Contact support@sharradsoftware.co.uk or visit www.sharradsoftware.co.uk

Users Guide

General Replace (GR)
And
General Replace Multi-file (GRX) Utility

Version Covered by this document: 1.79ZD

Last revised: 17th June 2011 v1.0Q
Andrew Sharrad

Index

Contact support@sharradsoftware.co.uk or visit www.sharradsoftware.co.uk

Whats New – Recent Changes to GR

GR 179Y to GR179Z…

- Add the /#:I and /#:J functions to ignore the first or the last matches (see page 41)
- Improved the /#:<number> function (see page 41)
- Add the /D:L and /D:R functions to preserve parts of the surrounding matched line (see page 26)
- Change the default "match if end of line" character from the percent symbol, to the exclamation mark (% to !) (see page 20)
- Various bug fixes mainly relating to the /#:a function
- This is an interim release with GR1.80 due in 2011.
- GRX downloads now have the executable files renamed to include the "X".
- Additions to the /K function

Please see the version history for more information at
http://www.sharradsoftware.co.uk/gr179doc.htm

Page left Intentionally Blank

Contact support@sharradsoftware.co.uk or visit www.sharradsoftware.co.uk

License Agreement

This software is provided without any guarantees as to performance, reliability or compatibility. It is not designed for commercial or mission critical usage.

We are not responsible for any consequential loss or damages from the use or mis-use of the software or documentation. Usage of any item from Sharrad Software is entirely at your own risk. No warranty implied or otherwise. Technical support, where available, is provided on a "best endeavours" basis and response times and any possible resolutions are not guaranteed.

If you do not agree with these terms do not use the software.

These instructions are provided to assist you in getting to know the product. These instructions are also provided without any guarantees as to accuracy, errors or omissions. If you do not agree with the terms for using these instructions please do not use the product.

Conditions of Use

This software is provided without charge, although voluntary donations are welcomed to assist with the cost of hosting the web site.

Sharrad Software permit the re-distribution of GR and/or GRX, but we ask that you include a link to http://www.sharradsoftware.co.uk in your release notes. However, even if a donation has been made, this software may not be modified. Modifying or reverse engineering the software constitutes a breaking of the terms of the license agreement and negates your permission to use the product.

Nagware and Digitally Signed Versions

Some versions of GR/GRX are Nagware.

This means you will get a reminder that the software is not a commercial product. Please contact us to receive a version without the Nagware reminder. We do ask please for a donation to assist with the cost of hosting these materials.

We can provide Digitally Signed versions of our Win32 applications. In Vista and Windows 7 this will allow the programs to run without a warning that the application has not been signed. These versions of Windows use a digital signature to prove the authenticity of the application. However, to go towards the cost of obtaining a certificate we ask you to make a **donation** of £10 or more, and then contact us to receive a signed version. This version will also be nagware free.

To test that our digital certificate meets your requirements please download and run the test program here: http://www.sharradsoftware.co.uk/certtest.zip (For Windows 32-bit compatible environments only).

Lite Versions

GR and GRX are both available in Lite versions. These versions have some features removed to slightly improve performance, and make the application even lighter weight, for example for even better use over a networking.

Lite versions do not have:

- Command line help
- Output redirection (/$[:A])
- Pad byte (/N) and wildcard replacement byte (/W) are always a space (ASCII 32), and cannot be changed
- Case control switch (/&[:a])
- The ability to disabling Command Line improvements (/Z)
- The ability to insert a special functionality character (/K:mC)
- Match control (/#:a)
- Loading <search> and <new> from external files (/L)

In addition to Lite Versions, we can also provide Lite or normal versions that are limited to files less than 64K in size to again reduce the program footprint.

Availability for Different Operating Systems

GR and GRX are available for the following operating systems:

DOS/Win16
Runs on true DOS, Windows 3.1 DOS prompt, Windows 95/98 (pre-OS Command prompt) . If needed, we can also provide a version that runs in character text mode on Windows 3.11 inside the GUI.

Limitations: The maximum file size that can be worked on is limited to the amount of free base memory, which is usually much less than 640K. Loading the search or new text from file is also limited to file sizes less than 64K.

Warning	We do not recommend that you run the 16-bit version of GR or GRX in a 32-bit environment. 16-bit versions of GR do not support long filenames. Files can only be loaded using their short name. Any creation or copying of a file will likely lose the long filename.

Win32 (95/98/NT/2K/XP/Vista/7)
These are the main versions used today. This version supports long file names and runs as a character mode executable. It supports long filenames and works with file sizes up to 2GB. This is the version that you are most likely to need.

OS/2 and EcomStation
This is a 32-bit version with full support for native OS/2 command windows. Again, it supports long filenames and works with file sizes up to 2GB. If you require we can provide a 16-bit version for legacy environments.

Minimal Testing: Recent versions GR and GRX have received minimal testing on DOS, Windows 3.1, 95 and 98 due to the legacy nature of these operating systems. OS/2 and Ecomstation use has been tested.

Windows Vista and the UAC (User Account Control) Feature

Windows Vista includes the User Account Control feature, or UAC. By default, Windows Vista Administrative users actually run as a standard user account. When Administrator functions are needed Windows Vista elevates the user or application to the rights needed. Depending upon the system policy, Vista will first give the user a confirmation prompt (and, in the case of a standard user being logged on, it will prompt for the credentials of a user with an administrative account).

GR as default does not need administrative privileges to run.

However, there are circumstances where you will need GR to have Administrative privileges, possibly because of the file that you want GR to work on for example.

There are three ways that you can do this.

1/ The first is the Vista inbuilt "RUNAS" command. You can use this command from a shortcut, from a batch file or directly from the command prompt. However this normally requires you to specify the administrative account name and then be prompted for a password, which makes it difficult to script in a completely automated fashion.

2/ You can create a shortcut to a batch file which runs the needed GR command, and choose "Run as Administrator". However this requires the end user to remember to do this, and again and prevents GR from running in a completed automated fashion.

3/ Win32 versions of GR are available in a "UAC_Elevate" version. This version of the program contains special code that will ask Windows Vista to automatically elevate GR to run with Administrative privileges in most circumstances. If an administrator is logged on you will get a UAC confirmation prompt depending upon the policy set on the system. However, if a standard user is logged on however, Vista will prompt for the username and password for an administrative account.

If you need GR to run on a Vista machine in a scripted fashion we recommend you test it using the account that's likely to be used when GR is put into action. The UAC_Elevate version of GR is provided as a convenience to assist you.

Usage Introduction

GR and GRX are useful utilities similar in style to GREP but offering alternative functionality. These programs could be used as part of scripts, or batch files. It is useful for batch processing. Its also available in several different variations, including a version that can handle multiple files and sub-directories (GRX) and versions for OS/2 or EComStation.

Basic operation:

Take a source file (from now on, referred to as <file>), and modify it, searching for text <search>, and then changing it using the command line switches provided, using the <new> replacement text as necessary.

From the command line:
```
GR <file> <search> [<new>] [/switches]
```

For example:

```
GR listofnames.txt "bob" "robert"
```

listofnames.txt
```
Tom
Barry
Bob
Bobby
Roger
```

The file `listofnames.txt` is modified – the original is changed and backup copies are not made.

Bob and Bobby in this example will get changed to Robert

Full coverage of the various switches and options are given in this document.

Arguments in <arrows> refer to general text – filenames, words to match on etc. Anything show in enclosing [square] brackets means that its optional.

Switches are differentiated from text by using a Slash or hyphen. They are also case insensitive.

```
GR dictionary.txt "find this" " add this onto it" /a
```

is equal to
```
GR dictionary.txt "find this" " add this onto it" -A
```

Many switches can be used together and combined in GR, so switches can also be used consecutively without a slash or hyphen to save space on the command line:
```
GR dictionary.txt "long search string" "short" /C /M /N:32
```

Can be shortened to:
```
GR dictionary.txt "long search string" "short" /CMN:32
```
Or
```
GR dictionary.txt "long search string" "short" /N:32 /CM
```

More details on using the command line are covered later in "Basic Principles – Using the command Line".

Page left Intentionally Blank

How to use the examples in this document

To try and make things easier to understand, I will try and use some simple samples of what would happen when using various switches, or what happens in different circumstances.

GR <filename> <search> [<new>] [/switches]

```
GR listofnames.txt "Peter" " Robert" /a
```

listofnames.txt

```
Tom
Malcolm
Peter
Roger
```

The file `listofnames.txt` is modified – the original is changed and backup copies are not made.

The green Peter and Bobby in this example shows what GR matched when searching. The blue Robert shows that is has been added by GR. We will use red to show deletions where possible.

The result from above:

listofnames.txt

```
Tom
Malcolm
Peter Robert
Roger
```

Best Practices

- Try the commands and switches first to familiarise yourself before putting them into proper use. Use test files, or test files and folders and check that the result you want is being achieved. Once GR has performed its work, there is no undo option. If you modify a file by mistake you will need to restore it from your backups, or re-create it.

