

# Cleanscape Fortran-lint Demo Startup Guide

Version 7.58



## **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](http://www.cleanscape.net)  
[sales@cleanscape.net](mailto:sales@cleanscape.net)  
[support@cleanscape.net](mailto:support@cleanscape.net)

Copyright © 2014-2023 by Cleanscape as an unpublished work.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.

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



## **WELCOME**

Thank you for your interest in Cleanscape products! 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 of programmers – and anyone tired of command prompts and/or desiring the productivity gains found in using a GUI.

As of version 7.58, there are 909 unique messages that can diagnose 1655 situations in Fortran code.

## **DOCUMENTATION**

This is the “quick start” guide for demo users 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 purpose is to introduce trial users to static analysis, and specifically Flint’s features and operation. There are two other documents which these users should refer:

Flint is very rich in analysis controls and reporting; to gain maximum benefit from your (ultimate) product purchase, we urge you to read and keep handy the companion document, [Flint Reference Manual](#) located in the ‘doc’ subdirectory.

The Flint GUI provides ease-of-use enhancements over the Flint command-line product. For details on the different user modes – GUI, IDE integration, and command line – see the [Flint User Interface Guide](#), also located in the ‘doc’ subdirectory.

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.

## **THE PURPOSE OF FLINT**

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:

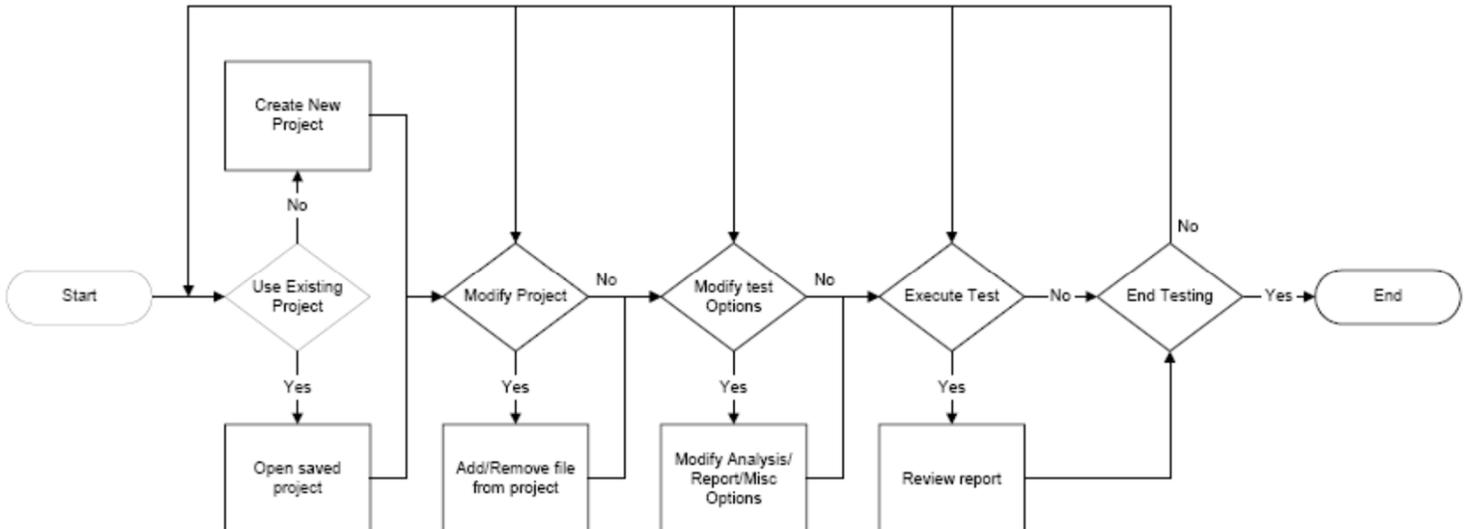
- a. Check source files before they are compiled
- b. Isolate obscure problems
- c. Identify problems before debugging is required – or available

## 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:



## **LINT (STATIC) ANALYSIS OVERVIEW**

For those of you new to static analysis, this is a form of testing which occurs pre-compile, i.e., no compiler is necessary and all you need are your source files. Static analyzers can detect defects in minutes (on par with compilation time) – issues that could take days using a debugger or print statements.

