X Window System, Version 11
Release 4
Release Notes

Jim Fulton

X Consortium
MIT Laboratory for Computer Science

ABSTRACT

The X Window System is a portable, network-transparent window system originally developed at MIT. It can be used on a wide variety of raster display devices, ranging from from simple monochrome frame buffers to deep, true color graphics processors. This document describes contents of the fourth public release of X, Version 11 from MIT and how it has changed from previous releases.

1. Overview

This is the fourth release of the X Window System, Version 11 from MIT. Substantial progress has been made in optimizing the sample server, window manager, and programming libraries. In addition, major improvements to the user interface of several of the key applications (in particular, *xmh*, *twm*, *xman*, and *xterm*) should make release noticably nicer to use. Sample implementations of the various new Consortium Standards are included as well as prototype implementations of several efforts currently under development. No incompatible changes have been made to either the core Protocol or to the *Xlib* programming library. The *Xt Intrinsics* should be source compatible with the previous release. Changes have been made to the *Xaw* widget set, but a configuration option for providing backwards compatibility interfaces is available.

Several new sets of fonts have been added: a new fixed width family of fonts, a Kanji and Kana font, the Lucida family from Bigelow & Holmes and Sun Microsystems, a terminal emulator font from Digital Equipment Corporation, and 100 dots-per-inch (dpi) versions of all 75dpi fonts.

This release contains two types of software: that which is supported by the staff of the X Consortium and which forms the core of the X Window System, and that which has been contributed by the user community and is provided as a public service without support from MIT. The core distribution also contains public implementations of certain software management utilities that may not be available on all platforms (such as *patch* and a *cpp* that can handle large numbers of symbols) as well as the tools that are used to build X software.

The servers, libraries, and applications provided in this release are *sample* rather than *reference* implementations. The X Consortium standardizes specifications, not particular instances of code. In particular, the document *X Window System Protocol, Version 11* is the final authority on what is and is not part of the core X Window System protocol. For additional information, see the *XStandards(1)* manual page.

Since the last release, the X Consortium has made significant additions to the Xlib standard (see *mit/doc/Xlib/R4Xlib.tbl.ms*) and to the X Toolkit Intrinsics standard, and has approved the following specifications as new standards:

Inter-Client Communications Conventions Manual
: The Inter-Client Communications Conventions Manual (ICCCM, whose specification may be found in *mit/doc/ICCCM/icccm.tbl.ms*) establishes a set of conventions that allow clients to cooperate in the areas of selections, cut buffers, window management, session management, and resources. Programming interfaces have been added to both *Xlib* and the *Xt Intrinsics* to simplify the task of writing compliant applications. The core *twm* window manager, as well as the user-contributed *gwm*, *olwm*, and *tekwm* window managers, is intended to be be compliant.

X11 Non-rectangular Window Shape Extension
: The SHAPE extension (whose specification may be found in *mit/doc/extensions/shape.ms*) provides non-rectangular, disjoint windows. Samples of its use may be found in the *Xaw* **Command** and **Mailbox** widgets, in the *twm* window manager, and in the *oclock* and *xeyes* clients.

X Display Manager Control Protocol
: The X Display Manager Control Protocol (XDMCP) (whose specification may be found in *mit/doc/XDMCP/xdmcp.ms*) is a datagram-based protocol for managing remote displays (particularly X terminals) in a network. Implementations of the various elements of the protocol are provided in the sample server and the *xdm* display manager.

Compound Text Encoding
: Compound Text (whose specification may be found in *mit/doc/CTEXT/ctext.tbl.ms*) is an interchange format for multiple character set data such as multi-lingual text. It is based on ISO standards for encoding and combining characters and is intended to be used in the following contexts: inter-client communication using selections, window properties, and resources. Routines for parsing Compound Text may be found in *mit/lib/Xmu/Xct.c*.

X Logical Font Description Conventions
: The X Logical Font Description Conventions (XLFD, whose specification may be found in *mit/doc/XLFD/xlfd.tbl.ms*) are a set of guidelines for naming fonts and font properties such that fonts can be uniquely named and queried in a consistent manner by applications. All of the text fonts in the core distribution follow the XLFD conventions. In addition, the new *xfontsel* program can be used to view and select fonts that have XLFD names.

This release been built on the following operating systems: Ultrix 3.1 (both VAX and RISC), SunOS 4.0.3, HP-UX 6.5, Domain/OS 10.1, A/UX 1.1, AIX RT-2.2 and PS/2-1.1, AOS-4.3, UTEK 4.0, NEWS-OS 3.2, UNICOS 5.0.1, and UNIX System V, Release 3.2 (AT&T 6386 WGS). It should work correctly, or with a minor amount of work, on a variety of other systems as well. Before building the release, see the *README* files in *mit/config/* and *mit/server/ddx/*/*, for any special instructions.

