Published: 11/03/67

Identification

User Control Overview and Terminology
J. H. Saltzer, C. Marceau

Purpose

This section provides an overview of how the user "plugs into" the system, that is, of the system's view of the user from the time he dials up until the time he logs out. Functional descriptions of the terms used in user control sections are included in this section. Section BQ.3.00, which is closely related to this section, discusses the processes in a user process group. That is, it describes how the functions listed here are carried out in a multiprogrammed system.

User Control

The term "user control" refers to the system's view of the user from the time he logs in until the time he logs out. The concerns of user control are 1) logging the user in and out; 2) supplying the user with the ability to interrupt the work he is doing and the ability to save the state of the work he is doing; 3) supplying the system with mechanisms for limiting the number of users and in general ensuring that the system will not lose track of users' processes. The remaining terms discussed in this section pertain to the functions of user control.

System Control and User Control

System Control is responsible for control of the operating system, from the time the system is "brought up" to the time it is "shut down". User control is of interest to system control because user control includes the system's handle on users. In particular, system control can place a limit on the number of users allowed on the system or can decide to shut down the system although many users are currently on the system. User control provides the mechanisms for enforcing these decisions.

Login

A "login" occurs whenever a user explicitly requests and receives admission to the system. A user (a person working on a project) is always either a potential user (that is, a user in a state of potentiality), or, after he has logged in, an active user of the resources of the system. A successful login implies three things on the part of the system:

1) positive identification of the user (he is who he claims to be);

2) a decision that the user may use the system at the time of the login (the system is not too busy to accommodate him);

3) an entry in the user log recording the successful login, with date and time and other pertinent information such as console location.

The statements "the user logs in" and "the system logs in the user" mean the same thing. Although the former phrase is more precise, the latter is sometimes also used, especially to refer to the actions which the system takes when a user logs in. Both imply the user's volition and the system's response.

Logout

The term "logout" refers to the actions taken to "remove" a user from the system, that is, to change his status from that of an active user to that of a potential user of system resources. A logout implies that the user ceases to actively use the system and that he must log in again in order to use the system. "Logout" differs from "login" in that it does not always imply the user's volition. The system may on its own initiative log a user out (for example, because of system overload). A logout which occurs on the initiative of the system rather than of the user is called an "automatic logout".

The user may, however, voluntarily log out. The statements "the user logs out" and "the system logs the user out" are not quite synonymous. Both imply that the system removes the user from the system and records the logout in the user log, with date and time. The first statement carries the further implication that the user initiated the logout. The statement, "the system logs the user out automatically", implies that the user did not initiate the logout.

User-Process-Group

The concept of the process-group is presented in BQ.0 and for a good introduction to the meaning of the term the reader should consult BQ.0. Here we state only that processes which cooperate closely for a common purpose are associated in process-groups, with, for example, common access to data and procedures. A user-process-group is simply a process-group serving a logged in user. When

the user logs in, his process-group is created and when he logs out, it is destroyed. The user-process-group can be divided into two conceptual units: the overseer module of the process-group is a part of user control, concerned with providing standard services to the user and with implementing the decisions of the system control. The second unit of the user process group is the computation of the user himself. This computation is what the user actually sees of the system. It consists of the execution of the user's commands and procedures, and may be multiprogrammed or not. The computation is sometimes called the user's work, as in "the user desires to interrupt his work" or his computation.

Quit/Start Terminology

If the user desires to interrupt his computation he does so by pressing a key at his console known as the "quit button". When the user presses the quit button, he causes the overseer to stop the computation. Stopping the computation in this way is known as a quit. One of the functions of the overseer module is to implement quits initiated by the user.

Stop is the name of a procedure in the overseer module of a user-process-group. This procedure, if invoked for any reason, will immediately stop the computation. In general, the stop procedure is called by other procedures as one of their steps. One such procedure is the quit procedure, which executes in two steps:

1) invoke the stop procedure, described above,

2) condition the process-group so that it may accept new commands. This is done by invoking the quit responder (see below) of the user in a new computation.

That is, a quit causes the current computation to be stopped and a new computation initiated. The user then interacts with his new computation, typically by typing a new command. The supporting actions of the overseer module are invisible to the user, who is thus only aware of being able to start over afresh.

Start is a companion procedure to quit. It too is part of the overseer module, and its job is to undo the work done by quit. The start procedure may be invoked by the user who types a start command immediately following a quit. Other invocations of the start procedure occur as a result of unshelving an absentee computation (see absentee user terminology) or the resume procedure (see save/resume terminology).

Two other procedures which are related to quit and start are hold and reset. Reset causes the computation which was quit to be destroyed. Normally there can exist only two levels of computation simultaneously: one level which is currently doing the user's work, and one level which was quit. If a second quit follows the first, the previously quit computation is destroyed and there are again only two levels of computation.

It may happen that after quitting a user wishes to operate on his previous computation, for example to save it (see save/resume terminology). By operating on his quitted computation he is treating it as data of the current computation. The hold procedure, part of the overseer module, is invoked to hold a quitted computation as part of the current computation. After a hold there is only one computation; the current one.

Commands such as save or hold are not part of a user's computation but directions to user control (in this case the overseer) concerning the computation. Thus these commands are system standards and are not dependent on the particular subsystem in which the user may be operating.

Subsystems

When a user interacts with his computation he "sees" a subset of commands and procedures, for example, the Multics Command System. The user control MSPM sections refer to such a subset as the user's subsystem. Most general purpose users will use the Multics Command System as their subsystem, but other users might use a special purpose subsystem, such as one which interprets every line typed by the user as a request concerning airline reservations.

