G95 Manual

Contents

SYNOPSIS

```
 g95 [ -c | -S | -E ] compile & assemble | produce assembly code | list source
      [-g] [-pg]               debug options
      [-Olevel]                Optimisation level
      [-s ]                    strip
      [-Wwarn...] [-pedantic]  Warning switches
      [-Idir...]               Include directory to search
      [-Ldir...]               Library directory to search
      [-Dmacro[=defn]...]      Define macro
      [-Umacro]                Undefine macro
      [-foption...]
      [-mmachine-option...]
      [-o outfile]             name of outfile
      infile...
```

G95 Options

```
Usage: g95 [options] file...
Options:
 -pass-exit-codes             Exit with highest error code from a phase
 --help                       Display this information
 --target-help                Display target specific command line options
                              (Use '-v --help' to display command line options
                              of sub-processes)
 -dumpspecs                   Display all of the built in spec strings
 -dumpversion                 Display the version of the compiler
 -dumpmachine                 Display the compiler's target processor
 -print-search-dirs           Display the directories in the compiler's search
                              path
 -print-libgcc-file-name      Display the name of the compiler's companion
                              library
 -print-file-name=<lib>       Display the full path to library <lib>
 -print-prog-name=<prog>      Display the full path to compiler component
                              <prog>
 -print-multi-directory       Display the root directory for versions of
```

```
                                libgcc
 -print-multi-lib               Display the mapping between command line options
                                and multiple library search directories
 -print-multi-os-directory      Display the relative path to OS libraries
 -Wa,<options>                  Pass comma-separated <options> on to the
                                assembler
 -Wp,<options>                  Pass comma-separated <options> on to the
                                preprocessor
 -Wl,<options>                  Pass comma-separated <options> on to the linker
 -Xassembler <arg>              Pass <arg> on to the assembler
 -Xpreprocessor <arg>           Pass <arg> on to the preprocessor
 -Xlinker <arg>                 Pass <arg> on to the linker
 -combine                       Pass multiple source files to compiler at once
 -save-temps                    Do not delete intermediate files
 -pipe                          Use pipes rather than intermediate files
 -time                          Time the execution of each subprocess
 -specs=<file>                  Override built-in specs with the contents of
                                <file>
 -std=<standard>                Assume that the input sources are for <standard>
 -B <directory>                 Add <directory> to the compiler's search paths
 -b <machine>                   Run gcc for target <machine>, if installed
 -V <version>                   Run gcc version number <version>, if installed
 -v                             Display the programs invoked by the compiler
 -###                           Like -v but options quoted and commands not
                                executed
 -E                             Preprocess only; do not compile, assemble or
                                link
 -S                             Compile only; do not assemble or link
 -c                             Compile and assemble, but do not link
 -o <file>                      Place the output into <file>
 -x <language>                  Specify the language of the following input
                                files. Permissible languages include: c c++
                                assembler none- 'none' means revert to the
                                default behavior of guessing the language based
                                on the file's extension

Options starting with -g, -f, -m, -O, -W, or --param are automatically passed
on to the various sub-processes invoked by g95.  In order to pass other
options on to these processes the -W<letter> options must be used.

Command line arguments:
Any program compiled with g95 can be executed with these arguments:
 --help                 Print this list
 --resume <corefile>    Resume program execution from a core file
```

Preprocessor Options

```
-cpp                    Force the input files to be run through the C
                        preprocessor
-no-cpp                 Prevent the input files from being C preprocessed
-D<macro=>              Define a preprocessor macro
-U<macro>               Undefine a preprocessor macro
-E                      Show preprocessed source only
-M                      Write dependencies in Makefile form
```

