Building and Installing the X Window System

Stephen Gildea

X Consortium

March 5, 1996

Updated For Release 6.3

Table of Contents

1. Easy Build Instructions

This quick summary is no substitute for reading the full build instructions later in this document.

Edit **xc/config/cf/site.def** for local preferences. If you want to build with *gcc* uncomment the **HasGcc2** line. If you want to install somewhere other than **/usr/X11R6.3**, change **ProjectRoot**. (Do *not* use **DESTDIR**.)

If any fixes have been released by the X Consortium, stop here and follow the instructions at the top of each patch, but don't do any of the *make* commands suggested in the patches. Then continue here.

Check the appropriate vendor-specific **.cf** file in **xc/config/cf/** to make sure that **OSMajorVersion** and **OSMinorVersion** are set correctly for your system. Override them in **site.def** if necessary.

See if there is a **BootstrapCFlags** mentioned in the comments in the vendor-specific **.cf** file. If there isn't one, *cd* to the **xc** directory and type:

    % make World >& world.log

If there is a **BootstrapCFlags**, take its value and type:

    % make World BOOTSTRAPCFLAGS="value" >& world.log

Do not call the output file "make.log" when doing "make World". After a successful build, you can install with:

    % make install >& install.log

You can install manual pages with:

    % make install.man >& man.log

While the system is building (or if things fail), read the rest of these installation instructions.

2. Building X

This document gives detailed instructions for building Release 6: getting it off the distribution medium, configuring, compiling, installing, running, and updating.

Release Notes are in **xc/RELNOTES.*** (various formats) in the distribution.

More recent information about newly-discovered problems may be found in the *Frequently Asked Questions* posting appearing monthly on the comp.windows.x newsgroup and xpert mailing list. It is also available via anonymous FTP on **ftp.x.org** in the file **contrib/faqs/FAQ.Z**, or on your local X mirror site.

2.1. Preparing the Site

If you are unpacking tar files, you will need about 130 Mb to hold the **xc/** part. To install requires 30-50 Mb assuming you have shared libraries (80-100 Mb without). You will need an equivalent amount of extra space to build, since you also need room for all the object files.

Distributed as tar files, Release 6.3 core is divided into parts as follows:

xc-1.tar	contains everything in xc/ that isn't in the other tar files
xc-2.tar	contains xc/fonts, xc/doc/specs, xc/util
xc-3.tar	contains xc/doc/hardcopy

If you define BuildFonts to NO, you only need to unpack xc-1.tar to build. If you build fonts, then you will also need xc-2.tar to build.

2.2. Unpacking the Distribution

The distribution normally comes as multiple tar files, either on tape or across a network, or as a CD-ROM.

2.2.1. Unpacking a Compressed FTP Distribution

If you have obtained compressed tar files over the network, create a directory to hold the sources and *cd* into it:

mkdir *sourcedir*
cd *sourcedir*

Then for each tar file **xc−*.tar.Z**, execute this:

zcat *ftp-dir*/xc−*N*.tar.Z | tar xf −

2.2.2. Unpacking a gzipped FTP Distribution

If you have obtained gzipped tar files over the network, create a directory to hold the sources and *cd* into it:

mkdir *sourcedir*
cd *sourcedir*

Then for each tar file **xc−*.tar.gz**, execute this:

gunzip −c *ftp-dir*/xc−*N*.tar.gz | tar xf −

2.2.3. Unpacking a Split Compressed FTP Distribution

If you have obtained compressed and split tar files over the network, create a directory to hold the sources:

mkdir *sourcedir*

Then for each directory **xc−***:

cd *ftp-dir*/xc−*N*
cat xc−*N*.?? | uncompress | (cd *sourcedir* ; tar xf −)

2.2.4. Unpacking the Tape Distribution

If you have obtained a tape, create a directory to hold the sources and untar everything into that directory:

> mkdir *sourcedir*
> cd *sourcedir*
> tar xf *tape-device*

2.2.5. Using the CD-ROM

If you have obtained a CD-ROM, you don't have to do anything to unpack it. However, you will have to create a symbolic link tree to build X. See the next section.

