Características del lenguaje Perl 5.0 y su aplicación como herramienta de desarrollo en la elaboración de un Servidor Web.

Byron I. Barquero Chaves
Universidad de Costa Rica, Escuela de las Ciencias de la Computación e Informática,
San José, Costa Rica
barquechak@gmail.com

y

William Méndez Rodríguez
Universidad de Costa Rica, Escuela de las Ciencias de la Computación e Informática,
San José, Costa Rica
willmwndez@gmail.com

Abstract

This article has the purpose of describing in a brief and clear forms the functionality and flexibility that presenting the interpreter languages, that at the present time have had a great height of popularity, and which they are different from the traditional compilers by his versatility with the applications Web. One of the most used with this intention is Perl 5.0, which offers many possibilities in multiple platforms, between which they are the handling for applications Web, connectivity with Data Bases, communication protocols with Web servers, design of interfaces, and other practical applications. In order to exemplify these characteristics, the implementation of a Web server will appear that uses the functionalities that the language offers.

Keywords: Perl, interpreter, regular expressions, Web server, protocol.

Resumen

Este artículo tiene la finalidad de describir en una forma breve y clara la funcionalidad y flexibilidad que presentan los lenguajes interpretativos, que en la actualidad han tenido un gran auge de popularidad, y que se diferencian de los compiladores tradicionales por su versatilidad con las aplicaciones Web. Uno de los más usados con este propósito es Perl 5.0, el cual ofrece muchas posibilidades en múltiples plataformas, entre las cuales destacan manejo para aplicaciones Web, conectividad con Bases de Datos, protocolos de comunicación con servidores Web, diseño de interfaces, entre otras aplicaciones prácticas. Para ejemplificar estas características, se presentará la implementación de un servidor Web que utiliza las funcionalidades que ofrece el lenguaje.

Palabras clave: Perl, intérprete, expresiones regulares, servidor Web, protocolo.

1 Introducción

Perl (*Practical Extraction y Report Language*) es un lenguaje de propósito general, el cual fue originalmente desarrollado para extraer informes de ficheros de texto y utilizar dicha información para preparar informes, dicho desarrollo motivado principalmente por el hecho de que no existía un lenguaje en ese momento que pudiera satisfacer sus necesidades.

No obstante, actualmente ha evolucionado y se ha diseccionado hacia un enfoque diferente con el que se creó el lenguaje, siendo capaz de realizar labores de administración en cualquier sistema operativo [1], tales como administración de sistemas, desarrollo Web, programación en red, desarrollo de GUI(*Graphical User Interface*), así como otras aplicaciones prácticas.

Este lenguaje debe gran parte de su popularidad a que se trata de un lenguaje pseudo-compilado que se distribuye de forma gratuita; un Script genérico de Perl puede ejecutarse en cualquier plataforma en la que tengamos un intérprete disponible.

Además, con el crecimiento acelerado de sitios Web, se generó la necesidad de realizar programas CGI (*Common Gateway Interface*) el cual define un protocolo de comunicación entre un servidor Web y una aplicación externa para ofrecer contenido dinámico a las páginas Web; con lo que Perl se convirtió en la elección natural para aquellos desarrolladores que se encontraban familiarizados con este lenguaje.

Para demostrar las características y funcionalidades de este lenguaje se va a presentar, como una sección del presente artículo, la implementación de un Servidor Web, desarrollado por los estudiantes Mariano Aguilera Retana y Leonor Anglada Gutiérrez en el curso de Aplicaciones Web de la Universidad de Costa Rica, esto con fines ilustrativos, para mostrar la sintaxis y la utilización del lenguaje Perl en el desarrollo de una aplicación.

En el inicio de este artículo, se realizará una descripción previa de las características y la estructura básica del lenguaje, para comprender posteriormente la implementación del Servidor Web, así como de la historia del mismo, el cual consideramos que es importante resaltar pues muestra la evolución de los lenguajes gratuitos que son desarrollados en conjunto por la comunidad de programadores.

1.1 Historia del lenguaje

Perl fue creado por Larry Wall a mediados de la década de los ochenta, motivado principalmente a que no había un lenguaje en ese momento que pudiera satisfacer sus necesidades relacionadas a la generación de reportes.

