Periódico de la Comunidad Perl de Capital Federal

http://cafe.pm.org/boletines/

Cafe Perl v0.a

CaFe Perl v0.a - Periódico de la Comunidad Perl de Capital Federal

Editorial

Hola Perl Mongers, bienvenidos a un nuevo número de CaFe Perl !!

Esta vez tenemos grandes novedades. Para empezar a partir de este número hay un nuevo colaborador de CaFe Perl, alguien que ven a menudo en la lista, y ese alguien es Martín Ferrari que a partir de ahora va a llevar la sección Peace of Code, pero ya no como un ejercicio para hacer durante el mes sino como una serie de one-liners muy interesantes. Martín : bienvenido y que lo disfrutes.

Por otra parte nuestra sección PERLitas nos trae una entrevista con Patrick Rael, quién está en proceso de creación de un robot. También la sección “Mordiditas...” nos trae una inmersión en el mundo de la precedencia de operadores.

Espero que lo disfruten.
Hasta la próxima taza de CaFe Perl !!! ... eso sí, café del bueno ;-).

Víctor A. Rodríguez (Bit-Man)

POC (peace of code)

Antes que nada las respuestas a las preguntas del número anterior :

```
my $home = $ENV{HOME}
            || $ENV{LOGDIR}
            || (getpwuid($<))[7]
            || die "Homeless boy\n"
```

- **qué significado tiene el hash %ENV ??** : permite el acceso de la variables de ambiente del sistema operativo (como PATH, HOME, etc.)
- **qué son $ENV{HOME} y $ENV{LOGDIR} ??** : contienen los valores de los paths correspondientes al directorio home y al de logs
- **que devuelve la función getpwuid() ??** : permite obtener una serie de datos del usuario cuyo UID es pasado a la funciín getpwuid() (en particular el octavo es el home dir0
- **qué contiene la variable $< ??** : el user id (UID) real del proceso actual
- **qué significado tiene el valor alojado en $home ??** : el path del home directory del usuario actual

Ahora el peace of code de esta semana : *inspeccionar la estructura de un HTML*.

Útil para cuando estamos haciendo alguna mecanización de una página, y necesitamos ver sintéticamente el árbol de elementos. Luego solemos usar el mismo TreeBuilder para extraer datos y eso.

Toma por entrada estándar el html y escupe un árbol de elementos.

```
perl -MHTML::TreeBuilder -e '$t = HTML::TreeBuilder->new; while(<>)
{$t->parse($_) }; $t->eof; print $t->dump'
```

Ejemplo, leer la página de google no es apto para cardíacos, pero este script lo hace más fácil. Aquí uso algunas cosas que bien podría poner dentro del one-liner, pero no tiene sentido, es preferible esperar un segundo más y escribir menos!

Explicación: obtengo el html de google.com, lo convierto de iso-8859-1 (el encoding por default para argentina devuelto por google) a nuestro locale en uso, y luego se lo paso a perl.

```
$ GET http://google.com/ | iconv -f iso-8859-1 | perl
-MHTML::TreeBuilder -e '$t = HTML::TreeBuilder->new; while(<>) {$t-
>parse($_) }; $t->eof; print $t->dump'
<html> @0
  <head> @0.0
    <meta content="text/html; charset=ISO-8859-1"
http-equiv="content-type"> @0.0.0
    <title> @0.0.1
      "Google"
    <style> @0.0.2
         "<!--\x0abody,td,a,p,.h{font-family:arial,sans-serif;}\x0a.h
{font-size:
..."
    <script> @0.0.3
          "\x0a<!--\x0afunction sf(){document.f.q.focus();}\x0a// --
>\x0a"
  <body alink="#ff0000" bgcolor="#ffffff" link="#0000cc"
marginheight=3 onload="sf()" text="#000000" topmargin=3
vlink="#551a8b"> @0.1
```

PERLitas

Entrevista y traducción realizada por Víctor A. Rodríguez

Robot Maximilian (http://howtoandroid.com/) es una cabeza de robot computerizada, que fue hecha por Patrick Rael para ... mejor vemos lo que tiene que decir y qué nos puede enseñar.

Por favor Patrick, una introducción para el grupo CaFe.pm

Hola a los miembros de CaFe.pm. Mi nombre es Patrick Rael. Me gusta experimentar con androides. Cualquiera que me conoce sabe que me gusta desdibujar la línea entre las computadoras y el mundo real. El mundo es un kernel, y nosotros somos procesos.

