Un lenguaje de casi 50 años de historia

Lisp (I)

David Arroyo Menéndez, José E. Marchesi

Lisp es una familia de lenguajes de programación con una larga historia. Desarrollado originalmente como una implementación de un modelo computacional, rápidamente se convirtió en el lenguaje favorito para hacer investigación en el ámbito de la inteligencia artificial.

Lisp ha sido pionero en el uso de estructuras de árbol (S-Expressions), recolección de basura, intérpretes y programación funcional. Hoy dialectos Lisp son usados en muchos campos, desde el desarrollo web a las finanzas, y también en la educación en ingeniería informática y, por supuesto, en investigación.

El nombre Lisp viene de "Procesamiento de Listas". La estructura de datos de listas y las primitivas para manejarlas son el denominador común de todos los dialectos Lisp, como ya explicaremos más adelante. Otras características comunes de los dialectos Lisp incluyen el tipado dinámico, el soporte a la programación funcional y la habilidad para manejar código fuente como datos.

Los lenguajes Lisp tienen una apariencia rápidamente reconocible. El código del programa es escrito usando la misma sintaxis de listas: la sintaxis de S-expressions. Cada subexpresión en un programa (o estructura de datos) está rodeada con paréntesis. Esto hace que los lenguajes Lisp sean fáciles de parsear y también de metaprogramar, esto es, crear programas que escriben otros programas. Esta es la mayor razón para su gran popularidad en los años 70 y 80; los programadores de inteligencia artificial creyeron que esta característica de Lisp resolvería fácilmente grandes problemas de la IA.

Lisp se especificó en 1958, convirtiéndose en el segundo lenguaje de programación de alto nivel que se usa hoy; solo Fortran es más viejo. Al igual que Fortran, Lisp ha cambiado bastante desde sus primeros día y numerosos dialectos han existido a través de su historia. Hoy, los dialectos Lisp más usados para programar son Common Lisp y Scheme.

Tabla 1.
Número de resultados en Google en agosto de 2005

Búsqueda	Número de resultados
"programming language" java	3.360.000
"programming language" c -c++	2.790.000
"programming language" c++	2.020.000
"programming language" perl	1.010.000
"programming language" python	881.000
"programming language" shell	568.000
"programming language" lisp	573.000
"programming language" ruby	292.000
"programming language" tcl	249.000

Tabla 2. Relación de lenguajes y número de proyectos en Sourceforge en agosto de 2005

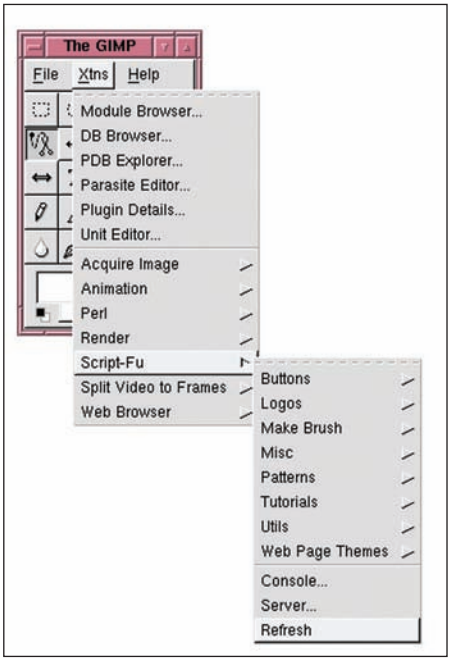
Lenguaje	Número de proyectos
C++	16.111
Java	15.849
C	15.396
Perl	6.039
Python+Zope	4.409
C#	2.646
Unix Shell	1.767
Tcl	892
Ruby	365
Scheme	195
Common Lisp	18
Emacs Lisp	9

Historia de Lisp

Information Processing Language (IPL) fue el primer lenguaje de Inteligencia Artificial (IA), desde 1955 o 1956, y ya incluía muchos de los conceptos, tales como procesamiento de listas y facilidades para la recursividad, que fueron usadas en Lisp.

Lisp fue inventado por John McCarthy en 1958 mientras él estaba en el MIT. McCarthy publicó su diseño en un artículo de Comunicaciones del ACM en 1960, titulado "Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I" (Funciones Recursivas de Expresiones Simbólicas y su Computación por Máquinas, Parte I). Curiosamente la segunda parte nunca fue publicada. Él mostraba que con unos pocos simples operadores y una notación para funciones, se puede construir un lenguaje de programación completo.

