# CUDA Tutorial 01 - Linux

## Getting up and running under Linux

Martin A. Kruse - Februrary 2014

This document is part of a series of tutorials intended for the people at the extraterrestrial physics research group at the Christian Albrechts Universität zu Kiel. Everyone else is invited to use and distribute (as is) these documents, however the computers mentioned here are not available to the public.

The full series of tutorials can be found online at
http://www.ieap.uni-kiel.de/et/people/kruse/,
though access to this site later than 2026 might
be impossible due to thermonuclear war.

Suggestions and corrections are very welcome at kruse@physik.uni-kiel.de.

This tutorial shows you which software to install on your PC in order to start developing CUDA software. The installation process has been tested with Ubuntu 12.04 64 bit desktop edition and Linux Mint 16 (Petra). All necessary packages are installed on Prometheus, so if you plan on working that computer you may skip this tutorial, log on to Prometheus and get going with programming.

## 1 Getting CUDA 6.0 RC up and running

Not thirty seconds after I finished writing this tutorial, NVIDIA sent an email announcing the release candidate (RC) of CUDA 6.0. As I found out that CUDA 6.0 brought a significant improvement for beginners over CUDA 5.5, I rewrote the installation process to include the updated version, which is also simpler than installation of CUDA 5.5. The old installation process can be found below under section 2. Try the procedure described in this section first and go to section 2 if you encounter any problems.

### *1.1 Get NVIDIA drivers and CUDA toolkit*

Go to NVIDIAs driver webpage (http://www.geforce.com/drivers) and get the newest available driver .run-file (let's assume you downloaded 331.41, otherwise adjust the version number where appropriate). At the time of this writing, there is no official release of CUDA 6.0 available, though NVIDIA has released a release candidate (RC) of the upcoming iteration. Most probably by the time you read this, there will be a stable official release, until then you need to get the package by registering for a developer account or, more easily, by downloading the file

```
/data/etph/kruse/tutorials/cuda/cuda_6.0.26_rc_linux64.run
```

### *1.2 Installing NVIDIA driver and CUDA toolkit*

Now thing become interesting. A piece of advise: Depending on distribution, after this step you might not be able to start your window manager when you reboot before installing the new driver version, so print this tutorial or open with a different computer so you know what to do when all that fancy windows are gone. Do not worry, nothing is lost when something gows awry, just ask the system administrator of your choice (or me), but I do not think that this will be necessary.

Hit <CTRL> + <ALT> + <F1> to open a terminal.

Get rid of possibly installed older NVIDIA drivers:

```
sudo apt-get purge nvidia*
```

Navigate to the downloaded files from step 1.1 and make them executable (adjust filenames where necessary):

```
sudo chmod +x NVIDIA-Linux-x86_64-331.41.run
sudo chmod +x cuda_5.5.22_linux_64.run
```

#### 1.2.1 Install NVIDIA driver

Shutdown your window manager by typing

```
sudo service lightdm stop
```

Substitute lightdm in the line above by your window manager. If lightdm doesn't do the trick, try

```
sudo service mdm stop
sudo service gdm stop
```

If that also achieves nothing, ask google what window manager your distribution is running.

Install the driver:

```
sudo ./NVIDIA-Linux-x86_64-331.41.run
```

Follow the instructions given by the installer, use default values if you are not sure what it wants from you, except when it asks to overwrite X configuration file. You want that to happen so the NVIDIA driver starts automatically, so chosse 'yes' in that situation.

### 1.2.2 Install CUDA toolkit

Install the CUDA toolkit via

```
sudo ./cuda_6.0.26_rc_linux_64.run
```

The installer will ask you several things. Choose as follows:

(if not running Ubuntu)Install on unsupported device -> well, at least try... Install NVIDIA Accelerated Graphics Driver.... -> no! you just installed a newer version (but be sure you really did, otherwise ugly runtime errors are sure to be experienced)
Install CUDA toolkit -> yes, of course, that's why we are here
Enter toolkit Location -> hit enter, the default value is ok
Do you want to install a symbolic link... ->yes, why not, but I don't think that you will need that
Install the CUDA samples -> yes, we want to play around with them
Enter CUDA samples location -> anything is fine, I chose /usr/local/cuda-6.0/samples

Now the installer will do its thing. There will be a warning due to partially unfinished installation. That is because we chose to not install the driver coming with the CUDA package but the newer one, so ignore this warning.

You may now restart the window manager

```
sudo service lightdm start
```

If you encounter problems in this step, take a look at section 2.4, maybe something there might help you out.

## *1.3 Adjust environment variables*

You now have to tell your system where to find the binaries and libraries of the toolkit. To do so, open/create .bashrc file (in your home folder by typing 'gedit ~/.bashrc') and append the following line:

```
export PATH=/usr/local/cuda-6.0/bin:${PATH}
```

Save and exit.

Then edit /etc/ld.so.conf (type 'sudo gedit /etc/ld.so.conf') and append the following two lines:

```
/usr/local/cuda-6.0/lib64
/usr/local/cuda-6.0/lib
```

Save and exit. Type in a terminal

```
sudo ldconfig
```

Now reboot your system.

### *1.4 Testing your system*

If everything went according to plan, you have now everything installed and setup to start using CUDA capabilities in your own programs. Let's test this hypothesis.

Navigate to the folder where you installed the samples and make all that stuff, so in my case, where the samples were stored as mentioned above, this would look like this:

```
cd /usr/local/cuda-6.0/samples/
sudo make
```

This will take a couple of minutes, so grab a coffe or do stuff. When that process is finished, go to the bin folder

```
cd /usr/local/cuda-6.0/samples/bin/x86\_64/linux/release/
```

and have some fun. Try what you want, but for our purposes here, deviceQuery is a nice choice because it will confirm that everything went fine during installation and additionaly shows some CUDA capabilities of your graphics card(s). So type

```
./deviceQuery
```

That's it. Good luck programming nice and fast CUDA programs. If you are eager to get your first program running, go on to tutorial 02.

