# Distributed multihead support with Linux and Xdmx

## Create multimonitor setups displaying one contiguous desktop

Nathan Harrington March 28, 2006

Learn about the tools available to develop your own multiscreen configuration and physical layout to enhance your computing experience. You can use Linux® and Xdmx to create one contiguous desktop across multiple display devices attached to separate computers. Combine your available laptop and desktop computers running Linux to create one large display for enhanced productivity. Explore large-scale display-wall setups and the creation of multihead setups without purchasing graphics cards.

Adding monitors to your computing setup will greatly enhance your productivity by removing the need for frequent task-switching. This and many other benefits are well known to those who use multihead graphics cards, or use several graphics cards in the same PC. With Linux and Xdmx, anyone with two display devices and two computers can enjoy the benefits of multihead setups without investing in new hardware.

Using Xdmx and some simple configuration advice, we will set up a computing environment with four monitors spread across one desktop and three laptop PCs. Read about high-performance visualization setups using Chromium and gigabit network speeds to deliver real-time 3-D graphics across multi-megapixel displays. Learn about tools available to develop your own multiscreen configuration and physical layout to enhance your computing experience.

## Requirements

This article was developed on four computers with three video cards, three screen sizes, and three separate resolution settings. Kernel levels 2.4.21-2.6.15 were used with X11 and xorg configurations across Red Hat Enterprise Linux (RHEL) V3.5, and Fedora Core 3 distributions. The point: Diversity of hardware and software is not an obstacle. You can install and use Xdmx on many distributions across many hardware options with success.

### Hardware

You'll need:

- Relatively modern CPUs in your setup -- anything above a 486 should be usable (although a bit slow)

- Fast networking -- Ethernet or better
- Graphics card capable of at least 16-bit color depth -- anything lower and you'll have difficulty with xinerama enabled displays

### Software

We developed on modern Red Hat and Fedora Core distributions, but many other distributions should suffice. If your distribution of choice supports RPMs, you should be able to get up and running without having to compile the Xdmx application from source.

## Xdmx acquisition and installation

### Software acquisition

Head on over to Sourceforge.net to grab Xdmx (see Related topics). For the purposes of this article, the RPM should be fine. Look for the `dmx-1.2.20040630-1.i386.rpm` file in the Download DMX section. If you're using a very old distribution or one that does not support RPMs, you may have to compile DMX from scratch. The source is in the Download DMX section, but compiling instructions are beyond the scope of this article.

### Installation

Now that you have the Xdmx RPM, you need to run the installation process. On RHEL V3.5-based systems, `rpm -Uvh dmx-1.2.20040630-1.i386.rpm` should install the Xdmx application with no errors. On Fedora Core 3 and higher systems, when running the RPM command, you may see an error similar to `file /usr/X11R6/lib/libdmx.a from install of dmx-1.2.20040630-1 conflicts with file from package xorg-x11-devel-6.8.2-31`.

As a workaround, use the `--force` option: `rpm --force -Uvh dmx-1.2.20040630-1.i386.rpm`. Warning: Forcing the installation of the Xdmx software may create instability in your development environment. I've been running development boxes with the Xdmx software "force-installed" with zero errors for more than a year, but your experience may differ. If you don't want to force the install, compile the Xdmx application from source, then install.

On Debian-based systems or those that use `apt-get`, use `apt-get install xdmx` to install and set up Xdmx.

Xdmx needs to be installed on every machine you want to have as part of your multihead display setup. Both the server and client are included in the Xdmx install package, so if you want to add a display node, change your control node, or have a hardware failure in an existing node, there is no need to reinstall software to support different configurations.

## Configure PCs for multihead operation

### Xdmx manual

Xdmx comes with fantastic manual information that will provide all you need to get started with distributed multihead displays. This article focuses on helping you surmount some of the

more difficult problems associated with disparate hardware setup. Read on for some real-world examples of how to use Xdmx and how to overcome some of the challenges you may face.

