Anexo: Lenguajes de Scripting

Departamento de Sistemas Telemáticos y Computación (GSyC)

gsyc-profes (arroba) gsyc.es

Febrero de 2009

Contenidos

Los lenguajes de scripting

- Son interpretados (opuesto a compilados)
- Suelen ser de alto nivel
- Orientados a tareas *sencillas*, interoperar con otros lenguajes, tareas de administración del S.O., procesar ficheros de texto
- Es frecuente que los programas hechos en lenguajes de scripting (también llamados *scripts*) sean cortos y se desarrollen rápidamente

En principio no son adecuados para:

- El driver de un dispositivo
- Un sistema de tiempo real
- Una aplicación bancaria
- Un compilador/intérprete de lenguaje
- Una aplicación que requiera mucho cálculo
- Una aplicación de la que dependan vidas humanas
- ...

Adecuados para

- Prototipado rápido
- Utilidades administración sistemas

Ejemplo :

```
Leer todos los correos que lleguen a un servidor POP3
Meter todos sus datos en una base de datos
Extraer (del texto) todos los enlaces a páginas cuyos
   servidores estén en mi subred
Extraer de los anexos todos los archivos jpg,
   integrarlos en un único jpg
Encriptarlos en PGP, enviarlos tal máquina por FTP y generar un
   informe en Latex, PDF y HTML
```

Los lenguajes de scripting podemos dividirlos en

- Clásicos: awk, shell de MS-DOS, Shells de Unix (ksh, csh, sh, bash) , tcl
- Modernos: Perl, Python, Visual Basic Script, JavaScript, Ruby

Lenguaje Compacto:
Aquel que tiene pocas particularidades y en el que resulta fácil conocer todas sus características[1]

- Ejemplos de lenguaje Compacto: Pascal, C, Python
- Lenguajes no compactos: C++, Perl, java

[1] La definición y estos ejemplos podrían ser discutibles

El lenguaje Perl

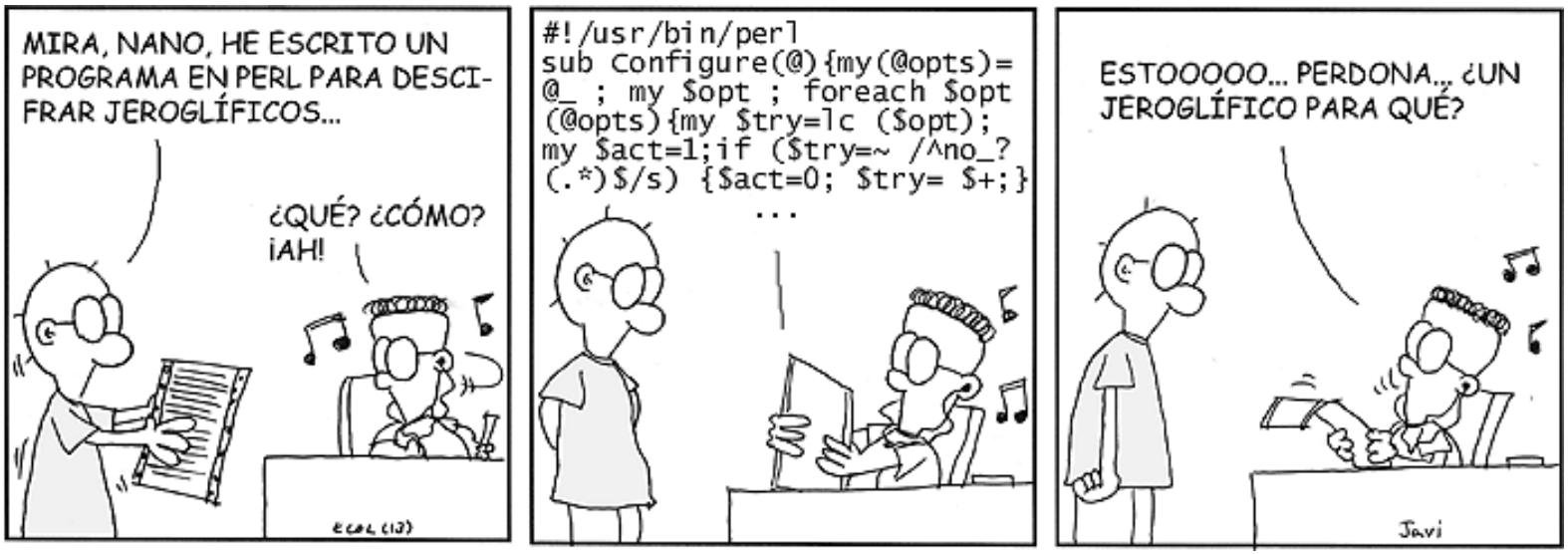
- Lenguaje de scripting de alto nivel. Primer lenguaje de scripting *moderno*
- Derivado de C, sed, awk
- Bastante eficiente, para ser interpretado
- Dispone de repositorio centralizado de librerías: CPAN *Comprehensive Perl Archive Network*
- Perl 1.0 aparece en 1987. La versión 4.0 en 1991. En su época no tenía rival. Muy popular para las primeras páginas web dinámicas

