Published: 10/28/66

## Identification

# The Multilevel Storage Monitor
Gerald F. Clancy

## Purpose

The multilevel storage monitor functions as an independent process and is empowered to solve secondary storage device saturation problems by moving segments from device to device until a harmonious distribution of segments over devices is achieved. This process is called to work whenever any file system device interface module (DIM) signals that the amount of space used on its associated device has surpassed the capacity threshold as set in the device disposition table (see section BH.1.01).

## Introduction

The multilevel move control module (section BH.1.01) supplies a mechanism whereby segments are moved from device to device as a result of their normal use by system users. This mechanism operates as part of the user's own process and, hence, segment motion can only occur upon their activation. It is expected that this type of moving will transport more segments up the device hierarchy than down due to the importance value computation because as a segment is used more its importance value will increase. Thus, normal system operation eventually results in a saturation of those secondary storage devices highest in the device hierarchy.

The multilevel storage monitor is designed primarily to relieve this type of crisis. Any DIM whose device's used space has surpassed its overflow threshold awakens the storage monitor via a call to the device distress primitive of the move control module.

The storage monitor conducts an exhaustive scan of all segments located within some allotted portion of the directory hierarchy. Each segment so encountered (directory or otherwise) is tested to determine if its importance value is such that the segment should now reside on a different storage device in light of the current set of device residency requirements. If this current importance value is incompatible with the segment's present device then movement to a different device is effected. While this search-move process is being executed, the storage monitor also gathers statistics relating to the current segment importance value versus device usage distribution. The importance value of a segment is a function of rate of past access by users and a measure of the segment's value as set by its owner (Section BH.1.01). The device-segment statistics are compiled onto a statistic file data base which, when complete, is used to adjust the importance value criterion range of each on-line device (see Section BH.1.01).

After one complete hierarchy scan has been conducted the statistics file contains a graph plotting secondary storage space versus importance value for all segments within the file system. This data is used to adjust the importance value acceptability range for each on-line device via successive calls to the set criterion primitives of move control. The range is set by 1) establishing the desired amount of space normally used for each device, 2) partitioning the statistics file graph so that the required amounts of data are distributed among the devices and 3) setting the importance value partition points to be the acceptable importance value range for each device.

The storage monitor is composed of three modules: main control module, hierarchy scan, the mover module and the statistics file data base. Figure 1 shows a block diagram of the complete process. The following is a brief introduction to each of its constituents.

1. The Main Control Module - This module serves as the main logic program for the process. Its function is to select a hierarchy tree node defining a sub-tree to be scanned by the process and then to initiate the hierarchy scan module. Whenever control returns from the call to scan indicating that the specified search is complete, another node is selected and scan again invoked. When all the allotted subtrees have been processed, the main control module makes a normal return to its caller.

2. Hierarchy Scan - This procedure is initiated by main control. It systematically traverses the specified hierarchy. A copy of each directory entry encountered in its path is made and passed to the mover module where the movement criterion is applied and segment movement enacted if required. When the sub-tree has been searched exhaustively once, control returns to the caller.

