CET BASIC
BSORT Users Guide

CET BASIC Bsort Users Guide

Published and printed in the United States of America.

First Edition	1989
Second Edition	March, 1995
Third Edition	March, 1996

The Table of Contents

CHAPTER 1

The Installation Guide

Introduction

Bsort is an extremely fast and efficient sort utility designed to process selected data elements from CET BASIC data files and output them in a sorted sequence in a fraction of the time it takes using BASIC alone. It is available for use with all of the CET products.

Bsort is a full-featured, record-level sorting utility. It provides the ability to sort a disk file by any field, a portion of a field or a combination of fields. The input file may be organized as sequential, direct or indexed and may contain either formatted binary or unformatted ASCII records. The output file may also be sequential, direct or indexed.

Bsort is extremely flexible so that it can be used to enhance any CET application. For example, business programs written in CET BASIC can sort user files and quickly return to BASIC to display them. Some outstanding features are:

- Bsort command parameters may be stored in a disk file. This feature allows you to change the parameters without having to recompile your BASIC program.
- Bsort may be used to sort 32 or more input files into a single output file.
- Bsort allows you to skip from 1 to 255 records at the beginning of the input file. This is especially important when you do not want to include control or header records with specific program information in the sorted output.
- Character, field or subfield position specifications allow fixed and variable-length fields to be sorted.
- Outputs full records, the key field or any portion of the record to meet your specific programming needs. This feature not only gives you the flexibility you need, but will speed up the sorting process when the entire record does not need to be output.
- Any combination of fields (or portions of a field) may be output regardless of the order in which the records are sorted.
- Specific fields may be rearranged within the output record.
- Up to 32 instructions may be used to select specific records that match your criteria. Only these records will be sorted and output according to the specifications.
- A range of records in direct or indexed files may be selected based on the key value.
- Alternate directories may be specified for the temporary work files.

The syntax of the Bsort command and the options which may be specified are covered in the next chapter. A more detailed discussion of the sort parameters and the general operation may be found in Chapter 3. Specific examples are given in the Appendix.

Installation Guide

The Bsort utility is distributed as an executable with an error message file. The Bsort executable should be placed in a directory in your search path. Typically, this is the same directory used to store your other CET programs.

The error messages for Bsort are stored in **Bsort_err** (under UNIX) or **bsort.err** (under MS-DOS or Windows). While the error file is not required for Bsort to operate, it must be in the directory with the executable if error messages are to be displayed.

CHAPTER 2

The BSORT Command Syntax

The Bsort parameters may be obtained from the operating system prompt, from a CSI/CSH statement within a CET BASIC program or from a sequential text file. The syntax is as follows:

Bsort *input-file(s) output-file* [*workdir*] { *keyspec* [*selspec*] [*outspec*] }

Where:

input-file(s)

is the name of one or more files that contain the data to be sorted. File names may be in the form of the operating system in use or in a THEOS format. (THEOS library file names are now supported in the current version.)

Multiple input files may be sorted by using the following syntax. Note that although the number of input files that may be sorted is limited by the amount of memory available, it should always be at least 32.

Bsort *input-file1 + input-file2 + input-file3 ... + input-fileN = output-file*

Bsort will search for the input files in the directory specified by the environment variable B_FPATH, and then in the current working directory in the same way CET BASIC programs search for files.

Note that Bsort is able to read sort parameters that are stored in an ASCII sequential file with an extension of **.qsp**. When this type of file is specified for input, any subsequent arguments on the line will be ignored.

Refer to the next chapter for detailed information on the types of input and output files that may be used.

output-file

is the name of the file to be used to store the sorted data. File names may be in either an MS-DOS, UNIX or THEOS format. If the output file does not exist, Bsort will create it.

By default, the output file will be a direct file except when the input file is sequential and the limited output option has not been used. In that case, the output file will also be sequential.

