B211 Internet Computing

Introduction to Programming, Scripting and Mark-up Languages

Lecture Outline

- Why study Programming, Scripting and Mark-up Languages?
- Categories of Languages

Programming, Scripting and Mark-up Languages

- In current architecture of computing machines, we use software to control the hardware, so that the hardware can execute the tasks we want.
- The reason for having software is so that we don't have to make new hardware for every task we wish done.
- The purpose of having *development languages* is to create the software.

Languages (cont'd)

- Development languages used to create software in the past have fallen into two major categories:
  1) programming languages, and
  2) scripting languages.
- With the adoption of the WWW, a third category called mark-up languages, have rose to popularity.
  - Although it is debatable whether we can say mark-up languages are used to *create* software, it is nevertheless unquestionable that mark-up languages play a critical role in developing web technologies.

Mark-up vs “Execution” Languages

- One general way of differentiating between a mark-up language and the traditional programming/scripting languages is by what code in the languages does:
  - Code in mark-up languages are mainly tags which identify (or “mark up”) sections of text.
  - Code in programming/scripting languages contains mainly instructions to be executed by the operating system and machine.

Mark-up vs “Execution” Languages (cont’d)

- But we shall see later that this very broad description of programming/scripting languages as “execution” languages may not be adequate is come cases.

Categorization of Development Languages

- Categorizing languages into programming, scripting or mark-up languages is not the only way of classifying them.
- For the rest of this lecture, we will look at different ways of categorizing languages, as well as how development languages have evolved over the years.

Computational Paradigms

- A *computational paradigm* refers to a way of computing, or more precisely, the process of coming up with a solution to solve a problem using computational machines.
  - Some refer to it as paradigms of thinking as well, since it reflects the way we as humans solve problems.
- Three major paradigms which have arisen through the history of computing and programming are:
  - imperative (or procedural)
  - functional
  - logic

Imperative (or Procedural) Programming

- This is far and away the most dominant type of programming that has developed in computing.
  - The architecture of almost all current machines (called the *von Neumann architecture, made up of stacks, registers, logic gates, etc*) is founded on the imperative way of problem solving.
- Primarily consisting of sequences of commands (or *imperatives*)

Imperative Programming (cont'd)

- All imperative languages contain constructs for *sequences*, *conditionals* (eg. `if…then…else` statements), and *loops* (eg. `while` statements).
- Inherently time-based
  - Do this first, then this second, then …
- Parallel programming, Object-oriented programming, event-based programming all fall into this category.

Imperative Programming (cont'd)

- Examples of imperative languages:
  - C and C++
  - Java
  - Perl
  - Pascal
  - Fortran
  - Cobol
  - Visual Basic

Functional Programming

- All computations by functions
- Ideally, no concept of time
  - but most functional languages still have imperative constructs like loops, so that it executes efficiently in von Neumann machines.
- Two special properties compared to imperative languages:
  - Functions as first-class objects – functions can be treated just like variables - be passed as parameters, assigned at run-time, etc.
  - Lazy evaluation for all data – eg. parameters and values evaluated only when it is required.

Functional Programming

- Examples of functional programming languages:
    - Lisp
    - Miranda
    - Standard-ML

Logic Programming

- Using a logical system of reasoning to find solutions
- Inherently *declarative*, as oppose to procedural
    - Do not define how to solve the problem, only what the facts are.
    - The solution is found by using the logical system to resolve the given facts.
- Examples of logic programming languages:
    - Prolog
    - CLP
    - Gödel

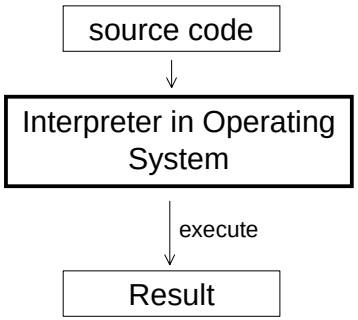
Common Features of All Languages

- Regardless of paradigm, all languages share a few common features (although they use different names to refer to them):
    - *data types*: different ways of storing data (eg. integers, strings, etc) and different ways of operating on the data (eg. +, -, append, etc).
    - *variables*: locations which can take on different values at different times.
    - *subroutines*: block of code which can be reused.
    - *parameters*: data passed from one block of code to another.

