CUDA Quick Start Guide

DU-05347-301_v11.1 | October 2020

Table of Contents

Chapter 1. Introduction.... 1

Chapter 2. Windows....2

2.1. Network Installer.... 2

2.2. Local Installer.... 4

Chapter 3. Linux.... 6

3.1. Linux x86_64....6

3.1.1. Redhat / CentOS....6

3.1.1.1. RPM Installer.... 6

3.1.1.2. Runfile Installer.... 7

3.1.2. Fedora.... 7

3.1.2.1. RPM Installer.... 8

3.1.2.2. Runfile Installer.... 8

3.1.3. SUSE Linux Enterprise Server....9

3.1.3.1. RPM Installer.... 9

3.1.3.2. Runfile Installer.... 9

3.1.4. OpenSUSE.... 10

3.1.4.1. RPM Installer.... 10

3.1.4.2. Runfile Installer.... 10

3.1.5. WSL.... 11

3.1.6. Ubuntu....11

3.1.6.1. Debian Installer....12

3.1.6.2. Runfile Installer.... 12

3.1.7. Debian.... 13

3.1.7.1. Debian Installer....13

3.1.7.2. Runfile Installer.... 13

3.2. Linux POWER8....14

3.2.1. Ubuntu....14

3.2.1.1. Debian Installer....14

3.2.2. Redhat / CentOS....15

3.2.2.1. RPM Installer.... 15

Chapter 1. Introduction

This guide covers the basic instructions needed to install CUDA and verify that a CUDA application can run on each supported platform.

These instructions are intended to be used on a clean installation of a supported platform. For questions which are not answered in this document, please refer to the Windows Installation Guide and Linux Installation Guide.

The CUDA installation packages can be found on the CUDA Downloads Page.

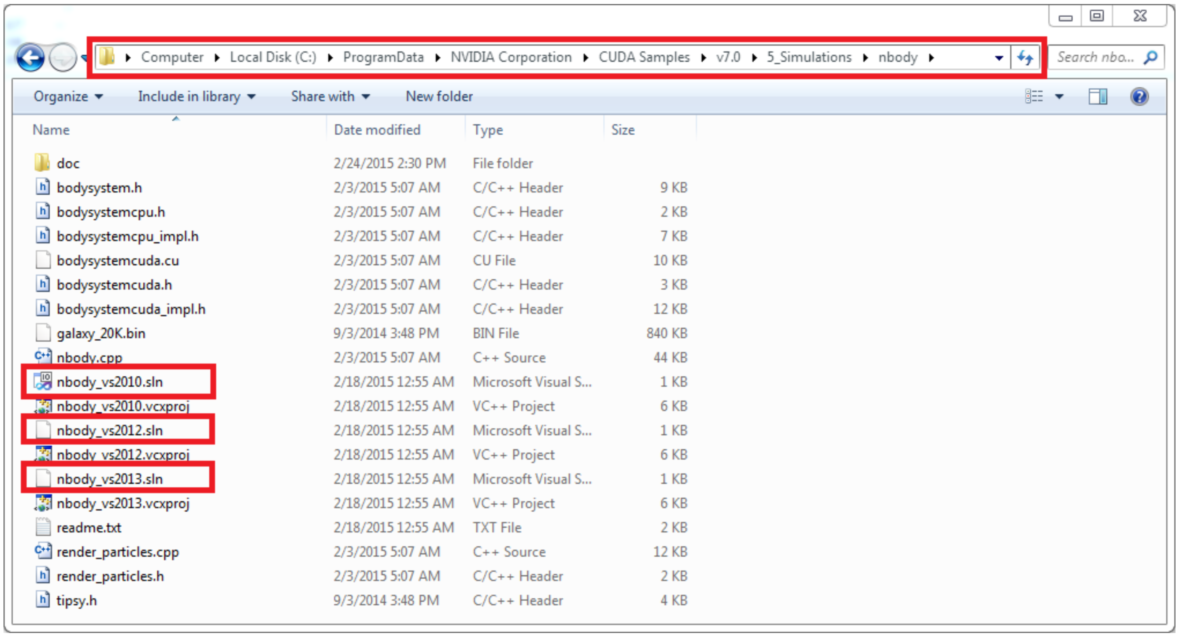
Chapter 2. Windows

When installing CUDA on Windows, you can choose between the Network Installer and the Local Installer. The Network Installer allows you to download only the files you need. The Local Installer is a stand-alone installer with a large initial download. For more details, refer to the Windows Installation Guide.

2.1. Network Installer

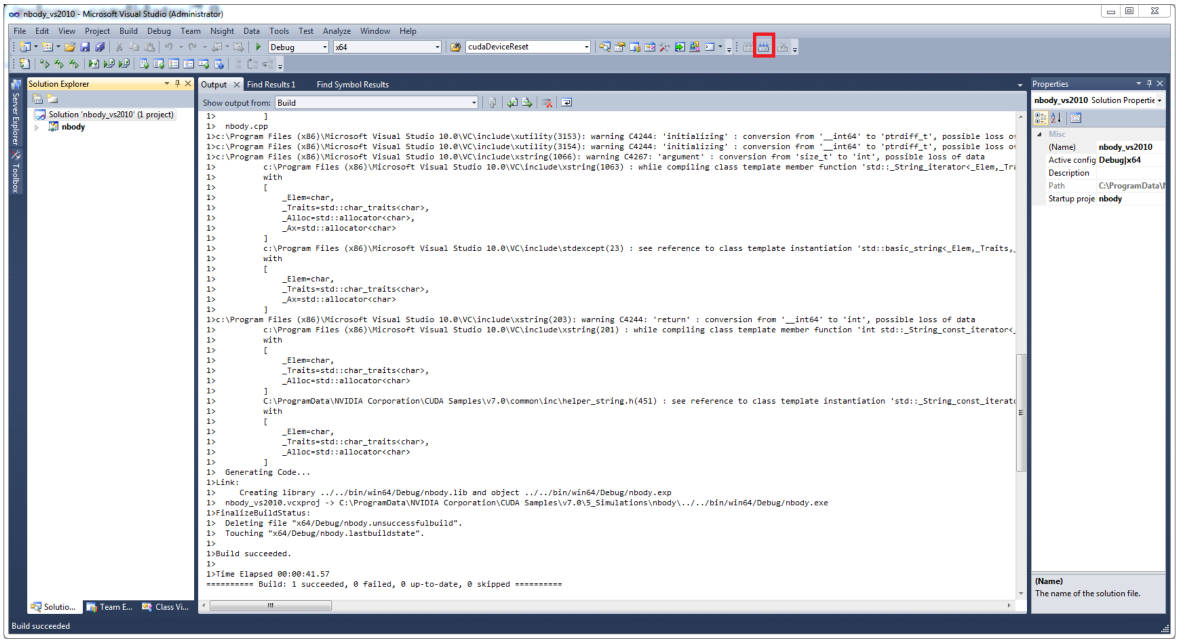
Perform the following steps to install CUDA and verify the installation.

