

Cleanscape Fortran-lint Quick Start Guide

Version 7.x



Sales and Service Office

PO Box 616
Franklin Springs, GA 30639
Toll-free 800-944-LINT
Direct 706-245-1070
Fax 706-432-1720

www.cleanscape.net
sales@cleanscape.net
support@cleanscape.net

Note: Licensed users may photocopy for distribution.

**Direct comments concerning this manual to the address on the title page or
support@cleanscape.net**

Copyright © 1987-2020

**CLEANSCAPE
NOTICE OF COPYRIGHTS**

Copyrighted by Cleanscape as an unpublished work. All rights reserved. In claiming any copyright protection which may be applicable, Cleanscape reserves and does not waive any other rights that it may have (by agreement, statutory or common law, or otherwise) with respect to this material. See Notice of Proprietary Rights.

NOTICE OF PROPRIETARY RIGHTS

This manual and the material on which it is recorded are the property of Cleanscape. Its use, reproduction, transfer and/or disclosure to others, in this or any other form, is prohibited except as permitted by a written License Agreement with Cleanscape. Cleanscape reserves the right to update this document without prior notification.

FortranLint is a trademark of Cleanscape Software International.

All other trademarks and registered trademarks are the property of their respective owners.

NOTE: Some screenshots in this document may be of older versions. In such cases the salient feature(s) on that screen are unchanged in the newer version.

Table of Contents

PART I Introduction	5
1.1 WELCOME	5
1.2 DOCUMENTATION.....	5
1.3 PURPOSE	5
A. Function.....	5
B. Application.....	6
C. Advantages	6
D. Flow of Analysis	6
PART II Requirements, Installation, and Uninstallation	7
2.1 WINDOWS	7
A. System Requirements.....	7
B. Software Setup Procedure.....	7
C. Uninstallation.....	8
2.2 UNIX/LINUX	9
A. System Requirements.....	9
B. Software Setup Procedure.....	9
C. Uninstallation – manual process.....	9
PART III Activating Flint	11
A. Registration Process – Windows	11
B. Registration Process – Linux/Unix.....	11
PART IV Running the Flint GUI	13
A. Overview	13
B. Components.....	16
C. Creating a new project.....	19
D. Opening an existing project	19
E. Saving a project.....	20
F. Modifying a project	21
G. Execute test	23
H. Review reports	23
I. Online Help	28
J. Sub-Menu Functions.....	28
K. Operating the GUI using the Keyboard; Keyboard Shortcuts.....	30
L. Changing fonts / sizes	31
PART V Running Flint from the Command Line	33
A. Introduction.....	33
B. Operation.....	33
C. Return Codes	33
PART VI Running Flint from IDEs	34
A. Overview	34
B. Installation	34
C. Operation	35
D. Uninstallation.....	37
PART VII MISCELLANEOUS INFORMATION	39
7.1 ADDITIONAL STEPS FOR WINDOWS 2000+.....	39
A. Applicability	39
B. Details.....	39
7.2 ADDING AN EXTERNAL EDITOR TO THE GUI USING SETEDITOR	41
A. Introduction.....	41
B. Operation.....	41
7.3 TESTING SELECTED FILES IN PROGRAM CONTEXT USING USESCAN.....	43
A. Introduction.....	43
B. Definition Mode	43
C. Resolution Mode	44

PART I Introduction

1.1 WELCOME

Thank you for your product purchase! With Cleanscape Fortran-lint (Flint), you have the most powerful static source (lint) analysis available for Fortran 77→08 code. Flint in its command-line form has been assisting Fortran programmers for over a quarter century; the GUI is an ease-of-use enhancement to the venerable Flint product for a new generation programmers – and anyone tired of command prompts or desiring the productivity gains available with a GUI.

As of version 7, there are 871 unique messages that can diagnose 1519 situations in Fortran code.

1.2 DOCUMENTATION

This is the “quick start” guide for the Flint static analyzer. There are three modes of Flint operation on Unix/Linux, and three on Windows:

- A. *Cleanscape GUI*
- B. *Command line*
- C1. *Visual Studio integration using Cleanscape automation (Windows only)*
- C2. *Xlint graphical browser (Unix/Linux only). This product remains under support, but the Flint GUI effectively supersedes its functionality.*

This document's sole purpose is to describe the ease-of-use enhancements provided by Cleanscape GUI over the Flint command-line product. Flint is very rich in analysis controls and reporting; to gain maximum benefit from your product purchase, we urge you to read and keep handy the companion document, [Flint Reference Manual](#).

While on the topic of documentation: if you choose Cleanscape GUI, be sure to check out the Online Help facility! It's concise yet useful information. The Table of Contents and many interrelated items in the help text are hyperlinked to make information access quick and easy.

New features in the latest release are marked in this document with **NEW v7.0**.

1.3 PURPOSE

A. Function

1. Flint is a programming tool that simplifies the debugging and maintenance of both large and small Fortran programs. The Flint GUI provides ease-of-use enhancements to the venerable Flint command line product.
2. The Flint source code analyzer that can detect a wide range of potential problems, including:
 - a. Inappropriate arguments passed to functions
 - b. Inappropriate library calls
 - c. Non-portable code
 - d. Type usage conflicts across different modules
 - e. Unused variables and dead code

B. Application

1. Flint can be used to:

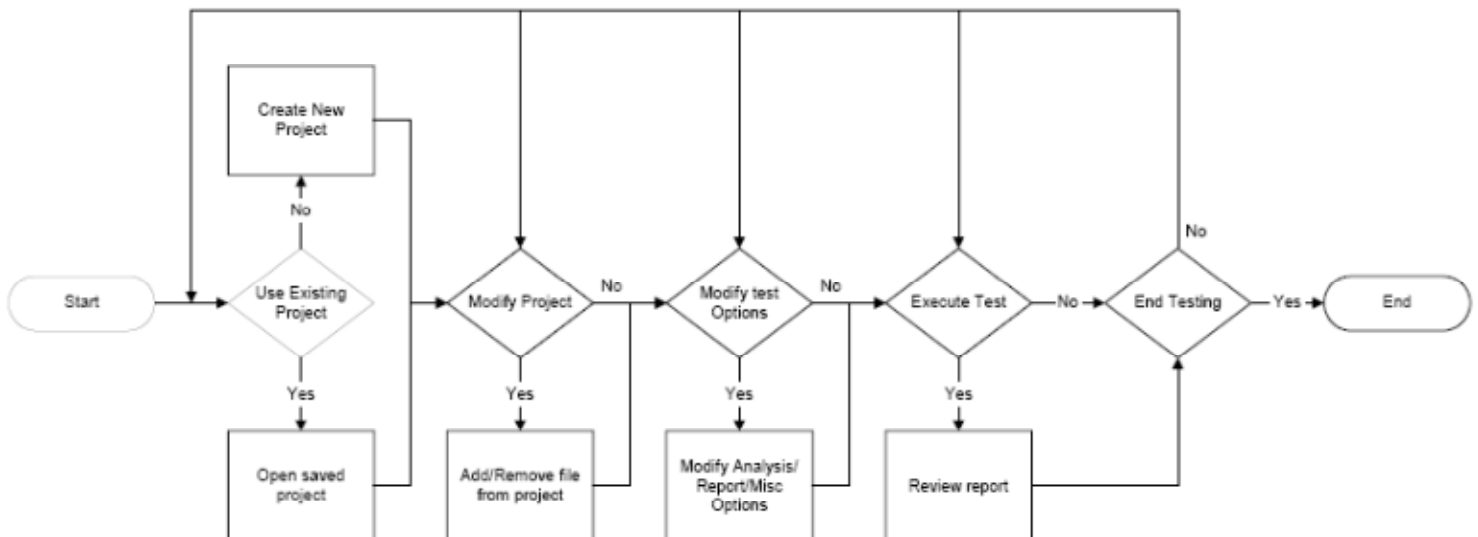
- a. Check source files before they are compiled
- b. Isolate obscure problems
- c. Identify problems before debugging is required

C. Advantages

1. The diagnostic messages produced by Flint are more detailed than those produced by standard compilers, and cover a much wider range of syntactic and semantic problems.
2. Flint analyzes source files both individually and as a group, and can therefore identify problems that are beyond the scope of a compiler – especially the global (program) scope.
3. Flint is effective in reducing development time and improves Fortran programming style.
4. Analysis and other report results are hyperlinked from the message to the related line of source using your favorite editor.
5. Cleanscape exclusive report content enhances your ability to comprehend and manage your program. These reports include call trees, cross reference for the entire program, USE trees, and include trees.

D. Flow of Analysis

1. The following flowchart illustrates the Flint test process:



PART II Requirements, Installation, and Uninstallation

Using FortranLint is subject to the terms and conditions contained in [shrinkwrap_license.pdf](#), located in the 'doc' subdirectory. If you do not agree to the terms of that agreement, discontinue use of the associated software product immediately and contact sales@cleanscape.net within 15 days of purchase to arrange for return.

2.1 WINDOWS

A. System Requirements

1. Hardware

Any configuration sufficient to run Windows is sufficient for Flint.

2. Operating System

- a. Microsoft Windows 10
- b. Microsoft Windows 8
- c. Microsoft Windows 7
- d. Legacy support: Microsoft Windows Vista®, Microsoft Windows XP® SP2, Microsoft Windows 2000® SP2, Microsoft Windows NT® 4.0 SP6a, Microsoft Windows 98® and 98® SE

3. Web Browsers

- a. Chrome® and Chromium-based browsers
- b. Firefox®
- c. Opera®
- d. Microsoft Edge or Internet Explorer®

B. Software Setup Procedure

1. Installation

- a) Download `flintgui<ver>_win.exe` to a temporary directory, then run it.
- b) An installer window will appear and extract a number of files to the installation directory you specify (hereinafter referred to as `<install_dir>`; the default is `c:\cleanscape\flint`). The installer exits automatically, and no reboot is required, though you must close/reopen any command prompts.

NEW v7.1 The installer will upgrade you to x64 versions of key executables if applicable.

- c) The installer automatically creates a shortcut for the Flint GUI on the desktop. To run the GUI, double-click the shortcut follow the instructions to obtain a license key as described in Section 3.
- d) The installer searches for recent versions of Visual Studio on your machine and if found, automatically installs the Cleanscape tools. All installers are available in `<install_dir>\ideinstall`

Care has been taken to allow users who are not logged in as "owner" to successfully install the Visual Studio tools; if you encounter problems, reinstall as owner (or have your Administrator install) and notify [us](#).

NOTE: Automatic installation is not possible on Windows 98; if you are a Win98 IDE user, please contact support@cleanscape.net.

- e) Finally, the installer adds the `main` subdirectory to your system PATH – necessary for running Flint (or any of its associated support programs) from the command line. To do this manually, enter the following command:
- ```
set PATH=<install_dir>\main;%PATH%
```

## 2. Additional steps for Windows 2000+ user privileges / access control

If you're installing Flint under Windows 2000 or later as Administrator, and you want to make the program accessible to ordinary Users, some additional steps are required. For more information, see Section 6.1.

## C. Uninstallation

**NOTE:** You will need owner privileges if that is how the product was installed.

### 1. Manual uninstallation

- a) Delete the installation directory and its subdirectories.
- b) Delete the Flint GUI icon from the desktop
- c) Remove environment variable `FLINTHOME` and the Cleanscape directory from your PATH:
  - In Windows 98, delete the appropriate “set flinthome=” and “set path=” statements from your `c:\autoexec.bat` file.
  - In Windows NT or later,
    - right click your “My Computer” icon on the desktop, then select “Properties”
    - click the “Advanced” tab or click “Advanced System Settings”
    - click the “Environment Variables” button and in the System Variables section:
      - \$ delete the `FLINTHOME` line
      - \$ double-click the text field “Path” in the System Variables area, and from that string, delete `<install_dir>\main`
- d) If Microsoft Visual Studio is installed on your machine, tools were automatically integrated upon Flint installation. Delete these tools using the following steps:
  - Open your Visual Studio IDE.
  - Select the Tools dropdown menu.
  - Select “External Tools...” (for VS 6, select “Customize...”, then click the “Tools” tab in the dialog).
  - Click on each Cleanscape tool in turn, then click on the Delete button (for VS 6, delete each tool by clicking the red **X** in the top right).

### 2. Restore your system to the point just before Flint installation – not available for Windows NT/2k

The installer created a system restore point just prior to installation. If you have not added new programs in the interim, you can safely roll your system back to this point. For Win98, use `scanreg /restore`



## 2.2 UNIX/LINUX

### A. System Requirements

#### 1. Hardware

A minimum of 256 MB memory is required for Flint.

#### 2. Operating System. Note the GUI version may differ amongst the various hosts.

- a. Most GNU/Linux OSes, including RedHat®, SuSE®, Debian®, Ubuntu®
- b. Mac OS-X® Mojave (backwards compatible to Tiger)
- c. HP HP-UX®
- d. IBM AIX®
- e. SGI Irix®
- f. Sun Solaris®
- g. FreeBSD

#### 3. Web Browsers

- a. Firefox®
- b. Opera®
- c. Seamonkey®
- d. Mozilla® or Netscape Navigator®

### B. Software Setup Procedure

Installation – installation as root is easier and recommended. Refer to the installation notes for details. The ‘#’ below represents the root prompt.

