

Introducción a Tiny Cobol

Jonathan Riddell, jr@jrippell.org

Una breve introducción a Tiny Cobol para aquellos que empiezan con Tiny Cobol o con Cobol en general (traducción libre de Juan J. Martínez <reidrac@usebox.net>, Abril de 2003).

Este Documento

Objetivo

Esta es una breve introducción a Tiny Cobol para aquellos que empiezan con Tiny Cobol o Cobol en general, llevando al lector a mirar a través de Cobol, mediante un programa sencillo y otro más complicado usando make.

Nuevas versiones de este documento

La última versión de este documento se puede encontrar en la página web de Tiny Cobol en <http://tiny-cobol.sourceforge.net>.

Copyrights

N.t.: Las condiciones de copia y distribución se mantienen en el idioma original.

This document is Copyright Jonathan Riddell 2001.

You may copy and distribute it only under the terms of the GNU Free Documentation License, available at <http://www.gnu.org/copyleft/fdl.html>

Agradecimientos

Gracias al equipo de Tiny Cobol y especialmente a Rildo Pragana (<http://www.pragana.net/>), nuestro líder lleno de fe.

Gracias a Richard Bland, mi instructor y maestro en todo lo relacionado con Cobol.

Agradecimientos también a las gente de Debian por incluir un programa sgmltools que funciona. Se aprecia mucho.

Traducciones

No hay traducciones de este documento todavía, pero cualquier ayuda es bienvenida.

N.t.: En el momento de esta traducción ya se dispone de traducciones al italiano, brasileño y español (castellano).

Y sí, este documento está en inglés británico. Los paréntesis son así (), y los periodos son algo que las chicas tienen cada mes. Considérese afortunado de que no lo escribiera en escocés (<http://scots.jriddell.org>).

Introducción a Cobol y a Tiny Cobol

Sobre Cobol

Cobol es uno de los lenguajes de tercera generación más veteranos. Fue desarrollado en 1959 (en la misma época que Fortran) para reemplazar a lenguajes ensamblador específicos del procesador (los lenguajes de segunda generación). El primer estándar ANSI fue el del 68 (aunque era usado comunmente desde 1961). Los estándares posteriores fueron el del 74 y el del 85. La mayoría de los programas usados hoy día se ajustan al estándar del 85 (y así lo hace Tiny Cobol).

Las principales ventajas de Cobol sobre lenguajes competidores (como C) eran la alta velocidad en entrada/salida que podía conseguir, la aritmética de coma fija para contabilidad exacta, y sintaxis similar al inglés que favorece la documentación y la legibilidad. Esto lo hizo muy útil para trabajos con procesamiento de grandes cantidades de datos como hacer el seguimiento de un millón de cuentas bancarias cada noche o procesar pagos combinados. La sintaxis similar al inglés hizo que los programas fueran fáciles de entender, así las aplicaciones para los negocios podían ser fácilmente modificadas durante su tiempo de vida según iban cambiando las reglas del negocio.

Cobol es significativamente diferente de lenguajes estructurados en bloques como pueden ser Pascal, C y los descendientes de C. No tiene bloques de estructuras y por lo tanto no hay forma de esconder variables; hay más libertad a la hora de escribir el programa; los números son más parecidos a la aritmética humana, por ejemplo: los números de coma fija o los números decimales se usan generalmente en lugar de la coma flotante; la entrada/salida está orientada a registros, no orientada a flujo; la recursividad no está permitida y el lenguaje en sí mismo es mucho mayor porque no se emplean librerías.

N.t.: Se ha empleado librerías en todo el documento, aunque la traducción correcta es bibliotecas. Coloquialmente se traduce library de esta forma, por lo que he optado por la claridad.

Se puede encontrar más información sobre Cobol en el FAQ de Cobol en <http://www.cobolreport.com/faqs/cobolfaq.htm>

Sobre Tiny Cobol

Tradicionalmente la clase de gente que usa Software Libre no es la clase de gente que usa Cobol. Esto es lo que el archivo de jerga (en inglés, *The Jargon File*) dice sobre el tema:

COBOL /koh'bol/ n. [COmmon Business-Oriented Language] (Sinónimo de malvado.) Un débil, prolijo y flácido lenguaje empleado por los operadores de tarjetas perforadas para hacer cosas aburridas sin importancia en dinosaurios mainframe. Los hackers creen que todos los programadores de COBOL son

amoldadores de código que no hay que tener en cuenta, y ningún hacker que se respete a si mismo admitirá nunca haber aprendido este lenguaje. Su propio nombre se pronuncia raramente sin expresiones rituales de disgusto u horror. Una de estas expresiones, muy popular, es la famosa observación de Edsger W. Dijkstra que dice: "El uso de COBOL lisia la mente; su enseñanza debería, por consiguiente, ser tomada como un acto criminal".

