The Linux XFree86 HOWTO

by Matt Welsh, mdw@sunsite.unc.edu v3.0, 15 March 1995

This document describes how to obtain, install, and configure version 3.1.1 of the XFree86 version of the X Window System (X11R6) for Linux systems. It is a step-by-step guide to configuring XFree86 on your system.

Contents

1 Introduction

The X Window System is a large and powerful (and somewhat complex) graphics environment for UNIX systems. The original X Window System code was developed at MIT; commercial vendors have since made X the industry standard for UNIX platforms. Virtually every UNIX workstation in the world runs some variant of the X Window system.

A free port of the MIT X Window System version 11, release 6 (X11R6) for 80386/80486/Pentium UNIX systems has been developed by a team of programmers originally headed by David Wexelblat (dwex@XFree86.org). The release, known as XFree86, is available for System V/386, 386BSD, and other x86 UNIX implementations, including Linux. It includes all of the required binaries, support files, libraries, and tools.

In this document, we'll give a step-by-step description of how to install and configure XFree86 for Linux, but you will have to fill in some of the details yourself by reading the documentation released with XFree86 itself. (This documentation is discussed below.) However, using and customizing the X Window System is far beyond the scope of this document—for this purpose you should obtain one of the many good books on using the X Window System.

2 Hardware requirements

As of XFree86 version 3.1.1, released in February 1995, the following video chipsets are supported. The documentation included with your video adaptor should specify the chipset used. If you are in the market for a new video card, or are buying a new machine that comes with a video card, have the vendor find out exactly what the make, model, and chipset of the video card is. This may require the vendor to call technical support on your behalf; in general vendors will be happy to do this. Many PC hardware vendors will state that the video card is a "standard SVGA card" which "should work" on your system. Explain that your software (mention Linux and XFree86!) does not support all video chipsets and that you must have detailed information.

You can also determine your videocard chipset by running the `SuperProbe` program included with the XFree86 distribution. This is covered in more detail below.

The following standard SVGA chipsets are supported:

- Tseng ET3000, ET4000AX, ET4000/W32
- Western Digital/Paradise PVGA1
- Western Digital WD90C00, WD90C10, WD90C11, WD90C24, WD90C30, WD90C31, WD90C33
- Genoa GVGA
- Trident TVGA8800CS, TVGA8900B, TVGA8900C, TVGA8900CL, TVGA9000, TVGA9000i, TVGA9100B, TVGA9200CX, TVGA9320, TVGA9400CX, TVGA9420
- ATI 18800, 18800-1, 28800-2, 28800-4, 28800-5, 28800-6, 68800-3, 68800-6, 68800AX, 68800LX, 88800
- NCR 77C22, 77C22E, 77C22E+
- Cirrus Logic CLGD5420, CLGD5422, CLGD5424, CLGD5426, CLGD5428, CLGD5429, CLGD5430, CLGD5434, CLGD6205, CLGD6215, CLGD6225, CLGD6235, CLGD6420
- Compaq AVGA
- OAK OTI067, OTI077
- Avance Logic AL2101
- MX MX68000, MX680010
- Video 7/Headland Technologies HT216-32

The following SVGA chipsets with accelerated features are also supported:

- 8514/A (and true clones)

- ATI Mach8, Mach32
- Cirrus CLGD5420, CLGD5422, CLGD5424, CLGD5426, CLGD5428, CLGD5429, CLGD5430, CLGD5434, CLGD6205, CLGD6215, CLGD6225, CLGD6235
- S3 86C911, 86C924, 86C801, 86C805, 86C805i, 86C928, 86C864, 86C964
- Western Digital WD90C31, WD90C33
- Weitek P9000
- IIT AGX-014, AGX-015, AGX-016
- Tseng ET4000/W32, ET4000/W32i, ET4000/W32p

Video cards using these chipsets are supported on all bus types, including VLB and PCI.

All of the above are supported in both 256 color and monochrome modes, with the exception of the Avance Logic, MX and Video 7 chipsets, which are only supported in 256 color mode. If your video card has enough DRAM installed, many of the above chipsets are supported in 16 and 32 bits-per-pixel mode (specifically, some Mach32, P9000, S3 and Cirrus boards). The usual configuration is 8 bits per pixel (that is, 256 colors).

The monochrome server also supports generic VGA cards, the Hercules monochrome card, the Hyundai HGC1280, Sigma LaserView, and Apollo monochrome cards. On the Compaq AVGA, only 64k of video memory is supported for the monochrome server, and the GVGA has not been tested with more than 64k.

This list will undoubtedly expand as time passes. The release notes for the current version of XFree86 should contain the complete list of supported video chipsets.

