Tutorial 2
Get familiar with G++ & Makefile and String & Stream Applications

Sep. 19, 2024

Fang Zihao (USTF)

(SDS, 122090106@link.cuhk.edu.cn)

Objectives today

1. Some background information about this course.
2. Concepts clarification
3. Get familiar with some common operations of string
4. Three examples with string and stream operations:
    1. Palindromes 回文数
    2. Acronym 首字母缩略词
    3. Integer or String? 整数还是字符串？
5. Learn to use g++ and Makefile to run the examples above.
6. Q&A time: Make sure you can run the C++ code on your computer now! (If you can’t, solve it today!)

1. Background

- **The course changed a lot in this semester. Now, you can:**
- **Use Qt Creator to write code. (Recommended)**
  - Most examples in our textbook has implementations with the Stanford Library.
  - It is difficult to embed the Library into other IDEs and there is a blank or empty project provided with the Library already in stalled.
  - So if you want to use Stanford Library, you probably need to use Qt.
- **Use VS Code to write code. (Highly Recommended)**
  - Familiar, Light-weight, easy to use.
  - Learn to use command lines and Makefile.
  - Allow writing and running single .cpp file without create a heavy “project”.
- **Use other IDEs like Visual Studio or CLion. (Optional)**
- **Use Vim or Emacs. (Recommended if you are an expert in programming)**

2. Concepts

- Compiler
  - GNU GCC (Recommended)
    - Acronym for GNU Compiler Collection (formerly GNU C Compiler), it includes front ends fro C, C++, Objective-C, Fortran etc.
    - Most widely used compiler.
    - In this course, we will use its C++ front end: g++.
  - LLVM Clang (Highly Recommended)
    - Cross-platform and more user friendly.
    - Compatible with GNU GCC.
    - In this course, we will use its C++ front end: clang++.
  - MSVC
    - Only available for Microsoft Windows
    - Heavy-weight and bulky.

2. Concepts

- Build System
  - GNU Make (Will be taught in tutorial)
    - Simpler build system for small or old projects.
  - CMake
    - More advanced build system for bigger projects.
  - Meson
    - Based on Python, which is easier to use.
    - More advanced build system for bigger projects.

3. Get familiar with some common operations of string

- Two types of String:
- C-style string (char array)
  - `char c[6] = {'h', 'e', 'l', 'l', 'o', '\0'};`
  - `char *c = "hello";`
- C++ std::string
  - `include<string>`
  - `std::string s = "hello";`
- Changing C-style string to std::string
  - `std::string s(c);`
  - `std::string s = c;`
- Changing std::sting to C-style string
  - `s.c_str();`

3. The <cctype> (ctype.h) Interface

This header declares a set of functions to **classify** and **transform individual characters**.

isalnum	checks if a character is alphanumeric (function)
isalpha	checks if a character is alphabetic (function)
islower	checks if a character is lowercase (function)
isupper	checks if a character is an uppercase character (function)
isdigit	checks if a character is a digit (function)
isxdigit	checks if a character is a hexadecimal character (function)
iscntrl	checks if a character is a control character (function)
isgraph	checks if a character is a graphical character (function)
isspace	checks if a character is a space character (function)
isblank (C++11)	checks if a character is a blank character (function)
isprint	checks if a character is a printing character (function)
ispunct	checks if a character is a punctuation character (function)
tolower	converts a character to lowercase (function)
toupper	converts a character to uppercase (function)

For more, please visit https://en.cppreference.com/w/cpp/header/cctype

3. The <cstring> (string.h) Interface

This header file defines several functions to **manipulate C strings** and **arrays**.

strcpy	copies one string to another (function)
strlen	returns the length of a given string (function)
strcmp	compares two strings (function)
strstr	finds the first occurrence of a substring of characters (function)
memset	fills a buffer with a character (function)
memcpy	copies one buffer to another (function)

For more, please visit https://en.cppreference.com/w/cpp/header/cstring

3. Operators on the String Class

- To convert the C++ string objects into C string literals, simply apply the `c_str` method to the C++ string.
- The operators are overloaded to support the following operations:

`str[i]` Returns the i^{th} character of `str`. Assigning to `str[i]` changes that.
`s1 + s2` Returns a new string consisting of `s1` concatenated with `s2`.
`s1 = s2;` Copies the character string `s2` into `s1`.
`s1 += s2;` Appends `s2` to the end of `s1`.
`s1 == s2` (and similarly for `<`, `<=`, `>`, `>=`, and `!=`) Compares to strings lexicographically.
`str.c_str()` Returns a C-style character array.

