Saltzer file - proposal relevant to Multics

April 4, 1973

Mr. Charles T. Clingen
Cambridge Information Systems Laboratory
Honeywell Information Systems
575 Technology Square
Cambridge, Massachusetts

RE: Consistent calling sequences to Multics system procedures

Dear Charlie:

I have finally gotten to dabbling in hcs_. A glaring deficiency in the system is the lack of consistency in passing parameters to the various entry points of hcs_.

It would appear that some of these inconsistencies have been incurred in pursuit of some sub-optimized concept of efficiency. I am going to have to produce large pieces of readable code on Multics and train programmers on my staff to readily write Multics code. These little uglies that have been perpetrated on Multics sub-system writers clearly defeat both goals. In order to use an hcs_ entry a programmer must know not only the name of the entry and the parameters passed into and out of that entry, but also know whether this particular entry point has a path name that is passed as a fixed length string with a count field, a varying string, a pointer to a fixed length string, a pointer to a varying string, etc. It appears that even a proficient programmer will have to constantly have a manual at his side to remind him which particular convention is in vogue for each entry point each time he has to code it.

We originally became interested in Multics because of the cleanliness of its design and the ease with which we could write subsystems. When we made the initial Multics examination we were taken back by the exorbitant price schedule Honeywell had chosen for the 6180. The only point that kept our interest was the unusual cleanliness of the system which was represented by how well everything seemed to fit together. That cleanliness could partially offset the hardware cost by significantly reducing our system development cost.

As we have looked deeper into the system we have found that the cleanliness of the original design has not always been followed. Somewhere between the writing of the overviews and the implementation of Multics the beautiful way in which the system meshed with its various components was at least partially subverted. hcs_ is one of the more serious of these I have discovered but there are others (e.g., I find myself executing entirely too many "new_proc" commands).

What I think Honeywell needs is a way of detecting these uglies, placing them in a priority list and acquiring the resources to fix them. This clearly is a Honeywell procedural matter which might be best taken up with the users group.

As far as hcs_ goes I think there are several solutions. I can think of two right off the top of my head. Provide a new set of consistent entry points which can coexist along side the older, inconsistent entry points which clearly should be allowed to survive for reasons of compatibility. If the use of the old entry points is discouraged the natural turn over in code will slowly reduce the need for their existence. Another alternative is to provide a slightly different philosophical approach to the problem. For example, you could have a short and long initiate call. The long call would return everything under the sun and would allow the description of entry points, etc. The shorter call might allow only a path name and possibly an entry name and would return only a pointer and a code. All other utilities use only the pointer returned by the initiate call to access various status data. Most of the inconsistency in argument passing occurs in the passing of character strings. This solution would isolate the use of character strings to the initiate procedure. It would then be inconsequential if we chose a very convenient but possibly slightly inefficient method of representing characters strings (e.g., as varying strings). While I am on that point I think it would also be nice if the procedure would simply expand a relative path name rather than require the input of absolute path names.

I am a novice Multics user and I really don't expect my suggestions would be the best way of fixing this problem. They are,however, indicative of the kind of solution which would be acceptable.

Well Charlie this makes two nasty letters. Maybe next time I can write you a nice letter. Incidentally, your responses to my complaints about MPM were good. The new revisions have been more helpful and I think the new chapter 4 is outstanding. I learned more from that section about what you can and cannot get away with in Multics than I did from all my other readings in the manuals. It is now a whole lot easier, to understand the more cryptic sections of the manuals. Incidentally, I might as well throw in one more MPM dig. Whoever wrote the description of abbrev did an absolutely abominable job. I really would like to know how abbrev works, but that description is just lousy (I guess I really didn't like it).

Sincerely,

Peter A. Alsberg

PAA/smb

cc: Jerry Whitmore