Cuál fue su motivación para construir a "Robot Maximilian" ??

Tuve dos motivaciones. (1) Tengo el sueño de que un algún día las computadoras serán súper fáciles de usar. “Súper fácil” entendido como trivial. Robot Max es un intento parcial de crear una computadora que es trivial de usar. En ese sentido, R.Max es un trabajo en progreso. Todavía no hay programación alguna para su mente. (2) Otra motivación es resolver el reto de crear una mente robótica que podría demostrar ser inteligente y consciente. Esto también es un trabajo en progreso.

Hace siete años hice algún diseño de alto nivel para un modelo de mente robótica auto-consciente. Lo llamo M1[1]. Pero hasta recientemente no la llevé hasta el siguiente nivel. Hace muy poco, tuve una idea de cómo implementar M1 #9 que es “el foco”. La idea es curiosa porque es una idea visual. O sea, pueda verla en mi mente. Ahora estoy analizando esa imagen y descomponiéndola en algo que pueda ser puesto en palabras. A veces pienso visualmente. La idea es cobre una forma de mantener el foco en una idea sobre un período de tiempo T. La idea se enlaza con la figura en ese documento M1 que muestra un lazo de realimentación en la mente de los robots. Creo que ahora veo cómo diseñar “el foco” y la conciencia, todo basado en esa figura en mi mente.

Y hablando sobre la mente de M1 ... la implementaría en Perl ?? Qué lenguaje cree que es el más calificado para esta tarea ??

Normalmente hago prototipo en Perl de la mayoría de las cosas que hago porque es un buen lenguaje para hacer algo rápido. Perl será mi primer elección para M1 a esta altura. Sin embargo, M1 es deliberadamente vago ya que sólo dice que se necesitan diez piezas. Cualquiera podría modelar M1 en una tarde pero eso no garantiza que el robot alcance la auto-conciencia y sentiencia. Todo es acerca de cómo sea modelado M1 para llegar a ser una mente viviente. Ese es el reto.

Encontré algunos módulos de perl para redes Neuronales en CPAN. Son muy interesante. Estoy comenzando a jugar con ellos. También encontré el programa C llamado “motion” para Linux. Es n lindo programa que puede ser usada para detectar movimiento en las imágenes de las webcams y colocar un cuadrado alrededor del area que va cambiando. Creo que lo voy a adoptar para que R.Max mueva sus ojos cámara centrados en el movimiento. A este punto temprano en la exploración de M1, usaré cualquier lenguaje que sea conveniente hasta que el diseño se solidifique.

Tiene alguna ayuda de otros proyectos Open Source ?? (código, consejos, etc.)

Me gusta usar Open Source y software libre en todo lo que hago y en la medida de lo posible. Perl, Linux, herramientas GNU, Java y miles de otros freeware. La herramientas Open Source son indispensable para el trabajo que hago.

---

1 N. del T. el resto de la pregunta se basa en la arquitectura de M1 (http://howtoandroid.com/Architecture.html)

Qué habilidades, relacionadas con Perl o no, obtuvo mientras hacía este proyecto ??

Aprendí Perl/Tk, javax.comm y algunos trucos de stty. Creo que el truco de stty es uno de los más contra-intuitivos. Lo que quiero decir es que, seguramente cualquiera que usa Unix puede ejecutar un programa y redireccionar la entrada desde un archivo. El resultado es que el contenido del archivo es procesado por el programa. Pero en cambio se puede pensar en ejecutar un programa y redireccionar la entrada desde un archivo, PERO NO SE INGRESA NI UN SOLO CARACTER DE ESE ARCHIVO NI SE PIENSA HACERLO ? Eso es extraño.

El comando stty que está abajo es sólo un ejemplo :

```
% stty 9600 cs8 -parenb -cstopb < /dev/ttyS0
```

Encontré este truco en el libro "Tricks of the Unix Masters"[1], sobre el final donde hay varias páginas de poderosos “one-liners”[2] listados. Este programa manipula “datos”, pero no los datos en el archivo.

Qué consejo le daría a los futuros diseñadores y hobistas que enfrentan un nuevo proyecto ??

