TO: MSPM Distribution
FROM: C. Marceau
SUBJ: Logging in and logging out
DATE: 07/07/67

Sections BQ.2.03 (Logging in) and BQ.2.03A (Logging out) should be discarded. This section (BQ.2.03) replaces both. The differences are numerous. Major changes are:

1) Logging in and logging out now occur in a separate process, the User Control Process.

2) They do not use the standard error handling technique.

3) The listener (BX.2.02) is no longer involved in a user-initiated logout.

4) BQ.2.03 does not describe logging in and logging out from the user's point of view. The user's view of login and logout can be found in BX.3.01 and BX.3.04 respectively.

Published: 07/07/67
(Supersedes: BQ.2.03, 01/16/67;
BQ.2.03A, 01/16/67)

Identification

User Control Processes
C. Marceau

Purpose

When a user dials up from a remote console the system must take action to enable him to use the computer, and at the same time must verify his right to use the computer (is he a bonafide user, is he solvent?). Similarly, when a user leaves the system, some module must inform other system modules that the user's process-group is gone, and in general, clean up after the user. The first action is called logging in the user's process-group, and the second is called logging out the user's process-group. Both actions are handled by what is called a User Control Process (which controls not the user himself, but logging in and logging out the user's process-group). Note: This section discusses logging in from the point of view of the system rather than of the user. If you want to log in, see section BX.3.01.

The Login Command

Readers of this section should know the format of a login command line:

```
     login name -project_id- -account_id- -wdir-
```

where name is the user's name and project-id his project id. Account-id specifies the account to which he wants to charge this console session. Wdir is the path name of the directory he wishes to use as working directory. If any of the arguments which are surrounded by dashes are not given, the user's standard default is found.

An alternative login input line allows user A to log in for user B as a proxy, i.e., after login he seems to be user B. At login, however, he must give his own password, and the User Log shows that A logged in for B rather than B for himself. To log in as a proxy for B, A types:

```
     proxy A for B -project_id- -account_id- -wdir-
```

The last 3 arguments are the same as for the login command. The arguments A and B and the word "for" may not be omitted.

<u>Discussion</u>

For the purposes of this section, we will speak of "logging in the user" as synonymous with "logging in the user's process-group". Both phrases apply, since at login it is essential to find out who the user is and to add a new process-group to the system to work for the user. For symmetry, we speak of "logging out the user" as well as of "logging out the user's process-group", even though at logout the system is concerned less with the identity of the user than with deleting his process-group from the system. We also say that "the user logs in" to mean that the user issues a login command or in some other way causes the system to log him in.

To log a user in it is necessary to:

1) identify the user - the dialup might be caused by a chimpanzee or an imposter;

2) find out whether the user may use the system - he might be broke, the system might be operating at capacity;

3) set up a process-group for the user so that he can do work.

The user, after logging in, works in the process-group which was created for him. (User-process-groups are discussed in BQ.3.00.) After a time he finishes his work. Alternatively, the system may decide that it can accommodate him no longer, for example, because the entire system is being shut down for a few minutes for replacement of a hard-core ring module. It may even happen that some blackguard cuts the wire leading from the user's console to the computer. In any of these cases, the User Control Process logs the user out:

1) destroys the user's process-group;

2) informs interested system modules that the process-group is gone.

Logging in and logging out are discussed in greater detail below.

The User Control Process

As stated in the purpose of this section, the User Control Process is responsible for logging the user in and logging the user out. It is active, therefore, only at the two extremes of the user's console session. When the user dials up the computer from a remote console, the Answering Service (see BQ.2.01) receives the dialup and creates a User Control Process to respond to it. There is one User Control Process for each attempt to log in. The Answering Service calls the user_control procedure in the User Control Process, giving it one piece of information - the registry file name of the user's console. (Registry file names permit unique identification of consoles and other system resources - see BF.3.00.)

The User Control Process has a modular design. In particular, four activities of the User Control Process are in distinct modules, called by the user_control procedure.

User_control calls the user_who procedure (see below) to determine the identity of the user. User_who reads the user's login command, checks his password, and makes sure he is a user known to the system. User_who is familiar with the organization of the User Identification Data Bases described in BQ.4.00; changes to that organization are reflected in user_who, but in no other module of the User Control Process.

User_control calls the user_in procedure to determine whether the user is allowed to add a process-group to the system at this time. User_in calls the relevant system modules, which are Accounting (see BO.0) and the Process-group Ranker (see BQ.5.00). (If some other module should become relevant or cease to be relevant to adding a user-process-group to the system, user_in should be modified to include a call to that module.)

