Published: 06/16/67

Identification

Summary of Reserver Calls
J. H. Saltzer, R. C. Daley

Purpose

This section provides in summary form the calling sequences of all calls to the initial Reserver, declarations of arguments, and brief comments as to meaning of calls and arguments. The Reserver is described elsewhere, in overview in section BT.3.00, and in detail in later parts of BT.3.

Summary of calls

Reservation-making calls are described first, followed by reservation-using (allocating) calls. Errors are reflected by setting the argument status to a non-zero value. The possible status values are described at the end.

Reservation making:

```
call reserve$resource(resource_name,start_time,hold_time,
     status);
```

makes a reservation in the name of this user for resource resource_name from time start_time to stop_time.

```
call reserve$type(resource_type,start_time,hold_time,status);
```

makes a reservation in the name of this user for some resource of type resource_type from start_time to stop_time.

```
call reserve$hold(resource_name,user_id,hold_time,status);
```

will hold the resource resource_name for the interval of time hold_time in a state such that only the process group with id user_id can perform successful allocation calls. (This call will be accepted only if the caller has the device allocated to himself.)

```
call reserve$group(group_name,resource_list,early_start_time,
     late_start_time,start_time_list,hold_time_list,
     assigned_start_time,status);
```

will, if possible, set up a reservation group for the current user for all of the resources in the array resource_list, as described in section BT.3.00. It will associate the name group_name with the

reservation group. The reservation group will begin at the time assigned_start_time, which will be set by reserve_group to some time between early_start_time and late_start_time. The reservation for the resource named in resource_list(j) will start at time start_time_list(j) after assigned_start_time, and will be for the interval hold_time_list(j).

(Entry reserve$group is not provided in the initial implementation of the reserver.)

```
call release$group(group_name,status);
```

will release the reservation group previously made under name group_name.

```
call release$resource(resource_name,status);
```

releases the reservation or hold on resource_name for this user.

```
call release$type(resource_type,status);
```

releases the reservation for a resource of type resource_type for this user.

Allocating calls:

```
call alloc$resource(type_name,resource_name,status);
```

allocates the resource resource_name to the calling process group.

```
call alloc$type(type_name,resource_name,status);
```

allocates a resource of type type_name to the calling process group. Resource_name will be set to contain the name of the particular resource of type type_name which has been allocated.

```
call de_alloc$resource(resource_name,status);
```

will release the allocation of resource resource_name.

```
call de_alloc$all;
```

will release all allocations currently held by the user process group.

Argument declarations.

```
declare resource_type character(32),
        resource_name character(32),
        group_name character(32),
        resource list(*) character(32),
        start_time_list(*) bit(72),
        hold_time_list(*) bit(72),
        early_start_time bit(72),
        late_start_time bit(72),
        assigned_start_time bit(72),
        start_time bit(72),
        hold_time bit(72),
        user_id character(50),
        status integer;
```

resource_type and resource_name are most commonly I/O system device types or names. Start_times are standard calendar clock times, and hold_times are standard calendar clock intervals. User_id is the user-process-group identification assigned by login.

Status returns.

The following values of the return argument status are defined:

1 A requested reservation cannot be made, for lack of resources.

2 A requested reservation cannot be made because of a possible error in the calling sequence (attempt to reserve unreservable resource type, stop_time precedes start_time, etc.)

3 A requested hold is not permitted.

4 Attempt to release non-existent reservation.

5 Allocation call made without previous reservation, and therefore rejected.

6 Allocation call made without previous reservation, but accepted. User takes chance on being bumped by a later reservation holder. (Only allowed on some devices.)

7 Allocation call cannot be accepted because of lack of resource; previously made reservation cannot be honored. Very sorry. (See project administrator for reparations.)

8 Allocation call cannot be accepted because of lack of resource. No previous reservation. Tough luck.

9 Allocation call cannot be accepted because of possible error in calling sequence. (e.g., illegal device name)

10 Allocation call is redundant--process group is already allocated this resource. Allocation remains.

11 Attempt to deallocate a resource which was never allocated.