This default behavior may be overridden by using the output option to specify the desired file type; either sequential, indexed, direct or the same as the input file.

workdir

is an optional parameter that may be entered to indicate the directory to use if temporary work files are required. If this parameter is omitted, Bsort will use the following search sequence to determine which directory to use:

the directory specified by the environment variable TEMP
the \tmp directory
the directory where the (first) input file is stored

keyspec

defines the type of input file and the fields to be sorted. Up to 32 fields may be specified. The order in which field definitions are stated defines the ranking of the fields in the sort. Sort key specifications are entered in the following format. Note that spaces may be used instead of the comma separator.

{ [B] [*#nnn*], [*field#*] *field-desc, ...* }

Where:

{ }

indicate the start and end of the instructions. Under UNIX, the left curly brace has special meaning to the shell. In that case, enter a backslash (\) before the character.

B

is used at the beginning of the *keyspec* to indicate that the input file consists of binary formatted records. An optional space may be used between this and the next option (e.g. #80) for clarity, but a comma in this position is **not** allowed. For example:

```
Bsort infile outfile {B #80,...}
```

#nnn

is the maximum record length. This parameter **must** be specified when the input file is sequential.

field#

is the position of the field which contains the data to be used in the sort operation. The key field in indexed files is indicated with zero, and data fields start with one. Note that this parameter is only used when sorting binary formatted records. ASCII files consist of unformatted records with one string field per record.

field-desc

describes the contents of the field. One of the following may be specified.

F indicates a floating point number. (This parameter is **only** used in binary records. ASCII files consist of string data.)

I indicates that the data is an integer value. (This is **only** used in binary records.) For example, use the following syntax to sort the input file first by field #3 which contains an integer value. Note that the space after the "B" is not required. It is used to make the command more readable.

```
{B  3I  ...
```

[-]from-to[N] indicates a string starting with the *from* character position through the *to* position. To sort by the first five string characters in field 6 enter:

```
{B  6-1-5  ...
```

The option N may be used to indicate that the string is a number and should be converted to a numeric for proper ranking.

The total length specified (in the *keyspec*) may **not** be greater than the key length when dealing with indexed file keys or the total record length when dealing with data fields. If the length of the actual string is shorter than the specification, the data will be padded with spaces.

To sort an ASCII sequential file with unformatted records containing a single 40-character string by the first 4 characters:

```
{#40  1-4 …
```

When sorting an ASCII formatted indexed file (created with the PRINT statement), it is important to note that the record key is concatenated to the data field. This means that any *from-to* specifications will be offset by the length of the key.

D indicates that the field is to be sorted in descending sequence instead of in ascending order which is the default action.

A file may be sorted by up to 32 fields with a single command. If more than one field is specified, separate each of the *keyspec* instructions with a comma or a space character. Examples are included in the next chapter and in the Appendix for your convenience.

selspec

Selection specifications may be used to select records from the input file based on the contents of the specified fields. Only the records matching the criteria will be sorted and written to the output file.

Enter the specifications in the following format using either commas or spaces as separators. The *field#* (in binary formatted files) and the *field-desc* variables are used in the same way they were described previously.

S [*field#*] *field-desc* [operator] *'constant'*... [OR]...

Where:

S

specifies the criteria to use to select the records which are to be sorted. This character should immediately precede the first selection field definition. A space may be used as a separator, but not a comma. This option is terminated when an L, P, O or Q instruction or the end of the command line is encountered.

operator

indicates the relationship the data must have to the following *'constant'* before it will be selected. Currently, these operators must be entered in upper case letters.

EQ - equal to	NE - not equal to
GT - greater than	GE - greater than or equal to
LT - less than	LE - less than or equal to

'constant'

defines the constant to be used in the comparison. Be sure to enter the constant in the appropriate case mode as this operation is case sensitive.

Since the constant **must** be enclosed in single quotes, a quote may not be used as part of the item. Under UNIX, the single quote has special meaning to the shell and must be preceded with a backslash character (\).

