Advantages of Scripting Languages

Introduction

- Traditional programming languages are concerned with building a self contained environment that receives input and produces output.
- Most ``real-world'' computing however involves the use of *multiple* programs.
- For example:
  - Imagine a system that stores a list of numbers in a txt file, 1 per line.
  - Each number must be passed to a database as a parameter for an SQL query.
  - Each query returns a string representing an OS command that must be executed.
- What is the best way of achieving this scenario?

Option 1

- We could write all of this as a Java program:

```
public int getNumFromFile () {
    …
    return Integer.parseInt(BufferedReader.readLine());
}

public String executeSQL ( int c ) {
    …
    return SQL(“select command from table where command = ”+c);
}

public void executeCommand ( String command ) {
    …
    Runtime.exec(command);
}
```

Option 1 cont…

- What are the drawbacks of this approach?
- Languages such as Java stress efficiency, portability and maintainability.
- Their type systems are based upon hardware-level concepts.
- Examples of this include: fixed sized integers, floating point numbers, characters and arrays.
- So how can we re-write the problem?
- Through the use of a scripting language.

Scripting Languages

- Scripting languages stress flexibility, rapid development and dynamic checking.
- Their type systems embrace very high level concepts such as tables, patterns, lists and files.
- There a number of distinct groups that fall under the scripting language family.
- Languages such as Perl and Python are known as ``glue'' languages because they were designed to glue existing programs together.
- There are other extensible types of scripting languages used in the WWW also.

Option 2

- Consider the first problem again:

```
read –r  var1 < commands.txt
while $var1 –ne “”
    do
        echo “select command from table where command =“$var1 > query.txt
        mysql < query.txt > command
        read –r var2 < command
        exec $var2
        read –r var1 < commands.txt
    done
```

- Using a simple bash script we can solve the whole problem in less than 10 lines!*

* May not be exactly correct

What exactly is a scripting language?

- Scripting languages descend from two main types of ancestors.
- The first set of scripting languages are those designed to execute terminal commands in batch mode.
- Examples of this are bash scripts and ``.bat'' files for MS-DOS.
- The other type of scripting languages are those designed for text-processing and report generation.
- Examples of these include the very useful **sed** and **awk**.
- From these two languages grew Perl.
- Some other general purpose scripting languages about include Tcl, Ruby, Python, and VBScript.

Scripting and the WWW

- In the early-mid nineties, Perl was adopted to become the language of choice for server-side processing.
- One guy kept a lot of handy Perl scripts to track access to his home page. These scripts eventually morphed into the a fully fledged independent language known as PHP.
- This is the most popular server side scripting language.
- Other competitors to PHP include JSP and VBScript.
- For client side processing, there is Javascript, a very watered down version of Java created by Netscape 10 years ago.

Common Characteristics of Scripting languages.

1. Both Batch and Interactive use.
2. Economy of Expression.
3. Lack of declarations; simple scoping rules
4. Flexible dynamic typing.
5. Easy access to other programs.
6. Sophisticated Pattern matching.
7. High-level data types.

1 – Batch and Interactive Use

- Perl has a JIT compiler that reads the entire program before execution.
- Other languages are quite happy to only execute a line as soon as it has been read.
- Python, Tcl and Ruby will also accept commands from the keyboard.

2 – Economy of Expression

- To support rapid development and interactive use, scripting languages require very little boilerplate code.
- This can be illustrated very clearly by the following example:

```
public static void main(String args[])
{
    System.out.println(“Hello!”);
}

print “Hello!\n”
```

- Another example is reading a file word by word.