NAME

freesort -- variable length record sort

fixedsort -- fixed length record sort

SYNOPSIS

```
freesort (argc, argv, comp)
fixedsort (argc, argv, comp)
int argc;
char **argv;

int (*comp)(pp1,pp2);
char **pp1, **pp2;
extern char DELIM = '\n'; /* for freesort() only */
extern char RCDSIZ = 0; /* for fixedsort() only */
extern int cmpflg;

extern struct {
     int e_code;
     char e_msg[50];
} errsort;
```

DESCRIPTION

freesort() and _fixedsort_() are provided as alternatives to the standard UNIX sort for those instances that the standard comparison routine is not appropriate. These routines use standard I/O. For "large" sorting jobs, _freesort_() and _fixedsort_() may use up to 10 streams above and beyond that of the routine that calls them; if 10 file descriptors are not available, then the subroutine will fail.

The comparison routine should return a value greater than 0 when the record corresponding to the first argument is "greater than" (i.e. should precede in the output) the record corresponding to the second argument. Analogously, a negative value should be return by the comparison routine when the first record is "less than" (i.e. should follow in the output) the second record. When the comparison routine returns the value 0, it means that the routine does not care which is first.

By default, the algorithm is not stable (i.e. it does not preserve the order of records with identical sort codes. However, options are available to either preserve or reverse original order which are rather efficient for most applications.

freesort() and _fixedsort_() work by allocating almost all of the available memory, repeatedly filling that core with records which they sort and dump to disc, and then they merge the disc files (if necessary). Therefore, these subroutines may not work well unless lots of unallocated memory is available to this routine. However, with separated I&D space, it should be very seldom when the calling routine takes up so much of the available memory that these routines cannot run efficiently compared to the costs of executing a smaller main which executes _freesort_() or _fixedsort_().

Upon exiting, even in the case of an error, all streams are closed and the allocated memory is returned to the system. _freesort_() and _fixedsort_() can be called repeatedly from the same routine as a part of a larger algorithm (the old versions could not). _freesort_() and _fixedsort_() catch interrupts, hangups, and quits in order to clean up the temporary files which they make. They return to main the value 6 after they have cleaned up in order to give the main routine the same opportunity to clean up. In all cases, when _freesort_() and _fixedsort_() return to main, the process is in a mode to ignore interrupts, hangups and quits; thus the calling routine may wish to reset those signals.

freesort() tosses trivial records (those which only contain the delimiter). For each input file which does not end with a delimiter, _freesort_() behaves as if a delimiter were added to the input file. However, if _fixedsort_() encounters an odd part of a record at the end of any input file, that partial record is discarded.

The first element of _argv_ is ignored. The remaining argument strings will have the following interpretation:

-m merge only. All input files are assumed to be sorted.

-u output records with unique sort keys only.

-o the next argument is taken to be the output file. If none are given or the output file is "-", then _stdout_ is assumed.

-s<char> or -s<size>
For _freesort_(), <char> is the delimiter. The delimiter defaults to '\n'. For _fixedsort_(), <size> is the record size. It defaults to zero, which if left there, causes an error.

-c<value> The external variable, _cmpflg_, which may be used as a flag by the comparison routine is set to the value of the ascii string, <value>.

-t<threshold> After partitioning the records into sets of identical sort keys, only the sets with <threshold> or more records are output.

-l<size> For _freesort_(), the limit of the record size. Records which exceed this size are truncated to <size> bytes (including the delimiter).

-T<string> For _freesort_() when a record is truncated, <string>

will be placed at the end of the record. If the "-l<size>" is not specified, then _freesort_() chooses the maximum size so that at least three records can fit into the allocated data space. If neither the "-T" or "-l" options are used, then _freesort_() will return abnormally if it encounters a record which is too large to handle.

<filename> Arguments which do not begin with "-" or follow an "-o" argument are assumed to be input files. No more than thirty input files are allowed. If there are no input file arguments, then _stdin_ is assumed.

-d For _freesort_(), a delimiter will be placed as the first character in the output stream so that all records are "surrounded" by delimiters.

-P In the output, preserve the order of the records on input if they have identical sort codes.

-R In the output, reverse the order of the records on input if they have identical sort codes.

LIBRARY

/lib/lib1.a

DIAGNOSTICS

These routines return 0 for normal execution. A variety of non-zero returns occur when the subroutine does not terminate normally. When that occurs, the return value will also be written in _errsort_._e code_ and an error message will be written in _errsort_._e msg_. The error message may help the calling routine construct an error message for the user. No message is written when the return value is 0 (normal) or 6 (interrruption by interrupt, hangup or quit signal). The structure of _errsort_, named ERRSORT, is found in "/compool/sorterr.h".