IEEE P1003.2 Draft 11.2 – September 1991

Copyright © 1991 by the
Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, NY 10017, USA
All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change. The publication, distribution, or copying of this draft, as well as all derivative works based on this draft, is expressly prohibited except as set forth below.

Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities only, and subject to the restrictions contained herein.

Permission is hereby also granted for member bodies and technical committees of ISO and IEC to reproduce this document for purposes of developing a national position, subject to the restrictions contained herein.

Permission is hereby also granted to the preceding entities to make limited copies of this document in an electronic form only for the stated activities.

The following restrictions apply to reproducing or transmitting the document in any form: 1) all copies or portions thereof must identify the document's IEEE project number and draft number, and must be accompanied by this entire notice in a prominent location; 2) no portion of this document may be redistributed in any modified or abridged form without the prior approval of the IEEE Standards Department.

Other entities seeking permission to reproduce this document, or any portion thereof, for standardization or other activities, must contact the IEEE Standards Department for the appropriate license.

Use of information contained in this unapproved draft is at your own risk.

IEEE Standards Department
Copyright and Permissions
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331, USA
+1 (908) 562-3800
+1 (908) 562-1571 [FAX]

```
touch ./file
```

in this case.

The −r option was added because several comments requested this capability. This option was named −f in an earlier draft, but was changed because the −f option is used in the BSD version of touch with a different meaning.

At least one historical implementation of touch incremented the exit code if −c was specified and the file did not exist. This standard requires exit status zero if no errors occur.

4.64 tr — Translate characters

4.64.1 Synopsis

tr [−cs] *string1 string2*

tr −s [−c] *string1*

tr −d [−c] *string1*

tr −ds [−c] *string1 string2*

4.64.2 Description

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

4.64.3 Options

The tr utility shall conform to the utility argument syntax guidelines described in 2.10.2.

The following options shall be supported by the implementation:

−c Complement the set of characters specified by *string1* . See 4.64.7.

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

−s Replace instances of repeated characters with a single character, as described in 4.64.7.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

4.64.4 Operands

The following operands shall be supported by the implementation:

string1
string2 Translation control strings. Each string shall represent a set of characters to be converted into an array of characters used for the translation. For a detailed description of how the strings are interpreted, see 4.64.7.

4.64.5 External Influences

4.64.5.1 Standard Input

The standard input can be any type of file.

4.64.5.2 Input Files

None.

4.64.5.3 Environment Variables

The following environment variables shall affect the execution of `tr`:

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_COLLATE This variable shall determine the behavior of range expressions and equivalence classes.

LC_CTYPE This variable shall determine the locale for the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments) and the behavior of character classes.

LC_MESSAGES This variable shall determine the language in which messages should be written.

4.64.5.4 Asynchronous Events

Default.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

13219 ### 4.64.6 External Effects

13220 #### 4.64.6.1 Standard Output

13221 The `tr` output shall be identical to the input, with the exception of the specified
13222 transformations.

13223 #### 4.64.6.2 Standard Error

13224 Used only for diagnostic messages.

13225 #### 4.64.6.3 Output Files

13226 None.

13227 ### 4.64.7 Extended Description

13228 The operands *string1* and *string2* (if specified) define two arrays of characters or
13229 collating elements. The following conventions can be used to specify characters or
13230 collating elements:

13231 *character* Any character not described by one of the conventions below
13232 shall represent itself.

13233 *octal* Octal sequences can be used to represent characters with
13234 specific coded values. An octal sequence shall consist of a
13235 backslash followed by the longest sequence of one-, two-, or
13236 three-octal-digit characters (01234567). The sequence shall
13237 cause the character whose encoding is represented by the one-,
13238 two-, or three-digit octal integer to be placed into the array. If 1
13239 the size of a byte on the system is greater than nine bits, the 1
13240 valid escape sequence used to represent a byte is 1
13241 implementation-defined. Multibyte characters require multi- 1
13242 ple, concatenated escape sequences of this type, including the 1
13243 leading \ for each byte. 1

13244 *character* The backslash-escape sequences in Table 2-15 (see 2.12) shall
13245 be supported. The results of using any other character, other
13246 than an octal digit, following the backslash are unspecified.

