

Introduzione a Tiny Cobol

Jonathan Riddell, jr@jridell.org

Una breve introduzione a Tiny Cobol per chi non conosce Tiny Cobol o il Cobol in generale (traduzione di Fabio Teatini, <teafab@tiscali.it>).

1. Questo documento

1.1. Obiettivo

Una breve introduzione a Tiny Cobol per coloro che non conoscono Tiny Cobol o il Cobol in generale. Il lettore verrà accompagnato in un giro panoramico attraverso il Cobol, realizzerà un primo programma, e un altro più complicato mediante l'uso di make.

1.2. Nuove versioni di questo documento

L'ultima versione di questo documento può essere trovata presso il sito web di Tiny Cobol <http://tiny-cobol.sourceforge.net>. (NdT: Le informazioni contenute in questo documento sono aggiornate alla versione 0.55 di Tiny Cobol)

1.3. Nota sui copyright (in lingua originale)

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>

1.4. Nota sui copyright (in lingua italiana)

Questo documento è coperto dal Copyright Jonathan Riddell 2001.

Può essere copiato e distribuito soltanto rispettando i termini della licenza GNU Free Documentation License, disponibile presso <http://www.gnu.org/copyleft/fdl.html>

1.5. Ringraziamenti

Grazie alla squadra di Tiny Cobol, e in special modo a Rildo Pragana (<http://members.zoom.com/rpragana>), la nostra guida fidata.

Grazie a Richard Bland, il mio tutore e maestro di tutto quel che riguarda il Cobol.

Un plauso va anche alla gente di Debian, per avervi introdotto un programma sgmltools funzionante. É stato molto apprezzato.

1.6. Traduzioni

Non esistono ancora traduzioni di questo documento (oltre a questa italiana, ndT), e qualsiasi aiuto è benvenuto.

Inoltre, questo documento è in Inglese britannico. Le parentesi graffe sono queste {}, e i punti (che in inglese sono detti "period" ed assomiglia al nostro "periodo", ndT) sono cose che le ragazze hanno ogni mese. Ritenetevi fortunati, piuttosto, dal momento che questo documento non è scritto in lingua scozzese (<http://scots.jriddell.org>).

2. Introduzione al Cobol e a Tiny Cobol

2.1. Cobol

Il Cobol è uno dei primi linguaggi di terza generazione. È stato sviluppato nel 1959 (è coetaneo del Fortran) per sostituire i linguaggi assembly specifici dei processori (i linguaggi di seconda generazione). Il primo standard ANSI è lo standard 68 (benché fosse comunemente utilizzato fin dal 1961). Gli standard successivi furono gli standard 74 e 85. La maggior parte dei programmi Cobol, utilizzati a tutt'oggi, sono conformi allo standard 85 (e anche Tiny Cobol è conforme a questo standard).

I principali vantaggi del Cobol sui linguaggi concorrenti (come il C) erano: la maggiore velocità di input/output che consentiva, l'aritmetica con il punto decimale fisso che lo rendevano utile nei programmi per la contabilità, e la sintassi simile a quella della lingua inglese lo rendevano ottima la leggibilità e la documentazione. Queste caratteristiche lo hanno reso molto utile nei job per l'elaborazione di grandi moli di dati, o per effettuare l'aggiornamento notturno di un milione di conti correnti bancari, o ancora per l'elaborazione delle buste-paga. La sintassi simile all'inglese ha reso i programmi più facili da comprendere, cosicché i programmi aziendali possono essere modificati facilmente e superare il loro normale ciclo di vita.

Il linguaggio Cobol è significativamente diverso dai linguaggi strutturati in blocchi come sono i linguaggi Pascal, C, e i suoi discendenti. Il Cobol non è strutturato in blocchi, e quindi non c'è modo di nascondere le variabili; c'è maggiore libertà mentre si scrivono i programmi; i numeri sono trattati in maniera più vicina alla rappresentazione aritmetica umana, ad esempio la virgola fissa o i numeri decimali vengono solitamente usati al posto della virgola mobile; l'I/O è orientato al record, e non allo stream o flusso; la ricorsione non è permessa e il linguaggio stesso è più vasto, in quanto non usa librerie.

