Published: 12/23/66

Identification

Summary of the Dumping Processes
G. F. Clancy

Purpose

The backup system provides protection against the destruction of any data known to the file system. It insures retrievability from detachable storage of any segment destroyed while residing on secondary storage and the ability to retrieve such a lost segment from more than one place in the detachable storage archives should a portion of the archives be destroyed. The insurance provided by the backup system is effected by several dumping methods. These methods employ some redundancy in order to provide multiple sources in the archives for each segment, to facilitate execution of the secondary storage reload processes and to provide, to the extent possible, user directed backup.

In general, backup strategy consists of the following:

1. Search out and dump in duplicate onto detachable storage all newly created or modified segments or, in the case of directory segments, all newly created or modified directory entries. This is known as primary incremental dumping.

2. Search out and dump in duplicate onto detachable storage secondary copies of all segments whose primary incremental copies were produced some time ago and whose secondary copies have not yet been produced. This is secondary incremental dumping. Primary and secondary dumping produce four backup copies of each segment that resided on secondary storage for a sufficient period of time. Each copy is produced with a spatial or temporal separation between it and any other. Thus maximum reliability from detachable storage is achieved.

3. Periodically dump in duplicate all segments used since some set time in the past onto a body of detachable storage which is physically distinct from incrementally produced storage. This user checkpoint dump has the effect of consolidating copies of oft used segments together where they may be found swiftly by the reloading processes.

4. Periodically dump in duplicate onto a third distinct body of detachable storage a copy of the hierarchy skeleton (all file system directories) and a minimal set of accounting and system segments. This system checkpoint dump consolidates sufficient data where it can be pro-

cessed quickly by the reload processes and hence enable a swift return to normal Multics operation following a secondary storage catastrophe.

5. Allow a user and the multilevel storage management system to forceably backup a particular segment without waiting for the incremental dumper.

6. Allow a user to insure that a particular hierarchy subtree be scanned by the incremental dumper in a consistent way, i.e., all newly created or modified segments in the subtree are forceably dumped in a mutually consistent state.

Forced backup (item 5 above) is done via the services of the single segment dumping process whose description appears in section BH.2.04. That process dumps segments onto the same detachable storage used by the incremental dumping process and produces dumping records which make it appear as if the incremental dumper itself had done the work.

Primary and secondary incremental dumping and consistent dumping (item 6 above) are effected by the incremental dumping process which operates continually within Multics and produces one continuous body of incrementally produced detachable storage. The system checkpoint dump is carried out by the system checkpoint dumping process and the user checkpoint dump by the user checkpoint dumping process.

The program structure of each of these last three processes is identical except for a single module that applies the criterion to data selected for dumping. This module, the decision module, is different for each process. The remainder of this section describes the common modules of the dumping processes while sections BH.2.01 - BH.2.03 detail the selection logic applied by the individual processes.

BH.2.01 The Incremental Dump Decision Module

BH.2.02 The System Checkpoint Dump Decision Modules

BH.2.03 The User Checkpoint Dump Decision Module

Thus, a dumping process with entirely different selection logic may be created by specifying the appropriate decision module.

Output produced on detachable storage by the incremental dumper is kept permanently or until consolidation occurs (section BH.1.04). Checkpoint dumps (of either type) are temporary in that only the latest two or three are preserved at any time, the outdated ones being discarded as newer ones are produced. All data written onto detachable

storage by any of the four dumpers is in identical logical record format. That format is: mandatory header and preamble records followed by an optional segment copy. Detachable storage data format is thoroughly described in section BH.4.03. The header contains certain identifying information about the logical record. The preamble consists of a string of successively inferior directory entries that uniquely positions the object segment (following the preamble) in the directory hierarchy. Since each logical record uniquely positions its own data in the hierarchy, each such record is entirely independent of any other.

The Process

The basic function of a dumping process is to conduct a search of some prespecified portion or sub-tree of the directory hierarchy dumping, according to its particular objectives, individual directory entries and segments.

The modular constituents of the dumping process are a dump control module, hierarchy scan, a replaceable decision module, a dump module and a data base called the position segment. These modules are introduced briefly below while Figure 1 shows their block configuration.

1. Dump Control - 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 the scan module invoked again.

2. Hierarchy Scan - This procedure is initiated by dump control. It systematically traverses a specified hierarchy sub-tree. A copy of each directory entry encountered in this path is made and passed to a decision module where the dump criterion is applied. Dumping decisions are recorded by that module and acted upon whenever the scanner invokes the dump module. When the specified sub-tree has been exhaustively searched once, control returns to the caller.

