
Linux Standard Base (LSB) —
Part 1-6:
Graphics and Gtk3 specification

Noyau de base normalisé Linux (LSB) —

Partie 1-6: Spécification graphique et Gtk3





COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2021

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by the Linux Foundation as Linux Standard Base (LSB): Graphics and gtk3 specification and drafted in accordance with its editorial rules. It was assigned to Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*, and adopted by National Bodies.

This first edition of ISO/IEC TS 23360-1-6 cancels and replaces ISO/IEC 23360-1:2006, which has been technically revised.

This document is based on “The GNU Free Documentation License, version 1.1”. The license is available at <https://www.gnu.org/licenses/old-licenses/fdl-1.1.html>

A list of all parts in the ISO/IEC 23660 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Contents

Foreword	iii
Introduction	v
I Introductory Elements	1
1 Scope	2
2 Normative References.....	3
3 Requirements	4
3.1 TUM Libraries	4
3.2 Gtk3 Libraries	4
4 Terms and Definitions.....	5
5 Documentation Conventions.....	7
II PNG15 library	8
6 Libraries	9
6.1 Interfaces for libpng15	9
6.2 Interface Definitions for libpng15	1 3
III GTK+ Stack Libraries	7 8
7 Libraries	7 9
7.1 Introduction.....	7 9
7.2 Interfaces for libgdk-3.....	7 9
7.3 Data Definitions for libgdk-3.....	9 1
7.4 Interfaces for libgtk-3.....	1 7 1
7.5 Data Definitions for libgtk-3.....	2 4 5
Annex A Alphabetical Listing of Interfaces by Library	4 4 6
A.1 libpng15.....	4 4 6
A.2 libgdk-3.....	4 5 0
A.3 libgtk-3.....	4 5 9

Introduction

The Trial Use Specification describes components which may or may not be present on an otherwise conforming system. The purpose is to indicate that these components are on a Standards Track, that is, they are intended to become part of the LSB Specification in a future edition.

This document should be used in conjunction with the documents it references. Information referenced in this way is as much a part of this document as is the information explicitly included here.

I Introductory Elements

1 Scope

The Trial Use Specification defines components which are not required parts of the LSB Specification.

2 Normative References

The specifications listed below are referenced in whole or in part by the Trial Use Specification. Such references may be normative or informative; a reference to specification shall only be considered normative if it is explicitly cited as such. The Trial Use Specification may make normative references to a portion of these specifications (that is, to define a specific function or group of functions); in such cases, only the explicitly referenced portion of the specification is to be considered normative.

Table 2-1 Informative References

Name	Title	URL
Gdk 3.6.4 Reference Manual	Gdk 3.6.4 Reference Manual	http://developer.gnome.org/gdk3/3.6
Gtk 3.6.4 Reference Manual	Gtk 3.6.4 Reference Manual	http://developer.gnome.org/gtk3/3.6
ISO C (1999)	ISO/IEC 9899:1999 - Programming Languages -- C	

3 Requirements

3.1 TUM Libraries

The libraries listed in Table 3-1 shall be available on a Linux Standard Base system which implements the Trial Use TUM module, with the specified runtime names. This list may be supplemented or amended by the architecture-specific specification.

Table 3-1 Standard Library Names

Library	Runtime Name
libpng15	libpng15.so.15

These libraries will be in an implementation-defined directory which the dynamic linker shall search by default.

3.2 Gtk3 Libraries

The libraries listed in Table 3-2 shall be available on a Linux Standard Base system which implements the Trial Use Toolkit_Gtk3 module, with the specified runtime names. This list may be supplemented or amended by the architecture-specific specification.

Table 3-2 Standard Library Names

Library	Runtime Name
libgdk-3	libgdk-3.so.0
libgtk-3	libgtk-3.so.0

These libraries will be in an implementation-defined directory which the dynamic linker shall search by default.

4 Terms and Definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 2382, ISO 80000-2, and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- ISO Online browsing platform: available at <https://www.iso.org/obp>
- IEC Electropedia: available at <http://www.electropedia.org/>

4.1

archLSB

Some LSB specification documents have both a generic, architecture-neutral part and an architecture-specific part. The latter describes elements whose definitions may be unique to a particular processor architecture. The term archLSB may be used in the generic part to refer to the corresponding section of the architecture-specific part.

4.2

Binary Standard, ABI

The total set of interfaces that are available to be used in the compiled binary code of a conforming application, including the run-time details such as calling conventions, binary format, C++ name mangling, etc.

4.3

Implementation-defined

Describes a value or behavior that is not defined by this document but is selected by an implementor. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence of the value or behavior. An application that relies on such a value or behavior cannot be assured to be portable across conforming implementations. The implementor shall document such a value or behavior so that it can be used correctly by an application.

4.4

Shell Script

A file that is read by an interpreter (e.g., awk). The first line of the shell script includes a reference to its interpreter binary.

4.5

Source Standard, API

The total set of interfaces that are available to be used in the source code of a conforming application. Due to translations, the Binary Standard and the Source Standard may contain some different interfaces.

4.6**Undefined**

Describes the nature of a value or behavior not defined by this document which results from use of an invalid program construct or invalid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

4.7**Unspecified**

Describes the nature of a value or behavior not specified by this document which results from use of a valid program construct or valid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

In addition, for the portions of this specification which build on IEEE Std 1003.1-2001, the definitions given in *IEEE Std 1003.1-2001, Base Definitions, Chapter 3* apply.

5 Documentation Conventions

Throughout this document, the following typographic conventions are used:

`function()`

the name of a function

command

the name of a command or utility

CONSTANT

a constant value

parameter

a parameter

variable

a variable

Throughout this specification, several tables of interfaces are presented. Each entry in these tables has the following format:

name

the name of the interface

(symver)

An optional symbol version identifier, if required.

[*refno*]

A reference number indexing the table of referenced specifications that follows this table.

For example,

forkpty(GLIBC_2.0) [SUSv4]

refers to the interface named `forkpty()` with symbol version `GLIBC_2.0` that is defined in the reference indicated by the tag `SUSv4`.

Note: For symbols with versions which differ between architectures, the symbol versions are defined in the architecture specific parts of of this module specification only. In the generic part, they will appear without symbol versions.

