Cleanscape gr8utils User's Guide

Version 1.0

CLEANSCAPE
software international

Sales and Service Office
5850 Eubank Blvd. NE, B49-180
Albuquerque, NM 87111
Toll-free 800-94-4LINT
505-246-0267
Fax 505-247-2678
www.cleanscape.net
sales@cleanscape.net
support@cleanscape.net

PART I Introduction

1.1 WELCOME

Thank you for your product purchase! With Cleanscape's gr8utils, you have added powerful, quick, and convenient utilities for managing your computer and its files.

1.2 DOCUMENTATION

This is the user's guide for the eight gr8utils. Each utility has its own chapter. NOTE: You may not have purchased all the utilities described in this document; to add more, visit www.cleanscape.net/products/gr8utils.

For each utility, a command summary is presented along with a rating for each type of user and for each host it can be run on. Examples with screenshots are provided, as are hints and tricks to maximize the effectiveness of using the utility.

1.3 PURPOSE

A. Function

The gr8utils provide users with powerful command-line control of eight common functions relating to file management, locate, search, and display. Some of the functions mimic those found on Linux systems; all provide powerful features, fantastic tech support, and flexible licensing.

B. Application

The gr8utils offer the most functionality to the software developer, followed by system administrators, and finally to the knowledgeable general user. From a platform perspective, they are most useful to Windows users, followed by Unix users, followed by Linux users.

Arguably, we could have renamed the package something different on Unix or Linux, since not all the utilities have the same, er, utility on those platforms (e.g., mag7, htnml6, fab5, fntstc4...) but some would probably argue that the one abbreviation "gr8" is one too many.

C. Advantages

1. Command line programs often operate faster than the equivalent GUI function, if the GUI function exists at all!
2. They require less windowing, navigation, and mouse clicks.
3. They provide finer levels of control.
4. They are excellent for remote system administration or as diagnostic aids for deployed software.
5. They provide consistent results for managing heterogeneous networks or code deployed in heterogeneous environments.

PART II Requirements, Installation, and Uninstallation

2.1 WINDOWS

A. System Requirements

1. Hardware
   Any configuration sufficient to run Windows is sufficient for the gr8utils.
2. Operating System
   a. Microsoft Windows 98® and 98® SE
   b. Microsoft Windows NT® 4.0 with Service Pack 6a (SP6a)
   c. Microsoft Windows 2000® with Service Pack 2 (SP2)
   d. Microsoft Windows XP® with Service Pack 2 (SP2)
   e. Microsoft Windows Vista®

B. Software Setup Procedure

Please read the No-Nonsense License Agreement first

1. Installation
   a) Copy `gr8utils<ver>_win.exe` to a temporary directory, then run it.
   b) An installer window should appear. Click the OK button. This should extract a number of files to the directory you specified. The installer exits automatically, and no reboot is required.
   c) The installer adds the installation subdirectory to your system PATH – necessary for running the gr8utils (or any of the associated support programs) from the command line. To do this manually, run this command:
   ```
   set PATH=<install_dir>;%PATH%
   ```
   d) Follow the instructions to obtain a license key as described in Part 3.
2. Additional steps for Windows 2000 and later

   If you're going to install Flint under Windows 2000 as Administrator, and you want to make the program accessible to ordinary "Users", some additional steps are required. For more information, see Section 5.1.

C. Uninstallation – manual process

a) Delete the installation directory and its subdirectories.
b) Delete the installation directory from your `PATH`:
- In Windows 98, delete the appropriate "`set path=`" statement from your `c:\autoexec.bat` file.
- In Windows NT/2K/XP/Vista, right click your "My Computer" icon on the desktop, select "Properties", click the "Advanced" tab, click the "Environment Variables", double-click the text field "Path" in the System Variables area, and from that string, delete the installation directory.

You can also restore your system to the point just before gr8utils installation – NOT available for Windows NT!

The installer created a Windows 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, run this command: `scanreg /restore`

2.2 UNIX/LINUX

A. System Requirements

1. Hardware
   A minimum of 256 MB memory is required for the gr8utils.

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® Tiger
   c. Sun Solaris®
   d. HP HP-UX® (PA-RISC *and* Itanium)
   e. SGI Irix®
   f. IBM AIX®

B. Software Setup Procedure

Please read the No-Nonsense License Agreement first

Installation – installation as root is easier and recommended. The '#' below represents the root prompt.

a) Download the latest version of `gr8utils<`*`ver`*`>_<`*`OS`*`>.taz` to a temporary directory, e.g., `/tmp`.

b) Create installation directory, e.g., `/usr/local/gr8utils`, and `cd` to it.

c) Use the following commands to extract the files:
```
# gunzip  /tmp/gr8utils<ver>_<OS>.taz
# tar xvf /tmp/gr8utils<ver>_<OS>.tar
```

d) Set the environment (bash example):
```
# export GR8UTILS=<install_dir>
# export PATH=$GR8UTILS:$PATH
```
e) Follow the instructions to obtain a license key as described in Part 3.

C. Uninstallation – manual process

a) Delete the installation directory and its subdirectories.
b) Delete the installation directory from `PATH`

PART III Activating gr8utils

1. Run the command, `gr8utils –license=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. On Windows machines, it starts with "30720/"; on Unix/Linux, it is all numeric.

   You will have been sent a temp key as part of your purchase; enter it at this point.

   To obtain your permanent key, contact Cleanscape Software and provide this server code. Contact Cleanscape at 800-944-5468 or email support@cleanscape.net.

2. Once the activation key is entered, the gr8utils you have purchased are registered and operational.

3. If you ever need your server code again, it is in the version info: `gr8utils –ver`

NOTE: Even though the installer installs all eight gr8utils, you will only have access to the products you have purchased. Getting access to an additional utility is easy; order it from www.cleanscape.net/products/gr8utils and all you need is an updated key.

Contact sales@cleanscape.net if you are interested in network licensing.

PART IV Individual Utilities

0. General Notes.

All utilities support the following command-line options:

`-VER`	supply version information to *stdout* and quit
`/?` *or* `-?`	supply help information to *stdout* and quit

Single-character flags *cannot* be combined, and *must* be separated by spaces. The following will not work:

```
fs /w/s/c "*.c" "int main"
fs -wsc *.c "int main"
```

Unless otherwise noted, results are written to *stdout*, while error/informational messages are written to *stderr*. The latter can be suppressed by redirecting *stderr* (`2>NUL` or `2>/dev/null`), but if you do this, be sure to inspect the return codes (see next)!

Win98 Note: It is not possible to redirect *stderr* from the command line! So we created a "clowd9" utility, `nostderr`; if you do not want error/informational messages, try this:

```
    nostderr gr8util gr8util_parameters
```

The gr8 utilities have a return code of 0 for the normal result situation. See the individual descriptions for what "normal" means for each utility, but in general:

RC=0	the program ran successfully and returned a single result
RC=1	the program ran successfully and returned zero results
RC=2	the program ran successfully and returned multiple results
RC=3..5	warning/informational message(s) have been generated and three (3) has been added to return code [0..2] . This is useful when *stderr* messages have been redirected to `/dev/null` or `NUL`, since there are potential problems that should be reviewed (rerun the utility without redirect).
RC=8	the program did not run successfully (e.g., bad input)
RC=10	the program aborted (e.g., fatal error, OS not supported)
RC=12	an error during the acquisition of a license (e.g., key expired, not authorized for the particular utility being started)

In the following chapters, a subjective "ranking" of each utility is supplied, both for the type of user and the host platform. (A "general" user is *not* a consumer-type or casual user, but instead refers to a professional who uses a computer every day.)

For each utility, there is also a(n often-lengthy) set of Programming Notes, which may provide background information, usage details, and performance tradeoffs.

Finally, the command-line switches are usually depicted in this document (and each program's online help) in ALL CAPS; however, case is not significant. Also, unless otherwise indicated, switches are Unix-style, preceded by '-', not '/' as in Windows.

Important notes for Unix/Linux users:

- The `GR8UTILS` environment variable must be set to the installation directory.
- If you don't have read access to a directory, no file information can be obtained.

Regarding Distribution Licensing:

All gr8utils may be licensed for distribution with your products; the most suitable are indicated with the box shown at right. Redistribution requires a separate license agreement and a fee (greatly reduced rates based on volume); contact sales@cleanscape.net for further information, including customizations for your distribution needs.

Redistribution Licensing Available

1. l2s

USABILITY (1=low, 5=high)			UTILITY		
Developer	Sysadmin	General	Windows	Unix	Linux
5	5	3	5	2	2

Redistribution Licensing Available

Purpose. l2s converts a Windows long filename possibly containing spaces to a non-space format, or to strict "8.3" format. As anyone programming Windows knows, spaces in filenames are a real annoyance, since spaces are natural delimiters. There are even Windows commands (e.g., move) that to this day don't handle spaces in names!

l2s is also useful for *nix programmers because it handles tilde- and relative-paths, converting them to a fully qualified pathname with no trailing slash.

l2s can be redistributed as a common code element for mixed *nix/Windows applications (since the shortened names are machine-dependent).

Real-World Uses.

- This utility can be redistributed (license sold separately) in your applications.
  - ◇ A bug-free solution for multi-platform product distributions (we have tested it: 1.5% of the time, Windows reports the wrong short name!)
  - ◇ Its optional non-strict output makes it more human readable than strict 8.3 names, and so is advantageous to use in tables, reports, GUI dropdown lists...
- Use it to quickly determine a short filename while at the command prompt. *Example:* the following will not work on a Windows XP command line (sometimes copy does):
  ```
  C:\chris\utils> move "demo\my   data\name with spaces.txt" \temp
  The system cannot find the path specified.
  ```
  See how l2s easily assists the operation:
  ```
  C:\chris\utils> l2s "demo\my   data\name with spaces.txt"
  C:\chris\utils\demo\MYDATA~1\NAMEWI~1.TXT
  C:\chris\utils> move C:\chris\utils\demo\MYDATA~1\NAMEWI~1.TXT \temp
  ```

Syntax. `l2s { <"path possibly with spaces"> [-STRICT | -FAST] } | -VER | (/?|-?)`

path	Required: the path to convert. Add double quotes if it has spaces. May be piped or redirected output.
-STRICT	return strict 8.3 filename format; default off. Ignored in *nix.
-FAST	not recommended unless converting several files; default off. If input has no spaces, the input is returned unchanged; otherwise, *always* returns 8.3. This mode is N/A Win98 and gets a warning and RC+=3.
-VER	Return version information and quit.
/? *or* -?	Print a help listing, then quit.

Returns. The converted name(s) if successful, null string if not, to *stdout*. Informational messages are written to *stderr*. NOTE: Directory results are *always* sans trailing '/' or '\'!

Return Codes. 0 for normal result, 1 for no result, +3 is added to the previous if there are *stderr* messages that should be reviewed, and 10 for fatal error.

Examples. Each is numbered [x].

```
[1] C:\> l2s "C:\Program Files\cleanscape\Copy of LPLUS\alongfilename.c.bak"
    C:\PROGRA~1\cleanscape\COPYOF~1\alongfilename.c.bak

[2] C:\> l2s "C:\Program Files\cleanscape\Copy of LPLUS\alongfilename.c.bak" -strict
    C:\PROGRA~1\CLEANS~1\COPYOF~1\ALONGF~1.BAK
```

```
[3] C:\Program Files\Microsoft Visual Studio 8\VC\atlmfc\src\mfc> dir /b *.c | l2s
    C:\PROGRA~1\MID05A~1\VC\atlmfc\src\mfc\mfcmanifest.c
    C:\PROGRA~1\MID05A~1\VC\atlmfc\src\mfc\rawdllmainproxy.c
```

Programming Notes.

Because the "space in filename" problem occurs often in Windows development and administration, this utility has a *lot* of, well, utility in Windows for programmers and sysadmins. It is also worthwhile for mostly-Unix programmers who have to port code to Windows, or sysadmins who maintain scripts across multiple platforms. For these reasons it can be a good candidate for redistribution for some companies.

Note that directory conversions *never* have a trailing (back)slash. This is useful if obtaining a directory from an environment variable, or any other time that (back)slash's presence is unknown. For certain scripts (batch files and Unix sh) it is very hard to determine this in advance, but simply adding a (back)slash at the right place is trivial.

`l2s` will return a null string for double-backslashed (UNC) network paths, since Windows does not maintain an 8.3 version of a share name. Use subst or `net` use to assign a drive letter. Contact your IT Dep't or support@cleanscape.net if you need assistance.

`–STRICT` can be used for comparing filenames (for instance, gr8util `which` uses this mode to absolutely compare directory names).

`–FAST` should be specified *only* if you're converting several files at once. Otherwise, use the default (not `–FAST`); you will get a human-readable format in human-real-time.

With `–FAST` enabled, conversion time has been tested to 0.0458 seconds on a random sampling of 50,000 filenames on this author's XP hard disk. Without `–FAST`, average conversion time climbs to 0.127 seconds. Interestingly, WinXP appears quite a bit slower; a factor may be the complexity of filenames on that platform (avg. 85 vs. 70 chars/filename). Here is a table comparing hosts:

Platform	Count	Errors	Avg. Time	OS err %	Comment
Win98	31,585	0	0.0576 sec	N/A=0%	-FAST not available Win98
WinNT	37,639	0	0.0187 sec	2.12%	
Win2k	32,411	0	0.0090 sec	1.08%	
WinXP	47,821*	0	0.0652 sec*	1.36%*	*= avg. of 3 systems
WinVS	50,000	0	0.0098 sec	0.00%	

So, why wouldn't one always use `–FAST`? Because it returns *only* 8.3 format (except in the trivial case where the filename doesn't contain spaces), which is not very human-readable.

Side note: the Windows routines used to implement fast conversion have an error rate of over 2% on some platforms! `l2s` handles this by using a hybrid model, detecting the error and calling a custom routine, which has been confirmed to have *zero* errors, but is slower.

