Published: 07/28/69

Identification

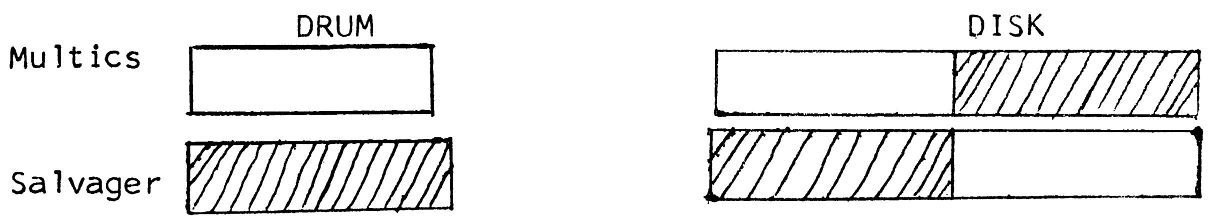
Salvager
N.I. Morris

Purpose

The Multics Salvager is a set of programs designed to be run after a Multics system crash in order to restore the integrity of the file system hierarchy. It is run as a separate bootload immediately after a system crash and emergency shutdown. The entire directory hierarchy is scanned recursively by the Salvager and all inconsistencies are noted and corrected. At the completion of the salvaging operation a normal Multics warm boot should be possible. The only segments lost will be those that were in a user's process directory. Pages of segments may be lost if they reside on illegal or conflicting device addresses.

The salvaging environment

The Multics Salvager requires secondary storage of its own (for paging purposes) orthogonal to the secondary storage used by the Multics System. Yet, the Salvager must be able to access the secondary storage used by the Multics System in order to examine and correct the Multics file system hierarchy. Thus, a partitioned storage scheme has been devised accommodating at least 2 separate partitions, one for normal Multics operation and one for the operation of the Salvager. This scheme allows a system running in one partition to access secondary storage records in another partition, but it prevents the file system from assigning pages for the running program in anything but its own partition. To give an example of this scheme, assume that the Multics partition uses all of the fire hose drum and one group on the disk, and that the Salvager partition uses a second group on the disk. The secondary storage available might be diagrammed as follows:



The shaded areas represent areas of secondary storage not available for paging purposes but accessible as data.

The Salvager is supported by a subset of the Multics System. This subset includes all of page control but not directory control. This provides a virtual memory environment similar to that used by the Multics Checker. In this environment, all segments comprising the Salvager and its data bases will remain active since no directory hierarchy exists for the Salvager itself. Thus, the Salvager is capable of supporting more program and data than can fit in the available core storage, but it does not have the capabilities of a full Multics System (i.e., it cannot take segment faults).

The only modification necessary to be made to the standard Multics System in order to support the Salvager is that of the partitioned secondary storage. This is accomplished in an extrememly simple fashion by adding an extra parameter to the "INTK" card in the BOS configuration deck. This parameter is a four character partition name. This name will be searched for on a new type of BOS configuration card, the "PART" card. This card has the following format:

```
    PART NAME DRUM_FIRST_REC DRUM_NREC DISK_FIRST_REC DISK_NREC
```

For the partitions described previously, these cards might appear as follows:

```
    PART MULT      0          10000       0          10000
    PART SALV      0            0       10000        10000
```

The "DRUM" and "DISK" cards appear in the BOS configuration deck as usual. Under Multics, "initialize_dims" searches for the "PART" card specified on the "INTK" card, as well as the "DRUM" and "DISK" cards, and initializes the file system DIM's accordingly. The drum DIM must be initialized even if the partition specifies no drum storage, since the Salvager requires drum access even if it does not page onto the drum. Note that the "DRUM" and "DISK" cards specify the total configuration of secondary storage, while the "PART" card specifies the portion of that configuration to be used for paging purposes.

Manipulation of Data Segments in the Salvager

Since the Salvager does not have a full Multics system available to it, it cannot access the directories in the Multics hierarchy in a straightforward manner. Segments cannot simply be initiated, then referenced. A segment, therefore, must be accessed in a rather indirect manner. If it is known where in secondary storage a segment resides, that is, if the secondary storage device address of each page of the segment is known, then the contents of that segment can be referenced. This is done by manufacturing an Active Segment Table (AST) entry and a page table for the segment, and by filling in the page table words with the device addresses. If an SDW exists which points to that table, then references through that SDW will automatically cause page control to retrieve pages of that segment, even though the secondary storage addresses are not part of the salvager's partition. When work on the segment is completed, the pages of the segment are written out by a call to "pc$write." Those pages modified will be rewritten onto secondary storage (in the Multics partition); those not modified will simply be deleted.

