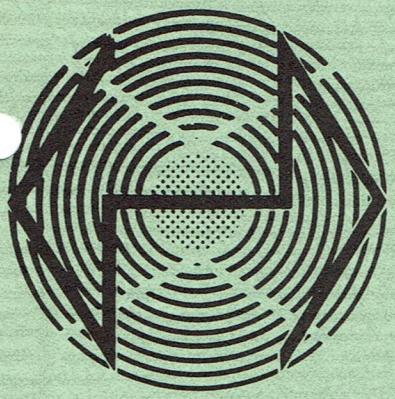


A/S REGNECENTRALEN

RC SYSTEM LIBRARY FALKONERALLE 1 DK-2000 COPENHAGEN F

RCSL No: 21-V015

Edition: February 1977

Author: Jørgen Winther

Title: SORTBS - SORTPROCBS

Keywords: RC 4000, RC 8000, ALGOL, FORTRAN, SYSTEM80, DISC, SORTING.

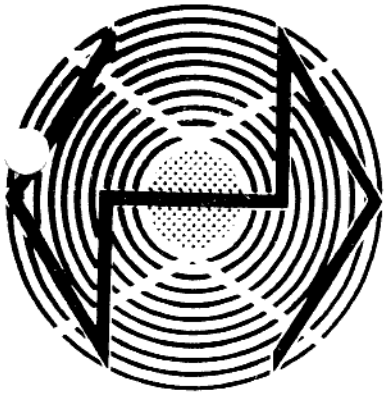
Abstract: SORTBS is a program which sorts a disc file holding records of either fixed or variable length.

The output file is delivered on the scope specified.

The sorting is performed by means of the external algol_procedure SORTPROCBS which delivers the output file on scope temp.

It is the hope of the author that the current versions of mdsort and mdsortproc in a short time will be replaced by sortbs and sortprocbs.

In a period of about a year the md- and bs-versions will coexist with identical content, and after that the names sortbs and sortprocbs will be removed from the SYSTEM80 system tape.
Note that the new type 'absbyte' is not yet available in user-programs through sortprocbs as the official version of the internal sorting system has not yet been updated.
6 + 6 pages.



A/S REGNECENTRALEN

RC SYSTEM LIBRARY FALKONERALLE 1 DK-2000 COPENHAGEN F

RCSL No: 21-V015

Edition: February 1977

Author: Jørgen Winther

Title: SORTBS - SORTPROCBS

Keywords: RC 4000, RC 8000, ALGOL, FORTRAN, SYSTEM80, DISC, SORTING.

Abstract: SORTBS is a program which sorts a disc file holding records of either fixed or variable length.

The output file is delivered on the scope specified.

The sorting is performed by means of the external algol_procedure SORTPROCBS which delivers the output file on scope temp.

It is the hope of the author that the current versions of mdsort and mdsortproc in a short time will be replaced by sortbs and sortprocbs.

In a period of about a year the md- and bs-versions will coexist with identical content, and after that the names sortbs and sortprocbs will be removed from the SYSTEM80 system tape.
Note that the new type 'absbyte' is not yet available in user-programs through sortprocbs as the official version of the internal sorting system has not yet been updated.
6 + 6 pages.

Purpose.

Sortbs, sort_backing_storage, is a sorting program for fast sorting of one disc file holding records of either fixed or variable length.

File format.

Records of fixed length are handled by means of inrec6/outrec6.
records of variable length are handled by invar/outvar.
The length of a file is either given in the catalog entry of the input file:

```
inputfile=set <segments> <storage device> <number of records>
```

or by a special end of file record. See the call definition below.

Method.

The program is based upon the external algol procedure sortprocbs. This procedure performs the sort in 2 phases:

1: The input file is read, and sorted strings of the maximum length are output consecutively in one area.

2: The strings generated in phase 1 are merged together in the needed number of passes.

The procedure will optimize the sorting by variation of the number of passes, the blocklengths, and the number of shares, and by utilization of 2 disc stores if available.

the mutual ordering of records with equal values in all keyfields, i.e. synonyms, is not changed by the sorting.

Requirements.