While a compiler also produces error messages, its main purpose is to generate code that runs, so its error analysis is mainly intended to remove syntax errors. While Flint also detects syntax errors, its purpose is to help you produce code that runs *correctly*. Static analysis can locate both immediate problems and issues that could cause problems perhaps years down the road.

Key features of Flint include:

- Over 1650 checks of your source code – many more than a compiler can produce – the most thorough analyzer on the market!
- It is parser-based (like your compiler), not a scanner, meaning more accurate analysis results.
- Unlike a compiler, Flint can assess your code as an entire program (rather than analyze each file as an independent unit). This *global analysis mode* can, for instance, check for datatype mismatches between caller/calling routines, or that a dummy variable mistakenly has the allocatable attribute, or that a constant argument has been assigned a new value in the called routine.
- We also check for common coding errors based on what IS allowed in the spec. A classic example here is implicit typing, e.g., undeclared variable 'i' becomes implicitly an integer and a real assigned to it becomes cast to an integer. Other examples include our FYI messages, which are programming subtleties reported to us by our customers and we codified as potential problems to be checked for in your code.
- Dataflow analysis: this advanced feature parses conditionals to determine set/reference issues. For instance, variable `foo` is set in the `if` portion of a branch but not in the `else` branch, then is subsequently referenced.
- It is even possible to create a set of prototypes for 3<sup>rd</sup> party code or libraries that tell Flint the argument datatypes and whether a variable is set or referenced in the 3<sup>rd</sup> party code, thus allowing for 100% accurate global interface checks between any body of code! We supply definitions and prototypes for IEEE modules, ISO\_C\_BINDINGS, MPI, and NetCDF.
- Flint is used in over 5000 unique projects worldwide. Key customer markets include defense/aerospace (including many DoD facilities), oil/gas/nuclear energy (including DOE labs), weather models, and flow simulation. Incidentally, Flint is the only domestically produced analyzer for Fortran (as is our sister product, C++-lint) - an important factor if your customer is the US government.
- Flint is well supported: our support staff is constantly receiving kudos for its professionalism, responsiveness, and quality of service.

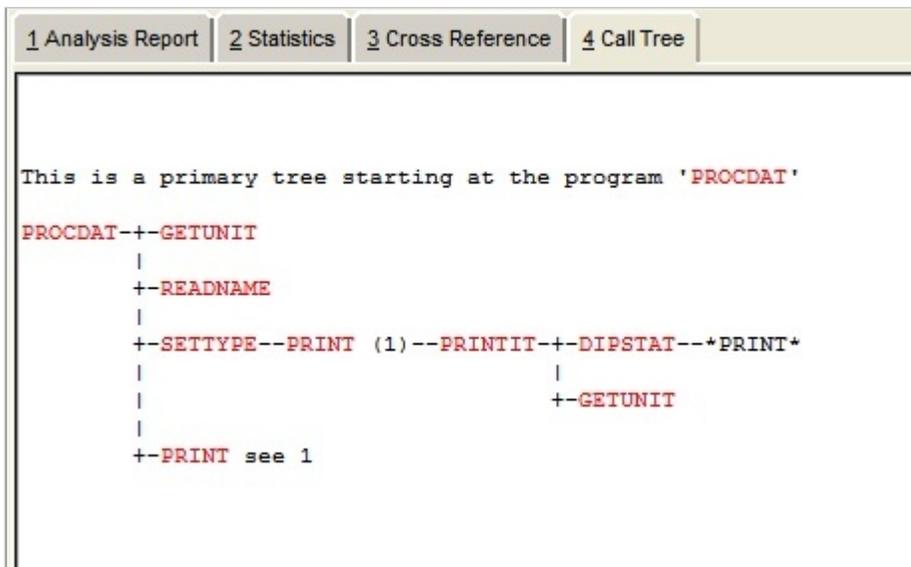
We believe Flint is best static analyzer on the market and we hope this free demo proves it to be a useful addition to *your* software toolkit!