II PNG15 library

6 Libraries

6.1 Interfaces for libpng15

Table 6-1 defines the library name and shared object name for the libpng15 library

Table 6-1 libpng15 Definition

Library:	libpng15
SONAME:	libpng15.so.15

The behavior of the interfaces in this library is specified by the following specifications:

[LSB] This Specification

6.1.1 libpng15 interfaces

6.1.1.1 Interfaces for libpng15 interfaces

An LSB conforming implementation shall provide the generic functions for libpng15 interfaces specified in Table 6-2, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-2 libpng15 - libpng15 interfaces Function Interfaces

png_access_version_number(PNG15_0) [LSB]	png_benign_error(PNG15_0) [LSB]	png_build_grayscale_palette(PNG15_0) [LSB]
png_calloc(PNG15_0) [LSB]	png_chunk_benign_error(PNG15_0) [LSB]	png_chunk_error(PNG15_0) [LSB]
png_chunk_warning(PNG15_0) [LSB]	png_convert_from_struct_tm(PNG15_0) [LSB]	png_convert_from_time_t(PNG15_0) [LSB]
png_convert_to_rfc1123(PNG15_0) [LSB]	png_create_info_struct(PNG15_0) [LSB]	png_create_read_struct(PNG15_0) [LSB]
png_create_read_struct_2(PNG15_0) [LSB]	png_create_write_struct(PNG15_0) [LSB]	png_create_write_struct_2(PNG15_0) [LSB]
png_data_freer(PNG15_0) [LSB]	png_destroy_info_struct(PNG15_0) [LSB]	png_destroy_read_struct(PNG15_0) [LSB]
png_destroy_write_struct(PNG15_0) [LSB]	png_error(PNG15_0) [LSB]	png_free(PNG15_0) [LSB]
png_free_data(PNG15_0) [LSB]	png_free_default(PNG15_0) [LSB]	png_get_IHDR(PNG15_0) [LSB]
png_get_PLTE(PNG15_0) [LSB]	png_get_bKGD(PNG15_0) [LSB]	png_get_bit_depth(PNG15_0) [LSB]
png_get_cHRM(PNG15_0) [LSB]	png_get_cHRM_XYZ(PNG15_0) [LSB]	png_get_cHRM_XYZ_fixed(PNG15_0) [LSB]
png_get_cHRM_fixed(PNG15_0) [LSB]	png_get_channels(PNG15_0) [LSB]	png_get_chunk_cache_max(PNG15_0) [LSB]

png_get_chunk_malloc_max(PNG15_0) [LSB]	png_get_color_type(PNG15_0) [LSB]	png_get_compression_buffer_size(PNG15_0) [LSB]
png_get_compression_type(PNG15_0) [LSB]	png_get_copyright(PNG15_0) [LSB]	png_get_current_pass_number(PNG15_0) [LSB]
png_get_current_row_number(PNG15_0) [LSB]	png_get_error_ptr(PNG15_0) [LSB]	png_get_filter_type(PNG15_0) [LSB]
png_get_gAMA(PNG15_0) [LSB]	png_get_gAMA_fixed(PNG15_0) [LSB]	png_get_hIST(PNG15_0) [LSB]
png_get_header_version(PNG15_0) [LSB]	png_get_header_version(PNG15_0) [LSB]	png_get_iCCP(PNG15_0) [LSB]
png_get_image_height(PNG15_0) [LSB]	png_get_image_width(PNG15_0) [LSB]	png_get_int_32(PNG15_0) [LSB]
png_get_interlace_type(PNG15_0) [LSB]	png_get_io_chunk_name(PNG15_0) [LSB]	png_get_io_chunk_type(PNG15_0) [LSB]
png_get_io_ptr(PNG15_0) [LSB]	png_get_io_state(PNG15_0) [LSB]	png_get_libpng_version(PNG15_0) [LSB]
png_get_mem_ptr(PNG15_0) [LSB]	png_get_offs(PNG15_0) [LSB]	png_get_pCAL(PNG15_0) [LSB]
png_get_pHYs(PNG15_0) [LSB]	png_get_pHYs_dpi(PNG15_0) [LSB]	png_get_pixel_aspect_ratio(PNG15_0) [LSB]
png_get_pixel_aspect_ratio_fixed(PNG15_0) [LSB]	png_get_pixels_per_inch(PNG15_0) [LSB]	png_get_pixels_per_meter(PNG15_0) [LSB]
png_get_progressive_ptr(PNG15_0) [LSB]	png_get_rgb_to_gray_status(PNG15_0) [LSB]	png_get_rowbytes(PNG15_0) [LSB]
png_get_rows(PNG15_0) [LSB]	png_get_sBIT(PNG15_0) [LSB]	png_get_sCAL(PNG15_0) [LSB]
png_get_sCAL_fixed(PNG15_0) [LSB]	png_get_sCAL_s(PNG15_0) [LSB]	png_get_sPLT(PNG15_0) [LSB]
png_get_sRGB(PNG15_0) [LSB]	png_get_signature(PNG15_0) [LSB]	png_get_tIME(PNG15_0) [LSB]
png_get_tRNS(PNG15_0) [LSB]	png_get_text(PNG15_0) [LSB]	png_get_uint_16(PNG15_0) [LSB]
png_get_uint_31(PNG15_0) [LSB]	png_get_uint_32(PNG15_0) [LSB]	png_get_unknown_chunks(PNG15_0) [LSB]
png_get_user_chunk_ptr(PNG15_0) [LSB]	png_get_user_height_max(PNG15_0) [LSB]	png_get_user_transform_ptr(PNG15_0) [LSB]
png_get_user_width_max(PNG15_0) [LSB]	png_get_valid(PNG15_0) [LSB]	png_get_x_offset_inches(PNG15_0) [LSB]
png_get_x_offset_inches_fixed(PNG15_0) [LSB]	png_get_x_offset_microns(PNG15_0) [LSB]	png_get_x_offset_pixels(PNG15_0) [LSB]

png_get_x_pixels_per_inch(PNG15_0) [LSB]	png_get_x_pixels_per_meter(PNG15_0) [LSB]	png_get_y_offset_inches(PNG15_0) [LSB]
png_get_y_offset_inches_fixed(PNG15_0) [LSB]	png_get_y_offset_microns(PNG15_0) [LSB]	png_get_y_offset_pixels(PNG15_0) [LSB]
png_get_y_pixels_per_inch(PNG15_0) [LSB]	png_get_y_pixels_per_meter(PNG15_0) [LSB]	png_handle_as_unknown(PNG15_0) [LSB]
png_info_init_3(PNG15_0) [LSB]	png_init_io(PNG15_0) [LSB]	png_longjmp(PNG15_0) [LSB]
png_malloc(PNG15_0) [LSB]	png_malloc_default(PNG15_0) [LSB]	png_malloc_warn(PNG15_0) [LSB]
png_permit_mng_features(PNG15_0) [LSB]	png_process_data(PNG15_0) [LSB]	png_process_data_pause(PNG15_0) [LSB]
png_process_data_skip(PNG15_0) [LSB]	png_progressive_combine_row(PNG15_0) [LSB]	png_read_end(PNG15_0) [LSB]
png_read_image(PNG15_0) [LSB]	png_read_info(PNG15_0) [LSB]	png_read_png(PNG15_0) [LSB]
png_read_row(PNG15_0) [LSB]	png_read_rows(PNG15_0) [LSB]	png_read_update_info(PNG15_0) [LSB]
png_reset_zstream(PNG15_0) [LSB]	png_save_int_32(PNG15_0) [LSB]	png_save_uint_16(PNG15_0) [LSB]
png_save_uint_32(PNG15_0) [LSB]	png_set_IHDR(PNG15_0) [LSB]	png_set_PLTE(PNG15_0) [LSB]
png_set_add_alpha(PNG15_0) [LSB]	png_set_alpha_mode(PNG15_0) [LSB]	png_set_alpha_mode_fixed(PNG15_0) [LSB]
png_set_bKGD(PNG15_0) [LSB]	png_set_background(PNG15_0) [LSB]	png_set_background_fixed(PNG15_0) [LSB]
png_set_benign_errors(PNG15_0) [LSB]	png_set_bgr(PNG15_0) [LSB]	png_set_cHRM(PNG15_0) [LSB]
png_set_cHRM_XYZ(PNG15_0) [LSB]	png_set_cHRM_XYZ_fixed(PNG15_0) [LSB]	png_set_cHRM_fixed(PNG15_0) [LSB]
png_set_check_for_invalid_index(PNG15_0) [LSB]	png_set_chunk_cache_max(PNG15_0) [LSB]	png_set_chunk_malloc_max(PNG15_0) [LSB]
png_set_compression_buffer_size(PNG15_0) [LSB]	png_set_compression_level(PNG15_0) [LSB]	png_set_compression_mem_level(PNG15_0) [LSB]
png_set_compression_method(PNG15_0) [LSB]	png_set_compression_strategy(PNG15_0) [LSB]	png_set_compression_window_bits(PNG15_0) [LSB]
png_set_crc_action(PNG15_0) [LSB]	png_set_error_fn(PNG15_0) [LSB]	png_set_expand(PNG15_0) [LSB]

