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-4.0.1/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

G95 can handle files that contain C preprocessor constructs.

-cpp	Force the input files to be run through the C preprocessor
-no-cpp	Prevent the input files from being preprocessed
-Dname[=value]	Define a preprocessor macro
-Uname	Undefine a preprocessor macro
-E	Show preprocessed source only
-Idirectory	Append 'directory' to the include and module files search path. Files are searched for in various directories in this order: Directory of the main source file, the current directory, directories specified by -I, directories specified in the G95_INCLUDE_PATH environment variable and finally the system directories.

Fortran Options

-Wall	Enable most warning messages
-Wimplicit-none	Same as -fimplicit-none
-Wline-truncation	Warn about truncated source lines
-Wobsolete	Warn about obsolete constructs
-Wno=numbers	Disable a comma separated list of warning numbers

-Wunused-vars	Warn about unused variables
-Wunset-vars	Warn about unset variables
-Wunused-module-vars	Warn about unused module variables. Useful for ONLY clauses
-Wprecision-loss	Warn about precision loss in implicit type conversions
-fbackslash	Interpret backslashes in character constants as escape codes. Use -fno-backslash to treat backslashes literally.
-fdollar-ok	Allow dollar signs in entity names
-fendian=	Force the endianness of unformatted reads and writes. The value must be 'big' or 'little'. Overrides environment variables.
-ffixed-form	Assume that the source file is fixed form
-ffixed-line-length-132	132 character line width in fixed mode
-ffixed-line-length-80	80 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
-fintrinsic-extensions	Enable g95-specific intrinsic functions even in a -std= mode
-fintrinsic-extensions=proc1,proc2,...	Include selected intrinsic functions even in a -std= mode. The list is comma-separated and case insensitive.
-fmod=directory	Put module files in directory
-fmodule-private	Set default accessibility of module-entities to PRIVATE
-ftr15581	Enable the TR15581 allocatable array extensions even in -std=F or -std=f95 modes.
-M	Produce a Makefile dependency line on standard output
-std=F	Warn about non-F features
-std=f2003	Strict fortran 2003 checking
-std=f95	Strict fortran 95 checking
-i4	Set kinds of integers without specification to kind=4 (32 bits)
-i8	Set kinds of integers without specification to kind=8 (64 bits)
-r8	Set kinds of reals without kind specifications to double precision
-d8	Implies -i8 and -r8.

Code Generation Options

-fbounds-check	Check array and substring bounds at runtime
-fcase-upper	Make all public symbols uppercase
-fleading-underscore	Add a leading underscore to public names
-fonetrip	Execute DO-loops at least once. (Buggy fortran 66)
-fpack-derived	Try to layout derived types as compact as possible. Requires less memory, but may be slower
-fqkind=n	Set the kind for a real with the 'q' exponent to n

-fsecond-underscore	Append a second trailing underscore in names having an underscore (default). Use -fno-second-underscore to suppress.
-fshort-circuit	Cause the .AND. and .OR. operators to not compute the second operand if the value of the expression is known from the first operand.
-fsloppy-char	Suppress errors when writing non-character data to character descriptors
-fstatic	Put local variables in static memory where possible. This is not the same as linking things statically (-static).
-funderscoring	Append a trailing underscore in global names (default). Use -fno-underscoring to suppress.
-max-frame-size=n	How large a single stack frame will get before arrays are allocated dynamically
-finteger=n	Initialize uninitialized scalar integer variables to n
-flogical=	Initialize uninitialized scalar logical variables. Legal values are none, true and false.
-freal=	Initialize uninitialized scalar real and complex variables. Legal values are none, zero, nan, inf, +inf and -inf.
-fpointer=	Initialize scalar pointers. Legal values are none, null and invalid.
-fzero	Initialize numeric types to zero, logical values to false and pointers to null. The other initialization options override this one.

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

Environment Variables

The g95 runtime environement 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	Unit number that will be preconnected to standard input. No preconnection if negative, default is 5.
G95_STDOUT_UNIT	Integer	Unit number that will be preconnected to standard output. No preconnection if negative, default is 6.
G95_STDERR_UNIT	Integer	Unit number that will be preconnected to standard error. No preconnection if negative, default is 0.
G95_USE_STDERR	Boolean	Sends library output to standard error instead of standard output.