## CALL TREE

A call tree provides a visual depiction of how routines are called within your program. They are generally rooted from the main program, but in Flint you can specify any routine you'd like the tree to start from.

In the GUI, a routine name in red can be clicked on, whereupon you will jump to the entry point of that routine in your favorite code editor. This is a supremely useful as a

- documentation/ code maintenance tool, or
- code familiarization tool, either for new programmers to a project, or when integrating 3<sup>rd</sup> party code (e.g., weather models, water/oil/gas flow simulations) into an existing project.



The screenshot shows a software interface with a menu bar at the top containing four items: '1 Analysis Report', '2 Statistics', '3 Cross Reference', and '4 Call Tree'. The '4 Call Tree' item is selected. Below the menu bar is a text area containing the following text:

```
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
```

**CROSS REFERENCE (XREF)**

A detailed listing of *all* symbols: Program, Subroutines, Functions, Modules, Types, Parameters, Structure components, Common blocks, Structures, Records, Variables, and Arrays. Both local and global variables are recorded in the xref. This global mode is quite powerful: relying on a compiler's symbol table outputs, if you wanted to see how variable `foo` was used throughout your program, you'd have to review possibly hundreds of local cross references and mentally stitch all that information together.

The line(s) of a variable's declaration, set, and reference are provided in the xref. Clicking on one of these line numbers in the GUI will jump you to that source line in your favorite code editor.

Report may be filtered to include/exclude symbols; for instance, it is possible to exclude all variables used as loop counters whose names are length one, thus excluding loop variables like `i`, `j`.

The cross reference is extremely useful for documentation, program debugging, and refactoring, as it pinpoints all accesses (read and write) of a particular variable, including COMMON blocks.

```

1 Analysis Report  2 Statistics  3 Cross Reference  4 Call 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:

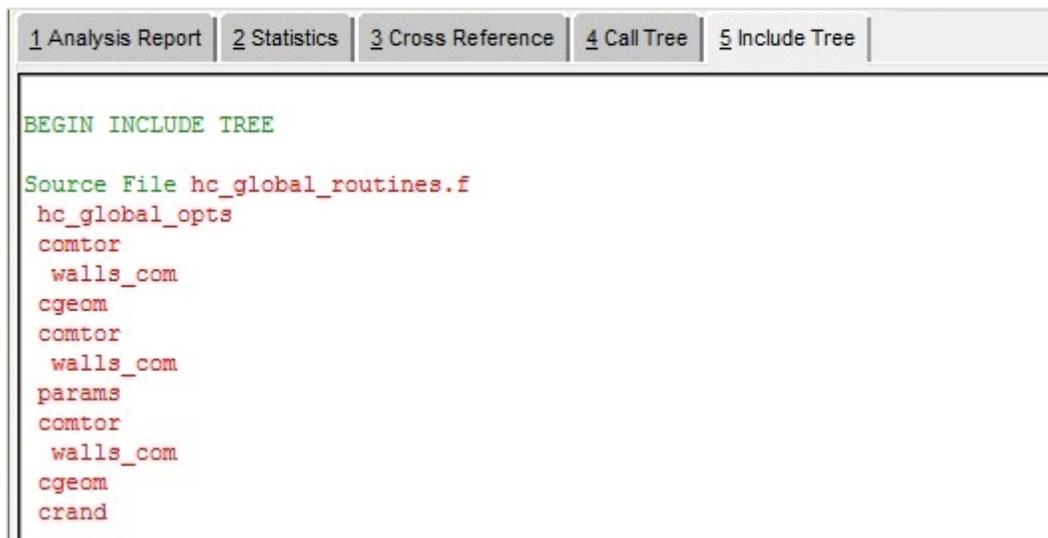
AVE : I*4 : public entity of module M
           in (demo90.f90:M) is  10-D
           in (demo90.f90:MAIN) is  49-S
DUM (:,:) : R*4 : local
           in (demo90.f90:MAIN::MAIN_INNER) is  (demo90.inc)3-P
                                                (demo90.inc)4-D
                                                (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