To mount the CD-ROM, see the mount(8) manual page on your system or the liner notes that came with the CD-ROM. Some systems, e.g., Solaris 2, can automatically mount the CD-ROM for you.

2.3. Apply Patches

If there are fixes released that are more recent than your distribution, apply them now. Follow the instructions at the top of each patch, but don't do any make commands. See the section "Public Patches" later in this document. Then continue here.

2.4. Symbolic Link Trees

If you expect to build the distribution on more than one machine using a shared source tree, or you are building from CD-ROM, or you just want to keep the source tree pure, you may want to use the program **xc/config/util/lndir.c** to create a symbolic link tree on each build machine. The links may use an additional 10 megabytes, but it is cheaper than having multiple copies of the source tree.

It may be tricky to compile *lndir* before the distribution is built. If you have a copy from a previous release, use that. **Makefile.ini** can be used for building *lndir* the first time. You may have to specify **OSFLAGS=−D***something* to get it to compile. What you would pass as **BOOTSTRAPCFLAGS** might work. The command line looks something like this:

> make −f Makefile.ini OSFLAGS=−D*flag*

To use a symbolic link tree, create a directory for the build, *cd* to it, and type this:

> lndir *sourcedir*

where *sourcedir* is the pathname of the directory where you stored the sources. All of the build instructions given below should then be done in the build directory on each machine, rather than in the source directory.

xc/config/util/mkshadow/ contains *mkshadow*, an alternative program to *lndir*.

2.5. Configuration Parameters

Build information for each source directory is in files called **Imakefile**. An **Imakefile**, along with local configuration information in **xc/config/cf/**, is used by the program *imake* to generate a **Makefile**.

Most of the configuration work prior to building the release is to set parameters so that *imake* will generate correct files. Most of those parameters are set in **xc/config/cf/site.def**. You will also need to check the appropriate vendor-specific **.cf** file in **xc/config/cf/** to make sure that OSMajorVersion, OSMinorVersion, and OsTeenyVersion are set correctly for your system. Override them in **site.def** if necessary.

The **site.def** file has two parts, one protected with "#ifdef BeforeVendorCF" and one with "#ifdef AfterVendorCF". The file is actually processed twice, once before the **.cf** file and once after. About the only thing you need to set in the "before" section is **HasGcc2**; just about everything else can be set in the "after" section.

The sample **site.def** also has commented out support to include another file, **host.def**. This scheme may be useful if you want to set most parameters site-wide, but some parameters vary from machine to machine. If you use a symbolic link tree, you can share **site.def** across all machines, and give each machine its own copy of **host.def**.

The config parameters are listed in **xc/config/cf/README**, but here are some of the more common parameters that you may wish to set in **site.def**.

ProjectRoot
: The destination where X will be installed. This variable needs to be set before you build, as some programs that read files at run-time have the installation directory compiled in to them. Assuming you have set the variable to some value /*path*, files will be installed into /*path*/bin, /*path*/include/X11, /*path*/lib, and /*path*/man.

HasGcc
: Set to **YES** to build with *gcc* version 1.

HasGcc2
: Set to **YES** to build with *gcc* version 2. Both this option and **HasGcc** look for a compiler named *gcc*, but **HasGcc2** will cause the build to use more features of *gcc* 2, such as the ability to compile shared libraries.

BuildXInputExt
: Set to **YES** to build the X Input Extension. This extension requires device-dependent support in the X server, which exists only in *Xhp* in our implementation.

BuildPexExt
: Set to **NO** to not build the PEX server extension and fonts.

DefaultUsrBin
: This is a directory where programs will be found even if PATH is not set in the environment. It is independent of ProjectRoot and defaults to **/usr/bin**. It is used, for example, when connecting from a remote system via *rsh*. The *rstart* program installs its server in this directory.

InstallServerSetUID
: Some systems require the X server to run as root to access the devices it needs. If you are on such a system and will not be using *xdm*, you can set this variable to **YES** to install the X server setuid to root. Note that the X server has not been analyzed by the X Consortium for security in such an installation; talk to your system manager before setting this variable.