png_set_expand_16(PNG15_0) [LSB]	png_set_expand_gray_1_2_4_to_8(PNG15_0) [LSB]	png_set_filler(PNG15_0) [LSB]
png_set_filter(PNG15_0) [LSB]	png_set_filter_heuristics(PNG15_0) [LSB]	png_set_filter_heuristics_fixed(PNG15_0) [LSB]
png_set_flush(PNG15_0) [LSB]	png_set_gAMA(PNG15_0) [LSB]	png_set_gAMA_fixed(PNG15_0) [LSB]
png_set_gamma(PNG15_0) [LSB]	png_set_gamma_fixed(PNG15_0) [LSB]	png_set_gray_to_rgb(PNG15_0) [LSB]
png_set_hIST(PNG15_0) [LSB]	png_set_iCCP(PNG15_0) [LSB]	png_set_interlace_handling(PNG15_0) [LSB]
png_set_invalid(PNG15_0) [LSB]	png_set_invert_alpha(PNG15_0) [LSB]	png_set_invert_mono(PNG15_0) [LSB]
png_set_keep_unknown_chunks(PNG15_0) [LSB]	png_set_longjmp_fn(PNG15_0) [LSB]	png_set_mem_fn(PNG15_0) [LSB]
png_set_oFFs(PNG15_0) [LSB]	png_set_pCAL(PNG15_0) [LSB]	png_set_pHYs(PNG15_0) [LSB]
png_set_packing(PNG15_0) [LSB]	png_set_packswap(PNG15_0) [LSB]	png_set_palette_to_rgb(PNG15_0) [LSB]
png_set_progressive_read_fn(PNG15_0) [LSB]	png_set_quantize(PNG15_0) [LSB]	png_set_read_fn(PNG15_0) [LSB]
png_set_read_status_fn(PNG15_0) [LSB]	png_set_read_user_chunk_fn(PNG15_0) [LSB]	png_set_read_user_transform_fn(PNG15_0) [LSB]
png_set_rgb_to_gray(PNG15_0) [LSB]	png_set_rgb_to_gray_fixed(PNG15_0) [LSB]	png_set_rows(PNG15_0) [LSB]
png_set_sBIT(PNG15_0) [LSB]	png_set_sCAL(PNG15_0) [LSB]	png_set_sCAL_fixed(PNG15_0) [LSB]
png_set_sCAL_s(PNG15_0) [LSB]	png_set_sPLT(PNG15_0) [LSB]	png_set_sRGB(PNG15_0) [LSB]
png_set_sRGB_gAMA_and_chRM(PNG15_0) [LSB]	png_set_scale_16(PNG15_0) [LSB]	png_set_shift(PNG15_0) [LSB]
png_set_sig_bytes(PNG15_0) [LSB]	png_set_strip_16(PNG15_0) [LSB]	png_set_strip_alpha(PNG15_0) [LSB]
png_set_swap(PNG15_0) [LSB]	png_set_swap_alpha(PNG15_0) [LSB]	png_set_tIME(PNG15_0) [LSB]
png_set_tRNS(PNG15_0) [LSB]	png_set_tRNS_to_alpha(PNG15_0) [LSB]	png_set_text(PNG15_0) [LSB]
png_set_text_compression_level(PNG15_0) [LSB]	png_set_text_compression_mem_level(PNG15_0) [LSB]	png_set_text_compression_method(PNG15_0) [LSB]

png_set_text_compression_strategy(PNG15_0) [LSB]	png_set_text_compression_window_bits(PNG15_0) [LSB]	png_set_unknown_chunk_location(PNG15_0) [LSB]
png_set_unknown_chunks(PNG15_0) [LSB]	png_set_user_limits(PNG15_0) [LSB]	png_set_user_transform_info(PNG15_0) [LSB]
png_set_write_fn(PNG15_0) [LSB]	png_set_write_status_fn(PNG15_0) [LSB]	png_set_write_user_transform_fn(PNG15_0) [LSB]
png_sig_cmp(PNG15_0) [LSB]	png_start_read_image(PNG15_0) [LSB]	png_warning(PNG15_0) [LSB]
png_write_chunk(PNG15_0) [LSB]	png_write_chunk_data(PNG15_0) [LSB]	png_write_chunk_end(PNG15_0) [LSB]
png_write_chunk_start(PNG15_0) [LSB]	png_write_end(PNG15_0) [LSB]	png_write_flush(PNG15_0) [LSB]
png_write_image(PNG15_0) [LSB]	png_write_info(PNG15_0) [LSB]	png_write_info_before_PLTE(PNG15_0) [LSB]
png_write_png(PNG15_0) [LSB]	png_write_row(PNG15_0) [LSB]	png_write_rows(PNG15_0) [LSB]
png_write_sig(PNG15_0) [LSB]		

An LSB conforming implementation shall provide the generic deprecated functions for libpng15 interfaces specified in Table 6-3, with the full mandatory functionality as described in the referenced underlying specification.

Note: These interfaces are deprecated, and applications should avoid using them. These interfaces may be withdrawn in future releases of this specification.

Table 6-3 libpng15 - libpng15 interfaces Deprecated Function Interfaces

png_get_io_chunk_name(PNG15_0) [LSB]		
--------------------------------------	--	--

6.2 Interface Definitions for libpng15

The interfaces defined on the following pages are included in libpng15 and are defined by this specification. Unless otherwise noted, these interfaces shall be included in the source standard.