- a) Download the latest version of flintgui<ver>\_<OS>.taz to a temporary directory, e.g., /tmp.

**NEW v7.4** There is a 64-bit version of Flint for Linux and Mac.

- b) Create installation directory, e.g., /usr/local/cleanscape, and cd to it.

- c) Use the following command to extract the files.

```
tar xzpvf /tmp/flintgui<ver>_<OS>.tar
```

(For non-gnu tar, one would first run gunzip, then tar xpvf)

- d) If you are running the demo version of the product (Linux or Mac), no key is necessary. Otherwise, obtain and install a license key as described in Section 3.

- e) If this is a server-based application, start the daemon on the server as root:

```
startup
```

**NOTE:** The daemon must be running on the server before clients can access/ use the product.

- f) If you intend to run Cleanscape Flint from the command line, be sure you have started the daemon as described in step e). These additional commands are required (examples below are for sh/bash):

```
export CSIAPPPBASE=<install_dir>
export FLINTHOME=$CSIAPPPBASE/flintgui.dir/main
export PATH=$FLINTHOME:$PATH
```

### C. Uninstallation – manual process

- a) Delete the installation directory and its subdirectories.

- b) Delete files `myeditor.lst` and `ftemplate.csi` from all users' `$HOME` directories.**

## PART III Activating Flint

**Note1:** If you are a demo user (Windows/Linux/Mac), *no key is necessary; proceed to page 12.*

**Note 2:** The license managers are different between Windows and Unix/Linux.

### A. Registration Process – Windows

You should have received a temporary key good for 30 days (the warranty period); if not, contact [sales@cleanscape.net](mailto:sales@cleanscape.net).

For most users, you simply copy the keyfile to %FLINTHOME%. Detailed instructions are provided in the email with the keyfile, as well as instructions for obtaining a permanent key.

There is practically no license management for single user licenses; floating license management is described in detail in doc\readme\_floating.txt.

### B. Registration Process – Linux/Unix

License registration is accomplished via the command line. You should have received a temporary key good for 30 days (the warranty period); if you didn't receive a tempkey, email [sales@cleanscape.net](mailto:sales@cleanscape.net).

0. Remember to set up the environment variables per the instructions in Section 2.2.B.f above.

1. Run the command, `flint activate`

Hit <Enter> to leave the number of license servers at its default of 1.

The next line from the activation program will contain your server code. Save this if you are requesting a permanent key (next step).

Enter your temporary key when prompted. Flint is registered and operational.

2. Run the command, `startup`

This starts the license manager daemon. Note: `startup` should be added to your server's initialization so it will restart with every server reboot; for assistance, contact [support@cleanscape.net](mailto:support@cleanscape.net).

3. To obtain your permanent key, email [sales@cleanscape.net](mailto:sales@cleanscape.net) and provide the server code displayed during step 1.

The next three sections describe in detail the operation of Flint

- from the GUI [Part IV](#)
- from the command line [Part V](#)
- integrated within IDEs [Part VI](#)

## PART IV Running the Flint GUI

### A. Overview

The Cleanscape GUI is a tried-and-true graphical interface used successfully for years. It is also the interface for our C/C++ offerings, and the planned interface for Java analyzers and test tools.

The Cleanscape GUI provides hyperlinking between the various reports (in the Reports frame) and the line of source in the source file that caused the message, using your favorite editor.

Advantages of the Cleanscape GUI include:

- Fast
- Easy to learn, navigate, and use
- Information readily at the programmer's fingertips
- Point-and-click control for options-laden Flint command-line product.
- Access code at the relevant point using your favorite editor!

**NEW v7.0** The GUI has two new reports: USE tree, which displays the USE hierarchy for the selected files, and include tree for both the Fortran INCLUDE line and #include directives. For details, see Sections 4.B and 4.H below.

Supported code editors are listed below. It is also possible for users to integrate their own editor; see Section 7.2 for details on the `seteditor` program. For user-contributed editors, visit [http://www.cleanscape.net/products/contributed\\_editors.html](http://www.cleanscape.net/products/contributed_editors.html).

#### *Windows editors:*

- |                               |             |                      |
|-------------------------------|-------------|----------------------|
| • Crimson Editor              | • GVim      | • UltraEdit          |
| • Emacs                       | • Notepad++ | • Visual Studio *    |
| • Epsilon Programmer's Editor | • Slickedit | • Visual Studio Code |

#### *Unix/Linux editors:*

- |         |          |          |          |
|---------|----------|----------|----------|
| • Elvis | • Joe *  | • Pico * | • Xemacs |
| • Emacs | • Nano * | • Vi *   |          |
| • Jed * | • Nedit  | • Vim *  |          |

\* Multiple instances of these editors will open with each link click.

All elements of the GUI are also controllable from the keyboard; this is discussed in [Section K](#) below.

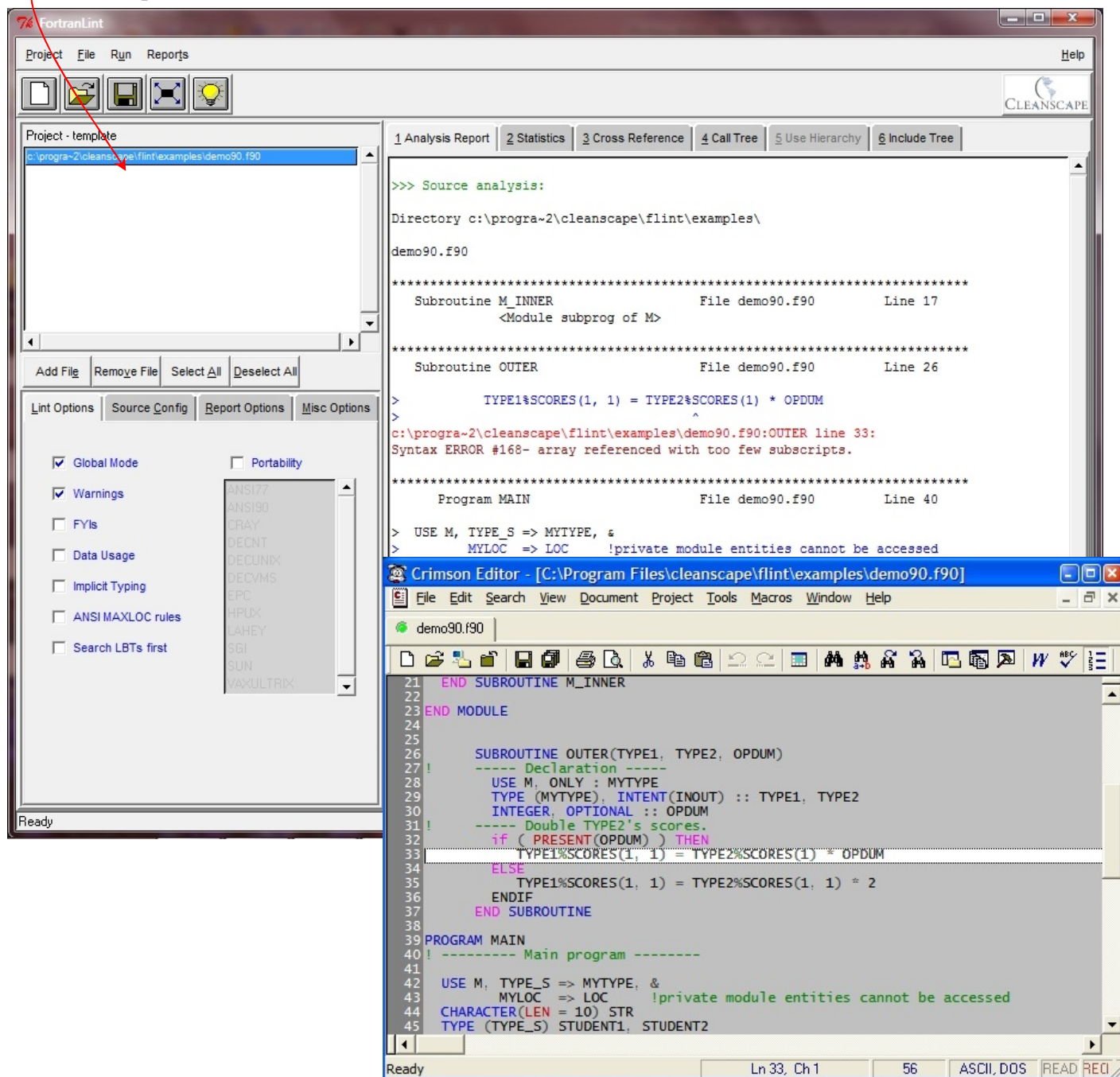
The following screenshots depict a sample Flint session.

**NOTE:** A sample Unix/Linux screen shot is shown in addition to one from Windows. All subsequent screen shots are Windows-based, but the functionality is identical between the two environments.

The Crimson Editor (previously selected as shown in the Reports tab in the lower left frame) was activated when the red line containing “33” in the Analysis Report was left-mouse-clicked. Flint positioned the editor to the line in the source file that caused the analysis result.

It is possible to open any file listed in the Project window (upper left frame of the GUI) by right-mouse-clicking on the desired filename.

The Flint GUI remembers settings (e.g., checkboxes, include path, external editor – but not filenames) from the previous session by creating a template file in the bin subdirectory or your \$HOME directory on Unix/Linux. There is no template file upon installation.



X-Deep/32 Root Window (:0 SW Mode)

Project File Run Reports Help

Project - ftemplate

/usr/local/cleanscape/flintgui.dir/examples/demo90.f90

/usr/local/cleanscape/flintgui.dir/examples/demo90.f90

Add File Remove File Select All Deselect All

Lint Analysis Source Config Reports Misc Options

Global Mode

Warnings

FYIs

Data-flow analysis

Data Usage

Implicit Typing

ANSI MAXLOC rules

Search LBTs first

Portability

ANSI77

ANSI90

CRAY

DECNT

DECUNIX

DECVMS

EPC

HPUX

LAHEY

SGI

SUN

VAXULTRIX

1 Analysis Report 2 Statistics 3 Cross Reference 4 Call Tree

>>> Source analysis:

Directory /usr/local/cleanscape/flintgui.dir/examples/  
demo90.f90

\*\*\*\*\*

Subroutine M\_INNER File demo90.f90 Line 16  
<Module subprog of M>

\*\*\*\*\*

Subroutine OUTER File demo90.f90 Line 25

>

TYPE1%SCORES(1, 1) = TYPE2%SCORES(1) \* OPDUM

.../cleanscape/flintgui.dir/examples/demo90.f90:OUTER line 32:  
SYNTAX ERROR #168- array referenced with too few subscripts.

>

TYPE1%SCORES(1, 1) = TYPE2%SCORES(1) \* OPDUM

.../cleanscape/flintgui.dir/examples/demo90.f90:OUTER line 32:  
PORT ERROR #456- ANSI-F90 does not allow an array to be referenced with  
too few subscripts.

\*\*\*\*\*

Program MAIN File demo90.f90 Line 38

> USE M, TYPE\_S => MYTYPE, &

> MYLOC => LOC !private module entities cannot be accessed

>

.../cleanscape/flintgui.dir/examples/demo90.f90:MAIN line 42:  
SYNTAX ERROR #661- entity not accessible in module M.

> AVE = MAIN\_INNER( STUDENT1%SCORES )

.../cleanscape/flintgui.dir/examples/demo90.f90:MAIN line 49:  
INTERFACE ERROR #252- I\*4 array passed to dummy arg which is a R\*4 array.

.../cleanscape/flintgui.dir/examples/demo90.f90:MAIN line 46:  
USAGE ERROR #126- local variable STUDENT2 is referenced but never set.

.../cleanscape/flintgui.dir/examples/demo90.f90:MAIN line 48:  
USAGE WARNING #127- local variable STR is set but never referenced.

\*\*\*\*\*

Function MAIN\_INNER File demo90.f90 Line 53  
<Internal subprog of MAIN>

.....

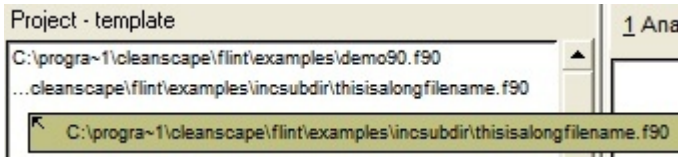
Ready

## B. Components

Where possible, each component features “balloon” help which will appear if you hover the mouse over an item or control description. Additional help for each item may be found in the Online Help (see [Section 4.I](#)).

