To: Distribution

From: C. T. Clingen

Date: March 25, 1975

Subject: Sort/Merge Facility

Attached is draft documentation describing the sort/merge facility developed for Multics by Joel Berson's group in Billerica. The interfaces have already undergone several rounds of internal review and are now being published for general Project distribution before a Multics Change Request for formal approval of the interfaces is submitted.

Still unresolved is the issue of whether the two write-ups slated for publication in the Subsystem Writers' Guide should remain there or be moved to a Program Logic Manual instead.

I will be happy to collect comments you may have and send them on to Joel Berson.

Command
03/26/75

Name: sort

The command "sort" provides a generalized file sorting function in Multics.

The basic function of the Sort is to sort a file of records which are not ordered, to create a new file of ordered (or "ranked") records.

The description given here of the "sort" command is sufficient for the situation where the Sort is "free standing"; that is, where all procedures executed are part of the Sort and none are supplied as user-written procedures. (User-written procedures are called "exit procedures".) For additional information necessary to execute the command "sort" with user-written procedures, consult the description of the subroutine "sort_" in the Multics Programmers' Manual.

The user can specify one input file to be sorted. The input file muSt be in the Multics Storage System. The organization can be either sequential or indexed. Alternatively, the user can specify an input file procedure, which is then responsible for giving records to the Sort.

The user can specify an output file to contain the ordered records. The output file must be in the Multics Storage System. The organization must be sequential. Alternatively, the user can specify an output file procedure, which is then responsible for retrieving records (ordered by the Sort) from the Sort.

A file in the Storage System can be either a segment or a multi-segment file.

Records can be either fixed or variable length.

In addition to arguments to the "sort" command, other information is necessary to specialize the Sort for a particular execution. This information, called the "Sort Description", can be supplied either in a segment or through the user's terminal.

The user can specify the key fields to be used in ranking the input records. These key descriptions are specified in the Keys statement of the Sort Description. The Sort's standard key comparison procedure is then used. Alternatively, the user can specify his own comparison procedure for ranking records.

If the Sort's standard key comparison procedure is used, the key descriptions must describe a single ASCII character string field. ASCII collating sequence is used. Ranking can be either

ascending or descending. ("Normal ranking" means ascending.) The original order of records with equal keys is preserved.

The Sort provides exits at specific points during the sorting process. Procedures written by the user can be supplied at these points. (The user-written input file procedure, output file procedure, and comparison procedure mentioned earlier are examples of such exit procedures.) Exit procedures are specified via the Exits statement of the Sort Description.

Usage:

```
sort  pn_sort_desc
      -input_file  pn1 ... pnn
      -output_file pn_out
      -work_file   pn_work
      -option1-  ...  -optionn-
```

1) pn_sort_desc

Path name of the segment containing the Sort Description.

The following value may also be used instead of the path name:

-console_input
-ci

Indicates console input: the Sort Description is entered via the I/O switch (i/o stream) "user_input" (which normally is the user's terminal).

2) -input_file pn1 ... pnn
-if pn1 ... pnn

Indicates that the user is specifying the name(s) of the input file(s). If the user is supplying an input file procedure, then this argument must be omitted and the input file procedure must be specified via the "input_file" exit description of the "Exits" statement.

If this argument is supplied, it must be followed by one or more path names of the input files. The star convention may be used for any pni.

At present, only one input file path name can be specified, and the star convention cannot be used.

3) -output_file pn_out
-of pn_out

Indicates that the user is specifying the name of the output file. If the user is supplying an output file procedure, then this argument must be omitted and the output file procedure must be specified via the "output_file" exit description of the "Exits" statement.

If this argument is supplied, it must be followed by the path name of the output file. The equals convention may be used; if it is, it is applied against the first input file path name.

The following value may also be used instead of the path name:

-replace, -rp — Replace the first input file by the output file. That input file will be overwritten during the merge pass of the Sort.

4) -work_file pn_work, -wf pn_work — Indicates that the user is specifying the path name of the directory which will contain the work files. If this argument is omitted, work files will be contained in the user's process directory.

If this argument is supplied, it must be followed by the path name of the directory which will contain the work files. The equals convention may be used; if so, it is applied against the first input file path name.

The following values may also be used instead of the path name:

-process_directory, -pd — Work files will be contained in the user's process directory;

-working_directory, -wd — Work files will be contained in the user's current working directory.

5) -option<u>i</u>- — Options are selected from the following list:

-brief, -bf — Indicates that the Sort will suppress its summary report. The default is to write a summary report, via "user_output".

Notes:

The Sort Description argument must appear first. Other arguments and options may appear in any order, provided (as specified above) path names or values appear immediately following their respective key words.

Any path name can be relative (to the current working directory) or absolute.

The input file (-if) and output file (-of) path names are the names of segments or multi-segment files. The work file (-wf) path name is the name of a directory. The Sort Description path name is the name of a segment.