Para ello, se trató de llenar el vacío que había entre la programación en bajo nivel, tal como C++ ó Ensamblador; y otros lenguajes de alto nivel que hacían uso del shell del sistema. De esta forma fue como surgió la primera versión de Perl, la cual se puede considerar como una fusión entre ambas técnicas de programación.

Posteriormente, al añadirle varios componentes que fortalecieron el lenguaje, Larry Wall distribuyó el lenguaje a la comunidad de programadores a través de Internet en forma gratuita. Dichos usuarios alrededor del mundo le han dado respaldo, mantenimiento y mejoras, algo que Larry Wall nunca se imaginó al momento de que empezó la creación de las primeras versiones básicas del lenguaje.

Como resultado de este lanzamiento, Perl fue creciendo y fortaleciéndose como lenguaje de programación, tanto en funcionalidades como en portabilidad, haciendo que aquel pequeño lenguaje disponible para un par de sistemas operativos de UNIX creciera hasta poder aportar en la actualidad cientos de páginas de libre documentación, decenas de libros, una importante cantidad de desarrolladores e implementaciones en casi todos los sistemas operativos que se usan hasta el día de hoy.

En la actualidad, Larry Wall aún sigue a cargo de Perl, en donde sigue involucrado con el desarrollo y la toma de importantes decisiones acerca del lenguaje, aunque está apoyado por un equipo de treinta personas, además de los cientos de ayudantes que se han integrado desde diversas partes del mundo.

A continuación se van a describir las principales características que posee este lenguaje en la actualidad, realizando un especial énfasis en las ventajas y desventajas de un intérprete, así como la forma en que el mismo puede ser instalado por cualquier usuario en los dos sistemas operativos más utilizados: Windows y Linux.

2 Descripción del lenguaje Perl 5.0

2.1 Descripción básica del lenguaje Perl

Perl es un lenguaje visualizado para la manipulación de cadenas de caracteres, archivos y procesos. Esta manipulación se ve simplificada por el importante número de operadores a disposición del usuario.

Para trabajar con Perl se requieren dos requerimientos básicos y fáciles de conseguir:

- Un editor de texto para poder escribir los programas Perl.
- El intérprete de Perl, que ejecute los programas realizados con Perl.

Es una combinación de las características de los lenguajes más usados por los programadores de sistemas, como son los *shell* del sistema operativo UNIX, los utilidad (que incluye un lenguaje interpretado propio) awk para formateo y tratamiento de texto e incluso características de Pascal, aunque su potencia se basa en la similitud con las mejores características del lenguaje estructurado C.

Es por esto que el lenguaje Perl se percibe habitualmente como un lenguaje intermedio entre los *shell scripts* y la programación en C. Esto debido a que los programas en Perl son una sucesión de instrucciones y son similares a los *shell scripts* porque no existe un procedimiento principal como la subrutina *main* en C. Sin embargo, se parece al lenguaje C en su sintaxis y en el número importante de funciones que permiten la manipulación de cadenas de caracteres y archivos.

Por todo esto, Perl es un lenguaje muy utilizado en los dos campos siguientes:

1. La administración de sistemas operativos, debido a que, por sus características Perl es muy potente en la creación de pequeños programas que pueden ser usados como filtros para obtener información de ficheros, realizar búsquedas, entre otras cosas. Además, aunque Perl nació en un entorno Unix, hay versiones para casi todas las plataformas existentes.

2. La creación de formularios en la Web. Se ha usado desde los primeros días del Web para escribir *scripts* CGI *(Common Gateway Interface)*, los cuales realizan el intercambio de información entre aplicaciones externas y servicios de información, es decir, se encargan de tratar y hacer llegar la información que el cliente WWW manda al servidor a través de un formulario. Es una de las "tres P's (Perl, Python y PHP), que son los lenguajes más populares para la creación de aplicaciones Web, y es un componente integral de la popular solución LAMP (Linux Apache MySQL Perl) para el desarrollo Web. Grandes proyectos escritos en Perl son Slash, IMDb y UseModWiki, un motor de Wikipedia. En otras palabras, muchos sitios Web con alto tráfico de usuarios.