3. Decision Module - A different decision module exists for each type of dumper. In general, it examines the directory entry last extracted from the hierarchy by the scan module. If dumping of the entry itself or of the segment defined by the entry is required, switches are set detailing the required operation and will be read later and acted upon by the dump module. Dumping decisions are usually reached by testing specific items in the directory entry itself.

4. Dump Module - The dump module is called periodically by the hierarchy scanner and is asked to examine the dumping switches set by the decision module for that directory entry

corresponding to the current hierarchy position of the process. It may find switches set compelling it to dump that entry itself or its associated segment. If any of these operations is to occur, the request is placed in an output queue where it will be read and effected by the output process (BH.4.01).

5. The Position Segment -

The position segment is used by the process:

1) to keep track of its ever changing hierarchy position,

2) to communicate dumping decisions from the decision module to the dump module,

3) as a staging area for the construction of the preamble record and

4) to store entry copies while dumping decisions are being made.

The current hierarchy position of the process is determined by a list of successively inferior directory entries that eventually lead to the directory in which the dumper is currently working. The position segment contains, for each of these entries, in addition to a copy of the entry itself, the following:

1. The date/time the entry was read from the hierarchy.

2. The switches set by the decision module when it considered the entry.

3. The count of the number of entries of the current type presently in the directory.

4. The date/time when that count (item 3) was last changed.

5. The unique position of the entry within the directory (this is described later as slot number).

The significance of these items will soon become evident.

The Dump Control Module

Whenever a dumping process begins execution, the dump control module assumes control. Its only function is to determine the set of hierarchy sub-trees to be scanned by the process. For each distinct member of this set the hierarchy scan module is invoked. Whenever that list is exhausted, dump control returns if it is a checkpoint dumper (thus eliminating the process) or goes blocked for a period of time before its list of subtrees is again searched if it is an incremental dumper.

A dumping process is initiated by the following call into dump control.

```
call type_dumper;
```

where "type" is used to specify the desired dumper (e.g., call incremental_dumper).

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 is a collection of links, branches and a common access control list (CACL). Since all links and branches are uniquely numbered within the directory, a directory search consists of an orderly sequence of requests for entry via the getentry primitive of directory control for each link, the CACL followed by a series of consecutive requests for each branch.

Each entry so returned by getentry is preserved in the position segment 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 originates at the hierarchy root node and uniquely positions the scanner at an entry at level n. If a recursive call is made to level n+1 another item is appended to the present list and if a return is made from level n, 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 a directory, the decision module for the process is called and determines whether or not dumping operations are necessary (since the decision 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). The nature of the actual dumping process is that whenever some hierarchy data is dumped that data must be accompanied by a form of the current entry list (the preamble) which uniquely positions the data in the hierarchy. Therefore, the entry considered by the decision module may be a branch defining an inferior directory and in addition, a decision might be made requiring that the entry itself be dumped. In

this case, since a recursive call to scan an inferior directory is required, the entry is not dumped immediately but this action is delayed since if dumping occurs at some lower level the required dump of the current entry will automatically be accounted for as a consequence of dumping the entire entry position list (of which the current entry is a part). Thus, a return from the dump module which just considered a directory branch will be followed immediately by a recursive call to the scan module to search level n+1. When return is made from scan back to level n, the dump module is called to see if there still exists a dumping operation to be performed at the current level (n). A return from any non-directory branch call to the decision module is followed by a call to the dump module to accomplish any required dump of that entry since the next action taken will be to either replace the current entry at the present level or return to the previous level; both of which will destroy the current entry information. Figure 2 presents a diagram of the scan module.

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

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

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

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

In these calls n is a number signifying the depth into the tree from the root at which scanning is to occur and therefore is the depth of node which defines the directory to be considered. n also defines the position in the entry position list to be used when scanning the directories at level n. node||branch defines some directory immediately inferior to node. psp is a pointer to the base of the position segment in which the entry copies are stored. decision is the entry name of the decision module to be called by the scan module. These arguments are declared as follows:

```
dcl n fixed,
    node char (*),
    decision entry ext,
    psp ptr;
```

Decision Module

There exists a unique decision module for each different type of dumping process used by the backup system. Each

is designed to examine a directory entry as presented by the scan module and to flag for dumping those which are significant or which define segments significant to its own backup function. The decision module also takes measures to insure the proper disposition of such data by the dump module. In general a decision module may request that the entry being examined be dumped or that the segment defined by the entry be dumped.