Sort Description

The Sort Description contains additional information to specialize the Sort for a particular execution. This information is:

Keys - Description of one or more key fields (also called "control fields") for ranking.

Exits - Definition of which exit points are used and the names of the corresponding user-written procedures.

The Sort Description is required. At a minimum, the user must either specify key fields or specify an exit to a user-written comparison procedure.

The Sort Description may be supplied either via a segment or via the user's terminal ("user_input").

If the Sort Description is supplied in a segment, its path name is specified as the first argument of the command.

If the Sort Description is to be supplied via the terminal, the Sort prints "Input." via the I/O switch "user_input" and waits for input. After typing in the Sort Description, the user types a line consisting of "." followed by a carriage return, which terminates input. (This line is not part of the Sort Description.)

1. Syntax of the Sort Description

A Sort Description consists of a series of statements. Each statement must begin with a function keyword. The function keyword is followed by a function keyword delimiter. The statement itself consists of one or more parameters, separated by parameter delimiters. Each statement must end with a statement delimiter.

The following delimiters are used:

: (colon). Function keyword delimiter. Optional.

(space). Parameter delimiter.

((left parenthesis). Parameter delimiter, but permitted only where shown below.

) (right parenthesis). Parameter delimiter, but permitted only where shown below.

, (comma). Parameter delimiter. Optional.

; (semicolon). Statement delimiter. Required.

2. Keys Statement

keys
key

The Keys statement specifies key fields to be used to sort the records of the input file(s).

The Keys statement consists of a series of key descriptions. The key descriptions are specified in order, from the first (or major) key to the last (or most minor) key.

A key description is the specification of a single key field. Each key description consists of from one to four elements, which must be written in the specified order. A key description has the following form:

datatype(size), position, reverse

datatype

Data type of the field. This element is required. See the tables below for the encoding of "datatype".

size

Size of the field, expressed in a form which depends on the data type. This element is required.

The size element has two forms:

length

For string data types, the length of the field.

precision

For arithmetic data types, the precision of the field. (No scale factor can be given.)

The size element must be a decimal integer. The unit depends on the data type.

For string data types, the exact amount of space occupied is expressed by "length". For arithmetic data types, the space occupied is determined by the combination of "precision"

with the data type and the alignment. Factors such as base (binary or decimal), type (fixed or float), and the presence of a sign are considered. For an aligned binary (fixed or float) field, the space occupied is increased if necessary to an integral number of words.

See the tables below for the semantics of the size element.

position — Position of the field, relative to the beginning of the record. This element is required. There are two formats:

w — w = Word offset of the field from the beginning of the record. Words are numbered from zero (0) for the word containing the first character of the record.

This format specifies that the field is aligned on a word or double word boundary.

w(b) — w = Word offset, as above.

b = Bit offset of the field within the word. Bits are numbered from zero (0) to 35.

This format implies that the field is not aligned, although the Sort functions correctly if it is (speed of execution may be affected).

reverse
rv — Reverse the normal ranking of the Sort for this key field. This element may be omitted; the default is "normal ranking" for this key field.

At present, only one key field may be specified - that is, the Keys statement must contain only one key description - and the data type must be ASCII character string ("char" or "ascii").

[This page intentionally left blank.]

DATA TYPE SYNTAX AND SEMANTICS

Data Type	Encoding of datatype	Unit
Character string:		
ASCII	char ascii	character: 9 bits
Bit string	bit	1 bit
Fixed binary	bin	1 bit
Floating binary	float bin	1 bit
Fixed decimal, unpacked:		
Leading sign	dec	digit: 9 bits
Floating decimal, unpacked:	float dec	digit: 9 bits

SEMANTICS OF SIZE ELEMENT

Encoding of datatype	Semantics of length|precision Unit	Range	Space Occupied
char ascii	char: 9 bits	1 - 4095	n characters
bit	bit	1 - 4095	n bits
bin	bit	1 - 71	Aligned: $1 \leq n \leq 35$: one word $36 \leq n \leq 71$: two words Unaligned: n + 1 bits
float bin	bit	1 - 63	Aligned: $1 \leq n \leq 27$: one word $28 \leq n \leq 63$: two words Unaligned: n + 9 bits
dec	digit: 9 bits	1 - 59	n + 1 digits
float dec	digit: 9 bits	1 - 59	n + 2 digits

Notes on encoding of datatype: In addition to the forms shown in the tables above, the following variants are also permitted:

1) The following alternate spellings may be used:

```
char|character
bin|binary
dec|decimal
```

2) The word "fixed" may be used (or omitted). For example:

```
fixed bin|bin
fixed dec|dec
```

3) Where datatype is encoded with more than one word, the words may be written in any sequence. For example:

```
float bin|bin float
```

