AIMS-IWR Compact Course 2018

Day 2: Introduction to C++ (lab part)

Ole Klein

Interdisciplinary Center for Scientific Computing
Heidelberg University
`ole.klein@iwr.uni-heidelberg.de`

December 11, 2018

Editing C++ Source Files

C++ source files are just normal text files and can be edited with any text editor. While there are text editors specialized for programming (Emacs, vim), and integrated development environments (IDEs) like Eclipse, the standard editors are enough for simple programs.

To open the GNOME standard editor, type

```
[user@host ~] gedit
```

To open the KDE standard editor, type

```
[user@host ~] kate
```

Other options are `geany` and `qtcreator`. You can also open an editor from the dropdown menu (but than you have to navigate to the file in the terminal afterwards).

Compiling C++ Programs

- C++ programs have to be compiled before they can be run
- The standard C++ compiler under Linux is called g++
- Another option is the compiler `clang++` from the LLVM project, which often produces error messages that are easier to understand
- One should always use the option `-Wall`, so that the compiler warns about issues that might be legal C++ but lead to unforseen results
- The option `-o` is used to define the name of the resulting executable file (program)

```
[user@host ~] g++ -Wall -std=c++14 -o example example.cc
```

- The resulting program may be executed by writing `./` in front:

```
[user@host ~] ./example
```

Common Compiler Flags

Debugging information:

- `-g`: include debug symbols in program

Optimization:

- `-O0`: completely disable optimization
- `-Og`: flags of `-O1` that don't interfere with debugging
- `-O1`: use optimizations that don't impact on compile time
- `-Os`: flags of `-O2` that don't increase program size
- `-O2`: use most optimizations, increases compile time
- `-O3`: flags of `-O2` plus expensive optimizations
- `-Ofast`: flags of `-O3` plus `-ffastmath`, not standards compliant

Recommendation: `-Og -g` for development, `-O3` or `-Ofast` for application

Others:

- `-march=native`: compile for local processor architecture (fast but non-portable)
- `-Wall`: enable all warning about possible coding errors
- `-Werror`: turn warnings into hard errors

First C++ Compilation

Write the following program, using an editor of your choice:

```
#include <iostream>

int main(int argc, char** argv)
{
  std::cout << "Hello world!" << std::endl;
  return 0;
}
```

Compile the program and then execute it:

```
[user@host ~] g++ -Wall -o helloworld helloworld.cc
[user@host ~] ./helloworld
```

Understanding C++ Compiler Messages

Create a file with the following content:

```
#include <iostrean>
int main(int argc, char** argv)
{
  std::cout << "Typing is difficult" << endl;
  int ret = 0;
  return retv;
}
```

This source file contains some errors so you can get familiar with compiler error messages. Compile the program, try to understand the resulting messages and fix the bugs. Repeat until no more error messages appear.

```
[user@host ~] g++ -Wall -o errors errors.cc
```

First Real Program

Assume we want to compute q raised to the power of n, where both q and n are natural numbers. This is repeated multiplication:

$$q^0 := 1, \qquad \forall\, n > 0\colon \quad q^n := q^{n-1} \cdot q = \prod_{i=1}^{n} q$$

Try to write the following four versions of such a program:

- a recursive implementation using n multiplications
- an iterative implementation using n multiplications
- a more efficient recursive implementation
- a more efficient iterative implementation

Note: the efficient versions should make use of the identity $q^n = q^{2 \cdot n/2} = \left(q^2\right)^{n/2}$ for n even

(These tasks are roughly sorted by difficulty.)