(coreutils) sort invocation

(coreutils) **Operating on sorted files** # *(coreutils)* **uniq invocation**

```
`sort': Sort text files
=======================

   `sort' sorts, merges, or compares all the lines from the given
files, or standard input if none are given or for a FILE of `-'.  By
default, `sort' writes the results to standard output.  Synopsis:

     sort [OPTION]... [FILE]...

   `sort' has three modes of operation: sort (the default), merge, and
check for sortedness.  The following options change the operation mode:

`-c'
`--check'
     Check whether the given files are already sorted: if they are not
     all sorted, print an error message and exit with a status of 1.
     Otherwise, exit successfully.

`-m'
`--merge'
     Merge the given files by sorting them as a group.  Each input file
     must always be individually sorted.  It always works to sort
     instead of merge; merging is provided because it is faster, in the
     case where it works.


   A pair of lines is compared as follows: if any key fields have been
specified, `sort' compares each pair of fields, in the order specified
on the command line, according to the associated ordering options,
until a difference is found or no fields are left.  Unless otherwise
specified, all comparisons use the character collating sequence
specified by the `LC_COLLATE' locale.  (1)

   If any of the global options `bdfgiMnr' are given but no key fields
are specified, `sort' compares the entire lines according to the global
options.

   Finally, as a last resort when all keys compare equal (or if no
ordering options were specified at all), `sort' compares the entire
lines.  The last resort comparison honors the `--reverse' (`-r') global
option.  The `--stable' (`-s') option disables this last-resort
comparison so that lines in which all fields compare equal are left in
their original relative order.  If no fields or global options are
specified, `--stable' (`-s') has no effect.

   GNU `sort' (as specified for all GNU utilities) has no limits on
input line length or restrictions on bytes allowed within lines.  In
addition, if the final byte of an input file is not a newline, GNU
`sort' silently supplies one.  A line's trailing newline is not part of
the line for comparison purposes.

   Upon any error, `sort' exits with a status of `2'.

   If the environment variable `TMPDIR' is set, `sort' uses its value
as the directory for temporary files instead of `/tmp'.  The
`--temporary-directory' (`-T') option in turn overrides the environment
variable.

   The following options affect the ordering of output lines.  They may
be specified globally or as part of a specific key field.  If no key
fields are specified, global options apply to comparison of entire
lines; otherwise the global options are inherited by key fields that do
not specify any special options of their own.  In pre-POSIX versions of
`sort', global options affect only later key fields, so portable shell
scripts should specify global options first.

`-b'
`--ignore-leading-blanks'
     Ignore leading blanks when finding sort keys in each line.  The
     `LC_CTYPE' locale determines character types.

`-d'
`--dictionary-order'
     Sort in "phone directory" order: ignore all characters except
     letters, digits and blanks when sorting.  The `LC_CTYPE' locale
     determines character types.
```

`-f'
`--ignore-case'
Fold lowercase characters into the equivalent uppercase characters when comparing so that, for example, `b' and `B' sort as equal. The `LC_CTYPE' locale determines character types.

`-g'
`--general-numeric-sort'
Sort numerically, using the standard C function `strtod' to convert a prefix of each line to a double-precision floating point number. This allows floating point numbers to be specified in scientific notation, like `1.0e-34' and `10e100'. The `LC_NUMERIC' locale determines the decimal-point character. Do not report overflow, underflow, or conversion errors. Use the following collating sequence:

* Lines that do not start with numbers (all considered to be equal).

* NaNs ("Not a Number" values, in IEEE floating point arithmetic) in a consistent but machine-dependent order.

* Minus infinity.

* Finite numbers in ascending numeric order (with -0 and +0 equal).

* Plus infinity.

Use this option only if there is no alternative; it is much slower than `--numeric-sort' (`-n') and it can lose information when converting to floating point.

`-i'
`--ignore-nonprinting'
Ignore nonprinting characters. The `LC_CTYPE' locale determines character types.