If the data field being compared to the constant is defined as numeric, then Bsort will convert the constant to a numeric for comparison purposes.

OR

is used to indicate the beginning of an alternate set of record selection criteria. If omitted, AND is implied.

outspec

Output specifications are used to indicate up to 32 fields from the input record which are to be written to the output file. These fields are completely independent of those used in the sorting process. If this option is omitted, the entire input record will be written to the output

file. For this reason, the option is also referred to as the limited output option.

Enter the specifications in the following format using spaces or commas as separators. Note that the *field#* and *field-desc* variables are used in the same way as explained under *keyspec* except that the N option for strings is not needed.

L *field# field-desc ...*

Where:

L

specifies that the following instructions indicate which fields are to be written to the output file. The character should immediately precede the first output field definition with or without a space separator. This option is terminated when an S, P, O or Q instruction or the end of the command line is encountered.

Pnnn

indicates that the first *nnn* records (1-255) in the input file are to be ignored and not included in the output file. This feature should be used whenever there are control or header records in the file that should be excluded from the output.

Q

indicates that Bsort should run in 'quiet' mode. When this option is used, the copyright notice and record counts will not be displayed.

O

indicates the access method to be used for the output file. The syntax is:

O=*type*

Valid file type codes are:

R - same access method as the input file
S - sequential file
D - direct file
I - indexed file

Note that when this option is **not** specified, the output file will be a direct file except when the input file is sequential and the limited output option has **not** been used. In that case, the output file will also be sequential.

C

is the continuation character that indicates the command line will continue on the next line. This option is used by entering the C after the comma or space which terminates the prior instruction and then a <carriage return>.

Note that this option may **only** be used from the operating system prompt or from a **.qsp** file.

CHAPTER 3

The Sorting Process

Introduction

There are three main phases of the sort process. The first phase is the specification phase, in which Bsort obtains the information necessary to perform the required job.

In the second phase, Bsort reads the input file and loads the selected records into memory and sorts them into a temporary work file.

In the third or output phase, the sorted records and/or the specified fields are written to the output file indicated in the sort instructions.

The Specification Phase

During the specification phase, Bsort is provided with the operating parameters needed to perform the task. These sort parameters may be obtained from the command line, a CSI/CSH statement in a BASIC program or from an ASCII sequential file. When the sort parameters require more than one line on the console, use the continuation option (C) to inform Bsort that more input will follow.

The sort parameters include an input file, an output file, key specifications, and any selection or output options that were covered in the previous chapter. (The specification and use of the sort parameters are described in detail below.)

Bsort will analyze the sort command line to determine that all the required parameters have been entered, and that the syntax is valid. The input file is located and verified that it is has the correct format.

Input File

The name of the input file(s) **must** be specified. File names are normally in the form recognized by the operating system in use. THEOS formatted names such as **cust.data** and **cp.data.cust** are also allowed. In that case, the name will be folded to upper case as it is converted (e.g. **/DATA/CUST**).

Bsort will search for the input file in the directory specified by the environment variable B_FPATH, and then in the current working directory in the same way CET BASIC programs search for files.

The input file may be any standard CET sequential, direct or indexed file. The data records may be formatted or unformatted, but all the records in the file must be similar. Note that the Pnnn option may be used when it is necessary to skip over any control records found at the beginning of the file.

Bsort also allows you to store the sort parameters in a special ASCII sequential file with an extension of **.qsp**. When this type of file is specified for input, any subsequent arguments on the line will be ignored.

Binary Formatted Files

A formatted file contains records with string, integer or floating point fields that are specified with special formatting information. Each field starts with a format code indicating the type of data it contains. Binary formatted records are automatically supported by CET BASIC when the record is written using the WRITE statement.

The CET Blist command displays the data fields in the record separated with commas.

Unformatted ASCII Files

Unformatted ASCII files contain records with only string characters and no field delimiters. The entire record is written as a single string. The end of record delimiter is a <carriage-return>/<linefeed> under MS-DOS or a <linefeed> character under UNIX.