Other interfaces listed in Section 6.1 shall behave as described in the referenced base document.

png_access_version_number

Name

png_access_version_number — return version of the run-time library

Synopsis

```
#include <png.h>
png_uint_32 png_access_version_number(void);
```

Description

png_access_version_number() returns version of the libpng15 library available at run-time.

Return Value

png_access_version_number() returns integer constructed from the major version, minor version with leading zero and leading number with leading zero. For example, the version number for version 1.2.8 is 10208.

png_convert_from_struct_tm

Name

png_convert_from_struct_tm — convert struct tm to png_time

Synopsis

```
#include <png.h>
void png_convert_from_struct_tm(png_timep ptime, struct tm * ttime);
```

Description

This interface shall convert from the struct tm time format to the png_time format.

The parameter *ttime* shall specify the struct tm value to convert to png_time format.

The output parameter *ptime* shall contain the converted value.

png_convert_from_time_t

Name

png_convert_from_time_t — convert time_t to png_time

Synopsis

```
#include <png.h>
void png_convert_from_time_t(png_timep ptime, time_t ttime);
```

Description

This interface shall convert from the time_t time format to the png_time format.

The parameter *ttime* shall specify the time_t value to convert to png_time format.

The output parameter *ptime* shall contain the converted value.

png_create_info_struct

Name

png_create_info_struct — allocate and initialize a png_info structure

Synopsis

```
#include <png.h>
png_infop png_create_info_struct(png_structp png_ptr);
```

Description

png_create_info_struct() shall allocate and initialize a png_info structure.

Return Value

Returns the pointer to png_info structure. Returns NULL if it fails to create the structure.

Errors

png_create_info_struct() shall return NULL if the allocation fails. The application should check for the return value.

png_create_read_struct

Name

`png_create_read_struct` — allocate and initialize a `png_struct` structure for reading PNG file

Synopsis

```
#include <png.h>
png_structp png_create_read_struct(png_const_charp user_png_ver,
png_voidp error_ptr, png_error_ptr error_fn, png_error_ptr warn_fn);
```

Description

`png_create_read_struct()` shall allocate and initialize a `png_struct` structure. The function shall return NULL if the allocation fails. The application should check for the return value. For handling errors and warnings, the application can pass desired error handling routines as arguments to `png_create_read_struct()`. Otherwise, the default error handling uses `stderr` and `longjmp`. The error handling routine must NOT return to the calling routine.

user_png_ver

version string of the library. Must be `PNG_LIBPNG_VER_STRING`

error_ptr

user defined struct for error functions.

error_fn

user defined function for printing errors and aborting.

warn_fn

user defined function for warnings.

Return Value

Returns the pointer to `png_struct` structure. Returns NULL if it fails to create the structure.

Errors

`png_create_read_struct()` shall return NULL if the allocation fails. The application should check for the return value.

png_create_read_struct_2

Name

png_create_read_struct_2 — register custom read function

Synopsis

```
#include <png.h>
png_structp png_create_read_struct_2(png_const_charp user_png_ver,
png_voidp error_ptr, png_error_ptr error_fn, png_error_ptr warn_fn,
png_voidp mem_ptr, png_malloc_ptr malloc_fn, png_free_ptr free_fn);
```

Description

This interface shall register a custom read function and allocate and initialize a `png_struct` structure. For handling errors and warnings, as well as allocating and deallocating memory, the application can pass functions as arguments. Otherwise, the default error handling functions `stderr` and `longjmp()` will be used. The error handling routine must not return to the calling routine.

The parameter `user_png_ver` shall specify the version string of the library, which must be `PNG_LIBPNG_VER_STRING`.

The parameter `error_ptr` shall specify a user-defined structure for error functions.

The parameter `error_fn` shall specify an optional user-defined function for printing errors and aborting.

The parameter `warn_fn` shall specify an optional user-defined function for warnings.

The parameter `mem_ptr` shall specify the memory to allocate.

The parameter `malloc_fn` shall specify an optional user-defined memory allocation function.

The parameter `free_fn` shall specify an optional user-defined memory deallocation function.

Application Usage (informative)

You should define `PNG_USER_MEM_SUPPORTED` before you call `png_create_read_struct2()`.

png_create_write_struct

Name

`png_create_write_struct` — allocate and initialize a `png_struct` structure for writing PNG file

Synopsis

```
#include <png.h>
png_structp png_create_write_struct(png_const_charp user_png_ver,
png_voidp error_ptr, png_error_ptr error_fn, png_error_ptr warn_fn);
```

Description

`png_create_write_struct()` shall allocate and initialize a `png_struct` structure. The function shall return NULL if the allocation fails. The application should check for the return value.

user_png_ver

version string of the library. Must be PNG_LIBPNG_VER_STRING

error_ptr

user defined struct for error functions.

error_fn

user defined function for printing errors and aborting.

warn_fn

user defined function for warnings.

Return Value

Returns the pointer to `png_struct` structure. Returns NULL if it fails to create the structure.

png_create_write_struct_2

Name

png_create_write_struct_2 — register custom write function

Synopsis

```
#include <png.h>
png_structp png_create_write_struct_2(png_const_charp user_png_ver,
png_voidp error_ptr, png_error_ptr error_fn, png_error_ptr warn_fn,
png_voidp mem_ptr, png_malloc_ptr malloc_fn, png_free_ptr free_fn);
```

Description

This interface shall register a custom write function and allocate and initialize a `png_struct` structure. For handling errors and warnings, as well as allocating and deallocating memory, the application can pass functions as arguments. Otherwise, the default error handling functions `stderr` and `longjmp()` will be used. The error handling routine must not return to the calling routine.

The parameter `user_png_ver` shall specify the version string of the library, which must be `PNG_LIBPNG_VER_STRING`.

The parameter `error_ptr` shall specify a user-defined structure for error functions.

The parameter `error_fn` shall specify an optional user-defined function for printing errors and aborting.

The parameter `warn_fn` shall specify an optional user-defined function for warnings.

The parameter `mem_ptr` shall specify the memory to allocate.

The parameter `malloc_fn` shall specify an optional user-defined memory allocation function.

The parameter `free_fn` shall specify an optional user-defined memory deallocation function.

Return Value

On success, returns a pointer to the `png_struct` that was created.

On failure, returns `NULL`.

png_data_freer

Name

png data freer – change the default behavior for freeing data

Synopsis

```
#include <png.h>
void png_data_freer(png_structp png_ptr, png_infop info_ptr, int freer,
png_uint_32 mask);
```

Description

This interface shall change the default behavior for freeing allocated data, from only freeing data allocated internally by libpng, to either freeing user-allocated data or to not freeing any data at all.

This interface can enable deallocation of user data allocated with `png_malloc()` or `png_zalloc()` and passed to libpng with one of the `png_set_*` interfaces. Users can call it both before and after passing the data. Calling it after reading PNG data but before passing the data controls whether the user or the interface is responsible for the existing data. Calling it after passing the data controls whether the user or the interface should deallocate the data.

