User manual for Netpbm

Netpbm

Updated: 24 August 2006
<?makeman .SH NAME ?> <?makeman netpbm   netpbm library overview ?>

Overview Of Netpbm

Netpbm is a package of graphics programs and a programming library.

There are over 220 separate programs in the package, most of which have "pbm", "pgm", "ppm", "pam", or "pnm" in their names. For example, **pamscale** and **giftopnm**.

For example, you might use **pamscale** to shrink an image by 10%. Or use **pamcomp** to overlay one image on top of another. Or use **pbmtext** to create an image of text. Or reduce the number of colors in an image with **pnmquant**.

Netpbm is an open source software package, distributed via the Sourceforge **netpbm** project.

Table Of Contents

History

Author

The Netpbm Programs

The Netpbm programs are generally useful run by a person from a command shell, but are also designed to be used by programs. A common characteristic of Netpbm programs is that they are simple, fundamental building blocks. They are most powerful when stacked in pipelines. Netpbm programs do not use graphical user interfaces and do not seek input from a user. The only programs that display graphics at all are the very primitive display programs **pamx** and **ppmsvgalib**, and they don't do anything but that.

Each of these programs has its own manual, as linked in the directory below.

The Netpbm programs can read and write files greater than 2 GiB wherever the underlying system can. There may be exceptions where the programs use external libraries (The JPEG library, etc.) to access files and the external library does not have large file capability. Before Netpbm 10.15 (April 2003), no Netpbm program could read a file that large.

Common Options

There are a few options that are present on all programs that are based on the Netpbm library, including virtually all Netpbm programs. These are not mentioned in the individual manuals for the programs.

You can use two hyphens instead of one on these options if you like.

-quiet Suppress all informational messages that would otherwise be issued to Standard Error. (To be precise, this only works to the extent that the program in question implements the Netpbm convention of issuing all informational messages via the **pm_message()** service of the Netpbm library).

-version Instead of doing anything else, report the version of the **libnetpbm** library linked with the program (it may have been linked statically into the program, or dynamically linked at run time). Normally, the Netpbm programs and the library are installed at the same time, so this tells you the version of the program and all the other Netpbm files it uses as well.

-plain If the program generates an image in Netpbm format, generate it in the "plain" (aka "ascii") version of the format, as opposed to the "raw" (aka "binary") version.

This option was introduced in Netpbm 10.10 (October 2002).

Directory

Here is a complete list of all the Netpbm programs (with links to their manuals):

Netpbm program directory

How To Use The Programs

As a collection of primitive tools, the power of Netpbm is multiplied by the power of all the other unix tools you can use with them. These notes remind you of some of the more useful ways to do this. Often, when people want to add high level functions to the Netpbm tools, they have overlooked some existing tool that, in combination with Netpbm, already does it.

Often, you need to apply some conversion or edit to a whole bunch of files.

As a rule, Netpbm programs take one input file and produce one output file, usually on Standard Output. This is for flexibility, since you so often have to pipeline many tools together.

Here is an example of a shell command to convert all your of PNG files (named *.png) to JPEG files named *.jpg:

```
for i in *.png; do pngtopnm $i | ppmtojpeg >`basename $i .png`.jpg; done
```

Or you might just generate a stream of individual shell commands, one per file, with awk or perl. Here's how to brighten 30 YUV images that make up one second of a movie, keeping the images in the same files:

```
ls *.yuv
  | perl -ne 'chomp;
  print yuvtoppm $_ | ppmbrighten -v 100 | ppmtoyuv >tmp$$.yuv;
  mv tmp$$.yuv $_
  '
  | sh
```

The tools **find** (with the **-exec** option) and **xargs** are also useful for simple manipulation of groups of files.

Some shells' "process substitution" facility can help where a non-Netpbm program expects you to identify a disk file for input and you want it to use the result of a Netpbm manipulation. Say the hypothetical program **printcmyk** takes the filename of a Tiff CMYK file as input and what you have is a PNG file **abc.png**. Try:

```
printcmyk <({ pngtopnm abc.png | pnmtotiffcmyk ; })
```

It works in the other direction too, if you have a program that makes you name its output file and you want the output to go through a Netpbm tool.

