MULTICS STAFF BULLETIN

MSB -30

To: Distribution

From: T. H. Van Vleck

Date: 3/1/72

Subject: Improved Multics Resource control

Some fairly easy changes to the resource control modules are now under development and will provide several desirable features, including

1. Limit stops on individual users' resource consumption, either by shift or by total, settable by the project administrator.

2. Elimination of the current anomaly which causes all CPU usage to be charged at the rate for the shift on which the user logged in.

3. A user command which shows month-to-date resource and funds usage, as of process creation time.

4. Simplification of the daily accounting run.

5. Space for additional items, such as resource meters and control items, which cannot be stored in existing data bases.

To implement these features requires the replacement of one answering service module, minor modifications to some existing modules, the creation of a new data base, and several simple support programs.

Current Situation

Currently, user control creates a "session record" entry in a file named "accounting" when a user logs in. This record is 20 words, or 80 characters, long and is often called a "session record card" although the information contained in it is no longer completely character. There is not enough space in this session record to store usage for more than one shift, leading to the "shift anomaly", and the session records for each day must be processed each night to attribute the sessions to the various users of the system.

When a session record file fills up, it is renamed and another file started. Manual procedures are required to complete the processing of the previous session record file, and if the system stays up too long after the session record file has filled up, the answering service will blow up due to out-of-bounds errors on the session record file.

Cutoff for overexpenditure is performed once a day only. This cutoff is only on a whole-project basis. Users are unable to obtain any information about their resource consumption on-line, and must call User Accounts during working hours to get figures that are a day old (or more, if the user was logged in while the session record file was being processed).

We have come about as far as we can with the present resource control method, which was written in 1969 and has been long overdue for a rewrite.

Overview and Definitions

A Project is a grouping of users for resource control purposes. Almost all resource control passes through the project level. Funding is not the same as resource control; "accounting" is the process of attributing resource consumption to an externally-specified funding entity called an account, or requisition, or purchase order, and possibly determining that this account is exhausted and that charges must be made to some other account (or that some other action should be taken, such as cutting off the source of charges).

Limit stops are the property of projects, not accounts. The question of funding is not material to resource control, in much the same way that the question of physical space available has no direct relationship to the total record quota allocated by the system. A project may choose to set no limit stops on its users. The resource control system will then never cut off a user on the project. The accounting system may decide to cut the whole project off, but that is its business, and has to do with a lot of parameters which are mixed up with installation policy, such as "credit ratings".

The accounting subsystem operates completely in the system administrators' processes. It couples to the resource-control system by generating modifications to the tables which the system administrators install, but it is not directly involved in the control of resource consumption. Many installations may wish to modify the operation of the accounting subsystem to meet their own local requirements; very few should need to modify the resource-control facilities, which are an integral part of the Multics system.

Organizing resource control in this manner does not preclude the provision of multiple accounts per user, or multiple projects per account, or other, more elaborate, structures. These facilities will be provided at the accounting subsystem level as necessary, and if "hooks" in the resource-control package are requred to allow these structures to operate smoothly, additional control parameters can be added incrementally to resource control. Similarly, this proposal does not preclude the eventual "distribution" of the resource-control functions and the eventual moving of the data base to ring 1.

New Method of Resource Control

The single answering service module which performs resource control is called "act_ctl_". By replacing this module with one which maintains an up-to-date set of usage figures for each user of the system, we will significantly improve the performance of the accounting and resource control systems and the appearance of Multics to the user.

The new data base will contain month-to-date usage figures for each resource, in resource units (e.g. microseconds), for each user. The module act_ctl_ will update its usage figures in place in the data base entry for the user as resources are consumed. A simple program will be made to copy the data base files daily from the system directory into the administrative directory, producing a file in the format of the "hist" file, which is the end result of the current daily processing and the driving file for the monthly billing. By doing this, we avoid requiring the simultaneous reprogramming of all programs which operate on the hist file.

User limit stops, determined by the project administrator, will be inserted into the project's PDT. These limits will be available for checking user overrun at login and every time an updating cycle occurs. The user limits would of course also be copied into the PIT whenever a process is created, so that the "resource_usage" command can access them.