The merge technique requires 2 backing storage areas able to hold the data.
One of these areas can be the input file if the program is allowed to clear it.
The job process must own at least 2 message buffers, but the machine can be utilized harder with a greater number, up to about 20, especially with great core size.
The safe minimum core size is given in bytes by the following expression:

```
12000 + 512*(inputblocklength + outputblocklength)
+ 4*maxlength + 48*noofkeys.
```

The blocklengths are given in segments, (512 bytes), and the maximum recordlength in bytes.
The minimum core size for blocklengths of 2 segments is thus about 15000 bytes, but it is emphasized that this core size will give a very inefficient sort, 30000 to maxsize would be more appropriate, depending on the data volume.

Example of program call.

For a more exhaustive definition, see the next section.

```
sortbs  in.file1  out.file2 , input and output files
        block,2             , input and output blocklengths in
                            , segments.
        var,34              , variable reclength, max 34 bytes.
        long,8              , first sorting criterion, ascending
        real,20,d           , second, descending
        absbyte,11,16,d     , criterion 3, 6 bytes without sign
        word,10
```

Program call definition.

The program call consists of the following parts:

```
sortbs  <sortfiles>  <sortspecifikation>
```

Both <sortfiles> and <sortspecifikation> are groups of fp-parameters which will be defined in the sequel.

Sortfiles.

Fp-parameters defining input and output.

```
<sortfiles>::=  in.<input file>  (.clear)0/1
                out.<output file> (.<output disc> (.<scope>)0/1 )0/1
```

<input file>, <output file>, and <output disc>::= <name>

<scope>::= temp ! login ! user ! project

The signature 0/1 means that the preceding quantity can be omitted.
The clear parameter defines, whether the input file should be cleared or not. The parameter may be necessary in connection with great data volumes.
The output file is created by the program, and placed on the output disc, if this parameter is given.
If the output disc is specified the scope of the output file will be either temp or the scope specified.
If an output disc is not specified the name of the output file is looked up with two possible results:

1. A file with this name does not exist on scope temp to project. In this case the output scope will be temp and the disc for the outout file is selected according to the most efficient sorting strategy.
2. A file with this name exists on scope temp to project. The name for the output disc and the scope for the output file is fetched from this file.

In any case all files of the name of the output file on scope temp, login, or of a scope lower than the scope selected for the output file, will be removed with a warning before the sorting starts (or in sortprocbs if it is the input file).

Sortspecification.

fp-parameters defining the details of the sort.

<sortspecification>::=
block.<input blocklength>(.<output blocklength>)0/1
<fix or var>.<maxlength>
(eof.<eof 1>.<eof 2>)0/1
<keyfield> 1/n

<input blocklength>, and <output blocklength>::= <integer>

Two integers specifying the blocklengths of the input file and the final output file as a number of segments, (512 bytes).
The maximum blocklength is 40 segments.
If only the input blocklength is specified, this blocklength is used for the output file as well.

<fix or var>::= fix ! var ! vnc

Defines whether the records of the input file was created by outrec or by outvar.
vnc means invar without checksum.

<maxlength>::= <integer>

Defines the fixed or the maximum length of a record measured in bytes.
It must be even, and not less than 2.
It is important for the efficiency of the sort that maxlength is given as accurate as possible in the case of variable record-length.

<eof 1>, and <eof 2>::= <integer>

If the eof parameter does not occur, then the length of the input file is given by the content of the catalog entry of the file. The word succeeding the name of the disc storage device in the tail of the entry must contain the number of records to be sorted:
inoutfile=set <segments> <storage dev.> <number of records>

But if the eof parameter is given, then the file end of the input file is defined by a special record having <eof 1>, and <eof 2> as the values of the first 2 words of the user part of the record, i.e., byte 1 to 4 of fixed length, and byte 5 to 8 of variable length records.
The number of sorted records is always inserted in the catalog entry of the output file, and an end of file record is written at the end of the file if eof is used.

<keyfield>::= <type>,<firstaddr> (,<lastaddr>)0/1 (,descending)0/1

This is the specification of one keyfield.
The order of specification gives the priority of the keyfields.

<type>::= byte ! word ! long ! real ! absbyte

The types correspond to the types 1 to 5 in the internal sorting system of rc4000 algol.