EXAMPLES OF KEY DESCRIPTIONS

char(10), 0(18)	Character string, 9 bit ASCII code, length ten characters, starting at bit 18 of word 0.
char(8), 0, reverse	Character string, 9 bit ASCII code, length eight characters, starting at bit 0 of word 0; reverse normal Sort ranking.
ascii(4), 0, rv	Character string, 9 bit ASCII code, length four characters, starting at bit 0 of word 0; reverse normal Sort ranking (equivalent to "char(4), 0, rv").
bin(35), 2	Fixed binary, precision 35; since aligned, occupying one word.
bin(17), 2	Fixed binary, precision 17; since aligned, occupying one word, and equivalent to "bin(35), 2".
bin(17), 2(18)	Fixed binary, precision 17; since unaligned, occupying 18 bits, starting at bit 18 of word 2 (that is, occupying the low order half of the word).
bin(1), 2(0)	Fixed binary, precision 1; since unaligned, occupying 2 bits.
bin(1), 2	Fixed binary, precision 1; since aligned, occupying one word.
bin(36), 2	Fixed binary, preecision 36; since aligned (double word), occupying two words, and equivalent to "bin(71), 2".
dec(6), 0(9)	Fixed decimal, unpacked (9 bit), precision 6, starting at bit 9 of word 0 and occupying 7 digits including sign (that is, through the end of word 1).
float dec(9), 0(9)	Floating decimal, unpacked (9 bit), precision 9, starting at bit 9 of word 0 and occupying 11 digits including exponent and sign (that is, through the end of word 2).

3. Exits Statement

exits
exit

The Exits statement specifies the names of one or more user-written exit procedures, which are to be called at the specified exit points in the sorting process.

The parameters of the Exits statement consist of a series of exit descriptions. Each exit description is composed of exactly two parameters, which must be written in the specified order. The exit descriptions themselves may be written in any order in the statement.

An exit description has the following form:

exit_name, user_name

exit_name Key word naming the exit point at which the user-written procedure is to be called. Values may be chosen from the following list:

input_file = input file procedure.
output_file = output file procedure.
compare = comparison procedure.

user_name Name of the entry point of the user-written procedure. This parameter has the same syntax and semantics as a command name. That is:

The user name can be either a (procedure) segment name (e.g., "segment") or a segment name followed by an entry point name (segment$entry_point"). In this case, the user's current search rules are applied to find the procedure. (If some segment is already known by the specified reference name, that segment is used.)

The user name can also be a path name; that is, can specify a directory hierarchy location, either relative (to the current working directory) or absolute. In this case, the search rules are not applied and the path name is used to find the procedure. (If some other segment is already known by the specified reference name, that segment is terminated first.)

Exit Procedures

The names of exit procedures are specified via the Exits statement of the Sort Description, as described above. The specifications of and requirements imposed on exit procedures are given in the description of the subroutine "sort_".

Examples

1) sort -ci -if sort_in -of sort_out
Input.
key: char(10), 0;
.

The arguments of the command state that the Sort Description is input via the user's terminal; there is one input file, named "sort_in"; the output file is named "sort_out"; by default the work files are contained in the user's process directory; by default a report is written.

The Sort Description states there is one key, a character string of length 10 characters, starting at word 0 bit 0 of the record. There are no exits specified.

2) sort sort_desc -wf >udd>pool -bf

The arguments of the command state that the Sort Description is contained in the segment named "sort_desc"; the work files are contained in the directory >udd>pool; the report is suppressed.

Let the segment sort_desc contain:

```
keys:   fixed bin(35) 0, char(8) 1;
exits:  input_file  user$input,
        output_file user$output;
```

The Sort Description states that there are two keys. The major key is an aligned fixed binary field of precision 35, contained in word 0 of the record. The minor key is a character string of length 8, contained in words 1 and 2 of the record.

There are two exits, an input file procedure exit and an output file procedure exit. The input file procedure is the user's entry point named "user$input"; the output file

procedure is the user's entry point named "user$output". These exits must be specified because the command did not specify either an input file or an output file.

3) sort sort_desc -if sort_in -of -replace -wf -wd

The arguments of the command state that the Sort Description is contained in the segment "sort_desc"; the input file is named "sort_in"; the output file is to replace the input file; work files are contained in the user's current working directory; by default a report is written.

(END)

Subroutine
03/26/75

Name: sort_

The procedure "sort_" provides a generalized file sorting function in Multics. The functions provided are almost identical to those provided by the command "sort", and some reference will be made to that command.

The basic function of the Sort is to sort a single file of records which are not ordered, to create a new file of ordered (or "ranked") records.

There are several points during the sorting process where the Sort can exit to user-written procedures. Such a user-written procedure is called an "exit procedure". An exit procedure can perform special processing, and then must return to the Sort to continue the sorting process.

When called at the entry "sort_", the Sort will call:

(a) an input file procedure, to obtain input records and release them to the sorting process;

(b) an output file procedure, to return records (ranked by the Sort) from the Sort and to produce output.

