Information for Linux Users

Orest Zborowski 13 November 1994

Contents

1 Linux versions on which XFree86 has been tested

XFree86 has been tested with Linux version 1.1.49. It should work with any version since 1.0 without change. The binaries and libraries are based on the 4.5.26 DLL C libraries, and the 1.4.4 dynamic linker `ld.so` (libc-4.5.26 or newer is *REQUIRED* !). The DLL X libraries are produced with the 2.16 DLL-tools by Eric Youngdale.

2 Backwards Compatibility

X11R6 is considered a major update from X11R5, so the shared libraries in XFree86 3.1 and later are not compatible with XFree86 2.1.1 and older libraries. To continue to run X11R5 applications, you must keep the old libraries somewhere on your machine. They can be moved from `/usr/X386/lib` elsewhere, but `/etc/ld.so.conf` must be updated. All X11R5 applications should work with the X11R6 servers without problems.

Very old binaries (linked to XFree86-1.2, XFree86-1.3 or XFree86-2.0 libraries) will continue to work, but may need an explicit symlink from `/lib/libX{11,t,aw}.so.3` to `/usr/X386/lib/libX{11,t,aw}.so.3`. Linking with `ld.so` against the XView3L5 distribution will give problems with pre-1.3 versions of `ld.so`, since the XView libraries contain the absolute path to the shared images, which is discouraged with `ld.so`.

A temporary fix is including / in `/etc/ld.so.config`, better recompile XView with relative library names or get a newer `ld.so`.

3 Installing XFree86

Starting with version 3.0, XFree86 is installed in `/usr/X11R6`. If you don't have enough space for both X11R5 and X11R6, you'll have to make space by discarding unnecessary parts of X11R5. In particular, the fonts and old X11R5 binaries can be removed. The shared libraries must be retained in order to run dynamically-linked X11R5 binaries.

Each binary distribution of XFree86 3.X comes with a detailed installation document relevant to that release. The most up-to-date information is kept there, while this document describes basic features of XFree86 3.X on Linux and how to compile the sources.

4 Running XFree86

XFree86 requires about 4mb of virtual memory to run, although having 8mb of RAM is probably the minimum comfortable configuration. A 387 coprocessor is helpful for 386 machines, although greater gains in interactive performance are obtained with an increase in physical memory. Also, a faster graphics card, bus or RAM, will improve server performance.

After unpacking the tar files, you need to include `/usr/X11R6/lib` in `/etc/ld.so.conf` (where it should already be by default) or in your `LD_LIBRARY_PATH` environment variable. Also, the configuration file `/usr/X11R6/lib/X11/XF86Config` *must* be properly filled out based on the host setup using `XF86Config.eg` as a starting point and `README.Config` as guideline. You may damage your hardware if you use a wrong `XF86Config` file, so *read the docs*, especially the man pages and the other `README` files in `/usr/X11R6/lib/X11/doc`.

XFree86 has the ability to perform VT switching to and from the X server. When first started, XFree86 will automatically locate the first available VT (one that hasn't been opened by any process), and run on that VT. If there isn't one available, XFree86 will terminate with an error message. The server can be run on a specific VT by using the "vt<nn>" option, where <nn> is the number of an available VT (starting from 1). If you don't have a free VT XFree86 cannot run. You can increase the number of available VTs by increasing the value of `NR_CONSOLES` in `include/linux/tty.h` and recompiling the kernel.

Once running inside X, switching to another VT is accomplished by pressing Ctrl-Alt-<Fnn> where nn is the number of the VT to switch to. To return to the server, press the proper key-combination that moves you back to the VT that XFree86 is using: by default, this is Alt-<Fmm>, where mm is the number of the VT the server is running on (this number is printed when the server is started). Note that this is NOT the VT from which the server was started.

NOTE: you can redefine the text-mode keybindings with the '`loadkeys`' command found in the `kbd-0.81.tar.gz` archive (or a later version thereof). With this, you can (for example) make Ctrl-Alt-<Fmm> work from text mode the same way it works under the XFree86 server.

