ImageMagick v6 Examples -- Usage under Windows

Index

Most of the commands in IM Examples were written specifically with LINUX and UNIX shell scripting in mind, as these systems are designed with batch processing and network servers in mind. However, more and more users of ImageMagick want to use it from the Windows Environment. This section provides details and examples of how you can use IM in that environment and, more important, how to convert a UNIX shell command (as used in the rest of the IM Examples) into its Windows DOS equivalent.

I wish to express specific thanks to Wolfgang Hugemann <ImageMagick_AT_Hugemann.de> who completely re-wrote the original notes and expanded them to cover a much larger range of topics, of specific concern to window users. What you see here is his work, and IM users are indebted to him for his time and patience.

Introduction

What's the use of an IM script on my Windows PC?

The following examples basically assume that you run IM on a Windows desktop computer, probably attached to a network. Well, there are a lot of readily available image manipulation programs, such as Adobe Photoshop, Corel's Paint Shop Pro, IrfanView (http://www.irfanview.com/) and even GIMP (http://www.gimp.org/). So why should you bother to perform image processing by IM's command line programs and scripts?

The genuine advantage of using ImageMagick instead of a mouse-driven interface is that you can completely automate routine manipulations, either for single files or for bulks of files. Tasks such as:

Bulk format conversion
: This is offered by quite a lot of Windows programs, such as IrfanView. However, IM's versatility when it comes to image formats is unsurpassed. You can for instance convert all the pages of a PDF into a series of JPEGs (if GhostScript is installed on your computer).

Shrinking and preprocessing digital photographs
: When embedding digital photographs into a word processor document, you should usually reduce their resolution, such that the document can be printed faster. The same holds if you convert the document into a PDF via a PDF printer driver such as FreePDF. Preprocessing could also comprise colour and lens correction routines.

Applying a series of operations to a bulk of digital photographs
Having worked out a series of working steps by use of a mouse-driven program, you might wish to automate these steps for future bulk processing. However, script languages (such as Adobe's Action Script) are not very common in Windows image processing programs.

Combining several images to a catalog image

Although some of these tasks (especially bulk-shrinking) are also offered by other freeware programs (especially by IrfanView's batch processing), you are never free of choice in what processing steps to apply: you have to live with those offered by the program. For instance, IrfanView's batch processing offers to place a text string into a bulk of photographs, but not a logo. It also offers to change the gamma value, but the histogram of the photograph cannot be clipped at its ends (as done by "`-contrast-stretch`" in IM, see Normalize and Contrast Stretch).

IM scripts are especially suited for productive use in a company network, because ready-made scripts can be applied by anyone – end users do not necessarily have to know about what's going on in the background. Thus standard workflow steps on images can be completely automised (and really standardised). Several of the scripts presented in the following were derived for productive use in our small company (working in the field of accident reconstruction).

I am neither an outstanding Windows script programmer nor most familiar with IM's command line tools. There are probably more elegant approaches to some of the problems treated in the following. The points I want to make are:

- to demonstrate some basic techniques of Windows script programming with IM's command line tools.
- to prove that the use of IM-based scripts is neither art for art's sake nor an academic pastime.
- to show that IM's command line tools can do real work in a local network (and not only on webservers).

As in the rest of IM's Example pages, we will only use IM's command line tools and leave its various programming interfaces aside. The scripts are intended to run within a local network with drive letters assigned to each network drive. Most of the scripts are intended to be run on the client computers of that network, few aim at the network (file) server.

Possible environments for IM's command line tools

Under Windows, simple IM commands are usually run in the Windows Command Shell (a "DOS Shell" run by starting `cmd.exe`). For complex operations, performed in a lengthy command line or in a series of command lines, you will better write a script. For a series of simple commands, this will most probably be a DOS batch file, executed in the Windows Command Shell. This approach, however, has its shortcomings, as the batch file command set is rather limited in comparison to those of common UNIX command shells.

When running IM under Windows, you basically have the following alternatives:

The Windows command shell ("DOS window")
This is run by `cmd.exe` (32-bit mode) on Windows NT 4.0, Windows XP and later versions and is present on any Windows computer. See Using the DOS Shell and Batch Files, as well as the special note about the The Convert Issue.

Cygwin
A bash-like command shell (http://www.cygwin.com/). When using this shell, the IM examples presented in the rest of this usage section can be run exactly as given, as you have access to a UNIX style command line shell. See Using Cygwin below.

