Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'sort.1p' command

$ man sort.1p

SORT(1P) POSIX Programmer's Manual SORT(1P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

sort ? sort, merge, or sequence check text files

SYNOPSIS

sort [-m] [-o output] [-bdfinru] [-t char] [-k keydef]... [file...]

sort [-c|-C] [-bdfinru] [-t char] [-k keydef] [file]

DESCRIPTION

The sort utility shall perform one of the following functions:

1. Sort lines of all the named files together and write the result to the specified output.
2. Merge lines of all the named (presorted) files together and write the result to the specified output.
3. Check that a single input file is correctly presorted.

Comparisons shall be based on one or more sort keys extracted from each line of input (or, if no sort keys are specified, the entire line up to, but not including, the terminating <newline>), and shall be per? formed using the collating sequence of the current locale. If this col? lating sequence does not have a total ordering of all characters (see

the Base Definitions volume of POSIX.1?2017, Section 7.3.2, LC_COL? LATE), any lines of input that collate equally should be further com? pared byte-by-byte using the collating sequence for the POSIX locale.

OPTIONS

The sort utility shall conform to the Base Definitions volume of POSIX.1?2017, Section 12.2, Utility Syntax Guidelines, except for Guideline 9, and the -k keydef option should follow the -b, -d, -f, -i, -n, and -r options. In addition, '+' may be recognized as an option de? limiter as well as '-'.

The following options shall be supported:

-c Check that the single input file is ordered as specified by the arguments and the collating sequence of the current lo? cale. Output shall not be sent to standard output. The exit code shall indicate whether or not disorder was detected or an error occurred. If disorder (or, with -u, a duplicate key) is detected, a warning message shall be sent to standard er? ror indicating where the disorder or duplicate key was found.

-C Same as -c, except that a warning message shall not be sent to standard error if disorder or, with -u, a duplicate key is detected.

-m Merge only; the input file shall be assumed to be already sorted.

-o output Specify the name of an output file to be used instead of the standard output. This file can be the same as one of the in? put files.

-u Unique: suppress all but one in each set of lines having equal keys. If used with the -c option, check that there are no lines with duplicate keys, in addition to checking that the input file is sorted.

The following options shall override the default ordering rules. When ordering options appear independent of any key field specifications, the requested field ordering rules shall be applied globally to all sort keys. When attached to a specific key (see -k), the specified or?

dering options shall override all global ordering options for that key.

-d Specify that only <blank> characters and alphanumeric charac? ters, according to the current setting of LC_CTYPE, shall be significant in comparisons. The behavior is undefined for a sort key to which -i or -n also applies.

-f Consider all lowercase characters that have uppercase equiva? lents, according to the current setting of LC_CTYPE, to be the uppercase equivalent for the purposes of comparison.

-i Ignore all characters that are non-printable, according to the current setting of LC_CTYPE. The behavior is undefined for a sort key for which -n also applies.

-n Restrict the sort key to an initial numeric string, consist? ing of optional <blank> characters, optional <hyphen-minus> character, and zero or more digits with an optional radix character and thousands separators (as defined in the current locale), which shall be sorted by arithmetic value. An empty digit string shall be treated as zero. Leading zeros and signs on zeros shall not affect ordering.

-r Reverse the sense of comparisons.

The treatment of field separators can be altered using the options:

-b Ignore leading <blank> characters when determining the start? ing and ending positions of a restricted sort key. If the -b option is specified before the first -k option, it shall be applied to all -k options. Otherwise, the -b option can be attached independently to each -k field_start or field_end option-argument (see below).

-t char Use char as the field separator character; char shall not be considered to be part of a field (although it can be included in a sort key). Each occurrence of char shall be significant (for example, <char><char> delimits an empty field). If -t is not specified, <blank> characters shall be used as default field separators; each maximal non-empty sequence of <blank> characters that follows a non-<blank> shall be a field sepa?

rator.

