imake Frequently Asked Questions

Paul DuBois
dubois@primate.wisc.edu

Wisconsin Regional Primate Research Center
Version 1.01
Revision date: 1 May 1997

Introduction

This document is the *imake* Frequently Asked Questions (FAQ) list. It tries to answer some of the commonly-asked questions about *imake*, a *Makefile* generator used for software configuration.

This document is available in the *imake-stuff* archive, which is located at:

```
http://www.primate.wisc.edu/software/imake-stuff/
ftp://ftp.primate.wisc.edu/software/imake-stuff/
```

If you have comments about this FAQ or suggestions for material to be included, please send them to the FAQ maintainer, Paul DuBois, dubois@primate.wisc.edu. At the moment, this FAQ is very new and in great need of suggestions for improvement.

General

What is imake? How does it work?

imake is a *Makefile*-generator that is intended to make it easier to develop software portably for multiple systems. Makefiles are inherently non-portable, so instead of writing them by hand, machine-dependencies are specified explicitly in a set of configuration files. Instead of writing a *Makefile*, you write an *Imakefile*, which is a machine-independent description of what targets you want to build. This way your description file doesn't need to change when you build your software on different systems. *imake* reads the Imakefile and combines the specifications in it with the proper machine dependencies from the configuration files to write a Makefile tailored for a specific system. The *Imakefile* and the configuration files are processed by *cpp*, so they are written using conventions that should be familiar to most C programmers.

What systems does imake run on?

imake runs primarily on UNIX sytems and Windows NT as distributed (see "Where can I get imake?"). There are also ports to other systems such as OS/2, AmigaOS, and reputedly OpenVMS (though I have not been able to confirm this). Port information is available at the *imake-stuff* archive.

Who owns imake? What is its redistribution status?

imake is a product of The Open Group:

```
The Open Group
11 Cambridge Center
Cambridge, MA 02142-1405
USA
http://www.opengroup.org/
```

imake is copyrighted by The Open Group, but is freely available and redistributable.

Where can I get imake?

imake is distributed as part of the X Window System (another product of The Open Group), which is available online at:

```
http://www.x.org/
ftp://ftp.x.org/
```

There are also several mirror sites; a list is available in the *GettingR6* file at the locations just shown.

If you don't want to get the X11 distribution, the *imake*-related parts of X11 are packaged as the *itools* distribution, which is available at:

```
http://www.primate.wisc.edu/software/imake-book/
ftp://ftp.primate.wisc.edu/software/imake-book/
```

You should use the most recent version of *imake* you can. The current version is the one distributed with X11R6.3. However, you need an ANSI C compiler to build it. If you don't have one, you may be able to build the version distributed with X11R6.1. There is also an R6.1-based *itools* distribution.

Many of the freely-available UNIX variants (e.g., FreeBSD, Linux) include *imake* in their standard distributions. Check the distribution for an X developer's package, since *imake* may not be installed if you install only the X server and X client packages.

Similarly, XFree86 includes *imake*, but it won't be installed unless you install the developer's part of the distribution.

What do I need in order to be able to run imake?

At a minimum, you need *imake* itself, a set of configuration files, *cpp*, *make*, and a bootstrapper such as *xmkmf* or *imboot*.

Also recommended are *makedepend*, *mkdirhier* (if your system doesn't support *mkdir −p*), and *bsdinst* or *install.sh* (if your system doesn't have a BSD-compatible install program).

All of these are available in the X11 or *itools* distributions.

Where can I find information about imake?

There are several sources of information about *imake*:

- The *imake* manual page
- O'Reilly & Associates publishes the "boa book" *Software Portability with imake*. There is an online archive at the book's home page, located at:

  ```
  http://www.primate.wisc.edu/software/imake-book/
  ftp://ftp.primate.wisc.edu/software/imake-book/
  ```

  The archive includes the Preface from the book and two appendixes, one of which contains the *itools* installation instructions.
- The *README* file in the *config/cf* directory of the X11 or *itools* distributions.
- The *imake-stuff* archive contains various papers, configuration files, and utilities:

  ```
  http://www.primate.wisc.edu/software/imake-stuff/
  ftp://ftp.primate.wisc.edu/software/imake-stuff/
  ```

- The imake-talk mailing list is available for questions and discussion. To subscribe, send an e-mail message to imake-talk-request@primate.wisc.edu that says:

```
subscribe imake-talk
```

Using imake

How do I build a project that has an Imakefile?

For projects that include an *Imakefile*, you need to run *imake* to generate the corresponding *Makefile*. However, you never run *imake* directly. Instead, you run a bootstrapper that invokes *imake* for you and passes to it any flags that are needed. (For instance, the bootstapper will tell *imake* what directory to look in to find the configuration files.)

If the project is for an X program, try generating the *Makefile* by running *xmkmf*. If that works, try *make* and *make install*. If you don't have *xmkmf*, see "Where can I get imake?"

If the program isn't an X program, then its documentation should say something about how to bootstrap the Makefiles.

To regenerate the *Makefile* (e.g., after making a change to the *Imakefile*), you can either run the bootstrapper again, or else run this command:

```
% make Makefile
```

If the project has multiple directories, you'll need to generate the Makefiles in any subdirectories:

```
% make Makefiles
```

I make a change to the Imakefile, but it's not reflected in the Makefile. Why?

