Lecture 1b: Compiling C++; Style

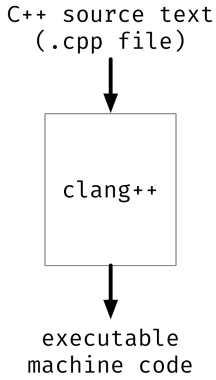
CS 70: Data Structures and Program Development

January 23, 2020

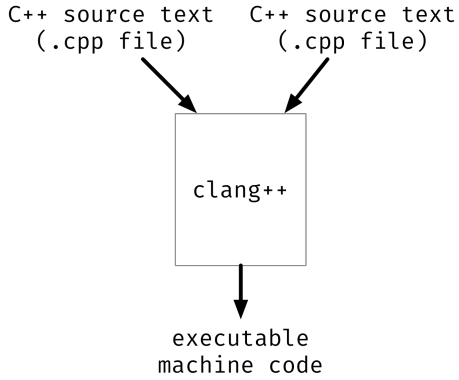
Learning Goals

- I can explain the steps to compile multi-file C++ code.
- I can contrast the design goals of Java and C++.
- I can identify code with bad style.
- I can reason about readable and elegant C++ code.

Compiling C++ (1 file)

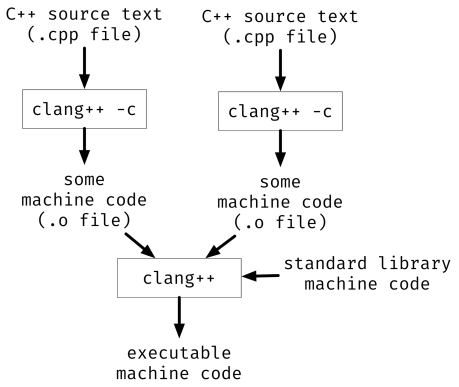


Compiling C++ (multiple files)



Compiling C++ (multiple files, better!)

Compiling C++ (multiple files, better!)



Exercise

1. **Suppose our program has three `.cpp` source files. Assume nothing has been compiled yet. To get a runnable program, how many times should the `clang++` command run?**

2. **Suppose now we change one definition in one `.cpp` file. To get an updated runnable program, how many times should `clang++` run?**

System Header Files

```
#include <iostream>
#include <string>

int main() {
  std::string message = "Hello, World!";
  std::cout << message << "\n";
  return 0;
}
```


User Header Files

```
------------------ exclaim.hpp ----------------
#include <string>

// Adds an !
std::string exclaim(std::string sentence);

----------------------main.cpp-----------------
#include <iostream>

#include "exclaim.hpp"

int main() {
  std::cout << exclaim("wow") << std::endl;
}
```


User Header Files (continued)

```
------------------ exclaim.hpp ----------------
#include <string>

// Adds an !
std::string exclaim(std::string sentence);


------------------ exclaim.cpp ----------------
#include "exclaim.hpp"
#include <string>

std::string exclaim(std::string sentence) {
  return sentence + "!";
}
```

I changed `exclaim.cpp`. Steps to recompile?

```
------------------ exclaim.cpp ----------------
#include "exclaim.hpp"
#include <string>

// Don't add ! if the string ends in !
std::string exclaim(std::string sentence) {
  size_t length = sentence.size();
  if (length > 0 && sentence[length-1] == '!') {
      return sentence;
  } else {
      return sentence + "!";
  }
}
```


Design Principles for C++

C++ is a statically typed, object-oriented, imperative language.

Design goals for the language include:

- "C++ is a better C"
- Efficiency (in time and space) **No Overhead**
- Trust (i.e., obey) the programmer

Design Principles for Java

Java is a statically typed, object-oriented, imperative language.

Design goals for the language include:

- Safety (ensure bad things can't happen)
- Portability (Write Once, Run Everywhere)
- Familiarity

Exercise: C++ or Java?

For each described behavior, indicate whether it would result from a C++ program *or* from a Java program.

1. If you try to access index 100 of a 10-element array of integers, an error (exception) will be reported.

2. If you try to access index 100 of a 10-element array of integers, anything could happen (but you'll most likely get back some bits taken from memory past the end of the array, interpreted as an integer.)

Style and Elegance

- **Readability should be your top priority**
- **And all other things being equal,**
  - Maximize maintainability / extensibility
  - Don't be *gratuitously* inefficient
- **Nobody writes beautiful code all the time**
  - Go back and fix things!
- **Use Consistency and Inconsistency to your advantage**
  - Similar things should *look* similar
  - Different things should *look* different

We will ask you to generally follow Google's C++ Style Guide (as implemented in the `cpplint` tool)

Consistency: Variable Names

Variable names and functions: camelCase

- `count`, `i`, `activeTask`, `launchMissiles()`

Data members (fields): camelCase + trailing underscore

- `front_`, `currentCapacity_`

Class names: Capitalized CamelCase

- `Gene`, `StudentTranscript`

Constants: All caps, underscore between words

- `VERSION`, `MAX_STUDENTS`

Consistency: Applying Idioms

```
const size_t NUM_LETTERS = 26;
std::string alphabet;

for (size_t i = 0; i < NUM_LETTERS; ++i) {
    alphabet += ('a' + i);
}

for (size_t i = 0; i < alphabet.size(); ++i) {
    std::cout << alphabet[i] << " " << i << "\n";
}
```

Exercise: Consider this code example

- What is wrong with it? Why is this a problem?
- How would you suggest fixing the problem(s)?

```
// MUST be set to 1!
Params.ParentalLevel = 3; // QA now insists this be 2
```


Exercise: Consider this code example

- What is wrong with it? Why is this a problem?
- How would you suggest fixing the problem(s)?

```
// if ((typec!="20") && (typec!="13") &&
(typec!="5") && (typec!="4"))
if (typec !="20") {
    if (typec != "13") {
        if (typec != "5") {
            if (typec != "4") {
                selectType("ALLOC");
                return;
            }
        }
    }
}
```