La notación original usada por McCartyhy estaba basada en "M-expressions". Esta fue rápidamente abandonada en favor de las S-expressions que él



Se puede automatizar trabajo en The Gimp gracias a scripts en Scheme Lisp.

originalmente propuso como una representación interna. Por ejemplo, la M-expresssion car[cons[A,B]] es equivalente a la S-expression (car (cons A B)).

Lisp fue originalmente implementado por Steve Russell en un ordenador IBM 704 y los dos lenguajes de macros para esta máquina llegaron a ser las operaciones primitivas para descomposición de listas: car (Contents of Address Register) y cdr (Contents of Decrement Register). Los dialectos Lisp todavía usan car y cdr para las operaciones que devuelven el primer ítem de una lista y el resto de la misma respectivamente.

El primer compilador de Lisp completo y escrito en Lisp fue implementado en 1962 por Tim Hart y Mike Levin. (AI Memo 39, 767 kB PDF (ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-039.pdf)). Este compilador ya introducía el modelo Lisp de compilación incremental, en el que las funciones compiladas e interpretadas pueden mezclarse libremente.

Desde su concepción, Lisp estaba estrechamente conectado con la comunidad investigadora en inteligencia artificial. En los años 70, se empieza a comercializar la IA y el rendimiento de los sistemas Lisp existentes llega a cuestionarse seriamente, en parte debido a utilizar un recolector de basura y en parte porque, en su representación de estructuras internas, Lisp llega ser difícil de arrancar en el limitado hardware de almacenamiento de memoria de aquellos tiempos. Esto motiva la creación de máquinas LISP: hardware de propósito específico para arrancar entornos Lisp y programas. Las técnicas de construcción de compiladores modernos y las gigantescas capacidades de los computadores de hoy en día han hecho esta especialización hardware innecesaria y hoy disfrutamos de entornos Lisp mucho más eficientes que otros lenguajes más populares.

Entre 1980 y 1990, se hace un gran esfuerzo para unificar los numerosos dialectos Lisp dentro de un lenguaje simple. El nuevo lenguaje, liderado por Guy Steele, se llamaría Common Lisp y era esencialmente un superconjunto de los dialectos reemplazados. En 1994, ANSI publicaba el estándar Common Lisp, "ANSI X3.226-1994 Information Technology Programming Language Common Lisp". En esos momentos el mercado que había entorno a Lisp ya era bastante pequeño, comparado con el de unos años atrás.

Mientras eso ocurría Richard Stallman anunció en 1983 el proyecto GNU, para crear un nuevo sistema operativo libre.

Puesto que lo que tendría que hacer el 90% del tiempo que empleara con la computadora para llevar a cabo su proyecto sería editar texto decidió empezar creando un editor llamado Emacs (editor de macros) y con Emacs surgió Emacs Lisp, también conocido como elisp. De este modo, el primer programa libre fue desarrollado por una comunidad de programadores Lisp. En 1990 el editor y el compilador de GNU estaría completos.

En 1986 en el mundo del software privativo surge otro lenguaje específico de la aplicación: AutoLisp (renombrado posteriormente como VisualLisp), un lenguaje para extender AutoCAD (un programa verdaderamente extendido en ingeniería y arquitectura) y que ha generado mucho código Lisp.

En 1995 en GNU están pensando en hacer aplicaciones para el escritorio y les gustaría tener un lenguaje de programación para poder escribirlas de un modo fácil y extensible como en el editor de gnu, es decir, GNU Emacs. En ese momento TCL está sirviendo muy bien para ese propósito. A Richard Stallman no le gustó básicamente porque, aunque hereda algunas ideas de Lisp, TCL no es Lisp y decidieron crear Scheme, el lenguaje de extensibilidad estándar para GNU.

No se eligió Common Lisp porque parecía bastante largo y prefirieron algo más sencillo. La idea era tener un intérprete de Scheme diseñado para ser enlazado dentro de aplicaciones tal y como TCL lo hacía. El principal beneficio de utilizar Scheme como lenguaje de extensión primario sería porque debido a la sintaxis de Lisp es fácil traducir código Lisp a TCL, pero a la inversa no. Esto es debido a la idea de S-Expression de Lisp, una idea recientemente reinventada con XML. Actualmente, Lisp no es tan usado como otros lenguajes; sin embargo, constituye una pieza fundamental para entender la historia de la informática y gente bastante inteligente sigue escribiendo código Lisp.