- If you receive an error message, note it down carefully to trouble shoot it. Most are self explanatory; the most command error message is related to mistakes being made on the command line and GR will attempt to show the argument it did not understand.

- Read this document and instructions carefully!

Page left Intentionally Blank

Differences between GR and GRX

GR can only operate on one file at a time. As such, it needs a filename of the file to be modified. GRX however operates on a file specification, such as *.doc. This should process all .doc files in the current folder.

```
GRX *.doc <search> [<new>]

GRX C:\scripts\*.bat <search> [<new>]

GRX C:\scripts\*.bat <search> [<new>] /R
```

The last switch, /R, tells GRX to process subdirectories. This switch (and a few others) are specific to GRX.

GRX	In this document we will try and highlight differences between GR and GRX by putting GRX information separately where possible.

Single Source Code

GR and GRX are available in many different versions and combinations. However, all of these variations are based on a single source file to improve consistency, both in terms of results achieved, the use of switches and standard error messages.

This means that if you need to move operating platform, or need to move from GR to GRX you can expect similar program behaviour. While making any changes we would always recommend you test your script or GR instructions, you should experience no problems.

GR is compiled in Open Watcom 1.8 for the fastest, most reliable program. It will also compile in Borland C 3.01, 5.01, and Microsoft Visual Studio 2003 – so if you have a specific requirement please let us know.

Basic Principles – Using the Command Line

Most users have a good understanding of how to run programs from the command line, however GR has some advanced features to make programming easier, quicker and more flexible.

From a command prompt (or OS/2 command window, hereon in referred to as a command prompt), we recommend you experiment using GR on some test files until you are happy that GR is doing what you need to do.

GRs basic structure for the command line is as follows:

```
C:\GR>GR /?
GR Utility v1.79X+ [TURBO] Copyright (C) Andrew Sharrad 2009

Command line: GR <filename> [-L ]<search>[ [-L ]<new>][ -switches]
```

Arguments in <arrows> refer to general text – filenames, words to match on etc. Anything show in enclosing [square] brackets means that its optional.

Switches are differentiated from text by using a Slash or hyphen. They are also case insensitive. For example -

```
GR drinklist.txt "apple" " Cider" /a
```

is equal to

```
GR drinklist.txt "apple " Cider " -A
```

drinklist.txt

```
Pear Cider
Apple Cider
Other cider
```

The word Cider has been added onto the word "Apple". Note the space before the word cider, included inside the double quotes on the command line, to put a space between the two words.

Many switches can be used together and combined in GR, so switches can also be used consecutively without a slash or hyphen to save space on the command line:

```
GR dictionary.txt "long search string" "short" /C /M /N:32
```

Can be shortened to:

```
GR dictionary.txt "long search string" "short" /CMN:32
```

Or

```
GR dictionary.txt "long search string" "short" /N:32 /CM
```

And Even

```
GR dictionary.txt "long search string" "short" /N:32:CM
```

The first colon ":" after /N shows that it's a sub-argument for the N switch. The next colon ends the sub-argument.

Switches, or arguments that don't have possible (mandatory or optional) sub-arguments can also be separated from the next argument by a colon.

```
GR dictionary.txt "long search string" "short" /N:32:C:M
```

However, the following isn't valid:

```
GR dictionary.txt "long search string" "short" /N:C:M
```

This is because the /N switch can (optionally) take a sub-argument. Sub-arugments are always separated from the main argument by a colon. By putting a colon directly after the /N switch the command line interpreter is looking for the sub-argument for the /N switch – and that is not what was required here.

Switches can appear anywhere in the command line without changing their functionality or performance, with one or two exceptions. For example, the /L switch specifies that the next text argument for <search> or <new> refers to a filename that the search string or new string is to be loaded from. So, the /L switch needs to be positioned in the correct place.

GR is flexible enough to ignore minor errors (for example, repeated simple arguments) but clever enough to reject arguments that it either doesn't recognise, or that conflict.

For example:

```
G:\BCW>gr179y test.txt search /N:R
GR Utility v1.79Y+ [TURBO] Copyright (C) Andrew Sharrad 2009

ERROR: Bad command line information: R
```

In this example, R was shown as being invalid – the /N argument was acceptable, but on the R sub-argument, because /N:x takes an number as a sub-argument.

Some more switches later cover other enhancements to the command line, but a brief summary is shown below:

`GR /L search.txt /L new.txt filetoworkon.doc` - is acceptable. The /L switch instructs GR to load the <search> pattern from the file search.txt.

```
GR filetoworkon.doc "searchtext" "replacement text" /$ outputfile.doc
```

In this case filetoworkon.doc is the source file, but its not modified, since the output has been redirected to outputfile.doc. The filename outputfile.doc must follow the /$ switch immediately afterwards.

Many switches in GR can be used at the same time to achieve the desired result. GR is capable for all the standard switches to be used at the same time and the work is processed in one pass, rather than you having to run GR multiple times with different switches.

GRX	GRX uses almost exactly the same command line arguments as GR. Instead of a filename, it uses a file specification, such as *.DOC, and the /$ switch by default refers to an output directory, instead of a filename.

Page left Intentionally Blank

Basic Principles – The File to be Modified

What can be in the file?

Almost anything at all.

GR was originally designed to be used on simple text files; and this remains the primary usage. It can be used to modify batch files, VB Scripts, INI or configuration files.

However, it can also be used to modify files than also contain machines code, or non-text data. How do? GR loads and modifies all fines in binary format, and does not rely on text formatting to perform the searches or make changes. However, any inbuilt file checksums are not recalculated, so the modification of JPEG or ZIP files directly would probably result in the file being reported as corrupted.

GR is also possibly suitable for modifying Microsoft Active Directory export or import information.

How big can the file be?

Since GR modifies the file by loading it into memory and then committing changes to disk, the file can be as big as physical or paged memory permits, or a maximum of 2GB, whichever comes first. In Win32 applications, Windows automatically limits each process to a maximum of 2GB. Note that loading a file of 2GB into memory is likely to take some time and Windows, assuming you have sufficient physical RAM in the first place, may then also page some of the used memory back to disk.

What if the file I wanted to work on didn't exist?

GR will report that the file was not found. Or, it can be programmed to create a new file, with the /T:C (Tag <new> onto the end of the file if there aren't any matches, creating a new file if the file wasn't found - this is the :C part) or /B:C (Add <new> to the beginning of the file if there aren't any matches, creating a new file if the file wasn't found – again, this is the :C part).

GRX	If you give GRX a single filename to work on, it will behave exactly as GR in the above scenario. If however you give a file specification, such as *.DOC, it cannot create a new file, since it doesn't know the filename that you want to create.

What if I don't want to modify the file, I just want to see if the matches are present?

Use the /#:0 switch and argument. The /# symbol tells GR that you want to restrict the matches which are worked on, and /#:0 tells GR that actually you don't want ANY changes made. The file will be opened in Read Only mode, to ensure that changes cannot be made to it. The number of skipped matches will be displayed, and the DOS errorlevel will be returned again showing the number of skipped matches.

What if the file cant be edited? How can I use GR on a file that's locked or read only?

Either copy the file, or use GR's output redirection function, /$. This means again that the original file will be opened in Read Only mode.

For example:

```
GR lockedfile.txt "User: Tom" "User: Bob" /$ output.txt
```

The /#:0 and /$ <output filename> commands can be used at the same time. The file will just be copied, and information about skipped matches displayed.

GRX	GRX can also use the output re-director, but by default its an output directory. `GRX files*.txt "User: Tom" "User: Bob" /$ c:\outputdir`

Basic Principles – <search> and <new> Text

```
C:\GR>GR /?
GR Utility v1.79X+ [TURBO] Copyright (C) Andrew Sharrad 2009

Command line: GR <filename> [-L ]<search>[ [-L ]<new>][ -switches]
```

The search text always needs to be specified for GR to operate. Without a search pattern, GR cannot operate. Remember, to can change <search> and <new> to become filenames to load these patterns by specifying the /L switch beforehand.

Enclose <search> and <new> in double quotes on the command line to ensure that they are passed to GR correctly.

For example:

`GR file.txt seach for this "found"` - is not valid. Because search for this is not enclosed in double quotes, GR gets passed three different text arguments instead of one.

The correct format is:

```
GR file.txt "search for this" "found"
```

If you use the load from file, /L switch, the entire file is loaded into memory. This means that it can include spaces and line breaks without needing additional double quotes to enclose the arguments.

For example:
```
GR file.txt /L search.txt /L new.txt
```

file.txt
```
I am Searching for the most unusual things
```

search.txt
```
am searching for the most
```

new.txt
```
found some of the more
```

Would result in:
file.txt
```
I found some of the more unusual things
```

GRX	GRX is useful here because the /L files are usually only loaded once into memory, no matter how many files are processed. This can result in improved performance, especially on large files or slower machines. The <new> file may be loaded again its been modified to contain randomised characters at your request, and the original file needs to be re-randomised. See the /K:mC switch.

