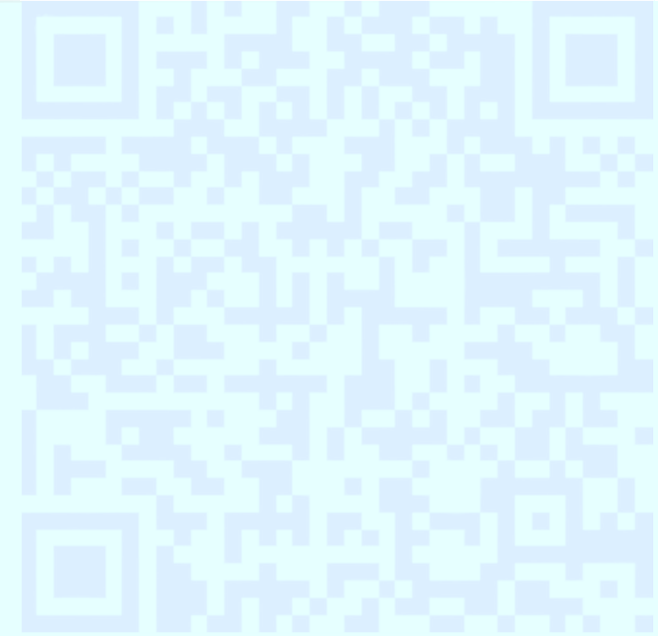


# CUDA 프로그래밍

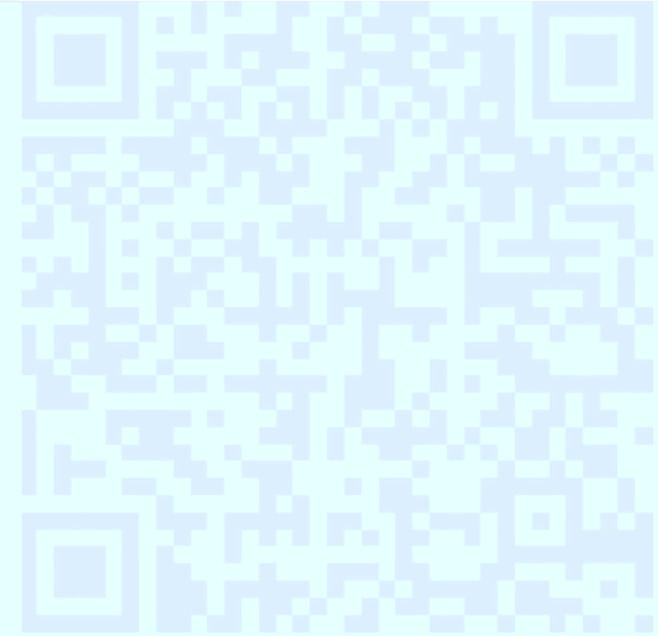
CUDA Programming



**biztripcru@gmail.com**

© 2021-2022. biztripcru@gmail.com. All rights reserved.  
모든 저작권은 biztripcru@gmail.com 에게 있습니다.

# Hello, Linux



**본** 동영상과, 본 동영상 촬영에 사용된 발표 자료는 저작권법의 보호를 받습니다.

**본** 동영상과 발표 자료는 공개/공유/복제/상업적 이용 등, **개인 수강 이외의 다른 목적으로 사용하지 못합니다.**

© 2021-2022. biztripcru@gmail.com. All rights reserved.  
모든 저작권은 biztripcru@gmail.com 에게 있습니다.

## 내용 contents

- **CUDA 프로그램을 좀더 향상시킨다**
  - Linux 에서도 돌아가게 한다
  - nvcc option을 설정한다

# Linux 에서의 CUDA 사용

- You should have a **CUDA-capable graphics card**.
- google search for “Linux CUDA install”
- in the case of “Ubuntu”,
  - \$ ubuntu-drivers devices
  - \$ **sudo apt install nvidia-driver-460** (or any new one)
    - ▶ reboot !
  - \$ nvidia-smi (only for checking your new NVIDIA driver)
    - ▶ visit CUDA-zone to get the **CUDA toolkit** (same as Win10 case)
  - \$ **sudo apt-get install build-essential** (to get **GCC** compilers)

# Linux 에서의 CUDA 사용 계속

- \$ nvcc -V
  - ▶ (now you should get the NVIDIA CUDA Compiler messages)

```
linux/cuda-work > nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2021 NVIDIA Corporation
Built on Mon_May__3_19:15:13_PDT_2021
Cuda compilation tools, release 11.3, V11.3.109
Build cuda_11.3.r11.3/compiler.29920130_0
linux/cuda-work > █
```