De hecho se trata de una crítica bastante dura. Pero para contradecir un poco esta postura hay muy buenas razones por las cuales Cobol es relevante hoy día.

- Hay alrededor de 100 billones de líneas de Cobol en uso hoy día.
- Gran cantidad de desarrollo aun se realiza en Cobol - se calcula que hay alrededor de un millón de desarrolladores en todo el mundo y que dos billones de líneas de Cobol se escriben cada año.
- Hay una demanda considerable de programadores de Cobol para trabajar en sistemas heredados.
- En muchos cursos universitarios de enseñanza Cobol.
- Usted podría discrepar con el autor del archivo de jerga.

Entonces, si Cobol es aun relevante hoy día, significa que será relevante para una buena cantidad de usuarios de Software Libre, y eso quiere decir que es útil y muy importante que exista un compilador Libre de Cobol.

En este punto es cuando Tiny Cobol entra en escena. Su creación está encabezada por Rildo Pragana y fue realizado en primer lugar para el entorno restringido de DOS. Tiny Cobol está ahora disponible para el entorno mucho más amigable para el programador, Linux, y también ha avanzado un gran trecho.

La Competición

- Cobol para GCC (<http://cobolforgcc.sourceforge.net/>), es un proyecto para desarrollar un compilador de Cobol que se integrará con la Colección de Compiladores de GNU, GCC. Aun no ha llegado a un estado usable.

Instalación

Obteniendo Tiny Cobol

Tiny Cobol solo funciona actualmente con Linux o FreeBSD en la arquitectura i386. Ha sido compilado satisfactoriamente en la plataforma Win32 empleando las herramientas de cygwin, pruebe buscando en los archivos de la lista de distribución si está interesado en ello (puede que hayan binarios disponibles). Parece que no será posible compilarlo en otros sistemas, aunque le invitamos a que lo intente.

N.t.: En el momento de esta traducción se dispone también de un port para BeOS, y Win32 posee ya un port nativo gracias a MinGW.

Puede descargar Tiny Cobol desde su página web en Sourceforge: <http://tiny-cobol.sourceforge.net> (<http://tiny-cobol.sourceforge.net/>). Están disponibles algunas versiones en RPM, pero la mayor parte de

la gente simplemente querrá descargar las versiones comprimidas con GZIP. El tamaño del fichero es de menos de medio megabyte.

Copie el fichero en algún lugar como `/usr/local/src` y descomprímalo (el paquete creará su propio directorio en el proceso). Entre en el directorio creado y ojee el fichero `README`, y lea con detenimiento el fichero `INSTALL`. El fichero `INSTALL` le dirá que librerías necesita; si no tiene esas librerías entonces instálelas desde los paquetes de su distribución o descárgelas de los sitios sugeridos. También verá como necesita GCC instalado, si no lo tiene instalado no llegará demasiado lejos así que instálelo ahora, nuevamente desde paquetes de su distribución o descargándolo desde GNU.

Compilando e Instalando

Para preparar la compilación, vaya al directorio de Tiny Cobol y escriba `./configure` (hay varias opciones para emplear con `configure`, consulte el fichero `INSTALL` para más detalles). Entonces escriba `'make'` para compilar. No necesita ser root para realizar este paso.

Ahora cambie de directorio, `'cd test_suite'` y escriba `'make tests'`. Usted podrá ver un flujo de resultados según el recién compilado compilador va trabajando con los programas de prueba. No todos estos programas funcionarán, recuerde que este software se encuentra en estado alfa, pero la mayoría deberían.

Ahora vaya al directorio original y escriba `'make install'`, lo que compilará la librería, el compilador y el preprocesador y otros pocos ficheros para que pasen a estar listos para usar. Necesitará ser root para este paso. Los binarios se instalan en `/usr/local/bin`, las librerías en `/usr/local/lib` y todos los otros ficheros en `/usr/local/share/htcobol`, a no ser que se especifique de otra forma con el programa `configure`.

¡Ayuda, no ha funcionado!