Whats the limit on the size of the <search> and <new> patterns?

These are limited normally by the amount of space on the command line. On DOS/Win16 versions, you can load files up to 64K in size each for <search> and <new> using the /L switches, depending on the amount of memory available.

On Win32 and OS2 versions, the size of file that can be loaded using the /L switch is up to 2GB in size, again depending on the amount of memory available.

What if I want to put a double quotation mark into <new> when using the command line?

The double quotation marks on a command line will confuse the operating system and pass GR the wrong arguments

```
GR book.txt "find this quotation and add quotes" "find this
"quotation" and add quotes"
```

The way around this is to use SINGLE quotes and then use the /' switch, which turns all single quotes in <new> to double quotes when writing the file.

```
GR book.txt "find this quotation and add quotes" "find this
'quotation' and add quotes" /'
```

What if I want to use the <search> as part of the replacement string?

For example, let suppose that the file boot.txt has the line "`Tom's car was shiny`" but you want to change it to "`Tom's next car was shiny yellow`"

You could of course replace the entire first text with the second, but there is an alternative, which comes in use in some situations.

```
GR book.txt "car was"  "next car & yellow"
```

The `&` symbol is the default special character to instead the found text in <new> at that position. More uses for this, and more detail, are covered later.

Does GR have any limitations as to the maximum number of matches in a file?

GR can work on as many matches as there any bytes in a 2GB file.

Basic Principles – Special Characters

GR has four special characters that perform different actions.

Special Character Number	Default Character	Active In	Action
1	@	<search>	Wildcard
2	&	<new>	This symbol in <new> will be replaced by <search>
3	#	<search>	When present at the start of <search>, GR will only match when <search> is at the start of a line of text
4	!	<search>	When present at the end of <search>, GR will only match when <search> is at the end of a line of text

Special Characters 3 and 4 can be used in conjuction, for example:

```
GR config.ini "#[Section Name]%" "[New Section Name]"
```

In this example, `[Section Name]` will only be matched when it is on a line on its own in the file.

I want to change these special characters to something else, to avoid conflict

Use /1:_ for example to change the special character @ to be an underscore instead.

Another example:

```
GR config.ini "*[Section Name]+" "[New Section Name]" "/3:*"
"/4:+"
```

TIP	Remember that some characters on the command line can be badly interpreted by the operating system before they reach GR. Enclose them in double quotes if you are not sure. Use the Sharrad Software COMTEST utility in Appendix 2 to check how arguments are being handled by the operating system.

How do I turn off these special characters?

Just use /2, on its own, for example, to turn off special character number 2.

TIP	Its important to remember that all of these special characters are operational even if you are loading the <search> and <new> text from files, using the /L switch. If you aren't certain about the contents of these files, and you don't want the special characters to work, turn them off

Basic Principles – Error Messages

Most error messages you may get are fairly self explanatory, but we have listed a few below:

Error:

`Bad filename or file does not exist`
The file couldn't be found, or the file contained invalid characters that aren't permitted

`Not enough memory to load file`
The operating system couldn't give GR enough RAM to load the file into memory

`File too big or 0 bytes`
The main file was either bigger than 2GB (or 64K for small file versions) or was zero in size.

`Bad file - could not read`
There was a disk I/O problem – likely a bad sector or other corrupt media

`Could not create new file`
Either the volume is write protected, or an existing write protected file exists

`Missing filename specified for /L switch`
You have used the /L switch and then not specified a filename afterwards

`Could not load file for /L switch into memory`
GR will then tell you the reason for this – usually the file either didn't exist or not enough memory was available.

`Bad or missing command line information`
GR couldn't understand or was missing important information and couldn't proceed. If GR can work out which part was wrong, it will tell you.

`Write fault encountered`
GR experience a file or disk problem when trying to write the new file to disk

`File is Read Only`
The file to be modified is protected from changes. Either the file or volume is read-only.

`Extended error`
GR encountered an internal fault, or possibly an unusual disk issue. Record the error code given and contact support.

TIP	If you are getting an error message you don't understand, try working on a different file as a test. Or, again using a test file, try removing arguments one at a time to see which one is causing the problem.

GRX	GRX has some additional error messages, which occur for example if GRX has problems reading the directory structure (disk problem), or creating the new directory structure (likely to be write protected volume). You can also get the message "No files to process" if you give a file specification – for example, *.DOC, yet no .DOC files exist.

Page left Intentionally Blank

Error Return Codes

GR can return a DOS Errorlevel to tell a batch file more about what happened.

For example:

```
@echo off
GR stocklist.txt "pens" /#:0
if errorlevel==1 goto found
goto end
:found
echo GR found the word pens in the file at least once
:end
```

The following return codes are used:

Return Codes			
	Return Code	**GR**	**GRX**
Normal Operation	0	Completed OK	Completed OK
	1	File not found	No files to process
	2	Bad command line info	Bad command line info
	3	Not enough memory to initialise	Not enough memory to initialise
	4	File too big	
	5	Cant read file	Cant read directory structure
	6	Cant create file	Cant create file or directory
	7	Cant load <search> or <new>	Cant load <search> or <new>
	8	Extended error / Write fault	
	9	File is Read only	
When performing searches only, reporting on matches (`/#:0` specified. Notes that `/#:R` still reports errors)	0 upwards	Reports the total number of matches. (File is not modified). Zero on zero matches or if there is a separate problem!	Reports the total number of matches that are present in all files. (Files are not modified). Zero on zero matches or if there is a separate problem!

Error codes in blue are generally specific to a single file, and as such are only reported by GR.

Error code 7 is in regards to <search> or <new> being loaded into memory using the /L switch.

Generally error code 6 means that a file could not be created – usually a file will only be requested to be created when using the /$, /T:C or /B:C switches.

GRX	The return on GR and GRX differ because GR is operating on ONE file only. Hence, it can report on problems on that one, specific file. GRX works on multiple files and so does not return codes regarding any specific file. Instead, it reports about GRX specific problems such as issues reading the directory structure, or serious problems that prevent it from carrying on processing, such as being unable to create an output file or directory required by the /$ switch (Error Code 6).

Command Reference - Append, Insert or Replace (/A /I switches)

These are the most basic switches in GR.

Without /A or /I, <search> text is completely replaced by <new>

numbers.txt
```
one, two, four, five, seven
```

Running
```
GR numbers.txt "two," " three,"
```

Would give you the result
```
one,  three, four, five, seven
```

Instead, to append to the matching pattern, use the /A switch:
```
GR numbers.txt "two," " three," /A
```

This would result in:
```
one, two, three, four, five, seven
```

You can use /I to insert before the matching pattern:
```
GR numbers.txt ", seven" ", six" /I
```

To give you the final result of:
```
one, two, three, four, five, six, seven
```

TIP	Remember, if you don't specify Append (/A) or Insert (/I) the default is replace. You cannot use /A and /I at the same time as they are mutually exclusive.

Command Reference – Case Sensitivity (/C)

Normally, searches in GR are case insensitive. That is, the word "Andrew" would match the word "andrew". If you want a case sensitive search, use the /C switch

```
GR names.txt "Andrew" "Roger" /C
```

This would ensure that only matches with the exact same capitalisation in the word "Andrew" are used.

Examples - with Case Sensitivity Checking (/C)…

Andrew	matches	Andrew
Andrew	does not match	Andrew
andrew	does not match	ANDREW

Without case sensitivity checking, all versions of Andrew are a match.

Command Reference – Working in Whole Lines (/D[:P])

You can tell GR to select a whole line of text when a match is found:

zoo.txt
```
We have fifteen Kangaroos.
Sometimes we have many elephants as too, for treatment.
We don't have any tigers.
Too many bees are in the hive.
```

If you use -
```
GR zoo.txt "elephants" "We don't have any large mammals anymore." /D
```

In this instance, with the /D switch, the entire line containing the word elephants is selected, and replaced with the new line.

```
We have fifteen Kangaroos.
We don't have any large mammals anymore.
We don't have any tigers.
Too many bees are in the hive.
```

There are times though when you may wish to delete the entire line:
```
GR zoo.txt "large mammals" /D
```

This will delete the entire line containing the matching pattern of "large mammals"

```
We have fifteen Kangaroos.
We don't have any tigers.
Too many bees are in the hive.
```

Notice that the entire line has been removed, including the trailing "enter", or carriage return and line feed symbols. This means that there is no gap left in the text.

If you wish to preserve this gap, use the /D:P switch instead.

```
GR zoo.txt "tigers" /D:P
```

This will result in:
```
We have fifteen Kangaroos.

Too many bees are in the hive.
```

The gap, or carriage return and line feed, has been preserved.

