

# libpnmio user manual

<b>Title</b>	libpnmio (I/O PNM library)
<b>Author</b>	Nikolaos Kavvadias 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020
<b>Contact</b>	<a href="mailto:nikolaos.kavvadias@gmail.com">nikolaos.kavvadias@gmail.com</a>
<b>Website</b>	<a href="http://www.nkavvadias.com">http://www.nkavvadias.com</a>
<b>Release Date</b>	10 March 2016
<b>Version</b>	1.2.6
<b>Rev. history</b>	
<b>v1.2.6</b>	2016-03-10 Add sftbyvec.c, related changes.
<b>v1.2.5</b>	2014-10-28 Fixed bugs in PFM format I/O.
<b>v1.2.4</b>	2014-10-28 Added names to prototype parameters.
<b>v1.2.3</b>	2014-10-28 Documentation update; add <code>get_pnm_type</code> and reference to the IrfanView image viewer.
<b>v1.2.2</b>	2014-10-26 <ul style="list-style-type: none"><li>• New test application, <code>rnwimg</code> for reading and writing PBM, PGM, PPM and PFM images.</li><li>• Numerous bug fixes to <code>read_p*m_header</code> routines.</li><li>• Added <code>get_pnm_type</code>.</li><li>• Removed <code>squares.binary.pgm</code> (all-black image).</li><li>• Removed <code>feep.ascii.ppm</code> (incomplete image).</li><li>• PFM reading still needs work.</li></ul>
<b>v1.2.1</b>	2014-10-25 Added sample PBM, PGM, PPM and PFM images for testing.

<b>v1.2.0</b>	2014-10-25 Added support for reading and writing PFM (Portable Float Map) image data files.
<b>v1.1.1</b>	2014-10-01 Date notation change.
<b>v1.1.0</b>	2014-09-23 Moved AUTHORS, LICENSE, README, VERSION to top-level.
<b>v1.0.1</b>	2014-06-14 Changed README to README.rst.
<b>v1.0.0</b>	2014-02-20 First public release.

## 1. Introduction

The `libpnmio` library provides an implementation and API for reading and writing [PNM](#) (some times termed as Portable AnyMap) images. The [PNM](#) convention is collectively used to address [PBM](#) (Portable Bitmap), [PGM](#) (Portable Greymap) and [PPM](#) (Portable Pixmap) images.

The current version of `libpnmio` supports the ASCII variation of the PNM formats, however, it will be extended in order to support the corresponding binary formats.

The library is accompanied by the following test applications:

- `randimg`: produces PBM/PGM/PPM image files filled with random data
- `doset`: generates a color illustration of the Mandelbrot set
- `rnwimg`: reads and writes PBM/PGM/PPM/PFM images for testing the library
- `sftbyvec`: reads an input ASCII PGM image, shifts its contents by a given vector and then writes it back.

Since version 1.2.0, support for the Portable Float Map format ([PFM](#)) has been added.

Additional information on the PFM format can be found at the [PFM page by Paul Bourke](#).

Reference documentation for LIBPNMIO can be found in the `/doc` subdirectory in plain text, HTML and PDF form.

## 2. File listing

The `libpnmio` distribution includes the following files:

<code>/libpnmio</code>	Top-level directory
<code>AUTHORS</code>	List of authors.
<code>LICENSE</code>	License agreement (modified BSD license).
<code>README.rst</code>	This file.

README.html	HTML version of README.rst.
README.pdf	PDF version of README.rst.
VERSION	Current version of the LIBPNMIO distribution.
rst2docs.sh	Shell script for generating the documentation using docutils.
/bin	Executables directory (initially empty)
/images	Image data (PBM, PGM, PPM, PFM) for testing
*.pbm, *.pgm, *.ppm, *.pfm	Sample images.
/lib	Compiled static library directory
/src	Source code directory
Makefile	Makefile for compiling the library and generating the executables.
doset.c	Generates a color visualization of the Mandelbrot set.
pnmio.c	Implementation of the <code>libpnmio</code> library in C.
pnmio.h	Header file (interface) of the <code>libpnmio</code> library.
randimg.c	Random PBM/PGM/PPM/PFM image generator.
rnwimg.c	Reads and writes PBM/PGM/PPM/PFM images for exercising the <code>libpnmio</code> API.
sftbyvec.c	Read an input ASCII PGM image, shift its contents by a given vector and then writes it back
/test	Test script directory
run-doset.sh	Bash script for running the Mandelbrot set example.
run-randimg.sh	Bash script for running the random image generator.
run-rnwimg.sh	Bash script for running the read-and-write API tests.
run-sftbyvec.sh	Bash script for running the shift-by-vector tests.