1. Program menu: 

2. Shortcut bar: 

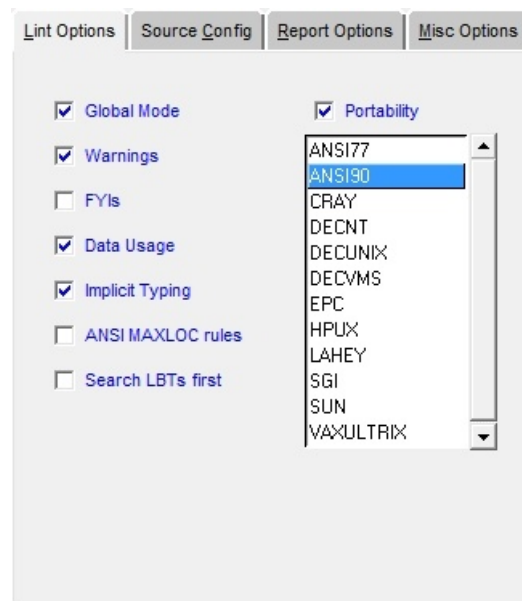
3. Project window: 

Any file listed in the Project window can be opened in the selected editor by right-mouse-clicking the filename. Any filenames too long to fit the window are shortened to ~60 characters and an ellipsis is prepended. The full filename appears in a balloon tip if hovering the mouse over the name, as shown above.

4. Project shortcut buttons: 

5. Lint Options tab (with “Portability” listbox activated). Flint provides 12 portability options to help determine issues porting your code to different hosts. The ANSI77 and ANSI90 options are the most commonly used.

This kind of point-and-click control makes using the options-laden Flint command-line product easy! **NOTE:** Your suggestions to improve this feature are appreciated – email suggestions to [sales@cleanscape.net](mailto:sales@cleanscape.net).





6. Source Config tab. "Dialect" is analogous to "Portability"; an example in English is, "Tell Flint that the incoming source was written for a Solaris compiler (dialect) and I want to know issues porting to a Lahey compiler (portability)". Also note the preprocessor option: if checked, Flint will search for `cpp` in your PATH; you can define a new path and/or preprocessor name (e.g., the `fpp` that came with your Fortran compiler) in the textbox at bottom.

The location text box will also accept directives preceded by a dash, passing them on unchanged to the preprocessor.

The screenshot shows the 'Source Config' tab with the following options:

- ☐ Debug Lines
- ☐ 132 Columns
- ☐ HPF directives
- ☒ Preprocessor
- ☐ Two-byte INTEGERS
- ☐ Ignore INCLUDE paths
- ☐ Ignore VMS logicals
- ☐ OpenMP directives

Language: Automatic (dropdown)  
Dialect: Automatic (dropdown)  
Source Format: Automatic (dropdown)

"Include" directories: [Browse...]  
Preprocessor location, if not in PATH: [Locate...]

## 7. Report Options tab

Examples for all reports in Section 4.H below.

**NEW v7.0** Include Tree: display the inclusion of header files from either Fortran `INCLUDE` lines or preprocessor `#include` directives.

7.4 Release Note: The USE Report available in prior releases has been temporarily removed while under rewrite. If you need this report, contact [sales@cleanscape.net](mailto:sales@cleanscape.net).

The screenshot shows the 'Report Options' tab with the following options:

- ☒ Cross-Reference
- ☒ Call Tree
- ☒ Free Form
- ☒ Condense Tree
- ☐ Tabular
- ☐ Squash Tree
- ☐ Lower case
- ☐ Sort Tree
- ☐ INCLUDE Tree
- ☒ Trim Tree
- ☒ Statistics
- ☐ No Library
- ☐ Source Listing
- ☐ No Undefined
- ☒ Auto-load reports
- ☒ Auto-save reports

External Editor: Visual Studio Code (dropdown)  
Editor location: [Locate...]  
c:\progra~1\Microsoft VS Code

## 8. Miscellaneous Options tab:

Preprocessor defines/undefines can be specified in the first two textboxes on this tab, separated by spaces.

Call tree root is the starting point of the call tree. Because Flint tracks all variables (even loop counters), cross reference filters can provide sophisticated control of what can rapidly become a very large cross reference. See Sections 7.4.1 and 8.3 of *flintman.pdf* (located in the 'doc' subdirectory) for details on these two features.

Individual analyses can be enabled/disabled by number in the appropriate text boxes on this tab.

Dataflow analysis can help identify problems with initialization, improper sequencing of set/reference instances, and identifying dead or wasteful code.

For example, with this option on, Fortran-lint will inspect both branches of an IF-THEN-ELSE conditional to determine if a variable has been initialized for both branches, and whether a variable has been set before referenced in either branch.

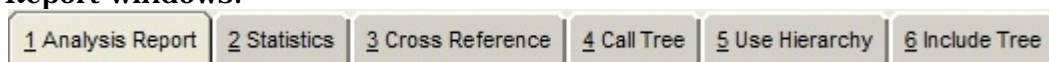
Using this option will add processing time to the analysis. The amount of extra time is determined by the complexity of the source, i.e., nested loops, IF blocks or excessive use of GOTOs. Indeed, some runs could take years! So dataflow analysis should be used sparingly and then only by senior programming staff.

### NEW v7.0

The Component Test option allows users to test a subset of their sourcebase within the context of their entire program. When checked, external program *usescan* (see Section 7.3) is invoked to resolve all USE statement targets, resulting in a minimum set of files required to fully test the UUT.

The screenshot shows the 'Misc Options' tab in the Flint software interface. It contains several input fields and checkboxes for configuring the analysis. The fields are labeled 'Define symbols:', 'Undefine symbols:', 'Call Tree Roots:', 'Cross Reference Filters:', 'Disable these messages:', and 'Enable these messages:'. The 'Cross Reference Filters' field contains the text 'no.unreferenced.parameters no.unused.common.variable:'. The 'Disable these messages:' field contains the text '76 207 261 176 531'. Below these fields, there is a red warning message: 'Below are ADVANCED options not to be used every run!'. At the bottom, there are two checkboxes: 'Data-flow analysis' and 'Component Test', both of which are currently unchecked.

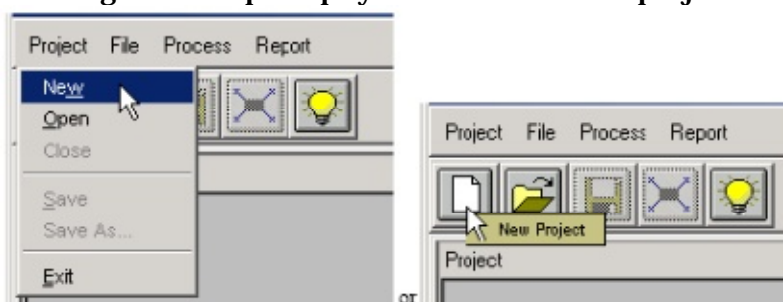
## 9. Report windows:



Example reports appear in [Section H](#) below.

### C. Creating a new project

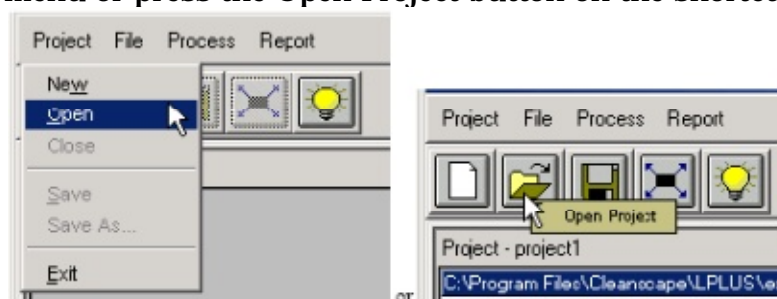
1. To create a new project, select Project/New from the menu or press the New Project button on the shortcut bar. Note: If a project is already open, a dialog box will prompt you to save the old project first.



2. A new project name appears in the title, which can be saved to any desired name later.

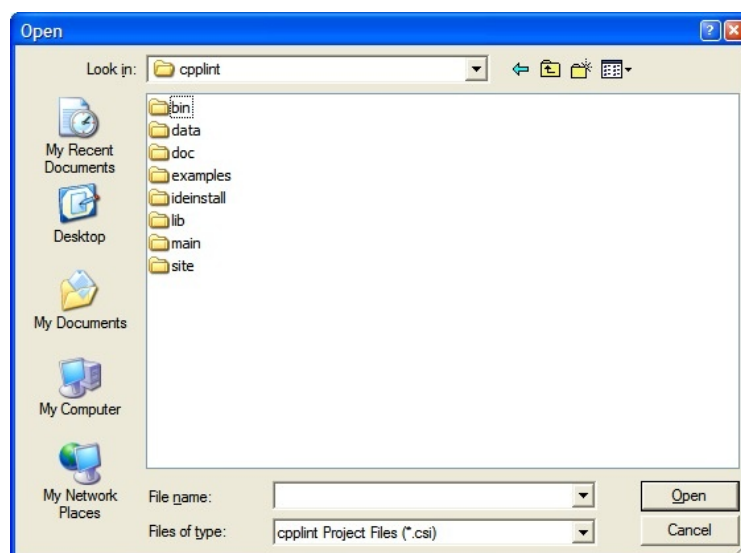
### D. Opening an existing project

1. To open an existing Cleanscape GUI project, select Project/Open from the menu or press the Open Project button on the shortcut bar:

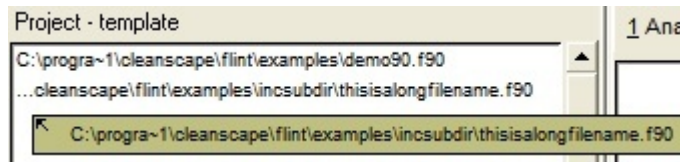


2. A standard Open dialog box will appear:

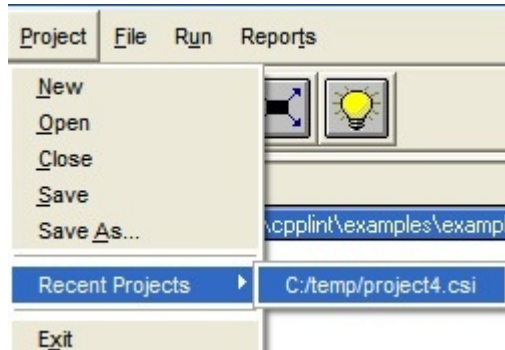
- a. Browse to find/select a project file (with extension .csi).
- b. When ready, press the Open button in the lower right corner.



3. Files associated with the project are displayed in the Project window:

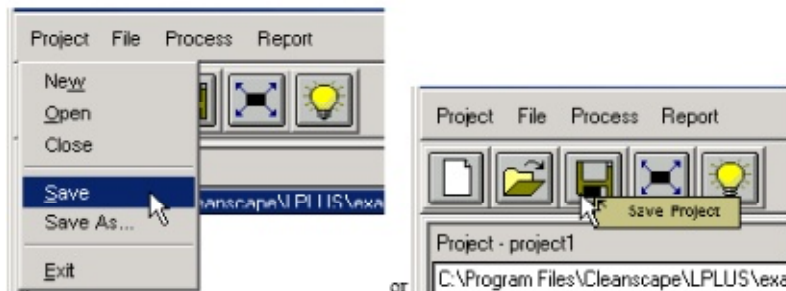


4. It is also possible to open recent projects using the Recent Projects menu:



#### *E. Saving a project*

1. To save the current state of a project, select Project/Save from the menu or press the Save Project button on the shortcut bar:

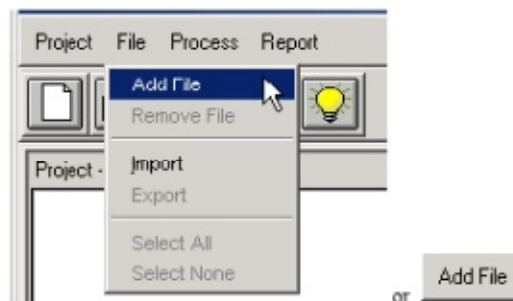


2. If this is a new project, the Save As window will appear.
  - a. Enter a name for the project.
  - b. When done, press the Save button.
  - c. You can also use the "Save As..." feature in the Project dropdown to save an existing project under a new name.

