# Sort Command Examples

Sort command in unix or linux system is used to order the elements or text. Sort command has the capability of sorting numerical values and strings. The sort command can order the lines in a text file.

The syntax of sort command is:

```
sort [options] filename
```

The options are:

```
-b : Ignores leading spaces in each line
-d : Uses dictionary sort order. Conisders only spaces and alphanumeric
characters in sorting
-f : Uses case insensitive sorting.
-M : Sorts based on months. Considers only first 3 letters as month. Eg: JAN,
FEB
-n : Uses numeric sorting
-R : Sorts the input file randomly.
-r : Reverse order sorting
-k : Sorts file based on the data in the specified field positions.
-u : Suppresses duplicate lines
-t : input field separator
```

**Sort Command Examples**:

Before practicing the examples create the below two files in your unix system:

```
> cat order.txt
Unix distributed 05 server
Linux virtual 3 server
Unix distributed 05 server
Distributed processing 6 system

> cat delim_sort.txt
Mayday|4
Janmon|1
Declast|12
```

1. Sorting lines of text

The default sort command uses alphabetical order (ASCII order) to sort the file. It treats each line as a string and then sorts the lines.

```
> sort order.txt
Distributed processing 6 system
Linux virtual 3 server
Unix distributed 05 server
Unix distributed 05 server
```

2. Sorting based on the field positions.

You can specify the field postions using the -k option of sort command. The sort command uses the space or tab as the default delimiter. To sort based on the data in the second field, run the below command:

```
> sort -k2 order.txt
Unix distributed 05 server
Unix distributed 05 server
Distributed processing 6 system
Linux virtual 3 server
```

You can also pecify more than field with k option as a comma separated list. The below command uses the second and fourth fields to sort the data.

```
> sort -k2,4 order.txt
```

3. Numeric sorting

Instead of the default alphabetical sorting order, you can make the sort command to sort in numeric order using the -n option. This is shown below:

```
> sort -nk3 order.txt
Linux virtual 3 server
Unix distributed 05 server
Unix distributed 05 server
Distributed processing 6 system
```

4. Sort in reverse order

By default, the sort command sorts the data in ascending order. You can change this to descending order using the -r option.

```
> sort -nrk3 order.txt
Distributed processing 6 system
Unix distributed 05 server
Unix distributed 05 server
Linux virtual 3 server
```

5. Suppressing duplicates or Print only unique values

You can produce only unique values in the output using the - u option of the sort command.

```
> sort -u order.txt
Distributed processing 6 system
Linux virtual 3 server
Unix distributed 05 server
```

Another way is piping the output of sort command to uniq command.

```
> sort order.txt | uniq
```

6. Delimited file input

In the second, third and fourth examples we have sorted the data based on the field positions. Here the fields are separted by space or tab character. What if the fields are specifed by any other character? In such cases, we have to specify the input delimiter with the -t option. An example is shown below:

```
> sort -t'|' -nrk2 delim_sort.txt
Declast|12
Mayday|4
Janmon|1
```

7. Sorting on months.

We can sort the data in the monthwise using the -M option of the sort command. This is shown below:

```
> sort -M delim_sort.txt
Janmon|1
Mayday|4
Declast|12
```

Treats the first 3 characters in the string as month and then sorts in months order.

r its first and second arguments; it is joining the sorted contents of both files and gives results similar to the below results.

```
1 onion mango
2 pepper grapefruit
3 tomato apple
4 beet orange
```

## 15 examples of sort command in Linux

**sort** command is used to sort a file, arranging the records in a particular order. By default, the sort command sorts file assuming the contents are ascii. Using options in sort command, it can also be used to sort numerically. Let us discuss it with some examples:

**File with Ascii data:**
Let us consider a file with the following contents:

```
$ cat file
Unix
Linux
Solaris
AIX
Linux
HPUX
```

**1. sort simply sorts the file in alphabetical order**:

```
$ sort file
AIX
HPUX
Linux
Linux
Solaris
Unix
```

All records are sorted alphabetically.

**2. sort removes the duplicates** using the -u option:

```
$ sort -u file
AIX
HPUX
Linux
Solaris
Unix
```

The duplicate 'Linux' record got removed. '-u' option removes all the duplicate records in the file. Even if the file have had 10 'Linux' records, with -u option, only the first record is retained.