3. Operators on the String Class

`str.length()` Returns the number of characters in the string `str`.
`str.at(index)` Returns the character at position `index`; most clients use `str[index]`
`str.substr(pos, len)` Returns the substring of `str` starting at `pos` and continuing for `len`
`str.find(ch, pos)` Returns the first index ≥ `pos` containing `ch`, or `string::npos` if not found.
`str.find(text, pos)` Similar to the previous method, but with a string instead of a character.

3. Operators on the Stream Class

Formatted input	
`operator>>`	extracts formatted data (public member function of `std::basic_istream<CharT,Traits>`)
Unformatted input	
`get`	extracts characters (public member function of `std::basic_istream<CharT,Traits>`)
`peek`	reads the next character without extracting it (public member function of `std::basic_istream<CharT,Traits>`)
Formatted output	
`operator<<`	inserts formatted data (public member function of `std::basic_ostream<CharT,Traits>`)
Unformatted output	
`put`	inserts a character (public member function of `std::basic_ostream<CharT,Traits>`)
`write`	inserts blocks of characters (public member function of `std::basic_ostream<CharT,Traits>`)

For more, please visit https://en.cppreference.com/w/cpp/io/basic_iostream

4.1. Palindrome

- A *palindrome* is a word that reads identically backward and forward, such as “level” or “noon”.
- Write a C++ program `isPalindrome` that checks whether a string is a palindrome.

```
bool isPalindrome(string str) {
    int n = str.length();
    for (int i = 0; i < n / 2; i++) {
        if (str[i] != str[n - i - 1]) return false;
    }
    return true;
}
```

```
bool isPalindrome(string str) {
    return str == reverse(str);
}
```

we define this function to reverse a string.

- Efficiency vs. Readability

4.2. Acronym

- An *acronym* is a word formed by taking the first letter of each word in a sequence, as in

  `"self-contained underwater breathing apparatus"` ⟶ `"scuba"`

- Write a C++ program that generates acronyms, as illustrated by the following sample run:

```
zhangsan@zhangsandeMacBook-Pro ~/Desktop/USTF/OneDrive_1_2024-9-10/Progr
am1_Acronym @ ./acronym
Program to generate acronyms
Enter string: Graphical User Interface
The acronym is "GUI"
Enter string: International Collegiate Programming Contest
The acronym is "ICPC"
Enter string: Super Computing Conference
The acronym is "SCC"
```

4.2. Acronym

```
string acronym(string str) {
   string result = "";
   bool inWord = false;
   int nc = str.length();
   for (int i = 0; i < nc; i++) {
      char ch = str[i];
      if (inWord) {
         if (!isalpha(ch)) inWord = false;
      } else {
         if (isalpha(ch)) {
         result += ch;
         inWord = true;
         }
      }
   }
   return result;
}
```

4.3. Integer or String?

- You need to judge if the following input is an integer or a string, and perform different operations on them.
- Integers are preceeded by “int:” while strings are preceeded by “str:”.
- For integers, double them, and for strings, duplicate them.
- Each string DOES NOT contain any space to ease implementation.
- Example run:

```
zhangsan@zhangsandeMacBook-Pro ~/Desktop/USTF/OneDrive_1_2024-9-10/Program3_Integer_or_String
? [-2] @ ./integer_or_string?
This program tests for palindromes.
Enter in a valid format: int:123
The result is: 246
Enter in a valid format: str:123
The result is: 123123
Enter in a valid format: str:hello
The result is: hellohello
```

