README for X11R6.9 on NetBSD

Rich Murphey, David Dawes, Marc Wandschneider, Mark Weaver, Matthieu Herrb

1. What and Where is X11R6.9?

X11R6.9 is an Open Source version of the X Window System that supports several UNIX(R) and UNIX-like operating systems (such as Linux, the BSDs and Solaris x86) on Intel and other platforms. This version is compatible with X11R6.6, and is based on the XFree86 4.4.0RC2 code base, which, in turn was based on the X consortium sample implementation.

See the Copyright Notice.

The sources for X11R6.9 are available from:

http://wiki.x.org

X11R6.9 also builds on other NetBSD architectures. See section *Building on other architectures* (section 7.4, page 6) for details.

2. New OS dependent features

See the Release Notes for non-OS dependent new features in X11R6.9.

2.1 New OS dependent features in 4.2.0

- Support of client side on NetBSD/sparc64
- Support for in-kernel MTRR and AGP support in NetBSD 1.5Y

2.2 New OS dependent features in 4.1.0

- Enable wide characters support in NetBSD 1.5P and later.

2.3 New OS dependent features in 4.0.2

- A fix for libXmu OS detection which was broken since `unix` isn't defined anymore by the C preprocessor.
- (limited) native wscons support. This is not activated by default.
- Updates to the aperture driver
- Support for multithread libraries with GNU pth
- Add `/usr/pkg/bin` to the default user path.

2.4 New OS dependent features in 4.0.1

- Support for NetBSD 1.5_ALPHA

- The Xsun server can be built on NetBSD/sparc

2.5 New OS dependent features in 4.0

- Preliminary APM support.

2.6 New OS dependent features in 3.9.18

- Soft-booting secondary cards through the int10 BIOS interface is now possible using the x86emu real mode emulator.

2.7 New OS dependent features in 3.9.17

- Support for *silken mouse* with the wsmouse protocol has been added.
- A new version of the Aperture driver which provides MTRR support is included.

3. Installing the Binaries

Refer to the Installation Document for detailed installation instructions.

4. Configuring X for Your Hardware

The `/etc/X11/xorg.conf` file tells the X server what kind of monitor, video card and mouse you have. You *must* create it to tell the server what specific hardware you have.

You'll need info on your hardware:

- Your mouse type, baud rate and its /dev entry.
- The video card's chipset (e.g. ET4000, S3, etc).
- Your monitor's sync frequencies.

For details about the `xorg.conf` file format, refer to the *xorg.conf(5)* manual page.

Once you've set up a xorg.conf file, you can fine tune the video modes with the `xvidtune` utility.

4.1 About mouse configuration

X11R6.9 has support for the mouse driver included in the **wscons** console driver introduced by NetBSD 1.4. Specify "`wsmouse`" as the protocol and "`/dev/wsmouse0`" as the device in `/etc/X11/xorg.conf` if you're using NetBSD 1.4 or later with a PS/2 mouse.

For older releases, the NetBSD **pms** mouse driver handles PS/2 style mice as Busmouse. Specify the protocol as "`busmouse`" in the mouse section of your `xorg.conf` file if you're using a PS/2 mouse with NetBSD 1.3 or former releases.

Only standard PS/2 mice are supported by this driver. Newest PS/2 mice that send more than three bytes at a time (especially Intellimouse, or MouseMan+ with a wheel) are not supported by NetBSD 1.3 and former releases.

See README.mouse for general instruction on mouse configuration.

5. Running X

The easiest way for new users to start X windows is to type:

```
startx >& startx.log
```

Error messages are lost unless you redirect them because the server takes over the screen.

To get out of X windows, type: "`exit`" in the console xterm. You can customize your X by

creating `.xinitrc`, `.xserverrc`, and `.twmrc` files in your home directory as described in the xinit and startx man pages.

5.1 Starting Xdm, the display manager

To start the display manager, log in as root on the console and type: "`xdm -nodaemon`".

You can start xdm automatically on bootup by changing the line

```
xdm=NO                  xdm_flags=""            # x11 display manager
```

to:

```
xdm=YES                 xdm_flags=""            # x11 display manager
```

in `/etc/rc.conf`.