1. Launch the downloaded installer package.
2. Read and accept the EULA.
3. Select "next" to download and install all components.
4. Once the download completes, the installation will begin automatically.
5. Once the installation completes, click "next" to acknowledge the Nsight Visual Studio Edition installation summary.
6. Click "close" to close the installer.
7. Navigate to the CUDA Samples' nbody directory.
8. Open the nbody Visual Studio solution file for the version of Visual Studio you have installed.


Computer ▸ Local Disk (C:) ▸ ProgramData ▸ NVIDIA Corporation ▸ CUDA Samples ▸ v7.0 ▸ 5_Simulations ▸ nbody ▸
Search nbo...
Organize
Include in library
Share with
New folder

Name	Date modified	Type	Size
doc	2/24/2015 2:30 PM	File folder	
bodysystem.h	2/3/2015 5:07 AM	C/C++ Header	9 KB
bodysystemcpu.h	2/3/2015 5:07 AM	C/C++ Header	2 KB
bodysystemcpu_impl.h	2/3/2015 5:07 AM	C/C++ Header	7 KB
bodysystemcuda.cu	2/3/2015 5:07 AM	CU File	10 KB
bodysystemcuda.h	2/3/2015 5:07 AM	C/C++ Header	3 KB
bodysystemcuda_impl.h	2/3/2015 5:07 AM	C/C++ Header	12 KB
galaxy_20K.bin	9/3/2014 3:48 PM	BIN File	840 KB
nbody.cpp	2/3/2015 5:07 AM	C++ Source	44 KB
nbody_vs2010.sln	2/18/2015 12:55 AM	Microsoft Visual S...	1 KB
nbody_vs2010.vcxproj	2/18/2015 12:55 AM	VC++ Project	6 KB
nbody_vs2012.sln	2/18/2015 12:55 AM	Microsoft Visual S...	1 KB
nbody_vs2012.vcxproj	2/18/2015 12:55 AM	VC++ Project	6 KB
nbody_vs2013.sln	2/18/2015 12:55 AM	Microsoft Visual S...	1 KB
nbody_vs2013.vcxproj	2/18/2015 12:55 AM	VC++ Project	6 KB
readme.txt	2/18/2015 12:55 AM	TXT File	2 KB
render_particles.cpp	2/3/2015 5:07 AM	C++ Source	12 KB
render_particles.h	2/3/2015 5:07 AM	C/C++ Header	2 KB
tipsy.h	9/3/2014 3:48 PM	C/C++ Header	4 KB

9. Open the "Build" menu within Visual Studio and click "Build Solution".

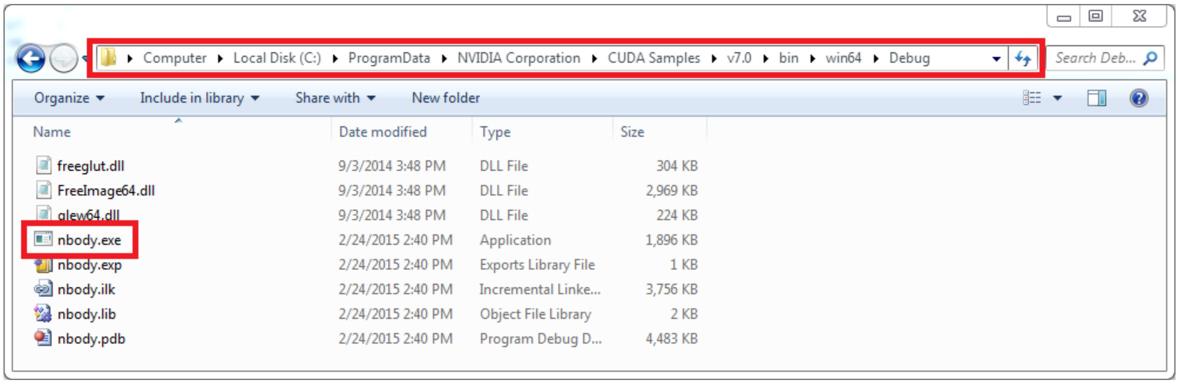


(Name) nbody_vs2010
Active config Debug|x64
Description
Path C:\ProgramData\
Startup proje nbody

