DRI Compilation Guide

VA Linux Systems, Inc. Professional Services - Graphics.

21 April 2001

1. Preamble

1.1 Copyright

Copyright © 2000-2001 by VA Linux Systems, Inc. All Rights Reserved.

Permission is granted to make and distribute verbatim copies of this document provided the copyright notice and this permission notice are preserved on all copies.

1.2 Trademarks

OpenGL is a registered trademark and SGI is a trademark of Silicon Graphics, Inc. Unix is a registered trademark of The Open Group. The 'X' device and X Window System are trademarks of The Open Group. XFree86 is a registered trademark of The XFree86 Project, Inc. Linux is a registered trademark of Linus Torvalds. Intel is a registered trademark of Intel Corporation. 3Dlabs, GLINT, and Oxygen are either registered trademarks or trademarks of 3Dlabs Inc. Ltd. 3dfx, Voodoo3, Voodoo4, and Voodoo5 are registered trademarks of 3dfx Interactive, Incorporated. Matrox is a registered trademark of Matrox Electronic Systems Ltd. ATI Rage and Radeon is a registered trademark of ATI Technologies, Inc. All other trademarks mentioned are the property of their respective owners.

2. Introduction

This document describes how to download, compile and install the DRI. The DRI provides 3D graphics hardware acceleration for the XFree86 project. This information is intended for experienced Linux developers. Beginners are probably better off installing precompiled packages.

Edits, corrections and updates to this document may be mailed to <brian@tungstengraphics.com>.

Last updated on 13 February 2002 by Brian Paul.

3. Prerequisites

You'll need the following:

- An installation of XFree86 4.1 or later. The DRI tree has been pruned down to minimize its size. But in order to build the DRI tree you need to have recent X header files, etc. already installed. If you don't have XFree86 4.1 (or later) installed you can probably install it from RPMs (or another package format). Or, you can download XFree86 as sources and compile/install it yourself.

- At least 200MB of free disk space. If you compile for debugging (the -g option) then you'll need about 600MB.
- GCC compiler and related tools.
- ssh (secure shell) if you're a DRI developer and don't want to use anonymous CVS download.
- A 2.4.x Linux Kernel. See below for details.
- FreeBSD support is not currently being maintained and may not work.

The DRI 3D drivers generally work on systems with Intel or AMD CPUs. However, limited support for Alpha and PowerPC support is underway.

For 3dfx Voodoo hardware, you'll also need the Glide3 runtime library (libglide3-v3.so for Voodoo3 or libglide3-v5.so for Voodoo4/5). These can be downloaded from the DRI website. You can compile them yourself, but it's often a painful process.

For Matrox G200/G400, Intel i810/i830 or ATI Rage128/Radeon hardware, you'll also need AGP support in your Linux kernel, either built-in or as a loadable module.

4. Linux Kernel Preparation

Only the Linux 2.4.x kernels are currently supported by the DRI hardware drivers. 2.5.x kernels may work, but aren't tested.

Most of the DRI drivers require AGP support and using Intel Pentium III SSE optimizations also requires an up-to-date Linux kernel. Configuring your kernel correctly is very important, as features such as SSE optimizations will be disabled if your kernel does not support them. Thus, if you have a Pentium III processor, you must configure your kernel for the Pentium III processor family.

Building a new Linux kernel can be difficult for beginners but there are resources on the Internet to help. This document assumes experience with configuring, building and installing Linux kernels.

Linux kernels can be downloaded from www.kernel.org

Here are the basic steps for kernel setup.

- Download the needed kernel and put it in /usr/src. Create a directory for the source and unpack it. For example:

```
cd /usr/src
rm -f linux
mkdir linux-2.4.x
ln -s linux-2.4.x linux
bzcat linux-2.4.x.tar.bz2 | tar xf -
```

  It is critical that /usr/src/linux point to your new kernel sources, otherwise the kernel headers **will not** be used when building the DRI. This will almost certainly cause compilation problems.

- Read /usr/src/linux/Documentation/Changes. This file lists the minimum requirements for all software packages required to build the kernel. You must upgrade at least gcc, make, binutils and modutils to at least the versions specified in this file. The other packages may not be needed. If you are upgrading from Linux 2.2.x you must upgrade your modutils package for Linux 2.4.x.