InstallXdmConfig
: By default set to NO, which suppresses installing xdm config files over existing ones. Leave it set to NO if your site has customized the files in **/usr/X11R6.3/lib/X11/xdm**, as many sites do. If you don't install the new files, merge any changes present in the new files.

MotifBC
: Causes Xlib and Xt to work around some bugs in older versions of Motif. Set to **YES** only if you will be linking with Motif version 1.1.1, 1.1.2, or 1.1.3.

GetValuesBC

Setting this variable to **YES** allows illegal XtGetValues requests with NULL ArgVal to usually succeed, as R5 did. Some applications erroneously rely on this behavior. Support for this will be removed in a future release.

The following vendor-specific **.cf** files are in the release but have not been tested recently and hence probably need changes to work: **apollo.cf**, **bsd.cf**, **convex.cf**, **DGUX.cf**, **luna.cf**, **macII.cf**, **Mips.cf**, **moto.cf**, **Oki.cf**, **pegasus.cf**, **x386.cf**. **Amoeba.cf** is known to require additional patches.

The file **xc/lib/Xdmcp/Wraphelp.c**, for XDM-AUTHORIZATION-1, is not included in this release.

2.6. System Build Notes

This section contains hints on building X with specific compilers and operating systems.

If the build isn't finding things right, make sure you are using a compiler for your operating system. For example, a pre-compiled *gcc* for a different OS will not have right symbols defined, so *imake* will not work correctly.

2.6.1. gcc

gcc version 2 is in regular use at the X Consortium on Sparc platforms. Set the variable **HasGcc2**. X will not compile on some systems with *gcc* version 2.5, 2.5.1, or 2.5.2 because of an incorrect declaration of memmove() in a gcc include file.

If you are using a *gcc* version older than 2.7 on Solaris x86, you need to specify **BOOTSTRAPCFLAGS="−Dsun"** in the “make World” command.

2.6.2. Other GNU tools

Use of the GNU assembler, *as*, or linker, *ld*, is not supported. GNU *make* is not supported.

2.6.3. SparcWorks 2.0

If you have a non-threaded program and want to debug it with the old SparcWorks 2.0 dbx, you will need to use the thread stubs library in **xc/util/misc/thr_stubs.c**. Compile it as follows:

```
cc -c thr_stubs.c
ar cq libthr_stubs.a thr_stubs.o
ranlib libthr_stubs.a
```

Install libthr_stubs.a in the same directory with your X libraries (e.g., **/usr/X11R6.3/lib/libthr_stubs.a**). Add the following line to **site.def**:

```
#define ExtraLibraries −lsocket −lnsl $(CDEBUGFLAGS:−g=−lthr_stubs)
```

This example uses a *make* macro substitution; not all *make* implementations support this feature.

2.6.4. CenterLine C under Solaris 2

If you are using the CenterLine C compiler to compile the distribution under Solaris 2, place the following line in your **site.def**:

```
#define HasCenterLineC YES
```

If clcc is not in your default search path, add this line to **site.def**:

```
#define CcCmd /path/to/your/clcc
```

If you are using CodeCenter 4.0.4 or earlier, the following files trigger bugs in the *clcc* optimizer:

```
xc/programs/Xserver/cfb16/cfbgetsp.c
xc/programs/Xserver/cfb16/cfbfillsp.c
xc/programs/Xserver/cfb/cfbgetsp.c
```

Thus to build the server, you will have to compile these files by hand with the **−g** flag:

```
% cd xc/programs/Xserver/cfb16
% make CDEBUGFLAGS="−g" cfbgetsp.o cfbfillsp.o
% cd ../cfb
% make CDEBUGFLAGS="−g" cfbgetsp.o
```

This optimizer bug appears to be fixed in CodeCenter 4.0.6.

2.6.5. IBM AIX 4.1.4

On AIX 4.1.4, the file **lib/font/Type1/objects.c** must be compiled without optimization (**−O**) else the X server will exit when Type 1 fonts are used.