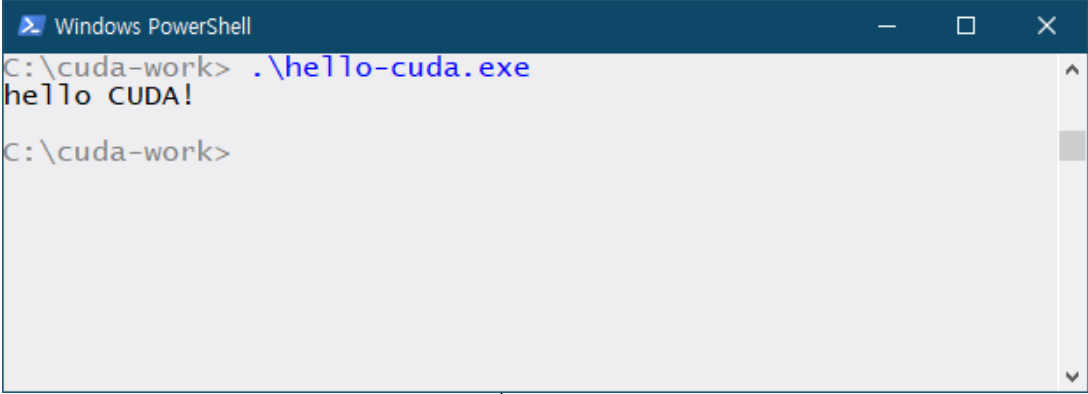
# hello-cuda.cu

```
#include <stdio.h>

__global__ void hello( void ) {
    printf( "hello CUDA!\n" );
}

int main( void ) {
    hello<<<1,1>>>>();
    fflush( stdout );
    return 0;
}
```

- `__global__`
  - **CUDA 함수** 임을 표시
- `<<<1,1>>>`
  - $1 \times 1 = 1$ 번만 실행
- 주의: **Linux**에서는 실패할 수 있음



```
Windows PowerShell
C:\cuda-work> .\hello-cuda.exe
hello CUDA!
C:\cuda-work>
```

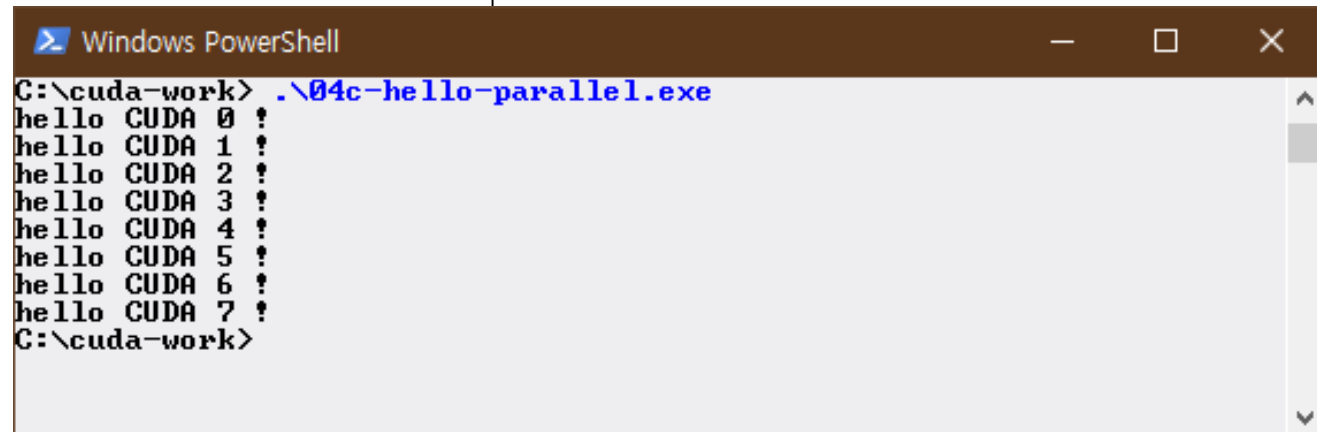
# hello-parallel.cu

```
#include <stdio.h>

__global__ void hello( void ) {
    printf( "hello CUDA %d !\n", threadIdx.x );
}

int main( void ) {
    hello<<<1,8>>>>();
    fflush( stdout );
    return 0;
}
```

