FortranLint (Flint)
A FORTRAN SOURCE CODE ANALYZER
Version 7.0

Cleanscape Software International
172 College Street, STE A
Spencer, TN 38385
Telephone: (931) 528-LINT   Fax: (931) 933-7658
E-mail: sales@cleanscape.net -or- support@cleanscape.net

Copyright 1987-2014 Cleanscape Software International

CLEANSCAPE
software international

NOTE: This is a print copy of the online help listing; none of the hyperlinks shown like this
in the document are enabled. To get the fully interactive online help listing
- start the Flint GUI;
- select the Help menu; then
- select Contents and Index

1. Introduction

FortranLint is a Fortran source code analyzer with FORTRAN 77 and Fortran 90/95 support. FortranLint (or FLINT) checks for a range of potential problems, including:

- `invalid arguments passed to library routines`
- `inconsistencies in common-block declarations`
- `data-type usage conflicts`
- `portability problems`
- `unused variables and/or routines`
- `variables which are referenced, but not set`

FortranLint checks source files for problems both locally and as a group. This allows FortranLint to detect problems that compilers may miss.

FortranLint also produces subroutine-call trees, cross-reference tables (with optional filters), a use hierarchy, an include tree, source listings, and statistical reports.

FortranLint supports the ANSI FORTRAN 77 and Fortran-Extended (Fortran 90) standards. FortranLint also supports vendor-specific dialects and language extensions (such as HPF and OpenMP).

FortranLint runs under UNIX (including Linux) and Windows 9x through Vista. The program may be used either as a command-line tool or through this GUI interface.

FortranLint is a registered trademark of Cleanscape Software International.

2. Running Flint

FortranLint operates on one "project" at a time. To run the program, proceed as follows:

```
Create a new project     - use Project: New
Add files to project     - use File: Add
Set analysis options     - use Analysis tab
Select source format     - use Source Config tab
Select reports           - use Reports tab
Set misc. options        - use Misc Options tab
Select files to process  - use mouse or Select buttons
Process the files        - use Run: Go
View the results         - use individual report tabs
Save the project state   - use Project: Save
View the documentation   - use the appropriate item in the Help dropdown
Exit                     - use Project: Exit
```

For more information, see the main FortranLint manual. For more information on keyboard shortcuts, see the Section 4.K in the FortranLint GUI User's Guide.

3. Analysis options

3.1. Overview. FortranLint provides the following analysis modes and options:

```
Component Test        - Test file(s) in program context
Local  mode           - Intra-module checking
Global mode           - Add inter-module tests
Warnings              - Report problems (only)
FYIs                  - Add "informational" messages
Data-flow analysis    - Perform data-flow analysis
Data Usage            - Reference/set problems, etc.
Implicit Typing       - Report un-typed variables
ANSI MAXLOC rules     - Modified MAXLOC/MINLOC rules
Search LBTs first     - Check LBTs before intrinsics
Portability checking  - Check multiple target systems
```

To change analysis settings, use the Analysis Options tab. Individual options are discussed below.

Note: For more information on analysis options, see chapter 5 of the main FortranLint manual.

3.2. Component Testing.

Command-line switch: N/A

If Component Test is selected, the FortranLint GUI invokes external program `usescan` (located in the `bin` subdirectory) to scan all occurrences of the USE keyword and build a list of files containing the module definitions for those USE calls.

This, in combination with Flint's automatic INCLUDE scanning, will result in comprehensive testing of the selected files (components) without necessarily having to run Flint over the entire source tree.

If Component Testing is turned off, FortranLint analyzes the selected source files only, and will probably report errors for missing modules.

Component Testing can be important to locate and resolve module definitions for the USE Tree report.

NOTE: Component testing, while reducing analysis time, may require non-trivial time to scan for all USE statements and build the resulting list of source files required to thoroughly test the specified component source files.

3.3. Local mode and global mode.

Command-line switch: -g

If Global mode is selected, FortranLint checks subprograms as a group for problems. For example, this mode detects mismatched argument lists and common-block inconsistencies.

If Global mode is turned off, FortranLint operates in Local mode. In this mode, FortranLint checks subprograms on an individual basis; for example, call interface checking is not performed.

Note: The Cross-Reference option requires Global mode. If Global mode is turned off, the Cross-Reference is disabled.

3.4. Warnings and FYIs.

Command-line switches: -w and -f

If the Warnings option is selected, FortranLint generates warnings in addition to error messages. This option is enabled, by default.

