Perl para apresurados

Juan Julián Merelo Guervós

Historial de revisiones

Revisión 1.0 Jul 2006
Preparando la primera versión para el curso de Extremadura

1. ¿Quién eres tú?

Eso mismo te estarás preguntando, que quién diablos eres, que a qué dedicas el tiempo libre, todo eso. Así que te vamos a echar una mano. Supongo que ya sabes programar, que el concepto de *variable* no va para ti asociado a la nubosidad ni el de *bucle* a la cabeza de Nellie Olleson. Puede que conozcas el C, sólo para precavidos, o hables con lengua de serpiente (pitón (http://www.python.org)), o incluso que el símbolo mayor y menor van para tí asociados de forma indisoluble a un acrónimo capicúa (http://www.php.org).

Vamos, que puede extrañarte las formas ignotas en las que un nuevo lenguaje de programación repite cachos de código o mete valores en variables o representa listas de datos, pero los conceptos en sí no son nada nuevo para tí. A tí, pues, va dirigido este mini-tutorial.

Supongo también que tienes prisa. Si no, no estarías leyendo este tutorial para *apresurados*. Estarías leyendo uno titulado, por ejemplo, *Perl para los que tienen todo el tiempo del mundo*. Es decir, que es necesariamente breve, con la idea de poder ser impartido (y espero que asimilado) en unas dos horas. Igual no te da tiempo a teclear todos los ejemplos de código, pero este ordenador que estás mirando tiene una cosa maravillosa llamada "corta y pega" con la que sin duda estás familiarizado, y que podrás usar para tu provecho y el de la Humanidad.

Y quizás todavía no lo sabes, pero *necesitas* saber Perl. Para vigilar ese fichero de registro y crear alertas que te avisen de lo inesperado. Para ese CGI terriblemente complicado. Para convertir una página web demasiado compleja en algo que también es complejo, pero que puedes leer con tu lector de cosas complejas favorito. Para hacer lo que siempre quisiste hacer: escribir poesía (http://www.perlmonks.org/index.pl?node=Perl%20Poetry) en tu lenguaje de programación favorito. En fin, donde quiera que haga falta convertir cosas en otras cosas, ahí hace falta saber Perl.

> **Sugerencia:** Y con ello damos entrada a la primera *flamewar* de este tutorial, que es donde tú, que estás entre el público, dices aquello de *Pues yo hago todo eso, y más, en (Fortran|Postscript|Haskell)*. Que vale, que si. Los lenguajes de programación son universales. Se puede hacer de todo con ellos. Y siempre es más fácil hacer algo en el lenguaje que uno conoce mejor. Pero al menos tendrás más donde elegir, ¿no?

Sobre todo, que no cunda el pánico. Y no te olvides de la toalla (http://es.wikipedia.org/wiki/Dia_de_la_Toalla).

2. Todo listo para despegar

> Si ya has usado algún lenguaje de scripting, lo más probable es que te aburras como un bivalvo en esta sección. Así que ahórrate un bostezo y pasa directamente a la siguiente. O si no descárgate los fuentes (http://www.cpan.org/src/latest.tar.gz) y echas un ratillo compilándolos en silencio, para no desmoralizarme a la parroquia.

Lo primero que necesitas en tu lista de comprobación son las cualidades de todo programador en Perl: la pereza, el orgullo y la curiosidad. No te preocupes si no tienes ninguna de ellas, las irás adquiriendo con el tiempo. Sobre todo la pereza. Y una cierta habilidad de entender lenguas muertas como el caldeo y el dálmata.

Segundo, necesitas amar a los camélidos. El Perl no es como esos otros lenguajes que incitan a la avaricia a través de la adoración de las piedras preciosas (http://www.ruby-lang.org), o a la hiperactividad por ingestión de bebidas excitantes (http://www.java.com). Los camellos son buenos. Los camellos son útiles. Llevan cosas encima. Tienen joroba. Amemos a los camélidos (las llamas también son camélidos (http://es.wikipedia.org/wiki/llama)).

No menos importante es tener un ordenador con sistema operativo. Incluso sin él (http://www.windows.com). Ejecuta lo siguiente para saber si lo tienes: **`perl -v`** a lo que el ordenador debidamente contestará algo así:

Figura 1. Contestación de un ordenador educado a `perl -v`

```
This is perl, v5.8.7 built for i486-linux-gnu-thread-multi
(with 1 registered patch, see perl -V for more detail)

Copyright 1987-2005, Larry Wall

Perl may be copied only under the terms of either the Artistic License or the
GNU General Public License, which may be found in the Perl 5 source kit.

Complete documentation for Perl, including FAQ lists, should be found on
this system using ‘man perl’ or ‘perldoc perl’.  If you have access to the
Internet, point your browser at http://www.perl.org/, the Perl Home Page.
```

si es que está instalado. Si no lo está, es poco probable que conteste eso. Incluso imposible. Dirá algo así como `bash: perl: command not found` e incluso pitará. El muy desagradable.