The Windows Script Host
The Windows Script Host is based on the .COM technology. It is present on any contemporary Windows computer and WSH scripts are much more powerful than simple DOS batch files. The Windows Script Host offers several programming interfaces, with VBScript (Visual Basic Script) and JScript (Java Script) being the most common. The IM command line tools can be invoked by using the DOS shell commands `Run` or `Exec` of the Shell object. See Visual Basic Script (VBS) below.

The Windows Powershell
The much more powerful successor of the ancient DOS shell, based on the .NET 2.0 technology. The Powershell shipped with Windows 7 and is run by `powershell.exe`. It can be downloaded for Windows XP and Vista at Microsoft's website.

Running scripts effectively

Let's assume you have a perfect Windows script (a DOS batch file, a VBScript, or whatever) that takes the name(s) of the input file(s) as command line parameter(s), performs some manipulation and spits out the result as a single image or a series of images. You surely won't like to start a DOS command shell (or an alternative) everytime and provide the script with the filenames by typing them. To avoid such cumbersome ways of proceeding, you can basically use **Drag & Drop** or **SendTo**:

When using **Drag & Drop**, you place the DOS batch file or the VBScript (or whatever) in a location that is easily accessible, like the desktop or the directory which holds the image files to be processed. You then select the files to be processed in the Windows Explorer and just drop them onto the script file. The filenames will be handed over to the script as the command line parameters and can be referred to in the script.

As an alternative, you can place your script (or a link to it) in the **SendTo** folder. The programs in this folder appear in the context menu of the Windows Explorer when right-clicking in the Explorer's file pane. Again, the names of the selected files are handed over to the script as command line parameters. The SendTo folder is named `SendTo`. Its location seems to move with each new Windows version. A bullet-proof way to find it is typing `shell:sendto` into the run box.

A single command line under Windows XP or later can be 8192 (= 2^{13}) characters long. So if you invoke an IM tool directly from the command line, either directly or via a batch file that names all the files needed directly, you will hardly run into this limitation. Drag & Drop however uses the ExecShell routine, which is limited to strings of "only" 2048 (= 2^{11}) characters. As all files are passed with fully qualified filenames (i.e. drive + path + name), this can become a problem for batch files and VBScripts run via WSH when the path names become too long. This errors cannot be properly handled by the script once it occurs, as the error occurs **before** the script is actually run. The solution under Windows XP is usually to place the files in a location where the pathnames are shorter.

The Convert issue

IM's Windows installation routine adds IM's program directory to the search path, such that you can call IM's command line tools directly from the command prompt, without providing a path name. However, the names of IM's command line tools are rather unspecific (e.g. Convert, Identify, Compare ...) which provokes name conflicts with other programs.

The FAT → NTFS convert tool was first shipped with Windows XP and generated a name conflict with IM's command line tool "`convert`" as IM's program directory was **appended** to the DOS search path (i.e. the PATH environment variable), the system tool was found first and simple calls to "`convert`" in older scripts weren't resolved correctly. Current versions of IM's Windows setup program however place IM's program directory at the head of the search path, thereby insuring that in case of conficting names, IM's command line tool is usually found first. However, other utilities with the same name (e.g. Deplhi's report converter) ran into the same problem and went for the same solution, i.e. placing their program path at the head of the path variable, which means that this solution is not bullet-proof: If Delphi is installed after IM, a simple call of Convert will invoke the report converter tool, not IM's Convert.

The introduction of the Convert tool with Windows XP caused a lot of legacy scripts to crash. The common solution was to rename IM's Convert tool to something else, such as "`IMconvert`" (Note that you can not rename the system command, as the next Windows service pack would probably just restore it, ignoring the renamed version.) This solution, although unnecessary now, can still be found all over the Internet.

