Chapter 6: An Introduction to System Software and Virtual Machines

Invitation to Computer Science, C++ Version, Third Edition

Objectives

In this chapter, you will learn about:

- System software
- Assemblers and assembly language
- Operating systems

Introduction

- Von Neumann computer
  - “Naked machine”
  - Hardware without any helpful user-oriented features
  - Extremely difficult for a human to work with
- An interface between the user and the hardware is needed to make a Von Neumann computer usable

Introduction (continued)

- Tasks of the interface
  - Hide details of the underlying hardware from the user
  - Present information in a way that does not require in-depth knowledge of the internal structure of the system

Introduction (continued)

- Tasks of the interface (continued)
  - Allow easy user access to the available resources
  - Prevent accidental or intentional damage to hardware, programs, and data

System Software: The Virtual Machine

- System software
  - Acts as an intermediary between users and hardware
  - Creates a virtual environment for the user that hides the actual computer architecture
- Virtual machine (or virtual environment)
  - Set of services and resources created by the system software and seen by the user
Types of System Software

- System software is a collection of many different programs
- Operating system
  - Controls the overall operation of the computer
  - Communicates with the user
  - Determines what the user wants
  - Activates system programs, applications packages, or user programs to carry out user requests

Types of System Software (continued)

- User interface
  - Graphical user interface (GUI) provides graphical control of the capabilities and services of the computer
- Language services
  - Assemblers, compilers, and interpreters
  - Allow you to write programs in a high-level, user-oriented language, and then execute them

Types of System Software (continued)

- Memory managers
  - Allocate and retrieve memory space
- Information managers
  - Handle the organization, storage, and retrieval of information on mass storage devices
- I/O systems
  - Allow the use of different types of input and output devices
- Scheduler
  - Keeps a list of programs ready to run and selects the one that will execute next
- Utilities
  - Collections of library routines that provide services either to user or other system routines
Assemblers and Assembly Language: Assembly Language

- Machine language
  - Uses binary
  - Allows only numeric memory addresses
  - Difficult to change
  - Difficult to create data

Assembly Language (continued)

- Assembly languages
  - Designed to overcome shortcomings of machine languages
  - Create a more productive, user-oriented environment
  - Earlier termed second-generation languages
  - Now viewed as low-level programming languages

Assembly Language (continued)

- Source program
  - An assembly language program
- Object program
  - A machine language program
- Assembler
  - Translates a source program into a corresponding object program

Assembly Language (continued)

- Advantages of writing in assembly language rather than machine language
  - Use of symbolic operation codes rather than numeric (binary) ones
  - Use of symbolic memory addresses rather than numeric (binary) ones
  - Pseudo-operations that provide useful user-oriented services such as data generation
Examples of Assembly Language Code (continued)

**Algorithmic operations**

- Set the value of \( i \) to 1 (line 2).
- Add 1 to the value of \( i \) (line 7).

Examples of Assembly Language Code (continued)

**Arithmetic expression**

\[ A = B + C - 7 \]

(Assume that \( B \) and \( C \) have already been assigned values)

Examples of Assembly Language Code (continued)

**Problem**

- Read in a sequence of non-negative numbers, one number at a time, and compute a running sum.
- When you encounter a negative number, print out the sum of the non-negative values and stop.
Translation and Loading

Before a source program can be run, an assembler and a loader must be invoked

Assembler
- Translates a symbolic assembly language program into machine language

Loader
- Reads instructions from the object file and stores them into memory for execution

Operating Systems

System commands
- Carry out services such as translate a program, load a program, run a program
- Types of system commands
  - Lines of text typed at a terminal
  - Menu items displayed on a screen and selected with a mouse and a button: point-and-click
  - Examined by the operating system

Functions of an Operating System

Five most important responsibilities of the operating system
- User interface management
- Program scheduling and activation
- Control of access to system and files
- Efficient resource allocation
- Deadlock detection and error detection
The User Interface

- Operating system
  - Waits for a user command
  - If command is legal, activates and schedules the appropriate software package

- User interfaces
  - Text-oriented
  - Graphical

System Security And Protection

- The operating system must prevent
  - Non-authorized people from using the computer
    - User names and passwords
  - Legitimate users from accessing data or programs they are not authorized to access
    - Authorization lists

Efficient Allocation Of Resources

- The operating system ensures that
  - Multiple tasks of the computer may be underway at one time
    - Processor is constantly busy
      - Keeps a “queue” of programs that are ready to run
      - Whenever processor is idle, picks a job from the queue and assigns it to the processor

The Safe Use Of Resources

- Deadlock
  - Two processes are each holding a resource the other needs
  - Neither process will ever progress
  - The operating system must handle deadlocks
    - Deadlock prevention
    - Deadlock recovery

Historical Overview Of Operating Systems Development

- First generation of system software (roughly 1945–1955)
  - No operating systems
  - Assemblers and loaders were almost the only system software provided
Historical Overview of Operating Systems Development (continued)

- Second generation of system software (1955–1965)
  - Batch operating systems
  - Ran collections of input programs one after the other
  - Included a command language

Historical Overview of Operating Systems Development (continued)

- Third-generation operating systems (1965–1985)
  - Multiprogrammed operating systems
  - Permitted multiple user programs to run at once

Historical Overview of Operating Systems Development (continued)

- Fourth-generation operating systems (1985–present)
  - Network operating systems
  - Virtual environment treats resources physically residing on the computer in the same way as resources available through the computer's network

The Future

- Operating systems will continue to evolve
- Possible characteristics of fifth-generation systems
  - Multimedia user interfaces
  - Parallel processing systems
  - Completely distributed computing environments
Summary

- System software acts as an intermediary between the users and the hardware
- Assembly language creates a more productive, user-oriented environment than machine language
- An assembler translates an assembly language program into a machine language program

Summary

- Responsibilities of the operating system
  - User interface management
  - Program scheduling and activation
  - Control of access to system and files
  - Efficient resource allocation
  - Deadlock detection and error detection