Options Controlling Fortran Dialect

-d8	Set the default real and integer kinds to double precision
-i8	Set kinds of integers without kind specifications to double default precision
-r8	Set kinds of reals without kind specifications to double default precision
-fcase-upper	Make all public symbols uppercase
-fbackslash	Interpret backslashes in character constants as escape code. Use -fno-backslash to treat backslashes literally.
-fdollar-ok	Allow dollar signs in entity names
-ffixed-form	Assume that the source file is fixed form
-ffixed-line-length-80	80 character line width in fixed mode
-ffixed-line-length-132	132 character line width in fixed mode
-ffree-form	Assume that the source file is free form
-fimplicit-none	Specify that no implicit typing is allowed, unless overridden by explicit IMPLICIT statements
-fmodule-private	Set default accessibility of module entities to PRIVATE
-fonetrip	Force DO-loops to execute at least once (buggy FORTRAN 66)
-fpack-derived	Try to layout derived types as compact as possible
-fqkind=<n>	Set the kind for a real with the 'q' exponent to 'n'
-fstatic	Put local variables in static memory where possible.
-max-frame-size=<n>	How large a single stack frame will get before arrays are allocated dynamically
-fsloppy-char	Prevent type checks when printing formatted characters variables.
-std=f95	Strict fortran 95 checking
-std=f2003	Strict fortran 2003 checking
-std=F	Check for non-F features and warn

Directory Options

-I<directory>	Append 'directory' to the include and module files search path
-L<directory>	Append 'directory' to the library search path
-fmod=<directory>	Put module files in 'directory'

Warning Options

Warnings are diagnostic messages that report constructions which are not inherently erroneous but which are risky or suggest there might have been an error. You can request many specific warnings with options beginning -W. Each of these specific warning options also has a negative form beginning -Wno- to turn off warnings. This manual lists only one of the two forms, whichever is not the default. These options control the amount and kinds of warnings produced by g95:

-Wall	Enable most warning messages
-Wno=<n1,n2,,>	Disable warnings (comma separated list of warning numbers).
-Wimplicit-none	Same as -fimplicit-none
-Wline-truncation	Warn about truncated source lines
-Wprecision-loss	Warn about precision loss in implicit type conversions

```
-Wunused-label          Warn when a label is unused
-Wunused-module-vars    Warn about unused module variables. Used to build ONLY
                        clauses.
-Wunused-vars           Warn about unused variables
-Wunset-vars            Warn about unset variables
```

Code Generation Options

```
-fbounds-check          Check array bounds at runtime
-fleading-underscore    Add a leading underscore to public names
-funderscoring          Append a trailing underscore in global names
                        (default). Use -fno-underscoring to suppress.
-fsecond-underscore     Append a second trailing underscore in names having an
                        underscore (default). Use -fno-second-underscore to
                        suppress.
```

Environment Variables

The g95 runtime environment provides many options for tweaking the behaviour of your program once it runs. These are controllable through environment variables. Running a g95-compiled program with the --help option will dump all of these options to standard output.

The values of the various variables are always strings, but the strings can be interpreted as integers or boolean truth values. Only the first character of a boolean is examined and must be 't', 'f', 'y', 'n', '1' or '0' (uppercase OK too). If a value is bad, no error is issued and the default is used.

```
G95_STDIN_UNIT    Integer     Default: 5
                  Unit number that will be preconnected to standard input (No
                  preconnection if negative)

G95_STDOUT_UNIT   Integer     Default: 6
                  Unit number that will be preconnected to standard output
                  (No preconnection if negative)

G95_STDERR_UNIT   Integer     Default: 0
                  Unit number that will be preconnected to standard error
                  (No preconnection if negative)

G95_USE_STDERR    Boolean     Default: Yes
                  Sends library output to standard error instead of standard
                  output.

G95_ENDIAN        String      Default: NATIVE
                  Endian format to use for I/O of unformatted data. Values are
                  BIG, LITTLE or NATIVE. Default is NATIVE

G95_CR            Boolean     Default: Yes
                  Output carriage returns for formatted sequential records.
                  Default true on windows, false elsewhere.

G95_IGNORE_ENDFILE Boolean    Default: No
                  Ignore attempts to read past the ENDFILE record in
                  sequential access mode. Default false.
```

G95_TMPDIR String Default: ""
Directory for scratch files. Overrides the TMP environment variable. If TMP is not set /var/tmp is used.

G95_UNBUFFERED_ALL Boolean Default: No
If TRUE, all output is unbuffered. This will slow down large writes but can be useful for forcing data to be displayed immediately.