If the FYI option is selected, FortranLint generates informational messages in addition to warnings and/or error messages.

3.5. Data-Flow Analysis.

Command-line switch: -Fon

If this option is enabled, FortranLint performs local data-flow analysis. For more information on this feature, see section 5.6 of the main FortranLint manual.

NOTE: Dataflow analysis can require significant processing power, memory, and time; indeed, it is possible for dataflow analysis to take years to complete! Contact support@cleanscape.net for details and assistance using this advanced feature.

3.6. Data Usage.

Command-line switch: -u

If this option is enabled, FortranLint checks for data usage problems, such as variables used but never set.

3.7. Implicit Typing.

Command-line switch: -m

The Implicit Typing option is similar to the Fortran IMPLICIT NONE statement. If this option is enabled, FortranLint reports instances of implicit typing.

3.8. ANSI MAXLOC rules.

Command-line switch: -Mansi_maxloc

If HPF directives are enabled, FortranLint normally uses the following rules for MAXLOC and MINLOC (non-DEC targets only):

```
MAXLOC (ARRAY, DIM, MASK)
MINLOC (ARRAY, DIM, MASK)
```

ARRAY   must be integer or real array
DIM   is optional; if present, must be integer scalar
MASK   is optional; if present, must be of local type and conformable with ARRAY

To override the HPF rules, go to the Analysis Options screen and enable ANSI MAXLOC rules. This option applies the ANSI X3.198-1992 rules (disallowing the DIM argument).

This option doesn't apply to DEC targets.

3.9. Search LBTs first.

Command-line switch: -Muselbt

LBT files are library template files that describe call interfaces for user libraries or system libraries. (For a complete explanation, see chapter 9 of the main FortranLint manual.)

FortranLint normally searches for definitions as follows (highest precedence first):

```
1. Check Fortran source files
2. Check system intrinsic table
3. Check user's LBT files
4. Check unixlib.lbt or vmslib.lbt
```

By default, the intrinsic table takes precedence over the user's LBT files. If Search LBTs first is checked, FortranLint searches the user's LBT files before the intrinsic table. (Steps 2 and 3 here are reversed.)

Note: The system-library templates unixlib.lbt and vmslib.lbt have the lowest precedence, whether or not this option is selected.

3.10. Portability checking.

Command-line switch: -P system-name

FortranLint will optionally check code portability based on one or more target environments.

To use portability checking, go to the Analysis Options screen and enable Portability, then select any of the target environments shown in the associated listbox.

4. Source Configuration

4.1. Overview.

FortranLint provides language and source-code options related to:

```
Language            - FORTRAN 77 or Fortran 90
Dialect             - Fortran compiler
Source format       - Free vs. fixed format, etc.
Special directives  - 'C' preprocessor, HPF, etc.
INCLUDE directories - Used by Fortran INCLUDEs
Misc. features      - Two-byte integers, etc.
```

To change Source Configuration settings, use the Source Config tab options discussed below.

Note: For more information, see chapter 4 of the main FortranLint manual.

4.2. Language.

Command-line switch: -7 or -9

The Language setting specifies the version of Fortran used. The default mode (Automatic) determines the version automatically.

Note: FORTRAN 77 sources can be processed in Fortran 90 mode; however, this may produce version-related warning messages.

4.3. Dialect.

Command-line switch: -V system-name

The Dialect setting specifies the host environment; i.e., the Fortran compiler used. The default setting (Automatic) assumes that the current environment will be used.

4.4. Source format.

4.4.1. Full explanation.

FortranLint understands the following source-code formats:

```
FORTRAN 77 standard format
FORTRAN 77 132-column format
Fortran 90 fixed format
Fortran 90 free  format
```

In FORTRAN 77 standard format, column 6 is reserved for a continuation indicator, columns 7 through 72 are reserved for source code, and columns from 73 onward are reserved for comments.

To select FORTRAN 77 standard format, go to the Source Config screen, set Language to FORTRAN 77, and turn the 132 Columns option off.

To process FORTRAN 77 source files containing statements which are longer than 72 columns, turn the 132 Columns option on. Note that in-line comments are not allowed, in this mode. Comment lines (with a "C" in column one) are not affected.

Fortran 90 fixed format is similar to FORTRAN 77 standard format. Fortran 90 adds a new comment delimiter; i.e., "!" may be used to start an in-line comment.

