

Richiamo compilatore FORTRAN 77

Table of Contents

<u>Richiamo compilatore FORTRAN 77</u>	1
<u>Indice dei contenuti</u>	1
<u>Generalità sul compilatore f77</u>	1
<u>Nomi di file (filename)</u>	1
<u>Opzioni</u>	2
<u>Note per utenti OpenVMS</u>	3

Richiamo compilatore FORTRAN 77

In questa pagina vengono date delle note essenziali sul richiamo del compilatore FORTRAN 77 sotto Digital Unix.

Maggiori informazioni si possono ottenere consultando lo help on-line di Unix con il comando:

```
k$ man f77
```

Con il comando:

```
k$ man f77 | col -b > nome_file
```

si ottiene nel file *nome_file* il contenuto dell'help on-line in formato editabile e stampabile.

Indice dei contenuti

1. [Generalità sul compilatore f77](#)
2. [Nomi di file \(filename\)](#)
3. [Opzioni](#)
4. [Note per utenti OpenVMS](#)
 - ◆ [Compilazione, link, esecuzione](#)
 - ◆ [Uso di librerie](#)
 - ◆ [Avvertenze sulla gestione dei file dati](#)
 - ◆ [Opzione -vms](#)

Generalità sul compilatore f77

La sintassi del comando per richiamare il compilatore FORTRAN 77 in Digital Unix è:

```
k$ f77 [ options [argument] ] filename [ options ]
```

Supponendo di voler compilare i sorgenti *gauss.f*, contenente il main program, e *sub.f*, contenente delle subroutine utilizzate dal main program, il comando è:

```
k$ f77 gauss.f sub.f
```

che produce, nella directory corrente, un file eseguibile denominato (di default) *a.out*.

E' lo stesso compilatore che provvede infatti a richiamare il loader (link editor), passando il file oggetto intermedio e i nomi delle librerie di sistema da utilizzare in fase di linking. Al termine il file oggetto viene automaticamente cancellato.

Nomi di file (filename)

Il comando *f77* accetta come nomi di file per i sorgenti FORTRAN 77 (o 66) i nomi che terminano con i suffissi ".f", ".for", ".FOR". Ad esempio sono nomi corretti: *gauss.f*, *gauss.solver.f*, *gauss.for*, *GAUSS.FOR*, *GAUSS.f*.

Il suffisso standard in Unix per i sorgenti Fortran è però ".f": si consiglia quindi di utilizzare solo questo suffisso.

Il compilatore per denominare i file che crea sostituisce il suffisso dei file sorgente. Ad esempio i file oggetto hanno suffisso ".o", i file contenenti il listato del programma ".l".

Opzioni

Le opzioni del comando `f77` per lo più precedono il filename, ma per alcune è obbligatorio citarle dopo il filename.

-o

specifica il nome del file eseguibile:

```
k$ f77 -o prova prova.f
```

produce il file eseguibile di nome *prova*. Per eseguire il programma basta citare il nome del file eseguibile:

```
k$ prova
```

-c

elimina la fase di loading (linking): viene prodotto un file oggetto con suffisso ".o":

```
k$ f77 -c prova.f
```

produce un file *prova.o*.

Per richiamare il loader (*link editor*) si utilizza il comando *ld*:

```
k$ ld [ option ... ] file ...
k$ ld prova.o
```

Si sconsiglia di utilizzare questa forma "in due passi" di compilazione e link: il compilatore quando richiama il loader passa automaticamente i nomi delle librerie di sistema da richiamare, mentre utilizzando direttamente il loader *ld* le stesse librerie vanno citate esplicitamente.

-Ldir

Indica al loader (linker) di ricercare nella directory *dir* le librerie specificate nell'opzione *-l*.

-lstring

indica al link editor di utilizzare la libreria di nome *libstring.a*. La libreria viene cercata nelle directory di default per le librerie di sistema nel caso non si sia utilizzata l'opzione *-L*, oppure nella directory indicata esplicitamente nell'opzione *-L*.

L'opzione *-l* deve seguire il filename.

Il comando:

```
k$ f77 gauss.f -lnag
```

utilizza la libreria *libnag.a* installata nell'opportuna directory di sistema, mentre il comando:

```
k$ f77 -L$HOME/librerie gauss.f -lmatrici
```

indica al loader di richiamare la libreria *libmatrici.a*, contenuta nella subdirectory *librerie* della home directory dell'utente.

Il loader cerca di risolvere gli external indefiniti del programma prima scandendo le entry della libreria *libmatrici.a*, e, successivamente, quelle delle librerie standard.

-g

produce un eseguibile che può venir esaminato utilizzando il debugger *dbx*. Questa opzione disabilita tutte le ottimizzazioni effettuate di default dal compilatore.

-C oppure -check_bounds

genera un eseguibile con del codice aggiuntivo, che permette il controllo a run-time della congruità fra il dimensionamento di vettori e stringhe e i riferimenti agli stessi. Se in fase di esecuzione gli

indici eccedono il dimensionamento viene visualizzata una segnalazione d'errore con l'indicazione della variabile coinvolta. Di default questa opzione non è attiva.

–*V*

crea un file di suffisso ".l" contenente il listato del programma.

–*cross_reference*

usata assieme all'opzione –*V* genera la cross reference delle variabili usate nel programma:

```
k$ f77 -V -cross_reference gauss.f
```

produce un file *gauss.l* con il listato del programma e la cross reference.

–*sdt*

in fase di compilazione segnala con dei "warning" le estensioni semantiche e sintattiche rispetto allo standard FORTRAN 77 ANSI utilizzate nel programma.