User_control calls the create_overseer procedure to create an Overseer Process for the user-process-group. Create_overseer and user_in are also used by the Absentee Monitor (see BQ.2.04) when it creates an absentee process-group (i.e., a process-group that does not interact with a user at a console).

After create_overseer has created an Overseer Process and returned, user_control causes the Overseer Process to begin execution in the overseer procedure. The overseer procedure is described in more detail in BQ.3.01. Briefly, it enables the user to "quit" his work (by depressing the "quit button" at his console) and it initializes a subsystem for the user. The subsystem which it initializes is determined by the overseer procedure. The term "subsystem" in this section simply refers to the user's interface with the Multics after he has logged in. Users will normally use the Multics Command Subsystem - see BX.0.00 - as their subsystem. While a user is logging in, he may be considered to be in the "User Control Subsystem", to which he returns at logout time. Section BQ.3.01, on the overseer procedure, discusses how the user passes from the User Control Subsystem to some other subsystem, such as the Multics Command Subsystem.

After starting up the Overseer Process, the User Control Process goes blocked, waiting for a completion event from the Overseer Process signifying that the Overseer is "returning" and that the user is to be logged out. The overseer procedure sends such an event in response to one of four situations:

1) the user desires to log out;

2) the user hangs up his remote console (or the connection from his console is broken);

3) the user runs out of funds;

4) the system can no longer support the user's process-group (viz., because system load is too heavy).

When the overseer sends the completion event to the User Control Process, user_control calls the user_out procedure. User_out's job is to log out the user.

Descriptions of the individual modules in the User Control Process make up the remainder of this section. Figures 1 and 2 illustrate the order of calls in the User Control Process, and its relationship to other processes.

<u>User control (procedure)</u>

User_control is called with a single argument, the registry file name of the console on which the user dialed up. User_control consists largely of subroutine calls but does a few things itself:

1. Call user_who to identify the user.

2. Call user_in to determine whether the user may log in at this time.

3. Call the reserver to hold the console for the user so that only he may attach it (see reserve$hold, BT.3.01).

4. Call create_overseer to create the Overseer Process of a process-group for the user, and to cause the Overseer Process to begin executing the overseer procedure.

5. Wait, through a call to the Wait Coordinator (BQ.6.06) for an automatic logout event or a return (completion) from the interprocess call, i.e., a completion event signifies that the overseer is returning to its caller. The overseer does this only when it wants the process group to be logged out.

6. When an automatic logout event occurs, reflect the event to the Overseer Process and wait for a completion event from the Overseer. When a completion event from the overseer occurs, call user_out to log out the user.

7. On return from user_out, user_control returns. This return causes a completion event to be sent to the Answering Service, which had issued an interprocess call to the user_control procedure. The Answering Service interprets a completion event from user_control to mean that the user has been logged out. The Answering Service now destroys the User Control Process and awaits further dialups from the console on which the user had been logged in.

<u>User who</u>

User_who's function is to identify the user who is attempting to log in. User_who is closely tied to the user identification data bases described in BQ.4.00. To understand user_who the reader should be familiar with section BQ.4.00. User_who takes the following steps to identify the user:

1. User_who receives as an argument the registry file name of the console from which the user dialed up. It first checks this name against the names on the <u>dedicated console list</u> (see BQ.4.01). If

the console name appears on the list, that is sufficient to identify the user (i.e., that user is the only one who dials up on that console) and user_who returns with the name and project id of the user.

2. If the registry file name of the user's console does not appear on the dedicated console list, the user is expected to write a line at his console. User_who calls the read_login_line procedure to read this line and to interpret it. A login line has one of two forms as described above. Read_login_line announces its presence, waits a short time (2 minutes) for the user to input a line, checks the line for first word "login" or "proxy", and then reads the line to get the following items:

   a) name of a proxy (if any) logging in for the user,
   b) name of the user,
   c) project id of the user (optional),
   d) account id of the user (optional),
   e) working file directory path name (optional).

   Read_login_line returns these items to user_who.

3. If the user omitted any optional arguments, user_who looks up the defaults for that user. (This is not done as a single step, but each default is looked up as it becomes necessary.)

4. User_who calls ask_password to ask for the password of the user, or, in the case of a proxy login, the password of the proxy. User_who checks the typed password against that recorded in the user's (or proxy's) personal identification file (see BQ.4.02).

5. User_who checks the personal file (see BQ.4.02) of the user to be sure that the proxy (if any) is allowed to act as the user's proxy.

6. User_who checks the project directory directory (see BQ.4.00) to make sure that the project specified by the login line is a known project and that the user works on it.

7. If the user has successfully run this gauntlet, user_who returns to user_control each of the items in the login input line.