1. Ve por él!
2. Nunca dejes que la intratabilidad te desacelere
3. Nunca pospongas hasta mañana lo que tu androide puede hacer durante la noche
4. Los robots hacen el trabajo duro (la llamo la ley de Pat de la Robótica)
5. Una costura[3] a tiempo salva nueve dimensiones temporales

En qué partes del código le aconsejaría mirar al codificador novato de Perl, para tener una experiencia placentera de aprendizaje ??

Para RobotControl, que es perl, diría que lo más divertido no está siquiera en el código perl. De otro ejemplo en mi site web http://HowToAndroid.com/RobotControl-PerlTk.html, las cuatro líneas de Unix que están sobre el ejemplo son interesantes. Por ejemplo, si ejecutas RobotControl y no enviás la salida a través del port serial, o dejás que la salida vaya a tu terminal, podés arruinar la terminal. Si lo hacés todo lo que escribas se van a ver como garabatos, entonces intentá el truco de escribir ^Jreset^j. Estos son los 7 caracteres a escribir, y no buscar el reset de tu computadora.

De otra forma, dentro de RobotControl, diría que en el manejo de eventos, la callback modelo de las aplicaciones Perl/Tk es interesante y también los es la sub scale_cb(), la subrutina que es llamada cuando el usuario arrastra uno de los controles en la gui para ajustar el servo a una nueva posición. RobotControl tiene tan pocas líneas porque el circuito Mini-SSC-II establece una conexión serial y un protocolo de paquetes de 3 bytes para mover los servos. Dado este protocolo de 3 bytes, incluso podría controlarse los servos desde el shell. El primer byte identifica el inicio del paquete, el siguiente el ID del servo, y el último es la posición de 0..255 para mover el servo.

Cómo se puede colaborar con "Robot Maxamilian" ??

1 N. del T. : Trucos de los amos de Unix
2 N. del T. : programas o comandos que resuelven un problema en una sola línea
3 N. del T. : stitch

Actualmente, mi propio Robot Max es mi trabajo en curso. Sin embargo, sugiero que si otros están interesados, pueden construir si propio R.Max. Tengo muchos e-mails de gente diciendo que van a hacer exactamente eso: hacer su propio R.Max. Les digo que lo hagan, que tengan mucha suerte y diversión.

Que funcionalidades no están implementadas, y cuáles cree que agregará pronto ??

La programación de la mente de R.Max aún no está hecha. RobotControl es simplemente un juguete para demostrar que los servos pueden ser controlados por un usuario externo. Cuando la mente de M1 esté implementada, entonces R.Max controlará sus propios servos.

Algún módulo favorito de CPAN ??

Creo que el módulo que más uso es Getopt::Long. Pero como favorito, no se si tengo uno. El módulo Tk es uno, y también DBI::DBD. Hay tanto módulos que usé que no puedo decir que alguno en particular es mi favorito.

Algún programador favorito, miembro de la comunidad o persona abocada a Perl

Creo que Larry Wall y sus ayudantes se merecen una gran ronda de aplausos. Y después una de cerveza también.

Hay algún grupo Perl en su localidad ??

Desconozco.

Algo más que nos quiera decir y que no le preguntamos ??

Si tuviera alguna queja con Perl, y no estoy diciendo que la tenga. Pero si hay algo que cambiaría, sería la palabra “Bless” para crear objetos de una determinada clase. s/Bless/Boing/g.

Si alguna vez voy a Argentina, voy a buscar a CaFe.pm.

SudorNews

Un comienzo atinado

Todos nos ponemos contentos cuando Perl gana una batalla ... pero que tal cuando pierde una ?? Ocurre que a alguien le pasó e investigando llegó a una buena cantidad de conclusiones. (http://www.perl.com/pub/a/2005/12/21/a_timely_start.html)

Parser vs. Lexer

Normalmente solemos usar las expresiones regulares en todo lugar que quepan, y muchas veces se pueden tornar un tanto complicadas, sobre todo cuando tenemos que hacer el parsing de HTML o estructuras por el estilo : el lexing viene a nuestra ayuda (http://www.perl.com/pub/a/2006/01/05/parsing.html)

Entrevistas a Chris Nandor y brian d foy

De este personaje, famoso por sus intervenciones en Slashdot y use perl, pueden escuchar un podcast en http://perlcast.com/2006/01/05/interview-with-chris-nandor/ y brian d foy en http://www.perlcast.com/audio/Perlcast_Interview_018_foy.mp3

Parrot 0.4.1 "Foghorn Leghorn" Released!

