CMsort

User Manual

Revision: 2.1b
Date: 2020-06-30

Contents

Notes
In this documentation, trademarks are used without further notice. Usage of a trademark without the ® sign does not mean that the name can be used without restrictions.
All examples contain fictional data. Any similarity to real names, persons, companies, data etc. are purely coincidental.

1 Overview

CMsort is a 32-bit command line tool to sort text files with DOS/ Windows, Linux/ Unix, MAC, or mixed end-of-line marks. It can also handle CSV files.

CMsort allows you to define one or more sort keys. Each field can independently have an ascending (default) or descending sort order. The concatenation of all partial sort keys builds the complete sort key (superkey) for a record.

In contrast to old 16-bit DOS applications, CMsort can be executed on both 32 and 64-bit Versions of Microsoft Windows.

The versions 2.0 and above offer increased speed, enable sorting of files greater than 2 GB and have the additional capability to sort CSV files.

Technical Information

- As a 32 bit application, CMsort runs under both 32 and 64 bit versions of Microsoft Windows XP/ Vista/ 7 /8.x/ 10.
- The file size is limited only by the operating system and/or the available disk size.
- The line (i.e. record) count must be less or equal to $2^{63} - 1 = 9{,}223{,}372{,}036{,}854{,}775{,}807$[a]
- The length of a single line must be less or equal to 32 KB = 32,768 bytes (including end-of-line marks).
- The quicksort algorithm is used for sorting, which is *not stable*, i.e. the *order* of lines with *identical* sort keys in the output file *may differ* from the order within the input file.
- The case insensitive sort is done by converting all characters to uppercase according to the current Windows locale. Then the comparison is done evaluating the 8-bit ordinal value of each character (refer to chapter 5).
- CMsort is built with Free Pascal and Lazarus.[b]

CMsort is freeware. You can use it as long as you accept my Licence Agreement and Copyright (see below).

www.chmaas.handshake.de
chmaas@handshake.de

If you like CMsort, want to appreciate my work, and/or support the development, I would be pleased to receive a donation from you via PayPal. Click to the link below to make a donation:

http://www.chmaas.handshake.de/delphi/freeware/cmsort/cmsort.htm

[a] Due to the record counter.
[b] The sourcecode of the current version has been successfully compiled into a 64 bit executable. But at the moment, this offers no advantage, especially concerning speed (on the contrary, execution is about 15% slower).

2 Installation

CMsort is a 32 bit console application and requires Windows XP, Vista, 7, 8.x or 10 (32 or 64 bit versions).

No special installation procedure is required, i.e. CMsort is a *portable application*.

Simply create a directory e.g. `C:\CMsort` or `C:\PortableApps\CMsort`.

> Note for Windows Vista and above: Don't create a directory below the directory `C:\Program Files` (like `C:\Program Files\CMsort`).

Now simply unzip the archive file CMsort.zip into this directory: You'll see the following files:

CMsort.exe	the executable
CMsort.pdf	this user manual
README.TXT	

No data will be written into the registry, no DLLs are copied to your hard disk.

3 Uninstallation

To uninstall CMsort completely, simply delete the directory and all containing files.

4 Version history

Versions of CMsort prior 2.0 were developed with Delphi. These versions were restricted to file sizes less than 2 GB[c] and could not handle CSV files. The first version 1.0 was released on 2001-02-11, the last version 1.71 on 2011-04-08.

In 2013, development was moved to Free Pascal (http://www.freepascal.org/) and Lazarus (http://www.lazarus.freepascal.org/).

4.1 Version history of the program

	Release date	Features/Bugfixes
2.0	2013-09-24	• Increased speed (up to 10 times faster for small files, up to 3 times faster for large files) • File size > 2 GB possible • Processing of CSV files • `/X` option to exclude records with a given character in a given column
2.01	2014-06-23	• Any error now causes errorlevel 1 (before, it was errorlevel 0). • Option `/B` now works with CSV files, too. In CSV mode, both empty or blank lines and empty CSV records (e.g. containing only separator, quotation sign and blanks) are discarded. • Elapsed time shown as hh:mm:ss (instead of mm:ss) • For CSV files, empty fields can be sorted with `F=1` and `L=0` (see below for details).
2.02	2014-09-08	• Bugfix: Case insensitive sorting in descending mode (only this combination) didn't work correctly. All versions since 1.0 are affected, so please upgrade to version 2.02.

