NAME

ctype, isdigit, isxdigit, islower, isupper, isalpha, isalnum, isspace, iscntrl, ispunct, isprint, isgraph, isascii – character handling

SYNOPSIS

#include <ctype.h>

int isalpha(int *c***);**

int isupper(int *c***);**

int islower(int *c***);**

int isdigit(int *c***);**

int isxdigit(int *c***);**

int isalnum(int *c***);**

int isspace(int *c***);**

int ispunct(int *c***);**

int isprint(int *c***);**

int isgraph(int *c***);**

int iscntrl(int *c***);**

int isascii(int *c***);**

MT-LEVEL

MT-Safe with exceptions

DESCRIPTION

These macros classify character-coded integer values. Each is a predicate returning non-zero for true, zero for false. The behavior of these macros, except **isascii( )**, is affected by the current locale (see **setlocale**(3C)). To modify the behavior, change the **LC_TYPE** category in **setlocale( )**, that is, **setlocale(LC_CTYPE,** *newlocale***)**. In the **C** locale, or in a locale where character type information is not defined, characters are classified according to the rules of the US-ASCII 7-bit coded character set.

The macro **isascii( )** is defined on all integer values; the rest are defined only where the argument is an **int**, the value of which is representable as an **unsigned char**, or **EOF**, which is defined by the **stdio.h** header and represents end-of-file.

isalpha( )　tests for any character for which **isupper( )** or **islower( )** is true, or any character that is one of an implementation-defined set of characters for which none of **iscntrl( )**, **isdigit( )**, **ispunct( )**, or **isspace( )** is true. In the **C** locale, **isalpha( )** returns true only for the characters for which **isupper( )** or **islower( )** is true.

isupper( )　tests for any character that is an upper-case letter or is one of an implementation-defined set of characters for which none of **iscntrl( )**, **isdigit( )**, **ispunct( )**, **isspace( )**, or **islower( )** is true. In the **C** locale, **isupper( )** returns true only for the characters defined as upper-case ASCII characters.

islower( ) tests for any character that is a lower-case letter or is one of an implementation-defined set of characters for which none of **iscntrl( )**, **isdigit( )**, **ispunct( )**, **isspace( )**, or **isupper( )** is true. In the **C** locale, **islower( )** returns true only for the characters defined as lower-case ASCII characters.

isdigit( ) tests for any decimal-digit character.

isxdigit( ) tests for any hexadecimal-digit character (**[0–9]**, **[A–F]** or **[a–f]**).

isalnum( ) tests for any character for which **isalpha( )** or **isdigit( )** is true (letter or digit).

isspace( ) tests for any space, tab, carriage-return, newline, vertical-tab or form-feed (standard white-space characters) or for one of an implementation-defined set of characters for which **isalnum( )** is false. In the **C** locale, **isspace( )** returns true only for the standard white-space characters.

ispunct( ) tests for any printing character which is neither a space nor a character for which **isalnum( )** is true.

isprint( ) tests for any printing character, including space (**" "**).

isgraph( ) tests for any printing character, except space.

iscntrl( ) tests for any ‘‘control character’’ as defined by the character set.

isascii( ) tests for any ASCII character, code between **0** and **0177** inclusive.

All the character classification macros and the conversion functions and macros use a table lookup.

Functions exist for all the above-defined macros. To get the function form, the macro name must be undefined (for example, **#undef isdigit**).

RETURN VALUES

If the argument to any of the character handling macros is not in the domain of the function, the result is undefined.

FILES

/usr/lib/locale/*locale***/LC_CTYPE**

SEE ALSO

chrtbl(1M), **setlocale**(3C), **stdio**(3S), **ascii**(5), **environ**(5)

NOTES

isdigit( ), **isxdigit( )**, **islower( )**, **isupper( )**, **isalpha( )**, **isalnum( )**, **isspace( )**, **iscntrl( )**, **ispunct( )**, **isprint( )**, **isgraph( )** and **isascii( )** can be used safely in a multi-thread application, as long as **setlocale**(3C) is not being called to change the locale.