

## MacOSX

This wiki page is about S60 3rd edition development under Mac OS X. More exactly on an Intel Mac. This page doesn't cover every detail, just the ones that are not covered elsewhere on the Internet.

There is a [really good article](#) on creating a development environment on the Mac by Martin Storsjö. It also covers development on GNU/Linux. If you own a PowerPC based Macintosh you should look there for information. The only thing missing from that is details on how to do the same on an Intel Mac.

For Intel Macintosh computers you will need an additional step. That is compiling gcc yourself. The following steps were tested on Mac OS X Tiger 10.4.9 with gcc 4.0.1.

When creating the development environment, a script converts the SDK to a form which will work on the Unix platform. For compiling programs for the phone the environment will use mainly gcc but some tools are run with the help of [Wine](#). So for the whole process you will need an SDK, the gcc crosscompiler and Wine. This page only covers the build process of the gcc crosscompiler. For the other necessary steps read Martin's article.

## Building the gcc crosscompiler

We will use CodeSourcery's gcc crosscompiler. You can download it from [http://www.codesourcery.com/gnu\\_toolchains/arm/releases/2005Q1C](http://www.codesourcery.com/gnu_toolchains/arm/releases/2005Q1C). Choose the source format.

Put the downloaded package to a directory and uncompress it:

```
$ mkdir gnu-csl
$ cd gnu-csl
$ tar xvjf gnu-csl-arm-2005Q1C-arm-none-symbianelf.src.tar.bz2
```

The package contains an installing script but we will need to modify this and the directory structure. We will create some directories: *release-config* for the configuration files, *install* for the installed gcc crosscompiler, *log* for the log files, *obj* for the temporary files, and *pkg* and *src* for the compressed and uncompressed files. Do these with the following commands:

```
$ mkdir release-configs
$ mv ARM* release-configs/
$ mkdir install log obj pkg src
$ mv binutils-csl-arm-2005Q1C.tar.bz2 gcc-csl-arm-2005Q1C.tar.bz2 obj
$ cd obj
$ tar xvjf binutils-csl-arm-2005Q1C.tar.bz2
$ tar xvjf gcc-csl-arm-2005Q1C.tar.bz2
$ cd ..
$ mv obj/binutils-csl-arm-2005q1 obj/gcc-csl-arm/ src/
```

We will need to make the compiling tools available through the PATH environmental variable and we will also need to create a symlink for gcc with the name *i686-apple-darwin8-gcc* because the building script will look for it by this name. We will need root rights for this. The following commands will work with the 4.0.1 gcc version from XCode. On a different configuration you may need to change these. We will also set the path of the building script.

```
$ export PATH=/usr/libexec/gcc/i686-apple-darwin8/4.0.1:$PATH
$ sudo ln -s /usr/bin/i686-apple-darwin8-gcc-4.0.1 /usr/bin/i686-apple-darwin8-gcc
```

## MacOSX

```
$ export CSL_SCRIPTDIR=$(pwd)
```

The building script was created by CodeScourcery mainly for internal use. So we will need to modify some bits of it to work in a more general environment. I've put the necessary modifications in a patch file: [gnu-csl.patch](#).

```
$ patch -p1 < gnu-csl.patch
```

The only remaining thing is to set up the path variable where you want gcc to be installed. Edit *ARM-2005Q1.inc* under *release-config*. Add the following line (edited with your chosen installation path):

```
installdir="/Users/LacKac/SDK/csl-gcc"
```

Finally we have to run the building script with the correct arguments. You shouldn't change this command:

```
$ ./gnu-release -n -s $(pwd)/src -o $(pwd)/obj -p $(pwd)/pkg -i $(pwd)/install -l $(pwd)/log ARMS
```

Move the built gcc to the installation path you've set in the configuration file:

```
$ mv install /Users/LacKac/SDK/csl-gcc
```

With this we've completed the building process and we own a working crosscompiler. Use this one instead of the one provided in Martin's article

## Signing SIS files under Mac OS X

One other thing is needed if you want to be able to get from the source to the installation file for a 3rd edition phone. The *signsis* utility is not working with wine. For SIS signing we will use *ensymble* which was created mainly to help Python based development for the Symbian platform on Unix environments. You can download it from <http://ensymble.googlecode.com/>. Download the pre-squeezed one for your system's Python version (Mac OS X Tiger 10.4.9 has Python 2.3).

The program will ask for the password of the keyfile. For the this to work we will have to make it able to create popup window. Change the first line of the script file you download for the following one:

```
#!/usr/bin/env pythonw -W ignore::DeprecationWarning
```

You can use the program in the following way:

```
$ ensymble.py signsis --cert=mycert.cert --privkey=mykey.key HelloWorld.SIS HelloWorld.sisx
```

By all of these we made the main part of the development process available for [Intel Mac OS X computers](#).

--[LacKac](#) 10:00, 8 June 2007 (UTC)