The Netpbm Formats

All of the programs work with a set of graphics formats called the "netpbm" formats. Specifically, these formats are pbm, pgm, ppm, and pam. The first three of these are sometimes known generically as "pnm". Many of the Netpbm programs convert from a Netpbm format to another format or vice versa. This is so you can use the Netpbm programs to work on graphics of any format. It is also common to use a combination of Netpbm programs to convert from one non-Netpbm format to another non-Netpbm format. Netpbm has converters for about 100 graphics formats, and as a package Netpbm lets you do more graphics format conversions than any other computer graphics facility.

The Netpbm formats are all raster formats, i.e. they describe an image as a matrix of rows and columns of pixels. In the PBM format, the pixels are black and white. In the PGM format, pixels are shades of gray. In the PPM format, the pixels are in full color. The PAM format is more sophisticated. A replacement for all three of the other formats, it can represent matrices of general data including but not limited to black and white, grayscale, and color images.

Programs designed to work with PBM images have "pbm" in their names. Programs designed to work with PGM, PPM, and PAM images similarly have "pgm", "ppm", and "pam" in their names.

All Netpbm programs designed to read PGM images see PBM images as if they were PGM too. All Netpbm programs designed to read PPM images see PGM and PBM images as if they were PPM. See Implied Format Conversion.

Programs that have "pnm" in their names read PBM, PGM, and PPM but unlike "ppm" programs, they distinguish between them and their function depends on the format. For example, **pnmtopng** creates a black and white PNG output image if its input is PBM or PGM, but a color PNG output image if its input is PPM. And **pnmrotate** produces an output image of the same format as the input. A hypothetical **ppmrotate** program would also read all three PNM input formats, but would see them all as PPM and would always generate PPM output.

Programs that have "pam" in their names read all the Netpbm formats: PBM, PGM, PPM, and PAM. They sometimes treat them all as if they are PAM, using an implied conversion, but often they recognize the individual formats and behave accordingly, like a "pnm" program does. See Implied Format Conversion.

If it seems wasteful to you to have three separate PNM formats, be aware that there is a historical reason for it. In the beginning, there were only PBMs. PGMs came later, and then PPMs. Much later came PAM, which realizes the possibility of having just one aggregate format.

The formats are described in the specifications of **pbm**, **pgm**, **ppm**, and **pam**.

Implied Format Conversion

A program that uses the PGM library subroutines to read an image can read a PBM image as well as a PGM image. The program sees the PBM image as if it were the equivalent PGM image, with a maxval of 255. note: This sometimes confuses people who are looking at the formats at a lower layer than they ought to be because a zero value in a PBM raster means white, while a zero value in a PGM raster means black.

A program that uses the PPM library subroutines to read an image can read a PGM image as well as a PPM image and a PBM image as well as a PGM image. The program sees the PBM or PGM image as if it were the equivalent PPM image, with a maxval of 255 in the PBM case and the same maxval as the PGM in the PGM case.

A program that uses the PAM library subroutines to read an image can read a PBM, PGM, or PPM image as well as a PAM image. The program sees a PBM image as if it were the equivalent PAM image with tuple type **BLACKANDWHITE**. It sees a PGM image as if it were the equivalent PAM image with tuple type **GRAYSCALE**. It sees a PPM image as if it were the equivalent PAM image with tuple type **RGB**. But the program actually can see deeper if it wants to. It can tell exactly which format the input was and may respond accordingly. For example, a PAM program typically produces output in the same format as its input.

Netpbm and Transparency

In many graphics formats, there's a means of indicating that certain parts of the image are wholly or partially transparent, meaning that if it were displayed "over" another image, the other image would show through there. Netpbm formats deliberately omit that capability, since their purpose is to be extremely simple.

In Netpbm, you handle transparency via a transparency mask in a separate (slightly redefined) PGM image. In this pseudo-PGM, what would normally be a pixel's intensity is instead an opaqueness value. See **pgm**. **pamcomp** is an example of a program that uses a PGM transparency mask.

Another means of representing transparency information has recently developed in Netpbm, using PAM images. In spite of the argument given above that Netpbm formats should be too simple to have transparency information built in, it turns out to be extremely inconvenient to have to carry the transparency information around separately. This is primarily because Unix shells don't provide easy ways to have networks of pipelines. You get one input and one output from each program in a pipeline. So you'd like to have both the color information and the transparency information for an image in the same pipe at the same time.