Si algo ha ido mal y no sabe como arreglarlo, lea este documento otra vez. Asegúrese de que tiene instalados todos los elementos importantes, como GCC y las librerías necesarias (debería ser posible conseguir ambas cosas independientemente de la distribución de Linux que emplee). Después lea la página web de Tiny Cobol en <http://tiny-cobol.sourceforge.net>.

¿Qué aun no ha conseguido solucionarlo? Suscríbase a la lista de correo de `tiny-cobol-users` y pregunte a las personas allí apuntadas. Puede apuntarse en

<http://lists.sourceforge.net/mailman/listinfo/tiny-cobol-users> o enviando un correo electrónico a `tiny-cobol-users-request@lists.sourceforge.net` (<mailto:tiny-cobol-users-request@lists.sourceforge.net>) con la palabra `subscribe` en el cuerpo del mensaje. La dirección para enviar es `tiny-cobol-users@lists.sourceforge.net` (<mailto:tiny-cobol-users@lists.sourceforge.net>). También debería revisar los archivos por si su pregunta se hubiera preguntado con anterioridad. Los archivos se encuentran en <http://lists.sourceforge.net/archives/tiny-cobol-users/>.

También hay una lista de distribución en brasileño (portugués) en

<http://listas.cipsga.org.br/cgi-bin/mailman/listinfo/cobol-br> y una página en brasileño en

<http://br.tinycobol.org>.

Primer programa

Editores

Como en la mayoría de los lenguajes de programación, un programa en Cobol es simplemente un fichero de texto, el cual usted edita con su editor de textos favorito. Afortunadamente, para evitar guerras religiosas, tanto Vim como Emacs trabajan con Cobol.

Emacs

No hay un modo Cobol que venga como estándar para GNU Emacs. En su lugar, descargue una copia de `cobol.el` del proyecto Cobol para GCC. Usted puede descargar el fichero directamente desde su CVS en <http://cvs.sourceforge.net/cgi-bin/cvsweb.cgi/gcc/cobol/cobol.el?cvsroot=CobolForGCC>.

Abra Emacs y compile el fichero Lisp con M-x byte-compile-file. Mueva el fichero `cobol.elc` resultante a `/usr/share/emacs/20.7/lisp/textmodes/` o el equivalente para su instalación. Finalmente añada este código al fichero `.emacs` (o `.gnu-emacs` en algunas distribuciones) en su directorio home:

```
(autoload 'cobol-mode "cobol" "COBOL Editing mode" t)
(setq auto-mode-alist
  (append '(("\\.cpp$" . c++-mode)
            ("\\.hpp$" . c++-mode)
            ("\\.lsp$" . lisp-mode)
            ("\\.scm$" . scheme-mode)
            ("\\.pl$" . perl-mode)
            ("\\.cbl$" . cobol-mode)
            ("\\.CBL$" . cobol-mode)
            ("\\.COB$" . cobol-mode)
            ("\\.cob$" . cobol-mode)
            ("\\.CPY$" . cobol-mode)
            ("\\.cpy$" . cobol-mode)
          ) auto-mode-alist))

;; Auto font lock mode
(defvar font-lock-auto-mode-list
  (list 'c-mode 'c++-mode 'c++-c-mode 'emacs-lisp-mode 'lisp-mode
        'perl-mode 'scheme-mode 'scheme-mode 'shell-script-mode 'cobol-mode
        'dired-mode)
  "List of modes to always start in font-lock-mode")

(defvar font-lock-mode-keyword-alist
  '(c++-c-mode . c-font-lock-keywords)
    (perl-mode . perl-font-lock-keywords)
    (cobol-mode . cobol-font-lock-keywords)
    (dired-mode . dired-font-lock-keywords))
  "Associations between modes and keywords")

(add-hook 'cobol-mode-hook
  '(lambda ()
    (set (make-local-variable 'dabbrev-case-fold-search) t)
    (set (make-local-variable 'dabbrev-case-replace) t)))
```

Abra un fichero Cobol en Emacs y escriba 'M-x cobol-mode' y 'M-x font-lock-mode' para conseguir bonitos colores. Gracias al equipo de Cobol para GCC por hacer este modo para Emacs.

VIM

Para una guía para trabajar con Cobol en Vim vaya a <http://dimensional.com/~sitaram/cobol/>.