[c] Due to the record counter.

2.1	2019-10-27	• Bugfix: In CSV mode, case insensitive sorting in descending mode (only this combination) didn't work correctly (same problem as solved with v2.02 for fixed-record[d] files). All versions since 2.0 are affected, so please upgrade to version 2.1.
		• Bugfix: An end-of-file (EOF) mark *within* the file (i.e. followed by additional records) resulted in an endless loop (EOF marks at the end of the file were handled properly). Although EOF marks should normally only be present at the end of a file, this is now fixed. This case results in a warning that all records after EOF were ignored.
		• Bugfix: Sorting with `/C=F,0-` or `/S=F,0-` (i.e. length 0 in combination with descending sort order) could produce wrong results if the records did not have the same length (sorting of *CSV files* was *not* affected). Explanation: The part keys according to `/C=F,0-` or `/S=F,0-` can have different length. On the other hand, the creation of a superkey makes it impossible to "invert" the "whole" comparison of two superkeys (by swapping them or by multiplying the result of e.g. the `CompareStr` function by `-1`). Rather, each part key in descending order has to be "inverted" separately; but this produces correct results only as long as each part key has the same length.
		Therefore, these parameters are now prohibited. To fix this problem and to enable sorting of files which could not be sorted prior to this version, the following new features have been introduced:
		  ○ New parameters for "simple" sorts using complete line as keys (refer to 5.2.1).
		  ○ New handling of non-numeric keys `/S=F,L` and `/C=F,L` with a value `F` beyond end of record or `L` too large: These cases will no longer raise an error. Instead, the (part-) key is filled up with binary zeroes (if needed, completely build up by binary zeroes). The first occurrence will produce a warning.
		• Error/warning messages now include line numbers.
		• `/V` option: Correction in documentation (default separator is semicolon, not comma).
		• `/V=S` alone allowed, e.g. `/V=$2C` for comma as separator (instead default semicolon). In this case, the double quotation mark (default) is used as quotation sign.
		• Numeric sort keys for CSV files: documentation corrected.
		• Special case for sorting CSV files allowed: `/SV=I,1,0` (i.e. `F=1`, `L=0`) is valid, even if I-th CSV field is always empty. This case produces a warning that this field has no impact on the sort key.
		• Elapsed time with milliseconds.

[d] Fixed-record files are no longer supported, refer to section 4.2

4.2 Version history of the manual (only for the current version)

	Release date	Notes/Corrections
2.1a	2019-10-27	• Initial version of manual.
2.1b	2020-06-30	• Support for files with fixed-length records (`/F` option) had to be removed on 2018-03-10 (i.e. between the versions 2.02 and 2.1). This is now correctly reflected within the manual. Explanation for removing the `/F` option: A built-in library function is internally calling another function, but the types of parameters are inconsistent. This caused an endless loop after reading 0,5*2^32 = 2^31 = 2147483648 records. • Explanation of the `/C-` option corrected (section 5.2.1)

5 Usage

The general syntax to invoke CMsort is:

```
CMsort [sort key][-] ... [option] ... <input file> <output file>
```

> Options and sort keys must begin with either a / (slash) or a – (minus sign). In this documentation, only the slash is used.

If a file name contains spaces, it must be enclosed within double quotation marks:

```
cmsort "my file unsorted.txt" "my file sorted.txt"
```

The input and/or output file may be located in a different directory:

```
cmsort C:\input\data.txt C:\output\data.sor
```

You can use relative paths:

```
cmsort .\input\data.txt .\output\data.sor
```

The comparison is based on the 8-bit ordinal value of each character. This means especially the following ascending order of digits and letters:

1. digits 0 to 9
2. capital letters A to Z
3. small letters a to z

> Note: In case sensitive order, capital letters have a “lower” value than lower case (this may differ from lexical sort order).

If the output file already exists, it will be overwritten without confirmation.