- <<<1,8>>>
  - 1 x 8 = 8개의 core 사용
  - 함수를 8번 동시 실행
  - 병렬 컴퓨팅 parallel computing
- 주의: Linux에서는 실패할 수 있음



```
Windows PowerShell
C:\cuda-work> .\04c-hello-parallel.exe
hello CUDA 0 !
hello CUDA 1 !
hello CUDA 2 !
hello CUDA 3 !
hello CUDA 4 !
hello CUDA 5 !
hello CUDA 6 !
hello CUDA 7 !
C:\cuda-work>
```

# CUDA header files

- **#include <cuda.h>**
  - 가장 기본적인 header file
- **#include <cuda\_runtime\_api.h>**
  - compiler 사용시, runtime function을 사용 가능
  - API = application programming interface (보통, 함수 정의를 포함)
- **#include <cuda\_runtime.h>**
  - 추가로, CUDA built-ins, types 사용 가능
- **nvcc 사용시,**
  - compiler 가 **자동으로 include** 시키는 경우가 많음



# CUDA header files

- **#include <helper\_cuda.h>**
- **#include <helper\_functions.h>**
  - not CUDA standard header file
  - provided by CUDA samples
  - you can find them in CUDA samples / inc directory

# CUDA function rules

- 모든 CUDA 함수는 “**cuda**”로 시작
- 대부분은 에러 코드 error code를 리턴 (성공시는 **cudaSuccess**).

- **Example:**

```
if ( cudaMalloc( &devPtr, SIZE ) != cudaSuccess ) {  
    exit(1);  
}
```

# cudaDeviceSynchronize

- `cudaError_t cudaDeviceSynchronize( void );`
  - wait for compute device to finish.
  - Blocks until the device has **completed** all preceding requested tasks.
  - returns: `cudaSuccess`
- 즉, 모든 **CUDA device** 들이 주어진 일을 **완료(finish)**할 때까지 기다림
  - Linux에서는 필요한 경우가 많음

# CUDA 세대 구분

- **CUDA hardware 구조가 계속 발전해 왔음**
  - 새로운 NVIDIA 그래픽 카드 → 새로운 CUDA 세대 generation
- **CUDA compute capability**
  - 1.0, 1.1, 1.2, 1.3 – Tesla
  - 2.0, 2.1 – Fermi
  - 3.0, 3.2, 3.5, 3.7 – Kepler
  - 5.0, 5.2, 5.3 – Maxwell
  - 6.0, 6.1, 6.2 – Pascal
  - 7.0, 7.2 – Volta
  - **7.5 – Turing** (GeForce RTX 2070)
  - 8.0, 8.6 – Ampere
  - 9.0 – Hopper

# 그래픽 카드 GPU 체크 계속

- NVIDIA CUDA zone 방문
  - 구글에서 “CUDA capable GPU list”

developer.nvidia.com > cuda-gpus ▾ 이 페이지 번역하기

## CUDA GPUs | NVIDIA Developer

Are you looking for the compute **capability** for your **GPU**, then started with **CUDA** and **GPU** Computing by joining our free-to-join NVIDIA Developer Program. Check the **list** above to see if your **GPU** is on it. If it is ...

이 페이지를 3번 방문했습니다. 최근 방문 날짜: 20. 1. 24

### CUDA Legacy GPUs


CUDA Legacy GPUs. Find the compute capability of the latest ...

[nvidia.com 검색결과 더보기 »](#)


### Data Science

NVIDIA-Data Science workstations

### CUDA-Enabled Tesla Products



### CUDA-Enabled Quadro Products




### Quadro Desktop Products


GPU	Compute Capability
Quadro RTX 8000	7.5
Quadro RTX 6000	7.5
Quadro RTX 5000	7.5
Quadro RTX 4000	7.5
Quadro GV100	7.0
Quadro GP100	6.0

# CUDA Compute Capability Check

- visit <https://developer.nvidia.com/cuda-gpus>

captured by biztripcru@gmail.com

 **NVIDIA DEVELOPER** [HOME](#) [BLOG](#) [FORUMS](#) [DOCS](#) [DOWNLOADS](#) [TRAINING](#) [Q](#) [ACCOUNT](#)

 **CUDA-Enabled GeForce and TITAN Products**

### GeForce and TITAN Products

GPU	Compute Capability
GeForce RTX 3090	8.6
GeForce RTX 3080	8.6
GeForce RTX 3070	8.6
NVIDIA TITAN RTX	7.5
Geforce RTX 2080 Ti	7.5