Note that `l2s` (or any long-to-short conversion) is machine-specific; for example, if `micros~1.txt` already exists, the conversion will create a `micros~2.txt`, but the existence of `micros~1.txt` is unknown beforehand. (Microsoft could have chosen a lossless conversion utility, but we're way past that now and the issue is therefore moo~1.)

`l2s` is useful on *nix because it processes tilde paths and provides a consistent path format (i.e., no trailing slash, ever), making it suitable for scripts.

If the reverse process (short to long) is interesting to you, contact sales@cleanscape.net and let us know you'd like an "`s2l`" program. We've never needed one and can't think of a scenario for using it, but who are we to limit our users?

2. which

USABILITY (1=low, 5=high)			UTILITY		
Developer	Sysadmin	General	Windows	Unix	Linux
5	4	3	5	4	2

Redistribution Licensing Available

Purpose. which searches for a given name within the directories of a given environment variable to determine which instance will be encountered first (and hence used) by the OS. It also reports other instances later in the execution hierarchy.

The most common use is finding which executable the OS will run. This is not necessarily the first instance in `PATH`: there are (DOSkey) aliases, (batch) scripts, and intrinsic commands; on Windows there are also numerous filename extensions (e.g., `.COM, .EXE, .BAT, .VBS`...) that could take precedence over the command you *think* you're running!

If *name* is not specified, which will delimit and list the contents of the specified environment variable, one per line – useful for searching and certainly more human-readable.

Real-World Uses.

- Verify the program you want executed is indeed the one that is run. *Real world example:* Our shipping programs on Windows needed the OS "find" command. Some of our customers were getting bizarre results we couldn't replicate. We sent them a copy of which as a troubleshooting aid, and found they had installed gnu binutils and had a Unix-style "find" command in their PATH before the Windows version!
- This utility can be redistributed (license sold separately) in your applications, especially as a diagnostic aid to confirm the program you are calling is really the one you want, or to confirm the order of directories in an environment variable is as intended.
- Determine whether the correct version of header file is being included based on the `INCLUDE` environment variable.
- Locate duplicate or bad entries in `PATH` or `INCLUDE` to optimize performance, or simply review the contents 1 directory/line without having to read a long, word-wrapped, mashed-together sequence of directories.

Syntax. `which {(-PATH | -INCLUDE | -VAR ev) (-SORT | [-STRICT] name)}| -VER | (/?|-?)`

name	Leaving off file extension is recommended unless you need an exact match; see Programming Notes section for details.
-PATH	With *name*, search for *name* in directory list within the `PATH` env. var. Else, list the directories in `PATH`, one per line. Useful when combined with `grep` or `find`. Ex: `which -path | find /i "sdk"`
-INCLUDE	Same as `-PATH` except use `INCLUDE` environment variable.
-VAR *ev*	Same as `-PATH` except use the specified environment variable *ev*. *NOTE:* `-PATH == -VAR path` and `-INCLUDE == -VAR include`
-SORT	Optional: Alphabetically sort the directories in the environment variable. Implies `-PATH` if nothing is specified. *name* overrides `-SORT` if both are specified.
-STRICT	Optional: Output results in 8.3 format (perhaps for subsequent use in a program). Useful to avoid "space in filename" issues when using this utility's output in code. Ignored in Unix/Linux.
-VER	Return version information and quit.
-? *or* /?	Print a help listing, then quit.

Returns. The full pathname of the first occurrence of *name*, or null string if no match, to *stdout*. Additional matches, if any, are written to *stderr*, as are informational messages.

Return codes. 0 for one match (or help, version, or list environment variables), 1 if no match, 2 if >1 matches; +3 is added if there are *stderr* messages that should be reviewed (including conflicts in user-specified parameters). 8 if an error (e.g., bad input) or 10 if severe (e.g., internal) errors were detected.

Examples. Each is numbered [x].

```
[1a] C:\> which which

     FYI: Redundant entries for the following directories were found in PATH.
          Only one instance has been retained for this program's search.
          c:\progra~1\resource kit == C:\Program Files\Resource Kit

     which ==> .\which.exe
     which ==> c:\chris\utils\which.exe

[1b] C:\> which which 2>NUL
     which ==> .\which.exe

[1c] C:\> echo %errorlevel%
     5      //RC=5 because more than one entry was found (RC=2) & there were messages (+3)

[2a] C:\> which fhaie 2>NUL   //Duplicate "resource kit" was fixed in PATH between Ex. 1 & 2

[2b] C:\> echo %errorlevel%
     1

 [3] C:\> which -var lib ole32.lib
     ole32.lib ==> C:\Program Files\Microsoft Platform SDK\Lib\ole32.lib
     ole32.lib ==> C:\Program Files\Microsoft Visual Studio 8\VC\PlatformSDK\lib\ole32.lib

 [4] C:\> which -path 2>NUL
     c:\windows
     c:\progra~1\resource kit
     C:\Program Files\Microsoft Visual Studio 8\Common7\IDE
     C:\Program Files\Microsoft Visual Studio 8\VC\BIN
     C:\Program Files\Microsoft Visual Studio 8\Common7\Tools
     C:\Program Files\Microsoft Visual Studio 8\Common7\Tools\bin
     C:\Program Files\Microsoft Visual Studio 8\VC\PlatformSDK\bin
     C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\bin
     C:\Program Files\Microsoft Visual Studio 8\VC\VCPackages
     C:\WINDOWS\system32
     C:\WINDOWS\System32\Wbem
     c:\chris\utils
     c:\chris\batch
     C:\PROGRA~1\MICROS~4\Office

 [5] C:\> which -include -sort

     Here are the 4 entries in INCLUDE, sorted by name:
     C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\include
     C:\Program Files\Microsoft Visual Studio 8\VC\ATLMFC\INCLUDE
     C:\Program Files\Microsoft Visual Studio 8\VC\INCLUDE
     C:\Program Files\Microsoft Visual Studio 8\VC\PlatformSDK\include
```

Programming Notes.

This program is *incredibly* useful for Windows software developers, because it can tell not only which executable will be run, but also tell which header file will be selected from the INCLUDE list of directories. It is also very useful for sysadmins and moderately useful for general, experienced users.

This program is also very useful for *nix non-bash programmers who have a very weak which built-in. For instance, now you can get a full listing of all matches (alias, intrinsic, and within PATH) that the system is aware of.

Windows key point: If you're searching for an executable on Windows, it's really best to leave off the file extension, since "find.exe" may not be the first executable "find" that runs – it could be "find.bat", or a DOSkey macro, or even a user-specified file extension that's been prepended to PATHEXT!

As a safeguard for this situation, which will omit the extension for the search, output results that do not have the correct extension to *stderr*, write the match, if any, to *stdout*, print a warning message to *stderr*, and set the return code accordingly (+3).

**nix key point:* Because programs run in subshells and subshells are not an exact copy of the parent, we have provided scripts to grabs alias/ environment information and stuff it into temp files for which to consume. In your installation directory, look for files which.bash and which.tcsh, and then set one of these aliases:

(sh family) `alias which='. $GR8UTILS/which.bash'`
(tcsh family) `alias which 'source $GR8UTILS/which.bash'`
NOTE: The above won't work with pure csh, as it does not accept arguments to source.

which will work fine without the scripts, using the environment/aliases as they appear at login. If you encounter any problems, let us know at support@cleanscape.net – there are some theatrics in determining the parent shell.

Remember, not only can you list of contents of an environment variable, but if the contents are directory names, you can also search within the directories for a file, as shown in Example [3].

It's also useful to match, for instance, .ini files to the corresponding executable:

```
[6] C:\src> which moreplus.ini
moreplus.ini ==> .\moreplus.ini
moreplus.ini ==> c:\progra~1\cleanscape\gr8utils\moreplus.ini
moreplus.ini ==> c:\chris\utils\moreplus.ini
```

which is smart enough to equate the various ways a path can be specified in Windows (long vs. short vs. in-between), as shown in Example [1a].

The shells for which which knows intrinsics (builtins) are as follows: bash, csh, ksh, sh(-posix), tcsh, Win2k, Win98, WinNT, WinXP, Win-Vista, and zsh.

which doesn't have a lot of utility on Linux, since similar functionality is available from the which and type commands, and there really isn't an INCLUDE environment variable to worry about. The ability to print out any environment variable one line at a time is mildly interesting (for instance, spreading out and then grepping $LS_COLORS); scanning other environment variables (e.g., $LD_LIBRARY_PATH) for a filename is more interesting.

3. *catalog*

USABILITY (1=low, 5=high)			UTILITY		
Developer	Sysadmin	General	Windows	Unix	Linux
5	5	5	5	4	1

Purpose. catalog and locate allow Windows or Unix users to catalog all relevant drives/ directories, and then quickly locate a filename of interest anywhere in the catalog. catalog creates a read-only catalog stored as `c:\udb.cat` (Windows) or `/var/udb.cat` (Unix).

Real-World Uses (see also locate)

- Unix servers or Windows machines can easily contain 200,000 or more files; searching for individual files of interest can take several minutes using existing OS services.
- On Windows, there's not a convenient method to search across multiple disks (the dir command does not provide for this).

Syntax. `catalog [{(+|-)`*drive_a_to_z*`: | +NFS | -`*ftype*` } …] | -VER | (/?|-?)`

+*D*: *or* -*D*:	Optional, *Windows only:* Add or exclude drive *D*: to/from the catalog. *D* is any valid drive letter (a..z). Mixed sequences of +*D*: and -*D*: may be repeated. catalog validates that added drives are up and available.
+NFS *or…*	Optional, *Unix only:* refine list of filesystems to traverse. By default, the
… -*ftype*	program will catalog everything *but* nfs mounts. -*ftype* may be repeated.
-VER	Return version information and quit.
-? *or* /?	Print a help listing, then quit.

Returns. The action is to create the udb.cat catalog. Messages are written to *stderr*.

Return Codes. 0 for normal result, 1 for not found, +3 added for informational messages, 8 for user error (e.g., syntax errors), 10 for fatal error.

Example. `catalog +G: -D: +Q:`
catalogs valid local drives, adds G: and Q:, and excludes local drive D:.

Programming Notes.

The functionality is very similar to the updatedb/locate routines on Linux. However, the output catalog's straightforward structure makes for easy access/ customization of additional search routines. The format of udb.cat (and example):

```
<fully_qualified_pathname>?<filename>
C:\corporate\Cleanscape Agreement.doc?Cleanscape Agreement.doc
```

On Unix, root access is much more thorough and recommended. If you are not root, only your `$HOME` directory (and its subdirectories) gets cataloged into `$HOME/udb.cat`.

As disk content changes, the catalog becomes less accurate. The tradeoff is the much faster response that can be obtained, especially on modern disks with hundreds of thousands of files to search through. If you need up-to-the-second file searching, consider using gr8util `fff` instead, which was consciously designed *not* to use the catalog catalog. <plug>Don't have it? Visit www.cleanscape.net/products/gr8utils today!</plug>

On Linux, use the native routines, as they are more versatile, especially as regards regexp handling.

Bundled with: locate
See also: `fff`

4. locate

USABILITY (1=low, 5=high)			UTILITY		
Developer	Sysadmin	General	Windows	Unix	Linux
5	5	5	5	4	1

Purpose. locate and catalog allow any Windows or Unix user to catalog all relevant drives/ directories, and then quickly locate a filename of interest anywhere in the catalog. locate reads catalog c:\udb.cat (Windows) or /var/udb.cat (Unix) for the specified string. By default, the string is assumed to be a filename unless –A is also specified.

The filelist can be piped to gr8util fff or morePLUS (if you are licensed); you may want to preview the list before piping (for instance, is it too long to interactively browse?).

Real-World Uses (see also catalog)

- System security check: what are all those processes running on your Windows machine? locate makes it a snap to find the executable named in Windows Task Manager or from Unix ps, neither of which contain the full path to that executable.
- *Real-world example:* When updating a user manual for multiple versions of a product in production, development, and test directories, locate quickly found all those locations where the new manual needed replace the old one.

Syntax. `locate { [-A] <string>[$] | "'<regexp>'" } | -STAT | -VER | (/?|-?)`

string	Required: The string to search for. If terminated with $, *string* matches only if at *end* of a catalog listing (see example [2]).
"'regexp'"	Alternate: specify a regular expression enclosed in *both* single- then double-quotes. *NOTES:* No checking of *regexp* is performed; one is limited to the regexp syntax of findstr on Windows NT+ or egrep on Unix; this functionality is not available on Win98.
-A	Optional: Search for <string> within the entire catalog, not just as a filename (i.e., could be part of a directory name as well).
-STAT	Print statistics regarding the current catalog, then quit.
-VER	Return version information and quit.
-? *or* /?	Print a help listing, then quit.

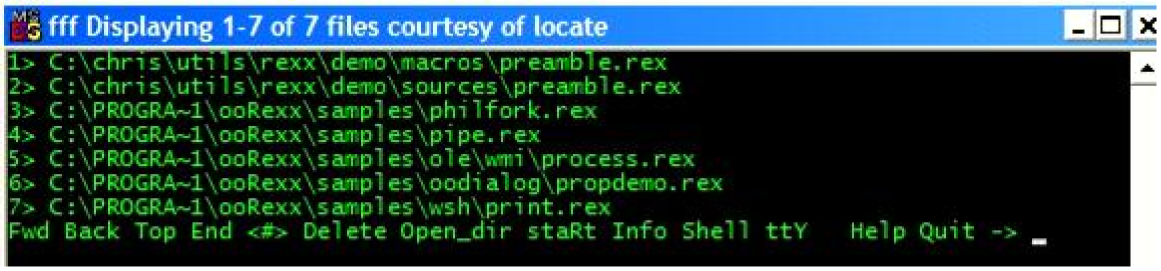
Returns. The full path name(s) matching *string*, or null string if no matches, to *stdout*. Informational messages are written to *stderr*.

Return codes. 0 for one match (or for help, version, or statistics), 1 if no match, 2 if >1 matches; +3 is added if there are *stderr* messages that should be reviewed. 8 if an error (e.g., catalog missing or input errors) or 10 if severe (e.g., internal) errors were detected.