Under Fortran 90 free format, lines may be up to 132 columns long. The "!"-comment convention is supported. (The older FORTRAN "C in column one" convention can only be used in fixed-format mode.)

To select Fortran 90 fixed or free format, go to the Source Config screen and set Language to Fortran 90. To select fixed or free format based on filename extensions, set Source Format to Automatic.

To force fixed-format mode, select Fixed Format instead.

To force free-format mode, select Free Format.

4.4.2. Free vs. Fixed format.

Command-line switch: -R free or -R fixed

To specify Free or Fixed format, use the Source Format selection box.

There are three settings: Automatic, Free, and Fixed.

If Automatic is selected:

    .f   sources are treated as "fixed format"
    .f90 sources are treated as "free format"

If Free Format is selected, FortranLint ignores filename extensions and treats source files as free-format, regardless.

If Fixed Format is selected, FortranLint treats source files as fixed-format, regardless.

Note: These are Fortran 90 options. If FORTRAN 77 is selected, these settings are ignored. FORTRAN 77 sources are always treated as fixed-format files.

4.4.3. 132 Columns.

Command-line switch: -e

To enable 132 Columns mode, use the 132 Columns checkbox on the Source Config screen.

If FORTRAN 77 and/or Fixed Format are selected, FortranLint normally assumes that input lines are 72 columns long. Characters after column 72 are ignored.

If the 132 Columns option is enabled, FortranLint extends the 72-column limit to 132 columns.

4.5. Special directives.

4.5.1. 'C' preprocessor.

Command-line switches: -p, -#*full_path_to_preproc*

FortranLint supports local 'C' preprocessors; in other words, 'C' directives such as #define or #include can be used. By convention, capitalized file extensions (.F, .F90) will automatically invoke the C preprocessor (as long as it is in your system PATH).

To send all Fortran source files through the local 'C' preprocessor, go to the Source Config screen and enable 'C' preprocessor. If the preprocessor is not in your PATH, specify it using the associated textbox or Browse button on this tab.

4.5.2. "Debug" lines.

Command-line switch: -d

If FORTRAN 77 and/or Fixed Format are selected, FortranLint assumes that input lines beginning with "D" are "debug" lines. "Debug" lines are normally treated as comments.

To process "debug" lines, go to the Source Config screen and enable Debug Lines.

4.5.3. HPF directives.

Command-line switch: -Mhpf

FortranLint supports High Performance Fortran (HPF). By default, HPF directives are treated as normal comments. To process HPF directives, go to the Source Config screen and enable HPF directives.

4.6. OpenMP directives.

Command-line switch: Contact Cleanscape

For information on OpenMP support, contact Cleanscape Software at (800) 94-4LINT or sales@cleanscape.net.

4.7. INCLUDE directories.

Command-line switch: -Idirectory

Users may specify one or more INCLUDE directories for a project. If the project sources use Fortran INCLUDE statements, FortranLint searches these directories for files referenced by the INCLUDE statements. As of version 7.0, the GUI also uses these directories to search for module definitions when building a USE hierarchy.

To specify INCLUDE directories, add them to the Include directories field on the Source Config screen. Note: Use complete paths (such as /opt/include under UNIX or C:\project\include under MS-Windows).

4.8. Misc. source options.

4.8.1. Two-byte integers.

Command-line switch: -2

By default, INTEGER and LOGICAL declarations are treated as INTEGER*4 and LOGICAL*4. If the Two-byte INTEGERs option is enabled, INTEGER and LOGICAL declarations are treated as INTEGER*2 and LOGICAL*2. This option does not affect declarations which include length specifiers.

4.8.2. Ignore INCLUDE paths.

Command-line switch: -Mignore_paths

The Ignore INCLUDE paths option may be useful when Fortran files are moved from one machine to another. If this option is selected, FortranLint ignores directory paths inside INCLUDE statements.

Note: To specify the INCLUDE directories which should be used, add them to the Include directories field.

4.8.3. Ignore VMS logicals.

Command-line switch: -Mignore_logicals

The Ignore VMS logicals option may be useful when VMS Fortran files are processed under MS-Windows or UNIX.

If this option is selected, FortranLint ignores VMS "logicals" inside INCLUDE statements.

Note: To specify the INCLUDE directories which should be used locally instead of the "logicals", add them to the Include directories field.