For that reason, some new (and recently renovated) Netpbm programs recognize and generate a PAM image with tuple type RGB_ALPHA or GRAYSCALE_ALPHA, which contains a plane for the transparency information. See the PAM specification.

The Netpbm Library

The Netpbm programming library, **libnetpbm**, makes it easy to write programs that manipulate graphic images. Its main function is to read and write files in the Netpbm formats, and because the Netpbm package contains converters for all the popular graphics formats, if your program reads and writes the Netpbm formats, you can use it with any formats.

But the library also contain some utility functions, such as character drawing and RGB/YCrCb conversion.

The library has the conventional C linkage. Virtually all programs in the Netpbm package are based on the Netpbm library.

netpbm-config

In a standard installation of Netpbm, there is a program named **netpbm-config** in the regular program search path. We don't consider this a Netpbm program -- it's just an ancillary part of a Netpbm installation. This program tells you information about the Netpbm installation, and is intended to be run by other programs that interface with Netpbm. In fact, **netpbm-config** is really a configuration file, like those you typically see in the */etc/* directory of a Unix system.

Example:

```
$netpbm-config --datadir
/usr/local/netpbm/data
```

If you write a program that needs to access a Netpbm data file, it can use such a shell command to find out where the Netpbm data files are.

netpbm-config is the only file that must be installed in a standard directory (it must be in a directory that is in the default program search path). You can use **netpbm-config** as a bootstrap to find all the other Netpbm files.

There is no detailed documentation of **netpbm-config**. If you're in a position to use it, you should have no trouble reading the file itself to figure out how to use it.

Memory Usage

An important characteristic that varies among graphics software is how much memory it uses, and how. Does it read an entire image into memory, work on it there, then write it out all at once? Does it read one and write one pixel at a time? In Netpbm, it differs from one program to the next, but there are some generalizations we can make.

Most Netpbm programs keep one row of pixels at a time in memory. Such a program reads a row from an input file, processes it, then writes a row to an output file. Some programs execute algorithms that can't work like that, so they keep a small window of rows in memory. Others must keep the entire image in memory. If you think of what job the program does, you can probably guess which one it does.

When Netpbm keeps a pixel in memory, it normally uses a lot more space for it than it occupies in the Netpbm image file format.

The older programs (most of Netpbm) use 12 bytes per pixel. This is true even for a PBM image, for which it only really takes one bit to totally describe the pixel. Netpbm does this expansion to make implementing the programs easier -- it uses the same format regardless of the type of image.

Newer programs use the "pam" family of library functions internally, which use memory a little differently. These functions are designed to handle generic tuples with a variable numbers of planes, so no fixed size per-tuple storage is possible. A program of this type uses 4 bytes per sample (a tuple is composed of samples), plus another 4 bytes per tuple. In a graphic image, a tuple is a pixel. So an ordinary color image takes 16 bytes per pixel.

When considering memory usage, it is important to remember that memory and disk storage are equivalent in two ways:

> Memory is often virtual, backed by swap space on disk storage. So accessing memory may mean doing disk I/O.
>
> Files are usually cached and buffered, so that accessing a disk file may just mean accessing memory.

This means that the consequences of whether a program works from the image file or from a memory copy are not straightforward.

Note that an image takes a lot less space in a Netpbm format file, and therefore in an operating system's file cache, than in Netpbm's in-memory format. In non-Netpbm image formats, the data is even smaller. So reading through an input file multiple times instead of keeping a copy in regular memory can be the best use of memory, and many Netpbm programs do that. But some files can't be read multiple times. In particular, you can't rewind and re-read a pipe, and a pipe is often the input for a Netpbm program. Netpbm programs that re-read files detect such input files and read them into a temporary file, then read that temporary file multiple times.

A few Netpbm programs use an in-memory format that is just one bit per pixel. These are programs that convert between PBM and a format that has a raster format very much like PBM's. In this case, it would actually make the program more complicated (in addition to much slower) to use Netpbm's generic 12 byte or 8 byte pixel representation.

By the way, the old axiom that memory is way faster than disk is not necessarily true. On small systems, it typically is true, but on a system with a large network of disks, especially with striping, it is quite easy for the disk storage to be capable of supplying data faster than the CPU can use it.