Actualmente existen dos versiones altamente populares de Perl, la 4.3 y la 5.0, habiendo diferencias importantes entre una versión y otra, esto debido a que no son totalmente compatibles. La versión 5 de Perl es una reescritura en donde se añadieron características para soportar estructuras de datos complejas, y un modelo de programación orientado a objetos. Éstos incluyen referencias, paquetes y una ejecución de métodos basada en clases y la introducción de variables de ámbito léxico, que hizo más fácil escribir código robusto.

Esto rompe en gran medida con la filosofía tradicional de Perl de una programación más parecida a los Shells de Unix que al modular lenguaje C, de modo que las librerías, por ejemplo para creación de CGI´s, no funcionan de un método a otro.

Además, el lenguaje Perl no es precompilado, pero aún así es más rápido que la mayoría de lenguajes interpretados. Esto se debe a que los programas en Perl son analizados, interpretados y compilados por el interprete *perl* antes de su ejecución. [8]

Esto quiere decir que no hace falta un fichero binario para poder ejecutar las instrucciones que hemos codificado usando este lenguaje, es decir, es interpretado, aunque el intérprete de Perl "compila" los programas antes de ejecutarlos [3]. Por esta razón es que se dice que es un lenguaje pseudo compilado, o sea, una fusión entre compiladores e intérpretes.

Es importante en este punto la descripción de lo que es un intérprete y cual es su diferencia con un compilador, así como sus ventajas y desventajas, ya que los lenguajes interpretativos se están convirtiendo en una tendencia con mucha fuerza en la actualidad.

2.2 Descripción de un intérprete y sus diferencias con un compilador

En primera instancia, se va a definir el concepto de un lenguaje compilado y de un lenguaje interpretado.

Un lenguaje compilado es un término un tanto impreciso para referirse a un lenguaje de programación que típicamente se implementa mediante un compilador. Esto implica que una vez escrito el programa, éste se traduce a partir de su código fuente por medio de un compilador en un archivo ejecutable para una determinada plataforma.

Un lenguaje interpretado es aquel en el que las instrucciones se traducen o interpretan una a una en tiempo de ejecución a un lenguaje intermedio o lenguaje máquina o a través de una máquina virtual, siendo típicamente unas diez veces más lentos que los programas compilados, y normalmente no guardan el resultado de dicha traducción [4].

En otras palabras, un lenguaje compilado es aquel que, en teoría, es traducido a código máquina y las instrucciones generadas, son interpretadas directamente por la máquina. Un lenguaje interpretado, es aquel que es traducido a un lenguaje intermedio, o sea, entiéndase no máquina, en la cual cada instrucción es interpretada y traducida a lenguaje máquina en tiempo de ejecución.

Por esta razón a lenguajes como Perl en inglés se les llama *scripting languages* y a los programas escritos en Perl se les llama *scripts* [7].

Este tipo de lenguajes interpretados posee la gran ventaja de la portabilidad porque puede ser compilado en y para cualquier plataforma o sistema operativo, ya que permiten ofrecer al programa interpretado un entorno no dependiente de la máquina donde se ejecuta el intérprete, sino del propio intérprete, lo que se conoce comúnmente como máquina virtual.

Otra ventaja es que son más flexibles en entornos de programación y depuración, lo que se traduce, por ejemplo, en una mayor facilidad para reemplazar partes enteras del programa o añadir módulos completamente nuevos.

Su desventaja primordial es la velocidad, debido a la necesidad de traducir el programa mientras se ejecuta. Este aspecto se debe evaluar a fondo al crear software con este tipo lenguajes, pues se debe equilibrar la portabilidad con la velocidad que se está sacrificando. A menos que las prestaciones de los equipos informáticos sean bastante altas, en el caso cual, se podría despreciar este aspecto.

Otra desventaja importante es que este tipo de lenguajes no poseen verificación de tipos, lo cual quiere decir que no se van a detectar errores propios del programador debido a que Perl y otros lenguajes no hacen reconocimiento de si el dato utilizado es un entero o una cadena de caracteres, y por ende, el mismo intérprete realiza el debido *casting* para trabajar con dicho dato, lo cual puede originar problemas referentes a la veracidad e integridad de los datos, como por ejemplo, al momento del paso de parámetros.

