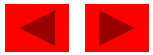
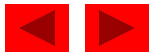
Chapter 1 – Introduction

Outline



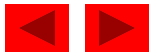
Chapter 1 – Introduction

Outline



Objectives

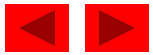
- In this chapter, you will learn:
  - To understand basic computer concepts.
  - To become familiar with different types of programming languages.
  - To become familiar with the history of the C programming language.
  - To become aware of the C standard library.
  - To understand the elements of a typical C program development environment.
  - To appreciate why it is important to learn C in a first programming course.
  - To appreciate why C provides a foundation for further study of programming languages in general and of C++ and Java in particular.



1.1 Introduction

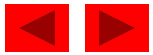
- We will learn
  - C programming language
  - *Structured programming techniques*
- This book also covers (but not covered in this class)
  - C++
    - Chapter 15 – 23 introduce the C++ programming language
  - Java
    - Chapters 24 – 30 introduce the Java programming language
- This course is appropriate for

  Technically oriented people with little or no programming experience



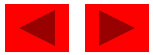
1.2 What is a Computer?

- Computer
  - Device capable of *performing computations* and *making logical decisions*
  - Computers process data under the control of sets of instructions called *computer programs*
- Hardware
  - Various devices comprising a computer
  - Keyboard, screen, mouse, disks, memory, CD-ROM, and processing units
- Software
  - **Programs** that run on a computer (operation systems, application programs)
  - Structured programming, top-down stepwise refinement, functionalization, and object-oriented programming



