DESCRIPCIÓN DE LOS ASPECTOS FUNDAMENTALES DEL LENGUAJE DE PROGRAMACIÓN PERL

Andrey Francisco Pérez Hidalgo
Universidad de Costa Rica, Escuela de Ciencias de la Computación e Informática
San José, Costa Rica
andreyfperez@gmail.com

Abstract

The syntax of the programming languages spreads to look like each other more and more to the human natural language, an example of this is Perl. The language Perl takes characteristic of other languages like C, Lisp, AWK, sh, among others. This language is used to develop applications in many fields like for example the web. Perl uses an interpreter program called perl, and its structure is formed mainly by different types of data, control structures and subroutines. The great advantage of Perl is that it is available many operating systems.

Key words: Perl, programming language, interpreter, perl, structures of the language, regular expressions.

Resumen

Las sintaxis de los lenguajes de programación tienden a parecerse cada vez más al lenguaje natural humano, un ejemplo de esto es Perl. El lenguaje Perl toma características de otros lenguajes como C, Lisp, AWK, sh, entre otros. Este lenguaje es usado para desarrollar aplicaciones en muchos campos como por ejemplo la *web*. Perl usa un intérprete llamado perl, y su estructura está formada principalmente por diferentes tipos de datos, estructuras de control y subrutinas. La gran ventaja de Perl es que está disponible para gran cantidad de sistemas operativos.

Palabras claves: Perl, lenguaje de programación, intérprete, perl, estructura del lenguaje, expresiones regulares.

1. Introducción

La idea que se tiene al desarrollar un lenguaje de programación nuevos es hacerlo su sintaxis lo más parecida al lenguaje natural de las persona, esto con el fin de hacerlo fácil de aprender, de entender y de usar, así facilitar la labor de los programadores, uno de los lenguajes que han surgido bajo esta idea es Perl.

En este artículo se hará una descripción general del lenguaje Perl, luego se analizarán las características y usos que se le dan. Posteriormente, se analizará el interprete del lenguaje llamado perl; luego, se describirá la forma como está estructurado el lenguaje. Finalmente, se analizarán la disponibilidad con la que cuenta Perl, así como las principales ventajas y desventajas tanto del lenguaje como del intérprete del lenguaje.

2. El lenguaje Perl

Perl es un lenguaje diseñado por Larry Wall en 1987, que originalmente fue desarrollado para ser un lenguaje de manipulación de texto [4], sin embargo con el pasar de los años se ha formado una verdadera comunidad de personas que utilizan Perl para el desarrollo de interfaces gráficas de usuario (GUI), desarrollo de páginas *web*, administración de sistema, programación en red, entre otras aplicaciones [2] [3].

Perl fue creado con el fin de que fuera un lenguaje fácil de usar, completo y eficiente, en lugar de que fuera compacto y elegante. Originalmente este lenguaje fue llamado "Pearl", que viene de "la parábola de la perla", sin embargo ya existía un lenguaje llamado PEARL, por lo que Wall decidió cambiar el nombre a "Perl", lo cual no significa nada, aunque muchas personas lo llaman PERL ("*Practical Extraction and Report Language*" o Lenguaje Práctico para la Extracción e Informe) [1].

Muchas de las características de este lenguaje están basadas en otros lenguajes como C/C++, el lenguaje interpretado *shell*, AWK, Lisp, entre otros []. En la siguiente sección se hará una descripción de estas y otras características.

3. Características de Perl

Perl tiene como principales características la facilidad de uso, el soporte para diferentes tipos de programación como lo son la orientada a objetos, la estructural y la programación funcional, además posee un robusto sistema de procesamiento de texto y una gran cantidad de módulos (lo que en otros lenguajes de programación como Java es llamado "bibliotecas") [5].

Como se menciono en la sección anterior, Perl toma características de otros lenguajes de programación. Su estructura está basada en bloques al estilo de C, lo que lo convierte en un lenguaje imperativo, con variables, expresiones, asignaciones, delimitación de bloques de código mediante llaves, estructuras de control y subrutinas [4] [5].

Perl también toma características del lenguaje interpretado *shell*, usando símbolos identificadores de tipo (*sigil*) para las variables y gran cantidad de funciones para la realización de tareas comunes y acceso a los recursos del sistema. Otras características heredadas de otros lenguajes son: las listas (Lisp), los *hashes* (AWK) y las expresiones regulares (*sed*) [4] [5].

