NAME

chrtbl, wchrtbl – generate character classification and conversion tables

SYNOPSIS

chrtbl [*filename*]

wchrtbl [*filename*]

DESCRIPTION

chrtbl creates character type and numeric layout files for single byte locales. **wchrtbl** does the same for multibyte locales. The two commands are links to each other.

Character classification tables contain information on character attributes, upper- to lowercase conversion, and codeset character width. The **LC_CTYPE** file is an array of bytes encoded so simple table lookups can determine character type or perform case mapping, using **ctype**(3C) or **wctype** (see **iswalpha**(3I)) library routines. Other routines can find the byte count and screen width of characters in supplementary code sets. The **LC_NUMERIC** file contains format information for numbers. The first byte specifies the decimal delimiter, and the second byte specifies the thousands separator.

Both commands read character classification and conversion information from *filename* and create three output files in the current directory. If no input file is given, these commands read from standard input. The example section below contains the source *filename* for ISO 8859-1. For multibyte locales this example needs to be extended.

First Output File

The first output file, **[w]ctype.c** is a C language source file, which application programs can use as needed. It contains a (257∗2)+7 byte array generated from processing *filename*. Review the contents of the C source to verify that the array is set up as planned. The first 257 bytes of the array are used for character classification. Symbols used for initializing these bytes represent character classifications defined in **<ctype.h>**; for example, **_L** means a character is lower case and **_S | _B** means the character is both a spacing character and a blank. The second 257 bytes of the array are used for character conversion. These bytes are initialized so that characters without conversion information are converted to themselves. If you provide conversion information, the first value of the pair is stored where the second one would normally be stored, and vice versa. For example, if you provide **<0x41 0x61>**, then **0x61** is stored where **0x41** would normally be stored, and **0x61** is stored where **0x41** would normally be stored. The last 7 bytes are used for character width information for up to three supplementary code sets.

Second Output File

The second output file is binary data containing the same information, but structured for efficient use by the **ctype**(3C) and **wctype** (see **iswalpha**(3I)) routines. The name of this output file is the value you assign to the keyword **LC_CTYPE** in *filename*. The superuser should install this file as **/usr/lib/locale/***locale***/LC_CTYPE/ctype**. It must be readable by user, group, and other; execute permission is not necessary. Application programs consult this file when the **LC_CTYPE** environment is set appropriately, upon calling **setlocale**(3C).

Third Output File

The third output file is binary data created only if numeric formatting information is specified. The name of this output file is the value you assign to the keyword **LC_NUMERIC** in *filename*. The superuser should install this file as

/usr/lib/locale/*locale***/LC_NUMERIC**. It must be readable by user, group, and other; execute permission is not necessary. Application programs consult this file when the **LC_NUMERIC** environment is set appropriately, upon calling **setlocale**(3C).

For supplementary codesets, there are three sets of tables. The first set contains three pointer arrays that point to supplementary codeset information tables. If supplementary codeset information is not specified, the contents of the pointers are zeros. The arrays are full of null pointers. The second set contains three supplementary codeset information tables, each specifying minimum and maximum code values to be classified and converted, and also pointers to character classification and conversion tables. If there is no corresponding table, the contents of the pointers are zeros. The third set contains character classification and conversion tables that contain the same information as the single byte table, except codes are represented as process codes and table size is variable. The characters used for initializing values of the character classification table represent character classifications defined in **<wctype.h>**; **_E1** through **_E8** are for international use and **_E9** through **_E24** are for language-dependent use.

filename Syntax

The syntax of *filename* provides for data file naming, assignment of characters to character classifications, upper- to lower-case mapping, byte and screen widths for up to three supplementary code sets, plus numeric formatting information. The keywords recognized by **[w]chrtbl** are:

LC_CTYPE	name of the first data file created by **[w]chrtbl**
isupper	character codes classified as upper-case letters
islower	character codes classified as lower-case letters
isdigit	character codes classified as numeric
isspace	character codes classified as spacing (delimiter) characters
ispunct	character codes classified as punctuation characters
iscntrl	character codes classified as control characters
isblank	character code for the blank (space) character
isxdigit	character codes classified as hexadecimal digits
ul	relationship between upper- and lower-case characters
cswidth	byte count and screen width information
LC_NUMERIC	name of the second data file created by **[w]chrtbl**
decimal_point	decimal delimiter, may be \NNN octal or \xNN hexadecimal
thousands_sep	thousands separator, may be \NNN octal or \xNN hexadecimal
LC_CTYPE1	begin definition of supplementary codeset 1
LC_CTYPE2	begin definition of supplementary codeset 2
LC_CTYPE3	begin definition of supplementary codeset 3

isphonogram(iswchar1)	character codes classified as phonograms in supplementary code sets
isideogram(iswchar2)	character codes classified as ideograms in supplementary code sets
isenglish(iswchar3)	character codes classified as English letters in supplementary code sets
isnumber(iswchar4)	character codes classified as numeric in supplementary code sets
isspecial(iswchar5)	character codes classified as special letters in supplementary code sets
iswchar6	character codes classified as other printable letters in supplementary code sets
iswchar7 - iswchar8	reserved for international use
iswchar9 - iswchar24	character codes classified as language-dependent letters/characters

Any lines with a sharp (**#**) in the first column are treated as comments and are ignored, as are blank lines.