Existen tres situaciones específicas de detallan porque algunos lenguajes no pueden compilarse por completo al lenguaje de la máquina [10], las cuales son:

1. Se han eliminado totalmente la declaración de las variables, de tal modo que una variable tiene siempre el tipo del último valor que se le asignó.
2. Se ha eliminado la gestión dinámica de la memoria, confiándole al intérprete la eliminación automática de la memoria no utilizada.

3. Porque la presencia del intérprete durante la ejecución es necesaria por razones de seguridad o de independencia de la máquina.

Es importante recalcar nuevamente que Perl es un lenguaje interpretado por excelencia, aunque se compilan a código intermedio en tiempo de ejecución, cosa que acelera su ejecución. Existen también herramientas que generan un código cercano a la máquina para estos dos lenguajes y cachean contenido, pero a la larga, siguen siendo interpretados. [9]

Habiendo especificado de mejor forma las características de un lenguaje interpretado, vamos a seguir describiendo las características propias del lenguaje de Perl.

2.3 Características del lenguaje Perl

Existen varias características acerca de Perl que son importantes de rescatar, las cuales se mencionan a continuación:

1. Es fácil de usar, aunque es difícil de aprender. Cuando se ha programado en Perl por varias horas, Perl se va a ir haciendo cada vez más fácil de implementar. Este lenguaje se desarrolló pensando en que el lenguaje fuera práctico (fácil de usar, eficiente, y completo) en lugar de pequeño, elegante y mínimo, además de que esta enfocado hacia un desarrollador que posee cierta cantidad de conocimientos sobre el lenguaje y no así hacia un estudiante que esta aprendiendo.

2. Existen muy pocas cosas que no se pueden hacer con este lenguaje, por lo que se le puede considerar como un lenguaje que no tiene fronteras. Con Perl se puede programar cualquier necesidad que se tenga, ya que existen librerías y módulos para casi cualquier cosa que se requiera, pero hay aplicaciones que requieren mucha rapidez en las cuales es mejor utilizar otros lenguajes que no sean interpretados. [3].

3. Es rápido de crear, ya que no posee funciones que, aunque sean bastante interesantes, hagan disminuir la velocidad de desarrollo de una aplicación del lenguaje.

4. El lenguaje es feo, siendo esta una de las principales razones para su difícil aprendizaje, aunque esto se compensa con el poder de alcance del lenguaje. Es por esto que Larry Wall escogió el camello como el logotipo del lenguaje, ya que aunque es feo, siempre trabaja fuerte aún en condiciones complicadas.

5. Se puede utilizar en varios entornos, como puede ser Windows 95, OS/2, Linux, entre muchos otros, sin realizar cambios de código, siendo únicamente necesario la introducción del interprete Perl correspondiente a cada sistema operativo [2].

6. Tiene características que soportan una variedad de paradigmas de programación, como la estructural, funcional y la orientada a objetos. Al mismo tiempo, Perl no obliga a seguir ningún paradigma en particular, ni obliga al programador a elegir alguna de ellas. No obstante, esta característica es solo accesible en la versión 5.0.

7. Tiene incorporado un poderoso sistema de procesamiento de texto y una enorme colección de módulos disponibles.

8. Ofrece una ayuda en línea desde la consola de comandos. Por ejemplo, para obtener ayuda sobre la función print, se debe de escribir en una ventana MSDOS: perldoc –f print

9. Se ejecuta desde la línea de comandos de una ventana del sistema operativo.

10. Es un lenguaje *case-sensitive*, lo cual hace referencia a la propiedad de un texto para presentarse en mayúsculas o en minúsculas [5].

Antes de continuar con la estructura básica del lenguaje de Perl, se consideró importante mencionar una descripción de como es que se puede conseguir el intérprete de Perl, así como el procedimiento para ponerlo a funcionar correctamente en la plataforma en la cual se está desarrollando.

2.4 Obtención e instalación del intérprete de Perl

Se debe de recalcar que para el lenguaje Perl 5.0 no existe lo que se denomina como compilador, como se explico anteriormente en la sección 2.2, de nombre “Descripción de un intérprete y sus diferencias con un compilador”. Por tal razón se va a citar el procedimiento a realizar para poder desarrollar aplicaciones mediante el lenguaje Perl en una computadora personal cualquiera, que posee ya sea Windows o Linux.