No hay que dejarse descorazonar por tal eventualidad. Encomendándonos al *Gran Camélido*, y sin necesidad de ver una vez más Ishtar, diremos en voz alta "Abracadabra" mientras que escribimos **`sudo`**

`yum install perl` o bien **`sudo apt-get install perl`** Si es que están en un linux no-debianita (en el primer caso) o en uno debianita (en el segundo). Habrá gente que incluso lo haga sin necesidad de bajarse del ratón. Pero los apresurados no usan el ratón salvo que sea estrictamente necesario. Que no es el caso. En otros sistemas operativos, lo mejor es ir a Perl.com (si es que no has ido todavía) (http://www.perl.com) y te bajes la versión compilada.

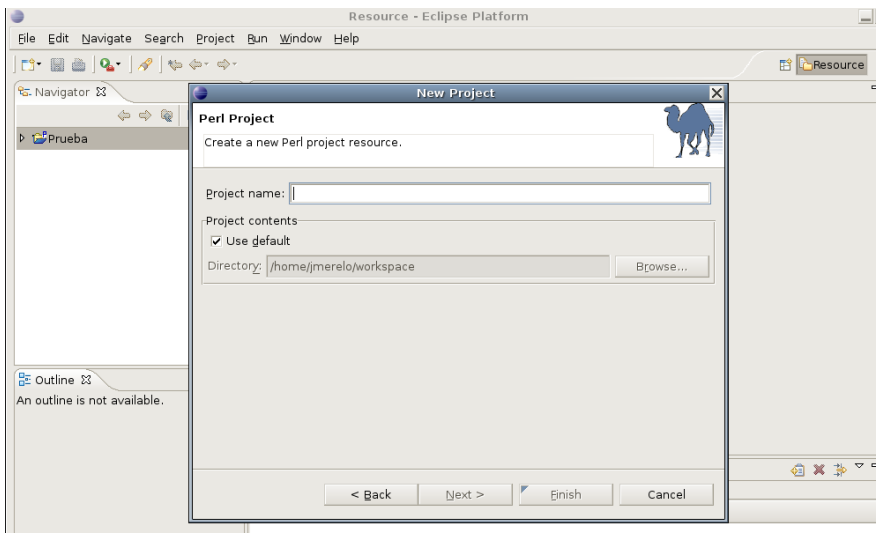
También puedes compilarlo tú. Pero no creo que lo hagas, porque eres un apresurado, y la compilación no está hecha para los apresurados (si eres usuario de Gentoo (http://www.gentoo.org), es el momento de abandonar este tutorial).

Lo que tienes o has instalado es un intérprete de Perl. Perl es generalmente un lenguaje interpretado, con lo que no hace falta ningún encantamiento intermedio para pasar de un programa escrito en Perl a su ejecución. Si te hará falta un editor. No *un* editor. *El* editor.

> **Sugerencia:** Los que apoyen al ínclito (x)emacs de este lado del *flamewar*, los que se queden con el sólido pero escuálido vi(m), de este otro lado. Los que estén con kate, jot, o incluso el kwrite, que elijan armas y padrinos y que pidan hora.

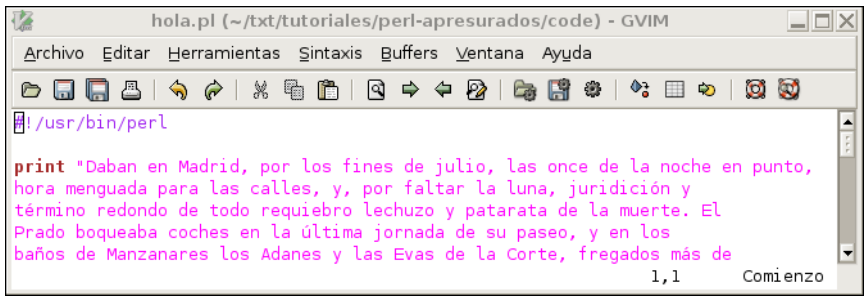
Vuelvo contigo entre el fragor de la batalla hablarte de otras opciones. No es que haya muchas, pero hay alguna. Por ejemplo, puedes usar el conocido entorno Eclipse (http://eclipse.org) con el plugin EPIC (http://e-p-i-c.sourceforge.net) para desarrollar proyectos, como se muestra en la figura siguiente.

Figura 2. Iniciando un proyecto en Perl con EPIC/Eclipse



Otros entornos de desarrollo, como PerlIDE o Komodo, o bien no siempre funcionan o bien son de pago. Si consigues que te lo compren, suertudo de ti. Si no, apoya proyectos de software libre. Suficientes personas han estado desarrollando sobre esos entornos durante el suficiente tiempo como para que presenten la sana apariencia que se muestra en la figura de abajo.

