C++ Mechanics

CS 150 – C++ Programming I
Chapter 1

"Hello World" the C Manual

1.1 Getting Started

The only way to learn a new programming language is to write programs in it. The first program to write is the same for all languages:

Print the words

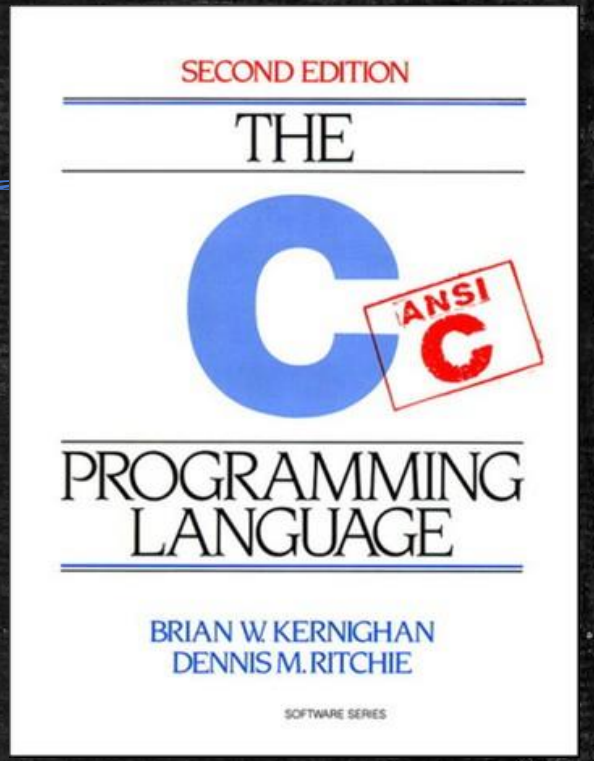
```
hello, world
```

This is the big hurdle; to leap over it you have to be able to create the program text somewhere, compile it, load it, run it, and find out where your output went. With these mechanical details mastered, everything else is comparatively easy.

In C, the program to print "`hello, world`" is

```
#include <stdio.h>

main() {
    printf("hello, world");
```