5. Flint reports

5.1. Overview.

Flint produces some or all of the following reports, depending on the options selected:

```
Analysis          - Diagnostics (Lint) Results
Call Tree         - Program "calling" structure
Cross Reference   - Program symbol tables
Use Hierarchy     - USE module tree
Include tree      - INCLUDE and #include diagram
Statistics        - Produce summary/stat information
Stack Usage       - Show auto storage and stack info
Auto-load/save    - Load or save reports automatically
Report color      - Selects color of output reports
External editor   - Selects editor for hyperlinking
```

All reports are color-coded and any lines, numbers, or symbol names in bright red are hyperlinks. Clicking on any of these hyperlinks will jump to the associated source file / line number associated with that link, provided an External Editor has been selected.

To select or configure individual reports, use the **Reports** tab. Reports are discussed individually below. To generate reports after configuration is completed, see section 5.11. To access generated reports, see section 5.12.

5.2. Analysis Report.

5.2.1 Main Report. Command-line switch: N/A

C++lint always produces an Analysis report. The Analysis report describes all of the problems found, including syntax errors, global interface problems, etc. The level of detail depends on the options selected on the Analysis Options screen.

For more information on the Analysis report, see Chapter 6 of the main FortranLint manual located in the doc subdirectory.

5.2.2 Source Listing. Command-line switch: -l

The Analysis report includes an optional source listing. The source listing shows errors in context. To produce a source listing, go to the Reports screen and enable Source Listing.

5.2.3 Call Tree Report. Command-line switch:-t

FortranLint generates optional call trees, which are graphical representations of a program's "calling" structure. The sample call tree below shows that PROCDAT calls three routines, including PRINT, which calls one routine.

```
PROCDAT-+-GETUNIT
        |
        +-READNAME
        |
        +-PRINT--PRINTIT-+-DIPSTAT--*PRINT*
                         |
                         +-GETTING
```

By default, the call tree covers the entire program. To specify a starting point for the call tree, go to the Misc Options screen and add a function or subroutine name to the Call Tree Roots field.

To generate multiple call trees for individual pieces of the program, add more names to the Call Tree Roots field.

To change the call-tree format, use the Call Tree sub-options on the Reports screen:

```
Squash Tree    - Remove white space to make a shorter tree.
Condense Tree  - If a routine is called more than once, merge all of the
                 calls to that routine.
No Library     - Ignore calls to library routines, as defined by LBT files.
                 Note: For an explanation of LBT files, see chapter 9 of the
                 main FortranLint manual.
No Undefined   - Omit calls to undefined routines.
Sorted Tree    - Sort call trees alphabetically by routine name.
```

To use this feature, go to the Reports screen and enable Call Tree. For more information on call trees, see chapter 7 of the main FortranLint manual.

5.2.4 Cross Reference Report.

5.4.1 Basic Information. Command-line switch: -x

FortranLint produces an optional Cross-Reference report. The Cross-Reference provides usage information for local variables, subroutine arguments, and global variables, including equivalence variables. For example:

```
*** Vars/Arrays:
AVE : I*4 : public entity of module M
        in (demo90.f90:M) is   11-D
        in (demo90.f90:MAIN) is   50-S
I : I*4 : local
        in (demo90.f90:MAIN::MAIN_INNER) is   (demo90.inc)6-RS
J : I*4 : local
        in (demo90.f90:MAIN::MAIN_INNER) is   (demo90.inc)7-RS
STR : CHAR*10 : local
        in (demo90.f90:MAIN) is   44-D  49-S

*** Parameters:
GRADE (6) : CHAR*2
        in (demo90.f90:MAIN) is   49-R

*** Structure components:
TYPE1%NAME : CHAR*10
        in (demo90.f90:M::M_INNER) is   20-S
```

To use this feature, go to the Reports screen and enable Cross-Reference.

Note: The Cross-Reference report requires Global mode. If the Cross-Reference option is greyed out, go to the Analysis Options screen and turn Global mode on.

For more information on the Cross-Reference report, see chapter 8 of the main FortranLint manual.

5.4.2. Cross-Reference formats.

To change the default Cross-Reference format, use the Cross-Reference sub-options on the Reports screen:

```
Free Form   - Selects a variable-width format that includes line numbers.
Tabular     - Selects a fixed-width (132 column) format.
```

Note: The format must be Free Form if hyperlinks are desired in either the Cross Reference or the Call Tree reports.

For more information on the Cross-Reference report, see chapter 8 of the main FortranLint manual.

