

**NAME**

sort – sort, merge, or sequence check text files

**SYNOPSIS**

```
/usr/bin/sort [-bcdfimMnru] [-k keydef] [-o output]
[-S kmem] [-t char] [-T directory] [-y [kmem]]
[-z recsz] [+pos1 [-pos2]] [file]...
```

```
/usr/xpg4/bin/sort [-bcdfimMnru] [-k keydef] [-o output]
[-S kmem] [-t char] [-T directory] [-y [kmem]]
[-z recsz] [+pos1 [-pos2]] [file]...
```

**DESCRIPTION**

The **sort** command sorts lines of all the named files together and writes the result on the standard output.

Comparisons are based on one or more sort keys extracted from each line of input. By default, there is one sort key, the entire input line. Lines are ordered according to the collating sequence of the current locale.

**OPTIONS**

The following options alter the default behavior:

**/usr/bin/sort**

**-c** Checks that the single input file is ordered as specified by the arguments and the collating sequence of the current locale. The exit code is set and no output is produced unless the file is out of sort.

**/usr/xpg4/bin/sort**

**-c** Same as **/usr/bin/sort** except no output is produced under any circumstances.

**-m** Merges only. The input files are assumed to be already sorted.

**-o *output*** Specifies the name of an output file to be used instead of the standard output. This file can be the same as one of the input files.

**-S *kmem*** Specifies the maximum amount of swap-based memory used for sorting, in kilobytes (the default unit). *kmem* can also be specified directly as a number of bytes (b), kilobytes (k), megabytes (m), gigabytes (g), or terabytes (t); or as a percentage (%) of the installed physical memory.

**-T *directory*** Specifies the *directory* in which to place temporary files.

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

- y *kmem*** (obsolete). This option was used to specify the amount of main memory initially used by **sort**. Its functionality is not appropriate for a virtual memory system; memory usage for **sort** is now specified using the **-S** option.
- z *recsz*** (obsolete). This option was used to prevent abnormal termination when lines longer than the system-dependent default buffer size are encountered. Because **sort** automatically allocates buffers large enough to hold the longest line, this option has no effect.

### Ordering Options

The default sort order depends on the value of `LC_COLLATE`. If `LC_COLLATE` is set to **C**, sorting is in **ASCII** order. If `LC_COLLATE` is set to **en\_US**, sorting is case insensitive except when the two strings are otherwise equal and one has an uppercase letter earlier than the other. Other locales have other sort orders.

The following options override the default ordering rules. When ordering options appear independent of any key field specifications, the requested field ordering rules are applied globally to all sort keys. When attached to a specific key (see **Sort Key Options**), the specified ordering options override all global ordering options for that key. In the obsolescent forms, if one or more of these options follows a *+pos1* option, it affects only the key field specified by that preceding option.

- d** Dictionary order: only letters, digits, and blanks (spaces and tabs) are significant in comparisons.
- f** Folds lower-case letters into upper case.
- i** Ignores non-printable characters.
- M** Compares as months. The first three non-blank characters of the field are folded to upper case and compared. For example, in English the sorting order is "JAN" < "FEB" < ... < "DEC". Invalid fields compare low to "JAN". The **-M** option implies the **-b** option (see below).
- n** Restricts the sort key to an initial numeric string, consisting of optional blank characters, optional minus sign, and zero or more digits with an optional radix character and thousands separators (as defined in the current locale), which is sorted by arithmetic value. An empty digit string is treated as zero. Leading zeros and signs on zeros do not affect ordering.
- r** Reverses the sense of comparisons.

### Field Separator Options

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

- b** Ignores leading blank characters when determining the starting and ending positions of a restricted sort key. If the **-b** option is specified before the first sort key option, it is applied to all sort key options. Otherwise, the **-b** option can be attached independently to each **-k** *field\_start*, *field\_end*, or *+pos1* or *-pos2* option-argument (see below).