Since the resource-usage entry for the user will be located before his process is created, it will be easy to pass usage figures to the newly created user process in his PIT. We will provide a user-callable subroutine to extract these figures, and to produce a month-to-date usage report which is up-to-date as of process creation. (Since the hardcore is not conscious of shift changes, we cannot determine how much CPU usage was one shift and how much another, if the user's process was created before the most recent shift change.)

These modifications will proceed in at least three stages. The first stage will install enough changes to allow us to run the two accounting methods in parallel. This will enable us to compare the new system with the old, and provide backup in case of bugs in the new method. Once the new facility has been checked out, those features of the package which are visible to users will be added and announced, and the accounting subsystem will be modified to use the new method. Desirable enhancements will be done as a third stage.

Design of data base

It seems best to create a separate usage segment for each project. There is already a per-project segment with an entry for each user, namely the PDT. The new data organization is to re-format the PDT entries to carry usage and limit stops as well as privilege attributes, initial procedure, etc. This method avoids the problems which may arise if so many segments become active in the initializer that the KST becomes full, and eliminates the possibility that the usage and limit data bases may get out of step due to a partial reload or other system failure. This organization will also enable us to allow the project administrator direct read access to the usage data for his project, and enable him to generate any usage reports he desires. (A subroutine will be provided for the administrators to extract the data from the PDT.) The price we pay for these advantages is a lower limit on the number of users allowed in a project (255 instead of about 800, until segment size is increased, then 1023), some complications during the installation of PDT's to inusre that deleted users are still charged for their resource usage (we do something like this for the PNT already), and the necessity of reformatting all PDT's mechanically when the new method is installed (easy).

An interlock will be required in each PDT entry, because more than one process will sometimes modify the entry as a result of daemon charging and monthly reset operations performed by the system administrator.

An SPS section for the new PDT format is attached as an appendix.

