcode に戻って⌘Rで実行！
  ひとまず完成だが、各種アイコンの作成と登録が完了していないので、inkscape で作成

- アプリケーションアイコンは、以前と同様、**57x57**と**114x114**のPNGファイル
  タブバーのアイコンは、約**30x30**と**60x60**のPNGファイルで書き出しておく

IPv4/v6アドレス範囲の算出ツールの作成(5)
各種アイコンの作成と登録

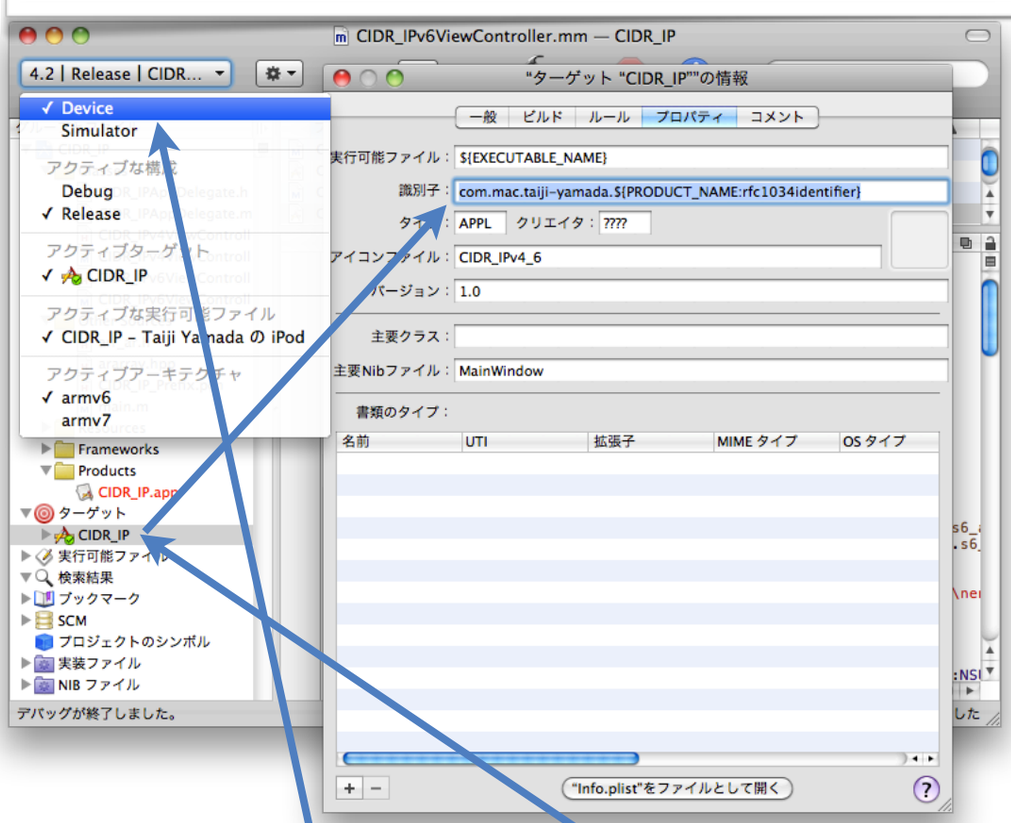


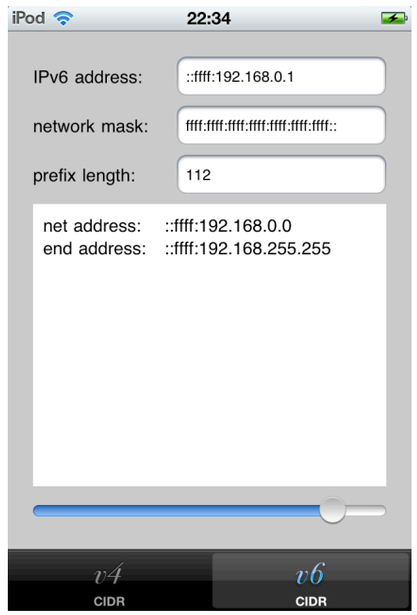
アプリケーションアイコン**CIDR_IPv4_6.png**と**～@2x.png**、タブバーのアイコン**v4.png**と**～@2x.png**、**v6.png**と**～@2x.png** を Resources にコピー

