J. H. Saltzer

516

DRAFT
date 1/24/67

Identification

Major Entries to the Process-Group Ranker

Robert R. Fenichel

Purpose

For various reasons, it is useful for the system to maintain a single linear ordering of all logged-in process-groups. This ranking is maintained by the Process-Group Ranker (PGR), whose major entries and functions are described here.

Discussion of the substructure which is actually concerned with ranking policy has been segregated in Section BQ.5.02. A diagram covering all of BQ.5 appears as the last page of this section, BQ.5.01.

Definitions

Using mechanisms discussed elsewhere, a process-group (P-G) may be quit and saved, and its processes may then be destroyed. For purposes of the Central Supervisor, this P-G no longer exists.

Within section ~~ET~~ BQ, the saved segments will be referred to as an inanimate process group, in the naming tradition which brought us mock turtle and alleged criminal. A P-G properly so called will be here known as an animate process group.

For the interactive process group, being animate is the same as being logged in. An absentee P-G may be logged in without being animate; more on this suspended state appears below.

The PGR maintains a list of all logged-in process groups. This list, simply referred to as the Ranking all through ~~ET.1~~ BQ,5, is ordered by juniority. This latter word is meant to be an antonym to "seniority;" having low juniority is better than having high juniority.

In particular, two threshholds are defined. At any given time, only those users with juniority ≤ <u>userceiling</u> are allowed to remain animate, and only those users with juniority ≤ <u>usermargin</u> may become animate. Userceiling is never less than usermargin.

The process of forcibly removing a user from animate status is known as <u>bumping</u>. Bumping is automatic logout in the case of interactive users.

In the case of absentee users, bumping is <u>suspension</u>. Suspension is very similar to automatic logout--all processes are quit and saved. But the difference is that a suspended process group is subject to <u>rekindling</u> by the PGR with no human intervention.

Let us return to the ranking of process-groups. This ranking is composed of a (very large) fixed number of <u>slots</u>. Each slot is permanently identified with a unique juniority. At any given time, a given slot either

- (a) is empty,
- (b) is <u>tenanted</u> by an animate process-group,
- (c) is tenanted by a suspended process-group,
- (d) holds the label of an active reservation group (see BT.3), or
- (e) holds the label of an active reservation group <u>and</u> is tenanted by an animate process-group.

There are no daemons to cause a process group's juniority to change gradually through time. However, there are a number of reasons for <u>sudden</u> change in a process-group's juniority; many of these are discussed in BT.1.2. NO

For example, consider the case of the process group whose reservation group reaches its normal termination. This process-group will immediately suffer a great increase in juniority.

Entries Used Following User Action

The login_absentee command results in the call

```
try_to_insert(username, projid, 0, response, juniority, loginid)
```

where username and projid are the first two elements of a P-G identification (BO.1.05). ~~These two~~ The first three arguments are the only ones supplied by the caller. If the PGR decides that it cannot accept an absentee P-G from this user, response is 0; for example, the PGR might believe that this user should not have more simultaneous logged-in absentee jobs than he already has. Otherwise, response is 1 and the assigned juniority and loginid (BO.1.05) are returned.

The mechanism determining the response and juniority of the new man* is described in BT.1.2. If the response is to be 1, the loginid is computer by utilizing the login_id procedure of BQ.5.04.

Similarly, the login command results in the call

```
try_to_insert(username, projid, 1, response, juniority, loginid)
```

The third argument~~s~~ here makes ~~are the same as with insert, but~~ the response 0 is distinctly ~~ctly~~ more likely. For as in the case ~~of insert,~~ when this argument is 1, the juniority-computing procedure may refuse to allow the login. But even after the juniority has been

*Some though was given to the possibility of calling the ~~insert~~ try_to_insert entry "P_G_Newman." Puns, however, have no place in this Manual. Et tu, alligator.

computed, the login will still be unsucessful (response 0) if this juniority is greater than usermargin.

When the first reservation of a reservation group (BT.3) is started, the reserver makes the call

```
          reservelot(username, projid, loginid, reserverjuniority,
                              reservation-group-name, response)
```

Here, all but the last argument is supplied by the reserver. The *reserverjuniority* parameter is a number computed by the reserver to serve the function of juniority. It may be used by the PGR in computing juniority for purposes of the ranking.

If this latter juniority is beyond the usermargin, the response 0 tells the reserver that this reservation-group Cannot be honored.

Whenever logout is terminating a process-group, the

```
          outsert(username, projid, loginid)
```

call causes the slot to be cleared. The juniorities of some other process-groups may be changed at this time. For example, this user may have other logged-in process groups of higher juniority. Quite possibly, one of these will be moved up into the just-vacated slot. For more of these matters, see BQ.5.02.

Entries Used Following System Action

It may be desirable to reduce the system's load. The

```
               easeout (n)
```

call will reduce usermargin until *n* nonempty slots are newly beyond usermargin. For the effect of this call, see try_to_insert and *reserveslot* above.

The need for load reduction may be more urgent. The

squeezeout (n)

call may be used. This causes userceiling to be reduced until n non-empty slots are newly beyond userceiling. Usermargin is set to coincide with userceiling.

