Building XFree86® from a Source Distribution

David Dawes, Matthieu Herrb

16 March 2005

Abstract

This document describes how to build XFree86 from the **source** distribution and is designed to be used in conjunction with the operating system (OS) specific README files.

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

We recommend using gcc to build XFree86, but XFree86 generally builds with the native compiler for each OS platform.

1. How to get the XFree86 4.6.0 source

The recommended way of getting the XFree86 4.6.0 source is to obtain it directly from the XFree86 CVS repository. There are several ways of doing that, and they are described at our CVS web page `<URL:http://www.xfree86.org/cvs/>`.

The CVS tag for this release is "`xf-4_6_0`". The tag for the maintenance branch for this release is "`xf-4_6-branch`".

Another method of getting the XFree86 4.6.0 source is to either download the 4.6.0 source tarballs from the XFree86 ftp site. The procedure for this is as follows:

- The XFree86 4.6.0 source is contained in the files:

```
XFree86-4.6.0-src-1.tgz

XFree86-4.6.0-src-2.tgz

XFree86-4.6.0-src-3.tgz

XFree86-4.6.0-src-4.tgz

XFree86-4.6.0-src-5.tgz

XFree86-4.6.0-src-6.tgz

XFree86-4.6.0-src-7.tgz
```

These can be found at ftp://ftp.xfree86.org/pub/XFree86/4.6.0/source/. `XFree86-4.6.0-src-4.tgz` and `XFree86-4.6.0-src-5.tgz` contains the fonts. `XFree86-4.6.0-src-6.tgz` contains the documentation source. `XFree86-4.6.0-src-7.tgz` contains the hardcopy documentation. `XFree86-4.6.0-src-1.tgz`, `XFree86-4.6.0-src-2.tgz` and `XFree86-4.6.0-src-3.tgz` contains everything else.

If you do not need either the documentation or the fonts, then you need only `XFree86-4.6.0-src-1.tgz`, `XFree86-4.6.0-src-2.tgz` and `XFree86-4.6.0-src-3.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, with a similar amount being required for the compiled binaries):

```
gzip -d < XFree86-4.6.0-src-1.tgz | tar vxf -

gzip -d < XFree86-4.6.0-src-2.tgz | tar vxf -

gzip -d < XFree86-4.6.0-src-3.tgz | tar vxf -

gzip -d < XFree86-4.6.0-src-4.tgz | tar vxf -

gzip -d < XFree86-4.6.0-src-5.tgz | tar vxf -

gzip -d < XFree86-4.6.0-src-6.tgz | tar vxf -

gzip -d < XFree86-4.6.0-src-7.tgz | tar vxf -
```

Alternatively, if you already have a pristine copy of the XFree86 4.5.0 source, you can download patches from ftp://ftp.xfree86.org/pub/XFree86/4.6.0/patches/ that will allow you to convert it to 4.6.0. Information about which patch files to download and how to apply them can be found in the "How to get XFree86" section of the README for this release.

Each of the methods outlined above will produce one main source directory called `xc`.

2. Configuring the source before building

In most cases it should not be necessary to configure anything before building.

If you do want to make configuration changes, you should start by going to the `xc/config/cf` directory, and copying the file `xf86site.def` to `host.def`. After that, read thoroughly the the `host.def` file, which is heavily commented, and set your configuration parameters. Usually you can find the default settings by checking the `.cf` file(s) relevant to your OS.

A good rule of thumb is to only change that you understand, as it is very easy to create build problems by changing the default configuration unnecessarily. Before making too many modifications, check the configuration parameters specified in the `xc/config/cf/README` file.

If you are using just the `XFree86-4.6.0-src-1.tgz`, `XFree86-4.6.0-src-2.tgz` and `XFree86-4.6.0-src-3.tgz` parts of the source dist, you will need to define **BuildFonts** to **NO**.

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

We recommend that you use a shadow directory of symbolic links to do the build of XFree86, because it allows you to keep the source directory unmodified during the build process. It has the following benefits:

- Foreign files not under CVS's control are not touched.
- Greater flexibility in building XFree86 for several different Operating Systems or hardware architectures from the same sources; shared by read-only NFS mounts.
- Different configuration options can be created by putting a real copy of the `host.def` file in each build tree and by customizing it separately in each build tree, and then customizing it separately.

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

- create the directory at the top of the build tree. This is often created at the same level that the `xc` directory.

  `cd` *the directory containing the xc directory*

  mkdir build

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

  ```
  cd build

  lndir ../xc
  ```

  Note: You can refer to the `xc` directory by referencing it with an absolute path also.

  See the lndir(1) manual page for details.

If the `lndir` command is not already available 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
```

Occasionally there may be stale links in the build tree. This happens when files in the source tree are either removed or renamed. These stale links can be cleaned by running the "`cleanlinks`" script from the build directory (see the cleanlinks(1) manual page for further details).

Rarely will there be changes that require the build tree to be re-created from scratch. If you think that this may be the case, look for build problems, which could indicate that this is the problem. The best solution in this scenario is to remove the build tree,and then re-create it using the steps outlined above.

4. Building and installing the distribution

Before building the distribution, read through your OS-specific documentation in `README` file in `xc/programs/Xserver/hw/xfree86/doc`. After that go to your build directory which will either be the `xc` directory or the shadow tree which was create previously, and run "`make World`".

It is also advisable to that you redirect standard output *stdout* and standard error output *stderr* to `World.Log`, so that you can track down problems that might occur during the build.

With Bourne-like shells (examples include the Bash, the Korn, and `zsh` shells) use the following command:

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

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

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

You can then follow the progress of the build by running:

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

in a terminal window.

When the build is finished, you should check the `World.Log` file to see if there were any problems. If there were not, then install the binaries. By default "make World" stops when the first error is found. To restart that same build process after correcting the various problems, run just 'make'.

If Imakefiles or part of the build configuration were changed as part of correcting the problem, then either re-run "make World", or run "make Everything".

If instead, you want "make World" to continue past errors, then instead execute it as follows:

for Bourne-like shells:

```
make WORLDOPTS=-k World > World.log 2>&1
```

for C-shell variants:

```
make WORLDOPTS=-k World >& World.log
```

For installation, run "`make install`" and "`make install.man`" and ensure that there is enough space in the `/usr/X11R6` directory.

If instead you want to install XFree86 on a filesystem other than `/usr`, make a symbolic link to `/usr/X11R6` before running the install process.

5. Reconfiguring the server (using the source distribution)

If you would like to build several different sets of servers or server(s) with a various sets of drivers then you should follow this procedure:

1. Ensure that the source for any new drivers is in the correct place (e.g., all driver source should be in a subdirectory of `xc/programs/Xserver/hw/xfree86/drivers`).
2. Change the settings of the server definitions 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` replicates `make World` except for the cleaning of the source tree. `make Everything` very quickly rebuilds the tree after a source patch, but there are times when it is better to force a full build by using `make World`.

clean
: This 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
: This 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
: This generates all generated header files and in-tree symbolic links needed by the build. These files are removed by a `make clean`.

depend
: This 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
: This 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>.

CONTENTS

$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/BUILD.sgml,v 3.20 2006/04/12 01:39:22 dawes Exp $