The best solution to avoid possible future name conflicts is to call IM's command line tools by their full pathname in any script. That is, storing its location in a variable or a constant. So every batch file should start with lines like

```
SETLOCAL EnableDelayedExpansion
SET IMCONV="%PROGRAMFILES%\ImageMagick\Convert"
...
%IMCONV% -size 128x128 xc:white test.gif
```

The code assumes that IM was installed in a folder named "ImageMagick" below the program folder, which is **not** the standard naming of its installation folder. (See Installing ImageMagick under Windows for details.) `%PROGRAMFILES%` is an environment variable which expands to the name of the program directory, i.e. "Program Files" in the English Windows version and "Programme" in the German Windows version. `SETLOCAL` will limit the definition of new environment variables (such as IMCONV) to the scope of the batch file. `EnableDelayedExpansion` is not really needed over here, but it is a good habit to use this option each time you are using `SETLOCAL`; see Guidelines for Batch Programming. for details. There are more sophisticated and bullet-proof ways to find out about IM's installation folder, which will be treated in Editing, Debugging and Runtime Error Testing. The equivalent VB-Script code would be something like

```
Set wsh = WScript.CreateObject("WScript.Shell")
IMconv = wsh.ExpandEnvironmentStrings("%PROGRAMFILES%") & "\ImageMagick\Convert"
```

For reasons of simplicity, we will not use this code everytime in the following, but you should keep this in mind as a good habit. For a good alternative summery and solutions to this problem see Ron Savage: MS Windows and convert.exe.

Character Encoding

ImageMagick encodes strings in Unicode, more precisely in UTF-8. To the contrary, DOS uses codepages to encode characters (mostly in 8-bits). This generates problems when writing strings into images, such as when working with 'label' or '-title': When using non-ASCII charcters, things will go wrong in the easy approach.

For example trying to create a label of German umlauts such as 'ä', 'ö', 'ü', you can simply use the following in Linux...

```
convert label:äöü test.png
```

But this would not create the desired string under windows. You can however read and UTF-8 coded string from a textfile:

```
convert label:umlauts.txt test.png
```

and you can even create this textfile by the use of "`echo`, if you switch the codepage to UTF-8 in advance:

```
CHCP 65001
ECHO äöü > umlauts.txt
convert label:umlauts.txt test.png
```

But if you want to process the output of a DOS command, for example when trying to title an index print of the JPEGs contained in a certain directory with the name of this directory, you are in trouble. Switching to codepage 65001 will not work with most of the DOS commands, especially when looping through directory trees. And switching the codepage to and fro between, say 1252 (West European Latin) and 65001 will usually not work either or become at least rather tricky.

The safest approach is to convert strings when they are needed, using an external command line program, such as "`Iconv.exe`", a UNIX tool which is also available for Windows. Download the setup file from SourgeForge and install the files into the standard directory `C:\Program Files\GnuWin32`. Then dump the output of the DOS command to a text file in your script and convert that file to UTF-8 in the following:

```
CHCP 1252
DIR /B äöü.jpg > temp.txt
"C:\Program Files\Gnuwin32\bin\iconv.exe" -f ISO-8859-1 -t UTF-8 temp.txt > title.txt
convert label:@title.txt äöü.jpg -append äöü_labelled.jpg
```

The parameters tell Iconv to transcode from ISO-8859-1 (codepage 1252) to UTF-8. Iconv writes its output to stdout, so that you have to redirect it to a file in order to use it with 'label'. Please note that dumping the output to a textfile is recommendable anyway, because it tells ImageMagick to interpretate the files contents literally, without taking the Windows backslash ("\") for an escape character.

Of course, the code in the above example does not make much sense as it is presented here for demonstration purposes. In a practical DOS batch file, the file name will probably be generated in a FOR loop. A practical example is given below (See Batch processing a (sub-)directory tree).

For more on UTF handling see the other IM examples and information in Unicode or UTF8 Format Text Handling.

Installing ImageMagick under Windows

program of the current binary release can be found at http://www.imagemagick.org/script/binary-releases.php#windows.

You can also download HDRI (Floating Point Quality) and FFT (Fast Fourier Transform) capable version of ImageMagick from Astronomy and Astrophotography Blog.

By default, IM installs itself in a program directory called `C:\Program Files\ImageMagick x.x.x-x`, where "x.x.x-x" stands for its version number. By default, the setup program suggest to extend the PATH environment variable when IM is installed for the first time (i.e. "Add application path to your system path" is checked). If you forget to de-install older versions, you will quickly have a nice collection of various ImageMagick versions with their corresponding PATH extensions.

In our office, we therefore took the habit to install IM into `C:\Program Files\ImageMagick`, installing newer versions just over the older ones and leaving the PATH environment variable untouched. We couldn't find anything wrong with this way of proceeding in years of usage. If you really want to know the IM version number, you can always call "`convert -version`" and a dummy-proof script can evaluate the version number in case that it relies on a certain minimum version number. See section "Editing, Debugging and Runtime Error Testing" for more information on this.