If the user becomes responsible for data allocated by libpng, `png_free()` must be called to free it. If libpng becomes responsible for user-allocated data, the data must only have been allocated with `png_malloc()` or `png_zalloc()`.

The parameter *png_ptr* shall specify the PNG file.

The parameter *info_ptr* shall specify the PNG info structure.

The parameter *freer* shall specify one of these constants:
 PNG_DESTROY_WILL_FREE_DATA, PNG_SET_WILL_FREE_DATA,
 PNG_USER_WILL_FREE_DATA.

The parameter `mask` shall specify the data to free, as described under `png_free_data()`.

Application Usage (informative)

A `row_pointers` array allocated in a single block must not be freed with `png_set_rows()` or `png_read_destroy()`, because they would attempt to free the elements of the array as well.

Do not free `text_ptr` with `libpng` if some of its members were allocated separately, because this will actually only free `text_ptr.key`. If responsibility for freeing `text_ptr` moves from `libpng` to the user, the user must not free the members separately.

png_destroy_info_struct

Name

png_destroy_info_struct — free memory in PNG info structure

Synopsis

```
#include <png.h>
void png_destroy_info_struct(png_structp png_ptr, png_infopp info_ptr_ptr);
```

Description

This interface shall free any memory in a single PNG info structure.

The parameter *png_ptr* shall specify the PNG file.

The parameter *info_ptr_ptr* shall point to a pointer to the png_info_struct to destroy.

png_destroy_read_struct

Name

png_destroy_read_struct — free the memory associated with read png_struct

Synopsis

```
#include <png.h>
void png_destroy_read_struct(png_structpp png_ptr_ptr, png_infopp info_ptr_ptr, png_infopp end_info_ptr_ptr);
```

Description

png_destroy_read_struct() frees the memory associated with the read png_struct struct that holds information from the given PNG file, the associated png_info struct for holding the image information and png_info struct for holding the information at end of the given PNG file.

png_destroy_write_struct

Name

png_destroy_write_struct — free the memory associated with write png_struct

Synopsis

```
#include <png.h>
void png_destroy_write_struct(png_structpp png_ptr_ptr, png_infopp info_ptr_ptr);
```

Description

png_destroy_write_struct() frees the memory associated with the write png_struct struct that holds information for writing the PNG file and the associated png_info struct for holding the image information.

png_error

Name

png_error — default function to handle fatal errors

Synopsis

```
#include <png.h>
void png_error(png_structp png_ptr, png_const_charp error_message);
```

Description

png_error() is the default error handling function for fatal errors. The default error handling functionality may be changed by using png_set_error_fn() to replace the error function at run-time.

Return Value

Errors

png_free

Name

png_free — free a pointer allocated by png_malloc()

Synopsis

```
#include <png.h>
void png_free(png_structp png_ptr, png_voidp ptr);
```

Description

png_free() shall free memory pointed to by "ptr" previously allocated by png_malloc().

png_free_data

Name

png_free_data — free internally allocated data

Synopsis

```
#include <png.h>
void    png_free_data(png_structp    png_ptr,    png_infop    info_ptr,
png_uint_32 free_me, int num);
```

Description

This interface shall free data that was internally allocated by libpng.

The parameter *png_ptr* shall specify the PNG file.

The parameter *info_ptr* shall specify the PNG info structure containing members to free that are pointing to memory allocated by libpng.

The parameter *mask* shall specify the data to free with a mask consisting of the logical OR of at least one of the following constants: PNG_FREE_HIST, PNG_FREE_ICCP, PNG_FREE_PCAL, PNG_FREE_PLTE, PNG_FREE_ROWS, PNG_FREE_SCAL, PNG_FREE_SPLT, PNG_FREE_TEXT, PNG_FREE_TRNS, PNG_FREE_UNKN. (To specify all constants, use PNG_FREE_ALL.)

The parameter *num* shall specify the sequence number of the item to free. The value -1 specifies all items.

png_get_IHDR

Name

png_get_IHDR — get PNG_IHDR chunk information from png_info structure

Synopsis

```
#include <png.h>
png_uint_32 png_get_IHDR(png_structp png_ptr, png_infop info_ptr,
png_uint_32 * width, png_uint_32 * height, int * bit_depth, int *
color_type, int * interlace_method, int * compression_method, int *
filter_method);
```

Description

png_get_IHDR() gets PNG_IHDR chunk type information from png_info structure.

width

holds the width of the image in pixels up to 2³¹.

height

holds the height of the image in pixels up to 2³¹.

bit_depth

holds the bit depth of one of the image channels. Valid values are 1, 2, 4, 8, 16 and also depend on the color_type.

color_type

describes which color/alpha channels are present. Supported color types shall include:

PNG_COLOR_TYPE_GRAY (bit depths 1, 2, 4, 8, 16)
 PNG_COLOR_TYPE_GRAY_ALPHA (bit depths 8, 16)
 PNG_COLOR_TYPE_PALETTE (bit depths 1, 2, 4, 8)
 PNG_COLOR_TYPE_RGB (bit depths 8, 16)
 PNG_COLOR_TYPE_RGB_ALPHA (bit depths 8, 16)
 PNG_COLOR_MASK_PALETTE
 PNG_COLOR_MASK_COLOR
 PNG_COLOR_MASK_ALPHA

filter_method

holds the filter method. If this argument is NULL, the filter method will not be retrieved. Valid values after retrieval are

PNG_FILTER_TYPE_BASE
 PNG_INTRAPIXEL_DIFFERENCING

compression_method

holds the compression method. If this argument is NULL, the compression method will not be retrieved. Valid values after retrieval are
 PNG_COMPRESSION_TYPE_BASE

interlace_method

holds the interlace method. If this argument is NULL, the interlacing method will not be retrieved. Valid values after retrieval are

PNG_INTERLACE_NONE
PNG_INTERLACE_ADAM7

Return Value

On success, `png_get_HDR()` shall return 1. Otherwise, `png_get_IHDR()` shall return 0.

png_get_PLTE

Name

`png_get_PLTE` — get image palette information from `png_info` structure

Synopsis

```
#include <png.h>
png_uint_32 png_get_PLTE(png_structp png_ptr, png_infop info_ptr,
    png_colorp * palette, int * num_palette);
```

Description

`png_get_PLTE()` gets palette information from `png_info` structure. "palette" holds an array of color values with "num_palette" entries.

Return Value

On successful retrieval of palette information, `png_get_PLTE()` shall return PNG_INFO_PLTE. Otherwise, `png_get_PLTE()` shall return 0.

png_get_bKGD

Name

`png_get_bKGD` — get background color for given image

Synopsis

```
#include <png.h>
png_uint_32 png_get_bKGD(png_structp png_ptr, png_infop info_ptr,
    png_color_16p * background);
```

Description

`png_get_bKGD()` shall return the background color to "background" if the validity flag for background is set.