13247 *c*–*c* Represents the range of collating elements between the range 2
13248 endpoints, inclusive, as defined by the current setting of the
13249 LC_COLLATE locale category. The starting endpoint shall pre-
13250 cede the second endpoint in the current collation order. The
13251 characters or collating elements in the range shall be placed in
13252 the array in ascending collation sequence. No multicharacter
13253 collating elements shall be included in the range.

13254 `[:`*class*`:]` Represents all characters belonging to the defined character
13255 class, as defined by the current setting of the LC_CTYPE locale

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

13256 category. The following character class names shall be
13257 accepted when specified in *string1* :

13258 `alnum cntrl lower space`
13259 `alpha digit print upper`
13260 `blank graph punct xdigit`

13261 When the `−d` and `−s` options are specified together, any of the
13262 character class names shall be accepted in *string2* . Otherwise,
13263 only character class names `lower` or `upper` shall be accepted in
13264 *string2* and then only if the corresponding character class
13265 (`upper` and `lower`, respectively) is specified in the same rela-
13266 tive position in *string1* . Such a specification shall be interpreted
13267 as a request for case conversion. When `[:lower:]` appears in
13268 *string1* and `[:upper:]` appears in *string2,* the arrays shall
13269 contain the characters from the `toupper` mapping in the
13270 LC_CTYPE category of the current locale. When `[:upper:]`
13271 appears in *string1* and `[:lower:]` appears in *string2,* the
13272 arrays shall contain the characters from the `tolower` mapping
13273 in the LC_CTYPE category of the current locale. The first char-
13274 acter from each mapping pair shall be in the array for *string1*
13275 and the second character from each mapping pair shall be in
13276 the array for *string2* in the same relative position.

13277 Except for case conversion, the characters specified by a charac-
13278 ter class expression shall be placed in the array in an
13279 unspecified order.

13280 If the name specified for *class* does not define a valid character
13281 class in the current locale, the behavior is undefined.

13282 `[=`*equiv*`=]` Represents all characters or collating elements belonging to the
13283 same equivalence class as *equiv* , as defined by the current set-
13284 ting of the LC_COLLATE locale category. An equivalence class
13285 expression shall be allowed only in *string1* , or in *string2* when
13286 it is being used by the combined `−d` and `−s` options. The char-
13287 acters belonging to the equivalence class shall be placed in the
13288 array in an unspecified order.

13289 `[`*x*`*`*n*`]` Represents *n* repeated occurrences of the character or collating
13290 symbol *x*. Because this expression is used to map multiple
13291 characters to one, it is only valid when it occurs in *string2* . If *n*
13292 is omitted or is zero, it shall be interpreted as large enough to
13293 extend the *string2* -based sequence to the length of the *string1* -
13294 based sequence. If *n* has a leading zero, it shall be interpreted
13295 as an octal value. Otherwise, it shall be interpreted as a
13296 decimal value.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

When the −d option is not specified:

— Each input character or collating element found in the array specified by *string1* shall be replaced by the character or collating element 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 without −d, the complement 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—shall be 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 or collating elements found in the array specified by *string1* shall be deleted.

— When the −c option is specified with −d, all characters except those specified by *string1* shall be deleted. The contents of *string2* shall 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) shall be required.

When the −s option is specified, after any deletions or translations have taken place, repeated sequences of the same character shall 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 shall 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 shall contain only those characters defined as the second characters in each of the `toupper` or `tolower` character pairs, as appropriate.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

13332 ### 4.64.8 Exit Status

13333 The `tr` utility shall exit with one of the following values:

13334 0 All input was processed successfully.

13335 >0 An error occurred.

13336 ### 4.64.9 Consequences of Errors

13337 Default.

13338 ### 4.64.10 Rationale. *(This subclause is not a part of P1003.2)*

13339 **Examples, Usage**

13340 If necessary, *string1* and *string2* can be quoted to avoid pattern matching by the
13341 shell.

13342 The following example creates a list of all words in *file1* one per line in *file2*,
13343 where a word is taken to be a maximal string of letters.