- Sintaxis y una semántica muy rica. Críptica y farragosa
- Su diseño original va contra principios elementales de Ingeniería del Software: variables globales declaradas implícitamente, débilemente tipado, paso de parámetros a funciones complicado. . . Posteriormente se mejora, pero resulta un *parche*
- Da mucha libertad al programador, libertad para cometer errores

Perl puede ser muy oscuro

Ejemplo: Subrutina para ordenar un array de cadenas

```
sub ordenar_resultado{
@resultado =
        map  { $_->[0] }
        sort { $a->[1] <=> $b->[1] or $a->[2] <=> $b->[2] }
        map  { [ $_, (split /\s*,\s*/)[0,1] ] } @resultado;
}
```



Un principio básico en Perl es
There's more than one way to get things done
Eso tiene alguna ventaja, pero muchos inconvenientes, especialmente cuando trabajan varias personas, hace difícil homogeneizar código, depurar, mantener
Ruby mantiene este principio

Ejemplo:

```
while (1) {
    $buf = $fp->read($blocksize);
    if (not $buf) { last }
    $conn->send($buf);
}

LOOP:
while (1) {
    $buf = $fp->read($blocksize);
    if (!$buf) {
        last LOOP;
    }
    $conn->send($buf);
}

while (1) {
    last unless $buf = $fp->read($blocksize);
    $conn->send($buf);
}

while (1) {
   last unless $buf = read $fp $blocksize;
   send $conn $buf;
}
```

Opuesto a lenguajes como Python donde las cosas se hacen de 1 forma. Ej:

```
while 1:
    buf = fp.read(blocksize)
    if not buf:
        break
    conn.send(buf)
```

Principio: *Do The Simplest Thing That Could Possibly Work*

El lenguaje Ruby

- Yukihiro Matsumoto, 1995. Licencia libre
- Intenta mejorar perl, del que parte
- Orientado a Objetos puro
  Ejemplo: el entero 1 es una instancia de la clase `Fixnum`
- Mucha variabilidad sintáctica (como perl). Pero manteniendo la legibilidad (no como perl)
- Buena parte del éxito actual de ruby se debe a *Ruby on Rails*.
  - Entorno de trabajo (*framework*) para hacer aplicaciones web. Aparece en 2004.
  - Posteriormente (año 2005) aparece *django*, entorno similar pero para python

```
If you like Perl, you will like Ruby and its syntax.
If you like Smalltalk, you will like Ruby and its semantics.
If you like Python, you may or may not be put off by the huge
difference in design philosophy between Python and Ruby/Perl.
```

(The Ruby FAQ)

El lenguaje Python

(Ver tema "Programación en python")