Tenants of the slots passed over are bumped; reservations associated with these slots are ~~dishonored~~ listed among those not honored by the system,

In happier times, the Thermostat (BQ.5.03) may react to system underloading by calling

letin (n)

This call lifts userceiling past n suspended or unstarted process groups. Each of these p.g.'s is rekindled, ~~or started.~~ Usermargin is set to coincide with userceiling.

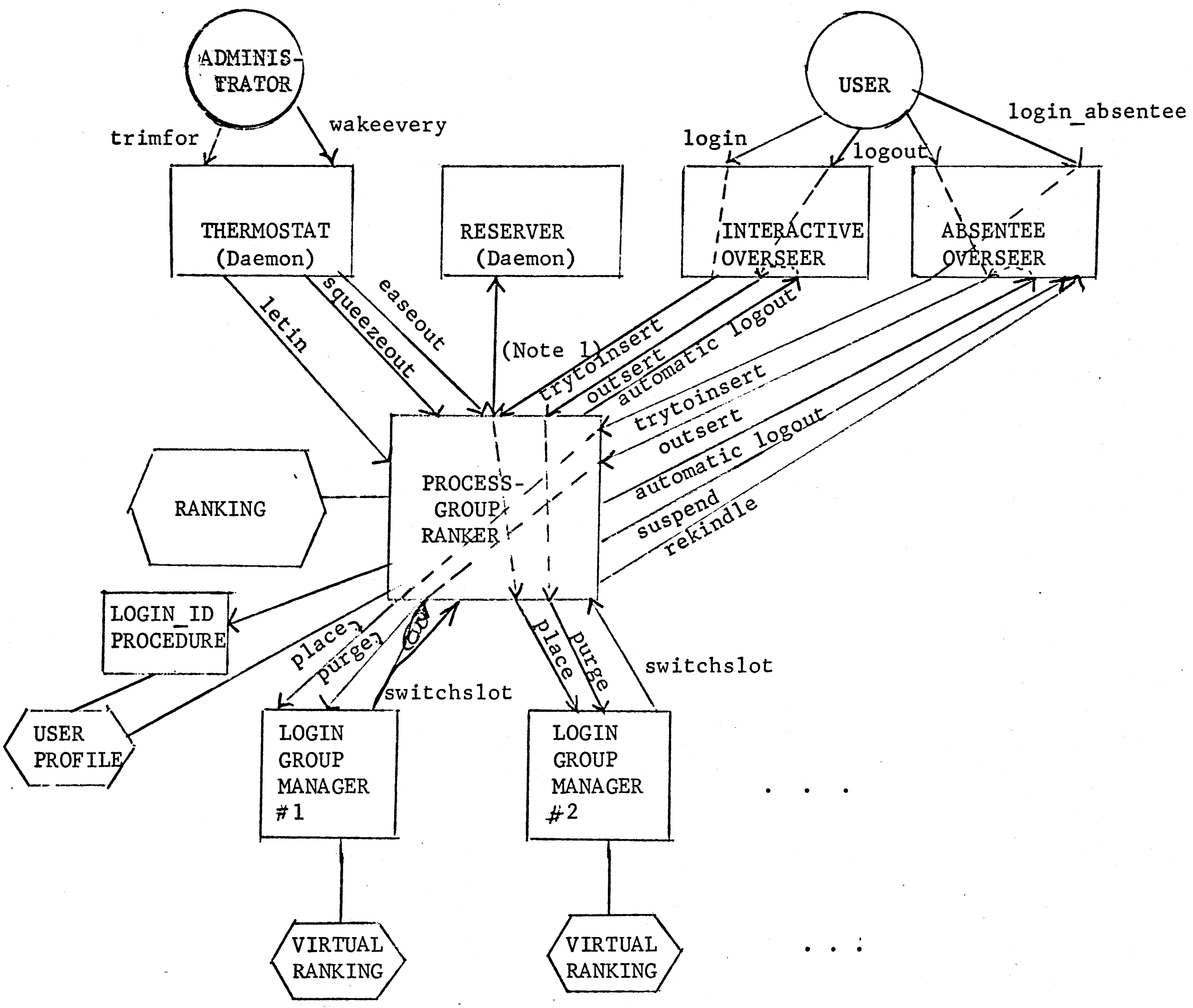
In order to allow certain users of the letin, easeout, and squeezeout entries, the integer-valued, no-argument, self-explanatory entries

getuserceiling

getusermargin

get_number_of_animate_users

are available.



Note 1: This arrow covers the reserveslot call, the PGR-to-Reserver "can't honor this reservation" call, and possibly some undesigned calls in this area.

Identification

Process-Group Ranker Policy

Robert R. Fenichel

DRAFT
date 1/24/67

Purpose

As described in Section BQ.5.01, the Process-Group Ranker is concerned with the care and feeding of a single great ranking of all logged-in process groups. The procedures determining slot* assignments are described in this section.

Much of this section is dependent upon the concepts and jargon of the Reserver, BT.3.