- InterfaceBuilder で「CIDR_IPv4」タブの Image を v4.png に。v6 も同様
- Xcode のターゲット**CIDR_IP**の情報(⌘I)のプロパティでアイコンファイルを登録

IPv4/v6アドレス範囲の算出ツールを実機で

App ID の作成とオーガナイザへの登録

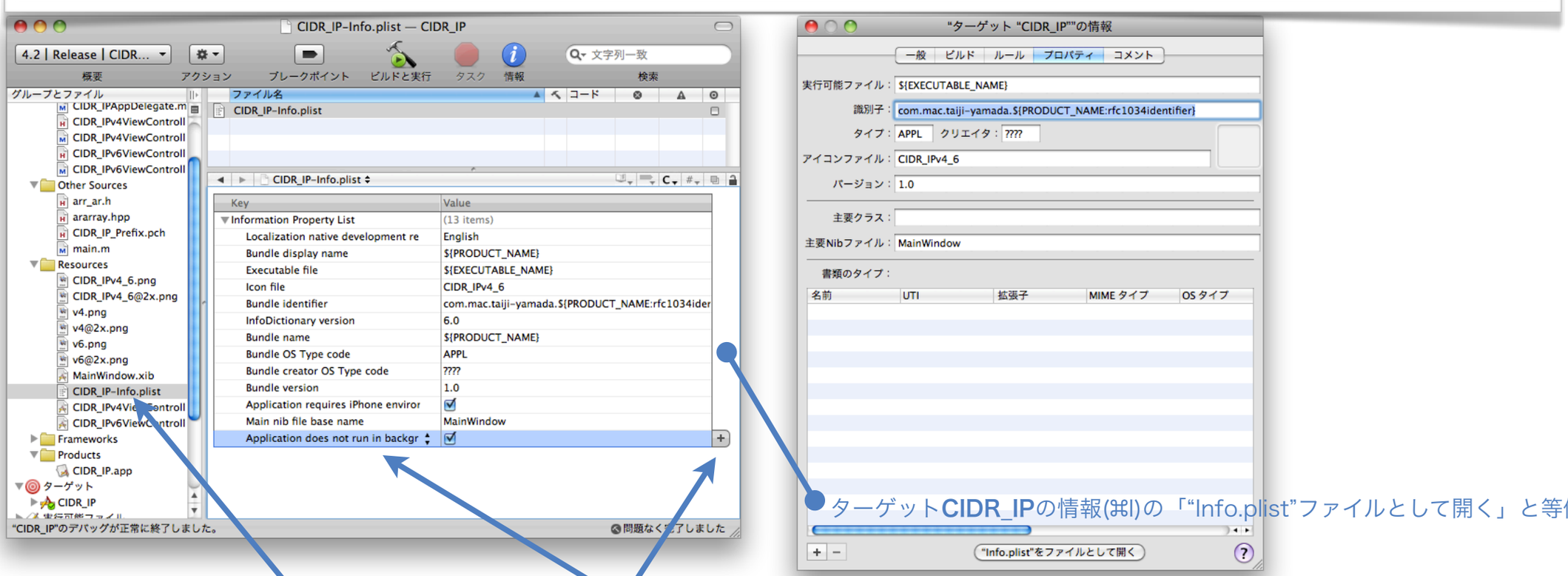




- Apple の Provisioning Portal で CIDR_IP のための App ID を取得
- Xcode のオーガナイザに App ID を登録
- Xcode のターゲット**CIDR_IP**の情報(⌘I)のプロパティで識別子を App ID に対応させる
- ビルドターゲットを Device にして、⌘R で実行！