2. Building the Release

The software in this release is divided into two distributions: one for the *core* software that is supported by the staff of the X Consortium (located in the directory tree *./mit/*), and one for *user-contributed* software containing everything else (located in the directory tree *./contrib/*). Great pains have been taken to make the core distribution easy to reconfigure, build and install on a wide range of platforms. The user-contributed distribution, on the other hand, has not been compiled or tested by the staff of the X Consortium and will require building by hand. With the addition of function prototypes in the Xlib include files, it is virtually certain that some user-contributed will fail to compile under picky compilers (such as *hc*).

Almost all *Makefiles* in the core software are generated automatically by a utility called *imake*. The program combines machine-independent descriptions (called *Imakefiles*) of targets to be built with machine-dependent sets of parameters. Initial versions of all of the *Makefiles* are included for those sites that cannot use *imake* (they will undoubtedly require patching for specific machines). However, on many systems, X should build correctly right off the tape. For the user-contributed distribution, and for your own applications, the *xmkmf* script in *mit/util/scripts/* can be used to build a *Makefile* from an *Imakefile* once the core has been built and installed.

2.1. Installation Summary

To load and install this release of the X Window System, you will need to:

1. Finish reading these Release Notes.

2. Create a directory into which you will read the distribution tapes (usually named something like */usr/local/src/X* or */src/R4/*). You will need roughly 50 megabytes to hold the core software and up to 90 megabytes for the user-contributed software. Note that compiling will require anywhere from 50 to 110% more disk space, depending on your machine (e.g. RISC vs. CISC and whether or not you have shared libraries).

3. Unload the core tape into the directory created in step #2. Since the user-contributed software must be built by hand, you may wait and load it in later. Each of the tapes contains one (very large) UNIX *tar* file stored at 1600 bits per inch.

4. Read the file *mit/config/README* for instructions on how to configure the build for your particular site. Also, make sure that you follow the directions in *README* files in *mit/server/ddx/* directories for which you plan to build servers. In particular, you must make sure that the *OSMajorVersion* and *OSMinorVersion* configuration parameters, as well as those indicated at the top of *mit/config/site.def*, are appropriate for your system.

‡ Ultrix and VAX are trademarks of Digital Equipment Corporation; SunOS is a trademark of Sun Microsystems, Inc.; HP-UX and Domain are trademarks of the Hewlett-Packard Company; PostScript is a trademark of Adobe Systems, Inc.; A/UX is a trademark of Apple Computer; UNICOS is a trademark of Cray Research; AIX and AOS are trademarks of the IBM Corporation; UNIX is a registered trademark of AT&T Bell Laboratories; OPEN LOOK is a trademark of AT&T; Times, Helvetica, and New Century Schoolbook are registered trademarks of Linotype; Lucida is a registered trademark of Bigelow & Holmes; and Charter is a registered trademark of Bitstream, Inc.

5. If you plan to compile the release on more than one machine and have a distributed file system, you may wish to use the script *mit/util/scripts/lndir.sh* to create symbolic link trees on each of the target machines. This allows all of the platforms on which you wish to run X to share a single set of sources. In either case, the phrase *build tree* will be used to refer to the directory tree in which you are compiling (to distinguish it from the *source tree* which contains the actual files).

6. If you are building on a Macintosh II, make sure you read the file *mit/server/ddx/macII/README* and follow the directions for running the *X11R4* script in that directory. If you are using GNUC on the Mac (highly recommended; sources are available for anonymous ftp from the machine *apple.com*), you will need to remove the "*-s*" flag on the *egrep* command in the GNUC *fixincludes* script. Otherwise, you will have to build and install the C preprocessor in *mit/util/cpp/*.

7. If you are running on a VAX or 680x0 processor, you should consider using the GNU C compiler (available via anonymous ftp from the machine *prep.ai.mit.edu*) to compile the server. It can result in up to a factor of 2 improvement in performance. See the *HasGcc* parameter in the files *sun.cf*, *ultrix.cf*, *macII.cf*, and *site.def* in *mit/config/*

8. Check the *imake* configuration parameters in *mit/config/imakemdep.h* and *mit/config/Imake.tmpl*.

9. Once you are satisfied with the configuration, you are ready to build the core distribution. Look at the *.cf* file for your system. There may be a line in it that sets an *make* variable named *BootstrapCFlags*. If you don't find such a variable, you can use the following command to start the build:

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

   If you do find the variable, you should append that definition to the command line, using the *make* variable BOOTSTRAPCFLAGS. This is used by *imake* to set particular *cpp* symbols for all compiles (if you are porting to a different platform, see *mit/util/imake/imakemdep.h*). Special BOOTSTRAPCFLAGS are required on the following systems for which *.cf* files are supplied: MacII, AT&T 6386, IBM workstations, and the Tektronix 4310 series. For example:

   ```
   % make BOOTSTRAPCFLAGS=-DmacII World >& make.world &
   ```

   Do not call the output file *make.log* as the *make clean* done by *make World* removes all files of this name. This will rebuild all of the *Makefiles* and execute a *make -k all* to compile everything in the core distribution. This will take anywhere from 15 minutes (on a Cray Y-MP) to 12 hours, depending on your machine.

