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...]
     [-ooutfile]            name of outfile
     infile...
```

G95 Options

```
Usage: g95 [options] file...

-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.

By default g95 provides no optimization. For information on all the GCC options available when compiling with g95, see: http://gcc.gnu.org/onlinedocs/gcc-3.4.3/gcc/.

Command line arguments:
A program compiled with g95 may 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'
-frelative	Search relative to the source file 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_ABORT Boolean	If this is true and the program is ending abnormally, then this will cause a core dump.
G95_STDIN_UNIT Integer	Unit number that will be preconnected to standard input (No preconnection if negative) Default: 5
G95_STDOUT_UNIT Integer	Unit number that will be preconnected to standard output. (No preconnection if negative) Default: 6
G95_STDERR_UNIT Integer	Unit number that will be preconnected to standard error. No preconnection if negative) Default: 0
G95_USE_STDERR Boolean	Sends library output to standard error instead of standard output. Default: Yes
G95_ENDIAN String	Endian format to use for I/O of unformatted data. Values are BIG, LITTLE or NATIVE. Default: NATIVE
G95_CR Boolean	Output carriage returns for formatted sequential records. Default: true on windows, false elsewhere.
G95_IGNORE_ENDFILE Boolean	Ignore attempts to read past the ENDFILE record in sequential access mode. Default: false
G95_TMPDIR String	Directory for scratch files. Overrides the TMP environment variable. If TMP is not set /var/tmp is used. Default: ""

G95_UNBUFFERED_ALL Boolean	If TRUE, all output is unbuffered. This will slow down large writes but can be useful for forcing data to be displayed immediately. Default: No
G95_SHOW_LOCUS Boolean	If TRUE, print filename and line number where runtime errors happen. Default: Yes
G95_OPTIONAL_PLUS Boolean	Print optional plus signs in numbers where permitted. Default FALSE.
G95_DEFAULT_RECL Integer	Default maximum record length for sequential files. Most useful for adjusting line length of preconnected units. Default 500000000
G95_LIST_SEPARATOR String	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	For formatted output, print otherwise unprintable characters with \-sequences Default: FALSE
G95_QUIET Boolean	Suppress bell characters (\a) in formatted output. Default FALSE.
G95_SYSTEM_CLOCK Integer	Number of ticks per second reported by the SYSTEM_CLOCK() intrinsic in microseconds. Zero disables the clock. Default: 100000
G95_SEED_RNG Boolean	If true, seeds the random number generator with a new seed when the program is run. Default: FALSE.
G95_MINUS_ZERO Boolean	If true, allows minus zeros to be printed correctly, contrary to the standard. Default TRUE.
G95_MEM_INIT String	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	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	If true, shows the maximum number of bytes allocated in user memory during the program run. Default: No
G95_MEM_MXFAST Integer	Maximum request size for handing requests in from fastbins. Fastbins are quicker but fragment more easily. Default 64 bytes

G95_MEM_TRIM_THRESHOLD Integer	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. Default: 262144
G95_MEM_TOP_PAD Integer	Extra space to allocate when getting memory from the OS. Can speed up future requests. Default: 0
G95_SIGHUP String	Whether the program will IGNORE, ABORT or SUSPEND on SIGHUP. Default: ABORT
G95_SIGINT String	Whether the program will IGNORE or ABORT or SUSPEND on SIGINT. Default: ABORT
G95_FPU_ROUND String	Set floating point rounding. Values are NEAREST, UP, DOWN, ZERO. Default: NEAREST
G95_FPU_PRECISION String	Precision of intermediate results. Value can be 24, 53 and 64. Default 64
G95_FPU_DENORMAL Boolean	Raise a floating point exception when denormal numbers are encountered. Default: No
G95_FPU_INVALID Boolean	Raise a floating point exception on an invalid operation. Default: No
G95_FPU_ZERODIV Boolean	Raise a floating point exception when dividing by zero. Default: No
G95_FPU_OVERFLOW Boolean	Raise a floating point exception on overflow. Default: No
G95_FPU_UNDERFLOW Boolean	Raise a floating point exception on underflow. Default: No
G95_FPU_INEXACT Boolean	Raise a floating point exception on precision loss. Default: No
G95_FPU_EXCEPTIONS Boolean	Whether masked floating point exceptions should be shown after the program ends. Default: No
G95_UNIT_x	Default unit names
G95_UNBUFFERED_x	Unit buffering overrides
G95_INCLUDE_PATH	A colon separated list of directories

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

The following intrinsic procedures are available:

COMMAND_ARGUMENT_COUNT
GET_COMMAND
GET_COMMAND_ARGUMENT
GET_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, and structure components.

