MEMORANDUM

TO: Prof Saltzer, Prof Corbató, Steve Webber, John Gintell

FROM: Roger R. Schell

SUBJECT: Proposal for Improved Directory Format

DATE: April 12, 1971

1. Plans are currently being made for changing the Multics directory format to provide a variable length file map. I would like to propose that two additional changes be included:

   a) Move all the file maps into a group of a few segments to provide a convenient environment for secondary storage reconfiguration--including dynamic file migration.

   b) Simplify the design for locking directories by moving the lock variable into the active segment table (AST) and possibly reduce the I/O activity for directories by concentrating the frequently modified information.

2. To remove a portion of secondary storage from use it is necessary for reconfiguration procedures to find all the file maps that contain addresses within some range. With the current design this would require an exhaustive search of the hierarchy to look at the file map in each directory. If the problem of finding the file map is avoided (e.g., by using an address hash table as proposed for drum multilevel) then there is a multiple copy problem -- reconfiguration primitives can access secondary storage directly by using absolute device addresses, while other processes may make a Multics segment reference that simultaneously access the same location using the file map.

3. To provide a more convenient environment for secondary storage reconfiguration it is proposed that all the file maps be placed in a single file map segment (perhaps implemented as multiple segments) so that they can be searched directly: for each device a storage map (indexed by device address) can then be provided with a relative pointer to the file map entry for each address. Each entry in a directory, of course, has a relative pointer to the file map for the entry. The file map segment should also contain the AST pointer (currently in the directory), and changes in file maps should be under a global lock, possibly the current AST lock.

4. With the proposed file map segment, implementation of secondary storage reconfiguration is aided by including with the file map a relative pointer to the file map for the parent and a pointer (relative to the base of the parent directory) to the

directory entry for the segment. To facilitate a general dynamic file migration the file map entry for each page should include a current device id and a "new" device id, if any. To move a page, the segment is made active, the page is read into core, storage on the "new" device is assigned and the page is written out. Such a segment can be put at the head of the list for deactivation, and the file map updated with the new address upon deactivation.

5. The proposed file map segment(s) would be retrieved from secondary storage and created with AST entries during system initialization in a manner similar to that used for the FSDCT. The FSDCT could record its address and length. Variable length file maps should be managed and allocated in a manner similar to variable length AST entries with the processor retrofit. A "static" cleanup facility to group together all file maps for a single directory would probably be useful. In addition the file maps can be grouped according to the highest speed device they contain in order to possibly reduce the number of page faults, for example, when using the standard search path through directories on the drum. This proposal for a file map segment does introduce the potential for an additional page fault when referencing a directory entry.

6. A file migration scheme has been proposed where pages on the drum would be "stand-ins" for disk pages--the file map would continue to reflect the disk address. The primary intrinsic benefit of this approach is that pure page moved back down to the disk need not be written out; however, this I/O should be relatively infrequent if file migration is a viable concept, and the disk storage is of course not available for use while the page in on the drum. I think this design with core as a stand-in for drum which may in turn be a stand is for disk can be more complex (with its "multiple copy" problem) than a true multilevel where the page is actually moved in the file map, as more easily done with the proposed file map segment.

7. The current design for locking directories contains a "probabilistic design" for changing directory access to prevent accidental writing in a directory. Write access is required to lock the directory but segment fault assumes that only locked directories have write access--the design assumes there is a low probability that a directory with write access will be deactivated, leading to a segment fault when trying lock. It is proposed this problem be solved by placing the lock at a location external to the directory, specifically as part of the AST entry. This requires that a directory remain active while it is locked. In the current implementation a "lock array" is maintained so locks can be reset if there is a crawl-out: this overhead could be eliminated for directory locks since the locks could be searched for directly in the AST. I also note that when searching a directory only the lock and the modify switch are actually written, and if both were in the (wired down) AST there

what happened - there is a failure?

would be no I/O required to write out the directory page.

8. I have made a series of proposals for changes to the directory format to provide an environment for reconfiguration of secondary storage and a simpler locking design. This proposal is intended to stimulate thought and is not intended to provide the answers to all implementation details; however, I will be happy to discuss the details with anyone who finds the proposal interesting.