One problem faced by the XFree86 developers is that some video card manufacturers use non-standard mechanisms for determining clock frequencies used to drive the card. Some of these manufacturers either don't release specifications describing how to program the card, or they require developers to sign a non-disclosure statement to obtain the information. This would obviously restrict the free distribution of the XFree86 software, something that the XFree86 development team is not willing to do. For a long time, this has been a problem with certain video cards manufactured by Diamond, but as of release 3.1 of XFree86, Diamond has started to work with the development team to release free drivers for these cards.

The suggested setup for XFree86 under Linux is a 486 machine with at least 8 megabytes of RAM, and a video card with a chipset listed above. For optimal performance, we suggest using an accelerated card, such as an S3-chipset card. You should check the documentation for XFree86 and verify that your particular card is supported before taking the plunge and purchasing expensive hardware. Benchmark ratings comparisons for various video cards under XFree86 are posted routinely to the USENET newsgroups `comp.windows.x.i386unix` and `comp.os.linux.x`.

As a side note, my personal Linux system is a 486DX2-66, 20 megabytes of RAM, and is equipped with a VLB S3-864 chipset card with 2 megabytes of DRAM. I have run X benchmarks on this machine as well as on Sun Sparc IPX workstations. The Linux system is roughly 7 times faster than the Sparc IPX (for the curious, XFree86-3.1 under Linux, with this video card, runs at around 171,000 xstones; the Sparc IPX at around 24,000). In general, XFree86 on a Linux system with an accelerated SVGA card will give you much greater performance than that found on commercial UNIX workstations (which usually employ simple framebuffers for graphics).

Your machine will need at least 4 megabytes of physical RAM, and 16 megabytes of virtual RAM (for example, 8 megs physical and 8 megs swap). Remember that the more physical RAM that you have, the less that the system will swap to and from disk when memory is low. Because swapping is inherently slow (disks are very slow compared to memory), having 8 megabytes of RAM or more is necessary to run XFree86 comfortably. A system with 4 megabytes of physical RAM could run *much* (up to 10 times) more slowly than one with 8 megs or more.

3 Installing XFree86

The Linux binary distribution of XFree86 can be found on a number of FTP sites. On `sunsite.unc.edu`, it is found in the directory `/pub/Linux/X11`. (As of the time of this writing, the current version is 3.1.1; newer versions are released periodically).

It's quite likely that you obtained XFree86 as part of a Linux distribution, in which case downloading the software separately is not necessary.

If you are downloading XFree86 directly, This table lists the files in the XFree86-3.1 distribution.

One of the following servers is required:

XF86-3.1.1-8514.tar.gz

Server for 8514-based boards.

XF86-3.1.1-AGX.tar.gz

Server for AGX-based boards.

XF86-3.1.1-Mach32.tar.gz

Server for Mach32-based boards.

XF86-3.1.1-Mach8.tar.gz

Server for Mach8-based boards.

XF86-3.1.1-Mono.tar.gz

Server for monochrome video modes.

XF86-3.1.1-P9000.tar.gz

Server for P9000-based boards.

XF86-3.1.1-S3.tar.gz

Server for S3-based boards.

XF86-3.1.1-SVGA.tar.gz

Server for Super VGA-based boards.

XF86-3.1.1-VGA16.tar.gz

Server for VGA/EGA-based boards.

XF86-3.1.1-W32.tar.gz

Server for ET4000/W32-based boards.

All of the following files are required:

XF86-3.1.1-bin.tar.gz

The rest of the X11R6 binaries.

XF86-3.1.1-cfg.tar.gz

Config files for `xdm`, `xinit` and `fs`.

XF86-3.1.1-doc.tar.gz

Documentation and manpages.

XF86-3.1.1-inc.tar.gz

Include files.

XF86-3.1.1-lib.tar.gz

Shared X libraries and support files.

XF86-3.1-fnt.tar.gz

Basic fonts.

The following files are optional:

XF86-3.1-ctrb.tar.gz

Selected contrib programs.

XF86-3.1-extra.tar.gz

Extra XFree86 servers and binaries.

XF86-3.1-lkit.tar.gz

Server linkkit for customization.

XF86-3.1-fnt75.tar.gz

75-dpi screen fonts.

XF86-3.1-fnt100.tar.gz

100-dpi screen fonts.

XF86-3.1-fntbig.tar.gz

Large Kanji and other fonts.

XF86-3.1-fntscl.tar.gz

Scaled fonts (Speedo, Type1).

XF86-3.1-man.tar.gz

Manual pages.

XF86-3.1-pex.tar.gz

PEX binaries, includes and libraries.

XF86-3.1-slib.tar.gz

Static X libraries and support files.

XF86-3.1-usrbin.tar.gz

Daemons which reside in /usr/bin.

XF86-3.1-xdmshdw.tar.gz

Shadow password version of xdm.

The XFree86 directory should contain `README` files and installation notes for the current version.

