GPatch

Ralf Gruner

COLLABORATORS

	TITLE : GPatch		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Ralf Gruner	August 26, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

GPatch

1.1 GPatch Manual

```
                ***RGR***

GCompare / GPatch:  The patch system for program updates

Version 2.5

Author: Ralf Gruner
```

1.2 Introduction

Introduction

GCompare generates patch files for the distribution of updates for any files.

Such a patch file contains only the differences between the old and the new files. This reduces considerably the size of the data you have to distribute.

GPatch applies these patches.

The features of GCompare/GPatch are:

-The patch file can contain patches for any number of files. So you can distribute the update for a lot of versions in one file. If the directory structure of your product is unchanged, then the patcher will find the required patches without any additional actions of your update script. And you can put the patches for different files of your product in one patch file.

-If the patchfile contains the changes from each version of your program to the next then it is not necessary to store all old versions.

-The format of the patchfile is very high optimized. GCompare can try some different coding algorithms and select that with the shortest result.

I do not know too many similar programs, but I think in the most cases GCompare produces the shortest patch files of all available patch programs.

-To avoid corrupt files, the programs contain a very safe error checking (CRC32 signatures for all files). So you can be sure that the result of the patch process is perfect if no error messages appear.

1.3 GCompare

GCompare

GCompare is a shell command. The format is

GCompare <old file> <new file> <patch file> [MODE <n> [VARY <n>] | AUTO [DEEP]] [VERBOSE]

GCompare compares the old file with the new file and stores all changes in the patch file.

If GCompare finds an existing patch file, then it appends the patches after a check on whether the patch file do not contain already patches for the old file.

If a patch file already exists, then GCompare searches for matching data in this file too.

Later GPatch can build the new file using the old file and the patch file.

In case of errors GCompare displays an error message.

Any fatal error produces a return code of 15, and if the result is a corrupt patch file then it will be deleted.

The options MODE, VARY, AUTO and DEEP select the operating mode.

Without any option GCompare works in its fastest mode (the quick mode of the previous GCompare versions). You should use this only if you have no time to wait for other modes (maybe for very large files) or if your files are only modified in few bytes, otherwise the patch file will be much larger than in the other operating modes.

With the option AUTO GCompare selects the optimal type. The additional argument DEEP varies the value for the minimum size of matching data from -1 to 1.

With AUTO DEEP GCompare produces the shortest possible patch file.

If you additional type the option VERBOSE then GCompare tells you after the compare passes the settings for the best result.

With the option MODE you can select the file format immediately. The MODE values <n> can be the numbers 1 to 4.

The option VARY changes the internal minimum values for accepting data as match blocks (automatically tested in the DEEP mode). The minimum is -3.

With the option VERBOSE GCompare prints in all operating modes but AUTO a detailed list of the patch data.

GCompare needs enough free memory for old file, new file and the patch file.

The search speed of GCompare depends now on the remaing memory.

The fastest search algorithm needs altogether 9 times the filesize of the old file plus 514 KB.

The simple list search needs altogether 9 times the filesize of the old file plus 2 KB.

If GCompare can not allocate this memory too, then it works in the mode 3 using linear search.

For the search in the already existing patch file it is the same.

The fastest search needs 9 times the filesize of the patch file plus 514 KB and the simple search needs 9 times the filesize plus 2 KB.

If this memory is not available then GCompare searches only in the old file.

1.4 GPatch

GPatch

GPatch is a shell command. The format is

GPatch <old file> <patch file> <new file> [RECURSIVE] [NOVERSION | QUIET]

GPatch creates the new file using the old file and the patch file.

It selects the matching patches by file size and CRC signature of the old file.

In case of errors GPatch displays an error message.

Any fatal error produces a return code of 15, and if the result is a corrupt new file then it will be deleted.

The argument RECURSIVE selects the recursive operating mode. In this mode GPatch tries again to patch after each successful patch until no matching patches are found, using the new file as old file.

You can use this operating mode if you do not want to store all your old program versions.

Then the patch file should not contain the changes of all old versions to the newest version, but the changes from each version to the next.

The argument NOVERSION suppresses the version information print and the argument QUIET suppresses all text output (without error messages).

1.5 Hints

Discussion and hints

I have got some mail about the speed and the size of the resulting patch file. Some of you found the patch file size much shorter than of other patchers, some of you not, and almost all want a faster compare program.

Since version 2.5 we should not have to speak about speed any more.