5.4.3. Cross-Reference filters.

FortranLint supports Cross-Reference "filter" commands. Filter commands may be used to limit Cross-Reference tables to areas of interest.

For example, the following filter command will produce a Cross-Reference limited to variables that are either used or equivalenced:
```
    used_variables and_equivalenced_variables
```

For instructions on defining filter commands, see section 8.4 of the main FortranLint manual. To use a given filter command, go to the Misc Options screen and put the command in the Cross Reference Filters field.

5.5. USE Hierarchy.   Command-line switch: N/A

If this option is selected, FortranLint generates one or more USE trees. USE trees show the relationships between use modules, the related module definitions, and the calling subprograms (which may be programs, subroutines, functions, or modules).

Hyperlinks (colored red) are available from the USE tree report to the appropriate source file / line that is the entry point to the module definition or subprogram caller, in the selected editor.

Uncolored (normal foreground color) function names do not have hyperlink information and thus no hyperlinking is available. In most cases this is because the source code is missing or otherwise

not available. If this is the case, try resolving missing code by using the Component Test analysis option.

Example:

```
BEGIN USE TREE

hc_global_routines.f
 global_hc_init
  comhc
 global_hc_begin
  comhc
  hc_start
 global_hc_end
  comhc
  hc_start
 global_hc_store_raw_data
  hc_init_div_diag
   hc_storage_setup
    comhc
  comhc
```

To enable USE Trees, go to the Report Options tab and turn USE Tree on. By default, the nodes of the tree are in order of access; if you'd like to sort at the subprogram level, click the Sort Subprograms checkbox.

5.6. Include Trees. Command-line switch: N/A

Fortranlint can generate one Include report for each selected source file.

If the INCLUDE Tree box is checked on the Report Options tab, a tree is generated that demonstrates the nesting structure of the Include files used by the selected Fortran source file(s).

The GUI reports both INCLUDE and #include (preprocessor) directives. Filenames in angle brackets indicate system preprocessor include files.

Hyperlinks (colored red) are available from the Include report to the appropriate header source file, in the selected editor.

Uncolored (normal foreground color) function names do not have hyperlink information and thus no hyperlinking is available. In most cases this is because the source code is not available.

Example:

```
main.f90
 foo.inc
  bar.inc
   comhc.f
   hc_init.f
 baz.inc
  comhc.f
```

5.7. Statistics. Command-line switch: -s

FortranLint produces an optional Statistics report. The Statistics report includes a list of errors detected, sorted by frequency. This report also provides some source-level statistics (number of functions, etc.).

To use this feature, go to the Reports screen and enable Statistics. For more information on the Statistics report, see section 6.4 of the main FortranLint manual.

5.8. Auto-load / Auto-save Reports. Command-line switch: N/A

If selected, these options will load and/or save the output reports along with the rest of the project.

5.9. Report Font Color. Command-line switch: N/A

To make space available for other report options, the font color option is relegated to the `flint.ini` text file located in the `main` subdirectory. Simply edit the file to enter the desired color (red, blue, black) where indicated.

Default report width (75) can also be changed in this file in the same fashion.

5.10. External editor and hyperlinking. Command-line switch: N/A

Hyperlinks are generated between each analysis message and the source file / line number causing the message. The link itself is the line number, which will appear in bright red in the Analysis Report. The following editors are supported out-of-the-box:

Windows:
```
    Borland CodeWright   Starbase CodeWright
    Crimson Editor       Epsilon Programmer's Editor
    Emacs                GWD Text Editor
    GVim                 MultiEdit
    TextPad              UltraEdit
    Visual SlickEdit     Visual Studio 6--2010 *
```

Unix/Linux:
```
    Elvis                NEdit
    Emacs                Pico *
    Jed *                Vi *
    Joe *                Vim *
    Nano *               XEmacs
```

* These editors are not easily controlled via external tools; in all cases, multiple instances of the editor will be started with each link click.

The GUI knows the default installation directory for each of these editors and will load this location in the **Editor Location** textbox. If you installed your editor in a different location, use the `Locate...` button and browse to that location. NOTE: "Locate" is preferred to typing the location directly in the textbox.

NEW: Use program `seteditor` located in the bin subdirectory to add your favorite editor to the GUI! See Section 6.2 of the GUI manual for more information.

5.11. Generating reports.

To generate (or regenerate) all currently-selected reports, select the files to be processed and use the **Analyze Files** command on the **Process** menu.