Altre informazioni sul Cobol possono essere trovate nelle FAQ relative al Cobol situate presso <http://www.cobolreport.com/faqs/cobolfaq.htm>

2.2. Tiny Cobol

Tradizionalmente, nella tipologia di persone che utilizzano il software Libero non sono comprese quelle che usano il Cobol. Questo è quanto il Jargon File afferma in merito:

COBOL /koh'bol/ n. [COMmon Business-Oriented Language] Un linguaggio fiacco,

prolisso, e cadente, usato da perforatori di schede per compiere azioni stupide e noiose sulle macchine mainframe dinosauriche. Gli hacker credono che tutti i programmatori COBOL siano macinatori di codice, e nessun hacker che si rispetti ammetterà mai di aver imparato il linguaggio. Il suo nome esteso viene raramente proferito senza espressioni rituali di disgusto o di orrore. C'è una famosa osservazione di Edsger W. Dijkstra, secondo cui "L'uso del COBOL storpia la mente; il suo insegnamento dovrebbe, quindi, essere considerato come un crimine."

Indubbiamente questa è una critica alquanto violenta. Ma a contrastarla ci sono buone ragioni che spiegano perché il Cobol sia tutt'oggi rilevante.

- Ad oggi, sono utilizzate circa 100 miliardi di righe di Cobol.
- Moltissimo sviluppo viene tuttora effettuato in Cobol - si pensa che nel mondo ci sia 1 milione di sviluppatori, e che 2 miliardi di righe di Cobol vengano scritte ogni anno
- C'è una notevole domanda di programmatori Cobol da impiegare su sistemi già esistenti.
- In molti corsi universitari viene insegnato il Cobol
- Si può non essere concordi con l'autore del Jargon File

Se quindi il Cobol ha ancora oggi la sua importanza, significa che sarà importante per molti utenti di software Libero, e da ciò deriva l'utilità e l'importanza di un compilatore Libero per il Cobol.

E qui arriviamo al punto che riguarda Tiny Cobol. La sua creazione è stata diretta da Rildo Pragana ed è stato inizialmente rivolto all'ambiente ristretto del DOS. Ora Tiny Cobol è rivolto all'ambiente di programmazione molto più amichevole di Linux, ed è progredito moltissimo.

2.3. La concorrenza

- Cobol per GCC (<http://cobolforgcc.sourceforge.net/>), è un progetto per sviluppare un compilatore Cobol che sarà integrato nella GNU Compiler Collection, GCC. Non

è ancora in uno stato utilizzabile.

3. Installazione

3.1. Come procurarsi Tiny Cobol

Attualmente Tiny Cobol funziona solo con Linux o FreeBSD su un computer i386. Di recente è stato compilato con successo anche sulla piattaforma win32 con l'impiego degli strumenti di cygwin e quindi, se siete interessati, potete provare a cercare informazioni in merito negli archivi della mailing list (sono comunque disponibili anche i binari). Molto probabilmente, Tiny Cobol non sarà utilizzabile su qualche altra piattaforma, ma siete invitati a provare comunque.

Potete scaricare Tiny Cobol dal suo sito web presso Sourceforge:

<http://tiny-cobol.sourceforge.net> (<http://tiny-cobol.sourceforge.net/>). Lì sono disponibili alcune versioni in formato RPM, ma la maggior parte delle persone vorrà scaricare le versioni compresse con GZIP. La dimensione del file è inferiore al mezzo megabyte.

Copiate questo file in qualcosa come `/usr/local/src` e decomprimetelo (verrà così creata la relativa directory). Entrate nella directory, date uno sguardo al file `README`, e leggete tutto il file `INSTALL`. Il file `INSTALL` vi dirà di quali librerie avete bisogno; se ve ne manca qualcuna, installatela dal CD della vostra distribuzione, o scaricatela dai siti suggeriti. Inoltre questo file vi dirà che dovete avere installato GCC; se non è installato, ora non dovrete andare molto lontano per farlo, usando ancora il CD della vostra distribuzione, o un server ftp, oppure scaricandolo dal sito di GNU.