G95 Extensions - Intrinsic Procedures

abort
access
besj0
besj1
besjn
besy0
besy1
besyn
chdir
chmod
command_argument_count
dbesj0
dbesj1
dbesjn
dbesy0
dbesy1
dbesyn
DCMPLX()
derf
derfc
DFLOAT()
DREAL()
dtime
erf
erfc
etime
exit
fdate
float
flush
fstat
g95_runtime_start
getarg
get_command
get_command_argument
getcwd
get_environment_variable
getlog
getpid
hostnm
isnan
lstat
new_line
rand
rename
signal
sizeof
sleep
srand
stat
system
time
unlink
%val & %ref

abort

CALL abort() | INTEGER FUNCTION abort()

Prints a message and quits the program with a core dump.

access

```
INTEGER FUNCTION access(filename, mode)
CHARACTER :: filename
CHARACTER :: mode
```

Checks whether the file ‘filename’ can be accessed with the specified mode, where ‘mode’ is one or more of the letters ‘rwx’.

besj0

```
REAL FUNCTION besj0(x)
REAL :: x
```

Returns double-precision bessel function value (first kind, zero order).

besj1

```
REAL FUNCTION besj1(x)
REAL :: x
```

Returns double-precision bessel function value (first kind, first order).

besjn

```
REAL FUNCTION besjn(n,x)
INTEGER :: n
REAL :: x
```

Returns double-precision bessel function value (first kind, nth order).

besy0

```
REAL FUNCTION besy0(x)
REAL :: x
```

Returns double-precision bessel function value (second kind, zero order).

besy1

```
REAL FUNCTION besy1(x)
REAL :: x
```

Returns double-precision bessel function value (second kind, first order).

besyn

```
REAL FUNCTION besyn(n,x)
INTEGER :: n
REAL :: x
```

Returns double-precision bessel function value (second kind, nth order).

chdir
CALL chdir(dir) | INTEGER FUNCTION chdir(dir)
CHARACTER :: dir
Sets the current working directory to 'dir'.

chmod
INTEGER FUNCTION chmod(file,mode)
CHARACTER :: file
INTEGER :: mode
Change permissions for a file.

command_argument_count
INTEGER FUNCTION command_argument_count
Returns the number of arguments on the command line.

dbesj0
REAL FUNCTION dbesj0(x)
REAL :: x
Returns a double-precision bessel function value (first kind, zero order).

dbesj1
REAL FUNCTION dbesj1(x)
REAL :: x
Returns a double-precision bessel function value (first kind, first order).

dbesjn
REAL FUNCTION dbesjn(n,x)
INTEGER :: n
REAL :: x
Returns a double-precision bessel function value (first kind, nth order).

dbesy0
REAL FUNCTION dbesy0(x)
REAL :: x
Returns a double-precision bessel function value (second kind, zero order).

dbesy1
REAL FUNCTION dbesy1(x)
REAL :: x

Returns a double-precision bessel function value (second kind, first order).

dbesyn

```
REAL FUNCTION dbesyn(n,x)
INTEGER :: n
REAL :: x
```

Returns a double-precision bessel function value (second kind, nth order).

dcmplex()

Double precision CMPLEX()

derf

```
REAL FUNCTION derf(x)
REAL :: x
```

Returns the error function of x.

derfc

```
REAL FUNCTION derfc(x)
REAL :: x
```

Returns the complementary error function of x: derfc(x) = 1 - derf(x).

dfloat()