Depending on arguments supplied to "sort_", the caller may choose either to use the Sort's standard input file processing procedure, or to use his own input file procedure called via the "input_file" exit. Similarly, the caller may choose either the Sort's standard output file processing procedure, or his own output file procedure called via the "output_file" exit. The Sort initializes itself to call the appropriate input file and output file procedures (Sort-supplied or user-written) as determined by those arguments.

If the Sort's standard input file procedure is used, the input file must be in the Multics Storage System. The organization can be either sequential or indexed.

If the Sort's standard output file procedure is used, the output file must be in the Multics Storage System. The organization must be sequential.

A file in the Storage System can be either a segment or a multi-sement file.

Records can be either fixed length or variable length.

The caller can use either the Sort's standard key comparison procedure, or his own comparison procedure called via the "compare" exit.

If the Sort's standard comparison procedure is used, a key description must be supplied. The key description must describe a single ASCII character string field. ASCII collating sequence is used. Ranking can be either ascending or descending. ("Normal ranking" means ascending.) The original order of records with equal keys is preserved.

Exit procedures are described in detail later, under the heading "Writing an Exit Procedure".

Usage:

```
dcl  sort_ entry((*)char(168), char(*), char(*),
                 ptr, char(*), fixed bin(35));

call sort_      (input_file, output_file, work_file,
                 parameters, report, code);
```

1) input_file Array containing the path name(s) of the input file(s). (Input)

At present, only one input file path name can be specified (that is, the array extent must be 1).

If the user is supplying his own input file procedure, then the first input file path name must be "" and the input file procedure must be specified via the entry variable "input_file_exit" in the "io_exits" substructure of the Sort Description.

2) output_file Path name of the output file. (Input)

If the user is supplying his own output file procedure, then this argument must be "" and the output file procedure must be specified via the entry variable "output_file_exit" in the "io_exits" substructure of the Sort Description.

3) work_file Path name of the directory which will contain the work files. (Input)

The following values may also be used:

"-pd" = work files will be contained in the user´s process directory;

"-wd" = work files will be contained in the user´s current working directory.

If this parameter is "", it is equivalent to specifying "-pd".

4) parameters Pointer to the Sort Description (see below). (Input)

5) report Specifies the type of summary report to be produced by the Sort. The report is written via

the I/O switch (i/o stream) "user_output". (Input)

The following values are permitted:

"" = normal summary report.

"-brief"
"-bf" = suppress the report.

6) code Standard Multics status code returned by the Sort. Possible values are listed below. (Output)

Notes:

Any path name - whether for an input, output, or work file - can be either relative (to the current working directory) or absolute. The star and equals conventions can not be used.

The input file and output file path names are the names of segments or multi-segment files. The work file path name is the name of a directory.

Status Codes:

The following status codes may be returned by the entry "sort_" (all codes are in error_table_):

0	Normal return (no errors).
out_of_sequence	The current call is not in the sequence required by the Sort; e.g., sort_ has been called after some other call to the Sort.
fatal_error	The Sort has encountered an error in calling upon some other function of the Multics system, such as the File System or the I/O System. The Sort will have previously printed a specific message related to the condition via error_output.
bad_arg	One or more arguments specified to the Sort, including those in the Sort Description, was invalid or inconsistent. The Sort will have previously printed specific diagnostic messages via error_output.

Sort Description

The Sort Description contains additional information to specialize the Sort for a particular execution. This information is:

Keys - Description of one or more key fields (also called control fields) for ranking.

Exits - Definition of which exit points are used and the names of the corresponding user-written exit procedures.

The Sort Description is required. At a minimum, the user must either specify key fields or specify an exit to a user-written comparison procedure.

The Sort Description may be supplied either in "source form" or in "internal form".

The source form of the Sort Description is an ASCII character string, in a format identical to the Sort Description as specified for the "sort" command. The source form is particularly useful when the Sort Description is a segment; the caller can initiate the segment and set a pointer to it.

The internal form of the Sort Description is the following structure:

```
dcl 1 sort_desc_ based(parameters),
      2 header                char(8) init("internal"),
      2 keys,
        3 id                  char(4) init("keys"),
        3 version             fixed bin init(1),
        3 number              fixed bin,
        3 key_desc(user_number refer(keys.number)),
          4 datatype          char(8),
          4 size              fixed bin(24),
          4 word_offset       fixed bin(18),
          4 bit_offset        fixed bin(6),
          4 rv                char(2),
      2 io_exits,
        3 id                  char(4) init("io_e"),
        3 input_file_exit     entry,
        3 output_file_exit    entry,
      2 exits,
        3 id                  char(4) init("exit"),
        3 number              fixed bin init(1),
        3 compare_exit        entry,
      2 end                   char(4) init("end ");
```

The internal Sort Description consists of five substructures: header, keys, io_exits, exits, and end. The first substructure must be "header" and the last must be "end". The other substructures must be written in the order given above, although one or more of them may be omitted.