Alternatively, /D:L preserves just the left hand portion of the line, /D:R preserves the right.

/D Switch Setting	Result
(no setting; default)	Text matched on its own, the rest of the surrounding line is not affected
/D	Entire line is selected when matching text is found
/D:P	Entire line is selected when matching text is found, however any resulting empty lines (line breaks) are preserved
/D:L	The entire line is selected except for the text to the left of the match, or preceding the match – this is preserved
/D:R	The entire line is selected except for the text to the right of the match, or after the match – this is preserved

Command Reference – Getting Help (/H[:x])

GR includes in built, basic help to remind you about the switches and commands.

The /H or /? switches can be used to display these on screen.

Command	Result
/? or /H	Display the first help screen
/?:x or /H:x	Display help screen number x, where x is 1 to 3
/?:A or /H:A	Display all help screens
/?:P or /H:P	Display all help screens, with a key press in between each screen
/?:V ot /H:V	Displays the version string only, useful for recording as part of a batch file which version of GR is in use. For example, to store the result in `grvers.txt`, run `GR /H:V >GRVERS.TXT`

TIP	If you aren't sure what's going wrong with a command you are trying to run, its tempting to simply add /? on to the end of an existing command line. This will display the help page and the file will not be processed. Bear in mind though that if you have specified the /V:0 verbosity switch as part of your command line nothing will be displayed on screen. All work prior to processing the file itself though is carried out, so we recommend that if you need to refer to the inbuilt help screen, that you use it on its own: `GR /?:P` for pages of help.

GRX	The help screens for GRX are slightly different to those for GR, so if you are getting to grips with GRX or GR for the first time, make sure you use the right help screens for the program you want to eventually use.

Command Reference – Overwriting (/O)

Normally, when you replace a portion of the file the non-matched part of the file is not changed or overwritten. However this can be changed with the /O switch and takes effect when the <new> text is larger than the <search> text.

For example:
codes.txt

```
The code for chocolate is 56-673339. Its different to the code for
sweets.

GR codes.txt "56-" "*DELETED*"
```

Would give you the result:

```
The code for chocolate is *DELETED*-673339. Its different to the code
for sweets.
```

Instead, going back to the original file, this is the result with the /O switch:

```
GR codes.txt "56-" "*DELETED*" /O

The code for chocolate is *DELETED*. Its different to the code for
sweets.
```

The -673339 portion has been overwritten.

Command Reference – Writing Unmodified (/U) or Modified (M) lines only

These arguments are mutually exclusive and change drastically what appears in the output file.

With the /U switch, only lines (text separated by carriage return and line feed combinations) that have not had any changes made to them will appear in the output file.

lines.txt
```
One
Two
Three
Four
Five
Six
```

```
GR lines.txt "tw" /U
```

Would give the following result:
```
One
Three
Four
Five
Six
```

"tw" has been matched with "Two" and the entire line has been removed, since only Unmodified lines are left in the file.

Using the /M switch, the opposite happens. Only modified lines are left in the file

```
GR lines.txt "e" "e" /M
```

This now gives the following:
```
One
Three
Five
```

In this command, any line that has the letter "e" in it is preserved, even though the "e" is exchanged for another "e".

TIP	The number of matches reported in the last example is 4, even though only three lines were preserved. This is because "three" has two e's in it.

Command Reference – Pause before exiting (/P)

This command simply instructs GR to obtain a key press after it has done its work.

This is an alternative to the DOS `PAUSE > NUL` command and is provided for convenience.

Command Reference – Verbosity Level (/V[:x])

Verbosity refers to how much information GR displays on screen about the work in progress. You can use this switch to add or remove the amount of information displayed.

Verbosity Level	**GR**	**GRX**
0	Nothing displayed – Silent operation	As GR, left
1	Program message displayed; /P keyppress message displayed	As GR, plus any messages about directory levels that were too deep are ignored, or directories that were detected as being the same as the source directory
2 (default)	As above plus details about matches, skipped matches, and error message displayed	As above and GR, left
3 (selected when you use the /V switch on its own)	As above plus details about original vs. new file sizes displayed	As above and GR, left

Command Reference – Special Characters (/n[:b])

GR has four special characters that perform different actions.

Special Character Number	Default Character	Active In	Action
1	@	<search>	Wildcard
2	&	<new>	This symbol in <new> will be replaced by <search>
3	#	<search>	When present at the start of a line, GR will only match when <search> is at the start of a line of text
4	%	<search>	When present at the start of a line, GR will only match when <search> is at the end of a line of text

Use the following commands:

Command	Result		Command	Result
/1	Disable Special Character No. 1		/1:+	Change to + symbol
/2	Disable Special Character No. 2		/2:$	Change to $ symbol
/3	Disable Special Character No. 3		/3:r	Change to r (case sensitive)
/4	Disable Special Character No. 4		/4:*	Change to * symbol

WARNING	GR deliberately does not check to see if the special characters have been duplicated, in case for some reason you need to use the same symbol. Use wisely if you need to use the same symbol!

Special Characters 3 and 4 can be used in conjunction with each other, for example:

```
GR config.ini "#[Section Name]%" "[New Section Name]"
```

In this example, `[Section Name]` will only be matched when it is on a line on its own in the file.

I want to change these special characters to something else, to avoid conflict

Use /1:_ for example to change the special character @ to be an underscore instead.

Another example:

```
GR config.ini "*[Section Name]+" "[New Section Name]" "/3:*"
"/4:+"
```

TIP	Remember that some characters on the command line can be badly interpreted by the operating system before they reach GR. Enclose them in double quotes if you are not sure. Use the Sharrad Software COMTEST utility to check how arguments are being handled by the operating system.

REMINDER	Its important to remember that all of these special characters are operational even if you are loading the <search> and <new> text from files, using the /L switch. If you aren't certain about the contents of these files, and you don't want the special characters to work, turn them off

Page left Intentionally Blank

Command Reference – Tag to the Beginning (/B[:C]) or End (/T[:C]) of file

GR has an ability to add the <new> pattern onto the Beginning or Tag onto the end of the file being worked on, if the <search> pattern is not found.

stationerylist.txt
```
pens
stapler
ruler
protractor
photo paper
```

```
GR stationerylist.txt "80gm paper" "100gm paper" /T
```

In this example, "80gm paper" isn't found in the file, so "100gm paper" will get added onto the end. Note that it will literally get tagged onto the end of the file, without any additional "enter", that is, a carriage return and line feed.

The result will be:
```
pens
stapler
ruler
protractor
photo paper
100gm paper
```

However there wont be a CR + LF on the end of "`100gm paper`" because GR hasn't been told to add one. So, if you run a similar command again, you will get this result:

```
GR stationerylist.txt "paperclips" "paperclips" /T
```

The result will then be:
```
pens
stapler
ruler
protractor
photo paper
100gm paperpaperclips
```

There are two ways around this:

- Use the /D switch to make GR work in whole lines. The new line, when added, will always have a CR+LF added on the end
- Or, Use /K:6x to add a CR+LF onto the end of the string

For example:
```
GR stationerylist.txt "paperclips" "paperclips?" /T /K:6?
```

In this example, the `/K:6?` means that the "`?`" in paperclips is replaced with a CR+LF. The /K switch is covered in more detail later.

TIP	The example above can be shortened to: `GR stationerylist.txt "paperclips" "&?" /T /K:6` On this line, by default the "`&`"acts as a special character and the original <search> pattern is used in its place.

The /T and /B switches have an optional sub-argument, which is :C

```
GR doesntexist.txt "wheels" "chairs" /B:C
```

If the file "`doesntexist.txt`" doesn't exist, with the /B:C option GR will actually create a new file. The new file will contain the word "`chairs`".

GRX	GRX does include /T:C and /B:C functionality. If you use GRX with a file specification, such as "*.doc", GRX can only be working on files already found on the disk. A new file will never be created. However, specifying a single, non-ambiguous file to GRX that doesn't exist, using the /T:C or /B:C switch will result in a new file being created.

Command Reference – Force Next Argument is a Text Argument (/G)

In DOS, Windows or OS2 legacy command line applications, switches – arguments that change program behaviour – are preceded by a hyphen or slash to differentiate them from normal text argument.

```
GR /I file.txt "search for this" "new" "-N:65"
```

In the line above, `/I` and `-N:65` are both switches. Putting the `-N:65` in double quotes does not stop it working as a switch.

But what if you want to use GR and the <search> or <new> arguments, or even the filename, start with a slash or hyphen?

```
GR ---old.txt "/old url" "/new url"
```

In this example, GR (and most applications using the command line) would believe that "`---old.txt`", "`/old url`" and "`/new url`" are all switches.

GR has a work-around for this.

Simply put a /G switch before the text, and GR will treat it as text and ignore any hyphen or slash that would usually tell it that it's a switch. This only works for the next separate argument, one time only.