Questa opzione ha lo stesso effetto dell'opzione –*stand* con argomenti *semantic* e *syntax*.

L'opzione –*stand* prevede anche un argomento *source_form*, che segnala l'uso di <TAB> o di caratteri minuscoli nel sorgente.

–*vms*

produce un eseguibile che simula (parzialmente) a run-time il comportamento che avrebbe su un sistema OpenVMS.

Note per utenti OpenVMS

In queste note sono raccolte indicazioni e avvertenze da seguire nel porting di programmi FORTRAN 77 da OpenVMS a (Digital) Unix.

Compilazione, link, esecuzione

La sequenza di comandi tipica in OpenVMS per compilare, linkare ed eseguire un programma FORTRAN 77 è:

```
VMS_$ FORTRAN GAUSS[.FOR]
VMS_$ LINK GAUSS[.OBJ]
VMS_$ RUN GAUSS[.EXE]
```

dove fra parentesi quadre sono indicate le estensioni di default rispettivamente del file sorgente, oggetto ed eseguibile. La sequenza di comandi equivalente in (Digital) Unix è:

```
k$ f77 -o gauss gauss.f
k$ gauss
```

E' lo stesso compilatore che richiama infatti il loader (link editor). Si noti che va specificata l'opzione –*o* seguita dal nome del file eseguibile, altrimenti Unix denomina automaticamente *a.out* tutti i file eseguibili. Per eseguire il programma basta citare il nome del file eseguibile.

In Unix non esiste il concetto di "estensione" di un file (distinta dal "nome" del file) tipico dell'OpenVMS, ma il compilatore richiede che il nome del file sorgente termini con il suffisso ".f".

In Digital Unix sono accettati anche i suffissi ".for" e ".FOR", il che permette di spostare sorgenti Fortran da OpenVMS (ove hanno di default l'estensione .FOR) e poterli compilare senza rinominarli.

Per i sorgenti di nuova creazione si consiglia di utilizzare sempre il suffisso standard ".f".

Uso di librerie

Per "linkare" le subroutine contenute in una libreria installata la sintassi OpenVMS è:

```
VMS_$ FORTRAN GAUSS
VMS_$ LINK GAUSS, NAG/LIBRARY
VMS_$ RUN GAUSS
```

La sequenza di comandi equivalente in (Digital) Unix è:

```
k$ f77 -o gauss gauss.f -lnag
k$ gauss
```

Nel caso di librerie prodotte dall'utente (ROSSI), ad esempio si voglia "linkare" la libreria *MATRICI.OLB* che risiede nella directory *[ROSSI.LIBRERIE]*, la sintassi OpenVMS è:

```
VMS_$ FORTRAN GAUSS
VMS_$ LINK GAUSS, [ROSSI.LIBRERIE]MATRICI/LIBRARY
VMS_$ RUN GAUSS
```

La sequenza di comandi equivalente in (Digital) Unix (presupponendo che la libreria sia denominata *libmatrici.a* e stia nella subdirectory *librerie* della home directory utente) è:

```
k$ f77 -o gauss -L$HOME/librerie gauss.f -lmatrici
k$ gauss
```

Si noti che il nome della libreria va citato nell'opzione *-l* senza il prefisso "*lib*" e il suffisso "*.a*".

Avvertenze sulla gestione dei file dati

Va posta particolare attenzione al fatto che Unix, a differenza di OpenVMS, è "case sensitive".

I nomi *DATI.DAT* e *dati.dat*, che in OpenVMS designano lo stesso file, in Unix designano due file diversi. Se lo stesso file dati viene utilizzato da più programmi (o da più istruzioni OPEN nello stesso programma), si deve porre attenzione nell'indicare correttamente il nome file.

Ad esempio, ipotizzando un programma che scriva un file dati e un programma di lettura dello stesso file, in OpenVMS è possibile scrivere:

```
PROGRAM SCRIVI
OPEN (logical_unit, FILE='DATI.DAT', STATUS='NEW' ...)
...
END

PROGRAM LEGGI
OPEN (logical_unit, FILE='dati.dat', STATUS='NEW' ...)
...
END
```

ed ottenere una corretta esecuzione (infatti il nome file *dati.dat* citato nel programma LEGGI è tradotto dal sistema operativo in *DATI.DAT*).

Gli stessi programmi portati su Unix non funzionano più correttamente: il programma SCRIVI crea infatti un file *DATI.DAT*, mentre il programma LEGGI cerca un file *dati.dat*, che ovviamente non esiste, terminando così con un errore (File not found).

Va anche tenuto conto che in Unix non esiste il concetto di versione dello stesso file. Questo può portare lo stesso programma ad avere comportamenti diversi nei due ambienti, in particolare utilizzando il parametro

'NEW' in una istruzione OPEN.

Si supponga che *dati.dat* sia un file già esistente. In OpenVMS uno statement del tipo:

```
OPEN (logical_unit, FILE='dati.dat', STATUS='NEW' ...)
```

è legittimo. Ha infatti l'effetto di creare una nuova versione del file.

In Unix lo stesso statement, sempre nell'ipotesi che file indicato già esista, provoca invece una situazione di errore (Cannot overwrite existing file).

Opzione *-vms*

Il compilatore f77 su Digital Unix mette a disposizione una opzione *-vms* che permette di avere nell'esecuzione del programma un comportamento che simula per certi aspetti quello su un sistema OpenVMS.

Gli aspetti di gestione dei file sopra indicati non vengono però risolti attraverso questa opzione.