IEEE P1003.2 Draft 11.2 – September 1991

Copyright © 1991 by the
Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, NY 10017, USA
All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change. The publication, distribution, or copying of this draft, as well as all derivative works based on this draft, is expressly prohibited except as set forth below.

Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities only, and subject to the restrictions contained herein.

Permission is hereby also granted for member bodies and technical committees of ISO and IEC to reproduce this document for purposes of developing a national position, subject to the restrictions contained herein.

Permission is hereby also granted to the preceding entities to make limited copies of this document in an electronic form only for the stated activities.

The following restrictions apply to reproducing or transmitting the document in any form: 1) all copies or portions thereof must identify the document's IEEE project number and draft number, and must be accompanied by this entire notice in a prominent location; 2) no portion of this document may be redistributed in any modified or abridged form without the prior approval of the IEEE Standards Department.

Other entities seeking permission to reproduce this document, or any portion thereof, for standardization or other activities, must contact the IEEE Standards Department for the appropriate license.

Use of information contained in this unapproved draft is at your own risk.

IEEE Standards Department
Copyright and Permissions
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331, USA
+1 (908) 562-3800
+1 (908) 562-1571 [FAX]

C.2 `fort77` — FORTRAN compiler

This utility is optional. It shall be provided on systems that support the FORTRAN Development Utilities Option.

C.2.1 Synopsis

`fort77 [-c][-g][-L` *directory* `]... [-O` *optlevel* `][-o` *outfile* `][-s][-w]`
operand ...

C.2.2 Description

The `fort77` utility is the interface to the FORTRAN compilation system; it shall accept the full FORTRAN language defined by ISO 1539 {2}. The system conceptually consists of a compiler and link editor. The files referenced by *operand* s are compiled and linked to produce an executable file. (It is unspecified whether the linking occurs entirely within the operation of `fort77`; some systems may produce objects that are not fully resolved until the file is executed.)

If the `-c` option is present, for all pathname operands of the form *file* `.f`, the files

`$(basename` *pathname* `.f).o`

shall be created or overwritten as the result of successful compilation. If the `-c` option is not specified, it is unspecified whether such `.o` files are created or deleted for the *file* `.f` operands.

If there are no options that prevent link editing (such as `-c`) and all operands compile and link without error, the resulting executable file shall be written into the file named by the `-o` option (if present) or to the file `a.out`. The executable file shall be created as specified in 2.9.1.4, except that the file permissions shall be set to

S_IRWXO | S_IRWXG | S_IRWXU

(see POSIX.1 {8} 5.6.1.2) and that the bits specified by the *umask* of the process shall be cleared.

C.2.3 Options

The `fort77` utility shall conform to the utility argument syntax guidelines described in 2.10.2, except that:

— The `-l` *library* operands have the format of options, but their position within a list of operands affects the order in which libraries are searched.

— The order of specifying the multiple `-L` options is significant.

— Conforming applications shall specify each option separately; that is, grouping option letters (e.g., `-cg`) need not be recognized by all implementations.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

The following options shall be supported by the implementation:

`-c` Suppress the link-edit phase of the compilation, and do not remove any object files that are produced.

`-g` Produce symbolic information in the object or executable files; the nature of this information is unspecified, and may be modified by implementation-defined interactions with other options.

`-s` Produce object and/or executable files from which symbolic and other information not required for proper execution using the POSIX .1 {8} *exec* family has been removed (stripped). If both `-g` and `-s` options are present, the action taken is unspecified.

`-o` *outfile* Use the pathname *outfile* , instead of the default `a.out`, for the executable file produced. If the `-o` option is present with `-c`, the result is unspecified.

`-L` *directory*
Change the algorithm of searching for the libraries named in `-l` operands to look in the directory named by the *directory* pathname before looking in the usual places. Directories named in `-L` options shall be searched in the specified order. Implementations shall support at least ten instances of this option in a single `fort77` command invocation. If a directory specified by a `-L` option contains a file named `libf.a`, the results are unspecified.