The original sources for the images included in the `/libpnmio/images` directory are the following:

- *prague, squares, lena32, fruit, blocks*
  - <http://graphics.stanford.edu/~jowens/223b/examples.html>
- *letter\_j, feep, ppmex255.ascii, ppmex1*
  - [http://en.wikipedia.org/wiki/Netpbm\\_format](http://en.wikipedia.org/wiki/Netpbm_format)
- *haus*
  - <http://goo.gl/DBbPpF>
- *ppmex255.binary*
  - <http://wiki.multimedia.cx/index.php>

### 3. API description

This section summarizes the intended functionality of the functions supported by the `libpnmio` application programming interface.

#### 3.1 `get_pnm_type`

```
int get_pnm_type(FILE *f);
```

Read the header contents of a PBM/PGM/PPM/PFM file up to the point of extracting its type. Valid types for a PNM image are as follows:

- `PBM_ASCII = 1`
- `PGM_ASCII = 2`
- `PPM_ASCII = 3`
- `PBM_BINARY = 4`
- `PGM_BINARY = 5`
- `PPM_BINARY = 6`
- `PAM = 7` (unimplemented)
- `PFM_RGB = 16`
- `PFM_GREYSCALE = 17`

The result (`pnm_type`) is returned.

#### 3.2 `read_pbm_header`

```
void read_pbm_header(FILE *f, int *img_xdim, int  
*img_ydim, int is_ascii);
```

Read the header contents of a PBM (portable bit map) file. A PBM image file follows the format:

```
P1  
<X> <Y>  
<I1> <I2> ... <IMAX>
```

A binary PBM image file uses P4 instead of P1 and the data values are represented in binary. Comment lines start with `#`. `< >` denote integer values (in decimal). For the PBM format, they can take only the 0 and 1 values. `img_xdim` and `img_ydim` correspond to X and Y, respectively. If `is_ascii` is 1, an ASCII PBM file is assumed; otherwise a binary PBM file is.

### 3.3 read\_pgm\_header

```
read_pgm_header(FILE *f, int *img_xdim, int *img_ydim,  
int *img_colors, int is_ascii);
```

Read the header contents of a PGM (portable grey map) file. A PGM image file follows the format:

```
P2  
<X> <Y>  
<levels>  
<I1> <I2> ... <IMAX>
```

A binary PGM image file uses P5 instead of P2 and the data values are represented in binary. Comment lines start with #. < > denote integer values (in decimal). `img_xdim`, `img_ydim`, and `img_colors` correspond to X, Y and levels, respectively. If `is_ascii` is 1, an ASCII PGM file is assumed; otherwise a binary PGM file is.

### 3.4 read\_ppm\_header

```
void read_ppm_header(FILE *f, int *img_xdim, int  
*img_ydim, int *img_colors, int is_ascii);
```

Read the header contents of a PPM (portable pix map) file. A PPM image file follows the format:

```
P3  
<X> <Y>  
<levels>  
<R1> <G1> <B1> ... <RMAX> <GMAX> <BMAX>
```

A binary PPM image file uses P6 instead of P3 and the data values are represented in binary. Comment lines start with #. < > denote integer values (in decimal). `img_xdim`, `img_ydim`, and `img_colors` correspond to X, Y and levels, respectively. Each color component, R, G, and B can take any value from 0 to levels. If `is_ascii` is 1, an ASCII PPM file is assumed; otherwise a binary PPM file is.

### 3.5 read\_pfm\_header

```
void read_pfm_header(FILE *f, int *img_xdim, int  
*img_ydim, int *img_type, int *endianess);
```

Read the header contents of a PFM (portable float map) file. A PFM image file follows the format:

```
[PF|Pf]  
<X> <Y>  
(endianess)  
{R1}{G1}{B1} ... {RMAX}{GMAX}{BMAX}
```

A PFM image file has its data values represented in binary. Comment lines start with #. < > denote integer values (in decimal). ( ) denote floating-point values (in decimal). { } denote floating-point values (coded in binary). `img_xdim` and `img_ydim` correspond to X and Y, respectively. If `img_type` is equal to 1, the PFM image encodes RGB (color) information, otherwise if it is equal to 0, it stores greyscale information. If `endianess` is negative (-1), the binary data are encoded in little-endian ordering, otherwise if `endianess` is positive (+1), the data follow big-endian ordering.

### 3.6 read\_pbm\_data

```
void read_pgm_data(FILE *f, int *img_in, int is_ascii);
```

Read the data contents of a PBM (portable bit map) file. `img_in` denotes an array of integer values representing image data. If `is_ascii` is 1, an ASCII PBM file is assumed; otherwise a binary PBM file is.

### 3.7 read\_pgm\_data

```
void read_pgm_data(FILE *f, int *img_in, int is_ascii);
```

Read the data contents of a PGM (portable grey map) file. `img_in` denotes an array of integer values representing image data. If `is_ascii` is 1, an ASCII PGM file is assumed; otherwise a binary PGM file is.

### 3.8 read\_ppm\_data

```
void read_ppm_data(FILE *f, int *img_in, int is_ascii);
```

Read the data contents of a PPM (portable pix map) file. `img_in` denotes an array of integer values representing image data. If `is_ascii` is 1, an ASCII PPM file is assumed; otherwise a binary PPM file is.