3. The Mover Module - This procedure is called by the hierarchy scan for each directory branch to be considered.

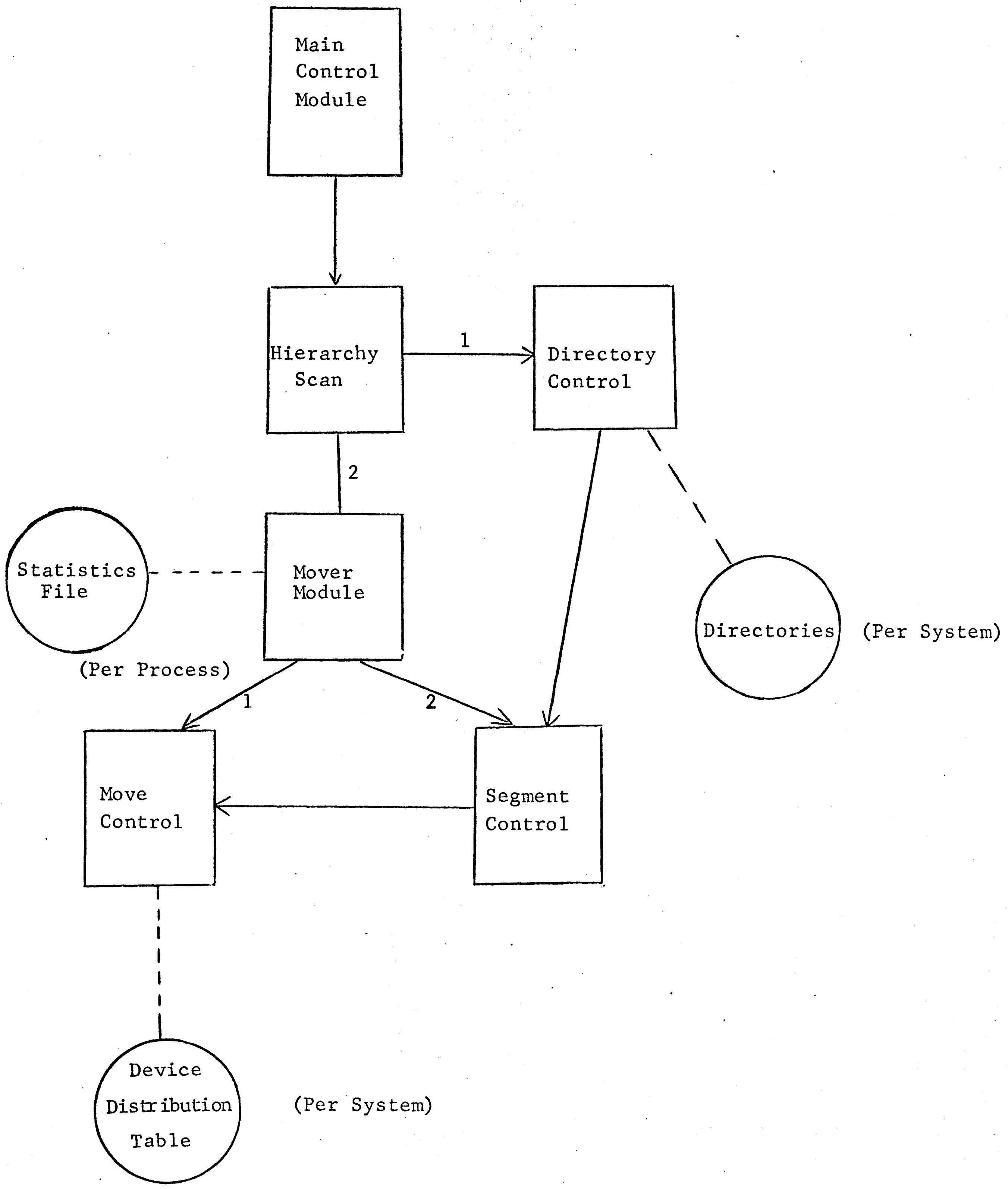

Figure 1 - The Multilevel Storage Monitor

It decides if segment movement is required and if so determines and executes those procedures which will effect the desired result. Usually, the mover initiates segment moves via the moveseg primitive of segment control. The mover may also move segments off to detachable storage. The current length of each segment examined by the mover is accumulated into the statistics file even if movement did not take place.

4. Statistics File - This data base is created anew by the process prior to each hierarchy scan. Its contents consist of a distribution graph in the form of a histogram plotting total space used as a function of importance value. As segments are examined, their importance value is computed and their current length added to the proper position in the table.

## The Main Control Module

Whenever the storage monitor begins execution, the main control module assumes control. Its only function is to determine the set of hierarchy sub-trees to be searched by the process. For each distinct member of this set, the hierarchy scan module is invoked. Whenever that list is exhausted, dump control returns to its caller. The storage monitor is initiated by the following call to the main control module.

```
call storage_monitor;
```

## Hierarchy Scan Module

The hierarchy scan module is a procedure which conducts a linear scan of all entries in a single directory and which is capable of calling itself recursively whenever an entry in a directory defines another inferior directory. In this way an exhaustive search of any hierarchy sub-tree is effected.

A directory search consists of an orderly sequence of requests for a copy of entry data via the getentry primitive of directory control (BG.8.02) for each branch by a series of consecutive requests.