`-O` *optlevel* Specify the level of code optimization. If the *optlevel* option-argument is the digit `0`, all special code optimizations shall be disabled. If it is the digit `1`, the nature of the optimization is unspecified. If the `-O` option is omitted, the nature of the system's default optimization is unspecified. It is unspecified whether code generated in the presence of the `-O 0` option is the same as that generated when `-O` is omitted. Other *optlevel* values may be supported.

`-w` Suppress warnings.

Multiple instances of `-L` options can be specified.

C.2.4 Operands

An *operand* is either in the form of a pathname or the form `-l` *library* . At least one operand of the pathname form shall be specified. The following operands shall be supported by the implementation:

file.`f` The pathname of a FORTRAN source file to be compiled and optionally passed to the link editor. The file name operand shall be of this form if the `-c` option is used.

file.`a` A library of object files typically produced by `ar` (see 6.1), and passed directly to the link editor. Implementations may recognize implementation-defined suffixes other than `.a` as denoting

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

192 object file libraries.

193 *file*.o An object file produced by `fort77 -c`, and passed directly to the
194 link editor. Implementations may recognize implementation-
195 defined suffixes other than `.o` as denoting object files.

196 The processing of other files is implementation defined.

197 `-l` *library* (The letter ell.) Search the library named:

198 `lib`*library*`.a`

199 A library is searched when its name is encountered, so the place-
200 ment of a `-l` operand is significant. Several standard libraries
201 can be specified in this manner, as described in C.2.7. Implemen-
202 tations may recognize implementation-defined suffixes other than
203 `.a` as denoting libraries.

204 ### C.2.5 External Influences

205 #### C.2.5.1 Standard Input

206 None.

207 #### C.2.5.2 Input Files

208 The input file shall be one of the following: a text file containing FORTRAN source
209 code; an object file in the format produced by `fort77 -c`; or a library of object
210 files, in the format produced by archiving zero or more object files, using `ar`.
211 Implementations may supply additional utilities that produce files in these for-
212 mats. Additional input files are implementation defined.

213 A <tab> character encountered within the first six characters on a line of source
214 code shall cause the compiler to interpret the following character as if it were the
215 seventh character on the line (i.e., in column 7).

216 #### C.2.5.3 Environment Variables

217 The following environment variables shall affect the execution of `fort77`:

218 **LANG** This variable shall determine the locale to use for the
219 locale categories when both **LC_ALL** and the correspond-
220 ing environment variable (beginning with **LC_**) do not
221 specify a locale. See 2.6.

222 **LC_ALL** This variable shall determine the locale to be used to over-
223 ride any values for locale categories specified by the set-
224 tings of **LANG** or any environment variables beginning
225 with **LC_**.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

226 **LC_CTYPE** This variable shall determine the locale for the interpreta-
227 tion of sequences of bytes of text data as characters (e.g.,
228 single- versus multibyte characters in arguments and
229 input files).

230 **LC_MESSAGES** This variable shall determine the language in which mes-
231 sages should be written.

232 **TMPDIR** This variable shall be interpreted as a pathname that
233 should override the default directory for temporary files, if
234 any.

235 ### C.2.5.4 Asynchronous Events

236 Default.

237 ## C.2.6 External Effects

238 ### C.2.6.1 Standard Output

239 None.

240 ### C.2.6.2 Standard Error

241 Used only for diagnostic messages. If more than one file operand ending in `.f` (or
242 possibly other unspecified suffixes) is given, for each such file:

243 `"%s:\n"`, <*file*>

244 may be written to allow identification of the diagnostic message with the
245 appropriate input file.

246 This utility may produce warning messages about certain conditions that do not
247 warrant returning an error (nonzero) exit value.

248 ### C.2.6.3 Output Files

249 Object files, listing files, and/or executable files shall be produced in unspecified
250 formats.

251 ## C.2.7 Extended Description

252 ### C.2.7.1 Standard Libraries

253 The `fort77` utility shall recognize the following `-l` operand for the standard
254 library:

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

255 `-l f` This library contains all library functions referenced in ISO 1539
256 {2}. An implementation shall not require this operand to be
257 present to cause a search of this library.