アプリケーションの常駐を抑止

Property List Editor でアプリの Info.plist を編集



- iOS 4.0 からアプリのマルチタスキング、バックグラウンド機能が提供されている。
  そのひとつである Fast App Switching が有効になっているので、不要な場合は無効化する

- Xcode で **CIDR_IP-Info.plist** の編集を開始

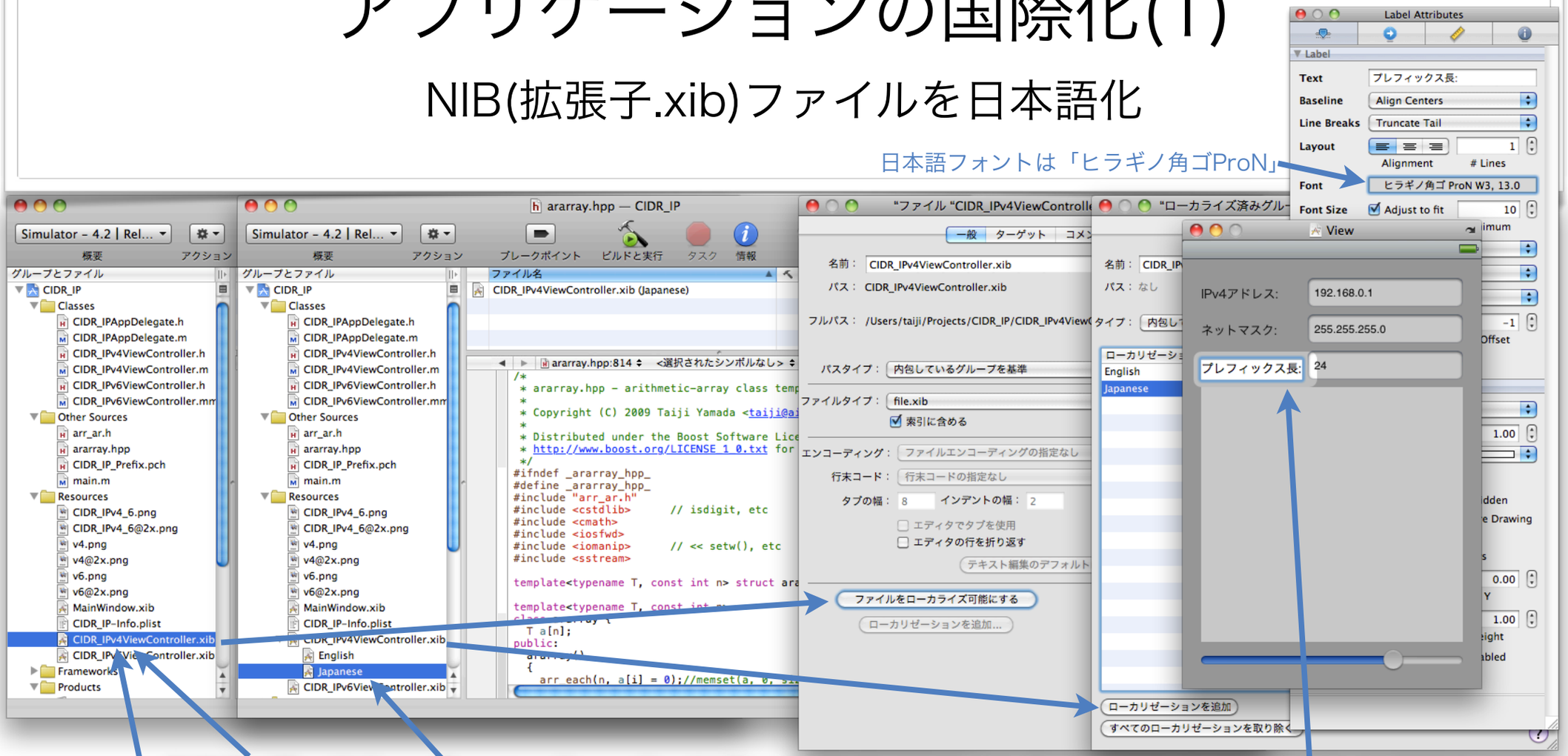
- エントリを「+」タブで追加し、Key を **Application does not run in background** にして Value にチェックを入れればよい ちなみにこの Key は plist ファイルをみると **UIApplicationExitsOnSuspend** となっている