`-M'
`--month-sort'
An initial string, consisting of any amount of whitespace, followed by a month name abbreviation, is folded to UPPER case and compared in the order `JAN' < `FEB' < ... < `DEC'. Invalid names compare low to valid names. The `LC_TIME' locale category determines the month spellings.

`-n'
`--numeric-sort'
Sort numerically: the number begins each line; specifically, it consists of optional whitespace, an optional `-' sign, and zero or more digits possibly separated by thousands separators, optionally followed by a decimal-point character and zero or more digits. The `LC_NUMERIC' locale specifies the decimal-point character and thousands separator.

Numeric sort uses what might be considered an unconventional method to compare strings representing floating point numbers. Rather than first converting each string to the C `double' type and then comparing those values, `sort' aligns the decimal-point characters in the two strings and compares the strings a character at a time. One benefit of using this approach is its speed. In practice this is much more efficient than performing the two corresponding string-to-double (or even string-to-integer) conversions and then comparing doubles. In addition, there is no corresponding loss of precision. Converting each string to `double' before comparison would limit precision to about 16 digits on most systems.

Neither a leading `+' nor exponential notation is recognized. To compare such strings numerically, use the `--general-numeric-sort' (`-g') option.

`-r'
`--reverse'
Reverse the result of comparison, so that lines with greater key values appear earlier in the output instead of later.

Other options are:

`-o OUTPUT-FILE'
`--output=OUTPUT-FILE'

Write output to OUTPUT-FILE instead of standard output. If necessary, `sort' reads input before opening OUTPUT-FILE, so you can safely sort a file in place by using commands like `sort -o F F' and `cat F | sort -o F'.

On newer systems, `-o' cannot appear after an input file if `POSIXLY_CORRECT' is set, e.g., `sort F -o F'. Portable scripts should specify `-o OUTPUT-FILE' before any input files.

`-S SIZE'
`--buffer-size=SIZE'
Use a main-memory sort buffer of the given SIZE. By default, SIZE is in units of 1024 bytes. Appending `%' causes SIZE to be interpreted as a percentage of physical memory. Appending `K' multiplies SIZE by 1024 (the default), `M' by 1,048,576, `G' by 1,073,741,824, and so on for `T', `P', `E', `Z', and `Y'. Appending `b' causes SIZE to be interpreted as a byte count, with no multiplication.

This option can improve the performance of `sort' by causing it to start with a larger or smaller sort buffer than the default. However, this option affects only the initial buffer size. The buffer grows beyond SIZE if `sort' encounters input lines larger than SIZE.

`-t SEPARATOR'
`--field-separator=SEPARATOR'
Use character SEPARATOR as the field separator when finding the sort keys in each line. By default, fields are separated by the empty string between a non-whitespace character and a whitespace character. That is, given the input line ` foo bar', `sort' breaks it into fields ` foo' and ` bar'. The field separator is not considered to be part of either the field preceding or the field following. But note that sort fields that extend to the end of the line, as `-k 2', or sort fields consisting of a range, as `-k 2,3', retain the field separators present between the endpoints of the range.

`-T TEMPDIR'
`--temporary-directory=TEMPDIR'
Use directory TEMPDIR to store temporary files, overriding the `TMPDIR' environment variable. If this option is given more than once, temporary files are stored in all the directories given. If you have a large sort or merge that is I/O-bound, you can often improve performance by using this option to specify directories on different disks and controllers.

`-u'
`--unique'
Normally, output only the first of a sequence of lines that compare equal. For the `--check' (`-c') option, check that no pair of consecutive lines compares equal.

`-k POS1[,POS2]'
`--key=POS1[,POS2]'
Specify a sort field that consists of the part of the line between POS1 and POS2 (or the end of the line, if POS2 is omitted), _inclusive_. Fields and character positions are numbered starting with 1. So to sort on the second field, you'd use `--key=2,2' (`-k 2,2'). See below for more examples.

`-z'
`--zero-terminated'
Treat the input as a set of lines, each terminated by a zero byte (ASCII NUL (Null) character) instead of an ASCII LF (Line Feed). This option can be useful in conjunction with `perl -0' or `find -print0' and `xargs -0' which do the same in order to reliably handle arbitrary pathnames (even those which contain Line Feed characters.)