258 In the absence of options that inhibit invocation of the link editor, such as `-c`, the
259 `fort77` utility shall cause the equivalent of a `-l f` operand to be passed to the
260 link editor as the last `-l` operand, causing it to be searched after all other object
261 files and libraries are loaded.

262 It is unspecified whether the library `libf.a` exists as a regular file. The imple-
263 mentation may accept as `-l` operands names of objects that do not exist as regu-
264 lar files.

265 ### C.2.7.2 External Symbols

266 The FORTRAN compiler and link editor shall support the significance of external 1
267 symbols up to a length of at least 31 bytes. The compiler may fold case (i.e., may 1
268 ignore uppercase/lowercase distinctions between identifiers). The action taken 1
269 upon encountering symbols exceeding the implementation-defined maximum sym-
270 bol length is unspecified.

271 The compiler and link editor shall support a minimum of 511 external symbols
272 per source or object file, and a minimum of 4095 external symbols total. A diag-
273 nostic message is written to standard output if the implementation-defined limit
274 is exceeded; other actions are unspecified.

275 ### C.2.8 Exit Status

276 The `fort77` utility shall exit with one of the following values:

277 0 Successful compilation or link edit.

278 >0 An error occurred.

279 ### C.2.9 Consequences of Errors

280 When `fort77` encounters a compilation error, it shall write a diagnostic to stan-
281 dard error and continue to compile other source code operands. It shall return a
282 nonzero exit status, but it is implementation defined whether an object module is
283 created. If the link edit is unsuccessful, a diagnostic message shall be written to
284 standard error, and `fort77` shall exit with a nonzero status.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

C.2.10 Rationale. *(This subclause is not a part of P1003.2)*

Examples, Usage

The following are examples of usage:

`fort77 -o foo xyz.f`	Compiles `xyz.f` and creates the executable `foo`.
`fort77 -c xyz.f`	Compiles `xyz.f` and creates the object file `xyz.o`.
`fort77 xyz.f`	Compiles `xyz.f` and creates the executable `a.out`.
`fort77 xyz.f b.o`	Compiles `xyz.f`, links it with `b.o`, and creates the executable `a.out`.

History of Decisions Made

The file inclusion and symbol definition (`#define`) mechanisms used by the `c89` utility were not included in POSIX.2—even though they are commonly implemented—since there is no requirement that the FORTRAN compiler use the C preprocessor.

The `–onetrip` option was not included in this specification, even though many historical compilers support it, because it is a relic from FORTRAN-66; it is an anachronism that should not be perpetuated.

Some implementations produce compilation listings. This aspect of FORTRAN has been left unspecified because there was opposition within the balloting group to the various methods proposed for implementing it: a `–V` option overlapped with historical vendor practice and a naming convention of creating files with `.l` suffixes collided with historical `lex` file naming practice.

There is no `–I` option in this version of POSIX.2 to specify a directory for file inclusion. An INCLUDE directive has been a part of the FORTRAN-8X discussions, but it is not clear whether it will be retained.

It is noted that many FORTRAN compilers produce an object module even when compilation errors occur; during a subsequent compilation, the compiler may patch the object module rather than recompiling all the code. Consequently, it is left to the implementor whether or not an object file is created.

The name of this utility was changed to `fort77` in Draft 9 to parallel the renaming of the C compiler. The name `f77` was not chosen to avoid collision with historical implementations.

A reference to MIL-STD-1753 was removed from an earlier draft in response to a request from the POSIX.9 working group. It was not the intention of this document to require certification of the FORTRAN compiler and the forthcoming POSIX.9 standard does not specify the military standard or any special preprocessing requirements. Furthermore, use of that document would have been inappropriate for an international standard.

The specification of optimization has been subject to changes through early drafts. At one time, `–O` and `–N` were Booleans: optimize and do not optimize (with an

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

unspecified default). Some historical practice lead this to be changed to:

`-O 0` No optimization.

`-O 1` Some level of optimization.

`-O` *n* Other, unspecified levels of optimization.

