

CUDA ON OS X

The following code was tested to work on OS X Lion 10.7.3 with Xcode 4.3, CUDA Toolkit 4.1, CUDA DevDriver 4.1.28, and GPU Computing SDK 4.0.17.

Word of caution: problems with installation of CUDA SDK were reported on OS X Mountain Lion 10.8 Developer's Preview.

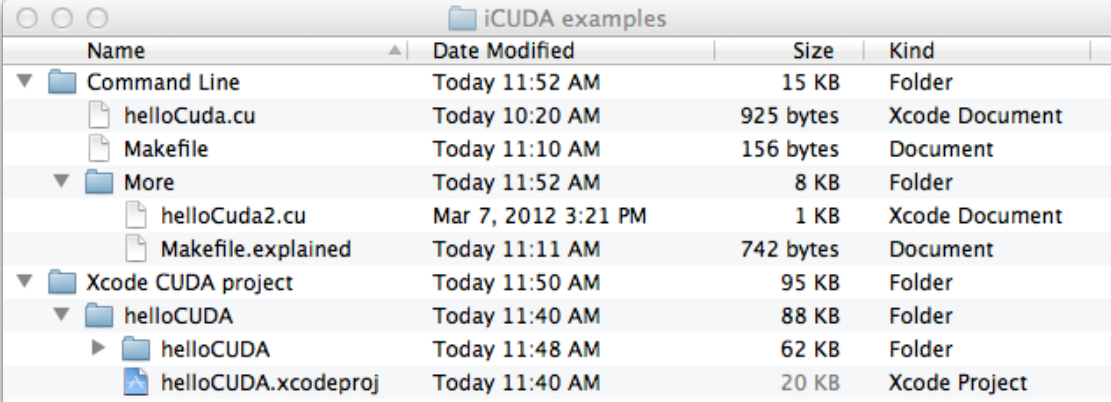
INSTALLATION

A sufficient guide for installing Xcode and CUDA is available on nVidia Dev site:

http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_Getting_Started_Mac.pdf

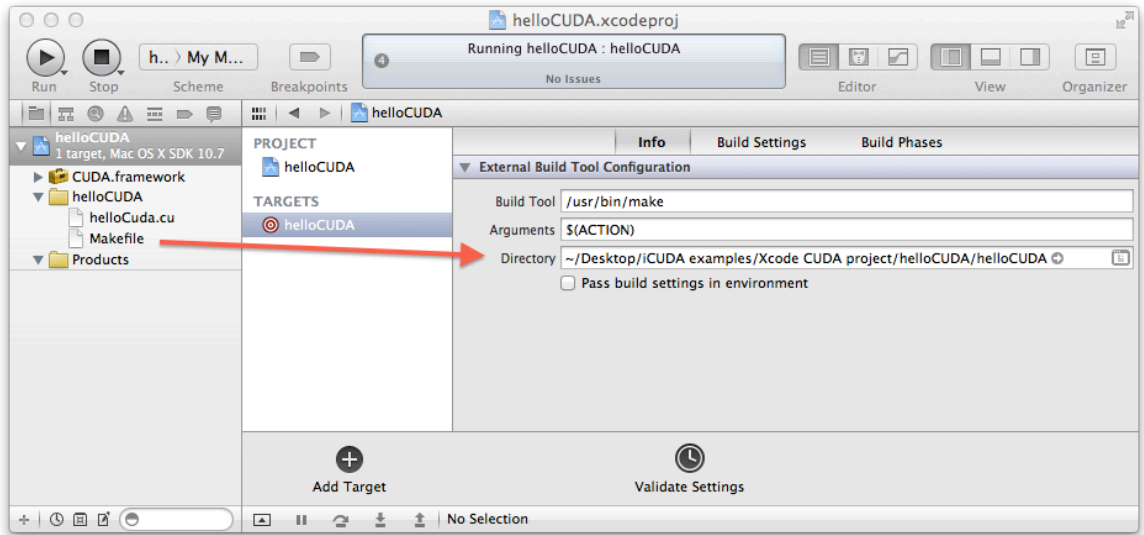
ABOUT THE INCLUDED EXAMPLES

The example CUDA scripts are located in `/iCUDA examples/Command Line`. Another version of a 'hello world' script is in the `More` folder along with a `Makefile.explained` that includes comments on relevant and optional compiler instructions.



Name	Date Modified	Size	Kind
▼ Command Line	Today 11:52 AM	15 KB	Folder
helloCuda.cu	Today 10:20 AM	925 bytes	Xcode Document
Makefile	Today 11:10 AM	156 bytes	Document
▼ More	Today 11:52 AM	8 KB	Folder
helloCuda2.cu	Mar 7, 2012 3:21 PM	1 KB	Xcode Document
Makefile.explained	Today 11:11 AM	742 bytes	Document
▼ Xcode CUDA project	Today 11:50 AM	95 KB	Folder
▼ helloCUDA	Today 11:40 AM	88 KB	Folder
▶ helloCUDA	Today 11:48 AM	62 KB	Folder
helloCUDA.xcodeproj	Today 11:40 AM	20 KB	Xcode Project