Otros

Hay gran cantidad de editores disponibles. Uno interesante es THE, un editor que imita editores mainframe tales como Xedit. Está disponible desde <http://www.lightlink.com/hessling/>

Primer programa

Aquí hay un Hola Mundo en Cobol:

```
* Hello World Program
* GPL Copyleft Jonathan Riddell 2001
  IDENTIFICATION DIVISION.
  PROGRAM-ID.    hello.
  ENVIRONMENT DIVISION.
  DATA DIVISION.

  PROCEDURE DIVISION.
    DISPLAY "hello ," WITH NO ADVANCING
    DISPLAY "world!"
    STOP RUN.
```

La estructura del programa es bastante simple, cuatro secciones de las cuales dos tienen contenido. El PROGRAM-ID es un nombre simple para el programa, usted puede opcionalmente proporcionar otra información en IDENTIFICATION DIVISION. ENVIRONMENT DIVISION puede (opcionalmente ahora) contener información acerca su configuración, tal como el compilador que está usando. DATA DIVISION contiene declaraciones de variables que podríamos necesitar. Finalmente PROCEDURE DIVISION contiene las instrucciones del propio programa.

Usted notará que hay gran cantidad de mayúsculas en el código. Usar mayúsculas para las palabras reservadas de Cobol mantiene el programa más compatible hacia atrás con otros compiladores y asegura que las palabras reservadas se mantienen diferenciadas de los literales y variables. Con Tiny Cobol usted es perfectamente libre de no usar mayúsculas, lo que hará que su Bloq Mayús no se estropee tan rápido.

La otra cosa a notar es el espaciado. Debido a los viejos días de las tarjetas perforadas, Cobol mantiene un control estricto de dónde debe encontrarse el cada parte del código:

```
columna 1-6    : número de línea.
```

columna 7 : area de indicación,
asterisco para comentar lineas,
menos para continuación de lineas,
barra para saltar página en listados de
compilación, y en otro caso espacio.
columna 8-11 : Margen A, aquí comienza una división, sección,
identificadores de párrafo y algunos números de
nivel.
columna 12-72 : Margen B, para todo lo que no pertenece al Margen A.
columna 73-80 : área de identificación del programa.

Tiny Cobol no le fuerza a seguir este régimen, pero hacerlo mantendrá sus programas más ordenados, evitará que otros compiladores se quejen por el espaciado y hará feliz al modo Cobol en Emacs. No obstante, Tiny Cobol insiste en tener el asterisco que denota comentario en la primera o segunda columna.

Ahora a por la compilación del programa. Guarde el código anterior como hello.cob y escriba el siguiente comando:

```
htcobol hello
```

Hay posibilidades de que htcobol advierta de que \$COBDIR no está presente, en el caso de que haya instalado en el directorio por defecto esto no debería ser un problema, pero si usted cambió el directorio o si no desea ser molestado por extraños mensajes de advertencia, entonces añada 'export COBDIR=/usr/local/share/htcobol' o similar a sus fichero .bashrc o .profile. También le dirá que está procesando parámetros del compilador (literal: 'Processing compiler parameters'), eso es buena señal.

htcobol producirá los ficheros hello.lis y hello.s. hello.lis es una conversión del código Cobol a algo más amigable para el compilador. hello.s es código ensamblador, el equivalente al código máquina en términos legibles por el ser humano.

Actualización: Tiny Cobol ya no necesita compilar a código ensamblador y en su lugar compila directamente a programa ejecutable. Por lo tanto, los siguientes dos pasos pueden descartarse.

Si su shell no puede encontrar un programa llamado htcobol en primer lugar compruebe que su variable \$PATH le permite acceder a /usr/local/bin/htcobol o a lugar donde usted lo haya instalado (empee echo \$PATH). Si apunta bien, entonces vuelva a comprobar que se configuró, compiló e instaló correctamente.

Si usted escribió mal el código anterior, habrá notado que Tiny Cobol no es muy bueno todavía informando de errores. De hecho, lo único que usted obtendrá es un fallo de segmentación (literal: 'Segmentation Fault'. Compruebe otra vez que escribió todo correctamente, que no olvidó alguna puntuación y que los asteriscos de los comentarios se encuentran en la primera columna.

N.t.: El informe de errores ha evolucionado mucho desde que se escribió el artículo original, esta información está anticuada.