A subsystem, in the user control sense of the word, is defined by three elements: a login responder, or the program of the subsystem which interprets the first command a user types after login; a quit responder, which interprets the first command a user types after a quit; an automatic logout save flag, which, if up, indicates that the user's computation should be saved in case of an automatic logout or if the user's console hangs up.

Note that when a user types "start" after a quit, his command is interpreted by his quit responder. The Multics command system quit responder interprets the start command as an instruction to user control and not as a regular command line (the line is not interpreted by the Shell - see BX.3.06). Another subsystem might prevent the user

from communicating with user control simply by refusing to forward his requests to user control. In such a subsystem the user could not execute commands concerning his computation, but could only execute commands within his computation.

Save/resume terminology

It is possible to save the state of a computation so that it may be resumed at a later date. For example, in case of an automatic logout, if the automatic logout save flag is up, the overseer saves the user's current computation.

Saving a computation usually consists of preserving those processes which are performing the computation (see BQ.3.00 for the functions of processes in a process-group). To preserve a process means to freeze it in a stable state so that it can be resurrected and continue to run as before. A preserved process is a snapshot of a process and not actually a process. Thus it is not in the running, ready, nor blocked states. The process can be resurrected and placed in the blocked state, at which time it becomes capable of execution again. A preserved process retains its process I.D. but does not have a process directory (an entry in the process directory directory signifying that it is a known process) and, as stated above, is neither running, ready, nor blocked. (All this seems theologically appropriate.)

To save a computation it is necessary to preserve all processes executing the computation and to save the state of all input/output of interest to the computation. To resume a computation it is necessary to resurrect all processes which are part of the computation (that is, to restore the computation) and to start the computation.

Absentee User Terminology

An absentee user is one who requests a computation to be executed in his absence, that is, while he is not interacting with it from a console. Such a computation executes in an absentee user-process-group. The typical user is an interactive user, and his user-process-group is called an interactive group: the terminology introduced in this section up until now is concerned with the interactive user process-group. We now discuss those aspects of an absentee group which distinguish it from the interactive case:

1) The overseer module of an absentee process-group does not set up the mechanisms necessary for the user to request any of the functions of quit, start, etc., since the absentee user is presumably not directing the progress of his computation from a console.

2) As a general rule the system attempts to provide response to the interactive user's requests so that the interaction rate will be useful. However, response quality of the absentee group may typically be sacrificed in favor of both interactive groups and system efficiency.

3) If system load becomes very heavy or if the system must be brought down for a short time, the absentee group will be shelved (see below). A shelved group may be unshelved on the initiative of the system at some later time, over which the user has no control.

The common feature of these three characteristics of an absentee group is that the system assumes that no interaction between user and computation is taking place, and also that the user relinquishes control of his computation to the system. It is conceivable that an absentee user's computation will in fact attach a console and attempt interaction. However, the system is not designed to support an interactive computation in an absentee group. A user who tries such a combination will find himself with an ineffectual quit button (since the overseer module is not attached to the quit button) and with poor interaction time (since the system makes no special effort to secure him good response time).

An absentee computation can be submitted only by another computation (possibly absentee), in the form of a file of commands to be executed. This file is known as the absentee source file for the computation.

The computation is submitted to the absentee monitor process, a system process whose mission is to control the number of absentee computations which are currently competing for computer resources. That is, the user, through his computation, attempts to log in an absentee computation. With respect to this absentee computation the user is termed an absentee user. The absentee monitor process handles the login request. This type of login has the same characteristics as an interactive login: 1) The absentee monitor authenticates the identity of the user requesting login by noting the process group id of the process group requesting the login. 2) The absentee monitor decides whether the system will accept a new process-group of this user at this time. 3) The absentee monitor records the login in the user log.

An absentee computation begins in the shelved state. A shelved absentee computation consists of a collection of segments only, and is known only to the absentee monitor process. In addition to the absentee source file, associated with a shelved absentee computation is an absentee saved file, preserving the state of the absentee-computation at its last time of execution. (An absentee computation which has not yet begun to execute for the first time has no absentee saved file.)

From time to time, as it observes that resources are available, the absentee monitor process unshelves one or more absentee computations. Unshelving involves creating an absentee user process group and instructing it to execute the computation. The overseer module for the absentee group allocates all necessary resources to the process group and resumes the absentee saved file. If this computation is being unshelved for the first time, the absentee overseer simply causes the computation to begin in the user's login responder, which begins executing commands from the absentee source file for this computation.

The absentee monitor process may also observe that there are too many unshelved absentee groups in progress for the current collection of available resources. If so, it takes the initiative to shelve some of the executing absentee computations. A computation may be shelved by sending to its overseer an order to suspend operations (see BQ.3.01). The overseer will then perform the suspend procedure, which involves calling the stop and save procedures, deallocating all resources, and then returning control to the absentee monitor process. The save procedure saves the current state of the computation in an absentee saved file. The absentee monitor process then destroys the overseer process group and the absentee job is now considered shelved.

Note that an absentee job does not need to be logged out if the system is taken down. Instead, it is merely shelved, and when the system is restarted, it appears among the candidates to continue absentee operations. However the absentee computation itself may initiate logout, when it is complete, or the user who submitted the absentee computation may request that it be logged out by logging in (interactive or absentee) and sending the absentee monitor process an appropriate event signal (as the result of executing an appropriate command). The absentee monitor process will force the computation in question into the shelved state if it is not already in that state, and then complete the actions necessary to log out the computation.

The user may now paw about in the absentee saved file to determine the state of the computation, when it was cut off. He may also resume the computation in an interactive group (after suitable modifications to ensure that his computation takes commands from his console and not from the absentee source file).