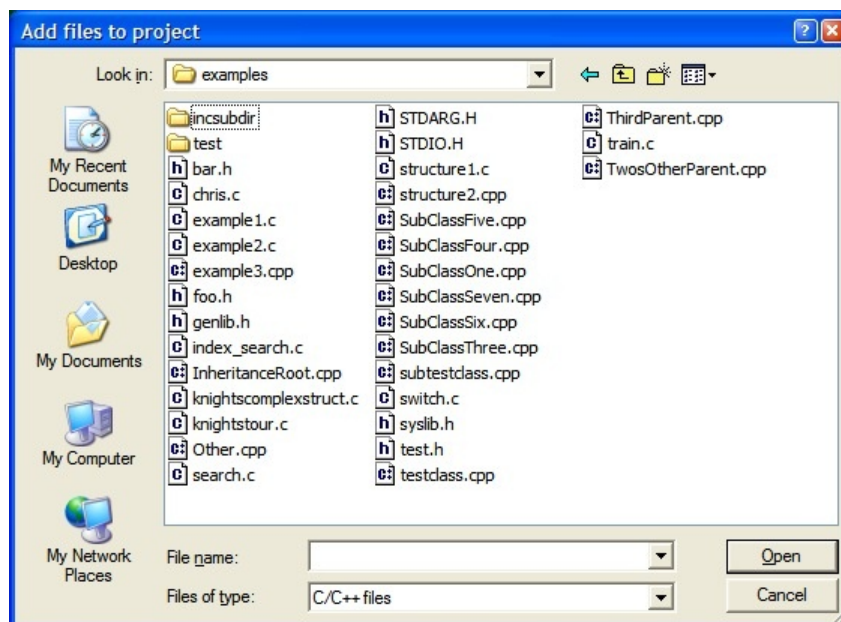
## F. Modifying a project

### 1. Add files to a project

- a. To add one or more files to a project, select File/Add File from the menu to add files into the project or press the Add File button on the project shortcut bar:



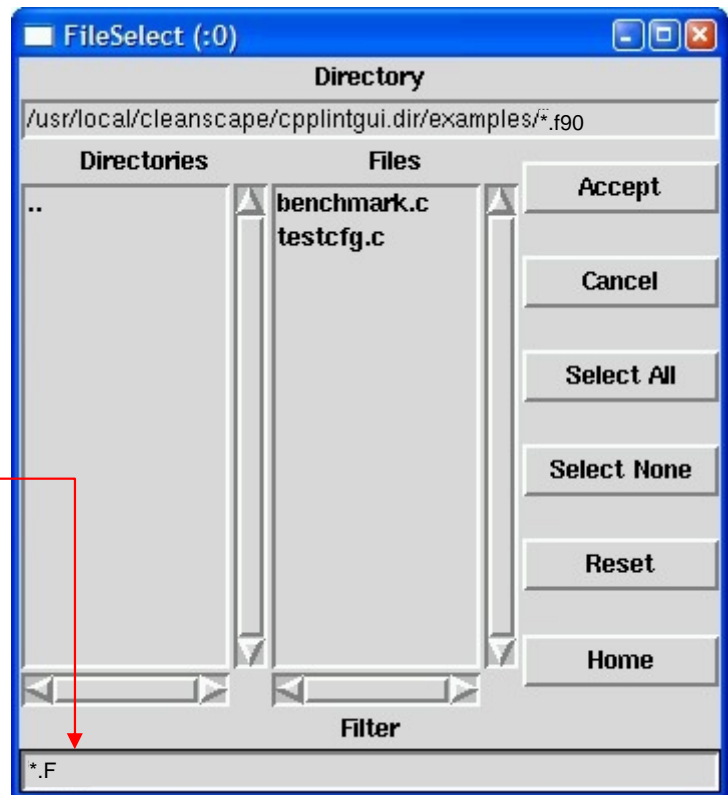
- b. The Add file window will appear:



- c. For the Flint GUI, Fortran source files will be the default file type (.f, .f90).

- d. **UNIX NOTE:** The default file type is .f90, which can be modified by entering the appropriate type (e.g., \*.F) in the Filter textbox at the bottom of the dialog. It is also possible to permanently modify the filter type by editing the "Default Add File filter" line in text file

~/cpplint.ini.

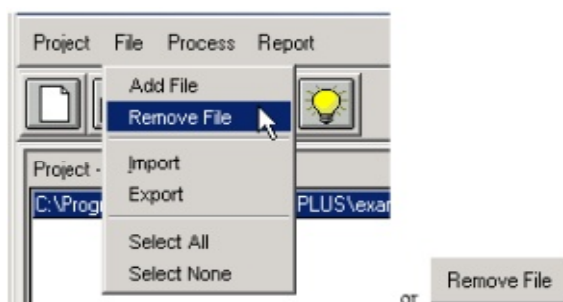


e. Multi-file selection:

- 1) The file-selection dialog supports multiple-file selection under both MS-Windows and \*nix.
- 2) To add multiple files individually, use <Control> + Left Mouse Button. Each selected file will be highlighted.
- 3) To add a group of files:
  - (i) Left-click on the first file.
  - (ii) Hold down the <Shift> key.
  - (iii) Click the last file. The first, last, and all in-between will be highlighted.
  - (iv) When done, press the Open (Windows) or Accept (\*nix) button.

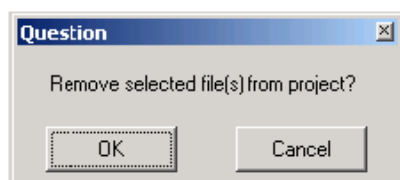
## 2. Removing files from a project

- a. To remove individual source files from a project, select the files to be removed, and then press the Remove File button. To remove all files from a project (i.e., to clear the file list), first press Select All, and then press the Remove File button.





- b. Press the OK button to confirm the removal operation:



- c. The updated file list is displayed in the project window.
- d. Note that this operation has no effect on the actual file on-disk.

#### G. Execute test

1. Create a new project or open an existing project for testing.

To create a new project, see [Section 4.C](#).

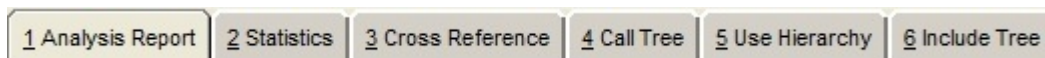
To open an existing project, [Section 4.D](#).

2. Select the files to be analyzed as explained in [Section 4.F.1](#).
3. Modify options as necessary, using the tabs in the lower left frame of the GUI, as displayed in [Sections 4.B.5-8](#). See balloon help, Online Help, and the [Flint Reference Manual](#).
4. To analyze the selected files, use Process/Analyze Files from the menu or press the Execute test button on the shortcut bar:



#### H. Review reports

1. To view the generated reports, click on the appropriate report tab:



2. To print reports, or to save them to disk, use the Report menu dropdown at the top of the screen. Reports may be printed or saved collectively or individually.
3. Samples of each of the reports are depicted below. Remember that clicking any entry in **red** will open the source file at the appropriate source line in the specified External Editor.

7.4 Release Note: The USE Report available in prior releases has been temporarily removed while under rewrite. If you need this report, contact [sales@cleanscape.net](mailto:sales@cleanscape.net).

|                   |              |                   |             |                 |                |
|-------------------|--------------|-------------------|-------------|-----------------|----------------|
| 1 Analysis Report | 2 Statistics | 3 Cross Reference | 4 Call Tree | 5 Use Hierarchy | 6 Include Tree |
|-------------------|--------------|-------------------|-------------|-----------------|----------------|

```

> TYPE1%SCORES(1, 1) = TYPE2%SCORES(1) * OPDUM
>
...\\examples\\incsubdir\\thisisalongfilename.f90:OUTER line 33:
PORT ERROR #456- ANSI-F90 does not allow an array to be referenced with
too few subscripts.

Program MAIN File thisisalongfilename.f90 Line 39
> USE M, TYPE_S => MYTYPE, &
> MYLOC => LOC !private module entities cannot be accessed
>
...\\examples\\incsubdir\\thisisalongfilename.f90:MAIN line 43:
SYNTAX ERROR #661- entity not accessible in module M.

> AVE = MAIN_INNER(STUDENT1%SCORES)
>
...\\examples\\incsubdir\\thisisalongfilename.f90:MAIN line 50:
INTERFACE ERROR #252- I*4 array passed to dummy arg which is a R*4 array.

...\\examples\\incsubdir\\thisisalongfilename.f90:MAIN line 47:
USAGE ERROR #126- local variable STUDENT2 is referenced but never set.

...\\examples\\incsubdir\\thisisalongfilename.f90:MAIN line 49:
USAGE WARNING #127- local variable STR is set but never referenced.

Function MAIN_INNER File thisisalongfilename.f90 Line 54
<Internal subprog of MAIN>

Global checking:

USAGE WARNING #743- module entity set but not referenced: M:AVE

```



1 Analysis Report 2 Statistics 3 Cross Reference 4 Call Tree 5 Use Hierarchy 6 Include Tree

```
>>> Statistics:
```

Number of source files: 1

Source files: 54 lines, 1262 bytes ( 18% comments, 82% code )  
 Include files: 14 lines, 352 bytes ( 5% comments, 95% code )  
 Total parsed: 68 lines, 1614 bytes ( 15% comments, 85% code )

Total subprograms: 5  
 Subroutines: 2  
 Functions: 1  
 Program: 1  
 Block Data: 0  
 Module: 1

Individual message summary

```

```

USAGE ERR #126- 2x: local variable \* is referenced but never set.  
 USAGE WARN #127- 1x: local variable \* is set but never referenced.  
 SYNTAX ERR #168- 1x: array referenced with too few subscripts.  
 INTRFC ERR #252- 1x: \* array passed to dummy arg which is a \* array.  
 PORT ERR #456- 1x: \* does not allow an array to be referenced with too  
 few subscripts.  
 USAGE WARN #509- 1x: array subscript is not integer data type.  
 SYNTAX ERR #661- 1x: entity not accessible in module \*.  
 USAGE ERR #742- 1x: module entity referenced but not set: \*, \*  
 USAGE WARN #743- 1x: module entity set but not referenced: \*, \*

Total messages: 10

|                | Errors | Warnings | FYIs   |
|----------------|--------|----------|--------|
| Syntax:        | 2      | 0        | <supp> |
| Interface:     | 1      | 0        | <supp> |
| Data usage:    | 3      | 3        | <supp> |
| ANSI-F90 port: | 1      | 0        | 0      |
| Lahey port:    | 1      | 0        | 0      |

1 Analysis Report 2 Statistics 3 Cross Reference 4 Call Tree 5 Use Hierarchy 6 Include Tree

This is a primary tree starting at the program 'PROC DAT'

```
PROC DAT--+-GETUNIT
 |
 +-READNAME
 |
 +-SETTYPE--PRINT (1)--PRINTIT--+-DIPSTAT---*PRINT*
 | |
 | +-GETUNIT
 |
 +-PRINT see 1
```

|                   |              |                   |             |                 |                |
|-------------------|--------------|-------------------|-------------|-----------------|----------------|
| 1 Analysis Report | 2 Statistics | 3 Cross Reference | 4 Call Tree | 5 Use Hierarchy | 6 Include Tree |
|-------------------|--------------|-------------------|-------------|-----------------|----------------|

```

*** Records:

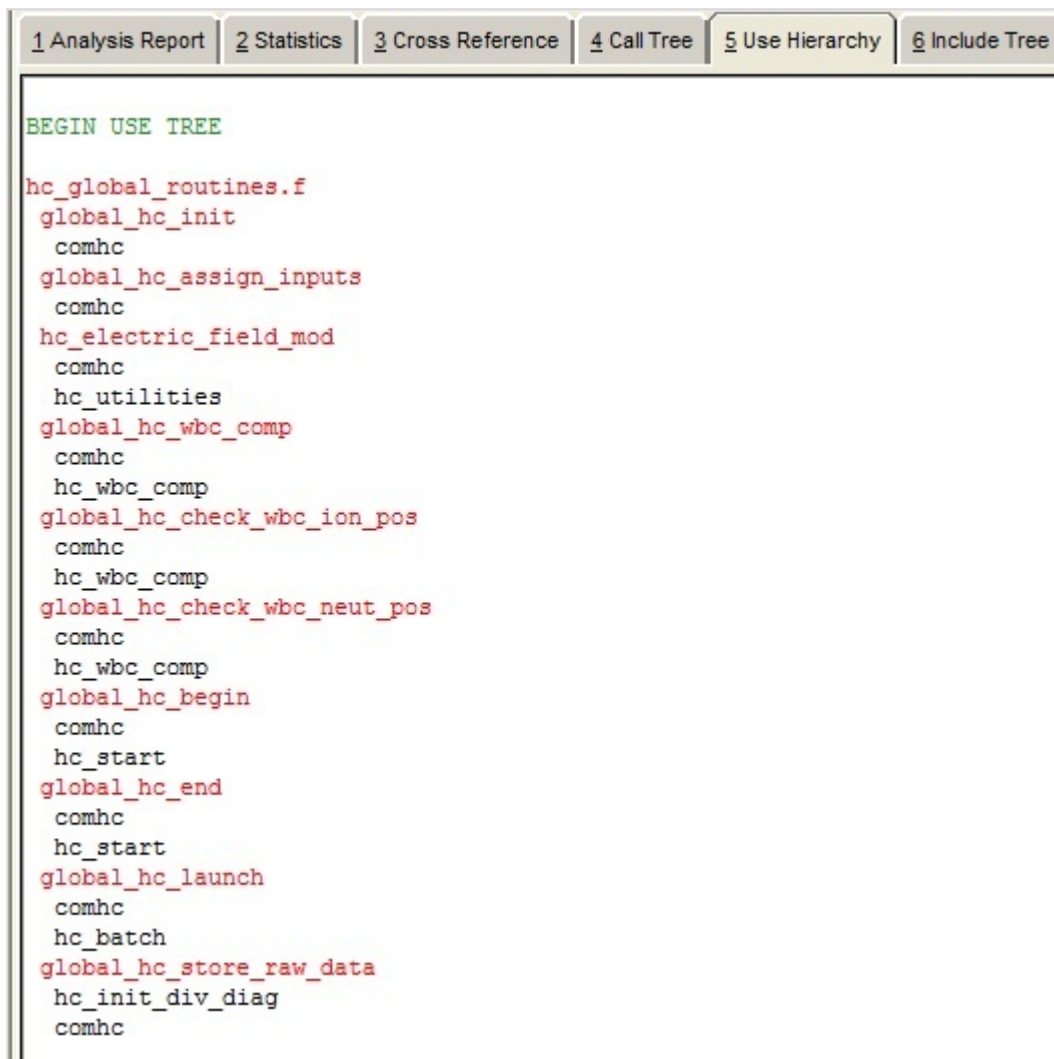
STUDENT1 : type TYPE_S : local
 in (demo90.f90:MAIN) is 44-D 46-SA 47-SA 49-RA
STUDENT2 : type TYPE_S : local
 in (demo90.f90:MAIN) is 44-D 46-RA 47-RA
TYPE1 : type MYTYPE : local
 in (demo90.f90:M::M_INNER) is 16-P 17-D 19-S
 in (demo90.f90:OUTER) is 25-P 28-D 32-S 34-S
TYPE2 : type MYTYPE : local
 in (demo90.f90:M::M_INNER) is 16-P 18-D 19-R
 in (demo90.f90:OUTER) is 25-P 28-D 32-R 34-R

*** Vars/Arrays:

AVE : I*4 : public entity of module M
 in (demo90.f90:M) is 10-D
 in (demo90.f90:MAIN) is 49-S
DUM (:,:) : R*4 : local
 in (demo90.f90:MAIN::MAIN_INNER) is (demo90.inc)3-P
 (demo90.inc)4-D
 (demo90.inc)6-RA
 (demo90.inc)7-RA
 (demo90.inc)9-R
FOO : R*4 : public entity of module M
 in (demo90.f90:M) is 9-RB
I : I*4 : local
 in (demo90.f90:MAIN::MAIN_INNER) is (demo90.inc)6-RS
 (demo90.inc)9-R
J : I*4 : local
 in (demo90.f90:MAIN::MAIN_INNER) is (demo90.inc)7-RS
 (demo90.inc)9-R
LOC (adj) : R*4 : private entity of module M
 in (demo90.f90:M) is 9-D
OPDUM : I*4 : local
 in (demo90.f90:OUTER) is 25-P 29-D 31-RA 32-R
STR : CHAR*10 : local