La máquina virtual que soportará a Perl 6 y otros lenguajes dinámicos ha liberado su versión 0.4.1 (http://www.cpan.org/authors/id/L/LT/LTOETSCH/parrot-0.4.1.tar.gz). Parrot aún se encuentra en desarrollo, así que si quieren colaborar pueden empezar por http://www.parrotcode.org/source.html

Sin bichos ni bugs

Hay situaciones en las que un bug no sólo es molesto (siempre los son) sino que además puede llegar a ser inadmisible. Hay gente que hace que esto no suceda, y con herramientas que (a veces) todos tenemos a mano (http://www.spectrum.ieee.org/sep05/1454)

pmtools de nuevo en CPAN

Hace tiempo Tom Christiansen escribió una serie de herramientas para manejo de módulos perl, y estaban desaparecidas hasta hace poco que Mark Fisher las rescató y las uso en CPAN (http://search.cpan.org/~mlfisher/). Si bien no es una obra de arte si se puede decir que “le falta redondear los bordes”, pero no es ni mas ni menos que un buen trabajo y una muy buena prueba conceptual (http://use.perl.org/~Mark%20Leighton%20Fisher/journal/28339).

Mapa conceptual de software libre

Algo bonito para ver, conceptual o no es este mapa conceptual del software libre (http://bulma.net/body.phtml?nIdNoticia=2260), y también en el mismo tenor tenemos al típico poster con la historia de la evolución de los lenguajes de programación más conocidos (http://oreilly.com/news/languageposter_0504.html) y el de la historia de Unix (http://www.levenez.com/unix/history.html)

Existe la comunidad open source ?

Extraído de barrapunto.com : La comunidad open source no existe". Este es el provocador título de un artículo recién aparecido en O'Reilly OnLamp.com (http://www.onlamp.com/pub/a/onlamp/2006/01/12/no_oss_community.html) que dice: "Que pasaría si descubres que todo lo que aprendiste sobre open source está equivocado? ¿Que la narrativa que enfrenta a open source con "gigantes malos" del software no es en realidad correcta? Y si te enteras de que los líderes reconocidos del movimiento open source no son más que figuras publicas de un proceso que ya existía? ¿Y si te enteras que el open source no es ni bueno ni malo, sino la simple manifestación de tendencias económicas de toda la vida? ¿De que las compañías que utilizan el open source no están tomando una postura moral superior, sino aplicando de forma despiadada una ventaja competitiva?" El artículo es largo, pero muy interesante

Perl Mongers ahora en Wikipedia

bran d foy ha comenzado un artículo sobre este tema en la Wikipedia (http://en.wikipedia.org/wiki/Perl_mongers). Colaboradores bienvenidos sean !!!

Salió la nueva ley ...

Se largó a rodar el primer draft de GPLv3 (http://gplv3.fsf.org) y, como de costumbre, levantó mucha polémica y polvareda :

- Eben Moglen, abogado principal de FSF y co-autor de GPLv3 explicó en http://news.zdnet.co.uk/business/legal/0,39020651,39247976,00.htm que DRM es fundamentalmente incompatible con GPLv3
- Linus Torvalds comentó en la lista del kernel de Linux (http://lkml.org/lkml/2006/1/25/273) que el kernel permanecería con GPLv2
- Sun considera agregar a Solaris, su sistema operativo, la licencia GPLv3 (http://blogs.sun.com/roller/page/jonathan?entry=hp_and_sun_partnering_around)

Comentarios y opiniones en http://gplv3.fsf.org/comments/

Programación Perl con DB2 Universal Database

Un artículo que explica cómo escribir programas Perl sencillos que extraen o manipulan datos (http://www.codejava.org/?idxpagina=9&destacada=1&idxcomunidad=1378&idxnota=43300) almacenados en DB2 UDB . Va desde una tarea tan sencilla como puede ser elegir una fila de una base de datos dentro de un programa Perl hasta tópicos más avanzados, como lidiar con objetos grandes o invocar procedimientos almacenados.