(Name)
The name of the solution file.

```
1>          ]
1>  nbody.cpp
1>c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\xutility(3153): warning C4244: 'initializing' : conversion from '__int64' to 'ptrdiff_t', possible loss of
1>c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\xutility(3154): warning C4244: 'initializing' : conversion from '__int64' to 'ptrdiff_t', possible loss of
1>c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\xstring(1066): warning C4267: 'argument' : conversion from 'size_t' to 'int', possible loss of data
1>          c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\xstring(1063) : while compiling class template member function 'std::_String_iterator<_Elem,_Tra
1>          with
1>          [
1>              _Elem=char,
1>              _Traits=std::char_traits<char>,
1>              _Alloc=std::allocator<char>,
1>              _Ax=std::allocator<char>
1>          ]
1>          c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\stdexcept(23) : see reference to class template instantiation 'std::basic_string<_Elem,_Traits,_
1>          with
1>          [
1>              _Elem=char,
1>              _Traits=std::char_traits<char>,
1>              _Ax=std::allocator<char>
1>          ]
1>c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\xstring(203): warning C4244: 'return' : conversion from '__int64' to 'int', possible loss of data
1>          c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\xstring(201) : while compiling class template member function 'int std::_String_const_iterator<_
1>          with
1>          [
1>              _Elem=char,
1>              _Traits=std::char_traits<char>,
1>              _Alloc=std::allocator<char>
1>          ]
1>          C:\ProgramData\NVIDIA Corporation\CUDA Samples\v7.0\common\inc\helper_string.h(451) : see reference to class template instantiation 'std::_String_const_iterat
1>          with
1>          [
1>              _Elem=char,
1>              _Traits=std::char_traits<char>,
1>              _Alloc=std::allocator<char>
1>          ]
1>  Generating Code...
1>Link:
1>     Creating library ../../bin/win64/Debug/nbody.lib and object ../../bin/win64/Debug/nbody.exp
1>  nbody_vs2010.vcxproj -> C:\ProgramData\NVIDIA Corporation\CUDA Samples\v7.0\5_Simulations\nbody\../../bin/win64/Debug/nbody.exe
1>FinalizeBuildStatus:
1>  Deleting file "x64/Debug/nbody.unsuccessfulbuild".
1>  Touching "x64/Debug/nbody.lastbuildstate".
1>
1>Build succeeded.
1>
1>Time Elapsed 00:00:41.57
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

10. Navigate to the CUDA Samples' build directory and run the nbody sample.



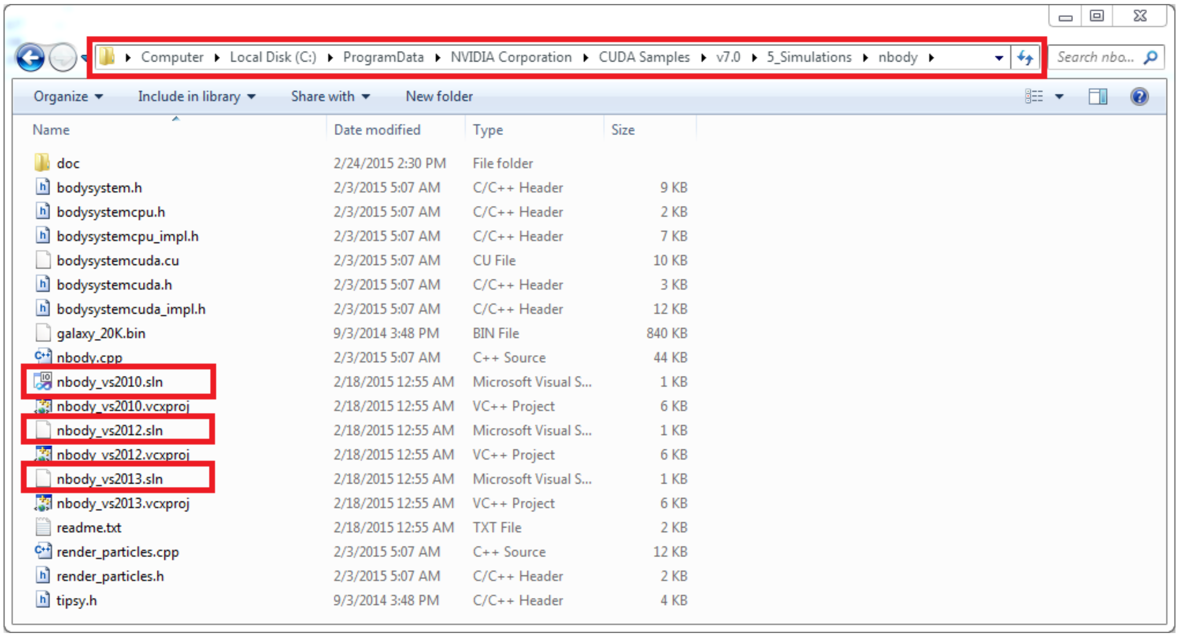
Computer ▸ Local Disk (C:) ▸ ProgramData ▸ NVIDIA Corporation ▸ CUDA Samples ▸ v7.0 ▸ bin ▸ win64 ▸ Debug

Name	Date modified	Type	Size
freeglut.dll	9/3/2014 3:48 PM	DLL File	304 KB
FreeImage64.dll	9/3/2014 3:48 PM	DLL File	2,969 KB
glew64.dll	9/3/2014 3:48 PM	DLL File	224 KB
nbody.exe	2/24/2015 2:40 PM	Application	1,896 KB
nbody.exp	2/24/2015 2:40 PM	Exports Library File	1 KB
nbody.ilk	2/24/2015 2:40 PM	Incremental Linke...	3,756 KB
nbody.lib	2/24/2015 2:40 PM	Object File Library	2 KB
nbody.pdb	2/24/2015 2:40 PM	Program Debug D...	4,483 KB

2.2. Local Installer

Perform the following steps to install CUDA and verify the installation.

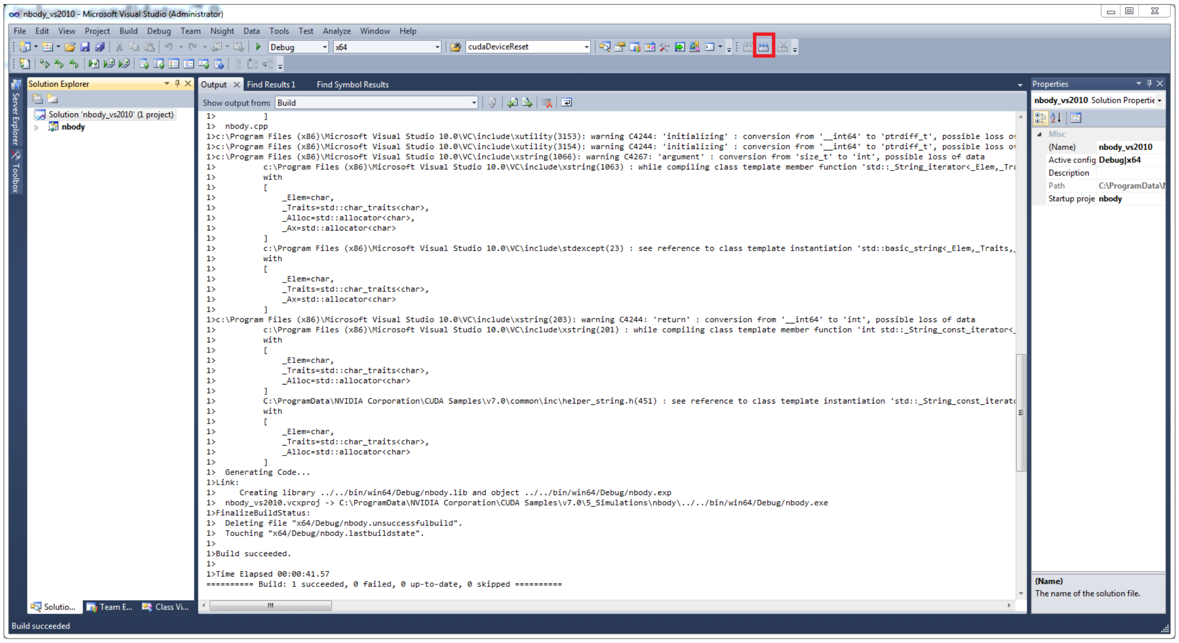
1. Launch the downloaded installer package.
2. Read and accept the EULA.
3. Select "next" to install all components.
4. Once the installation completes, click "next" to acknowledge the Nsight Visual Studio Edition installation summary.
5. Click "close" to close the installer.
6. Navigate to the CUDA Samples' nbody directory.
7. Open the nbody Visual Studio solution file for the version of Visual Studio you have installed.



Computer ▸ Local Disk (C:) ▸ ProgramData ▸ NVIDIA Corporation ▸ CUDA Samples ▸ v7.0 ▸ 5_Simulations ▸ nbody ▸

Name	Date modified	Type	Size
doc	2/24/2015 2:30 PM	File folder	