- Configure your kernel. You might, for example, use `make menuconfig` and do the following:
  - Go to *Code maturity level options*
  - Enable *Prompt for development and/or incomplete code/drivers*
  - hit ESC to return to the top-level menu
  - Go to *Processor type and features*
  - Select your processor type from *Processor Family*
  - hit ESC to return to the top-level menu
  - Go to *Character devices*
  - Disable *Direct Rendering Manager (XFree86 DRI support)* since we'll use the DRI code from the XFree86/DRI tree and will compile it there.
  - Go to */dev/agpgart (AGP Support) (EXPERIMENTAL) (NEW)*
  - Hit SPACE twice to build AGP support into the kernel
  - Enable all chipsets' support for AGP
  - It's recommended that you turn on MTRRs under *Processor type and Features*, but not required.
- Configure the rest of the kernel as required for your system (i.e. Ethernet, SCSI, etc)
- Exit, saving your kernel configuration.
- Edit your /etc/lilo.conf file. Make sure you have an image entry as follows (or similar):

```
image=/boot/vmlinuz
      label=linux.2.4.x
      read-only
      root=/dev/hda1
```

  The important part is that you have /boot/vmlinuz without a trailing version number. If this is the first entry in your /etc/lilo.conf AND you haven't set a default, then this will be your default kernel.
- Compile the new kernel.

```
cd /usr/src/linux-2.4.x
make dep
make bzImage
make modules
make modules_install
make install
```

  Note that last make command will automatically run lilo for you.
- Now reboot to use the new kernel.

5. CPU Architectures

In general, nothing special has to be done to use the DRI on different CPU architectures. There are, however, a few optimizations that are CPU-dependent. Mesa will determine at runtime which CPU-dependent optimizations should be used and enable them where appropriate.

5.1 Intel Pentium III Features

The Pentium III SSE instructions are used in optimized vertex transformation functions in the Mesa-based DRI drivers. On Linux, SSE requires a recent kernel (such as 2.4.0-test11 or later) both at compile time and runtime.

5.2 AMD 3DNow! Features

AMD's 3DNow! instructions are used in optimized vertex transformation functions in the Mesa-based DRI drivers. 3DNow! is supported in most versions of Linux.

5.3 Alpha Features

On newer Alpha processors a significant performance increase can be seen with the addition of the -mcpu= option to GCC. This option is dependent on the architecture of the processor. For example, -mcpu=ev6 will build specifically for the EV6 based AXP's, giving both byte and word alignment access to the DRI/Mesa drivers.

To enable this optimization edit your xc/config/host.def file and add the line:

#define DefaultGcc2AxpOpt -O2 -mcpu=ev6

Additional speed improvements to 3D rendering can be achieved by installing Compaq's Math Libraries (CPML) which can be obtained from http://www.support.compaq.com/alpha-tools/software/index.html

Once installed, you can add this line to your host.def to build with the CPML libraries:

#define UseCompaqMathLibrary YES

The host.def file is explained below.

6. Downloading the XFree86/DRI CVS Sources

The DRI project is hosted by SourceForge. The DRI source code, which is a subset of the XFree86 source tree, is kept in a CVS repository there.

The DRI CVS sources may be accessed either anonymously or as a registered SourceForge user. It's recommended that you become a registered SourceForge user so that you may submit non-anonymous bug reports and can participate in the mailing lists.

6.1 Anonymous CVS download:

1. Create a directory to store the CVS files:

```
cd ~
mkdir DRI-CVS
```

   You could put your CVS directory in a different place but we'll use `~/DRI-CVS/` here.

2. Check out the CVS sources:

```
cd ~/DRI-CVS
cvs -d:pserver:anonymous@cvs.dri.sourceforge.net:/cvsroot/dri login
  (hit ENTER when prompted for a password)
cvs -z3 -d:pserver:anonymous@cvs.dri.sourceforge.net:/cvsroot/dri co xc
```

   The -z3 flag causes compression to be used in order to reduce the download time.

6.2 Registered CVS download:

1. Create a directory to store the CVS files:

```
cd ~
mkdir DRI-CVS
```

   You could put your CVS directory in a different place but we'll use `~/DRI-CVS/` here.

2. Set the CVS_RSH environment variable:

```
setenv CVS_RSH ssh      // if using csh or tcsh
export CVS_RSH=ssh      // if using sh or bash
```

3. Check out the CVS sources:

```
cd ~/DRI-CVS
cvs -z3 -d:ext:YOURID@cvs.dri.sourceforge.net:/cvsroot/dri co xc
```

   Replace `YOURID` with your CVS login name. You'll be prompted to enter your sourceforge password.

   The -z3 flag causes compression to be used in order to reduce the download time.

6.3 Updating your CVS sources

In the future you'll want to occasionally update your local copy of the DRI source code to get the latest changes. This can be done with:

```
cd ~/DRI-CVS
cvs -z3 update -dA xc
```

The -d flag causes any new subdirectories to be created and -A causes most recent trunk sources to be fetched, not branch sources.

7. Mesa

Most of the DRI 3D drivers are based on Mesa (the free implementation of the OpenGL API). The relevant files from Mesa are already included in the XFree86/DRI source tree. *There is no need to download or install the Mesa source files separately.*

Sometimes a newer version of Mesa will be available than the version included in XFree86/DRI. Upgrading Mesa within XFree86/DRI is not always straightforward. It can be an error-prone undertaking, especially for beginners, and is not generally recommended. The DRI developers will upgrade Mesa when appropriate.