The decision module must distinguish the entry type and decide what must be done. In general there are three distinct alternatives.

1. If the entry is a link or a common access control list, then no other segment is defined and the entry itself is a terminus of the hierarchy tree. The decision module must decide if such a terminal entry need be dumped.

2. If the entry is a non-directory branch, then it too is a terminal entry of the tree but associated with it is a non-directory segment. The decision module must first decide if the entry need be dumped and, second, if the segment itself fulfills the decision module's specialized requirements for dumping.

3. When a directory branch is encountered, the decision module must determine if the entry contents information is such that dumping is required and if the scan module subsequently should scan or ignore the subtree defined by the branch or if the entire associated directory should be dumped.

Since the dump module is not necessarily called after returning from the decision module (a recursive call to the scan module may intervene), the dumping decisions relevant to the entry at each successive level in the entry position list are recorded by the decision module in such a way that the dump module may later read and interpret them. Requests made by the decision module refer only to the particular entry being processed and are of the following form

1. Dump the entry itself.

2. Dump the segment defined by the entry (the entry is a non-directory branch).

3. Dump the associated inferior directory defined by this entry (the entry is a directory branch).

If one of the above operations is necessary the decision module makes use of a set of decision switches permanently attached to each entry of the entry position segment.

A complete set of switches exists for each entry and allows one setting for each operation defined above.

A decision module is called by the following:

```
call decision_module (dsw,psp, deeper);
```

In this call deeper is the return to be made when the current entry under consideration is a directory branch and the associated inferior directory is to be scanned subsequently. If a normal return is made, scanning continues in the current directory. Note that the decision module can cause an entire sub-tree to be ignored.

dsw, the dump switch is returned OFF if no dump switches have been set by the decision module and otherwise left ON. It is a signal that the scan module must call the dump module for the entry only if ON on return from the decision module. psp is a pointer to the base of the position segment.

These arguments are declared as follows:

```
dcl dsw bit(1),

    psp ptr,

    deeper label;
```

The Dump Module

The dump module exists in conjunction with the decision modules. Its purpose is to perform dumping duties as communicated via instructions found attached in the current (last) entry in the position segment. When the dump module is called it is always asked to examine only the last entry in the list which corresponds to the current depth into the hierarchy of the scan. Those functions which may be performed are the dumping of an entry or a segment.

The dump request switches attached to the entry copy in the entry position list are the following.

1. entry switch - if this switch is ON for the entry being examined then a copy of the entry itself is to be dumped.

2. segment switch - if this is ON, then the entry defines a non-directory segment which is to be dumped.

3. directory switch - if ON, then the entry is a directory branch whose associated directory is to be completely dumped.

4. update switch - this switch is a signal (if ON) to the dump module that the process is the incremental dumper. As such, the detachable storage produced will be permanently kept in the off-line archives and may be used for user retrieval of segments. Therefore, if the update switch is ON, the date/time at which anything is dumped and the exact location of all segment copies within detachable storage must be updated into the directory hierarchy. If the update switch is ON in conjunction with the entry, segment or directory switches then the date/time the entry was read from the hierarchy replaces the date/time-last-dumped in the hierarchy entry. If the update switch is ON in conjunction with the segment or directory switches, then the position on detachable storage where the dumped segment may be found is copied into the hierarchy entry to facilitate the segment's retrieval at a later date.

5. Consistent dump switch.

   If a user desires that a particular hierarchy sub-tree be dumped in a mutually consistent state, then this intention is signaled by the setting of a switch (consw) in the directory entry defining this subtree to the value 1. When that entry is considered by the incremental dump decision module, the consistent dump switch attached to the entry is set ON and consw in the dumper's entry copy is set to 2. When the dump module recognizes the switch, the sub-tree has been dumped in a mutually consistent state. The consistent dump switch in a branch is set to 0 and the entry dumped again in this state. (This is so because the dump module reads the switches for a particular entry only after all dumping inferior to that entry has been effected. During a reload, the latest dumped version of this entry (whose branch consistent dump switch (consw) is zero) will be reloaded. If the dumper did not scan the entire specified sub-tree due to a catastrophe during the dump then the latest version of the entry has its branch consistent dump switch still set to 2 as a warning to the user.