Sort keys can be specified using the options:

-k keydef The keydef argument is a restricted sort key field defini?
tion. The format of this definition is:

field_start[type][,field_end[type]]

where field_start and field_end define a key field restricted to a portion of the line (see the EXTENDED DESCRIPTION sec? tion), and type is one or more modifiers from the list of characters 'b', 'd', 'f', 'i', 'n', 'r'. The 'b' modifier shall behave like the -b option, but shall apply only to the field_start or field_end to which it is attached. The other modifiers shall behave like the corresponding options, but shall apply only to the key field to which they are attached; they shall have this effect if specified with field_start, field_end, or both. If any modifier is attached to a field_start or to a field_end, no option shall apply to ei? ther. Implementations shall support at least nine occurrences of the -k option, which shall be significant in command line order. If no -k option is specified, a default sort key of the entire line shall be used.

When there are multiple key fields, later keys shall be com? pared only after all earlier keys compare equal. Except when the -u option is specified, lines that otherwise compare equal shall be ordered as if none of the options -d, -f, -i, -n, or -k were present (but with -r still in effect, if it was specified) and with all bytes in the lines significant to the comparison. The order in which lines that still compare equal are written is unspecified.

OPERANDS

The following operand shall be supported:

file A pathname of a file to be sorted, merged, or checked. If no file operands are specified, or if a file operand is '-', the standard input shall be used. If sort encounters an error

when opening or reading a file operand, it may exit without writing any output to standard output or processing later op? erands.

STDIN

The standard input shall be used only if no file operands are speci? fied, or if a file operand is '-'. See the INPUT FILES section.

INPUT FILES

The input files shall be text files, except that the sort utility shall add a <newline> to the end of a file ending with an incomplete last line.

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of sort:

LANG Provide a default value for the internationalization vari? ables that are unset or null. (See the Base Definitions vol? ume of POSIX.1?2017, Section 8.2, Internationalization Vari? ables for the precedence of internationalization variables used to determine the values of locale categories.)

LC_ALL If set to a non-empty string value, override the values of all the other internationalization variables.

LC_COLLATE

Determine the locale for ordering rules.

LC_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files) and the behavior of character classification for the -b, -d, -f, -i, and -n options.

LC_MESSAGES

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard er? ror.

LC_NUMERIC

Determine the locale for the definition of the radix charac? ter and thousands separator for the -n option.

NLSPATH Determine the location of message catalogs for the processing of LC_MESSAGES.

ASYNCHRONOUS EVENTS

Default.

STDOUT

Unless the -o or -c options are in effect, the standard output shall contain the sorted input.

STDERR

The standard error shall be used for diagnostic messages. When -c is specified, if disorder is detected (or if -u is also specified and a duplicate key is detected), a message shall be written to the standard error which identifies the input line at which disorder (or a duplicate key) was detected. A warning message about correcting an incomplete last line of an input file may be generated, but need not affect the final exit status.

OUTPUT FILES

If the -o option is in effect, the sorted input shall be written to the file output.

EXTENDED DESCRIPTION

The notation:

-k field_start[type][,field_end[type]]

shall define a key field that begins at field_start and ends at field_end inclusive, unless field_start falls beyond the end of the line or after field_end, in which case the key field is empty. A miss? ing field_end shall mean the last character of the line.

A field comprises a maximal sequence of non-separating characters and, in the absence of option -t, any preceding field separator.

The field_start portion of the keydef option-argument shall have the form:

field_number[.first_character]

Fields and characters within fields shall be numbered starting with 1.

The field_number and first_character pieces, interpreted as positive decimal integers, shall specify the first character to be used as part

of a sort key. If .first_character is omitted, it shall refer to the first character of the field.

The field_end portion of the keydef option-argument shall have the form:

field_number[.last_character]

The field_number shall be as described above for field_start. The last_character piece, interpreted as a non-negative decimal integer, shall specify the last character to be used as part of the sort key. If last_character evaluates to zero or .last_character is omitted, it shall refer to the last character of the field specified by field_num?ber.

