De Tout Et De Rien

Tag - armv7

Entries feed - Comments feed

Thursday, May 6 2010

06May2010

Building OpenSSL for iPhone

- apple
- arm
- armv6
- armv7
- fat binary
- ipad
- iphone
- lipo
- openssl
- universal

With the latest iPhone SDK 4.0 release, Apple has introduced the concept of fat binaries to the iPhone/iPad platform. Now, applications for these devices are built for both ARMv6 and ARMv7 architectures.

This has an impact on the third-party libraries you may choose to use, especially the one that don't come with a XCode project; OpenSSL is one of them. This post describes the basic steps to build an universal version of OpenSSL to target the development of iPhone/iPad applications.

Getting OpenSSL

The OpenSSL website (http://www.openssl.org/) only offers source code packages. Be sure to get the latest one, as the library is evolving quickly. At the time of writing, the latest version is the 1.0.0.

Once you got the archive, uncompress it where you want. Open a Terminal and go the uncompressed folder.

As we want to target all the architecture, we need **THREE** different builds: the simulator (i386), ARMv6 and ARMv7.

Patching OpenSSL

Open the `./crypto/ui/ui_openssl.c` file and replace the line

```
static volatile sig_atomic_t intr_signal;
```

with

```
static volatile int intr_signal;
```

This is because the `sig_atomic_t` typedef does not exists for iPhone.

Building OpenSSL

Building for the simulator (i386)

Create a folder for the distribution, for example `iPhoneSimulator-i386`.

Launch the configuration:

```
./Configure BSD-generic32 --openssldir=iPhoneSimulator-i386
```

Now, edit the `Makefile` file and change:

```
CC= cc
```

```
CFLAG= -DOPENSSL_THREADS ...
```

with

```
CC= /Developer/Platforms/iPhoneSimulator.platform/Developer/usr/bin/gcc -arch i386
```

```
CFLAG= -isysroot
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.0.sdk -
DOPENSSL_THREADS ...
```

Once, all this is done, build the OpenSSL:

```
make; make install
```

Building for iPhoneOS (ARMv6)

Create a folder for the distribution, for example `iPhoneOS-armv6`.

Launch the configuration:

```
./Configure BSD-generic32 --openssldir=iPhoneOS-armv6
```

Now, edit the `Makefile` file and change:

```
CC= cc
```

```
CFLAG= -DOPENSSL_THREADS ...
```

with

```
CC= /Developer/Platforms/iPhoneOS.platform/Developer/usr/bin/gcc -arch armv6
```

```
CFLAG= -isysroot /Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS4.0.sdk
-DOPENSSL_THREADS ...
```

Once, all this is done, build the OpenSSL:

```
make; make install
```

Building for iPhoneOS (ARMv7)

Create a folder for the distribution, for example `iPhoneOS-armv7`.

Launch the configuration:

```
./Configure BSD-generic32 --openssldir=iPhoneOS-armv7
```

Now, edit the `Makefile` file and change:

```
CC= cc
```

```
CFLAG= -DOPENSSL_THREADS ...
```

with

```
CC= /Developer/Platforms/iPhoneOS.platform/Developer/usr/bin/gcc -arch armv7

CFLAG= -isysroot /Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS4.0.sdk
-DOPENSSL_THREADS ...
```

Once, all this is done, build the OpenSSL:

```
make; make install
```

Merging OpenSSL

Now that you have the three distinct architecture, a simple way to use them is to merge them.

Create a folder for the distribution, for example `iPhoneOS`, and two subfolders `iPhoneOS/include` and `iPhoneOS/lib`.

Copy the include files from one of the distribution:

@@cp -R iPhoneOS-armv7/include/ iPhoneOS/include

Build a fat binary for each static library:

```
lipo -create iPhoneSimulator-i386/lib/libcrypto.a iPhoneOS-armv6/lib/libcrypto.a
iPhoneOS-armv7/lib/libcrypto.a -output iPhoneOS/lib/libcrypto.a

lipo -create iPhoneSimulator-i386/lib/libssl.a iPhoneOS-armv6/lib/libssl.a iPhoneOS-
armv7/lib/libssl.a -output iPhoneOS/lib/libssl.a
```

Conclusion

The `iPhoneOS` now contains all you need to develop OpenSSL application for iPhone/iPad.