4.3. Integer or String?

```
string manipulate(string str) {
    istringstream iss(str.substr(4));
    if (str.substr(0, 4) == "int:") {
        int i;
        iss >> i;
        ostringstream oss;
        oss << i * 2;
        return oss.str();
    }
    else if (str.substr(0, 4) == "str:") {
        string s;
        iss >> s;
        s += s;
        return s;
    }
    else {
        throw "Invalid type";
    }
}
```

5.1. Run the code via command lines (terminal)

- If you don’t know how, or fail to set the VS Code makefile extension (settings.json), you can **ALWAYS** use **COMMAND LINES** to compile and run your C++ code.
- Only pre-requisite: you can run “make --version” and “g++ --version” in the command lines.

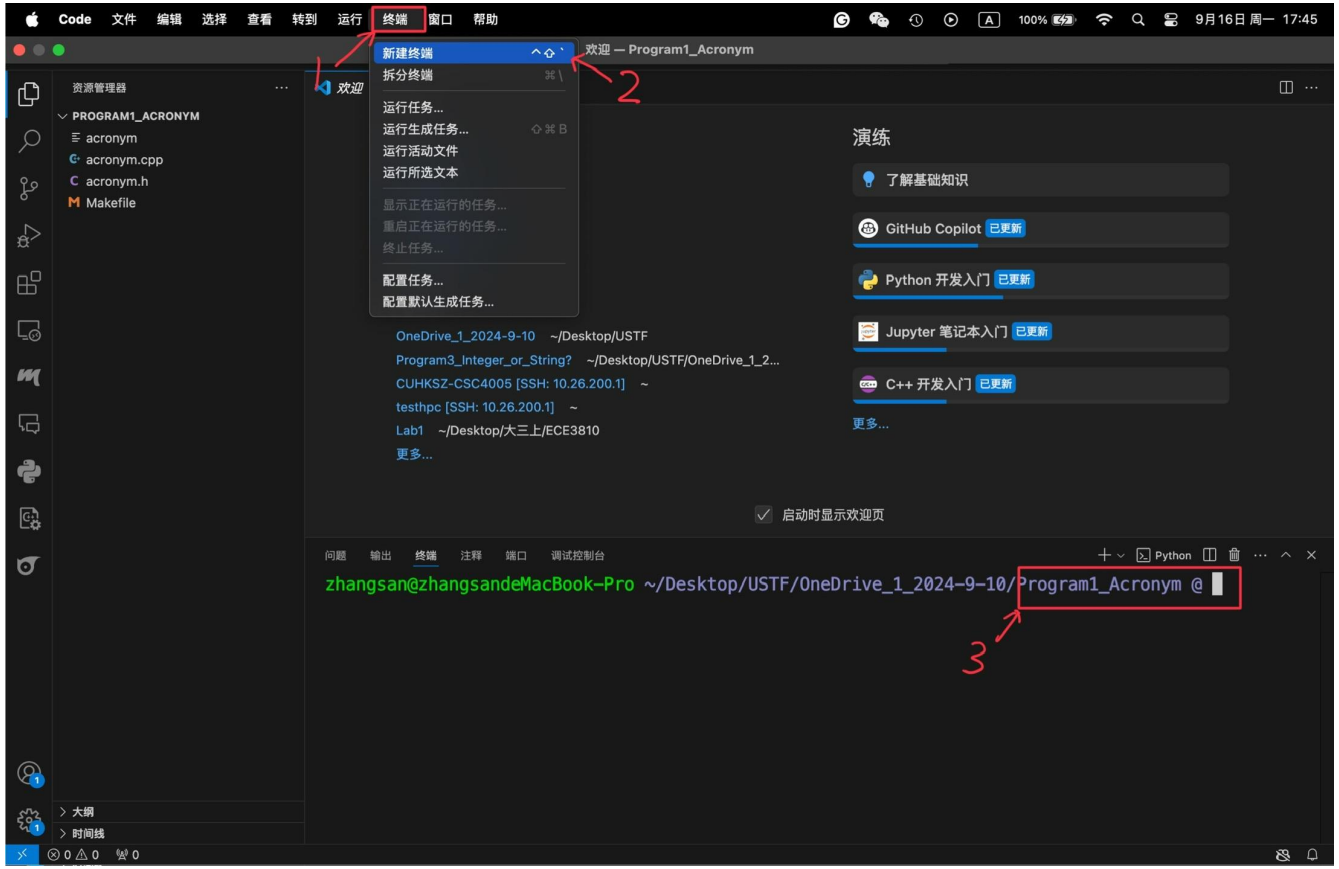
```
zhangsan@zhangsandeMacBook-Pro ~ @ g++ --version
Apple clang version 16.0.0 (clang-1600.0.26.3)
Target: arm64-apple-darwin24.0.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
zhangsan@zhangsandeMacBook-Pro ~ @ make --version
GNU Make 3.81
Copyright (C) 2006  Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.

This program built for i386-apple-darwin11.3.0
```

