MTSS DESIGN NOTEBOOK

Appendix F

FROM: Louis Pouzin

SUBJ: Library Subroutines, Commands, File Organization

I. Library.

1.1. Data conversion programs.

In most classic languages, it seems that data-conversion is considered only during I/Ø operations. E.g. although (IØH) may be used for core-to-core conversion, no routines are provided in order to do so.

There should be a complete set of conversion routines, (as parts or not of a big pure common procedure) for the following conversion.

BCD (6 bits)	to and fro	BCD (8 bits)
BCD (6 or 8 bits)	to and fro	binary
BCD (6 or 8 bits)	to and fro	floating point (single and double precision)
fixed binary	to and fro	floating binary
Left justification		
Right justification		
Leading zeroes	to and fro	leading blanks
Character substitution	(1 character for another, until some terminating condition)	
Packing and Unpacking		
Text breaking into data words		
String concatenation from data words.		

1.2. Monitoring routines. User's identification.

Parameters such as date, time, processor used, file directory currently used, user's name and number, should be available through subroutines, for the average user.

For privileged users, there should be access to monitoring information, namely all parameters used to allocate the system resources, and all accounting information as to the resources spent and for what.

1.3. Utility routines.

Subroutinized sequences to determine length of variable calling sequences, type of arguments. Modification of the structure of data in calling programs, to meet the requirements of the system machinery. E.g. gather data lists, specified as a collection of arguments, into a single fixed presentation for disk, or console I/O. And converse.

1.4. Subroutinized string processor

Although such a facility is not crucial, it may turn out to be a mighty tool for improving man-machine interactions. Due to the user's console, most input-outputs tend to be in string structures, rather than in fixed dimensioned fields. Thus, it would be handy to call a SNOBOL like processor, with strings as arguments, for the execution of one, or a set, of rules. This may be compared to the availability of a string language interpretor at the program level.

II. Commands

2.1. SAVE, RESTØR

These commands will have to take into account the new concept of loading. SAVE will be more like an unbinding program rather than a dumping one. Some facilities ought to be provided so that the user could specify which common procedures he wants to be saved, and later on restored, as private procedures. The reason is that he may want to protect himself against possible common procedure changes.

2.2. Compilers

They should systematically accept free format input program, rather than card images. On the other hand, card images are not to be suppressed, as there will be always decks supplied from other installations. Not only a compiler could be called automatically to translate a program, no binary copy of which exists yet, but also, it would be very powerful to call a compiler, from inside a program in order to compile and insert the machine code corresponding to a statement, or a group of statements, provided as data arguments.

2.3. Files housekeeping

The concept of LISTF is to be revised. What the user needs is a semi-automatic secretary for taking care of his files. A list of files being printed out is just an example of question and answer that the user is likely to exchange with his secretary-program.

The basic behavior of a file-keeping command should be:
DØ something with FILES meeting CØNDITIØNS
DO may be LIST, PRINT, DELETE, CHMØDE, RENAME, ARCHIV, CØMPRESS, EXPAND, REQUEST, UPDATE, SEND (to some user), etc.
There is likely the possibility to call any command, from this file keeping command level.
CONDITIØNS may be any concatenation of simple conditions as:
before a date
this date
after that date
longer than x tracks
such mode
part of the name being: "a string"

Simple logical conditions would be linked by logical boolean operations such as: ØR, AND, NOT.

The file-keeping command should be able to associate to a file name a string of characters which would be used for dassification purposes. All strings could be stored in a particular file which one could associate with the file-keeper at one time. As several tag-string files could exist, this would allow to work on the file directory from various criteria of sorting.

2.4. MAIL

Inter-users communication should be improved so as to forgo the console constraint. A MAIL command could send a file to another user, or another group of users specified by some criteria, a all user. Users, who the information is sent to, would read a message when they log in, indicating that some mail is waiting for delivery. The same MAIL command, with a particular argument, would delete all messages waiting in the delivery pool, and write them into the user's directory. Then the user could trash the whole thing if he wants to.

2.5. EDITING

The following intends solely to enhance some points which may not be sufficiently well solved by the tools presently available.

2.5.1. - Formatted input. The development of string oriented languages, and compilers accepting free format input program, may reduce the problem of formatted input, but probably not to the point that it will vanish. By formatted input, we mean the modification of the input text according to some set of rules, such as:

.set this field to a fixed length,(expand or truncate)
.set next field beginning at a fixed position (tabs expansion)
.repeat field, (from the previous record)
.change repeated field, (for duplication into following records)
.continue into next record, with continuation flag
.ignore character, or field, although it is typed on the console.

2.5.2. - Sort control words. The only option handled sofar is the numeric part of the serialization field in card images files. It should be possible to set, temporarily, the sub-string of the character string to be used as a sorting key, either by context or by character count reference. Successive settings would allow the sorting on some key at a level, and on some different key at a different level; E.g. by paragraph number at a level, and by alphabetic title in each paragraph. Moving around pieces of files would amount to change an appropriate key-word in order to meet the desired order.

2.5.3. - Automatic creation of labels. As an implication of the previous requirements, it should be possible to set generators of labels.Up to now, only line numbers are generated that way. This possibility should be extended to alphabetic symbols, by having N generators, set to an initial value, and an increment. Whenever mentioned, these generators would yield their current value, in place of the part of the text which mentioned them; then the current

value would be incremented for the next use. In alphabetic, for example, the increment A could mean skip to the next letter, alphabetically speaking.

2.5.4. - Scope definition. For any request, print, modify, delete, sort, file, etc., it should be possible to specify the limits into which it is to be applied. As it may be akward to specify too many things in a single request, it might be more convenient to set two pointers into the file, as protection pointers. Their positioning would be set separately and independently, through the means of context reference, which might be the most suitable. Then no request would be allowed to act outside the boundaries set by the pointers, until they have been reset to other values.

3. File Organization

3.1. The linking facility of the new disk routines provides a way of referencing a file, or a pointer to a file. One could add the possibility to reference a file of pointers. Throught that mechanism, a logical file could be a physical collection of single, or compound, files. Reading a compound file would amount to open automatically a new file (the pointer file) whenever one isencountered. End of file returns would result in going on through the pointer file.

3.2. There should be a mode whereby any message sent to the console would rather be written into a file stored in the user's tracks. Symetrically, another mode would read any input message from a file. Modes should allow saving and restoring.

3.3. Several applications would be conveniently solved if the supervisor itself could write its own files, almost as a regular user. E.g. statistics too large to fit in the supervisor storage area are practically neglected, even if their importance is obvious.