The internal form is particularly useful when the caller is constructing the Sort Description from other information.

1. Keys Substructure

1)	id	Identifies this substructure as the "keys" substructure.
2)	version	Version number of this substructure.
3)	number	Number of key fields specified. See the Notes below.
4)	key_desc	Array of key descriptions. Each key description is one element of the array. The key descriptions must be specified in order, from the first (or major) key to the last (or most minor) key.

5) datatype — Data type of the key field. See the tables below. The value of datatype must be left justified within this item.

6) size — Size of the field, in units which depend on the data type.

"size" must always express the exact length (characters or bits) or precision (binary or decimal digits) of the field. For example, for an aligned fixed binary field of one word, size must be 35; for an aligned floating binary field of two words, size must be 63.

"size" is not always the exact amount of space occupied by the field, since factors such as sign and floating point exponent must be considered.

See the tables below for the semantics of "size".

7) word_offset — Word offset of the field from the beginning of the record. Words are numbered from zero (0) for the word containing the first character of the record.

8) bit_offset — Bit offset of the field within the word. Bits are numbered from zero (0) to 35.

If the field is aligned on a word boundary, then bit_offset = 0.

9) rv — Indicates whether the normal ranking of the Sort for this key field is to be reversed:

"rv" = reverse the normal ranking of the Sort.

"" = use the normal ranking of the Sort.

DATA TYPE SYNTAX AND SEMANTICS

Data Type	Encoding of datatype	Unit
Character string:		
ASCII	char ascii	character: 9 bits
Bit string	bit	1 bit
Fixed binary	bin	1 bit
Floating binary	flbin	1 bit
Fixed decimal, unpacked:		
Leading sign	dec	digit: 9 bits
Floating decimal, unpacked:	fldec	digit: 9 bits

SIZE SEMANTICS

Encoding of datatype	Semantics of size: Unit	Range	Space Occupied
char ascii	char: 9 bits	1 - 4095	n characters
bit	bit	1 - 4095	n bits
bin	bit	1 - 71	n + 1 bits
flbin	bit	1 - 63	n + 9 bits
dec	digit: 9 bits	1 - 59	n + 1 digits
fldec	digit: 9 bits	1 - 59	n + 2 digits

2. IO Exits Substructure

1) id	Identifies this substructure as "io_exits".
2) input_file_exit	Entry variable specifying the entry point of the user-written input file procedure. If the caller expects the Sort to use its own standard input file procedure, then this exit is not used and the input file path name(s) must be given via the "input_file" argument of the call to "sort_".
3) output_file_exit	Entry variable specifying the entry point of the user-written output file procedure. If the caller expects the Sort to use its own standard output file procedure, then this exit is not used and the output file path name must be given via the "output_file" argument of the call to "sort_".

3. Exits Substructure

1) id	Identifies this substructure as "exits".
2) number	Number of exit points specified.
3) compare_exit	Entry variable specifying the entry point of the user-written comparison procedure. If the caller expects the Sort to use its own standard key comparison procedure, then this exit is not used and the keys must be described in the "keys" substructure.

4. Notes on Entry Variables

In the "io_exits" and "exits" substructures, each exit point is specified via an entry variable. The entry variable must be set (either initialized or assigned) by a user procedure, normally the procedure which calls "sort_". The entry variable can identify either an internal entry point (that is, an internal procedure) or an external entry point of that user procedure; or it can identify an external entry point, either primary or secondary, of another user procedure.

If none of the exits declared in a given substructure (io_exits or exits) is to be taken, then that substructure can be

omitted. If a substructure is included but a given exit is not to be taken, then the corresponding entry variable must be set to "sort_$noexit", which is declared

```
dcl  sort_$noexit  entry external;
```

An exit point may not be altered after the initial call to the Sort. Any change to the entry variable thereafter will have no effect.

Writing an Exit Procedure

The following exit points are currently provided:

input file procedure

output file procedure

comparison procedure

A user-written exit procedure is called by the Sort to perform a specific function. It must perform that function, and may do other special processing. It then must return to the Sort.

The input file procedure and the output file procedure are special, in that they must themselves call the Sort in a prescribed manner (as described below). All other exit procedures must not call the Sort; they merely return to it.

A user-written input file, output file, or comparison procedure replaces the corresponding standard procedure of the Sort.

The Sort exercises control over the sequencing of calls to it, and does not permit an improper sequence to be executed. Improper sequences are indicated by status code values when the Sort returns to its caller.

The entry names of all exit procedures are defined by the user. Specific names are shown below only for convenience in discussion.

1. Input File Procedure

The Sort will call an input file procedure. If that input file procedure is user-written, it must conform to the specifications given below.

Usage:

```
input_file_procedure:  proc(code);

dcl  code  fixed bin(35) parameter;
```

1) code — Standard Multics status code which must be returned by the input file procedure. (Output)

If the value is not zero, the Sort will print the corresponding message from error_table and will return to its caller (normally the caller of sort_) with the status code "fatal_error".