Objetos inside-out

Sin querer ya tratamos este tema en el café Perl 0.4 (http://cafe.pm.org/boletines/CaFePerl-Issue04.html) pero acá ... acá tiene una vuelta más de rosca : http://dagolden.com/talks/20060117_whats_all_the_fuss.pdf

Fue liberado Perl 5.8.8 RC1 y 5.9.3

La versión de mantenimiento 5.8.8. involucra bug fixes e incorpora algunos módulos de CPAN nuevos. Pueden bajarlo desde en formato tar + gz desde

ftp://ftp.cpan.org/pub/CPAN/authors/id/N/NW/NWCLARK/perl-5.8.8-RC1.tar.gz y ver qué hay de nuevo en en http://search.cpan.org/~nwclark/perl-5.8.8-RC1/pod/perl588delta.pod.

También se liberó la versión que más adelante se convertirá en Perl 5.10 (5.9.3) y pueden probarla en ftp://ftp.cpan.org/pub/CPAN/src/perl-5.9.3.tar.gz así como ver que hay de nuevo en http://search.cpan.org/~rgarcia/perl-5.9.3/pod/perl593delta.pod y un pequeño roadmap en http://search.cpan.org/~rgarcia/perl-5.9.3/pod/perltodo.pod

Qué es Perl 6 ?

Un buen artículo de chromatic donde explica, a grandes rasgos, que és y cómo empezar a empaparse en Perl 6 : http://www.perl.com/pub/a/2006/01/12/what_is_perl_6.html

PostgreSQL y Perl

Sabían que no sólo esta base de datos tiene conectividad con con Perl sino que además acepta "stored procedures" hechos en Perl ??? http://www.oreillynet.com/pub/a/databases/2006/01/19/more-perl-in-postgresql.html !!!

Usando Perl para evaluar

Un profesor encontró tedioso y aburrido evaluar unas páginas HTML que sus alumnos debían hacer, entonces decidió pedirle una mano a Perl para que lo ayude ... y casualmente usar el mismo módulo que usamos en el PoC de este mes !!! (http://www.perl.com/pub/a/2006/01/19/analyzing_html.html)

Probando código C

Parece y suena ridículo, sobre todo en una publicación dedicada a Perl, pero la verdad es que se trata de que para probar código C hay una implementación del protocolo usado por los módulos dentro del namespace de Perl llamado Test (TAP – test anything protocol – http://use.perl.org/~petdance/journal/22057) llamada libtap. Disfrútenlo : http://www.onlamp.com/pub/a/onlamp/2006/01/19/libtap.html

Kevin Mitnick opina sobre el open source

Ahora que es un especialista en seguridad comentó que prefiere trabajar sobre código open source que propietario, mayormente porque es más fácil conseguirlo, no hay que recurrir a artimañas como ingeniería reversa o tener una copia ilegal, además de ser más seguro (http://www.tectonic.co.za/view.php?src=rss&id=839).

ActiveState es liberado

Esta famosa empresa es la que hace la distribución de Perl para Windows, y ha sido vendida nuevamente por la compañía Sophos (http://activestate.com/Corporate/Communications/Releases/Press1138659156.html)

Mudándose a OpenOffice.org

Si bien no está centrado en Perl nos da una muy buena idea de cómo convertir documentos en forma batch (http://www.xml.com/pub/a/2006/01/11/from-microsoft-to-openoffice.html)

Mordiditas de aquí y de allá

Operators

Artículo escrito por Víctor A. Rodríguez

Esta nueva entrega de “Mordiditas ...” está enfocada en el capítulo 3 de “Programming Perl” : Operators.

En el capítulo anterior (Atoms, Molecules y Built-in data types) se pueden ver una variedad de términos que por si solos no resuelven sino una pequeña parte de lo que un lenguaje necesita, para esto tenemos que relacionarlos entre si, pudiendo realizar operaciones con ellos (que sentido tiene tener datos si estos no pueden ser manipulados, interpretados, re-interpretados, y todo lo que sea necesario para convertirlos en información).

