PROGRAMMING STAFF NOTE 54

November 8, 1965

FROM: T. H. VanVleck

SUBJ: Proposal for resource allocation and accounting system to be integrated with new LØGIN, described in PSN 50.

Purposes

1. Implement dynamic allocation, described in PSN 43.
2. Provide sensible and efficient tools for administrators' management of resources.
3. Increase system security.
4. Take care of problems revised by by new LØGIN.
5. Neatness.

The Current Setup

At MAC, resources are allocated by the 18 group leaders, who create a file GRPxx GAF with the program ALOCAT. These files are picked up by Mills whenever they have changed, and run against UACCNT TIMACC to produce a new UACCNT TIMACC.

At the Center, allocation is done completely by Steinberg.

The drawbacks to the current system are:

1. The 18 group-leader passwords are valuable, since the GAF files contain passwords.
2. Mills must login to have changes in resource allocation become effective.
3. The allocation programs are difficult to use, and are not general (i.e. they are word-boundary dependent, and allow only one level of allocation).
4. Billing and summary output is difficult, or nonexistent.
5. Kludge and other privileges are not handled well.
6. It is possible to "re-use" time, by giving time to one number, having it used up, and then giving it to another.
7. There is no check on the total time or records allocated by a group.
8. Administrators cannot get a picture of the instantaneous status of their group.
9. At the beginning of the month, users logged in when TUSED TIMACC is deleted do not have their time used reset to zero.

Summary of Proposal

1. As described in the LØGIN writeup, PSN 50, each user will have a file containing his resource allocation, restrictions, time-used, etc.

2. Administrative divisions will have a "pool file" associated with them, giving the current resource allocation among the members of the group, who may be other groups or individual users. A "transaction file" is also kept.

3. Four new commands are to be added.

   a) EXTEND what howmuch -whofor-
   b) DPOSIT what howmuch -whofor-
   c) ALØCAT who -group-
   d) CHPASS

4. Three commands must be modified:

   a) LØGIN
   b) LISTF
   c) TTPEEK

5. All files are private, protected, read- and write-only, under a special author number common to both installations and belonging to the system administrator.

The group pool file will contain

1. group name, group leader name
2. pointer to superior pool file, if any exists
3. Time and record quota allotments
4. Time and record quota extension limits
5. Time and record quota "slack" - allotable
6. Entry for each group member
   a) pointer to account file: prob,prog,name1,name2
   b) time and record quota currently alloted
   c) name, if a user
   d) flags, or whatever else proves useful; we will have to see as we write the commands.

There will also be a transaction file, containing a record of all withdrawals and deposits.

Format has not been fixed for any files; there seems to be no reason to have them free-field if a system of editing can be worked out.

COMMANDS

1. EXTEND what howmuch -whofor- ....

The EXTEND command operates on a file in the current directory, and attempts to withdraw resources for it from a superior file.

what	is a code for which resource is to be increased. It may be T1....T5 or RQ1...RQ3
howmuch	represents the desired increment in minutes or records.
whofor	if present, represents the goup name: thus a group allocation supervisor may increase his total allocatable resources.

The following steps are taken:

1.) examine the extension limits for this resource versus the current allocation, to see if withdrawal is possible; if no, stop.

2.) Force a link to the superior file.

3.) Read the file to see if slack exists. Give up if non available.

4.) Rewrite the superior file with increased allocation, decreased slack.

5.) Force a link to and append to the transaction file.

6.) Update the user account file (or group file).

7.) If this withdrawal is for a user, patch core A for time allotments or call ALLOT for record quotas.

2. DPØSIT what howmuch -whofor- ....

DPØSIT will return resources by a procedure similar to that of EXTEND. However, step 1 checks the amount used instead of the limit, to prevent "re-use" of time, or cutting record quota below current use.

3. ALØCAT who group
ALØCAT (file) n1 n2

ALØCAT is the command used by the group allocation leader to modify current allocations.

group	represents the identification of the allocation group and group pool file to be used.
who	represents an entry in this file.

The command does the following:

1.) Open the group file, check that user is indeed the group leader.

2.) Locate the entry in the file; if it is not there, say so. If found, summarize current allocations and use.