3.2. Compilazione e installazione

Per preparare la compilazione, recatevi nella directory di Tiny Cobol e digitate `./configure` (esistono svariate opzioni per adattare `configure`, si veda il file `INSTALL`

per leggerne i dettagli). Poi digitate 'make' per compilare. Non avrete bisogno di essere root per farlo.

Ora cambiate directory, 'cd test_suite' e digitate 'make tests'. Vedrete un flusso di risultati dovuti al lavoro di compilazione dei programmi di prova, ottenuti impiegando il compilatore appena compilato. Non tutti supereranno la fase di compilazione, (si ricordi che questo software è in stato alpha), ma la maggior parte dovrebbe farcela.

Adesso andate nella directory originale e digitate 'make install'; questo copierà una libreria, il compilatore, il preprocessore, e pochi altri file, così da renderli pronti all'uso. Dovete essere root per farlo. I binari vengono posizionati in /usr/local/bin, le librerie in /usr/local/lib, e tutti gli altri file vanno in /usr/local/share/htcobol/, tranne il casi in cui venga specificato diversamente attraverso il programma configure.

3.3. Aiuto, non funziona!

Se qualcosa va storto e non sapete come aggiustarlo, leggete questo documento di nuovo. Assicuratevi di avere installato tutto ciò che serve, come GCC e le librerie necessarie (con le quali dovrete essere in grado di cavarvela, qualsiasi distribuzione Linux utilizzate). Successivamente leggete la pagina web di Tiny Cobol presso <http://tiny-cobol.sourceforge.net>.

Non siete ancora sulla buona strada? Iscrivetevi alla mailing list tiny-cobol-users e chiedete gentilmente ai partecipanti. Potete iscrivervi presso <http://lists.sourceforge.net/mailman/listinfo/tiny-cobol-users> o inviando una e-mail a tiny-cobol-users-request@lists.sourceforge.net (<mailto:tiny-cobol-users-request@lists.sourceforge.net>) con 'subscribe' nel corpo del messaggio. L'indirizzo per pubblicare è tiny-cobol-users@lists.sourceforge.net (<mailto:tiny-cobol-users@lists.sourceforge.net>). Dovreste anche ricercare negli archivi, nel caso in cui alla vostra domanda fosse già stato risposto prima. Gli archivi sono presso <http://lists.sourceforge.net/archives/tiny-cobol-users/>.

4. Primo programma

4.1. Gli editor

Come la maggior parte dei linguaggi di programmazione, un programma Cobol è soltanto un file di testo che potete scrivere con il vostro editor preferito. Sia vim che Emacs fanno un buon lavoro col codice Cobol, e questo ci evita le guerre di religione.

4.1.1. Emacs

Non esiste una modalità Cobol che venga fornita come standard in GNU Emacs. In compenso, scaricatevi una copia del file `cobol.el` dal progetto 'Cobol for GCC'. Potete scaricare il file direttamente dal loro CVS presso <http://cvs.sourceforge.net/cgi-bin/cvsweb.cgi/gcc/cobol/cobol.el?cvsroot=CobolForGCC>.