Under NetBSD 1.4 and later with the wscons console driver, you must enable a virtual console for the X server first. To do this follow these steps:

- Make sure the device file exists. If not, "`cd /dev ; ./MAKEDEV wscons`".
- Next, make sure your kernel wants to do wscons. (see *below* (section 6.1, page 4)).
- Next, make sure "`wscons=YES`" in `/etc/rc.conf`.
- Next, make sure `/etc/wscons.conf` exists. The relevant bits:

```
#screen 0       -       vt100
screen  1       -       vt100
screen  2       -       vt100
screen  3       -       vt100
screen  4       -       -
screen  5       -       vt100
```

(Thanks to Mason Loring Bliss <mason@acheron.middleboro.ma.us> for this explanation)

6. Kernel Support for X

To make sure X support is enabled under NetBSD, the following line must be in your config file in `/sys/arch/i386/conf`:

```
options XSERVER, UCONSOLE
```

6.1 Console drivers

The server supports the standard NetBSD/i386 console drivers: pccons, pcvt and wscons (in pcvt compatibility mode). They are detected at runtime and no configuration of the server itself is required.

The pccons driver is the most widely tested and is the console driver contained in the NetBSD binary distribution's kernels.

The pcvt console driver was bundled with NetBSD until 1.4. The pcvt X mode is compatible with the pccons driver X mode. It offers several virtual consoles and international keyboard support. In order to use this driver, change the line:

```
device          pc0     at isa? port "IO_KBD" irq 1
```

to

```
device          vt0     at isa? port "IO_KBD" irq 1
```

in your kernel config file, and rebuild and install your kernel.

Wscons is the current console driver, included in NetBSD 1.4 and later. For now, X supports wscons using the pcvt compatibility mode, so be sure to have the lines:

```
options         WSDISPLAY_COMPAT_PCVT           # emulate some ioctls
options         WSDISPLAY_COMPAT_SYSCONS        # emulate some ioctls
options         WSDISPLAY_COMPAT_USL          # VT handling
options         WSDISPLAY_COMPAT_RAWKBD         # can get raw scancodes
```

in your kernel configuration file if you're using wscons. Refer to the *wscons(4)* and *wsmouse(4)* manual pages for informations on how to configure wscons into the kernel.

6.2 Aperture Driver

By default NetBSD include the BSD 4.4 kernel security feature that disable access to the `/dev/mem` device when in multi-users mode. But X.Org foundation X servers can take advantage (or require) linear access to the display memory.

Most X11R6.9 card drivers require linear memory access. There are two ways to allow X to access linear memory:

The first way is to disable the kernel security feature by adding "`option INSECURE`" in the kernel configuration file and build a new kernel.

The second way is to install the aperture driver, included in source form in `xc/programs/Xserver/hw/xfree86/etc/apNetBSD.shar` in the X11R6.9 source distribution. Unpack it in a new directory of your choice by running:

```
sh apNetBSD.shar
```

By default the aperture driver will be installed in `/usr/local/aperture`. You can change this default directory by editing `Makefile.inc` before building it.

Then run "`make build`" as root to install it. To enable it, add the following line to `/etc/lkm.conf`:

```
/usr/local/aperture/lkm/xf86.o  -  - /usr/local/aperture/lkm/xf86_mod_install - -
```

and set "`lkm=YES`" in `/etc/rc.conf`

Reboot your system. X will auto-detect the aperture driver if available.

Warning 1: if you boot another kernel than `/netbsd`, loadable kernel modules can crash your system. Always boot in single user mode when you want to run another kernel.

Warning 2: the aperture driver only allows one access at a time (so that the system is in the same security state once X is launched). This means that if you run multiple servers on multiples VT, only the first one will have linear memory access. Use "option INSECURE" if you need more that one X server at a time.

Starting with XFree86 3.9.17, the XFree86 aperture driver also supports MTRR write combining on Pentiums II and AMD K6 class processors.

6.3 MIT-SHM

NetBSD 1.0 and later supports System V shared memory. If X detects this support in your kernel, it will support the MIT-SHM extension.

To add support for system V shared memory to your kernel add the lines:

```
# System V-like IPC
options         SYSVMSG
options         SYSVSEM
options         SYSVSHM
```

to your kernel config file.

7. Rebuilding the X Distribution

You should configure the distribution by editing `xc/config/cf/host.def` before compiling. To compile the sources, invoke "`make World`" in the xc directory.

