Building XFree86 from a Source Distribution

David Dawes, Matthieu Herrb

27 May 2001

Abstract

This document describes how to build XFree86 from the **source** distribution. It is designed to be used in conjunction with the OS-specific README files.

NOTE: Refer to the appropriate OS-specific README file before attempting to build XFree86. These files contain additional information that you may need to successfully build under your OS.

We highly recommend using gcc to build XFree86, but it generally also builds with the native compiler for each platform;

1. How to get the XFree86 4.2.1 source

There are a few starting points for getting the XFree86 source. One option is to start directly with the XFree86 4.2.0 source distribution and updating it to 4.2.1 with a source patch. In this case, the procedure is as follows:

- The XFree86 source is contained in files `X420src-1.tgz`, `X420src-2.tgz` and `X420src-3.tgz`. These can be found at ftp://ftp.xfree86.org/pub/XFree86/4.2.0/source/ and similar locations on XFree86 mirror sites. `X420src-2.tgz` contains the fonts and documentation source. `X420src-3.tgz` contains the hardcopy documentation. `X420src-1.tgz` contains everything else. If you don't need the docs or fonts you can get by with only `X420src-1.tgz`.
- Extract each of these files by running the following from a directory on a filesystem containing enough space (the full source requires around 270MB, and a similar amount is required in addition to this for the compiled binaries):

```
gzip -d < X420src-1.tgz | tar vxf -

gzip -d < X420src-2.tgz | tar vxf -

gzip -d < X420src-3.tgz | tar vxf -
```

- A patch relative to 4.2.0 is available at ftp://ftp.xfree86.org/pub/XFree86/4.2.1/patches/. The patch file is `4.2.0-4.2.1.diff.gz`.

The patch can be applied by running:

cd *the directory containing the* xc *directory*

gzip -d < 4.2.0-4.2.1.diff.gz | patch -s -p0 -E

Look for special patching instructions in the Release Notes.

Another option is to get the source by anonymous CVS or CVSup. See http://www.xfree86.org/cvs/ for details on the different procedure available.

All methods will produce one main source directory called `xc`.

2. Configuring the source before building

It is recommended that you start the configuration process by going to the `xc/config/cf` directory, and copying the file `xf86site.def` to `host.def`. Then read through the `host.def` file (which is heavily commented), and set any parameters that you want for your configuration. You can usually find out what the default settings are by checking the `.cf` file(s) relevant to your OS.

Unlike previous versions, imake can now automatically detect and set the various **OS*Version** parameters, so you shouldn't need to enter those settings explicitly.

If you are using just the `X421src-1.tgz` part of the source dist, you will need to define **BuildFonts** to **NO**.

3. Using a shadow directory of symbolic links for the build

A recommended practice is to use a shadow directory of symbolic links to do the build of XFree86. This permits to keep the source directory unmodified during the build, which has the following benefits:

- In the case where CVS is used to maintain the source tree up to date, the update process is not disturbed by foreign files not under the control of CVS.
- It is possible to build XFree86 for several different Operating System or architectures from the same sources, shared by NFS.
- It is possible to build XFree86 with different configuration options, just by putting a real copy`host.def` in each build tree and by customizing it separately in each build tree.

To make a shadow directory of symbolic links, use the following steps:

- create the directory at the top of the build tree. It is often created at the same level that the `xc` directory, but this is not mandatory.

  cd *the directory containing the* xc*directory*

  mkdir build

- use the `lndir`command to make the shadow tree:

  ```
  lndir ../xc
  ```

  Note that you can refer to the `xc` directory with an absolute path if needed.

  See the lndir(1) manual page for details.

If `lndir` is not already installed on your system, you can build it manually from the XFree86

sources by running the following commands:

```
cd xc/config/util

make -f Makefile.ini lndir

cp lndir some directory in your PATH
```

4. Building and installing the distribution

Before building the distribution, read through the OS-specific `README` file in `xc/programs/Xserver/hw/xfree86/doc` that is relevant to you. Once those OS-specific details have been taken care of, go your build directory (either the `xc` directory or the shadow tree created before) and run "`make World`" with the **BOOTSTRAPCFLAGS** set as described in the OS-specific README (if necessary, but most systems supported by XFree86 don't need **BOOTSTRAPCFLAGS**). It is advisable to redirect stdout and stderr to `World.Log` so that you can track down problems that might occur during the build.

With Bourne-like shells (Bash, the Korn shell, `zsh`, etc.) use a command like:

```
make World > World.log 2>&1
```

Witch C-shell variants (`csh`, `tcsh`, etc), use:

```
make World >& World.log
```

You can follow the progress of the build by running:

```
tail -f World.log
```

in a terminal.

When the build is finished, you should check `World.Log` to see if there were any problems. If there weren't any then you can install the binaries. To do the install, run "`make install`" and "`make install.man`". Make sure you have enough space in `/usr/X11R6` for the install to succeed. If you want to install on a filesystem other than `/usr`, make a symbolic link to `/usr/X11R6` before installing.

5. Reconfiguring the server (source distribution)

To build a different set of servers or servers with a different set of drivers installed:

1. Make sure the source for any new drivers is in the correct place (e.g., driver source should be in a subdirectory of `xc/programs/Xserver/hw/xfree86/drivers`).
2. Change the settings of the server defines in `host.def` to specify which servers you wish to build. Also, change the driver lists to suit your needs.
3. From `xc/programs/Xserver`, run:

```
make Makefile
make Makefiles
 make includes
make depend
make
```

6. Other useful make targets

There are some other useful targets defined in the top level `Makefile`of XFree86:

- **Everything** after a `make World`, `make Everything`does everything a `make World` does, except the cleaning of the tree. It is a way to quickly rebuild the tree after a source patch, but it is not 100% bullet proof. There are cases were it is better to force a full build by using `make World`.

- **clean** does a partial cleaning of the source tree. Removes object files and generated manual pages, but leaves the Makefiles and the generated dependencies files in place. After a `make clean` you need to re-run

```
make includes
make depend
make
```

  to rebuild the XFree86.

- **distclean** does a full cleaning of the source tree, removing all generated files. After a `make distclean`, `make World` is the only option to rebuild XFree86.
- **includes** generates all generated header files and in-tree symbolic links needed by the build. These files are removed by a `make clean`.
- **depend** recomputes the dependencies for the various targets in all Makefiles. Depending on the operating system, the dependencies are stored in the Makefile, or as a separate file, called `.depend`. This target needs the generated include files produced by `make includes`.
- **VerifyOS** displays the detected operating system version. If the numbers shown do not match your system, you probably need to set them manually in `host.def` and report the problem to XFree86@XFree86.org.

`$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/BUILD.sgml,v 3.6.4.1 2002/09/04 03:33:11 dawes Exp $`

CONTENTS

1. How to get the XFree86 4.2.1 source 1
2. Configuring the source before building 2
3. Using a shadow directory of symbolic links for the build 2
4. Building and installing the distribution 3
5. Reconfiguring the server (source distribution) 3
6. Other useful make targets 4