Lecture 3:
Introduction to Computers, the Internet and The World Wide Web (cont)

Ioan Raicu
Department of Electrical Engineering & Computer Science
Northwestern University

EECS 211
Fundamentals of Computer Programming II
March 31st, 2010

1.13 Key Software Trend: Object Technology

- Key benefit of object-oriented programming is that the software is
  - more understandable
  - better organized and
  - easier to maintain, modify and debug
- Significant because perhaps as much as 80 percent of software costs are associated not with the original efforts to develop the software, but with the continued evolution and maintenance of that software throughout its lifetime.

1.14 Typical C++ Development Environment

- C++ systems generally consist of three parts: a program development environment, the language and the C++ Standard Library.
- C++ programs typically go through six phases: edit, preprocess, compile, link, load and execute.

1.14 Typical C++ Development Environment

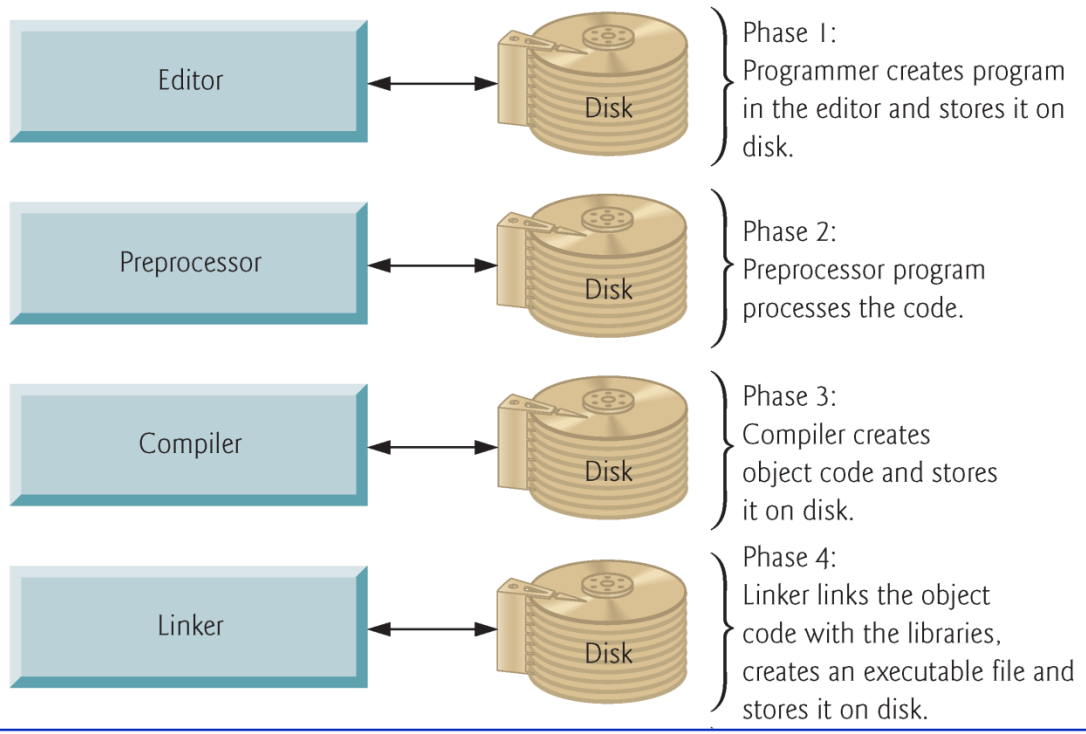

Fig. 1.1 | Typical C++ environment. (Part 1 of 2)