The user can therefore tell the exact state of the subtree by testing his local dump consistent variable in the entry. The possibilities are:

0. The tree is consistent. Either a dump was successfully requested, completed, and reloaded or no dump was requested.

1. The tree is consistent but waiting to be dumped in the consistent state.

2. The tree is inconsistent. Either the dump was aborted while in the subtree or the entire subtree was not reloaded.

6. Secondary Dump Switch - This switch is used only for non-directory branches. It is a signal that the secondary back-up copy of the associated segment is to be dumped. This switch is used only by the incremental dump process and hence must be ON only in conjunction with the update switch. When a secondary segment copy is dumped its position in detachable storage is recorded in the hierarchy branch as a second (alternate) set of retrieval arguments.

If dumping is required (as determined from the setting of the file, directory, entry, secondary or consistent dump switch) then the following actions are taken.

1. The preamble record to be dumped as part of the logical record on detachable storage is created. Copies of all entries in the position segment specifying the current scan position are converted to a certain bit string format. This format is described in Section BH.4.03. A distinct preamble segment is created by the successive concatenation of the bit string form of all entries.

2. If the directory switch is ON, copies of all entries in the directory segment specified by the last entry in the list are fetched via successive calls to getentry.

   Each entry from this directory is converted to the same bit string format as were the components of the preamble segment, and all are concatenated together to form a distinct directory copy segment.

3. Next, the names of one or two segments (preamble segment and possibly a non-directory segment or the directory copy segment) are placed in an output service queue where they will be noticed and copied onto detachable storage by the output process (section BH.4.01). When the request is given output service, the exact position where the data may be found on detachable storage is returned. If the update switch is ON, and either the directory, file or secondary switch is ON, this position is recorded into the hierarchy data of the current entry.

4. Since the entire current position entry list was dumped, requests that any entry superior to the current entry be dumped (entry switch ON) have also been fulfilled. The switches corresponding to all entries in the position list are scanned, and for each entry whose entry switch and update switch are ON, the date/time read from the hierarchy is updated as the date/time last dumped and both switches

are turned OFF. If just the entry switch is ON then it is merely turned OFF since the entry dump request has been fulfilled.

The dump module is called by the following:

```
call dump (psp, name);
```

In this call psp is a pointer to the base of the position segment; name is the tree name of the current entry with the position segment. The PL/I declaration of the arguments is as follows.

```
dcl psp ptr,

    name char (*);
```

The Position Segment

The position segment is a data base common to all modules of the dumping process. It acts as temporary storage for the various data necessary for the execution of the dumping process. Essentially it is composed of a one dimensional array of structures; one level being used for each entry needed to determine the current process hierarchy position. The following is a brief outline of the contents of the position segment.

1. Current scan depth into the hierarchy.
2. Array of entry data (one array location for each level).
   a) decision switches
   b) date/time the entry was read from the hierarchy
   c) packed switch
   d) slot number (integer format)
   e) slot number (character format)
   f) pointer to packed bit string of entry information
   g) pointer to entry information structure
   h) current count of entries of this type in the directory (type count)
   i) date/time type count last changed
3. Area for entry structure and bit string allocations.

The following is an explanation of the items in the position segment.

1. Current scan depth into the hierarchy - This number is used as an index into the array of entry data. When the decision module or dump module is called, the current scan depth locates the entry to be considered by the call.

2. Array of entry data - This array contains the data pertinent to an entry at any level in the hierarchy. Each entry currently required to specify the current scan position is itemized by an appropriate item of the array. Each item of the array contains the following data concerning its associated entry.

   2a. Switches - When the decision module is called to consider this entry it sets ON one or more of the following switches.

   The switches are eventually interpreted and their commands executed by the dump module.

   1) segment switch (SDW) - the segment switch is set ON by the decision module if the non-directory segment defined by the entry is to be dumped.

   2) directory switch (DSW) - the directory switch is set ON by the decision module if the directory segment defined by the entry is to be dumped.

   3) entry switch (ESW) - the entry switch is set ON by the decision module if the entry itself is to be dumped.

   4) update switch (USW) - the update switch is set ON by the decision module only if the process is the incremental dumper. This is a signal, if ON, to the dump module that a record of any data dumped should be written into the hierarchy.

   5) consistent dump switch (CSW) - this is a signal to the dump module (if ON) that the hierarchy sub-tree defined by this entry is to be dumped in a consistent state.