8. Compiling the XFree86/DRI tree

8.1 Make a build tree

Rather than placing object files and library files right in the source tree, they're instead put into a parallel *build* tree. The build tree is made with the `lndir` command:

```
cd ~/DRI-CVS
ln -s xc XFree40
mkdir build
cd build
lndir -silent -ignorelinks ../XFree40
```

The build tree will be populated with symbolic links which point back into the CVS source tree.

Advanced users may have several build trees for compiling and testing with different options.

8.2 Edit the host.def file

The `~/DRI-CVS/build/xc/config/cf/host.def` file is used to configure the XFree86 build process. You can change it to customize your build options or make adjustments for your particular system configuration

The default `host.def` file will look something like this:

```
            #define DefaultCCOptions -Wall
(i386)      #define DefaultGcc2i386Opt -O2
(Alpha)     #define DefaultGcc2AxpOpt -O2 -mcpu=ev6 (or similar)
            #define LibraryCDebugFlags -O2
            #define BuildServersOnly YES
            #define XF86CardDrivers vga tdfx mga ati i810
            #define LinuxDistribution LinuxRedHat
            #define DefaultCCOptions -ansi GccWarningOptions -pipe
            #define BuildXF86DRI YES
            /* Optionally turn these on for debugging */
            /* #define GlxBuiltInTdfx YES */
            /* #define GlxBuiltInMga YES */
            /* #define GlxBuiltInR128 YES */
            /* #define GlxBuiltInRadeon YES */
            /* #define DoLoadableServer NO */
            #define SharedLibFont NO
```

The `ProjectRoot` variable specifies where the XFree86 files will be installed. We recommend installing the DRI files over your existing XFree86 installation - it's generally safe to do and less error-prone. This policy is different than what we used to recommend.

If XFree86 4.x is not installed in /usr/X11R6/ you'll have to add the following to the `host.def` file:

```
#define ProjectRoot pathToYourXFree86installation
```

Note the *XF86CardDrivers* line to be sure your card's driver is listed.

If you want to enable 3DNow! optimizations in Mesa and the DRI drivers, you should add the following:

```
#define MesaUse3DNow YES
```

You don't have to be using an AMD processor in order to enable this option. The DRI will look for 3DNow! support and runtime and only enable it if applicable.

If you want to enable SSE optimizations in Mesa and the DRI drivers, you **must** upgrade to a Linux 2.4.x kernel. Mesa will verify that SSE is supported by both your processor *and* your operating system, but to build Mesa inside the DRI you need to have the Linux 2.4.x kernel headers in /usr/src/linux. If you enable SSE optimizations with an earlier version of the Linux kernel in /usr/src/linux, Mesa **will not compile**. You have been warned. If you do have a 2.4.x kernel, you should add the following:

```
#define MesaUseSSE YES
```

If you want to build the DRM kernel modules as part of the full build process, add the following:

```
#define BuildXF86DRM YES
```

Otherwise, you'll need to build them separately as described below.

8.3 Compilation

To compile the complete DRI tree:

```
cd ~/DRI-CVS/build/xc/
make World >& world.log
```

Or if you want to watch the compilation progress:

```
cd ~/DRI-CVS/build/xc/
make World >& world.log &
tail -f world.log
```

With the default compilation flags it's normal to get a lot of warnings during compilation.

Building will take some time so you may want to go check your email or visit slashdot.

WARNING: do not use the -j option with make. It's reported that it does not work with XFree86/DRI.

8.4 Check for compilation errors

Using your text editor, examine `world.log` for errors by searching for the pattern `***`.

After fixing the errors, run `make World` again. Later, you might just compile parts of the source tree but it's important that the whole tree will build first.

If you edited your host.def file to enable automatic building of the DRI kernel module(s), verify that they were built:

```
cd ~/DRI-CVS/build/xc/programs/Xserver/hw/xfree86/os-support/linux/drm/kernel
ls
```

Otherwise, build them now by running

```
cd ~/DRI-CVS/build/xc/programs/Xserver/hw/xfree86/os-support/linux/drm/kernel
make -f Makefile.linux
```

For the 3dfx Voodoo, you should see *tdfx.o*. For the Matrox G200/G400, you should see *mga.o*. For the ATI Rage 128, you should see *r128.o*. For the ATI Radeon, you should see *radeon.o*. For the Intel i810, you should see *i810.o*.

If the DRI kernel module(s) failed to build you should verify that you're using the right version of the Linux kernel. The most recent kernels are not always supported.