1.14 Typical C++ Development Environment

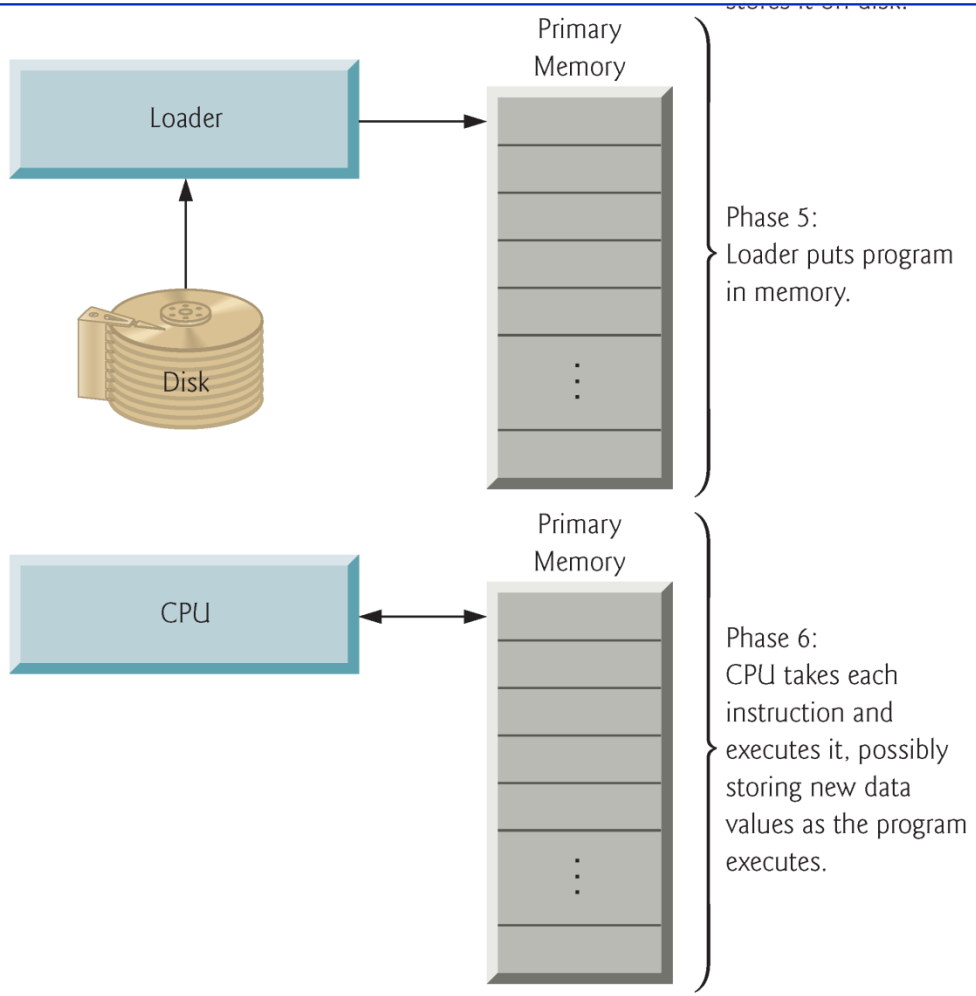

Fig. 1.2 | Typical C++ environment. (Part 2 of 2)

1.14 Typical C++ Development Environment (cont.)

- Programs do not always work on the first try.
- If this occurs, you'd have to return to the edit phase, make the necessary corrections and proceed through the remaining phases again to determine that the corrections fix the problem(s).

1.14 Typical C++ Development Environment (cont.)

Common Programming Error 1.1

Errors such as division by zero occur as a program runs, so they're called **runtime errors** *or* **execution-time errors**. **Fatal runtime errors** *cause programs to terminate immediately without having successfully performed their jobs.* **Nonfatal runtime errors** *allow programs to run to completion, often producing incorrect results. [Note: On some systems, divide-by-zero is* not *a fatal error. Please see your system documentation.]*

1.14 Typical C++ Development Environment (cont.)

- Most programs in C++ input and/or output data.
- Certain C++ functions take their input from `cin` (the standard input stream; pronounced "see-in"), which is normally the keyboard, but `cin` can be redirected to another device.
- Data is often output to `cout` (the standard output stream; pronounced "see-out"), which is normally the computer screen, but `cout` can be redirected to another device.
- When we say that a program prints a result, we normally mean that the result is displayed on a screen.
  - Data may be output to other devices, such as disks and hardcopy printers.
- There is also a standard error stream referred to as `cerr` that is used for displaying error messages.

1.14 Typical C++ Development Environment (cont.)

Portability Tip 1.3

Although it's possible to write portable programs, there are many problems among different C and C++ compilers and different computers that can make portability difficult to achieve. Writing programs in C and C++ does not guarantee portability. You often will need to deal directly with compiler and computer variations. As a group, these are sometimes called ***platform*** *variations.*

1.14 Typical C++ Development Environment (cont.)

Good Programming Practice 1.3

If, after reading your C++ language documentation, you still are not sure how a feature of C++ works, experiment using a small test program and see what happens. Set your compiler options for "maximum warnings." Study each message that the compiler generates and correct the program to eliminate the messages.

1.16 Test-Driving a C++ Application

- In this section, you'll run and interact with your first C++ application.
- You'll begin by running an entertaining guess-the-number game, which picks a number from 1 to 1000 and prompts you to guess it.
- If your guess is correct, the game ends.
- If your guess is not correct, the application indicates whether your guess is higher or lower than the correct number.
- There is no limit on the number of guesses you can make.
- [*Note:* For this test drive only, we've modified this application from the exercise you'll be asked to create in Chapter 6, Functions and an Introduction to Recursion. Normally this application randomly selects the correct answer as you execute the program. The modified application uses the same correct answer every time the program executes (though this may vary by compiler), so you can use the same guesses we use in this section and see the same results as we walk you through interacting with your first C++ application.]