The return code of CMsort is 0 (zero) if the program terminates normally. Otherwise, the return code is 1. This is useful especially within CMD files, where you can evaluate the return code as follows:

```
cmsort /SV=3,1,0 /NV=4,1,0- /V /D /H=1 flo3.in flo3.sor
IF ERRORLEVEL 1 GOTO ERROR
rem everything is ok
...
:ERROR
rem below, an error can be treated
```

5.1 Understanding numeric sort keys

To avoid problems with converting numbers (e.g. large numbers or country specific settings) and to improve speed, the numeric interpretation *doesn’t convert the field value into a number*. Internally, numbers are treated as strings with a special handling for negative numbers.

It doesn't matter if the decimal point is a comma and the thousand separator is a point (i.e. 1,000.00 and 1.000,00 are equivalent). The only condition is that the characters used for decimal point and thousand separator are the *same* for *each* sort field (in other words: *different* sort fields can have *different* representations)

5.2 Sort keys for files with fixed-length fields

5.2.1 Simple sort keys (use *exactly one* of the following possibilities)

Parameter	Explanation
	Sort key is complete line case sensitive ascending (no sort key specified; this is equivalent to `/S`)
`/S`	Sort key is complete line case sensitive ascending (default)
`/C`	Sort key is complete line case insensitive ascending
`/S-`	Sort key is complete line case sensitive descending
`/C-`	Sort key is complete line case insensitive descending

5.2.2 Complex sort keys (combinations possible)

Parameter	Explanation
`/S=F,L`	case sensitive string from position F with length L ascending
`/C=F,L`	case insensitive string from position F with length L ascending
`/N=F,L`	interpreted numeric from position F with length L ascending
`/S=F,L-`	case sensitive string from position F with length L descending
`/C=F,L-`	case insensitive string from position F with length L descending
`/N=F,L-`	numeric from position F with length L descending

`F` is counted from 1, the beginning of the record. `L` is the number of characters used for the sort key. `L=0` is allowed only for the last non-numeric sort key and means "until end of record"; but `L=0` is prohibited in combination with descending sort order, i.e. `/C=F,0-` or `/S=F,0-` are invalid.

If **no sort key** is indicated, **the complete line** is used as **case sensitive sort key**. In this case, the records can have different length.

The **case insensitive** sort is based on the **ANSI** character table (all letters will be converted into **uppercase** using the **current Windows locale**). Therefore, is **not possible to sort UTF-8 files** (doing so will normally produce wrong results).

For **numeric sort keys**, under any circumstances, there must be **`L` characters present at position `F`**.

For non-numeric keys, the following rules apply:

- If there is **no valid character present at position `F`**, the complete part key will consist of **binary zeroes** (the first occurrence will trigger a warning).
- If there are **less than `L` characters present**, the part key will be **filled up with binary zeroes** (the first occurrence will throw a warning).
- The usage of `L=0` for the last non-numeric sort key will **not result in filling up the key with binary zeroes**.

5.2.3 Fixed-length fields and numeric sort keys

The following prerequisites must be met for all numbers belonging to the same sort field:[e]

- Numbers must be *aligned at the decimal point:*
  ```
  123.4
   56.78
    9.0
  ```
- If a decimal point is present within one number, it must be present within all numbers (but the precision of the numbers may be different):
  1.7
  1.55
  1.
- If only integer numbers without decimal point are present, they must be properly aligned.
  ```
  123
   45
    6
  ```
- If (at least) *one* number has one or more thousand separator(s), *all* numbers having a value qualifying for thousand separators must have them at the given position(s):
  ```
  16,384
     512
   1,048
  ```
- Leading zeroes are allowed (obviously, leading blanks must be present as long as necessary for correct alignment of the decimal point).
- Leading zeroes and leading blanks can be “mixed”
  ```
  001
    2
   03
  ```
- Negative numbers must contain a minus sign in three possible styles (the style must not be unique for all numbers within the file or a record, i.e. every number can have its own style):
  - at the end of the field
  - in the first column of the field
  - anywhere left from the first significant digit
- Positive numbers can contain a plus sign (syntax is the same as for the minus sign).

Refer to the appendix for examples.