ASCII files are normally created with the CET BASIC PRINT statement. Each record consists of 'fields' that are a fixed length so the beginning field positions remain constant throughout the file. Fields in this type of record are often right padded with spaces to maintain their position in the record.

Output Files

By default, Bsort will output the sorted records to a direct file. The output will be directed to a sequential file when the input file is sequential and the limited output option (L) has **not** been used.

The output option may be used to override this behavior and specify the type of file to be used. In that case, the output file may be sequential, direct, indexed or the same type as the input file.

Bsort will create the output file if it does not already exist.

Sort Key Specifications

Sort key specifications are preceded with a left curly brace. These instructions define the type of input file and the fields to be sorted.

When Bsort is used to sort a binary formatted file, the first option specified **must** be the letter "B". A space may be used to separate the character from the next option, but a comma is not allowed. For example:

{B 5 ...	field #5 in a binary formatted indexed or direct file
{B#500, ...	a binary formatted sequential file with the maximum record length of 500 bytes
{B 3-1-10 ...	the first ten string characters in the third field of a binary formatted record

If the input file is sequential, the maximum record length to be read **must** be specified. For example, B#500 indicates a binary formatted sequential file that has a maximum record length of 500 bytes. If Bsort detects a record longer than the specified length an error will be detected.

At least one sort key specification must also be indicated to define the character(s), data field or portion of the field in the input record which should be used to perform the sort operation. Up to 32 sort keys may be used. When more than one sort key is specified, the first key is the most significant in determining the sequence of the output file.

The key specifications are determined by whether the input file contains binary formatted or ASCII unformatted records.

Binary formatted records have field delimiters which separate each field. Records may contain string, integer or floating point fields. To sort a binary file, you **must** indicate the location of the field that contains the data to be used. If the data is an integer or floating point value, a field descriptor must also be specified as in the following examples:

2F	a floating point value found in field #2
3I	an integer value found in field #3
6	all the string characters in field #6
6-1-5	the first 5 string characters in field #6
6-1-5D	the first 5 string characters in field #6 sorted in descending ASCII order
6-1-5N	the first 5 string characters in field #6 converted to a numeric
6-1-5ND	the first 5 string characters in field #6 converted to a numeric and sorted in descending order

Notice that no special character is needed to indicate string fields. Also note that a portion of the string (starting from through the to position) may be used for sorting purposes. If the actual data is less than the specification, it will be right padded with spaces. The specification may not be greater than the key length when dealing with indexed file keys or the total record length when dealing with data fields.

The N option may be used if the string contains a valid number, the digits 0-9, a leading plus or minus sign or an embedded decimal point. Bsort will convert the string to a numeric for sorting purposes. Otherwise, the sort will be performed using the ASCII value of the string characters.

By default, Bsort will sort data in ascending order. If the D option is specified, the output will be sorted in descending order with the highest value first.

Sorting ASCII unformatted files in much simpler. If the input file is sequential, the maximum record length to be read **must** be specified as before. For example, #500 indicates an ASCII unformatted sequential file that has a maximum record length of 500 bytes.

Since each record consists of a single string field, fewer options are needed to define the sort key specifications. Only the from-to, N and D options apply. For example:

1-5	the first 5 string characters in the record
1-5D	the first 5 string characters sorted in descending ASCII order
1-5N	the first 5 string characters converted to a numeric
1-5ND	the first 5 string characters converted to a numeric and sorted in descending order

When sorting an ASCII formatted indexed file (created with the PRINT statement), it is important to note that the record key is concatenated to the data field. This means that any *from-to* specifications will be offset by the length of the key.

Selection Specifications

Bsort allows you to select records from the input file based on field content. Only the records matching the criteria will be sorted.

Selection specifications begin with the letter "S" immediately followed by the field number (for binary formatted files) and the field description explained in the previous section.