Figura 3. Editando un programilla con gvim



Un editor decente tiene que tener colorines. Y también cerrar paréntesis. Ninguno va a evitar que cometas errores, pero va a hacértelo lo más complicado posible.

> **Ejercicios.** ¿Tienes un intérprete de Perl instalado en tu sistema? ¿Tienes un editor (chulo, si es posible) para editar programas en Perl? Si la respuesta a alguno de ellos es *no*, ¿a qué esperas para tenerlos? Venga, te espero.

3. Comenzando una nueva carrera

Si has llegado hasta aquí, supongo que se te llevarán todos los diablos, porque con la hora que es, las camas están sin hacer y lo que s dice picar código, todavía no has picado nada. Y eso está bien: hay que convertir esa rabia en energía creativa, y aprovechando que uno de los diablos que se te llevan es cojuelo, escribir el siguiente fragmento de literatura:

```
#!/usr/bin/perl                                                    ❶
print "Daban en Madrid, por los fines de julio, las once de la noche en punto..."; ❷
print "\n";                                                        ❸
```

❶ Tratándose de diablos, lo mejor es usar los conjuros lo antes posible. En esta línea, clásica de los lenguajes interpretados y denominada shebang se escribe el camino completo donde se halla el intérprete del lenguaje en cuestión. Si está en otro sitio, pues habrá que poner otro camino. Por ejemplo, podría ser **#!/usr/local/bin/perl** o bien **#!/usr/bin/perl6.0.por.fin**. O **#!perl** y que se busque la vida. Si se trabaja (es un decir) en Windows, esa línea no es necesaria, pero es

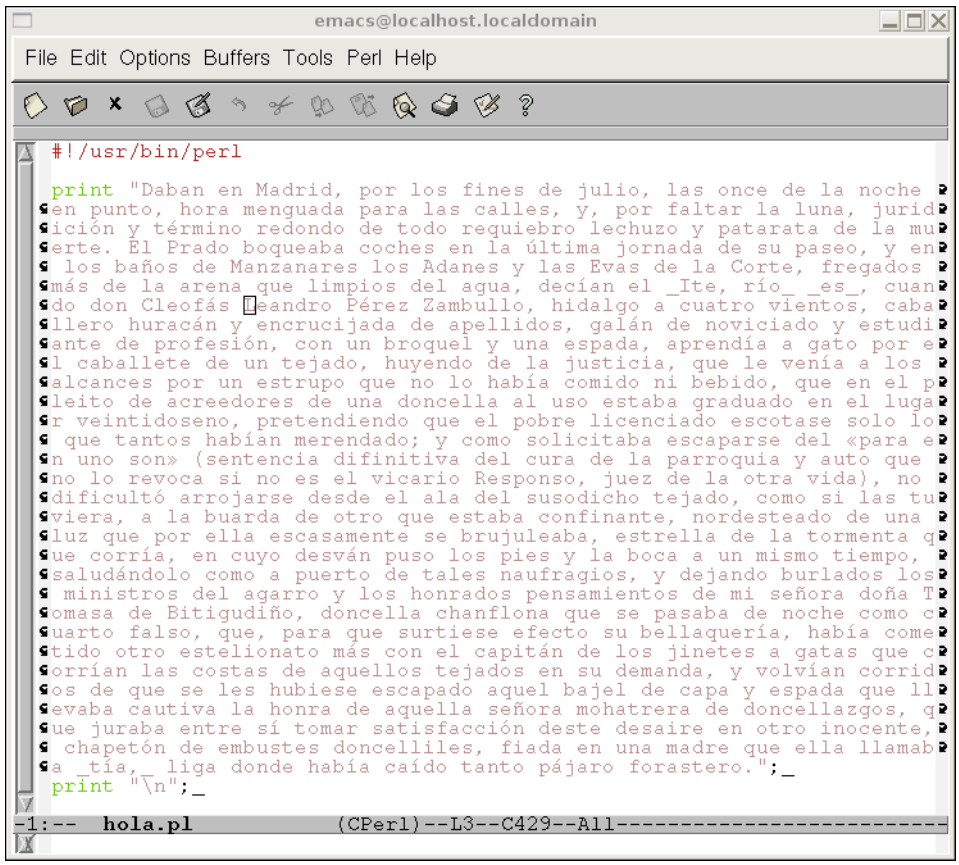
convenientepara que el programa sea compatible con otros sistemas operativos. Cuando un Unix/GNU/Linux decente y trabajador encuentra esa línea, carga ese programa y le pasa el resto del fichero para que lo interprete.