10. When the *make* is done, check the log file for any problems. There should be no serious errors. A/UX *pcc* users can ignore compiler warnings about enumeration type clashes, and Apollo and IBM users can ignore optimizer warnings. Most optimizers will also give warnings about the C code that is generated by *lex* and *yacc* in *mit/clients/twm*; these may be safely ignored.

11. If you are satisfied that everything has built correctly, test the various critical programs (servers, *xterm*, *xinit*, etc.) by hand. You may need to be root to run the server or *xterm*. A second workstation or terminal will be particularly useful if you run into problems.

12. Make backup copies of your old X header files, binaries, fonts, libraries, etc.

13. Go to the top of the build tree and type

    ```
    # make install >& make.install
    ```

    You will either have to do this as root, or have write access to the appropriate directories (see

DIRS_TO_BUILD in the top level *Imakefile* and *mit/config/Imake.tmpl*). The *xterm* program should be installed setuid to root on most systems and the *xload* program should be installed setgid to whichever group the file */dev/kmem* belongs to (it is installed setuid to root by default).

14. If you would like to install the manual pages, type the following at the top of the build tree:

```
# make install.man
```

15. If you would like to create and install lint libraries, type the following at the top of the build tree:

```
% make install.ln
```

If you are installing X for the first time, you may also need to do some of the steps listed below. Check the various README files in the *mit/server/ddx* directories for additional instructions.

16. Add device drivers or reconfigure your kernel.

17. Create additional pseudoterminals. See your operating system script */dev/MAKEDEV* and site administrator for details.

18. Read the manual page for the Display Manager *xdm* and configure it for your site. This program provides a portable way of running X automatically and has many hooks for creating a nice interface for novice users. **Warning: the –L flag is no longer supported by xterm. If you are running xterm from /etc/init you will have to convert to xdm or else save your old binary.**

19. Make sure that all X11 users have the directory */usr/bin/X11* in their search paths.

20. Give it a try!

Release 4 of Version 11 of the X Window System should now be ready to use.

2.2. Operating System Requirements

One of the reasons why X is so popular is that it does not require very much operating system support. Although this distribution only contains sample implementations for BSD and UNIX derivative platforms, support for other operating systems is available from a wide variety of vendors. The servers in this release have been built on the following systems:

Ultrix 3.1 (both VAX and RISC)
SunOS 4.0.3
HP-UX 6.5
Domain/OS 10.1
A/UX 1.1
AIX RT-2.2 and PS/2-1.1
IBM AOS-4.3
UTEK 4.0

The client libraries and applications have been built on all of the above systems, plus:

NEWS-OS 3.2
UNICOS 5.0.1

UNIX System V, Release 3.2 (AT&T 6386 WGS)

If you are using versions prior to these, you may well run into trouble. In particular, the server will not run on IBM 4.2A release 2 and there is no longer support for Apollo SR9.7. The *README* files in the various *mit/server/ddx/* describe particular requirements such as compilers, libraries, preprocessors, etc. As was noted above, A/UX 1.0 users will need to build a new version of the C preprocessor.

You should verify that your networking and interprocess communication facilities are working properly before trying to use X. If programs such as *talk* and *rlogin* don't work, X probably won't either.

2.3. Reading in the Release Tapes

This release may be obtained electronically from the Internet, the UUNET Project, several consulting firms, and various UUCP archive sites. In addition, a set of 2400 foot, 1600 BPI magnetic tapes (MIT does not distribute on cartridge tapes or floppy disks) and printed documentation is available from the MIT Software Center; call (617) 258-8330 for details.

Each tape from MIT contains one large *tar* archive with software and documentation for part of the release. If you have a limited amount of disk space, you should load tape #1, prune out any servers that you don't need, and generate listings of the user-contributed tapes for later retrieval. In particular, you will probably want to extract the *mit/doc* directory from tape #2. All filenames are given as relative paths (i.e. beginning with a period instead of a slash) so that the release may be placed anywhere in your file system.

Before reading in the tapes, make sure that you have enough disk space. Each of the tapes contains roughly 35 megabytes, split as follows:

Tape #	*Contents*	*Status*
1	core software for *make World*	required
2	core docs and contrib clients	recommended
3	contrib libs and toolkits	optional
4	contrib Andrew and games	optional

The compiled versions of the programs will occupy yet again as much disk space (particularly on RISC machines without shared libraries), so plan accordingly.

Create a directory into which you will put all of the sources. In this directory, execute the appropriate operating system commands to read in the core tapes. If your site is set up so that *tar* uses a 1600bpi tape drive by default, you will probably type something like:

```
% mkdir /usr/local/src/X
% cd /usr/local/src/X
% tar xv
```

See your system administrator for help.