CPU Usage

People sometimes wonder what CPU facilities Netpbm programs and the Netpbm programming library use. The programs never depend on particular features existing (assuming they're compiled properly), but the speed and cost of running a program varies depending upon the CPU features.

One area of particular importance is floating point arithmetic. The Netpbm image formats are based on integers, and Netpbm arithmetic is done with integers where possible. But there is one significant area that is floating point: programs that must deal with light intensity. The Netpbm formats use integers that are proportional to brightness, and brightness is exponentially related to light intensity. The programs have to keep the intermediate intensity values in floating point in order not to lose precision. And the conversion (gamma function) between the two is heavy-duty floating point arithmetic. Programs that mix pixels together have to combine light intensity, so they do heavy floating point. Three of the most popular Netpbm programs do that: **pamscale** (shrink/expand an image), pamcomp (overlay an image over another one), and pamditherbw (Make a black and white image that approximates a grayscale image).

The Netpbm image formats use 16 bit integers. The Netpbm code uses "unsigned int" size integers to work with them.

Companion Software

PHP-NetPBM

If you're using Netpbm to do graphics for a website, you can invoke the Netpbm programs from a PHP script. To make this even easier, check out PHP-NetPBM, a PHP class that interacts with Netpbm. Its main goal is to decrease the pain of using Netpbm when working with images in various formats. It includes macro commands to perform manipulations on many files.

I can't actually recommend PHP-NetPBM. I spent some time staring at it and was unable to make sense of it. Some documentation is in fractured English and other is in an unusual character set. But a PHP expert might be able to figure it out and get some use out of it.

Other Graphics Software

Netpbm contains primitive building blocks. It certainly is not a complete graphics software library.

Graphics Viewers

The first thing you will want to make use of any of these tools is a viewer. (On GNU/Linux, you can use **ppmsvgalib** in a pinch, but it is pretty limiting). **zgv** is a good full service viewer to use on a GNU/Linux system with the SVGALIB graphics display driver library. You can find **zgv** at **ftp://ftp.ibiblio.org/pub/Linux/apps/graphics/viewers/svga.**

zgv even has a feature in it wherein you can visually crop an image and write an output file of the cropped image using **pamcut**. See the **-s** option to **zgv**.

For the X inclined, there is also **xzgv**.

xloadimage and its extension **xli** are also common ways to display a graphic image in X.

gqview is a more modern X-based image viewer.

qiv is a small, very fast viewer for X.

To play mpeg movies, such as produced by **ppmtompeg**, try **mplayer** or **xine.**

See **ftp://metalab.unc.edu/pub/Linux/apps/graphics/viewers/X**.

Visual Graphics Software

Visual graphics software is modern point-and-click software that displays an image and lets you work on it and see the results as you go. This is fundamentally different from what Netpbm programs do.

ImageMagick is like a visual version of Netpbm. Using the X/Window system on Unix, you can do basic editing of images and lots of format conversions. The package does include at least some non-visual tools. **convert**, **mogrify**, **montage**, and **animate** are popular programs from the **ImageMagick** package. **ImageMagick** runs on Unix, Windows, Windows NT, Macintosh, and VMS.

xv is a very old and very popular simple image editor in the Unix world. It does not have much in the way of current support, or maintenance, though.

The Gimp is a visual image editor for Unix and X, in the same category as the more famous, less capable, and much more expensive Adobe Photoshop, etc. for Windows. See **http://www.gimp.org**.

Electric Eyes, **kuickshow**, and **gthumb** are also visual editors for the X/Window system, and **KView** and **gwenview** are specifically for KDE.

Programming Tools

If you're writing a program in C to draw and manipulate images, check out gd. Netpbm contains a C library for drawing images, but it is probably not as capable or documented as **gd**. You can easily run any Netpbm program from a C program with the **pm_system** function from the Netpbm programming library, but that is less efficient than **gd** functions that do the same thing.

Ilib is a C subroutine library with functions for adding text to an image (as you might do at a higher level with **pbmtext**, **pamcomp**, etc.). It works with Netpbm input and output. Find it at **k5n.us**. Netpbm also includes character drawing functions in the **libnetpbm** library, but they do not have as fancy font capabilities (see **ppmlabel** for an example of use of the Netpbm character drawing functions).