G95_SHOW_LOCUS Boolean Default: Yes
If TRUE, print filename and line number where runtime errors happen.

G95_OPTIONAL_PLUS Boolean Default: No
Print optional plus signs in numbers where permitted. Default FALSE.

G95_DEFAULT_RECL Integer Default: 500000000
Default maximum record length for sequential files. Most useful for adjusting line length of preconnected units. Default 500000000

G95_LIST_SEPARATOR String Default: " "
Separatator to use when writing list output. May contain any number of spaces and at most one comma. Default is a single space.

G95_EXPAND_UNPRINTABLE Boolean Default: No
For formatted output, print otherwise unprintable characters with \-sequences Default FALSE

G95_QUIET Boolean Default: No
Suppress bell characters (\a) in formatted output. Default FALSE.

G95_SYSTEM_CLOCK Integer Default: 100000
Number of ticks per second reported by the SYSTEM_CLOCK() intrinsic in microseconds. Zero disables the clock.

G95_SEED_RNG Boolean Default: No
If true, seeds the random number generator with a new seed when the program is run. Default FALSE.

G95_MINUS_ZERO Boolean Default: Yes
If true, allows minus zeros to be printed correctly, contrary to the standard. Default TRUE.

G95_MEM_INIT String Default
How to initialize ALLOCATEd memory. Default value is NONE for no initialization (faster), NAN for a Not-a-Number with the mantissa 0x40f95 or a custom hexadecimal value

G95_MEM_SEGMENTS Integer Default: 25
Maximum number of still-allocated memory segments to display when program ends. 0 means show none, less than 0 means show all. Default 25

G95_MEM_MAXALLOC Boolean Default: No
If true, shows the maximum number of bytes allocated in user memory during the program run.

G95_MEM_MXFAST Integer Default: 64
Maximum request size for handing requests in from fastbins. Fastbins are quicker but fragment more easily. Default 64 bytes

G95_MEM_TRIM_THRESHOLD Integer Default: 262144
Amount of top-most memory to keep around until it is returned to the system. -1 prevents returning memory to the system. Useful in long-lived programs.

G95_MEM_TOP_PAD Integer Default: 0
Extra space to allocate when getting memory from the OS. Can speed up future requests.

G95_SIGHUP String Default: ABORT
Whether the program will IGNORE, ABORT or SUSPEND on SIGHUP.

G95_SIGINT String Default: ABORT
Whether the program will IGNORE or ABORT or SUSPEND on SIGINT.

G95_FPU_ROUND String Default: NEAREST
Set floating point rounding. Values are NEAREST, UP, DOWN, ZERO.

G95_FPU_PRECISION String Default: (Unknown)
Precision of intermediate results. Value can be 24, 53 and 64. Default 64

G95_FPU_DENORMAL Boolean Default: No
Raise a floating point exception when denormal numbers are encountered.

G95_FPU_INVALID Boolean Default: No
Raise a floating point exception on an invalid operation.

G95_FPU_ZERODIV Boolean Default: No
Raise a floating point exception when dividing by zero.

G95_FPU_OVERFLOW Boolean Default: No
Raise a floating point exception on overflow.

G95_FPU_UNDERFLOW Boolean Default: No
Raise a floating point exception on underflow.

G95_FPU_INEXACT Boolean Default: No
Raise a floating point exception on precision loss.

G95_FPU_EXCEPTIONS Boolean Default: No
Whether masked floating point exceptions should be shown after the program ends.

G95_UNIT_x Default unit names

G95_UNBUFFERED_x Unit buffering overrides

Runtime Error Codes

Running a g95-compiled program with the --help option will dump this list of error codes to standard output

```
 -2   End of record
 -1   End of file
  0   Successful return
      Operating system errno codes (1 - 199)
200   Conflicting statement options
201   Bad statement option
202   Missing statement option
203   File already opened in another unit
204   Unattached unit
205   FORMAT error
206   Incorrect ACTION specified
207   Read past ENDFILE record
208   Bad value during read
209   Numeric overflow on read
210   Out of memory
211   Array already allocated
212   Deallocated a bad pointer
213   Bad record number in direct-access file
214   Corrupt record in unformatted sequential-access file
215   Reading more data than the record size (RECL)
216   Writing more data than the record size (RECL)
```

SEE ALSO:
For further information see the following man and info entries: gpl(7), gfdl(7), fsf-funding(7), cpp(1), gcov(1), gcc(1), as(1), ld(1), gdb(1), adb(1), dbx(1), sdb(1) and the Info entries for gcc, cpp, as, ld, binutils and gdb.

