

PERL

¿Que es Perl?

Perl es un lenguaje interpretado, Perl compila los programas antes de ejecutarlos. Por eso se habla de “*scripts*”, y no de programas. **Perl** significa “*Practical Extraction and Report Language*”, algo así como “*Lenguaje Práctico de Extracción y de Informes*”

Programa Básico

```
#!/usr/bin/perl  
#  
# Programa para hacer lo evidente  
#  
print 'Hola mundo.';
```

La primera línea

Esta sentencia “**#!/usr/bin/perl**” indica al sistema operativo que lo que sigue es un “**script**” de Perl y con este es que se debe ejecutar. Dicho programa está en “**/usr/bin**”. Por tanto la secuencia “**#!**” es reconocida por Unix no por Perl.

Comentarios y sentencias

Los comentarios pueden ser insertados en un programa con el símbolo “**#**”, y cualquier cosa desde el símbolo “**#**” hasta el final de la línea es ignorada (con la excepción de la primera línea). La única forma de alargar los comentarios sobre varias líneas es usar “**#**” en cada línea; todo lo demás es una sentencia en Perl la cual debe terminar con un punto y coma, como la última línea de arriba.

Impresión simple

La función “***print***” muestra al exterior alguna información. En el ejemplo anterior, imprime literalmente la cadena “***Hola mundo***”. y por supuesto la sentencia termina en un punto y coma.

Asegúrese de que el fichero es ejecutable usando el comando

```
chmod u+x progname
```

En la línea de comandos de UNIX, donde “**programa**” es el nombre del archivo del programa. Ahora, para ejecutar el programa teclee cualquiera de la siguientes instrucciones de línea de comandos.

perl programa

./programa

programa

Tipos de datos:

Perl permite representar los tipos de datos reales, enteros, cadenas de caracteres y el tipo booleano.

Los tipos numéricos (reales y enteros).

Los valores numéricos expresados literalmente se presentan en forma de valores reales codificados en doble precisión. Este formato interno se utiliza para todas las operaciones aritméticas. Por ejemplo:

$\$x = 0.897;$ *# un real*

$\$y = 6.23e-24;$ *# un real*

$\$n = 567;$ *# un entero*

$\$i = -234;$ *# un entero*

Nota: *El que todas las variables contengan un \$ significan que representan un escalar.*

Las cadenas de caracteres.

Las cadenas de caracteres se especifican literalmente por medio de una sucesión de caracteres delimitada por comillas ("..") o apóstrofes ('..'). Estas dos representaciones se distinguen por la interpretación hecha por Perl de las cadenas de caracteres. Cuando van delimitadas por comillas (".."), toda variable referenciada en el interior de la cadena se evalúa y se reemplaza por su valor. Por ejemplo, las instrucciones siguientes:

```
$wld = "mundo";  
$str = "¡Hola $wld!";
```

Las cadenas de caracteres delimitadas por apóstrofes se dejan intactas. Por ejemplo:

```
$str = '¡Hola $wld!';
```

asigna al escalar “***\$str***” la cadena de caracteres “***¡Hola \$wld!***”.

Otra forma

```
$msg = <<SALUDO;  
hola,  
buenos días,  
adios,  
SALUDO
```

El tipo booleano.

El tipo booleano existe, al igual que en C, de modo implícito, es decir, un número es falso si es igual a cero y verdadero en cualquier otro caso. Como el cero está asociado a la ristra vacía (""), ésta también equivale al valor falso.

Los escalares.

El escalar representa el tipo básico en Perl. Permite representar enteros, reales y cadenas de caracteres. Las variables de tipo escalar van precedidas por el símbolo "\$". A continuación veremos algunos ejemplos:

```
$real = 4.53;
```

```
$entero = -45;
```

```
$cadena = "Hola";
```

Las variables en Perl se asignan de manera dinámica y se les asigna un valor predeterminado en función del contexto. En un contexto numérico el valor predeterminado es 0, mientras que en un contexto de cadena de caracteres el valor predeterminado es la cadena vacía ("").

Los arrays.

Un arreglo “**array**” es una lista de datos de tipo escalar. Cada elemento de la lista es una variable escalar a la que se le asocia un valor. Las variables de tipo array se identifican por el prefijo arroba “@”. Por ejemplo:

```
@numeros = (2, 1, 667, 23, 2.2, 5, 6);
```

```
@letras = ("perro", "gato", "león");
```

```
@mezcla = ("hola", 23, "adios", 31.234);
```

Las listas asociativas (arrays de indexación literal).

Una lista asociativa está indexada por cadenas en lugar de por números. Se utiliza % para definir el tipo de lista asociativa y un elemento está indexado por el anterior formando ambas parejas del tipo (*clave*, *valor*).

```
%animales = ("perros", 1000, "gatos", 256);
```

```
$id = $animales{"gatos"}; # $id = 256
```

Estructura de decision.

Los bloques siempre requieren que se encierren entre corchetes { }, a diferencia de la programación en C.

```
if ($cadena)  
{  
    print "La cadena no esta vacía\n";  
}  
else  
{  
    print "Cadena vacía\n";  
}
```

Estructuras de repetición.

Foreach

```
@animales = ("perro", "gato", "león");  
foreach $lista (@animales)  
{  
    print "$lista\n";  
}
```

For

```
for ($i = 0; $i < 10; ++$i)  
{  
    print "$i\n";  
}
```

while and until

```
#!/usr/local/bin/perl  
print "Nombre de usuario? ";  
$usuario = <STDIN>;  
print "palabra clave? ";  
$clave = <STDIN>;  
chop $clave;  
while ($clave ne "luis")  
{  
    print "Nombre de usuario o clave incorrecta? ";  
    $clave = <STDIN>;  
    chop $clave;  
}
```

Manejo de archivos

En el siguiente ejemplo ilustra como se pueden manipular los archivos desde un programa en perl.

```
#!/usr/local/bin/perl  
$archivo = '/etc/passwd';  
open(ARCH, $archivo);  
@lineas = <ARCH>;  
close(ARCH);  
print @lineas;
```

Los archivos pueden ser abiertos en diferentes modos, como por ejemplo solo de lectura, lectura y escritura, etc. a continuación se ilustra las diferentes formas:

```
open(ARCH, $archivo); # Entrada  
open(ARCH, ">$archivo"); # Salida  
open(ARCH, ">>$archivo"); # Concatenar
```