## One PC, one laptop

For this initial example, we will use one PC and one laptop in a two-screen setup, with the desktop stretched across the display. To begin, designate one computing resource the control node -- in our case, it will be the PC.

## X configuration file setup

A big problem with using disparate hardware is the differing feature support among computing resources. You'll need to make sure the systems support the same set of options before Xdmx will work. In this example, all the systems do not have the same fonts installed. Xdmx has an option to handle this, which we will cover later. For now, note that GLX support and default display bit depth are common issues that will prevent Xdmx from running correctly. To spread your desktop across multiple screens, make sure the root window bit depths are the same on every screen. If you want multiple independent desktops, bit depth matching between displays is not necessary. Since we want one big display for this example, modify your Xorg.conf or XF86Config file to select 16-bit color depth as the default. (You can select 24 or whatever bit depth all of your equipment will support.) For this example, 16-bit is the highest color depth all of the hardware will support.

Make a backup copy of your Xorg.conf or XF86Config file and make the changes shown below. For example, in the RHEL 3.5 box, the XF86Config file has the lines:

## Listing 1. sample XF86Config file

```
Section "Screen"
        Identifier "Screen0"
        Device     "Videocard0"
        Monitor    "Monitor0"
        DefaultDepth     24
        SubSection "Display"
                Depth     24
                Modes    "1400x1050" "1280x1024" \
                "1280x960" "1152x864" "1024x768" "800x600" "640x480"
        EndSubSection
EndSection
```

Change `DefaultDepth 24` to `DefaultDepth 16` and the line under the SubSection for Display from `Depth 24` to `Depth 16`.

If you have custom drivers specific to your hardware, you may need to revert to default settings or try compatible hardware support. See your hardware documentation for detailed information on what your hardware can support. For example, the RHEL V3.5 laptop in this example contains an ATI FireGL Mobility T2 card, and associated drivers. The change to the XF86Config file as shown above will produce an error because the fglrx driver does not support the 16-bit depth setting. The solution is to run `system-config-xfree86` (system-config-display on Fedora Core) and select a VESA mode for compatibility support. In the Advanced tab, select **VESA Driver (generic)** as your video card. In the Monitor Type section, configure the settings to support the maximum resolution

your display can handle. Go back to the Display tab and select your resolution and **Thousands of Colors** as your bit depth.

Now that the bit depths have been matched, you are ready to try a multiscreen setup. On the client node, start an X session. If you are in runlevel 3, try the command `xinit` to start a bare-bones X session. If you are already logged into a graphical window manager session, such as GNOME or KDE, start an `Xterm`. In the Xterm window on the client node screen, type `xhost + control_node_ip`, where the `control_node_ip` is the dotted quad of your control node. For this example, the control node is 192.168.1.101, so I would run the command `xhost + 192.168.1.101` on the client node.

## Xdmx start command

On the control node, start your default X Window System session. If you are running xdm, for example, you'll be presented with your login screen on boot-up; then start a GNOME or KDE session. Regardless, start an `Xterm` on your control node. Start a new Xdmx session that spans both nodes running the twm window manager. The window manager is your choice, of course, but note that GNOME does not support running two instances on the same computer. Run the following command on the control node:

## Listing 2. Sample Xdmx start command

```
startx `which twm` --          \
  /usr/bin/X11R6/Xdmx :1       \
  -display control_node_ip:0   \
  -display client_node_1_ip:0 \
  -ignorebadfontpaths          \
```

Where `control_node_ip` is 192.168.1.101 and the `client_node_1_ip` is the IP address of the client node. That command is broken down in Table 1.

## Table 1. The start command explained

| Each line in start ... | ... And what it means |
|---|---|
| startx `which twm` | Start a new X session with the twm window manager |
| -- /usr/bin/X11R6/Xdmx :1 | Start the Xdmx program, as X session on :1 |
| -display control_node_ip:0 | Use the display on the control node as the first monitor |
| -display client_node_1_ip:0 | Use the display on the first client node as the second monitor |
| -ignorebadfontpaths | Never liked them anyway |
| +xinerama | Treat the display as one desktop |

If the command completed successfully, you should see a hash background across both screens. Move your mouse around on the control node to verify that you have one desktop available across both screens to the input devices on the control node. Press **Ctrl+Alt+q** to exit Xdmx.