El abocarnos a analizar particularidades de cada uno de los operadores es una tarea ardua y que por suerte se haya muy bien tratada en este capítulo del libro, pero hay algo que me llamó poderosamente la atención y es lo relacionado con la precedencia. Qué es esto ? Seguramente lo debés haber usado más de una vez, y hasta casi seguro tenés una serie de reglas en tu cabeza para sobrellevarla. Te voy a dar un ejemplo : cuánto es 10 + 3 * 5 + 6 * 2. Es fácil, primero hacemos todas multiplicaciones y divisiones, y después todas las sumas y restas

```
10 + 3 * 5 + 6 * 2 = 10 + 15 + 12 = 37
```

Esto significa, ni más ni menos, que cuando se necesita resolver una ambigüedad o conflicto hay una serie de reglas que seguir que indican qué operaciones se hacen primero y cuáles siguen después (en orden). Son las llamadas reglas de precedencia.

En Perl no estamos ajenos a esta problemática. Si ejecutamos el siguiente código :

```
if ( $ARGV[0] < 10 && $ARGV[0] > 3 ) {
      print “entre 3 y 10”;
} else {
      print “fuera de rango”;
}
```

cómo interpretamos lo que está como condición del if :

```
1. $ARGV[0] < ( 10 && $ARGV[0] ) > 3
2. $ARGV[0] < ( 10 && $ARGV[0] > 3 )
3. ( $ARGV[0] < 10 ) && ( $ARGV[0] > 3 )
4. ( $ARGV[0] < 10 && $ARGV[0] ) > 3
```

A simple vista sabemos que la opción correcta es la tercera, porque lo hemos hecho un millar de veces para validar rangos de valores numéricos ... entonces a qué viene tanta alharaca con la precedencia ? Formalmente decimos que es interpretado de esa forma porque la operación < tiene precedencia sobre && (ver ítem 1 de Infografía) con lo cual el primero tiene prioridad sobre el segundo al momento de tomar un argumento.

Muy bien, hasta acá esta claro, pero que pasa en esos casos especiales donde uno tiende a hacer lo de todos los días, pero perl se empecina que hay un error? Y dónde está el error ? Tomemos el siguiente ejemplo :

```
while (<FH>) {
      next if length < 80;
      performLineBreaking( $_ );
};
```

A simple vista iteramos sobre cada línea de un archivo, si es de una longitud menor a 80 caracteres leemos la siguiente línea y en caso contrario la cortamos en líneas de 80 caracteres con performLineBreaking($_). Lo queremos ejecutar pero tenemos el siguiente error :

```
Use of "length" without parentheses is ambiguous
```

Lo que ocurre aquí es que por un lado length sin parámetros devuelve la longitud de lo que hay en $_, pero también es cierto que trata de tomar todo lo que está a su derecha como parámetro (en lugar de usar directamente $_) con lo que aquí debemos salvar la ambigüedad usando paréntesis :

```
next if length() < 80;
next if length $_ < 80;
```

Todo esto nos lleva a pensar que deberíamos manejarnos continuamente con la tabla de precedencias para cada línea que debamos hacer, lo que se hace más que engorroso. Entonces vamos a tratar de utilizar algún método que nos sirva en forma genérica. Un extremo de la solución sería poner todos los paréntesis (necesarios o no) para forzar la interpretación que deseamos. Con esto los ejemplo anteriores quedarían como :

```
10 + (3 * 5) + (6 * 2)
if ( ($ARGV[0] < 10) && ($ARGV[0] > 3) ) { ...
next if ( length() < 80 );
```

lo que si bien fuerza su interpretación correcta hace que la legibilidad se torne un poco complicada. Sabemos que no poner paréntesis en absoluto nos lleva hacia el extremo de una escritura más ágil, pero una interpretación por el parser que no siempre va a ser la misma que la que podamos entender a simple vista.

Para terminar de resolver nuestro acertijo veamos un poco más en detalle el siguiente ejemplo. Queremos hacer que si un número es divisible por dos entonces se le sume 10, de lo contrario se le sume 2, para lo cual se nos ocurre el siguiente código :

```
$a % 2 ? $a += 10 : $a += 2
```

Ahora bien, ocurre que si no los colocamos, la interpretación que hace perl es distinta de la nuestra :

```
(($a % 2) ? ($a += 10) : $a) += 2   # es interpretado como esto otro
($a % 2) ? ($a += 10) : ($a += 2)   # pero nosotros queremos esto
```

Con lo cual, en este caso, no nos queda otra opción que utilizar todos los paréntesis para que sea interpretado como la segunda opción.