All that is required to install XFree86 is to obtain the above files, create the directory `/usr/X11R6` (as `root`), and unpack the files from `/usr/X11R6` with a command such as:

```
gzip -dc XF86-3.1.1-bin.tar.gz | tar xfB -
```

Remember that these tar files are packed relative to `/usr/X11R6`. so it's important to unpack the files there.

After unpacking the files, you first need to link the file `/usr/X11R6/bin/X` to the server that you're using. For example, if you wish to use the SVGA color server, `/usr/bin/X11/X` should be linked to `/usr/X11R6/bin/XF86_SVGA`. If you wish to use the monochrome server instead, relink this file to `XF86_MONO` with the command

```
ln -sf /usr/X11R6/bin/XF86_MONO /usr/X11R6/bin/X
```

The same holds true if you are using one of the other servers.

If you aren't sure which server to use, or don't know your video card chipset, you can run the `SuperProbe` program found in `/usr/X11R6/bin` (included in the `XF86-3.1-bin` listed above). This program will attempt to determine your video chipset type and other information; write down its output for later reference.

You need to make sure that `/usr/X11R6/bin` is on your path. This can be done by editing your system default `/etc/profile` or `/etc/csh.login` (based on the shell that you, or other users on your system, use). Or you can simply add the directory to your personal path by modifying `/etc/.bashrc` or `/etc/.cshrc`, based on your shell.

You also need to make sure that `/usr/X11R6/lib` can be located by `ld.so`, the runtime linker. To do this, add the line

```
/usr/X11R6/lib
```

to the file `/etc/ld.so.conf`, and run `/sbin/ldconfig`, as `root`.

4 Configuring XFree86

Setting up XFree86 is not difficult in most cases. However, if you happen to be using hardware for which drivers ar under development, or wish to obtain the best performance or resolution from an accelerated graphics card, configuring XFree86 can be somewhat time-consuming.

In this section we will describe how to create and edit the `XF86Config` file, which configures the XFree86 server. In many cases it is best to start out with a "basic" XFree86 configuration, one which uses a low resolution, such as 640x480, which should be supported on all video cards and monitor types. Once you have XFree86 working at a lower, standard resolution, you can tweak the configuration to exploit the capabilities of your video hardware. The idea is that you want to know that XFree86 works at all on your system, and that something isn't wrong with your installation, before attempting the sometimes difficult task of setting up XFree86 for real use.

In addition to the information listed here, you should read the following documentation:

- The XFree86 documentation in `/usr/X11R6/lib/X11/doc` (contained within the `XFree86-3.1-doc` package). You should especially see the file `README.Config`, which is an XFree86 configuration tutorial.
- Several video chipsets have separate `README` files in the above directory (such as `README.Cirrus` and `README.S3`). Read one of these if applicable.
- The man page for `XFree86`.
- The man page for `XF86Config`.
- The man page for the particular server that you are using (such as `XF86_SVGA` or `XF86_S3`).

The main XFree86 configuration file is `/usr/X11R6/lib/X11/XF86Config`. This file contains information on your mouse, video card parameters, and so on. The file `XF86Config.eg` is provided with the XFree86 distribution as an example. Copy this file to `XF86Config` and edit it as a starting point.

The `XF86Config` man page explains the format of this file in detail. Read this man page now, if you have not done so already.

We are going to present a sample `XF86Config` file, piece by piece. This file may not look exactly like the sample file included in the XFree86 distribution, but the structure is the same.

Note that the `XF86Config` file format may change with each version of XFree86; this information is only valid for XFree86 version 3.1.

Also, you should not simply copy the configuration file listed here to your own system and attempt to use it. Attempting to use a configuration file which doesn't correspond to your hardware could drive the monitor at a frequency which is too high for it; there have been reports of monitors (especially fixed-frequency monitors) being damaged or destroyed by using an incorrectly configured `XF86Config` file. The bottom line is this: Make absolutely sure that your `XF86Config` file corresponds to your hardware before you attempt to use it.

Each section of the `XF86Config` file is surrounded by the pair of lines

```
Section "section-name"
  ...
EndSection
```

The first part of the `XF86Config` file is `Files`, which looks like this:

```
Section "Files"
    RgbPath     "/usr/X11R6/lib/X11/rgb"
    FontPath    "/usr/X11R6/lib/X11/fonts/misc/"
    FontPath    "/usr/X11R6/lib/X11/fonts/75dpi/"
EndSection
```

The `RgbPath` line sets the path to the X11R6 RGB color database, and each `FontPath` line sets the path to a directory containing X11 fonts. In general you shouldn't have to modify these lines; just be sure that there is a `FontPath` entry for each font type that you have installed (that is, for each directory in `/usr/X11R6/lib/X11/fonts`).

The next section is `ServerFlags`, which specifies several global flags for the server. In general this section is empty.

```
Section "ServerFlags"
# Uncomment this to cause a core dump at the spot where a signal is
# received.  This may leave the console in an unusable state, but may
# provide a better stack trace in the core dump to aid in debugging
#    NoTrapSignals

# Uncomment this to disable the <Crtl><Alt><BS> server abort sequence
#    DontZap
EndSection
```

Here, we have all lines within the section commented out.