For example:

```
GR /G ---old.txt /G "/old url" /G "/new url"
```

The /G switch will still be affective even if other switches appear between it and the text arugment, for example:

```
GR /GIDX ---old.txt /G "/old url" /G "/new url"
```

In this example, the I, D and X switches still work as normal.

TIP	The /G switch can still be combined successfully with the /L switch, to force the next argument to be text, and a filename for <search> or <new>, for example: `GR file.txt /L /G "---search.txt" "/G /L "---new.txt"`

Command Reference – Next Argument is an Environment Variable Name (/E)

In batch processing it is often useful or necessary to use environment variables as arguments.

Environment variables:

```
path=c:\;c:\dos;c:\windows
file=c:\old file.txt
new=hello galaxy
```

For example:

```
GR %file% "hello world" %new%
```

The argument `%file%` and `%new%` are taken from environment variables.

However, GR has its own functionality for this which can be useful if the environment variable doesn't exist.

Firstly, environment variables can be loaded directly using the following:

```
GR /E file "hello world" /E new
```

The /E switch tells GR that the next text argument is an environment variable name. If the environment variable doesn't exist then GR will be missing the required information.

NOTE	The /E or /EG switches cannot be used to load other switch information from an environment variables. `GR filename.txt search /E switchlist` would result in the contents of the environment variable `switchlist` being used as the next text argument, in this case <new>

However, where /E comes in more useful, is in conjunction with the /G switch and when the environment variable is missing or blank.

Environment variables:

```
path=c:\;c:\dos;c:\windows
file=
new=hello galaxy

GR /GE file "hello world" /E /G
```

In this example, with /G and /E appearing before the text argument, since file is blank as an environment variable, the filename that GR will use will actually be the name of the environment variable itself, this time, "file".

Perhaps a better way of looking at this is the other way around. It may be useful to have GR use a set filename in a batch file, but only have the name redirected in certain circumstances:

```
GR /GE normalfilename.txt "hello world" %new%
```

In this example, if on unusual circumstances you needed to make "`normalfilename.txt`" redirect to another file, simply create an environment variable called `normalfilename.txt` and assign it to the file that you want GR to use instead.

For example:

```
set normalfilename.txt="emergency redirected filename.txt"
GR /GE normalfilename.txt "hello world" %new%
```

Would translate to:

```
GR "emergency redirected filename.txt" "hello world" "hello galaxy"
```

Command Reference - Insert a Special Functionality Character (/K:mC)

GR has a second set of special characters that work exclusively on <new>. These are known as special functionality characters.

Unlike the normal special characters (which can be turned off using /1, /2 etc., or reprogrammed using /1:x where x is a character to be used), special functionality characters are not active by default.

Use the /K switch to enable special functionality characters. Use each of the /K switch requires two sub arguments, the m for mode (or function), and the character C which is to be replaced in <new>.

For example:

```
GR file.txt "end of the line." "&d" /K:6d
```

In this example, mode (or function) `6` is used and the character `d` is specified. The `d` in <new> will be replaced according to function 6.

/K replaces character[s] in <new>	
Mode (or function)	What to insert
1	Random numeric
2	Random alpha - lower case
3	Random alpha - upper case
4	Random alphanumeric - lower case
5	Random alphanumeric - upper case
6	A carriage return (ASCII Characters 13 and 10)
7	A tab character
8	An EOF (End-of-File) marker (ASCII Character 26)
9	A null (ASCII Character 0)
0	A double quotes (ASCII Character 34)
C	Percent symbol (useful for batch files, ASCII Character 37)
G	Greater than symbol (>)
H	Random hex - lower case
I	Random hex - upper case
L	Less than symbol (<)
Commands	
RM	/K:RM – Reset randomisation after every match
RF	/K:RM – Reset randomisation after every file (GRX only)
Note: You can separate lots of /K requests using a comma. For example, /K:1e /k:2f can be shortened to /K:1e,2f	

File.txt

```
A new carriage return will appear at the end of the line.
There will be a space between these two lines.

GR file.txt "end of the line." "&d" /K:6d
```

File.txt will now change to:

```
A new carriage return will appear at the end of the line.

There will be a space between these two lines.
```

TIP	The /K:mC functionality is case sensitive within <new> `GR file.txt "end of the line." "&d" /K:6D` This would not work as "d" does not match "D".

The /K special function character is often useful if you want to generate a random string. For example:

GUIDlist.txt
```
guid,
guid,
```

```
GR GUIDlist.txt "guid," /A " UUUUUUUU-LLLL-LLLL-UUUU-UUUUUUUUUUUU"
/K:IU /K:HL
```

This will generate a new GUID string after the "guid," and will look something like this:

GUIDlist.txt
```
guid, 62EA743B-9a3e-49ab-5F0B-65A48CC582D0
guid, 62EA743B-9a3e-49ab-5F0B-65A48CC582D0
```

Note that the two GUID strings are the same. GR has used the same random sequence for each match since <new> is only evaluated and built once.

However, its possible to force GR to rebuild <new> on every match with the /K:RM switch. If GR detects that you are using any form of randomisation by using /K:mC, GR will rebuild <new> on every match.

Going back to our example above, this is what would happen with the additional switch:

```
GR GUIDlist.txt "guid," /A " UUUUUUUU-LLLL-LLLL-UUUU-UUUUUUUUUUUU"
/K:IU,HL,RM
```

GUIDlist.txt
```
guid, 62EA743B-9a3e-49ab-5F0B-65A48CC582D0
guid, 0F65BE83-a031-98cb-BA58-28CAF68BA328
```

TIP	The number of different /K:mC usages in GR is limited to 10. If you use more than 10 you will get `Error: Bad Command Line Information` displayed. Using the commands like /K:RM is not included in the count of 10.

Note	Using /K with any randomisation feature, and using /L to load a file into memory for <new>, causes the file to be reloaded every time new randomisation is required. This can cause slower program execution if the file being loaded by /L for <new> is large, or if many /K operations are needed. GR will fail with an error message if the /L filesize is changed between randomisations or cannot be re-read.

GRX	GRX also includes the /K:RF switch. Unlike /K:RM which resets randomisation (if needed) after every match, /K:RF only resets randomisation when GRX moves onto the next file. In GRX, /K:RM implies /K:RF since the randomisation is always changing. `GR GUID*.txt "guid," /A " UUUUUUUU-LLLL-LLLL-UUUU-UUUUUUUUUUUU" /K:IU /K:HL /K:RF` In this example, all of the matches within a file would be written with the same GUID string, but each file would be different.

Page left Intentionally Blank

Command Reference – Specify which matches to work on (/#:a)

This command is useful to tell GR which matches to operate on, or to tell GR to simply report the number of matches back as an ERRORLEVEL code.

novel.txt
```
John said "I'd like no sugar in my tea please!"
"Yes" said Jill, "no sugar at all for John today please"
"He should have said that earlier" retorted the cook "I've already
put in two lumps!"

GR novel.txt "said" /#:0
```

In this example, GR will simply report back to the operating system how many matches it found. The file will not be modified.

It could be used in a batch file in this way:
```
@echo off
echo Test batch file for GR
GR novel.txt "said" /#:0 /V:0
rem using /V:0 to make GR silent
rem and using /#:0 to report matches only
if errorlevel==4 echo The word said was used four times
if errorlevel==3 echo The word said was used three times
if errorlevel==2 echo The word said was used twice
if errorlevel==1 echo The word said was used once
if errorlevel==0 echo The word said wasn't used at all
echo.
```

The answer would be:
```
Test batch file for GR
The word said was used three times
```

Summary of /#:a switch functionality		
Switch <a>	Meaning	Affect on file
/#:O	Act on Odd matches – 1st, 3rd, 5th etc.	File is modified as specified
/#:E	Act on Even matches – 2nd, 4th, 6th etc.	
/#:I	Ignore the first match only – act on the 2nd, 3rd, 4th etc.	
/#:J	Ignore the last match only – act on the 1st, 2nd etc. but not the last match	
/#:L	Act on only the last match	

/#:<number> [,number2]	Act on the match specified – 1 for first match only, 2 for the 2nd match only etc. Specify a number from 1 to 999999999 without any leading zeros. Up to ten numbers can be specified, either by using commas, or or by separate commands:
	`GR novel.txt "said" /#:12,23,44`
	`GR novel.txt "said" /#:12 /#:23,24`
	These two commands are identical.	

/#:0 (zero)	Don't act on any matches. Just report the number of matches back to the operating system. Reports zero on error, such as file missing.	File is not modified. File can be Read Only and GR will still operate.
/#:R	Don't act on any matches. Just report the number of matches back to the operating system, or any error codes if any occur (including file empty, missing etc.)	

Note	GR can only use the /#:a switch once with the exception of /#:<number>, which can be used up to 10 times. However all of these switches are exclusive – only one mode can be used. If you use more than one type of /#:a switch more than once on the command line the last requested switch only will be used.

