NAME

colltbl – create string collation routines

SYNOPSIS

colltbl [*filename*]

DESCRIPTION

The **colltbl** command reads locale specifications for collation order from *filename*, then creates a shared library composed of four functions: **strxfrm**(3C), **wsxfrm**(3I), **strcoll**(3C), and **wscoll**(3I). The last two transform their arguments and perform the comparison directly. If no input file is supplied, **colltbl** reads from standard input.

The name of the output file is the value you assign to the keyword **codeset** in *filename*. The superuser should install this file as **/usr/lib/locale/***locale***/LC_COLLATE/coll.so**. It must be readable and executable by user, group, and other. Application programs consult this file when the **LC_COLLATE** environment is set appropriately, after having called **setlocale**(3C).

The **colltbl** command can support languages whose collating sequence can be completely described by the following cases:

- Ordering of single characters within the codeset. For example, in English **B** is sorted after **A**, but before **C** and **D**.
- Equivalence class definition. A collection of characters is defined to have the same primary sort value. For example, in Finnish the letters French **V** and **W** compare equal. Both come after **U** but before **X**.
- Ordering of double characters in the collation sequence. For example, in Spanish **ch** is collated after **c**, and **l** is collated after **ll**.
- Ordering of one character as if it consists of two characters. For example, in German the "es-zet" **ß** is sorted as if it were **ss**. This is a special instance of the case below.
- Substitution of one character string with another character string. For example, spelled-out numbers, month and day names, and so forth, can be transformed so that they sort correctly.
- Null character mapping, so that certain characters in the codeset are ignored during collation. For example, if "**-**" were ignored during collation, then the strings **re-locate** and **relocate** would compare equal.
- Secondary ordering between characters. In the case where two characters are sorted together in the collation sequence, (that is, they have the same "primary" ordering), there is sometimes a secondary ordering that is used if two strings are identical except for characters that have the same primary ordering. For example, in French, the letters **e** and **è** have the same primary ordering but **e** comes before **è** in the secondary ordering. Thus the word **lever** would be ordered before **lèver**, but **lèver** would be sorted before **levitate**. (Note that if **e** came before **è** in the primary ordering, then **lèver** would be sorted after **levitate**.)

USAGE

The specification file consists of three types of statements:

1. **codeset** *filename*

   filename is the name of the output file to be created by **colltbl**.

2. **order is** *order_list*

   order_list is a list of symbols, separated by semicolons, that defines the collating sequence. The special symbol **...** is short-hand for symbols that are lexically sequential. For example,

   order is a;b;c;d;...;x;y;z

   specifies the list of lower_case letters. Of course, this could be further shortened to **a;...;z**. Note that symbols surrounding **...** must single character symbols; parentheses or braces are not allowed.

   A symbol can be up to two bytes in length and can be represented in any one of the following ways:

   - the symbol itself (for example, **a** for the lower-case letter **a**),
   - in octal representation (for example, **\141** or **0141** for the letter **a**), or
   - in hexadecimal representation (for example, **\x61** or **0x61** for the letter **a**).

   Any combination of these may be used as well.

   The backslash character, \ , is used for continuation. No characters are permitted after the backslash character.

   Symbols enclosed in parenthesis are assigned the same primary ordering but different secondary ordering. Symbols enclosed in curly brackets are assigned only the same primary ordering. For example,

   order is a;b;c;ch;d;(e;è);f;...;z;
   {1;...;9};A;...;Z

   In the above example, **e** and **è** are assigned the same primary ordering and different secondary ordering, digits 1 through 9 are assigned the same primary ordering and no secondary ordering. Only primary ordering is assigned to the remaining symbols. Notice how double letters can be specified in the collating sequence (letter **ch** comes between **c** and **d**).

   If a character is not included in the **order is** statement it is excluded from the ordering and will be ignored during sorting.

3. **substitute** *string* **with** *repl*

   The **substitute** statement substitutes the string *pattern* with the string *repl*. This can be used, for example, to provide rules to sort abbreviated month names numerically:

   substitute "Jan" with "01"
   substitute "Feb" with "02"
   ...
   substitute "Dec" with "12"

A simpler use of the **substitute** statement mentioned above is to substitute one character with two characters, as with the substitution of **ss** for **ß** in German.

Null character mapping can also be performed with **substitute**, as follows:

substitute "-" with ""

The **substitute** statement is optional. The **order is** and **codeset** statements are required.

Any lines in the specification file with a **#** in the first column are treated as comments and are ignored. Empty lines are also ignored.

EXAMPLES

The following example shows the collation specification required to support a hypothetical telephone book sorting sequence.

The sorting sequence is defined by the following rules:

- Upper and lower case letters must be sorted together, but upper case letters have precedence over lower case letters.
- All special characters and punctuation must be ignored.
- Digits must be sorted as their alphabetic counterparts (**0** as **zero**, **1** as **one**).
- The **CH**, **Ch**, **ch** combinations must be collated between **C** and **D**.
- **V** and **W**, **v** and **w** must be collated together.

The input specification file to **colltbl** should contain:

```
codeset     telephone
order is    (A;a);(B;b);(C;c);(CH;Ch;ch);(D;d);(E;e);(F;f);(G;g);\
            (H;h);(I;i);(J;j);(K;k);(L;l);(M;m);(N;n);(O;o);(P;p);\
            (Q;q);(R;r);(S;s);(T;t);(U;u);{V;W};{v;w};(X;x);(Y;y);(Z;z)
substitute "0" with "zero"
substitute "1" with "one"
substitute "2" with "two"
substitute "3" with "three"
substitute "4" with "four"
substitute "5" with "five"
substitute "6" with "six"
substitute "7" with "seven"
substitute "8" with "eight"
substitute "9" with "nine"
```

FILES

/usr/lib/locale/*locale*/LC_COLLATE/coll.so
shared library containing collation routines for *locale*

/opt/SUNWspro/bin/cc or any C compiler that supports these options:

−G	to output dynamically linked library
−o	to specify output filename
−O	to optimize code
−K pic	to generate position independent code

SEE ALSO

memory(3C), **setlocale**(3C), **strcoll**(3C), **strxfrm**(3C), **wscoll**(3I), **wsxfrm**(3I), **environ**(5)

NOTES

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