

# CUDA 在 linux 系统上安装指南

## 适用的操作系统

Fedora 7, 8, 9, 10

Redhat Enterprise 3.x,4.x,5.x

SUSE Linux Enterprise Desktop 10-SP1, 10.2, 11.0

OpenSUSE 10.1,10.2, 10.3, 11.0, 11.1

Ubuntu 7.04 , 7.10., 8.04, 8.10, 9.04

-----  
下载和操作系统配套的  
驱动程序, SDK, toolkit

地址: [http://www.nvidia.com/object/cuda\\_get.html](http://www.nvidia.com/object/cuda_get.html)

-----  
安装程序 (TESLA 搭配非 NVIDIA 显卡使用, 可不须安装该显卡驱动程序)

## 在 Terminal 中安装 ( 不要进 XWindow )

以 linux as5.2 下安装 cuda2.1 为例

### 1. 安装 CUDA 运算驱动程序

命令行下执行: `sh NVIDIA-Linux-x86_64-180.22-pkg2.run`

根据提示回车执行各步安装过程

关于如何安装 NVIDIA 的 Linux 驱动程序, 请参考

NVIDIA Accelerated Linux Driver Set README and Installation Guide

<http://us.download.nvidia.com/XFree86/Linux-x86/1.0-9755/README/index.html>

安装完毕可以在 Terminal 中执行[nvidia-xconfig -query-gpu-info]以查看所安装的  
NVIDIA GPU

执行结果请见下图



```
root@nb-dick: ~/NVIDIA_CUDA_SDK/bin/linux/release - Shell - Konsole
Session: Edit View Bookmarks Settings Help

[root@nb-dick release]# nvidia-xconfig -query-gpu-info
Number of GPUs: 4

GPU #0:
  Name      : Tesla C870
  PCI BusID : PCI:0:0:0

  Number of Display Devices: 0

GPU #1:
  Name      : Tesla C870
  PCI BusID : PCI:10:0:0

  Number of Display Devices: 0

GPU #2:
  Name      : Tesla C870
  PCI BusID : PCI:135:0:0

  Number of Display Devices: 0

GPU #3:
  Name      : Tesla C870
  PCI BusID : PCI:137:0:0

  Number of Display Devices: 0

[root@nb-dick release]# nvidia-smi
Gpus found in probe:
Found Gpuid 0x8000
Found Gpuid 0xa000
Found Gpuid 0x87000
Found Gpuid 0x89000
Attaching all probed Gpus...OK
Setting unit information...OK
Getting all static information...
[root@nb-dick release]#
```

## 2. 安装 NVIDIA CUDA Toolkit

命令行下执行: `sh cudatoolkit_2.1_linux64_rhel5.2.run`

安装程序会要求你输入安装路径或是接受默认值, 推荐以 `root` 身份安装并使用默认路径 (`/usr/local`) ,

在之后我们将会以 `<CUDA_INSTALL_PATH>` 来代替实际的安装路径

增加 `CUDA` 二进制文件(`nvcc`)及函数路径(`libcuda.so`)到 `PATH` 及 `LD_LIBRARY_PATH` 的环境变量

安装完毕可以执行 `[nvidia-smi]` 以查看所安装的 `CUDA GPU`

[nvidia-smi]是 NVIDIA 提供可以让我们确认安装在机器中的 GPU 是否都能正常运行 CUDA 的新工具

执行结果请见下图



```
root@nb-dick:~/NVIDIA_CUDA_SDK/bin/linux/release - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@nb-dick release]# nvidia-xconfig -query-gpu-info
Number of GPUs: 4

GPU #0:
  Name       : Tesla C870
  PCI BusID  : PCI:8:0:0

  Number of Display Devices: 0

GPU #1:
  Name       : Tesla C870
  PCI BusID  : PCI:10:0:0

  Number of Display Devices: 0

GPU #2:
  Name       : Tesla C870
  PCI BusID  : PCI:135:0:0

  Number of Display Devices: 0

GPU #3:
  Name       : Tesla C870
  PCI BusID  : PCI:137:0:0

  Number of Display Devices: 0

[root@nb-dick release]# nvidia-smi
Gpus found in probe:
Found Gpuid 0x8000
Found Gpuid 0xa000
Found Gpuid 0x8700
Found Gpuid 0x8900
Attaching all probed Gpus...OK
Setting unit information...OK
Getting all static information...
[root@nb-dick release]#
```

### 3. 安装 NVIDIA CUDA SDK

命令行下执行: shcuda-sdk-linux-2.10.1215.2015-3233425.run

安装程序会要求输入安装路径或是接受默认值, 默认安装路径为用户的家目录 (/NVIDIA\_CUDA\_SDK)。

在之后我们将会以<SDK\_INSTALL\_PATH>来代替实际的安装路径在家目录下的.bash\_profile 中, 加入以下几行

```
PATH=$PATH: <CUDA_INSTALL_PATH>/bin
LD_LIBRARY_PATH=$LD_LIBRARY_PATH: <CUDA_INSTALL_PATH>/lib64
export PATH
export LD_LIBRARY_PATH
```

注意<CUDA\_INSTALL\_PATH> 用系统中安装的实际路径替代

然后启用该配置

```
source .bash_profile
```

#### 4. 建构 SDK project 范例程序

```
cd <SDK_INSTALL_PATH>
```

Build:

- release 输入 "make".
- debug 输入 "make dbg=1".
- emurelease 输入 "make emu=1".
- emudebug 输入 "make emu=1 dbg=1".make

在<SDK\_INSTALL\_PATH>执行 make 创建范例程序所使用的 libcutil 这个公共工具  
libcutil 是为了方便使用而提供的，不属于 CUDA 的一部分

注意：

在 make 时一些用到 opengl 的范例会发生有关 gl 的编译错误，这是因为没有安装 opengl 的库造成的，需要另外下载安装 gl 的库。

其他的范例应该编译正常。

可以直接到 /root/NVIDIA\_CUDA\_SDK/projects 下的各个范例中直接执行 make 进行编译：

如：矩阵乘

```
cd /root/NVIDIA_CUDA_SDK/projects/matrixMul
Make
```

#### 5. 执行范例

在范例程序当中的 deviceQuery 是让我们得到装在这台机器上可以进行 CUDA 运算的 GPU 信息

建构范例程序

```
cd <SDK_INSTALL_PATH>/projects/deviceQuery
```

make

然后在 `<SDK_INSTALL_PATH>/bin/linux32/release/deviceQuery` 执行范例程序 `deviceQuery` 执行结果如下图

而执行 `release`, `debug`, `emurelease` 或 `emudebug` 等  
其目录位于 `/bin/linux32/[release|debug|emurelease|emudebug]`



```
root@nb-dick: ~/NVIDIA_CUDA_SDK/bin/linux/release - Shell - Konsole
Session: Edit View Bookmarks Settings Help
[ront@nb-dick release]$ ./deviceQuery
There are 4 devices supporting CUDA

Device 0: "Tesla C870"
Major revision number:      1
Minor revision number:      0
Total amount of global memory: 1610350592 bytes
Total amount of constant memory: 65536 bytes
Total amount of shared memory per block: 16384 bytes
Total number of registers available per block: 8192
Warp size: 32
Maximum number of threads per block: 512
Maximum sizes of each dimension of a block: 512 x 512 x 64
Maximum sizes of each dimension of a grid: 65535 x 65535 x 1
Maximum memory pitch: 262144 bytes
Texture alignment: 256 bytes
Clock rate: 1350000 kilohertz

Device 1: "Tesla C870"
Major revision number:      1
Minor revision number:      0
Total amount of global memory: 1610350592 bytes
Total amount of constant memory: 65536 bytes
Total amount of shared memory per block: 16384 bytes
Total number of registers available per block: 8192
Warp size: 32
Maximum number of threads per block: 512
Maximum sizes of each dimension of a block: 512 x 512 x 64
Maximum sizes of each dimension of a grid: 65535 x 65535 x 1
Maximum memory pitch: 262144 bytes
Texture alignment: 256 bytes
Clock rate: 1350000 kilohertz

Device 2: "Tesla C870"
Major revision number:      1
Minor revision number:      0
Total amount of global memory: 1610350592 bytes
Total amount of constant memory: 65536 bytes
Total amount of shared memory per block: 16384 bytes
Total number of registers available per block: 8192
Warp size: 32
Maximum number of threads per block: 512
Maximum sizes of each dimension of a block: 512 x 512 x 64
Maximum sizes of each dimension of a grid: 65535 x 65535 x 1
Maximum memory pitch: 262144 bytes
Texture alignment: 256 bytes
Clock rate: 1350000 kilohertz

Device 3: "Tesla C870"
Major revision number:      1
Minor revision number:      0
Total amount of global memory: 1610350592 bytes
Total amount of constant memory: 65536 bytes
Total amount of shared memory per block: 16384 bytes
Total number of registers available per block: 8192
Warp size: 32
Maximum number of threads per block: 512
Maximum sizes of each dimension of a block: 512 x 512 x 64
Maximum sizes of each dimension of a grid: 65535 x 65535 x 1
Maximum memory pitch: 262144 bytes
Texture alignment: 256 bytes
Clock rate: 1350000 kilohertz

Test PASSED
Press ENTER to exit...
3 4 root@nb-dick:~/NVIDIA
```

---

## 创建自己的程序

---

使用 CUDA SDK 可以很容易的创建新的 CUDA 程序。

以复制及修改 CUDA SDK 提供的项目"template"的方式来符合你的需求  
步骤如下

1. 复制整个"template"项目（这边以 myproject 代表你所要创建的项目）  
`cd <SDK_INSTALL_PATH>/projects`  
`cp -r template <myproject>`

2. 把项目的文件名称改成你要的文件名称

```
mv template.cu myproject.cu
```

```
mv template_kernel.cu myproject_kernel.cu
```

```
mv template_gold.cpp myproject_gold.cpp
```

3. 把项目内容的文件名称改成你要的文件名称

编辑 Makefile 及原始档

把所有的"template"用"myproject"取代

4. 编译

```
make
```

5. 在下面的位置执行新的程序

```
../../bin/linux32/release/myproject
```

执行结果应该是"Test PASSED"

6. 最后再将程序代码改成符合你的运算需求即可

**此部份请参考 [CUDA Programming Guide](#)**