```

## INCLUDE TREE

Flint scans both Fortran INCLUDE lines and preprocessor #include directives to produce a tree of included files. While not required very often, the subtle issues raised by including the wrong (version of a) file could take a long time to diagnose otherwise!

As with other reports, clicking on a filename in red will open that file in your favorite code editor.



```
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
```

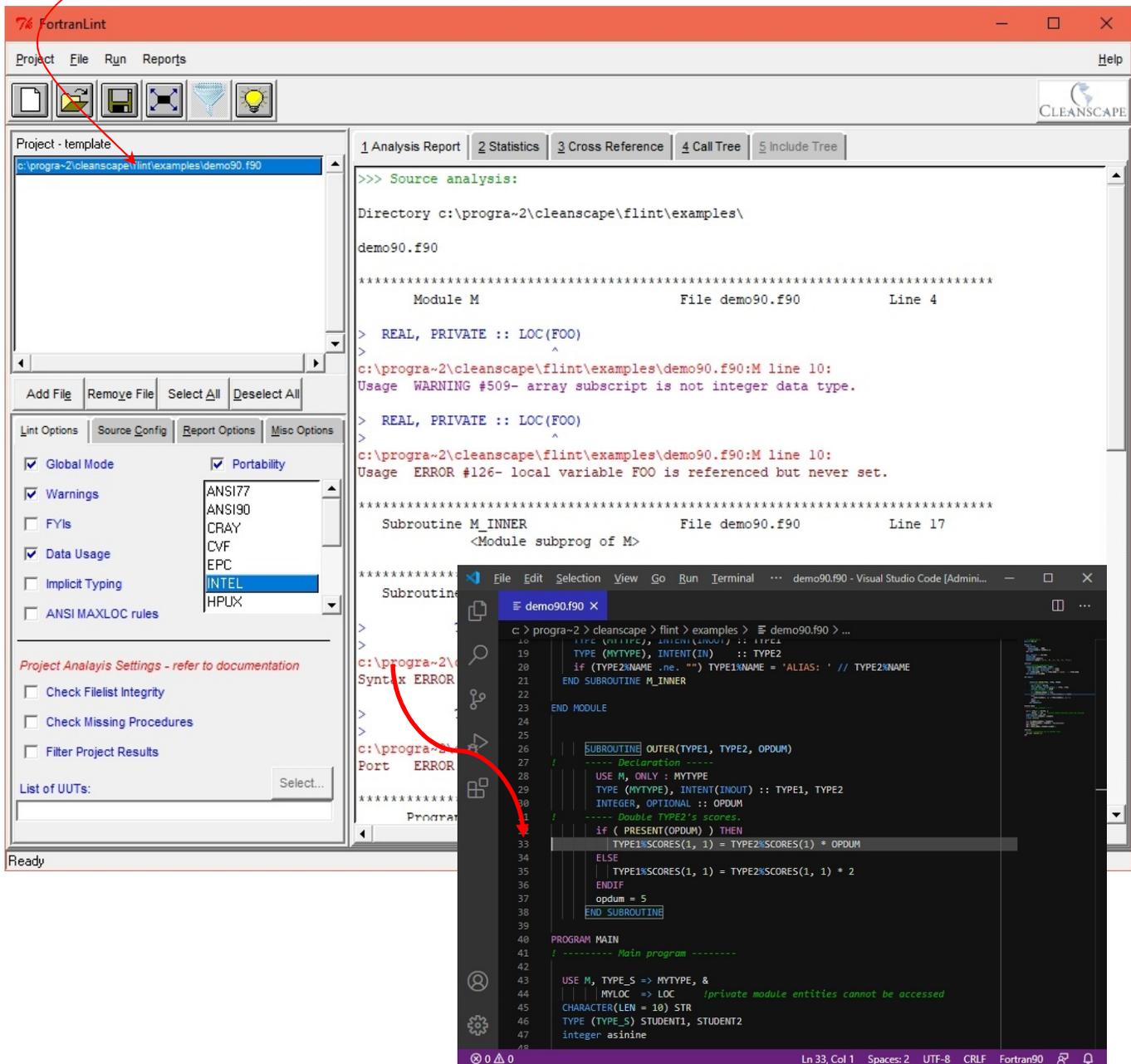
## THE GUI TO BIND THEM ALL

The Cleanscape GUI is a tried-and-true graphical interface used successfully for years. It is also the interface for our C/C++ analyzer offerings and test tools. Advantages of the Cleanscape GUI include:

- Fast; easy to learn, navigate, and use
- Point-and-click control for options-laden Flint command-line product
- One-click access to relevant code in your favorite code editor! Perfect for
  - quick review/patching of source code
  - source browsing

In the example below, Flint opened the external editor VS Code to source file line 33 when the user clicked the red line containing “33” in the Analysis Report.

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



## WINDOWS INSTALLATION

### A. System Requirements

#### 1. Hardware

Any configuration sufficient to run Windows is sufficient for Flint.

#### 2. Operating System

- a. Microsoft Windows 11
- b. Microsoft Windows 10
- c. Microsoft Windows 8
- d. Legacy support: Microsoft Windows 7, 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

### 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. The installer will upgrade you to 64-bit versions of key files as applicable.
- c) The installer automatically creates a shortcut for the Flint GUI on the desktop. (To run the GUI, double-click the shortcut.)
- 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](#).

- 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 in a command prompt, enter the following:  
`set PATH=<install_dir>\main;%PATH%`

#### 2. Permissions/ACLs

Since Flint writes to the installation directory, make sure you have write permissions to this directory and all subdirectories when you install Flint. We recommend installing with Administrator privileges.

