To: Distribution

From: Steve Herbst

Subject: Multi-Segment Files

Date: 11/15/74

REASONS FOR A HARDCORE IMPLEMENTATION

The ultimate goal is for a multi-segment file to be one branch. Multiple VTOC entries in the branch tell where the components are. The components are implicitly labeled 0,...,n-1 but there are no names on them as such. Two broad advantages are gained by this kind of implementation. First, nobody has to worry about the consistency of redundant information. Second, it is impossible to perform directory operations on components.

Some of the problems caused by redundant information are:

1) The ACL and initial ACL of the MSF directory and the ACL´s of the components have to be updated simultaneously. Modifying a component´s ACL sometimes requires temporarily forcing access on the MSF directory.

   In a hardcore implementation, an MSF has one ACL and no initial ACL.

2) Every component has a current length, max length and bit count. The max lengths have to be the same and all but the last component have to be full. (File dims perform calculations based on these facts.) The bit count of an MSF is an implicit number calculated when needed from the bit counts of the components. The current length of an MSF is an implicit number calculated when needed from the lengths of the components and the records used by the MSF directory. A change in any component´s bit count has to be reflected in the bit count author of the MSF directory.

   In a hardcore implementation, current length and bit count of an MSF are stored in the branch. max length is the max length of the whole MSF; a component max length is also needed.

3) date-time modified, date-time dumped and date-time used of components have to be propagated up to the MSF directory.

4) Ring brackets of components can be different, unless processes of validation level unequal to that of component 0 or a segment (SSF) are forbidden from creating components.

5) Safety switches on components have to be the same so that deletion affects an entire MSF.

Problems caused by components being segments:

6) MSF´s can be made inconsistent by deleting, adding and renaming components. None of these operations affects the MSF directory´s branch information. For example, renaming component "2" of a six-component MSF can make user programs think that only the first two components exist.

7) To shrink an MSF to an SSF or grow an SSF to an MSF, the user must have sma access to the containing directory. Furthermore, these conversions change author and unique ID.

No hardcore implementation of MSF´s can be completely transparent, ie. allow the user to treat an MSF like a segment. Hardware limits the size of pointer offsets and thereby prevents using a pointer to an MSF the way one can use a pointer to a segment. msf_manager_ will still be needed to get pointers to components. The internal workings may change, however. File control blocks may prove unnecessary when the information they presently contain is in the MSF branch.

Programs differentiate an MSF directory from a non-MSF directory by the fact that the bit count field (msf indicator) is non-zero. They differentiate an MSF component from another segment only by the fact that it is contained in an MSF directory. The msf indicator can be changed easily and should not be relied on. A hardcore implementation of MSF´s assumes a unique MSF branch type. This branch type is set in ring 0 by programs that create an MSF from a segment and vice-versa. These programs have to go inside the MSF branch and maintain multiple file maps.

Our discussion of a fictitious hardcore implementation of MSF´s ends here. The necessary hardcore changes must wait until resources are available and the new storage system is in place. The new storage system should allow room for an MSF branch type. When hardcore MSF´s are implemented, the kind of user ring MSF described in the next section can continue to exist. msf_manager_ and the file dims can be made to work interchangeably on both kinds of MSF and differentiate between them by the branch type.

USER RING IMPLEMENTATION

Most of the problems listed above can be faced in a user ring implementation by going to some extra trouble. The one that can´t is the vulnerability of MSF components to directory operations. There is no way to make user commands and subroutines such as set_acl and adjust_bit_count_ refuse to work on MSF components. Nor can a user be prevented from creating segments in an MSF directory or building his own MSF´s. Installed commands and subroutines should work correctly on MSF´s but the individual user must be responsible for the results of any unorthodox methods that he uses.

Answers to the problems listed in the previous section are:

1) The msf_manager_ entries acl_add, acl_delete, acl_list and acl_replace handle MSF ACL´s correctly. The ACL commands (set_acl, etc.) call these entries for MSF´s and force necessary access on the MSF directory. Since the MSF directory´s initial ACL is the same as every component´s ACL, a newly created component has the right ACL.

2) msf_manager_$adjust handles the bit counts and bc_authors of components correctly. The adjust_bit_count_ subroutine and therefore the abc command calls msf_manager_$adjust for MSF´s. truncate and set_bit_count should be changed to call $adjust.

   msf_manager_ derives the max length of a new component from the max length of component 0 or of a segment it is growing into an MSF. All components must therefore have the same max length. Setting the max length of the MSF directory is not allowed.

   The current length and bit count of an MSF have to be calculated when they are needed by looking at all components and the MSF directory.

3) date-time modified, date-time dumped and date-time used are automatically reflected in the parent directory of a segment. A component is only dumped when that component changes.

4) msf_manager_ should check validation level when it creates a component of an existing MSF. A new entry, set_ring_brackets, should be added to msf_manager_.

5) delete_ on an MSF deletes components and any other segments in the MSF directory by calling del_dir_tree_, which forces all safety switches to zero. The only safety switch that counts, therefore, is the one on the MSF directory.

There are no solutions to these:

6) Nothing prevents a user from mistreating MSF components. status should warn of non-component segments in an MSF.

7) The need for access on the containing directory to grow or shrink an MSF is an instrinsic problem caused by the discontinuity between SSF and MSF. A branch has to be created in order to convert one to the other.

A DOUBLE STANDARD

Altering the insides of an MSF is not always destructive. In many cases, it is advantageous to create by hand a special kind of MSF that cannot be created by msf_manager_ or a file dim. System programs should work for these MSF´s whenever it is reasonable for them to do so. A definition of what constitutes a multi-segment file ought to be general enough to encompass these deviations. Within the definition of MSF´s respected by the system we can enclose a "standard" definition of MSF´s created by the system.

The following set of rules is proposed:

1) The bit count (msf indicator) of an MSF directory is non-zero. System programs depend on this fact to recognize an MSF.

   STANDARD: msf_manager_ keeps the msf indicator equal to the number of components when creating components or adjusting bit counts. The status command reports an inconsistent msf indicator.

2) Components are segments and links in the MSF directory. Links are chased, except when deleting or copying. The status and list commands and the file dims should be made to chase links.

   STANDARD: msf_manager_ does not create links.

3) Components are named 0,...,n-1. These names are necessary so that msf_manager_$get_ptr can find the components. Additional names are allowed. If a component has the names i<n and j<n, the MSF effectively contains two copies of the component.

   STANDARD: msf_manager_ does not put additional names on components.

4) All components have the same max length and all but the last are full. Targets of links are exceptions.

STANDARD: If the segment from which an MSF is grown has max length equal to sys_info_$max_seg_size, as is true for a segment created by msf_manager_, that is the max length of every component. copy_seg_ causes the created MSF to have the same component max length as the original.

5) ACL´s and ring brackets can be anything. Those of targets of links, certainly, can be anything. Programs that call msf_manager_$get_ptr should only stop looking if the error code returned is error_table_$noentry.

STANDARD: Components created in the MSF directory take their ACL´s from the initial ACL of the MSF directory. ACL entries in msf_manager_ maintain consistent ACL´s, and ACL commands call these entries.