```

**NEW v7.0** Use Tree. Note that items in normal color (black in this case) are missing their module definitions and can't be hyperlinked. In such case, try adding Component Test analysis on the Misc. Options tab (see Section 4.B.8).



The screenshot shows the 'Use Tree' analysis report. The top navigation bar includes tabs for '1 Analysis Report', '2 Statistics', '3 Cross Reference', '4 Call Tree', '5 Use Hierarchy', and '6 Include Tree'. The main content area displays a hierarchical tree structure starting with 'BEGIN USE TREE'. The tree lists various components and their sub-components, with some items in red text indicating missing module definitions. The components listed are:

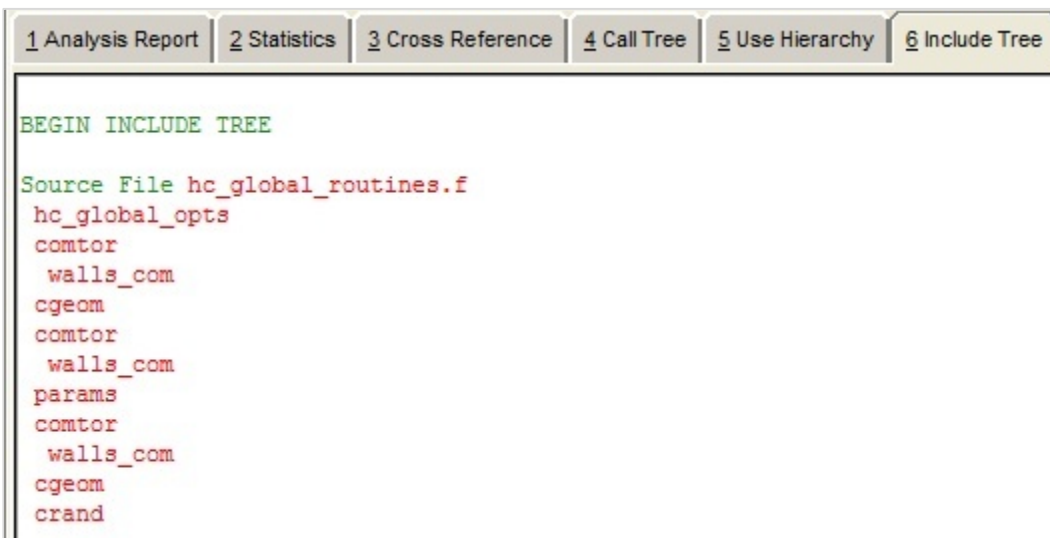
```

BEGIN USE TREE

hc_global_routines.f
 global_hc_init
 comhc
 global_hc_assign_inputs
 comhc
 hc_electric_field_mod
 comhc
 hc_utilities
 global_hc_wbc_comp
 comhc
 hc_wbc_comp
 global_hc_check_wbc_ion_pos
 comhc
 hc_wbc_comp
 global_hc_check_wbc_neut_pos
 comhc
 hc_wbc_comp
 global_hc_begin
 comhc
 hc_start
 global_hc_end
 comhc
 hc_start
 global_hc_launch
 comhc
 hc_batch
 global_hc_store_raw_data
 hc_init_div_diag
 comhc

```

**NEW v7.0** Include Tree. Flint scans both INCLUDE lines and #include directives.



The screenshot shows the 'Include Tree' analysis report. The top navigation bar includes tabs for '1 Analysis Report', '2 Statistics', '3 Cross Reference', '4 Call Tree', '5 Use Hierarchy', and '6 Include Tree'. The main content area displays a list of source files and their included components, starting with 'BEGIN INCLUDE TREE'. The components listed are:

```

BEGIN INCLUDE TREE

Source File hc_global_routines.f
 hc_global_opts
 comtor
 walls_com
 cgeom
 comtor
 walls_com
 params
 comtor
 walls_com
 cgeom
 crand

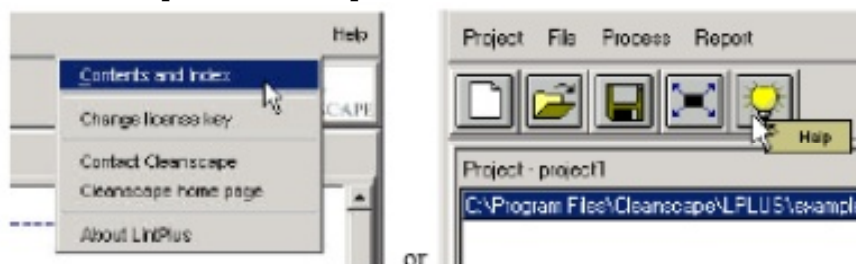
```

## I. Online Help

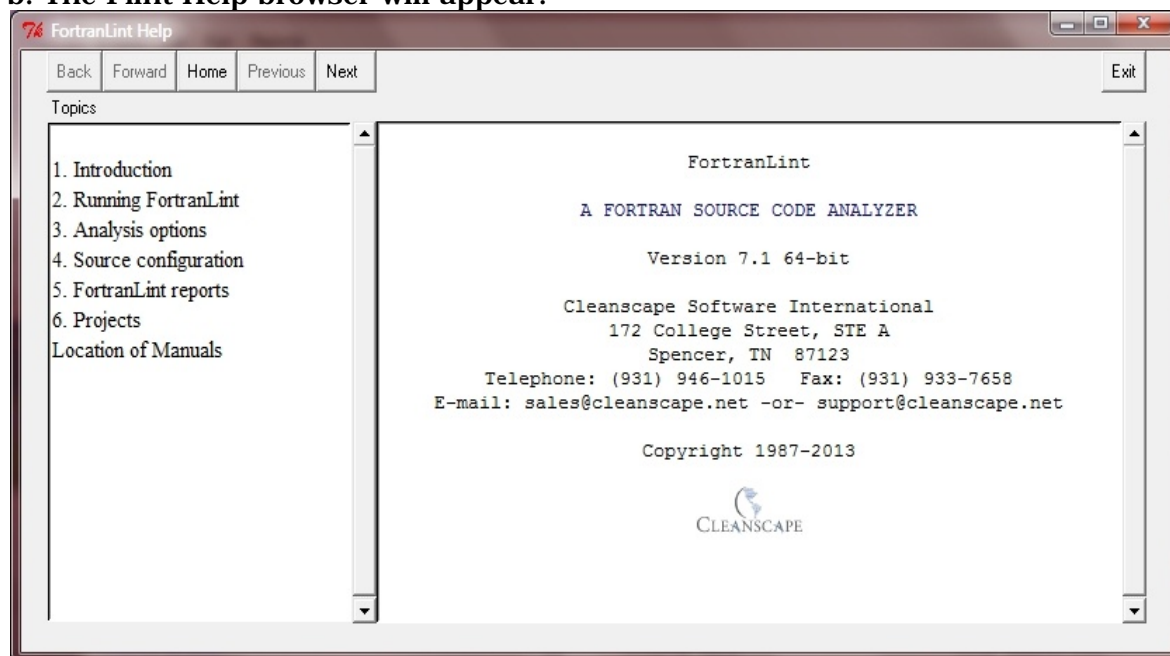
The Online Help System contains concise yet useful information for running the Cleanscape GUI. The Table of Contents and many interrelated items in the help text are hyperlinked to make information access quick and easy.

### 1. Accessing the Help System

- a. To access the online help system, select Help/Contents and Index from the menu or press the Help button:



- b. The Flint Help browser will appear:



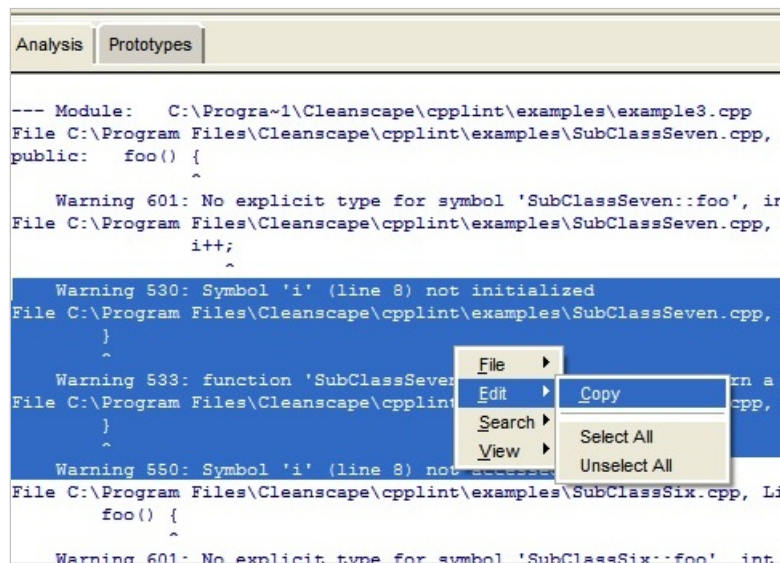
## J. Sub-Menu Functions

There are several “right-mouse-click” options available while in the Reports frame on the right hand side of the GUI. These features should be self-explanatory for those familiar with graphical environments. The more commonly used features are shown in detail below.

### 1. Copy

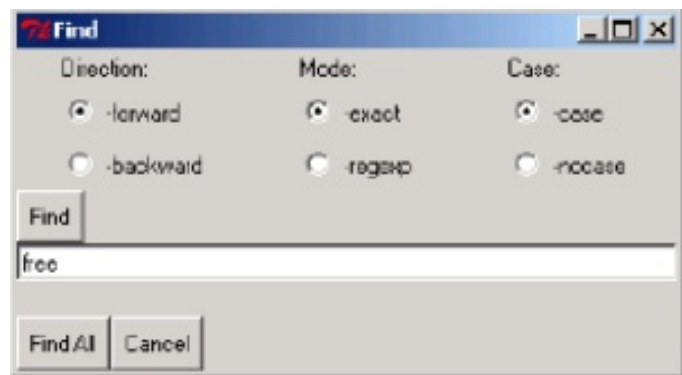
- a. Click and hold the left mouse button while dragging to select text.
- b. Press the right mouse button inside reports frame
- c. Select Edit -> Copy
- d. The text can now be pasted into other applications (e.g., Microsoft Word).