Return Value

On success, `png_get_bKGD()` shall return PNG_INFO_bKGD. Otherwise, `png_get_bKGD()` shall return 0.

png_get_bit_depth

Name

png_get_bit_depth — return image bit_depth

Synopsis

```
#include <png.h>
png_byte png_get_bit_depth(png_structp png_ptr, png_infop info_ptr);
```

Description

Returns the image bit_depth.

Return Value

Returns 0 if png_ptr or info_ptr is NULL, bit_depth otherwise.

png_get_cHRM

Name

png_get_cHRM — get CIE chromacities and referenced white point for given image

Synopsis

```
#include <png.h>
png_uint_32 png_get_cHRM(png_structp png_ptr, png_infop info_ptr,
double * white_x, double * white_y, double * red_x, double * red_y,
double * green_x, double * green_y, double * blue_x, double * blue_y);
```

Description

png_get_cHRM() shall return the CIE x,y chromaticities of the red, green and blue display primaries used in the image and the referenced white point from the cHRM chunk in the image.

Return Value

On success, png_get_cHRM() shall return PNG_INFO_cHRM. Otherwise, the function shall return 0.

png_get_channels

Name

png_get_channels — get number of color channels in image

Synopsis

```
#include <png.h>
png_byte png_get_channels(png_structp png_ptr, png_infop info_ptr);
```

Description

png_get_channels() shall return the number of data channels per pixel for the color type of the image. The number of channels shall range from 1-4 depending on the color type as given below.

- 1 - PNG_COLOR_TYPE_GRAY or PNG_COLOR_TYPE_PALETTE
- 2 - PNG_COLOR_TYPE_GRAY_ALPHA
- 3 - PNG_COLOR_TYPE_RGB
- 4 - PNG_COLOR_TYPE_RGB_ALPHA or PNG_COLOR_TYPE_RGB+filler
byte

Return Value

On success, `png_get_channels()` shall return the number of channels ranging from 1-4. Otherwise, `png_get_channels` shall return 0.

png_get_color_type

Name

`png_get_color_type` — return image color type

Synopsis

```
#include <png.h>
png_byte png_get_color_type(png_structp png_ptr, png_info_ptr info_ptr);
```

Description

Returns the image color type.

Return Value

Returns 0 if `png_ptr` or `info_ptr` is NULL, `color_type` otherwise.

png_get_error_ptr

Name

`png_get_error_ptr` — return `error_ptr` for user-defined functions

Synopsis

```
#include <png.h>
png_voidp png_get_error_ptr(png_structp png_ptr);
```

Description

`png_get_error_ptr()` returns the `error_ptr` associated with user-defined functions.

Return Value

Returns `error_ptr`

png_get_gAMA

Name

png_get_gAMA — get the gamma value for given image

Synopsis

```
#include <png.h>
png_uint_32 png_get_gAMA(png_structp png_ptr, png_infop info_ptr,
double * file_gamma);
```

Description

Returns the gamma value of an image to "file_gamma" if the gAMA chunk information is valid for the image.

Return Value

PNG_INFO_gAMA - if png_ptr, info_ptr, file_gamma are not NULL and gAMA chunk information is valid 0 - otherwise.

png_get_hIST

Name

png_get_hIST — get the histogram for given image

Synopsis

```
#include <png.h>
png_uint_32 png_get_hIST(png_structp png_ptr, png_infop info_ptr,
png_uint_16p * hist);
```

Description

Returns the histogram of an image to *hist if the hIST chunk information is valid for the image.

Return Value

PNG_INFO_hIST - if png_ptr, info_ptr, hist are not NULL and hIST chunk information is valid 0 - otherwise.

png_get_header_ver

Name

png_get_header_ver — get version information for libpng header files

Synopsis

```
#include <png.h>
png_charp png_get_header_ver(png_structp png_ptr);
```

Description

This interface shall return the version of the header files used when building libpng as a short string in the format "1.0.0" through "99.99.99zz".

png_get_iCCP

Name

png_get_iCCP — get the embedded ICC profile data for given image

Synopsis

```
#include <png.h>
png_uint_32 png_get_iCCP(png_structp png_ptr, png_infop info_ptr,
png_charpp name, int * compression_type, png_charpp profile,
png_uint_32 * proflen);
```

Description

png_get_iCCP() shall return the embedded ICC profile data in iCCP chunk. "name" shall contain the profile name, *compression_type shall contain the compression type, profile shall contain the International Color Consortium color profile data and *proflen shall contain the length of the profile data in bytes. *compression_type must always be set to PNG_COMPRESSION_TYPE_BASE.

Return Value

On success, png_get_iCCP() shall return PNG_INFO_iCCP. Otherwise, the function shall return 0.

png_get_image_height

Name

png_get_image_height — return image height

Synopsis

```
#include <png.h>
png_uint_32 png_get_image_height(png_structp png_ptr, png_infop info_ptr);
```

Description

png_get_image_height() returns the image height in pixels.

Return Value

Returns 0 if png_ptr or info_ptr is NULL, image_height otherwise.

png_get_image_width

Name

png_get_image_width — return image width

Synopsis

```
#include <png.h>
png_uint_32 png_get_image_width(png_structp png_ptr, png_info_ptr info_ptr);
```

Description

png_get_image_width() returns the image width in pixels.

Return Value

Returns 0 if png_ptr or info_ptr is NULL, image_width otherwise.

png_get_interlace_type

Name

png_get_interlace_type — returns interlace method

Synopsis

```
#include <png.h>
png_byte png_get_interlace_type(png_structp png_ptr, png_info_ptr info_ptr);
```

Description

png_get_interlace_type() returns the interlace method used for the image.

Return Value

Valid values are PNG_INTERLACE_NONE, PNG_INTERLACE_ADAM7. Returns 0 if png_ptr or info_ptr is NULL.

png_get_io_ptr

Name

png_get_io_ptr — return pointer for user-defined I/O

Synopsis

```
#include <png.h>
png_voidp png_get_io_ptr(png_structp png_ptr);
```

Description

Returns the pointer associated with user-defined input-output functions.

png_get_libpng_ver**Name**

`png_get_libpng_ver` — get the library version string

Synopsis

```
#include <png.h>
png_charp png_get_libpng_ver(png_structp png_ptr);
```

Description

`png_get_libpng_ver()` shall return the library version as a short string in the format "1.0.0" through "99.99.99zz".

png_get_oFFs**Name**

`png_get_oFFs` — get screen offsets for the given image

Synopsis

```
#include <png.h>
png_uint_32 png_get_oFFs(png_structp png_ptr, png_info_ptr info_ptr,
png_int_32 * x_offset, png_int_32 * y_offset, int * unit_type);
```

Description

`png_get_oFFs()` shall read the positive offset from the left edge of the screen from `info_ptr` to `x_offset` and the positive offset from the top edge of the screen to `y_offset`. The unit type shall be returned in `unit_type`, which can take the following values

PNG_OFFSET_PIXEL
PNG_OFFSET_MICROMETER

Return Value

On success, `png_get_oFFs()` shall return PNG_INFO_oFFs. Otherwise, `png_get_oFFs()` shall return 0.

png_get_pHYs

Name

png_get_pHYs — get the physical resolution for given image

Synopsis

```
#include <png.h>
png_uint_32 png_get_pHYs(png_structp png_ptr, png_infop info_ptr,
    png_uint_32 * res_x, png_uint_32 * res_y, int * unit_type);
```

Description

png_get_pHYs() shall return the physical pixel resolution of the image and the unit of resolution. Upon success, *res_x shall contain the horizontal resolution and *res_y shall contain the vertical resolution in pixels per unit. *unit_type will be set to PNG_RESOLUTION_METER if the resolution is expressed in pixels per meter. Otherwise *unit_type will be PNG_RESOLUTION_UNKNOWN.