The next section is `Keyboard`. This should be fairly intuitive.

```
Section "Keyboard"
    Protocol    "Standard"
    AutoRepeat  500 5
    ServerNumLock
EndSection
```

Other options are available as well—see the `XF86Config` file if you wish to modify the keyboard configuration. The above should work for most systems.

The next section is `Pointer` which specifies parameters for the mouse device.

```
Section "Pointer"

    Protocol    "MouseSystems"
    Device      "/dev/mouse"

# Baudrate and SampleRate are only for some Logitech mice
#    BaudRate   9600
#    SampleRate 150
```

```
# Emulate3Buttons is an option for 2-button Microsoft mice
#    Emulate3Buttons

# ChordMiddle is an option for some 3-button Logitech mice
#    ChordMiddle

EndSection
```

The only options that you should concern yourself with now are `Protocol` and `Device`. `Protocol` specifies the mouse *protocol* that your mouse uses (not the make or brand of mouse). Valid types for `Protocol` (under Linux—there are other options available for other operating systems) are:

- `BusMouse`
- `Logitech`
- `Microsoft`
- `MMSeries`
- `Mouseman`
- `MouseSystems`
- `PS/2`
- `MMHitTab`

`BusMouse` should be used for the Logitech busmouse. Note that older Logitech mice should use `Logitech`, but newer Logitech mice use either `Microsoft` or `Mouseman` protocols. This is a case in which the protocol doesn't necessarily have anything to do with the make of the mouse.

`Device` specifies the device file where the mouse can be accessed. On most Linux systems, this is `/dev/mouse`. `/dev/mouse` is usually a link to the appropriate serial port (such as `/dev/cua0`) for serial mice, or to the appropriate busmouse device for busmice. At any rate, be sure that the device file listed in `Device` exists.

The next section is `Monitor`, which specifies the characteristics of your monitor. As with other sections in the `XF86Config` file, there may be more than one `Monitor` section. This is useful if you have multiple monitors connected to a system, or use the same `XF86Config` file under multiple hardware configurations. In general, though, you will need a single `Monitor` section.

```
Section "Monitor"

    Identifier  "CTX 5468 NI"

    # These values are for a CTX 5468NI only! Don't attempt to use
    # them with your monitor (unless you have this model)

    Bandwidth    60
```

```
    HorizSync    30-38,47-50
    VertRefresh  50-90

    # Modes: Name      dotclock  horiz               vert

    ModeLine "640x480"  25       640 664 760 800      480 491 493 525
    ModeLine "800x600"  36       800 824 896 1024     600 601 603 625
    ModeLine "1024x768" 65       1024 1088 1200 1328  768 783 789 818

EndSection
```

The `Identifier` line is used to give an arbitrary name to the `Monitor` entry. This can be any string; you will use it to refer to the `Monitor` entry later in the `XF86Config` file.

they are listed below.

`HorizSync` specifies the valid horizontal sync frequencies for your monitor, in kHz. If you have a multisync monitor, this can be a range of values (or several comma-separated ranges), as seen above. If you have a fixed-frequency monitor, this will be a list of discrete values, such as:

```
    HorizSync    31.5, 35.2, 37.9, 35.5, 48.95
```

Your monitor manual should list these values in the technical specifications section. If you do not have this information available, you should either contact the manufacturer or vendor of your monitor to obtain it. There are other sources of information, as well;

`VertRefresh` specifies the valid vertical refresh rates (or vertical synchronization frequencies) for your monitor, in Hz. Like `HorizSync` this can be a range or a list of discrete values; your monitor manual should list them.

`HorizSync` and `VertRefresh` are used only to double-check that the monitor resolutions that you specify are in valid ranges. This is to reduce the chance that you will damage your monitor by attempting to drive it at a frequency for which it was not designed.

The `ModeLine` directive is used to specify a single resolution mode for your monitor. The format of `ModeLine` is

```
ModeLine name clock horiz-values vert-values
```

`name` is an arbitrary string, which you will use to refer to the resolution mode later in the file. `dot-clock` is the driving clock frequency, or "dot clock" associated with the resolution mode. A dot clock is usually specified in MHz, and is the rate at which the video card must send pixels to the monitor at this resolution. `horiz-values` and `vert-values` are four numbers each which specify when the electron gun of the monitor should fire, and when the horizontal and vertical sync pulses fire during a sweep.

How can you determine the `ModeLine` values for your monitor? The file `VideoModes.doc`, included with the XFree86 distribution, describes in detail how to determine these values for each resolution mode that your monitor supports. First of all, `clock` must correspond to one of the dot clock values that your video card can produce. Later in the `XF86Config` file you will specify these clocks; you can only use video modes which have a `clock` value supported by your video card.