User in

The user_in procedure accepts as arguments the name, project id, and account id of the user trying to log in. The account id specifies the account to which the user wishes to charge this console session, and is one of the optional arguments in the login input line. User_in ascertains whether the identified user can be allowed to use the system at this time.

1. Check the permission files (which associate with each device the user access to the device - see BT.1.01) to be sure that this user is allowed to use this console.

2. Call the signon entry in accounting (see BO.3.02) to charge the activities of this process to the identified user. Signon returns, possibly with a status indicator that the user is insolvent.

3. Call the process-group Ranker (see BQ.5.00) to determine whether system load will allow another process-group to be logged in. The Process-group Ranker returns a yes or no answer. In the case of a "yes" answer, it also indicates the rank of the process-group, i.e., its liability to be automatically logged out, and an instance tag for the process-group. The instance tag of the process-group distinguishes it from other process groups of the same user which are running concurrently.)

4. Call the log keeper to enter in the User Log that this user has logged in. Information recorded in the User Log entry includes the user's name and project id, the instance tag of this process-group, the console name, and the date and time.

5. Return the instance tag of the process-group-to-be to user_control.

Create overseer

The create_overseer procedure calls the Process Control module (see BJ.1) to create an Overseer Process for a user-process-group. User_control calls create_overseer with two arguments: the process-group id of the new process-group (user name, project id, and instance tag) and the account id of the process-group (as specified by the user).

1. Call Process Control to create an Overseer Process, specifying process-group id and account id.

2. Return the process_id of the Overseer to user_control. After create_overseer returns, user_control sends an interprocess call causing the Overseer Process to begin executing the overseer procedure.

<u>User out</u>

User_control calls user_out after receiving a completion event from the Overseer. The user_out procedure has the function of getting the user's process-group out of the system. It assumes that the Overseer of the process-group has deallocated all resources of the process-group and done all other necessary cleanup before signalling a completion event to the User Control Process. That is, its sole responsibility is to destroy the Overseer Process of the group, and to notify other system modules (notably the Process-Group Ranker and the User Log) that the process-group is logged out. See BQ.3.01 for a description of the Overseer's activities at logout time.

1. Call the Log Keeper to record in the User Log that the process-group is logged out, specifying the process-group id, and the date and time of the logout.

2. Call the Process-group Ranker to inform it that the process-group is no longer logged in.

3. Call Process Control to destroy the Overseer Process of the user-process-group.

4. Return to user_control.

<u>Ring and Directory Residence</u>

This section has introduced 7 procedures:

user_control
user_who
read_login_line
ask_password
user_in
create_overseer
user_out

These procedures (with the exception of ask_password) should be executed only in a User Control Process. They should be written by very responsible programmers, and it must be possible to verify quickly the authorship of one of these modules. (Ask_password is also called by the password command - see BX.3.03.)

Certain of the above procedures access sensitive data bases or issue calls to hardcore or administrative ring modules which may not be issued by user ring procedures: For example, create_overseer and user_out call Process Control to create or destroy a new process-group (i.e., by creating or destroying its Overseer Process). User_in calls the Process-Group Ranker to make known a new process-group. User_who accesses passwords. User_control calls the interprocess communication module. These procedures reside in the administrative ring. The other procedures of this section may reside in the user ring.

Interaction with the User

Reading from the user's console is confined to the procedures read_login_line and ask_password. These can be reprogrammed, for example, to recognize the word "substitute" for "proxy" or "logon" for "login".

Before issuing an interprocess call to the Overseer, user_control arranges for reallocation of the console to the newly-created process-group. This it does by calling the hold entry in the Reserver (see BT.0).

This section has not discussed error comments, instruction of the ignorant, and the like. It is necessary to inform the user of his success or failure in logging in and, without divulging unnecessary information, to indicate the causes of failure (e.g., "password incorrect"). An indexed table contains all messages which user control modules might print. For example, message #10 might be "incorrect password". A German Multics installation would use a different table, which contained as message #10 "falsches Losungswort". The actual printing (i.e., the calls to the I/O procedure write - see BF.1.12) is done by the procedure login_print (in the user ring) which reads the indexed table of messages and calls write.

