PROGRAMMING STAFF NOTE 43

CC-254 MAC-M-244

SUBJ: Proposal for Commands for Dynamic Resource Allocation

FROM: G. Schroeder

DATE: April 17, 1965

Introduction:

The need for a dynamic means of drawing time and track allotments from a common group pool has long been recognized by many and was made painfully apparent to all by the recent efforts to make reasonable time and track allotments on a group basis. In most cases, a group allocation can be made with reasonable accuracy, but division of these resources among individuals of that group is often a difficult - if not impossible - task. The need for crash efforts on the part of some and no needs on the part of others cannot be foreseen in time to set in motion the means for changing the information inside the system.

The commands here proposed would allow the system administrator A to allocate time and tracks to a group. The group leader L would then be totally responsible for how these allocations are used. L can make definite allotments to each individual, or he can make minimal allotments to each and leave the rest of the resources to be drawn by the individuals of his group when they need them. He can also mix these two policies by making definite allotments for some and allowing others to draw from the pool.

If a user is allowed to draw from the pool, the resources he requested are credited to him immediately. If he should draw more than he needs, he can return resources to the pool; and they will be immediately available to other members of his group.

Method:

At the beginning of each month A attaches to L's file directory and writes a special file, RESOURCE file, containing the total resources alloted to this group. This file is private to A. A then looks for a file, PARENT file, which contains the allocations for the individuals in the

group. The total allocations in this file are subtracted from the allotments in RESOURCE. If they are less than or equal to the allotments, the file RACCNT replaces the file called CACCNT. CACCNT is a part of the accounting files used by LOGIN. It is private to A; but it resides in L's file directory. If RACCNT is not there, CACCNT is checked directly. If neither RACCNT nor CACCNT is there, A writes L into CACCNT with a small quantum of time and tracks which he takes from RESOURCE.

Whatever is left of the resources in RESOURCE is called the pool. L may determine who can withdraw from the pool by writing a permission file, ALLOW, containing the problem number, programmer number of each individual he would like to be able to withdraw. He may also write a limit that each individual may withdraw from each pool.

The users who have permission may then withdraw from the pool and deposit in the pool by the use of privileged commands. A file is kept in L's directory which is a record of how much (net) each person has taken from the pool. This file is called the TRANSACTION file.

Changes to Time-Accounting File UACCNT:

The time accounting file UACCNT which is kept in the system file and maintained by A would now contain for each user, his problem number, programmer number, name, standby indicator, line multiplier, unit group number, party group number, restriction code, and allocation group number. LOGIN would read the allocation group number from UACCNT. If the allocation group number is zero, it is followed by the time allotments and track allotments and password. If it were a legal group number, LOGIN would attach to the appropriate U.F.D. and read the user's password. If the problem number, name and password match that of the person logging in, the time and track information would be read from CACCNT. (The password is put in CACCNT to make possible a facility for allowing an individual user to change his own password should anyone have time to implement this.) If the allocation group number of a user is zero in UACCNT, all information for that user is read from UACCNT.

Description of Proposed Commands:

The command to withdraw resources might look like this:

EXTEND GRPCC TO N T1 N T2 0 T3 0 T4 0 (INFORM PROGNO)

where GRPCC is the mnemonic for the user's group. The pairs "T2 0", T3 0, etc., indicate what and how much the user wants.

If (PROPNO PROGNO) is specified, the resources are given to that user rather than the user who issued the command. It is possible to use this option to allow the groups to control common files also. This might introduce a control problem in that the group leader can't tell who withdrew for the common file. It also would require an additional change to the UACCNT file. It is the prerogative of the implementer to add this facility.

The command will attach to the GRP06 administrative file directory and search the ALLOW file for the user. If he is not found, an error message will be written and the command will terminate. If (PROPNO PROGNO) is specified, the file will be searched for that person also.

After the person's permission has been assured, his demands are checked against the amounts still in the pools. The amount he is asking for is also checked by adding it to the amount he has already taken since the ALLOW file was made up and comparing it to his limit; it must be less than his limit for the command to proceed. If any demand cannot be met, he is given what is left. A message is printed advising him of what he really got and urging him to contact his group leader.

If the user has received more time, this time is added into his TA vector in core-A. The change will also be updated into GACCNT file. If the parameter (PROGNO PROGNO) is specified, that person's entry in GACCNT is updated.

If the user has received more tracks, this information will be sent to the disk routine by a call to ALLØT. The GACCNT and BACCNT file will also be updated with this information.

In this way time and track extensions take effect immediately. Allotments remain in effect until specifically changed by the group supervisor, or until the BACCNT file replaces the GACCNT file at the beginning of the month.

After the user has been credited with new allotments, the TRANSACTION file is updated for him. A message is printed on his console stating how much is left in the pool he has drawn from. In this way, impending disasters can be averted by his informing his supervisor when pools get very low.

The command to return resources might look like this:

DPØSIT GRP06 T0 M T1 N T2 O T3 T4 O

The command will attach to the GRP06 administrative file directory and search for the user in the GACCNT file. If he isn't found the command will print a message and

terminate.

If he wishes to deposit time, his TA vector in core-A is checked; and if his allotment minus the time he has used (if) is greater than the amount specified for deposit, the amount is subtracted from his TA and added to the pool. If the amount specified is equal to the amount of time he has left, a message is printed and an affirmative response must be received before the command will subtract from his TA.

After the time has been subtracted from his TA, it is also subtracted from his allotments in CACCNT. The TRANSACTION file is then updated for him (it is possible for his entry in this file to become negative if he gives back more than he takes).

If he wishes to deposit tracks, a call is made to the disk routine ST@RGF to get his allotment and amount used. If his allotment minus the amount he wishes to deposit is greater than the amount he has used, the amount he wishes to deposit is subtracted from his quota and added to the pool. His new quota is set by a call to ALLOT and his entries in CACCNT, RACCNT and the TRANSACTION file are updated.

The resources deposited are now available to any other user in his group.

In order to use EXTEND and DP@SIT intelligently, the user should be able to get an up-to-date snapshot of his resources and their use any time while he is logged in. This could be done with a command which might look like this:

TTPEEK

which would print the information usually printed by LOGIN:

SHIFT ALLOTTED	MINUTES USED
1 at1	ut1
2 at2	ut2
3 at3	ut3
4 at4	ut4
5 at5	ut5

ST@RAGE	DEVICE	QU@TA	USED

DRUM	DRQ	DRU
DISK	DIQ	DIU
TAPE	TPQ	TPU

Implementation:

The work involved in implementing this method of allocation consists of the following:

Writing the commands EXTEND and DEPØSIT. The commands can probably be written in MAD with small FAP functions for accessing core-A. The ALOCAT command must be modified to write RACCNT instead of the file it now produces (a slight difference in format) and to check against RESOURCE file. It must also be able to allow L to modify GACCNT subject to the restrictions of the RESOURCE file. The command TTEEEK must be written; most of the code could be lifted from LOGIN.

The RDACCT subroutine must be modified to read the GACCNT files.

A command should be written to do A's monthly resource allocation and checking.