An operator is used to indicate the relationship the data must have to a specified '*constant*' before it will be selected.

The valid operators are as follows:

EQ - equal to	NE - not equal to
GT - greater than	GE - greater than or equal to
LT - less than	LE - less than or equal to

Currently, only operators in upper case letters are recognized. Also, be sure to enter the constant used in the comparison in the appropriate case mode as this operation is case sensitive.

The constant **must** be enclosed in single quotes so a quote may not be part of the item. Under UNIX, the single quote has special meaning to the shell and must be preceded with a backslash character (\).

If the data field being compared to the constant is defined as numeric, then the constant will be converted to a numeric for comparison purposes.

The OR keyword may be used to indicate the beginning of an alternate set of record selection criteria. If omitted, AND is implied when multiple selection specifications are used. (These keywords must be in upper case letters.)

For example, the following specifications may be used to select records based upon:

S4-1-1,EQ,\'A\'	the first character in field #4 is an 'A'
S6,EQ,\'CA\'	the data in field #6 equals 'CA'
S9-1-3N,LT,\'100\'	the value in the first 3 characters in field #9 is less than 100

Regardless of the format of the file, it may be necessary to indicate that the first *nnn* records (1-255) in the input file are to be ignored. This feature is especially important when the file contains header records which are used to control the operation of the BASIC program.

The Sort Phase

After the sort parameters have been determined and verified, Bsort reads the input file and selects the records according to the specifications, if any.

The records are then loaded into the available memory for sorting. If the total number of records will not fit in the available memory, a temporary work file is created on the disk. If the *workdir* option has been specified, the work file will be created in that directory. Otherwise, Bsort will use the following search sequence to determine which directory to use.

the directory specified by the environment variable TEMP
the \tmp directory
the directory where the (first) input file is stored

The Output Phase

In the third phase, the sorted records are written to the output file specified in the command line. Output specifications may be used to limit the fields which are to be written to the output file. These fields are completely independent of those used in sorting. If this option is omitted, the entire record will be output.

The letter "L" is used to indicate that the following instructions define which fields are to be output. The character should immediately precede the first output field definition with no comma separator. (A space separator is allowed.) This option is terminated when an S, P, O or Q instruction or the end of the command line is encountered.

The *field#* (for binary formatted records) and the *field-desc* variables are used as before except that the N option to convert a string to a numeric value is not needed in this phase.

Certain options (P, O, Q and C) may also be used to modify the behavior of Bsort and indicate that the beginning (control) records in a file need to be skipped, the access method to use for the output file, the copyright and record count display should be suppressed and the sort instructions have been entered on more than one line.

APPENDIX A

Bsort Examples

This section illustrates some typical commands to sort records in a UNIX environment. The equivalent commands in an MS-DOS or Windows environment could use a \ (backslash) in front of file names and would not require a backslash to delimit the single quote character in a selection specification. For clarity, extra spaces have been used to separate the various parameters in the commands.

Sorting Binary Records

Example 1:

In the first example, **./testfile** is a sequential file created with the WRITE statement. Each record contains six fields. The first field in the record contains an integer and the rest of the fields contain strings. The maximum record length is 80 bytes.

```
Bsort  ./testfile  ./out { B  #80  6-1-1  1I  2-1-5 }
```

This command sorts the records in ./**testfile** and outputs the entire record to a new sequential file named ./**out** since the limited output option L has not been used.

First, the records are sorted by the first character of the sixth field in ascending order. The records are then sorted by the integer value in field one. A third sort is done using the first five characters in the string stored in the second field.

Example 2:

The following command is similar to the previous one except that the records are sorted by the first five characters in the second field after they have been converted to numeric values.

```
Bsort  ./testfile  ./out  { B  #80  1I  2-1-5N }
```

Example 3:

This example will select records in ./**testfile** if the first character in field four is an A. The selected records will then be sorted by the first forty characters in field four. Only those characters will be written to the direct file named ./**out**.

```
Bsort ./testfile ./out {B #80 4-1-40 S 4-1-1 EQ \'A\' L 4-1-40}
```

Note that Bsort will not output to a sequential file when the limited output option L is used. The O option may be used to force output to a different type of file.

