Llenguatges de Scripting

Un llenguatge de scripting (de guions) és un llenguatge de programació destinat a escriure programes que s'integrin i es comuniquin amb altres programes.

1.Motivació

L'any 1986, Donald Knuth proposa una tasca: escriure un programa per llegir un fitxer de text, determinar les n paraules més freqüents, i escriure una llista d'aquestes paraules en ordre alfabètic juntament amb la seva freqüència.
Knuth va presentar una solució en Pascal de 10 pàgines i en resposta, Douglas McIlroy va escriure un script de poques línies.

2. Història

Ancestres:

- shells: programari que proporciona una interfície al sistema operatiu.
- Processadors de textos i/o generador d'informes.

Evolucions:

- Rexx (IBM, 1979) o Perl (1987).
- Altres de propòsit general: Tcl, Python, Ruby, VBScript (Windows) i AppleScript (Mac).

World Wide Web (1990s):

- Perl més usat per scripting als servidors web.
- Scripts en Perl van evolucionar a PHP (1995).
- Perl també va influir el desenvolupament de Python (1991). També influït per molts llenguatges de programació.
- Altres competidors de Perl, PHP, i Python (codi obert) són JSP (Java Server Pages; per HTML o XML; SUN) i VBScript (Microsoft, 1996).

3. Característiques

LSs vs LPs tradicionals:

- Els LS estan pensats per crear aplicacions combinant components.
- Desenvolupar amb LSs és 5 a 10 vegades més ràpid.
- Executar amb LSs és de 10 a 20 vegades més lent.

- Els LS són normalment interpretats, amb compilació Just-in-time o les dues.
- Molts LSs són feblement tipats (Excepció: Python).
- Molts LS tenen tipat dinàmic (Excepció: RPG).

Alguns llenguatges de scripting han evolucionat i poden ser usats com a llenguatges tradicionals (amb menys eficiència).

Compilació just-in-time (JIT):

- Es compila el codi en temps d'execució (a codi màquina o a algun tipus de bytecode)
- Pot aplicar optimizacions que depenen del context d'execució.
- Positiu: el codi resultant pot ser molt més eficient.
- Negatiu: la compilació en execució és ineficient.
- Es pot aplicar parcialment: només en certes construccions. Per exemple: expressions regulars (matching).
- Pot combinar-se amb la compilació (estàtica) a bytecode:
    - Primer es compila el llenguatge a bytecode.
    - S'aplica JIT al bytecode per obtenir codi màquina més eficient.

Característiques principals:

- Permeten tant ús en batch com interactiu.
- La majoria tracten l'entrada línia a línia. (Excepció: perl que requereix llegir tot el programa abans de tractar-lo).
- Altres accepten instruccions per línia de comandes (ex: Rexx, Python, Tcl i Ruby). Han de poder ser interpretats (sense compilació just-in-time).
- Economia d'expressions:
    - Afavorir el desenvolupament ràpid i l'ús interactiu.
    - Fort ús de símbols de puntuació i identificadors molt curts (Perl).
    - Més paraules en anglès i menys puntuació (Rexx, Tcl,...).
    - Eviten les declaracions extenses i les estructures de nivell superior

Java

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

Python

```python
print("Hello, world!")
```

PHP

```
Hello, world!
```

- Absència de declaracions. Regles simples d'establiment d'àmbit (scoping).

Declaracions:

- En molts LS no hi ha declaracions.
- Per exemple, l'assignació dóna el tipus.

Àmbit:

- En alguns tot és global per defecte (Perl).
- En alguns tot és local per defecte (PHP, Tcl).
- Pot haver-hi regles com ara que l'assignació defineix la localitat.

- Tipat dinàmic flexible:

  Relacionat amb l'absència de declaracions, la majoria dels LPs són tipats dinàmicament:

  - En alguns, el tipus d'un variable es comprova just abans de ser usada (PHP, Python, Ruby).
  - En altres, es pot interpretar diferent en diferents àmbits (Rexx, Perl, Tcl).
- Fàcil comunicació amb altres programes:

  Donen moltes opcions predefinides per executar programes o operacions directament sobre el SO:

  entrada/sortida
  manipulació de fitxer i directoris
  manipulació de processos
  accés a bases de dades
  sockets (APIs) per comunicacions entre processos
  sincronització i comunicació entre processos
  protecció i autorització
  comunicació en xarxa
- Pattern matching i manipulació de strings sofisticada:

  És una de les aplicacions més antigues:

  - Facilita la manipulació de l'extrada i la sortida textual de programes externs.
  - Tendeixen a tenir facilitats molt riques per fer pattern matching, cerca i manipulació de strings.
  - Normalment es basen en formes esteses de les expressions regulars.
- Tipus de dades d'alt nivell:
  - S'inclouen com predefinits tipus d'alt nivell com ara: sets, bags, maps, lists, tuples,...
  - No es troben en llibreries, sinó que fan part del llenguatge.
  - Per exemple, és habitual tenir arrays indexats per strings com part del llenguatge (que s'implementen amb taules de hash).
  - S'utilitzen *garbage collectors* per gestionar l'espai.