## Xdmx errors and resolution

If the command did not complete successfully, you may see an error like this:

## Listing 3. Sample Xdmx error message

```
(II) dmx: ===== End of Summary =====
(!!) dmx: The default visual for screen #0 does not match any of the
(!!) dmx: consolidated visuals from Xinerama (listed above)
(!!) dmx: The default visual for screen #1 does not match any of the
(!!) dmx: consolidated visuals from Xinerama (listed above)
(Fatal Error) dmx: dmxConnectionBlockCallback: invalid screen(s) found
XIO:  fatal IO error 104 (Connection reset by peer) on X server ":1.0"
      after 0 requests (0 known processed) with 0 events remaining.
```

This is one of the few apparent deficiencies of the Xdmx program -- cryptic error messages. It turns out that this particular error message is due to incompatible GLX configurations on the client and control nodes. Although these error messages may be cryptic to we mere mortals, Xdmx functionality always delivers simplicity. Add the option `-noglxproxy` to the command and try again:

## Listing 4. Sample Xdmx command with noglxproxy

```
startx `which twm` --          \
  /usr/bin/X11R6/Xdmx :1       \
  -display control_node_ip:0   \
  -display client_node_1_ip:0 \
  -ignorebadfontpaths          \
  +xinerama                    \
  -noglxproxy
```

Bingo: no more problems. You may have noticed that when Xdmx fails, it causes strange keyboard problems. This is not you. I've noticed that the Ctrl, Shift, and Alt keys fail to work properly when Xdmx fails. The best solution is to restart your window manager, and all will be well.

## Three client nodes, three displays

Now that we have a two-node setup, let's add a third client node. From runlevel 3, start a bare-bones X session on the third client node with `xinit`. Enter `xhost + contrl_node_ip` in the `xterm` that appears once your X session has started. Modify the Xdmx start command to include the third node, as in Listing 5.

## Listing 5. Sample Xdmx command with noglxproxy and third node

```
startx `which twm` --          \
  /usr/bin/X11R6/Xdmx :1       \
  -display control_node_ip:0   \
  -display client_node_1_ip:0 \
  -display client_node_2_ip:0 \
  -ignorebadfontpaths          \
  +xinerama                    \
  -noglxproxy
```

It really is that easy for adding another node to a sequential horizontal configuration. Here's a screenshot showing three client nodes displaying a mosaic image from another developerWorks article:

### Figure 1. Three-node configuration


## Four client nodes, 2x2 display

What about a different display topology? How about a 2x2 square of displays? Xdmx has the answer: use the configuration file options with geometry specifiers for the desired setup. For example, if you wanted to have a 1024x768 display in the top left, a 1400x1050 display in the top right, a 1600x1200 display in the bottom left, and a 1024x768 display in the bottom right, you would use the following configuration file:

### Listing 6. Sample Xdmx configuration file

```
# xdmx.conf  - Xdmx configuration file
# quad config setup 2x2

virtual quad_config {
  display "client_node_upper_left_ip:0"   @0x0;
  display "client_node_upper_right_ip:0"  @1024x0;
  display "client_node_lower_left_ip:0"   @0x768;
  display "client_node_lower_right_ip:0"  @1024x768;
}
```