5.12. Accessing reports.

To view reports, use the individual report tabs on the left side of the flint screen (Analysis, Statistics, etc.). To print reports, or to save them to disk, use the **Reports** menu at the top of the screen. Reports may be printed or saved collectively or individually.

Note: By default, Flint copies reports to disk automatically when a project is saved, and loads reports from disk automatically when a project is loaded. For more information, click here.

---

6. Projects

Projects are a collection of source filenames along with analysis/report settings. You can open, save, and close projects. You can specify a project when starting Cleanscape GUI by adding the full filename of the project to the Target value in the Properties dialog box of the GUI shortcut on your desktop.

Cleanscape GUI also uses a "template project", which is opened by default if no project file is specified at startup. The template contains all the analysis settings (the tabs in the lower left frame) from the last GUI session (e.g., warning level, editor selection, include directories), but the file list is empty.

The template can be omitted if `-blank` is added to the desktop shortcut's Target value (Windows) or to the GUI's invocation in a command prompt.

6.1. Saving a project. To save the current state of a project, use the **Save** option on the **Project** menu, or press the third button on the image toolbar. The resulting *project.csi* file holds the source-file list and analysis/report option settings.

If the project is new (in other words, if it hasn't been saved before), **Save** prompts the user for the directory path and base name to be used.

To create a new project file based on the one currently opened, use **Save As** instead of **Save**.

6.2. Loading an old project. To load a **.csi** file which was saved previously, use the **Open** option on the **Project** menu, or press the second button on the image toolbar.

If the project has been recently used, look for its name in the Recent Projects list.

6.3. Starting a new project. To start a new project, use the **New** command on the **Project** menu, or press the first button on the image toolbar.

Note: If a project is already open, the GUI will ask you if you want to save the old project first.

6.4. Source-file list. Source files whose fully qualified pathname are too long to fit in the Project window are truncated to ~60 characters, with an ellipsis ("...") prepended to the truncated name. Hovering the mouse over the entry will show the full filename in a balloon.

Any source file in the list can be opened in the previously selected External Editor by right-mouse-clicking (Win) or middle-clicking (Unix/Linux) the filename in the Project window.

6.4.1. Adding source files. To add source files to a project, use **Add File** on the **File** menu, or press the **Add File** button near the middle of the screen.

Note: The file-selection dialog supports multiple-file selection under both MS-Windows and UNIX. To add multiple files individually, use Control-Left Click. To add a group of files, left-click on the first file, then press Left Shift and drag the mouse.

6.4.2. Selecting source files.

To select all files for processing, use the **Select All** button. To select or deselect individual source files, left-click on each file while holding down the CONTROL key. To deselect all files, use the **Select None** button.

6.4.3. Removing source files. To remove individual source files from a project, select the files to be removed (as explained in the preceding section), then press the **Remove File** button.

To remove all files from a project (i.e., to clear the file list), press **Select All**, then press **Remove File**.

Note: **Remove File** modifies the source-file list. It does not delete the specified files.

6.5. Files used by Flint. Flint treats the directory chosen for a project's **.csi** file as a "home directory" for the project. Files are written to this directory when a project is saved, and they are loaded from this directory when a project is loaded.

Report and project lint files have the same base name as the **.csi** file. For example, if the user saves a project to **foo.csi**, the following names are used for the associated files:

```
foo.lnt - Analysis Report
foo.tre - Call Tree
foo.stt - Statistics Report
foo.xrf - Cross-Reference
foo.inc - Include Tree
foo.use - Use hierarchy
```

If a new project has been created, but not saved, Flint stores the project's reports in temporary files. When the project is saved for the first time, Flint copies the temporary files to the new project directory.

It is also not possible to create a *project.lnt* file without first saving the project (there's no project name until then).

To disable report auto-saves, go to the **Reports** option screen and turn **Auto-save reports** off.

To disable report auto-loads, go to the **Reports** option screen and turn **Auto-load reports** off.

Note: You can use the **Report** menu at the top of the GUI to print reports, or to save them to other locations. Reports may be printed or saved collectively or individually.

7. Documentation

The documentation for FortranLint is stored in the following directory on this system:

Windows: `<install_dir>\doc`
*nix: `<install_dir>/flintgui.dir/doc`

The filename for the Flint core engine is `flintman.pdf`.
The filename for the GUI documentation is `flintgui.pdf`.

Select the appropriate item in the Help dropdown to open either document.