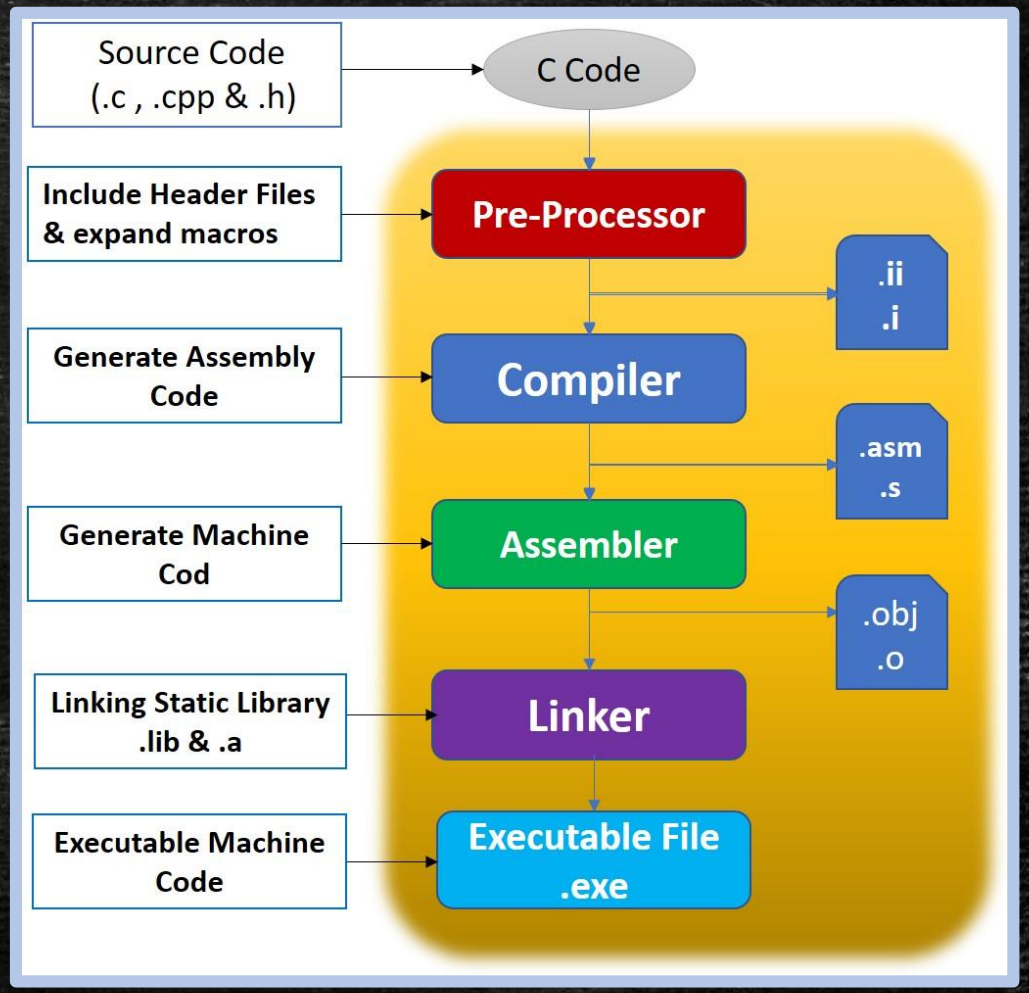


Hello World, C++ Style

- Let's write a C++ version of Hello World

C++ Mechanics

- Editor: C++ source-code
  - Extension `.cpp` (`.c` for C-language files)
- Driver (clang++, g++, CL) processes source
  - Preprocessing modifies source code
  - Parsing syntax, declaration & type errors
  - Generation produces assembly language
  - Assembler produces object-code
  - Linker combines object-code, library code and startup code to create executable
- Exercise: Open a new sandbox
  - Create `hello.cpp`



Simplest C++ program

- Smallest C++ program has one function `main()`
  - `main()` entry point called by startup code
  - Startup code added by the linker to initialize memory and set machine registers correctly
  - CS 150 "framework", `main()` calls: `run(…)`
- Function header (or interface): `int main()`
  - The kind of value produced by the function
  - The name of the function
  - Kind of arguments passed (if any)
- Function body (implementation) : `{ }`

Printing Output

- Python: use the `print()` function
  - Built-in to the language
  - Multiple arguments, adds a newline, no semicolon
- Java: use `System.out.print()` method
  - Standard library, not language, auto loaded object
  - Single argument, `println()` for newline, semicolon
- C++: use `cout` object and `<<` insertion operator
  - Library, not language, not auto loaded, in namespace
  - Multiple arguments, multiple operators, semicolon
  - Manually add newline with `\n` or `endl` object

Functions, Methods, Operators

- Function: a named piece of code that may take arguments and returns a value (or does an action)
  - `a = sqrt(52.5);`
- Member Function (method in Java): a function, defined in a class, that takes an implicit reference to an object as its first argument.
  - `n = str.length(); -> n = length(str);`
- Overloaded operator: a function with a special name, called using infix notation
  - `cout << "hello"; -> operator<<(cout, "hello");`

Standard Library and Preprocessor

- I/O part of standard C++ library NOT language
  - Not auto-included like `java.lang` package in Java
- Use library features by `#including` header
  - `#include <iostream>` to insert declarations and definitions from standard lib `iostream` header
  - More than 70 headers in C++11
- Lines that start with # are preprocessor instructions
  - Replaced with actual text of header file
- Exercise: `g++ -E hello.cpp -o hello.ii`
  - Pre-processed source code is called a translation unit

Namespaces

- Namespaces "group" related classes and functions
  - Similar to Math class in Java
- Standard library functions & classes are in `namespace std`:
- 1. Explicitly qualify the object or function name
  - Preface with scope-resolution operator `::`
  -
```
#include <cmath> // std namespace
double ans = std::sqrt(5.25);
```
- 2. Employ a using declaration: `using std::sqrt;`
- 3. A using directive: `using namespace std;`
  - Fine for single implementation files

Building and Running your Program

- Step 2 – Code Generation: `g++ -S hello.ii`
  - Converts preprocessed translation unit to assembly language
- Step 3 - Assembly: `g++ -c hello.s`
  - Converts assembly language into object (machine) code
  - Type `xxd hello.o` to see the machine code
- Step 4 – Link Object Code: `g++ hello.o -o hello`
  - Output of the linker: executable code
- Quite a few steps, so Linux has a tool that runs them automatically. Just type: `make hello` instead

Different Kinds of Errors

- Compiler Errors: discovered during parsing
  - Type errors: trying to store wrong value in variable
  - Syntax errors: forgetting semicolon, closing quote …
  - Declaration errors: using an unknown name
- Linker Errors: discovered during linking phase
  - Cannot find object referred to in compiled files
  - Missing main, missing file, etc.
- Undefined Behavior: logic error; output unknown
  - Runs OK? ONLY ACCIDENTALLY
  - Example: using a name without correct header