❷ Aquí se imprime, con el *nihil obstat* obtenido previamente. Obsérvense las comillas y el punto y coma. Las órdenes en Perl acaban con un punto y coma, para que quede bien claro dónde acaban y se puedan meter varias sentencias en una sola línea, con el objetivo de crear programas innecesariamente ofuscados. Lo que no se puede hacer en *esos otros lenguajes*. El `print` es herencia de aquellos primeros tiempos de los ordenadores, cuando el único periférico de salida era un convento de monjes trapenses dedicados a la sana tarea de copiar textos (y que se quedaron sin trabajo cuando el señor Hewlett se unió al señor Packard y crearon la impresora). En aquella época, la salida de un programa venía encuadernada en piel de cabrito y con todas las primeras letras de párrafo bellamente miniadas. Ah, tiempos aquellos, de los que sólo nos queda el nombre de una orden.

❸ Y aquí pasamos a la línea siguiente. Borrón y cuenta nueva. Se acabó lo que se daba. Si ya conoces algún lenguaje de programación, que se supone que lo conoces, pillín, porque te lo he preguntado en la primera sección, no hace falta que te explique que `\n` es un *retorno de carro*, ¿verdad?[1]

Desde un editor que cambie el color (y los tipos de letra) de acuerdo con la sintaxis del programa que se está editando tal como el `emacs`, el resultado debería ser algo similar al que aparece en la captura siguiente

Figura 4. Editando hola.pl en emacs



Aviso

El usar este tipo de texto, que incluye caracteres con acento, es bastante intencionado. En algunos editores puede que aparezcan caracteres extraños; habrá que cambiar la *codificación* para que entienda el conjunto de caracteres `iso-8858-1` o `latin1`.

Ejercicios. Elegir un editor no es un tema baladí, porque te acompañará en tu vida como desarrollador. Prueba diferentes editores disponibles en tu sistema mirando sobre todo a las posibilidades que tienen de adaptación a diferentes cometidos (comprobar la sintaxis y depurar, por ejemplo). Nadie te ata a un editor de por vida, pero cuanto antes lo elijas, antes empezarás a ser productivo. Así que ya estás empezando a usar el (x)emacs.

4. Viéndole las tripas al producto

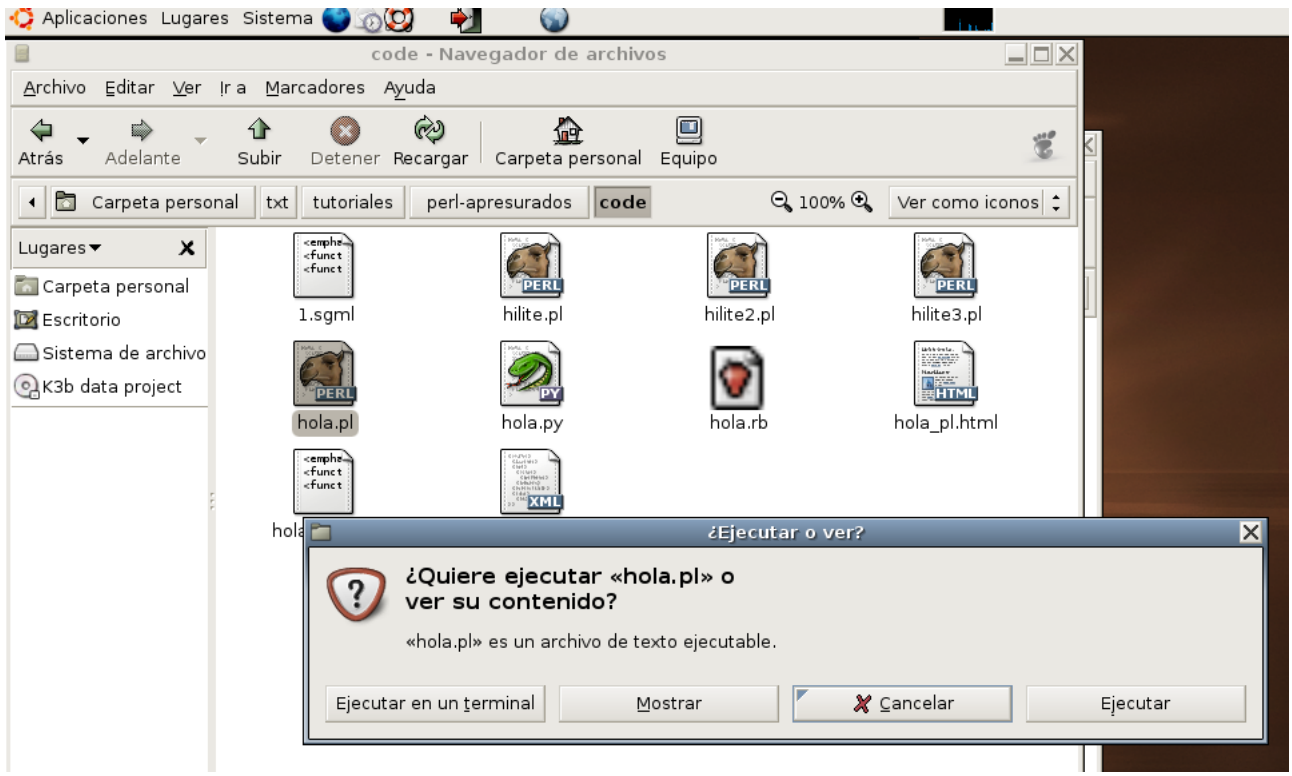
Mucho editar, mucho editar, pero de ejecutar programas nada de nada. Lo que hemos editado no deja de ser un fichero de texto, así que para ejecutarlo tendremos que llevar a cabo algún encantamiento para meterlo en el corredor de la ejecución.