GD is a library of graphics routines that is part of PHP. It has a subset of Netpbm's functions and has been found to resize images more slowly and with less quality.

Tools For Specific Graphics Formats

mencode, which is part of the **mplayer** package, creates movie files. It's like a much more advanced version of **ppmtompeg**, without the Netpbm building block simplicity.

To create an animated GIF, or extract a frame from one, use **gifsicle**. **gifsicle** converts between animated GIF and still GIF, and you can use **ppmtogif** and **giftopnm** to connect up to all the Netpbm utilities. See **http://www.lcdf.org/gifsicle**.

To convert an image of text to text (optical character recongition - OCR), use **gocr** (think of it as an inverse of **pbmtext**). See **http://altmark.nat.uni-magdeburg.de/˜jschulen/ocr/**.

http://schaik.com/pngsuite contains a PNG test suite -- a whole bunch of PNG images exploiting the various features of the PNG format.

Another version of Netpbm's **pnmtopng/pngtopnm** is at **http://www.schaik.com/png/pnmtopng.html**.

The version in Netpbm was actually based on that package a long time ago, and you can expect to find better exploitation of the PNG format, especially recent enhancements, in that package. It may be a little less consistent with the Netpbm project and less exploitive of recent Netpbm format enhancements, though.

pngwriter is a C++ library for creating PNG images. With it, you plot an image pixel by pixel. You can also render text with the FreeType2 library.

jpegtran Does some of the same transformations as Netpbm is famous for, but does them specifically on JPEG files and does them without loss of information. By contrast, if you were to use Netpbm, you would first decompress the JPEG image to Netpbm format, then transform the image, then compress it back to JPEG format. In that recompression, you lose a little image information because JPEG is a lossy compression. Of course, only a few kinds of lossless transformation are possible. **jpegtran** comes with the Independent Jpeg Group's (http://www.ijg.org) JPEG library.

Some tools to deal with EXIF files (see also Netpbm's **jpegtopnm** and **pnmtojpeg**): To dump (interpret) an EXIF header: Exifdump ((http://topo.math.u-psud.fr/~bousch/exifdump.py)) or Jhead.

A Python EXIF library and dumper: http://pyexif.sourceforge.net.

Here's some software to work with IOCA (Image Object Content Architecture): ImageToolbox ($2500, demo available). This can convert from TIFF -> IOCA and back again. Ameri-Imager ($40 Windows only).

pnm2ppa converts to HP's "Winprinter" format (for HP 710, 720, 820, 1000, etc). It is a superset of Netpbm's **pbmtoppa** and handles, notably, color. However, it is more of a printer driver than a Netpbm-style primitive graphics building block. See The Pnm2ppa /Sourceforge Project

Document/Graphics Software

There is a large class of software that does document processing, and that is somewhat related to graphics because documents contain graphics and a page of a document is for many purposes a graphic image. Because of this slight intersection with graphics, I cover document processing software here briefly, but it is for the most part beyond the scope of this document.

First, we look at where Netpbm meets document processing. **pstopnm** converts from Postscript and PDF to PNM. It effectively renders the document into images of printed pages. **pstopnm** is nothing but a convenient wrapper for Ghostscript, and in particular Netpbm-format device drivers that are part of it. **pnmtops** and **pbmtoepsi** convert a PNM image to a Postscript program for printing the image. But to really use PDF and Postscript files, you generally need more complex document processing software.

Adobe invented Postscript and PDF and products from Adobe are for many purposes the quintessential Postscript and PDF tools.

Adobe's free Acrobat Reader displays PDF and converts to Postscript. The Acrobat Reader for unix has a program name of "acroread" and the -toPostScript option (also see the -level2 option) is useful.

Other software from Adobe, available for purchase, interprets and creates Postscript and PDF files. "Distill" is a program that converts Postscript to PDF.

xpdf also reads PDF files.

GSview, ghostview, gv, ggv, and kghostview are some other viewers for Postscript and PDF files.

The program **ps2pdf**, part of Ghostscript, converts from Postscript to PDF.