⌘Rで実行して適当にいじり、シミュレータや実機のHomeボタンを押して、
再度アプリを実行すると、一旦終了して再度起動されたものと判る

アプリケーションの国際化(1)

NIB(拡張子.xib)ファイルを日本語化

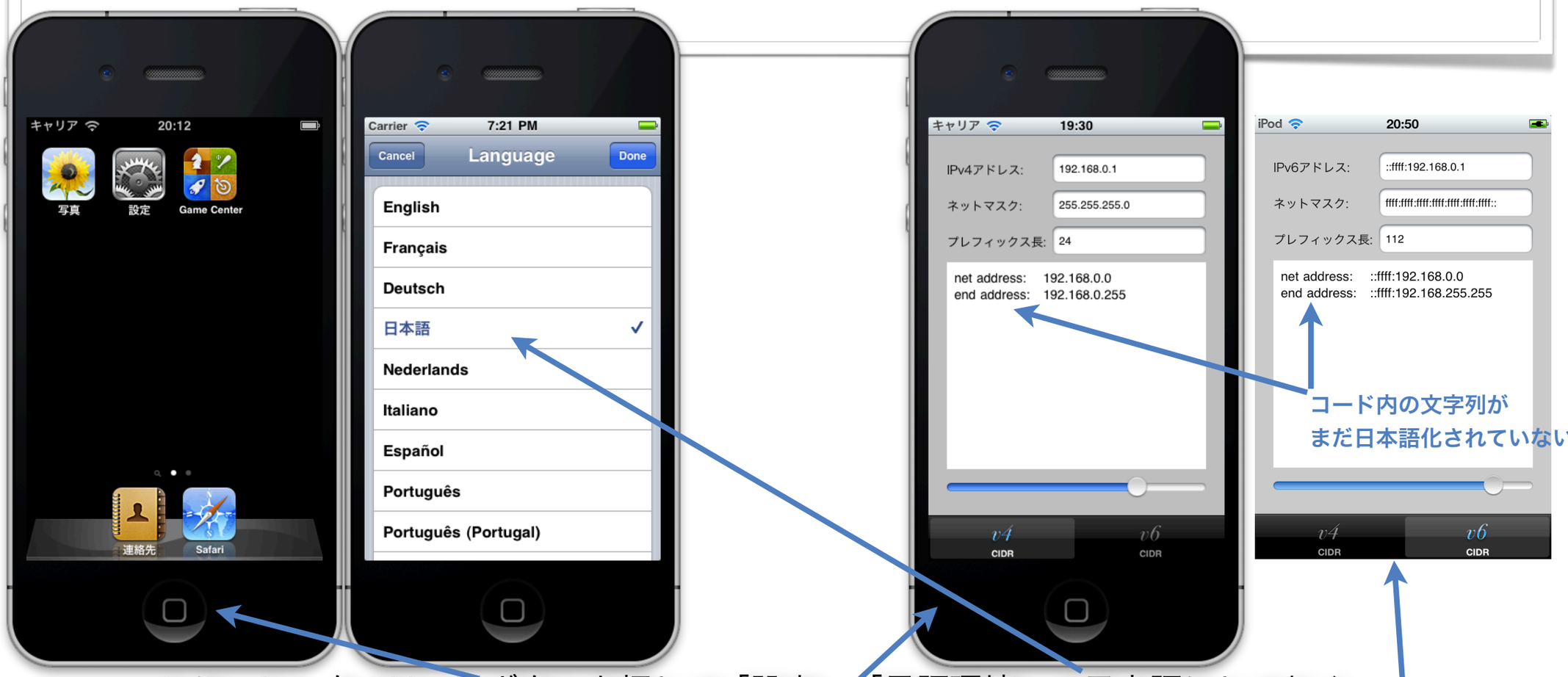
日本語フォントは「ヒラギノ角ゴProN」



- **CIDR_IPv4ViewController.xib** を選んで情報(⌘I)にて「ファイルをローカライズ可能にする」ボタンを押すと、それがグループ化される。
- **CIDR_IPv4ViewController.xib** グループを選んで情報(⌘I)にて「ローカリゼーションを追加」ボタンを押して、Japanese を指定する。
- **CIDR_IPv4ViewController.xib(Japanese)** を InterfaceBuilder で開いて日本語化する
  フォントの変更によりレイアウトに多少違和感を生じるので微調整
- **CIDR_IPv6ViewController.xib** についても同様に行う

アプリケーションの国際化(2)