There are two files included in the XFree86 distribution which may include `ModeLine` data for your monitor. These files are `modeDB.txt` and `Monitors`, both of which are found in `/usr/X11R6/lib/X11/doc`.

You should start with `ModeLine` values for the VESA standard monitor timings, which most monitors support. `modeDB.txt` includes timing values for VESA standard resolutions. In that file, you will see entries such as

```
# 640x480@60Hz Non-Interlaced mode
# Horizontal Sync = 31.5kHz
# Timing: H=(0.95us, 3.81us, 1.59us), V=(0.35ms, 0.064ms, 1.02ms)
#
# name         clock   horizontal timing     vertical timing      flags
 "640x480"     25.175  640  664  760  800    480  491  493  525
```

This is a VESA standard timing for a 640x480 video mode. It uses a dot clock of 25.175, which your video card must support to use this mode (more on this later). To include this entry in the `XF86Config` file, you'd use the line

```
ModeLine "640x480" 25.175 640 664 760 800 480 491 493 525
```

Note that the `name` argument to `ModeLine` (in this case `"640x480"`) is an arbitrary string—the convention is to name the mode after the resolution, but `name` can technically be anything descriptive which describes the mode to you.

For each `ModeLine` used the server will check that the specifications for the mode fall within the range of values specified with `Bandwidth`, `HorizSync` and `VertRefresh`. If they do not, the server will complain when you attempt to start up X (more on this later). For one thing, the dot clock used by the mode should not be greater than the value used for `Bandwidth`. (However, in many cases it is safe to use modes with a slightly higher bandwidth than your monitor can support.)

If the VESA standard timings do not work for you (you'll know after trying to use them later) then the files `modeDB.txt` and `Monitors` include specific mode values for many monitor types. You can create `ModeLine` entries from the values found in those two files as well. Be sure to only use values for the specific model of monitor that you have. Note that many 14 and 15-inch monitors cannot support higher resolution modes, and often resolutions of 1024x768 at low dot clocks. This means that if you can't find high resolution modes for your monitor in these files, then your monitor probably does not support those resolution modes.

If you are completely at a loss, and can't find working `ModeLine` values for your monitor, you can follow the instructions in the `VideoModes.doc` file included in the XFree86 distribution to generate `ModeLine` values from the specifications listed in your monitor's manual. While your mileage will certainly vary when attempting to generate `ModeLine` values by hand, this is a good place to look if you can't find the values that you need. `VideoModes.doc` also describes the format of the `ModeLine` directive and other aspects of the XFree86 server in gory detail.

Lastly, if you do obtain `ModeLine` values which are almost, but not quite, right, then it may be possible to simply modify the values slightly to obtain the desired result. For example, if while running XFree86 the image on the monitor is shifted slightly, or seems to "roll", you can follow the instructions in the `VideoModes.doc` file to try to fix these values. Also, be sure to check the knobs and controls on the monitor itself! In many cases it is necessary to change the horizontal or vertical size of the display after starting up XFree86 in order for the image to be centered and be of the appropriate size. Having these controls on the front of the monitor can certainly make life easier.

You shouldn't use monitor timing values or `ModeLine` values for monitors other than the model that you own. If you attempt to drive the monitor at a frequency for which it was not designed, you can damage or even destroy it.

The next section of the `XF86Config` file is `Device`, which specifies parameters for your video card. Here is an example.

```
Section "Device"
        Identifier "#9 GXE 64"

        # Nothing yet; we fill in these values later.

EndSection
```

This section defines properties for a particular video card. `Identifier` is an arbitrary string describing the card; you will use this string to refer to the card later.

Initially, you don't need to include anything in the `Device` section, except for `Identifier`. This is because we will be using the X server itself to probe for the properties of the video card, and entering them into the `Device` section later. The XFree86 server is capable of probing for the video chipset, clocks, RAMDAC, and amount of video RAM on the board.

Before we do this, however, we need to finish writing the `XF86Config` file. The next section is `Screen`, which specifies the monitor/video card combination to use for a particular server.

```
Section "Screen"
    Driver     "Accel"
    Device     "#9 GXE 64"
    Monitor    "CTX 5468 NI"
    Subsection "Display"
        Depth      16
        Modes      "1024x768" "800x600" "640x480"
        ViewPort   0 0
        Virtual    1024 768
    EndSubsection
EndSection
```

The `Driver` line specifies the X server that you will be using. The value values for `Driver` are:

- `Accel`: For the `XF86_S3`, `XF86_Mach32`, `XF86_Mach8`, `XF86_8514`, `XF86_P9000`, `XF86_AGX`, and `XF86_W32` servers;
- `SVGA`: For the `XF86_SVGA` server;
- `VGA16`: For the `XF86_VGA16` server;

- `VGA2`: For the `XF86_Mono` server;
- `Mono`: For the non-VGA monochrome drivers in the `XF86_Mono` and `XF86_VGA16` servers.