ImageMagick writes a few Registry keys to `HKEY Local Machine\Software\ImageMagick`. If you do deal with several installed versions of IM, the most important key is `HKEY Local Machine\Software\ImageMagick\Current`, where you can also find the path to IM's binarys, called `BinPath`. You can query this registry entry at the start of any script and thus determine the program path without having to rely on the PATH environment variable. See section "Editing, Debugging and Runtime Error Testing" for more information on this.

IM's setup program offers the option to install a COM+ object for ImageMagick by the option *Install ImageMagickObject OLE Control for VBscript, Visual Basic, and WSH*. The option is not checked by default and the installation of the COM+ object is **no prerequisite** to use IM in a VBScript, as will be proven in the following. Anyway, you should not rely on the COM+ object to be installed on a script target machine other than your own.

When working with PostScript files, ImageMagick relies on another program, "`Ghostscript`" for the reading and conversion of PostScript and PDF files into a image format it can use. That is to read such documents, Ghostscript needs to be installed on your computer. Its latest version can be downloaded at SourceForge.

The order in which you install GhostScript and ImageMagick does not matter. You do not have to install GhostScript prior to ImageMagick, and ImageMagick will run fine without it being installed. It is only needed if you want to work with Postscript or PDF files. IM will determines the location of Ghostscript at runtime, quering it from the Registry.

Specialities and pitfalls

Quite extraordinarily for Windows programs, IM allows images to be written to *stdout* and read from *stdin* and thus use piping to cascade image processing tasks. The operation

```
convert -size 128x128 xc:white gif:- | convert - test.gif
```

is equivalent to

```
convert -size 128x128 xc:white test.gif
```

In the first command of the pipe, the minus operator tells IM to write the image to *stdout*, while the minus operator in the second command of the pipe tells IM to read the image from *stdin*. This was of proceeding allows to avoid the explicit use temporary files. It is especially useful if some command does not offer certain operations, for example trimming:

```
montage -tile 2x2 -geometry 400x300+60+60 1.png 2.png 3.png 4.png miff:- | convert - -trim montage.png
```

As a consequence of this feature, textual output is usually written to *stderr* instead of *stdout*. For example: if you want to redirect the textual output of Compare to a text file, you would have to write

```
compare -metric PSNR 1.png 2.png dif_1_2.png 2>result.txt
```

So you have to redirect *stderr* ("2>") to the text file, not *stdout* ("1>" or just ">").

Getting Help

The command line options are extensively documented on IM's website, but it's the Usage Section of its website which really demonstrates how to get them to work. This section of the website is quite well structured, allowing a problem-oriented approach to the task you would like to perform. However the quick-and-dirty approach is a Google search on that section of the website in regard to the command line option that you have in mind. If you want to perform a montage of several images and want to find out about the *-tile* option you could perform a Google search on either

- Imagemagick Usage Tile
- tile site:www.imagemagick.org/Usage

and you will quickly find out about the essentials. Just keep in mind that this will reveal code intended for application in a UNIX / LINUX environment, which has to be slightly adjusted in order to be applied under Windows.

Another very helpful source of information is the IM discussion board, aka the Discourse Server User Forum, which you should incorporate in the bookmarks / favourites of your browser. Becoming a member will allow you to pose questions, which are mostly answered quickly be knowledgable users.

Auxiliary programs and alternatives

Yes, it's true. There **are** certain jobs that other programs can do better than ImageMagick. Typically ones which are designed with specific formats in mind, rather than general image manipulation that ImageMagick provides.

- IrfanView is probably the most common image viewer under Windows, which also allows some basic image manipulation.
- A GUI program like Adobe Photoshop or Gimp is more suited for direct editing and testing complex image manipulation steps.

- Extraction of JPEG streams from PDFs should be done with xPDF.
- Video processing should better be done with VirtualDub, best of all in combination with AviSynth and its editor AvsP.

This is not to say ImageMagick should be ignored for such image work. But you can do a lot more by combining them together.

Using Cygwin

As has been said above, ImageMagick was designed with UNIX and Linux in mind, so the easiest approach is probably to install a Bash shell on your Windows system and run the variety of IM scripts which have already been written for that system, for example Fred Weinhaus' scripts.

Cygwin is – as its developers put it, provides a "a Linux-like environment for Windows". It consists of all the tools which are normally available in the Linux shell. I have tested a few of Fred Weinhaus' bash scripts under Cygwin's Bash shell and found them to be fully functional.