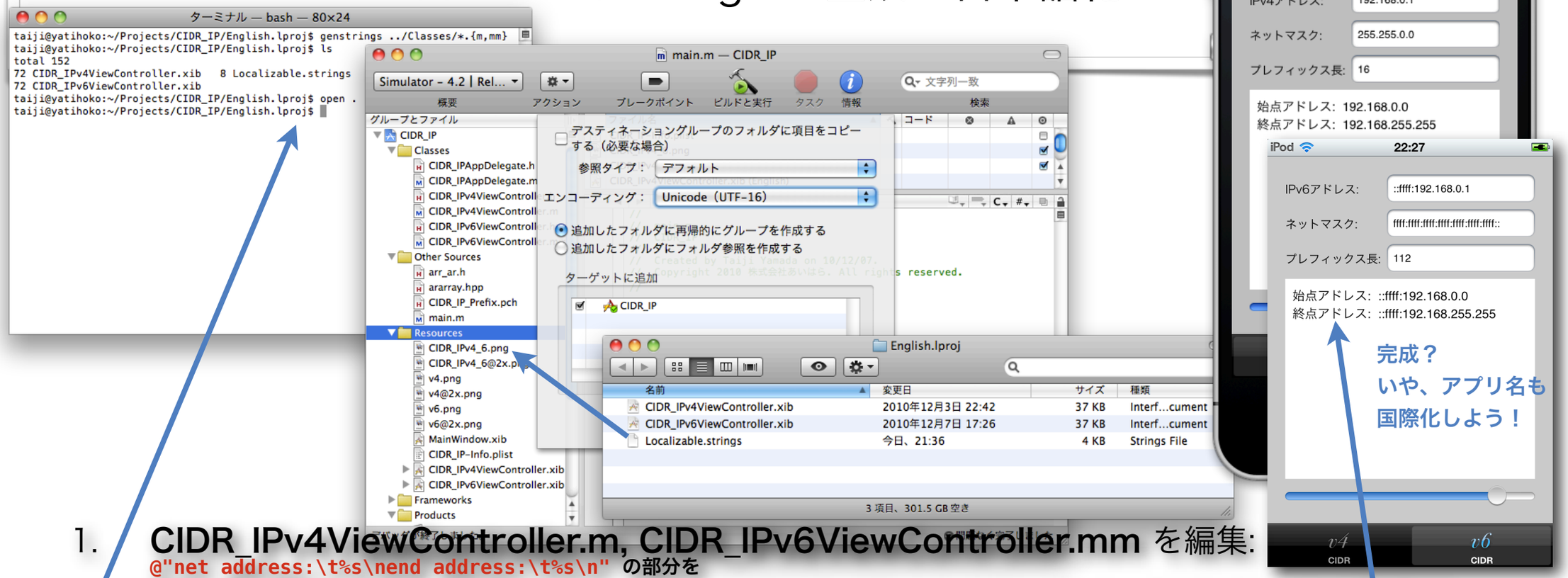
NIB(拡張子.xib)ファイルの日本語化をシミュレータと実機で確認



- シミュレータのHomeボタンを押して「設定」「言語環境」で日本語にしておく
- Xcode でメニュー、実行、すべてのターゲットのクリーニングをしてから⌘Rする
- それでも反映されないなら、シミュレータのメニューのすべてのコンテンツと設定をリセットしてから Xcode で⌘Rで実行
- 実機でも反映されないなら、Xcode のオーガナイザでアプリを Uninstall してから⌘R

アプリケーションの国際化(3)

Localizable.strings の生成と日本語化



1. **CIDR_IPv4ViewController.m, CIDR_IPv6ViewController.mm** を編集:
   `@"net address:\t%s\nend address:\t%s\n"` の部分を
   `NSLocalizedString(@"net address:\t%s\nend address:\t%s\n", @"format string of net and end address")` に置き換える

2. ターミナルで **~/Projects/CIDR_IP/English.lproj** ディレクトリにて、genstrings コマンドを使ってソースファイルから Localizable.strings ファイルを生成:
   `$ genstrings ../Classes/*.{m,mm}`

3. そのディレクトリを Finder で開いて Localizable.strings をプロジェクトの Resources に UTF-16 エンコーディングで追加、先のNIBと同様にそれをローカライズ可能にしてローカリゼーションの追加で Japanese を指定

4. Localization.string (Japanese) を編集:
   右辺値の `"net address:\t%1$s\nend address:\t%2$s\n"` を `"始点アドレス:\t%1$s\n終点アドレス:\t%2$s\n"` のようにする ⌘R！

アプリケーションの国際化(4)

InfoPlist.strings の作成と日本語化