Structure:

An input file procedure must have the following general form: For each record which is input to the sorting process, there must be one call to the entry "sort_$release". After the input file procedure has completed, it must return to the Sort.

2. Output File Procedure

The Sort will call an output file procedure. If that output file procedure is user-written, it must conform to the specifications given below.

Usage:

```
output_file_procedure:  proc(code);

dcl  code  fixed bin(35) parameter;
```

1) code — Standard Multics status code which must be returned by the output file procedure. (Output)

If the value is not zero, the Sort will print the corresponding message from error_table_ and will return to its caller (normally the caller of sort_) with the status code "fatal_error".

Structure:

An output file procedure must have the following general form: For each record to be retrieved there must be one call to the entry "sort_$return". If "sort_$return" is called but there are no more records to be retrieved, then it returns with a special value of the status code. This is the normal indication of end of data. If desired, the output file procedure may also terminate prior to receiving an end of data indication from "sort_$return". In any case, the output file procedure must return to the Sort.

3. Comparison Procedure

If the user specifies a comparison procedure, each time the sorting process is ready to rank two records (that is, to determine which is to be first in the sorted order) the Sort calls the comparison procedure. The comparison procedure receives as arguments pointers to each record. The comparison procedure must determine which of the two records is first - or must determine that they are equal in rank - and return a status code to the Sort.

Usage:

```
compare_procedure:  proc(buff_ptr_one, buff_ptr_two)
                    returns (fixed bin(1));

dcl (buff_ptr_one  ptr,
     buff_ptr_two  ptr) parameter;

dcl  result        fixed bin(1);

     ...

     return(result);
```

1) buff_ptr_one — Pointer to a byte-aligned (that is, not bit-aligned) buffer containg the first record of the pair to be compared. (Input)

2) buff_ptr_two — Pointer to a byte-aligned buffer containing the second record. (Input)

3) result — Result of the comparison. (Output)

Values are:

0 = records rank equal.

-1 = record one ranks first (has lower key values).

+1 = record two ranks first.

Structure:

The comparison procedure is invoked as a function. It must return to the Sort.

Notes:

Alignment of records relative to double word boundaries is preserved by the Sort. Thus if the comparison procedure applies a based declaration of a record which contains aligned data to the pointers, correct execution is ensured.

If the two records are ranked equal, the Sort preserves the original order of the records.

If the user requires the length of either record, it is available in the form

```
dcl rec_length fixed bin(21) aligned;
```

in the word preceding the beginning of the record.

Entry: sort_$release

The entry "sort_$release" is used each time the caller releases a record to the sorting process. The caller specifies the location and length of the record. The Sort accepts the record and stores it in its own work area. (Portions of the sorting process may also be performed.)

Usage:

```
dcl  sort_$release entry(ptr, fixed bin(21), fixed bin(35));

call sort_$release (buff_ptr, rec_len, code);
```

1) buff_ptr — Pointer to a byte-aligned (that is, not bit-aligned) buffer containing the record. (Input)

2) rec_len — Length of the record in bytes. (Input)

3) code — Standard Multics status code returned by the Sort. Possible values are listed below. (Output)

Notes:

Alignment of records relative to double word boundaries is preserved by the Sort.

Status Codes:

The following status codes may be returned by the entry "sort_$release" (all codes are in error_table_):

0 — Normal return (no error).

out_of_sequence — The current call is not in the sequence required by the Sort; e.g., sort_$release has been called before sort_.

fatal_error	The Sort has encountered an error in calling upon some other function of the Multics system, such as the File System or the I/O System. The Sort will have previously printed a specific message related to the condition via error_output.
long_record	This input record is longer than the maximum supported. The record is ignored by the Sort, and the caller may continue to release records to the Sort.
short_record	This input record is shorter than the minimum required to contain the key fields. The record is ignored by the Sort, and the caller may continue to releease records to the Sort.

Entry: sort_$return

The entry "sort_$return" is used each time the caller retrieves a record, in sorted order, from the Sort. Upon return from "sort_$return", the caller has available the location and length of the record.

If "sort_$return" is called but there are no more records to be retrieved, then "sort_$return" returns to the caller with a special value of the status code.

Usage:

```
dcl sort_$return entry(ptr, fixed bin(21), fixed bin(35));

call sort_$return (buff_ptr, rec_len, code);
```

1) buff_ptr — Pointer to a byte-aligned (that is, not bit-aligned) buffer containing the record. (Output)

2) rec_len — Length of the record in bytes. (Output)

3) code — Standard Multics status code returned by the Sort. Possible values are listed below. (Output)

Notes:

Alignment of records relative to double word boundaries is preserved by the Sort. Thus if the caller applies a based declaration of a record which contains aligned data to the pointers, correct execution is ensured.