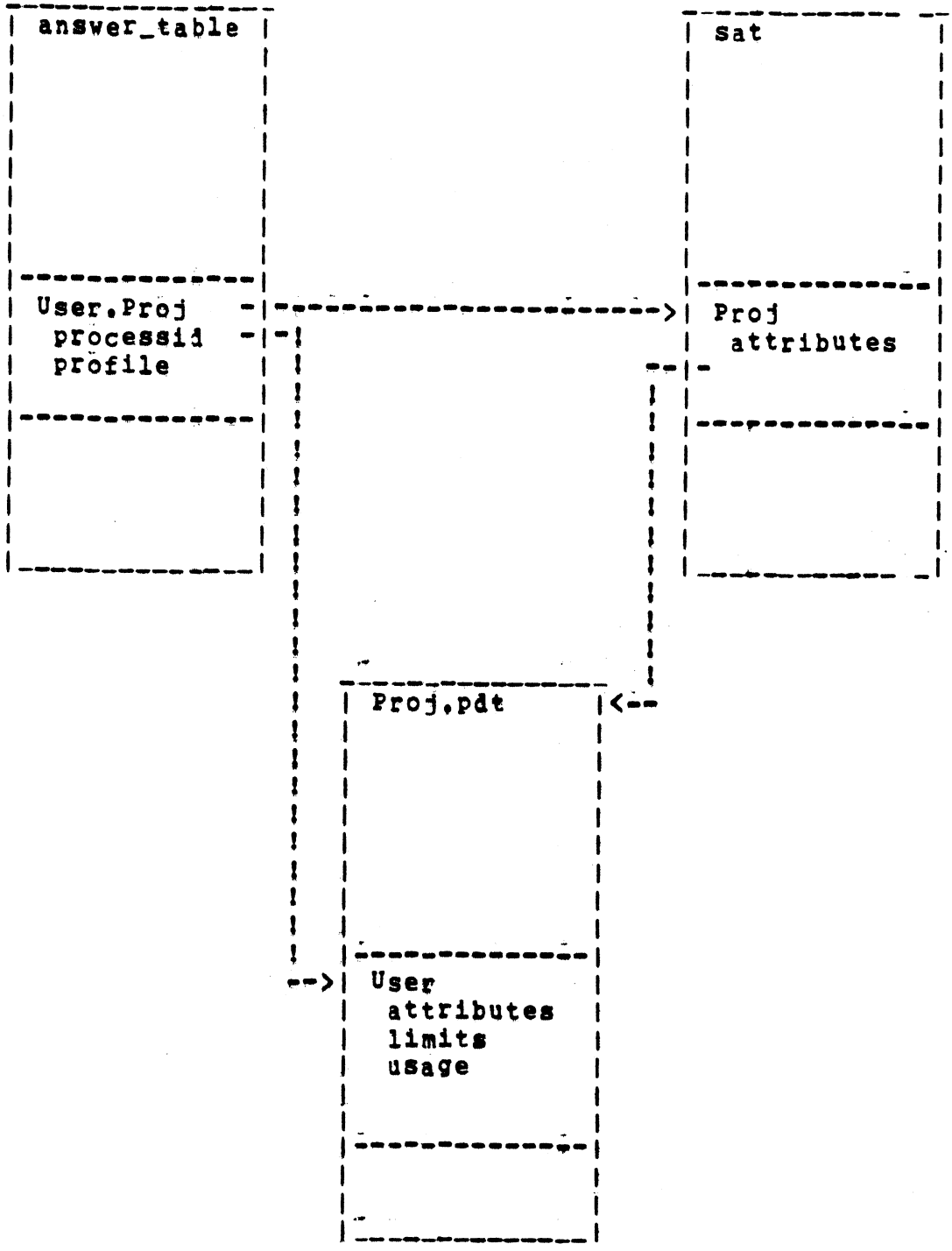

Figure 1 - Data Base Organization

User Interface

Since dollars are the most convenient unit for the project supervisor, resource limits will be expressed in dollar amounts. (The "dollar sign" should be installation-replaceable, so that installations which wish to call them "cost units" instead can do so.) The system prices are available in a per-system data base, and can be used to convert from resource measurement units into "dollars" wherever the programming is most convenient.

The problem, when designing resource-limiting schemes, is deciding when to stop. (This is meant in several senses.) In particular, although it is easy to suggest having a limit for each measured quantity, it is difficult to find a sensible interpretation for how the mechanism should behave when a limit is exceeded. (Suppose your core-usage quota runs out, but your CPU limit doesn't?) In addition, providing too many detailed limits will undoubtedly prove vexing to the project administrators. A single dollar limit per user, on the other hand, might not provide quite enough control for those projects which wish to compel their users to avoid relatively expensive ways of using the system. Therefore, dollar limits per shift will also be implemented.

The limits will be set by the project administrator's adding the line

```
limit:          500;
```

for example, to a user's PMF entry. This statement limits the user to 500 dollars' worth of resources, however consumed. To set shift limits, the project administrator might say

```
shift_limit:    99.95, 300, open, 300;
```

setting a limit of $99.95 on shift one, $300 on shifts two and four, and leaving shift three unlimited.

Enforcement of these limits will be done by refusing login. A message of the form

```
User resource limit of $99.95 on shift 1 exceeded.
No login.
```

will be printed to explain to the user why he may not use the system.

We will type a similar message and give the user an automatic logout if the user exceeds his limit in the course of a session.

Absentee and IO daemon usage will be controlled by the total dollar limit only. If project administrators request it, we may

later implement a permission bit which allows the user to use the high-priority queues only if the project administrator permits.

The initial implementation of the resource-control package will not check IO requests for over-expenditure, since the dollar amount involved is small and the change is rather complicated.

A simple modification will be made to absentee_overseer_ or init_admin_util_ to check the funds limit for an absentee user by looking in the PIT, and to write a nice message in the absentee output segment and log out if the limit is exceeded during the job. Once an absentee job is started, it will be allowed to finish even if the user exceeds his limit.

We will have to make it clear to the project administrators that there is some inaccuracy in the limits, since a user can consume a fair amount of resources between updates. User subsystems which implement tighter control can, of course, continue to be used.

