Published: 05/03/68

Identification

Operator Confirmation Command
op_here
W. R. Strickler

Purpose

Interprocess communication is required between any operator's process and System Control, in order that System Control can signal the assignment of functions to the operator's process whenever such an assignment is made. Individual functions require interprocess communication between the operator's process and system processes or modules associated with the functions. These communication facilities must be established and maintained despite quits and housekeeping. Ultimately, certain functions may require that devices other than consoles be attached to the service operator's process-group.

Discussion

The command op_here performs the responsibilities of the operator's process in establishing and maintaining interprocess communication with other system processes and modules.

The first time the command is called in an operator's process-group (immediately after the operator logs in), op_here creates the segment "op_function" (which will contain the operator's function table), creates an event channel over which System Control can signal the addition of new functions to the table, and signals the operator's presence to System Control. (The command recognizes that it has been called for the first time by the fact that the segment "op_function" is not in the operator's group-directory). After signaling operator presence to System Control, the procedure scans the function table. On subsequent calls to op_here, the presence of segment "op_function" indicates that the invocation was the result of housekeeping or a quit. Then the op_here procedure simply scans the function table.

A special entry, op_here$check, must be called whenever System Control puts a new function in the operator's function table. (Conceivably it could be called by the operator from his console when he wishes to snoop and see if he has any new functions that, for some reason, have not been reported to him). A call to op_here$check results in the function table being scanned for new functions.

The function table scan results in the following being done for each function in the table: if the function has been appended to the table since the last call to op_here, inform the operator of his responsibility to service the function, attach any devices other than consoles that may be required, and initiate any interprocess communication required. If the function was in the table during the previous scan, but this scan follows creation of a new process, reestablish any needed interprocess communication.

Following any scan of the function table, op_here returns to its caller; the operator is then placed at command level. If no functions are in his table, his role is simply that of the garden variety user.

Usage

The command op_here must be called after a user logs in under an operator project, and after he gets a new process (as a result of a quit or housekeeping). The entry op_here$check is called after a new function has been appended to the operator's function table. The special login responder, op_listener (BX.15.01), calls op_here automatically after login, quit and housekeeping; the procedure for monitoring operator functions, op_checker (BX.15.03), calls op_here$check when System Control signals that a new function has been assigned. If op_listener and op_checker are not available, the operator must invoke op_here at his console after he logs in or quits and after housekeeping (i.e.,after each command sequence, if the housekeep option is on); he should also type op_here$check periodically if he wishes to know whether he has been assigned new functions.

Implementation

Two data bases (the segments "operator_comm" and "operator_rep") in the System Control group directory are accessible to all operator processes.

Information needed by an operator's process to signal the operator's presence to System Control is contained in segment "operator-comm". All operator processes have read access to the segment, which contains the structure:

```
dcl  1 op_comm based (p),
     2 sys_ctl_pid bit (36),
         /* System Control process id */
     2 op_rep_chn bit (70),
         /* name of event channel, accessible to any
            operator's process, for signaling operator's
            presence */
     2 op)req_chn bit (70);
         /* name of event channel, accessible only to
            System Operator for system requests associated
            with the system function */
```

Part of the information needed by System Control, to signal an operator's process that an addition has been made to his function table, is placed by op_here into segment "operator_rep". Since all operator processes have write access to the segment, there is the possibility of race among operator processes in reporting operator identification via the segment. Consequently op_here utilizes the Locker facility (BQ.7) to eliminate race. The segment contains the structure:

```
dcl  1 op_rep based (op),
     2 lock (4) bit (36),
         /* used by Locker facility */
     2 op_name char (24),
         /* operator's name */
     2 op_tag char (2);
         /* instance tag of operator's process id */
```

To avoid permanent locking of the segment as the result of an operator's process being quit after locking and before unlocking of the segment, op_here must use the ring 1 procedure, quit_inhibit (BQ.3.06). Consequently op_here must reside in the administrative ring.