Status Codes:

The following status codes may be returned by the entry "sort_$return" (all codes are in error_table_):

0 — Normal return (not end of information, no error).

end_of_info — There are no more records to be returned from the Sort. This is the normal end of data condition. No record is returned to the caller.

out_of_sequence	The current call is not in the sequence required by the Sort; e.g., sort_$return has been called before sort_$commence.
fatal_error	The Sort has encountered an error in calling upon some other function of the Multics system, such as the File System or the I/O System. The Sort will have previously printed a specific diagnostic message related to the condition via error_output.
data_loss	End of data has been reached, but the number of records previously returned is less than the number of records released to the Sort. No record is returned to the caller.
data_gain	The number of records returned (including this record) is now larger than the number of records released to the Sort. The current record is returned to the caller, and the caller may continue to request records from the Sort.
data_seq_error	A sequence error has occurred in the records returned to the caller (as determined by the key fields of the record). The current record is returned to the caller, and the caller may continue to request records from the Sort. (This condition is not currently detected by the Sort; if the user desires protection, he must check for this condition himself.)

(END)

Command
03/26/75

Name: sort

The command "sort" provides a generalized file sorting function in Multics. Please refer to the description of the command "sort" in the Multics Programmers' Manual for the primary functions of the Sort. This description in the Subsystem Writers' Guide only describes additional optional arguments.

Usage:

```
sort  pn_sort_desc
      -input_file  pn1 ... pnn
      -output_file pn_out
      -work_file   pn_work
      -option1-  ...  -optionn-
```

5) -optioni- In addition to the options shown in the Multics Programmers' Manual description of the command "sort", the following options are also available:

-size filesize — Specifies the total size, in pages (1024 words), of the input file(s). If the value is inconsistent with the actual total input file size, then the actual value is used in further computations.

-string stringsize — Specifies the approximate size, in pages, of output strings from the presort.

-mergeway way — Specifies the way of the merge.

If neither -string nor -mergeway are specified, they are optimized by the Sort. If either is specified, the specified value is used and the other is computed by the Sort. If both are specified but are inconsistent with the actual total input file size, an error is indicated and the Sort chooses optimum values.

(END)

Subroutine
03/26/75

Name: sort_

The procedure "sort_" provides a generalized file sorting function in Multics. Please refer to the description of the subroutine "sort_" in the Multics Programmer's Manual for the primary functions of the Sort. This description in the Subsystem Writers' Guide describes only additional arguments and features.

Entry: sort_$size

The entry "sort_$size" provides functions almost identical to those described for the entry "sort_" in the Multics Programmers' Manual.

When called at the entry "sort_$size", the Sort is said to "drive" itself. That is, the Sort contains a driver which will call:

(a) an input file procedure, to obtain input records and release them to the sorting process;

(b) an output file procedure, to return records (ranked by the Sort) from the Sort and to produce output.

Depending on arguments supplied to the entry "sort_$size", the caller may choose either to use the Sort's standard input file processing procedure or to use his own input file procedure. Similarly, the caller may choose either the Sort's standard output file processing procedure or his own output file procedure. The Sort initializes itself to call the appropriate input file and output file procedures (Sort-supplied or user-written) as determined by those arguments.

If the user desires to interface more directly with the Sort, the entry "sort_$initiate" (described later) may be used.

Usage:

```
dcl  sort_$size entry((*)char(168), char(*), char(*),
                ptr, char(*), fixed bin(35),
                fixed bin(35), fixed bin(35), fixed bin(35));

call sort_$size (input_file, output_file, work_file,
                parameters, report, code,
                filesize, stringsize, mergeway);
```

1) to 6) Refer to the description of the subroutine sort_ in the Multics Programmer's Manual.

7) filesize Specifies the total size, in pages (1024 words), of the input file. If the value is inconsistent with the actual total input file size (or is zero), then the actual value is used in further computations. (Input)

8) stringsize Specifies the approximate size, in pages, of output strings from the presort. (input)

9) mergeway Specifies the way of the merge. (Input)

If both stringsize and mergeway are not specified (that is, are zero or negative), they are optimized by the Sort. If either is specified, the specified value is used and the other is computed by the Sort. If both are specified but are inconsistent with the actual total input file size, an error is indicated and the Sort chooses optimum values.

Entry: sort_$initiate

The entry "sort_$initiate" is used when the Sort is "driven" by its caller. (This entry plus the others described below support the ANSI Cobol Sort/Merge module, Level 1.)

The Sort is said to be driven if the caller supplies a driver which calls - or directly performs - the input file processing and output file processing procedures. Such a driver has the general form:

```
call sort_$initiate;

call input_file_procedure;

call sort_$commence;

call output_file_procedure;

call sort_$terminate;
```

where the requirements for user-written input file and output file procedures have been described in the Multics Programmers' Manual description of the subroutine sort_ under the heading "Writing an Exit Procedure", and the entry points "sort_$commence" and "sort_$terminate" are described below.

