

Mini how-to Debian 3.1 Sarge nVidia

Author: André Felipe Machado<andremachado@techforce.com.br>

Instructions for getting top 3D performance from your nVidia graphics card on your Debian system.

Easy compiling and creating your tuned deb package.

This min how-to is an improvement, with more instructions and hits for easier compilation, of the Debian documento that you will find at your Debian machine at `/usr/share/doc/nvidia-kernel-source/README.Debian`

There are a few additional steps and hints, basically on auto-apt instructions.

By following this document you DO NOT NEED, nor should, download and install the unified driver from nVidia site. That unified driver build and install outside from apt and put files at different places. This method here is more clean and suitable for Debian based systems using apt.

You will have a top performance 3D driver package, built and tuned at your machine.

This document does not yet leverage the new module-assistant package, created for this kind of task.

[nVidia 3D driver for Debian](#)[ATI Linux driver packages for Debian](#) **nvidia-graphics-drivers for Debian**

INTRODUCTION:

Welcome to Debian NVIDIA packages. This document contains notes on the kernel module interface for the NVIDIA driver offering.

The source package nvidia-graphics-drivers generates nvidia-kernel-source (which will need further compilation to generate specific kernel modules), nvidia-glx, nvidia-glx-dev, nvidia-glx-ia32.

VERIFY IF THE ALREADY COMPILED PACKAGES ARE ENOUGH:

At the Debian GNU / Linux

contrib and non-free repositories there are pre-compiled packages for some of the versions (stable, testing, unstable). They are not the newest available drivers from nVidia, but **can** be enough for your kernel and video board.

Execute `dpkg-query -f='${Package} ${Version} ${Architecture}\n'` on the command line. If the main server is unavailable, try [this](#) server.

If the ready made packages are enough, you can configure your sources.list file and install through apt-get or Synaptic.

If you want the newest available driver for newest kernel, or if there is not a driver package for your kernel version, continue reading this tutorial.

IF YOU HAVE A RECENT BOARD

The nVidia unified driver versions at official repositories MAY support your card. Give them a try first.

If unsupported, weird behaviour, poor performance, you will need to compile a recent version yourself. Keep reading.

IF YOU ARE USING DEBIAN STABLE

If the pre compiled drivers at official repositories are not freshen enough for your video card, use

the Backports repository. See the highlighted article about sources.list and preferences files.

IF YOU ARE USING DEBIAN STABLE WITH KERNEL 2.4.X

The Debian source package generate binaries ready to run on kernel 2.6.x.

In order to use nvidia drivers with kernel 2.4.x you **MUST** disable Thread Local Storage (TLS) from nVidia libraries **AFTER** the instalation and then restart the X server.

You **MUST** have to use the original kernel libraries or weird errors will happen during video board operation.

Coulored squares, duplicated / splited images, poor performance, lag, deturpated OpenGL features and other weird errors for each chipset.

As the **LAST** installation step, before you use the new video driver, you will execute, logged as root, the command:

```
dpkg-reconfigure --priority=low nvidia-glx
```

In order to disable TLS at each reboot.

Normally, the /etc/init.d/nvidia-glx script should detect the kernel version and disable / enable TLS correctly, but sometimes it could fail.

The command will edit the /etc/default/nvidia-glx file and enable

USE_TLS=0

For hard confirmation, you could execute, logged as root, the command:

```
/etc/init.d/nvidia-glx restart
```

and then restart X server. (Exit the session and use gdm/kdm/xdm or ctrl+alt+backspace).

KERNEL MODULE INSTRUCTIONS:

There are TWO ways to build the nvidia-kernel package. Which one depends on your situation with your kernel.

1. METHOD #1: You are running a Debian supplied kernel or built a kernel-headers package along with your own self-built kernel.
2. METHOD #2: You are running your own self-built kernel built from kernel source.
3. METHOD #3: use the new module-assistant procedure. It is the easiest.

Which method you choose really depends on what kernel headers you wish to use, those from a kernel-headers package or those from kernel

source (from which you built your own kernel).

As mentioned above, if you are running a Debian supplied kernel you will probably want to choose METHOD #1. If you compile your own kernels, METHOD #2.

PRELIMINARY:

- Decide where you want to build your module. By default it will build under /usr/src like other kernel modules but you may choose to build it under your home directory as some people prefer and writing to /usr violates the FHS.

SUPPORT FOR 2.6 KERNELS:

As of 1.0.5336-1, NVIDIA includes support for a 2.6 kernel. No extra steps are required.

HOTPLUG SUPPORT:

As of version 1.0.5336-8 there is hotplug module loading support provided in patches that are applied by default but by default, hotplug is set to

ignore PCI/AGP cards for display.

