Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'inttypes.h.0p' command

$ man inttypes.h.0p

inttypes.h(0P) POSIX Programmer's Manual inttypes.h(0P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

inttypes.h ? fixed size integer types

SYNOPSIS

#include <inttypes.h>

DESCRIPTION

Some of the functionality described on this reference page extends the ISO C standard. Applications shall define the appropriate feature test macro (see the System Interfaces volume of POSIX.1?2017, Section 2.2, The Compilation Environment) to enable the visibility of these symbols in this header.

The <inttypes.h> header shall include the <stdint.h> header.

The <inttypes.h> header shall define at least the following types:

imaxdiv_t Structure type that is the type of the value returned by the imaxdiv() function.

wchar_t As described in <stddef.h>.

The <inttypes.h> header shall define the following macros. Each expands to a character string literal containing a conversion specifier, possi?

bly modified by a length modifier, suitable for use within the format argument of a formatted input/output function when converting the cor? responding integer type. These macros have the general form of PRI (character string literals for the fprintf() and fwprintf() family of functions) or SCN (character string literals for the fscanf() and fws? canf() family of functions), followed by the conversion specifier, fol? lowed by a name corresponding to a similar type name in <stdint.h>. In these names, N represents the width of the type as described in <stdint.h>. For example, PRIdFAST32 can be used in a format string to print the value of an integer of type int_fast32_t.

The fprintf() macros for signed integers are:

PRIdN PRIdLEASTN PRIdFASTN PRIdMAX PRIdPTR

PRIiN PRIiLEASTN PRIiFASTN PRIiMAX PRIiPTR

The fprintf() macros for unsigned integers are:

PRIoN PRIoLEASTN PRIoFASTN PRIoMAX PRIoPTR

PRIuN PRIuLEASTN PRIuFASTN PRIuMAX PRIuPTR

PRIxN PRIxLEASTN PRIxFASTN PRIxMAX PRIxPTR

PRIXN PRIXLEASTN PRIXFASTN PRIXMAX PRIXPTR

The fscanf() macros for signed integers are:

SCNdN SCNdLEASTN SCNdFASTN SCNdMAX SCNdPTR

SCNiN SCNiLEASTN SCNiFASTN SCNiMAX SCNiPTR

The fscanf() macros for unsigned integers are:

SCNoN SCNoLEASTN SCNoFASTN SCNoMAX SCNoPTR

SCNuN SCNuLEASTN SCNuFASTN SCNuMAX SCNuPTR

SCNxN SCNxLEASTN SCNxFASTN SCNxMAX SCNxPTR

For each type that the implementation provides in <stdint.h>, the cor? responding fprintf() and fwprintf() macros shall be defined and the corresponding fscanf() and fwscanf() macros shall be defined unless the implementation does not have a suitable modifier for the type.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
intmax_t  imaxabs(intmax_t);
imaxdiv_t imaxdiv(intmax_t, intmax_t);
```

```
intmax_t  strtoimax(const char *restrict, char **restrict, int);
uintmax_t strtoumax(const char *restrict, char **restrict, int);
intmax_t  wcstoimax(const wchar_t *restrict, wchar_t **restrict, int);
uintmax_t wcstoumax(const wchar_t *restrict, wchar_t **restrict, int);
```

The following sections are informative.

EXAMPLES

```
#include <inttypes.h>
#include <wchar.h>
int main(void)
{
    uintmax_t i = UINTMAX_MAX; // This type always exists.
    wprintf(L"The largest integer value is %020"
        PRIxMAX "\n", i);
    return 0;
}
```

APPLICATION USAGE

The purpose of <inttypes.h> is to provide a set of integer types whose definitions are consistent across machines and independent of operating systems and other implementation idiosyncrasies. It defines, through typedef, integer types of various sizes. Implementations are free to typedef them as ISO C standard integer types or extensions that they support. Consistent use of this header will greatly increase the porta? bility of applications across platforms.

RATIONALE

The ISO/IEC 9899:1990 standard specified that the language should sup? port four signed and unsigned integer data types?char, short, int, and long?but placed very little requirement on their size other than that int and short be at least 16 bits and long be at least as long as int and not smaller than 32 bits. For 16-bit systems, most implementations assigned 8, 16, 16, and 32 bits to char, short, int, and long, respec? tively. For 32-bit systems, the common practice has been to assign 8, 16, 32, and 32 bits to these types. This difference in int size can create some problems for users who migrate from one system to another

which assigns different sizes to integer types, because the ISO C stan?
dard integer promotion rule can produce silent changes unexpectedly.
The need for defining an extended integer type increased with the in?
troduction of 64-bit systems.

FUTURE DIRECTIONS

Macro names beginning with PRI or SCN followed by any lowercase letter or 'X' may be added to the macros defined in the <inttypes.h> header.

SEE ALSO

<stddef.h>

The System Interfaces volume of POSIX.1?2017, Section 2.2, The Compila?
tion Environment, imaxabs(), imaxdiv(), strtoimax(), wcstoimax()

COPYRIGHT