NAME

tr - translate characters

SYNOPSIS

tr [**-Ccsu**] *string1 string2*
tr [**-Ccu**] **-d** *string1*
tr [**-Ccu**] **-s** *string1*
tr [**-Ccu**] **-ds** *string1 string2*

DESCRIPTION

The **tr** utility copies the standard input to the standard output with substitution or deletion of selected characters.

The following options are available:

-C Complement the set of characters in *string1*, that is "**-C** ab" includes every character except for ‘a’ and ‘b’.

-c Same as **-C** but complement the set of values in *string1*.

-d Delete characters in *string1* from the input.

-s Squeeze multiple occurrences of the characters listed in the last operand (either *string1* or *string2*) in the input into a single instance of the character. This occurs after all deletion and translation is completed.

-u Guarantee that any output is unbuffered.

In the first synopsis form, the characters in *string1* are translated into the characters in *string2* where the first character in *string1* is translated into the first character in *string2* and so on. If *string1* is longer than *string2*, the last character found in *string2* is duplicated until *string1* is exhausted.

In the second synopsis form, the characters in *string1* are deleted from the input.

In the third synopsis form, the characters in *string1* are compressed as described for the **-s** option.

In the fourth synopsis form, the characters in *string1* are deleted from the input, and the characters in *string2* are compressed as described for the **-s** option.

The following conventions can be used in *string1* and *string2* to specify sets of characters:

character

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

\octal A backslash followed by 1, 2 or 3 octal digits represents a character with that encoded value. To follow an octal sequence with a digit as a character, left zero-pad the octal sequence to the full 3 octal digits.

\character

A backslash followed by certain special characters maps to special values.

\a <alert character>
\b <backspace>
\f <form-feed>
\n <newline>
\r <carriage return>
\t <tab>
\v <vertical tab>

A backslash followed by any other character maps to that character.

c-c For non-octal range endpoints represents the range of characters between the range endpoints, inclusive, in ascending order, as defined by the collation sequence. If either or both of the range endpoints are octal sequences, it represents the range of specific coded values between the range endpoints, inclusive.

See the COMPATIBILITY section below for an important note regarding differences in the way the current implementation interprets range expressions differently from previous implementations.

[:class:] Represents all characters belonging to the defined character class. Class names are:

alnum	<alphanumeric characters>
alpha	<alphabetic characters>
blank	<whitespace characters>
cntrl	<control characters>
digit	<numeric characters>
graph	<graphic characters>
ideogram	<ideographic characters>
lower	<lower-case alphabetic characters>
phonogram	<phonographic characters>

print	<printable characters>
punct	<punctuation characters>
rune	<valid characters>
space	<space characters>
special	<special characters>
upper	<upper-case characters>
xdigit	<hexadecimal characters>

When "[:lower:]" appears in *string1* and "[:upper:]" appears in the same relative position in *string2*, it represents the characters pairs from the toupper mapping in the LC_CTYPE category of the current locale. When "[:upper:]" appears in *string1* and "[:lower:]" appears in the same relative position in *string2*, it represents the characters pairs from the tolower mapping in the LC_CTYPE category of the current locale.

With the exception of case conversion, characters in the classes are in unspecified order.

For specific information as to which ASCII characters are included in these classes, see ctype(3) and related manual pages.

[=equiv=]
Represents all characters belonging to the same equivalence class as *equiv*, ordered by their encoded values.

[#*n] Represents *n* repeated occurrences of the character represented by *#*. This expression is only valid when it occurs in *string2*. If *n* is omitted or is zero, it is be interpreted as large enough to extend *string2* sequence to the length of *string1*. If *n* has a leading zero, it is interpreted as an octal value, otherwise, it is interpreted as a decimal value.

ENVIRONMENT

The LANG, LC_ALL, LC_CTYPE and LC_COLLATE environment variables affect the execution of **tr** as described in environ(7).

EXIT STATUS

The **tr** utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

The following examples are shown as given to the shell:

Create a list of the words in file1, one per line, where a word is taken to be a maximal string of letters.

```
tr -cs "[:alpha:]" "\n" < file1
```

Translate the contents of file1 to upper-case.

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

(This should be preferred over the traditional UNIX idiom of "tr a-z A-Z", since it works correctly in all locales.)

Strip out non-printable characters from file1.

```
tr -cd "[:print:]" < file1
```

Remove diacritical marks from all accented variants of the letter ‘e’:

```
tr "[=e=]" "e"
```

COMPATIBILITY

Previous FreeBSD implementations of **tr** did not order characters in range expressions according to the current locale’s collation order, making it possible to convert unaccented Latin characters (esp. as found in English text) from upper to lower case using the traditional UNIX idiom of "tr A-Z a-z". Since **tr** now obeys the locale’s collation order, this idiom may not produce correct results when there is not a 1:1 mapping between lower and upper case, or when the order of characters within the two cases differs. As noted in the *EXAMPLES* section above, the character class expressions "[:lower:]" and "[:upper:]" should be used instead of explicit character ranges like "a-z" and "A-Z".

"[=equiv=]" expression and collation for ranges are implemented for single byte locales only.

System V has historically implemented character ranges using the syntax "[c-c]" instead of the "c-c" used by historic BSD implementations and standardized by POSIX. System V shell scripts should work under this implementation as long as the range is intended to map in another range, i.e., the command "tr [a-z] [A-Z]" will work as it will map the ‘[’ character in *string1* to the ‘[’ character in *string2*. However, if the shell script is deleting or squeezing characters as in the command "tr -d [a-z]", the characters ‘[’ and ‘]’ will be included in the deletion or compression list which would not have happened under a historic System V implementation. Additionally, any scripts that depended on the sequence "a-z" to represent the three characters ‘a’, ‘-’ and ‘z’ will have to be rewritten as "a\-z".

The **tr** utility has historically not permitted the manipulation of NUL bytes in its input and, additionally, stripped NUL’s from its input stream. This implementation has removed this behavior as a bug.

The **tr** utility has historically been extremely forgiving of syntax errors, for example, the **-c** and **-s** options were ignored unless two strings were specified. This implementation will not permit illegal syntax.

STANDARDS

The **tr** utility conforms to IEEE Std 1003.1-2001 ("POSIX.1"). The "ideogram", "phonogram", "rune", and "special" character classes are extensions.

It should be noted that the feature wherein the last character of *string2* is duplicated if *string2* has less characters than *string1* is permitted by POSIX but is not required. Shell scripts attempting to be portable to other POSIX systems should use the "[#*]" convention instead of relying on this behavior. The **-u** option is an extension to the IEEE Std 1003.1-2001 ("POSIX.1") standard.