5.1. Run the code via command lines

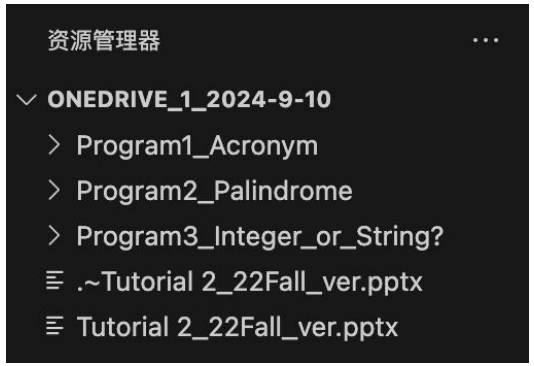
a) Compile by **pure** command lines, with “g++” compile command

- Open a terminal in the current code folder.



Correct: Makefile is in current program folder workspace, no path problem.

Wrong: Makefile in NOT in program folder workspace, may bring relative-path problem.



5.1. Run the code via command lines
a) Compile by **pure** command lines, with "g++" compile command

- In the terminal, type compile command:
- "g++ -std=c++17 <your source .cpp files> -o <output filename>"
- E.g. "g++ -std=c++17 helloworld.cpp foo.cpp -o helloworld"

- Then run the executable program:
- Different terminal have different calling method, maybe:
  - "./<filename>" (UNIX-like system including Linux and macOS)
  - ".\<filename>.exe" (Windows PowerShell)
  - "<filename>.exe" (Windows CMD)

5.1. Run the code via command lines

a) Compile by **pure** command lines, with “g++” compile command

```
zhangsan@zhangsandeMacBook-Pro ~/Desktop/USTF/OneDrive_1_2024-9-10/Progr
am1_Acronym @ g++ -std=c++17 acronym.cpp -o acronym
zhangsan@zhangsandeMacBook-Pro ~/Desktop/USTF/OneDrive_1_2024-9-10/Progr
am1_Acronym @ ./acronym
Program to generate acronyms
Enter string:
```

- Use <Ctrl>(^) + C to exit the program.

5.1. Run the code via command lines
b) Compile by **pure** command lines, with "makefile".

- Write your own "Makefile" script, or use the template given. (If you use Prof Kinley's template, remember to change two names!)
- In the terminal, type make command:
- "make"

- Then run the executable program:
- Different terminal have different calling method, maybe:
  - "./<filename>" (UNIX-like system including Linux and macOS)
  - ".\<filename>.exe" (Windows PowerShell)
  - "<filename>.exe" (Windows CMD)

5.1. Run the code via command lines
b) Compile by **pure** command lines, with “makefile”.

- One useful tip: Use “↑” “↓”(Up/down arrows on the keyboard) to view command history.
- Two useful tips: Use “⇥” (Tab) for automatic completion the command / filename
- Some shells, like fish and xonsh, can predict your command in gray color. Thus, use “→” (Right arrow on the keyboard) to accept the suggestion.

6. Q & A time

- Thank you for your listening!

- Fang Zihao (USTF)
- (SDS, 122090106@link.cuhk.edu.cn)