To: Distribution

From: T. H. Van Vleck

Date: July 7, 1975

Subject: Adding a "Property List" to the Branch

INTRODUCTION

From time to time there have been proposals to add data items to the directory branch for the convenience of application subsystems. Although several of these proposals have been of some interest, we have never been able to justify the effort required to modify the structure of directories on the basis of the improvement to a single application; and the difficulty of designing an extension which would satisfy the conflicting needs of more than one subsystem also led us to defer consideration of such an extension. This memorandum proposes that we now implement a mechanism which appears to satisfy all known needs in a straightforward way.

PROPOSAL

Briefly, the proposal is that an optional item be kept by the file system for each branch, called the *property list*. This item is a list of pairs associated with the segment or directory. Each pair contains a character-string *name* and a *value* associated with the name. No name can appear twice on a branch's list.

Structure of the Property List

The pairs on a property list will be allocated in the directory just as ACL entries are, but no attempt will be made to common the space if more than one branch has the same name-value pair. The branch will contain an 18-bit forward relative pointer to the first element on its property list (if any). The property list is threaded one way only since it must always be searched completely before adding an item and because order of items on the list is not significant.

The property name may be up to 32 characters long. The value of a property is a block of storage which may be up to 240 words (960 characters) long.

Primitives for Manipulating Property List

New hardcore primitives will be added for the maintenance of the property list, so that application programs can add or delete properties, change the value of items on the property list, and so forth. In the examples below, the "pointer" form of calls is shown. Similar calls which accept dirname and ename will also be provided.

Access control to the property list will be similar to access control restrictions currently enforced on the bit count of a segment. That is, modify access on the segment itself is required, and status on the containing directory is not required.

All of the primitives which manipulate single properties handle property values by means of a pointer and length pair. For return arguments, the pointer must be word aligned and point to sufficient storage to contain the largest possible property; the actual length will be returned,.

PUT_PROP_

This entry puts a name-value pair on the property list of a branch. If a property with this name existed already, its value is replaced and the old value returned. Otherwise a new entry is made.

```
     dcl put_prop_ entry (ptr, char (32), ptr, fixed bin,
         ptr, fixed bin, fixed bin (35));

     call put_prop_ (p, name, newp, newl, oldp, oldl, code);
```

If the property did not exist before, oldl will be zero and code will be error_table_$created_property. It is an error to attempt to create a property with no value; if newl is zero, code will be error_table_$badarg and no action will be taken.

DEL_PROP_

This entry deletes a property.

```
dcl del_prop_ entry (ptr, char (32), ptr, fixed bin,
    fixed bin (35));

call del_prop_ (p, name, oldp, oldl, code);
```

The old value is also returned.

GET_PROP_

This entry searches the property list for a given property.

```
dcl get_prop_ entry (ptr, char (32), ptr, fixed bin,
    fixed bin (35));

call get_prop_ (p, oldp, oldl, code);
```

If the property does not exist, oldl will be zero and code will be error_table_$no_property.

PUT_PROP_COND_

This entry does a conditional put_prop_ operation. It is indivisible with respect to other put_prop_ operations and so can be used to manipulate "semaphore" variables.

```
dcl put_prop_cond_ entry (ptr, char (32), ptr, fixed bin,
    ptr, fixed bin, ptr, fixed bin, fixed bin (35));

call put_prop_cond_ (p, name, testp, testl, newp, newl,
     oldp, oldl, code);
```

If the entry has a property named "name" and the value has length "testl" and matches the test value pointed to by "testp," then the value will be changed to the new value and code will be returned zero. Otherwise, code will be either error_table_$no_property or error_table_$match_fail; if the property exists at all its value will be returned.

LIST_PROP_

This entry obtains the whole property list for a segment.

```
dcl list_prop_ entry (ptr, ptr, fixed bin, fixed bin,
    fixed bin (35));

call list_prop_ (p, storage, max, actual, code);
```

The pointer "storage" points to a block of space large enough to contain "max" properties. If there is sufficient pad in the hcs_$status_ structures, it might be nice to return there the number of properties which a segment has.

Modifications to Existing System Software

Not many changes to the current system are necessary, since the property list is primarily designed for user applications.

COPY

The copy command should be able to copy property lists, perhaps under control of a new control argument, and certainly when "-a" is specified.

BACKUP AND RELOAD

The dumper and reloader will have to handle property lists. This change is probably the hardest to make to support the new facility, simply in terms of integration with the existing system.

It has been proposed that a new item be added to the branch giving the tape reel number on which a segment was dumped. The property list could, of course, be used to store this information. Since this would mean that every segment in the system had a property list, this may not be a good idea; probably space should be allocated in the branch itself. But if we had the property list available, it would be possible to experiment with new versions of the dumper and reloader which used such an item, before modifying the branch structure.