From Source Code to Execution

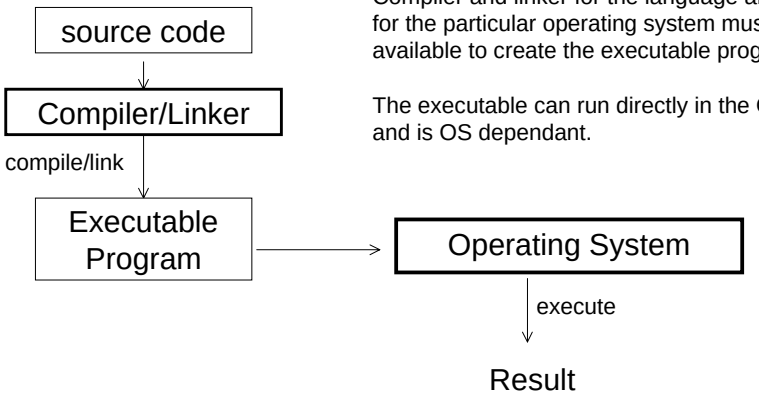
- There are 3 common ways program source code is translated and executed:
    - interpreted (eg. Perl, JavaScript)
    - byte-code compiled (eg. JAVA)
    - fully compiled (eg. C/C++, Pascal)
- Which method is used depends on the language.

Interpreted Programs



The interpreter must be available to execute the code.

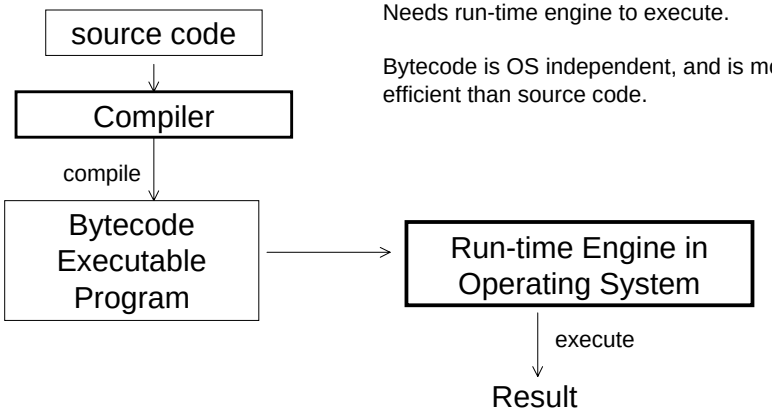
Fully Compiled Programs



Compiler and linker for the language and for the particular operating system must be available to create the executable program

The executable can run directly in the OS, and is OS dependant.

Bytecode Compiled Programs



Needs run-time engine to execute.

Bytecode is OS independent, and is more efficient than source code.

"Generations" of Languages

- 1st Generation: Machine Language – executable binary code
- 2nd Generation: Assembly Language
- 3rd Generation: High-level languages such as the imperative and functional languages.
- 4th Generation: Mostly built on top of 3rd generation languages, for specific applications
  - Eg. SQL for databases.
- 5th Generation: ???

"Generations" of Languages (cont'd)

- von Neumann's machines are based on the levels in the 1st - 3rd generations.

5th Generation Language

- There are major disagreements on what constitute 5th generation languages.
- The earliest mention was when Japan's Fifth Generation Project (started in the early 1980s) adopted Prolog.
  - Literature at that time calls "very-high level" languages like Prolog fifth generation languages.
- Today, some follow the language hierarchy and consider "environments" that uses visual or graphical development interfaces to create source language to be 5GL "languages".
  - Eg. IBM VisualAge for Java, Microsoft's Visual Studio.

In the lectures in this topic...

- We will look at different categories of languages important for Internet application development.
  - Mark-up languages: HTML, XML.
  - Web Scripting Languages: JavaScript, Jscript, VBScript, ECMAScript.
  - Other Languages: Java, Visual Basic, Perl, etc.

- Blank Page -