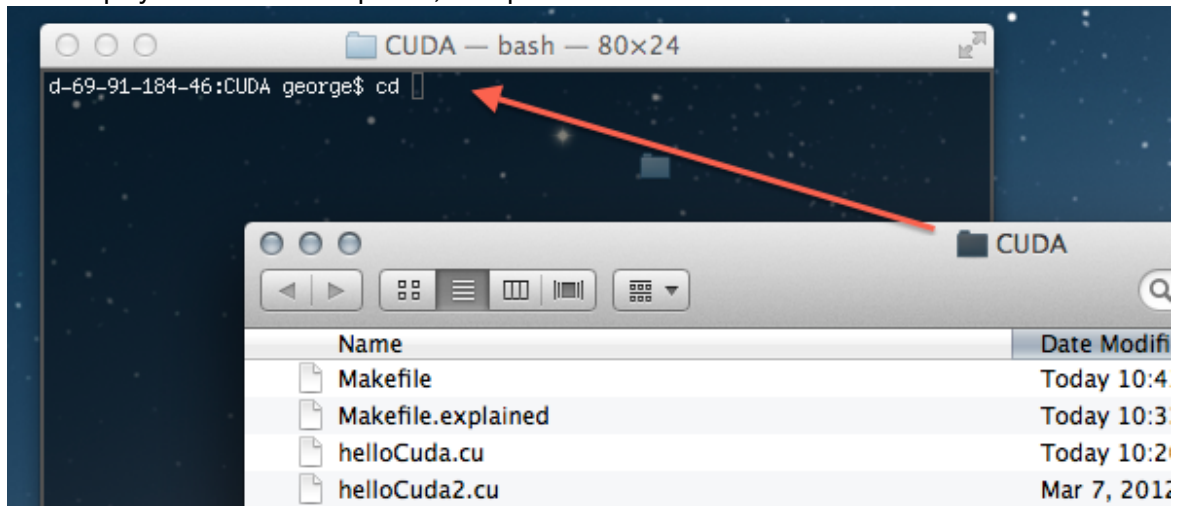
If you wish to use Xcode to write and compile your CUDA programs, open the included Xcode project. Before building and running the project, be sure to modify the path to the folder with the included Makefile (see screenshot below)



COMPILING

The following assumes you are using the default bash in Terminal.app and you have navigated to the folder with the file you wish to compile.

To do this, open **Terminal.app**, type in “**cd** ” (note the space) and drag the folder, where the script you want to compile is, and press Enter.



COMPILING A SINGLE FILE

IF YOU ARE NOT REFERENCING ANY SPECIFIC CUDA LIBRARIES

Compiling a single file can be done with the following command:

```
nvcc helloCuda.cu -o helloCUDA.a
```

where **helloCUDA.a** is the name of the compiled executable, and **helloCuda.cu** is the CUDA script you want to compile.

IF YOU ARE REFERENCING SPECIFIC CUDA LIBRARIES

```
nvcc -I/Developer/GPU\ Computing/C/common/inc
-L/Developer/GPU\ Computing/C/lib -lcutil_x86_64 -o helloCuda
helloCuda.cu
-I/$PATH - tells the compiler to include common CUDA headers (necessary if
you're including cuda.h or other in your code),
-L/$PATH - loads the required CUDA library.
```

COMPILING MULTIPLE FILES (C/C++/CUDA)

When your program spans multiple files (and languages) you can make use of Makefiles. A Makefile is a plaintext file with a set of specific instructions for the compiler. The following is an example of a Makefile:

```
EXECUTABLE := helloCUDA #name of compiled executable

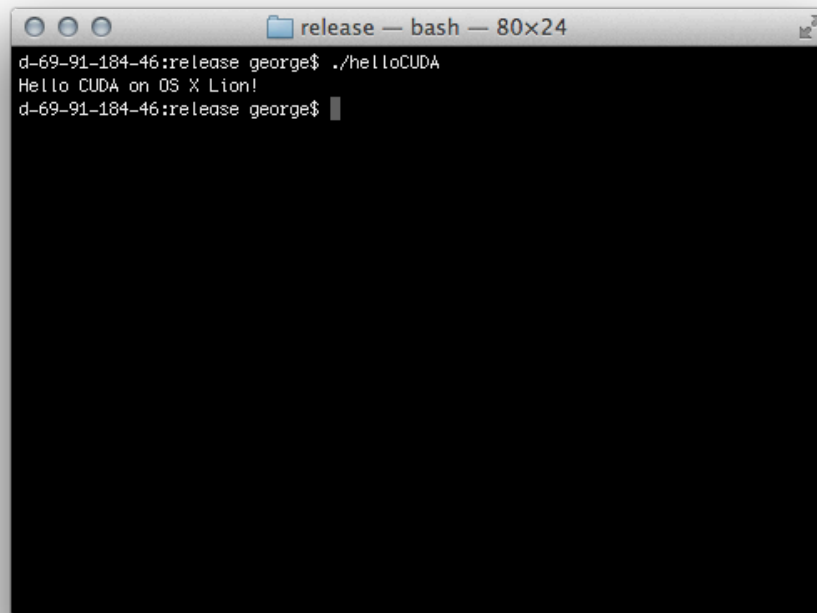
CCFILES := main.cpp #name(s) of C/C++ file(s) to compile
CUFILES := helloCuda.cu #name(s) of CUDA file(s) to compile

ROOTDIR := /Developer/GPU\ Computing/C/common
include $(ROOTDIR)/../common/common.mk
```

See the example [Makefile.explained](#) for more details.

RUNNING A COMPILED EXECUTABLE

Navigate to `/Developer/GPU Computing/C/bin/darwin/release` and either double click your executable, or “`.$EXECUTABLE_NAME`” in the Terminal:

A screenshot of a macOS Terminal window titled "release — bash — 80x24". The window shows a user named "george" at a prompt "d-69-91-184-46:release george\$". The user enters the command " ./helloCUDA". The terminal outputs "Hello CUDA on OS X Lion!". The prompt then changes to "d-69-91-184-46:release george\$" with a cursor at the end.

```
d-69-91-184-46:release george$ ./helloCUDA
Hello CUDA on OS X Lion!
d-69-91-184-46:release george$
```