Con este ejemplo final vislumbramos una solución y es usar paréntesis sólo cuando lo que intuitivamente pensamos que funciona de una forma perl lo interpreta de otra manera (es un buen intercambio entre velocidad de codificación y legibilidad). Imagínense que en Perl5 tenemos 24 niveles de precedencia, y aún en Perl 6 teniendo algo así como 18 niveles todavía es mucho, si no tenemos un método como este estamos perdidos en la nebulosa de los paréntesis.

Finalmente para acercarnos a Perl 6, y como nota curiosa, resulta que en el Apocalipsis 3 de Perl 6 (si este que nos ocupa y trata de operadores), y en el Larry Wall hizo algo curioso y poco común : en cierta forma documentó su propio proceso (enumerando los principios) para diseñar un sistema de operadores. Lo que viene a continuación y hasta el final del artículo es la traducción del mismo :

Déjenme enumerar algunos de los principios que sopeso cuando diseño un sistema de operadores.

- Distintas clases de operadores deberían verse distintos. Es por eso que los operadores de archivo lucen diferente que los operadores de strings o numéricos
- Clases de operadores similares deberían lucir similares. Es por eso que los los operadores de archivos lucen similares entre si
- Las operaciones más comunes deberían estar codificadas con el código de Huffman. O sea, los operadores más usados deberían ser más pequeños que los menos usados. Por su frecuencia de uso, el operador scalar de Perl 5 es demasiado largo, según mi estimación
- Preservar su cultura es importante. Perl tomó prestado muchos de sus operadores de otros lenguajes familiares. Por ejemplo, usamos el operador de exponenciación de Fortran ** . Según nos movemos a Perl 6, la mayoría de los operadores se tomarán prestados directamente de Perl 5
- Salirse de su cultura también es importante, porque es cómo aprendemos a otras culturas. Como un lenguaje multicultural explícito, Perl generalmente lo a hecho bien en este area, aunque siempre podemos hacerlo mejor. Ejemplos de intercambio cultural entre culturas de computación incluyen a XML y Unicode (No es sorpresa que estas características también permiten un mejor intercambio cultural entre culturas humanas. Es nuestra sincera esperanza)
- A veces los operadores deben responder a su contexto. Perl tiene muchos operadores que hacen cosas diferentes pero relacionadas según el contexto sea escalar o de lista
- A veces los operadores deben propagar el contexto a sus argumentos. El operador x actualmente hace esto con su argumento de la izquierda, mientras que los operadores de corto circuito lo hacen para los argumentos a su derecha
- A veces los operadores deben forzar el contexto a sus argumentos. Históricamente, los operadores matemáticos escalares de Perl fuerzan un contexto escalar en sus argumentos. Una de las RFCs discutidas propone revisar esto
- A veces los operadores deben responder polimórficamente a los tipos de sus argumentos. Las llamadas a métodos y el overloading trabajan de esta forma
- La precedencia de los operadores debería diseñarse para minimizar la necesidad de paréntesis. Se puede pensar en la precedencia de operadores como un ordenamiento parcial de los operadores de tal forma que minimice el número de apareamientos “no naturales” que requieren los paréntesis en un código tipo
- La precedencia de operadores debe ser lo más simple posible. La tabla de precedencia de Perl actualmente tiene 24 niveles. Esto puede o no ser demasiado. Probablemente podríamos reducirla a 18, si abandonamos la compatibilidad estricta de los operadores de C
- No se tiende a pensar mucho en la precedencia, así que esta debería ser diseñada para cumplir las expectativas. Desafortunadamente, las expectativas de alguien que conoce la tabla de precedencias no coinciden con las de alguien que no la conoce. Y Perl siempre cumplió las expectativas de los programadores de C, al menos hasta ahora. No hay mucho que pueda hacerse sobre las divergencia de las expectativas culturales

Infografía

1. Perl operators and precedence (http://www.perl.com/doc/manual/html/pod/perlop.html)
2. Just the FAQs: Precedence Problems (http://perl.plover.com/FAQs/Precedence.html)
3. Apocalypse 3: Operators (http://dev.perl.org/perl6/doc/design/apo/A03.html)
4. Exegesis 3 (http://dev.perl.org/perl6/doc/design/exe/E03.html)
5. Synopses 3 (http://dev.perl.org/perl6/doc/design/syn/S03.html)