Code to be written

The following programs will be installed first:

1. Rewrite of act_ctl_ to add code for in-place updating. This will initially be done only for interactive and absentee sessions.

2. Program to copy usage data from PDT and produce old-format "hist" file.

3. Modify up_pdt_ and up_sat_ to "carry" deleted users until reset.

4. Program to re-format PDT's.

5. Program to update daemon usage into PDT entries and increase total charge figure.

The next steps are the following:

6. Modify master.ec to eliminate processing of session record deck.

7. Modify cpg_ to put limits and resource usage into PIT.

8. User "resource_usage" command, to type out information passed in PIT. Also modifications to user_info_ to return usage figures.

9. Modify cv_pmf to place limit stops in user pdt entry. Sensible default statement and default default values required.

10. New program to reset usage at end of month.

Additional changes which may be made later:

11. Modify ed_installation_parms to put absentee and daemon prices into system parameter segment.

12. Modify absentee_overseer_ or init_admin_util_ to enforce funds limit.

13. Modify billing programs to use new data base format. Then copy program of (2) above can be deleted.

14. Implement subroutines to allow project administrators to construct subsystems for automatic management.

15. Project administrator's usage report command.

15. Wait-after-preempt timer to prevent misuse of bumping privilege.

17. Time of last logout and console for last session in PDT.

18. Add tape charges, multiple-console charges, etc.

19. Move usage data base to ring 1, as described in Voydock/Feiertag memo. We can then modify daemon charging to use ring 1 primitives and update in place, and delete the program in (5) above.

Name: pdt

The Project Definition Table (PDT) defines the legal users on a project and their attributes. It is also used to store resource usage data for each user on the project. Each Project Definition Table is kept in a separate segment. There is one for each legal project on the system. These segments are accessed by the user control programs during login and logout, and by the resource-accounting programs as a user consumes resources.

Project administrators create a PDT for their projects by using the cv_pmf command to compile a Project Master File into a binary file. They then use the install command to request that the system install the new copy.

PL1 DECLARATION

```
dcl 1 pdt based (pdtp) aligned,
    2 author aligned,
      3 proc_group_id char (32),
      3 process_id bit (36),
      3 ev_channel fixed bin (71),
      3 table char (4),
      3 w_dir char (64),
    2 max_size fixed bin,
    2 current_size fixed bin,
    2 version fixed bin,
    2 freep fixed bin,
    2 n_users fixed bin,
    2 project_name char (28) aligned,
    2 project_dir char (64) aligned,
    2 pad1 (199) fixed bin,
    2 user (255) aligned,
      3 pad (256) fixed bin;

dcl 1 user based (pdtep) aligned,
    2 state fixed bin,
    2 lock fixed bin,
    2 person_id char (24) aligned,
    2 password char (8) aligned,
    2 at aligned,
     (3 administrator bit(1),
      3 primary_line bit(1),
      3 nobump bit(1),
      3 guaranteed_login bit(1),
      3 anonymous bit(1),
      3 nopreempt bit(1),
      3 nolist bit (1),
      3 dialok bit (1),
      3 multip bit (1),
      3 bumping bit (1),
      3 brief bit (1),
```

```
      3 vinitproc bit (1),
      3 vhomedir bit (1),
      3 nostartup bit (1),
      3 sb_ok bit (1),
      3 pm_ok bit (1),
      3 eo_ok bit (1),
      3 pad bit(19)) unaligned,
    2 initial_procedure char (64) aligned,
    2 home_dir char (64) aligned,
    2 bump_grace fixed bin,
    2 high_ring fixed bin,
    2 init_ring fixed bin,
    2 pdtpad (14) fixed bin,
    2 dollar_limit float bin,
    2 dollar_charge float bin,
    2 shift_limit (0: 7) float bin,
    2 daton fixed bin (71),
    2 datof fixed bin (71),
    2 last_login_time fixed bin (71),
    2 last_login_unit char (4),
    2 last_login_type fixed bin,
    2 time_last_bump fixed bin (71),
    2 logins fixed bin,
    2 crashes fixed bin,
    2 interactive (0: 7),
      3 charge float bin,
      3 xxx fixed bin,
      3 cpu fixed bin (71),
      3 core fixed bin (71),
      3 connect fixed bin (71),
      3 process fixed bin (71),
    2 absentee (4),
      3 charge float bin,
      3 jobs fixed bin,
      3 cpu fixed bin (71),
      3 real fixed bin (71),
    2 iod (4),
      3 charge float bin,
      3 pieces fixed bin,
      3 cpu fixed bin (71),
      3 lines fixed bin (71),
    2 devices (16) float bin,
    2 pdtupad (29) fixed bin,
    2 chain fixed bin;
```