Examples. Each is numbered [x].

[1]
```
C:\scripts> locate *.pl | moreplus -getcol 1
```
displays all instances of .pl in the catalog, using interactive pager/ gr8util morePLUS.

[2]
```
C:\scripts> locate p*.rex$ | fff
```
displays all filenames in catalog starting with p and ending in *exactly* .rex, and routes the output to gr8util fff for interactive browsing/file management (see image, next page). fff also has the benefit of confirming the filelist before presenting it, i.e., if any files from the catalog are no longer valid, fff will suppress them.



```
fff Displaying 1-7 of 7 files courtesy of locate
1> C:\chris\utils\rexx\demo\macros\preamble.rex
2> C:\chris\utils\rexx\demo\sources\preamble.rex
3> C:\PROGRA~1\ooRexx\samples\philfork.rex
4> C:\PROGRA~1\ooRexx\samples\pipe.rex
5> C:\PROGRA~1\ooRexx\samples\ole\wmi\process.rex
6> C:\PROGRA~1\ooRexx\samples\oodialog\propdemo.rex
7> C:\PROGRA~1\ooRexx\samples\wsh\print.rex
Fwd Back Top End <#> Delete Open_dir staRt Info Shell ttY   Help Quit -> _
```

```
[3] C:\scripts\rexx> locate -A p*.rex
```

not only finds the items in [2] but also, e.g., c:\temp\foo.rexx, c:\files\tproj2.rexx, and c:\checkpoint\stats.rexx.info

Programming Notes.

The functionality is very similar to the updatedb/locate routines on Linux. For up-to-the-second file searching, use gr8util fff instead, which does *not* to use the catalog catalog.

Extensive coding/testing makes the input "natural"; characters '\ * .' have special meaning and so the user's input is massaged into the appropriate expression syntax, thereby allowing a natural input like `locate -a \proj1\src\p*.cpp` *One exception:* the use of '$' (end-of-line metacharacter), consistent with most regexp and Linux locate.

Note that, because of the structure of the database, ^ is not required (i.e., ^foo == foo*). Also because of the database structure, ? is not a valid wildcard (needed a character that couldn't be in a filename) and if present, is deleted from the search string.

If you want more extensive regexp handling, specify the regular expression within double- then single-quotes. This is passed directly to findstr (WinNT+) or egrep (Unix), so refer to those programs for details on format/handling. While there is no findstr in Win98, locate's extensive, custom expression handling makes it quite useful even there.

As on Linux, a warning message is presented if the catalog is >8 days old (and so rerun catalog). As disk content changes, catalog accuracy decreases (hence the warning). The advantage of using catalog/locate is much faster response than recurrent disk searches, especially as the number of files increases.

Suppose Windows Task Manager returns a short (8.3) filename. Try a combination of locate, l2s -strict, and find. *Example:* Assume you have filename checkp~1.exe:

```
[4] C:\scripts> locate checkp*.exe 2>NUL | l2s -strict | find /i "checkp~1.exe"
```

Full perl regexp was considered for this program (as in fs) but was deemed overkill for simply searching a catalog. If you disagree, email us at support@cleanscape.net.

While we could have a switch to confirm the existence of any file before output, this would confer more accuracy to the catalog than should be, since files could have been added/deleted since the last catalog run. If you want confirmation, pipe the output of locate to l2s or fff. For example, to list only .pl files from the catalog that exist today,

```
      locate *.pl | l2s      -or-      locate *.pl | fff
```

Again, this would not handle new .pl files created since the last catalog run.

Bundled with:	catalog
See also:	fff
	l2s
	morePLUS

5. *fff*

USABILITY (1=low, 5=high)			UTILITY		
Developer	Sysadmin	General	Windows	Unix	Linux
5	5	5	5	5	5

Purpose. `fff` stands for "fast file find". As the name implies, it searches for files on-disk and presents the results, either as a simple list or in an interactive session where operations may be conducted on each match. More powerfully, `fff` can also scan within ZIP and tar archives and add matching files to the results list.

A list of files may also be piped to `fff` and then manipulated in the interactive session.

Real-World Uses.

- Windows machines or Unix servers can easily contain 200,000 or more files; searching for individual files of interest can take several minutes, and spanning disks is difficult.
- *Real-world example:* In creating these utilities, significant changes were made and then a snippet of code from a prior version was (desperately) required. Since the prior version was somewhere in a zipped backup, running `fff` found it quickly!

Syntax. The syntax is slightly different depending on host:

```
Win:  fff {[-NI] [-NR] [-ZIP] [-TAR] [(+drive_a..z:) … ]         [path]<name> } |-VER|/?
*nix: fff {[-NI] [-NR] [-TAR] [-ZIP] [+LINKS] [+NFS|(-ftype)…] "[path]<name>"} |-VER|-?
```

name	Required: The filename to search for. Enclose in quotes if Unix and it contains wildcards. May be piped from other programs.
path	Optional: the path to start searching from. The default is the current directory *and* its subdirectories (use –NR to prevent subdirectory search).
+*D*:	Optional, *Windows only:* also search the drive specified, where *D* is any drive letter (a..z). The sequence may be repeated, and the program validates *D* is up and available.
+LINKS	Optional, *Unix only:* Follow symbolic links; default is off.
+NFS *or…* … -*ftype*	Optional, *Unix only:* refine list of filesystems to traverse. By default, the program will search everything *but* nfs mounts. -*ftype* may be repeated.
-NI	Non-interactive: Print the search results to *stdout*, then quit. Note that informational messages are still written to *stderr*, so use 2>NUL or 2>/dev/null if you want only the list of matches with no fluffery.
-NR	Do not recursively search subdirectories during the search. Also applies to -ZIP and -TAR, if specified.
-ZIP	Search within ZIP archives as well.
-TAR	Search within tarballs too.
-VER	Return version information and quit.
-? *or* /?	Print a help listing, then quit.

Returns. If -NI is specified, a list of filenames matching the input specification is returned, written to *stdout*. If not, the same list is loaded into an interactive session (see example [1]). Informational messages are written to *stderr*.

Return codes. 0 for one match (or help, version, or statistics were specified), 1 if no match, 2 if >1 matches; +3 is added to the previous if there are *stderr* messages that should be reviewed; 8 if an error (e.g., input errors) or 10 if severe (e.g., internal) errors.

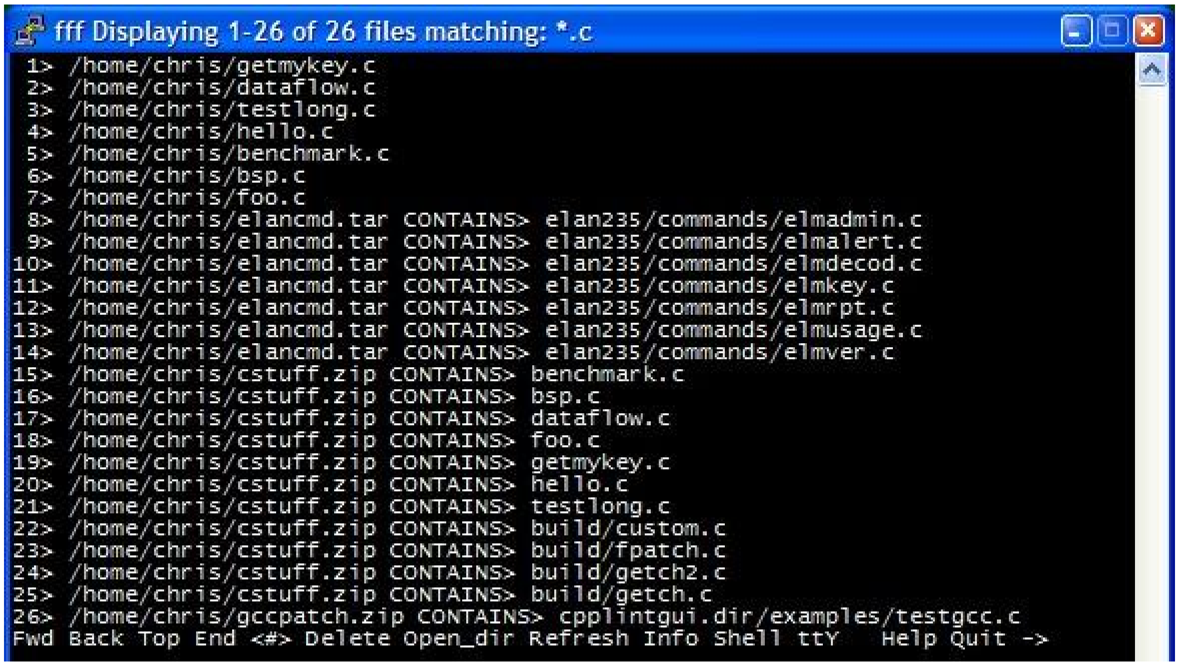
In interactive mode, `fff` presents a numbered list of matches; the user may enter one of the below actions at the prompt at the bottom of the screen. Unnumbered actions take place immediately on the topmost filename displayed. Enter a number followed by <CR> to specify a different file – the one corresponding to that number.

`f`	scroll Forward one screenful of files (default; `<CR>` and `<space>` also work).
`b`	scroll Backward one screenful of files (`–` also works).
`d`	Delete the file (with confirmation); works even if file is in a ZIP or tar archive.
`e`	go to the End of the listing.
`h`	display this Help listing (`?` also works).
`i`	display Information about the selected file and its archive, if applicable.
`o`	Open a shell (Unix) or Explorer (Windows) at the directory containing the file.
`q`	Quit the session.
`r`	*Unix/Linux:* Refresh the display (e.g., after gunzipping a `.tar.gz` to `.tar` in a previous `f` shell so that its contents can be displayed). *Windows:* staRt the file in its associated program (e.g., Excel for `.xls`).
`s`	start a command Shell (`sh` for *nix, `command` for Win98, `cmd` for WinNT+).
`t`	go to the Top of the listing (`1<CR>` also works).
`y`	ttY (interactively list) the file using morePLUS (`m` also works).

Numbers and commands can be combined a la <number><action_letter><CR>. In Windows, for instance, `8r<CR>` starts file #8 using its associated program (e.g., Excel for `.xls`).

Examples. Each is numbered [x]. Additional examples of `fff`'s interactive capabilities are shown in the Examples sections for `locate` and `fsi`.

[1] *A sample `fff` (Linux) session is depicted below. If the user entered `8d<CR>` at the prompt, file #8 in the tar archive, or the entire archive, could be deleted – after a confirmation dialog.*



```
fff Displaying 1-26 of 26 files matching: *.c
 1> /home/chris/getmykey.c
 2> /home/chris/dataflow.c
 3> /home/chris/testlong.c
 4> /home/chris/hello.c
 5> /home/chris/benchmark.c
 6> /home/chris/bsp.c
 7> /home/chris/foo.c
 8> /home/chris/elancmd.tar CONTAINS> elan235/commands/elmadmin.c
 9> /home/chris/elancmd.tar CONTAINS> elan235/commands/elmalert.c
10> /home/chris/elancmd.tar CONTAINS> elan235/commands/elmdecod.c
11> /home/chris/elancmd.tar CONTAINS> elan235/commands/elmkey.c
12> /home/chris/elancmd.tar CONTAINS> elan235/commands/elmrpt.c
13> /home/chris/elancmd.tar CONTAINS> elan235/commands/elmusage.c
14> /home/chris/elancmd.tar CONTAINS> elan235/commands/elmver.c
15> /home/chris/cstuff.zip CONTAINS> benchmark.c
16> /home/chris/cstuff.zip CONTAINS> bsp.c
17> /home/chris/cstuff.zip CONTAINS> dataflow.c
18> /home/chris/cstuff.zip CONTAINS> foo.c
19> /home/chris/cstuff.zip CONTAINS> getmykey.c
20> /home/chris/cstuff.zip CONTAINS> hello.c
21> /home/chris/cstuff.zip CONTAINS> testlong.c
22> /home/chris/cstuff.zip CONTAINS> build/custom.c
23> /home/chris/cstuff.zip CONTAINS> build/fpatch.c
24> /home/chris/cstuff.zip CONTAINS> build/getch2.c
25> /home/chris/cstuff.zip CONTAINS> build/getch.c
26> /home/chris/gccpatch.zip CONTAINS> cpplintgui.dir/examples/testgcc.c
Fwd Back Top End <#> Delete Open_dir Refresh Info Shell ttY   Help Quit ->
```

[2] `C:\> fff \back*.wbk`
Display all Microsoft Office backup files in an interactive `fff` session. Very useful to find/delete such backups that are no longer desired. Use Info (`i`) to differentiate them.

[3] `~$ fff -nr -TAR -ZIP "*.pl"`
displays all instances of `.pl` in the current directory, *including* inside zipfiles and tarballs but *excluding* subdirectories, within an interactive `fff` session.

[4] `C:\temp> fff +c: +d: +e: -ZIP -TAR \*.pl`
a "mother of all searches": find *all* instances of `.pl` on *all* listed drives in *all* directories, *including* inside ZIP *and* tar archives. You might make some popcorn during the wait, but if you find that one damn file you need, `fff` has done its job :-)

[5] `~/scripts$ fff -ni "*.pl" 2>/dev/null`
displays all instances of `.pl` in the current directory and subdirectories, then quits with no interactive `fff` session. Useful for scripting applications where only results are desired, but be sure to check the return code!

[6] `C:\batch> fff -ni *.bat 2>NUL | l2s`
displays all instances of `.bat` in the current directory and subdirectories, routes the resulting file list to `l2s`, which will return non-spaced filenames.

Programming Notes.

The "fast" part of `fff` comes from the simple interface and OS calls nearer the hardware level, rather than deal with the Windows wagging dog in the GUI search. `fff` does a full search of the disk(s) each time it is called; depending on the size of the search, it can be very time-consuming, although disk caching makes subsequent runs much faster.