2.4.1. Instalación en Plataforma Windows.

Se puede utilizar el intérprete ActivePerl el cual se puede descargar de la siguiente dirección electrónica http://www.activestate.com/. Luego es conveniente utilizar como directorio base de la instalación C:\Perl y añadir al PATH de las variables de entorno del sistema la ruta C:\Perl\bin [1].

2.4.2. Instalación en Plataforma Linux.

Hoy en día todas las distribuciones incluyen casi de serie un intérprete de Perl, es decir, el intérprete de Perl se instala casi por defecto (muy mínima debería ser las instalación para que no se incluyera o quitarlo a propósito). Debido a la característica que antes apuntábamos de su utilidad para realizar tareas de administración del sistema.

Para instalar Perl (o instalar una versión actualizada), lo primero que hay que decidir es si instalar una versión compilada (binaria) o compilar nosotros mismos. Ambas cosas (fuentes o binarios) se pueden encontrar en www.cpan.org para multitud de Sistemas Unix (y otros no Unix).

Si te bajas una versión compilada y específica para tu sistema, simplemente ejecutas el binario que te bajes (descomprimiéndolo si estuviera comprimido) o instalándolo con rpm -uhv perl_compilado.rpm si es un fichero rpm. El único problema que se puede encontrar es que haga falta alguna librería, y ahí es donde se complican las cosas, ya que también habrá que instalarla [3].

Habiendo descrito esto, se va a continuar con la sección referente a la estructura del lenguaje Perl.

3 Estructura del lenguaje Perl 5.0

En esta sección se va a describir de una forma bastante breve los tipos de datos que se pueden representar en este lenguaje. Vale recalcar que existen muchas más estructuras y tipos de datos a describir, tales como las expresiones regulares, las cuales no van a ser especificadas para no desviar la idea y propósito de este artículo.

3.1 Tipos de datos.

Un aspecto importante de este lenguaje es que, por defecto, no es necesario declarar las variables previamente a su uso. Las variables se pueden empezar a usar directamente en las expresiones, lo cual ofrece mucha flexibilidad al momento de estar desarrollando el código.

Existen tres tipos básicos de variables, los cuales se citan a continuación.

1. Escalar: empiezan por el carácter $. Un escalar puede almacenar números, strings, referencias a otras variables y descriptores de ficheros. Algunos ejemplos de la declaración de este tipo de datos son los siguientes: $a = 5; $b = "xxx"; $c = $a++;

2. Arreglos: las variables array empiezan por el carácter @, y sirven para agrupar un conjunto de variables de tipo escalar. En este aspecto también se debe de nombrar la existencia de matrices. Algunos ejemplos son los siguientes: @a = (95, 7, 'fff'); print @a;

3. Hash: las variables tipo *hash* o array asociativo empiezan por el carácter %. Se trata de un tipo característico de Perl, y consiste básicamente en un array en el cual se accede a sus distintos elementos a través de una clave en lugar de por un índice. Los elementos se accesan por claves y no se permiten claves duplicadas. Para crear un elemento de un hash se requiere una lista de dos

valores, siendo el primer elemento la clave y el segundo es el valor asociado a dicha clave, como se muestra en el siguiente ejemplo: %almacen = ('Peras', 5, 'Manzanas', 3); print $almacen{'Peras'};

4 Descripción de un Servidor Web e implementación en Perl.

Un servidor es un tipo de software que suministra servicios a los usuarios o terminales que lo solicitan. Por ejemplo, en una típica arquitectura cliente-servidor, el cliente podría ser un ordenador que realiza peticiones de información a través de un programa de correo, y el servidor le entrega los datos en forma de correos electrónicos en respuesta a su solicitud [7].

4.1 Descripción de un Servidor Web

Un servidor Web es un programa que sirve datos en forma de páginas Web, hipertextos o páginas HTML (*HyperText Markup Language*), textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonidos.

La comunicación de estos datos entre cliente y servidor se hace por medio un protocolo el cual define el conjunto de reglas que gobiernan el intercambio entre entidades dentro de una red; concretamente del protocolo HTTP (*Hyper Text Transfer Protocol*) el cual carece de estado; es decir, cada petición de un cliente a un servidor no es influida por las transacciones anteriores, por lo que el servidor trata cada petición como una operación totalmente independiente del resto. Además una de las características del protocolo HTTP es que no es permanente, es decir, cada operación HTTP implica una conexión con el servidor, que es liberada al término de la misma.