EXPLANATION OF VARIABLES

1.	author	validation data about table's author
2.	proc_group_id	process-group-id (personid.projectid.tag)
3.	process_id	author's process ID
4.	ev_channel	response event channel
5.	table	"PDT"

```
6.  w_dir              author's working directory
7.  max_size           max number of entries table can grow
8.  current_size       current size of table (in entries)
9.  version            table version
10. freep              index of first entry on free chain
11. n_users            number of entries actually used
12. project_name       name of project
13. project_dir        treename of project's directory
14. pad1               make header 256 words long
15. user               the project definition table entries
16. pad                each entry is 256 words long
17. user               declaration of a single PDT entry
18. state              1 = normal, 2 = deleted 0 = free
19. lock              entry lock
20. person_id          login name of user
21. password           password for anonymous user
22. at                 the user attributes
23. administrator      1 = system administrator privileges
24. primary_line       1 = user has primary-line privileges
25. nobump             1 = user cannot be bumped
26. guaranteed_login 1 = user has guaranteed login privileges
27. anonymous          not used
28. nopreempt          not used
29. nolist             1 = don't list user on "who"
30. dialok             1 = user may have multiple consoles
31. multip             1 = user may have several processes
32. bumping            1 = user may bump others in group
33. brief              1 = no login or logout message
34. vinitproc          1 = user may change initial procedure
35. vhomedir           1 = user may change homedir
36. nostartup          1 = user does not want start_up.ec
37. sb_ok              1 = user may be standby
38. pm_ok              1 = user may be primary
39. eo_ok              1 = user may be edit_only
40. pad                padding
41. initial_procedure initial procedure in user's process
42. home_dir           user's default working directory
43. bump_grace         number of minutes he is protected
44. high_ring          highest ring user may use
45. init_ring          ring user will start in
46. pdtpad             padding
47. dollar_limit       user dollar limit
48. dollar_charge      total dollars spent this month
49. shift_limit        user interactive dollar limit by shift
50. daton              date user added to system
51. datof              date user deleted
52. last_login_time    time of last login
53. last_login_unit    terminal id last used
54. last_login_type    terminal type
55. time_last_bump     time last bumped
56. logins             number of logins
57. crashes            sessions abnormally terminated
```

58.	interactive	interactive use, shifts 0-7
59.	charge	total dollar charge this shift
60.	xxx	padding
61.	cpu	cpu usage in microseconds
62.	core	core demand in page-microseconds
63.	connect	total console time in microseconds
64.	process	total process time in microseconds
65.	absentee	absentee use, queues 1-4
66.	charge	dollar charge this queue
67.	jobs	number of jobs submitted
68.	cpu	total cpu time in microseconds
69.	real	total real time in microseconds
70.	iod	io daemon use, queues 1-4
71.	charge	dollar charge this queue
72.	pieces	pieces of output requested
73.	cpu	amount of cpu time in microseconds
74.	lines	total record count of output
75.	devices	device charges
76.	pdtupad	padding
77.	chain	index of next entry on free list

The bit-flags in the substructure "at" are ANDed with the corresponding flags in the project's entry in the SAT, and any flags specified by the user in his login line, to produce the user's attributes for his session.

The items in the "author" substructure are used by up_sysctl_ in the process of installing a new copy of the segment into the system control directory where the PDT files are kept.

(END)