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

### Sort Key Options

Sort keys can be specified using the options:

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

**-k** *field\_start* [*type*] [,*field\_end* [*type*] ]

where:

*field\_start* and *field\_end*

define a key field restricted to a portion of the line.

*type*

is a modifier from the list of characters **bdfiMnr**. The **b** modifier behaves like the **-b** option, but applies only to the *field\_start* or *field\_end* to which it is attached and characters within a field are counted from the first non-blank character in the field. (This applies separately to *first\_character* and *last\_character*.) The other modifiers behave like the corresponding options, but apply only to the key field to which they are attached. They 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 applies to either.

When there are multiple key fields, later keys are compared only after all earlier keys compare equal. Except when the **-u** option is specified, lines that otherwise compare equal are 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 notation:

**-k** *field\_start*[*type*][,*field\_end*[*type*]]

defines 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 missing *field\_end* means 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 has the form:

*field\_number*[,*first\_character*]

Fields and characters within fields are numbered starting with 1. *field\_number* and *first\_character*, interpreted as positive decimal integers, specify the first character to be used as part of a sort key. If *.first\_character* is omitted, it refers to the first character of the field.

The *field\_end* portion of the *keydef* option-argument has the form:

*field\_number*[*.last\_character*]

The *field\_number* is as described above for *field\_start*. *last\_character*, interpreted as a non-negative decimal integer, specifies the last character to be used as part of the sort key. If *last\_character* evaluates to zero or *.last\_character* is omitted, it refers to the last character of the field specified by *field\_number*.

If the **-b** option or **b** type modifier is in effect, characters within a field are counted from the first non-blank character in the field. (This applies separately to *first\_character* and *last\_character*.)

[*+pos1* [*-pos2*]]

(obsolete). Provide functionality equivalent to the **-kkeydef** option.

*pos1* and *pos2* each have the form *m.n* optionally followed by one or more of the flags **bdfiMnr**. A starting position specified by *+m.n* is interpreted to mean the *n*+1st character in the *m*+1st field. A missing *.n* means **.0**, indicating the first character of the *m*+1st field. If the **b** flag is in effect *n* is counted from the first non-blank in the *m*+1st field; *+m.0b* refers to the first non-blank character in the *m*+1st field.

A last position specified by *-m.n* is interpreted to mean the *n*th character (including separators) after the last character of the *m*th field. A missing *.n* means **.0**, indicating the last character of the *m*th field. If the **b** flag is in effect *n* is counted from the last leading blank in the *m*+1st field; *-m.1b* refers to the first non-blank in the *m*+1st field.

The fully specified *+pos1 -pos2* form with type modifiers **T** and **U**:

**+w.xT -y.zU**

is equivalent to:

```
undefined (z==0 & U contains b & -t is present)
-k w+1.x+1T,y.0U    (z==0 otherwise)
-k w+1.x+1T,y+1.zU  (z > 0)
```

Implementations support at least nine occurrences of the sort keys (the **-k** option and obsolescent *+pos1* and *-pos2*) which are significant in command line order. If no sort key is specified, a default sort key of the entire line is used.

**OPERANDS**

The following operand is supported:

*file*      A path name of a file to be sorted, merged or checked. If no *file* operands are specified, or if a *file* operand is `-`, the standard input is used.

**USAGE**

See **largefile(5)** for the description of the behavior of **sort** when encountering files greater than or equal to 2 Gbyte (  $2^{31}$  bytes).

**EXAMPLES**

In the following examples, first the preferred and then the obsolete way of specifying **sort** keys are given as an aid to understanding the relationship between the two forms.