When the server is exited, it will return to the original VT it was started from, unless it dies unexpectedly, when the switch must be done manually. There still seem to be weird combinations of graphic cards and motherboards that have problems to restore the textfont when returning from XFree86 to the text mode. In these cases using the `runx` script from the **svgalib** distribution might help.

The XFree86 server now queries the kernel to obtain the key binding in effect at startup. These bindings are either the default map in place when the kernel was compiled, or reloaded using the '`loadkeys`' utility. Not all keys are bound: kernel-specific, multiple keysym, and dead keys are not handled by the server. All others are translated to their X equivalents. Note that the XFree86 server only allows for four modifier maps: unshifted, shifted, modeswitch unshifted and modeswitch shifted. Depending on what the modeswitch key is (it is configurable in your `XF86Config` and defaults to Alt), XFree86 will read those tables into its keymaps. This means if you use certain keys, like left-Control, for Linux modeswitch, that will not be mappable to X.

There are some quirks that may need some fixing depending on your configuration. In the past, Linux xload used the BSD method of obtaining the load average from the running kernel. In the XFree86-1.3 release, that has been replaced by reading the information from `/proc/loadavg` instead, which should be forward-compatible with future kernel releases. Xman has also been changed to support the GNU groff family instead of the BSD nroff family. A quick edit and recompile restores BSD conventions. The server has been modified to provide better console redirection support for clients, which includes changing the ownership of `/dev/console` and `/dev/tty0` as well as the VT used to run on. Unfortunately, the kernel requires that a process be setuid root to perform console redirection, so `xterm` and `xconsole` must be setuid root. As `xconsole` is not designed with running setuid root in mind, this opens a big security problem.

5 Compiling XFree86

There are no special instructions required for compiling XFree86. This version was compiled with gcc-2.5.8, the 4.5.26 DLL libraries and the 1.4.4 shared, dynamic linker ld.so. The tools-2.16 DLL package was used to generate the shared libraries, and the server sources have been updated to use the new (`/usr/bin`) location of the tools. The server has been compiled with `-m486`, which optimizes it for the 486 processor, but the binary will run on the 386 processor (there is a slight increase in binary size over using `-m386`, but no loss of performance).

All extensions and clients have been ported and tested. By default, the servers are compiled without any extra extensions (PEX, XIE, etc), but libraries for those extensions are always generated. By modifying `site.def`, extra extensions can be compiled into the server. Alternately, the link kit can be used to craft modified servers.

The distribution is very large, but it is possible to compile XFree86 on a single 64mb partition, if the source tree is carefully trimmed (no manpages, PEX or large clients). Simply run "`make Makefiles`" to create the Makefiles, then stop the make and run each piece individually. It is not necessary to run "`make depend`" as well, which saves some space. Having 100mb available makes compiling XFree86 a lot easier. You will need about 10mb of virtual memory to compile the entire server.

If patches are applied which significantly change the libraries, modified jump_xxx files will be needed. Those can be generated according to instructions given in the DLL tools package, and will be made available as XFree86 patches.

The JUMP_xxx defines used to compile the X libraries can also be used to compile external X shared libraries, like Xaw3d. Detailed instructions are provided in `/usr/X11R6/lib/X11/config/lnxLib.rules`, where the X library definitions are provided, as an example.

Thanks to Dirk Hohndel (*hohndel@aib.com*) for working out all the details and doing all the work in creating the first set of XFree86 DLL libs, on which the current method (developed by Orest Zborowski (*orestz@eskimo.com*)) is based.

6 Bug Notification

Bug reports should be sent to *XFree86@XFree86.org* or posted to the *comp.windows.x.i386unix* newsgroup. Questions or comments about Linux support or the Linux distribution should be made to *orestz@eskimo.com.*

```
$XConsortium: Linux.sgml,v 1.2 95/01/23 15:34:44 kaleb Exp $
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/Linux.sgml,v 3.5 1995/01/28 16:01:43 dawes Exp $
```