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", ".f03" and ".F03". 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.
`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'.

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
-D*name[=value]*	Define a preprocessor macro
-U*name*	Undefine a preprocessor macro
-E	Show preprocessed source only
-I*directory*	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.
-traditional	Performs traditional C preprocessing (default)
-nontraditional	Performs modern C preprocessing

Fortran options

`-Wall`	Enable most warning messages
`-Werror`	Change warnings into errors
`-Wextra`	Enable warning not enabled by -Wall
`-Wimplicit-none`	Same as -fimplicit-none
`-Wline-truncation`	Warn about truncated source lines
`-Wobsolescent`	Warn about obsolescent constructs
`-Wno=`*`numbers`*	Disable a comma separated list of warning numbers
`-Wuninitialized`	Warn about variables used before initialized. Requires -O2
`-Wunused-vars`	Warn about unused variables
`-Wunused-types`	Warn about unused module types. Not implied by -Wall
`-Wunset-vars`	Warn about unset variables
`-Wunused-module-vars`	Warn about unused module variables. Useful for ONLY clauses
`-Wunused-module-procs`	Warn about unused module procedures. Useful for ONLY clauses
`-Wunused-parameter`	Warn about unused parameters. Not implied by -Wall
`-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.
`-fd-comment`	Make D lines executable statements in fixed form.
`-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
`-fmultiple-save`	Allow the SAVE attribute to be specified multiple times
`-fone-error`	Force compilation to stop after the first error.
`-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). Default kinds are unchanged.
`-i8`	Set kinds of integers without specification to kind=8 (64 bits). Default kinds are unchanged.
`-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).
-ftrace	'-ftrace=frame' will insert code to allow stack tracebacks on abnormal end of program. This will slow down your program. '-ftrace=full' additionally allows finding the line number of arithmetic exceptions (slower). Default is '-ftrace=none'.
-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.
-fround=	Controls compile-time rounding. Legal values are nearest, plus, minus and zero. Default is round to nearest, plus is round to plus infinity, minus is minus infinity, zero is towards zero.
-fzero	Initialize numeric types to zero, logical values to false and pointers to null. The other initialization options override this one.

Running g95 programs

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 --g95 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 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. 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 50000000.

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_LIST_EXP	`Integer`	Last power of ten which does not use exponential format for list output. Default 6.
G95_COMMA	`Boolean`	Use a comma character as the default decimal point for I/O. Default false.
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 without a minus sign in formatted (non-list) output, 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

-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)

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.

Information on OpenGL binding can be found at Nick Yasko's G95/OpenGL page.