But a improvement is possible yet. GCompare would need more than 128 MB of memory for this algorithm. If sometimes this is a matter of course then I can implement it.

Because it is impossible to develop a format of the patch file with always the shortest result (because the file size depends on the distance of the matching data and the way to address it) GCompare has four algorithms.

The optimizer (activated with AUTO) selects the best format, but this takes four times of computing time.

The results of the DEEP mode depends on the compared files. In much cases DEEP produces the same results.

If you want not wait for the optimizer then I suggest you MODE 1. In the most cases this provides a good result.

If you compare the results of GCompare with the results of other programs, you should do this with compressed versions of the patch files too. GCompare has no built in compression (different from other compare programs with run length encoding), because I think all distribution files will always be compressed before they are published and double compressing would decrease the effectivity.

GCompare searches matching data in the patch file too if it already exists. The more patches of different versions of the same program the patch file contains, the more effective the resulting patch file will be, compared with single patch files.

If the versions differ not much, then you should store the changes from one version to the next in the patch file and run GPatch in the recursive mode. So you the patch file will be very small.

GPatch can not handle the old GCompare file formats (I want a small patch program). If you need an older version then use version 1.4 or 1.6.

If GPatch detects an old patch file then it tells you which version you need.

1.6 Technical Details

Technical Details

MODE 1

Mode 1 is the most effective if the most changes are shorter than 128 bytes.

MODE 2

Mode 2 is the most effective for many very small changes.

MODE 3

Mode 3 finds matching data only in a relative area of 64 KB.

It needs only linear search and is practical if you have too little memory.

MODE 4

Mode 4 is similar to mode 1, but better if you have fewer but larger changes.

Quick mode (no option selected)

The quick mode produces the same file format as mode 3, but it searches only in a relative area of 254 bytes for matching data.

After its work GCompare lists some data about the patch file. The listed values are all information stored in the patch file about your files.

The "type" value is the coding algorithm selected by the MODE argument. If you want to know, which value for VARY GCompare has selected in the AUTO DEEP mode, then type the VERBOSE option.

To distinguish the patch data of different files GPatch uses the file size and the CRC signature. The version data or file names are not used. This method is safe, because GCompare use it too to check wheter patch data for a file already exist before it appends new.

1.7 Example

Example

As GCompare example I have included scripts for a fictive program in the drawer "ExampleScripts".

MakePatchFile creates the patch file and UpdateMyProgram applies the patches.

The old versions of the example program "MyProgram" are in "Archive". The patches for the program and a manual "MyProgram.readme" will be applied on the distribution disk "MyProgram".

And for fast trials there is the CompareScript to use GCompare without the need to type shell commands.

I have received an installer example script from Thomas Baust. This script has the name UpdateMyProgram.installer and it patches a program "MyProgram" in a drawer selected by the user. The comments and dialog texts are in German language.

1.8 Versions for other operating systems

Versions for other operating systems

The drawer bin/IX contains versions of GCompare and GPatch for some other computer platforms. The used compiler and the operating systems you can find in the text file Machines.readme .

These programs are not good tested, because I have no access to the hardware.

If anyone have success (or not) with it, then please send a mail to me .

1.9 Copyright, Distribution, Disclaimer

Copyright

Ralf Gruner, An der Sense 5a, D-02779 Großschönau, Germany

ralf.gruner@t-online.de

Distribution

GCompare and GPatch are freeware programs. You can use it for all your needs without restrictions. Also use for commercial products is free.

If you want to see future updates of the programs, please send me an email that you use the programs. There are some things to improve yet, but if nobody need it I can do other things.

Any comments are welcome.

The versions for other operating systems are only for non commercial purposes freeware. For any commercial usage please contact me.

Disclaimer

THIS PROGRAM IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE. THOUGH EVERY CARE HAS BEEN TAKEN TO MAKE THIS PROGRAM SYSTEM-FRIENDLY AND BUGS-FREE, THE ENTIRE RISK AS TO THE RESULTS, RELIABILITY AND PERFORMANCE OF THIS PROGRAM IS ASSUMED BY YOU.

1.10 Credits

Credits

I have to thank for sending me emails, bug reports and test results:

Dirk Stöcker (suggested the recursive mode of GPatch and made the IX ports).

Thomas Baust

Christian Beck

Oliver Blumert

Domenic Gebauer

John Girvin

Matthew Gregan

Bernardo Innocenti

Richard Koerber

Michael Lünse

Tom Newton

Jürgen Reinert

Tobias Schaechtelin

Maik Schreiber

1.11 Index