3.) Accept pairs of the form code amount
where code is a code for the resource, being T1...T5, RQ1..RQ3, TL1..TL5 (extension limits), RQL1...RQL3 (extension limits).

4.) The command must check the limits in the pool file in order not to over-allocate. It must also check the current usage of any user being modified, to prevent "re-use", or cutting record quota below current use.

5.) The pool file and the user's account file are both modified: if the user is logged in, this may patch, too.

6.) If the entry is not found, and no such user is in the MFD, the user may be created. Perhaps a flag can be set in the group leader's account or in the pool file for permission to create new users.

7.) This command may also create a group file entry (perhaps again only if permission is granted). Such a subgroup may then manage its own users.

8.) The meta-argument '(FILE)' may be used to set up a large number of allocations by reading input from disk.

9.) There should be an optional "brief" mode.

4. TTPEEK

Should be modified to allow a group leader to get allocation and usage summaries for his group. Off line output should be available, and the time and date should show. (Alternatively, TTPEEK can be left as it is and a new command written (called STATUS?) to do this.) Eventually we may wish to make some sort of linear-programming scheme available for group leaders, and to formalize a resource-request scheme for things like privilege bits, as well as time and record allotments.

The user account file must contain PRØBNØ NAME of the group leader's file directory, to simplify the work of EXTEND and DPØSIT and of setting up new users. The group identification is also necessary, for the cases in which one administrator maintains several groups.

A flag meaning "password may not be changed" is necessary for student numbers and other cases in which a number may be used by many people.

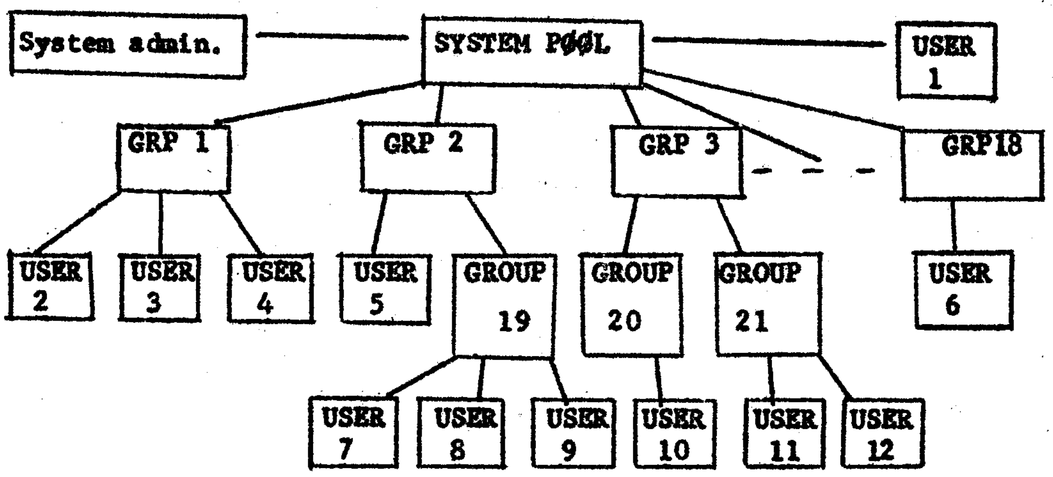
A special list in the user account file of an administrator, specifying the groups of which he is leader, might take care of situations in which a group account file is destroyed; otherwise all commands must depend on existence of the account file, or an external list, for validity checking. This will lead to startup problems or extra file-system work--and we have 432 words in the user account file.

A checksum on columns 1-72 of the user account file would be nice, but requires special editors. All commands must check for the system-administrator author number on the files they use. Any other possible validity checks should be done.

The problem of what to do if an anomaly is discovered is thorny, and probably depends on specific programming situations. Serious problems should perhaps be noted on line, less serious ones by some sort of MAIL-like approach. A user must always be able to log out, even if his account file is missing: if we tell him he can't, he will hang up.

Several programs are needed for use by the administrators, for end-of-period updating, statistics, and so forth. It is hoped that common subroutines can be used for all accounting programs.

Diagram



Points

1. A user belongs to only one group.

2. Any arbitrary depth of nesting.

3. EXTEND, etc. work for groups, too.

4. EXTEND, etc. never go more than one level.