2.4. Using Symbolic Links

This release uses links (symbolic, on machines that support them) in several places to avoid duplication of certain files (mostly header files). If you are building this release on a system for which configuration files have not been supplied, you should check the LN configuration parameter in the appropriate *mit/config/*.cf* file. If your operating system does not support soft links, LN should be set either to create hard links or to copy the source file.

If you need to move the release to another machine after it has been built, use *tar* instead of *cp* or *rcp* so that you preserve dates and links. This is usually done with a command of the form:

```
% (chdir /usr/local/src/X; tar cf - .) | rsh othermachine "(chdir /moredisk/X; tar xpBf -)"
```

See your system administrator for help.

2.5. Configuring the Release

This release makes extensive use of a utility called *imake* to generate machine-specific *Makefiles* from machine-independent *Imakefiles*. Another utility, called *makedepend*, is used to generate *Makefile* dependencies for C language files. Sample *Makefiles* are provided, although you are strongly urged to use *imake* and *makedepend* so that your software works across releases.

The configuration files for *imake* are located in the directory *mit/config/*. *Makefiles* are created from a template file named *Imake.tmpl*, a machine-specific *.cf* file, and a site-specific *site.def* file. With only a few exceptions, configuration parameters are *cpp* symbols that may be defined on a per-server basis or for all servers in a given site. The template file should *not* be modified.

The file *mit/config/README* describes each of the build parameters. The defaults have been chosen to work properly on a wide range of machines and to be easy to maintain. An overview of the configuration system may be found in the file *mit/doc/config/usenixws/paper.ms*. Site-specific configurations should be described in the file *site.def* using the following syntax:

```
#ifndef BuildParameter
#define BuildParameter site-specific-value
#endif
```

2.6. Compiling the Release

Once the configuration parameters are set, you should be able to type the following command at the top of the build tree to compile the core software:

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

Remember to set BOOTSTRAPCFLAGS if your system needs it. Don't redirect the output to *make.log* as this particular file is deleted as part of the build process. This will take anywhere up to 24 hours, depending on the machine used, and should complete without any significant errors on most machines.

If you need to restart the build after all of the *Makefiles* and dependencies have been created, type the following command at the top of the build tree:

```
% make -k >& make.out &
```

If you later decide to change any of the configuration parameters, you'll need to do another full *make World*.

2.7. Installing the Release

If everything compiles successfully, you may install the software by typing the following as root from the top of the build tree:

```
# make install
```

If you would rather not do the installation as root, make the necessary directories writable by you and do the install from your account. Then, check the ownership and protections on *xterm* and *xload* in the BINDIR directory (usually */usr/bin/X11/*): on most systems *xterm* must be installed setuid to root so that it can set the ownership of its pseudoterminal and update */etc/utmp*, and *xload* needs to be setuid to root or setgid to the group owning the file */dev/kmem* so that it can get the system load average.

If your */etc/termcap* and */usr/lib/terminfo* databases don't have entries for *xterm*, sample entries are provided in the directory *mit/clients/xterm/*. System V users may need to compile and install the *terminfo* entry with the *tic* utility.

If you plan to use the *xinit* program to run X, you should create a link named *X* pointing to the appropriate server program (usually named something like *Xmachine* in the directory */usr/bin/X11/*). However, *xinit* is not intended for most users; instead, site administrators are expected to either configure *xdm* or provide user-friendly interfaces.

If you would like to have manual pages installed, check the *ManDirectoryRoot*, *ManDir* and *LibManDir* configuration parameters in *mit/config/* and type the following at the top of the build tree:

```
# make install.man
```

If you would like to have lint libraries created and installed, type the following at the top of the build tree:

```
# make install.ln
```

Finally, make sure that all users have the BINDIR (usually */usr/bin/X11/*) in their PATH environment variable.

2.8. Notes on Kernels and Special Files

On some machines, it may be necessary to rebuild the kernel with a new device driver, or to at least reconfigure it. If you have never run X before and are using a system not listed in these notes, you might need verify that the CSR addresses in your kernel configuration file match your hardware. In addition, you should make sure that the kernel autoconfigures the display when booting.

You may need to create special devices for your display, mouse, or keyboard. For example,

```
# MAKEDEV cgfour          # for Sun 3/60
```

```
# MAKEDEV displays          # for displays on the RT/PC
```

The protection modes on the display device files should be set so that only the server can open them. If the server is started by */etc/init*, the protections can be root read/write, everyone else no access; otherwise, they will have to be read/write for everyone or else your server will have to be setuid to root.

On a Digital QVSS (VAXStation II) under older versions of Ultrix, you may need to use *adb* to make sure that the kernel variable *qv_def_scrn* is set to 2 so that the full width of the VR-260 monitor is used (otherwise there will be an unused black strip down the right edge of the screen). This can be done by changing the value either in */vmunix* directly or in */sys/vaxuba/qv.o* and relinking and reinstalling the kernel. You will need to reboot for the new value to take effect.