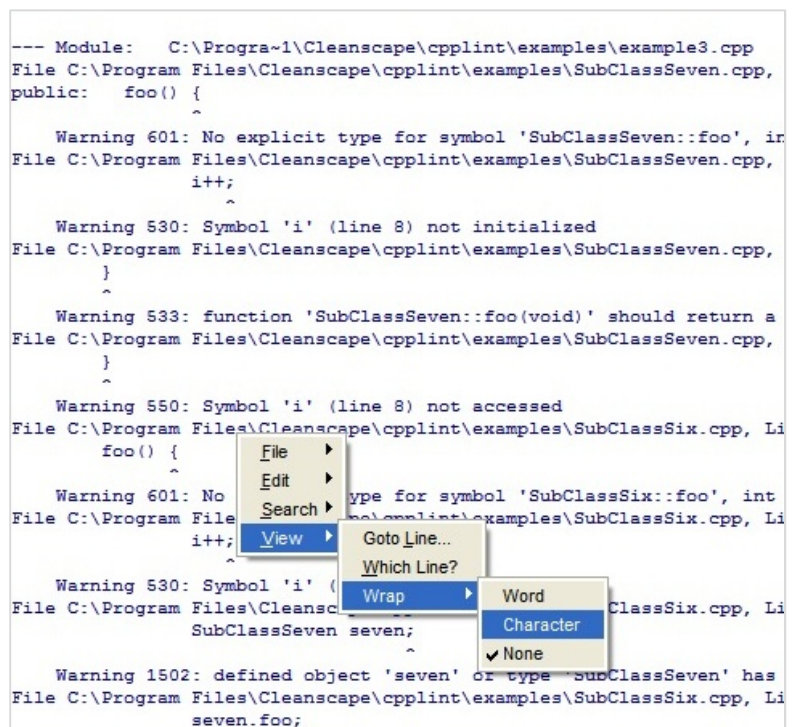
## 2. Search

- Press the right mouse button inside a report frame.
- Select Search -> Find.
- Enter string to search and select the desired options:
- The search result(s) will be highlighted.



## 3. Line Wrap

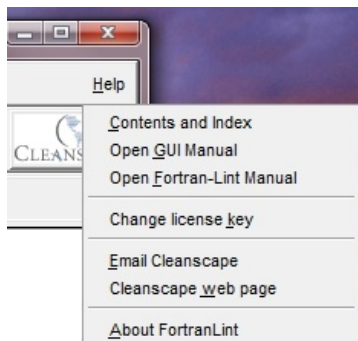
- Press the right mouse button inside a report frame
- Select View -> Wrap. The default is None.



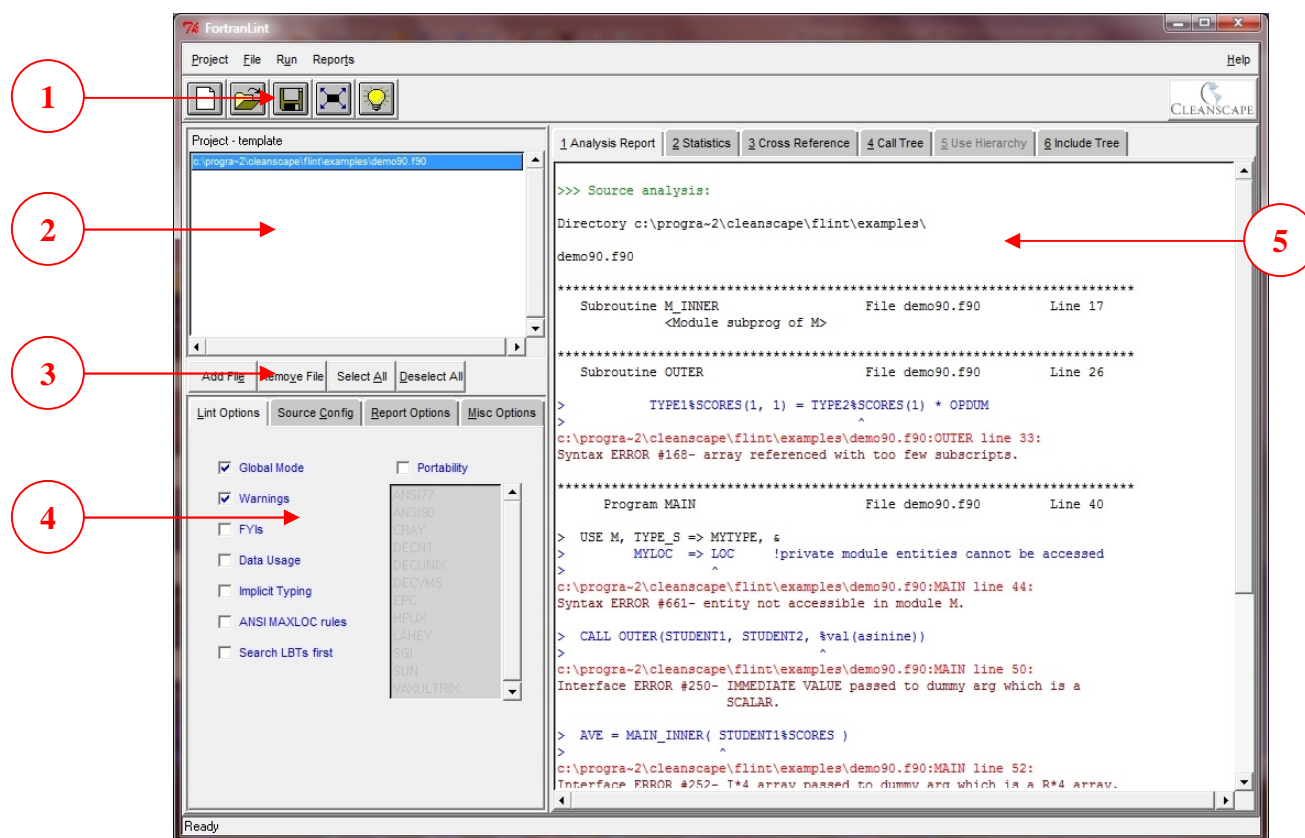
## K. Operating the GUI using the Keyboard; Keyboard Shortcuts

All aspects of the Flint GUI can be controlled from the keyboard. This capability was added to [comply](#) with the US Government's [Section 508](#) provisions.

1. Accessing dropdown menus and items using keyboard accelerators. This is the standard mode common to all Windows products.
  - a. Select the desired menu by holding down the <ALT> key, then pressing the underlined letter for that menu item. For instance, this screen image was obtained by pressing and holding <ALT>, then typing the "h" key:



- b. To open the GUI manual, release the <ALT> key and then press "g".
2. Navigating amongst screen elements. There are 5 screen elements in the GUI, as shown below:



- a. The <TAB> key scrolls between these five screen elements and all active items within each element. <SHIFT>+<TAB> reverses the scrolling. The

item with focus will have a dotted line around its border. *Note:* Because of the background color, the icon buttons in Element 1 will not show the dotted-line highlighting.

- b. For buttons (including radio buttons), pressing the space bar will “push” the button.
  - c. For checkboxes, pressing the space bar will “check/uncheck” the box.
  - d. For dropdown boxes, pressing the space bar will open the dropdown; the up/down arrows will navigate the dropdown, and the <ENTER> key will select.
3. Keyboard shortcuts.
- a. The standard Windows shortcuts are available. For instance, pressing <F1> will bring up the Help listing; <ALT>+<F4> exits the program.
  - b. Use the alt-key combination to access a menu, then type just the underlined letter to access a submenu item. For instance, to invoke Project-Save As, one would type <ALT>+<p>, then <a>. Alternately, the arrow keys can be used to navigate submenu selections once the menu dropdown has been activated with <ALT>+<p>.
  - c. The following keyboard shortcuts are also available within the GUI:

|           |                                                                       |
|-----------|-----------------------------------------------------------------------|
| <ALT>+<o> | <u>O</u> pen Project                                                  |
| <ALT>+<g> | R <u>u</u> n the Analysis ( <u>G</u> o)                               |
| <ALT>+<x> | <u>E</u> xit GUI                                                      |
| <ALT>+<l> | J <u>u</u> mp to <u>L</u> int analysis tab (in Element 4)             |
| <ALT>+<c> | J <u>u</u> mp to <u>S</u> ource <u>C</u> onfig tab (in Element 4)     |
| <ALT>+<r> | J <u>u</u> mp to <u>R</u> eports tab (in Element 4)                   |
| <ALT>+<m> | J <u>u</u> mp to <u>M</u> isc Options tab (in Element 4)              |
| <ALT>+<1> | J <u>u</u> mp to Report # <u>1</u> (Analysis report in Element 5)     |
| <ALT>+<2> | J <u>u</u> mp to Report # <u>2</u> (Statistics report in Element 5)   |
| <ALT>+<3> | J <u>u</u> mp to Report # <u>3</u> (Xref report in Element 5)         |
| <ALT>+<4> | J <u>u</u> mp to Report # <u>4</u> (Call tree report in Element 5)    |
| <ALT>+<5> | J <u>u</u> mp to Report # <u>5</u> (USE tree report in Element 5)     |
| <ALT>+<6> | J <u>u</u> mp to Report # <u>6</u> (Include tree report in Element 5) |

#### L. Changing fonts / sizes

To change the fonts and sizes used within the GUI, edit the text file `flint.ini` located in the `main` subdirectory on Windows or your `$HOME` directory if \*nix.

In that file, you will see a section starting with `[fonts]`. Change the values from default to a value specified as follows:

*name size style*

where *name* is any font name on your system (enclose in curly braces if spaced),  
*size* is an integer font size, and  
*style* is one of: `normal` `bold` `italic` `underline`

*Example:* `report text = {Lucida Console} 9 normal`

Specifying all three font characteristics is recommended.

**NOTES:** The GUI will attempt to substitute Helvetica 10 or Courier for invalid fonts. The case of the font name can be a factor.





## PART V Running Flint from the Command Line

### A. Introduction

Flint has a command line facility suitable for standalone operation or for inclusion in scripts, e.g., for “make lint” purposes.

For details on the actual operation of Flint and its control and reporting options, refer to the companion document, [Flint Reference Manual](#).

### B. Operation

To run Flint in command line mode, you need to have set the environment variables as defined in Section 2.1.B.1.d or 2.2.B.f and registered the product as described in [Section 3](#).

The format of the Flint command line is quite simple:

```
flint <parameters_to_be_supplied_to_PC_lint> <source_filename(s)>
```

Entering `flint` without parameters yields a command summary.

Details on all the command line parameters may be found starting in Chapter 3 of the [Flint Reference Manual](#).

Other important sections in the Flint reference manual include cross-reference format/content sub-options (Chapter 8), in which very finely honed cross-reference results may be obtained, and the Unix install guide (Appendix A), which also provides details on the license daemon (not used under Windows).

**NOTE 1:** USE and INCLUDE reports are available only from the Flint GUI, but they may then be saved to disk as text files; see Sections 4.B.5 and 4.H.2.

**NOTE 2:** If desired, a subset of the sourcebase can be analyzed within the context of the entire program using external program `usescan`. See Section 7.3.

### C. Return Codes

A return code of zero (0) indicates that Flint ran and ran successfully without encountering any source errors.

A return code >1 indicates that either

- There was a problem securing a valid license key to run the program, or
- There were one or more messages resulting from the Flint analysis over the source code.

A detailed description is of course available in the analysis report. If there was a problem starting the program or securing a key, contact [support@cleanscape.net](mailto:support@cleanscape.net). If you are under maintenance, you may also contact Cleanscape Support for questions regarding any analysis output message.

For more information Flint's return codes and their uses, see Section 6.5 of the [Flint Reference Manual](#).

## PART VI Running Flint from IDEs

### A. Overview

For customers who develop their code in Integrated Development Environments (IDEs), it is more useful to operate Flint from within the environment.

Advantages include:

- Flint obtains project information (especially the file list, include directories, and defines/undefines) – no need to re-specify.
- Deep integration: When applicable, the equivalent Flint command is derived from controls set within the project. For instance, setting Intel Visual Fortran's "Fixed Form Line Length" to 132 will enable Flint's `-e` option.
- Flint uses the output window of the IDE for results and, if available in the IDE, links to the source line using the IDE's internal editor.
- Fewer windows to shuffle between.
- **NOTE:** The number of tools varies between IDEs; see the Tools menu in your IDE for availability. Section C below is a full description of all tools.

**NEW v7.4** • When Project Setup is run, a GUI `.csi` control file is also created, meaning you can run Flint from the GUI (thus obtaining call tree and cross reference reports with hyperlinks from the GUI), while still having the option to run Flint's analysis and get results strictly within the IDE!

Current IDEs supported:

ü All Microsoft Visual Studio versions: VS6 → VS2015

ü Other IDEs may be supported via the External Editor option in the Cleanscape GUI (see Sec. 4.B.6).

**NEW v7.0** Eclipse integration is available. To obtain integration instructions or request deep integration for your IDE, email [sales@cleanscape.net](mailto:sales@cleanscape.net).

### B. Installation

**NOTE:** The installers will not operate in Windows 98! If you are a Win98 IDE user, please contact [support@cleanscape.net](mailto:support@cleanscape.net).

To run Flint from IDEs, you need to have registered and activated the product as described in Section 3.

Installation for purchased IDEs occurred automatically when you installed Flint. If you added a new version or an altogether new IDE, contact [support@cleanscape.net](mailto:support@cleanscape.net) for a simple manual process.

**NOTE:** Care has been taken to allow users who are not logged in as "owner" to successfully install the Visual Studio tools; if you encounter problems, install as owner (or have your Administrator install the product for you) and notify [support@cleanscape.net](mailto:support@cleanscape.net).

## C. Operation

The installer will place four “External Tools”, found in the “Tools” dropdown menu for your IDE.

In the titles for each tool, below as well as on the “Tools” dropdown menu, the underlined letter indicates the default “accelerator key” for that tool. Clicking ALT-T then the underlined accelerator key represents a shortcut to running that tool. For details on the actual operation of Fortran-lint and its control and reporting options, refer to the companion document, *flintman.pdf*, located in the ‘doc’ subdirectory.