Tampoco hace falta. Lo más universal es irse a un intérprete de comandos, colocarse en el directorio donde hayamos salvado el fichero, y escribir **`perl hola.pl`**. Pero ya que estás puesta, puedes hacer algo más: escribir **`chmod +x hola.pl`**, lo que convertirá al fichero en ejecutable, es decir, en algo que podemos correr simplemente tecleando su nombre, de esta forma:

```
jmerelo@vega:~/txt/tutoriales/perl-apresurados/code$./hola.pl
Daban en Madrid, por los fines de julio, las once de la noche en punto, hora menguada...
```

Pero el encantamiento este actúa también a otros niveles, pudiendo ejecutar el programa directamente desde esos inventos del averno llamados *entornos de ventanas*, como se muestra en la figura siguiente.

Figura 5. Ejecutando un programa en Perl desde Gnome



Como el Nautilus, el manejador de ficheros de Gnome es muy listo, se dice a si mismo (pero bajito):

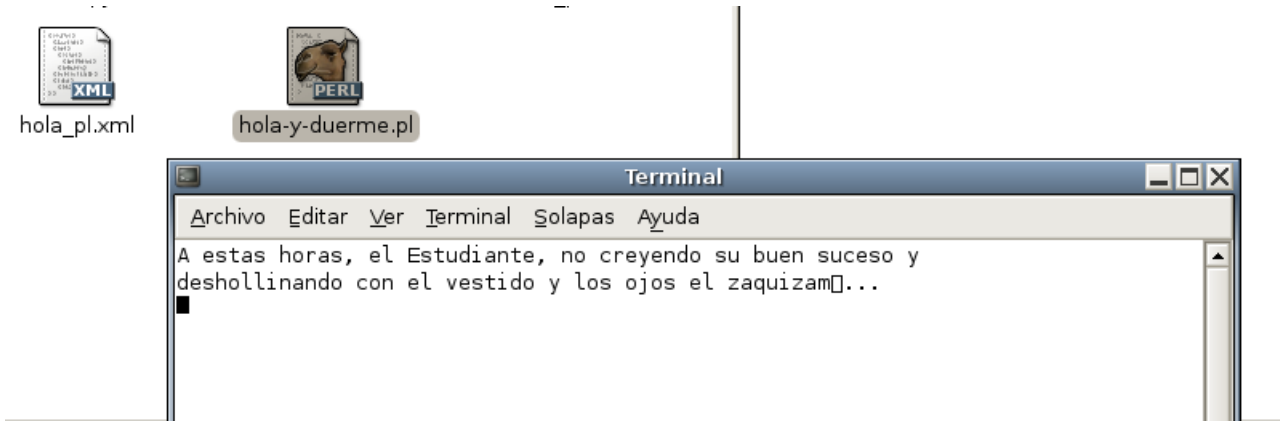
Pardiez, este fichero es ejecutable. ¿Qué puedo hacer con él? ¿Lo ejecuto? ¿Lo abro sin ejecutarlo? La duda me carcome. Se lo preguntaré al honorable usuario. El menú contextual (con el botón derecho del ratón) nos ofrecerá opciones similares. El problema es que si lo ejecutamos será visto y no visto.

Vamos a dejar entonces que el programa se quede clavado hasta nueva orden, con una pequeña modificación, que aparece en el siguiente listado

```
#!/usr/bin/perl
print "A estas horas, el Estudiante, no creyendo su buen suceso y
deshollinando con el vestido y los ojos el zaquizamí...\n";
sleep 10;
```

Que, con un poco de suerte, nos permitirá capturar una pantalla como la siguiente:

Figura 6. Terminal con el resultado de ejecutar el programa `hola-y-duerme.pl`



En otros sistemas operativos, cambiará el icono y la apariencia del terminal donde está el resultado, pero por lo demás, el resultado será el mismo. La única diferencia con el primer programa es la última línea, que le indica al programa que se quede quieto parao (dormido, de hecho) durante 10 segundos. Y que diga *cheeeeeese* (solo en ordenadores con interfaz gestual y/o emocional y/o audiomotriz/parlanchín).