4.2 Arquitectura y tipos servidores Web

La arquitectura de un servidor Web, generalmente está divida en la capa de servidor y en la capa soporte, las cuales contienen los siguientes subsistemas:

- Capa servidor. Esta capa contiene cinco subsistemas, que son los responsables de implementar la funcionalidad de un servidor Web.
    - Subsistema de recepción: representa la primera “línea de ataque” y su labor consiste en esperar las peticiones HTTP de los clientes que llegan por la red. También, analiza las peticiones y determina las capacidades de los navegadores (tipo de navegador, compatibilidad, etc.). Este subsistema contiene la lógica necesaria para manejar múltiples peticiones.
    - Analizador de peticiones: encargado de traducir la localización del recurso de la red al nombre del archivo local. Por ejemplo, la solicitud del recurso http://www.mec.es se traduce al fichero local /var/www/webfiles/index.html.
    - Control de acceso: sirve para autentificar y permitir el acceso.
    - Manejador de recursos: este subsistema es el responsable de determinar el tipo de recurso solicitado; lo ejecuta y genera la respuesta.
    - Registro de transacción: se encarga de registrar todas las peticiones y su resultado.
- Capa soporte. Esta capa actúa como una interfase entre el sistema operativo y el servidor Web y, entre los propios subsistemas de la capa superior.
    - Útil: contiene funciones que son utilizadas por el resto de subsistemas.
    - Capa abstracta del Sistema Operativo (OSAL): este subsistema encapsula el funcionamiento específico del sistema operativo para facilitar la portabilidad del servidor Web a diferentes plataformas.

Entre los tipos de servidores Web se encuentran los basados en procesos los cuales son el predecesor de todos los demás y que se basan en la obtención de paralelismo mediante la duplicación del proceso de ejecución, de donde surgieron los basados en hilos en los cuales se tiene la ventaja que la creación de un hilo no es tan costosa como la de un proceso. Además se encuentran los servidores basados en sockets no bloqueantes o dirigidos por eventos los cuales utilizan una llamada al sistema para examine el estado de

los sockets con los que trabaja. Por último se tienen los servidores implementados en el núcleo con lo que se busca acelerar la velocidad de un servidor Web mediante el movimiento de su código de espacio de usuario a espacio de núcleo o *kernel*.

4.3 Implementación de un servidor Web en Perl 5.0

Para realizar la implementación de un servidor Web, inicialmente se deben definir los parámetros bajo los cuales será construido el servidor, como el tipo de servidor, protocolo, el tipo de acción HTTP que se utilizará, los tipos de archivos que el servidor podrá servir, entre otros aspectos para la construcción del servidor. El servidor que se muestra posee las siguientes características:

- Tipo de servidor: servidor basado en sockets dirigidos por eventos.
- Protocolo: mecanismo HTTP el cual es el más utilizado en la Web. Es un protocolo cliente/servidor simple que funciona con sockets TCP/IP. En donde se tiene a grandes rasgos lo siguiente:
  - El cliente manda una petición al servidor: "Envíeme el documento con el nombre X"
  - El servidor responde al cliente: "Acá esta el dato que me solicitó " (o "Lo siento. No entiendo lo que me quiere solicitar").
- Tipo de acción HTTP GET para enviar los datos, en donde los datos se envían como parte del URL. Cada valor de campo aparece en el formato *nombre=valor*, además para asegurarse de que el servidor Web no confunda los caracteres especiales que podrían aparecer en los datos del formulario, cualquier carácter con un significado especial se codifica usando un encriptado especial [1].
- Tipos de archivos: archivos con extensiones .jpg, .gif, .html, .png, css, .zip, .rar, .pdf.
- Inclusión de una bitácora en la cual se registran las peticiones del cliente y las respuestas brindadas por el servidor.