If your build machine is running a different version of the kernel than your target machine (i.e. 2.2.x vs. 2.4.x), make will select the wrong kernel source tree. This can be fixed by explicitly setting the value of `LINUXDIR`. If the path to your kernel source is `/usr/src/linux-2.4.x`,

```
cd ~/DRI-CVS/build/xc/programs/Xserver/hw/xfree86/os-support/linux/drm/kernel
make -f Makefile.linux LINUXDIR=/usr/src/linux-2.4.x
```

or alternatively, edit Makefile.linux to set `LINUXDIR` before the `ifndef LINUXDIR` line.

8.5 DRI kernel module installation

The DRI kernel modules will be in `~/DRI-CVS/build/xc/programs/Xserver/hw/xfree86/os-support/linux/drm/kernel/`.

To load the appropriate DRM module in your running kernel you can either use ismod and restart your X server or copy the kernel module to `/lib/modules/2.4.x/kernel/drivers/char/drm/` then run depmod and restart your X server.

Make sure you first unload any older DRI kernel modules that might be already loaded.

Note that some DRM modules require that the `agpgart` module be loaded first.

9. Normal Installation and Configuration

Most users will want to install the new X server and use it in place of their old X server. This section explains how to do that.

Developers, on the other hand, may just want to test the X server without actually installing it as their default server. If you want to do that, skip to the next section.

9.1 Installation

Here are the installation commands:

```
su
cd ~/DRI-CVS/build/xc
make install
```

9.2 Update the XF86Config File

You may need to edit your XF86Config file to enable the DRI. The config file is usually installed as `/etc/X11/XF86Config-4`. See the DRI User Guide for details, but basically, you need to load the "glx" and "dri" modules and add a "DRI" section.

On the DRI web site, in the resources section, you'll find example XF86Config files for a number of graphics cards. These configuration files also setup DRI options so it's highly recommended that you look at these examples.

The XFree86 4.x server can generate a basic configuration file itself. Simply do this:

```
cd /usr/X11R6/bin
./XFree86 -configure
```

A file named `/root/XF86Config.new` will be created. It should allow you to try your X server but you'll almost certainly have to edit it. For example, you should add `HorizSync` and `VertRefresh` options to the `Monitor` section and `Modes` options to the `Screen` section. Also, the `ModulePath` option in the `Files` section should be set to `/usr/X11R6/lib/modules`.

9.3 Start the New X Server

The new X server should be ready to use now. Start your X server in your usual manner. Often times the `startx` command is used:

```
startx
```

10. Testing the Server Without Installing It

As mentioned at the start of section 9, developers may want to simply run the X server without installing it. This can save some time and allow you to keep a number of X servers available for testing.

10.1 Configuration

As described in the preceding section, you'll need to create a configuration file for the new server. Put the `XF86Config` file in your `~/DRI-CVS/build/xc/programs/Xserver` directory.

Be sure the `ModulePath` option in your XF86Config file is set correctly.

10.2 A Startup Script

A simple shell script can be used to start the X server. Here's an example.

```
#!/bin/sh
export DISPLAY=:0
./XFree86 -xf86config XF86Config & \
sleep 2
fvwm2 &
xset b off
xmodmap -e "clear mod4"
xsetroot -solid "#00306f"
xterm -geometry 80x40+0+0
```

You might name this script `start-dri`. Put it in your `~/DRI-CVS/build/xc/programs/Xserver` directory.

To test the server run the script:

```
cd ~/DRI-CVS/build/xc/programs/Xserver
./start-dri
```

For debugging, you may also want to capture the log messages printed by the server in a file. If you're using the C-shell:

```
./start-dri >& log
```

11. Where To Go From Here

At this point your X server should be up and running with hardware-accelerated direct rendering. Please read the DRI User Guide for information about trouble shooting and how to use the DRI-enabled X server for 3D applications.

CONTENTS

1. Preamble 1
   1.1 Copyright 1
   1.2 Trademarks 1
2. Introduction 1
3. Prerequisites 1
4. Linux Kernel Preparation 2
5. CPU Architectures 3
   5.1 Intel Pentium III Features 4
   5.2 AMD 3DNow! Features 4
   5.3 Alpha Features 4
6. Downloading the XFree86/DRI CVS Sources 4
   6.1 Anonymous CVS download: 4
   6.2 Registered CVS download: 5
   6.3 Updating your CVS sources 5
7. Mesa 5
8. Compiling the XFree86/DRI tree 5
   8.1 Make a build tree 5
   8.2 Edit the host.def file 6
   8.3 Compilation 7
   8.4 Check for compilation errors 7
   8.5 DRI kernel module installation 8
9. Normal Installation and Configuration 8
   9.1 Installation 8
   9.2 Update the XF86Config File 8
   9.3 Start the New X Server 8
10. Testing the Server Without Installing It 9
    10.1 Configuration 9
    10.2 A Startup Script 9
11. Where To Go From Here 9

$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/DRIcomp.sgml,v 1.21 2004/12/08 22:27:11 dawes Exp $