SALVAGER

A fairly simple change to the salvager is necessary to insure that when a directory is rebuilt, the property list is rebuilt. Thread checking on the property list itself can also be done. If any error is detected in a property list, it will probably be acceptable to have the salvager drop the whole list.

COMP_DIR_INFO, SAVE_DIR_INFO

These tools should be updated to save and check the property list.

New_Commands

No new commands are strictly required by this facility, since the applications will provide whatever property manipulation functions they need. However, a few system commands may be useful for cases where one is setting up an application.

ADD_PROPERTY

This command would add a named property to a segment or directory. Probably it would be limited to ASCII values only. The star convention should work.

LIST_PROPERTY

The list_property command will obtain the property list and print it, in ASCII if the value contains only printable characters, and in octal otherwise.

```
list_property path

list_property path property_name
```

The star convention should be accepted.

DELETE_PROPERTY

This command would delete a named property from a segment or directory. Star convention should work. The control argument "-a" should be acceptable to delete all properties.

APPLICATIONS

A few possible uses of the property list are described below.

Subsystem Use

The most important use of the property list is in extending the Multics storage system directory so that catalogue data about segments can be kept associated with the segment. The ability to do this frees the subsystem which needs one small data item maintained from constructing a "parallel directory" with the attendant problems of access control simulation, salvage, etc.

FAST

The FAST subsystem's need to keep the language type (basic, fortran, or data) associated with a source segment is an example of a use for the property list. The language type will be kept as the value of the "FAST.lang_type" property of the segments. This solution is far better than having FAST attempt to keep a "directory extension" segment, or mapping and un-mapping segment names, or hiding the language type beyond the bit count, or changing the syntax of the OLD and NEW commands. At the same time, this usage does not preempt other subsystems from accessing source programs written by FAST and attaching their own properties.

If an even closer simulation of DTSS was needed, other data items could be stored on the property list of segments under FAST, such as data for the DTSS file system's access control mechanism.

CONSISTENT SYSTEM

The Cambridge Project's Consistent System wishes to record a "data type" for each segment catalogued by the system, so that the data-manipulation operations defined by the system may find the internal representation of data segments in the Multics storage system. Currently this is accomplished by maintaining a table with one entry for each branch in the directory, with attendant overhead. By using the property list, the "consistent_system" property could be used to store this information. The Consistent System is also a system which unifies many statistical and data-organizing subsystems. If any of these subsystems need a catalogue item maintained, the property list can once again be used, without conflicting with other applications. An informal registry of property names may turn out to be a useful idea in the long run -- for a start, naming the property after the subsystem should be adequate.

Volume Description Segment

The property list provides the solution to a problem which has arisen in the planning for the Resource Control Package and its predecessors. Although each Volume Description Segment describing a tape or disk pack should logically be a separate segment, in order to take advantage of the Multics protection mechanism, we have been reluctant to waste 1024 words of storage to store about ten words of actual data. By placing the volume-registration data on the property list of a zero-length segment, we make efficient use of storage. Any user application which has a similar need for a "mini-segment" will be able to solve it in the same way.

Courtesy Lock

The conditional-set primitive can be used to implement a new convention, the "courtesy lock convention." Commands which honor this convention would set the "Multics.courtesy_lock" property to the lock_id of the process. If edm, qedx, archive, etc. all respected this property, then the possibility of two users editing the same segment at once could be minimized. The editors could print a warning if the property could not be set, and continue anyhow (since often the editors are used for interactive search), perhaps with the "w" request disabled. Higher-level process-coordination schemes, including analogues for notification when a lock is unlocked, can obviously be proposed once the property list is in place.

Use by vfile

Currently the PL/I input-output system cannot reliably determine whether a multisegment file is of sequential or indexed organization. By adding the "Multics.io_data_type" property with appropriate values (and default assumptions) this problem can be eliminated.

Use by msa manager

The multi-segment area management programs used in the library maintenance software were concealing segment numbers within the bit count of the area segments for a while. This problem was solved by other conventions, but the property list would have been the natural way to provide storage for a small data item which could not be placed in the segment itself, but which had to be associated with the segment.

Use by Replacement for send message

There has been some discussion about replacing the current send_message command with a version which would make use of the ring-1 mailbox. Hooks for this application have been defined in the implementation of the new mail command. The method of storing the event channel and process ID associated with a mailbox was not completely specified: the property list of the mailbox may be just the place, since it does not require modifying the internal structure of message segments.

Use by Profile Manager

When abbrev rehashed the profile, the date for check_info_segs used to get destroyed. If the date were kept in the "Multics.cis_date" property of the profile, then check_info_segs and abbrev would not have to know of each others' existence. Furthermore, new applications which need a per-user data base could be written and debugged without the temptation to create yet another per-user segment.