Pero incluso así, puede que sea demasiado rápido para apreciar la sutileza de cada una de las órdenes, y haya que ejecutarlo paso a paso. Más adelante tendrás que *depurar* tus programas, porque cometerás errores, si, errores y tendrás que corregirlos sobre la marcha. De la forma más inteligente, además. Pero no hay que preocuparse, porque Perl tiene un depurador integrado. Ejecuta el programa de esta forma:

```
jmerelo@vega:~/txt/tutoriales/perl-apresurados/code$
perl -d hola-y-duerme.pl
Loading DB routines from perl5db.pl version 1.28
Editor support available.
```

```
Enter h or ‘h h’ for help, or ‘man perldebug’ for more help.

main::(hola-y-duerme.pl:3):     print "A estas horas, el Estudiante, no creyendo su buen suceso y
main::(hola-y-duerme.pl:4):     deshollinando con el vestido y los ojos el zaquizamí..\n";
  DB<1>
```

La opción **-d** del intérprete te introduce en el depurador, así, sin más prolegómenos. A partir de esa línea de comandos, puedes evaluar las expresiones de Perl que quieras, y, por supuesto, depurar el programa, ejecutándolo paso a paso, mirando variables, y todo ese protocolo inherente al mester de la programación. Para empezar, vamos a ejecutarlo pasito a pasito.

```
DB<1> R
Warning: some settings and command-line options may be lost!

Loading DB routines from perl5db.pl version 1.28
Editor support available.

Enter h or ‘h h’ for help, or ‘man perldebug’ for more help.

main::(hola-y-duerme.pl:3):     print "A estas horas, el Estudiante, no creyendo su buen suceso y
main::(hola-y-duerme.pl:4):     deshollinando con el vestido y los ojos el zaquizamí..\n";
```

, lo que empieza a hacerse ya un poco repetitivo. La orden **R** comienza a ejecutar el programa. En realidad, antes lo único que habíamos hecho es indicarle (educadamente) al depurador el programa que íbamos a depurar; ahora es cuando lo estamos ejecutando en serio. Bueno, todavía no, porque en este punto todavía no hemos ejecutado ni siquiera la primera línea. La salida del depurador nos indica <`main::(hola-y-duerme.pl:3):` la siguiente línea del programa (`3`) que vamos a ejecutar (y la 4 de camino, que para eso la orden ocupa dos líneas).

```
  DB<0> n                                                                    ❶
A estas horas, el Estudiante, no creyendo su buen suceso y
deshollinando con el vestido y los ojos el zaquizamí..
main::(hola-y-duerme.pl:5):     sleep 10;
```

❶ **n**, de *next*, siguiente, ejecutal la línea siguiente, es decir, justamente la que aparece al final de el ejemplo anterior

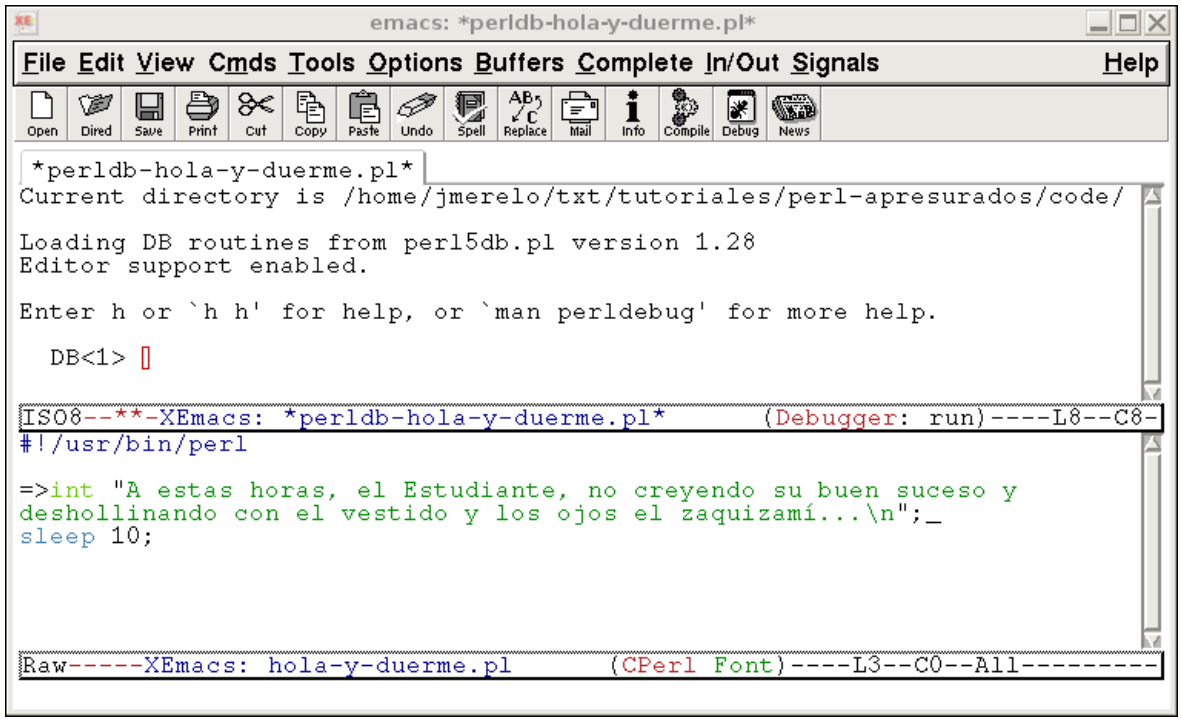
❷ Ésta es la salida de esa línea en particular; lo que hace es escribir lo que se encuentra entre las comillas.

