NAME

sort – sort, merge, or sequence check text files

SYNOPSIS

/usr/bin/sort [**–cmu**] [**–o** *output*] [**–T** *directory*] [**–y** [*kmem*]] [**–z** *recsz*]
[**–dfiMnr**] [**–b**] [**–t** *char*] [**–k** *keydef*] [**+***pos1* [*–pos2*]] [*file* . . .]

/usr/xpg4/bin/sort [**–cmu**] [**–o** *output*] [**–T** *directory*] [**–y** [*kmem*]] [**–z** *recsz*]
[**–dfiMnr**] [**–b**] [**–t** *char*] [**–k** *keydef*] [**+***pos1* [*–pos2*]] [*file* . . .]

AVAILABILITY

/usr/bin/sort

SUNWesu

/usr/xpg4/bin/sort

SUNWxcu4

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 Check 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 curcumstances.

–m Merge only. The input files are assumed to be already sorted.

–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.

–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 input files.

–T *directory*
The *directory* argument is the name of a directory in which to place temporary files.

–y *kmem* The amount of main memory initially used by **sort**. If this option is omitted, **sort** begins using a system default memory size, and continues to use more space as needed. If *kmem* is present, **sort** will start using that number of Kbytes of memory, unless the administrative minimum or maximum is exceeded, in which case the corresponding extremum will be used. Thus, **–y 0** is guaranteed to start with minimum memory. **–y** with no *kmem* argument starts with maximum memory.

–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 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 will affect 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 Fold lower-case letters into upper case.

–i Ignore non-printable characters.

–M Compare 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 Restrict 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 will be sorted by arithmetic value. An empty digit string is treated as zero. Leading zeros and signs on zeros do not affect ordering.

–r Reverse the sense of comparisons.

Field Separator Options

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

–b Ignore 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 –**k** *keydef* 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*.1**b** 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 will be used.

EXAMPLES

In the following examples, non-obsolescent and obsolescent ways of specifying **sort** keys are given as an aid to understanding the relationship between the two forms.

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
```

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
```

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

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

example% −t : −k 3,3n /etc/passwd
example% −t : +2 −3n /etc/passwd

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

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **sort**: **LC_COLLATE**, **LC_MESSAGES**, and **NLSPATH**.

LC_CTYPE Determine the locale for the behaviour 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

SEE ALSO

comm(1), **join**(1), **uniq**(1), **environ**(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.