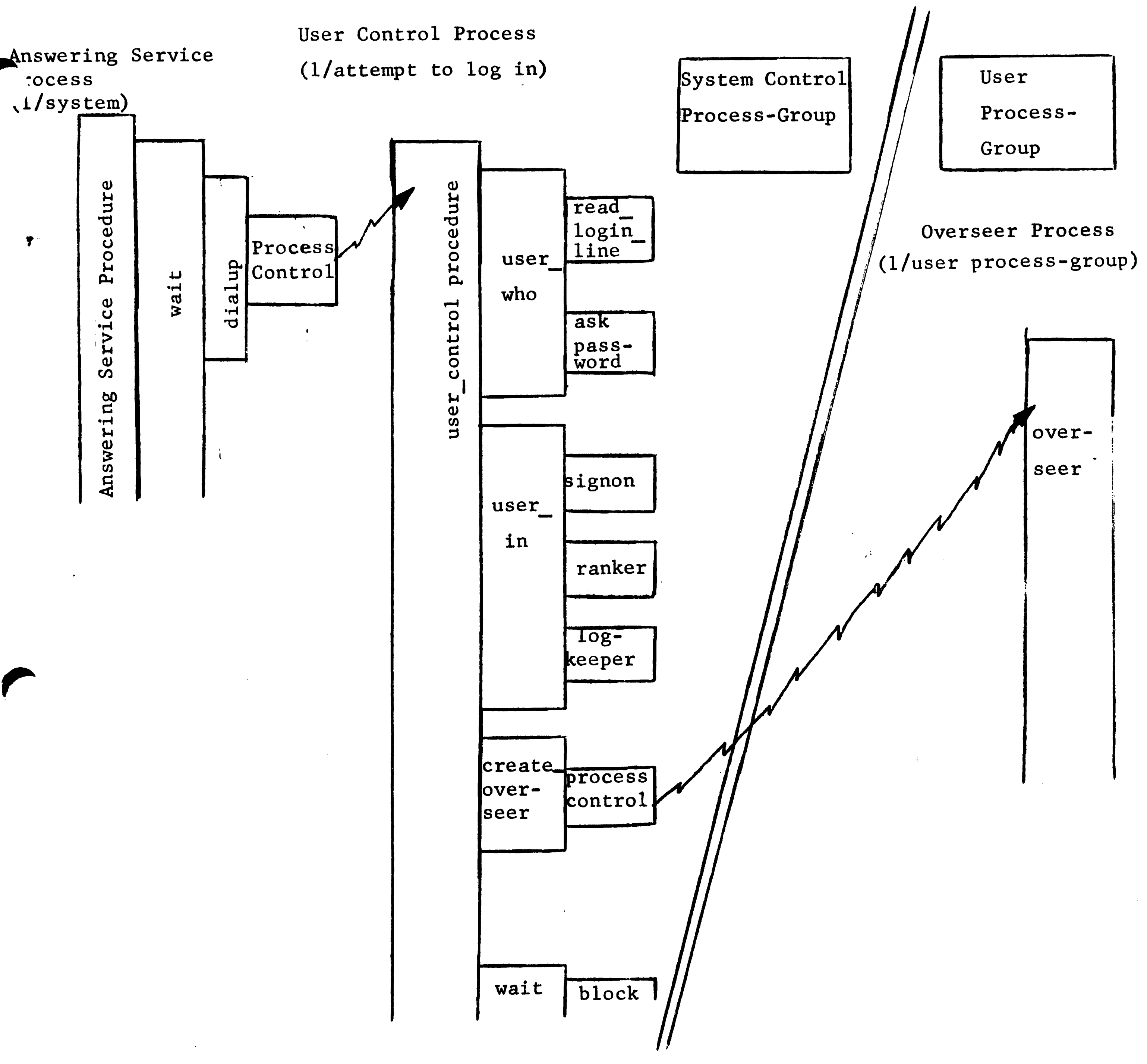

Figure 1: History of a Login

This figure shows the three processes involved in logging in a user process-group. The diagonal line is a boundary between process groups. The flow of control within each process is shown by a process portrait - a "building-block" picture of calls and returns which occur in the process. A horizontal line at the top of a "building-block" represents a call from the procedure represented by the "building-block" immediately to the left. A horizontal line at the bottom represents a return. A block without a bottom (return) line represents a procedure which has not returned by time a login is completed. A zap ($\rightsquigarrow$) from process control in one process to another process signifies that the sending process has created the receiving process and is causing a procedure to be executed by the receiving process, e.g., process creation in the Answering Service Process causes the user_control procedure to be executed in the User Control Process. In the login illustrated here, the User Control Process has created the Overseer Process and caused the overseer procedure to begin execution. The User Control Process is now waiting until it has to log out the process-group. The overseer will now create other processes in the process-group.