For more information, see the appropriate *README* files and manual pages in the *mit/server/ddx/* directories.

2.9. Testing the Release

Even if you plan on using *xdm* to run X all the time, you should first run it by hand from another terminal to check that everything is installed and working properly. Error messages from the X server will then appear on your terminal, rather than being written to the *xdm-errors* or to */usr/adm/X?msgs* (where *?* is the number of the display).

The easiest way to test the server is to go to */usr/bin/X11* (or wherever you have installed the various X programs), and run *xinit* as follows:

```
% cd /usr/bin/X11
% xinit
```

If all is well, you should see a gray stipple pattern covering the screen, a cursor shaped like an ‘‘X’’ that tracks the pointer, and a terminal emulator window. Otherwise, check the following:

1. If the gray background doesn't appear at all, check the permissions on any special device files (usually stored in */dev/*) described in the *README* in the appropriate *mit/server/ddx/* subdirectories.

2. If the background appears, but the cursor is a white square that doesn't change, make sure that the fonts have been installed (in particular, the font named *cursor.snf* in the directory */usr/lib/X11/fonts/misc/*; see the configuration parameter *DefaultFontPath*). Also make sure that there is a file named *fonts.dir* in each font directory. This file is created by the *mkfontdir* program and is used by the server to find fonts in a directory.

3. If the cursor appears but doesn't track the pointer, make sure that any special device files (often named something like */dev/mouse*) are installed (see the server's *README* file).

4. If the server starts up and then goes black a few seconds later, the initial client (usually *xterm* or *xdm*) is dying. Make sure that *xterm* is installed setuid to root and that you have created enough pseudoterminals. If you are running *xinit*, and have a file named *.xinitrc* in your home directory, make sure that it is executable and that the last program that it starts is run in the foreground (i.e. that there is no ampersand at the end of the line). Otherwise, the *.xinitrc* will finish immediately, which *xinit* assumes means that you are through.

Once you have the initial window working properly, try running some other programs from the *xterm*. To position a new window with the *twm* window manager, press Button1 (usually the left-most button on the pointer) when the flashing rectangle appears:

```
% xclock -g 200x200-0+0 &
% twm &
% xlogo &
% xeyes &
...
```

X should now be ready to use. Read the manual pages for the new programs, look at the new fonts, and have fun.

2.10. Creating Extra Pseudoterminals

Since each *xterm* will need a separate pseudoterminal, you should create a large number of them (you probably will want at least 32 on a small, multiuser system). 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'' lines configured (do an ''ls /dev''), you should have your system administrator add them. This is often done by running the MAKEDEV script in */dev*:

```
# cd /dev
# ./MAKEDEV pty0
# ./MAKEDEV pty1
```

2.11. Starting X from /etc/rc

The X Display Manager is used to run the X server and initial login window. It is normally started from the system startup file */etc/rc*, and is designed to be easily tailored to the needs of each specific site. *Xdm* takes care of keeping the server running, prompting for username and password and managing the user's session. The sample configuration currently uses shell scripts to provide a fairly simple environment. This will be an area of continuing work in future releases.

The key to *xdm*'s flexibility is its extensive use of resources, allowing site administrators to quickly and easily test alternative setups. When *xdm* starts up, it reads a configuration file (the default is */usr/lib/X11/xdm/xdm-config* but can be specified with the *-config* command line flag) listing the names of the various datafiles, default parameters, and startup and shutdown programs to be run. Because it uses the standard X Toolkit resource file format, any parameters that may be set in the *xdm-config* file may also be specified on the command line using the standard *-xrm* option.

The default configuration contains the following lines:

DisplayManager.servers:	/usr/lib/X11/xdm/Xservers
DisplayManager.errorLogFile:	/usr/lib/X11/xdm/xdm-errors
DisplayManager*resources:	/usr/lib/X11/xdm/Xresources
DisplayManager*startup:	/usr/lib/X11/xdm/Xstartup
DisplayManager*session:	/usr/lib/X11/xdm/Xsession
DisplayManager*reset:	/usr/lib/X11/xdm/Xreset

The *mit/servers* file contains the list of servers to start. The *errorLogFile* is where output from *xdm* is redirected. The *resources* file contains default resources for the *xdm* login window. In particular, this is where special key sequences can be specified (in the *xlogin*login.translations* resource). The *startup* file should be a program or executable script that is run after the user has provided a valid password. It is a hook for doing site-specific initialization, logging, etc. The *session* entry is the name of a session manager program or executable script that is run to start up the user's environment. A simple version has been supplied that provides an *xterm* window and *twm* window manager if the user does not have an executable *.xsession* file in his or her home directory. Finally, the *reset* program or executable script is run after the user logs out. It is a hook for cleaning up after the *startup* program.