Double precision REAL()

dreal()

Alias for DBLE()

dtime

```
CALL dtime(tarray,result) | REAL FUNCTION dtime(tarray)
REAL, OPTIONAL, INTENT(OUT) :: tarray(2)
REAL, OPTIONAL, INTENT(OUT) :: result
```

Returns the runtime in seconds since the start of the process, or since the last invocation.

erf

```
REAL FUNCTION erf(x)
REAL :: x
```

Returns the error function of x.

erfc

```
REAL FUNCTION erfc(x)
REAL :: x
```

Returns the complementary error function of x: erfc(x) = 1 - erf(x).

etime

```
CALL etime(tarray,result) | REAL FUNCTION etime(tarray)
REAL, OPTIONAL, INTENT(OUT) :: tarray(2)
REAL, OPTIONAL, INTENT(OUT) :: result
```

Returns in seconds the time since the start of the process' execution.

exit

```
CALL exit(code)
INTEGER, OPTIONAL :: code
```

Exit a program with status 'code' after closing open Fortran i/o units.

fdate

```
CALL fdate(date) | CHARACTER FUNCTION fdate()
CHARACTER :: date
```

Returns the current date and time as: Day Mon dd hh:mm:ss yyyy

flush

```
CALL flush(unit)
INTEGER :: unit
```

Flushes the Fortran file ‘unit’ currently open for output.

fstat

```
CALL fstat(unit,sarray,status) | INTEGER FUNCTION fstat(file,sarray)
INTEGER :: unit
INTEGER, INTENT(OUT) :: sarray(13)
INTEGER, INTENT(OUT) :: status
```

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

g95_runtime_start

```
void g95_runtime_start(int argc, char *argv[])
```

Force an initialization of the g95 runtime library from C. This may be required in C programs calling Fortran routines, and linked using g95. Use before calling Fortran routines. Call g95_runtime_stop() when done. For more information see: http://www.g95.org/docs.html#interface

getarg

```
CALL getarg(pos, value)
INTEGER :: pos
CHARACTER, INTENT(OUT) :: value
```

Sets 'value' to the pos-th command-line argument.

get_command

```
CALL get_command(command,length,status)
CHARACTER :: command
INTEGER, OPTIONAL :: length
INTEGER, OPTIONAL :: status
```

Returns the command that invoked the program.

get_command_argument

```
CALL get_command_argument(number,value,length,status)
INTEGER :: number
CHARACTER :: value
INTEGER, OPTIONAL, INTENT(OUT) :: length
INTEGER, OPTIONAL, INTENT(OUT) :: status
```

Returns the command line argument ‘number’ in ‘value’.

getcwd

```
INTEGER FUNCTION getcwd(name)
CHARACTER :: name
```

Returns the current working directory in ‘name’.

get_environment_variable

```
CALL get_environment_variable(name,value,length,status,trim_name)
CHARACTER :: name
CHARACTER, OPTIONAL, INTENT(OUT) :: value
INTEGER, OPTIONAL, INTENT(OUT) :: length
INTEGER, OPTIONAL, INTENT(OUT) :: status
LOGICAL, OPTIONAL :: trim_name
```

Returns the value of the environment variable ‘name’ in ‘value’, its length in ‘length’, and sets ‘status’ = 0 if successful. If ‘trim_name’ is .true., trailing blanks are trimmed.

getlog

```
CALL getlog(name)
CHARACTER, INTENT(OUT) :: name
```

Returns the login name for the process in ‘name’.

getpid()

```
INTEGER FUNCTION getpid()
```

Returns the process id for the current process.

getuid

```
INTEGER FUNCTION getuid()
```

Returns the user's id.

hostnm

```
INTEGER FUNCTION hostnm(name)
CHARACTER :: name
```

Fills 'name' with the system's host name.

isnan

```
LOGICAL FUNCTION isnan(x)
REAL :: x
```

Tests whether ‘x’ is Not-a-Number (NaN).

lstat

CALL lstat(file,sarray,status) | INTEGER FUNCTION stat(file,sarray)