Lisp como lenguaje de extensión

Lisp ha sido usado múltiples veces como lenguaje de extensión. Los ejemplos más populares en la informática han sido Emacs y AutoCAD, pero no han sido los únicos: gimp, gnucash, texmacs son aplicaciones interesantes para las que nos será útil haber aprendido lisp.

La primera comunidad que se generó alrededor de software libre se generó alrededor de Emacs y el primer proyecto de GNU fue GNU Emacs. No es un proyecto cualquiera, es un proyecto importante. Tened en cuenta que más del 90% de lo que hacemos cuando estamos delante del ordenador es editar texto, así es que la herramienta con la que hacerlo tiene que ser cómoda.

Cuando Stallman creó el proyecto GNU se le pasó por la cabeza hacerlo todo con Lisp, pero por razones de eficiencia y dada la velocidad de las máquinas de aquel entonces se decantó por hacer las herramientas de bajo nivel en C. A pesar de eso en GNU habría Lisp, y así fue.

Emacs tiene un pequeño núcleo escrito en C; no obstante, todos los modos están escritos en Lisp y así cambiar el comportamiento de cualquier cosa que estés haciendo en Emacs significa evaluar Lisp, lo cual se hace de una manera tan natural e integrada con el editor que a veces es incluso usando el ratón.

De este modo, se pueden crear abreviaturas para *supercalifragilísticoespialidoso* que es una palabra larga que por una serie de razones vas a tener que escribir muchas veces en distintos documentos. También puedes crearte fácilmente una función Lisp para comentar el código de un lenguaje raro con el que vas a tener que programar durante unos meses y al que además te gustaría asociarle un atajo de teclado. Todo esto y muchísimo más es tan sencillo como escribir y evaluar Lisp sin tener que apagar el Emacs. Y cuando decimos sencillo es que es sencillo; Stallman comentaba que incluso administrativas acababan creando programas Lisp debido a que nadie les decía que estaban programando, que les solucionaba problemas reales y que había una buena documentación integrada. Todo ello se debe a que Lisp es un lenguaje interpretado que atrae a gente inteligente con gusto por hacer las cosas bien.

AutoCAD es software privativo muy utilizado en el mundo de la arquitectura y la ingeniería. Como ya comentamos, lleva utilizando

Tabla 3. Lenguajes más utilizados en Debian

Lenguaje	Debian 2.1		Debian 2.2		Debian 3.0		Debian 3.1	
C	27.800.000	74.89%	40.900.000	69.12%	66.500.000	63.08%	130.847.000	57%
C++	2.800.000	7.57%	5.980.000	10.11%	13.000.000	12.39%	38.602.000	16.8%
Shell	1.150.000	3.10%	2.710.000	4.59%	8.635.000	8.19%	20.763.000	9%
Lisp	1.890.000	5.10%	3.200.000	5.41%	4.090.000	3.87%	6.919.000	3%
Perl	774.000	2.09%	1.395.000	2.36%	3.199.000	3.03%	6.415.000	2.8%
Python	211.000	0.57%	349.000	0.59%	1.459.000	1.38%	4.129.000	1.8%
Fortran	735.000	1.98%	1.182.000	1.99%	1.939.000	1.84%	2.724.000	1.2%

Lisp como lenguaje de extensión desde 1986. El problema real es el mismo que en la edición de textos: al editar los planos de un edificio hay ciertas tareas que se vuelven repetitivas y es bueno tener un lenguaje interpretado que las automatice. En las herramientas de CAD, además estas operaciones repetitivas suelen ser operaciones matemáticas. Las herramientas libres de CAD a veces incluyen otros lenguajes de script Python, Perl o lenguajes propios. Creemos que atraerían muchos usuarios de AutoCAD si además añadieran Lisp como otro lenguaje de extensión.

Gimp es otro proyecto importante en el mundo del software libre, la primera librería gráfica libre, gtk, se desarrolló alrededor de este proyecto. Gimp utilizó Scheme Lisp como lenguaje de extensión; de este modo, es fácil crear scripts para automatizar la creación de cosas como, por ejemplo, logos.