Return Value

PNG_INFO_pHYs - on success 0 - otherwise.

png_get_progressive_ptr

Name

png_get_progressive_ptr — return pointer to user-defined push read functions

Synopsis

```
#include <png.h>
png_voidp png_get_progressive_ptr(png_structp png_ptr);
```

Description

Returns the pointer to user-defined structure containing information about the callback functions.

png_get_rowbytes

Name

png_get_rowbytes — Return number of bytes for a row

Synopsis

```
#include <png.h>
png_uint_32 png_get_rowbytes(png_structp png_ptr, png_infop info_ptr);
```

Description

png_get_rowbytes() returns the number of bytes needed to hold a transformed row of an image.

Return Value

Returns 0 if png_ptr or info_ptr is NULL, number of bytes otherwise.

png_get_rows

Name

png_get_rows — retrieve image data from png_info structure

Synopsis

```
#include <png.h>
png_bytepp png_get_rows(png_structp png_ptr, png_infop info_ptr);
```

Description

png_get_rows() shall retrieve rows of image data from the info_ptr structure in an array of pointers to the pixel data for each row.

Return Value

On success, png_get_rows() shall return an array of pointers to the pixel data for each row of the image.

png_get_sBIT

Name

png_get_sBIT — get number of significant bits for each color channel

Synopsis

```
#include <png.h>
png_uint_32 png_get_sBIT(png_structp png_ptr, png_infop info_ptr,
png_color_8p * sig_bit);
```

Description

png_get_sBIT() shall return the number of significant bit for each of the gray, red, blue and green color channels.

Return Value

On success, png_get_sBIT() shall return PNG_INFO_sBIT. Otherwise, png_get_sBIT() shall return 0.

png_get_sRGB

Name

png_get_sRGB — get the rendering intent for given image

Synopsis

```
#include <png.h>
png_uint_32 png_get_sRGB(png_structp png_ptr, png_infop info_ptr, int
* srgb_intent);
```

Description

png_get_sRGB() shall return the rendering intent of an image to *srgb_intent if the sRGB chunk information is valid for the image.

Return Value

PNG_INFO_sRGB - if png_ptr, info_ptr, srgb_intent are not NULL and sRGB chunk information is valid 0 - otherwise.

png_get_tIME

Name

png_get_tIME — get last modification time for the image

Synopsis

```
#include <png.h>
png_uint_32 png_get_tIME(png_structp png_ptr, png_infop info_ptr,
png_timep * mod_time);
```

Description

png_get_tIME() shall return the time of last modification of the image if the tIME information is valid for the image.

Return Value

PNG_INFO_tIME - if png_ptr, info_ptr, mod_time are not NULL and tIME information is valid 0 - otherwise.

png_get_tRNS

Name

png_get_tRNS — get transparency data for images

Synopsis

```
#include <png.h>
png_uint_32 png_get_tRNS(png_structp png_ptr, png_infop info_ptr,
    png_bytep * trans, int * num_trans, png_color_16p * trans_values);
```

Description

png_get_tRNS() shall obtain the transparency data for paletted images and image types that don't need a full alpha channel from info_ptr. For a paletted image, the function retrieves the transparency values stored in the same order as the palette colors, starting from index 0. For non-paletted images, the function retrieves the single color value which is treated as fully transparent. If the transparency information is valid, i.e. PNG_INFO_tRNS bit is set for info_ptr->valid: *trans shall be set to the transparency values for a paletted image. Values for the data shall be in range [0,255], ranging from fully transparent to fully opaque, respectively. *num_trans shall be set to the number of transparency values *trans_values shall be set to the single color value specified for non-paletted images.

Return Value

Returns PNG_INFO_tRNS on successful return, 0 otherwise.

png_get_text

Name

png_get_text — get comments information from png_info structure

Synopsis

```
#include <png.h>
png_uint_32 png_get_text(png_structp png_ptr, png_infop info_ptr,
    png_textp * text_ptr, int * num_text);
```

Description

png_get_text() returns the text chunk information from the PNG stream in the array pointed to by text_ptr. It also returns the number of text chunks in num_text. text_ptr is an array of structure png_text whose members include:

compression

type of compression used on text. Valid values are:

PNG_TEXT_COMPRESSION_NONE
PNG_TEXT_COMPRESSION_zTXt
PNG_ITXT_COMPRESSION_NONE
PNG_ITXT_COMPRESSION_zTXt

key

keyword for comment. Must contain 1-79 characters.

text

text comment for current keyword. May be empty.

text_length

length of text string after decompression. 0 for iTXt.

Return Value

Returns 0 if *png_ptr* or *info_ptr* is NULL, returns the number of text chunks otherwise.

png_get_unknown_chunks

Name

`png_get_unknown_chunks` — retrieve the unknown chunks from a PNG file

Synopsis

```
#include <png.h>
png_uint_32 png_get_unknown_chunks(png_structp png_ptr, png_infop
info_ptr, png_unknown_chunkpp unknowns);
```

Description

This interface shall retrieve the unknown chunks from a PNG file.

The parameter *png_ptr* shall specify the PNG file.

The parameter *info_ptr* shall specify the PNG info structure.

The parameter *unknowns* shall specify an array of `png_unknown_chunk` structures containing the unknown chunks. The position of a structure in the array shall correspond to the order in which `png_set_unknown_chunks()` inserted its chunk into the PNG file, or in which it was read.

The `png_unknown_chunkpp` structure shall contain the following members.

name

the name of the chunk

data

the data of the chunk

size

the size of the chunk's data

location

the position of the chunk in the PNG file

Return Value

On success, this interface shall return the number of unknown chunks obtained.

On failure, this interface shall return 0.

png_get_user_chunk_ptr

Name

png_get_user_chunk_ptr — get pointer to user chunk data

Synopsis

```
#include <png.h>
png_voidp png_get_user_chunk_ptr(png_structp png_ptr);
```

Description

This interface shall return the pointer to the user chunk data associated with the specified PNG file.