[e] In the following examples, all lines have the same length, i.e. if necessary, they are filled up with trailing spaces.

5.3 Sort keys for CSV files

If a field is enclosed within quotes, only the field value itself (e.g. without the leading and trailing quote sign) is used to build the sort key. If a field *contains the separator* char (and the separator char is printable), the field *must* be enclosed within quotes (otherwise, the field can be enclosed or not). If the separator char is not printable (e.g. TAB), the field cannot contain the separator char. The output file contains the record in the same CSV format as the input file, i.e. the quotation is untouched.

A quotation mark of the same type as used for parsing the CSV file must not be present within the field value itself.

Parameter	Explanation
`/SV=I,F,L`	I-th CSV field as string from position F with length L (case sensitive) ascending
`/CV=I,F,L`	I-th CSV field as string from position F with length L (case insensitive) ascending
`/NV=I,F,L`	I-th CSV field numeric from position F with length L ascending
`/SV=I,F,L-`	I-th CSV field as string from position F with length L (case sensitive) descending
`/CV=I,F,L-`	I-th CSV field as string from position F with length L (case insensitive) descending
`/NV=I,F,L-`	I-th CSV field numeric from position F with length L descending

The CSV fields are counted beginning with 1 for the first field (i.e. use `I=1` for the first field). The position `F` within the field is counted from 1 (the first character of the field); if the field is enclosed within quotes, the quotes itself don't count.

`L` is the number of characters used for the partial sort key. `L=0` is allowed for all sort keys and means "until end of field". In this case, the file will be preprocessed to determine the maximum number of characters within the key (this information is used to auto-align the key).

Normally (like for plain text files), at position `F` at least one character must be present. There is one exception: empty fields can be sorted with `F=1` and `L=0`; in this case, the field is treated as a field containing as much blanks as the widest field contains.

`F=1` and `L=0`: If the appropriate field is always empty, this case is allowed, but produces a warning (because this field will have no impact on the sort key).

`F>1` and `L=0`: If the length of the field is less than `F`, an error is raised (because the field doesn't contain at least one significant character). Otherwise, the field is filled up with blanks to reach the maximum width. Then a part key is built with characters from `F` until the end of the filled-up field.

The complete line (as it is with all characters including separation and quotation signs) is used as sort key if no sort keys are indicated, i.e. the file is sorted the same way like a file with fixed-length fields without sort keys. It's up to you to decide if this makes sense with a given CSV file.

5.3.1 Numeric sort keys for CSV files

If `L` is greater than 0, the handling of numeric sort keys is the same as for fixed-length fields, i.e. numbers must be aligned at the decimal point.

If `L=0`, the following conditions must be met for all numbers belonging to the same field:

- All numbers must have the *same precision* (i.e. the same count of digits after the decimal point). **Note:** the precision of *different* fields (i.e. different sort keys) may be different (but for one field, it must be identical).
- If one number has a decimal point, it must be present within all numbers (even if it is an integer number).
- If all numbers are integers, a decimal point is not required (but in this case, no number must have a decimal point).
- All negative numbers must contain a *minus sign at the same position* (either a trailing minus sign, or a leading minus sign preceding the leftmost digit).
- Positive numbers can contain a plus sign preceding the leftmost digit, but a *trailing plus sign is not allowed*. **Exception**: If all numbers have an appropriate trailing sign (i.e. all negative numbers have a trailing minus, all positive numbers have a trailing plus sign), the sorting will be correct.

Leading zeroes and leading blanks are allowed.

For thousand separators, the same rules as for fixed-length fields apply.

Refer to the appendix for examples.