bodysystem.h	2/3/2015 5:07 AM	C/C++ Header	9 KB
bodysystemcpu.h	2/3/2015 5:07 AM	C/C++ Header	2 KB
bodysystemcpu_impl.h	2/3/2015 5:07 AM	C/C++ Header	7 KB
bodysystemcuda.cu	2/3/2015 5:07 AM	CU File	10 KB
bodysystemcuda.h	2/3/2015 5:07 AM	C/C++ Header	3 KB
bodysystemcuda_impl.h	2/3/2015 5:07 AM	C/C++ Header	12 KB
galaxy_20K.bin	9/3/2014 3:48 PM	BIN File	840 KB
nbody.cpp	2/3/2015 5:07 AM	C++ Source	44 KB
nbody_vs2010.sln	2/18/2015 12:55 AM	Microsoft Visual S...	1 KB
nbody_vs2010.vcxproj	2/18/2015 12:55 AM	VC++ Project	6 KB
nbody_vs2012.sln	2/18/2015 12:55 AM	Microsoft Visual S...	1 KB
nbody_vs2012.vcxproj	2/18/2015 12:55 AM	VC++ Project	6 KB
nbody_vs2013.sln	2/18/2015 12:55 AM	Microsoft Visual S...	1 KB
nbody_vs2013.vcxproj	2/18/2015 12:55 AM	VC++ Project	6 KB
readme.txt	2/18/2015 12:55 AM	TXT File	2 KB
render_particles.cpp	2/3/2015 5:07 AM	C++ Source	12 KB
render_particles.h	2/3/2015 5:07 AM	C/C++ Header	2 KB
tipsy.h	9/3/2014 3:48 PM	C/C++ Header	4 KB

8. Open the "Build" menu within Visual Studio and click "Build Solution".


nbody_vs2010 - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Nsight Data Tools Test Analyze Window Help
Debug
x64
cudaDeviceReset
Solution Explorer
Solution 'nbody_vs2010' (1 project)
nbody
Server Explorer
Toolbox
Output
Find Results 1
Find Symbol Results
Solutio...
Team E...
Class Vi...
Properties
nbody_vs2010 Solution Properties
Misc

Show output from: Build

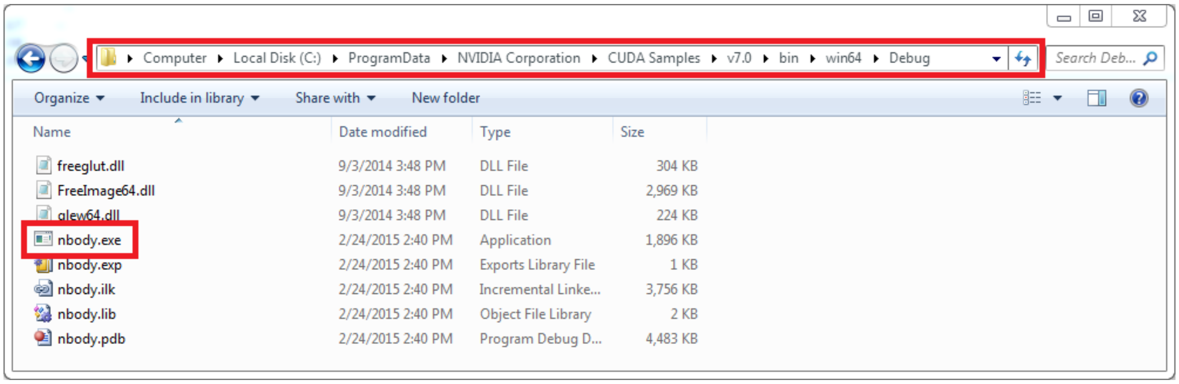
```
1>          ]
1>  nbody.cpp
1>c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\xutility(3153): warning C4244: 'initializing' : conversion from '__int64' to 'ptrdiff_t', possible loss o
1>c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\xutility(3154): warning C4244: 'initializing' : conversion from '__int64' to 'ptrdiff_t', possible loss o
1>c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\xstring(1066): warning C4267: 'argument' : conversion from 'size_t' to 'int', possible loss of data
1>          c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\xstring(1063) : while compiling class template member function 'std::_String_iterator<_Elem,_Tra
1>          with
1>          [
1>              _Elem=char,
1>              _Traits=std::char_traits<char>,
1>              _Alloc=std::allocator<char>,
1>              _Ax=std::allocator<char>
1>          ]
1>          c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\stdexcept(23) : see reference to class template instantiation 'std::basic_string<_Elem,_Traits,_
1>          with
1>          [
1>              _Elem=char,
1>              _Traits=std::char_traits<char>,
1>              _Ax=std::allocator<char>
1>          ]
1>c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\xstring(203): warning C4244: 'return' : conversion from '__int64' to 'int', possible loss of data
1>          c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\xstring(201) : while compiling class template member function 'int std::_String_const_iterator<_
1>          with
1>          [
1>              _Elem=char,
1>              _Traits=std::char_traits<char>,
1>              _Alloc=std::allocator<char>
1>          ]
1>          C:\ProgramData\NVIDIA Corporation\CUDA Samples\v7.0\common\inc\helper_string.h(451) : see reference to class template instantiation 'std::_String_const_iterat
1>          with
1>          [
1>              _Elem=char,
1>              _Traits=std::char_traits<char>,
1>              _Alloc=std::allocator<char>
1>          ]
1>  Generating Code...
1>Link:
1>     Creating library ../../bin/win64/Debug/nbody.lib and object ../../bin/win64/Debug/nbody.exp
1>  nbody_vs2010.vcxproj -> C:\ProgramData\NVIDIA Corporation\CUDA Samples\v7.0\5_Simulations\nbody\../../bin/win64/Debug/nbody.exe
1>FinalizeBuildStatus:
1>  Deleting file "x64/Debug/nbody.unsuccessfulbuild".
1>  Touching "x64/Debug/nbody.lastbuildstate".
1>
1>Build succeeded.
1>
1>Time Elapsed 00:00:41.57
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

