CS 162
Intro to Computer Science II

Separate Compilation
Compilation

```
#include <iostream>
int main() {
  std::cout << "Hello world" << std::endl;
  return 0;
}
```
Executable Files

Compile with:

\texttt{g++ program1.cpp}

Executable file is the default, \texttt{a.out}

Or

\texttt{g++ program1.cpp -o program1}

Executable file is named \texttt{program1}
Separate Compilation

• Create separate object files:
  
g++ program1.cpp
  
-o program1.o

• Logical parts of your program are in different files
  – Only need to change the file
  – Only recompiles the changed object files

• The linker brings together all the necessary files
Header Files

• Separate implementation from the interface

• Example
  – Definition for square() goes in functions.cpp
  – The function prototype (definition) goes into functions.hpp
  – Must include functions.hpp in functions.cpp
Example

functions.cpp
#include “functions.hpp”

int square(int x) {
    return x * x;
}

functions.hpp

int square(int x)
Extended Example

program1.cpp

#include “functions.hpp”

int main() {
    std::cout << square(12) << std::endl;
    return 0;
}
Extended Example

Compile with:

```
G++ program1.cpp functions.cpp -o program1
```

Produces an executable file called program1
Extended Example

To create an object file, just the compilation of one file use:

```
g++ functions.cpp -c -o functions.o
```

functions.o is not executable!

This command will pull the parts together to make an executable file:

```
g++ program1.cpp functions.o -o program1
```
Include- issue

• What if you use `#include foo.hpp` in 10 implementation files?
  – The code for `foo.cpp` gets included in your executable file 10 times!

• Use guards to prevent this
  – What’s a guard?
    ```
    ifndef FOO_HPP
    define FOO_HPP
    ...
    endif
    ```
Guard Example

functions.hpp

```c
#ifndef FUNCTIONS_HPP
#define FUNCTIONS_HPP

int square(int x);

#endif // FUNCTIONS_HPP
```

OSU Oregon State University
Modularization

• Separate compilation organizes your code
  – Each logical part of the program should be in a separate file
  – Other programmers do not need to see the “innards” of your code

• You should see classes as an extension of this concept
  – One of the driving forces for the development of object-oriented programming!