To run *xdm* using the default configuration, add the following line to your system boot file (usually named */etc/rc* or */etc/rc.local*):

```
/usr/bin/X11/xdm
```

Most sites will undoubtedly want to build their own configurations. We recommend that you place any site-specific *xdm-config* and other *xdm* files in a different directory so that they are not overwritten if somebody ever does a *make install*. If you were to store the files in */usr/local/lib/xdm*, the following command could be used to start *xdm*:

```
/usr/bin/X11/xdm -config /usr/local/lib/xdm/xdm-config
```

Many servers set the keyboard to do non-blocking I/O under the assumption that they are the only programs attempting to read from the keyboard. Unfortunately, some versions of */etc/getty* (A/UX's in particular) will immediately see a continuous stream of zero-length reads which they interpret as end-of-file indicators. Eventually, */etc/init* will disable logins on that line until somebody types the following as root:

```
# kill -HUP 1
```

Under A/UX, one alternative is to disable logins on the console and always run *xdm* from */etc/inittab*. However, make sure that you save a copy of the old */etc/inittab* in case something goes wrong and you have to restore logins from over the network or from single-user mode.

Another less drastic approach is to set up an account whose shell is the *xdmshell* program found in *mit/clients/xdm/*. This program is not installed by default so that site administrators will examine it to see if it meets their needs. The *xdmshell* utility makes sure that it is being run from the appropriate type of terminal, starts *xdm*, waits for it to finish, and then resets the console if necessary. If the *xdm* resources file (specified by the *DisplayManager*resources* entry in the *xdm-config* file) contains a binding to the *abort-display* action similar to the following

```
xlogin*login.translations: #override  Ctrl<Key>R: abort-display()
```

the console can then by restored by pressing the indicated key (Control-R in the above example) in the *xdm* login window.

The *xdmshell* program is usually installed setuid to root but executable only by members of a special group, of which the account which has *xdmshell* as its shell is the only member:

```
% grep xdm /etc/passwd
x:aB9i7vhDVa82z:101:51:Account for starting up X:/tmp:/etc/xdmshell
% grep 51 /etc/group
xdmgrp:*:51:
% ls -lg /etc/xdmshell
-rws--x---   1 root     xdmgrp     20338 Nov  1 01:32 /etc/xdmshell
```

If the *xdm* resources have not been configured to have a key bound to the *abort-display()* action, there will be no way for general users to login to the console directly. Whether or not this is desirable depends on the particular site.

3. How to Report Bugs or Request Enhancements

If you find a *reproducible* bug in the core distribution supported by MIT (i.e. not in *contrib/*) or would like to suggest an enhancement (preferably with sample code), please fill in a copy of the form located in *mit/doc/bugs/bug-report* and mail it to:

xbugs@expo.lcs.mit.edu

Please fill in all sections (even if the bug appears on all systems) and please include any test cases; a small sample program is almost always the best information we can receive. Sites that do not have access to the various networks may send printed copies of bug reports and tests cases to:

X Bugs
X Consortium
MIT Laboratory for Computer Science
room NE43-218
545 Technology Square
Cambridge, MA 02139

Bug reports that are not sent in electronic form cannot be guaranteed a response. Also, any media containing bug reports, contributions, etc. will not be returned.

Bugs in user-contributed software should be sent to the author of the particular program, **not** to the address listed above. The X Consortium will not track or forward bugs in code located in *contrib/*.

4. What's New in this Release

The primary focus of this release has been optimization of the server and improvements in the key applications.

4.1. Changes to the core distribution

The following additions, deletions, and modifications have been made to the software in the core distribution. Widget writers should read the new X Toolkit Intrinsics specification. Application developers who use the Athena Widget Set should read the list of changes in the file *mit/lib/Xaw/CHANGES* and the conversion document *mit/doc/Xaw/ConvertToR4*, and read the new *Athena Widget Set* documentation.

many, many bugs fixed

A large number of bugs have been fixed in the server, the libraries, and the clients. Servers are now robust enough that they have been known to run for more than 3 months without experiencing any problems. The server is now much stricter about disallowing extraneous bits in masks (particularly the *do_not_propagate_mask* window attribute), causing some improperly coded applications to generate protocol errors. A new, non-standard extension is provided (see *xset bc*) to enable backwards-compatibility for broken clients.

server optimized, data space reduced

A substantial number of optimizations to both the device-independent (dix) and device-dependent (ddx) code. The monochrome (mfb) and color (cfb) frame buffer code is now capable of driving many displays at memory speeds. In addition, the amount of heap memory that is used by the server has been reduced by roughly two-thirds since the last release.