Y como último caso de estudio tenemos GNU Cash. Se trata de un programa de contabilidad que también usa Scheme como lenguaje de extensión para importar ficheros QIF, generar informes y generar *pop ups* con el típico consejo del día.

Lisp en el mundo del software libre

Lisp es injustamente un gran desconocido especialmente en el mundo hispano e incluso dentro del software libre. Ciertos análisis poco rigurosos hacen parecer que Lisp es escasamente usado. Estos se basan en contar páginas en Google o Yahoo, o buscar en Sourceforge o Savannah.

Veamos, en la tabla 1, el número de resultados de buscar lisp, c++, java, python, tcl, perl y ruby en Google en agosto de 2005.

Estas búsquedas han sido realizadas en inglés, seguramente en castellano el panorama hubiera sido todavía más desolador para Lisp. No obstante, debemos considerar estos resultados como un indicador de uso de los lenguajes, no como el único indicador; una cosa es que se hable mucho de un lenguaje y otra distinta es que ese lenguaje se use mucho.

Ahora veamos, en la tabla 2, el número de proyectos albergados en Sourceforge. En realidad estos números lo único que quieren decir es la cantidad de proyectos que son aceptados en Sourceforge.

En "Introducción al software libre" (http://curso-sobre.berlios.de/introsobre/sobre.html/index.html) podemos leer un estudio bastante riguroso para entender la vitalidad de un lenguaje en el mundo del software libre. Éste se basa en tomar una distribución de software estable y ampliamente usada (por ejemplo RedHat, FreeBSD o Debian) y contar líneas de código en los paquetes que vienen en la distribución.

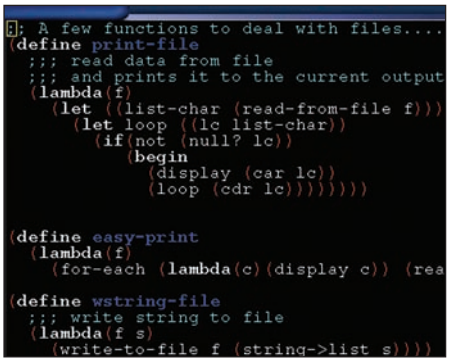
Al contar proyectos Sourceforge no distinguimos entre un proyecto maduro y ampliamente usado, como por ejemplo el kernel, de un proyecto que acaba siendo una idea no acabada; ambos cuentan como 1 y por ello eso es una mala estimación. Al contar páginas registradas en un buscador pasa algo parecido: una cosa es que se escriban muchas páginas acerca de un lenguaje y otra cosa es que realmente se use el lenguaje. Tomar una distribución de software y contar líneas de código de proyectos empaquetados (con cierta madurez) es una estimación bastante más acertada acerca de qué lenguajes estamos usando cuando usamos software libre.

Ahora veamos, en la tabla 3, los resultados en Debian.

Viendo estas estadísticas observamos que C es el lenguaje mayoritario y seguirá siéndolo durante bastante tiempo. C++ está en auge (se triplica desde la última distribución). Shell se sigue duplicando y está siendo el tercer lenguaje más usado. Lisp se mantiene como el cuarto lenguaje más usado; sin embargo, su número de líneas de código no crece al ritmo que crecen otros lenguajes. Python es un lenguaje que lleva un crecimiento más exponencial. Java es un lenguaje que empieza a considerarse dentro del mundo del software libre y ya llega a las 3.679.000 líneas de código y se sitúa en el 1.6% del tamaño de Debian.

Tal vez se podría decir que hay una cierta correlación entre los lenguajes que ocupan las primeras posiciones en Sourceforge y los lenguajes que crecen más rápidamente.

Seguramente la buena posición de Lisp dentro del mundo del software libre sea debida a que se usa en paquetes grandes e importantes como Emacs o Gimp.



```
; A few functions to deal with files....
(define print-file
  ;;; read data from file
  ;;; and prints it to the current output
  (lambda (f)
    (let ((list-char (read-from-file f)))
      (let loop ((lc list-char))
        (if (not (null? lc))
            (begin
              (display (car lc))
              (loop (cdr lc))))))))

(define easy-print
  (lambda (f)
    (for-each (lambda (c) (display c)) (rea

(define wstring-file
  ;;; write string to file
  (lambda (f s)
    (write-to-file f (string->list s))))
```

GNU emacs es un magnífico entorno de programación en general y de Lisp en particular.