Page left Intentionally Blank

Command Reference – Pad bytes (/N[:x])

Use this switch when <new> is smaller than <search> to pad the space to the right. The /N switch takes an optional argument which is the ASCII code of the padding byte you want to use. If you don't specify this optional ASCII code then the default of 32, a space character, will be used.

paddingexample.txt
```
this is an enormous word
```

```
GR paddingexample.txt "an enormous" "a tiny" /N:33
```

The result would be:
```
this is a tiny!!!! word
```

The ASCII code for the exclamation mark is 33.

The /N switch is useful if the overall file size must stay the same.

The /N:[x] switch accepts an ASCII code in the range 0 to 255. See the Appendix for a list of useful ASCII codes.

Note	The /N[:x] switch is designed for use when the <search> pattern is being replaced with <new>. It will product interesting results, but will work if you use /I (insert) or /A (append) – the new string will be written and still padded to match the size of <search>.

TIP	Don't use extra leading zeros when giving GR an ASCII code –
	`GR paddingexample.txt "an enormous" "a tiny" /N:0`
	`GR paddingexample.txt "an enormous" "a tiny" /N:12`
	`GR paddingexample.txt "an enormous" "a tiny" /N:188`
	…these are all OK while
	`GR paddingexample.txt "an enormous" "a tiny" /N:012`
	…is not – GR will read this as /N:0 and then treat 12 as separate arguments.

If joining together switches on the command line, GR will try and intelligently work out what you are trying to achieve. However, to prevent making mistakes we recommend you separate /N:x if you use a code of two bytes or less, and the next argument if it's a number.

For example,
```
GR paddingexample.txt "an enormous" "a tiny" /N:251
```

GR would understand this to mean that you wanted to use an ASCII code of 251, which is a valid code. But if the intention was to use /N:25 and then the special character command /1, its better to do the following:

```
GR paddingexample.txt "an enormous" "a tiny" /N:25:1
```
GR
```
GR paddingexample.txt "an enormous" "a tiny" /N:25 /1
```

The colon after 25 separates the 25 from a switch to disable the first special character, /1.

In other situations GR will be able to determine that the argument is separate if the resulting ASCII code is greater than 255:
```
GR paddingexample.txt "an enormous" "a tiny" /N:331
```

Command Reference – Replacement for wildcards in <new> text (/W[:x])

In the example below, wildcards have been used in <search> to make a match.

datelist.txt
```
Sharrad, Andrew, 01/01/1999
Sharrad, Stephen, 02/02/2000
```

```
GR datelist.txt "@@/@@/@@@@%"  "new column before date, &"
```

The % special character ensures that GR matches the date at the end of the line only. The & special character means that the matching text from <search> is inserted at that point. And lastly, the @ symbol in <new> acts as a wildcard.

This will give you the result:
```
Sharrad, Andrew, new column before date, 01/01/1999
Sharrad, Stephen, new column before date, 02/02/2000
```

However, there may be situations when the match isn't found and you are using the /T[:C] or /B[:C] commands to put the <new> string into the file. However, since <new> contains the & special character, the search string with wildcards is the only source.

```
GR datelist.txt "@@/04/2002"  "& date logged" /T
```

In this example, as the date isn't in datelist.txt, GR will be trying to Tag the string onto the end of the file. However it has no source for the & special character in <new>, except the <search> pattern on the command line. This <search> pattern contains wildcards. It may be useful to change the wildcards to another character and the /W[:x] switch allows you to do this.

So, the command would be:

```
GR datelist.txt "@@/04/2002"  "&, date logged" /T /W:45
```

The result would be:
```
Sharrad, Andrew, new column before date, 01/01/1999
Sharrad, Stephen, new column before date, 02/02/2000
--/04/2004, date logged
```

The ASCII code for the hyphen is 45.

The /W switch takes an optional argument which is the ASCII code of the padding byte you want to use. If you don't specify this optional ASCII code then the default of 32, a space character, will be used.

TIP	The /W[:x] switch is an extreme work-around for this situation. Its generally recommended not to use the combination of wildcards (default @), the insert new special character (default &) and the /T[:C] or /B[:C] switches together.

Please see the notes regarding the /N switch and using ASCII codes on the command line with GR.

Command Reference – Change single quote marks (') to double quotes (") in <new> [/']

This switch is similar to /K:0' – it allows you to put double quotes (ASCII character 34) into <new>, which is not possible to do on the command line.

```
GR list.txt "I want to find this text and enclose it in double
quotes" "'&'" /'
```

The & special character means that the matching text from <search> used at that point.

The double quotes in red get swallowed up by the operating system as it passes arguments to GR. The operating system uses double quotation marks to enclose or split up arguments.

Using single quote – ` – is permitted on the command line. The /' switch tells GR to replace all single quotes in <new> with double quotes.

The result of the above line would be:

```
"I want to find this text and enclose it in double quotes"
```

TIP	Use this switch – /' – as a quicker to use alternative to /K:0'. The /K switch however can turn other selected matching bytes into double quotes, instead of just single quotes.

Command Reference – Changing the <new> special character to be on a line on its own (/X)

The /X switch works in conjunction with the /D switch and the 'Insert <search> at this point in <new>' special character (which by default is "&")

In the example below, we want insert the missing line.

novel.txt

```
This is the first line of a story.
This is the third line of the story.
And this is the last line of the story.
```

```
GR novel.txt "third line" "This is the missing line of the story. &"
/D
```

We are using /D to select the whole line that "third line" appears on.

Without the /X switch, you will get the following:

```
This is the first line of a story.
This is the missing line of the story. This is the third line of the
story.
And this is the last line of the story.
```

Instead, with the /X switch, going back to the original file, this would be the result:

```
This is the first line of a story.
This is the missing line of the story.
This is the third line of the story.
And this is the last line of the story.
```

Essentially, and extra carriage return has been added between "`This is the missing line of the story. `" and "`This is the third line of the story.`"

Command Reference – Disabling Command Line improvements (/Z)

We have found that some operating systems can be confused when multiple arguments are passed to programs. Sometimes arguments can be corrupted, especially if long text arguments, containing a space and ending in a backslash (\), are then enclosed in double quotes, and then followed by more switches.

GR includes a fix for this which attempts to recover arguments when the operating system mis-handles the above argument.

However, if you encounter problems this command line improvement can be disabled with the /Z switch.

We do recommend generally, because of operating system restrictions, that you enclose arguments in double quotation marks. For example:

```
GR file.txt "search" /I &
```

This example should insert the <search> pattern before <search>, duplicating it. However some operating systems do not like the ampersand (&) symbol appearing on its own on the command line. It is better to enclose it in double quotation marks:

```
GR file.txt "search" /I "&"
```

TIP	If you are not sure how your operating system, use the Sharrad Software COMTEST program to test how argument are being passed. COMTEST does not include the above work-around and will show all arguments as passed.

Command Reference – Output redirector (/$[[:A] <filename>])

GR includes functionality to redirect the changes made to a file. Normally, GR will modify the file that you specify it to search for matches. However, you can specify that the file should be copied and modified at the same time.

```
GR inputfilename.txt "search" "new" /$ outputfilename.txt
```

In this example, inputfilename.txt is read for matches but not modified. outputfilename.txt is created and used to store the revised file. If outputfilename.txt already exists it will be deleted.

Its possible though to tell GR to append to the output file. If the output file does not exist, it will still be created. If it does exist, the additional data from the modified inputfilename.txt will be added:

```
GR inputfilename.txt "search" "new" /$:A addtothisfile.txt
```

The /$:A tells GR to append to the existing file.

TIP	Use the /$ redirector to output to the screen, if required. Use `CON` as the filename. Use in conjunction with /V:0 to suppress GR program information being sent to the screen so that only the output file is printed.

GRX

The /$ switch is also available in GRX, with slightly different functionality.

Without any other sub arguments, /$ causes GR to output to the current working directory. Or, a specific output directory can be specified.

```
GR *.doc "search" "new" /$ [outputdirectory]
```

The /$ switch in GRX can take one of optional sub-arguments.

Switch	Meaning	Text Argument [Optional]
/$	Output files to a different directory	[<output directory>] if not given, the current working directory will be used.
/$:A	Append to ONE output file	<output filename>
/$:B	Append to files in a different output directory	[<output directory>] if not given, the current working directory will be used.
/$:F	Send output to a single file only	<output filename> or device, such as CON

GRX uses a temporary file, placed in the target output directory, when /$ or /$:B is used. This will be in the form of 8 random characters, with a .GRX extension. This temporary file will be deleted when GRX has completed.

This temporary file allows GR to check if the source and target directories are the same, which is not permissible. GRX does not simply check the directory or volume names, since with the availability of SUBSTituted, mapped network drives or junction points the path may not look the same, while it could point to the same physical location. GRX will inform you if a directory – (and all its sub-directories) – have been skipped.