Op_here creates the segment "op_function" in the operator's process-group directory and in it places the operator's process-id and the name of the event channel over which System Control signals that a change has occurred in the function table. The function table is a substructure in "op_function" and is filled in by System Control as functions are assigned to the operator. Obviously System Control has read, write, and append access to the segment, which contains the structure:

```
dcl  1 op_fcn based (fp),
       2 op_pid bit (36),                 /*operator's process id*/
       2 new_tbl_chn bit (70),            /*channel for System
                                            Control to signal change
                                            in table*/
       2 n_fcns fixed bin (17),           /*number of functions
                                            assigned to this
                                            operator*/
       2 fcn (fp->op_fcn.n_fcns),
         3 unsol_req bit (1),             /*"1" b if this is an
                                            unsolicited-request
                                            function*/
         3 just_in bit (1),               /*"1" b if this function
                                            entered table on this
                                            writing*/
         3 wait_chn bit (70),             /*event channel for
                                            unsolicited-request
                                            function*/
         3 name_lgth fixed bin (17),      /*no. of significant
                                            characters in name of
                                            function*/
         3 fcn_name char (32),            /*name of function and
                                            service procedure*/
         3 n_hpc fixed bin (17),          /*no. of significant
                                            characters in
                                            here_proc*/
```

```
3 here_proc char (32),          /*name of function_
                                  here procedure*/

3 n_apc fixed bin (17),         /*no. of significant
                                  characters in
                                  attach_proc*/

3 attach_proc char (32),        /*name of procedure
                                  to attach new ions.
                                  I/O devices*/
```

The following steps are taken by op_here:

1. Set a 1-bit switch, qsw, to "1" b, indicating that the operator has just been given a new process.

2. Call generate_ptr$initiate (BY.13.02) to get a pointer to segment "op_function". If the pointer is not null (the segment exists, indicating that this call to op_here is the result of housekeeping or a quit), go to step 10 for a scan of the function table. Otherwise (this call to op_here immediately follows operator login), proceed:

3. Create segment "op_function" by calling smm$set_name_status (BD.3.05) and give RWA access to System Control by calling mode$set (BY.2.05). Use get_process_id (BY.18.01) to put the current operator process-id into fp→op_fcn.op_pid, make function table null by setting fp→op_fcn.n_fcns = 0.

4. Create an event channel (see BQ.6.04) over which System Control can signal changes in the operator's function table. Store the name of the channel in fp→op_fcn.new_tbl_chn, and give System Control access to the channel.

5. Call quit_inhibit$on (BQ.3.06) to avoid permanently locking segment "operator_rep" in step 6.

6. If the segment "operator_rep" is locked, wait until it is unlocked; then lock it for this process, by calling locker$wait (BQ.7.00).

7. Use substrings of get_group_id (BY.18.02) to put operator's name and process-group instance tag into op→op_rep.op_name and op→op_rep.op_tag.

8. Signal operator's presence to System Control over the channel named in p→op_comm.op_rep_chn. (See BQ.6.04) Call the Wait Coordinator (BQ.6.06) to wait for a reflection signal from System Control over the channel named in fp→op_fcn.new_tbl_chn.

9. Unlock segment "operator_rep" by calling locker$reset (BQ.7.00). Call quit_inhibit$off. Go to step 11.

10. New process following housekeeping or quit: use get_process_id to put current operator process_id into fp→op_fcn.n_fcns; then proceed:

11. Scan function table. For each function in the table do the following:

    If the function has been placed in the table since the last execution of op_here (fp→op_fcn.fcn(i).just_in = "1"b), then print a console message informing the operator this function is one of his duties. If devices other than consoles must be attached (as indicated by fp→op_fcn.fcn(i).n_apc being non-zero) then call fake_call (BY.10.04) to call the procedure, named in fp→op_fcn.fcn(i)attach_proc, to perform the necessary attachment. If a "function_here" procedure exists (fp→op_fcn.fcn(i).n_hpc is non-zero), call fake_call to call this procedure (named in fp→op_fcn.fcn(i).here_proc); the named procedure will establish any interprocess communication facilities that must be provided by the operator's process for this function.

    Otherwise, the function was in the table during a prior execution of op_here in this process-group. If this execution is in a new process (qsw = "1" b) and if this is an unsolicited-request function (fp→op_fcn.fcn(i).unsol_reg = "1" b), then call fake_call to call the function, if any, named in fp→op_fcn.fcn(i).here_proc.

12. Return

The check entry sets qsw = "0" b and goes to step 11.