Aprirete Emacs e compilate il file Lisp con `M-x byte-compile-file`. Spostate il file risultante `Cobol.elc` nella directory `/usr/share/emacs/20.7/lisp/textmodes/` oppure nella directory equivalente per la vostra installazione. Aggiungete infine questo codice al file `.emacs` (o al `.gnu-emacs` di qualche distribuzione) situato nella vostra directory personale (in `/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 'scribe-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)))
```

Aprite un file Cobol in Emacs, e digitate 'M-x cobol-mode' e 'M-x font-lock-mode' per ottenere la sintassi a colori. Grazie alla squadra 'Cobol for GCC' per aver realizzato questa modalità di Emacs.

4.1.2. VIM

Una guida per scrivere in Cobol con VIM, si trova in <http://dimensional.com/~sitaram/cobol/>.

4.1.3. Altri editor

Oltre a quelli detti, ci sono moltissimi altri editor. Tra questi, è interessante THE, un editor che imita gli editor del mainframe come Xedit. É disponibile presso <http://www.lightlink.com/hessling/>

4.2. un primo programma

Ecco il programma Hello World in 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 struttura del programma, alquanto semplice, è composta di quattro sezioni, due delle quali hanno un qualche contenuto. Il PROGRAM-ID è semplicemente un nome per il programma, ma nella IDENTIFICATION DIVISION possono esserci anche altre informazioni. La ENVIRONMENT DIVISION può (è facoltativo) contenere informazioni riguardo la vostra configurazione, come il compilatore che state usando. La DATA DIVISION contiene tutte le dichiarazioni delle variabili di cui avete bisogno. Infine, la PROCEDURE DIVISION contiene le istruzioni vere e proprie del programma.

Si noti l'uso abbondante di lettere maiuscole nel codice. L'uso delle maiuscole per le parole chiave del Cobol, permette di mantenere la compatibilità con altri compilatori e garantisce che le parole riservate siano in risalto rispetto alle stringhe e alle variabili.

Con Tiny Cobol siete completamente liberi di non usare le maiuscole, così da evitarvi il rapido logorìo del tasto di Caps Lock.

L'altra cosa da notare è la spaziatura. Fin dai tempi andati delle schede perforate, il Cobol ha mantenuto un controllo stretto sull'incolonnamento delle varie parti di codice:

```
colonna 1-6   : numero di riga.  
colonna 7     : indicatore dell'area,  
                asterisco per righe di commento,  
                segno meno per la continuazione delle righe,  
                barra diagonale per salto-pagina del listato  
                altrimenti spazio.  
colonna 8-11  : Margine A, ove iniziano le divisioni, le sezioni,  
                gli identificatori di paragrafo e qualche nu-  
                mero di livello.  
colonna 12-72: Margine B, per tutto ciò che non va messo  
                in Margine A.  
colonna 73-80: area di identificazione del programma.
```

Tiny Cobol non vi forza a mantenere questo regime, ma in questo modo potete mantenere più puliti i vostri programmi, e potete anche evitare che tutti gli altri compilatori facciano storie riguardo alla spaziatura, e inoltre la modalità Cobol di Emacs ne sarà lieta. Tiny Cobol, tuttavia, persiste sull'uso dell'asterisco nella prima o nella seconda colonna a rappresentare un commento.

Adesso passiamo alla compilazione del programma. Salvate il codice sopra riportato come `hello.cob` e digitate il seguente comando:

```
htcobol hello
```

htcobol potrebbe emettere l'avvertimento (warning) che `$COBDIR` non è impostato; fintanto che l'installazione è avvenuta nella directory predefinita, questo non dovrebbe essere un problema; ma se avete modificato la directory, o se siete disturbati da strani messaggi di warning, aggiungete `'export COBDIR=/usr/local/share/htcobol'` o qualcosa di simile al vostro file `.bashrc` (o a `.profile`). Se poi vi si presenta il messaggio `'Processing compiler parameters'`, si tratta di un buon segno.

htcobol produrrà i file hello.lis e hello.s. hello.lis è una conversione del codice Cobol in qualcos'altro di più gestibile da parte del compilatore. hello.s è il codice assembly, il codice in linguaggio-macchina in formato leggibile.

Se la vostra shell non riesce a trovare un programma di nome htcobol, verificate prima di tutto che la vostra variabile \$PATH punti a /usr/local/bin/htcobol o comunque alla directory in cui avete fatto l'installazione (digitate echo \$PATH). Se punta correttamente, allora verificate attentamente che sia stato tutto configurato, compilato e installato correttamente.

Se avete scritto scorrettamente il codice sopra riportato, vi renderete conto che Tiny Cobol non è ancora molto bravo a riferire gli errori. Infatti, tutto quello che otterrete è un Segmentation Fault. Verificate attentamente di aver scritto tutto correttamente, non tralasciate alcuna punteggiatura, e mettete gli asterischi dei commenti nella prima colonna.

Il prossimo passo è la compilazione del file assembly hello.s in codice macchina. il comando per farlo è:

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

as fa parte della collezione di compilatori GNU GCC, e se non l'avete installato dovete proprio avere una strana distribuzione (visto che avete appena adesso compilato Tiny Cobol). Cercate informazioni sull'installazione di as nella documentazione della vostra distribuzione.

hello.o è un file oggetto, ma non è eseguibile direttamente perché non è stato linkato con le librerie necessarie. Questo è il comando finale:

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

Se siete arrivati fin qui non dovrete avere avuto i problemi trattati prima ma, se gcc non ha trovato le librerie libhtcobol o libm, potreste dover aggiornare il database delle librerie con il comando ldconfig.

Alla fine avrete un programma funzionante che potrà essere eseguito con:

```
./hello
```

Dovreste poterne vedere l'output:

```
Hello, world!
```

Era facilmente prevedibile. Se osservate il codice del programma dovrebbe essere ovvio quello che fanno DISPLAY e WITH NO ADVANCING (stampa senza andare a capo).

5. Programmazione avanzata

5.1. Un altro programma

Ecco un programma leggermente più complicato; inseritelo in un file e salvatelo col nome 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 "Inserite un numero"  
    ACCEPT myvar  
    DISPLAY "Il numero è ", myvar
```

```
ADD a TO b ROUNDED
DISPLAY "b è ", b
ADD c TO d
DISPLAY "d è ", d
STOP RUN.
```

Potete compilare questo programma con lo stesso procedimento seguito per hello.cob e, se incontrate un qualche Segmentation Fault, controllate bene il vostro programma prima di reclamare.

Se siete a digiuno di Cobol, notate che vi troverete un inghippo. Inizialmente abbiamo dichiarato alcune variabili nella WORKING-STORAGE SECTION della DATA DIVISION. Il Cobol impone che tutte le variabili siano dichiarate in questa maniera. Le altre sezioni della DATA DIVISION si occupano dei file e dei link.

Tutte le dichiarazioni di variabile iniziano, in questo programma, con '77'. Si intende che la variabile è posta al livello 77: non ha struttura interna e non è definita alcuna relazione con le altre variabili, inoltre non fa parte di un record (che in C corrisponde a una struct). Il nome della prima variabile è myvar; i nomi delle variabili possono essere sia in lettere maiuscole che in lettere minuscole, ma con Tiny Cobol dovete mantenere coerenza in tutto il programma.

La clausola PIC specifica il tipo di variabile. Con i numeri 9 si intende che la variabile è un decimale, e utilizzando tre numeri 9 si ottiene una variabile che può assumere i valori da 0 a 999. Ci può anche essere una 'S' prima dei numeri 9, e ciò attribuisce il segno alla variabile. Una 'V' nella dichiarazione PIC significa che la variabile riserva una virgola decimale nella posizione con la V. Per variabili di grandi dimensioni si può scrivere 9(15), che è una variabile con 15 posizioni. L'impiego delle lettere A al posto dei numeri 9 trasforma la variabile in alfabetica, e utilizzando le X la variabile diventa alfanumerica. Infine, con VALUE può essere assegnato un valore iniziale alla variabile.

I programmatori non avvezzi al Cobol potrebbero rimanere stupiti di come venga segnalata la conclusione di un'istruzione. La fine di un blocco di codice viene indicata con un punto, e le istruzioni non sono segnalate in alcun modo, mentre invece il compilatore Cobol riconosce l'inizio di una nuova istruzione quando incontra una delle tante parole caratteristiche del Cobol. La fine di una riga non è importante in Cobol.

Un'altra cosa da notare nel codice qua sopra, sono le virgole aggiuntive, Le virgole sono del tutto facoltative in Cobol e potete metterle ovunque vogliate (anche da nessuna parte): non fa differenza. Usatele con parsimonia e manterrete il programma più ordinato.

Quando lanciate il programma vi viene chiesto un numero, che poi viene stampato insieme ai risultati dei due calcoli. Il numero dato non può occupare uno spazio maggiore delle 3 posizioni allocategli, in caso contrario vengono accettate solo le ultime tre cifre. Se vengono forniti meno di tre caratteri, il numero verrà stampato con degli zeri iniziali. L'uso dei caratteri non numerici condurrà all'emissione di un qualche codice d'errore.

