

# C++: A simple program

## Outline

- Source Code
- Compiling and Running (no IDE)
- Debugging (no IDE)
- IDE and compiler interaction
- Compiling, Running and Debugging with IDE

# Source Code – hello.cpp

```
// a small C++ program
#include <iostream> ←

int main() ←
{
    std::cout << "Hello, world!" << std::endl;
    return 0; ←
}
↑
```

# Something Different– header files

- Python and Java
  - classes are all in 1 file
  - import statements used to include references to classes from libraries
- C++
  - classes are in 2 files (.cpp and .h)
  - Include files reference a library (or object file)- linker includes it in executable
- C++ is more difficult to use in this respect

```
import matplotlib.pyplot as plt
```

```
import java.lang.String;
```

```
#include <string>
```

## Outline

- Source Code
- Compiling and Running (no IDE)
- Debugging (no IDE)
- IDE and compiler interaction
- Compiling, Running and Debugging with IDE

# Compilers

- see [https://en.wikipedia.org/wiki/List\\_of\\_compilers#C.2B.2B\\_compilers](https://en.wikipedia.org/wiki/List_of_compilers#C.2B.2B_compilers)

Compiler	Author	Windows	Unix-like	Other OSs	License type	IDE?	Standard conformance		
							C++11	C++14	C++17
C++Builder	Embarcadero (CodeGear)	Yes	OS X, iOS <sup>[2]</sup>	No	Proprietary	Yes	Yes/No	Yes/No	Yes/No
							(Supported via Clang. <sup>[3]</sup> )		
Turbo C++ Explorer	Embarcadero (CodeGear)	Yes	No	No	Freeware	Yes	?	?	?
C++ Compiler	Embarcadero (CodeGear)	Yes	No	No	Freeware	No	?	?	?
CINT	CERN	Yes	Yes	BeBox, DOS, Convex, etc.	X11/MIT	Yes	?	?	?
Borland C++	Borland (CodeGear)	Yes	No	DOS	Proprietary	Yes	No	No	No
Turbo C++ for DOS	Borland (CodeGear)	No	No	DOS	Proprietary	Yes	No	No	No
Clang	LLVM Project	Yes	Yes	Yes	BSD-like	Xcode, QtCreator (optional)	Yes	Yes	Partial
CodeWarrior	Metrowerks	Yes	Yes	Yes	Freeware	Yes	?	?	?
Comeau C/C++	Comeau Computing	Yes	Yes	Yes	Proprietary	No	No	No	No
CoSy compiler development system	ACE Associated Compiler Experts <sup>4</sup>	Yes	Yes	No	Proprietary	No	?	?	?
Digital Mars	Digital Mars	Yes	No	DOS	Proprietary	No	?	?	?
EDGE ARM C/C++	Mentor Graphics	Yes	Yes	Yes	Proprietary	Yes	?	?	?
Edison Design Group	Edison Design Group	Yes	Yes	Yes	Proprietary	No	Yes	Yes	Partial
GCC	GNU Project	MinGW, Cygwin	Yes	Yes	GPLv3	QtCreator, Kdevelop, Eclipse, NetBeans, Code::Blocks, Geany	Yes <sup>[4]</sup>	Yes	Yes
Visual C++	Microsoft	Yes	can target Linux, OS X, Android and iOS (since VS 2015)	No	Proprietary	Yes	Yes <sup>[5]</sup>	Yes	Incomplete