Example 4:

This example illustrates how to specify that the sort parameters are stored in an ASCII sequential file. This feature is especially useful when you want to change the sort parameters without having to recompile the BASIC program. It also allows you to enter very long sort instructions which require

the use of the continuation option C (to avoid exceeding the 127 character command processor limit imposed by MS-DOS).

Two requirements must be met to use this technique. First, the file name **must** be a UNIX or MS-DOS formatted file name, depending upon the operating system in use. The file name must also have the extension .**qsp**. For example:

```
Bsort ./example.qsp
```

The contents of two records in a .**qsp** file might be:

```
./testfile ./out {B#80,6-1-1,3-1-6,4-1-40,5-1-4,1I,C
2-1-5,S4-1-1,EQ,\'A\',L4-1-40,1I}.
```

Example 5:

In this example, the file to be sorted is named ./**isamfile.** It is an indexed file with three fields in each record. The first field contains an integer, the second a floating point number and the third a string.

The following command will sort the contents of ./**isamfile**. The option O has been specified so that the sorted records will be output to a sequential file named ./**out** (instead of a direct file) in descending order. The records are first sorted by the first five characters in the third field. The second and third order of ranking will be determined by the contents of the second and first fields.

```
Bsort  ./isamfile  ./out { B  3-1-5  2F  1ID  O=S}
```

Sorting ASCII Records

This section illustrates some typical commands to sort ASCII records.

Example 1:

In this example, the file to be sorted, ./**testfile**, is a sequential file and each record contains a single string 40 characters long.

```
Bsort  ./testfile  ./out  { #40  1-4}
```

The records will be sorted by the first four characters of the record. The entire record in ./**testfile** will be output to a sequential file named ./**out**.

Example 2:

The following command will sort the records in ./**isamfile**, an indexed file with a key length of four and a record length of 40. Since this is an indexed file, the record key is concatenated to the data field before the sort operation begins.

```
Bsort  ./isamfile  ./out { 5-9  L1-4 }
```

Bsort will sort the records in the input file using the first four characters in the data field and write the key to the record to a direct file named ./**out**.

APPENDIX B

Error Messages

The error messages for Bsort are stored in **Bsort_err** (under UNIX) or **bsort.err** (under MS-DOS or Windows). While the error file is not required for the operation of Bsort, it must be in the directory with the executable if error messages are to be displayed.

The following is a listing of the contents of the error file:

```
/* Copyright (c) 1986 by CET Software, Inc.  */

1       Insufficient memory.
2       Can't open file "{0}".
4       Can't create file "{0}".
5       Invalid option: "{0}".
6       Can't find file "{0}".
7       Can't unlink file "{0}".
11      Read failure: {0}.
12      Write failure: {0}.
54      Syntax error.
/*      the following are specific to Bsort   */
501     Can't close file "{0}".
502     STRING to REAL conversion error.
503     Unsupported/Invalid data type: {0}.
504     Invalid data type, expected INTEGER.
505     Invalid data type, expected REAL.
506     Invalid data type, expected STRING.
507     Lseek error: "{0}".
508     Sequential record exceeds buffer size.
509     Missing output filename.
510     Missing option separator.
511     Bypass count greater than file record count.
512     Key field in non-ISAM file.
513     'N' option used with a non-string field.
514     Invalid relational operator: "{0}".
515     Missing comparison constant.
516     {0}: substring exceeds file record size: {0}.
517     {0}: substring exceeds file key length: {0}.
518     {0}: FROM field > TO field.
519     Records skipped  = {0}.
520     Records read     = {0}.
521     Records selected = {0}.
522     Records written  = {0}.
523     Program interrupted (signal = {0}).
```