El siguiente paso es compilar el fichero en ensamblador file.s en código máquina. La orden para hacer esto es:

```
as -o hello.o hello.s
```

as forma parte del la Colección de Compiladores GNU, GCC, y su usted no lo tiene instalado es porque tiene una distribución muy divertida (teniendo en cuenta que usted ya ha compilado Tiny Cobol). Consulte la documentación de su distribución sobre como instalarlo.

hello.o es un fichero objeto, pero no un fichero ejecutable porque no ha sido enlazado con las librerías necesarias. Esta es la orden final:

```
gcc -o hello hello.o -lhtcobol -lm
```

Si usted ha llegado tan lejos, probablemente no tenga ningún problema con esta parte, pero si gcc no ha encontrado las librerías libhtcobol o libm entonces usted tendría que actualizar la base de datos de las librerías del sistema con la orden ldconfig.

Finalmente tenemos el programa funcional que puede ser ejecutado con:

```
./hello
```

Y debería obtener la siguiente salida:

```
Hello, world!
```

La cual era bastante predecible. Si observa el código del programa debería ser obvio lo que hace DISPLAY y que hace WITH NO ADVANCING (imprime sin un salto de línea).

Programación adicional

Otro programa

Aquí hay un programa ligeramente más complicado, cópielo en un fichero y grábelo como addition.cob.

```
IDENTIFICATION DIVISION.  
PROGRAM-ID.      addition.  
ENVIRONMENT DIVISION.  
  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
77  myvar PIC 999 .  
77  a  PIC 99V999 VALUE 1.777 .  
77  b  PIC 99V99  VALUE ZERO .  
77  c  PIC 9V9    VALUE 5.5 .  
77  d  PIC 9v9    VALUE 5.5 .  
  
PROCEDURE DIVISION.
```



```

DISPLAY "Please enter a number"
ACCEPT myvar
DISPLAY "That number was ", myvar
ADD a TO b ROUNDED
DISPLAY "b is ", b
ADD c TO d
DISPLAY "d is ", d
STOP RUN.

```

Usted puede compilarlo con los mismos pasos que empleó para hello.cob, si encuentra algún fallo de segmentación (literal: 'Segmentation Fault', compruebe el código del programa antes de abandonar.

N.t.: Se refiere a errores de compilación.

Si usted es novato con Cobol, hay un poco que entender. En primer lugar tenemos algunas variables declaradas en WORKING-STORAGE SECTION de DATA DIVISION. Cobol necesita que todas las variables se declaren de esta forma. Las otras secciones DATA DIVISION tratan con ficheros y el enlazado.

Aquí todas las declaraciones de variables empiezan por '77'. Esto significa que la variable se encuentra en el nivel 77: no tiene estructura interna y no hay una relación definida con otra variable, no forma parte de un registro (o lo que C llama estructuras). El nombre de la primera variable es myvar; los nombres de las variables pueden estar tanto en mayúsculas como en minúsculas, pero con Tiny Cobol debe ser consistente en todo su programa.

La cláusula PIC especifica el tipo de variable. Usando nueve se indica que se trata de una variable decimal y empleando 3 nueve decimos que la variable puede tomar los valores desde 0 a 999. Se puede poner una 'S' delante de los nueve para que la variable sea con signo. Una 'V' es la declaración PIC significa que la variable tiene un punto decimal asumido en la posición de V. Para variables grandes puede escribir 9(15), lo que describe a una variable que posee 15 posiciones. Empleando Aes (varias A) en lugar de nueve define una variable alfabética, y empleando Xes (varias X) se define una variable alfanumérica. Finalmente un valor inicial se puede asignar a la variable empleando VALUE.

Los programadores de Cobol sin experiencia pueden estar preguntándose como se indica un final de sentencia. Los finales de bloque se marcan con un punto y aparte, pero las sentencias no se marcan en absoluto, en su lugar el compilador de Cobol reconoce que una nueva sentencia comienza cuando lee una de las diferentes palabras reservadas de Cobol. Los finales de línea no son importantes en Cobol. Otra cosa a señalar en el código anterior son las comas añadidas. Las comas son completamente opcionales en Cobol, puede ponerlas en cualquier punto que desee o en ningún sitio y no hay ninguna diferencia. Úselas con medida y conseguirá un código mucho más claro.

Cuando se ejecuta el programa debería preguntar por un número, y entonces mostrar ese número junto a los dos cálculos que realiza. El número no puede ser mayor de los 3 espacios reservados para él, sino tan solo los tres últimos caracteres se cogen. Si se proporcionan menos de tres caracteres, se completa con ceros a la izquierda. Caracteres no numéricos mostrarán código para corregir errores.

