NAME

dcl2inc – postprocess ftnchek .dcl files to create separate INCLUDE files

SYNOPSIS

dcl2inc **.dcl*

DESCRIPTION

dcl2inc postprocessing declaration files output by **ftnchek**(1), replacing unique COMMON block definitions by Fortran *INCLUDE* statements. For each input *.dcl* file, a modified output *.dcn* file is produced, together with include files named by the COMMON block name, with filename extension *.inc*.

In addition, **dcl2inc** produces on *stdout* a list of *Makefile* dependencies for the UNIX **make**(1) utility. These can be appended to the project *Makefile* to ensure that any subsequent changes to *.inc* files provoke recompilation of source files that include them.

dcl2inc warns about COMMONs which differ from their first occurrence, and simply copies them to the output *.dcn* file, instead of replacing them with an INCLUDE statement. Thus, any COMMON statements that are found in the output *.dcn* files should be examined carefully to determine why they differ: they may well be in error.

Replication of identical data, and bugs arising from subsequent modification of only part of it, is a significant reason why Fortran programming projects should *require* that COMMON declarations occur in separate include files, so that there is only a *single* point of definition of any global object.

Even though the Fortran *INCLUDE* statement was tragically omitted from the 1977 Standard, it has long been implemented by virtually all compiler vendors, and is part of the 1990 Standard. In practice, there is therefore no portability problem associated with use of INCLUDE statements, *provided* that one avoids nonportable file names. As long as the code obeys Fortran's limit of six-character alphanumeric names, the filenames generated by **dcl2inc** will be acceptable on all current popular operating systems.

Fortran's default, or IMPLICIT, variable typing is deprecated in modern programming languages, because it encourages sloppy documentation, and worse, bugs due to misspelled variables, or variables that have been truncated because they extend past column 72. If all variables used are explicitly typed, and a compiler option is used to reject all program units with untyped variables, variable spelling and truncation errors can be eliminated.

Variable declarations that have been produced automatically by a tool like **ftnchek**(1) or **pfort**(1) have a consistent format that facilitates application of stream editors (e.g. to change array dimensions or rename variables), and simple floating-point precision conversion tools like **d2s**(1), **dtoq**(1), **dtos**(1), **qtod**(1), **s2d**(1), and **stod**(1).

CAVEAT

The current version (2.9) of **ftnchek**(1) does not produce Fortran EQUIVALENCE statements in *.dcl* files, so you must be careful to preserve them when replacing original declarations with new ones from *.dcl* or *.dcn* files.

SEE ALSO

d2s(1), **dtoq**(1), **dtos**(1), **ftnchek**(1), **make**(1), **pfort**(1), **qtod**(1), **s2d**(1), **stod**(1).

AUTHOR

Nelson H. F. Beebe, Ph.D.
Center for Scientific Computing
Department of Mathematics
University of Utah
Salt Lake City, UT 84112
Tel: +1 801 581 5254
FAX: +1 801 581 4148
Email: <beebe@math.utah.edu>