2.6.6. SunOS 4

SunOS 4.0 and earlier need BOOTSTRAPCFLAGS=-DNOSTDHDRS because they do not have unistd.h nor stdlib.h. Do *not* supply a BOOTSTRAPCFLAGS when building any SunOS 4.1 version.

2.6.7. Microsoft Windows NT

All of the base libraries are supported, including multi-threading in Xlib and Xt, but some of the more complicated applications, specifically *xterm* and *xdm*, are not supported.

There are also some other rough edges in the implementation, such as lack of support for non-socket file descriptors as Xt alternate inputs and not using the registry for configurable parameters like the system filenames and search paths.

The *Xnest* server has been made to run on NT. It requires a real X server for output still.

2.6.8. Omron Luna

The Omron Luna platform is no longer supported. The Luna version of the *make* program doesn't define the standard macro MAKE, so you must run it as ‘‘make MAKE=make’’ at top level, e.g., ‘‘make MAKE=make World’’.

2.7. The Build

On NT, type

```
nmake World.Win32 > world.log
```

On other systems, find the BootstrapCFlags line, if any, in the vendor-specific **.cf** file. If there isn't one, type

```
% make World >& world.log
```

otherwise type

```
% make World BOOTSTRAPCFLAGS="value" >& world.log
```

You can call the output file something other than "world.log", but do not call it "make.log" because files with this name are automatically deleted during the "cleaning" stage of the build.

Because the build can take several hours to complete, you will probably want to run it in the background and keep a watch on the output. For example:

```
% make World >& world.log &
% tail −f world.log
```

If something goes wrong, the easiest thing is to just start over (typing "make World" again) once you have corrected the problem.

2.8. Installing X

If everything is built successfully, you can install the software by typing the following as root:

```
% make install >& install.log
```

Again, you might want to run this in the background and use *tail* to watch the progress.

You can install the manual pages by typing the following as root:

```
% make install.man >& man.log
```

2.8.1. System Installation Notes

This section contains hints on installing and using X with specific compilers and operating systems.

2.8.1.1. The X Server on AIX 4

For IBM's AIX 4, you need to make sure the LFT device is associated with the correct graphics adapter. It's a one-time setup that does *not* happen automatically, even if there's only one graphics adapter in the system. To configure the LFT device properly, become root and start SMIT. Go to the "Devices" category, choose "LFT", then "Displays", then "Move the LFT to Another Display".

Select "Both" for when the change should take effect, then select the display adapter where you want to run the X server. Confirm the changes and exit SMIT; from now on, you should be able to run the server just fine.

To run *Xibm* from *xdm*, you must provide the “−force” flag on the server command line in the **Xservers** file.

2.9. Shared Libraries

The version number of some of the the shared libraries has been changed. On SunOS 4, which supports minor version numbers for shared libraries, programs linked with the R6 libraries will use the new libraries with no special action required. On other platforms you have the following choices:

1. Keep the old versions of the libraries around.
2. Relink all applications with the new libraries.
3. Create a link from the old name to the new name.

   For example, to have programs that were linked against libX11.so.6.0 use libX11.so.6.3, make this link:

   ```
   ln −s libX11.so.6.3 libX11.so.6.0
   ```

2.10. Setting Up xterm

If your **/etc/termcap** and **/usr/lib/terminfo** databases do not have correct entries for *xterm*, use the sample entries provided in the directory **xc/programs/xterm/**. System V users may need to compile and install the **terminfo** entry with the *tic* utility.

Since each *xterm* will need a separate pseudoterminal, you need a reasonable number of them for normal execution. You probably will want at least 32 on a small, multiuser system. On most systems, each pty has two devices, a master and a slave, which are usually named /dev/tty[pqrstu][0-f] and /dev/pty[pqrstu][0-f]. If you don’t have at least the “p” and “q” sets configured (try typing “ls /dev/?ty??”), you should have your system administrator add them. This is commonly done by running the *MAKEDEV* script in the **/dev** directory with appropriate arguments.

2.11. Starting Servers at System Boot

The *xfs* and *xdm* programs are designed to be run automatically at system startup. Please read the manual pages for details on setting up configuration files; reasonable sample files are in **xc/programs/xdm/config/** and **xc/programs/xfs/**.