## 2 Getting CUDA 5.5 up and running

This is the process for the old CUDA toolkit. Please try installing a newer version first (section 1) and find solutions to possible problems during that process here.

### *2.1 Get NVIDIA drivers and CUDA toolkit*

Go to NVIDIAs driver webpage (http://www.geforce.com/drivers) and get the newest available driver .run-file (331.38 at the time of this writing). Navigate to the CUDA developer homepage (https://developer.nvidia.com/cuda-downloads) and get the newest CUDA toolkit (5.5 / Ubuntu 12.10, also a .run-file, not the deb-package).

### *2.2 Install dependencies for driver and toolkit*

Get them by typing

```
sudo apt-get install freeglut3 freeglut3-dev build-essential libx11-dev libxmu-dev libxi-dev
    libgl1-mesa-glx libglu1-mesa libglu1-mesa-dev gcc g++ gcc-4.6 g++-4.6 linux-headers-generic
    linux-source
sudo ln -s /usr/lib/x86_64-linux-gnu/libglut.so.3 /usr/lib/libglut.so
```

### *2.3 Setup alternatives for gcc*

For some reason unknown to me, the CUDA toolkit needs a gcc version older than 4.7.3 to compile, so you have to adjust your system to support an older version that you probably have (let's assume you have 4.8 for the following commands, adjust accordingly). You already installed version 4.6 in the previous step, so now install the alternative by typing

```
sudo update-alternatives --remove-all gcc
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.6 10
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.8 50
```

*2.4 Installing NVIDIA driver and CUDA toolkit*

Now thing become interesting. A piece of advise: Depending on distribution, after this step you might not be able to start your window manager when you reboot before installing the new driver version, so print this tutorial or open with a different computer so you know what to do when all that fancy windows are gone. Do not worry, nothing is lost when something gows awry, just ask the system administrator of your choice (or me), but I do not think that this will be necessary.

Hit <CTRL> + <ALT> + <F1> to open a terminal.

Get rid of possibly installed older NVIDIA drivers:

```
sudo apt-get purge nvidia*
```

Navigate to the downloaded files from step 2.1 and make them executable (adjust filenames where necessary):

```
sudo chmod +x NVIDIA-Linux-x86_64-331.38.run
sudo chmod +x cuda_5.5.22_linux_64.run
```

### 2.4.1 Install NVIDIA driver

Use the newer version (4.8) of gcc by typing

```
sudo update-alternatives --config gcc
```

and choosing the appropiate value. Shutdown your window manager by typing

```
sudo service shutdown lightdm stop
```

Substitute lightdm in the line above by your window manager. If lightdm doesn't do the trick, try

```
sudo service shutdown mdm stop
sudo service shutdown gdm stop
```

If that also achieves nothing, ask google what window manager your distribution is running.

Now install the driver:

```
sudo ./NVIDIA-Linux-x86_64-331.38.run
```

Follow the instructions given by the installer, use default values if you are not sure what it wants from you, except when it asks to overwrite X configuration file. You want that to happen so the NVIDIA driver starts automatically, so chosse 'yes' in that situation.

### 2.4.2 Install CUDA toolkit

Switch to the old version (4.6) of gcc by again typing

```
sudo update-alternatives --config gcc
```

Install the CUDA toolkit via

```
sudo ./cuda_5.5.22_linux_64.run
```

The installer will ask you several things. Choose as follows:

Install NVIDIA Accelerated Graphics Driver.... -> no! you just installed a newer version
Install CUDA toolkit -> yes, of course, that's why we are here
Enter toolkit Location -> hit enter, the default value is ok
Install the CUDA samples -> yes, we want to play around with them
Enter CUDA samples location -> anything is fine, I chose /usr/local/cuda-5.5/samples

Now the installer will do its thing. There will be a warning due to partially unfinished installation. That is because we chose to not install the driver coming with the CUDA package but the newer one, so ignore this warning.

*2.5 Adjust environment variables*

You now have to tell your system where to find the binaries and libraries of the toolkit. To do so, open/create .bashrc file (in your home folder by typing 'gedit ~/.bashrc') and append the following line:

```
export PATH=/usr/local/cuda-5.5/bin:${PATH}
```

Save and exit.

Then edit /etc/ld.so.conf (type 'sudo gedit /etc/ld.so.conf') and append the following two lines:

```
/usr/local/cuda-5.5/lib64
/usr/local/cuda-5.5/lib
```

Save and exit. Type in a terminal

```
sudo ldconfig
```

Now reboot your system (and pray or sacrifice a goat, if you think that might help).

*2.6 Testing your system*

If everything went according to plan, you have now everything installed and setup to start using CUDA capabilities in your own programs. Let's test this hypothesis.

Navigate to the folder where you installed the samples and make all that stuff, so in my case, where the samples were stored as mentioned above, this would look like this:

```
cd /usr/local/cuda-5.5/samples/
sudo make
```

This will take a couple of minutes, so grab a coffe or do stuff. When that process is finished, go to the bin folder

```
cd /usr/local/cuda-5.5/samples/bin/x86\_64/linux/release/
```

and have some fun. Try what you want, but for our purposes here, deviceQuery is a nice choice because it will confirm that everything went fine during installation and additionaly shows some CUDA capabilities of your graphics card(s). So type

```
./deviceQuery
```

That's it. Good luck programming nice and fast CUDA programs. If you are eager to get your first program running, go on to tutorial 02.