At the bottom of the root page of the Cygwin website, you will find a link labeled Install or update now!, which will download an installation stub named Setup.exe. When you start this program, it will offer a list of site mirrors. After you have chosen one of them, the routine will provide a tree view of what tools it is going to install. The standard selection seems reasonable to me, so you might just proceed. The installation routine will then download the packages needed and install Cygwin on your computer.

When you start Cygwin, its Bash shell looks pretty much like a DOS box, that is a text window with black background. Like for the DOS window, the font can be chosen by clicking on the window's system menu, located left in the title bar. The basic commands are:

- You change the current directory with `CD` command, more or less like in DOS. However, the backslash ("\") has to be substituted by the forward slash ("/"). Drives are also changed via `CD`, as under DOS. As such `CD w:` will switch to drive w: Unlike in a nomal Unix environment, pathnames are case-insensitive. Special characters, such as umlauts, can be used. For some commands drivenames have to be passed in a special syntax, avoiding the colon. For example `/cygdrive/w/test`.
- Cywin reads the Windows PATH environment variable and sets its own PATH correspondingly. You can check that by typing `echo $PATH` into the Bash shell. Note: Unlike under Windows, the names of environment variables are case-sensitive, so you have to use capital letters when referring to PATH.
- Unlike under Windows, the current directory in **not** in the search path by default! If for example the shell script "autolevel1" resides in `W:\scripts`, you can not just to switch to that directory when calling the shell script. You must append at least a minimal directory location to the start of the script, like this: `./autolevel1` (for the script located in current directory). Or the script complete path, like this `/cygdrive/w/scripts`.
- Alturnatively you can append that script directory explicitly to the search path using `PATH=$PATH:/cygdrive/w/scripts`. As the colon is used as a path separator, you have to use this special nomenclature, such as `/cygdrive/w/scripts` instead of `w:/scripts`.

This description of the Cygwin shell will be extended in future versions of this page. For the moment, the above information should suffice to get you started.

Using the DOS Shell and Batch Files

Converting Scripts: UNIX Shell to Window DOS

When invoking IM commands directly from the DOS command shell (using `cmd.exe`) you have to modify the scripts presented on IM's Example site (if they don't stem from this section), as most examples provided (in other sections) are generally intended to be run in a UNIX or LINUX command shell.

In order to run them from a DOS command shell, you have to perform the following modifications:

- Most often, double quotes '"' have to be used in place of single quotes "'" so that the command arguments remain correct. Watch out for quotes within quotes such as in `-draw` operator. You can use single quotes within a DOS double quoted argument as these are passed to IM for handling, and not processed by the script.
- Backslashes '\' appearing on the end of the shown example lines represents a 'line continuation' which appends the next line to the same command line sequence.

  Replace them with '^' character to denote line continuation in DOS batch files. For DOS the next line will also need to start with a space, however that is fairly standard practice so should not cause much of a problem.

  You can also append all the lines onto one line and remove those backslashes, though this makes editing, and later reading the command much harder. As such the practice is not recommended.
- All reserved shell characters which are not in double quotes must be escaped with a '^' (caret or circumflex) if used in a literal sense (i.e. not fulfilling their usual purpose). These reserved shell characters are: '&', '|', '(', ')', '<', '>', '^'.

  This especially means that:
  - The special character '>' (used for resize geometry) needs to be escaped using '^'. For example "`-resize 100x100^>`"
  - Similarly the 'internal fit resize' flag '^' needs to be doubled to become '^^'.
- UNIX shell escaping backslashes '\' are not needed to escape parenthesis '()' or exclamation marks '!'.
- Otherwise the UNIX shell escaping backslashes '\' will need to be replaced with a circumflex '^' character, when that escape characters such as '<' and '>'.
  For example: "`-resize 100x100\>`" will become "`-resize 100x100^>`".
- In DOS batch files, the percent '%' character also has a special meaning as it references to the command line parameters. For example `%1, %2, ...` (in UNIX shell a '$' sign is used for the same general meaning). In a DOS batch file, single percent signs (as they appear in the "`FOR` command") will need to be doubled to '%%'

  ImageMagick itself generally only looks to see if a percentage sign is present, and does not care if more than one has been given. So unless it is part of a text label or comment string, doubling all percent signs generally does not hurt.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.