Podéis ver radiografías de otras distribuciones e incluso de proyectos concretos e importantes en el "Curso de Introducción al Software Libre" (http://curso-sobre.berlios.de/introsobre/) escrito por Jesús González Barahona, Joaquín Seoane Pascual y Gregorio Robles, a quienes se les agradece el buen trabajo realizado.

Lisp como lenguaje para la inteligencia artificial

La historia de la inteligencia artificial es larga y podemos remontarnos hasta la antigua Grecia para buscar sus orígenes. Sin embargo, el momento en el que se comienza a utilizar el término y este se populariza es en 1956, cuando John McCarthy organiza el Dartmouth Summer Research Project on Artificial Intelligence (http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html). Los años sucesivos a ese verano fueron especialmente fructíferos: Newell, Simon y Shaw escribirían el "Logic Theorist" (http://www.j-paine.org/students/tutorials/tute/node11.htm) y el "General Problem Solver" y John Mc Carthy crearía Lisp y escribiría el artículo "Programs with Common Sense".

Lisp era el lenguaje de alto nivel que necesitaban; fue el primer lenguaje en implementar programación funcional, acercando las matemáticas al lenguaje; permite entender el

Tabla 4. Buscando un lenguaje para la IA en Google

Búsqueda	Número de entradas
c++ "artificial intelligence"	622.000
java "artificial intelligence"	581.000
lisp "artificial intelligence"	214.000
perl "artificial intelligence"	203.000
python "artificial intelligence"	184.000
prolog "artificial intelligence"	179.000

lenguaje hasta el nivel del intérprete y es extensible, permitiendo crear nuevos paradigmas de programación en el lenguaje.

Hasta 1971 no se inventaría Prolog, que se usaría sobre todo en combinación con Lisp, en los sistemas basados en reglas, debido a sus facilidades para el encadenamiento hacia atrás, entre otras cosas.

Así es que históricamente Lisp ha estado asociado a la inteligencia artificial y viceversa. Esto se puede ver en la mayoría de la literatura existente acerca de inteligencia artificial. Y echando un vistazo a lo que se ha hecho y se sigue haciendo en los principales centros de investigación: Carnigie Mellon, MIT o Stanford.

Resulta difícil hacer una estimación realista de cuánto se usa Lisp en el mundo de la inteligencia artificial. Esto es debido a que mucho del software escrito para inteligencia artificial es no libre. Peter Norvig comenta que hay una tendencia de un cierto abandono de Lisp en la inteligencia artificial, dado el número de entradas al buscar $(lenguaje de programación) + "artificial intelligence". Veamos, en la tabla 4, los resultados en agosto de 2005.

Al igual que en el caso de Lisp, en el software libre deberíamos interpretar no como que Lisp o Prolog no se usan dentro de la IA, sino que simplemente se habla mucho de esos lenguajes y inteligencia artificial en la web.

De nuevo, aquí sería interesante poder saber cuáles son los programas más usados de inteligencia artificial y estudiar en qué lenguajes están escritos. Los autores agradecemos sugerencias para poder realizar tales estudios.

¿Lisp en la web?

Resulta incluso atrevido pensar que Lisp pueda servir para programar la web. Para eso están los lenguajes de script ¿no? :). Para escribir esta sección hemos consultado los escritos de dos programadores de Lisp con conclusiones contrarias; estos son Philip Greenspun y de Paul Graham.

Philip Greenspun había sido programador de Lisp en el MIT durante su juventud. En 1995 empieza a programar junto con Ben Adida y Brian Tivol para crear sitios web. En 1998 tienen un montón de código reutilizable y lanzan una compañía: ArsDigita Communities System, basada en software con licencia GPL pero utilizando Oracle como base de datos. La idea de negocio era hacer desarrollos específicos teniendo un toolkit para hacer desarrollos rápidos en web.