### 3. Upgrading to full version upon successful eval and purchase

Once you have purchased Flint, double-click the *ConvertToFull* file found in `<install_dir>`. This will enable full operation of Flint on your system.

#### C. Uninstallation

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

There are two Uninstallation options.

##### 1. Manual uninstallation – *recommended*

- 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`:
  - 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.
  - Click on each Cleanscape tool in turn, then click on the Delete button.

##### 2. Restore your system to the point just before Flint installation –

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.

## UNIX/LINUX INSTALLATION

### 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, including PPC)
- c. HP HP-UX® (PA-RISC or Itanium)
- 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`. Note there is a 64-bit distro 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) To start the GUI:

```
# flintgui &
```

e) If you intend to run Flint from the command line, these additional commands are required (examples below are for `sh/bash`) on the server and each client machine.

```
# export CSIAPPPBASE=<install_dir>
```

```
# export FLINTHOME=$CSIAPPPBASE/flintgui.dir/main
```

```
# export PATH=$CSIAPPPBASE:$FLINTHOME:$PATH
```

### C. Uninstallation - manual process

a) Delete the installation directory and its subdirectories.

b) Delete files `myeditor.lst`, `ftemplate.csi` and `.flint.ini` from each user's `$HOME` directory.

## LICENSE MANAGEMENT

There's no license management for the demo! Simply operate Flint according to one of the four modes indicated below. (Once you have acquired Flint, there is a license manager that's easy to install/operate and we will be happy to assist.)

## FULLY OPERATIONAL!

The demo is fully operational; the only restriction is limiting the number of specific results (i.e., the exact source line and line number) in the analysis reports.

**HINT:** Check out the summary listing, which will list *each error encountered* during the run *and the number of times each error was detected*.

## STARTING THE GUI

On Windows, simply double-click the FortranLint icon on your desktop.