5.4 Options

Option	Explanation
/B	Ignore blank or empty records (i.e. empty lines or lines containing only blanks). Such records are discarded, i.e. neither sorted neither written. For CSV files, both empty/ blank lines and empty CSV records (i.e. lines containing only separator, quotation sign and blanks) are treated as empty records. The /B switch will *not discard* records with keys filled up with binary zeroes (refer to 5.2.2), even if the key effectively used for comparison consists solely of binary zeroes.
/D	Ignore records with duplicate keys. Duplicate refers to *all* given sort keys, i.e. to records with the *same superkey*.
/D=<file>	Ignore records with duplicate keys, and write them to <file>.
/H=n	Don't sort the first n lines (default: n=0). If n is greater 0, the output contains the first n (unsorted) lines of the input file, followed by sorted lines.
/M=n	Use n KB memory (n >= 1,024 KB = 1 MB; default: 65,536 KB = 64 MB) Note: The exact amount of memory used may slightly differ from this value (in general, it should be less).
/Q	Quiet mode (no progress output).
/T=<path>	Set path for temporary files (use /T=TMP for Windows temporary file path). Otherwise, all temporary files are created in the current directory, i.e. the directory from where CMSort is called.
/V	Treat input file as CSV file with semicolon as separator and double quotation mark as quotation sign. With this option, only the sort keys /SV, /CV, /NV are allowed.
/V=S	Like before, but use the specified char S as separator. S must be indicated as $<hex> or #<decimal>, e.g. /V=$3B for default separator semicolon or /V=$2C (or equivalent /S=#44) for comma as separator. As quotation sign, the default (double quotation mark) is used.
/V=S,Q	Like before, but use S as separator and Q as quotation sign. S, Q must be indicated as $<hex> or #<decimal> (e.g. for default values semicolon/ double quotation mark, use /V=$3B,$22 which is equivalent to /V=#59,#34). Use $00 for Q if no quotation sign is used (e.g. /V=$2C,$00 for comma as separator, but no double quotation sign).
/W=n	Use n-way-merge of temporary files (2<=n<=5, default: n=5). Note: This switch normally has no impact on speed, i.e. there should be no need to change the default value.
/X=n,c	Exclude all lines containing char c (indicated as char, $<hex>, #<dec>) in column n (indicated as normal decimal integer).

The option /X can be used for CSV files, even if this should normally make no sense, because n is referring to the column within the complete line (not referring to the CSV field or a specific column of a specific CSV field).

6 Examples

6.1 Example for fixed-length fields

Assume you have the following input file customer.txt (fixed-length fields with two header lines):

```
12345678901234567890123456789012345678901234567890123
CustNo  Name         OrderDate       Return
1004711 Miller & Co. 1999-12-06    1,207.23
1004713 Topsoft      2000-01-04    2,521.95
1004747 MCP & Co.    2000-01-04    7,356.88
1004799 Eftpos       1999-12-06   23,122.56
```

To sort this file by *order date ascending* and by *return descending* without sorting the header lines, use:

```
cmsort /S=22,10 /N=33,11- /H=2 customer.txt customer.sor
```

The meaning of the parameters is:

`/S=22,10`	First part of key is a case sensitive string beginning at position 22, length 10, sort ascending (default)
`/N=33,11-`	Second part of key is numeric, beginning at position 33, length 11, sort descending (-)
`/H=2`	Don't sort the first two lines (i.e. lines 1 and 2), but include these two lines to the output file

The result will be the following file:

```
12345678901234567890123456789012345678901234567890123
CustNo  Name         OrderDate       Return
1004799 Eftpos       1999-12-06   23,122.56
1004711 Miller & Co. 1999-12-06    1,207.23
1004747 MCP & Co.    2000-01-04    7,356.88
1004713 Topsoft      2000-01-04    2,521.95
```

6.2 Example for ignoring records with duplicate keys

Duplicate records are recognized by the defined key(s), not by the whole line. If you want to exclude identical lines, you must perform an additional sort beforehand by using the whole line as sort key. The following log file is containing user ID, user name, and last access time:

```
055 Maas       2001-02-05 07:31:55
087 Mechenbier 2001-02-05 08:01:23
024 Hesselbein 2001-02-05 08:15:16
055 Maas       2001-02-05 08:44:24
089 Kruft      2001-02-05 09:05:07
087 Mechenbier 2001-02-05 09:31:13
```

Command line:

```
cmsort /S=1,3 /D log.txt log.sor
```

Result:

```
024 Hesselbein 2001-02-05 08:15:16
055 Maas       2001-02-05 08:44:24
087 Mechenbier 2001-02-05 09:31:13
089 Kruft      2001-02-05 09:05:07
```

6.3 Example for CSV files

Input file:

```
"Customer No";CustName;OrderDate;Return
1004711;Miller & Co.;1999-12-06;1,207.23
"1004713";"Topsoft";"2000-01-04";"2,521.95"
1004747;MCP & Co.;2000-01-04;7,356.88
1004799;Eftpos;1999-12-06;23,122.56
```

The quotation sign is not required unless a field contains the separation char. The presence of the quotation sign may be different for each field and/or record without having influence on the sorting.