Each entry so returned by getentry is preserved by the scan module until it is no longer of use. Thus, due to the recursive nature of the procedure, a scan module invoked at level n in the hierarchy adds to a list of n-1 already existing entry copies. Due to their origin, this list of entries forms a string of successively inferior entries which originate at the hierarchy root node and uniquely position the scanner at an entry at level n in the tree. If a recursive call is made to level n+1 another item is appended to the present list and if return is made from level n+1, the last entry in the list is deleted. In this way, the entry list always uniquely positions the process at some position in the hierarchy. Should the scan module move laterally from one entry to another in the same directory, then the current entry in the list at level n is replaced by another.

Once an entry has been safely extracted from directory control, the mover module is called and determines whether or not movement operations are necessary (since the mover module is called immediately after each entry is fetched, the entry copy to be considered is always the last one in the current entry list at level n).

Figure 2 presents a diagram of the scan module.

A scan of a hierarchy directory is initiated by the following call.

```
     call scan (n,node);
```

A recursive call to scan an inferior directory takes the following form.

```
     call scan (n+1, node||branch);
```

In these calls <u>n</u> is a number signifying the depth into the tree from the root at which scanning is to occur and therefore is the depth of <u>node</u> which defines the directory to be considered. <u>n</u> also defines the position in the entry pos tion list to be used when scann ng the dir ctory at level <u>n</u>. <u>node||branch</u> defines some directory immediately inferior to <u>node</u>. The PL/I declaration of the arguments is as follows.

```
     dcl n fixed,
          node char (*);
```

<u>The Mover Module</u>

This module is responsible for effecting device to device segment motion as required. It is designed to recognize only directory and non-directory branches since only a branch defines a segment occupying space on some storage device. The mover first examines the entry copy corresponding