You should be sure that `/usr/X11R6/bin/X` is a symbolic link to the server that you are using.

The `Device` line specifies the `Identifier` of the `Device` section corresponding to the video card to use for this server. Above, we created a `Device` section with the line

```
Identifier "#9 GXE 64"
```

Therefore, we use `"#9 GXE 64"` on the `Device` line here.

Similarly, the `Monitor` line specifies the name of the `Monitor` section to be used with this server. Here, `"CTX 5468 NI"` is the `Identifier` used in the `Monitor` section described above.

`Subsection "Display"` defines several properties of the XFree86 server corresponding to your monitor/video card combination. The `XF86Config` file describes all of these options in detail; most of them are icing on the cake and not necessary to get the system working.

The options that you should know about are:

- `Depth`. Defines the number of color planes—the number of bits per pixel. Usually, `Depth` is set to 8. For the `VGA16` server, you would use a depth of 4, and for the monochrome server a depth of 1. If you are using an accelerated video card with enough memory to support more bits per pixel, you can set `Depth` to 16, 24, or 32. If you have problems with depths higher than 8, set it back to 8 and attempt to debug the problem later.
- `Modes`. This is the list of video mode names which have been defined using the `ModeLine` directive in the `Monitor` section. In the above section, we used `ModeLines` named `"1024x768"`, `"800x600"`, and `"640x480"`. Therefore, we use a `Modes` line of

  ```
  Modes    "1024x768" "800x600" "640x480"
  ```

  The first mode listed on this line will be the default when XFree86 starts up. After XFree86 is running, you can switch between the modes listed here using the keys `ctrl-alt-numeric +` and `ctrl-alt-numeric -`.

  It might be best, when initially configuring XFree86, to use lower resolution video modes, such as 640x480, which tend to work on most systems. Once you have the basic configuration working you can modify `XF86Config` to support higher resolutions.
- `Virtual`. Sets the virtual desktop size. XFree86 has the ability to use any additional memory on your video card to extend the size of your desktop. When you move the mouse pointer to the edge of the display, the desktop will scroll, bringing the additional space into view. Therefore, even if you are running at a lower video resolution such as 800x600, you can set `Virtual` to the total resolution which your video card can support (a 1-megabyte video card can support 1024x768 at a depth of 8 bits per pixel; a 2-megabyte card 1280x1024 at depth 8, or 1024x768 at depth 16). Of course, the entire area will not be visible at once, but it can still be used.

The `Virtual` feature is a nice way to utilize the memory of your video card, but it is rather limited. If you want to use a true virtual desktop, we suggest using `fvwm`, or a similar window manager, instead. `fvwm` allows you to have rather large virtual desktops (implemented by hiding windows, and so forth, instead of actually storing the entire desktop in video memory at once). See the man pages for `fvwm` for more details about this; most Linux systems use `fvwm` by default.

- `ViewPort`. If you are using the `Virtual` option described above, `ViewPort` sets the coordinates of the upper-left-hand corner of the virtual desktop when XFree86 starts up. `Virtual 0 0` is often used; if this is unspecified then the desktop is centered on the virtual desktop display (which may be undesirable to you).

Many other options for this section exist; see the `XF86Config` man page for a complete description. In practice these other options are not necessary to get XFree86 initially working.

5 Filling in video card information

Your `XF86Config` file is now ready to go, with the exception of complete information on the video card. What we're going to do is use the X server to probe for the rest of this information, and fill it into `XF86Config`.

Instead of probing for this information with the X server, the `XF86Config` values for many cards are listed in the files `modeDB.txt`, `AccelCards`, and `Devices`. These files are all found in `/usr/X11R6/lib/X11/doc`. In addition, there are various `README` files for certain chipsets. You should look in these files for information on your video card, and use that information (the clock values, chipset type, and any options) in the `XF86Config` file. If any information is missing, you can probe for it as described here.

In these examples we will demonstrate configuration for a #9 GXE 64 video card, which uses the `XF86_S3` chipset. This card happens to be the one which the author uses, but the discussion here applies to any video card.

The first thing to do is to determine the video chipset used on the card. Running `SuperProbe` (found in `/usr/X11R6/bin`) will tell you this information, but you need to know the chipset name as it is known to the X server.

To do this, run the command

```
X -showconfig
```

This will give the chipset names known to your X server. (The man pages for each X server list these as well.) For example, with the accelerated `XF86_S3` server, we obtain:

```
XFree86 Version 3.1 / X Window System
(protocol Version 11, revision 0, vendor release 6000)
Operating System: Linux
Configured drivers:
  S3: accelerated server for S3 graphics adaptors (Patchlevel 0)
      mmio_928, s3_generic
```

The valid chipset names for this server are `mmio_928` and `s3_generic`. The `XF86_S3` man page describes these chipsets and which videocards use them. In the case of the #9 GXE 64 video card, `mmio_928` is appropriate.