(Name) nbody_vs2010
Active config Debug|x64
Description
Path C:\ProgramData\
Startup proje nbody

(Name)
The name of the solution file.

Build succeeded

9. Navigate to the CUDA Samples' build directory and run the nbody sample.



Computer ▸ Local Disk (C:) ▸ ProgramData ▸ NVIDIA Corporation ▸ CUDA Samples ▸ v7.0 ▸ bin ▸ win64 ▸ Debug

Name	Date modified	Type	Size
freeglut.dll	9/3/2014 3:48 PM	DLL File	304 KB
FreeImage64.dll	9/3/2014 3:48 PM	DLL File	2,969 KB
glew64.dll	9/3/2014 3:48 PM	DLL File	224 KB
nbody.exe	2/24/2015 2:40 PM	Application	1,896 KB
nbody.exp	2/24/2015 2:40 PM	Exports Library File	1 KB
nbody.ilk	2/24/2015 2:40 PM	Incremental Linke...	3,756 KB
nbody.lib	2/24/2015 2:40 PM	Object File Library	2 KB
nbody.pdb	2/24/2015 2:40 PM	Program Debug D...	4,483 KB

Chapter 3. Linux

CUDA on Linux can be installed using an RPM, Debian, or Runfile package, depending on the platform being installed on.

3.1. Linux x86_64

For development on the x86_64 architecture. In some cases, x86_64 systems may act as host platforms targeting other architectures. See the Linux Installation Guide for more details.

3.1.1. Redhat / CentOS

When installing CUDA on Redhat or CentOS, you can choose between the Runfile Installer and the RPM Installer. The Runfile Installer is only available as a Local Installer. The RPM Installer is available as both a Local Installer and a Network Installer. The Network Installer allows you to download only the files you need. The Local Installer is a stand-alone installer with a large initial download. In the case of the RPM installers, the instructions for the Local and Network variants are the same. For more details, refer to the Linux Installation Guide.

3.1.1.1. RPM Installer

Perform the following steps to install CUDA and verify the installation.

1. Install EPEL to satisfy the DKMS dependency by following the instructions at EPEL's website.
2. **Enable optional repos**:

   On **RHEL 7 Linux** only, execute the following steps to enable optional repositories.

   - **On x86_64 workstation:**
     ```
     $ subscription-manager repos --enable=rhel-7-workstation-optional-rpms
     ```
   - **On POWER9 system:**
     ```
     $ subscription-manager repos --enable=rhel-7-for-power-9-optional-rpms
     ```
   - **On x86_64 server:**
     ```
     $ subscription-manager repos --enable=rhel-7-server-optional-rpms
     ```
3. Install the repository meta-data, clean the yum cache, and install CUDA:
   ```
   $ sudo rpm --install cuda-repo-<distro>-<version>.<architecture>.rpm
   $ sudo yum clean expire-cache
   $ sudo yum install cuda
   ```

4. Reboot the system to load the NVIDIA drivers.
5. Set up the development environment by modifying the PATH and LD_LIBRARY_PATH variables:

   ```
   $ export PATH=/usr/local/cuda-11.1/bin${PATH:+:${PATH}}
   $ export LD_LIBRARY_PATH=/usr/local/cuda-11.1/lib64\
                            ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
   ```

6. Install a writable copy of the samples then build and run the nbody sample:

   ```
   $ cuda-install-samples-11.1.sh ~
   $ cd ~/NVIDIA_CUDA-11.1_Samples/5_Simulations/nbody
   $ make
   $ ./nbody
   ```

3.1.1.2. Runfile Installer

Perform the following steps to install CUDA and verify the installation.

1. Disable the Nouveau drivers:

   a). Create a file at `/etc/modprobe.d/blacklist-nouveau.conf` with the following contents:

   ```
   blacklist nouveau
   options nouveau modeset=0
   ```

   b). Regenerate the kernel initramfs:

   ```
   $ sudo dracut --force
   ```

2. Reboot into runlevel 3 by temporarily adding the number "3" and the word "nomodeset" to the end of the system's kernel boot parameters.
3. Run the installer silently to install with the default selections (implies acceptance of the EULA):

   ```
   sudo sh cuda_<version>_linux.run --silent
   ```

4. Create an xorg.conf file to use the NVIDIA GPU for display:

   ```
   $ sudo nvidia-xconfig
   ```

5. Reboot the system to load the graphical interface.
6. Set up the development environment by modifying the PATH and LD_LIBRARY_PATH variables:

   ```
   $ export PATH=/usr/local/cuda-11.1/bin${PATH:+:${PATH}}
   $ export LD_LIBRARY_PATH=/usr/local/cuda-11.1/lib64\
                            ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
   ```

7. Install a writable copy of the samples then build and run the nbody sample:

   ```
   $ cuda-install-samples-11.1.sh ~
   $ cd ~/NVIDIA_CUDA-11.1_Samples/5_Simulations/nbody
   $ make
   $ ./nbody
   ```

3.1.2. Fedora

When installing CUDA on Fedora, you can choose between the Runfile Installer and the RPM Installer. The Runfile Installer is only available as a Local Installer. The RPM Installer is available as both a Local Installer and a Network Installer. The Network Installer allows you to download only the files you need. The Local Installer is a stand-alone installer with a large initial download. In the case of the RPM installers, the instructions for the Local and Network variants are the same. For more details, refer to the Linux Installation Guide.