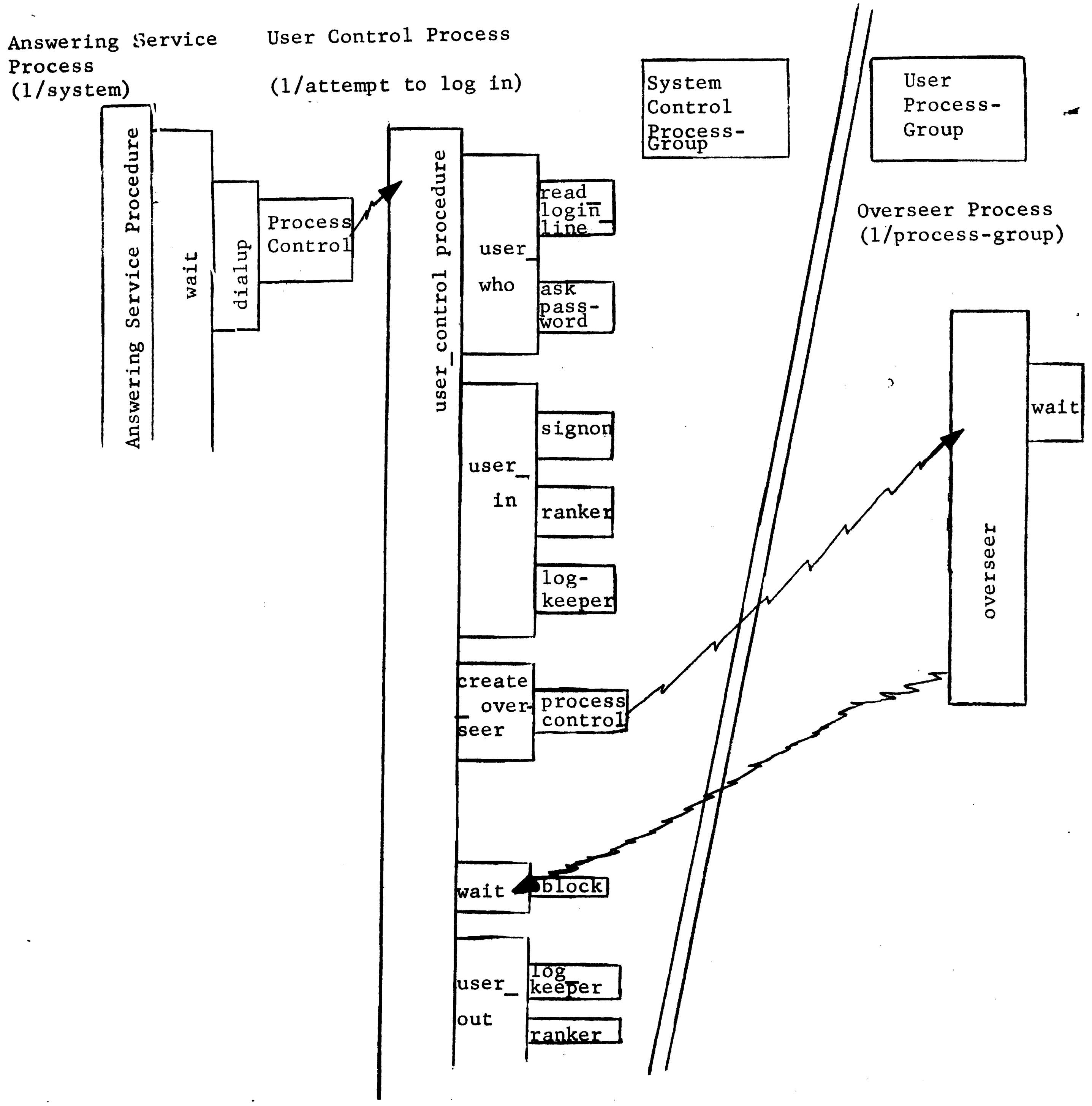

Figure 2: History of a Logout

This figure shows the three processes involved in logging out a user process-group. The diagonal double line is a boundary between process-groups. The flow of control within each process is shown by a process portrait - a "building-block" picture of calls and returns which occur in the process. A horizontal line at the top of a "building-block" represents a call from the procedure represented by the "building-block" immediately to the left. A horizontal line at the bottom represents a return. A block without a bottom (return) line represents a procedure which has not returned. A zap (⟿) from process control in one process to another process signifies that the sending process has created the receiving process and is causing a procedure to be executed by the receiving process, e.g., process control in the Answering Service Process causes the user control procedure to be executed in the User Control Process. A zap from a created process to its creator signifies the completion of the created process. In the logout illustrated here, the overseer procedure has returned, causing a completion event to be sent to the User Control Process. The User Control Process is now executing user_out, which will destroy the Overseer before returning. The Overseer Process has already destroyed the other processes in its process-group and entered