Software and Operating systems (OS)

Aniel Nieves-González

Fall 2015

Hardware and software I

- Hardware is the set of physical components or devices that constitute a computer.
- Software are the computer programs that are the embodiment of some computer algorithms (hence they perform certain task.
- Broadly, in the software category we have:
  - Operating systems (OSs).
  - Applications.
  - Firmware (control programs stored in chips).

Operating systems I

- An operating system (OS) is software that serves as an intermediary between the different hardware components of a computer and the users and applications (other software) of a computer.
- The main function of an OS is to manage the different resources provided by hardware and efficiently allocating such resources so that they can be used by users and applications.
- An OS is composed of a kernel and several other programs. The kernel is the part of the OS that is always running. The other programs support functions like file management, device control, etc.

Operating systems II

- Modern OS support interaction with the user via a graphical user interface (GUI) and via text interface (command line).
- OS give programmers function and tools that facilitate the development of applications.
- OS can be of different types.
  - *Single-tasking:* Only one program can be run at a time.
  - *Multi-tasking:* More than one program can be run at a time. This is achived by time-sharing, the available processor time is divided among the different running program (processes).
  - *Single-user:* There is no way to distinguish between different users.
  - *Multi-user:* This extends the concept of multi-tasking to distinguish between different users.

Operating systems III

- *Distributed:* It manages a set of computers and allow them to work as a single entity.
- *Embedded:* An OS designed to work on computer dedicated to a specific purpose (not general purpose computers).
- *Real-time:* In this case the OS guarantees that certain events or data are going to be process in certain (short) amount of time.

Applications: I

The applications consists in the computer programs that the users are interested in running because it solves their specific problems. Relevant to business administration we have:

- Desktop software.
- Enterprise software:
  - Enterprise Resourse Planning (ERP) system with multiple modules installed can touch many functions of the business such as sales, inventory, manufacturing, human resources, purchasing, order tracking, and decision support.
  - Customer relationship management (CRM) systems support customer-related sales and marketing activities.
  - Supply chain management (SCM) systems can help a firm manage aspects of its value chain.

Applications: II

- Business intelligence (BI) systems use data created by other systems to provide reporting and analysis for organizational decision making.
- Database management system (DBMS). Strictly speaking is more general than enterprise software. But is important in the context of an enterprise.

Discussion:

- What are the advantages and disadvantages of packaged Enterprise systems?

Discussion:

- What are the advantages and disadvantages of packaged Enterprise systems?
- For what it matters: What are the advantages and disadvantages of any packaged software?

Distributed computing

- Distributed computing refers whenever there is a cluster of computers (MIMD cluster) that is spread over different locations and is used as a single computing entity to solve a problem.

Server-client model.

- A *server* is either hardware of software that serves requests from clients.
- In terms of harware the clients can be users or another computers.
- In terms of software the clients are another programs. For instance: web server telnet, ssh, etc.

We'll see a practical example...

Writing software: Programming I

A *a programming language* is a formal language design to communicate instructions to a computer. Programming languages are used to write programs, which express (implement) algorithms.

1. A *high-level language* provides strong abstraction from the details of a computing system. For example: FORTRAN, Cobol, C, Java, SQL, MATLAB, R, etc.
2. A *low-level language* provides little abstraction from the inner workings of a computer system. For example: Any assembly language.

Writing software: Programming II

Ultimately, any programming language is translated into machine language (machine instructions), which are in binary form. *Loosely* speaking that translation process is called compilation or interpretation. The difference between compilation and interpretation is important.

- *Compilation* is a multi-step procedure in which the input is the source code (i.e., an algorithm written in a programming language) and the output is the program in machine readable format (binary or assembly).
- Steps in the compilation process include:
  - Syntax verification.
  - Code optimization.
  - Assembly language generation

Writing software: Programming III

- Informally, the work performed by the assembler (translate from assembly language to machine code) and the linker (link the source code to other code) is included in the compilation process.
- Examples of compiled languages are: FORTRAN, C, and the descendants of C.
- For an interpreted programming language the source code is translated into a low-level language called bytecode at run time. The bytecode is generated and executed by another piece of software called the interpreter.
- Examples of interpreted languages are: Java, R, and MATLAB.

Writing software: Programming IV

- In the case of Java, a source code written in Java is translated into bytecode which is interpreted by the Java Virtual Machine (JVM).

Writing software: Compilation vs. interpretation I

- Compilation produces an output that is generally faster (in terms of run time) than interpreted code. The reason is that in this case the interpreter is removed from the equation.
- Interpretation produces code that is highly portable in comparison with compiled code. Recall, that in contrast, to port the source code of a compiled language to another machine you have to compile the code again.

Writing software I

- Source code is written using an application called *text editor*.
- Text editors are not the same as word processing software. Text editors produce plain text files, word processors add special characters to produced a formatted and stylistic output.
- Operating systems always include a text editor.
- Besides being used to write source code, text editors are used to edit OS configuration files.
- In MS Windows the default text editor is notepad. In the UNIX/Linux world the main text editors are vim, emacs, etc.

Writing software II

- To develop software the applications used are text editor, compiler/interpreter, debugger for detecting error, etc.
- Some professional programmers use an integrated development environment (IDE) to write their code.
  - The IDE includes a text editor, a debugger, and other useful programming tools.
  - The IDE will also compile a programmers code.

Discussion I

Why do some technology projects fail?

- Unclear goals.
- Weak commitment.
- Inaccurate estimates of the resourses needed.
- Poor communication.
- Poor management.
- etc.