If you don't know which chipset to use, the X server can probe it for you. To do this, run the command

```
X -probeonly > /tmp/x.out 2>&1
```

if you use `bash` as your shell. If you use `csh`, try:

```
X -probeonly &> /tmp/x.out
```

You should run this command while the system is unloaded, that is, while no other activity is occurring on the system. This command will also probe for your video card dot clocks (as seen below), and system load can throw off this calculation.

The output from the above (in `/tmp/x.out` should contain lines such as the following:

```
XFree86 Version 3.1 / X Window System
(protocol Version 11, revision 0, vendor release 6000)
Operating System: Linux
Configured drivers:
  S3: accelerated server for S3 graphics adaptors (Patchlevel 0)
      mmio_928, s3_generic
   ...
(--) S3: card type: 386/486 localbus
(--) S3: chipset:   864 rev. 0
(--) S3: chipset driver: mmio_928
```

Here, we see that the two valid chipsets for this server (in this case, `XF86_S3`) are `mmio_928` and `s3_generic`. The server probed for and found a video card using the `mmio_928` chipset.

In the `Device` section of the `XF86Config` file, add a `Chipset` line, containing the name of the chipset as determined above. For example,

```
Section "Device"
        # We already had Identifier here...
        Identifier "#9 GXE 64"
        # Add this line:
        Chipset "mmio_928"
EndSection
```

Now we need to determine the driving clock frequencies used by the video card. A driving clock frequency, or dot clock, is simply a rate at which the video card can send pixels to the monitor. As we have seen, each monitor resolution has a dot clock associated with it. Now we need to determine which dot clocks are made available by the video card.

First you should look into the files (`modeDB.txt`, and so forth) mentioned above and see if your card's clocks are listed there. The dot clocks will usually be a list of 8 or 16 values, all of which are in MHz. For example, when looking at `modeDB.txt` we see an entry for the Cardinal ET4000 video board, which looks like this:

```
# chip    ram   virtual   clocks                          default-mode  flags
 ET4000   1024  1024 768   25  28  38  36  40  45  32   0  "1024x768"
```

As we can see, the dot clocks for this card are 25, 28, 38, 36, 40, 45, 32, and 0 MHz.

In the `Devices` section of the `XF86Config` file, you should add a `Clocks` line containing the list of dot clocks for your card. For example, for the clocks above, we would add the line

```
Clocks 25 28 38 36 40 45 32 0
```

to the `Devices` section of the file, after `Chipset`. Note that the order of the clocks is important! Don't resort the list of clocks or remove duplicates.

If you cannot find the dot clocks associated with your card, the X server can probe for these as well. Using the `X -probeonly` command described above, the output should contain lines which look like the following:

```
(--) S3: clocks:  25.18  28.32  38.02  36.15  40.33  45.32  32.00  00.00
```

We could then add a `Clocks` line containing all of these values, as printed. You can use more than one `Clocks` line in `XF86Config` should all of the values (sometimes there are more than 8 clock values printed) not fit onto one line. Again, be sure to keep the list of clocks in order as they are printed.

Be sure that there is no `Clocks` line (or that it is commented out) in the `Devices` section of the file when using `X -probeonly` to probe for the clocks. If there is a `Clocks` line present, the server will *not* probe for the clocks—it will use the values given in `XF86Config`.

Note that some accelerated video boards use a programmable clock chip. (See the `XF86_Accel` man page for details; this generally applies to S3, AGX, and XGA-2 boards.) This chip essentially allows the X server to tell the card which dot clocks to use. If this is the case, then you may not find a list of dot clocks for the card in any of the above files. Or, the list of dot clocks printed when using `X -probeonly` will only contain one or two discrete clock values, with the rest being duplicates or zero.

For boards which use a programmable clock chip, you would use a `ClockChip` line, instead of a `Clocks` line, in your `XF86Config` file. `ClockChip` gives the name of the clock chip as used by the video card; the man pages for each server describe what these are. For example, in the file `README.S3`, we see that several S3-864 video cards use an "ICD2061A" clock chip, and that we should use the line

```
ClockChip "icd2061a"
```

instead of `Clocks` in the `XF86Config` file. As with `Clocks`, this line should go in the `Devices` section, after `Chipset`.

Similarly, some accelerated cards require you to specify the RAMDAC chip type in the `XF86Config` file, using a `Ramdac` line. The `XF86_Accel` man page describes this option. Usually, the X server will correctly probe for the RAMDAC.

Some video card types require you to specify several options in the `Devices` section of `XF86Config`. These options will be described in the man page for your server, as well as in the various files (such as `README.cirrus` or `README.S3`. These options are enabled using the `Option` line. For example, the #9 GXE 64 card requires two options:

```
        Option "number_nine"
        Option "dac_8_bit"
```

Usually, the X server will work without these options, but they are necessary to obtain the best performance. There are too many such options to list here, and they each depend on the particular video card being used. If you must use one of these options, fear not—the X server man pages and various files in `/usr/X11R6/lib/X11/doc` will tell you what they are.