La suma de las variables a y b debería funcionar bien, aunque notará que la respuesta se redondea tal y como pedimos. La suma de c y de no se llevará a cabo debido a que la respuesta es demasiado grande para caber en la variable d. Cambie c a, digamos, 1.5 y todo funcionará correctamente.

Make

Actualización: Tiny Cobol ahora compila directamente a un programa ejecutable, en lugar de código ensamblador. Mucho de lo que sigue es por lo tanto obsoleto (aunque los conceptos deberían ser útiles).

Ya debería empezar a estar harto de la rutina de htconol, as, gcc. Bien, existe una forma mejor, make. Make toma como entrada un fichero llamado Makefile y usa las instrucciones en él para realizar el trabajo sobre lo que necesita ser compilado en un sencillo paso. También comprueba que fichero necesita ser compilado antes de hacer nada, si ninguna de las dependencias del programa han cambiado desde la última compilación, no es necesario hacer nada.

Las reglas de Makefile tienen la siguiente forma:

```
objetivo: dependencias
orden
```

El objetivo (el cual debe ser escrito en la primera columna) es el fichero que quiere conseguir. Las dependencias son los ficheros que son necesarios para conseguir el objetivo. La orden (que debe estar en una segunda línea y precedida de un TAB) es la orden que normalmente se escribe en el shell para generar el objetivo. Aquí podemos ver el Makefile para addition:

```
#makefile simple para un programa cobol en un solo fichero
```

```
addition:addition.o
gcc -o addition addition.o -lhtcobol -lm
```

```
addition.o:addition.s
as -o addition addition.s
```

```
addition.s:addition.cob
htcobol addition
```

Guárdelo como Makefile en el mismo directorio de addition.cob y escriba make. Debería compilar, ensamblar y enlazar por usted.

Por defecto make interpretará la primera regla en el fichero. Esta regla es para crear el binario addition, pero este último depende de addition.o, por lo que debe hacerse primero. addition.o depende de addition.s, así que esa es la primera regla que make ejecuta. Ahora se puede crear addition.o y finalmente addition.o está actualizado y permite crear addition. Si usted intenta escribir make otra vez, make no hará nada porque ninguno de los ficheros ha cambiado y no hay necesidad de hacer nada (make mira la fecha de los ficheros para determinarlo). Cambie addition.cob ligeramente y ejecute run, y el proceso se lleva a cabo otra vez.

Make avanzado

Pero el Makefile anterior tiene un problema: ¿qué ocurre si queremos adaptarlo para trabajar con otro programa? Hay algunos cuantos 'addition' a cambiar, y con un programa más complicado eso podría ser

un trabajo importante. Afortunadamente make tiene bastantes buenos trucos en la manga. Aquí tiene un makefile más avanzado:

```
#makefile para un programa cobol de un solo fichero

PROGRAM := addition
LIBS := -lhtcobol -lm
OBJECTS := $(PROGRAM).o

$(PROGRAM):$(OBJECTS)
gcc -o $(PROGRAM) $(OBJECTS) $(LIBS)

#regla para cualquier fichero acabando en .o
#depende de el mismo prefijo con .s
# $@ es el objetivo
%.o:%.s
as -o $@ $<

%.s:%.cob
htcobol $(PROGRAM)
```

El Makefile comienza con algunas variable. PROGRAM es el nombre del programa que estamos haciendo. Se puede observar que se usa en la primera regla (precedido de un signo dolar y delimitado por paréntesis). LIBS contiene las librerías que necesitamos para compilar el programa. OBJECTS contiene todos los nombres de objeto necesarios, en este caso es solo uno y tiene el mismo nombre que el programa pero con una .o al final, y así es como se ha declarado.

La segunda regla se ajusta a cualquier objetivo acabado en .o (%.o) y requiere el mismo fichero con un sufijo .s (%.s es una dependencia). La variable automática \$@ se ajusta al nombre del objetivo y \$< se ajusta a la primera dependencia. A partir de este punto usted ya debería poder seguir los pasos hasta el objetivo final.

Si quiere adaptar este Makefile a otro programa Cobol, todo lo que tiene que hacer es cambiar el valor de la variable \$PROGRAM.

El final

Esto es todo por ahora. Probablemente debería haber aquí otro capítulo con un programa que usara más de un fichero y el preprocesador, pero eso por hoy queda más allá de mis conocimientos. Quizás también falten algunos enlaces a recursos sobre Cobol.

Jonathan Riddell, jr@jridell.org

Traducción por Juan J. Martínez, reidrac@usebox.net