### GeForce Notebook Products

GPU	Compute Capability
Geforce RTX 2080	7.5
Geforce RTX 2070	7.5
Geforce RTX 2060	7.5
GeForce GTX 1080	6.1
GeForce GTX 1070	6.1

# NVCC 의 딜레마

- **NVCC – NVIDIA CUDA 컴파일러 compiler**
  - CUDA GPU 코드 생성 code generation
  - GPU 를 직접 구동하는 기계어 명령 machine instruction 생성
- **가능한 선택 – 모든 GPU를 지원하자**
  - NVCC 가 모든 compute capability 별로 code generation
  - 단점: 컴파일 시간 증가, exe 파일 크기 증가
- **현실적 대안 – PC에 장착된 NVIDIA 카드에 맞추자**
  - NVCC: `-gencode` 옵션 제공

# Linux NVCC 설정

- 예제: **NVIDIA GeForce RTX 2070 그래픽 카드**
  - compute capability 7.5
- **NVCC option**
  - `-gencode=arch=compute_75,code=\"sm_75,compute_75\"`  
`-arch=sm_75`
  - read the NVCC manual, for more details
  - set it in your “**makefile**” (or other equivalents)



# hello-linux.cu

```
#include <stdio.h>

__global__ void hello( void ) {
    printf( "hello CUDA #%%d!\n", threadIdx.x );
}

int main( void ) {
    hello<<<1,8>>>>();
    #if defined(__linux__)
        cudaDeviceSynchronize();
    #endif
    return 0;
}
```

- Linux 용으로 수정한 version

# hello-linux.cu

- 실행 결과

```
linux/cuda-work > make 04d-hello-linux.exe
nvcc -gencode=arch=compute_75,code=\"sm_75,compute_75\" -arch=sm_75 -O2 -o 04d-hello-linux.exe 04d-hello-linux.cu
linux/cuda-work > ./04d-hello-linux.exe
hello CUDA #0!
hello CUDA #1!
hello CUDA #2!
hello CUDA #3!
hello CUDA #4!
hello CUDA #5!
hello CUDA #6!
hello CUDA #7!
linux/cuda-work > █
```

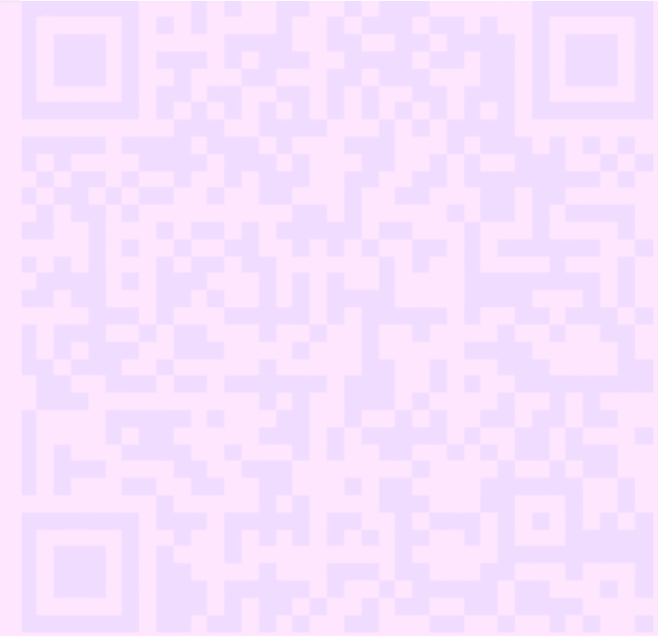
## 내용 contents

- **CUDA 프로그램을 좀더 향상시킨다**
  - Linux 에서도 돌아가게 한다
  - nvcc option을 설정한다

# Hello, Linux

폰트 끝단 일치 → 큰 교자 타고 혼례 치른 날  
정참판 양반댁 규수 큰 교자 타고 혼례 치른 날  
정참판 양반댁 규수 큰 교자 타고 혼례 치른 날  
본고딕 Noto Sans KR

© 2021-2022. biztripcru@gmail.com. All rights reserved.  
모든 저작권은 biztripcru@gmail.com 에게 있습니다.



The quick brown fox jumps over the lazy dog  
The quick brown fox jumps over the lazy dog  
The quick brown fox jumps over the lazy dog  
Source Sans Pro

Mathematical Notations  $O(n \log n)$   
Source Serif Pro