1. Resources でヘルクリックして追加、新規ファイルで Mac OS X/Resource/Strings File テンプレートで次、ファイル名 InfoPlist.strings を追加
2. **InfoPlist.strings** を編集:
   `CFBundleDisplayName = "CIDR calc.";` という行を追加する。
3. それをローカライズ可能にしてローカリゼーションの追加で Japanese を指定
4. **InfoPlist.strings(Japanese)** を編集:
   `CFBundleDisplayName = "CIDR計算";` という行を追加する。

⌘R！

まとめ

Objective-C++ programming
create Tab Bar Application

1. IPv4/v6アドレス範囲の算出ツールを制作した。
2. IPv6に関しては、C++ のテンプレートライブラリを活用
3. Tab Bar Application は Tab アイテムに View を持つコントローラ
4. View のレイアウトは Parent のリサイズに対応できるよう工夫
5. 各種アイコンには推奨サイズが規定されているので注意！
6. アプリケーションをHomeボタンで終了させるべきはさせる
7. NIBとLocalizable.strings、InfoPlist.stringsの国際化を行った

参考文献

Objective-C++ programming
create Tab Bar Application

1. Apple Inc., “iPhoneアプリケーションプログラミングガイド,” 2010.
   バックグラウンド実行などデバイスに関する内容や簡単な国際化の方法の紹介

2. Apple Inc., “iPhone開発ガイド,” 2010. 簡単なiPhoneアプリ作成から
   Property List 編集の説明、iOS Developer Program の活用などの紹介

3. Apple Inc., “iPhoneヒューマンインターフェイスガイドライン,”
   2010. ウインドウのリサイズへの対応のほか、遵守すべき項目の紹介

4. Apple Inc., “Objective-Cプログラミング言語,” 2009. Objective-C プロ
   グラミングの基礎から C++ との混在についてわかりやすく紹介