If the -b option or b type modifier is in effect, characters within a field shall be counted from the first non-<blank> in the field. (This shall apply separately to first_character and last_character.)

EXIT STATUS

The following exit values shall be returned:

0 All input files were output successfully, or -c was specified and the input file was correctly sorted.

1 Under the -c option, the file was not ordered as specified, or if the -c and -u options were both specified, two input lines were found with equal keys.

>1 An error occurred.

CONSEQUENCES OF ERRORS

The default requirements shall apply, except that if sort encounters an error when opening or reading a file operand, it may exit without writ?ing any output to standard output or processing later operands.

The following sections are informative.

APPLICATION USAGE

The default value for -t, <blank>, has different properties from, for example, -t"<space>". If a line contains:

<space><space>foo

the following treatment would occur with default separation as opposed to specifically selecting a <space>:

??

?Field ? Default ? -t "<space>" ?

??

? 1 ? <space><space>foo ? empty ?

? 2 ? empty ? empty ?

? 3 ? empty ? foo ?

??

The leading field separator itself is included in a field when -t is not used. For example, this command returns an exit status of zero, meaning the input was already sorted:

```
sort -c -k 2 <<eof
y<tab>b
x<space>a
eof
```

(assuming that a <tab> precedes the <space> in the current collating sequence). The field separator is not included in a field when it is explicitly set via -t. This is historical practice and allows usage such as:

```
sort -t "|" -k 2n <<eof
Atlanta|425022|Georgia
Birmingham|284413|Alabama
Columbia|100385|South Carolina
eof
```

where the second field can be correctly sorted numerically without re? gard to the non-numeric field separator.

The wording in the OPTIONS section clarifies that the -b, -d, -f, -i, -n, and -r options have to come before the first sort key specified if they are intended to apply to all specified keys. The way it is de? scribed in this volume of POSIX.1?2017 matches historical practice, not historical documentation. The results are unspecified if these options are specified after a -k option.

The -f option might not work as expected in locales where there is not a one-to-one mapping between an uppercase and a lowercase letter.

When using sort to process pathnames, it is recommended that LC_ALL, or at least LC_CTYPE and LC_COLLATE, are set to POSIX or C in the environment, since pathnames can contain byte sequences that do not form valid characters in some locales, in which case the utility's behavior would be undefined. In the POSIX locale each byte is a valid single-byte character, and therefore this problem is avoided.

If the collating sequence of the current locale does not have a total ordering of all characters, this can affect the behavior of sort in the following ways:

* As sort -u suppresses lines with duplicate keys, it suppresses lines that collate equally but are not identical.
* The output of sort (without -u) can contain identical lines that are not adjacent, if it does not implement the recommended further byte-by-byte comparison of lines that collate equally. This affects the use of sort with comm and uniq; see the APPLICATION USAGE for those utilities.

EXAMPLES

1. The following command sorts the contents of infile with the second field as the sort key:

   sort -k 2,2 infile

2. The following command sorts, in reverse order, the contents of infile1 and infile2, placing the output in outfile and using the second character of the second field as the sort key (assuming that the first character of the second field is the field separator):

   sort -r -o outfile -k 2.2,2.2 infile1 infile2

3. The following command sorts the contents of infile1 and infile2 using the second non-<blank> of the second field as the sort key:

   sort -k 2.2b,2.2b infile1 infile2

4. The following command prints the System V password file (user database) sorted by the numeric user ID (the third <colon>-separated field):

   sort -t : -k 3,3n /etc/passwd

5. The following command prints the lines of the already sorted file

infile, suppressing all but one occurrence of lines having the same third field:

```
sort -um -k 3.1,3.0 infile
```

RATIONALE

Examples in some historical documentation state that options -um with one input file keep the first in each set of lines with equal keys. This behavior was deemed to be an implementation artifact and was not standardized.