Two packages that produce more kinds of Encapsulated Postscript than the Netpbm programs, including compressed kinds, are bmeps and imgtops.

dvips converts from DVI format to Postscript. DVI is the format that Tex produces. Netpbm can convert from Postscript to PNM. Thus, you can use these in combination to work with Tex/Latex documents graphically.

wvware converts a Microsoft Word document (.doc file) to various other formats. While the web page doesn't seem to mention it, it reportedly can extract an embedded image in a Word document as a PNG.

Document Printer converts various print document formats (Microsoft Word, PDF, HTML, etc.) to various graphic image formats. ($38, Windows only).

Latex2html converts Latex document source to HTML document source. Part of that involves graphics, and Latex2html uses Netpbm tools for some of that. But Latex2html through its history has had some rather esoteric codependencies with Netpbm. Older Latex2html doesn't work with current Netpbm. Latex2html-99.2beta8 works, though.

Other

The **file** program looks at a file and tells you what kind of file it is. It recognizes most of the graphics formats with which Netpbm deals, so it is pretty handy for graphics work. Netpbm's **anytopnm** program depends on **file.** See **ftp://ftp.astron.com/pub/file**.

The Utah Raster Toolkit serves a lot of the same purpose as Netpbm, but without the emphasis on format conversions. This package is based on the RLE format, which you can convert to and from the Netpbm formats. The website of the Geometric Design And Computation group in the Department of Computer Science at University of Utah used to (ca. 2002) have information on the Utah Raster Toolkit, but now it doesn't.

Ivtools is a suite of free X Windows drawing editors for Postscript, Tex, and web graphics production, as well as an embeddable and extendable vector graphic shell. It uses the Netpbm facilities. See **http://www.ivtools.org**.

The program **morph** morphs one image into another. It uses Targa format images, but you can use **tgatoppm** and **ppmtotga** to deal with that format. You have to use the graphical (X/Tk) Xmorph to create the mesh files that you must feed to **morph**. **morph** is part of the Xmorph package. See **http://www.colorado-research.com/~gourlay/software/Graphics/Xmorph**.

Other Graphics Formats

People never seem to tire of inventing new graphics formats, often completely redundant with pre-existing ones. Netpbm cannot keep up with them. Here is a list of a few that we know Netpbm does not handle (yet).

Various commercial Windows software handles dozens of formats that Netpbm does not, especially formats typically used with Windows programs. ImageMagick is probably the most used free image format converter and it also handles lots of formats Netpbm does not.

VRML (Virtual Reality Modelling Language)

CAL (originated by US Department Of Defense, favored by architects). http://www.landfield.com/faqs/graphics/fileformats-faq/part3/section-24.html

array formats dx, general, netcdf, CDF, hdf, cm

CGM+

Windows Meta File (.WMF). Libwmf converts from WMF to things like Latex, PDF, PNG. Some of these can be input to Netpbm.

Microsoft Word .doc format. Microsoft keeps a proprietary hold on this format. Any software you see that can handle it is likely to cost money.

RTF

DXF (AutoCAD)

IOCA (Image Object Content Architecture) The specification of this format is documented by IBM: Data Stream and Object Architectures: Image Object Content Architecture Reference. See above for software that processes this format.

OpenEXR is an HDR format (like PFM). See http://www.openexr.com.

Xv Visual Schnauzer thumbnail image. This is a rather antiquated format used by the Xv program. In Netpbm circles, it is best known for the fact that it is very similar to Netpbm formats and uses the same signature ("P7") as PAM because it was developed as sort of a fork of the Netpbm format specifications.

YUV 4:2:0, aka YUV 420, and the simlar YUV 4:4:4, YUV 4:2:2, YUV 4:1:1, YUV 4:1:1s, and YUV 4:1:0. Video systems often use this.

History

Netpbm has a long history, starting with Jef Poskanzer's Pbmplus package in 1988. The file *HISTORY* in the Netpbm source code contains a historical overview as well as a detailed history release by release.

Author

Netpbm is based on the Pbmplus package by Jef Poskanzer, first distributed in 1988 and maintained by him until 1991. But the package contains work by countless other authors, added since Jef's original work. In fact, the name is derived from the fact that the work was contributed by people all over the world via the Internet, when such collaboration was still novel enough to merit naming the package after it.

Bryan Henderson has been maintaining Netpbm since 1999. In addition to packaging work by others, Bryan has also written a significant amount of new material for the package.