### 1. Flint Project Setup - tool by Cleanscape

**NOTE:** This option must be run first to successfully analyze projects (tool #3).

This option does not run any analyses, but instead parses the IDE project file for the list of source files, include directories, defines/undefines, and other settings comprising the current project. These values are then placed in a Flint `<IDEproject>.cfg` file, created and maintained by this tool.

Rerun this tool each time the project changes (changes to the source file list, include directories, or defines). Because the `<IDEproject>.cfg` file will be rewritten each time the tool is run, do not edit the file!

**HINT:** Be sure to save your project before running this tool!

Separate from `<IDEproject>.cfg`, a `std.cfg` file is used to specify analysis control settings to Flint. `std.cfg` is never changed automatically, and is therefore the place to set Flint options on a per-project basis.

**NEW v7.4** A third file, `<IDEproject>.csi`, is also created. This file translates both `.cfg` files into a control file for the Flint GUI. In so doing, a complete view of your project is also available to the GUI, meaning that other reports more readily viewed and browsed (via hyperlinking) from the GUI, such as call tree and cross reference, are easily obtained.

Like the project file, this file will also be rewritten each Project Setup, so do not edit it! Instead, save any customizations to your `std.cfg` file. For assistance, contact [support@cleanscape.net](mailto:support@cleanscape.net). Just in case, a backup of the existing `.csi` file is made before a rewrite occurs.

All these files are stored in the Visual Studio project directory.

Rerun this tool each time the project contents change (e.g., when files are added or deleted). If a `std.cfg` file exists, its contents are preserved.

### 2. Flint Single File - tool by Cleanscape

Use this tool to run Flint over a single file in your project. Analysis options are read from file `std.cfg` located in the Visual Studio project directory.

Analysis results appear in the Output window. When available in your IDE, double-clicking on any error message will cause the IDE's built-in editor to jump to the line number in the source file related to the analysis message.

The system include directories for the IDE you're running are obtained automatically for each analysis – great if you have multiple versions installed or you don't have the INCLUDE environment variable set.

### 3. Flint Entire Project - tool by Cleanscape

Use this tool (after having set up the project by using tool #1 above) to run Flint over the *entire* project sourcebase. As with the single module tool (#2 above), analysis results appear in the Output window; when available in your IDE, double-clicking on any error message will cause the IDE's built-in editor to jump to the line number in the source file related to the analysis message.

The system include directories for the version of Visual Studio you're running are obtained automatically for each analysis – great if you have multiple versions installed or you don't have the INCLUDE environment variable set.

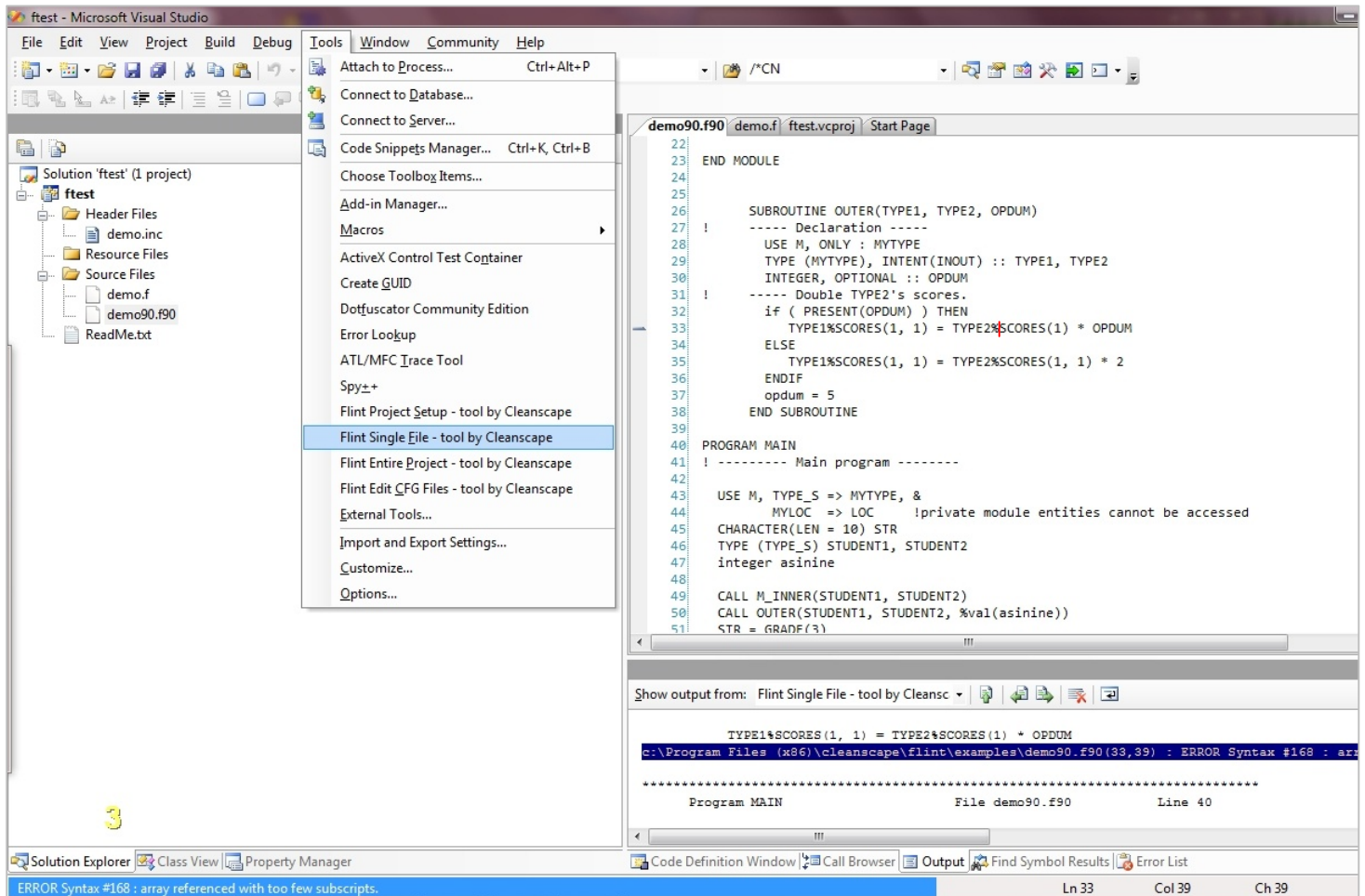
### 4. Flint Edit CFG Files - tool by Cleanscape

This tool conveniently invokes Notepad to edit the `std.cfg` and the `<IDEproject>.cfg` files (described in #2 above) associated with the current project. Do not edit the `<IDEproject>.cfg` file as it can be overwritten at any time; it is opened here for quick reference only.

Remember, it is possible to run Flint analysis from the GUI after running Project Setup (tool #1) above. By so doing, you can take advantage of other reports, like the cross reference and call tree, that are not as readily viewed in the IDE's output window. Plus, such reports are hyperlinked in the GUI, meaning you can click on any entry and the relevant line of source is presented in your favorite code editor! At present, Visual Studio's internal editor is opened when you click on a link in the GUI after using Project Setup; another editor can be chosen within the GUI, and any such choice is preserved later.

The screenshot below shows the Cleanscape tools installed in Visual Studio's Tools dropdown menu. It also shows the analysis results of the current project.

The highlighted ( **blue background** ) line in the Output window (bottom) was double clicked, resulting in Visual Studio's editor (top window) jumping to the source file / source line (line 33) that caused the analysis message.



#### D. Uninstallation

Manually remove any tool(s) from Visual Studio:

0. Open the Visual Studio IDE.

1. From the Tools dropdown menu, select "External Tools..." (in VS 6 select "Customize..." then click on the "Tools" tab in the resulting dialog).
2. Click the tool you wish to delete.
3. Click the "Delete" button in the right side of the dialog box (in VS6 click on the red 'X' in the upper right corner).



## PART VII MISCELLANEOUS INFORMATION

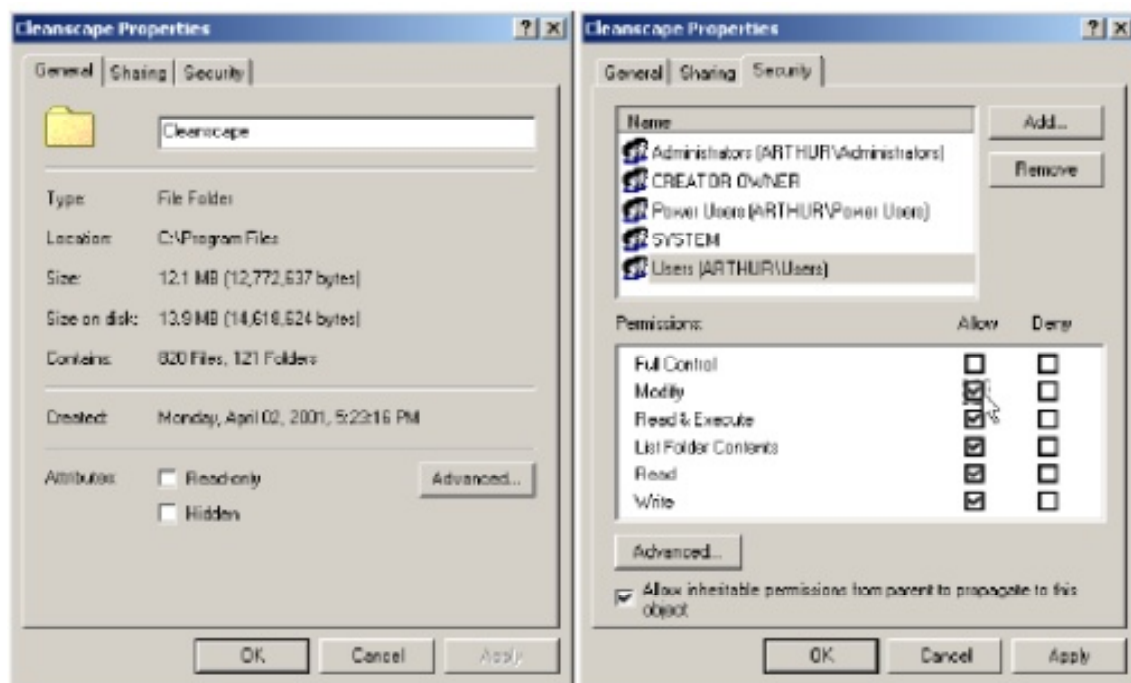
### 7.1 ADDITIONAL STEPS FOR WINDOWS 2000+

#### A. Applicability

1. This section applies to users running Windows 2000+ who belong to the “Users” group, and only to that group.

#### B. Details

1. For Flint to run correctly, users must have “write” and “modify” access rights to the installation directory and all its subdirectories.
  - a. Log in as “administrator” and finish installing Flint.
  - b. Double-click on the “My Computer” icon on the desktop.
  - c. Navigate to and double-click on the installation folder. Select Properties from the sub-menu.
  - d. Select “Security” tab on the Properties screen:



- e. Select the “Users” group and enable “Modify” and “Write” permissions.
- f. Click the “Apply” button.
- g. Click the “OK” button. This should close the Properties window.
- h. Flint is now ready to run on Win2k for the “Users” group.





## 7.2 ADDING AN EXTERNAL EDITOR TO THE GUI USING *SETEDITOR*

### A. Introduction

By popular demand, Cleanscape has added the ability for users to specify their own favorite editor to any Cleanscape product (as opposed to submitting a feature request to Cleanscape Support). This is implemented via an external program called *seteditor*, located in the 'bin' subdirectory.

User contributions welcome! Send them to [support@cleanscape.net](mailto:support@cleanscape.net); any contributions will receive appropriate credit and be placed in a "master" file located at [http://www.cleanscape.net/products/contributed\\_editors.html](http://www.cleanscape.net/products/contributed_editors.html).

### B. Operation

On any platform, it is possible to edit file *myeditor.lst* manually; see the comments inside the file, which is located in *bin* subdirectory on Windows or your *\$HOME* directory on Unix/Linux. The Unix/Linux session on the next page shows the contents of *myeditor.lst* (which looks substantially similar under Windows).

#### Windows.

You can either run *seteditor* from the command line or via Explorer.

From a DOS shell (cmd or command prompt), run the following command:

```
"<install_dir>\bin\seteditor"
```

From Explorer, navigate to the above directory and then double-click *seteditor.exe*.