Since the Multics directory hierarchy is recursively tree-structured, and since the Salvager searches this hierarchy recursively, it is necessary for the Salvager to have access to more than one segment in the hierarchy at a given time. The maximum recursion depth has been arbitrarily set to 32, so the Salvager must have the capability of accessing 32 segments simultaneously. The segment accessing primitives of the Multics Salvager provide this capability by reserving 32 special AST entries, 32 page tables, and 32 SDW's for the purpose of accessing directory segments recursively. A push-down list is maintained which is "pushed" whenever an inferior directory segment is to be accessed and "popped" when that segment is no longer needed. These primitives fill in the appropriate special AST entry and page table, given a directory entry including file map, making access to that segment possible.

It is not possible for the Salvager to create new segments in the Multics hierarchy. If a directory is to be completely replaced with a new copy, that copy must first be created in a temporary segment (residing in the Salvager's partition). Then, it is copied over the original. It is intended that only one segment need be copied at a given time, hence only one 65K temporary segment is provided in the Salvager for this purpose.

Salvaging Directories

The most common problem with directories is the presence of locked directories in the Multics hierarchy. The presence of a lock set in a directory may indicate that the directory was in the process of being read or in the process of being modified. If the directory was simply locked for reading, the lock can be reset by the Salvager and no further action need be taken. However, if the directory is in the process of being modified, there are many tests which should be performed to insure the integrity of the directory. Currently, there is no way to determine if a directory was locked for reading or modification, so the Salvager must assume the worst case. If a lock is found to be set, the Salvager will rebuild the entire directory. Such items as branch counts, link counts can and should be verified. Other items such as the CACL and the hash table can also be examined for consistency. The initial Salvager is designed to only verify the counts. More sophisticated designs will appear later.

Salvaging Directory Entries

As each directory entry is examined, certain actions must be taken depending on whether the entry is a branch or a link and, if a branch, whether that branch is a directory or not. The most common problems with branch entries will probably prove to be inconsistent file maps and active entries. For every branch entry in a directory, each address in the file map must be validated. This is accomplished in the following manner: Two free storage bit maps are maintained by the Salvager (in addition to the one for the salvager partition maintained by page control). One of these is the free storage bit map in use by Multics at the time of the system crash or shutdown. The second of these is initially empty (i.e., all records in the Multics partition are marked as free). The first bit map is checked to insure that a device address is "protected" (i.e., indicated as being used). The second bit map is checked to insure that the device address has not already been used. The first occurrence of a device address is considered by the Salvager as the only valid occurrence of that address. Thus, when the Salvager discovers a reused record, that address will be made null, setting the contents of that page of the segment to zero. At the completion of the salvaging operation, the second bit map is the one written back onto Multics secondary storage. Thus, device addresses previously indicated as being used, but not appearing in the file map of any segment are freed up.

Active entries are those with a non-zero AST entry pointer. This condition indicates that the segment was active at the time that the Multics system crashed. New pages and modified pages of this segment will be lost if a successful emergency shutdown has not taken place. The only action the Salvager can take in this case is to zero the ASTE pointer. In addition, the Salvager will rebuild the directory containing this entry.

Other items in a branch entry which are validated include the current length, the device i.d., and the move device i.d. The current length should always be equal to the highest non-zero page of a segment. The only legal device i.d.'s are currently 1 (drum) and 2 (disk). Illegal device i.d.'s will cause the segment to be truncated. Such items as the entry names and ACL of each entry can also be examined. The initial Salvager does not concern itself with these, however.

Reporting Errors Found While Salvaging

The initial Salvager reports all errors found in the Multics directory hierarchy on the on-line printer by using the "formline" routine. Error messages for each segment are preceded by the full path name of the segment being salvaged. This name is maintained in static storage. As the salvager "pushes" recursively into the Multics hierarchy, name components separated by ">" are added to the segment path name. As it "pops" out of the hierarchy, name components are removed. Thus, the correct path name of the segment being salvaged is always available. It is intended that error comments will be self-explanatory. (see MOSN 59).

Validation of Pointers

Directories contain many pointers to different types of data structures. The validity of each of these must be subject to suspicion when referenced by the Salvager. In order to prevent unexpected faults or scrambled directories during salvaging, each pointer must be validated before use by the Salvager. The most important criterion for a pointer is that it points into a page assigned in the Multics partition. It may not point to an unassigned page since a reference to such a page would cause the Multics file system to assign that page to the Salvager partition. Also, the data structure pointed to by the pointer must not reside on unassigned pages, and the address of the end of the data structure must not be out-of-bounds. A Salvager primitive exists which will validate a pointer. The size in words of the data structure is also supplied as an argument to this primitive. Each entry in the file map corresponding to a page occupied by the data structure is examined to insure that that page is assigned. In addition, this primitive checks relative pointers to insure that they are not zero. An error code is returned if any of the above criteria are not met.

Conclusion

The initial Multics Salvager is designed to correct articipated problems in the Multics directory hierarchy after a system crash. It is expected that there will be several unforeseen problems that will appear when the Salvager starts to be used extensively. As experience is gained in the use of the Salvager, some of the algorithms described above may require modification in order to correctly handle these contingencies.