❸ Y muestra la línea siguiente a ejecutar.

Como persona precavida vale por dos diablillos, no es mala idea tener siempre el depurador abierto para ir probando cosas. Te ahorrará más de una vuelta al editor a reescribir lo que ya está escrito. Además, es muy fácil. Si has elegido Un Buen Editor (o sea, el XEmacs), y te ha reconocido el programa como un

fichero Perl, tendrás una opción del menú llamada **perl**; desplegando ese menú, te aparecerá la opción **debugger**, eligiéndola te dará un resultado similar al que se muestra en la siguiente captura de pantalla:

Figura 7. Depurando un programa en el mismo editor XEmacs. La flecha está situada sobre la siguiente línea a ejecutar.



Desde este depurador se trabaja de la misma forma que en la versión de la línea de comandos, pero se pueden colocar puntos de ruptura usando el ratón, por ejemplo, y puedes ver las líneas que se están ejecutando en su contexto.

Con esto, ya estamos listos para abordar empresas más elevadas, y que nos llevarán mucho más lejos.

> **Ejercicios.** Familiarizarse con el depurador, creando un programa con las dos o tres cosas que se conocen, y viendo las diferentes órdenes; por ejemplo, cómo ejecutar un programa sin parar, o hasta una línea determinada, y cómo hacer que la ejecución se pare en una línea determinada. Recuerda, **h** es tu amiga.

5. Usando la sabiduría colectiva

Escribir está bien. Hay dos o tres personas que incluso se ganan la vida con ello[2]. Pero hace falta hacer algo más. Copiar a Faulkner, por ejemplo. Pero no sólo copiarlo. Ser más Faulkner que Faulkner. O mezclar Faulkner con, pongamos por caso, David Sedaris. O quizás Hemingway con Sedaris. Y llegado a este punto, te voy a contar un secreto. No hace falta que programes absolutamente nada. Ya hay gente que ha hecho lo que tú piensas programar en este preciso instante. De hecho, un vietnamita y un chavalote de Mondoñedo que acaba de terminar un módulo de FP segundo grado. Pero ambos dos son buenas personas, y legan su trabajo a la humanidad toda (inclusive tú). Si hay una sola cosa que haga al Perl superior a otros lenguajes de programación, son esas cosas que ha hecho la gente, empaquetado y colocado en un sitio común, llamado CPAN (http://search.cpan.org). CPAN significa, como probablemente ya habías adivinado, *comprehensive Perl Archive Network*, y es un sitio donde hay cienes, qué digo cienes, millardos de módulos que hacen todas esas cosas que se te hayan podido ocurrir, y otras cuantas que, ni harto de vino, se te podrían haber ocurrido. Pero hay que saber usarlo, claro.

> **Nota:** Si has tenido que pedirle a alguien que te instale el Perl, posiblemente sea el momento de que tengas a mano otra vez su teléfono o móvil, porque vas a volver a necesitarlo. No ahora. Más tarde. Mientras tanto, aunque no sea el día de apreciación del administrador del sistema (http://www.sysadminday.com/), aprovecha para pensar en él con cariño. Antes de que la falta de calor humano lo convierta en un operador bastardo del infierno (http://es.wikipedia.org/wiki/Bofh). Para instalar módulos de CPAN para que sean accesibles para todo el mundo hace falta tener privilegios de operador; sin embargo, puedes instalarlos sin problemas en tu propio directorio (por ejemplo, en `/home/miusuario/lib/perl`).

