Published: 06/09/67

Identification

Overview of the Reserver
Robert R. Fenichel

Purpose

(1) The Reserver provides a means for preallocating system resources (that is, for making reservations), and

(2) The Reserver is responsible for allocating certain resources at the time of use.

This document is primarily concerned with the reservation-making function of the Reserver.

Organization of This Section

This section is organized into nine subsections, as follows:

1. State-description of the Reserver
2. Reserver events and the Timetable
3. Making reservations
4. Benchmark events
5. Reservations which cannot be honored
6. Priorities of Reservations
7. Interface with the Accounting System
8. Interface with the Load Control System
9. Interface with the I/O System

1. State-description of the Reserver

At any given time, the Reserver has a single, central idea of what the system consists of. This world-view is called the state-description of the Reserver.

In many respects, a state-description resembles a dump of the Registry Files (BF.3). In particular,

(a) The Reserver is aware of the existence of certain fixed types of resources. The types known to the Reserver will primarily be a subset of those listed in the Registry Files (e.g., 9-track tapes), but some non-Registered

items may also appear. For example, one may be able to reserve "access". This reservation would not give one any devices, but it would tend to preserve one from bumping. For a complete accounting of the types known to the Reserver, see BT.3.2.

(b) For each type, the Reserver believes in the existence of some number of instances of this resource. The Reserver will believe, for example, that there is some specific number of tape drives on the machine.

According to installation policy, this number of instances may be larger or smaller than the number of actual devices known to the Registry Files. Airlines expect no-shows, and reserve more seats than they have; theaters hold house seats, and reserve fewer seats than they have. In any case, a reservation is a promise which has a non-zero but hopefully small probability of being broken.

(c) The Reserver also holds the following control information on a type-wide basis:

(1) May this resource be used without a reservation?

(2) What is the minimum permitted length of a reservation for this resource?

(3) What is the maximum permitted length of a reservation for this resource?

(d) For each instance of each resource, the Reserver performs a registry function. That is, it keeps track of the disposition of the instance: Is this instance now reserved? If so,

(1) By whom?

(2) For the purpose of describing his reservations, by what name, if any, does this user know this instance (e.g., "tape_drive_6")?

(3) With what physical device, if any, is this instance associated (e.g., "tape_drive_5")?

2. Reserver Events and the Timetable

The Reserver's state-description changes through time, in accordance with discrete events which are set into the Timetable. Care and feeding of the Timetable is the central task of the Reserver.

Two sorts of events are recognized: administrative events and user events. Administrative events generally cause type-wide parameters to be changed: number of known instances, minimum reservation length, etc. Administrative events are described in BT.3.3.

User events are what the Reserver was made for. Each user event either

(a) causes some instance to become reserved, or

(b) releases some instance from reservation.

The two types of user events are described in BT.3.4.

The Timetable is built up as reservations are made and as administrative edicts are issued. It is maintained in such a way that the events may be scanned in order of their intended occurrence. For more on the Timetable, see BT.3.5.

3. Making Reservations

A user will generally regard as a unit all those Reserver events necessary to run a single job. For example, a user might wish

(a) to reserve 6 tape drives,

(b) forty-five minutes later, to reserve 1 disk-pack drive,

(c) fifteen minutes later than that, to release 5 tape drives, and

(d) ten minutes later than that, to release the disk pack drive and the remaining tape drive.

This scheme will require 14 separate reserver events, and yet the unity of the scheme is clear. The user may not much care when the scheme is initiated, so long as its parts have the specified temporal relationships to each other.

Schemes like the one outlined above will be called reservation groups. A user will be able to specify a reservation group with a single call to the Reserver.

In the same call, the user must specify a name for the reservation group. The user is not permitted to own two pending reservation groups with the same name.

Finally, the user's call must indicate a bracketing calendar period during which the reservation group may acceptably (to him) be initiated. For example, the user might have specified that the reservation group shown should be started between 0900 and 1700 on April 1, 1968.

In order to fit the offered reservation group into the Timetable, the Reserver simply simulates forward through the Timetable, searching for a starting time within the bracketing period such that the reservation group, if started at that time, will not cause requests for instances which are not available.

The mechanisms of specifying and inserting reservation groups are described in BT.3.5.

4. Benchmark events

The simulation procedure just described may be exorbitantly time-consuming if the proposed reservation group is in the distant future. Benchmark events are available to ease this problem.

A benchmark event may be administratively placed at any point in the Timetable. This event causes no action whatever when its time comes.

During simulation, however, each benchmark event encountered is used as a repository for the current simulated state. The simulation process can then be described as follows:

1. Find a benchmark event set to "occur" shortly before the bracketing period. Set the simulated state from the contents of this event.

2. Simulate forward through the Timetable, up to the start of the bracketing period.

3. Try to find a suitable reservation start time within the bracketing period. A time t is suitable if it is possible

   (a) to pretend that the proposed reservation group has been inserted into the Timetable starting at t and suitably strung out.

   (b) to simulate forward from t until reaching at t' beyond the end of the reservation group a benchmark event whose remembered state is identical to the simulated state.

4. Alter all benchmark events between t and t' to show the up-to-date anticipated states.

Benchmark events are described more fully in BT.3.3.

5. Reservations Which Cannot be Honored

System resources may occasionally be overcommitted, so that certain reserved facilities are not delivered. This subsection is concerned with describing

(a) situations in which the Reserver will consider itself to have failed to deliver as promised, and

(b) what the Reserver will do in those situations.