Fortran 2003 Features

G95 implements a few features of Fortran 2003. For a discussion of all the new features of Fortran 2003, see:
http://www.kcl.ac.uk/kis/support/cit//fortran/john_reid_new_2003.pdf

COMMAND_ARGUMENT_COUNT ()
An inquiry function that returns the number of command arguments as a default integer scalar.

CALL GET_COMMAND ([COMMAND,LENGTH,STATUS])
Returns the entire command by which the program was invoked.

CALL GET_COMMAND_ARGUMENT (NUMBER[,VALUE,LENGTH,STATUS])
Returns a command argument.

CALL GET_ENVIRONMENT_VARIABLE (NAME[,VALUE,LENGTH,STATUS,TRIM_NAME])
Obtains the value of an environment variable.

Real and double precision DO loop index variables are not implemented in g95.

Square brackets [...] may be used as an alternative to (/ ... /) for array constructors and delimiters.

TR 15581 - allocatable derived types. Allows the use of the ALLOCATABLE attribute on:

* dummy arguments
* function results
* structure components

G95 Extensions - Intrinsic Procedures

Abort	Float	Rename
ChDir	Flush	Signal
DCMPLX()	FStat	Sleep
DFLOAT()	g95_runtime_start	SRand
DREAL()	GetArg	Stat
Dtime	GetLog	System
ErF	GetPId	Time
ErFC	HostNm	Unlink
ETime	IsNaN	%Val & %Ref
Exit	LStat	
Fdate	Rand	

CALL Abort()
Prints a message and quits.

CALL ChDir(Dir, Status)
Dir: CHARACTER; scalar; INTENT(IN)
Status: INTEGER(KIND=1); OPTIONAL; scalar; INTENT(OUT)
Sets the current working directory to 'Dir'.

DCMPLX()
Double precision CMPLEX()

DFLOAT()
Double precision REAL()

DREAL()
Alias for DBLE()

Dtime(TArray)
TArray: REAL(KIND=1); DIMENSION(2); INTENT(OUT)
REAL(KIND=1) function. Returns the runtime in seconds since the start of the process, or since the last invocation.

ErF(X)
X: REAL; scalar; INTENT(IN)
REAL function, the `KIND=' value of the type being that of argument X. Returns the error function of X.

ErFC(X)
X: REAL; scalar; INTENT(IN)
REAL function, the `KIND=' value of the type being that of argument X. Returns the complementary error function of X: ERFC(R) = 1 - ERF(R).

```
Etime(TArray)
TArray: REAL(KIND=1); DIMENSION(2); INTENT(OUT)
REAL(KIND=1) function. Returns in seconds the time since the start of the
process' execution.

CALL Exit(Status)
Status: INTEGER; OPTIONAL; scalar; INTENT(IN)
Exit a program with status 'Status' after closing open Fortran i/o units.

Fdate()
CHARACTER*(LEN=*) function. Returns the current date and time as:
Day Mon dd hh:mm:ss yyyy

Float(A)
A: INTEGER; scalar; INTENT(IN)
REAL(KIND=1) function. Archaic form of `REAL()' that is specific to one type
for A.

CALL Flush(Unit)
Unit: INTEGER; OPTIONAL; scalar; INTENT(IN)
Flushes Fortran unit(s) currently open for output. Without the optional
argument, all such units are flushed, otherwise just the unit specified by
'Unit'.

FStat(Unit, Sarray)
Unit: INTEGER; scalar; INTENT(IN)
SArray: INTEGER(KIND=1); DIMENSION(13); INTENT(OUT)
INTEGER function. Obtains data about the file open on Fortran I/O unit 'Unit'
and places them in the array 'SArray'. The values in this array are extracted
from the stat structure as returned by fstat(2) q.v., as follows:

      1.    File mode
      2.    Inode number
      3.    ID of device containing directory entry for file
      4.    Device id (if relevant)
      5.    Number of links
      6.    Owner's uid
      7.    Owner's gid
      8.    File size (bytes)
      9.    Last access time
      10.   Last modification time
      11.   Last file status change time
      12.   Preferred i/o block size
      13.   Number of blocks allocated

CALL g95_runtime_start()
Initialize the g95 runtime library. May be required in c programs calling
Fortran routines and linked using g95. Use before calling Fortran routines.

CALL GetArg(Pos, Value)
Pos: INTEGER; scalar; INTENT(IN)
Value: CHARACTER; scalar; INTENT(OUT)
Sets 'Value' to the Pos-th command-line argument.

CALL GetLog(Login)
Login: CHARACTER; scalar; INTENT(OUT)
Returns the login name for the process in ‘Login’.
```