En CPAN hay módulos para todo. En particular, para manejar textos en diferentes idiomas. Por ejemplo, un módulo para dividir en sílabas texto en castellano llamado Lingua::ES::Silabas (http://search.cpan.org/~marcos/Lingua-ES-Silabas-0.01/). Un módulo es simplemente una biblioteca de utilidades para un fin determinado (o ninguno) escritas en Perl, o, al menos, empaquetadas para que se pueda acceder a ellas desde un programa en Perl. Una librería crea una serie de funciones a las que podemos acceder desde nuestros programas. Pero antes hay que instalarla. Y antes todavía, hay que ejecutar CPAN por primera vez:

```
jmerelo@vega:~$ sudo cpan
cpan shell -- CPAN exploration and modules installation (v1.83)
ReadLine support enabled

cpan>
```

Si realmente es la primera vez que lo ejecutas, te preguntará una serie de cosas. En la mayoría es razonable contestar la opción que te ofrezcan por defecto, pero en un par de ellas si tienes que elegir:

- Si no tienes privilegios de superusuario, tendrás que elegir un subdirectorio alternativo para colocar los módulos instalados.
- Es conveniente usar los repositorios más accesibles desde tu país, y por orden de frecuencia de actualización, para tener garantía de frescura de los módulos. Por ejemplo, dos buenas opciones

pueden ser http://debianitas.net/CPAN/ y http://cpan.imasd.elmundo.es/; aunque los otros repositorios con la extensión **.es** también suelen funcionar relativamente bien.

Una vez configurado todo, ya se puede instalar el módulo susodicho. Lo puedes hacer directamente desde la línea de comandos con

```
install Lingua::ES::Silabas
CPAN: Storable loaded ok
LWP not available
Fetching with Net::FTP:
  ftp://ftp.rediris.es/mirror/CPAN/authors/01mailrc.txt.gz
Going to read /home/jmerelo/.cpan5.9.3/sources/authors/01mailrc.txt.gz
CPAN: Compress::Zlib loaded ok
LWP not available
[...más cosas...]
Fetching with Net::FTP:
  ftp://ftp.rediris.es/mirror/CPAN/authors/id/M/MA/MARCOS/CHECKSUMS

  CPAN: Module::Signature security checks disabled because Module::Signature
  not installed.  Please consider installing the Module::Signature module.  You may also need to be al
  keyservers like pgp.mit.edu (port 11371).

Checksum for /home/jmerelo/.cpan5.9.3/sources/authors/id/M/MA/MARCOS/Lingua-ES-Silabas-0.01.tar.gz ok
Scanning cache /home/jmerelo/.cpan5.9.3/build for sizes
Lingua-ES-Silabas-0.01/
Lingua-ES-Silabas-0.01/Silabas.pm
Lingua-ES-Silabas-0.01/README
Lingua-ES-Silabas-0.01/Makefile.PL
Lingua-ES-Silabas-0.01/Changes
Lingua-ES-Silabas-0.01/MANIFEST
Lingua-ES-Silabas-0.01/test.pl

  CPAN.pm: Going to build M/MA/MARCOS/Lingua-ES-Silabas-0.01.tar.gz

Checking if your kit is complete...
Looks good
Writing Makefile for Lingua::ES::Silabas
cp Silabas.pm blib/lib/Lingua/ES/Silabas.pm
Manifying blib/man3/Lingua::ES::Silabas.3
  /usr/bin/make  -- OK
Running make test
PERL_DL_NONLAZY=1 /usr/local/bin/perl5.9.3 "-Iblib/lib" "-Iblib/arch" test.pl
1..9
# Running under perl version 5.009003 for linux
# Current time local: Mon Jul 10 23:34:36 2006
# Current time GMT:   Mon Jul 10 21:34:36 2006
# Using Test.pm version 1.25
ok 1
ok 2
ok 3
ok 4
```

```
ok 5
ok 6
ok 7
ok 8
ok 9
  /usr/bin/make test -- OK
Running make install
Installing /usr/local/lib/perl5/site_perl/5.9.3/Lingua/ES/Silabas.pm
Installing /usr/local/share/man/man3/Lingua::ES::Silabas.3
Writing /usr/local/lib/perl5/site_perl/5.9.3/i686-linux-thread-multi-ld/auto/Lingua/ES/Silabas/.packl
Appending installation info to /usr/local/lib/perl5/5.9.3/i686-linux-thread-multi-ld/perllocal.pod
  sudo make install  -- OK
```

que, efectivamente, descarga el módulo del repositorio espejo de CPAN más cercano (en este caso ftp.rediris.es), lo "compila", hace una serie de tests (sin los cuales no se instalaría siquiera), y efectivamente lo instala para que esté disponible para todos los programas que quieran usarlo (que no creo que sean muchos, pero alguno puede caer).

Vamos a ver ahora como se usa ese pozo de sabiduría.

Notas

1. Lo que es recuerdo también de aquellos mismos tiempos en que `STDOUT` era un convento, en el que para seguir en la página siguiente tenían que esperar que retornara el carro que les traía las pieles de becerro curtidas en las que escribían lo que el programador les ordenaba.
2. Los monjes trapenses de la congregación periférica de E/S ya no, desgraciadamente, y se dedican a la elaboración de un delicioso licor de alcachofa