If you want quicker results, check out gr8 utilities `catalog` and `locate`, which create/access a catalog of the disk. Using them, you're searching a catalog rather than the disk in real-time; read more about them in their respective chapters of this document.

A word about gr8util morePLUS: by bundling it with `fff`, numerous additional options are available for viewing/manipulating the selected file. For instance, it's possible to open the file in the user's favorite editor; this means that a file inside a tar archive could be opened, edited, then saved to a location on the disk (and then manually updated into the tarball in a separate `f` shell session). Read more about morePLUS in its chapter in this document.

If your system does not have `unzip` or `tar`, or your `tar` cannot delete files within a tarball (most Unices), `fff` invokes Info-zip `unzip` or gnu `tar`, both located within the gr8utils `bin` subdirectory. These can be used freely, subject to their respective license agreements.

Owing to its Unix roots, files within a `tar` archive are case-sensitive, and `fff` respects this rule. On Windows, searches within ZIP archives are case-insensitive (`unzip -C` option).

If you're searching multiple drives on Windows, be sure any path you specify exists on all of them, or `cd` to the desired directory on each drive and then don't specify a path. If no drive letter appears in the search path and no `+D:` construct is used, the current drive is automatically selected.

On Unix, if you're accessing a remote system, you may need `+NFS` to get all results.

There is a bug in the `for` and `dir` commands in Windows NT/2k/XP/Vista: if the file extension is exactly three characters long, Windows will match *any extension starting with those three characters*. `fff` contains a fix for this bug; if you need to match the Windows bug (i.e., you want the greedy-style match), add a '*' to the end, e.g., `*.bat*`
See the Programming Notes section for gr8 utility `fs` for a complete discussion.

Bundled with: morePLUS

See also:	`fsi`
	`locate`

6. *fsi*

USABILITY (1=low, 5=high)			UTILITY		
Developer	Sysadmin	General	Windows	Unix	Linux
3	5	3	5	5	5

Purpose. `fsi` provides a list of the largest or most recent files on-disk. In full, the disk/filesystem information provided can be:

- ● Information about all mounted drives (type, file system, size, free space),
- ● The number of files and subdirectories encountered during the walk,
- ¤ The full path of the newest files encountered in the directory walk, *–or–*
- £ The largest directories by bytesize and filecount, *–and–*
- £ The full path of the largest files encountered.

The filelist can be routed to gr8util `fff` using the `–TOFFF` option, or can be piped to `fff` (using `-Q`) or `moreplus -getcol 22`; `–TOFFF` is best since no other operands are required and you can preview the list – and possibly add more results – beforehand.

Real-World Uses.

- Provides easy answers to questions not otherwise readily obtainable, such as:
  - Where are the unwanted picture files, old installers, spam ad-tachments or other disk clutter that I can delete to recover space?
  - What did that last installation put on my computer, and where did it put them?
  - What programs/directories are the most taxing to my filesystem and are they worth it?
- *Real-world example:* To retrieve disk space this author routinely runs `fsi` to find the biggest disk hogs, such as installers or ZIP downloads that are no longer necessary.
- *Bonus example:* When testing `fsi` on a Unix system, yours truly noticed some test files that were *huge* (>20 MB) when they should have only been a few hundred bytes. Turns out the test program had a malformed directory path and the results were garbage! As a reward for finding it, I got to rerun the tests…
- *Final example and why `fsi` was created:* When installing a product from Microsoft, one of our developers noticed the MS installer was writing files willy-nilly to an empty, second hard drive – one he was going to install Linux on for a dual-booting! (He had specified the installation directory and made no mention of the second drive. The only way to solve this was to *disable* the second drive in Device Manager!)

Syntax. The syntax is slightly different depending on host:

```
Win:  fsi { (-SIZE|-DATE|-EURO|-ISO) [-CT n] [root] [-ALL] [ {(+|-)drive_a..z:} … ]
          [-REVERSE] [-SKIP 'dir' …] [-DI] [-Q] [-TOFFF] } | -VER | (/?|-?)

*nix: fsi { (-SIZE | -DATE)          [-CT n] [root] [-ALL] [+LINKS]
          [-REVERSE] [-SKIP 'dir' …] [-DI] [-Q] [-TOFFF] } | -VER | (/?|-?)
```

root	The starting directory; default is the current directory.
`+D:` *or* `-D:`	Optional, Windows only: Add or exclude drive `D:` to/from the search. `D` is any valid drive letter (a..z). Mixed sequences of `+D:` and `-D:` may be repeated. `fsi` validates that added drives are up and available.
`-ALL`	*Windows:* search all local drives. Use `+D:` to specify, e.g., network drives. *Unix:* search all filesystems; really only matters if *root* is '/'.
`+LINKS`	Optional, *Unix only:* Follow symbolic links; default is off.
`-SIZE`	List the largest files (default).

`-DATE`	List newest files instead; takes precedence over `-SIZE`. Use `-EURO` for date formats *dd/mm/yyyy* or `-ISO` for date formats *yyyy/mm/dd*. `-EURO` or `-ISO` infer `-DATE`. Default format is USA: *mm/dd/yyyy*. *Win Note:* You *must* specify the date format if other than USA so `fsi` can read system dates correctly! *nix uses `-ISO` and cannot be overridden.
`-REVERSE`	Reverse sort order: smallest (-SIZE) or oldest (-DATE) files.
`-CT` *n*	Set file list count to *n*; default is 20.
`–TOFFF`	When complete, invoke gr8util `fff` (if present) to operate on the file list.
`-SKIP` *dir*	Omit files in (sub)directories named *dir*. Enclose in single quotes (escaped in *nix) if *dir* contains spaces. *NOTES:* Sequence may be repeated as necessary; on Windows, position in single-, *then* double-quotes if the dir name has multiple spaces in a row, e.g., `"'my    data'"`
`-DI`	List basic filesystem information (default off).
`-Q`	Quiet: list only results and set return code.
`-VER`	Return version information and quit.
`-?` *or* `/?`	Print a help listing, then quit.

Returns. A sorted list of filenames along with the relevant size or date data. The resulting list can also be routed to the `fff` program for interactive processing using `–TOFFF`.

Everything, including informational messages, is written to *stdout*. `-DI` also returns file system information derived from system utilities. If `-Q` is specified, only the filenames are returned, including `-DI` information if requested.

Return codes. 0 for successful operation (or help/version info requested), 1 if the specified directory is not found. +3 is added if there are informational messages that should be reviewed. 8 is returned if the specified path was invalid; 10 if internal error was detected.

Examples. Each is numbered [x]. The default is 20 files; 10 was used here to save space.

```
[1] C:\> fsi \windows -DI
    Gathering system info. If you have network drives, this could take ~30 sec...

    System Name: DELL_LAPTOP

    Drive C:
       Drive Type  Fixed local disk
       Status      Online
       File System NTFS
       Drive Size    33.64 GB
       Free Space    12.03 GB

    Drive E:
       Drive Type  Removable disk
       Status      Online
       File System FAT
       Drive Size    62.31 MB
       Free Space    52.09 MB

    Drive L:
       Drive Type  Network drive
       Status      Offline

    Processing 45,483 lines; please wait...

    --- File System Info for tree rooted at C:\windows ---

    Total number of files      : 29,280
    Total number of directories:  2,314
```

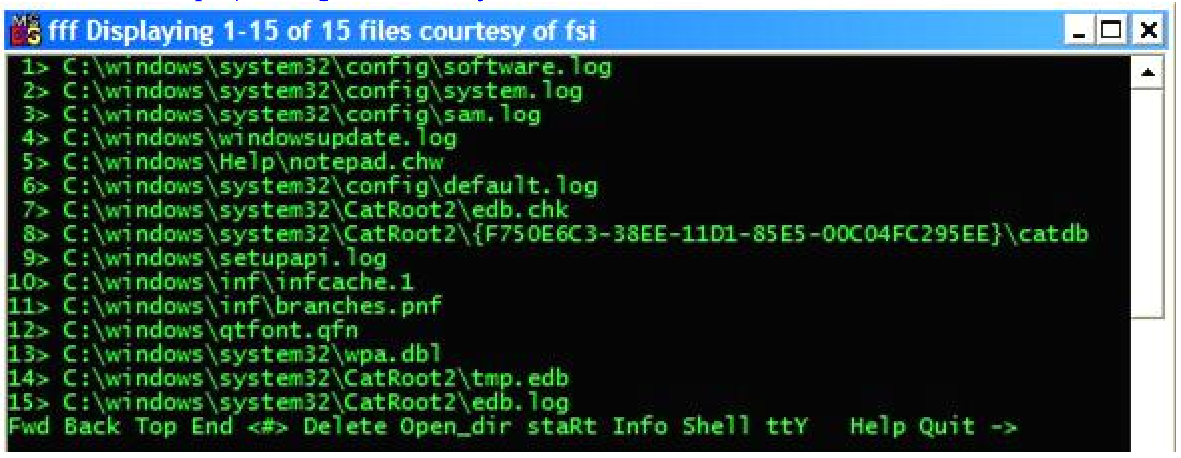
```
Dir most files  :     3,530  C:\windows\$NtServicePackUninstall$
Dir largest size: 996.34 MB  C:\windows\Installer\$PatchCache$\Managed\0E8BA7349
6BF22242B086AF4D32E5219\8.0.50727

The Top 10 Biggest Disk Hogs are:
  1   455.08 MB  C:\windows\Installer\33a33522.msp
  2    73.15 MB  C:\windows\Driver Cache\i386\driver.cab
  3    46.25 MB  C:\windows\system32\config\software.sav
  4    43.50 MB  C:\windows\system32\config\software
  5    42.46 MB  C:\windows\repair\software
  6    37.92 MB  C:\windows\system32\wbem\Repository.001\FS\objects.data
  7    22.75 MB  C:\windows\Driver Cache\i386\sp3.cab
  8    22.75 MB  C:\windows\ServicePackFiles\i386\sp3.cab
  9    22.75 MB  C:\windows\SoftwareDistribution\Download
                 \dd9ab5193501484cf5e6884fa1d22f9e\sp3.cab
 10    21.21 MB  C:\windows\Driver Cache\i386\sp2.cab
```

```
[2a] C:\> fsi \windows -DATE
     Processing 45,483 lines; please wait...

     --- File System Info for tree rooted at C:\windows ---

     Total number of files      : 29,280
     Total number of directories:  2,314

     The Top 10 Freshest Files are:
       1  07/04/08 11:48  C:\windows\Prefetch\gr8utils.exe-12f83a80.pf
       2  07/04/08 11:48  C:\windows\Prefetch\iptdcd.exe-23f5ce85.pf
       3  07/04/08 11:48  C:\windows\Prefetch\cmd.exe-034b0549.pf
       4  07/04/08 11:48  C:\windows\Prefetch\iptlma.exe-359f7867.pf
       5  07/04/08 11:48  C:\windows\Prefetch\fsum.exe-28288389.pf
       6  07/04/08 11:48  C:\windows\system32\config\software.log
       7  07/04/08 11:46  C:\windows\Prefetch\find.exe-0eead1a7.pf
       8  07/04/08 11:45  C:\windows\Prefetch\wmiprvse.exe-0d449b4f.pf
       9  07/04/08 11:45  C:\windows\Prefetch\cscript.exe-0a13a05c.pf
      10  07/04/08 11:45  C:\windows\Prefetch\fsi.exe-10185691.pf
```

```
[2b] C:\> fsi \windows -DATE -skip prefetch -skip softwaredistribution -skip wbem
          -skip pchealth       //Successively added –skips eliminate unwanted results

     --- File System Info for tree rooted at C:\windows ---
         Skipping directories whose names begin with:
         prefetch
         softwaredistribution
         wbem
         pchealth

     Total number of files      : 22,928     //See how these change as directories were
     Total number of directories:  2,308     //reduced by the –skip directives

     The Top 10 Freshest Files are:
       1  07/04/08 11:51  C:\windows\system32\config\software.log
       2  07/04/08 10:35  C:\windows\system32\config\system.log
       3  07/04/08 09:43  C:\windows\system32\config\sam.log
       4  07/04/08 09:35  C:\windows\windowsupdate.log
       5  07/03/08 22:12  C:\windows\Help\notepad.chw
       6  07/03/08 18:44  C:\windows\system32\config\default.log
       7  07/02/08 08:13  C:\windows\system32\CatRoot2\edb.chk
       8  07/02/08 08:13  C:\windows\system32\CatRoot2
                          \{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\catdb
       9  07/02/08 08:11  C:\windows\setupapi.log
      10  07/02/08 08:11  C:\windows\inf\infcache.1
```

```
[3] C:\> fsi \windows -DATE -skip prefetch -skip softwaredistribution -skip wbem
         -skip pchealth -Q -CT 15 -TOFFF
```

-TOFFF instructs fsi to load program fff with the resulting file list of 15 names, clear the screen, and start the fff session below. –Q suppresses the normal fsi output (seen in the previous example) and goes directly to fff.



```
 1> C:\windows\system32\config\software.log
 2> C:\windows\system32\config\system.log
 3> C:\windows\system32\config\sam.log
 4> C:\windows\windowsupdate.log
 5> C:\windows\Help\notepad.chw
 6> C:\windows\system32\config\default.log
 7> C:\windows\system32\CatRoot2\edb.chk
 8> C:\windows\system32\CatRoot2\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\catdb
 9> C:\windows\setupapi.log
10> C:\windows\inf\infcache.1
11> C:\windows\inf\branches.pnf
12> C:\windows\qtfont.qfn
13> C:\windows\system32\wpa.dbl
14> C:\windows\system32\CatRoot2\tmp.edb
15> C:\windows\system32\CatRoot2\edb.log
Fwd Back Top End <#> Delete Open_dir staRt Info Shell ttY   Help Quit ->
```

Programming Notes.

This program can be very useful to sysadmins who are looking for the largest files or directories on a system, perhaps to recover disk space. It can also be useful to anyone who wants to know what files were most recently added to the system, perhaps during a new program's installation. See the examples in the introduction for why we like it.