# Getting a compiler

- Visual C++ - comes with MS compiler

- GCC – depends on OS

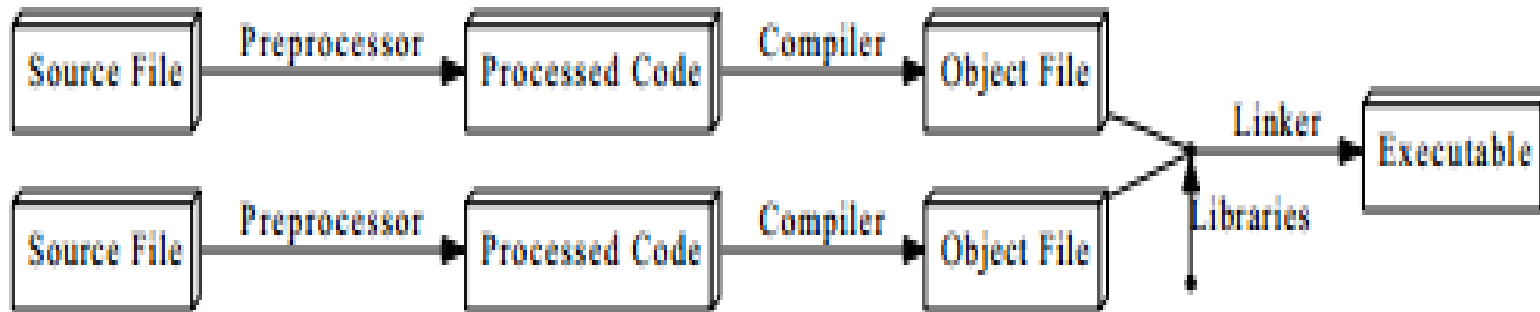
- Linux install build essentials to get GCC

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install build-essential
$ gcc -v
$ make -v
```

- Windows – minGW or Cygwin for GCC

- [http://www.mingw.org/wiki/HOWTO\\_Install\\_the\\_MinGW\\_GCC\\_Compiler\\_Suite](http://www.mingw.org/wiki/HOWTO_Install_the_MinGW_GCC_Compiler_Suite)
    - <https://www.cygwin.com/>

# Compiling/Linking - overview



**Source File** – .cpp .hpp .h files

**Preprocessor** – program that performs text substitution

**Compiler**- converts preprocessed source code to object code for a particular processor

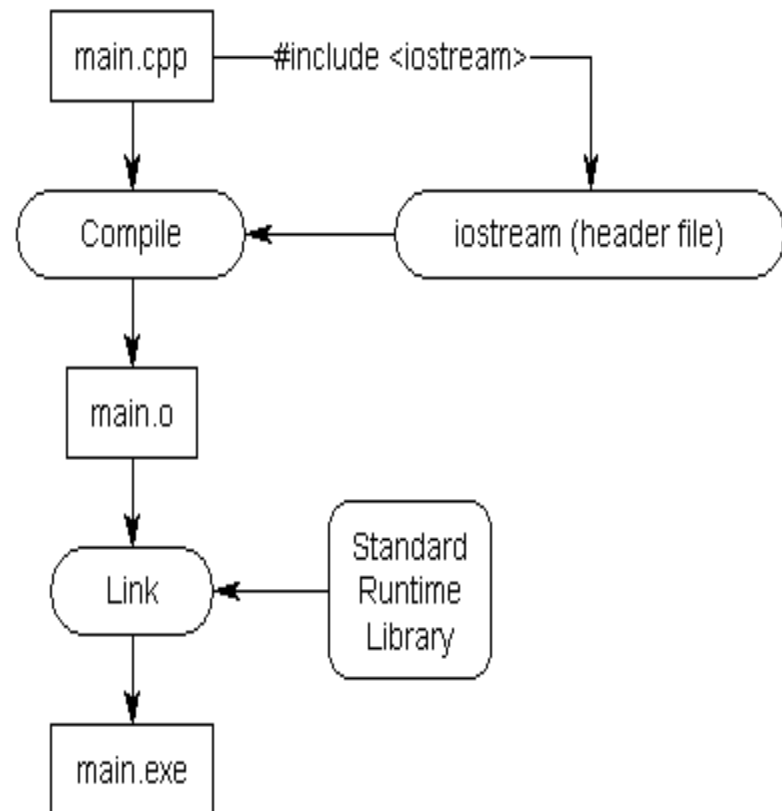
**Linker** – Links object files and external libraries to form exe (or library)  
Will always link the Cruntime and StandardLibrary



# Compiling/Linking