Perl es un lenguaje práctico, lo que quiere decir que no determina estrictamente una estructura a la hora de programar (incluye características a la hora de ser usadas, tiene tolerancia de excepciones, y utiliza heurística para resolver ambigüedades sintácticas). Sin embargo, esta practicidad hace que muchas veces sea muy difícil la detección de errores [4].

A continuación, se mencionarán algunos de los principales usos que se le han dado al lenguaje desde su creación.

4. Usos del lenguaje

Gracias a la gran variedad que han surgido desde la creación de Perl, este lenguaje se ha convertido en pieza fundamental para el diseño de aplicaciones en diversos campos. Algunos de estos usos son [1]:

- Junto con Phyton y PHP es uno de los lenguajes más usados en la implementación de aplicaciones *web*.
- Es usado desde que surgió la *web* para escribir *scripts* CGI [6].
- Es un componente integral de la solución LAMP para el desarrollo de aplicaciones *web* [7].
- Hay muchos proyectos grandes escritos en Perl como lo son: Slash, IMDb y UseModWiki.
- Utilización en sitios de Internet con gran cantidad de visitas como “Amazon” y “Ticketmaster”.
- Usado como “lenguaje pegamento”, para unir interfaces, componentes y sistemas enteros.
- Utilizado en aplicaciones que requieren procesar gran cantidad de datos, y para la creación de informes.
- Creación de programas de administración de sistemas de propósito general.
- Utilizado también en aplicaciones que requieran un desarrollo rápido y un manejo seguro de gran cantidad de datos (ejemplo: finanzas y bioinformática).

En la siguiente sección se hará una descripción del intérprete, denominado “perl”, sobre el que está implementado el lenguaje Perl.

4. Interprete perl

Perl está implementado en un interprete escrito en C llamado perl (la diferencia con el nombre del lenguaje es la “p” minúscula), dicho interprete está basado en una arquitectura orientada a objetos, y todos los elementos de Perl (listas, *hashes*, etc) son representadas como estructura C [5].

4.1 Ejecución de un programa Perl

La ejecución de un programa escrito en Perl se divide en dos fases [5]:

Tiempo de compilación: en esta fase se crea el árbol sintáctico del texto del programa, luego el árbol es optimizado antes de iniciar la ejecución del programa**.**

Tiempo de ejecución: en esta fase se ejecuta el programa siguiendo el árbol creado en la fase anterior

Lo anterior pone en evidencia una de las desventajas de Perl y su interprete, la cual es que cada vez que se corre un programa debe ser compilado, lo que lo hace más lento en tiempo de ejecución que otros lenguajes [5].

4.2 Parseo de Perl

Perl posee una gramática dependiente del contexto, la cual puede ser afectada en la ejecución de un programa, por lo que no se pueden utilizar parseadores como Lex/Yacc. Por esta razón el intérprete incluye su propio analizador léxico que se combina con Bison para parsear Perl [5].

4.3 Mantenimiento del intérprete

Debido a que existe toda una comunidad de colaboradores alrededor de Perl, expandiendo y optimizando tanto el intérprete como la colección de módulos, el dar mantenimiento al intérprete se vuelve cada vez más difícil.

Para lograr un óptimo mantenimiento y asegurar que el intérprete funcione correctamente, Perl posee gran cantidad de *test* funcionales, los cuales se encargan de probar al intérprete y sus módulos [5].

A continuación, se hará una descripción de la estructura del lenguaje Perl.

5. Estructura de Perl

La estructura de Perl está formada principalmente por diferentes tipos de datos, estructuras de control y subrutinas.

5.1 Tipos de datos

Perl tiene tres tipos de datos, cada uno se identifica mediante un *sigil*, los tipos son:

Escalares: Son valores como números, hileras (*strings*) o referencias. Se identifican mediante el *sigil* "$" [5]. Ejemplos:

```
$numero = 26;
$hilera1 = "Hola";
$hilera2 = 'Adiós';
```

Las hileras pueden convertirse en números y viceversa, además se pueden usar como booleanos.

Ejemplo:

```
$numero = “35 años”;  # = 35
$bool = ‘0’;    # = falso
```

Listas: Es una colección ordenada de datos escalares, la variable asociada se denomina arreglo (*array*) y los elementos pueden accederse mediante un índice. Se identifican mediante el *sigil* “*@*” [5].

