Introduction to C++

C and C++

- C was developed for doing system level programming on Unix (OS kernel, device drivers, networking software, etc.)
- C's language constructs map directly to native hardware operations, making it possible to write very efficient programs
- C++ is an object-oriented derivative of C that supports classes, inheritance, polymorphism, etc.
- C is a subset of C++
- Historically, C and C++ have been used for both system and application programming
- Application programming is moving away from C and C++ to languages such as Java, C#, Visual Basic, etc.
- System programming is still dominated by C and C++
- Performance-critical parts of applications are commonly implemented in C or C++

Simple C++ Program

simple.cpp

```
#include <iostream>
using namespace std;

int main() {
   cout << "Every age has a language of its own\n";
   return 0;
}
```

Compiling and Running

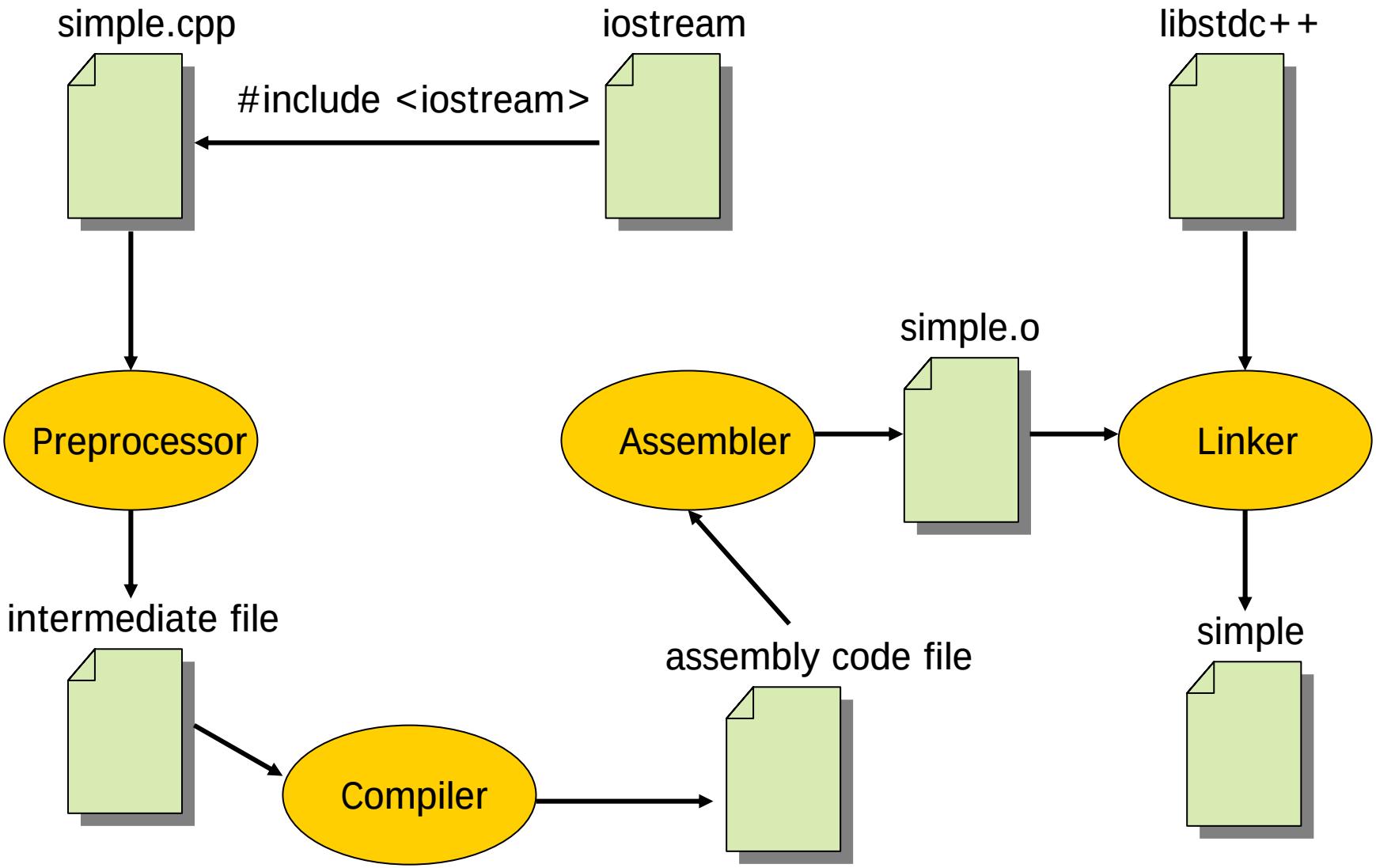
- Use `g++` to compile the program

```
$ g++ -o simple simple.cpp
```

- The -o option tells `g++` what the name of the executable file should be
  - The default name is a.out
- We then list the names of one or more `C++` files that contain the program's source code
  - One of these files must contain the main function
- After compiling, we are ready to run the program

```
$ ./simple
Every age has a language of its own
$
```

The Compilation Process



Demonstration of the phases of compilation

- View the output from the preprocessor
  - g++ -E simple.cpp
- View the output from the compiler (assembly code)
  - g++ -S simple.cpp
  - Output in simple.s
- View output from the assembler (object code)
  - g++ -c simple.cpp
  - Output in simple.o
- View the output from the linker (executable file)
  - g++ -o simple simple.cpp
  - Output is in simple

Preprocessor directives

- #include
  - #include <iostream>
- #define
  - #define SIZE 26
  - #define MAX(a, b) ((a > b) ? a : b)
- Demo - preprocess.cpp

Compiler Warnings

- Turn on all of the compiler's warnings in order to find possible bugs
  - g++ -Wall simple.cpp

- Demo - warnings.cpp

Defining constants in C++

- #define SIZE 26
- const int SIZE = 26;
- enum MONTH {JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC};
- Using const is preferred over using #define
  - debuggers keep track of consts

Command Line Arguments

- You can pass command line arguments into your program and use them in the program
- int main(int argc, char * argv[])
- argc = # of command line arguments (including the name of the executable)
- argv[0] = name of the executable
- argv[1] = first argument given
- etc.
- Demo - echo.cpp