```
// a small C++ program  
#include <iostream>
```

```
int main()  
{  
    std::cout << "Hello, world!" << std::endl;  
    return 0;  
}
```

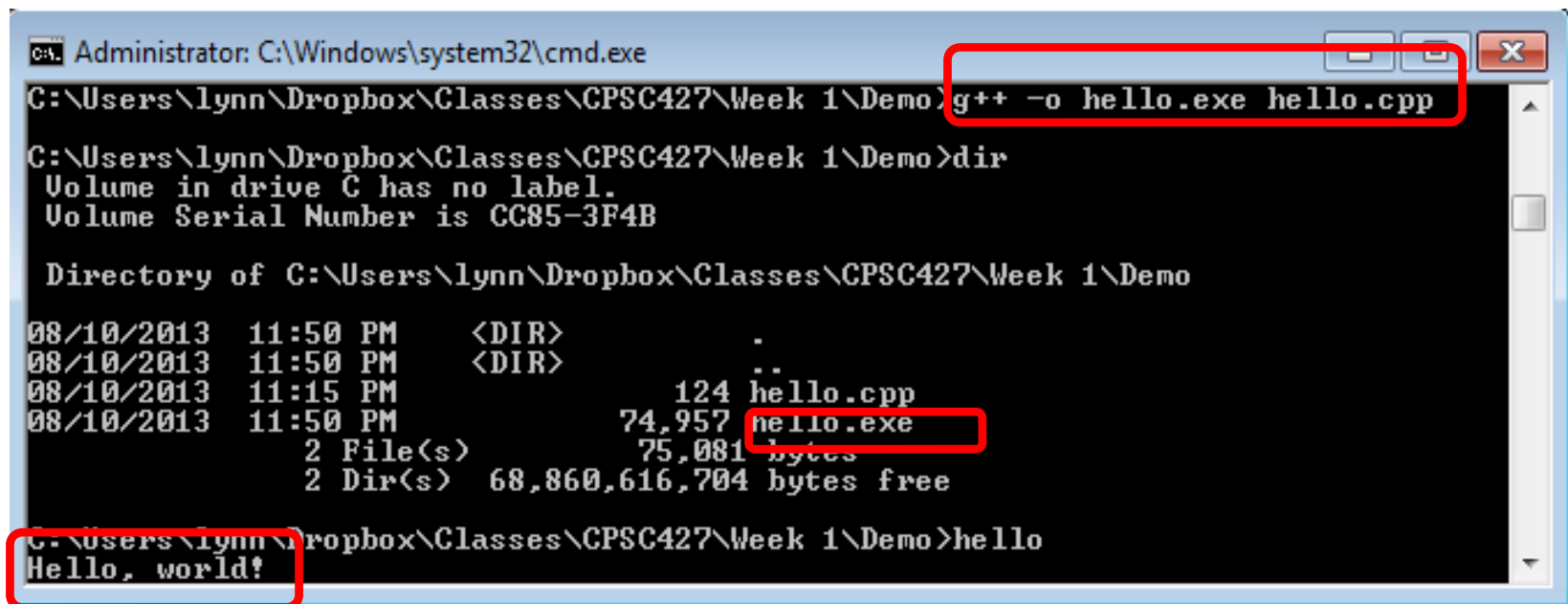


See [http://www.ntu.edu.sg/home/ehchua/programming/cpp/gcc\\_make.html](http://www.ntu.edu.sg/home/ehchua/programming/cpp/gcc_make.html) for more information

Diagram from <http://www.learncpp.com/cpp-tutorial/19-header-files/>

# Compiling/Linking – Example 1

- As simple as `g++ -o hello.exe hello.cpp`
- Can become very complex
- Commands reside in make file



The screenshot shows a Windows command prompt window titled "Administrator: C:\Windows\system32\cmd.exe". The current directory is `C:\Users\lynn\Dropbox\Classes\CPSC427\Week 1\Demo`. The command `g++ -o hello.exe hello.cpp` is entered and executed, with the command itself highlighted by a red rectangle. Below this, the `dir` command is used to list the files in the directory. The output shows a directory listing with columns for date, time, and file size. The file `hello.exe` is listed with a size of 74,957 bytes, and this line is also highlighted by a red rectangle. At the bottom, the command `hello` is entered and executed, with the output `Hello, world!` displayed, and this entire line is highlighted by a red rectangle.