Ejemplo:

```
@lista = (1, 2, 3, 4);
```

Hashes: Es un mapeo de hileras (claves) a escalares (valores), también se le llama memoria asociativa y sus elementos pueden accederse mediante la clave. Se identifican mediante el *sigil* “%” [5].

Ejemplo:

```
%edad = (Pedro => 22,
         Juan => 35);
```

5.2 Estructuras de control

Las estructuras de control de Perl son orientadas al bloque, similar a las de lenguajes como C. Los diferentes tipos de estructuras de control son [5]:

- *etiqueta* **while** (*condición*) {...}

- *etiqueta* **while** (*condición*) {...} **continue** {...}

- *etiqueta* **for** (*expresión inicial*; *expresión condicional*; *expresión incremental*) {...}

- *etiqueta* **foreach** *variable* (*lista*) {...}

- *etiqueta* **foreach** *variable* (*lista*) {...} **continue** {...}

- **if** (*condición*) {...}

- **if** (*condición*) {...} **else** {...}

- **if** (*condición*) {...} **elsif** (*condición*) {...} **else** {...}

También existe una sintaxis para estructuras que controlan una sola declaración [5]:

- *declaración* **if** *condición*
- *declaración* **unless** *condición*
- *declaración* **while** *condición*
- *declaración* **until** *condición*
- *declaración* **foreach** *condición*

Los operadores lógicos son **and** y **or** y se usan para expresiones. También existen dos construcciones para el manejo de bucles:

grep: devuelve todos los elementos de una lista que cumplen con lo indicado en el bloque.
resultados = **grep** {...} *lista*

map: evalúa el bloque por cada elemento de una lista y devuelve una lista con los valores resultantes.
resultados = **map** {...} *lista*

5.2 Subrutinas

Las subrutina son definidas se definen con el identificador “sub” y se invocan de dos maneras: si no se ha declarado se colocan dos paréntesis redondos después del nombre de la subrutina (ejemplo: subrut();); si ya ha sido declarada simplemente se nombra la subrutina (ejemplo: subrut;) [5].

Ejemplo:

```
subrutina();            # No declarada
sub subrutina {...};    # Declaración
subrutina;              # Declarada
```

En cuanto a los parámetros de las subrutinas, estos no necesitan ser declarados, por lo que pueden variar en tipo y número de un llamado a otro [5]. La forma de indicar los parámetros (escalares, listas o *hashes*) es la siguiente:

```
subrutina $param1, @param2, %param3
```

Los parámetros son almacenados en una lista especial llamada @_, luego esta lista es utilizada en la subrutina para acceder a los argumentos (por medio de un índice). Sin embargo, esto hace muy complicado el código, por lo que se acostumbra guardar los valores de @_ en una lista temporal antes de ser usados [2].

```
# “my” indica que es una variable local
my($val1, $val2, $val3) = @_;
```

Por otra parte, las rutinas pueden devolver uno o más valores, a diferencia de otros lenguajes que solo permiten devolver un valor [5]. La forma de hacerlo es la siguiente:

```
return $val1, @val2, %val3
```

Cabe recalcar que las listas y *hashes* son devueltos en una lista de escalares. Además, si en una subrutina no se especifican valores de retorno, entonces se devolverá la última expresión evaluada en el cuerpo de la subrutina.

A continuación, se describirá como se manejan las expresiones regulares en Perl [5].

6. Expresiones regulares

El lenguaje Perl posee una sintaxis especial para el manejo de las expresiones regulares, además el interprete perl está equipado con un motor para el emparejamiento de hileras y expresiones regulares [5]. Es importante indicar que en los últimos años, muchos lenguajes como PHP, Java y el mismo servidor HTTP Apache han adoptado las expresiones regulares de Perl [1].

Las sintaxis para el manejo expresiones regulares está formada por varios operadores:

Operador m: sirve para comprobar un emparejamiento.

```
# Evaluar si $x empareja con “aabb”
$x =~ m/aabb/;

# Captura cualquier letra entre ‘a’ y ‘c’, y la almacena en la variable interna $1
$x =~ m/a(.)c/;
```

Operador s: sirve para hacer una búsqueda y reemplazo.

```
# Cambia ‘A’ por ‘a’
$x =~ s/Abc/abc;
```

Operador split: sirve para especificar delimitadores de campo, retorna los valores no emparejados.

```
#Divide $hilera en los valores separados por ‘,’
@x = split m/,/, $hilera
```

Además, Perl incorpora varios modificadores que sirven para modificar el significado de una expresión.