**Example 1** Sorting with the Second Field as a sort Key

Either of the following commands sorts the contents of **infile** with the second field as the sort key:

```
example% sort -k 2,2 infile
example% sort +1 -2 infile
```

**Example 2** Sorting in Reverse Order

Either of the following commands 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):

```
example% sort -r -o outfile -k 2.2,2.2 infile1 infile2
example% sort -r -o outfile +1.1 -1.2 infile1 infile2
```

**Example 3** Sorting Using a Specified Character in One of the Files

Either of the following commands sorts the contents of **infile1** and **infile2** using the second non-blank character of the second field as the sort key:

```
example% sort -k 2.2b,2.2b infile1 infile2
example% sort +1.1b -1.2b infile1 infile2
```

**Example 4** Sorting by Numeric User ID

Either of the following commands prints the **passwd(4)** file (user database) sorted by the numeric user ID (the third colon-separated field):

```
example% sort -t : -k 3,3n /etc/passwd
example% sort -t : +2 -3n /etc/passwd
```

**Example 5** Printing Sorted Lines Excluding Lines that Duplicate a Field

Either of the following commands prints the lines of the already sorted file **infile**, suppressing all but one occurrence of lines having the same third field:

```
example% sort -um -k 3.1,3.0 infile
example% sort -um +2.0 -3.0 infile
```

**Example 6** Sorting by Host IP Address

Either of the following commands prints the **hosts(4)** file (IPv4 hosts database), sorted by the numeric **IP** address (the first four numeric fields):

```
example$ sort -t . -k 1,1n -k 2,2n -k 3,3n -k 4,4n /etc/hosts
example$ sort -t . +0 -1n +1 -2n +2 -3n +3 -4n /etc/hosts
```

Since '.' is both the field delimiter and, in many locales, the decimal separator, failure to specify both ends of the field leads to results where the second field is interpreted as a fractional portion of the first, and so forth.

**ENVIRONMENT VARIABLES**

See **environ(5)** for descriptions of the following environment variables that affect the execution of **sort**: LANG, LC\_ALL, LC\_COLLATE, LC\_MESSAGES, and NLSPATH.

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

**LC\_NUMERIC** Determine the locale for the definition of the radix character and thousands separator for the **-n** option.

**EXIT STATUS**

The following exit values are 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.

**FILES**

**/var/tmp/stm???** Temporary files

**ATTRIBUTES**

See **attributes(5)** for descriptions of the following attributes:

**/usr/bin/sort**

tab() box; cw(2.75i) |cw(2.75i) lw(2.75i) |lw(2.75i) ATTRIBUTE TYPEATTRIBUTE VALUE \_  
AvailabilitySUNWesu \_ CSIEnabled

**/usr/xpg4/bin/sort**

tab() box; cw(2.75i) |cw(2.75i) lw(2.75i) |lw(2.75i) ATTRIBUTE TYPEATTRIBUTE VALUE \_  
AvailabilitySUNWxcu4 \_ CSIEnabled \_ Interface StabilityStandard

**SEE ALSO**

**comm(1)**, **join(1)**, **uniq(1)**, **nl\_langinfo(3C)**, **strptime(3C)**, **hosts(4)**, **passwd(4)**, **attributes(5)**, **environ(5)**, **largefile(5)**, **standards(5)**

**DIAGNOSTICS**

Comments and exits with non-zero status for various trouble conditions (for example, when input lines are too long), and for disorders discovered under the **-c** option.

**NOTES**

When the last line of an input file is missing a **new-line** character, **sort** appends one, prints a warning message, and continues.

**sort** does not guarantee preservation of relative line ordering on equal keys.

One can tune **sort** performance for a specific scenario using the **-S** option. However, one should note in particular that **sort** has greater knowledge of how to use a finite amount of memory for sorting than the virtual memory system. Thus, a **sort** invoked to request an extremely large amount of memory via the **-S** option could perform extremely poorly.

As noted, certain of the field modifiers (such as **-M** and **-d**) cause the interpretation of input data to be done with reference to locale-specific settings. The results of this interpretation can be unexpected if one's expectations are not aligned with the conventions established by the locale. In the case of the month keys, **sort** does not attempt to compensate for approximate month abbreviations. The precise month abbreviations from **nl\_langinfo(3C)** or **strptime(3C)** are the only ones recognized. For printable or dictionary order, if these concepts are not well-defined by the locale, an empty sort key might be the result, leading to the next key being the significant one for determining the appropriate ordering.