The best approach is successive iterations, adding –SKIP *dir* in each iteration to eliminate directories that are not of interest, such as shown in examples 2a and 2b above.

fsi is different from the other utilities in that it has a –QUIET mode of operation: to suppress informational output, use –Q rather than redirect *stderr* to NUL or /dev/null. The tabular data (date and size info) must be suppressed if only the filenames are desired. Just remember to check those return codes and rerun without –Q if RC>2!

If you only want the nice -DI output, add -CT 0 (zero) to avoid the data gathering process.

A nicety: If you are scanning an entire disk or multiple disks (Windows), or from the root directory (*nix), and the specified count is <100, fsi goes ahead and stores the top 100 files in the likely event you want to see further down the list, or even save the list. This is done by looping through the 100 files *count* at a time, querying you at the start of each loop. For example, with –CT 15 (but not –QUIET) specified on the command line,

```
Since this was an entire-disk search, we've stored 100 values, just in case.
Would you like another 15 entries, or save the results to file? (y/n/s)
```

If –TOFFF was specified, any additional files viewed are also forwarded to fff. None of this interactivity occurs if –QUIET was specified.

If the filenames are *very* long on Windows, fsi will invoke the –STRICT mode of l2s to return names in 8.3 format, as can be seen in examples 1a and 1b above.

The significance on Unix/Linux is regarded as less than that on Windows, as it is possible to use system utilities to replicate the functionality. However, the linkage to other utilities like fff and morePLUS is a powerful addition, and it is a pre-built, tested program.

See also: fff

7. *fs*

USABILITY (1=low, 5=high)			UTILITY		
Developer	Sysadmin	General	Windows	Unix	Linux
5	3	3	5	5	5

Redistribution Licensing Available

Purpose. fs is probably the handiest gr8util. It searches specified files (possibly with wildcards) for a given search text (possibly a perl regular expression), then routes the matches to morePLUS so that they can be interactively browsed.

Real-World Uses.

- Scan large source code bases for particular strings, using regular expressions to really narrow down the false positives – a vast time-saver compared to find or grep!
- Quickly locate keywords in HTML, XML, text, or log files.
- Anyplace you need to sift through mounds of words or phrases and time is short.
- *Real-world example:* We had to locate all the places where an epoch issue in our 3rd party license manager arose in ~100 files including headers. Windows' findstr produced too many false positives: its regexp engine is poor (as are some Unix greps) and having too many results to reject increased the chances of missing a valid result.

Syntax. The syntax is slightly different depending on host:

```
Win:  fs { [/r] [/i] [/s] [/w] [/c] [/n] [(+drive_a..z:) … ]
         ( <[path]fileset>    | -FL <e_v> ) <"string"> } | -VER | /?
*nix: fs { [-r] [-i] [-s] [-w] [-c] [-n] [+LINKS] [+NFS | (-ftype) …]
         ( <"[path]fileset"> | -FL <e_v> ) <"string"> } | -VER | -?
```

fileset	Required: the list of files to search; may contain wildcards	*–or–*
-FL *e_v*	Required: a list of filenames (separated by ';' if Windows or ':' if Unix) specified in environment variable *e_v*	*–and–*
string	Required: the search string; may be a perl regular expression.	
path	Optional: the path to start searching from. The default is *only* the current directory, *not* its subdirectories (use -r for recursion into subdirectories).	
+*D*:	Optional, *Windows only:* search drive *D*, where *D* is any drive letter (a..z). The sequence may be repeated, and fs validates *D* is up and available.	
+LINKS	Optional, *Unix only:* Follow symbolic links; default is off.	
+NFS *or*… … -*ftype*	Optional, *Unix only:* refine list of filesystems to traverse. By default, the program will search everything *but* nfs mounts. -*ftype* may be repeated.	
-r *or* /r	Recursively search within directory and all its subdirectories.	
-i *or* /i	Ignore case.	
-s *or* /s	Span search across newlines within each source file.	
-w *or* /w	single blank in search string matches Whitespace = possibly multiple blanks or tabs, but *not* newlines (combine with -s if that is desired).	
-c *or* /c	provide one line of Context before and after the matching line.	
-n *or* /n	Non-interactive: Print the search results, then quit. Note that informational messages are still written to *stderr*, so use 2>NUL or 2>/dev/null if you want only the list of matches with no fluffery.	
-VER	return version information and quit.	
-? *or* /?	Print a help listing, then quit.	

Returns. If -n is specified, a list of file names with their matching text lines is returned, written to *stdout*. The number of the matching line in the file is prefixed to the actual text line. Without -n, the same list is loaded into an interactive morePLUS session. Informational messages are written to *stderr*.

Return codes. 0 for one match (or help, version, or statistics were specified), 1 if no match, 2 if >1 matches; +3 is added to the previous if there are *stderr* messages that should be reviewed; 8 if an error (e.g., input errors) or 10 if severe (e.g., internal) errors.

Examples. Depicting fs of is a little difficult since it usually outputs to morePLUS. The first example uses –n just to show the "normal" format of the program's output, including informational messages being posted to *stderr*. Each example below is numbered [x].

```
[1] C:\perl\> fs -n /r /i /e "lib\time\*.pm" time
    WARNING: Parameter not recognized and ignored: /e
    Searching 'lib\time\*.pm' for 'time'
    Options: non-interactive recursive ignore-case NT-ext-fix
    4 files match fileset criterion; beginning search...
```

For information on NT-ext-fix, *see the Programming Notes section.*

Results below are illustrative and have been truncated to save space. Note that the name of the file containing the match line(s) precedes each block of matches. The number in brackets (e.g., [9]*) is the line number in that file containing the match.*

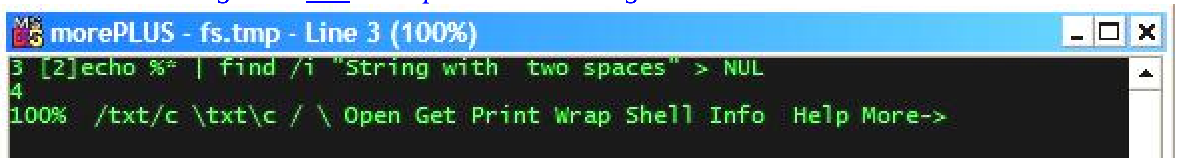
```
---------- C:\perl\lib\time\gmtime.pm
[1]package Time::gmtime;
[3]use Time::tm;
[9]    @ISA         = qw(Exporter Time::tm);
[23]    my $tmob = Time::tm->new();
[32]sub gmtime (;$)    { populate CORE::gmtime(@_ ? shift : time)}

---------- C:\perl\lib\time\Local.pm
[1]package Time::Local;
[144]These routines are the inverse of built-in perl fuctions localtime()
[145]and gmtime().  They accept a date as a six-element array, and return
[146]the corresponding time(2) value in seconds since the Epoch (Midnight,

. . .
```

```
[2] C:\chris\batch> fs -FL BATLIST /r /w "find /i \"String with two spaces\""
    Searching %BATLIST% for 'find /i "String with two spaces"'
    Options: recursive blanks-eq-whitespace NT-ext-fix
    1 results were found in 9 files.
    Ready to invoke morePLUS...
```

fs parses ';' (Windows) or ':' (Unix) -delimited environment variable BATLIST for the list of files. Without –n, fs then sends the results to morePLUS as depicted below (for details on morePLUS, see its chapter). It also turns on line numbering for user convenience (the first column) and has pre-aligned morePLUS to the first (in this case, only) match in the results file.

If the user enters g (Get) at the prompt (last line), the file containing the match will be opened in the user's specified editor to that line (2 in this case). Because of the pre-alignment, the filename is offscreen; you could enter – or < at the prompt to scroll backwards.

/w literally converted each single space in the search string to "\s+" (with some caveats; see the Programming Notes section) and thus found the string with two spaces in it. Note that escaped double-quotes are handled appropriately, and that /i was correctly incorporated in the search string and not interpreted as the ignore-case directive.



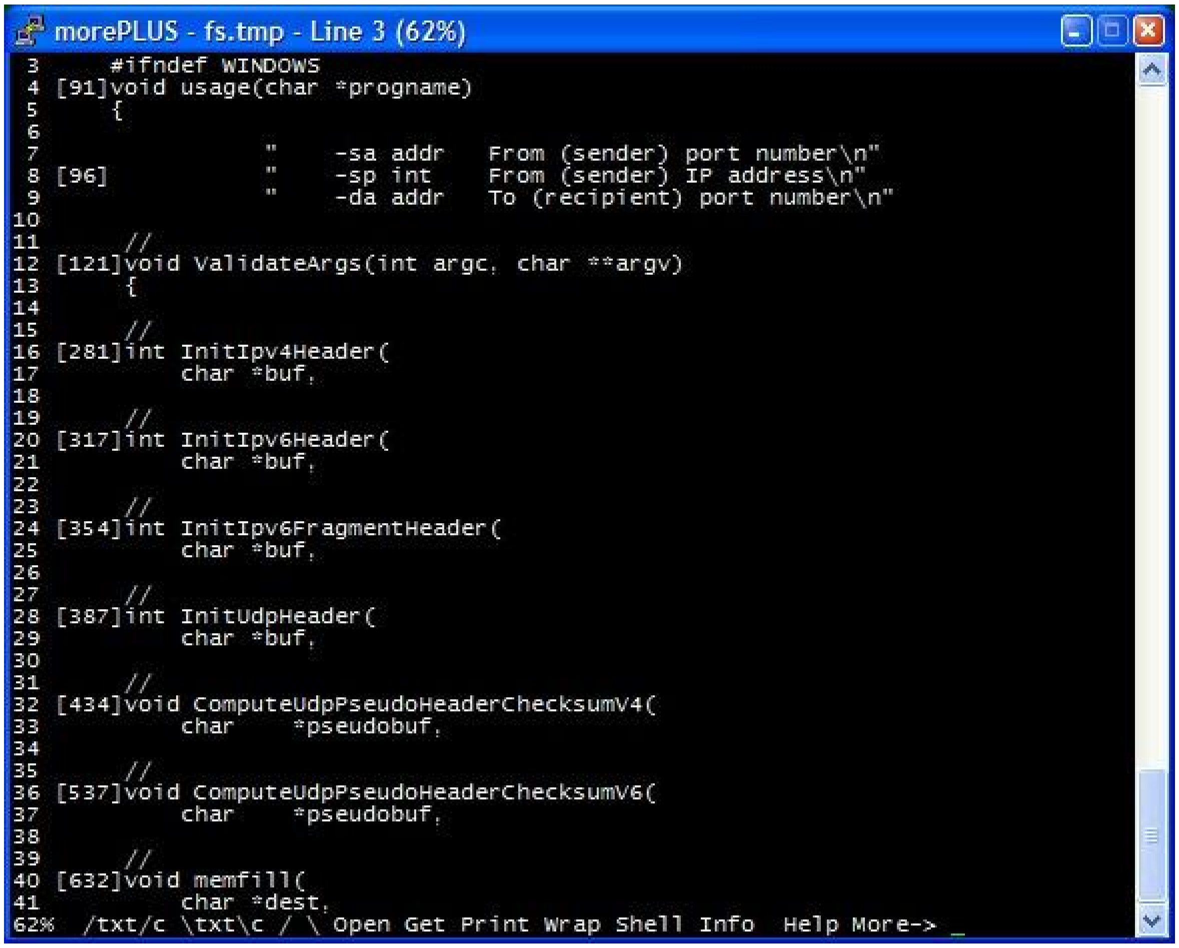
```
3 [2]echo %* | find /i "String with  two spaces" > NUL
4
100%  /txt/c \txt\c / \ Open Get Print Wrap Shell Info  Help More->
```

```
[3] redhat:~$ fs -w -s -c "*.c" "(int|void) \w+ \(.*?\).*?\{"
    Searching '*.c' for '(int|void) \w+ \(.*?\).*?\{'
    Options: blanks-as-whitespace  span-newlines  provide-context
    15 results were found in 2 files.
    Ready to invoke morePLUS...
```

The user is applying regular expressions to find function declarations; for an improved regexp with fewer false positives, see the Programming Notes Example [4].

Note that, because of –CONTEXT, morePLUS is aligned at the line prior to the match line. Usefully, `fs` has preloaded the “find text” field, so the user can enter / at the prompt (bottom of screen) to jump to line 4, the first match. We can see that this programmer (not me) prefers her/his ‘{‘ to reside on a separate line and indeed we’re searching until a ‘{‘ is found.

If the user then enters g, the file (whose name is offscreen because of pre-alignment) will be opened in the user’s preferred editor to line number 91. Sequences of / and g may be repeated to open each file at the line matching the search criteria supplied to `fs`.



```
 3     #ifndef WINDOWS
 4 [91]void usage(char *progname)
 5     {
 6
 7                    "    -sa addr   From (sender) port number\n"
 8 [96]               "    -sp int    From (sender) IP address\n"
 9                    "    -da addr   To (recipient) port number\n"
10
11       //
12 [121]void ValidateArgs(int argc, char **argv)
13       {
14
15       //
16 [281]int InitIpv4Header(
17          char *buf,
18
19       //
20 [317]int InitIpv6Header(
21          char *buf,
22
23       //
24 [354]int InitIpv6FragmentHeader(
25          char *buf,
26
27       //
28 [387]int InitUdpHeader(
29          char *buf,
30
31       //
32 [434]void ComputeUdpPseudoHeaderChecksumV4(
33          char    *pseudobuf,
34
35       //
36 [537]void ComputeUdpPseudoHeaderChecksumV6(
37          char    *pseudobuf,
38
39       //
40 [632]void memfill(
41          char *dest,
62%  /txt/c \txt\c / \ Open Get Print Wrap Shell Info  Help More-> _
```

Programming Notes.

I believe it was Chaucer who wrote,

Once thou goest perl regexp,

Goest thee ne’er hinward.

If you know that “regexp” means “regular expressions” and you don’t need a primer, proceed to ♣♣♣ later in this section.