Page left Intentionally Blank

Command Reference – Disabling intelligent file writing (/~)

GR uses internal checks to minimise the amount of DISK I/O carried out, which can be of benefit especially on slow drives or network volumes.

This intelligent file writing only writes portions of files that have actually changed. For example, if the file size hasn't changed after the matches so far – in other words, <search> and <new> have evaluated down to the same size – the rest of the file may not need to be re-written.

/~ allows you to disable this feature for trouble-shooting reasons. Expect slower performance as a result.

TIP	Any use of the output redirector (/$) disables intelligent file writing anyway.

Command Reference – Loading <search> and <new> from external files (/L)

Switches can usually appear anywhere in the command line without changing their functionality or performance, with one or two exceptions.

One large exception is the /L switch. This switch specifies that the next text argument for <search> or <new> refers to a filename that the search string or new string is to be loaded from. So, the /L switch needs to be positioned in the correct place.

```
GR inputfilename.txt /L searchfilename.txt /L newfilename.txt
```

In this example, `inputfilename.txt` is searched for the contents of `searchfilename.txt`, and if matched, its replace with the contents of newfilename.txt

The /L switch does not have to be used on both <search> and <new>. For example:

```
GR filetobemodified.txt "find this token" /L replacewiththisfile.txt
```

The /L switch needs to precede the filename directly. The filename needs to be in the place where the corresponding <search> or <new> would be normally.

Note	There are some things to bear in mind with the /L switch. First of all, the size of the files to be loaded with /L need to fit into available memory after the main file to be worked on has been loaded. The maximum theoretical size of the files loaded in this manner is 2GB, however in practice it will work out less than this, depending on how much memory your operating system can supply. Usually most 32-bit operating systems can supply a total of 2GB memory per process, for all files used. In DOS/Win16 (or OS/2 16-bit) versions of GR, the maximum size of the file than can be loaded by the /L switch is 64K per file. (However, the maximum size of the main input file on these operating systems can be larger than 64K, depending on available memory).

Command Reference – Case control switch (/&[:a])

The /& switch is used to control the casing of the text used by the 'Insert <search> at this point in <new>' special character (which by default is the "&" symbol).

By default, without the /& switch, when you use the above special character, the casing is taken from the source file.

For example:

caseexample.txt
```
This should be in LOWER case
And this is a mixture of Cases
THIS SHOULD BE IN Upper CASE

GR caseexample.txt "upper"  "THE &"
```

The word "upper" is matched and the word THE inserted before the original match, "upper":

caseexample.txt
```
This should be in LOWER case
And this is a mixture of Cases
THIS SHOULD BE IN THE Upper CASE
```

However, we can control the case of the text inserted at &, by using the /& switch.

For example, using:
`GR caseexample.txt "uppER"  "&"` … (see below)

Switch	Action	Example Result
(no /& switch specified)	Case taken from the source file	THIS SHOULD BE IN THE Upper CASE
/& (no sub-argument)	Case taken from the command line (the upper appears in lower case)	THIS SHOULD BE IN THE uppER CASE
/&:C	As Above	As Above
/&:U	Case is changed to UPPER case	THIS SHOULD BE IN THE UPPER CASE
/&:L	Case is changed to LOWER case	THIS SHOULD BE IN THE upper CASE

Another example, from above:
GR caseexample.txt "Lower" "&" /&:L

This would result in:
```
This should be in lower case
And this is a mixture of Cases
…
```

Command Reference – GRX Specific Commands - Process subdirectories [/R]

GRX	GRX takes a file specification, such as *.DOC, for files*.txt, to work on multiple files at the same time.
	
	It can also process that file specification in subdirectories, if the /R switch is given:
	
	`GRX *.txt /R <search> <new>`
	This will do all txt files in the current working directory and below
	
	`GRX C:\Projects\*.CFG <search> <new> /R`
	This will make GRX work on all CFG files in the Projects directory and below, on the C drive.
	
	Note that GRX is limited to a fixed number of sub-directories, since it operates with a fixed amount of memory for processing sub-directories (this method is more efficient and faster than allocating directory memory for each directory encountered). GRX will warn if a sub-directory (and its sub-directories) have been skipped.
	
	We can re-compile GRX to support more than the default of 18 directories deep if required. The first – current or otherwise – counts as one of the 18. Please contact us if you need a re-compiled version.

Command Reference – GRX Specific Commands - Stop GRX on the first error [/S]

GRX	When GRX is processing its files it will normally carry on with all of the files to be processed even if there is a problem reading or writing one of the files to be worked on. However it is possible to ask GRX to stop processing as soon as it encounters an error on a file, by using the /S switch.

Command Reference – GRX Specific Commands – Reset /K randomisation after every file (/K:RF)

GRX	The /K:mC Special Functionality Characters option has the ability to generate random characters in <new> at positions determined by the C byte matching bytes in <new>
	
	For example:
	`GR *.txt "find this" "replace with this random number ++++" /K:1+`
	
	Any matches in a text file (.txt) would contain this:
	`replace with this random number 5727`
	
	The same random number (5727) will be used for all matches by default.
	
	GR includes the functionality to re-generate random bytes after every match using the /K:RM option.
	
	However, there may be circumstances where you want all instances or matches within a file to contain the same match, but regenerate the random bytes after every file. To do this, GRX includes the /K:RF switch which cases GRX to re-do the randomisation when GRX moves onto the next file.
	
	`GR *.txt "find this" "replace with this random number ++++" /K:1+ /K:RF`
	
	In GRX, /K:RM implies /K:RF since the randomisation is always changing. It is not possible to re-do the randomisation after every match but then re-use the same randomisations in the next file.

Reminder	Remember, any use of /K:RF or /K:RM, when <new> is being loaded from a file using the /L switch, will cause <new> to have to be re-read from disk each time new randomisation is required. This will significantly slow program execution.

Command Reference – GRX Specific Commands – Changes and Additions to the Output re-director (/$[:A])

There are some differences in the functionality of the /$ switch between GR and GRX.

First of all, in GR:

Switch	Meaning	Text Argument Required
/$	Output to a different filename	<output filename>
/$:A	Output to a different filename, appending if possible	<output filename>

GRX

GRX works differently because the basis of GRX is that it is working on a file-specification – a directory or directories containing files.

Therefore the default action of the /$ switch is to output to a different directory, rather than just a different file.

The /$ switch in GRX can take one of three optional sub-arguments.

Switch	Meaning	Text Argument [Optional]
/$	Output files to a different directory	[<output directory>] if not given, the current working directory will be used.
/$:A	Append to ONE output file	<output filename>
/$:B	Append to files in a different output directory	[<output directory>] if not given, the current working directory will be used.
/$:F	Send output to a single file only	<output filename> or device, such as CON

For example:

```
GRX c:\inputfolder\*.csv "hello" "goodbye" /$
c:\outputfolder /R
```

The /R switch tells GRX to process subdirectories.

As a reminder, GRX uses a temporary .GRX file, placed in the target output directory to identify it, when /$ or /$:B is used. This temporary file will be deleted when GRX has completed.

If GRX determines that the source and destination directories overlap, or are the same, GRX will inform you when that directory and its subdirectories have been skipped.

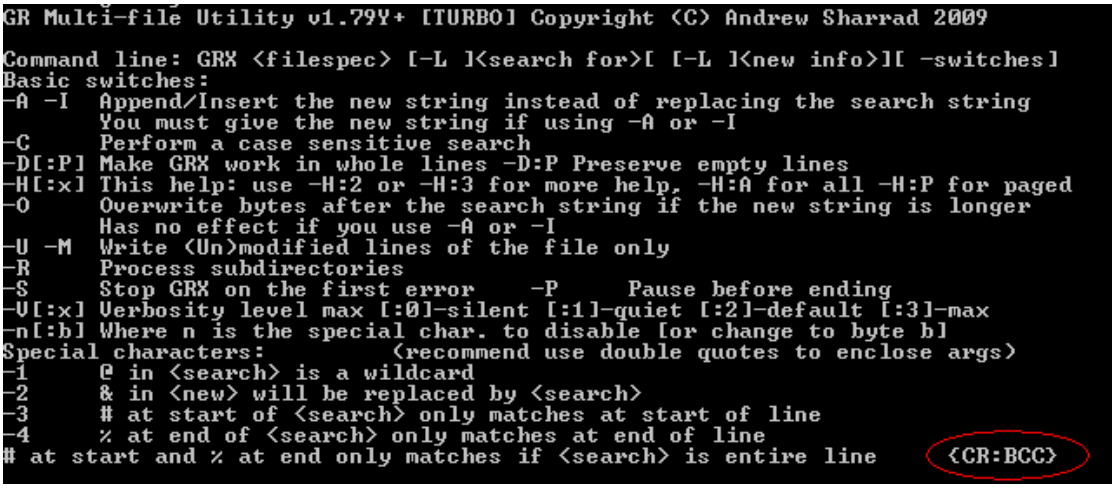
Page left Intentionally Blank

Compiler Information

GR was originally developed in Borland C++ 3.01, however it has evolved across platforms and compilers.

