Historical MacOS Installation Notes

Nowadays, installing FFTW on MacOS X is much like on any othe Unix system. In ancient times, a lot more work was required, and some of those obsolete instructions are preserved below for historical interest.

Installing FFTW on MacOS X

To install FFTW on MacOS X, all you should have the Apple developer tools installed, so that you can open up a terminal window and compile FFTW just as you would on any other Unix system: `./configure && make` to compile, and e.g. `sudo make install` to switch to `root` and install in `/usr/local`. (Note that `sudo` asks for *your* password, not `root`'s; alternatively, you can just use `su`, assuming that your `root` account is enabled (it isn't by default).)

The multi-threaded version (with POSIX threads) also works fine (`./configure --enable-threads`).

Precompiled MacOS X libraries via Fink

Instead of the above, the Fink project has put together precompiled/prepackaged versions of FFTW 2.x and FFTW 3.x for MacOS X.

Fink provides a collection of free-software Unix tools packaged for MacOS X, based on the excellent package and system-maintainance tools developed for Debian GNU/Linux

Precompiled packages for CodeWarrior (ancient compiler)

Greg Allen graciously posted a Mac package of FFTW 2.1.3 and BenchFFT, for CodeWarrior 5 I believe.

In the past, I had created precompiled packages of FFTW for Metrowerks CodeWarrior, including PPC and 68k libraries. Since my version of CodeWarrior (Pro 2) was becoming more and more out of date, I stopped doing this. The last version I packaged in this way was FFTW 2.0.1: fftw-2.0.1.sit.bin.

If you are interested in creating precompiled CodeWarrior packages of FFTW 3.x (e.g. for MacOS 9), please let us know (and give us a ride on your Tardis).

Compiling FFTW 2.x on MacOS 9

Compiling FFTW yourself on the MacOS is fairly straightforward. For example, this is the outline of the steps to compile the complex-transform library using CodeWarrior.

1. Download and unpack the FFTW archive (`.tar.gz` format); Stuffit Expander (available gratis) should have no problem with this. (Alternatively, you can download standalone gunzip and untar programs.)
2. Create an *empty* CodeWarrior project for the library, and drag the `fftw` subfolder of the FFTW package onto the project (this will add all the `.c` and `.h` files).
3. Go into the project preferences, change the project type to a library, and turn all the optimizations on. You will also also need to go to "Access Paths" and move the `fftw` folder

into the "System Paths" section (since our code includes it as `<fftw.h>`).
4. Compile.

To compile the corresponding test program (`fftw_test`), you'll create a "console ANSI C" project, adding the library created above and the files `fftw_test.c` and `test_main.c` (in the `tests` subfolder); you'll also need to modify the access paths as above.

Compiling the rfftw transforms is similar, except that you use the `rfftw` folder, and `rfftw_test.c` for the test program.

CodeWarrior Bugs

CodeWarrior Pro 4 reportedly generates incorrect code when compiling FFTW 2.x at the highest optimization level (level 4). Supposedly, this problem is fixed in CodeWarrior Pro 5 with all the latest updates applied. (Thanks to Dan Melomedman for the report, and for bugging Metrowerks about this.) We haven't heard of problems with other versions of CodeWarrior.

Using FFTW 2.x with the Absoft Compilers

Daniel Barth sent us a couple of notes regarding the compilation of FFTW 2.x using Absoft's C/C++ and Fortran compilers, version 6.2 (circa 2001).

First, the Absoft C compiler seems to have trouble with the Unix line endings (line feeds) in the source files. The source files can be converted to use Mac line endings (carriage returns) via a program like NetStripper.

Second, in order to link with Fortran programs using Absoft's ProFortran, add a `#define FFTW_FORTRANIZE_UPPERCASE 1` statement to the `fftw/config.h` file.

Go back to the FFTW download page.