You need to rerun *imake* to regenerate the *Makefile* from the modified *Imakefile*. See "How do I build a project that has an Imakefile?"

How do I write an Imakefile?

That depends on the configuration files you're using, although the general principles are similar for any set of configuration files. Probably the easiest way to write an *Imakefile* is to find a project similar to yours, and modify a copy of the *Imakefile* from that project. For further reading, the *config/cf/README* file in the X11 or *itools* distributions contains some tips on *Imakefile* writing, and the boa book discusses this topic extensively.

Problems Generating Makefiles

I get this error: "Imake.tmpl: no such file or directory"

This message means that *imake* is unable to find the configuration files. (The message names *Imake.tmpl* because that is the first file that *imake* tries to read.) This error message can occur for several reasons:

- You're trying to generate a *Makefile* by invoking *imake* directly, rather than by using a bootstrapper such as *xmkmf* or *imboot*. Consult the documentation for the project you're trying to build to find out which bootstrapper to use. If your program is X-based, you should probably generate the initial *Makefile* by running *xmkmf*.
- You're using a bootstrapper, but it's misconfigured. Bootstrappers typically pass one or more *−Idir* arguments to *imake* to tell it where the configuration files are located. If the bootstrapper doesn't pass the correct location(s), *imake* won't find the files. To fix this, you need to take a look at your bootstrapper (it's probably a script and can be examined using a text editor), and change the location(s) that it passes to *imake*.

- You may simply have no configuration files installed. In this case, you'll need to get the set required by the project you're trying to build, and install them.

I get XCOMM symbols all through my Makefile and make complains about them. What's going on?

`XCOMM` is used in Imakefiles and configuration files to write lines that should end up in Makefiles as comments. For instance, this line:

```
XCOMM this is a comment
```

should end up as:

```
# this is a comment
```

The `XCOMM` symbol is supposed to be translated to # during *Makefile* generation.

If translation fails and you see `XCOMM` symbols in your *Makefile*, you're probably using a newer set of configuration files (e.g., from X11R6 or later) that expect `XCOMM` to be translated to # by *imake* and an older version of *imake* (e.g., from X11R5 or before) that expects the configuration files to handle the translation. The solution is to upgrade to a version of *imake* from X11R6 or later.

Why can't imake find cpp under AIX?

Try setting the environment variable `IMAKECPP` to `/usr/lpp/X11/Xamples/util/cpp/cpp`.

When I run "make install", it complains about not being able to find mkdirhier.sh.

Install *mkdirhier.sh* in a public directory as *mkdirhier* (no *.sh* suffix), make it executable, and change `MKDIRHIER` in *Imake.tmpl* from this:

```
#ifdef UseInstalled
    ... other stuff ...
    MKDIRHIER = BourneShell $(BINDIR)/mkdirhier
#else
    ... other stuff ...
    MKDIRHIER = BourneShell $(SCRIPTSRC)/mkdirhier.sh
#endif
```

to this:

```
    MKDIRHIER = mkdirhier
#ifdef UseInstalled
    ... other stuff ...
#else
    ... other stuff ...
#endif
```

Another solution is to upgrade to a more recent set of configuration files, since this problem occurs with the X11R4 and X11R5 configuration files, but has been fixed in later X11 releases.

How can I get imake to work with my OpenWindows setup?

See Appendix J of the boa book (2nd edition) or check the *imake-stuff* archive for the "imake vs. OpenWindows" documentation. The documentation describes why OpenWindows systems have problems with *imake* and how to fix them. The archive also contains a software distribution that makes it easier to use *imake* with OpenWindows.

How can I use gcc with OpenWindows imake support?

See previous question. The software distribution mentioned there contains configuration file patches for *gcc* support.

How can I build Motif 2.x using imake?

Check the *imake-stuff* archive for the "Motif Support" documentation.

Debugging imake

How can I see what flags imake passes to cpp?

Try this command:

```
% imake -v -s/dev/null -f/dev/null -T/dev/null
```

If your shell supports `{...}` expansion, the following command is equivalent but simpler to type:

```
% imake -v -{s,f,T}/dev/null
```

How can I see what imake writes to Imakefile.c?

When *imake* runs, it creates a file *Imakefile.c* that contains the initial input to be given to *cpp*. (This assumes you're running an X11R6 or later version of *imake*). Sometimes it's useful to be able to see what is being passed to *cpp*, but *imake* removes *Imakefile.c* when *cpp* terminates.

However, if you create *Imakefile.c* as a link to another file before running *imake*, then that file will still exist and contain the contents of *Imakefile.c* after *imake* runs, even though *Imakefile.c* will have been removed. Then you can look at the file to see what's in it.

You can use this trick with a sequence of commands such as the following:

```
% touch fake
% ln fake Imakefile.c
% make Makefile                (or xmkmf, imboot, etc.)
% more fake
```

This works by creating *Imakefile.c* as a link to a file *fake* before running *imake*. *imake* writes the contents of *Imakefile.c*, then removes it later, but since *fake* is a link to *Imakefile.c*, it remains in existence after *imake* runs and you can take a look at it.

Note that you'll need to re-run the *ln* command each time you run *imake* to be able to see what gets written to *Imakefile.c*.