3.1.2.1. RPM Installer

Perform the following steps to install CUDA and verify the installation.

1. Install the RPMFusion free repository to satisfy the Akmods dependency:

   ```
   $ su -c 'dnf install --nogpgcheck http://download1.rpmfusion.org/free/fedora/
   rpmfusion-free-release-$(rpm -E %fedora).noarch.rpm'
   ```

2. Install the repository meta-data, clean the dnf cache, and install CUDA:

   ```
   $ sudo rpm --install cuda-repo-<distro>-<version>.<architecture>.rpm
   $ sudo dnf clean expire-cache
   $ sudo dnf install cuda
   ```

3. Reboot the system to load the NVIDIA drivers.
4. Set up the development environment by modifying the PATH and LD_LIBRARY_PATH variables:

   ```
   $ export PATH=/usr/local/cuda-11.1/bin${PATH:+:${PATH}}
   $ export LD_LIBRARY_PATH=/usr/local/cuda-11.1/lib64\
                            ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
   ```

5. Install a writable copy of the samples then build and run the nbody sample:

   ```
   $ cuda-install-samples-11.1.sh ~
   $ cd ~/NVIDIA_CUDA-11.1_Samples/5_Simulations/nbody
   $ make
   $ ./nbody
   ```

3.1.2.2. Runfile Installer

Perform the following steps to install CUDA and verify the installation.

1. Disable the Nouveau drivers:

   a). Create a file at `/usr/lib/modprobe.d/blacklist-nouveau.conf` with the following contents:

   ```
   blacklist nouveau
   options nouveau modeset=0
   ```

   b). Regenerate the kernel initramfs:

   ```
   $ sudo dracut --force
   ```

   c). Run the below command:

   ```
   $ sudo grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

   d). Reboot the system.

2. Reboot into runlevel 3 by temporarily adding the number "3" and the word "nomodeset" to the end of the system's kernel boot parameters.
3. Run the installer silently to install with the default selections (implies acceptance of the EULA):

   ```
   sudo sh cuda_<version>_linux.run --silent
   ```

4. Create an xorg.conf file to use the NVIDIA GPU for display:

   ```
   $ sudo nvidia-xconfig
   ```

5. Reboot the system to load the graphical interface.
6. Set up the development environment by modifying the PATH and LD_LIBRARY_PATH variables:

   ```
   $ export PATH=/usr/local/cuda-11.1/bin${PATH:+:${PATH}}
   $ export LD_LIBRARY_PATH=/usr/local/cuda-11.1/lib64\
   ```

```
                         ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

7. Install a writable copy of the samples then build and run the nbody sample:

   ```
   $ cuda-install-samples-11.1.sh ~
   $ cd ~/NVIDIA_CUDA-11.1_Samples/5_Simulations/nbody
   $ make
   $ ./nbody
   ```

3.1.3. SUSE Linux Enterprise Server

When installing CUDA on SUSE Linux Enterprise Server, you can choose between the Runfile Installer and the RPM Installer. The Runfile Installer is only available as a Local Installer. The RPM Installer is available as both a Local Installer and a Network Installer. The Network Installer allows you to download only the files you need. The Local Installer is a stand-alone installer with a large initial download. In the case of the RPM installers, the instructions for the Local and Network variants are the same. For more details, refer to the Linux Installation Guide.

3.1.3.1. RPM Installer

Perform the following steps to install CUDA and verify the installation.

1. Install the repository meta-data, refresh the Zypper cache, and install CUDA:

   ```
   $ sudo rpm --install cuda-repo-<distro>-<version>.<architecture>.rpm
   $ sudo SUSEConnect --product PackageHub/15/x86_64
   $ sudo zypper refresh
   $ sudo zypper install cuda
   ```

2. Add the user to the video group:

   ```
   $ sudo usermod -a -G video <username>
   ```

3. Reboot the system to load the NVIDIA drivers.
4. Set up the development environment by modifying the PATH and LD_LIBRARY_PATH variables:

   ```
   $ export PATH=/usr/local/cuda-11.1/bin${PATH:+:${PATH}}
   $ export LD_LIBRARY_PATH=/usr/local/cuda-11.1/lib64\
                            ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
   ```

5. Install a writable copy of the samples then build and run the vectorAdd sample:

   ```
   $ cuda-install-samples-11.1.sh ~
   $ cd ~/NVIDIA_CUDA-11.1_Samples/0_Simple/vectorAdd
   $ make
   $ ./vectorAdd
   ```

3.1.3.2. Runfile Installer

Perform the following steps to install CUDA and verify the installation.

1. Reboot into runlevel 3 by temporarily adding the number "3" and the word "nomodeset" to the end of the system's kernel boot parameters.
2. Run the installer silently to install with the default selections (implies acceptance of the EULA):

   ```
   sudo sh cuda_<version>_linux.run --silent
   ```

3. Create an xorg.conf file to use the NVIDIA GPU for display:

   ```
   $ sudo nvidia-xconfig
   ```

4. Reboot the system to load the graphical interface.

5. Set up the development environment by modifying the PATH and LD_LIBRARY_PATH variables:

```
$ export PATH=/usr/local/cuda-11.1/bin${PATH:+:${PATH}}
$ export LD_LIBRARY_PATH=/usr/local/cuda-11.1/lib64\
                         ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

6. Install a writable copy of the samples then build and run the vectorAdd sample:

```
$ cuda-install-samples-11.1.sh ~
$ cd ~/NVIDIA_CUDA-11.1_Samples/0_Simple/vectorAdd
$ make
$ ./vectorAdd
```

3.1.4. OpenSUSE

When installing CUDA on OpenSUSE, you can choose between the Runfile Installer and the RPM Installer. The Runfile Installer is only available as a Local Installer. The RPM Installer is available as both a Local Installer and a Network Installer. The Network Installer allows you to download only the files you need. The Local Installer is a stand-alone installer with a large initial download. In the case of the RPM installers, the instructions for the Local and Network variants are the same. For more details, refer to the Linux Installation Guide.

3.1.4.1. RPM Installer

Perform the following steps to install CUDA and verify the installation.

1. Install the repository meta-data, refresh the Zypper cache, and install CUDA:

```
$ sudo rpm --install cuda-repo-<distro>-<version>.<architecture>.rpm
$ sudo zypper refresh
$ sudo zypper install cuda
```

2. Add the user to the video group:

```
$ sudo usermod -a -G video <username>
```

3. Reboot the system to load the NVIDIA drivers.
4. Set up the development environment by modifying the PATH and LD_LIBRARY_PATH variables:

```
$ export PATH=/usr/local/cuda-11.1/bin${PATH:+:${PATH}}
$ export LD_LIBRARY_PATH=/usr/local/cuda-11.1/lib64\
                         ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

5. Install a writable copy of the samples then build and run the nbody sample:

```
$ cuda-install-samples-11.1.sh ~
$ cd ~/NVIDIA_CUDA-11.1_Samples/5_Simulations/nbody
$ make
$ ./nbody
```

3.1.4.2. Runfile Installer

Perform the following steps to install CUDA and verify the installation.

1. Disable the Nouveau drivers:

   a). Create a file at `/etc/modprobe.d/blacklist-nouveau.conf` with the following contents:

   ```
   blacklist nouveau
   options nouveau modeset=0
   ```

   b). Regenerate the kernel initrd:

```
$ sudo /sbin/mkinitrd
```

2. Reboot into runlevel 3 by temporarily adding the number "3" and the word "nomodeset" to the end of the system's kernel boot parameters.
3. Run the installer silently to install with the default selections (implies acceptance of the EULA):
```
sudo sh cuda_<version>_linux.run --silent
```
4. Create an xorg.conf file to use the NVIDIA GPU for display:
```
$ sudo nvidia-xconfig
```
5. Reboot the system to load the graphical interface.
6. Set up the development environment by modifying the PATH and LD_LIBRARY_PATH variables:
```
$ export PATH=/usr/local/cuda-11.1/bin${PATH:+:${PATH}}
$ export LD_LIBRARY_PATH=/usr/local/cuda-11.1/lib64\
                    ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```
7. Install a writable copy of the samples then build and run the nbody sample:
```
$ cuda-install-samples-11.1.sh ~
$ cd ~/NVIDIA_CUDA-11.1_Samples/5_Simulations/nbody
$ make
$ ./nbody
```

3.1.5. WSL

These instructions must be used if you are installing in a WSL environment. Do not use the Ubuntu instructions in this case.

1. **Install repository meta-data**
```
$ sudo dpkg -i cuda-repo-<distro>_<version>_<architecture>.deb
```
2. **Install the CUDA public GPG key**

   When installing using the local repo:
```
$ sudo apt-key add /var/cuda-repo-<version>/7fa2af80.pub
```
   When installing using the network repo:
```
$ sudo apt-key adv --fetch-keys https://developer.download.nvidia.com/compute/
cuda/repos/<distro>/<architecture>/7fa2af80.pub
```
   Pin file to prioritize CUDA repository:
```
$ wget https://developer.download.nvidia.com/compute/cuda/repos/
<distro>/<architecture>/cuda-<distro>.pin
$ sudo mv cuda-<distro>.pin /etc/apt/preferences.d/cuda-
repository-pin-600
```
3. **Update the Apt repository cache and install CUDA**
```
$ sudo apt-get update
$ sudo apt-get install cuda
```

3.1.6. Ubuntu

When installing CUDA on Ubuntu, you can choose between the Runfile Installer and the Debian Installer. The Runfile Installer is only available as a Local Installer. The Debian Installer is available as both a Local Installer and a Network Installer. The Network Installer allows you to download only the files you need. The Local Installer is a stand-alone installer with a large initial download. In the case of the Debian installers, the instructions for the Local and Network variants are the same. For more details, refer to the Linux Installation Guide.

3.1.6.1. Debian Installer

Perform the following steps to install CUDA and verify the installation.

1. Install the repository meta-data, install GPG key, update the apt-get cache, and install CUDA:

   ```
   $ sudo dpkg --install cuda-repo-<distro>-<version>.<architecture>.deb
   $ sudo apt-key adv --fetch-keys https://developer.download.nvidia.com/compute/
   cuda/repos/<distro>/<architecture>/7fa2af80.pub
   $ sudo apt-get update
   $ sudo apt-get install cuda
   ```

2. Reboot the system to load the NVIDIA drivers.
3. Set up the development environment by modifying the PATH and LD_LIBRARY_PATH variables:

   ```
   $ export PATH=/usr/local/cuda-11.1/bin${PATH:+:${PATH}}
   $ export LD_LIBRARY_PATH=/usr/local/cuda-11.1/lib64\
                            ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
   ```

4. Install a writable copy of the samples then build and run the nbody sample:

   ```
   $ cuda-install-samples-11.1.sh ~
   $ cd ~/NVIDIA_CUDA-11.1_Samples/5_Simulations/nbody
   $ make
   $ ./nbody
   ```

3.1.6.2. Runfile Installer

Perform the following steps to install CUDA and verify the installation.

1. Disable the Nouveau drivers:

   a). Create a file at `/etc/modprobe.d/blacklist-nouveau.conf` with the following contents:

   ```
   blacklist nouveau
   options nouveau modeset=0
   ```

   b). Regenerate the kernel initramfs:

   ```
   $ sudo update-initramfs -u
   ```

2. Reboot into runlevel 3 by temporarily adding the number "3" and the word "nomodeset" to the end of the system's kernel boot parameters.
3. Run the installer silently to install with the default selections (implies acceptance of the EULA):

   ```
   sudo sh cuda_<version>_linux.run --silent
   ```

4. Create an xorg.conf file to use the NVIDIA GPU for display:

   ```
   $ sudo nvidia-xconfig
   ```

5. Reboot the system to load the graphical interface.
6. Set up the development environment by modifying the PATH and LD_LIBRARY_PATH variables:

   ```
   $ export PATH=/usr/local/cuda-11.1/bin${PATH:+:${PATH}}
   $ export LD_LIBRARY_PATH=/usr/local/cuda-11.1/lib64\
   ```

```
                         ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

7. Install a writable copy of the samples then build and run the nbody sample:

```
$ cuda-install-samples-11.1.sh ~
$ cd ~/NVIDIA_CUDA-11.1_Samples/5_Simulations/nbody
$ make
$ ./nbody
```

3.1.7. Debian

When installing CUDA on Debian 10, you can choose between the Runfile Installer and the Debian Installer. The Runfile Installer is only available as a Local Installer. The Debian Installer is available as both a Local Installer and a Network Installer. The Network Installer allows you to download only the files you need. The Local Installer is a stand-alone installer with a large initial download. For more details, refer to the Linux Installation Guide.