<firstaddr> and <lastaddr>::= <integer>

Specifies the position of the keyfield.
The keyfield must be entirely within a maximum length record and only a byte or absbyte keyfield may have an odd position.
The keyfield consists of at least one simple field of the type specified. This field will have <firstaddr> as the field address.
If <lastaddr> is used, it must have the value:

<lastaddr> = <firstaddr> + (n - 1) * type_length

where n is the number of simple keyfields and type_length the length of one simple field, 1, 2, or 4 bytes.
The composite keyfield counts as n in the calculation of number of keys (noofkeys).
The maximum value of noofkeys is 50.

.descending

If the sorting order should be descending, this parameter must be used.

Note on the syntax of the fp-parameters.

The individual parametergroups may occur in any order.
A parametergroup is defined as a sequence of parameters, separated by points.
The last occurrence is used except for the keyfields, where all occurrences and their mutual ordering is significant.
Only the first 3 characters of the keywords are checked, so for example output can be used instead of out and blok instead of block.
For in and descending only the 2 first and the first character is checked, and for out and block the forms ud and segm are allowed.

Variable length records.

The sum check facility of invar is used during the reading of the input file, unless the parameter vnc is used instead of var.
The record length must not excede maxlength, and it must not be odd.
The minimum record length is given by the greatest of the two values: 4 and the position of the first keyfield.
Thus some of the keyfields of a short record may in fact be situated outside the record.
Such a record is sorted as if all the bits of keyfields outside the record were equal to zero.

Bs system files.

If the content field of the catalog tail of the input file is equal to 20, it is supposed that the file conforms to the conventions of the bs system developed at SUS.
In this case, the block parameter is not necessary, and the eof parameter is irrelevant.
The input file is read with the blocklength specified in the catalog tail, and the output file is written with the same blocklength if the block parameter is absent.

Printed output and the execution of the program.

1. The program call is listed on current output.
   Two different errorindications may be printed among the fp-parameters:
   <*> the preceding fp-parameter is illegal.
   <*< the preceding parametergroup is incomplete.

2. Alarms are printed if the accepted fp-parameters are incomplete.

3. The input file is looked up, and the name, the tail of its catalog entry, and its scope is printed.

4. If any errors have been detected up to this point, the run is stopped by a runtime alarm setting the ok-bit false.

5. All files of the name given for the output file which have to be removed before the sorting are looked up and shown in the same way as the input file, just before the removal.
   The first of these files may actually be the input file, it is not removed at this point but in sortprocbs.

6. The text: sort start: is printed just before the call of the sorting procedure sortprocbs.

7. The sorting may be stopped by a runtime alarm from sortprocbs.

8. After return from sortprocbs the text: sort ok: is printed.

9. The output file is scoped and looked up.

10. The number of segments and records produced in the output file and the real time and the cpu time is shown.

11. If any errors occurred at 5. or 9. return is performed by a runtime alarm, otherwise the ok-bit will be true.

Alarms.

The possible alarmsources are multiple, but the user errors should be trapped by either sortbs or sortprocbs.
These alarms are preceded by the text: ***sortbs alarm: and ***sortprocbs alarm: respectively.

***sortbs alarm:

error in fp-parameters: <n> wrong parametergroups
in.<inputfile> not given
out.<outputfile> not given
block.<blocklength> not given
fix, var or vnc.<maxlength> not given
no keyspecifikation
more than 50 keyfields
negative number of records from tail(6)
inputfile does not exist
no resources for scope of outputfile
monitor procedure: <n> result: <r>

These alarms are supposed to explain themselves.
The last one concerns peculiar results from calls of monitor procedures and should not occur.

***sortprocbs alarm:

alarm text.	alarm integer.	comment
param	1	wrong input blocklength.
	3	wrong output blocklength.
	5	wrong maxlength.
	6	noofkeys > maxlength.
keyfield	keyfield no.	illegal position of keyfield.
infile	tail(1)	input file is not an area.
r.length	record length	illegal variable length.
remove	monitor result	input file cannot be cleared.
size	-lacking bytes.	the size of the job must be increased so much before sortprocbs will do the sorting.
disc	-lacking entr.	too few catalog entries in main catalog.
	segments	not enough segments for one work file of this size.
out disc	-1	the output disc is not mounted.
	segments	not room for file of this size on the output disc.

In addition alarms from stderror may occur if record or file formats are strongly illegal.

The alarm r.length, and stderror alarms occurring during the reading of the input file are also preceded by a line, specifying the number of input records accepted before the error was detected.

Purpose.