The first line is a "header line" that must not be sorted (but should appear as first line within the output line; so we use `/H=1`). The command line to sort by *order date ascending* and by *return descending* is as follows:

```
cmsort /SV=3,1,0 /NV=4,1,0- /V /H=1 customercsv.txt customercsv.sor
```

Result:

```
"Customer No";CustName;OrderDate;Return
1004799;Eftpos;1999-12-06;23,122.56
1004711;Miller & Co.;1999-12-06;1,207.23
1004747;MCP & Co.;2000-01-04;7,356.88
"1004713";"Topsoft";"2000-01-04";"2,521.95"
```

6.4 Example for empty fields in CSV files

Input file `csvempty.in`:

```
Euclid;;Greek
Thales;;Greek
Banach;Stefan;Polish
de Fermat;Pierre;French
Cantor;Georg;German
```

You can sort this file according to the second CSV field, although it's empty in some records:

```
cmsort /V /SV=2,1,0 csvempty.in csvempty.sor
```

The resulting file `csvempty.sor` will contain

```
Thales;;Greek
Euclid;;Greek
Cantor;Georg;German
de Fermat;Pierre;French
Banach;Stefan;Polish
```

In this example, the sort key `/SV=2,2,0` is not allowed (or any other key with F>1), like explained above.

This example shows also the fact that the sorting algorithm (quicksort) is not stable, so the order of the first two records (with the same empty sort key) may not be preserved.

6.5 Examples for numeric sort keys for files with fixed-length fields

Alignment

OK	Comment	Wrong	Comment
`  0.01`	All numbers have	`  0.01`	Numbers not aligned
`  0.2`	a decimal point	`   0.2`	at decimal point.
`10.`	and are aligned at	`10`	Number without
`  .05`	this decimal		decimal point (even if
`  0.0`	point.		aligned) is wrong,
`00.0`	Trailing zeroes		because at least
`0010.0`	allowed.		one other number
or			has a decimal point.
`   0`	All numbers are		
`   1`	integers without		
`  10`	decimal point and		
`  100`	are properly aligned.		
`   00`	Trailing zeroes		
`000001`	allowed.		

Thousand separator

OK	Comment	Wrong	Comment
`    0.01`	No thousand	`  1000.99`	Thousand separator
`     .2`	separator	`1,000.99`	not present within
`   12.34`	present	`10100.34`	all numbers
`1000.99`	within		(even if correct
`1200300.50`	all records.		alignment at
or			decimal point).
`1,000.99`	Always		
`10,200.50`	thousand		
`   12.34`	separator		
`1,000.99`	present		
`1,200300.50`	where needed.		
or			
`1.000,99`	Decimal point		
`   12.34`	is comma.		

Sign

OK	Comment
`    1.23`	
`   -1.24`	
`    1.25-`	
`-   1.26`	
`-000001.27`	
`   +1.24`	
`    1.24+`	
`+000001.25`	

6.6 Examples for numeric sort keys for CSV files

The following examples apply for `L=0` (otherwise, the same rules as for fixed-length fields are valid).

Precision			
OK	**Comment**	**Wrong**	**Comment**
`0.01`	All	`0.01`	Numbers
`0.20`	numbers	`0.2`	not with
`10.00`	have	`1.`	same precision.
`.05`	same		
`1000.10`	precision.	`0.01`	Not with same
`00.01`	Trailing zeroes	`0.020`	precision!
`.01`	and/or blanks		
`00.01`	allowed.		
or			
`1`	All numbers	`1.0`	Integers with and
`10`	are	`10`	without decimal
`100`	integers.	`100.`	point mixed.
`01`	Trailing zeroes		
`1`	and/or blanks		
`01`	allowed.		
or			
`1,200,300.00`	Thousand	`1,000.00`	Numbers with and
`1,000.00`	separators can be	`2,000.00`	without thousand
`0.00`	used.	`3000.00`	separator mixed.
or			
`1.000,00`	Decimal point		
`0,00`	is		
`1.200.300,00`	comma.		