3.1.7.1. Debian Installer

Perform the following steps to install CUDA and verify the installation.

1. Install the repository meta-data, install GPG key, update the apt-get cache, and install CUDA:

```
$ sudo dpkg -i cuda-repo-<distro>_<version>_<architecture>.deb
$ sudo apt-key adv --fetch-keys https://developer.download.nvidia.com/compute/
cuda/repos/debian10/x86_64/7fa2af80.pub
$ sudo add-apt-repository contrib
$ sudo apt-get update
$ sudo apt-get -y install cuda
```

2. Reboot the system to load the NVIDIA drivers.
3. Set up the development environment by modifying the PATH and LD_LIBRARY_PATH variables:

```
$ export PATH=/usr/local/cuda-11.1/bin${PATH:+:${PATH}}
$ export LD_LIBRARY_PATH=/usr/local/cuda-11.1/lib64\
                         ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

4. Install a writable copy of the samples then build and run the nbody sample:

```
$ cuda-install-samples-11.1.sh ~
$ cd ~/NVIDIA_CUDA-11.1_Samples/5_Simulations/nbody
$ make
$ ./nbody
```

3.1.7.2. Runfile Installer

Perform the following steps to install CUDA and verify the installation.

1. Disable the Nouveau drivers:

   a). Create a file at `/etc/modprobe.d/blacklist-nouveau.conf` with the following contents:

   ```
   blacklist nouveau
   options nouveau modeset=0
   ```

   b). Regenerate the kernel initramfs:

   ```
   $ sudo update-initramfs -u
   ```

2. Reboot into runlevel 3 by temporarily adding the number "3" and the word "nomodeset" to the end of the system's kernel boot parameters.

3. Run the installer silently to install with the default selections (implies acceptance of the EULA):

```
$ sudo sh cuda_<version>_linux.run --silent
```

4. Create an xorg.conf file to use the NVIDIA GPU for display:

```
$ sudo nvidia-xconfig
```

5. Reboot the system to load the graphical interface.
6. Set up the development environment by modifying the PATH and LD_LIBRARY_PATH variables:

```
$ export PATH=/usr/local/cuda-11.1/bin${PATH:+:${PATH}}
$ export LD_LIBRARY_PATH=/usr/local/cuda-11.1/lib64\
                         ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

7. Install a writable copy of the samples then build and run the nbody sample:

```
$ cuda-install-samples-11.1.sh ~
$ cd ~/NVIDIA_CUDA-11.1_Samples/5_Simulations/nbody
$ make
$ ./nbody
```

3.2. Linux POWER8

For development on the POWER8 architecture.

3.2.1. Ubuntu

When installing CUDA on Ubuntu on POWER8, you must use the Debian Installer. The Debian Installer is available as both a Local Installer and a Network Installer. The Network Installer allows you to download only the files you need. The Local Installer is a stand-alone installer with a large initial download. The instructions for the Local and Network variants are the same. For more details, refer to the Linux Installation Guide.

3.2.1.1. Debian Installer

Perform the following steps to install CUDA and verify the installation.

1. Install the repository meta-data, update the apt-get cache, and install CUDA:

```
$ sudo dpkg --install cuda-repo-<distro>-<version>.<architecture>.deb
$ sudo apt-get update
$ sudo apt-get install cuda
```

2. Reboot the system to load the NVIDIA drivers.
3. Set up the development environment by modifying the PATH and LD_LIBRARY_PATH variables:

```
$ export PATH=/usr/local/cuda-11.1/bin${PATH:+:${PATH}}
$ export LD_LIBRARY_PATH=/usr/local/cuda-11.1/lib64\
                         ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

4. Install a writable copy of the samples then build and run the vectorAdd sample:

```
$ cuda-install-samples-11.1.sh ~
$ cd ~/NVIDIA_CUDA-11.1_Samples/0_Simple/vectorAdd
$ make
$ ./vectorAdd
```

3.2.2. Redhat / CentOS

When installing CUDA on Redhat on POWER8, you must use the RPM Installer. The RPM Installer is available as both a Local Installer and a Network Installer. The Network Installer allows you to download only the files you need. The Local Installer is a stand-alone installer with a large initial download. The instructions for the Local and Network variants are the same. For more details, refer to the Linux Installation Guide.

3.2.2.1. RPM Installer

Perform the following steps to install CUDA and verify the installation.

1. Install EPEL to satisfy the DKMS dependency by following the instructions at EPEL's website.
2. Install the repository meta-data, clean the yum cache, and install CUDA:

   ```
   $ sudo rpm --install cuda-repo-<distro>-<version>.<architecture>.rpm
   $ sudo yum clean expire-cache
   $ sudo yum install cuda
   ```
3. Reboot the system to load the NVIDIA drivers.
4. Set up the development environment by modifying the PATH and LD_LIBRARY_PATH variables:

   ```
   $ export PATH=/usr/local/cuda-11.1/bin${PATH:+:${PATH}}
   $ export LD_LIBRARY_PATH=/usr/local/cuda-11.1/lib64\
                            ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
   ```
5. Install a writable copy of the samples then build and run the vectorAdd sample:

   ```
   $ cuda-install-samples-11.1.sh ~
   $ cd ~/NVIDIA_CUDA-11.1_Samples/0_Simple/vectorAdd
   $ make
   $ ./vectorAdd
   ```

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

VESA DisplayPort

HDMI

OpenCL

Trademarks

Copyright

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com