La somma delle variabili a e b dovrebbe essere effettuata correttamente, e comunque vedrete che il risultato viene arrotondato come da noi desiderato (mediante `ROUNDED, ndT`). la somma di c con d non produrrà nulla, infatti il risultato è troppo grande perche sia ospitato dalla variable d. Provate a cambiare c in 1.5, ad esempio, e tutto andrà bene.

5.2. Make

A questo punto potreste anche esservi un po' stancati della sequenza `htcobol, as, e gcc`. Ebbene, c'è un modo migliore di procedere, con `make`. `Make` prende in input un file chiamato `Makefile`, e ne usa le istruzioni contenute per effettuare in un colpo solo tutte le compilazioni che servono. Inoltre, `make` controlla che un file non debba essere compilato prima di fare qualche altra cosa; se nessuna delle dipendenze del programma è cambiata dall'ultima compilazione, non viene effettuato nulla.

Le regole del `makefile` sono della forma:

```
obiettivo:dipendenze
  comando
```

L'obiettivo (che deve iniziare in prima colonna) è il file che volete ottenere. Le dipendenze sono i file che da cui esso dipende. Il comando (che deve trovarsi sulla

seconda riga e deve essere preceduto da un tab) è quello che verrebbe solitamente immesso dalla shell per fabbricare il vostro file. Ecco qui un Makefile per il programma addition:

```
#un semplice makefile per un programma cobol composto da un singolo file

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
```

Salvate questo file col nome 'Makefile' nella stessa directory di addition.cob, e digitate 'make': questo comando compila, assembla e linka tutto al posto vostro.

Come comportamento predefinito, make effettua la compilazione del file relativo alla prima regola presente nel Makefile, così da produrre il binario di addition; ma questo file dipende da addition.o, che così deve essere creato prima. addition.o dipende da addition.s, e questo è il primo comando che make esegue. Ora può essere creato addition.o, e infine addition.o viene compilato per creare addition. Se provate a eseguire nuovamente make, non succederà nulla; in effetti, poiché nessuno dei file ha subito modifiche, make non ha nessun compito da svolgere (e questo, make lo determina consultando la data dell'ultimo salvataggio dei file). Fate una qualche modifica ad addition.cob, ed eseguite make: verrà ripetuto l'intero procedimento.

5.3. Uso avanzato di make

Il precedente Makefile, però, ha un problema: come si fa a trasformarlo per fargli compilare un altro programma? Nel nostro caso vanno sostituiti solo alcuni "addition"

ma, con programmi più complicati di questo, ciò potrebbe richiedere delle ore. Fortunatamente, make ha parecchi assi nella manica. Ecco un makefile più avanzato:

```
#un makefile per un programma cobol composto da un singolo file

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

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

#la regola per tutti i file che terminano con .o
#dipende dai file con stesso prefisso e che finiscono con .s
# $@ è l'obbiettivo
%.o:%.s
as -o $@ $<

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

Il Makefile comincia con la definizione di alcune variabili. PROGRAM è il nome del programma da creare, e potete vederne l'uso nella prima regola (la variabile è preceduta da un simbolo del dollaro, e delimitata dalle parentesi). Le LIBS sono semplicemente le librerie che vi servono per compilare il programma. OBJECTS è qualsiasi file-oggetto necessario, in questo caso è soltanto quello che ha lo stesso nome del programma ed ha un .o alla fine, che è quanto è stato dichiarato.

La seconda regola si applica a qualsiasi obbiettivo terminante con .o (%.o), e richiede il file con lo stesso nome ma col suffisso .s (%.s esprime una dipendenza). La variabile automatica \$@ corrisponde al nome dell'obbiettivo e \$< corrisponde alla prima dipendenza. A questo punto dovrete riuscire ad arrivare all'obbiettivo finale.

Se volete adattare questo Makefile ad un altro programma Cobol, tutto quel che dovette fare è una variazione della variabile \$PROGRAM.

6. Fine

Questo è tutto, per ora. Qui ci sarebbe lo spazio per un altro capitolo che tratti di un programma composto da più di un file e che utilizzi il preprocessore, ma ciò va oltre le mie attuali conoscenze. Forse ci andrebbe anche qualche link a risorse sul Cobol.

Jonathan Riddell, jr@jridell.org

