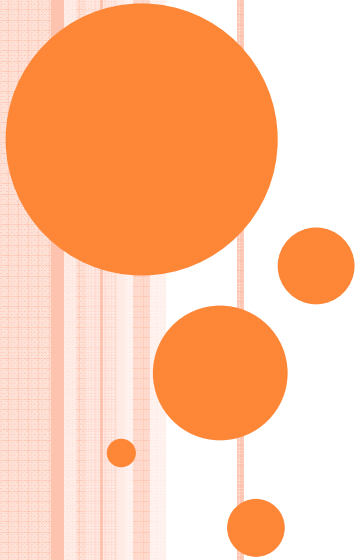

Lenguajes de scripting

CYPLP

Lenguajes de scripting

- Los lenguajes de programación tradicionales están destinados principalmente para la construcción de aplicaciones auto-contenidas:
  - Programas que aceptan una suerte entrada, la procesan de una manera bien entendida y finalmente generan una salida apropiada.
  - Sin embargo,

    muchos de los usos actuales en diferentes entornos, requieren la coordinación de múltiples programas.

Lenguajes de programación tradicionales vs LBS

- Los lenguajes convencionales tienden a **mejorar eficiencia**, mantenibilidad, portabilidad y detección estática de errores. Los tipos se construyen alrededor de conceptos a nivel hardware como enteros de tamaño fijo, punto flotante, caracteres y arreglos.

- Los lenguajes script tienden a **mejorar flexibilidad**, desarrollo rápido y chequeo dinámico. Su sistema de tipos se construye sobre conceptos de mas alto nivel como tablas, patrones, listas y archivos.

LBS

- Los lenguajes script de propósito general (Perl, Python) suelen conocerse como glue-languages.
- Se diseñaron para “pegar” programas existentes a fin de construir un sistema mas grande.
- Se utilizan como lenguajes de extensión, ya que permiten al usuario adaptar o extender las funcionalidad de las herramientas script.
- Son interpretados
- En general, débilmente tipados (tipificados), por lo tanto, … muy flexibles, aunque suelen ser menos eficientes en la ejecución

¿QUÉ ES UN LENGUAJES SCRIPT?

- Es difícil definirlos con precisión, aunque hay varias características que tienden a tener en común
- Estos lenguajes tienen dos tipos de ancestros:
  - interpretes de líneas de comando o "shells"
  - herramientas para procesamiento de texto y generación de reportes.

"Los lenguajes script asumen la existencia de componentes útiles en otros lenguajes. Su intención no es escribir aplicaciones desde el comienzo sino por combinación de componentes"

(John Ousterhout – creador de TCL)

Aspectos principales

- En los LBS las declaraciones son escasas o nulas y proveen reglas simples que gobiernan el alcance de los identificadores
- Por ejemplo, en Perl, cada identificador es global por omisión (hay opciones para limitar el alcance)
- En otros lenguajes (e.g., PHP y Tcl), cada cosa es local por omisión (un objeto global debe ser explícitamente importado)
- Python adopta la regla: “a una variable que se le asigna un valor es local al bloque donde aparece dicha asignación” (se puede cambiar esta regla con una sintaxis especial)

ASPECTOS PRINCIPALES

- Dado la falta de declaraciones, muchos LBS incorporan tipificación dinámica
- En algunos lenguajes el tipo de la variable es chequeada inmediatamente antes de su uso: e.g., PHP, Python, Ruby, y Scheme.
- En otros, el tipo de una variable (por ende su valor) será interpretado de manera diferente según el contexto: e.g., Rexx, Perl, y Tcl.

ASPECTOS PRINCIPALES DE LOS LBS
EXPRESIBILIDAD

JAVA

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

ADA

```
with ada.text IO; use ada.text IO;
procedure hello is
    begin
        put line(”Hello, world!”);
   end hello;
```

Perl, Python, o Ruby

```
print ”Hello, world!\n”
```

TIPIFICACIÓN DINÁMICA = FLEXIBILIDAD

- Ejemplo en Perl

```
$a = ”4”;
print $a . 3 . ”\n”; # ’.’ es la concatenación
print $a + 3 . ”\n”; # ’+’ es la suma
...
```

Dará la siguiente salida:

```
43
7
```

Facilidades avanzadas para

- Procesamiento de texto y generación de reportes
- Manipulación de Entrada/Salida de programas externos
- Pattern matching, búsqueda y manipulación
- La mayoría de los comandos están basados en Expresiones Regulares Extendidas (dir *.*)

Tipos de datos de alto nivel

- Los LBS son ricos en:
  - Conjuntos
  - Diccionarios
  - Listas
  - Tuplas, etc.

Por ejemplo:

- En muchos LBS es común poder indexar arreglos a través de cadenas de caracteres, lo que implica una implementación de tablas de hash y manejo de almacenamiento usando ”garbage collection”.

Dominios de Aplicación (general)

Principales Ejemplos:

- Lenguaje de comandos (shell)
- Procesamiento de texto y Generación de Reportes
- Matemática y Estadística
- Lenguajes de ”pegado”(GLUE) y de propósito general
- Extensión de Lenguajes

WWW como ejemplo especial

- CGI (Common Gateway Interface)
- Scripts Embebidos en Servidores
- Scripts Embebidos en Clientes
- Applets de Java
- Otros: XML.

EJEMPLO: JAVASCRIPT

```
<HTML>
 <HEAD>
      <SCRIPT LANGUAGE="JavaScript">
        <!--- function HolaMundo() { alert("¡Hola, mundo!"); }
        // --->
      </SCRIPT>
 </HEAD>
 <BODY>
 <FORM>
 <INPUT TYPE="button" NAME="Boton" VALUE="Pulsame"
 onClick="HolaMundo()">
  </FORM>
 </BODY>
</HTML>
```

Dentro de estos elementos se puede poner funciones en JavaScript

llamada al método alert (que pertenece al objeto window) que es la que se encarga de mostrar el mensaje en pantalla

onClick es un *evento*. Cuando el usuario pulsa el botón, el evento onClick se dispara y ejecuta el código que tenga entre comillas, en este caso la llamada a la función HolaMundo(), que tendremos que haber definido con anterioridad.