Maybe the mention of "regexp" has your eyes glazing over; relax! First of all, as you can see in example [1], you don't have to do anything different than you would if using the operating system's `find` or `grep` commands. And some of the regexp-iness is handled by `fs` with switches like `/i` and `/w`, as seen in example [2].

But if you really need some power lifting like that shown in example [3], regular expressions can't be beat, and perl expressions in particular are the most powerful and flexible out there.

Let's do one real-world tutorial here to show how regular expressions can really narrow down search results. Here is a sample perl regular expression, a more refined search than Example [3]. Note that in `fs` you only specify the part in teal.

```
m/^[ \t]*(int|void|char)\s+\w+\s*\(.*?\).*?\{/gms
```

- There are three sections to a regular expression, all separated by the same symbol. The choice of symbol is up to you; many people use '/' (as shown in red) while others use '@'. Choose a symbol that is not part of the regular expression you are building!
- The first part of the expression (plum in our example) is called the *operation*. Common values are *m* for "matching" or *s* for "substitute". `fs` uses only *m*.
- The last part of the expression is called the *modifier* (blue in the example) and is used to refine the operation results. They can be any combination of *g*, *i*, *s*, or *m*. fs always specifies *g* (global) and *m* (multi-line); *i* (case-insensitive) and *s* (span newlines) can be specified by the user; in `fs`, it's by adding the `–i`(`/i`) or `–s`(`/s`) switches.
- The part in the middle (teal) is the regular expression to be matched. It is read left-to-right, parsing each character one-at-a-time. (A backslash '\' followed by any character together count as one; the combination is known as an "escaped character".)
- Let's use this regular expression to improve the search for function definitions in a C source file (like Example [3] above). It's not a perfect match for any C function (for instance, we've only specified a few return types, and we don't handle being inside comments or quoted strings), but it goes a long way toward being there.

  If you have a C file(s) lying around, try this out yourself. Maneuver to the directory containing C source files; some or all of which contain C functions like

  ```
  int foo(int a, char* b)
      { …
      }
  ```

  The complete statement we will run is

```
[4] C:\src> fs /s *.c "^[ \t]*(int|void|char)\s+\w+\s*\(.*?\).*?\{"
```

  but let's do it in pieces to show how each piece refines the search. First, remember that `/s` instructs `fs` to span across lines; this is for people who (in this author's view, annoyingly) separate the open brace of a function or block on a separate line in their source file. For that matter, `/s` is necessary since C is free-form, and the entire function declaration could be spread across many lines.

  1. `fs /s *.c "^[ \t]*(int|void|char)\s+\w+\s*\(.*?\).*?\{"`
     searches at line starts for zero or more of either a space or the tab character (this eliminates starting in the middle of a word), followed by one of the strings *int*, *void*, or *char*. This will match quite a number of times, especially variable declarations.

2. `fs /s *.c “^[ \t]*(int|void|char)\s+\w+\s*\(.*?\).*?\{”`
   gets us a lot closer. The search is refined to next find one or more whitespaces, followed by at least one valid word character, followed by zero or more whitespaces, followed by an open parenthesis (preceded by ‘\’ because ‘(‘ can also be a regular expression component). This is grabbing the function name, but is not complete; it could match a function declaration in addition to the definition, for instance.
3. `fs /s *.c “^[ \t]*(int|void|char)\s+\w+\s*\(.*?\).*?\{”`
   finishes off a function declaration by looking for the open brace. After the open parenthesis character, allow any characters, followed by the *first* occurrence of a close parenthesis character, then allow any characters, and wind up with the *first* occurrence of an open brace.

   ‘?’ instructs perl to match the *first* occurrence of the next character; perl by default is “greedy”, meaning it would return a string all the way to the *last* occurrence of, in this example, a close parenthesis or an open brace.

There are abundant perl regexp resources on the web; some great places to start are:
http://en.wikipedia.org/wiki/Regular_expression
http://www.perl.com/doc/manual/html/pod/perlre.html
http://perldoc.perl.org/perlretut.html
http://www.cs.tut.fi/~jkorpela/perl/regexp.html

♣♣♣

It should be noted that, while `fs` will accept full perl regexp, complex procedure-like expressions or multiple subexpressions in parentheses produce diminishing results; after all, `fs` is intended to be a generally useful text search utility. Anything more should be custom-coded in a strong regexp language like perl.

Unix shells don't handle unbalanced, escaped quote marks well; for this reason, always enclose search strings containing escaped characters in single quotes.

If you're searching multiple drives on Windows, any path you specify should exist on all of them, or `cd` to the desired directory on each drive and then don't specify a path. If there is no drive letter in the search path and no *+D*: construct is used, the current drive is used.

There is a bug in Windows NT/2k/XP/Vista where the ‘for’ and ‘dir’ commands match extra files. Specifically, if the file extension is exactly three characters long, Windows will match *any extension starting with those three characters*. For example, if your directory contains files `test.bat`, `test.batt`, `test.baty`, and `test.batty`, the command

```
dir *.bat
```

will return all of these files, not just the one named `test.bat`! Note this does not occur if you enter `dir *.ba` *-or-* `dir *.batt`
which indicates that Windows' behavior is at least inconsistent and in our view, a bug.

`fs` contains a fix for this bug and on Windows you will always see `NT-ext-fix` in the search options field, as a reminder this is occurring. If you already know about this bug and your assumption is the greedy-style match, you will see results different from what you expect. To get your anticipated result in this situation, add a ‘*’ to the end, e.g., `*.bat*`

While we could have searched all drives in Windows as some other gr8utils do, we determined this could cause searches to become prohibitively long. If you do need to search multiple disks, simply add their drive letter using the *+D*: construct.

Bundled with: morePLUS

8. morePLUS

USABILITY (1=low, 5=high)			UTILITY		
Developer	Sysadmin	General	Windows	Unix	Linux
4	4	3	5	3	3

Redistribution Licensing Available

Purpose. morePLUS is an interactive file browser, or *pager*, intended to replace the pitiful Windows more command. It is also better than Unix more, and provides more programmatic control than less. Its features include navigation, search, line numbering, opening in your favorite editor… PLUS more!

It is possible to pipe data/filenames or redirect a file into morePLUS.

Real-World Uses.

- Anyplace you'd use more (or less) to view text, source, HTML, XML, or log files.
- Because the pager on Windows is so poor, it may be that many users have never actually used one before. Here are some possibilities:
  - Read through a file one screenful (or specified number of lines) at a time, moving forwards or backwards, or beginning to end, at will. Continue browsing the file until *you* want to quit (not just till you've reached the end of the file!).
  - Search forwards or backwards for a desired text string. Easily repeat the same search throughout the document.
  - Open the file in your favorite editor, to the line you're viewing, with one keypress.
  - Turn on/off word wrap, set tabsize, toggle original indentation on/off, PLUS more!
- *Real-world Example:* if you got a temporary key from us to demo gr8utils, that key is stored in a table along with keys for all the other Cleanscape products. Read about it in Example [2].

Syntax. There are really two modes: (1) operating mode and (2) configuration mode:

Mode 1 `moreplus <`*`file`*`> ['`*`title`*`'] [-INI .`*`ini`*`] [-GETCOL `*`col`*`] [-`*`cmd1`*` -`*`cmd2`*` …] | -VER | -?`

file	Required: the input file you wish to browse. You can also pipe or redirect data/filelists as input to morePLUS.
'title'	text at top of each page (tickmarks required); entering '1' uses the first line of the file as the page title. *nix only: enclose single-quoted string within double quotes, or escape the single quotes.
-USINI	specify a different .ini file besides default <*install_dir*>\moreplus.ini (Windows) or $HOME/moreplus.ini (*nix). Particularly useful on *nix if you sometimes run (shell-only) system console sessions and X GUI sessions other times.
-GETCOL	specify starting *col*umn of filename to be opened by command g *NOTE:* For a detailed example, see the Programming Notes section.
-*cmd1* …	one or more morePLUS commands (see below), each prefixed by '-'.
-VER	return version information and quit.
-? *or* /?	Print a help listing, then quit.

Command Details When Supplied As Parameters:

§ Use formats as described below, except prefix each command with '–'
§ Commands are parsed left-to-right, *except* for info (-i), shell (-s), and quit (-q), which are added at the end.
§ Enclose `->` and `-<` in double quotes to pass through the command interpreter, or use `-.` and `-,` instead.

<u>*Mode 2*</u> `moreplus -CONFIG [ROWS ##] [COLS ##]`

`-CONFIG`	Required: enter/save morePLUS preferences in file `moreplus.ini` located in the installation directory (Windows) or `$HOME` (*nix).
`ROWS`	set the number of rows in the command session (default 24).
`COLS`	set the shell window width (default 80).

Row/col values, if supplied, are confirmed; system values are computed and the user may select that value or specify another one.

In configuration mode, morePLUS will look for editor definitions in an existing `moreplus.ini`, then a `myeditor.lst`, file. If nothing is found (or the user declines previously set values), external program seteditor is invoked for the user to specify the editor name and path. Read more about it in Section 5.2.

Finally, `moreplus.ini` can be manually edited to add lines of the form,

```
    PREF .ext -cmd1 -cmd2 …
```

to set preferences for files based on extension.

For example, if `moreplus.ini` contained the line,

```
    PREF .c -w -n
```

then any C source file (ending in `.c`) would be opened in morePLUS with line numbering on and line wrapping off.

Prefs and command parameters are all applied; flags toggle with each incidence.

Once in the interactive session, the user may then enter one of the below actions at a prompt at the bottom of the screen. Unnumbered and non-search actions take place immediately; numbered (e.g., 12<) and search requests (e.g., /foo) must be terminated with <CR> so that the entire command can be read in.

`f , <CR> , <space>`	scroll Forward one screen
`b` *or* `-`	scroll Backward one screen
`<number>`	go to absolute line number in the file
`[`*N*`]>` *or* `[`*N*`].`	scroll ahead *N* lines, including wraps (default 5)
`[`*N*`]<` *or* `[`*N*`],`	scroll back *N* lines, including wraps (default 5)
/txt/ *or* `\`*txt*`\`	find next *txt* forwards/backwards from here, ignore case
/txt/`c` *or* `\`*txt*`\c`	same as above, Case-sensitive search
/txt/`r` *or* `\`*txt*`\r`	(internal) same as above, strip Regexp symbol '\' from search string
`/` *or* `\`	find next/previous instance of previously specified *txt*
`[`*N*`]a`	equate tAb to *N* spaces for this file (default is 4)
`e`	jump to End of file
`g`	Open file named at column `GETCOL`; type `moreplus -gex` for example
`h`	display this Help listing
`i`	provide Information about this file
`n`	toggle line Numbering on/off
`o`	Open the file using editor specified in `moreplus.ini` or during `-CONFIG`
`p`	Print the file to default device
`q`	Quit
`s`	start a command Shell (`sh` for *nix, `command` for Win98, `cmd` for WinNT+)
`t`	jump to Top of file (same as entering `1<CR>`)
`v`	toggle preserVe indentation - default ON
`[`*N*`]w`	lineWrap – default is ON, 2 spaces. If 0≤*N* ≤9, indent *N* spaces past current indent. Else: toggle linewrap on/off

Returns. An interactive morePLUS session. Informational messages at startup are written to *stderr*, while everything during the interactive session is sent to *stdout*. Given the interactive nature of morePLUS, suppressing warnings is not recommended.

Return codes. 0 for normal exit (or help, version, or statistics were specified), 8 if an error (e.g., file not found) or 10 if severe (e.g., internal) errors were detected.

Examples. Quite possibly the most powerful combination of gr8 utilities are `fs` and `moreplus`; see examples [2] and [3] in the fs chapter. <plug>If you didn't get `fs`, why not order it now? Visit www.cleanscape.net/products/gr8utils today!</plug>

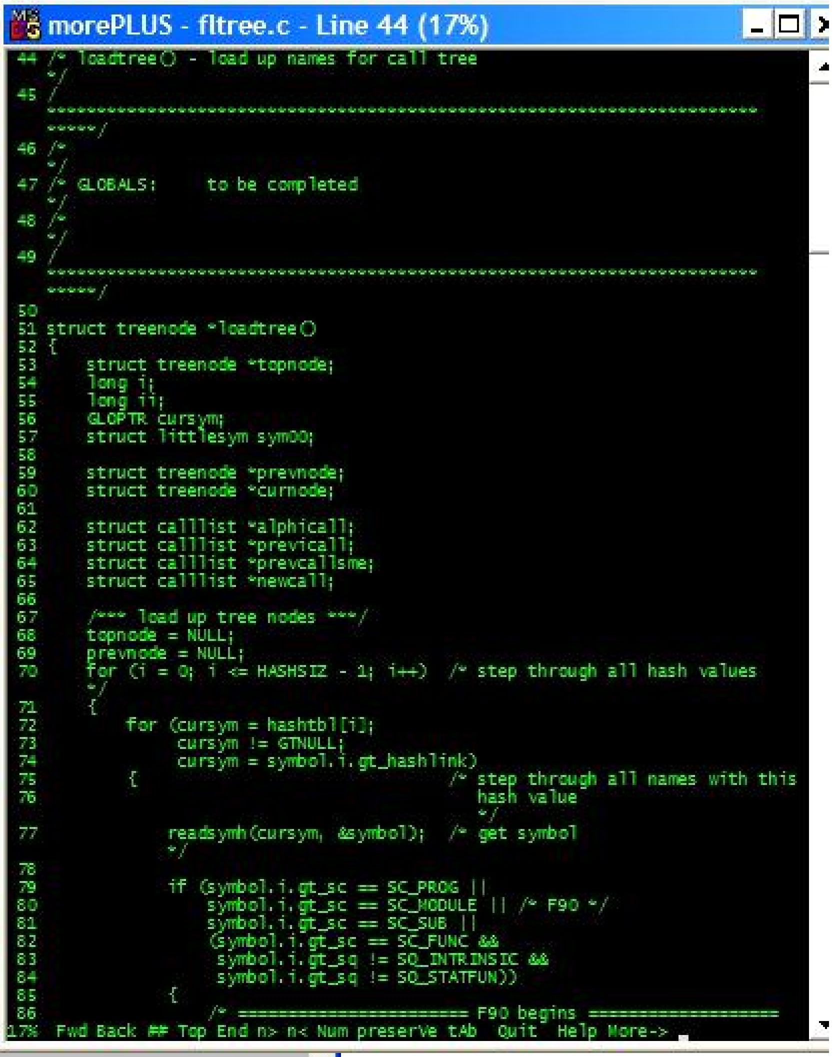
Each example is numbered [x].