SHAPE extension

Non-rectangular windows are now supported by the new SHAPE extension. Round windows such as round clocks (see *oclock*), oval buttons (see *xmh* and *xcalc*), and shaped desktop icons (see *xbiff*) are now possible. This extension is a Consortium standard.

prototype extensions

Prototypes of two extensions that are currently under development are provided in this release. The *Multi-Buffering* extension provides the ability to do simple animation (see *ico -dbl*), and the *XInputExtension* provides access to alternate input devices. These extensions are **draft** Consortium standards and are subject to change.

build configuration moved and simplified

The configuration files have been moved to *mit*/*config*/ and have been rewritten to make better use of preprocessor symbols and macros. Support for System V with and without the STREAMS transport layers has been added.

new servers

New support has been added to the sample server for the following platforms: Sun *cgthree* and *cgsix* frame buffers, Digital DECstation frame buffers, Tektronix 4319 frame buffer, and all HP framebuffers. Reorganizations within the machine-independent (mi) graphics code make porting to new platforms even easier than it was before.

security hooks

Programming hooks in *Xlib* and the server are provided for passing authorization information at connection setup time. A sample implementation (called MIT-MAGIC-COOKIE-1) based on secret tokens is used by *xdm* and the server to provide greater security than the host-based mechanism.

new fonts

Adobe Systems and Digital Equipment Corporation have jointly donated 100dpi versions of the 75dpi fonts that they provided in the last release. In addition, Digital has donated a set of terminal emulator fonts. Bigelow & Holmes and Sun Microsystems have jointly donated a collection of fonts from the *Lucida* family. Sun has also donated a set of OPEN LOOK glyph fonts. Sony has donated a set of Kanji and Kana fonts, and several individuals have donated additional fixed-width fonts.

ICCCM support

Xlib, the *X Toolkit Intrinsics*, *twm*, and various clients are now hoped to be ICCCM-compliant.

The following window managers in *contrib/windowmgrs/* also claim to be compliant: *gwm*, *olwm*, and *tekwm*.

new rgb color database

A new color database containing many new colors, gray scales, and color spectra tuned for some of the common monitors is included.

function prototypes

ANSI C function prototypes have been added to the *Xlib* and *Xt* header files; the include files should now also be usable from C++ without modification. The *Xlib* prototypes are enabled by default (on systems that support them), while the *Xt* prototypes are disabled (they were added too late in our release cycle). Picky compilers (such as *hc*) will now catch many type incompatibilities.

shared libraries

Support for SunOS-style shared libraries has been added to *Xlib*, *Xt*, *Xaw*, and *Xmu*. This substantially reduces the amount of disk spaced used for executable programs.

new Xt Intrinsics

The *X Toolkit Intrinsics* now provide windowless objects, varargs-style interfaces, better caching of resources, fallback resources, locale-driven finding of data files.

Athena widget enhancements

Most of the *Xaw* library has been rewritten to substantially improve functionality, robustness, and performance. New **SimpleMenu** and **MenuButton** widgets support pop-up and pull-down menus. The **Text** widget has been rewritten and is now quite usable for general editing. The **VPaned** widget has been generalized to include horizontal paning (and is now called **Paned**). The **Label** widget now supports multi-line labels. A new **Toggle** widget has been provided for implementing radio-buttons. Finally, the **Command** widget has been enhanced to use the SHAPE extension to provide true round buttons.

standard colormap routines

A new set of routines for manipulating standard colormaps (see the *XStandardColormap* structure in the *Xlib* documentation) has been added to the *Xmu* library. The *xstdcmap* client uses these routines to create standard colormaps.

additional converters

A variety of new converters have been added to *Xmu*. In addition, multi-display programs should now be able to use these converters (and might find the utilities for managing the multiple display data structures useful).

new window manager

The *uwm* window manager has been moved from the core distribution to the user-contributed distribution. A substantially rewritten version of the *twm* window manager is now supported.

improvements in xdm

The *xdm* display manager has been rewritten to reduce the number of processes it requires and to make it much more robust. This is now the only supported means for starting the server at boot time (the *-L* command line option has been removed from *xterm*).

new utilities

Several new utility programs have been provided: *appres* for determining which resources are loaded into particular applications, *listres* for printing the resource hierarchy for a widget, *oclock* for people who like truly round clocks, *xauth* for manipulating authorization files, *xditview* for previewing *ditroff* files, *xfontsel* for interactively selecting fonts, *xlsatoms* for determining the value of various atoms, *xlsclients* for listing the clients currenting being run, and *xstdcmap* for manipulating standard colormaps.

new demos

A new demo of how various GC attributes (*xgc*) affect what is displayed on the screen is provided.

new features in *xterm*

The following features have been added to *xterm*: 8-bit input and output, on-the-fly changing of the current font through escape sequences and a new menu, new resources for controlling whether or not synthetic key events are ignored, increased portability, and improved menus using the new Athena **SimpleMenu** widget.

new CLX and documentation

A substantially improved version of CLX, the Common Lisp interface to X, is provided. In addition, comprehensive documentation of the CLX interface is provided, courtesy of Texas Instruments.

sample copyright notice in *mit/COPYRIGHTS*

The file *mit/COPYRIGHTS* in the top level directory contains a sample copyright notice recommended for people who are interested in contributing software to the public releases.