Modificador i: sirve para evitar que se haga distinción entre minúsculas y mayúsculas.

```
$x =~ m/aabb/i;
```

Modificador g: sirve para lograr un efecto global de la expresión.

```
# Remplaza todas las ‘A’ por a
$x =~ s /A/a/g;
```

Modificador g: sirve para poder colocar comentarios y espacios dentro de una expresión regular.

```
$x =~ m/a       # empareja ‘a’
      b         # empareja ‘b’
      /x;
```

A continuación se presentará un análisis de la disponibilidad del lenguaje Perl.

7. Disponibilidad a los usuarios

Perl es software libre (bajo licencia GNU y licencia artística), lo que significa que no es necesario pagar para obtenerlo [9].

Además, existen distribuciones y adaptaciones para gran cantidad de sistemas operativos como Linux, Unix, Windows, Mac, etc. De hecho, plataformas como Linux y Unix traen Perl instalado por defecto [1].

Seguidamente se presentará un análisis de las principales ventajas y desventajas de utilizar el lenguaje Perl.

8. Ventajas y desventajas de Perl

Las principales ventajas de utilizar Perl se encuentran justamente en los objetivos que se tenían cuando se creo el lenguaje: Perl es un eficiente, completo y fácil de usar [4].

Otra de las ventajas de Perl está en la cantidad de aplicaciones que se le pueden dar al lenguaje en campos como la administración de sistemas, aplicaciones *web*, entre otros [1].

Además , el hecho de que Perl esté disponible para gran cantidad de sistemas operativos [1], lo hacen un lenguaje accesible a cualquier usuario, sirviendo esto para extender la comunidad que existe acreedor del lenguaje y así convertir a Perl en un lenguaje muy usado y muy confiable.

La principal desventaja de Perl se encuentra en el tiempo de ejecución de un programa, ya que un programa Perl es compilado cada vez que se ejecuta, por lo que puede resultar más lento que un programa similar escrito en otro lenguaje. Sin embargo, se han implementado técnicas para mejorar esta situación como guardar el compilado del programa en memoria y retrasar la compilación hasta que sea necesitada [2].

9. Conclusiones

Gracias a la cantidad de personas que utilizan Perl, el lenguaje ha sido extendido y es usado para aplicaciones como desarrollo de interfaces gráficas de usuario (GUI), desarrollo de páginas *web*, administración de sistema, programación en red, entre otros.

Aunque la sintaxis de Perl puede llegar a desconcentrar y hasta a asustar la primera vez que se le ve, cuando se esta familiarizado resulta muy fácil y cómodas para el desarrollo rápido y eficiente de aplicaciones.

El hecho de que Perl sea portable entra tantos sistemas operativos hace que la comunidad de personas que lo usan se muy extensa, lo que ayuda no solo en la popularidad del lenguaje sino también a extenderlo y mejorar su interprete y módulos (debido a que son código abierto).

Debido a que Perl toma características de otros lenguajes de programación (Lisp, C, AWK, etc.), se puede asegurar que la estructura resultante del lenguaje es lo suficientemente robusta

y confiable, algo que ya se ha demostrado en la cantidad de aplicaciones en las que se utiliza Perl.

7. Referencias

[1] Página en Internet de Larry Wall. URL: http://www.perl.com/pub/au/Wall_Larry. Consultada el 18 de junio del 2007

[2] Programación en Perl. URL:http://es.wikibooks.org/wiki/Programaci%C3%B3n_en_Perl. Consultada el 19 de junio del 2007

[3] Trascripción de una entrevista realizada a Larry Wall. URL: http://www.nntp.perl.org/group/perl.perl6.meta/2000/10/msg424.html. Consultada el 18 de junio del 2007

[4] Manual de UNIX. URL: http://perldoc.perl.org/perlintro.html. Consultada el 17 de junio del 2007

[5] Wall, L. Christiansen, T. Orwant, J. *Programming Perl*, Tercera edición. Julio del 2000.

[6] Common Gateway Interface (CGI). URL: http://es.wikipedia.org/wiki/Common_Gateway_Interface. Consultada el 17 de junio del 2007

[7] Soluciones LAMP. URL: http://es.wikipedia.org/wiki/Solouciones_LAMP. Consultada el 17 de junio del 2007