FROM: T. M. Hastings, J. F. Kellwig, L. Poussin

SUBJ: Standardization of Disk Files for the Time-Sharing System

During the development of the time-sharing system approximately four ways of storing disk files have emerged. This first proposal is an attempt to curb this file population explosion in its infancy. Currently the following four types of files are in use:

a. BCD card images - 14 words per card image
b. BSS card images - 28 words per card image
c. BCD Variable length records - record length in first word (?GETDAU input and output routines).
d. BINARY words with no logical subdivisions.

Fortunately all four of these methods are stored physically in the same way; the disk hardware makes no distinction between BCD or BINARY.

It seems wise to limit the kinds of files to the four kinds listed above. The user should of course be able to write a file in any way he sees fit. However, if these were accepted as standard formats, the parts of the system which manipulate disk files would be written so as to handle these meaningfully and painlessly. It would then be recommended that all users conform to these standards. Moreover the system should always generate files in one of the four standard forms.

A convenient way to establish this standardization of file formats would be to strengthen the significance of the secondary file name. The secondary or class name at present has a rather ambiguous status, inasmuch, as it is important in certain contexts (e.g. load, file), of local significance in others (e.g. data files referenced by Mad and Fortran I/O statements), and meaningless in others (e.g., print, and disk editor control cards).

It is proposed that the class name be used to specify the format of the file in addition to the special modes implied by the names fap, madtran, bss, saved, etc. Thus the following sets of class names would correspond to the four formats outlined above:

a. MAP MAD MADTRN CARDS*
b. BSS*
c. BCD*
d. SAVED BINARY*

where the '*' denotes the general class name to be used for each format where a special name is unnecessary.

Further, I/O routines which simulate reading and writing tape should generate file names like

TAPE) MOD for MAIN OUTPUT TAPE 3

TAPE12) BINARY for TAPE TAPE 12.

In general, all files which are created by system programs for the user, where the user does not specify the file name, should use names with a terminal) or enclosing parentheses. (See Programmer's Guide, Pg. 24.)

There are a number of CMS commands which operate on a user's file. They are:

PRINT, COMBIN, SPLIT, INPUT, MOD

These manipulatory functions should be provided for all of the four standard file forms. There appear to be at least two solutions to allow this:

a. Have four separate commands for each of the five functions:

I. printing files
II. combining files
III. separating files
IV. generating new files
V. editing existing files

b. Retain **only the five commands**, one for each function. In order to perform the appropriate function on a particular form of file, the command would examine the class name.

If a CMS command is asked to operate on a file with an undefined class name, it would assume a file of BINARY form. The advantages of this second scheme are:

1. It continues the precedent of attaching special significance to class names.

2. It reduces the number of commands needed to operate on files from 19 to 5 (inputting a BSS file is not provided directly).

3. The user's file director indicates to him how his files are stored.

4. Much disk storage for CMS commands will be saved since much of the coding would be the same regardless of the form of file.

A disadvantage is that the command program would have to be changed each time a new class name is introduced into the system. However, it seems that this would not be happening so frequently that it would be a serious problem.