X Standards in *mit/Standards.man*

The *XStandards(1)* manual page contains a description of what is and is not an MIT X Consortium standard for the X Window System. For further information about the X Consortium, see the manual page *XConsortium(1)*.

4.2. Highlights of the User-Contributed Distribution

The user-contributed distribution is set up in a tree very similar to that used by the core distribution. New versions of several packages are available, and a variety of new donations have been received. Since this is **not** a superset of the previous user-contributed tape, sites are encouraged to save any R3 user-contributed software that they use. Note that this distribution is of no relation to the */contrib* directory available for anonymous ftp on *expo.lcs.mit.edu*.

XView and olwm

The *XView* toolkit and an ICCCM-compliant OPEN LOOK window manager from Sun Microsystems have been added to this release. This toolkit implements the OPEN LOOK graphical user interface guidelines and the SunView application programming interface.

Gwm

The *Generic Window Manager* from Groupe Bull has been added. Unlike other window managers, *gwm* provides a programming language for tailoring its user interface. It is believed to be ICCCM-compliant.

Tektronix Window Manager

The *Tektronix Window Manager*, derived from the *awm* window manager in the previous release, is also new to this release. Like *gwm* and *olwm*, *tekwm* is believed to be ICCCM-compliant.

Sigma toolkit and window manager
: The Sigma Project has donated its Sigma User Interface Toolkit (SUIT) and window manager (*m_swm*).

toolkits updated
: New versions of *InterViews*, *Xw*, *andrew*, and *clue* are included. New toolkits include: *xgks* and *Xcu*.

Serpent UIMS
: The *Serpent* user interface management system is included in this release.

new libraries
: Several libraries for doing Japanese input (see *XJ* and *Wnn*), multi-language input (see *mlx* and *im*), and compose processing (see *XCompose*) are provided.

new programming examples
: The examples from the O'Reilly and Associates books on *Xlib* and *Xt* have been provided.

new demos
: A variety of eye-catching demos have been added.

new clients
: A number of useful packages have been added: image analysis (see *NCSA* and *img*), multi-language libraries and utilities (see *kinput*, *kterm*, and *mlxterm*), a user interface prototyping system (see *winterp*), plus improved versions of *xfed*, *xcolors*, *xpic*, *xplaces*, *xtek* (formerly *texx*), *xtroff*, and *xwebster*.

games
: A variety of new games have been contributed since the new release.

5. Writing Portable X Software

A favorite saying around here goes: There's no such thing as portable software, only software that has been ported. However, there are few rules of thumb that go a long way towards making programs written for X easy to getting running on a wide variety of machines:

1. Keep all source filenames to 12 characters or less. This is the maximum number of characters that older System V file systems allow when using a source code control system.

2. Use *Imakefiles*. They are the only way to generate correct *Makefiles*. The *xmkmf* shell script in *mit*/*util*/*scripts*/ makes it trivial create *Makefiles* outside of the core source tree. The easiest way to construct an *Imakefile* is to start with one that does something similar and modify it. The various macros that are used are defined in the file *mit*/*config*/*Imake.rules*.

3. If you absolutely must use *Makefiles* instead of *Imakefiles*, link against *-lX11* instead of *-lX*. If you are using *imake*, use the symbolic names $(XAWLIB), $(XMULIB), $(XTOOLLIB), $(EXTENSIONLIB), and $(XLIB). *Xaw* clients may use the symbol *XawClientLibs* to refer to the appropriate libraries.

4. Include header files using the syntax <*X11*/*file.h*> instead of "*X11*/*file.h*", <*X*/*file.h*>, or "*X*/*file.h*".

5. Include *<X11/Xos.h>* if you need *types.h*, *string.h* or *strings.h* (then use the routines *index* and *rindex* instead of *strchr* and *strrchr*), *file.h*, *time.h*, or *unistd.h*.

6. If you need to put in System V vs. BSD dependencies, use *#ifdef SYSV*. If you need SVR3 vs. SVR2, use *#ifdef USG*.

7. Do not assume that the root window's Visual (returned by the *DefaultVisual* macro) is the only one available. Some color screens may use a black and white window for the root or could provide StaticColor as well as PseudoColor visuals. Unfortunately, most libraries do not have adequate support for locating visuals to use. In the mean time, use *XGetVisualInfo()*.

8. Use *-display displayname* to specify the X server to contact. Do not simply assume that if a command line argument has a colon in it that it is a *displayname*. If you accept command line abbreviations, make sure that you also accept the full *-display*.

9. Use *-geometry geomspec* to specify window geometry. Do not simply assume that if a command line argument begins with an equal sign that it is a window geometry. If you accept command line abbreviations, make sure that you also accept the full *-geometry*.

10. Use the *.man* suffix for program manual page sources.

11. If you are interested in contributing software to the MIT public release, please use a copyright notice that is no more restrictive than the one shown in the files *./COPYRIGHTS* and *contrib/COPYRIGHTS*.

We hope you enjoy Release 4.