4. Dominis d'aplicació

Llenguatges de comandes shell (JCL, csh, tcsh, ksh, bash…):

- Ús interactiu.
- Processament batch (no molt sofisticats).
- Manipulació de noms de fitxers, arguments i comandes.

- Enganxar (glue) diversos programes.

Llenguatges de comandes shell.

- Exemple bash:

```
for f in *.ps
do
    ps2pdf $f
done
```

- Exemple bash:

```
for arg in "$@"
do
    index=$(echo $arg | cut -f1 -d=)
    val=$(echo $arg | cut -f2 -d=)
    case $index in
        X) x=$val;;
        Y) y=$val;;
    esac
done
((result=x+y))
echo "X+Y=$result"
```

`bash exemple.sh X=45 Y=30` dóna `X+Y=75`.

Processament de textos i generació d'informes (sed, awk, Perl, …):

- Els LS estan fortament orientats al tractament de strings.
- Les comandes són strings que es divideixen en paraules.
- Els valors de les variables són strings.
- Se'ns permet extreure substrings.
- Concatenar i moltes més opcions…

Processament de textos i generació d'informes.

- Exemple awk:

```
awk '{print $1 " " $2 " " ($1 + $2)/2} < notes.txt'
```

- Exemple awk:

```
BEGIN {
    print "user\thome"
    FS=":"
}

{
    print $1 "\t" $6
}

END {
    print "end"
}
```

```
awk -f exemple.awk < /etc/passwd
```

Matemàtiques i estadístiques (Maple, Mathematica i Matlab):

- gran suport pels mètodes numèrics.
- manipulació simbòlica de fòrmules.
- visualització de dades.
- modelat matemàtic.

Orientats a aplicacions científiques i en l'enginyeria.

Llenguatges com S i R per computació estadística:

- Inclou arrays i llistes multidimensionals.
- funcions de primera classe.
- laziness (call-by-need).
- operacions de selecció (slice) sobre arrays.
- extensió il·limitada (noms, objectes, …).

Exemple Matlab:

```
% Create and plot a sphere with radius r.
[x,y,z] = sphere;       % Create a unit sphere.
r = 2;
surf(x*r,y*r,z*r)       % Adjust each dimension and plot.
axis equal              % Use the same scale for each axis.

% Find the surface area and volume.
A = 4*pi*r^2;
V = (4/3)*pi*r^3;
```

Exemple R: Multiplica `[1,2,3]` per `[[3,1,2], [2,1,3], [3,2,1]]`.

```
v1 <- c(1,2,3)
v2 <- matrix(c(3,1,2,2,1,3,3,2,1), ncol = 3, byrow = TRUE)
v1 %*% t(v2)
```

Scripting de propòsit general: Perl, Tcl, Python, Ruby.

Llenguatges d'extensió: llenguatges de scripting que permeten fer scripts per una determinada aplicació:

- Per gràfics Adobe (Photoshop) es poden fer scripts JavaScript, Visual Basic o AppleScript.
- Per GIMP es poden fer scripts en Scheme, Tcl, Python i Perl.
- Per emacs hi ha un dialecte de Lisp anomenat Emacs Lisp.

- Exemple: Cridant LUA des de C.

```
print "Start"
for i = 1,10 do
    print(i)
end
print "End"
```

```
#include <lua.h>

int main() {
    lua_State* L = lua_open();
    lua_baselibopen(L);
    lua_dofile(L, "exemple.lua");
    lua_close(L);
}
```

Web scripting: Perl, PHP, Python, JavaScript, Cold Fusion, ...

- Exemple PHP: Pàgina de registre a Jutge.org:

```php
function registration_page () {
    extract(vars());

    $pag->use_captcha = true;

    if (isset($_POST[submit])) {

        $record = array(
            name            => trim($_POST[name]),
            email           => trim($_POST[email]),
            parent_email    => trim($_POST[parent_email]),
            birth_year      => trim($_POST[birth_year]),
            country_id      => trim($_POST[country]),
            agreement       => trim($_POST[agreement]),
            captcha         => $_POST['g-recaptcha-response'],
        );
        ⋮
    } else {
        $countries = $dbc->Countries->select(country_id, eng_name);
        ⋮
    }
}
```