GetPId()
INTEGER(KIND=1) function. Returns the process id for the current process.

HostNm(Name)
Name: CHARACTER; scalar; INTENT(OUT)
INTEGER(KIND=1) function. Fills 'Name' with the system's host name.

IsNan()
Elemental LOGICAL*4 function. Tests whether REAL or DOUBLE PRECISION numbers are Not-a-Number (NaN)

LStat(File, Sarray)
File: CHARACTER; scalar; INTENT(IN)
SArray: INTEGER(KIND=1); DIMENSION(13); INTENT(OUT)
INTEGER(KIND=1) function. Obtains data about the given 'File' and places them in the array 'SArray'. If 'File' is a symbolic link it returns data on the link itself, so the routine is available only on systems that support symbolic links. See Fstat() for details.

Rand(Flag)
Flag: INTEGER; OPTIONAL; scalar; INTENT(IN)
REAL(KIND=1) function. Returns a uniform quasi-random number between 0 and 1. If 'Flag' is 0, the next number in sequence is returned; if 'Flag' is 1, the generator is restarted by calling `srand(0)'; if 'Flag' has any other value, it is used as a new seed with srand.

CALL Rename(Path1, Path2, Status)
Path1: CHARACTER; scalar; INTENT(IN)
Path2: CHARACTER; scalar; INTENT(IN)
Status: INTEGER; OPTIONAL; scalar; INTENT(OUT)
Renames the file Path1 to Path2. If the Status argument is supplied, it contains 0 on success or an error code otherwise upon return.

CALL Signal(Number, Handler)
Number: INTEGER; scalar; INTENT(IN)
Handler: Signal handler (INTEGER FUNCTION or SUBROUTINE) or dummy/global INTEGER(KIND=1) scalar
If ‘Handler’ is an EXTERNAL routine, arranges for it to be invoked with a single integer argument (of system-dependent length) when signal ‘Number’ occurs. If ‘Number’ is an integer it can be used to turn off handling of signal ‘Handler’ or revert to its default action. Note that ‘Handler’ will be called with C conventions, so its value in Fortran terms is obtained by applying %loc (or loc) to it.

CALL Sleep(Seconds)
Seconds: INTEGER(KIND=1); scalar; INTENT(IN)
Causes the process to pause for ‘Seconds’ seconds.

CALL Srand(Seed)
Seed: INTEGER; scalar; INTENT(IN)
Reinitialises the random number generator with the seed in 'Seed'.

Stat(File, Sarray)
File: CHARACTER; scalar; INTENT(IN)
SArray: INTEGER(KIND=1); DIMENSION(13); INTENT(OUT)
INTEGER(KIND=1) function. Obtains data about the given File and places them in the array ‘SArray’. See Fstat()

CALL System(Command, Status)
Command: CHARACTER; scalar; INTENT(IN)
Status: INTEGER(KIND=1); OPTIONAL; scalar; INTENT(IN)
Passes the command 'Command' to a shell.