Historical (BSD and System V) implementations of `sort' have differed in their interpretation of some options, particularly `-b', `-f', and `-n'. GNU sort follows the POSIX behavior, which is usually (but not always!) like the System V behavior. According to POSIX, `-n' no longer implies `-b'. For consistency, `-M' has been changed in the same way. This may affect the meaning of character positions in field specifications in obscure cases. The only fix is to add an explicit `-b'.

A position in a sort field specified with the `-k' option has the form `F.C', where F is the number of the field to use and C is the number of the first character from the beginning of the field. In a

start position, an omitted `.C' stands for the field's first character. In an end position, an omitted or zero `.C' stands for the field's last character. If the `-b' option was specified, the `.C' part of a field specification is counted from the first nonblank character of the field.

A sort key position may also have any of the option letters `Mbdfinr' appended to it, in which case the global ordering options are not used for that particular field. The `-b' option may be independently attached to either or both of the start and end positions of a field specification, and if it is inherited from the global options it will be attached to both. Keys may span multiple fields.

On older systems, `sort' supports an obsolete origin-zero syntax `+POS1 [-POS2]' for specifying sort keys. POSIX 1003.1-2001 (

Standards conformance) does not allow this; use `-k' instead.

Here are some examples to illustrate various combinations of options.

* Sort in descending (reverse) numeric order.

```
sort -nr
```

* Sort alphabetically, omitting the first and second fields. This uses a single key composed of the characters beginning at the start of field three and extending to the end of each line.

```
sort -k 3
```

* Sort numerically on the second field and resolve ties by sorting alphabetically on the third and fourth characters of field five. Use `:' as the field delimiter.

```
sort -t : -k 2,2n -k 5.3,5.4
```

  Note that if you had written `-k 2' instead of `-k 2,2' `sort' would have used all characters beginning in the second field and extending to the end of the line as the primary _numeric_ key. For the large majority of applications, treating keys spanning more than one field as numeric will not do what you expect.

  Also note that the `n' modifier was applied to the field-end specifier for the first key. It would have been equivalent to specify `-k 2n,2' or `-k 2n,2n'. All modifiers except `b' apply to the associated _field_, regardless of whether the modifier character is attached to the field-start and/or the field-end part of the key specifier.

* Sort the password file on the fifth field and ignore any leading white space. Sort lines with equal values in field five on the numeric user ID in field three.

```
sort -t : -k 5b,5 -k 3,3n /etc/passwd
```

  An alternative is to use the global numeric modifier `-n'.

```
sort -t : -n -k 5b,5 -k 3,3 /etc/passwd
```

* Generate a tags file in case-insensitive sorted order.

```
find src -type f -print0 | sort -t / -z -f | xargs -0 etags --append
```

  The use of `-print0', `-z', and `-0' in this case means that pathnames that contain Line Feed characters will not get broken up by the sort operation.

  Finally, to ignore both leading and trailing white space, you could have applied the `b' modifier to the field-end specifier for the first key,

```
sort -t : -n -k 5b,5b -k 3,3 /etc/passwd
```

  or by using the global `-b' modifier instead of `-n' and an explicit `n' with the second key specifier.

```
sort -t : -b -k 5,5 -k 3,3n /etc/passwd
```

---------- Footnotes ----------

(1) If you use a non-POSIX locale (e.g., by setting `LC_ALL' to `en_US'), then `sort' may produce output that is sorted differently

```
than you're accustomed to.  In that case, set the `LC_ALL' environment
variable to `C'.  Note that setting only `LC_COLLATE' has two problems.
First, it is ineffective if `LC_ALL' is also set.  Second, it has
undefined behavior if `LC_CTYPE' (or `LANG', if `LC_CTYPE' is unset) is
set to an incompatible value.  For example, you get undefined behavior
if `LC_CTYPE' is `ja_JP.PCK' but `LC_COLLATE' is `en_US.UTF-8'.
```

 *(coreutils)* **Operating on sorted files** #  *(coreutils)* **uniq invocation**