Con estos parámetros definidos, lo primero que se utiliza dado que el servidor es del tipo basado en sockets es el paquete IO::Socket de Perl, el cual proporciona una interfase muy sencilla, y para crear un socket basta con llamar al constructor de la clase IO::Socket::INET, además se define el puerto de escucha del servidor:

```
#!c:/program files/perl/bin/wperl.exe
 use IO::Socket;
 use Net::hostent;              # for OO version of gethostbyaddr
 use File::stat;

 $PORT = 3500;                  # pick something not in use

 $server = IO::Socket::INET->new( Proto => 'tcp', LocalPort => $PORT, Listen => SOMAXCONN, Reuse
=> 1);
```

Luego el servidor espera la petición del cliente; de esta forma una vez que el cliente realiza la petición, el servidor procede a capturar los datos del cliente, donde obtiene de la petición HTTP encabezados como Host (Servidor al que se realizó la petición original), Accept (Tipos MIME (*Multipurpose Internet Mail Extensions*) aceptados por el cliente), Referer (URL del último documento visualizado por el cliente) y Content-Length (Longitud en bytes de los datos de respuesta), para lo cual se utilizan expresiones regulares de Perl.

```
while ($client = $server->accept()) {

   $client->autoflush(1);
   …

#   Captura los datos del cliente
   while($linea =~ /(\w)|(\d)/){
            $linea = <$client>;
```

```
        if($linea =~ /^Host: (.+)/){
                $host = $1;
                …
        }
        …
  }
```

Luego se verifica que la petición es HTTP GET.

```
if ($request =~ m|^GET /(.+) HTTP/1.[01]|){ … }
```

Posteriormente se verifica si se esta solicitando un script Perl o es una petición de un documento, de ser así se tiene que analizar el tipo de MIME TYPE (donde se utilizan expresiones regulares para identificarlo) que esta solicitando el cliente.

```
sub asignarContenido{
        #Cada uno de lo if pregunta por la extensión y lo asigna a la variable $content
          if ($_[0] =~ /(.+\.jpg)/gi){
                $content = "image/jpg";
          }
          #.+(\.s?html|\.htm)
          if ($_[0] =~ /(.+\.(s?html|htm|x(ht)?ml))/){
                $content = "text/html";
          }
          …
 }
```

Luego se debe analizar el encabezado Accept, de manera que si el cliente acepta el tipo de archivos que esta pidiendo entonces se retorne verdadero.

```
#Analiza el encabezado Accept, de manera que si el cliente acepta el tipo de archivos que esta
 #pidiendo entonces retorna verdadero.
 sub hacerAccept{
        if(($accept =~ /$content/)||($accept =~ /\*\/\*/)){
            return 1;
        }else{
           return 0;
        }
 }
```

Si la respuesta es verdadera, entonces se crea un conjunto de encabezados que se enviarán al cliente compuesto por las etiquetas: HTTP/1.0 200 OK, Content-Type, Content-length, Date y Server, de caso contrario enviar un error al cliente notificando que no se pudo procesar correctamente la petición.

```
#En caso de que ocurra un error en la petición, se envía un código de error al cliente.
#400: Bad Request.
#404: El archivo no existe.
#406: El tipo de archivo no es aceptable

 sub error{
        if($_[0] == 400){…}
        if($_[0] == 404){…}
        if($_[0] == 406){…}
 }

 #Envía un conjunto de encabezados al cliente compuesto por las etiquetas:
 #HTTP/1.0 200 OK, Content-Type, Content-length, Date y Server.
 sub armarHeader{
        local $date = scalar localtime;         #fecha y hora local
```

```
        print $client "HTTP/1.0 200 OK\nContent-Type: $content\n";
        $tamano = stat($_[0])->size;
        print $client "Content-length: $tamano\n";
        print $client "Date: $date\n";
        print $client "Server: perServer 0"."\n\n";
        #imprime en la bitacora
        printf $file "HTTP/1.0 200 OK\n";
}
```

Luego de enviar estos encabezados al cliente, se procede a enviar la página o los datos al cliente.

```
#Envía la página al cliente o recibe los datos de este y los guarda en la bitácora
sub get{
        #si es un form haciendo GET
        if($_[1]){
            $datos = $_[1];
            …
        }
        #si es un GET normal
        else{
            open(my $f,"<$_[0]");
            while(<$f>) {
             print $client $_;
        }
 }
```