[1] `C:\src> moreplus fltree.c`

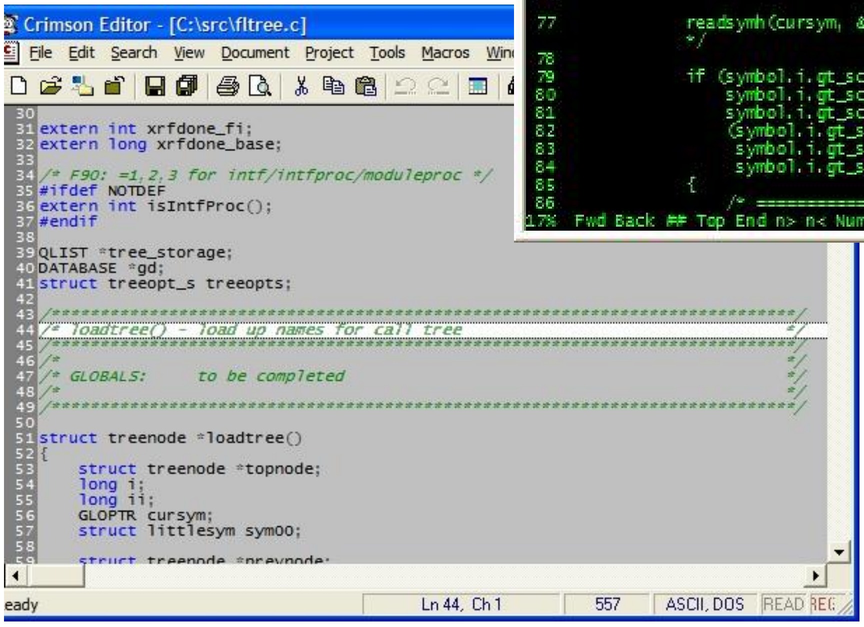
Because numbering (-n) is included in a PREF line in the `moreplus.ini` file, morePLUS opens `fltree.c` (and any .c file) with line numbering on.

As can be seen to the right, morePLUS creates space for the line numbers and wraps lines according to the indentation of the prior line. Wrapped lines are not numbered. Indentation is preserved by default, but this can be toggled by typing v or adding -v on a PREF line in `moreplus.ini`.

Typing o at the prompt will open `c:\src\fltree.c` in the specified editor (Crimson Editor in this case) at the line currently shown at the top of the screen (line 44), as shown below.


morePLUS - fltree.c - Line 44 (17%)

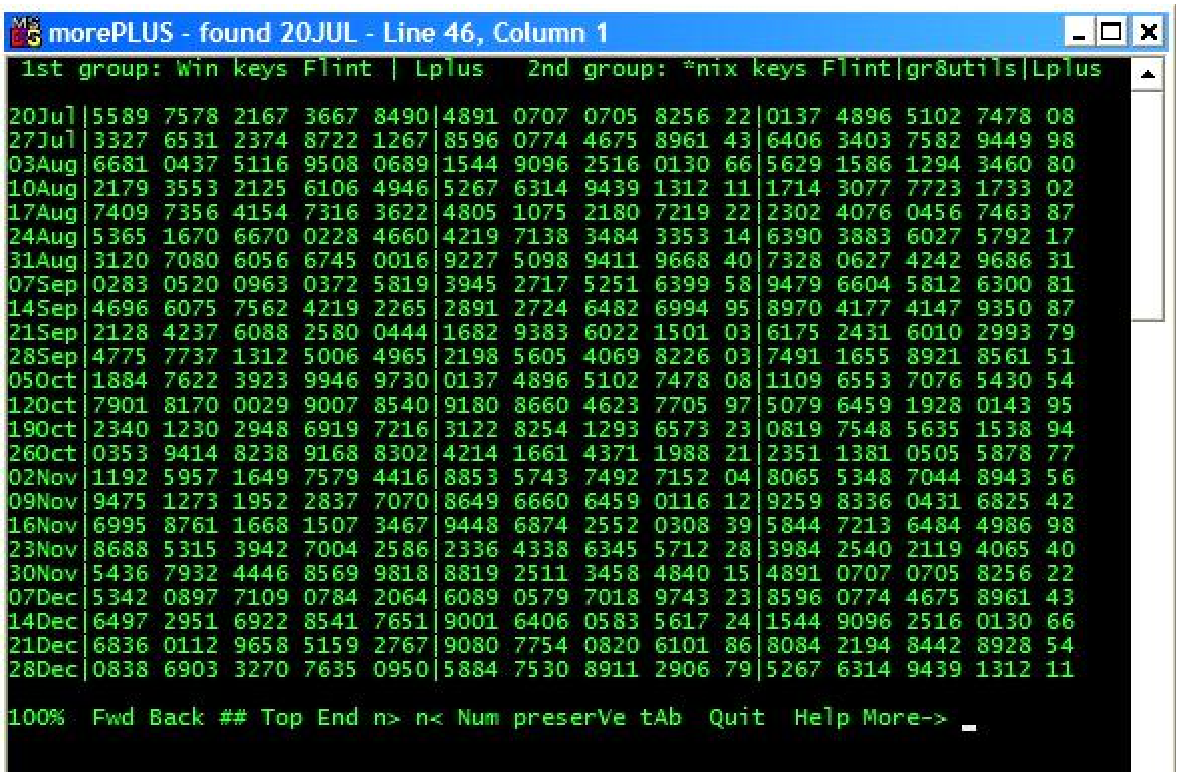
```
44 /* loadtree() - load up names for call tree
   */
45 /
   ****************************************************************************
   *****/
46 /*
   */
47 /* GLOBALS:     to be completed
   */
48 /*
   */
49 /
   ****************************************************************************
   *****/
50
51 struct treenode *loadtree()
52 {
53     struct treenode *topnode;
54     long i;
55     long ii;
56     GLOPTR cursym;
57     struct littlesym sym00;
58
59     struct treenode *prevnode;
60     struct treenode *curnode;
61
62     struct calllist *alphicall;
63     struct calllist *previcall;
64     struct calllist *prevcallsme;
65     struct calllist *newcall;
66
67     /*** load up tree nodes ***/
68     topnode = NULL;
69     prevnode = NULL;
70     for (i = 0; i <= HASHSIZ - 1; i++)  /* step through all hash values
       */
71     {
72         for (cursym = hashtbl[i];
73              cursym != GTNULL;
74              cursym = symbol.i.gt_hashlink)
75         {                                    /* step through all names with this
76                                                 hash value
                                                   */
77             readsymh(cursym, &symbol);  /* get symbol
               */
78
79             if (symbol.i.gt_sc == SC_PROG ||
80                 symbol.i.gt_sc == SC_MODULE || /* F90 */
81                 symbol.i.gt_sc == SC_SUB ||
82                 (symbol.i.gt_sc == SC_FUNC &&
83                  symbol.i.gt_sq != SQ_INTRINSIC &&
84                  symbol.i.gt_sq != SQ_STATFUN))
85             {
86                 /* ====================== F90 begins ===================
17%  Fwd Back ## Top End n> n< Num preserve tAb  Quit  Help More->
```


Crimson Editor - [C:\src\fltree.c]

```
30
31 extern int xrfdone_fi;
32 extern long xrfdone_base;
33
34 /* F90: =1,2,3 for intf/intfproc/moduleproc */
35 #ifdef NOTDEF
36 extern int isIntfProc();
37 #endif
38
39 QLIST *tree_storage;
40 DATABASE *gd;
41 struct treeopt_s treeopts;
42
43 /***************************************************************************/
44 /* loadtree() - load up names for call tree                                */
45 /***************************************************************************/
46 /*                                                                         */
47 /* GLOBALS:     to be completed                                            */
48 /*                                                                         */
49 /***************************************************************************/
50
51 struct treenode *loadtree()
52 {
53     struct treenode *topnode;
54     long i;
55     long ii;
56     GLOPTR cursym;
57     struct littlesym sym00;
58
59     struct treenode *prevnode;
```

```
[2] C:\src> moreplus c:\temp\tempkeys.txt -/20Jul/ -// ‘1’
```

To support demo requests, yours truly uses morePLUS to locate the appropriate row in a keyfile, organized by date and with two sections, one for Windows and one for *nix. I use line 1 of the keyfile as a title so I don’t have to remember which key goes with which product. The morePLUS call is in a script (to compute the expiration), but here’s what the command looks like for fetching a Unix tempkey:



```
 1st group: Win keys Flint | Lplus   2nd group: *nix keys Flint|gr8utils|Lplus

20Jul|5589 7578 2167 3667 8490|4891 0707 0705 8256 22|0137 4896 5102 7478 08
27Jul|3327 6531 2374 8722 1267|8596 0774 4675 8961 43|6406 3403 7582 9449 98
03Aug|6681 0437 5116 9508 0689|1544 9096 2516 0130 66|5629 1586 1294 3460 80
10Aug|2179 3553 2125 6106 4946|5267 6314 9439 1312 11|1714 3077 7723 1733 02
17Aug|7409 7356 4154 7316 3622|4805 1075 2180 7219 22|2302 4076 0456 7463 87
24Aug|5365 1670 6670 0228 4660|4219 7138 3484 3353 14|6390 3883 6027 5792 17
31Aug|3120 7080 6056 6745 0016|9227 5098 9411 9668 40|7328 0627 4242 9686 31
07Sep|0283 0520 0963 0372 5819|3945 2717 5251 6399 58|9479 6604 5812 6300 81
14Sep|4696 6075 7562 4219 2265|2891 2724 6482 6994 95|8970 4177 4147 9350 87
21Sep|2128 4237 6088 2580 0444|1682 9383 6022 1501 03|6175 2431 6010 2993 79
28Sep|4775 7737 1312 5006 4965|2198 5605 4069 8226 03|7491 1655 8921 8561 51
05Oct|1884 7622 3923 9946 9730|0137 4896 5102 7478 08|1109 6553 7076 5430 54
12Oct|7901 8170 0029 9007 8540|9180 8660 4623 7705 97|5079 6459 1928 0143 95
19Oct|2340 1230 2948 6919 7216|3122 8254 1293 6573 23|0819 7548 5635 1538 94
26Oct|0353 9414 8238 9168 8302|4214 1661 4371 1988 21|2351 1381 0505 5878 77
02Nov|1192 5957 1649 7579 4416|8853 5743 7492 7152 04|8065 5348 7044 8943 56
09Nov|9475 1273 1952 2837 7070|8649 6660 6459 0116 12|9259 8336 0431 6825 42
16Nov|6995 8761 1668 1507 3467|9448 6874 2552 0308 39|5844 7213 6484 4986 98
23Nov|8688 5315 3942 7004 2586|2336 4338 6345 5712 28|3984 2540 2119 4065 40
30Nov|5436 7932 4446 8569 9818|8819 2511 3458 4840 15|4891 0707 0705 8256 22
07Dec|5342 0897 7109 0784 2064|6089 0579 7018 9743 23|8596 0774 4675 8961 43
14Dec|6497 2951 6922 8541 7651|9001 6406 0583 5617 24|1544 9096 2516 0130 66
21Dec|6836 0112 9658 5159 2767|9080 7754 0820 6101 86|8084 2194 8442 8928 54
28Dec|0838 6903 3270 7635 0950|5884 7530 8911 2906 79|5267 6314 9439 1312 11

100%  Fwd Back ## Top End n> n< Num preserVe tAb  Quit  Help More-> _
```

If I wanted instead a Windows key, I could enter \ to find the previous 20-Jul!

Programming Notes.

We don’t position morePLUS as a superior pager, but rather a competent pager with superior programmatic control. morePLUS is really a companion utility, a capable extension of one’s favorite code editor and thus a significant productivity enhancer!

The thought was to use the same commands as in Unix `more` or `less`, but because there are so many options, a 1:1 correspondence was not feasible. Besides, if / means “search forward”, doesn't \ make a more natural choice to search backward than ?? (\ is hard in *nix because it’s almost universally the escape-character symbol.)

Since there is a fair amount of reformatting, very large files may take prohibitively long to view in morePLUS. In that case, open the file directly in your favorite editor.

In the current release, there is no regexp searching: you search for literal text only. Vote on this – if you’d really like to have regexp searches in morePLUS, email sales@cleanscape.net!