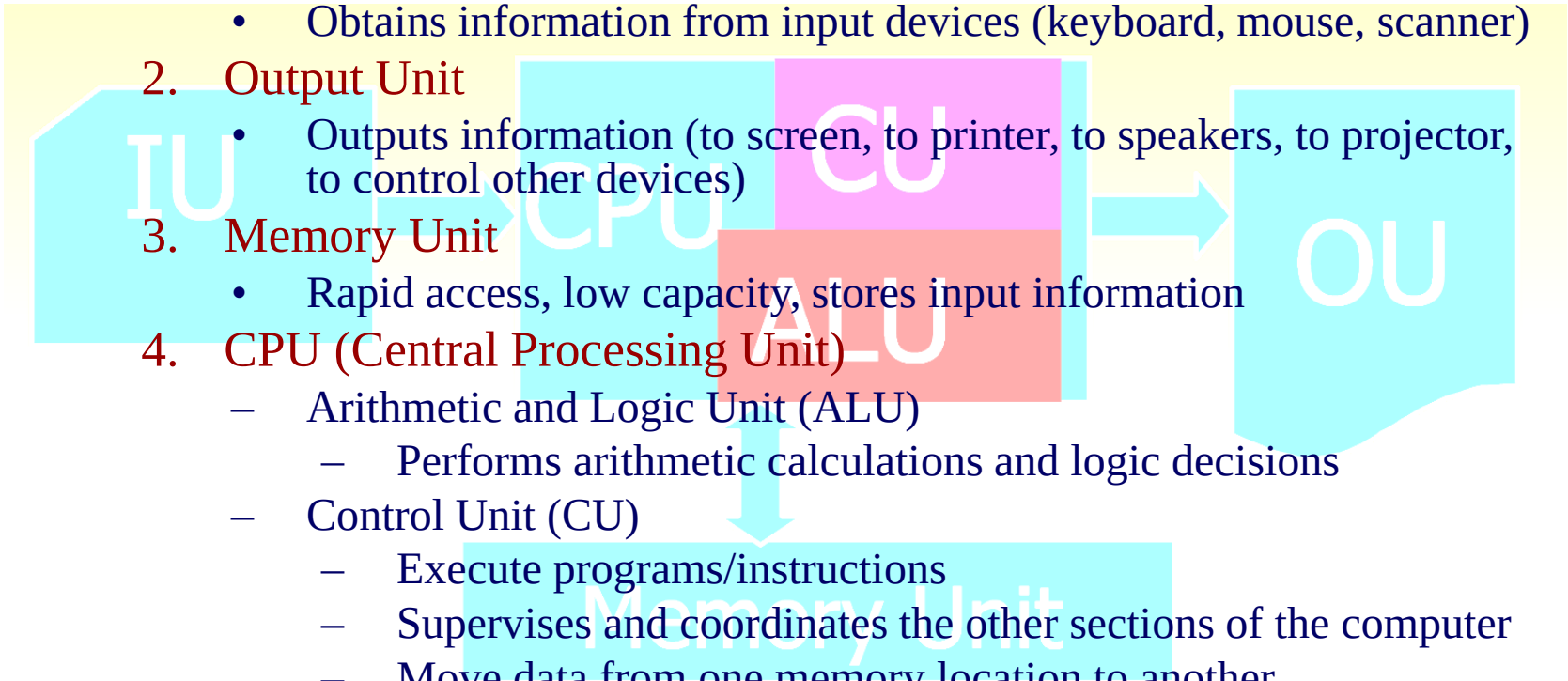
Terminology

- **Bi**nary Digi**t**s (bit): 1 and 0
  - The computer can combine the two digital states to represent letters, numbers, colors, sounds, images, shapes, and even odors.
  - An "on" or "off" electronic state is represented by a bit, short for binary digit
- Encoding Systems: Bits (位元) and Bytes (位元組)
  - Bits are combined according to an encoding system to represent letters, numbers, and special characters, collectively referred to as alphanumeric characters
  - The combination of bits used to represent a character is called a byte (**B**inar**y** **Te**rm, 8 bits/byte)
  - 8 bits = byte
- Representation of a Character
  - ASCII (American Standard Code for Information Interchange) is the most popular encoding system for PCs and data communication
    - ASCII – 7 bits
    - ANSI – 8 bits/byte
    - UNICODE – 16 bits
    - Big5 – 16 bits
- Storage Capacities
  - KB (kilobyte) = 2^{10} Bytes = 1,024 Bytes ≈ 10^3 Bytes
  - MB (megabyte) = 2^{20} Bytes = 1,024 KB = 1,048,576 Bytes ≈ 10^6 Bytes
  - GB (gigabyte) = 2^{30} Bytes = 1,024 MB ≈ 10^9 Bytes
  - TB (terabyte) = 2^{40} Bytes = 1,024 GB ≈ 10^{12} Bytes



1.3 Computer Organization

- Five logical units in every computer:
  1. Input Unit
     - Obtains information from input devices (keyboard, mouse, scanner)
  2. Output Unit
     - Outputs information (to screen, to printer, to speakers, to projector, to control other devices)
  3. Memory Unit
     - Rapid access, low capacity, stores input information
  4. CPU (Central Processing Unit)
     - Arithmetic and Logic Unit (ALU)
       - Performs arithmetic calculations and logic decisions
     - Control Unit (CU)
       - Execute programs/instructions
       - Supervises and coordinates the other sections of the computer
       - Move data from one memory location to another
  5. Secondary Storage Unit
     - Cheap, long-term, high-capacity storage (e.g., Hard Disks, Memory Sticks)
     - Stores inactive programs



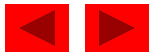
1.6 Machine Languages, Assembly Languages, and High-level Languages

1. Machine languages (機器語言)
   - Strings of numbers giving machine specific instructions
   - Example:

```
+1300042774
+1400593419
+1200274027
```

2. Assembly languages (組合語言)
   - English-like abbreviations representing elementary computer operations (translated via assemblers)
   - Example:

```
LOAD    BASEPAY
ADD     OVERPAY
STORE   GROSSPAY
```

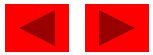



1.6 Machine Languages, Assembly Languages, and High-level Languages

3 High-level languages (高階語言)

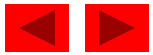
- Codes similar to everyday English
- Use mathematical notations (translated via compilers)
- Example:

```
grossPay = basePay + overTimePay
```