Sortprocbs, sort_procedure_backing_storage, is a procedure, intended for fast sorting of one backing storage area.
The procedure can be called from a program coded in algol or fortran for RC 4000, RC 6000, and RC 8000.

Function.

The procedure sorts a backing storage file holding records of either fixed or variable length, using backing storage throughout.
The basic sorting method, is the merge technique:
Sorted strings, as long as possible, are generated by internal sorting during the first reading of the input file.
After that, these strings are merged repeatedly until only one sorted string is left.

The procedure will try to minimize the sorting time by variation of mergepowers, blocklengths, use of single- or double-buffering, and by utilization of two disc-stores, if available.
The procedure needs in total a backing storage area of about twice the size of the data to be sorted.
It can be specified that the input file shall be cleared, so its area can be used for the merge.
The free core, when the procedure is called, must be more than about 10000 bytes, depending on the blocklengths and recordlengths specified.
The value of 10000 bytes is valid for blocklengths up to 2 segments.
The time for the sorting is inversely proportional to the amount of free core, and it can in general be decreased if 2 backing storage devices are available.

Call.

sortprocbs (param, keydescr, names, eof, noofrecs, result, explanation)

param(1:7) (call value, integer array)
This array holds various parameters of type integer and type boolean, describing the files and the records.

param(1) segsperinblock.
Blocklength of the input file, given as a number of segments. 1 <= segsperinblock <= 40.

param(2) clearinput.
1: The input file is cleared, and its area can be used for the merge.
0: The input file must not be cleared.

param(3) segsperoutblock.
Blocklength of the final output file, given as a number of segments. 1 <= segsperoutblock <= 40.

param(4) fixedlength.
1: Fixed recordlength. Inrec6/outrec6 are used.
0: Variable recordlength. Invar/outvar are used.
3: variable recordlength but no checksum.

param(5) maxlength.
The maximum length of variable length records, and the length of fixed length records, measured in bytes.
Maxlength >= 2 and maxlengh <= param(1) * 512 and maxlength <= param(3) * 512.
Maxlength must be even.
It is important for the efficiency of the sort that maxlength reflects the real maximum length of variable length records.

param(6) noofkeys.
The number of keyfields in the sorting key.
1 <= noofkeys <= maxlength and <= 169.

param(7) concerns the reaction on resource troubles.
0: Resource troubles will not stop the execution, the procedure returns with result > 1.
<>0: Resource troubles causes runtime alarm.

keydescr(1: noofkeys, 1:2)

(call value, integer array)
The description of the sorting key.
Keyfield n is specified as: +/- type, position, in keydescr(n, 1:2).
The type ranges from 1 to 5, indicating: signed byte, integer, long, real, or absbyte.
The sign of the type specifies the sequencing: + for ascending, and - for descending order.
The position of the keyfield is specified as the number of the last byte in the field, as for algol 6 field variables.
The entire keyfield must be within a maximum length record.
The length of variable length records must not be less than the position of keyfield 1, the highest priority keyfield.
records having equal values in all keyfields are sorted according to their occurrence in the input file, i.e. their mutual order is not changed.

names(1:6)

(call and return value, real array)
Contains 3 file and disc names.

names(1:2)

inputfile.
The name of a backing storage area.
The procedure asumes that the size of the area reflects the amount of data to be sorted.

names(3:4)

outputfile.
If names(3) = real<::> then the name of the outputfile is returned in names(3:4), otherwise the name given is used for the final output file.
An existing file of this name on scope temp is cleared without warning.
The sort is not able to use the resources of such a file.

names(5:6)

outdisc.
If names(5) = real<::> then the output disc is selected according to the most efficient strategy, otherwise the disc specified is used.

eof	(call value, real) If the parameter noofrecs is negative, then the end of the input file is indicated by a record holding the bitpattern given by eof in the first 4 bytes of the userpart. Byte 1 to 4 in case of fixed length and byte 5 to 8 in case of variable length records. The final output file is terminated by an endoffile record of maximum length in this case.
noofrecs	(call and return value, integer) If noofrecs is non negative, then the number of records in the input file is given by the value of noofrecs and an eof record is not created. The number of sorted records is returned in any case in this parameter.
result	(return value, integer) The value of result specifies the result of the call of the procedure. In general, resource problems will yield a result different from 1, whereas errors concerning the parameters or hard errors will stop the execution by a runtime alarm. If param(7) <> 0 only result = 1 will occur, the other results are transformed to alarms.
explanation	(return value, integer) The value of this parameter should give a further explanation of result. See the next section.