Finding Slots for Reservation Groups

The best (lowest-juniority) slots are used for reservation groups. The reserver's opinion of the relative merits of various reservation groups (that is, the reserverjuniority parameter of the reserveslot call) is taken as gospel, and the PGR will occasionally lock the Ranking until the reservation-group slots have been reshuffled.

The reserverjuniority parameter will probably be the calendar time at which the reservation group was inserted into the Timetable by the Reserver.

Finding Slots for Process Groups

When the PGR receives a~~n~~ ~~insertion~~ trytoinsert call, it first checks the Ranking for untenanted reservation-group slots belonging to th~~e~~is user. If any such slots exist, this process-groups is given the least junior of these slots.

NO UNDERLINE

A user whose reservation-group is waiting is consequently protected from being unable to log in. Of course, it is possible for a user's process-group A to land in a slot which he has intended to use with process-group B.

*For vocabulary, see BT.1.1.

In order to allow the user to utilize his reservation-groups as he sees fit, a special call is provided:

movetoreservationslot(nameofreservationgroup)

will move the calling process-group ;to the slot held by the named reservation-group. If this name is 0, the calling p-g is treated as if it were just logging in, with no reservation slots waiting.

Similarly, the termination of a reservation group will cause the group's slot's tenant, if any, to move to the position of a new logger-in with no reservations.

In the normal case, a user loggin in will have no reservations unclaimed. In this case, the user's juniority is computed via the important mechanism of login groups.

A login group is uniquely determined by a user's name, project id, and mode of use (interactive or absentee). A given slot in the Ranking is associated with exactly one login group. Thus, a would-be logged-in process-group is competing for one of the slots in the login group of this user and mode of use. This competition is independent of similar competitions which may be in progress for slots belonging to other login groups.

In brief, the scheme is straightforward. The login-group of the user is determined, and control is transferred to a login-group-manager peculiar to that group. The login-group-manager may choose to reject the login, but more likely it will compute a "virtual juniority" for the user. That is, it will compute the juniority of the user relative to other members of the login group. Finally, this "virtual juniority" is decoded into true juniority by the PGR.

For example, the absentee mode of use might be necessary and sufficient for membership in a login group owning one excellent slot and many mediocre ones. The login-group-manager of this login groups could use FIFO algorithms to perform most of the functions of the CTSS FIB monitor.

In detail, the procedure is as follows:

1. The user's login group number (say, k) is found in his user profile.

2. The PGR calls manage_login_group_k with the call

```
                  STET
place(username,projid,absentee/interactive)
```

   a. The manage_login_group_k routine thinks it is allotting slots numbers 1, 2, ..., n. Actually, of course, it is responsible for n slots whose true numbers are known only to the PGR.

   b. Place many not wish to admit this process group at all. In this case, place returns ~~to~~ the value 0.

   c. In any other case, place returns an integer between 1 and n. This is the virtual slot to which the new process group is assigned.

   d. As it tries to find a slot for the new process group, place may wish to shuffle another p-g from virtual slot i to virtual slot j. To do this, it gives the

   ```
   switchslot(i,j)
   ```

   call back to the PGR. The value of switchslot is

   0 if there is some error (virtual slot i empty, virtual slot j not empty)

   1 if the result of this call has been the ~~logout~~ bumping of the moved p-g (slot designated by j above userceiling)

2 otherwise

If j is 0, the p-g designated by i is always logged out. If the p-g designated by i was suspended ~~or unstarted,~~ and it is moved to a slot below usermargin, it is rekindled, ~~or started~~.

3. The PGR uses a function

realslot(i,K)

to determine what manage_login_group_k is talking about when it says "i". The value of place is translated via realslot, as are the arguments of switchslot.

4. If place wanted to put an interactive user into a slot beyond usermargin, or if a user belonging to login group k logs out, the PGR calls manage_login_group_k with

purge (i)

where i is the virtual slot number. In reaction to purge, manage_login_group_k may choose to shuffle its remaining family with switchslot.

e. To guide place, the PGR has the integer-valued, no-argument entries

virtual_user_margin
virtual_user_ceiling

Identification

The Thermostat

Robert R. Fenichel

DRAFT
date 1/24/67

Purpose

The Thermostat is a daemon module which tries to control the ratio of capacity to offered load.

Discussion

One can imagine the use of very sophisticated techniques to detect overloading of a large computer system. Until further notice, the Thermostat won't be using any such techniques. Indeed, the initial implementation could be done non-daemonically. However, the daemon mechanism seems like a good precedent.

The Thermostat will accept two calls:

```
wake_every (n)
trim_for (p)
```

Subsequently, it will awaken every *n* seconds to see if the number of animate process-groups is different from *p*. If it is, *letin* or *eas~~squeez~~eout* (BQ.5.01) will be used.

An important thing to notice is that there is no essential relationship between *p* and usermargin or userceiling (BQ.5.01). That is, a given value of *p* may be associated with various values of these other parameters. A system trimmed for 100 users might have very low usermargin during the day, when members of preferred login groups might all be active. At night, usermargin might increase to fill the machine with the rabble.