		Default is Yes.
G95_ENDIAN	String	Endian format to use for I/O of unformatted data. Values are BIG, LITTLE or NATIVE. Default is BIG.
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. No 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 is False.
G95_SHOW_LOCUS	Boolean	If TRUE, print filename and line number where runtime errors happen. Default is 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 is 1G.
G95_LIST_SEPARATOR	String	Separator 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 No.
G95_QUIET	Boolean	Suppress bell characters (\a) in formatted output. Default No.
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, prints minus zero with a minus sign, contrary to the standard. Default false.
G95_ABORT	Boolean	If true, dumps core on abnormal program end. Useful for finding the locus of the problem. Default FALSE.
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 operating 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 mode. Values are NEAREST, UP, DOWN, ZERO. Default is NEAREST.
G95_FPU_PRECISION	String	Precision of intermediate results. Value can be 24, 53 and 64. Default 64. Only available on x86 and IA64 compatibles.
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	String	Overrides the default unit name for unit x.
G95_UNBUFFERED_x	Boolean	If true, unit x is unbuffered

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
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_ARGUMENT
GET_COMMAND	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.

Stream I/O - F2003 stream access allows a Fortran program to read and write binary files without worrying about record structures. For example:

```
character(len=5) :: a
open(7, file='output', status='old', access='stream')
read(7, pos=5) a
close(7)
print *, a
end
```

Reads five bytes directly from position 5 of the 'output' file. I/O can be formatted or unformatted. The INQUIRE statement has also been enhanced to add a POS= tag which returns the current position of the stream file, for a future READ or WRITE.

Note: ACCESS='transparent' is equivalent to access='stream'

Clive Page has written some documentation on this feature, available at.
http://www.star.le.ac.uk/~cgp/streamIO.html

IMPORT - can be used in an interface body to enable access to entities of the host scoping unit.

G95 Extensions - Intrinsic Procedures

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

fnum
INTEGER FUNCTION fnum(unit)
INTEGER, INTENT(IN) :: unit
Returns the file descriptor number corresponding to ‘unit’. (Unix)

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. More information here.

```
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
CALL unlink(file,status) | INTEGER FUNCTION unlink(file)
CHARACTER :: file
INTEGER, INTENT(OUT) :: status
Unlink the file 'file'. (Unix)

%val() and %ref()
Allow Fortran procedures to call C functions.

Interfacing with g95 programs

While g95 produces stand-alone executables, it is occasionally desirable to interface with other programs, usually C. The first difficulty that multi-language program will face is the names of the public symbols. G95 follows the f2c convention of adding an underscore to public names, or two underscores if the name contains an underscore. The -fno-second-underscore and -fno-underscoring can be useful to force g95 to produce names compatible with your C compiler.

Use the 'nm' program to look at the .o files being produce by both compilers. G95 folds public names to lowercase as well, unless -fupper-case is given, in which case everything will be upper case. Module names are represented as module-name_MP_name.

After linking, there are two main cases: Fortran calling C subroutines and C calling fortran subroutines. For C calling fortran subroutines, the fortran subroutines will often call fortran library subroutines that expect the heap to be initialized in some way.

To force a manual initialization from C, call g95_runtime_start() to initialize the fortran library and g95_runtime_stop() when done. The prototype of the g95_runtime_start() is:

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

The library has to be able to process command-line options. If this is awkward to do and your program doesn't have a need for command-line arguments, pass argc=0 and argv=NULL.

On OSX/Tiger, include '-lSystemStubs' when using g95 to run the linker and linking objects files compiled by gcc.

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/4.0.1/
 ./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.1/f951
 ./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.1/crtendS.o
 ./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.1/crtend.o
 ./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.1/crtbeginT.o
 ./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.1/crtbeginS.o
 ./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.1/crtbegin.o
 ./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.1/cc1
 ./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.1/libf95.a
 ./g95-install/lib/gcc-lib/i686-pc-linux-gnu/4.0.1/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-MinGW.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", ".F95", and ".f03" and ".F03" for F2003 files. 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", ".F95", ".f03" and ".F03" 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
Cool Features:	http://www.g95.org/cool.html
Compiling g95:	http://www.g95.org/src.html
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