MULTICS TECHNICAL BULLETIN MTB-068

TO: Distribution

FROM: Ross E. Klinger

SUBJECT: A generalized sorting facility

DATE: April 23, 1974

This bulletin describes the changes made to the user interfaces of sort_items_ and sort_items_indirect_. These procedures provide a highly effecient, yet generalized, sorting facility.

The procedures implement adaptations of the QUICKERSORT algorithm of M. H. van Emden; algorithm A402; Comm. ACM; Vol 13; No 11; Nov, 1970; pp 693-4. A description of the algorithm may be found in Comm. ACM; Vol 13; No 9; Sep, 1970; pp 563-7. In the case of sort_items_, the algorithm was modified to reorder an array of unaligned pointers to the data items, rather than the data items themselves. In the case of sort_items_indirect_, an array of indices into the pointer array is reordered. In both cases, the algorithm has been made non-recursive.

The procedures incorporate the modification to detect ordered sequences, as suggested by Robert E. Wheeler in his Remark on Algorithm 402 [M11].

Please address all comments or suggestions to:

Ross E. Klinger
MIT Bldg. 39, Rm. 584 x3224

Or, send mail to: Klinger.PDO

The calling sequences have been changed to minimize unnecessary argument manipulation caused by alignment considerations. The arguments used are as follows:

vP	(Input). A pointer to a structure containing an array of pointers to the data items to be sorted.
iP	(Input). A pointer to the structure into which the ordered array of indices to the pointer array will be placed.
lP	(Input). A pointer to a structure containing the lengths of adjustable string data items to be sorted.
length	(Input). The length of fixed-length string data items to be sorted; a fixed bin (24) number.
function	(Input). An entry variable used to call a user-supplied function which can determine the comparative relation between two data items of arbitrary format.

The structure pointed to by vP should be declared as follows, where n is the value of v.n:

```
declare 1 v aligned,
        2 n fixed bin (24),
        2 vector (n) ptr unaligned;
```

The structure pointed to by iP or lP should be declared as follows, where n is the value of a.n:

```
declare 1 a aligned,
        2 n fixed bin (24),
        2 array (n) fixed bin (24);
```

The function should be defined as follows:

```
dcl function entry (ptr unal, ptr unal) returns (fixed bin (1));
```

The pointers refer to data items 1 and 2, respectively.

```
if data_item_1 < data_item_2, the function should return -1.
if data_item_1 = data_item_2, the function should return 0.
if data_item_1 > data_item_2, the function should return +1.
```

The following entry points are defined:

For fixed-length unaligned bit strings:

```
sort_items_$bit (vP, length);
sort_items_indirect_$bit (vP, iP, length);
```

For fixed-length unaligned character strings:

```
sort_items_$char (vP, length);
sort_items_indirect_$char (vP, iP, length);
```

For fixed bin (35, 0) numbers:

```
sort_items_$fixed_bin (vP);
sort_items_indirect_$fixed_bin (vP, iP);
```

For float bin (63) numbers:

```
sort_items_$float_bin (vP);
sort_items_indirect_$float_bin (vP, iP);
```

For data items of arbitrary format:

```
sort_items_$general (vP, function);
sort_items_indirect_$general (vP, iP, function);
```

For adjustable length character strings:

```
sort_items_indirect_$adj_char (vP, lP, lP);
(not implemented with sort_items_).
```