It is not always clear whether ''good code generation'' is the same thing as optimization. Simple optimizations of local actions do not usually affect the semantics of a program. The `-O 0` option has been included to accommodate the very fussy nature of scientific calculations in a highly optimized environment; compilers make errors. Some degree of optimization is expected, even if it is not documented here, and the ability to shut it off completely could be important when porting an application. An implementation may treat `-O 0` as ''do less than normal'' if it wishes, but this is only meaningful if any of the operations it performs can affect the semantics of a program. It is highly dependent on the implementation whether doing less than normal makes sense. It is not the intent of this to ask for sloppy code generation, but rather to assure that any semantically visible optimization is suppressed.

The specification of standard library access is consistent with the C compiler specification. Implementations are not required to have `/usr/lib/libf.a`, as many historical implementations do, but if not they are required to recognize `'f'` as a token.

External symbol size limits are in a normative subclause; portable applications need to know these limits. However, the minimum maximum symbol length should be taken as a constraint on a portable application, not on an implementation, and consequently the action taken for a symbol exceeding the limit is unspecified. The minimum size for the external symbol table was added for similar reasons.

The Consequences of Errors subclause clearly specifies the compiler's behavior when compilation or link-edit error occur. The behavior of several historical implementations was examined, and the choice was made to be silent on the status of the executable, or `a.out`, file in the face of compiler or linker errors. If a linker writes the executable file, then links it on disk with *lseek* ()s and *write* ()s, the partially-linked executable can be left on disk and its execute bits turned off if the link edit fails. However, if the linker links the image in memory before writing the file to disk, it need not touch the executable file (if it already exists) because the link edit fails. Since both approaches are existing practice, a portable application shall rely on the exit status of `fort77`, rather than on the existence or mode of the executable file.

The `−g` and `−s` options are not specified as mutually exclusive. Historically these two options have been mutually exclusive, but because both are so loosely specified, it seemed cleaner to leave their interaction unspecified.

The requirement that portable applications specify compiler options separately is to reserve the multicharacter option namespace for vendor-specific compiler options, which are known to exist in many historical implementations. Implementations are not required to recognize, for example, `−gc` as if it were `−g −c`; nor

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

are they forbidden from doing so. The synopsis shows all of the options separately to highlight this requirement on applications.

Echoing filenames to standard error is considered a diagnostic message, because it would otherwise difficult to associate an error message with the erring file. They are describing with ‘‘may’’to allow implementations to use other methods of identifying files and to parallel the description in `c89`.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

Annex D
(inform ative)

Bibliography

{B1} ISO 639: 1988, *Code for the representation of names of languages.* [1)]

{B2} ISO 2022: 1986, *Information processing— ISO 7-bit and 8-bit coded character sets—Code extension techniques.*

{B3} ISO 2047: 1975, *Information processing—Graphical representations for the control characters of the 7-bit coded character set.*

{B4} ISO 3166: 1988, *Code for the representation of names of countries.*

{B5} ISO 6429: 1988, *Information processing—Control functions for 7-bit and 8-bit coded character sets.*

{B6} ISO 6937-2: 1983, *Information processing—Coded character sets for text communication—Part 2: Latin alphabetic and non-alphabetic graphic characters.*

{B7} ISO 8802-3: 1989, *Information processing systems—Local area networks—Part 3: Carrier sense multiple access with collision detection (CSMA / CD) access method and physical layer specification.*

{B8} ISO 8806: 1988, *Data elements and interchange formats—Information interchange —Representation of dates and times.*

{B9} ISO 8859, *Information processing—8-bit single-byte coded graphic character sets.* (Parts 1 to 8 published.)

{B10} ISO/IEC 10367: . . . ,[2)] *Information processing—Repertoire of standardized coded graphic character sets for use in 8-bit codes.*

{B11} ISO/IEC 10646: . . . ,[3)] *Information technology—Universal Coded Character Set (UCS).*

---

1) ISO documents can be obtained from the ISO office, 1, rue de Varembe´, Case Postale 56, CH-1211, Genè ve 20, Switzerland/Suisse.

2) To be approved and published.

3) To be approved and published.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.