   6) secondary dump switch (SSW) - this is set ON by the decision module if the secondary copy of the segment defined by the associated entry is to be dumped.

2b. Date/time the entry was read from the hierarchy - This date/time is used to update the date/time-the-entry-was-last-dumped in the hierarchy entry information if that act is required (update switch ON). The time dumped must coincide with the time read since the dumped version may differ from the hierarchy version after that time. The date/time read is set to the current time immediately preceding the getentry request.

2c. Packed switch - The entry data is returned from getentry as a structure. Before dumping of the entry can occur, this structure must be converted to a packed bit string with certain format (see section BH.4.03). Since one entry may be dumped as part of more than one preamble the packed switch is used to signal that (if ON) the packing operation has been done and that this string exists elsewhere in the position segment.

2d.e. Slot number - The slot number is used to position the entry within the directory from which it was extracted. A directory consists of any number of links, branches and possibly a common access control list (CACL). A slot number of zero defines the entry as the CACL for the directory, -n, the nth link and +m the mth branch. The slot number is used in the getentry call to retrieve any entry from the hierarchy. It exists in 2 forms, character and fixed point binary. The character representation is the following:

">sdddddd"

where s is either "+" or "-" and d is either a digit or a blank. If blanks occur they must be to the right of all digits. Preceding zeroes are allowed. An example might be

>+413bbb

for slot number plus 413 (i.e. a branch). Another example is >-04bbbb for the link with slot number minus 4 (the 4th link).

2f. Pointer to packed bit string - Once the entry information has been converted to its bit string form it is placed in the allocation area, the packed switch is set ON and this pointer set so that the bit string may be referenced.

2g. Pointer to entry structure - Whenever an entry copy is to be returned by a call to getentry, space is allocated for the storage of the structure. This pointer is then set so that the entry items may be referenced by the process.

2h. Current entry count - The entry is either of type branch, link or CACL. The current entry count is returned by getentry and specifies the current number of slots in the directory of the same type as this entry.

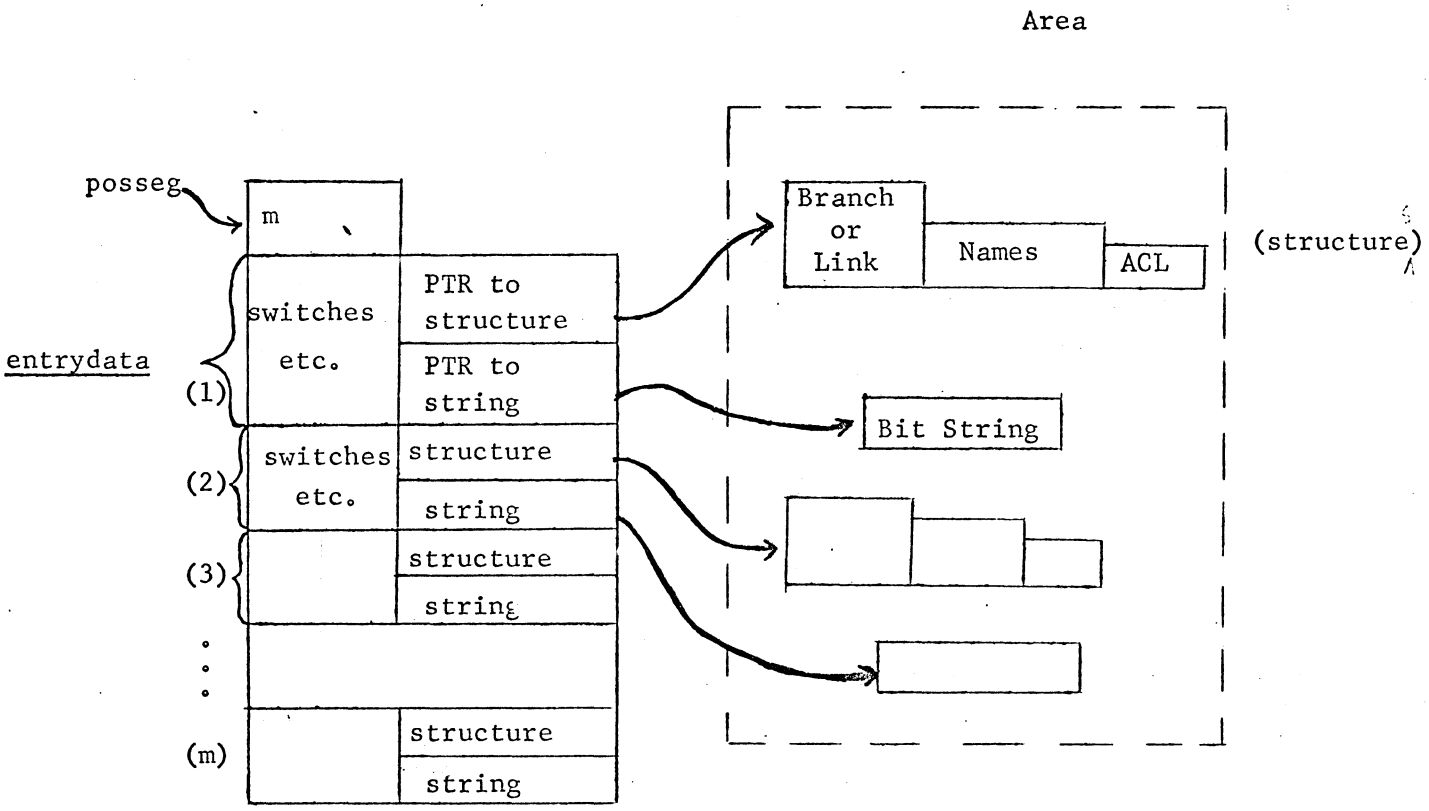
2i. Date/time count last changed - This date/time, returned by getentry, specifies when the current entry count (2h) in the directory last changed.