On \*nix, run the command `flintgui &`

See the [Flint Quick Start Guide](#), located in the 'doc' subdirectory for detailed operation and examples.

## STARTING THE COMMAND LINE

It is also possible to run Flint from the command line; return codes are generated, which makes Flint useful for integration into builds. This mode can also support remote users and anyone who is not interested in using a GUI.

You need to set the environment variables as described in the Installation section of this document. You then simply run the command `flint` to see a command summary.

See the [Flint Reference Manual](#) located in the 'doc' subdirectory for further information. Note that this document also provides details on each Flint command.

## STARTING XLINT – \*NIX ONLY

There is a legacy GUI called Xlint on Unix/Linux platforms. Its functionality has been supplanted by the Flint GUI, but continues to be provided to its many legacy users. See Section 12 of the [Flint Reference Manual](#) located in the 'doc' subdirectory should you require this interface.

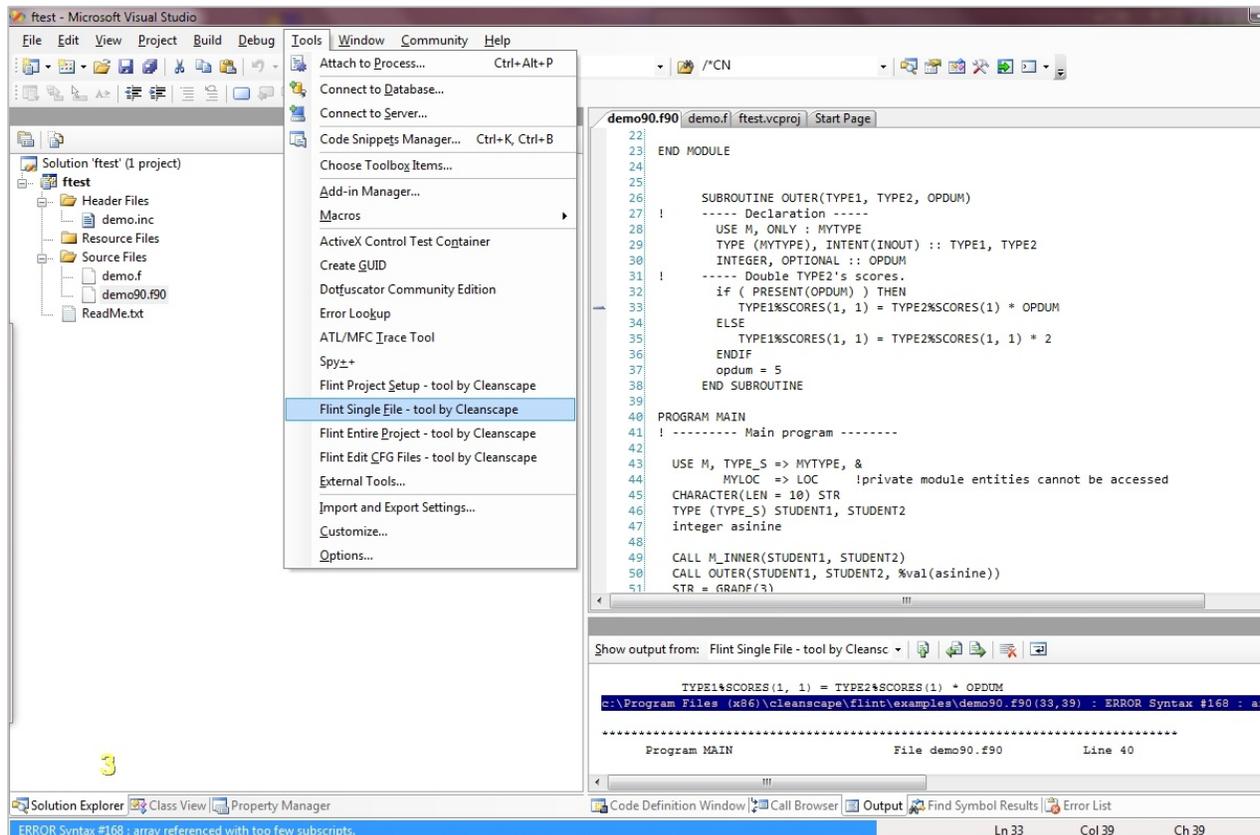
## STARTING VISUAL STUDIO INTEGRATION – WINDOWS ONLY

When you installed Flint, external tools were created in any Visual Studio implementation you have on your Windows system. Open a Fortran project, and you will see Flint tools on the Tools menu. 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.
- 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!