Sign

OK	**Comment**	**Wrong**	**Comment**
+0.01	All	2.01+	Trailing/leading
-0.01	numbers	2.01-	minus sign mixed,
-0.02	have	-2.02	trailing plus sign.
+0.02	leading sign.		
or			
0.01+	All		
0.01-	numbers		
0.02-	have		
0.02+	trailing sign.		
or			
0.01	Minus sign		
-0.01	always leading,		
-0.02	plus sign omitted		
+0.02	or leading.		
or			
0.01	Minus sign		
0.01-	always trailing,		
0.02-	plus sign omitted		
+0.02	or leading.		

7 Appendix

7.1 EOL marks

CMsort is able to sort text files with DOS/ Windows, Linux/ Unix, MAC, or mixed end-of-line marks (EOL). The following table shows the possible EOL marks.

EOL mark	Meaning	Operating systems (examples)
CR	Carriage return '\r' 0x0D (13 in decimal)	Commodore 8-bit machines, ZX Spectrum, TRS-80, Apple II family, Mac OS up to version 9/ OS-9
LF	Line feed '\n' 0x0A (10 in decimal)	Unix and Unix-like systems (Linux, Mac OS X, FreeBSD, AIX, Xenix, etc.), BeOS, Amiga, RISC OS
CR+LF	CR followed by LF '\r\n' 0x0D0A	MS-Windows, MS-DOS, CP/M, Atari TOS, OS/2
LF+CR	LF followed by CR '\n\r' 0x0A0D	Acorn BBC

CMsort enables you to sort on a Windows machine files written under other operating systems or files with mixed EOL marks (generated e.g. by concatenating files of different origin).

For an input file with (whatever) EOL marks, the resulting (sorted) file will always have EOL marks in the CR+LF format (Windows/ DOS), i.e. the original EOL marks may not be preserved.

7.2 Possible file structure of input files

The following table shows the possible file structures.

EOL mark	Record structure	Options used

Files with EOL marks (EOL marks will automatically be detected)	Fixed-length fields	`/S=F,L`
		`/C=F,L`
		`/N=F,L`
		`L=0` allowed only for the last non-numeric sort key (but not in combination with descending sort order)

	CSV	`/SV=I,F,L`
		`/CV= I,F,L`
		`/NV= I,F,L`
		`/V` or `/V=S,Q` (required)
		`L=0` allowed for all sort keys

For CSV files, you can specify the separation char and the quotation sign with the option `/V=S,Q` (see above). The default values are semicolon and double quotation mark (simply use `/V` for these default values).

Recommendations for large input files
If you have influence on the creation of the input files, you should prefer files with fixed-length fields, because they will be sorted faster than CSV files.

If you must deal with CSV files, create them with fixed-length fields when possible. If you can avoid the usage of a quotation sign, you can sort them like non-CSV files with fixed-length fields. Otherwise avoid using `L=0`, which will also make sorting faster.

If possible, generate CSV files with TAB (0x09 or decimal 09) as separator and without quotation mark (in this case, option `/V=$09,$00` can be used). This has nothing to do with speed, but it ensures that under all circumstances,

- the separation char will not be present within any data field;
- the quotation sign can be used without restrictions within any data field.

7.3 Sort keys in detail

CMsort allows the definition of one or more sort keys. Each sort key has a

- type (S for string case sensitive, C for string case insensitive, N for numeric);
- 1-based number of CSV field (only when sorting in CSV mode);
- beginning position referring to the 1-based position within the record; when sorting in CSV mode, referring to the 1-based position within the field (without possible quotation sign);
- length (can be zero for all keys when sorting in CSV mode; otherwise, the last non-numeric sort key can have zero length; if length is zero, the partial key goes until the end of the record);
- an optional appending minus sign for descending sort order (default is ascending).

