To: MTB Distribution

From: A. Kobziar

Subject: New Storage System Salvager Implementation

Date: 3 September 1975

The following MTB proposes a three stage salvager implementation for the new Storage System (NSS). The goal of this process is to achieve an efficient and reliable hierarchy repairing facility as described in MTB-220. However, since MTB-220 proposes drastic structural and operational changes, cost calculations will be done on a running NSS system before the decision to implement is made.

1) The first step is to implement a NSS salvager as quickly as possible. This will consist of the creation of a hardcore partition during bootload as described in MTB-213 and the writing of a volume (disk pack) salvager which will check for reused addresses and reconstruct the volume map. The current salvager will be modified to salvage the NSS directory hierarchy by removing the code which checks file maps. Metering will be added to this salvager to accurately calculate the cost of the MTB-220 proposed structural changes.

This salvager will be integrated into the Multics boot tape. A BOS command of the form "BOOT SALV LONG" will be provided. The sequence of operations will be to first salvage all disk packs and then to salvage the directory hierarchy. The directory salvager will access each branch's volume table of contents entry (vtoce) to verify the uid pathname as well as record the records used. Completion of salvaging will proceed directly to the rest of boot and to initializer command level.

As soon as this salvager is ready, NSS will be installed on CISL service. Exposure of NSS to users will not only reveal bottlenecks, but will also show what types of hierarchy errors occur. It will also indicate if a slowdown of directory control due to the inclusion of structure checking is acceptable.

2) The second step is to change the directory structure as proposed in MTB-220. Directory control will be recompiled, but the new structure fields will not be used until step 3.

The directory salvager will be rewritten to have the following properties:

a) It will operate on the new structure, but the code for checking the new fields will be commented out.

b) The interface will be changed to one supporting MTB-220. The directory salvager will not recurse down the hierarchy, but rather will operate on one directry only. It will rebuild the directory into a temporary segment without modifying the original. Its caller will the either move or discard the rebuilt version. Arguments to this salvager will include only information avaliable from a directory's active segment table entry; thus no vtoce references will be made by the salvager. Since this drops quota verification from the salvager's capabilities, some alternate method must be implemented for this purpose.

c) A new salvager driver will be written which recurses down the hierarchy, activates and deactivates directories, and replaces old directories by the rebuilt versions.

3) The last step is to change directory control to fill in the new structure fields and to check the values during normal operation. Code will be added to directory control which will trigger a rebuild when an error is detected. The salvager driver from step 2 will be modified to only pack salvage the root physical volume and to directory salvage some system libraries before going to command level. There will be an initializer command to salvage the rest of the hierarchy if needed/wanted.

The directory control checking described in MTB-220 will be implemented in a stepwise fasion. At each step a cost/performance evaluation will be made. A version which will become the field standard will be chosen from these evaluations.

The volume salvager will be modified to handle reused addresses on directories differently from thoes occuring on segments, as proposed in MTB-220. Also, integration with new backup will be done at this time.

(end)