Perl

Tomás P. de Miguel
DIT Univ. Politécnica de Madrid
tmiguel@dit.upm.es

¿Qué es Perl?

- Es un lenguaje de programación de propósito general
  - Nació para hacer informes
- Diseñado por administradores
  - para desarrollar procedimientos de administración
- Ideal para manipular el sistema
  - textos, ficheros y procesos
- Interpretado
  - #! /usr/bin/perl
  - Se puede compilar a otros lenguajes
- Mezcla de C, Shell y filtros de Unix
  - también aplicaciones de Internet
- Simple y potente
- Conciso, pero puede ser ilegible

Bibliografía

- Grupos de noticias: comp.lang.perl….
- Servidor Perl
  - http://www.perl.com
  - FAQ: http://www.perl.com/faq
- Centro de distribución de cosas Perl
  - Comprehensive Perl Archive Network
  - http://www.perl.com/CPAN
- Instituto Perl para controlar los desarrollos
  - http://www.perl.org
- Libros: la serie del camello de O´Reilly
  - Programming Perl, Larry Wall and Randal Schwartz

Intérprete de comandos

- Enviando los datos al intérprete
  - **`echo “print ‘Hola!’;” | perl –`**
- Como parámetro del intérprete
  - **`perl –e ‘print “Hola!”;’`**
- Escribiendo un script en un fichero
  - **Fichero hola:**

```
#! /usr/bin/perl –w
print “Hola!”;
```

  - **`> hola`**

Estructura

- Es un lenguaje de formato libre
- No es necesario declarar algo para poder utilizarlo
  - Excepto subrutinas y esquemas de formato
- El punto y coma es terminador
  - Dentro de un bloque no es necesario
- El espacio en blanco solo es necesario si la construcción es confusa
- Lo comentarios como en shell
  - # hasta fin de linea
  - = sección de documentación

Tipos de datos básicos

- Escalares
  - Son números o cadenas de caracteres
  - Perl hace conversión automática de tipos
  - Las variables se nombran con $
- Arrays
  - Son listas ordenadas de escalares
  - Se acceden con un índice
  - Las variables se nombran con @
- Tablas de claves
  - Son conjuntos de pares: clave/valor
  - Se acceden a través del índice clave
  - Las variables se nombran con %

Cadenas de caracteres

cadena	genérico	significado
´ ´	q//	literal
” ”	qq//	literal
` `	qx//	programa a ejecutar
()	qw//	lista de palabras
//	m//	reconocimiento de patrones
s///	s///	sustitución
y///	tr///	trasliteración

Listas

- Es una agrupación ordenada de valores escalares
  - `(1,2,3)`
  - `("uno","dos","tres")`
- La forma genérica de creación es:
  - `qw/lunes martes miércoles jueves/`
- Asignaciones
  - `$dia = 15;`
  - `@fecha= ($dia, 10, 2001);`
  - `$mes= $fecha[1]; # valor 10`
  - `($d,$m,$a) = @fecha;`
  - `split(/patrón/, cadena, [límite])`

Tablas de claves

- Son tablas hash
- Se ven como listas formadas por pares: clave, valor
- `%dep = (´.o´, ´.c´, ´.ps´, ´.tex´);`
- `$v= $dep{".o"};`
-
```
%dep = (
    ".o"      => ".c",
    ".ps"     => ".tex"
);
```

Declaraciones y ámbitos

- La declaración solo es obligatoria para la funciones y los formatos
- Las variables se crean automáticamente la primera vez que se usan.
- Una variable se puede declarar cuando se quiere limitar la ámbito
  - Ámbito dinámico
    - Visibilidad global, pero efecto local
      - por ejemplo, dentro de una función
    - local
  - Ámbito léxico
    - Visibilidad local y efecto local
    - my

Sentencias

- Sentencia simple
  - expresión;
- Sentencia compuesta
  - { }
- Bloque
  - Condicionales
  - Bucles
  - Modificadores
  - Control de bucle
  - Goto