So, when you're finished, you should end up with a `Devices` section which looks something like this:

```
Section "Device"
        # Device section for the #9 GXE 64 only!
        Identifier "#9 GXE 64"
        Chipset "mmio_928"
        ClockChip "icd2061a"
        Option "number_nine"
        Option "dac_8_bit"
EndSection
```

Most video cards will require a `Clocks` line, instead of `ClockChip`, as described above. The above `Device` entry is only valid for a particular video card, the #9 GXE 64. It is given here only as an example.

There are other options that you can include in the `Devices` entry. Check the X server man pages for the gritty details, but the above should suffice for most systems.

6 Running XFree86

With your `XF86Config` file configured, you're ready to fire up the X server and give it a spin. First, be sure that `/usr/X11R6/bin` is on your path.

The command to start up XFree86 is

```
startx
```

This is a front-end to `xinit` (in case you're used to using `xinit` on other UNIX systems).

This command will start the X server and run the commands found in the file `.xinitrc` in your home directory. `.xinitrc` is just a shell script containing X clients to run. If this file does not exist, the system default `/usr/X11R6/lib/X11/xinit/xinitrc` will be used.

A standard `.xinitrc` file looks like this:

```
#!/bin/sh

xterm -fn 7x13bold -geometry 80x32+10+50 &
xterm -fn 9x15bold -geometry 80x34+30-10 &
oclock -geometry 70x70-7+7 &
xsetroot -solid midnightblue &
```

```
exec twm
```

This script will start up two `xterm` clients, an `oclock`, and set the root window (background) color to `midnightblue`. It will then start up `twm`, the window manager. Note that `twm` is executed with the shell's `exec` statement; this causes the `xinit` process to be replaced with `twm`. Once the `twm` process exits, the X server will shut down. You can cause `twm` to exit by using the root menus: depress mouse button 1 on the desktop background—this will display a pop up menu which will allow you to `Exit Twm`.

Be sure that the last command in `.xinitrc` is started with `exec`, and that it is not placed into the background (no ampersand on the end of the line). Otherwise the X server will shut down as soon as it has started the clients in the `.xinitrc` file.

Alternately, you can exit X by pressing `ctrl-alt-backspace` in combination. This will kill the X server directly, exiting the window system.

The above is a very, very simple desktop configuration. Many wonderful programs and configurations are available with a bit of work on your `.xinitrc` file. For example, the `fvwm` window manager will provide a virtual desktop, and you can customize colors, fonts, window sizes and positions, and so forth to your heart's content. Although the X Window System might appear to be simplistic at first, it is extremely powerful once you customize it for yourself.

If you are new to the X Window System environment, we strongly suggest picking up a book such as *The X Window System: A User's Guide*. Using and configuring X is far too in-depth to cover here. See the man pages for `xterm`, `oclock`, and `twm` for clues on getting started.

7 Running Into Trouble

Often, something will not be quite right when you initially fire up the X server. This is almost always caused by a problem in your `XF86Config` file. Usually, the monitor timing values are off, or the video card dot clocks set incorrectly. If your display seems to roll, or the edges are fuzzy, this is a clear indication that the monitor timing values or dot clocks are wrong. Also be sure that you are correctly specifying your video card chipset, as well as other options for the `Device` section of `XF86Config`. Be absolutely certain that you are using the right X server and that `/usr/X11R6/bin/X` is a symbolic link to this server.

If all else fails, try to start X "bare"; that is, use a command such as:

```
X > /tmp/x.out 2>&1
```

You can then kill the X server (using the `ctrl-alt-backspace` key combination) and examine the contents of `/tmp/x.out`. The X server will report any warnings or errors—for example, if your video card doesn't have a dot clock corresponding to a mode supported by your monitor.

The file `VideoModes.doc` included in the XFree86 distribution contains many hints for tweaking the values in your `XF86Config` file.

Remember that you can use `ctrl-alt-numeric +` and `ctrl-alt-numeric -` to switch between the video modes listed on the `Modes` line of the `Screen` section of `XF86Config`. If the highest resolution mode doesn't look right, try switching to lower resolutions. This will let you know, at least, that those parts of your X configuration are working correctly.

Also, check the vertical and horizontal size/hold knobs on your monitor. In many cases it is necessary to adjust these when starting up X. For example, if the display seems to be shifted slightly to one side, you can usually correct this using the monitor controls.

The USENET newsgroup `comp.windows.x.i386unix` is devoted to discussions about XFree86, as is `comp.os.linux.x`. It might be a good idea to watch that newsgroup for postings relating to your video configuration—you might run across someone with the same problems as your own.

8 Copyright