When using two or more sort keys, the partial keys are concatenated into one string, the superkey (representing the complete sort key). Let's have a look at the following example file with fixed-length fields (each record is 29 byte long, Blank is represented as a point):

```
12345678901234567890123456789 (position indicator)
Product...Qty.Location..... (field names)
Apple......15.Kitchen........
Apple Pie...1.Kitchen table..
```

The sort keys `/H=2 /S=15,15 /S=1,10` will generate the following superkeys:

```
Kitchen........Apple.....
Kitchen table..Apple Pie.
```

The position of the second partial sort key within the superkey must always be the same, i.e. the preceding partial key (location) must always be 15 characters long. Otherwise, the second part key would be shifted and therefore not have the desired influence on the superkey.

Only if location is the last (or only) sort key, it can have a different length, and the input file must not contain the trailing blanks for the location field. In the latter case, the superkeys for `/S=1,10 /S=15,0` are:

```
Apple     Kitchen
Apple Pie.Kitchen table
```

For fixed-length fields, `L=0` can be used only for the last non-numeric sort key (but not in combination with descending sort order). In this case, the sort key begins at position `F` and goes until end of record.

In contrast, when sorting CSV files, the program can automatically align sort keys (because it can find the end of each field). This is the reason why you can use `L=0` for any sort key. If a sort key with `L=0` is present, the maximum length of this field will be analyzed beforehand.

For CSV files, `L=0` can be used for every sort key. The maximum field length will be determined, and all sort keys are automatically aligned by filling up with spaces
- on the *left* side for *string* sort keys
- on the *right* side for *numeric* sort keys

The complete line is used as one sort key if no sort keys are indicated. This has the same effect as using `/S` or `/S=1,0` as sort key (therefore, the lines can have different lengths).

7.4 Details about sorting

CMsort is reading records of an input file until the given amount of memory is reached. Then the records are sorted and written to a temporary file. This will be repeated until all records are processed. Finally, all temporary files are merged into the output file.

Unless specified by the `/T` option, the current directory (i.e. the *directory from where CMsort was started*) will be used as directory for the temporary files. The temporary files are named `$CMsort0.tmp, $CMsort1.tmp` etc.

If the RAM is not sufficient to read the complete input file, temporary files must be created; in this case, the additional space s needed during the sorting process on your hard drive can be estimated as follows:

$$s \leq 2n + 5m$$

where n is the size of the input file and m is the used RAM in bytes.

CMsort uses the quicksort algorithm which is *not a stable sort* by design. This means: In the output file, records having the same sort key(s) may not appear in the same order as within the input file.

8 Licence Agreement and Copyright

IMPORTANT - READ CAREFULLY **BEFORE USE**

This license and disclaimer statement constitutes a legal agreement ("License Agreement") between you (either as an individual or a single entity) and Christian Maas (the "Author") for this software product ("Software"), including any software, media, and accompanying on-line or printed documentation.

BY DOWNLOADING, INSTALLING, COPYING, OR OTHERWISE USING THE SOFTWARE, YOU AGREE TO BE BOUND BY ALL OF THE TERMS AND CONDITIONS OF THIS LICENSE AND DISCLAIMER AGREEMENT.

1. This software is freeware. You can use this software royalty-free for private and commercial purposes.
2. You can freely distribute copies of the main archive as long as no alterations are made to the contents and no charge is raised except a reasonable fee for distributing costs.
3. You may not modify, reverse engineer, decompile, or disassemble the object code portions of this software.
4. This software is owned by Christian Maas and is protected by copyright law and international copyright treaty. Therefore, you must treat this software like any other copyrighted material (e.g. a book).
5. This software is provided "as is" and without any warranties expressed or implied, including, but not limited to, implied warranties of fitness for a particular purpose.
6. In no event shall the author be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use of or inability to use this software or documentation, even if the author has been advised of the possibility of such damages.
7. Any feedback given to the author will be treated as non-confidential. The author may use any feedback free of charge without limitation.

If at least one of the previous conditions is not valid or applicable in your country, you are not allowed to use this software and/ or keep a copy of this software. In this case, you must immediately delete all copies.