2.11.1. On BSD-based systems using /etc/rc

If your system uses an **/etc/rc** file at boot time, you can usually enable these programs by placing the following at or near the end of the file:

```
if [ −f /usr/X11R6.3/bin/xfs ]; then
    /usr/X11R6.3/bin/xfs & echo −n ' xfs'
fi

if [ −f /usr/X11R6.3/bin/xdm ]; then
```

```
        /usr/X11R6.3/bin/xdm; echo −n ' xdm'
fi
```

Since *xfs* can serve fonts over the network, you do not need to run a font server on every machine with an X display. You should start *xfs* before *xdm*, since *xdm* may start an X server which is a client of the font server.

The examples here use **/usr/X11R6.3/bin**, but if you have installed into a different directory by setting (or unsetting) **ProjectRoot** then you need to substitute the correct directory.

If you are unsure about how system boot works, or if your system does not use **/etc/rc**, consult your system administrator for help.

2.11.2. On SystemV-based systems

There are two ways you can get On systems with a **/etc/inittab** file, you can edit this file to add the lines

```
xfs:3:once:/usr/X11R6.3/bin/xfs
xdm:3:once:/usr/X11R6.3/bin/xdm
```

On some systems, you can edit a file in /etc/init.d to run the X Consortium xdm instead of the vendor's product xdm. On Sony this file is /etc/init.d/consxdm. On IRIX edit /etc/init.d/xdm.

2.12. Using OPEN LOOK applications

You can use the X11R6 Xsun server with OPEN LOOK applications, but you must pass the **−swapLkeys** flag to the server on startup, or the OPEN LOOK Undo, Copy, Paste, Find, and Cut keys may not work correctly. For example, to run Sun's OpenWindows 3.3 desktop environment with an X11R6 server, use the command:

```
% openwin −server /usr/X11R6.3/bin/Xsun −swapLkeys
```

The keysyms reported by keys on the numeric keypad have also changed since X11R5; if you find that OpenWindows applications do not respond to keypad keys and cursor control keys when using the R6 server, you can remap the keypad to generate R5 style keysyms using the following *xmodmap* commands:

```
keysym Pause = F21
keysym Print = F22
keysym Break = F23
keysym KP_Equal = F24
keysym KP_Divide = F25
keysym KP_Multiply = F26
keysym KP_Home = F27
keysym KP_Up = Up
keysym KP_Prior = F29
keysym KP_Left = Left
keycode 100 = F31
keysym KP_Right = Right
keysym KP_End = F33
keysym KP_Down = Down
keysym KP_Next = F35
keysym KP_Insert = Insert
```

keysym KP_Delete = Delete

2.13. Rebuilding after Patches

You shouldn't need this right away, but eventually you are probably going to make changes to the sources, for example by applying any public patches that may be released.

Each patch comes with explicit instructions at the top of it saying what to do. Thus the procedure here is only an overview of the types of commands that might be necessary to rebuild X after changing it.

If you are building from CD-ROM, apply the patches to the symbolic link tree. The links to changed files will be replaced with local files containing the new contents.

If only source files are changed, you should be able to rebuild just by going to the **xc** directory in your build tree and typing:

```
% make >& make.log
```

If configuration files are changed, the safest thing to do is type:

```
% make Everything >& every.log
```

"Everything" is similar to "World" in that it rebuilds every **Makefile**, but unlike "World" it does not delete the existing objects, libraries, and executables, and only rebuilds what is out of date.

2.14. Formatting the Documentation

The PostScript files in **xc/doc/hardcopy** can be generated from the sources in **xc/doc/specs**. Most of the documentation is in troff using the −ms macros. The easiest way to format it is to use the Imakefiles provided.

Set the name of your local troff program by setting the variable **TroffCmd** in **xc/config/cf/site.def**. Then build the Makefiles:

```
cd xc/doc
make SUBDIRS=specs Makefiles
```

Finally, go to the directory you are interested in and type "make" there. This command will generate **.PS** files. You can also generate text files by specifying the document name with a **.txt** extension as a *make* target, e.g., "make icccm.txt".