TO: MSPM Distribution
FROM: M. A. Padlipsky
SUBJ: BT.3.02
DATE: 01/16/68

The attached revision describes the Initial Multics version of the Reserver.

Published: 01/16/68
(Supersedes: BT.3.02, 06/26/67;
BT.3.02, 06/16/67)

Identification

The Interim Reserver
S. I. Feldman

Purpose

This section describes the implementation of the Reserver planned for Initial Multics. See Section BT.3 for details on the final form of the Reserver.

Reserver Calls

I. Reservation-making calls:

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

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

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

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

II. Reservation-releasing calls:

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

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

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

III. Allocation-making calls:

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

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

IV. Allocation-releasing calls:

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

call de_alloc$all;

dcl type char(32),
```

```
resource_name char(32),
group_name char(32),
resource_list(*) char(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 char(50),
status fixed bin;
```

Implementation Plans

In initial Multics, it is impossible to reserve a device. Therefore, all calls in classes I and II (segments reserve and release) will be implemented by simple returns with a status argument of zero. Also, no per-user information will be kept regarding allocations of devices and media. Therefore, de alloc$all is implemented by returning a value of 12 for status. The other three calls are implemented, in a restricted sense.

Also, only I/O devices and media may be allocated and deallocated. Therefore, the Resource Assignment Module is not implemented, and the Interim Reserver calls the I/O Assignment Module (see BF.2.26) directly. (This will not be possible in Prototype Multics). The following describes the implementation of the three calls actually handled.

See the tables at the end of this section for a summary of the error codes returned by the Interim Reserver and by the I/O Assignment Module.

alloc$resource

To allocate a given resource to the present user, make the following call:

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

In response to this call, the Reserver makes the following call:

```
call ioam$assign(type,resource_name,cstatus);
dcl cstatus bit(18);
```

If cstatus equals zero, set status equal to 6 and return. If bit 4 of cstatus is ON, set status equal to 10 and return. If bit 10 of cstatus is ON, set status equal to 8 and return. Otherwise, set status equal to 9 and return.

alloc$type

In order to allocate a resource of given type, the user makes the following call:

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

In response to this call, the Reserver makes the following call:

```
call ioam$allocate(type,resource_name,cstatus);
dcl cstatus bit(18);
```

If cstatus is zero, set status equal to 6 and return. If bit 9 of cstatus is ON, set status equal to 8 and return.. Otherwise, set status equal to 9 and return.

de alloc$resource

If a user wishes to deallocate a resource, he make the following call:

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

In response to this call, the Reserver makes the following call:

```
call ioam$unassign(type,resource_name,"0"b,cstatus);
dcl cstatus bit(18);
```

If cstatus equals zero, set status equal to 12 and return. If bit 11 of cstatus is ON, set status equal to 11 and return. Otherwise, set status equal to 13 and return.

Summary of Status Codes used by the Interim Reserver

0 Unimplemented function
6 Successful allocation without reservation
8 Unavailable resource
9 Error in calling sequence of protection violation (resource may nott be allocated to this user)
10 Resource already allocated to this user
11 Attempt to deallocate a resource not allocated to this user
12 Successful deallocation
13 Unsuccessful deallocation (insufficient file permission)

Summary of Cstatus Bits for I/O Assignment Module

1 resource_name not found in IOAT
2 resource not assigned to this user
3 user not control user of device
4 resource already assigned to this user
5 user does not have write access to IOAT or IOAT does not exist
6 user does not have write access in type directory or type directory does not exist
7 user not permitted to assign device
8 no available resource in free pool
9 attempt to free an assigned device
10 attempt to delete resource assigned to a user other than the present user
11 resource assigned to other user
12 file already exists
13 no prototype file
18 system bug