The parameter *png_ptr* shall specify the PNG file.

png_get_valid

Name

png_get_valid — determine if given chunk data is valid

Synopsis

```
#include <png.h>
png_uint_32 png_get_valid(png_structp png_ptr, png_infop info_ptr,
png_uint_32 flag);
```

Description

png_get_valid() shall obtain the validity of chunk data specified by the bits set in "flag". The following bits may be set in flag: PNG_INFO_gAMA PNG_INFO_sBIT PNG_INFO_cHRM PNG_INFO_PLTE PNG_INFO_tRNS PNG_INFO_bKGD PNG_INFO_hIST PNG_INFO_pHYs PNG_INFO_oFFs PNG_INFO_tIME PNG_INFO_pCAL PNG_INFO_sRGB PNG_INFO_iCCP PNG_INFO_sPLT PNG_INFO_sCAL PNG_INFO_IDAT

Return Value

On success, png_get_valid() shall return "flag" with the chunk bits set. Otherwise, png_get_valid() shall return 0.

png_get_x_offset_pixels

Name

png_get_x_offset_pixels — return x offset in pixels from oFFs chunk

Synopsis

```
#include <png.h>
png_int_32 png_get_x_offset_pixels(png_structp png_ptr, png_infop
info_ptr);
```

Description

png_get_x_offset_pixels() shall obtain x offset in pixels for the image from its PNG_oFFs chunk data stored in info_ptr, if the unit for offset is pixels.

Return Value

On success, png_get_x_offset_pixels() shall return x offset in pixels. Otherwise png_get_x_offset_pixels() shall return 0.

png_get_x_pixels_per_meter

Name

png_get_x_pixels_per_meter — return horizontal pixel density per meter

Synopsis

```
#include <png.h>
png_uint_32 png_get_x_pixels_per_meter(png_structp png_ptr, png_infop
info_ptr);
```

Description

png_get_x_pixels_per_meter() shall obtain the horizontal pixel density in pixels per meter from its PNG_pHYs chunk data stored in info_ptr, if the unit for resolution is pixels per meter.

Return Value

On success, png_get_x_pixels_per_meter() shall return horizontal pixel density in pixels per meter. Otherwise png_get_x_pixels_per_meter() shall return 0.

png_get_y_offset_pixels

Name

png_get_y_offset_pixels — return y offset in pixels from oFFs chunk

Synopsis

```
#include <png.h>
png_int_32 png_get_y_offset_pixels(png_structp png_ptr, png_infop
info_ptr);
```

Description

png_get_y_offset_pixels() shall obtain y offset in pixels for the image from its PNG_oFFs chunk data stored in info_ptr, if the unit for offset is pixels.

Return Value

On success, png_get_y_offset_pixels() shall return y offset in pixels. Otherwise png_get_y_offset_pixels() shall return 0.

png_get_y_pixels_per_meter

Name

png_get_y_pixels_per_meter — return vertical pixel density per meter

Synopsis

```
#include <png.h>
png_uint_32 png_get_y_pixels_per_meter(png_structp png_ptr, png_infop
info_ptr);
```

Description

png_get_y_pixels_per_meter() shall obtain the vertical pixel density in pixels per meter from its PNG_pHYs chunk data stored in info_ptr, if the unit for resolution is pixels per meter.

Return Value

On success, png_get_y_pixels_per_meter() shall return vertical pixel density in pixels per meter. Otherwise png_get_y_pixels_per_meter() shall return 0.

png_info_init_3

Name

png_info_init_3 — initialize an info structure (DEPRECATED)

Synopsis

```
#include <png.h>
void      png_info_init_3(png_infopp      info_ptr,      png_size_t
png_info_struct_size);
```

Description

This interface shall initialize a PNG info structure. This interface is deprecated.

The parameter *info_ptr* shall specify the PNG info structure to initialize.

The parameter *png_info_struct_size* shall specify the size of the new structure.

png_init_io

Name

png_init_io — initialize input/output for the PNG file

Synopsis

```
#include <png.h>
void png_init_io(png_structp png_ptr, png_FILE_p fp);
```

Description

Initialize the default input/output functions for the PNG file to standard C streams. To replace the default read and write functions, use png_set_read_fn() and png_set_write_fn() respectively.

Return Value

Errors

png_malloc

Name

png_malloc — allocate memory

Synopsis

```
#include <png.h>
png_voidp png_malloc(png_structp png_ptr, png_uint_32 size);
```

Description

png_malloc() shall return a pointer to allocated memory of the specified size.

Return Value

Pointer to the block of memory allocated.

Errors

Invokes error handling function if the system is out of memory and sets PNG_FLAG_MALLOC_NULL_MEM_OK in member flags of png_struct.

png_permit_mng_features

Name

`png_permit_mng_features` — enable MNG extensions for PNG image wrapped in MNG datastream

Synopsis

```
#include <png.h>
png_uint_32 png_permit_mng_features(png_structp png_ptr, png_uint_32
mng_features_permitted);
```

Description

This interface shall enable some MNG extensions for a PNG image wrapped in a MNG datastream.

The parameter `png_ptr` shall specify the PNG image.

The parameter `mng_features_permitted` shall specify the logical OR of the features to be enabled, possibly including `PNG_ALL_MNG_FEATURES`, `PNG_FLAG_MNG_EMPTY_PLTE`, and `PNG_FLAG_MNG_FILTER_64`.

Return Value

This interface shall return the logical AND of the parameter `mng_features_permitted` with the set of MNG features supported by the local version of libpng.

Application Usage (informational)

This interface may not read or write a standalone PNG file; the PNG datastream must be embedded in a MNG datastream with an 8-byte MNG signature and MEND and MHDR chunks.

png_process_data

Name

`png_process_data` — read PNG file progressively

Synopsis

```
#include <png.h>
void png_process_data(png_structp png_ptr, png_info_ptr info_ptr,
png_bytep buffer, png_size_t length);
```

Description

`png_process_data()` shall process data progressively from the PNG stream using callback functions set within `png_set_progressive_read_fn()`. The data is passed in "buffer" and length of data to be processed is specified by "length".

png_progressive_combine_row

Name

`png_progressive_combine_row` — combines current row data with processed row

Synopsis

```
#include <png.h>
void png_progressive_combine_row(png_structp png_ptr, png_bytep old_row,
png_bytep new_row);
```

Description

For non-NULL rows of interlaced images during progressive read, `png_progressive_combine_row()` shall combine the data for the current row with the previously processed row data. `png_progressive_combine_row()` shall return for NULL rows of interlaced images and memcpy rows for non-interlaced images.

png_read_end

Name

`png_read_end` — read the end of PNG file

Synopsis

```
#include <png.h>
void png_read_end(png_structp png_ptr, png_infop info_ptr);
```

Description

`png_read_end()` reads the end of a PNG file after reading the image data, including any comments or time information at the end of the file. The function shall not read past the end of the file.

png_read_image

Name

`png_read_image` — read the entire image into memory

Synopsis

```
#include <png.h>
void png_read_image(png_structp png_ptr, png_bytepp image);
```

Description

`png_read_image()` reads the entire image into memory at once. For each pass of an interlaced image, use `png_read_rows()` instead.