(The Sort is said to drive itself when the entry "sort_$size" is used. In that case, the Sort itself will perform the calls listed above.)

Usage:

```
dcl  sort_$initiate  entry (char(*), ptr, ptr,
                    char(*), fixed bin(35),
                    fixed bin(35), fixed bin(35), fixed bin(35));

call sort_$initiate (work_file, keys_ptr, exits_ptr,
                     report, code,
                     filesize, stringsize, mergeway);
```

1) work_file — Path name of the directory which will contain the work files. (Input)

   The following values may also be used:

   "-pd" = Work files will be contained in the user's process directory;

   "-wd" = Work files will be contained in the user's current working directory.

2) keys_ptr — Pointer to the structure "keys" describing the field(s) to be used as keys for sorting (see the Notes below). If the caller is supplying his own comparison procedure, then keys_ptr = null and the comparison procedure must be specified in the "exits" structure. (Input)

3) exits_ptr — Pointer to the structure "exits" specifying which exit points from the Sort are to be taken and giving the entry names of the corresponding user-written procedures (see the Notes below). If no exits are to be taken, then exits_ptr = null. (Input)

4) report — Indicates what type of summary report is to be written by the Sort. The report is written via the I/O switch "user_output". (Input)

   The following values are permitted:

   "" = normal report.

   "-brief"
   "-bf" = suppress the report.

5)	code	Standard Multics status code returned by the Sort. Possible values are list4d below. (Output)
6)	filesize	Refer to the description of "filesize" for the entry sort_$size (above). (Input)
7)	stringsize	Refer to the description of "stringsize" for the entry sort_$size (above). (Input)
8)	mergeway	Refer to the description of "mergeway" for the entry sort_$size (above). (Input)

Notes:

The structure pointed to by "keys_ptr" is identical to the substructure "sort_desc_.keys" of the internal form of the Sort Description.

The structure pointed to by "exits_ptr" is identical to the substructure "sort_desc_.exits" of the internal form of the Sort Description.

The internal form of the Cort Description is defined under the heading "Sort Description" in the Multics Programmers' Manual description of the subroutine sort_.

Entry variables in the "exits" substructure should normally be set (either initialized or assigned) by the procedure which calls "sort_$initiate".

An exit point may not be altered after the call to "sort_$initiate". Any change to the entry variable thereafter will have no effect.

In order that the Sort can be terminated properly in case of an abnormal return, the cleanup procedure of the caller of "sort_$initiate" must include a call to "sort_$terminate".

Status Codes:

The following status codes may be returned by the entry "sort_$initiate" (all codes are in error_table_):

0	Normal return (no errors).
out_of_sequence	The current call is not in the sequence required by the Sort; e.g., sort_$initiate has been called after some other call to the Sort.
fatal_error	The Sort has encountered an error in calling upon some other function of the Multics system, such as the File System or the I/O System. The Sort will have previously printed a specific message related to the condition via error_output.
bad_arg	One or more arguments specified to the Sort, including those in the Sort Description, was invalid or inconsistent. The Sort will have previously printed specific diagnostic messages via error_output.

Entry: sort_$commence

The entry "sort_$commence" is used after the input file procedure has completed releasing input records to the Sort. This call informs the Sort that end of input has been reached. Upon return from "sort_$commence", ordered records are available for retrieval by the output file procedure.

Usage:

```
dcl  sort_$commence  entry (fixed bin(35));

call sort_$commence (code);
```

1) code — Standard Multics status code returned by the Sort. Possible values are listed below. (Output)

Status Codes:

The following status codes may be returned by the entry "sort_$commence" (all codes are in error_table_):

0	Normal return (no errors).
out_of_sequence	The current call is not in the sequence required by the Sort; e.g., sort_$commence has been called before any calls to sort_$release.
fatal_error	The Sort has encountered an error in calling upon some other function of the Multics system, such as the File System or the I/O System. The Sort will have previously printed a specific message related to the condition via error_output.

Entry: sort_$terminate

The entry "sort_$terminate" is used after the output file procedure has retrieved the last record desired from the Sort. It causes the sorting process to be terminated; that is, the state of the Sort is reset so that it may be called again (serially) in the same process, and work files are deleted.

Usage:

```
dcl  sort_$terminate  entry(fixed bin(35));

call sort_$terminate (code);
```

1) code — Standard Multics status code returned by the Sort. Possible values are listed below. (Output)

Status Codes:

The following status codes may be returned by the entry "sort_$terminate" (all codes are in error_table_):

0	Normal return (no errors).
out_of_sequence	The current call is not in the sequence required by the Sort; e.g., sort_$terminate has been called prior to a call to sort_$initiate.
fatal_error	The Sort has encountered an error in calling upon some other function of the Multics system, such as the File System or the I/O System. The Sort will have previously printed a specific message related to the condition via error_output.

Notes:

A call to the enty "sort_$terminate" should be included in the cleanup procedure of the caller of "sort_$initiate".

(END)