```
C:\Users\lynn\Dropbox\Classes\CPSC427\Week 1\Demo>g++ -o hello.exe hello.cpp

C:\Users\lynn\Dropbox\Classes\CPSC427\Week 1\Demo>dir
Volume in drive C has no label.
Volume Serial Number is CC85-3F4B

Directory of C:\Users\lynn\Dropbox\Classes\CPSC427\Week 1\Demo

08/10/2013  11:50 PM    <DIR>          .
08/10/2013  11:50 PM    <DIR>          ..
08/10/2013  11:15 PM                124 hello.cpp
08/10/2013  11:50 PM            74,957 hello.exe
                2 File(s)              75,081 bytes
                2 Dir(s)  68,860,616,704 bytes free

C:\Users\lynn\Dropbox\Classes\CPSC427\Week 1\Demo>hello
Hello, world!
```

# Compiling/Linking – Example 2

- 2 source files; hello.cpp, myfunc.cpp
- 1 user defined header file myfunc.h. This file provides declarations (return type, function name, parameters) of functions in myfunc.cpp.

```
//hello.cpp
#include <iostream>
#include <string.h>
#include "myfunc.h"

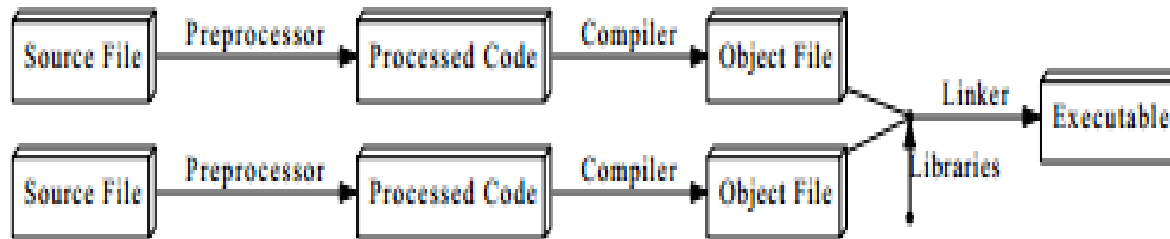
int main()
{
    std::string a = myfunc();
    std::cout << a << std::endl;
    return 0;
}
```

```
//myfunc.h
#include <iostream>
std::string myfunc();
```

```
//myfunc.cpp
#include "myfunc.h"

std::string myfunc()
{
    return "hello world";
}
```

# Compiling/Linking – Example 2



```
Administrator: Command Prompt

C:\AA_Demo>g++ -c myfunc.cpp
C:\AA_Demo>g++ -c hello.cpp
C:\AA_Demo>g++ -o hello.exe myfunc.o hello.o
C:\AA_Demo>dir
08/30/2013  12:15 AM                487 hello.cpp
08/30/2013  09:53 AM            28,033 hello.exe
08/30/2013  09:53 AM             1,927 hello.o
08/30/2013  01:45 AM              89 myfunc.cpp
08/30/2013  01:22 AM             427 myfunc.h
08/30/2013  09:52 AM             1,726 myfunc.o
        6 File(s)              32,689 bytes
        2 Dir(s)  122,903,212,032 bytes free

C:\AA_Demo>hello
hello world

C:\AA_Demo>
```

# Compiling/Linking – Example 2

```
//hello.cpp
#include <iostream>
#include <string.h>
#include <iostream>
std::string myfunc();
int main()
{
    std::string a = myfunc();
    std::cout << a << std::endl;
    return 0;
}
```

Preprocessor  
inserts myfunc.h  
here, expands all  
other includes