### 3.9 read\_pfm\_data

```
void read_ppm_data(FILE *f, float *img_in, int img_type,  
int endianess);
```

Read the data contents of a PFM (portable float map) file. `img_in` denotes an array of floating-point (float) values representing image data. If `img_type` is 1, color/RGB image data are assumed; otherwise (0) the image data are in greyscale. A negative `endianess` indicates little-endian ordering and positive one, big-endian.

### 3.10 write\_pbm\_file

```
void write_pbm_file(FILE *f, int *img_out, char  
*img_out_fname,  
int x_size, int y_size, int x_scale_val, int y_scale_val,  
int linevals, int is_ascii);
```

Write the contents of a PBM (portable bit map) file. Data stored in array `img_out` are written to file `f`. This file is assumed to be already opened under the name `img_out_fname`. The image data represent an image of size `x_size` by `y_size`. x-axis and y-axis scaling factors can be defined by `x_scale_val` and `y_scale_val`. `linevals` determines the emission of newline characters for easier reading of the PBM file data. If `is_ascii` is 1, an ASCII PBM file is assumed; otherwise a binary PBM file is.

### 3.11 write\_pgm\_file

```
void write_pgm_file(FILE *f, int *img_out, char
*img_out_fname,
int x_size, int y_size, int x_scale_val, int y_scale_val,
int img_colors,
int linevals, int is_ascii);
```

Write the contents of a PGM (portable grey map) file. Data stored in array `img_out` are written to file `f`. This file is assumed to be already opened under the name `img_out_fname`. The image data represent an image of size `x_size` by `y_size`. x-axis and y-axis scaling factors can be defined by `x_scale_val` and `y_scale_val`. `img_colors` determines the levels (0 to levels) for the common color component. `linevals` determines the emission of newline characters for easier reading of the PGM file data. If `is_ascii` is 1, an ASCII PGM file is assumed; otherwise a binary PGM file is.

### 3.12 write\_ppm\_file

```
void write_ppm_file(FILE *f, int *img_out, char
*img_out_fname,
int x_size, int y_size, int x_scale_val, int y_scale_val,
int img_colors, int is_ascii);
```

Write the contents of a PPM (portable pix map) file. Data stored in array `img_out` are written to file `f`. This file is assumed to be already opened under the name `img_out_fname`. The image data represent an image of size `x_size` by `y_size`. x-axis and y-axis scaling factors can be defined by `x_scale_val` and `y_scale_val`. `img_colors` determines the levels (0 to levels) for the common color component. Each R-G-B triplet is printed to a separate line. If `is_ascii` is 1, an ASCII PPM file is assumed; otherwise a binary PPM file is.

### 3.13 write\_pfm\_file

```
void write_pfm_file(FILE *f, float *img_out, char
*img_out_fname,
int x_size, int y_size, int img_type, int endianness);
```

Write the contents of a PFM (portable float map) file. Data stored in array `img_out` are written to file `f`. This file is assumed to be already opened under the name `img_out_fname`. The image data represent an image of size `x_size` by `y_size`. x-axis and y-axis scaling factors can be defined by `x_scale_val` and `y_scale_val`. If `img_type` is equal to 1, the PFM image encodes RGB (color) information, otherwise if it is equal

to 0, it stores greyscale information. If `endianess` is negative (-1), the binary data are encoded in little-endian ordering, otherwise if `endianess` is positive (+1), the data follow big-endian ordering.

## 4. Build and setup

In order to produce the static library, change directory to `/src` and run the Makefile as follows:

```
$ make clean ; make
```

This will produce the static library `libpnmio.a` and copy it to the `/lib` subdirectory of the distribution. The executable files for the reference applications will also be generated and copied to the `/bin` subdirectory.

## 5. Run tests

Two sample scripts are provided in the `/test` subdirectory. Change directory to `/test` and run the scripts as follows:

```
$ cd test
$ ./run-doset.sh
$ ./run-randimg.sh
$ ./run-rnwimg.sh
$ ./run-sftbyvec.sh
```

PBM, PGM and PPM files can be directly visualized by using freeware image viewers such as [XnView](#), [IrfanView](#) (non-commercial use only) and [Imagine](#). The informal/non-standardized PFM format was introduced by [Paul Debevec](#). A PFM viewer (`HDRView`) can be found here: <http://web.archive.org/web/20060614160328/http://www.debevec.org/FiatLux/hdrview/>.

## 6. Prerequisites

- Standard UNIX-based tools (tested with gcc-4.6.2 and gcc-4.8.1 on MinGW/x64).
  - make
  - bash (shell)

For this reason, MinGW (<http://www.mingw.org>) or Cygwin (<http://sources.redhat.com/cygwin>) are suggested, since POSIX emulation environments of sufficient completeness.