Philip escribió un libro titulado "Philip and Alex's Guide to Web Publishing" (http://philip.greenspun.com/panda/), cuya lectura recomendamos a cualquiera que quiera desplegar un sitio web en Internet, incluso si no pretende ser programador. En el capítulo "Sites That Are Really Programs" (http://philip.greenspun.com/panda/server-programming) describe los lenguajes de programación que pueden ser interesantes para la web. Su opinión acerca de estos se puede resumir en la siguiente frase: "Un lenguaje de script es mejor que Java porque no tiene que ser compilado. Un lenguaje de script puede ser mejor que Lisp porque la manipulación de cadenas es simple". E ilustraba este hecho con 2 interesantes ejemplos de código:

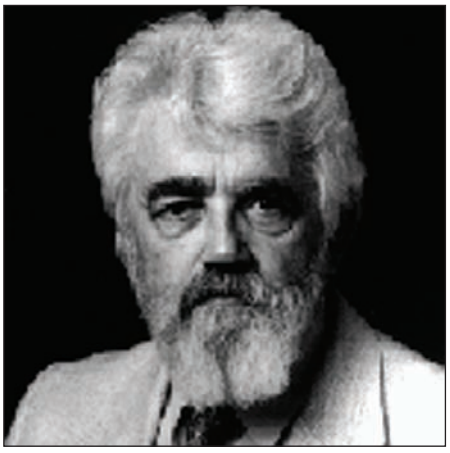
```
"posted by $email on $posting_date."
```

```
(concatenate 'string "posted by
" email " on " posting-date)
```

El primer ejemplo de código serviría para tcl, perl, php... y el segundo para Common Lisp. La primitiva *concatenate* es mucho más abstracta y poderosa, permitiéndo concatenar secuencias de paquetes TCP, secuencias de arrays tridimensionales... Pero si el 99,9% del trabajo va a ser concatenar texto el primer sistema es más cómodo. En la web los datos que se manejan salen de bases datos y ficheros, eso es, texto plano.

Philip Greenspun tuvo éxito, su software, actualmente es usado por importantes universidades e instituciones (http://dotlrn.org/partners/) y hay un buen número de empresas (http://openacs.org/community/companies/) que ofrecen: soporte, hosting, desarrollos, etcétera. De hecho, él ya no participa en su desarrollo. Prefiere aprender a pilotar helicópteros, ir a conciertos de jazz y dedicarse a sus clases en el MIT.

Paul Graham vio las cosas de manera distinta. Para él la programación en la web es un sitio donde puedes elegir el lenguaje que desees y, si puedes elegir lo que quieras, ¿por qué no usar Lisp? Lisp tiene funcionalidades poderosas que otros lenguajes no tienen, por ejemplo, las macros de Lisp, de las cuales Graham hizo un buen uso.



John McCarthy, creador de Lisp.

Con estas ideas en mente Paul Graham empezó una pequeña empresa para crear tiendas on-line, que se llamó Viaweb. Hoy esa pequeña empresa es Yahoo Store y Paul Graham ganó 40 millones de dólares con la venta ¿gracias a Lisp? Él asegura que sí en los documentos que escribió acerca del tema:

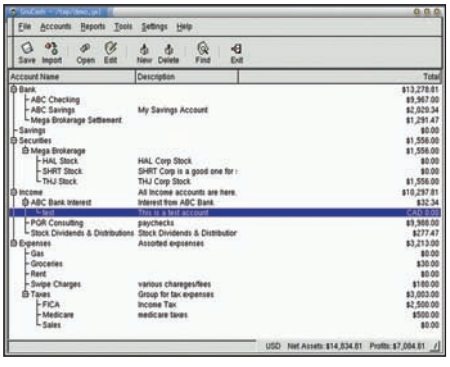
- Beating the Averages (http://www.paulgraham.com/avg.html)
- Lisp for Web-Based Applications (http://www.paulgraham.com/lwba.html)
- Revenge of the Nerds (http://www.paulgraham.com/icad.html)

Lecturas desde luego muy interesantes para quien planee desarrollar la próxima aplicación asesina de la web.

Conclusiones

En este artículo se ha visto la evolución histórica del lenguaje y se ha hecho un repaso a las áreas de aplicación del mismo haciendo un especial énfasis en lo que al software libre y, en especial, GNU/Linux se refiere.

El mes que viene se explicará cómo este lenguaje puede aprenderse en 10 minutos, explicando la esencia del mismo. Esperamos que hayáis disfrutado con la lectura.



En GNU Cash se usa Scheme como lenguaje de extensión.