```
//myfunc.cpp
#include <iostream>
std::string myfunc();
std::string myfunc()
{
    return "hello world";
}
```

↓  
hello.o

Compiler compiles preprocessed  
Pure c++ files to object files, sets aside  
Memory that calls function myfunc() that  
Returns a string

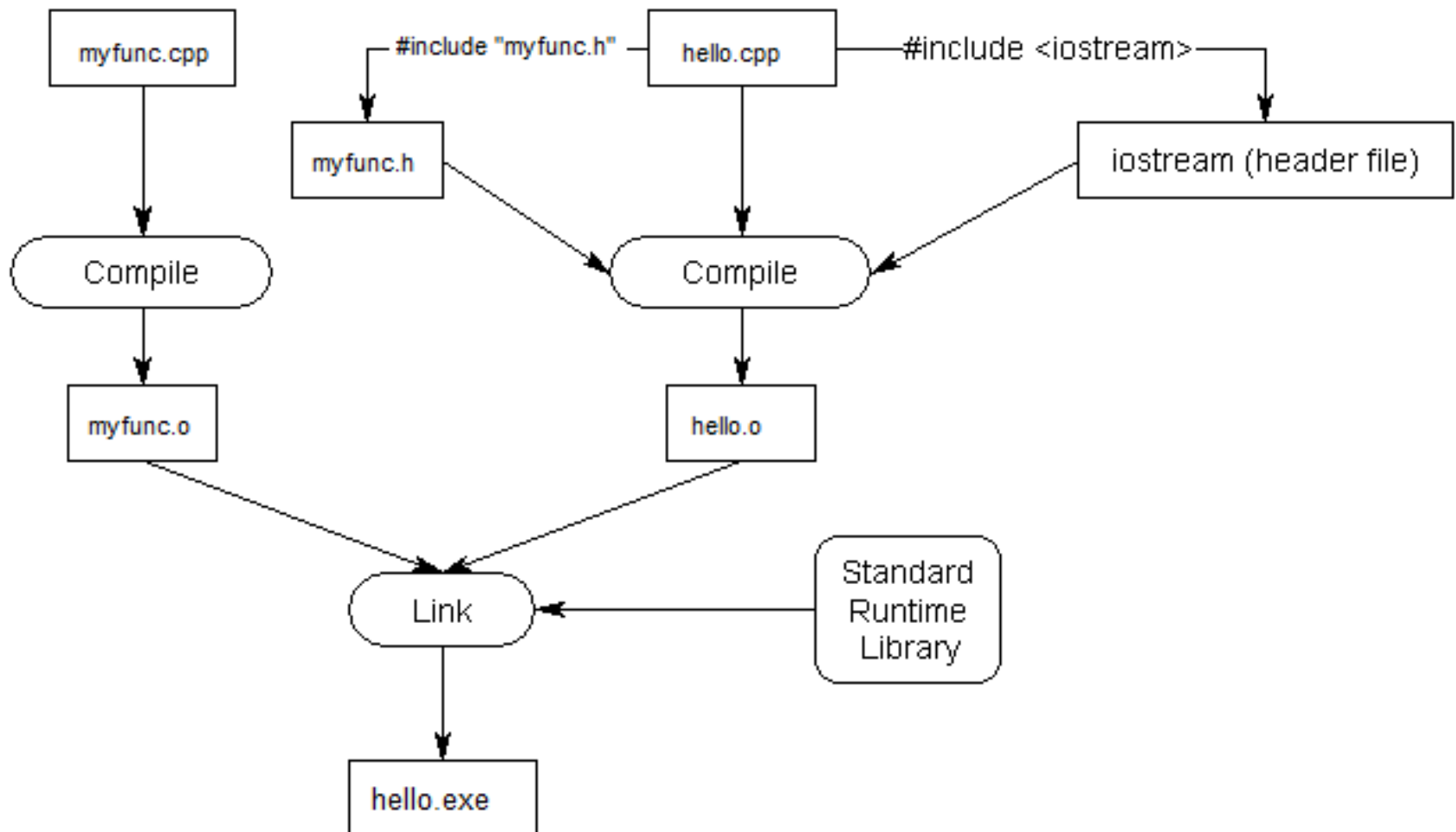
↓  
myfunc.o

Linker links o files and standardlibrary  
to single contiguous exe file. Resolves  
call in main to myfunc() with address

Standard library

↓  
hello.exe

# Compiling/Linking – Example 2



# Compiling/Linking – Makefiles

- Typing all these commands is error prone
- Instead use 'make'. A framework for building simple to very complex programs.
- Example: makefile from 1\_gcc\_demo