Con lo que se tiene finalmente el siguiente flujo de ejecución, que realiza el control desde la asignación de del contenido hasta el envió final del documento o un mensaje error al cliente.

```
        #si esta haciendo una solicitud a un script perl
                if($1 =~ /(.+)\.pl/){
                    getScript();
                }
                elsif($1 =~ /(.+)\?(.+)/){
                    print "prueba: $1  ";
                    print "$2";
                    asignarContenido($1);
                    if(hacerAccept($1)){
                      if (-e $1) {
                          armarHeader($1);
                          get($1, $2);
                      }else {
                          error(404, $1);
                      }
                }else{
                     error(406, $1);
                }
```

Por último se cierra el archivo así como el cliente, y de esta manera el servidor quedará esperando la siguiente petición de otro cliente.

```
close $file;
close $client;
```

5 Conclusiones

El primer punto que podemos nombrar acerca del lenguaje Perl 5.0 son las diversas ventajas de utilizar lenguaje interpretativo como herramienta de desarrollo, tanto para aplicaciones web como para escritorio, por el hecho de ser un lenguaje script, lo cual implica que no requiere de un compilador y no se genera un

código objeto, sino que se va interpretando línea por línea. Esto ofrece la funcionalidad de portabilidad de la aplicación debido a que no requiere una plataforma específica para implementar la aplicación, lo cual es una característica muy importante y deseada en toda aplicación web. No obstante, también tiene desventajas importantes, tales como la verificación de tipos como su velocidad de ejecución.

Un segundo importante a recalcar es su facilidad de uso, tanto en lo referente a la obtención e instalación del mismo, como por las ventajas a nivel de programación que ofrece. Perl 5.0 es un lenguaje libre que solo requiere de un editor de texto y un intérprete fácil, el cual es fácil de instalar y de obtener, debido a que es un lenguaje no privativo. Además, ofrece la posibilidad de trabajar con expresiones regulares y otros tipos de objetos que no son comunes en lenguajes diferentes a los scripts.

El último punto a nombrar son las facilidades que presenta el lenguaje Perl 5.0 para la implementación de aplicaciones Web. Existe una gran diversidad de funcionalidades añadidas a la versión 5.0 para el manejo de programación en red, así como el fácil uso de las expresiones regulares, las cuales son muy utilizadas por los desarrolladores para validar páginas web a nivel de formularios y envío de peticiones HTTP GET por parte de los clientes en los navegadores Web, entre otras aplicaciones, que ofrece una mayor velocidad de respuesta y seguridad al momento del descifrado.

Por tanto, se puede decir que el lenguaje de Perl es útil cuando se requiera hacer un programa pequeño en donde se debe de dar mucho énfasis al manejo de cadenas de caracteres, pero no es bueno para proyectos muy grandes, sin importar las funcionalidades que el proyecto requiera.

6 Referencias

[1] Prellezo, J. Perl 5.0: Un Lenguaje Multiuso, URL:http://www.open-eslack.org/mans/programm/perl/perl.pdf Versión 1.6, Octubre, 2002
[2] Tutorial de Perl, Flanagan, URL: http://flanagan.ugr.es/perl/
[3] Intérprete Informático, Wikipedia: La enciclopedia libre, URL: http://es.wikipedia.org/wiki/Int%C3%A9rprete_inform%C3%A1tico
[4] Case sensitive, Wikipedia: La enciclopedia libre, URL: http://es.wikipedia.org/wiki/Case_sensitive
[5] Perl, Wikipedia: la enciclopedia libre, URL:http://es.wikipedia.org/wiki/Perl, Enero, 2006
[6] Perl en bioinformática, URL: http://www.ccg.unam.mx/~contrera/bioinfoPerl/
[7] El servidor Web: Arquitectura y funcionamiento, Ministerio de Educación y Ciencia, URL:http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=366, España, Julio, 2006
[8] Perl : Introducción, Universidad de Las Palmas de Gran Canaria, URL:http://www.ulpgc.es/otros/tutoriales/perl/cap1.htm#Introduccion, España, Enero, 2003
[9] Apuntes Web y Notas de Programación – Dev Blog, Ricardo Obregón, URL: http://robregonm.blogspot.com/2006_02_26_archive.html, Bogotá, Colombia.
[10] Capítulo 8. Intérpretes, Manuel Fonseca, Universidad Autónoma de Madrid, URL: http://arantxa.ii.uam.es/~alfonsec/docs/compila8.htm, Madrid, España.