[illegible]

Programming Staff Note 37

Computation Center

November 6,1964

FROM: Louis Pouzin

SUBJ: An Approach to a Standard Error Procedure in Chains of Commands

## 1. What the Problem is

When a command is executed, it may encounter some conditions which make it impossible to run to completion. Some situations may be handled by interaction with the console, as 'NEED' messages, or 'DO YOU WANT TØ DELETE' an R1 mode file. Some other conditions turn out to be fatal, as an illegal card in loading a BSS file, or at least a console procedure would be too complicated or hazardous in order to restart a normal execution. Appropriate messages are then printed on the console, and the command is terminated.

This visual procedure may be satisfactory enough when commands are executed as isolated programs. It is not satisfactory when they are part of a more general program which controls their execution by setting them into chains of commands. Indeed, a failure in a particular command may or may not be fatal for the chain; and depending on the circumstances, it would be necessary to stop, or execute a predetermined error procedure, or even just continue and ignore the failure.

Evidently, no systematic decision, built into the command, is satisfactory. Neither is it usually possible to modify a command from another command. In a sense this complete independence between commands is purposely carried out in order to keep larger flexibility in the system.

Actually there is a wide variety in the way commands terminate on errors. E.g.

- Go to CHNCOM with core image
- Go to CHNCOM without core image
- Go to DORMNT

- Go to DEAD
- Print 'TYPE START TØ GØ ON', and go to DORMNT. If the user types START, then go to CHNCOM.
- Protection mode violation.

## 2. An Approach to a Solution

Since commands are not pieces of a closed package, but rather an open list, incremented with additions from various users groups, and designed for any particular usage, it seems desirable to keep the set of conventions as thin as possible. This eliminates practically the idea of having commands returning an abundant collection of arguments, through some core A buffer, or written onto the disk. Even though such a possibility may be very valuable to one command, it does not seem to be flexible enough to make it a mandatory convention for all commands. The minimum requirement should be simple, cost few machine instructions, and not make any assumption as to the way an error procedure might be carried out. In other words, an error exit should be a standard exit, as straightforward as calling CHNCØM.

## 3. Some Suggestions

Another requirement is that an error exit should not disturb whatever has been already set by other commands into the supervisor buffers. Namely, command lists and command counter do not belong to the current command, and they should not be destroyed as a result of some unfortunate situation. On the other hand, the content of the current command buffer will not be of any use after completion of the command. Hence, it may be used to return to the supervisor enough information in order to initiate a standard error procedure.

E.g. the command might call NEXCØM as an error exit, which allows the very important facility of executing an extra-command not previously set in the chain. On the other hand, nothing else is destroyed whatever.

The drawback of NEXCØM is that it allows for replacement of only the first two arguments in the command buffer. Even if this restriction is bearable, or even convenient in some cases, it is unquestionable that the whole command buffer could be of better use.

NEXCØM being what it is, it is not suggested to change its behavior, because this would raise problems of compatibility. But one may think of using SETCLS, as:

```
        TSX    SETCLS,4
        PZE    BUF,,0
```

meaning: set the current command buffer to the content of BUF...BUF+19.

Let us notice that a symetrical call:

```
                TSX    GETCLS,4
                PZE    BUF,,0
```

could return into BUF...BUF+19, the whole content of the current command buffer.

This suggestion has the advantage of using existing calls by simply allowing the command list number of zero to mean the current command buffer.

In order to start the command, CHNCØM may not be used, since it would go to the next command in the chain. But a different call, such as:

```
        TSX    REPCØM,4            for REPeat CØMmand
```

could start the execution of the new current command. NEXCØM may be used, if one resets the first two words in the command buffer.

Through the simple machinery outlines above, it would be possible to call any 'extra' command, without disturbing the setting of the current chain.

## 4. Standard Error Procedure

A standard error exit from a command could be to start the command:

```
        ERRØR     arg 1         arg 2        ... arg n
```

which could be a core A or core B command. The argi's are arbitrarily set to whatever information seems useful to hand over in the particular error condition encountered by the command.

The ERRØR command would execute very few things. E.g. Print: ERROR BREAK-POINT. TYPE START TO GO ON and write a disk file containing a copy of: current command buffer, all command lists, and command counter. This file would be called by a special name, (ERRØR FILE) e.g., and created as temporary, i.e. would not be stalled by a track quota exhausted. Then go to DØRMNT.

By typing START, the user could ignore the error, and force the continuation of the chain. On the other hand, if he does not type START, he may 'SAVE' the present status. Then he can examine the disk file in order to know in which context the error occurred, and thereafter fix up the trouble and restart.

## 5. Tailor-made Error Procedure

In the above paragraph, we sketched out an elementary error procedure, which could be the standard one. But we did not mention a major feature of ERRØR which allows bypassing the standard procedure completely.

In effect, ERRØR would check for the existence of an ERRØR SAVED file in the user's directory, and if there were any, it would transfer control to the user's command, by setting a RESUME ERRØR with all arguments as set by the original command. Then a particular procedure could be executed for any particular error condition, including possibly an automatic restarting of the chain.

Such a procedure would allow for taking into account the context of an error, before making a decision on whatever salvage procedure should be selected.