CHARACTER :: file

INTEGER, DIMENSION(13), INTENT(OUT) :: sarray

INTEGER, INTENT(OUT) :: status

If 'file' is a symbolic link it returns data on the link itself. See Fstat() for further details.

new_line

CHARACTER FUNCTION new_line(a)

CHARACTER :: a

Returns a new line character, achar(10)

rand

REAL FUNCTION rand(x)

INTEGER, OPTIONAL :: x

Returns a uniform quasi-random number between 0 and 1. If x is 0, the next number in sequence is returned; if x is 1, the generator is restarted by calling ‘srand(0)’; if x has any other value, it is used as a new seed with srand.

rename

CALL rename(path1, path2, status)

CHARACTER :: path1

CHARACTER, INTENT(OUT) :: path2

INTEGER, OPTIONAL, INTENT(OUT) :: status

Renames the file ‘path1' to ‘path2'. If the ‘status’ argument is supplied, it contains 0 on success or an error code otherwise upon return.

signal

CALL signal(signal,handler,status) | INTEGER FUNCTION (signal,handler)

INTEGER :: signal

PROCEDURE :: handler

INTEGER :: status

Calls the unix ‘signal’ routine.

sizeof

INTEGER FUNCTION sizeof(object)

The argument ‘object’ is the name of an expression or type.

Returns the size of ‘object’ in bytes.

sleep

```
CALL sleep(seconds)
INTEGER :: seconds
```

Causes the process to pause for ‘seconds’ seconds.

srand

```
CALL srand(seed)
INTEGER :: seed
```

Reinitialises the random number generator with the seed in 'seed'.

stat

```
CALL stat(file,sarray,status) | INTEGER FUNCTION stat(file,sarray)
CHARACTER :: file
INTEGER, INTENT(OUT) :: sarray(13)
INTEGER, INTENT(OUT) :: status
```

Obtains data about the given file and places it in the array ‘sarray’. See Fstat()

system

```
CALL system(cmd,result) | INTEGER FUNCTION system(cmd)
CHARACTER :: cmd
INTEGER, OPTIONAL :: result
```

Passes the command ‘cmd’ to a shell.

time

```
INTEGER FUNCTION time()
```

Returns the current time encoded as an integer in the manner of the UNIX function 'time'.

unlink

```
INTEGER FUNCTION unlink(file)
CHARACTER :: file
```

Unlink the file 'file'.

%val() and %ref()

Allow Fortran procedures to call C functions.

Using the Random Number Generator

```
random_number
CALL random_number(h)
REAL, INTENT(OUT) :: h
```

Returns a REAL scalar or an array of REAL random numbers in h, 0 <= h <1.

```
random_seed
CALL random_seed(sz,pt,gt)
INTEGER, OPTIONAL, INTENT(OUT) :: sz
INTEGER, OPTIONAL, INTENT(IN) :: pt(n1)
INTEGER, OPTIONAL, INTENT(OUT) :: gt(n2)
```

Argument ‘sz’ is the minimum number of integers required to hold the value of the seed; g95 returns 4.

Argument ‘pt’ is an array of default integers with size n1 >= sz, containing user provided seed values.

Argument ‘gt’ is an array of default integers with size n2 >= sz, containing the current seed.

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-pc-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.co.

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:

The g95 MinGW-based binary for Windows can provide two types of install. If MinGW is found, it installs into the MinGW file structure, otherwise it installs a complete stand-alone version with the supporting MinGW binutils files. Download g95 from http://www.g95.org/g95-MinGW32.exe. If you have MinGW, install g95 by executing the installer in the root MinGW directory. Set the PATH to find both the MinGW\bin and the g95\bin directories, and set the environment variable LIBRARY_PATH with:

```
SET LIBRARY_PATH = <path-to-MinGW/lib>.
```

Windows XP Users Note

MinGW currently allows about 8 mb for the heap on Windows XP. If your application requires access to more memory, try compiling with:

```
-Wl,--heap=0x01000000
```

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

Fortran 2003: http://j3-fortran.org/doc/standing/2003/007.pdf

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 STATEMENT