1.16 Test-Driving a C++ Application (cont.)

- Test-drive in a Linux shell

```
~$ cd examples/ch01/GuessNumber/GNU_Linux
~/examples/ch01/GuessNumber/GNU_Linux$
```

Fig. 1.10 | Changing to the **GuessNumber** application's directory.

```
~/examples/ch01/GuessNumber/GNU_Linux$ g++ GuessNumber.cpp -o GuessNumber
~/examples/ch01/GuessNumber/GNU_Linux$
```

Fig. 1.11 | Compiling the **GuessNumber** application using the g++ command.

```
~/examples/ch01/GuessNumber/GNU_Linux$ ./GuessNumber
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
?
```

Fig. 1.12 | Running the **GuessNumber** application.

```
~/examples/ch01/GuessNumber/GNU_Linux$ ./GuessNumber
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? 500
Too high. Try again.
?
```

Fig. 1.13 | Entering an initial guess.

```
~/examples/ch01/GuessNumber/GNU_Linux$ ./GuessNumber
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? 500
Too high. Try again.
? 250
Too low. Try again.
?
```

Fig. 1.14 | Entering a second guess and receiving feedback.

```
Too low. Try again.
? 375
Too low. Try again.
? 437
Too high. Try again.
? 406
Too high. Try again.
? 391
Too high. Try again.
? 383
Too low. Try again.
? 387
Too high. Try again.
? 385
Too high. Try again.
? 384
Excellent! You guessed the number.
Would you like to play again (y or n)?
```

Fig. 1.15 | Entering additional guesses and guessing the correct number.

```
Excellent! You guessed the number.
Would you like to play again (y or n)? y

I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
?
```

Fig. 1.16 | Playing the game again.

```
Excellent! You guessed the number.
Would you like to play again (y or n)? n

~/examples/ch01/GuessNumber/GNU_Linux$
```

Fig. 1.17 | Exiting the game.

1.17 Software Technologies

- Agile Software Development is a set of methodologies that try to get software implemented quickly with fewer resources than previous methodologies.
- Refactoring involves reworking code to make it clearer and easier to maintain while preserving its functionality.
- Game programming. The computer game business is larger than the first-run movie business.
- Open source software is a style of developing software in whuixh individuals and companies contribute their efforts in developing, maintaining and evolving software in exchange for the right to use that software for their own purposes, typically at no charge.

1.17 Software Technologies (cont.)

- Linux is an open source operating system and one of the greatest successes of the open source movement.
- MySQL is an open source database management system.
- PHP is the most popular open source server-side "scripting" language for developing Internet-based applications.
- LAMP is an acronym for the set of open source technologies that many developers used to build web applications—it stands for Linux, Apache, MySQL and PHP (or Perl or Python—two other languages used for similar purposes).
- Ruby on Rails combines the scripting language Ruby with the Rails web application framework developed by the company 37Signals.
- With Software as a Service (SaaS) the software runs on servers elsewhere on the Internet.
  - You typically access the service through a browser.

1.18 Future of C++: Open Source Boost Libraries, TR1 and C++0x

- The main goals for the new standard are to make C++ easier to learn, improve library building capabilities, and increase compatibility with the C programming language.
- Chapter 23 considers the future of C++—we introduce the Boost C++ Libraries, Technical Report 1 (TR1) and C++0x.
- The Boost C++ Libraries are free, open source libraries created by members of the C++ community.
- Boost has grown to over 80 libraries, with more being added regularly.

1.19 Software Engineering Case Study: Introduction to Object Technology and the UML

- Object-oriented design (OOD) models software in terms similar to those that people use to describe real-world objects.
  - Takes advantage of class relationships, where objects of a certain class, have the same characteristics.
  - Takes advantage of inheritance relationships, where new classes of objects are derived by absorbing characteristics of existing classes and adding unique characteristics of their own.
- Object-oriented design provides a natural and intuitive way to view the software design process.
- OOD also models communication between objects.

1.19 Software Engineering Case Study: Introduction to Object Technology and the UML (cont.)

- OOD encapsulates attributes and operations into objects.
- Objects have the property of information hiding.
  - Objects may know how to communicate with one another across well-defined interfaces, but normally they're not allowed to know how other objects are implemented.
  - Information hiding is crucial to good software engineering.

1.19 Software Engineering Case Study: Introduction to Object Technology and the UML (cont.)

- Languages like C++ are object oriented.
- Programming in such a language is called object-oriented programming (OOP), and it allows computer programmers to implement object-oriented designs as working software systems.
- C++ classes contain functions that implement operations and data that implements attributes.

1.19 Software Engineering Case Study: Introduction to Object Technology and the UML (cont.)

Software Engineering Observation 1.4

Reuse of existing classes when building new classes and programs saves time and money. Reuse also helps you build more reliable and effective systems, because existing classes and components often have gone through extensive testing, debugging and performance tuning.

Questions