Condicionales

- Sentencia if
  - if (expresión) {bloque} else {bloque}
  - if (expresión) {bloque}
    elsif (expresión2) {bloque}
    . . .
    elsif (expresiónN) {bloque}
    else {bloque}
- Sentencia unless
  - unless (expresión) {bloque} else {bloque}

Bucles

- **while (expresión) {bloque}**
  - `while (<FICH>) { print "$_\n"; }`
- **do {bloque} until (expresión)**
- **`continue`** se ejecuta antes de cada iteración
- **`last`** termina con el bucle
- **`next`** termina la iteración
- **`redo`** repite la iteración sin evaluar la expresión
- **`for ($i= 1; $i < 10; $i++) { . . . }`**
  - `$i= 1;`
    `while ($i < 10) {…}`
    `continue { $i++; }`
- **foreach var (lista) { bloque }**
  - Tambien puede tener un continue

Modificadores

- A cada sentencia simple se le puede aplicar una condición o incluso un bucle
- sentencia if (expresión);
- sentencia unless (expresión);
- sentencia while (expresión);
- sentencia until (expresión);

Variables especiales

- Son variables predefinidas
- Con un significado especial

$_	La entrada o el patrón encontrado por defecto
$ARG	La entrada o el patrón encontrado por defecto
$.	En un fichero, el número de la última línea leída
$0	Nombre del fichero script en ejecución
$/	Separador de registros (por defecto, fin de línea)
$$	Pid del proceso que ejecuta el script
$@	Mensaje de error del operador eval
$!	El valor de la variables errno
$,	Separador del operador print

Listas y tablas especiales

@ARGV	Línea de comandos con los argumentos del script
@INC	Lista de sitios donde se buscan los scripts que evaluan do, require o use
%INC	Tabla con los nombres de fichero de cada fichero que se incluye con do o require
%ENV	Tabla con la variables de entorno del proceso
%SIG	Tabla con los manejadores de señales configurados para el proceso

Manejadores de ficheros

ARGV	Accede a todos los ficheros de la línea de comandos. Se suele escribir como <>
STDIN	Entrada estándar
STDOUT	Salida estándar
STDERR	Salida de error

Operadores de asignación

=	**=	++=	*=	&=	<<=	&&=
		-=	/=	|=	>>=	||=
		.=	%=	^=		
			x=			

Búsqueda de patrones

- Es un potente sistema de búsqueda basado en expresiones regulares
- Se utiliza en muchos contextos:
  - Operador sustitución
    - s/patrón/nueva_cadena/
  - La expresión de un patrón /patrón/ aplica sobre la variable $_
    - if /patron/ {entonces} else {otra_cosa}
  - Se puede usar sobre cualquier variable
    - $variable =~ /patrón/
      - "1" si lo encuentra
      - "" si no lo encuentra

Sintaxis de expresiones regulares

- /ab*c/
- /^.*$/
- [aeiouAEIOU]
- [a-z0-9]
- $_ = "a xxx c xxxxxxxx c xxx d"; /a.*c.*d/;
- $_ = "a xxx c xxxxxxxx c xxx d"; /a.*?c.*d/;
- /fred(.)barney\1/;
- /a(.)b(.)c\2d\1/;
- /\bFred\b/; # no Frederich
- (song|blue)bird # songbird bluebird
- /^\/usr\/etc/ o usando m#^/usr/etc#
- $what = "bird"; /\b$what\b/

Abreviaturas de caracteres

construcción	equivalencia	opuesto	equivalencia
\d (a digit)	[0-9]	\D (digits, not!)	[^0-9]
\w (word char)	[a-zA-Z0-9_]	\W (words, not!)	[^a-zA-Z0-9_]
\s (space char)	[\r\t\n\f]	\S (space, not!)	[^ \r\t\n\f]

Funciones

- sub nombre {bloque}
- sub nombre (parámetros) {bloque}
- Se pueden crear en tiempo de ejecución
  - $func_ref = sub {bloque};
- Llamada
  - nombre(argumentos);
  - nombre argumentos;
  - &nombre;
- A través de una referencia
  - &$func_ref(argumentos);
  - &$func_ref;