Bundled with: `fff`
`fs` (by way of `fff`)

PART V MISCELLANEOUS INFORMATION

5.1 ADDITIONAL STEPS FOR WINDOWS 2000

A. Important note

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

B. Details

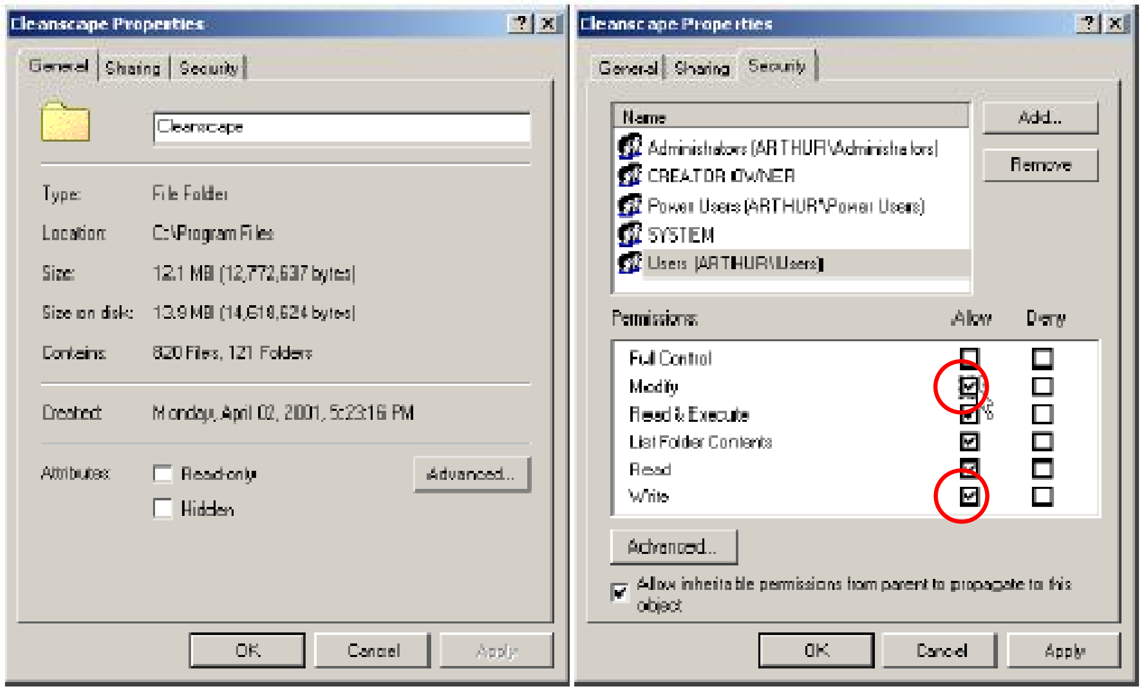
1. For the product to run correctly under Win2k, users must have "write" and "modify" access rights to the installation directory and all its subdirectories. This section explains the procedure used to change the access rights described above.

   a. Log in as "administrator" and finish installing the product.

   b. Double-click on the "My Computer" icon on the desktop.

   c. Double-click installation folder. Select Properties from the sub-menu.

   d. Select "Security" tab on the Properties screen:



Cleanscape

Type:	File Folder
Location:	C:\Program Files
Size:	12.1 MB (12,772,637 bytes)
Size on disk:	13.9 MB (14,618,624 bytes)
Contains:	820 Files, 121 Folders
Created:	Monday, April 02, 2001, 5:23:16 PM

Attributes: ☐ Read-only ☐ Hidden

Name
- Administrators (ARTHUR\Administrators)
- CREATOR OWNER
- Power Users (ARTHUR\Power Users)
- SYSTEM
- Users (ARTHUR\Users)

Permissions	Allow	Deny
Full Control	☐	☐
Modify	☑	☐
Read & Execute	☑	☐
List Folder Contents	☑	☐
Read	☑	☐
Write	☑	☐

☑ Allow inheritable permissions from parent to propagate to this object

   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. The product is now ready to run on Win2k for the "Users" group.

5.2 SPECIFYING AN EXTERNAL EDITOR FOR morePLUS USING SETEDITOR

A. Introduction

By popular demand, Cleanscape has added the ability for users to specify their own favorite editor (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; any contributions will receive appropriate credit and be placed in a "master" file located at http://www.cleanscape.net/products/contributed_editors.html.

B. Operation

The program is invoked automatically if running in morePLUS `-CONFIG` mode and no editors are currently present or selected. To start manually:

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

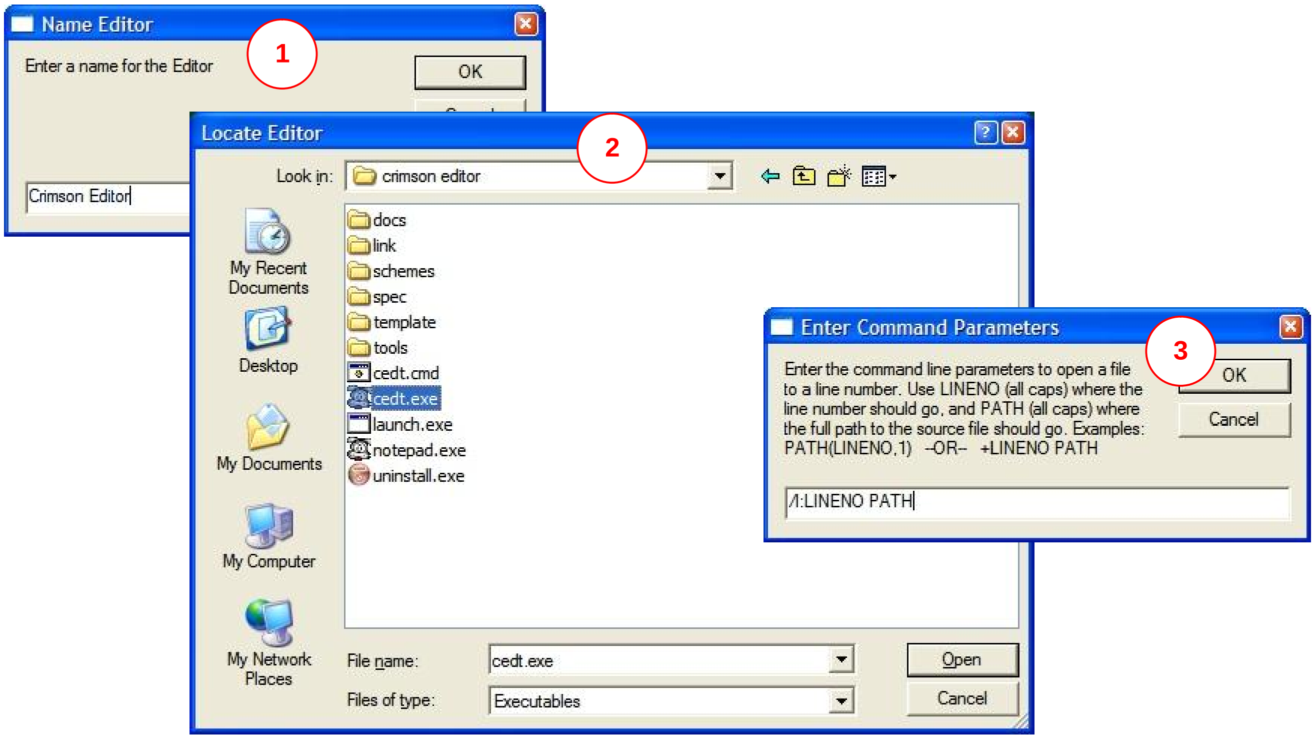
1. Naming the editor (a label identifier),
2. Locating the editor executable itself, and
3. Setting command line parameters to open a file and jump to a line number.

A sample Windows session depicting the dialogs for all three steps (and labeled as such) is shown on the next page, as is a *nix command 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 to the source line you're viewing in morePLUS.

Any number of editors may be added in this fashion; these additions are stored in file `myeditor.lst`, located the installation directory on Windows or your `$HOME` directory if running Unix/Linux. The info for the editor you select for morePLUS is then stored in file `moreplus.ini`. Once successfully added, email your `myeditor.lst` file to support@cleanscape.net for inclusion in a Master file to share with other Cleanscape customers!

It is also possible to edit `myeditor.lst` manually; see the comments inside the file. The Unix/Linux session below shows the contents of `myeditor.lst` (which looks substantially similar under Windows).



Enter a name for the Editor

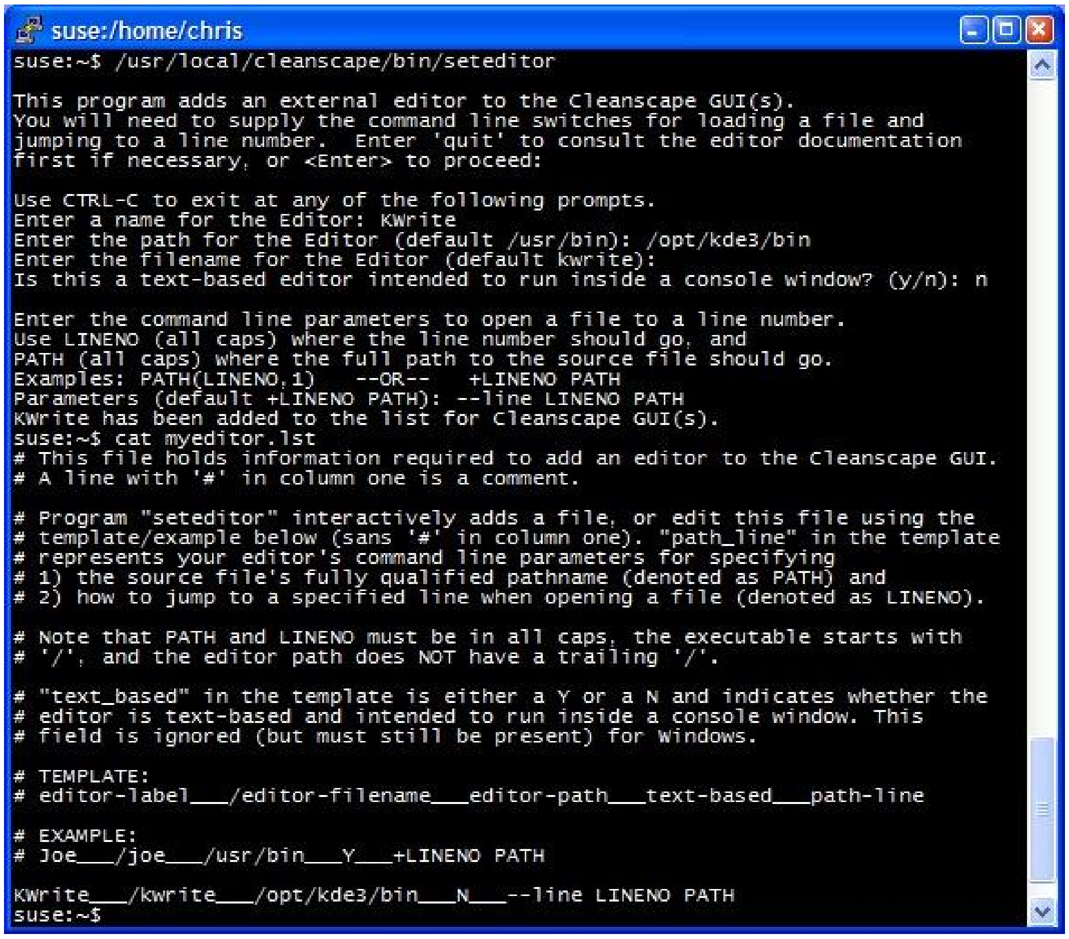
Crimson Editor

File name: cedt.exe

Files of type: Executables

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

/l:LINENO PATH



```
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:~$
```

PART VI No-Nonsense License Agreement

CLEANSCAPE NO-NONSENSE LICENSE AGREEMENT AND LIMITED WARRANTY

IMPORTANT - READ CAREFULLY
This license agreement and limited warranty constitutes a legal agreement ("License Agreement") between you (either as an individual or a corporate entity) and Cleanscape Corporation ("Cleanscape") for the *gr8utils* software product ("Software"), including any software, media, and accompanying on-line or printed documentation.

BY USING THE SOFTWARE, YOU AGREE TO BE BOUND BY ALL THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT. IF YOU DO NOT AGREE WITH THE TERMS AND CONDITIONS OF THIS AGREEMENT, DO NOT USE THE SOFTWARE AND NOTIFY CLEANSCAPE TO OBTAIN A FULL REFUND.

Upon your acceptance of the terms and conditions of the License Agreement, Cleanscape grants you the right to use the Software in the manner provided below.

This Software is owned by Cleanscape or its suppliers and is protected by copyright law and international copyright treaty. Therefore, you must treat this Software like any other copyrighted material (e.g., a book), except that you may either make one copy of the Software solely for backup or archival purposes or install the Software to a single hard disk provided you keep the original solely for backup or archival purposes.

You may transfer the Software and documentation on a permanent basis provided a) you retain no copies; b) the recipient agrees to the terms of this License Agreement; and c) you notify Cleanscape in writing that the transfer has taken place and to whom. Except as provided in the License Agreement, you may not transfer, rent, lease, lend, copy, modify, translate, sublicense, time-share or electronically transmit or receive the Software, media or documentation. To perform any of the above activities, you must enter into a separate agreement with Cleanscape, specifically, either a Reseller's Agreement or a Distribution License Agreement.

You acknowledge that the Software is a confidential trade secret of Cleanscape and/or its suppliers and therefore you agree not to modify the Software or attempt to reverse engineer, decompile, or disassemble the Software, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.

This Software is subject to US Commerce Department export restrictions.

ADDITIONAL LICENSE TERMS FOR SOFTWARE
Cleanscape grants to you as an individual, a personal, nonexclusive license to install and use the Software as described in its documentation. You may use the Software on more than one device so long as you are the only user of Software and Software is only used on one machine at a time. If you are an entity, Cleanscape grants you the right to designate one individual within your organization ("Named User") to have the right to use the Software in the manner provided above.

GENERAL TERMS THAT APPLY TO COMPILED PROGRAMS AND DISTRIBUTABLES
You may NOT distribute Software to the users of programs you create UNLESS you and Cleanscape mutually enter into a separate distribution agreement. Royalties are associated with distribution, and pricing depends on the components and quantities thereof. UNDER NO CIRCUMSTANCES MAY ANY COPIES OF SOFTWARE BE DISTRIBUTED WITHOUT A DISTRIBUTION AGREEMENT. You may not share copies of Software, distribution or not, with other co-workers or colleagues. You may not reproduce or distribute any Cleanscape documentation without Cleanscape's permission.

The Software might include files provided by a third party vendor ("Third Party Software"). Since use of Third Party Software might be subject to license restrictions imposed by the third party vendor, you should refer to the on-line documentation (if any) provided with Third Party Software for any license restrictions imposed by the third party vendor. In any event, any license restrictions imposed by a third party vendor are in addition to, not in lieu of, the terms and conditions of the License Agreement.

All Cleanscape software remains Cleanscape's exclusive property. Contact Cleanscape for the applicable royalties due and other licensing terms for all other uses and/or distribution of the Software.