#### Unix.

From a shell prompt, run the following command:

```
<install_dir>/bin/seteditor
```

Three pop-up dialogs (Windows) or a sequence of shell interactions (Unix/Linux) will guide you through

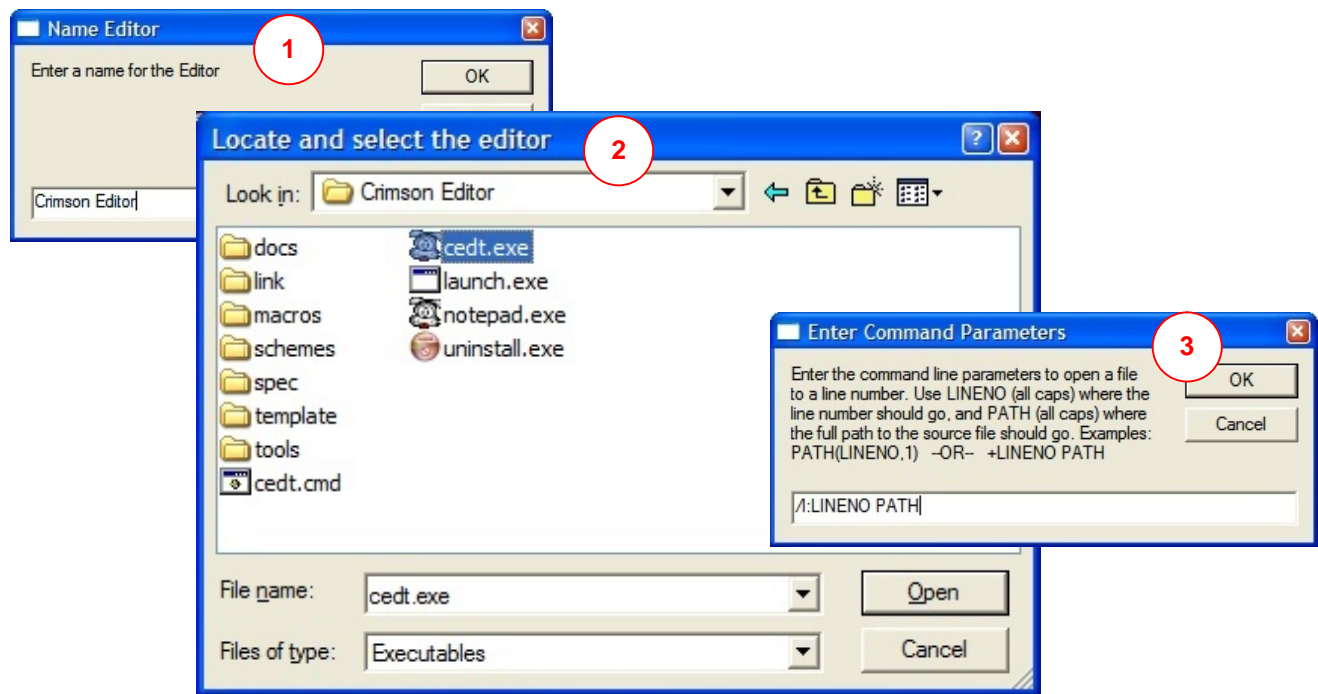
1. Naming the editor (a label identifier)
2. Locating the editor executable itself
3. Setting command line parameters to open a file and jump to a line number.

A sample Windows session depicting the dialogs for all three steps (and labeled as such) is shown on the next page, as is a Unix/Linux shell session.

**NOTE:** Refer to your editor's documentation to get the editor's command line information required (i.e., specifying the filename to open and the line number to jump to when opening the file). If your editor does not support jumping to line numbers from the command line, you can still invoke the editor but it will be impossible to align the analysis message to the "offending" source line.

We also recommend setting your editor to run as a single instance.

Any number of editors may be added in this fashion. Added file information is stored in file *myeditor.lst*; once successfully added, email your *myeditor.lst* file to [support@cleanscape.net](mailto:support@cleanscape.net) for inclusion in a Master file to share with other Cleanscape customers!



```
suse:/home/chris
suse:~$ /usr/local/cleanscape/bin/seteditor

This program adds an external editor to the Cleanscape GUI(s).
You will need to supply the command line switches for loading a file and
jumping to a line number. Enter 'quit' to consult the editor documentation
first if necessary, or <Enter> to proceed:

Use CTRL-C to exit at any of the following prompts.
Enter a name for the Editor: Kwrite
Enter the path for the Editor (default /usr/bin): /opt/kde3/bin
Enter the filename for the Editor (default kwrite):
Is this a text-based editor intended to run inside a console window? (y/n): n

Enter the command line parameters to open a file to a line number.
Use LINENO (all caps) where the line number should go, and
PATH (all caps) where the full path to the source file should go.
Examples: PATH(LINENO,1) --OR-- +LINENO PATH
Parameters (default +LINENO PATH): --line LINENO PATH
Kwrite has been added to the list for Cleanscape GUI(s).
suse:~$ cat myeditor.lst
This file holds information required to add an editor to the Cleanscape GUI.
A line with '#' in column one is a comment.

Program "seteditor" interactively adds a file, or edit this file using the
template/example below (sans '#' in column one). "path_line" in the template
represents your editor's command line parameters for specifying
1) the source file's fully qualified pathname (denoted as PATH) and
2) how to jump to a specified line when opening a file (denoted as LINENO).

Note that PATH and LINENO must be in all caps, the executable starts with
'/', and the editor path does NOT have a trailing '/'.

"text_based" in the template is either a Y or a N and indicates whether the
editor is text-based and intended to run inside a console window. This
field is ignored (but must still be present) for windows.

TEMPLATE:
editor-label__editor-filename__editor-path__text-based__path-line

EXAMPLE:
Joe__/joe__usr/bin__Y__+LINENO PATH

Kwrite__kwrite__opt/kde3/bin__N__--line LINENO PATH
suse:~$
```

### 7.3 TESTING SELECTED FILES IN PROGRAM CONTEXT USING *USESCAN*

#### A. Introduction

Also by popular demand, Cleanscape has added the ability for users to test only a few files within the context of the entire program. This is useful for a large, stable sourcebase where perhaps 1-2 sourcefiles are being modified. Running an analysis over the entire sourcebase can be time- and resource-consuming, only to find (perhaps) a trivial error in the unit under test (UUT).

Users specify the UUT file(s), then use Component Test (GUI) or external program *usescan* (command line) to automatically determine the other source files necessary to resolve *USED* modules, *INCLUDE*/*#include* files, and preprocessor (*-D*) macros.

To accomplish this, there are two modes to *usescan*: a Definition Mode, where the project is described in an interactive command line session, and a Resolution Mode, where information is extracted from the project data obtained during Definition Mode.

#### B. Definition Mode

The program is invoked without any operands in a command line shell:

```
<install_dir>/bin/usescan (*nix)
"<install_dir>\bin\usescan" (Windows)
```

An interactive session begins, in which the user is asked to provide all the sourcefiles that comprise the project, the directories containing *INCLUDE* or *#include* files, and any preprocessor (*-D*) macros used in the program.

Upon conclusion, two files are created in directory *<install\_dir>/projects*:

```
<projname>.cfp A project file listing all files, includes, and defines
<projname>.dep A recursive dependency list
```

A sample session follows. User input is in green. Note that the sample files are stored in the directory shown in the example: *<install\_dir>/examples/heeds*

```

READY (C:\cleanscape\flint\examples\heeds) c:\cleanscape\flint\bin\usescan

This interactive program creates a project description '.cfp' file for use by
CASAF or Flint. The goal is to have the test environment match the build en-
vironment as closely as possible, and provide contextual information when only
a portion of the entire sourcebase is being tested. There are 4 steps.

STEP 1: provide a single-word name for the project - no spaces, no extension...
heeds
Project file name is c:\cleanscape\flint\projects\heeds.cfp

STEP 2: enter ALL source filenames that comprise the project. Wildcards are
allowed. It may be useful to open a secondary command prompt or GUI-based file
manager to make this process easier. Gather info; press <ENTER> when ready...

Enter the first directory name containing project sourcefiles (or 'done'):
.

Enter the files in this directory that are part of the project, one per line.
Do NOT enter include files, #include files, or non-Fortran-source files.
Wildcards are allowed. When finished, enter 'done'.
*,f90
File(s) accepted. Enter next file, or 'done': done

Enter another directory name containing project sourcefiles (or 'done'):
done

STEP 3: enter include or #include paths. No need to relist source directories.
When done, enter 'done'.
Enter include or #include directory path #1:
done

STEP 4: enter any -D macros which are used in the build process, all on one
line, separated by spaces. Do not add the '-D', and do not use a space around
an '='. If there are no defines, enter 'none'.
Enter the list of -D macros:
none

Done! Project file is - c:\cleanscape\flint\projects\heeds.cfp

```

### C. Resolution Mode

#### GUI.

From the Flint GUI, select Component Test from the Misc Options tab in the lower left frame of the GUI (see Section 4.B.8). Necessary files are automatically added to the analysis, and any include directories or preprocessor macros are input to the analysis as well, if you specified them during project definition a la Part B above.

If you use preprocessor directives like #if or #include, be sure to

- Click the checkbox “Preprocessor” on the Source Config tab (and if necessary, specify the preprocessor binary by using the Locate... button at the bottom of that same tab).
- Specify any include directories for the Fortran INCLUDE directive or preprocessor #include statement. You can skip this step if you have already provided include information during project definition Part B.

- Define or Undefine any preprocessor macros (symbols) necessary for proper branching on the Misc Options tab. Again, no need to perform this step if you already provided this information during project definition Part B.

When the analysis is run, there is no outward appearance that files have been added: the file list in the GUI remains the same, listing only the UUTs you originally specified. However, the top of the analysis report will show you which files have been analyzed.

*Example:* Run the usescan command line session as shown in Part B (note: start in the `examples\heeds` subdir). Then, start the GUI. Click the Add File button, select `xml.io.f90` from the `examples\heeds` subdirectory. Check the Component Test box on the Misc Options tab, then run the analysis. You will see that three files were analyzed, despite your having specified only one UUT:

| 1 Analysis Report                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 2 Statistics | 3 Cross Reference | 4 Call Tree | 5 Include Tree |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|-------------------|-------------|----------------|
| <pre>&gt;&gt;&gt; Source analysis:  Directory c:\progra~2\cleanscape\flint\examples\heeds\  utilities.f90  university.f90  xml.io.f90  ***** Subroutine USAGE                      File utilities.f90          Line 326       &lt;Module subprog of UTILITIES&gt; ***** Subroutine CHECK_ARRAY_BOUND          File utilities.f90          Line 357       &lt;Module subprog of UTILITIES&gt; ***** Subroutine LOG_COMMENT                File utilities.f90          Line 389       &lt;Module subprog of UTILITIES&gt;</pre> |              |                   |             |                |

**Command line.**

The syntax of Resolution Mode is –

```
usescan -G -P <projfile> [-E <cmdfile>] <sourcefile1> [sourcefile2 ...]
```

where *projfile* is the name you supplied during the interactive session (Definition Mode). No path or extension information is required.

Without -E, data is written as a space-separated string to stdout.

With -E, data is written one per line to the user-specified Flint command file (intended to be run with Flint's -E command line operand).

Errors are written to stderr. Return codes are set as follows:

0 → success, 3-5 → warning, 8 → error, 10 → setup fail

**Example 1:** Having previously run usescan in Definition Mode as shown in Part B, run the command (from the 'examples/heeds' subdirectory)

```
usescan -G -P heeds xmllo.f90
```

to obtain the following, which has spaces between each datum (the output is one continuous line, but word wrapped in this document and in the shell):

```
-Ic:\cleanscape\flint\examples\heeds c:\cleanscape\flint\examples\
heeds\utilities.f90 c:\cleanscape\flint\examples\heeds\
university.f90 c:\cleanscape\flint\examples\heeds\xmllo.f90
```

**Example 2:** Same as above, but store the results in a Flint command file:

```
usescan -G -E \temp\heeds.cmd -P heeds xmllo.f90
```

The contents of \temp\heeds.cmd are as follows:

```
-Ic:\cleanscape\flint\examples\heeds
c:\cleanscape\flint\examples\heeds\utilities.f90
c:\cleanscape\flint\examples\heeds\university.f90
c:\cleanscape\flint\examples\heeds\xmllo.f90
```

**Example 3:** Specify a UUT that requires more modules:

```
usescan -G -P heeds server.f90

-Ic:\cleanscape\flint\examples\heeds c:\cleanscape\flint\main\
isobind.lsh c:\cleanscape\flint\examples\heeds\utilities.f90 c:\
cleanscape\flint\examples\heeds\university.f90 c:\cleanscape\flint\
examples\heeds\xmllo.f90 c:\cleanscape\flint\examples\heeds\html.f90
c:\cleanscape\flint\examples\heeds\editenlistment.f90 c:\cleanscape\
flint\examples\heeds\editstudent.f90 c:\cleanscape\flint\examples\
heeds\editchecklist.f90 c:\cleanscape\flint\examples\heeds\
editblocks.f90 c:\cleanscape\flint\examples\heeds\editsections.f90
c:\cleanscape\flint\examples\heeds\editteachers.f90 c:\cleanscape\
flint\examples\heeds\edituniversity.f90 c:\cleanscape\flint\examples\
heeds\initialize.f90 c:\cleanscape\flint\examples\heeds\server.f90
```

For Examples 1 and 3, copy/paste the output to a Flint analysis; for Example 2, run Flint with the -E\temp\heeds.cmd (you can add any other analysis controls you'd like to the .cmd file). Remember Flint option -p if your project requires running the code through a preprocessor before analysis!

To see a help listing from a shell prompt, run the following command:

```
<install_dir>/bin/usescan -? (*nix)
"<install_dir>\bin\usescan /?" (Windows)
```