NAME

tr – translate characters

SYNOPSIS

/usr/bin/tr [–cs] *string1 string2*
/usr/bin/tr –s | –d [–c] *string1*
/usr/bin/tr –ds [–c] *string1 string2*

/usr/bin/xpg4/tr [–cs] *string1 string2*
/usr/bin/xpg4/tr –s | –d [–c] *string1*
/usr/bin/xpg4/tr –ds [–c] *string1 string2*

AVAILABILITY

/usr/bin/tr SUNWcsu

/usr/xpg4/bin/tr SUNWxcu4

DESCRIPTION

The **tr** utility copies the standard input to the standard output with substitution or deletion of selected characters. The options specified and the *string1* and *string2* operands control translations that occur while copying characters and single-character collating elements.

OPTIONS

The following options are supported:

–c Complement the set of characters specified by *string1*.

–d Delete all occurrences of input characters that are specified by *string1*.

–s Replace instances of repeated characters with a single character.

When the –**d** option is not specified:

- Each input character found in the array specified by *string1* is replaced by the character in the same relative position in the array specified by *string2*. When the array specified by *string2* is shorter that the one specified by *string1*, the results are unspecified.
- If the –**c** option is specified, the complements of the characters specified by *string1* (the set of all characters in the current character set, as defined by the current setting of **LC_CTYPE**, except for those actually specified in the *string1* operand) are placed in the array in ascending collation sequence, as defined by the current setting of **LC_COLLATE**. Because the order in which characters specified by character class expressions or equivalence class expressions is undefined, such expressions should only be used if the intent is to map several characters into one. An exception is case conversion, as described previously.

When the –**d** option is specified:

- Input characters found in the array specified by *string1* will be deleted.
- When the –**c** option is specified with –**d**, all characters except those specified by *string1* will be deleted.
- The contents of *string2* will be ignored, unless the –**s** option is also specified.
- The same string cannot be used for both the –**d** and the –**s** option; when both options

are specified, both *string1* (used for deletion) and *string2* (used for squeezing) are required.

When the −**s** option is specified, after any deletions or translations have taken place, repeated sequences of the same character will be replaced by one occurrence of the same character, if the character is found in the array specified by the last operand. If the last operand contains a character class, such as the following example:

tr -s '[:space:]'

the last operand's array will contain all of the characters in that character class. However, in a case conversion, as described previously, such as

tr -s '[:upper:]' '[:lower:]'

the last operand's array will contain only those characters defined as the second characters in each of the **toupper** or **tolower** character pairs, as appropriate.

An empty string used for *string1* or *string2* produces undefined results.

OPERANDS

The following operands are supported:

string1

string2 Translation control strings. Each string represents a set of characters to be converted into an array of characters used for the translation.

The operands *string1* and *string2* (if specified) define two arrays of characters. The constructs in the following list can be used to specify characters or single-character collating elements. If any of the constructs result in multi-character collating elements, **tr** will exclude, without a diagnostic, those multi-character elements from the resulting array.

character Any character not described by one of the conventions below represents itself.

\ *octal* Octal sequences can be used to represent characters with specific coded values. An octal sequence consists of a backslash followed by the longest sequence of one-, two- or three-octal-digit characters (01234567). The sequence causes the character whose encoding is represented by the one-, two- or three-digit octal integer to be placed into the array. Multi-byte characters require multiple, concatenated escape sequences of this type, including the leading \ for each byte.

\ *character* The backslash-escape sequences **\a**, **\b**, **\f**, **\n**, **\r**, **\t**, and **\v** are supported. The results of using any other character, other than an octal digit, following the backslash are unspecified.

/usr/xpg4/bin/tr *c-c*

/usr/bin/tr [*c-c*] Represents the range of collating elements between the range endpoints, inclusive, as defined by the current setting of the **LC_COLLATE** locale category. The starting endpoint must precede the second endpoint in the current collation order.

The characters or collating elements in the range are placed in the array in ascending collation sequence.

[:*class***:]** Represents all characters belonging to the defined character class, as defined by the current setting of the **LC_CTYPE** locale category. The following character class names will be accepted when specified in *string1*:

alnum	**blank**	**digit**	**lower**	**punct**	**upper**
alpha	**cntrl**	**graph**	**print**	**space**	**xdigit**

In addition, character class expressions of the form **[:***name***:]** are recognized in those locales where the *name* keyword has been given a **charclass** definition in the **LC_CTYPE** category.

When both the −**d** and −**s** options are specified, any of the character class names will be accepted in *string2*. Otherwise, only character class names **lower** or **upper** are valid in *string2* and then only if the corresponding character class **upper** and **lower**, respectively, is specified in the same relative position in *string1*. Such a specification is interpreted as a request for case conversion. When **[:lower:]** appears in *string1* and **[:upper:]** appears in *string2,* the arrays will contain the characters from the **toupper** mapping in the **LC_CTYPE** category of the current locale. When **[:upper:]** appears in *string1* and **[:lower:]** appears in *string2,* the arrays will contain the characters from the **tolower** mapping in the **LC_CTYPE** category of the current locale. The first character from each mapping pair will be in the array for *string1* and the second character from each mapping pair will be in the array for *string2* in the same relative position.

Except for case conversion, the characters specified by a character class expression are placed in the array in an unspecified order.

If the name specified for *class* does not define a valid character class in the current locale, the behavior is undefined.

[=*equiv***=]** Represents all characters or collating elements belonging to the same equivalence class as *equiv*, as defined by the current setting of the **LC_COLLATE** locale category. An equivalence class expression is allowed only in *string1*, or in *string2* when it is being used by the combined −**d** and −**s** options. The characters belonging to the equivalence class are placed in the array in an unspecified order.

[*x***n*] Represents *n* repeated occurrences of the character *x*. Because this expression is used to map multiple characters to one, it is only valid when it occurs in *string2*. If *n* is omitted or is **0**, it is interpreted as large enough to extend the *string2*-based sequence to the length of the *string1*-based sequence. If *n* has a leading **0**, it is interpreted as an octal value. Otherwise, it is interpreted as a decimal value.

EXAMPLES

1. The following example creates a list of all words in *file1* one per line in *file2*, where a word is taken to be a maximal string of letters.

tr -cs "[:alpha:]" "[\n*]" <file1 >file2

2. The next example translates all lower-case characters in **file1** to upper-case and writes the results to standard output.

   tr "[:lower:]" "[:upper:]" <file1

   Note that the caveat expressed in the corresponding example is no longer in effect. This case conversion is now a special case that employs the **tolower** and **toupper** classifications, ensuring that proper mapping is accomplished (when the locale is correctly defined).

3. This example uses an equivalence class to identify accented variants of the base character **e** in **file1**, which are stripped of diacritical marks and written to **file2**.

   tr "[=e=]" e <file1 >file2

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **tr**: **LC_COLLATE**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

EXIT STATUS

The following exit values are returned:

0 All input was processed successfully.

>0 An error occurred.

SEE ALSO

ed(1), **sed**(1), **sh**(1), **ascii**(5), **environ**(5)

NOTES

Will not handle ASCII NUL in *string1* or *string2*; always deletes NUL from input.