**File with numbers**:
Let us consider a file with numbers:

```
$ cat file
20
19
5
49
200
```

**3.** The default sort 'might' give incorrect result on a file containing numbers:

```
$ sort file
19
20
200
49
5
```

In the above result, 200 got placed immediately below 20, not at the end which is incorrect. This is because the sort did  ASCII sort. If the file had not contained '200', the default sort would have given proper result. However, it is incorrect to sort a numerical file in this way since the sorting logic is incorrect.

**4. To sort a file numericallly**:

```
$ sort -n file
5
19
20
49
200
```

-n option can sort the decimal numbers as well.

**5. sort file numerically in reverse order**:

```
$ sort -nr file
200
49
20
19
5
```

'r' option does a reverse sort.

**Multiple Files**:
Let us consider examples with multiple files, say file1 and file2, containing numbers:

```
$ cat file1
20
19
5
```

```
49
200
$ cat file2
25
18
5
48
200
```

**6. sort can sort multiple files** as well.

```
$ sort -n file1 file2
5
5
18
19
20
25
48
49
200
200
```

The result of sort with multiple files will be a sorted and merged output of the multiple files.

**7. Sort, merge and remove duplicates**:

```
$ sort -nu file1 file2
5
18
19
20
25
48
49
200
```

-u option becomes more handy in case of multiple files. With this, the output is now sorted, merged and without duplicate records.

**<u>Files with multiple fields and delimiter</u>**:

Let us consider a file with multiple fields:

```
$ cat file
Linux,20
Unix,30
AIX,25
Linux,25
Solaris,10
HPUX,100
```

**8. sorting a file containing multiple fields**:

```
$ sort file
AIX,25
HPUX,100
Linux,20
```

```
Linux,25
Solaris,10
Unix,30
```
As shown above, the file got sorted on the 1st field, by default.

**9. sort file on the basis of 1st field**:

```
$ sort -t"," -k1,1 file
AIX,25
HPUX,100
Linux,20
Linux,25
Solaris,10
Unix,30
```
This is being more explicit. '-t' option is used to provide the delimiter in case of files with delimiter. '-k' is used to specify the keys on the basis of which the sorting has to be done. The format of '-k' is : '-km,n' where *m* is the starting key and *n* is the ending key. In other words, sort can be used to sort on a range of fields just like how the group by in sql does. In our case, since the sorting is on the 1st field alone, we speciy '1,1'. Similarly, if the sorting is to be done on the basis of first 3 fields, it will be: '-k 1,3'.

Note: For a file which has fields delimited by a space or a tab, there is no need to specify the "-t" option since the white space is the delimiter by default in sort.

**10. sorting file on the basis of the 2nd field**:

```
$ sort -t"," -k2,2 file
Solaris,10
HPUX,100
Linux,20
AIX,25
Linux,25
Unix,30
```

**11. sorting file on the basis of 2nd field , numerically**:

```
$ sort -t"," -k2n,2 file
Solaris,10
Linux,20
AIX,25
Linux,25
Unix,30
HPUX,100
```

**12. Remove duplicates from the file based on 1st field**:

```
$ sort -t"," -k1,1 -u file
AIX,25
HPUX,100
Linux,20
Solaris,10
Unix,30
```
The duplicate Linux record got removed. Keep in mind, the command "sort -u file" would not have worked here becuase both the 'Linux' records are not same, the values were different. However, in the

above, sort is told to remove the duplicates based on the 1st key, and hence the duplicate 'Linux' record got removed. According to sort, in case of a group of similar records, except the first one, the rest are considered duplicate.

**13. Sort the file numerically on the 2nd field in reverse order**:

```
$ sort -t"," -k2nr,2 file
HPUX,100
Unix,30
AIX,25
Linux,25
Linux,20
Solaris,10
```

**14. sort the file alphabetically on the 1st field, numerically on the 2nd field**:

```
$ sort -t"," -k1,1 -k2n,2 file
AIX,25
HPUX,100
Linux,20
Linux,25
Solaris,10
Unix,30
```

**15. sort a file based on the 1st and 2nd field, and numerically on 3rd field** on  a file containing 5 columns:

```
$ sort -t"," -k1,2 -k3n,3 file
```