Libnetpbm Image Processing Manual

Libnetpbm User's Guide

<?makeman .SH NAME ?> <?makeman libnetpbm_ug netpbm sample code ?>

The Libnetpbm programming library is part of Netpbm.

Example

Here is an example of a C program that uses **libnetpbm** to read a Netpbm image input and produce a Netpbm image output.

```
  /* Example program fragment to read a PAM or PNM image
     from stdin, add up the values of every sample in it
     (I don't know why), and write the image unchanged to
     stdout. */
  #include <pam.h>
  struct pam inpam, outpam;
  unsigned int row;
  pnm_init(&argc, argv);
  pnm_readpaminit(stdin, &inpam, PAM_STRUCT_SIZE(tuple_type));
  outpam = inpam; outpam.file = stdout;
  pnm_writepaminit(&outpam);
  tuplerow = pnm_allocpamrow(&inpam);
  for (row = 0; row < inpam.height; row++) {
     unsigned int column;
     pnm_readpamrow(&inpam, tuplerow);
     for (column = 0; column < inpam.width; ++column) {
        unsigned int plane;
        for (plane = 0; plane < inpam.depth; ++plane) {
           grand_total += tuplerow[column][plane];
        }
     }
     pnm_writepamrow(&outpam, tuplerow); }
  pnm_freepamrow(tuplerow);
```

Guide To Using Libnetpbm

libnetpbm classes

In this section, we cover only the PAM functions in **libnetpbm**. As described in the introduction to **libnetpbm**, there are four other classes of image processing functions (PBM, PGM, PPM, PNM). They are less important, since you can do everything more easily with the PAM functions, but if you're working on old programs or need the extra efficiency those older functions can sometimes provide, you can find them documented as here: PBM Function Manual, PGM Function Manual,PPM Function Manual, and PNM Function Manual.

In case you're wondering, what makes the PAM functions easier to use is:

Each function handles all the formats. It does so without converting to a common format, so your

program can treat the different formats differently if it wants. However, the interface makes it easy for your program to ignore the differences between the formats if that's what you want.

The PAM function parameter lists convey most information about the image with which you're working with a single **pam** structure, which you can build once and use over and over, whereas the older functions require you to pass up to 5 pieces of image information (height, width, etc.) as separate arguments to every function.

THE pam STRUCTURE

The PAM functions take most of their arguments in the form of a single **pam** structure. This is not an opaque object, but just a convenient way to organize the information upon which most the functions depend. So you are free to access or set the elements of the structure however you want. But you will find in most cases it is most convenient to call **pnm_readpaminit()** or **pnm_writepaminit()** to set the fields in the **pam** structure before calling any other pam functions, and then just to pass the structure unchanged in all future calls to pam functions.

The fields are:

size The storage size in bytes of this entire structure.

len The length, in bytes, of the information in this structure. The information starts in the first byte and is contiguous. This cannot be greater than **size**. **size** and **len** can be used to make programs compatible with newer and older versions of the Netpbm libraries.

file The file.

format The format code of the image. This is **PAM_FORMAT** unless the PAM image is really a view of a PBM, PGM, or PPM image. Then it's **PBM_FORMAT**, **RPBM_FORMAT**, etc.

There is an important quirk in the meaning of this member when you use the pam structure to write an image: Only the type portion of it is meaningful. A Netpbm format code conveys two pieces of information: The format type (PBM, PGM, PPM, or PAM) and the plainness (plain PBM vs raw PBM, etc.). But when writing, **libnetpbm** ignores the plainness part and instead takes the plainness from the **plainformat** member. So **PBM_FORMAT** and **RPBM_FORMAT** are identical when writing.

This quirk exists for historical purposes; it's necessary for consistency with the older functions such as **pnm_writepnmrow()** whose *format* and *forceplain* arguments are analogous.

Before Netpbm 10.32 (February 2006), **libnetpbm** did not ignore the plainness. This caused many programs to behave poorly, producing plain format output when they should, for backward compatibility at the very least, produce raw format output.

A common way to use this member is to copy it and the **plainformat** member from a pam for an input image to a pam for an output image. When you do that, your output image will be raw format regardless of whether your input image was plain or raw, and this is the conventional behavior of Netpbm programs.

plainformat This is a boolean value (0 = false, 1 = true), meaningful only when writing an image file. It means to write in the plain (text) version of the format indicated by **format** as oppposed to the raw (binary) version. Note that the format code in **format** would appear to completely specify the format, making **plainformat** redundant. But see the description of **format** for why that isn't true.

Until Netpbm 10.32 (Februrary 2006), this was defined a little differently. The **format** member did in fact completely identify the format and **plainformat** was redundant and existed as a separate member only for computational speed. But this was inconsistent with the older **libnetpbm** interface (e.g. **pnm_writepnm()**, and it made it difficult to write backward compatible programs. Before Netpbm 10.32, it affected reading as well as writing.

libnetpbm image reading functions set this field to false, for your convenience in building an output image pam from an input image pam.

height The height of the image in rows.

width The width of the image in number of columns (tuples per row).

depth The depth of the image (degree of or number of samples in each tuple).

maxval The maxval of the image. See definitions in pam.

bytes_per_sample The number of bytes used to represent each sample in the image file. See the format definition in pam. This is entirely redundant with **maxval**. It exists as a separate member for computational speed.

tuple_type The tuple type of the image. See definitions in pam. Netpbm does not define any values for this except the following, which are used for a PAM image which is really a view of a PBM, PGM, or PPM image: **PAM_PBM_TUPLETYPE**, **PAM_PGM_TUPLETYPE**, **PAM_PPM_TUPLETYPE**.

allocation_depth The number of samples for which memory is allocated for any tuple associated with this PAM structure. This must be at least as great as