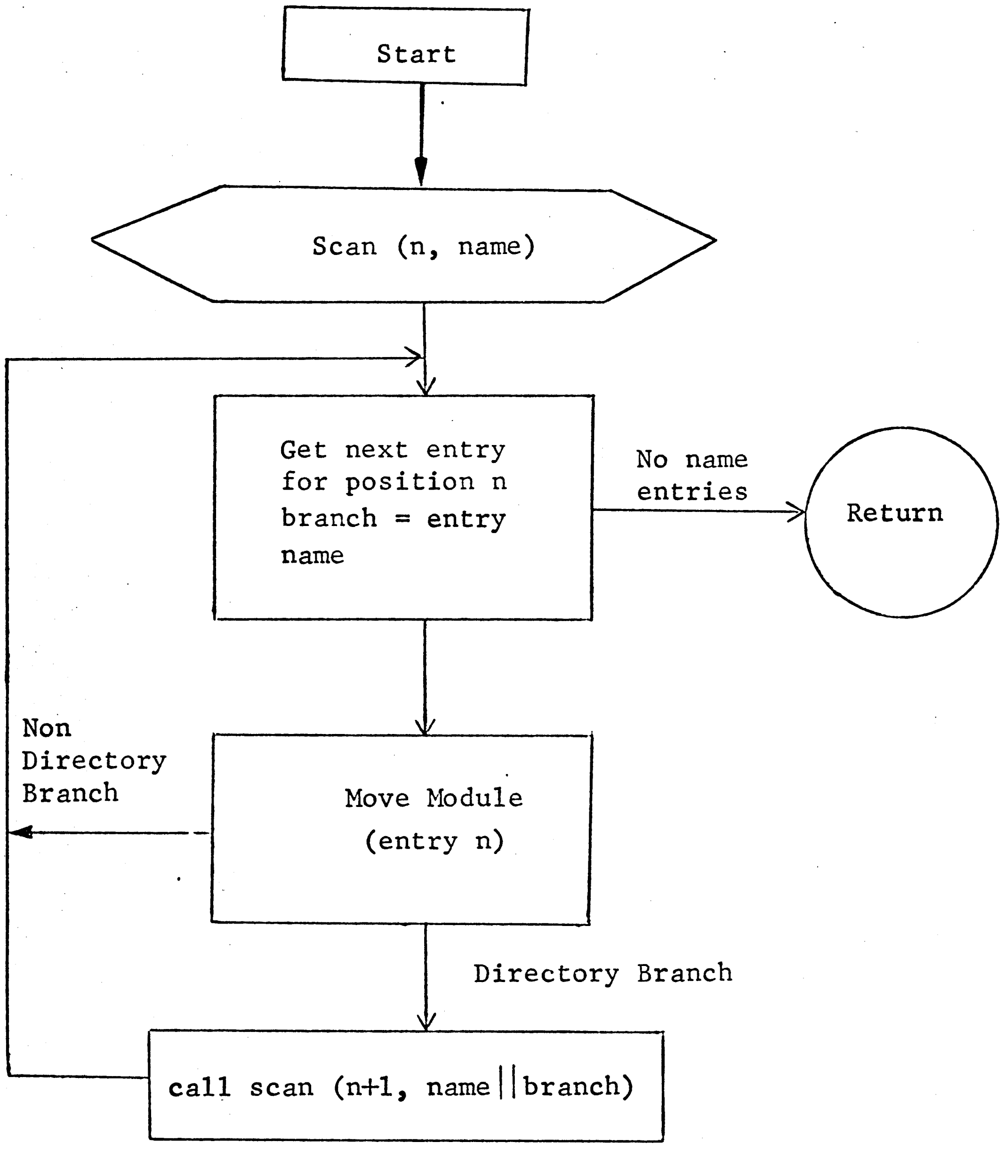

Figure 2 Hierarchy Scan Module

to the current processes scan position. A move decision is reached via the move advice primitive of move control (section BH.1.01). This call is made by extracting certain items from the entry data to complete the move advice calling sequence. The call is made with the move-off-to-detachable-storage switch ON, since the storage monitor is willing to eliminate some segments from secondary storage in order to achieve its goal. Return from this call to the move module supplies a new device identification (if a move is necessary) and the current importance value for the segment. Next, the current segment length is added to the proper location in the statistics file as determined by the importance value.

The movement requirements are then examined. In general, two distinct cases are possible.

1. The segment must be moved off to detachable storage. It is assumed that this type of move will only be necessary if the file is currently inactive and has not been accessed for some time. (This condition is, in fact, imposed as part of the importance value computation algorithm, section BH.1.01). If the segment backup status is adequate (date/time-last-modified is less than the date/time-last-dumped; indicating that a current version already exists on detachable storage) then it is truncated to zero length and the retrieval trap set ON. Otherwise, the segment must be dumped onto some permanent backup storage before truncation can occur. The single segment dump process (section BH.2.04) is awakened to dump the desired segment. Truncation then takes place.

2. The segment must be moved to another on-line storage device. In this case, a call is made to the moveseg primitive of segment control which initiates the actual segment movement. The new device identification returned by the previous call to move advice is supplied to moveseg. moveseg executes an error return if, for any reason, the segment cannot be moved (e.g. it is already being moved).

## The Statistics File

The statistics file is used to accumulate a graph of storage usage plotted against importance value for each complete hierarchy scan. Whenever any segment is examined for possible movement via a call to move advice, an importance value is returned to the mover which locates a cell in the statistics file corresponding to that value. Then the contents of that cell is incremented by the segment length.

The statistics file contents are:

1. Initial importance value

2. Difference (delta) between importance values of each successive cell.

3. Number of cells

4. For each cell

   a. accumulated storage space for this importance value.

5. Lock

The initial importance value determines the origin of the graph. Delta is the importance value increment between cells. Number of cells is the number of points to be plotted.

<u>PL/I Declaration for the Statistics File</u>

```
dcl 1 statisfile ctl (sfptr),

     2 lock bit (1),

     2 initvalue fixed, /* starting importance value */

     2 deltavalue fixed, /* increment */

     2 n fixed, /* number of cells less one */

     2 graph (0:n) fixed bin (35); /* the graph */
```

Using the above declaration, it is possible to increment the importance value <u>p</u> by <u>n</u> in the following way.

```
if p/(sfptr→statisfile . deltavalue)> n + 1 then x = n + 1;

     else x = p/(sfptr→statisfile . deltavalue);

sfptr→statisfile . graph (x) = sfptr→statisfile . graph (x) + n;
```