If you have a specific requirement we may be able to re-compile GR using a different compiler if you have a certain need. However we normally distribute GR compiled with what we think is the best compiler for the target platform.

To see which compiler your version of GR was made with, use the /? command and note the {CR:xxx} tag on the bottom right.



Tag	Compiler	Platforms used on
BCC	Borland C++	DOS/Win16 (BCC 3.01) *Also Win32 (BCC5.01)*
VST	Visual Studio 2003/2005	*Beta for Win64 (8GB file support)*
OWC	Open Watcom 1.8	Win32 OS/2 (16 and 32bit)

New Features Coming Soon!

New features coming soon in version 1.80 include proximity matching – the ability to specify the start and end portion of a file in which to conduct searches for <search>. It also will have features to exclude matches based on matching <not> text in the same line, more flexibility in pattern matching and many more features.

Stay tuned for more details!

Appendix 1 – Keyboard / ASCII Code List

Negative numbers represent the special double codes that some keys return. For example, -30, for ALT+A, represents the code of 0 and then 30 that is actually returned.

This table is provided for reference only. A standard UK keyboard was used.

Standard Alphabet						
Key	**Standard**	**Alt**	**Alt GR**	**Ctrl**	**Shift**	**Windows Key**
	(CAPS ON)				**(CAPS OFF)**	
A	65	-30	-30	1	97	As std / Shift
B	66	-48	-48	2	98	As std / Shift
C	67	-46	-46	3	99	As std / Shift
D	68	-32	-32	4	100	As std / Shift
E	69	-18	-18	5	101	As std / Shift
F	70	-33	-33	6	102	As std / Shift
G	71	-34	-34	7	103	As std / Shift
H	72	-35	-35	8	104	As std / Shift
I	73	-23	-23	9	105	As std / Shift
J	74	-36	-36	10	106	As std / Shift
K	75	-37	-37	11	107	As std / Shift
L	76	-38	-38	12	108	As std / Shift
M	77	-50	-50	13	109	As std / Shift
N	78	-49	-49	14	110	As std / Shift
O	79	-24	-24	15	111	As std / Shift
P	80	-25	-25	16	112	As std / Shift
Q	81	-16	-16	17	113	As std / Shift
R	82	-19	-19	18	114	As std / Shift
S	83	-31	-31	19	115	As std / Shift
T	84	-20	-20	20	116	As std / Shift
U	85	-22	-22	21	117	As std / Shift
V	86	-47	-47	22	118	As std / Shift
W	87	-17	-17	23	119	As std / Shift
X	88	-45	-45	24	120	As std / Shift
Y	89	-21	-21	25	121	As std / Shift
Z	90	-44	-44	26	122	As std / Shift
F1	-59	-104	-104	-94	-84	-59
F2	-60	-105	-105	-95	-85	-60
F3	-61	-106	-106	-96	-86	-61
F4	-62	-107	-107	-97	-87	-62
F5	-63	-108	-108	-98	-88	-63
F6	-64	-109	-109	-99	-89	-64
F7	-65	-110	-110	-100	-90	-65
F8	-66	-111	-111	-101	-91	-66
F9	-67	-112	-112	-102	-92	-67
F10	-68	-113	-113	-103	-93	-68
F11	-113	-139	-139	-137	-135	-113
F12	-134	-140	-140	-138	-136	-134
Separate Cursors & Insert/Delete etc. (NOT right hand keypad)						
Key	**Standard**	**Alt**	**Alt GR**	**Ctrl**	**Shift**	**Windows Key**
L.Cur	-75	-155	-155	-115	-75	-75
R.Cur	-77	-157	-157	-116	-77	-77
U.Cur	-72	-152	-152	-141	-72	-72
D.Cur	-80	-160	-160	-145	-80	-80
Home	-71	-151	-151	-119	-71	-71
End	-79	-159	-159	-117	-79	-79
PG UP	-73	-153	-153	-132	-73	-73
PG DN	-81	-161	-161	-118	-81	-81
INS	-82	-162	-162	-146	-82	-82
DEL	-83	-163	-163	-147	-83	-83
Right hand Keypad						
Key	**Standard**	**Alt**	**Alt GR**	**Ctrl**	**Shift**	**Windows Key**
	(NUM Lk Off)					
L.Cur	-75	4	4	-115	52	As std / Shift

R.Cur	-77	6	6	-116	54	As std / Shift
U.Cur	-72	8	8	-141	56	As std / Shift
D.Cur	-80	2	2	-145	50	As std / Shift
Home	-71	7	7	-119	55	As std / Shift
End	-79	1	1	-117	49	As std / Shift
5	-76	5	5	-143	53	As std / Shift
PG UP	-73	9	9	-132	57	As std / Shift
PG DN	-81	3	3	-118	51	As std / Shift
INS	-82				48	As std / Shift
DEL	-83				46	As std / Shift
/	47	-164	-164	-149	47	As std / Shift
*	42	55	55	-150	-150	As std / Shift
-	45	-74	-74	-142	45	As std / Shift
+	43	-78	-78	-144	43	As std / Shift
Enter	13 (13,10)	-166? (10)	-166? (10)	13	13	As std / Shift
Command Keys						
Key	**Standard**	**Alt**	**Alt GR**	**Ctrl**	**Shift**	**Windows Key**
ESC	27	-1	-1		27	27
TAB	9	-165	-165		-15	9
Space	32	32	32		32	32
PrintSCRN				16		
BackSpace	8	-127	-127		8	8
Enter	13 (13,10)	10	10		13	13 (13,10)
Other Keys						
Key	**Standard**	**Alt**	**Alt GR**	**Ctrl**	**Shift**	**Windows Key**
[	91	26?	26?	27	123 {	91
]	93	27	27	29	125 }	93
;	59	-39	-39	255	58 :	59
`	39	-40	-40	255	64 @	39
#	35	-43	-43		126 ~	35
,	44	-51	-51	255	60 <	44
.	46	-52	-52	255	62 >	46
/ (nxt to RH Sft.)	47	-53	-53	255	63 ?	47
\ (next to Z)	92			26	221 |	92
`	96	-41	-41	28	170 ¬	96
1	49	-120			33 !	49
2	50	-121		26?	34 "	50
3	51	-122			156 £	51
4	52	-123			36 $	52
5	53	-124			37 %	53
6	54	-125		30	94 ^	54
7	55	-126			38 &	55
8	56	-127			42 *	56
9	57	-128			40 (	57
0	58	-129			41)	58
-	45	-130		31	95 _	45
=	61	-131		255	43 +	61

Appendix 2 – COMTEST

The following command line arguments – and more – can be badly processed by operating systems.

```
GR filename.txt "search" & /M
```
Problem: The & character is used by the operating system
Resolution: Enclose the & symbol in double quotes – "&"

```
GR filename.txt "c:\" "c:\Program Files\" /M
```
Problem: The argument "`c:\Program Files\`" is damaged by the operating system and `/M` argument is lost.
Resolution: GR includes detection for this scenario and fixes it. Disable GR's work-around by using the /Z switch.

There are many instances where you may be trying to trouble shoot a complex command line; or maybe GR does not appear to be using arguments in the way you desired.

Use the COMTEST utility to run the desired command line to see how these arguments are being passed to the application.

For example:
```
C:>comtest "c:\program files\" &
COMTEST 1.03 to test Command-line arguments passed by the Operating
System
(C) Sharrad Software 1997.
Number of arguments reported: 2 (0 to 1)

**C:\COMTEST.EXE**
**c:\program files"
```

Download COMTEST from the following locations:
DOS
http://www.sharradsoftware.co.uk/comtest/comtestdos.zip

Win32
http://www.sharradsoftware.co.uk/comtest/comtestwin32.zip

OS/2
http://www.sharradsoftware.co.uk/comtest/comtestos2.zip

This software is provided "as-is" and is provided without any warranty implied or otherwise.

Appendix 3 – Unicode Support

GR will work on all file types, including EXE files and other binary files.

It will also work on Unicode (double-byte) text files with the following important caveat: All text entered on the command line is by nature ASCII (single-byte). This will not match to the desired text within files if those files are Unicode based.

A work-around is to store the search text in a Unicode file itself, and use the /L switch:

```
GR <Unicode file to be searched> /L <file containing Unicode search
text> /L <file containing replacement Unicode text>
```

A further work-around, to automatically convert command line input text to Unicode or the input file from Unicode to ASCII is planned in GR 1.80.

General Replace (GR)
And
General Replace Multi-file (GRX) Utility

Thank you for reading this manual
and for using our software.