7.1 Perl support

Starting with XFree86 4.0.2, perl is needed to build the fonts in XFree86. Since perl is not included with standard NetBSD installation, fonts that need perl are not built by default.

If you have installed perl (from the NetBSD packages, for instance), add the line

```
#define HasPerl YES
```

in `xc/config/cf/host.def` before rebuilding X.

7.2 Aperture driver

To build the X server with the Aperture driver enabled, you should unpack `apNetBSD.shar` and install it first.

Then edit `xc/config/cf/host.def` and add the line

```
#define HasNetBSDApertureDriver    YES
```

to it before rebuilding X.

7.3 Console drivers

X has a configuration option to select the console drivers to use in `host.def`:

- if you're using pccons put:

  ```
  #define XFree86ConsoleDefines -DPCCONS_SUPPORT
  ```

- if you're using pcvt put:

  ```
  #define XFree86ConsoleDefines -DPCVT_SUPPORT
  ```

If you don't define **XFree86ConsoleDefines** in `host.def` the pccons and pcvt drivers will be supported by default.

Experimental native support for the wscons console driver can be built by adding:

```
#define XFree86ConsoleDefines -DWSCONS_SUPPORT
```

to `xc/config/host.def` before rebuilding the server. This has not been thoroughly tested, except on the macppc.

For the i386, you should include both pcvt and wscons support in order to use the pcvt compatibility mode of wscons:

```
#define XFree86ConsoleDefines -DPCVT_SUPPORT -DWSCONS_SUPPORT
```

7.4 Building on other architectures

Note that the NetBSD project has now its own source tree, based on the X source tree, with some local modifications. You may want to start with this tree to rebuild from sources. The NetBSD xsrc source tree is available at: ftp://ftp.netbsd.org/pub/NetBSD/NetBSD-current/xsrc/

8. Building New X Clients

The easiest way to build a new client (X application) is to use `xmkmf` if an `Imakefile` is included in the sources. Type "`xmkmf -a`" to create the Makefiles, check the configuration if necessary and type "`make`". Whenever you install additional man pages you should update `whatis.db` by running "`makewhatis /usr/X11R6/man`".

When porting clients to *BSD systems, make use of the symbol **BSD** for code which is truly BSD-specific. The value of the symbol can be used to distinguish different BSD releases. For example, code specific to the Net-2 and later releases can use:

```
#if (BSD >= 199103)
```

To ensure that this symbol is correctly defined, include `<sys/param.h>` in the source that requires it. Note that the symbol **CSRG_BASED** is defined for *BSD systems in XFree86 3.1.1 and later. This should be used to protect the inclusion of `<sys/param.h>`.

For code that really is specific to a particular i386 BSD port, use __**FreeBSD**__ for FreeBSD, __**NetBSD**__ for NetBSD, __**OpenBSD**__ for OpenBSD, and __**bsdi**__ for BSD/386.

9. Thanks

Many thanks to all people who contributed to make XFree86 work on *BSD, in particular: **David Dawes**, **Todd Fries**, **Rod Grimes**, **Charles Hannum**, **Amancio Hasty**, **Christoph Robitschko**, **Matthias Scheler**, **Michael Smith**, **Ignatios Souvatzis**, **Jack Velte**, **Nate Williams** and **Pace Willison**.

CONTENTS

1. What and Where is X11R6.9? 1

2. New OS dependent features 1
   - 2.1 New OS dependent features in 4.2.0 1
   - 2.2 New OS dependent features in 4.1.0 1
   - 2.3 New OS dependent features in 4.0.2 1
   - 2.4 New OS dependent features in 4.0.1 1
   - 2.5 New OS dependent features in 4.0 2
   - 2.6 New OS dependent features in 3.9.18 2
   - 2.7 New OS dependent features in 3.9.17 2

3. Installing the Binaries 2

4. Configuring X for Your Hardware 2
   - 4.1 About mouse configuration 2

5. Running X 2
   - 5.1 Starting Xdm, the display manager 3

6. Kernel Support for X 3
   - 6.1 Console drivers 3
   - 6.2 Aperture Driver 4
   - 6.3 MIT-SHM 4

7. Rebuilding the X Distribution 5
   - 7.1 Perl support 5
   - 7.2 Aperture driver 5
   - 7.3 Console drivers 5
   - 7.4 Building on other architectures 6

8. Building New X Clients 6

9. Thanks 6