3. Area for allocation - Whenever a new entry structure must be fetched via getentry or whenever a structure must be converted to bit string form, space is allocated within this area for that purpose.

The following the PL/I declaration for the position segment:

```
dcl 1 posseg ctl (psp),          /* position segment */
    2 m fixed bin (17),          /* current scan depth */
    2 entrydata (maxen),
    3 (sdw,dsw,esw,usw,ssw,csw)
      bit (1),                   /* switches*/
    3 dtread bit (72)            /* date/time read from hierarchy /*
    3 packedsw bit (1),          /* =1 if entry has been packed */
    3 slotno fixed bin (17),     /* slot number (integer) */
    3 slot char (8),             /* slot number (character) */
    3 strptr ptr,                /* pointer to packed string */
    3 itemsp ptr,                /* pointer to items structure */
    3 date bit (72),             /* type max update date */
    3 count fixed bin (17);      /* type max */
    2 var area ((260000));       /* area for entry allocations */
```

Diagram of the Position Segment



A preamble record is formed by the concatenation of all entry bit strings pointed to in

entrydata (1) through entrydata (m).

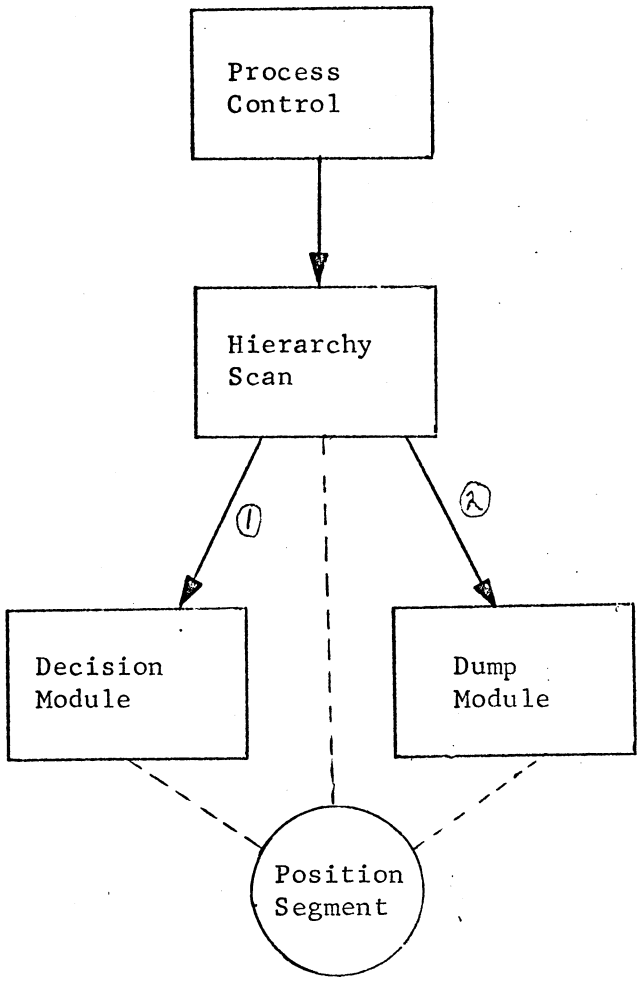


Figure 1

Block Diagram of the Dumping Process

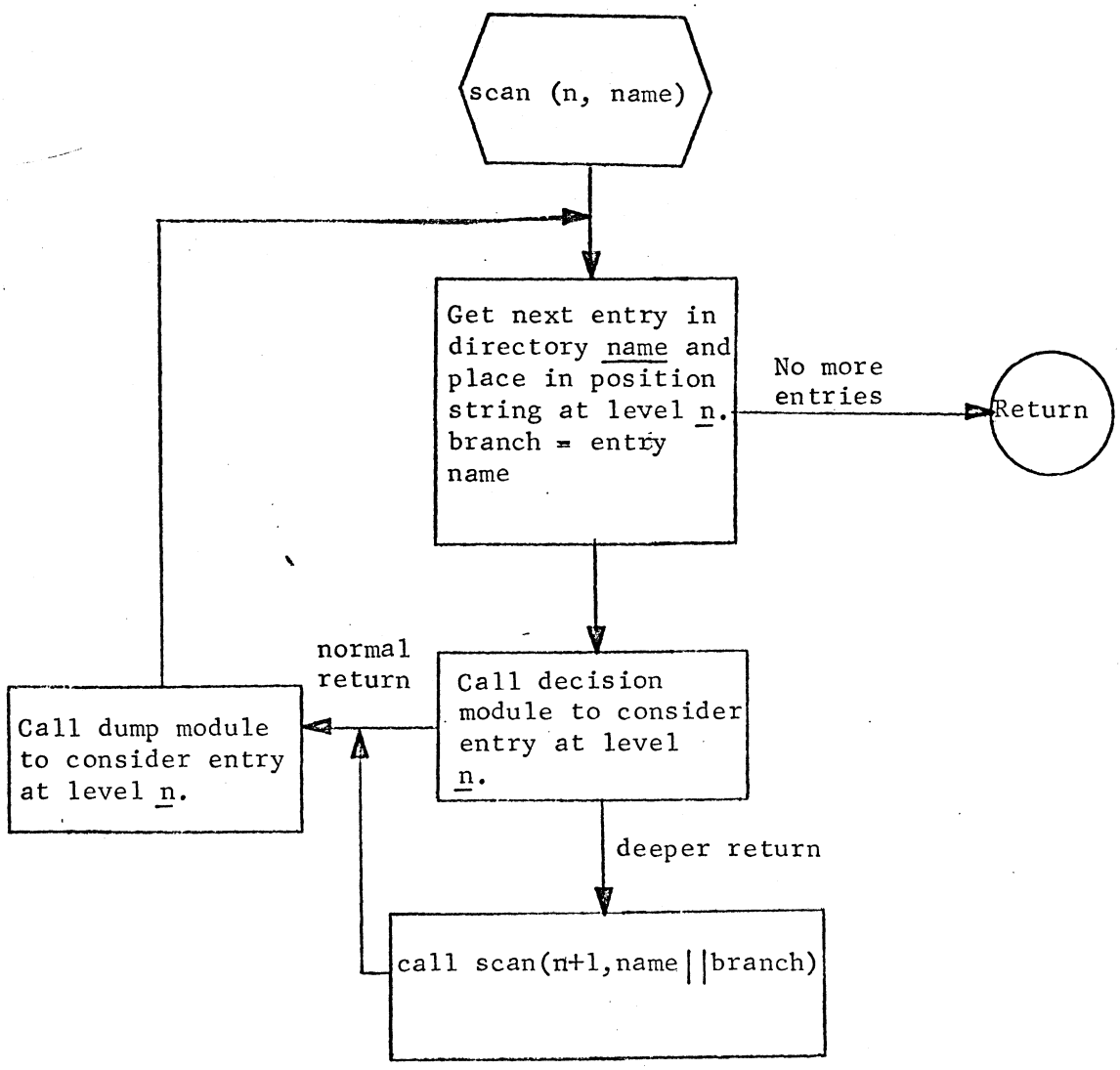


Figure 2

Hierarchy Scan Module