For detailed operation, see Section 6 of the [Flint Quick Start Guide](#), located in the 'doc' subdirectory.



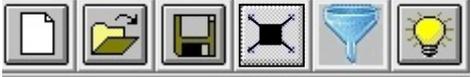
For purposes of the demo:

- Run Project Setup (tool #1) in the IDE.
- Start the Flint GUI and use Project-Open to open the `<IDEproject>.csi` file located in your Visual Studio project directory.
- Browse the various settings in the GUI. Note all your files are present (and any files Excluded from Build are not), plus the presences of include directories, defines/undefines, and any project-specific settings relevant to an analysis.

Running Flint over a large set of files is outside the scope of this demo. But as this tutorial shows, you can now 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!

## THE 10-MINUTE DEMO CHALLENGE

Cleanscape products have been engineered to be fast, easy to use, and produce meaningful results. We fully believe that our products speak for themselves, which is why we've created this challenge in 7 easy steps. Ready?

1. If you haven't already, finish installing the product (see page 9 or 11).  
*Windows:* Ensure you have write permission to the installation directory.  
*\*nix:* Set `$CSIAPPPBASE` and `$FLINTHOME` and prepend them to `$PATH`.
2. Start the GUI.  
*Windows:* double-click the FortranLint icon on your Desktop  
*\*nix:* type `flintgui &` at your command prompt.
3. Select your favorite code editor. In the *Report Options* tab in the lower left frame of the GUI, click the dropdown next to **External Editor** and select one. If the path to the editor is different on your system, click the **Locate...** button to specify the directory it's in on your system.  
  
If your favorite editor is not in the list, add it yourself!
  - a) Exit the GUI.
  - b) *Windows:* In Explorer, double-click *seteditor* in `<install_dir>\bin`.  
*\*nix:* Open a command prompt, and run `<install_dir>\bin\seteditor`
  - c) Restart the GUI according to Step 2 above, go to the *Report Options* tab, click the dropdown - your editor is now in there! Select it.
4. Click the Add File button located midway down the left side of the GUI. Navigate to the `examples` subdirectory and select file `demo90.f90`.
5. Click the Run button at the top of the GUI (it looks kinda like an X). Analysis will run. 
6. Click on the various report tabs in the right frame of the GUI. Review the features of those reports on pages 4-8 of this document.
7. Click on any item in red in those reports. What happened? Do you like it?

### DIFFICULTIES?

We want your evaluation to run smoothly and produce results for you quickly (duh). If you spend *more than 2 minutes* puzzling over something, call 800-944-LINT or +706-245-1070, or email [demo@cleanscape.net](mailto:demo@cleanscape.net). Maybe there's something we could have written better, or the product could operate better, but in any case, don't waste time, save time with Flint!

### CHALLENGE DONE. WHAT'S NEXT?

Now that you are familiar with the GUI's operation, add your own files. Start without Global Mode or Warnings. Later, enable them, then FYIs and implicit type-checking on the *Lint Options* tab, if you like. While the reports are limited in their detailed output, you can still see how much Flint found in your code by reviewing the Statistics report in the GUI's right frame. Then, if you like Flint...

Buy Flint! **Mention the Challenge for a special discount!** Then run *ConvertToFull*, located in your installation directory. Follow the instructions in Section 3 of [Flint Quick Start Guide](#), namely, email us the server code, and install the key we return. Then relax, you're In Like Flint.