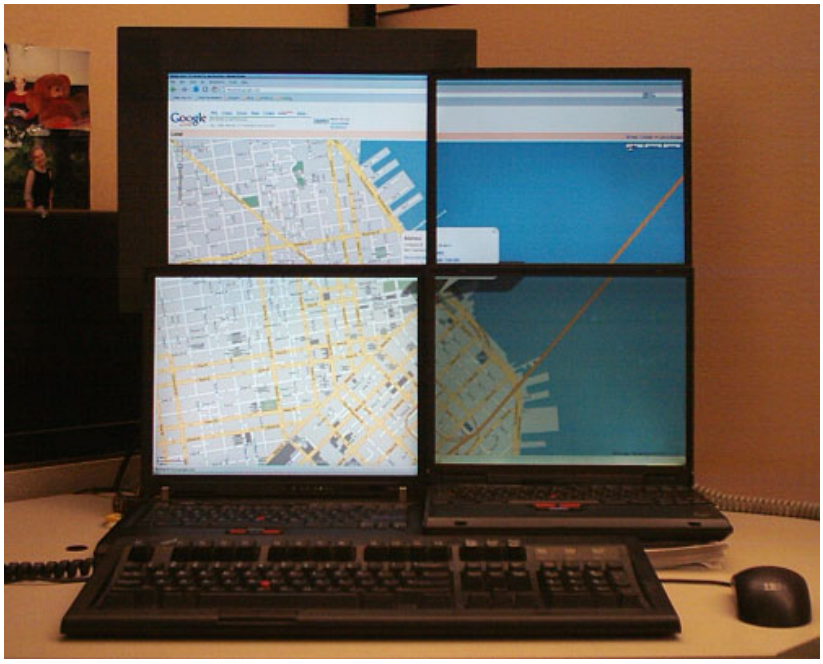
Run the above setup from the control node with the command:

### Listing 7. Sample Xdmx command using configuration file

```
startx `which twm`  -- /usr/bin/X11R6/Xdmx :1 \
  -configfile  xdmx.conf                      \
  -config  quad_config                        \
  -ignorebadfontpaths                         \
  -noglxproxy
```

I chose this physical display topology because it was the only way I could stack the display units close enough together to get the feel of a contiguous desktop. Here's a display of a 2x2 distributed multihead setup showing the San Francisco area on Google Maps:

### Figure 2. Four-node configuration



## Advanced configurations and display walls

Keep in mind that your displays need not be physically located next to each other. Following is a screenshot of one configuration focused on programming, which is four individual desktops using the Ion3 window manager:

### Figure 3. Four-node configuration, one flying

Note the display on the far left. The user can move windows to this display he wants to be able to view with a glance, but they remain out of peripheral vision to prevent distractions while busy creating more bugs. Use your Xdmx configuration file with the location and geometry specifiers to create any number of topologies that suit your working environment.

## 3-D processing setups

Chromium is a package designed to allow each node in your Xdmx setup to process OpenGL information separately and only for what is displayed on that node. This provides an excellent method for using the processing power of your client nodes (and graphics cards) to render large 3-D environments quickly. Please see the references below for more about Chromium and configuring systems for enhanced 3-D display.

### Display walls

The primary usage of Xdmx is in large-scale display systems at universities and research institutions focusing on visualization of large datasets. These setups frequently use Chromium for accelerated 3-D displays of complex datasets involving identical hardware on the client nodes and dedicated machine setups. This article focuses more on the desktop user and how to use existing hardware to create a distributed multihead setup. For some fantastic examples of large-scale display walls (16000x4800 anyone?) and auto-configuring your display nodes, see Related topics.

## Conclusion

With your experience setting up Xdmx multihead displays for your Linux desktop, you can greatly increase your productivity without investing in new hardware. It's easy to add machines to your display configuration easily with open source Xdmx and Linux. Impress your friends with your monstrous desktop size -- geek out to your very own display wall. Write more code, squash more bugs, or read three developerWorks articles at the same time.

## Related topics

- Learn all about Xdmx from the source.
- See the fantastic LambdaVision high-resolution display wall using Xdmx.
- Read the Xdmx Mini How-To.
- Visit sourceforge.net to read more about Chromium.
- Download a free trial version of WebSphere Application Server V6.0.
- Innovate your next open source development project with IBM trial software, available for download or on DVD.