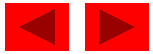
1.7 History of C

- C
  - C was created by Dennis Ritchie at the Bell Telephone Laboratories in 1972
  - Evolved from two previous programming languages, BCPL (Basic Computer Programming language, 1967) and B (developed by Ken Thompson of Bell Labs)
  - Used to develop UNIX
  - Used to write modern operating systems
  - Hardware independent (portable)
  - By late 1970's C had evolved to "Traditional C"
- Standardization
  - Many slight variations of C existed, and were incompatible
  - Committee (ANSI, the American National Standards Institute) formed to create a "unambiguous, machine-independent" definition - ANSI Standard C
  - Standard created in 1989, updated in 1999



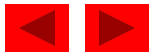
1.7 History of C

- Why Use C?
  - C is a powerful and flexible language
  - C is a popular language preferred by professional programmers
  - C is a portable language
  - C is a language of few words, containing only a handful of terms, called keywords, which serve as the base on which the language's functionality is built
  - C is modular. C code can (and should) be written in routines called functions.



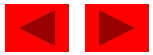
1.8 The C Standard Library

- C programs consist of pieces/modules called functions
  - A programmer can create his own functions
    - Advantage: the programmer knows exactly how it works
    - Disadvantage: time consuming
  - Programmers will often use the C library functions
    - Use these as building blocks
  - Avoid re-inventing the wheel
    - If a premade function exists, generally best to use it rather than write your own
    - Library functions carefully written, efficient, and portable



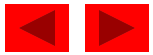
1.12 Other High-level Languages

- Other high-level languages
  - FORTRAN (FORmula TRANslator)
    - Used for scientific and engineering applications
    - Developed by IBM in 1950s
  - COBOL (COmmon Business Oriented Language)
    - Used to manipulate large amounts of data, e.g., for commercial applications
    - Developed in 1959
  - Pascal
    - Designed for teaching structured programming and rapidly became the preferred programming language in most colleges
    - Developed in 1971
  - Ada
    - Multitasking



1.13 Structured Programming

- Structured programming
  - Disciplined approach to writing programs
  - Clear, easy to test and debug and easy to modify
- Multitasking
  - Specifying that many activities run in parallel



1.14 Basics of a Typical C Program Development Environment

- Phases of C Programs:

1. *Edit*
2. *Preprocess*
3. *Compile*
4. *Link*
5. *Load*
6. *Execute*

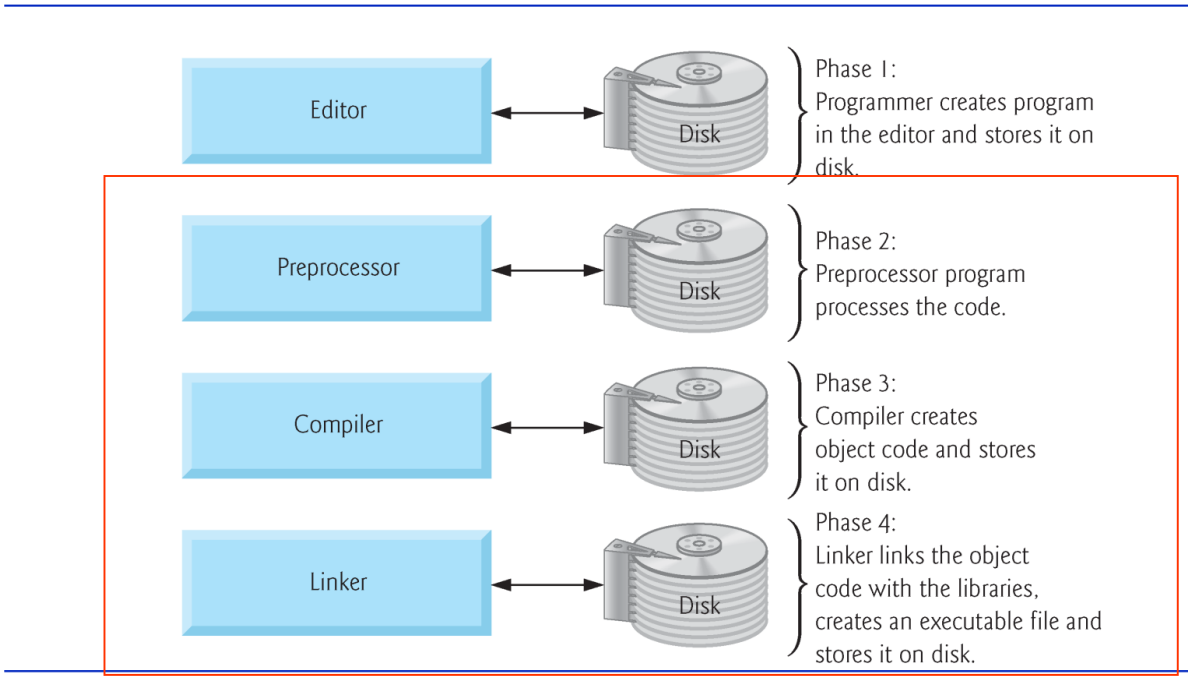


Fig. 1.1 | Typical C development environment. (Part 1 of 2.)

1.14 Basics of a Typical C Program Development Environment

- Phases of C Programs:

1. *Edit*
2. *Preprocess*
3. *Compile*
4. *Link*
5. *Load*
6. *Execute*

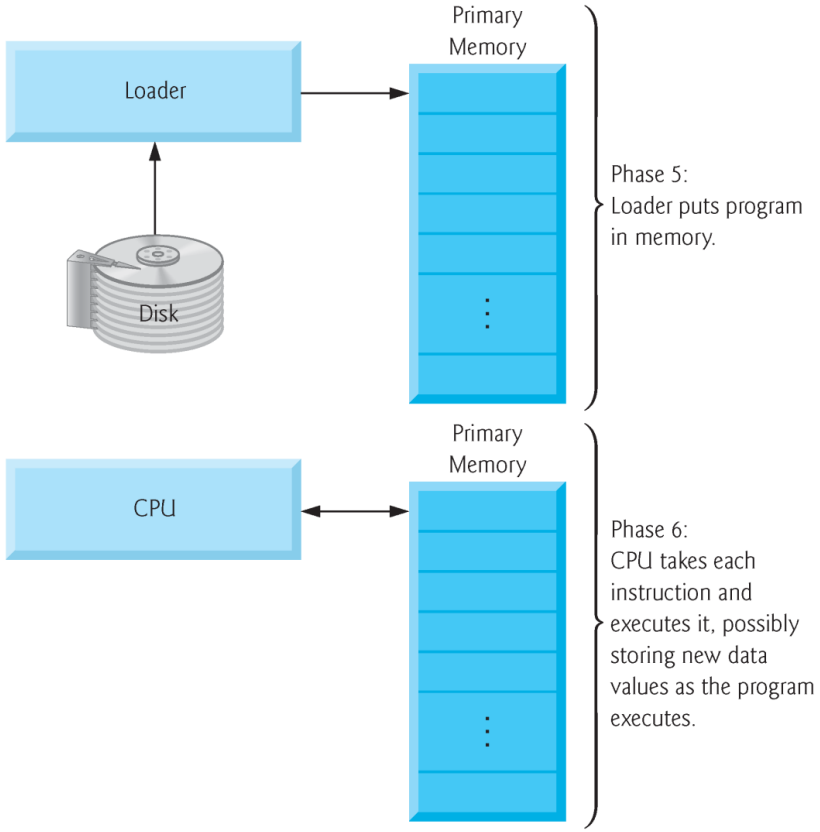


Fig. 1.1 | Typical C development environment. (Part 2 of 2.)