Time()
INTEGER(KIND=2) function. Returns the current time encoded as an integer in the manner of the UNIX function 'time'.

CALL Unlink(File, Status)
File: CHARACTER; scalar; INTENT(IN)
Status: INTEGER(KIND=1); OPTIONAL; scalar; INTENT(OUT)
Unlink the file 'File'. If the 'Status' argument is supplied, it contains 0 on success or an error code otherwise.

%VAL() and %REF()
Allow Fortran procedures to call c functions.

Installation Notes

Linux:
Open a console, and go to the directory in which you want to install g95. To download and install g95, run the following commands:

```
wget -O - http://www.g95.org/g95-x86-linux.tgz | tar xvfz -
ln -s $PWD/g95-install/bin/i686-x86-linux-gnu-g95 /usr/bin/g95
```

The following files and directories should be present:

```
./g95-install/
./g95-install/bin/
./g95-install/bin/i686-pc-linux-gnu-g95
./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.0/
./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.0/f951
./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.0/crtendS.o
./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.0/crtend.o
./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.0/crtbeginT.o
./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.0/crtbeginS.o
./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.0/crtbegin.o
./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.0/cc1
./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.0/libf95.a
./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.0/libgcc.a
./g95-install/INSTALL
./g95-install/G95Manual.pdf
```

The file cc1 is a symbolic link to f951 in the same directory.

Cygwin:
The -mno-cygwin option allows the Cygwin version of g95 to build executables that do not require access to the file cygwin1.dll in order to work, and so can be easily run on other systems. Also the executables are free of restrictions attached to the GNU GPL license. To install a Cygwin version with a working -mno-cygwin option, you will need the mingw libraries installed, available from the Cygwin site: http://cygwin.com/

Download the binary from http://www.g95.org/g95-x86-cygwin.tgz to your root cygwin directory (usually c:\Cygwin); start a Cygwin session, and issue these commands:

```
cd /
tar -xvzf g95-x86-cygwin.tgz
```

This installs the g95 executable in the /usr/local/bin directory structure.

Caution: Do not use Winzip to extract the files from the tarball or the necessary links may not be properly set up.

MinGW:
G95 requires the latest version of MinGW, including the binutils package. Install g95 by executing the self extracting install package g95-MinGW.exe from http://www.g95.org. Set the PATH to find both the MinGW\bin and the g95\bin directories.

Running G95

This section is provided to aid users unfamiliar with Unix compiler syntax.

Basic options:
-c Compile only, do not run the linker.
-o Specify the name of the output file, either an object file or the executable.

Multiple source and object files can be specified at once. Fortran files are indicated by names ending in ".f", ".F", ".for", ".FOR", ".f90", ".F90", ".f95", and ".F95". Multiple source files can be specified. Object files can be specified as well and will be linked to form an executable.

Files ending in uppercase letters are preprocessed with the C preprocessor by default, files ending in lowercase letters are not preprocessed by default.

Files ending in ".f", ".F", ".for", and ".FOR" are assumed to be fixed form source compatible with old f77 files. Files ending in ".f90", ".F90", ".f95" and ".F95" are assumed to be free source form.

Simple examples:

```
g95 -c hello.f90
```
Compiles hello.f90 to an object file named hello.o.

```
g95 hello.f90
```
Compiles hello.f90 and links it to produce an executable a.out (on Linux), or, a.exe (on MS Windows systems).

```
g95 -c h1.f90 h2.f90 h3.f90
```
Compiles multiple source files. If all goes well, object files h1.o, h2.o and h3.o are created.

```
g95 -o hello h1.f90 h2.f90 h3.f90
```
Compiles multiple source files and links them together to an executable file named 'hello', or 'hello.exe' on MS Windows systems.

Links

The g95 home page: http://www.g95.org
Documentation: http://www.g95.org/docs.html
This manual: http://www.g95.org/G95Manual.pdf
Source code: http://www.g95.org/g95_source.tgz
Authors: See the file AUTHORS in the g95 source for contributors to g95.
Bugs: Report bugs to andyv@firstinter.net

COPYRIGHT