```
CFLAGS = -Wall
#target exe
all: hello.o myfunc.o
    g++ $(CFLAGS) -o myexe hello.o myfunc.o

#rebuild if either of the files below change
hello.o: hello.cpp myfunc.h
    g++ $(CFLAGS) -c hello.cpp

#rebuild if either of the files below change
myfunc.o: myfunc.cpp myfunc.h
    g++ $(CFLAGS) -c myfunc.cpp

#type 'make clean' to remove following
clean:
    rm -f *.o myexe.exe
```

- See online makefile tutorial

## Outline

- Source Code
- Compiling and Running (no IDE)
- Debugging (no IDE)
- IDE and compiler interaction
- Compiling, Running and Debugging with IDE



# Debugging

```

Perkins@R343-M1 C:\test
$ g++ -g main.cpp
Perkins@R343-M1 C:\test
$ gdb a.exe
(gdb) break main
Breakpoint 1 at 0x1004010ed: file main.cpp, line 5.
(gdb) run
Starting program: /cygdrive/c/test/a.exe
[New Thread 7128.0x1ac8]
[New Thread 7128.0x670]
[New Thread 7128.0x1640]
[New Thread 7128.0x1e8c]

Breakpoint 1, main () at main.cpp:5
5         std::cout<<"hello world"<<std::endl;
(gdb) list
1      #include <iostream>
2
3      int main()
4      {
5          std::cout<<"hello world"<<std::endl;
6          int a=1;
7          int b=a+1;
8          return 0;
9      }(gdb) n
hello world
6          int a=1;
(gdb) n
7          int b=a+1;
(gdb) a
Ambiguous command "a": actions, add-auto-load-safe-p
(gdb) print a
$1 = 1
(gdb)

```

-g compile with debug info

start debugger  
break at beginning

run