To change this set IGNORE_PCI_CLASS_DISPLAY to false in /etc/default/hotplug or run

```
dpkg-reconfigure -plow hotplug
```

USE THE AUTO-APT ENVIRONMENT FOR EASIER BUILDING

The auto-apt building environment solve the time waste of fixing broken dependencies that are show stopers during compilation.

- configure your apt repositories.
- apt-get update
- apt-get upgrade
- apt-get install auto-apt
- auto-apt update

METHOD #1 Using a kernel-headers package

For example: 2.4.21-4-k7

3. Download and install package: kernel-headers-2.4.21-4-k7

It will install in /usr/src/

Make sure your kernel image and headers have matching release numbers to avoid possible problems in packages built from different sources.

4. Set some environment variables (if bash is your shell):

```
export KSRC=/usr/src/kernel-headers-2.4.21-4-k7
```

```
export KVERS=2.4.21-4-k7
```

5. Install nvidia-kernel-common:

If not installed already

```
apt-get install nvidia-kernel-common
```

6. Then build nvidia-kernel package:

6. Então compile seu pacote nvidia-kernel :

- `cd [YOUR BUILD LOCATION]/modules/nvidia-kernel`
- `auto-apt run debian/rules binary_modules`
 - it is likely that some files will be missing, and auto-apt will stop compilation, download the packages containing them, install them, and resume the compilation process.
- some packages may need configuration during installation through apt. Proceed as usual, answering the configuration questions and auto-apt will resume the compilation at the exact point it stopped.
- some files may be supplied by more than one package and auto-apt will ask WHICH ONE to install. Usually, the most broader one is pointed as the first option and should solve the issue.

(You can also combine step 4 and 5 into one line:

```
KSRC=/usr/src/kernel-headers-2.4.21-4-k7 KVERS=2.4.21-4-k7 debian/rules binary_modules)
```

7. Install the nvidia-kernel package:

```
dpkg -i ../nvidia-kernel-2.4.21-4-k7_1.0.7174-1+_Custom_i386.deb
```

(or similar filename)

8. Now see GENERAL NOTES below method #2

METHOD #2: Using your own kernel source headers

To build the nvidia-kernel deb you need to first make sure you have kernel-package installed, then do the following:

As root

1. `cd /usr/src`

```
tar xzvf nvidia-kernel-source.tar.gz -C [YOUR BUILD LOCATION]
```

(It will install in [YOU BUILD LOCATION]/modules)

- or -

```
tar xzvf nvidia-kernel-source.tar.gz (if building in /usr/src)
```

2. If you are NOT using the default modules location `/usr/src/modules` then you must set an environment variable that points to your modules location.

```
export MODULE_LOC=[YOUR BUILD LOCATION]/modules
```

This is needed by make-kpkg which is used later.

If [YOUR BUILD LOCATION] is /usr/src there is no need to set this.

3. Build the modules under MODULE_LOC i.e. [YOU BUILD LOCATION]/modules

```
cd linux (or your kernel source directory)
```

```
make-kpkg modules_image
```

4. Install nvidia-kernel-common:

```
apt-get install nvidia-kernel-common
```

5. Install the nvidia-kernel package:

```
cd [YOUR BUILD LOCATION] (e.g. /usr/src)
```

```
dpkg -i nvidia-kernel-KVER*.deb
```

Notes for method #2:

- It is advised not to clean the kernel source tree between "make-kpkg kernel_image" and "make-kpkg modules_image".

- The Riva framebuffer is known to conflict with the nvidia X driver. If you are using X it would be wise not to compile it in. The vesa framebuffer is known to work in some cases, and not in others. (people have recently been having problems with the vesa driver as well)

METHOD #3: use the module-assistant

This is, today, the easiest method.

Logged as root, execute the commands:

1. apt-get install module-assistant nvidia-kernel-common
2. m-a prepare
3. m-a auto-install nvidia

GENERAL NOTES:

Also you must add any users who wish to use OpenGL applications to the group video. You can do this with:

```
adduser username video
```

Also note for AGP issues and further information please see the NVIDIA README file (README.gz) in the nvidia-glx package

An old nvidia.o might be already loaded (run 'lsmod' to check) so do 'rmmod nvidia.o' to remove the module from memory.

For any news on this package check <http://people.debian.org/~rdonald> and <http://bugs.debian.org/nvidia-kernel-source>

-- Randall Donald [rdonald@debian.org]

[Troubleshooting Debian nVidia drivers installation](#)[Debian nVidia installation \(Andrew E. Schulman\)](#)[Instalação driver nVidia no Debian BR-CDD](#)