To indicate character codes, use either hexadecimal or octal constants. For example, the letter **a** can be represented as **0x61** in hexadecimal or **0141** in octal. Constants may be separated by one or more spaces and/or tabs. Use a dash (–) to indicate a range of consecutive numbers. Zero or more spaces may separate the dash from its numbers. Use a backslash (\) for line continuation; only the newline is permitted after a backslash. Character codes are EUC values minus, if any, the escape character prefix.

The relationship between upper- and lower-case letters (**ul**) is expressed as ordered pairs of octal or hexadecimal constants: **<**_upper-case_character lower-case_character_**>**. One or more space characters may separate these two constants. Zero or more space may separate angle brackets (**< >**) from numbers.

The following is the format of an input specification for **cswidth:** _n1:s1,n2:s2,n3:s3_ where:

_n1_	byte width for supplementary code set 1, required
_s1_	screen width for supplementary code set 1
_n2_	byte width for supplementary code set 2
_s2_	screen width for supplementary code set 2
_n3_	byte width for supplementary code set 3
_s3_	screen width for supplementary code set 3

decimal_point and **thousands_sep** are specified by a single character, the delimiter.

EXAMPLES

Here is the input file used to create the **iso_8859_1** codeset definition table.

```
LC_CTYPE        LC_CTYPE
isupper         0x41 - 0x5a     0xc0 - 0xd6     0xd8 - 0xde
islower         0x61 - 0x7a     0xdf            0xe0 - 0xf6     0xf8 - 0xff
isdigit         0x30 - 0x39
isspace         0x20            0x09 - 0x0d     0xa0
ispunct         0x21 - 0x2f     0x3a - 0x40     0x5b - 0x60     0x7b - 0x7e \
                0xa1 - 0xbf     0xd7            0xf7
iscntrl         0x0 - 0x1f      0x7f
isblank         0x20            0xa0
isxdigit        0x30 - 0x39     0x61 - 0x66     0x41 - 0x46
ul              <0x41 0x61>     <0x42 0x62>     <0x43 0x63>     <0x44 0x64> \
                <0x45 0x65>     <0x46 0x66>     <0x47 0x67>     <0x48 0x68> \
                <0x49 0x69>     <0x4a 0x6a>     <0x4b 0x6b>     <0x4c 0x6c> \
                <0x4d 0x6d>     <0x4e 0x6e>     <0x4f 0x6f>     <0x50 0x70> \
                <0x51 0x71>     <0x52 0x72>     <0x53 0x73>     <0x54 0x74> \
                <0x55 0x75>     <0x56 0x76>     <0x57 0x77>     <0x58 0x78> \
                <0x59 0x79>     <0x5a 0x7a>     <0xc0 0xe0>     <0xc1 0xe1> \
                <0xc2 0xe2>     <0xc3 0xe3>     <0xc4 0xe4>     <0xc5 0xe5> \
                <0xc6 0xe6>     <0xc7 0xe7>     <0xc8 0xe8>     <0xc9 0xe9> \
                <0xca 0xea>     <0xcb 0xeb>     <0xcc 0xec>     <0xcd 0xed> \
                <0xce 0xee>     <0xcf 0xef>     <0xd0 0xf0>     <0xd1 0xf1> \
                <0xd2 0xf2>     <0xd3 0xf3>     <0xd4 0xf4>     <0xd5 0xf5> \
                <0xd6 0xf6>     <0xd8 0xf8>     <0xd9 0xf9>     <0xda 0xfa> \
                <0xdb 0xfb>     <0xdc 0xfc>     <0xdd 0xfd>     <0xde 0xfe>
cswidth         1:1,0:0,0:0
LC_NUMERIC      LC_NUMERIC
decimal_point   ","
thousands_sep   " "
#
LC_CTYPE1
isupper         0xc0 - 0xd6     0xd8 - 0xde
islower         0xdf            0xe0 - 0xf6     0xf8 - 0xff
isspace         0xa0
ispunct         0xa1 - 0xbf     0xd7            0xf7
isblank         0xa0
ul              <0xc0 0xe0>     <0xc1 0xe1> \
                <0xc2 0xe2>     <0xc3 0xe3>     <0xc4 0xe4>     <0xc5 0xe5> \
                <0xc6 0xe6>     <0xc7 0xe7>     <0xc8 0xe8>     <0xc9 0xe9> \
                <0xca 0xea>     <0xcb 0xeb>     <0xcc 0xec>     <0xcd 0xed> \
                <0xce 0xee>     <0xcf 0xef>     <0xd0 0xf0>     <0xd1 0xf1> \
                <0xd2 0xf2>     <0xd3 0xf3>     <0xd4 0xf4>     <0xd5 0xf5> \
                <0xd6 0xf6>     <0xd8 0xf8>     <0xd9 0xf9>     <0xda 0xfa> \
                <0xdb 0xfb>     <0xdc 0xfc>     <0xdd 0xfd>     <0xde 0xfe>
```

FILES

/usr/include/ctype.h	declarations used by character classification and conversion routines
/usr/include/wctype.h	declarations used by wide character classification and conversion routines
/usr/lib/locale/*locale***/LC_CTYPE/ctype**	data file containing character classification, conversion, and codeset width information
/usr/lib/locale/*locale***/LC_NUMERIC**	data file containing numeric layout information

SEE ALSO

ctype(3C), **setlocale**(3C), **iswalpha**(3I), **environ**(5)

NOTES

Do not change files under the C locale, as this could cause undefined or nonstandard behavior.