Show lines around breakpoint

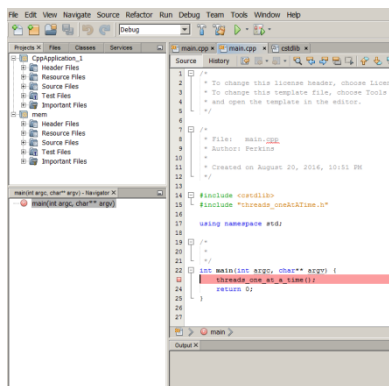
Next line

Print value of a

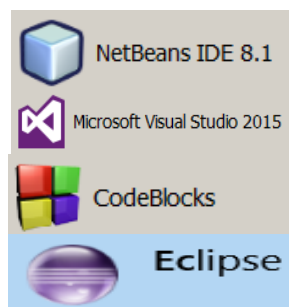
## Outline

- Source Code
- Compiling and Running (no IDE)
- Debugging (no IDE)
- IDE and compiler interaction
- Compiling, Running and Debugging with IDE

# IDE and compiler interaction



Integrated Development environment (IDE)  
Such as...



But an IDE makes it easier  
Especially on large projects

IDE uses  
compiler

IDE uses  
debugger

Compiler  
(like gcc)

Debugger  
(like gdb)

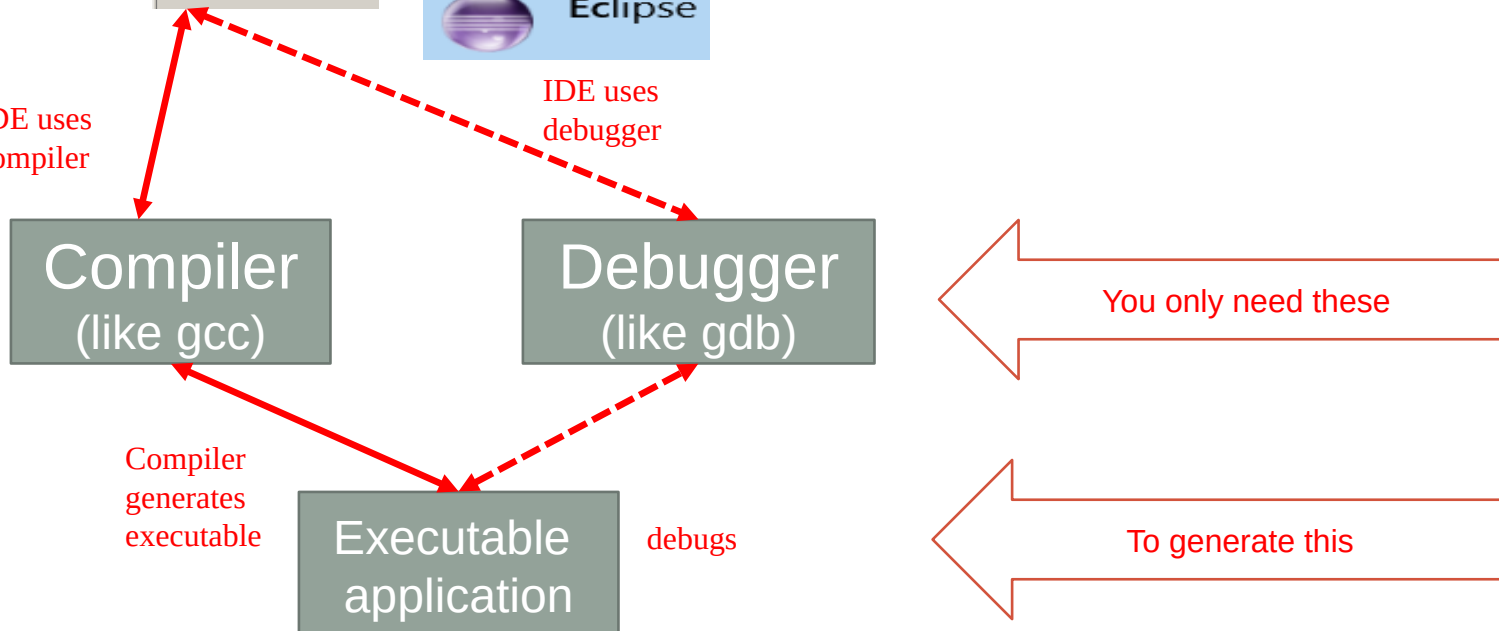
You only need these

Compiler  
generates  
executable

Executable  
application

debugs

To generate this




## Outline

- Source Code
- Compiling and Running (no IDE)
- Debugging (no IDE)
- IDE and compiler interaction
- Compiling, Running and Debugging with IDE

# Compiling/Linking – Using an IDE

- **Let Integrated Development Environment (IDE) handle all details**
- **(build settings still there just using default project settings)**
- Create C++ project
- Copy 3 files from example 2 to it
- Build it
- Here are some key shortcuts

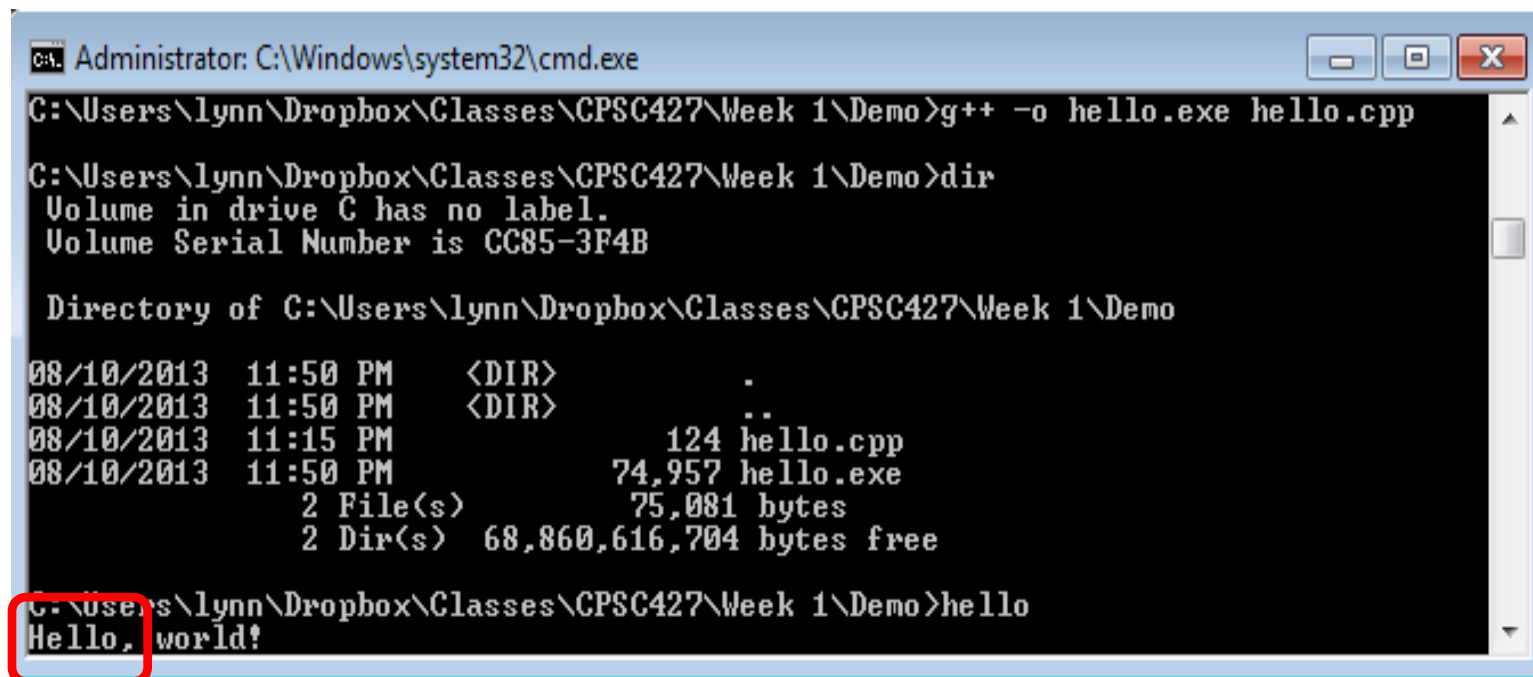


+		...	🗑
CTRL .	Next error		
CTRL - O	outline		
F3	Definition		
CTRL+SHIFT+O	Fix Imports		
ctrl-space	autocomplete		
CTRL+SHIFT+F	Format code		
CTRL+_,{	split horiz/vert		
CTRL + =,-	zoom in/out		
CTRL +	F11 Run		
F11	Debug		
f5,f6,f8	in/over/res		