13344 ```
tr -cs "[:alpha:]" "[\n*]" <file1 >file2
```

13345 If an ordinary digit (representing itself) is to follow an octal sequence, the octal
13346 sequence must use the full three digits to avoid ambiguity.

13347 When *string2* is shorter than *string1*, a difference results between historical
13348 System V and BSD systems. A BSD system will pad *string2* with the last charac-
13349 ter found in *string2*. Thus, it is possible to do the following:

13350 ```
tr 0123456789 d
```

13351 which would translate all digits to the letter `d`. Since this area is specifically
13352 unspecified in the standard, both the BSD and System V behaviors are allowed,
13353 but a conforming application cannot rely on the BSD behavior. It would have to
13354 code the example in the following way:

13355 ```
tr 0123456789 '[d*]'
```

13356 It should be noted that, despite similarities in appearance, the string operands
13357 used by `tr` are not regular expressions.

13358 On historical System V systems, a range expression requires enclosing square- 2
13359 brackets, such as: 2

13360 ```
tr '[a-z]' '[A-Z]'
```
2

13361 However, BSD-based systems did not require the brackets and this convention is 2
13362 used by POSIX.2 to avoid breaking large numbers of BSD scripts: 2

13363 ```
tr a-z A-Z
```
2

13364 The preceding System V script will continue to work because the brackets, treated 2
13365 as regular characters, are translated to themselves. However, any System V 2

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
```

13366 script that relied on `a-z` representing the three characters `a`, `-`, and `z` will have to 2
13367 be rewritten as `az-` or `a\-z`. 2

13368 **History of Decisions Made**

13369 In some earlier drafts, an explicit option, `–n`, was added to disable the historical
13370 behavior of stripping NUL characters from the input. It was felt that automati-
13371 cally stripping NUL characters from the input was not correct functionality. How-
13372 ever, the removal of `–n` in a later draft does not remove the requirement that `tr`
13373 correctly process NUL characters in its input stream. NUL characters can be
13374 stripped by using `tr –d '\000'`.

13375 Historical implementations of `tr` differ widely in syntax and behavior. For exam-
13376 ple, the BSD version has not needed the bracket characters for the repetition
13377 sequence. The POSIX.2 `tr` syntax is based more closely on the System V and
13378 XPG3 model, while attempting to accommodate historical BSD implementations.
13379 In the case of the short *string2* padding, the decision was to unspecify the
13380 behavior and preserve System V and XPG scripts, which might find difficulty with
13381 the BSD method. The assumption was made that BSD users of `tr` will have to
13382 make accommodations to meet the POSIX.2 syntax anyway, and since it is possible
13383 to use the repetition sequence to duplicate the desired behavior, whereas there is
13384 no simple way to achieve the System V method, this was the correct, if not desir-
13385 able, approach.

13386 The use of octal values to specify control characters, while having historical pre-
13387 cedents, is not portable. The introduction of escape sequences for control charac-
13388 ters should provide the necessary portability. It is recognized that this may cause
13389 some historical scripts to break.

13390 A previous draft included support for multicharacter collating elements. Several
13391 balloters pointed out that, while `tr` does employ some syntactical elements from
13392 regular expressions, the aim of `tr` is quite different; ranges, for instance, do not
13393 mean the same thing (''any of the chars in the range matches,'' versus ''translate
13394 each character in the range to the output counterpart''). As a result, the previ-
13395 ously included support for multicharacter collating elements has been removed.
13396 What remains are ranges in current collation order (to support, e.g., accented
13397 characters), character classes, and equivalence classes.

13398 In XPG3, the `[:class:]` and `[=equiv=]` conventions are shown with double
13399 brackets, as in regular expression syntax. Several balloters objected to this,
13400 pointing out that `tr` does not implement regular expression principles, just bor-
13401 rows part of the syntax. Consequently, the `[:class:]` and `[=equiv=]` should be
13402 regarded as syntactical elements on a par with `[x*n]`, which is not an RE bracket
13403 expression.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

13404 ## 4.65 `true` — Return true value

13405 ### 4.65.1 Synopsis

13406 `true`

13407 ### 4.65.2 Description

13408 The `true` utility shall return with exit code zero.

13409 ### 4.65.3 Options

13410 None.

13411 ### 4.65.4 Operands

13412 None.

13413 ### 4.65.5 External Influences

13414 #### 4.65.5.1 Standard Input

13415 None.

13416 #### 4.65.5.2 Input Files

13417 None.

13418 #### 4.65.5.3 Environment Variables

13419 None.

13420 #### 4.65.5.4 Asynchronous Events

13421 Default.

13422 ### 4.65.6 External Effects

13423 #### 4.65.6.1 Standard Output

13424 None.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.