Bs system files.

If the content field of the catalog tail of the input file is equal to 20, it is supposed that the file conforms to the conventions of the bs system developed at SUS.
In this case the output file is terminated according to the bs system, without the endoffile record.

Results.

result	explanation	comment
1	segments output	the sort was ok
2	-lacking core bytes	not sufficient core
3	see alarm segments	not sufficient backing storage
4	see alarm out disc	backing store specified by names (5:6) does not exist or has too few resources.

Results > 1 are only given if param(7) = 0.

The parameter noofrecs will contain the number of sorted records if result = 1 , otherwise it is unchanged.
The output file will, provided result = 1, be cut to the minimum size, and tail(6) of the catalog entry will contain the same value as noofrecs.

Requirements.

The available amount of core storage before the call of the procedure must satisfy the condition:

```
freebytes > 7000
            + 512*(segsperinblock + segsperoutblock)
            + 4*maxlength + 24*noofkeys.
```

The procedure requires as working areas two disc files of the size of the input file.
This means in the case when the input file is removed that the procedure must be able to create one work file of that size.

If the input file is not removed the procedure must be able to create two work files of the size of the input file.
If an output disc is specified this disc must be able to hold the final output file.
already at the beginning of the procedure it is checked that the output disc is capable of holding a file of the size of the input file. (this is of course the case if the inputfile is placed on the disc specified and has to be removed).
the blocking of records may be changed by the sorting, so the output file may have a greater size than the input file.
Work files are kept at minimum size by concatenation of the records without regards to block limits.

The procedure needs 2 catalog entries, 2 area processes, and 1 message buffer. So, the job process should at least be the owner of 4 area processes, and 2 message buffers.
But it is recommended to have a greater number of message buffers, (10 to 20), especially in the case of a sort of small records, (about 2 to 20 bytes), with great core size, (> 20000 bytes).

Variable length records.

The sum check facility of invar is used during the reading of the input file if param(4) = 0 (invar with checksum control).
The record length must not exceed maxlength.
The minimum record length is given by the greatest of the two values: 4 (if noofrecs < 0 then 8) and keydescr(1, 2).
Thus some of the keyfields of a short record may in fact be situated outside the record.
Such a record is sorted as if all the bits of keyfields outside the record were equal to zero.

Alarms.

Parameter errors and hard file errors will stop the run with a run time alarm.

alarmtext	integer	comment
param	param number	error at param(param number)
keyfield	keyfield	illegal position or type of keyfield <integer>.
create	monitor result	abnormal result in call of the monitor procedure create entry.
lookup	monitor result	abnormal result from lookup entry. This alarm will normally indicate that names(1:2) does not specify a catalog entry.
change	monitor result	abnormal result in call of change entry. Should not occur.
rename	monitor result	it is impossible to rename the final output file to names(3:4).
remove	monitor result	abnormal result in call of remove entry. The alarm will normally concern the original input file.
infile	tail(1)	names(1:2) does not point to a catalog entry describing an area.
r.length	record length	variable length record of a length greater than maxlength, less than keydescr(1, 2), or less than 8 if eof used.
passes	20	the sort could not be done in 20 merging passes. This alarm should never occur.
reccount	record count	this is a hard error or a programming error. The counts of records in the first pass and the last, are not equal. the last count is shown.
size	-lacking bytes	not enough core.
disc	-lacking entries	too few entries in main catalog.
	segments	not enough segments, the value of segments is the size of one workfile.
out disc	-1	the wanted output disc is not mounted.
	segments	not room for one workfile on the wanted output disc.

The last 3 alarms concerning resource troubles will only be given in case param(7) <> 0, otherwise the corresponding results, 2 to 4, with explanation will be given.
In addition, index-alarms may occur, if the parameter arrays are incorrectly declared, and alarms from invar, and stderror will occur, if records of strongly illegal length are input, or in case of hard errors.
Alarms, with the exception of index-alarms, are preceded by the text, ***sortprocbs alarm:.

The alarm r.length, and stderror alarms occurring during the reading of the input file are also preceded by a line, specifying the number of input records accepted before the error was detected.