There are exactly two ways in which the Reserver can be convinced of its own failure to honor a reservation.

(a) When an instance-reserving event occurs, the Reserver may learn from the Registry File Maintainer (BF.3) or some other real-world representation that no such device is available. For example, it might turn out that the Reserver contracted to deliver a specific tape drive which was destroyed by fire just before its intended use.

(b) In order to ease congestion, the Load Control Module (BQ.5) may bump a reservation-holder from the system. For example, 5 of the CPUs in a 6-CPU MULTICS might be simultaneously down.

In both these cases, the Reserver explicitly discovers some unavailability, either of an explicitly desired resource, or of general system access.

These cases of explicit unavailability must be distinguished from the case of improvidence. Perhaps in order to make effective use of a 12-tape, 2-hour reservation, a job needs at least 20 minutes of CPU time. Perhaps, too, a typical user only gets 3 CPU minutes per hour. In this case, the user reserving the 12 tapes should also reserve some special CPU treatment. If he does not do this, he has no claim against anyone but himself.

When a reservation cannot be honored, the Reserver really doesn't do much about it. If the system is down for ten minutes, it will hardly do to assume that all (or any) pending reservations may be pushed ten minutes back. Instead, the user is notified and, if he is absentee, the remainder of the reservation group is scratched. In addition, an administrator is notified so that repayment and reparation may be negotiated with the user.

6. Priorities of Reservations

Reservations which cannot be honored are a terrific embarrassment to the Reserver. Bumping of one reservation group by another will be performed only by administrative intervention. At the same time, it must be true that certain users should get a better chance than others at making reservations in the first place.

In order to satisfy these somewhat conflicting requirements, the following scheme is used:

(a) A portion of the state-description is the Timetable horizon, which is a calendar time in the future.

(b) Proposed reservation groups whose bracketing periods end before the horizon are accepted via the simulation mechanism described above and in BT.3.5.

(c) When a user submits a reservation group with a bracketing period not complete before the timetable horizon, the simulation mechanism is not used. Instead, the user's request is placed into a queue, one of several. The particular queue which this user's reservation group enters is specified in the user's profile.

(d) A special administrative event can cause the horizon to be pushed forward.

(e) Another administrative event causes the queues of (c) to be drained in a fixed order. Thus, users whose profiles associate them with early-to-drain queues stand a better chance of successfully reserving popular and overdemanded resources.

7. Interface with the Accounting System

This interface is described in BT.3.7. Its central notion is that all policy decisions in this joint area are made by the Accounting System (BO.3.07).

An example may be useful. Consider the user who wishes to reserve $X worth of resources.

(a) What if his account is adequate? Should the Reserver force him to tie up some funds in a "certified check" account for $X?

(b) What if his account has less than $X in it now? Perhaps it will "probably" be recredited to at least $X by reservation start time. But perhaps this user has never seen that much money in his life.

These questions are dodged by the Reserver. In general, the Reserver solves its Accounting problems by telling the Accounting system about almost everything that ever goes on in the Reserver. Blown by the prevailing fiscal winds, the Accounting system can react to the Reserver's raw data with what action it (the Accounting System) sees fit. In the initial implementation, the Accounting System will ignore calls from the Reserver.

8. Interface with the Load Control System

In order to allocate access to a finite system, the Load Control System (BQ.5) maintains a ranking of all logged-in process-groups. By manipulating certain threshholds, the Load Control System determines which users may log in and even, in some cases, which logged-in users must be bumped from the system.

The Reserver, then, is interested in the following aspects of this process:

(1) The Reserver must inform Load Control's Process-Group Ranker (PGR) at the time of the start of each reservation group. In this way, the PGR knows to give reservation-holders special slots in the ranking, so that few reservations will fail to be honored.

(2) The Reserver must further advise the Process-Group Ranker of the relative merits of the several reservation-holders. The system might be so disabled as to be able to support only a proper fraction of the current reservation-holders.

In the initial implementation, the Reserver considers one reservation-group worthier than another if the first was placed in the Timetable before the second.

(3) Out of respect for other users, the Reserver must inform the PGR when a reservation-group has been terminated. The PGR may then choose to reduce the security-against-bumping of the former reservation-holder.

(4) If a reservation-holder is bumped, the Reserver wants to know about it. This reservation-group is now known to be one which the Reserver could not honor, and the appropriate arrangements are made.

9. Interface with the I/O System

The Reserver stands as a sort of attach-verifier between the I/O System and the Resource Assignment Module. The

I/O system, in response to an attach call which implies that a device must be allocated to the user, makes an allocate call to the reserver.

When an allocate call is made without a reservation having been previously made,

(a) The Reserver rejects the allocate, on the grounds that this device must be reserved if it is to be allocated, or

(b) The Reserver rejects the allocate, on the grounds that this device is allocated to someone else, or

(c) The Reserver approves the allocate.

When an allocate call is made after a reservation has been made,

(a) The Reserver rejects the allocate, on the grounds that this device is already allocated to another reservation-holding user (this means that the system has overcommitted itself), or

(b) The Reserver approves the allocate, bumping the non-reservation-holding other user, if any. Of course, if allocation is set to be impossible without reservation, bumping will never be necessary.

This interface is more fully described in BT.3.1.