The -z option was omitted; it is not standard practice on most systems and is inconsistent with using sort to sort several files individually and then merge them together. The text concerning -z in historical doc?umentation appeared to require implementations to determine the proper buffer length during the sort phase of operation, but not during the merge.

The -y option was omitted because of non-portability. The -M option, present in System V, was omitted because of non-portability in interna?tional usage.

An undocumented -T option exists in some implementations. It is used to specify a directory for intermediate files. Implementations are encour?aged to support the use of the TMPDIR environment variable instead of adding an option to support this functionality.

The -k option was added to satisfy two objections. First, the zero-based counting used by sort is not consistent with other utility con?ventions. Second, it did not meet syntax guideline requirements.

Historical documentation indicates that ``setting -n implies -b''. The description of -n already states that optional leading <blank>s are tolerated in doing the comparison. If -b is enabled, rather than im?plied, by -n, this has unusual side-effects. When a character offset is used in a column of numbers (for example, to sort modulo 100), that offset is measured relative to the most significant digit, not to the column. Based upon a recommendation from the author of the original sort utility, the -b implication has been omitted from this volume of POSIX.1?2017, and an application wishing to achieve the previously men?

tioned side-effects has to code the -b flag explicitly.

Earlier versions of this standard allowed the -o option to appear after operands. Historical practice allowed all options to be interspersed with operands. This version of the standard allows implementations to accept options after operands but conforming applications should not use this form.

Earlier versions of this standard also allowed the -number and +number options. These options are no longer specified by POSIX.1?2008 but may be present in some implementations.

Historical implementations produced a message on standard error when -c was specified and disorder was detected, and when -c and -u were speci? fied and a duplicate key was detected. An earlier version of this stan? dard contained wording that did not make it clear that this message was allowed and some implementations removed this message to be sure that they conformed to the standard's requirements. Confronted with this difference in behavior, interactive users that wanted to be sure that they got visual feedback instead of just exit code 1 could have used a command like:

```
sort -c file || echo disorder
```

whether or not the sort utility provided a message in this case. But, it was not easy for a user to find where the disorder or duplicate key occurred on implementations that do not produce a message, especially when some parts of the input line were not part of the key and when one or more of the -b, -d, -f, -i, -n, or -r options or keydef type modi? fiers were in use. POSIX.1?2008 requires a message to be produced in this case. POSIX.1?2008 also contains the -C option giving users the ability to choose either behavior.

When a disorder or duplicate is found when the -c option is specified, some implementations print a message containing the first line that is out of order or contains a duplicate key; others print a message speci? fying the line number of the offending line. This standard allows ei? ther type of message.

Implementations are encouraged to perform the recommended further byte-

by-byte comparison of lines that collate equally, even though this may affect efficiency. The impact on efficiency can be mitigated by only performing the additional comparison if the current locale's collating sequence does not have a total ordering of all characters (if the im? plementation provides a way to query this) or by only performing the additional comparison if the locale name associated with the LC_COLLATE category has an '@' modifier in the name (since locales without an '@' modifier should have a total ordering of all characters ? see the Base Definitions volume of POSIX.1?2017, Section 7.3.2, LC_COLLATE). Note that if the implementation provides a stable sort option as an exten? sion (usually -s), the additional comparison should not be performed when this option has been specified.

FUTURE DIRECTIONS

A future version of this standard may require that if the collating se? quence of the current locale does not have a total ordering of all characters, any lines of input that collate equally when comparing them as whole lines are further compared byte-by-byte using the collating sequence for the POSIX locale.

SEE ALSO

comm, join, uniq

The Base Definitions volume of POSIX.1?2017, Section 7.3.2, LC_COLLATE, Chapter 8, Environment Variables, Section 12.2, Utility Syntax Guide? lines

The System Interfaces volume of POSIX.1?2017, toupper()

COPYRIGHT

at http://www.opengroup.org/unix/online.html .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.ker?nel.org/doc/man-pages/reporting_bugs.html .