CTRL+SHIFT+L	list shortcuts		

Key bindings I use

# Running

- Its an Executable! (no virtual machine)
- Can run from command line or IDE
- Fast Demo Various bits of IDE



A screenshot of a Windows command prompt window titled "Administrator: C:\Windows\system32\cmd.exe". The window shows the following commands and output:

```
C:\Users\lynn\Dropbox\Classes\CPSC427\Week 1\Demo>g++ -o hello.exe hello.cpp

C:\Users\lynn\Dropbox\Classes\CPSC427\Week 1\Demo>dir
Volume in drive C has no label.
Volume Serial Number is CC85-3F4B

Directory of C:\Users\lynn\Dropbox\Classes\CPSC427\Week 1\Demo

08/10/2013  11:50 PM    <DIR>          .
08/10/2013  11:50 PM    <DIR>          ..
08/10/2013  11:15 PM                124 hello.cpp
08/10/2013  11:50 PM            74,957 hello.exe
                2 File(s)              75,081 bytes
                2 Dir(s)  68,860,616,704 bytes free

C:\Users\lynn\Dropbox\Classes\CPSC427\Week 1\Demo>hello
Hello, world!
```

The output "Hello, world!" is highlighted with a red rectangle.

# What have we learned

C++ has lots of similarities to Java (more as we go) and some with Python

How to write a simple C++ program

How to compile using command line

How to use an IDE to create a program

**For this class and most likely professionally, let the IDE manage your builds.**

Basic IDE usage (Debug/release build, variables, breakpoints etc)

How to run a program

**PRACTICE PLEASE**