Published: 07/25/67

Identification

The Interprocess Group Event Channel Manager
Event channel protection
Michael J. Spier

Purpose

A number of event channel manager modules reside in the hardcore ring rather than in the administrative ring and are collectively known as the Interprocess Group Event Channel Manager (IPGECM).

One of these modules (set_event) is implemented in ring 0 for reasons of protection (access privileges to event channels), the others are functionally required to reside in the hardcore ring.

All IPGECM modules have one thing in common in that they all handle interprocess group communications, contrary to the ring one ECM modules which handle interprocess communications within the receiving process' group only.

This section covers two topics, it discusses the problems associated with event channel protection and the solutions adopted, then defines the modules of which the IPGECM is comprised.

The reader is assumed to be familiar with Interprocess Communication as described in MSPM sections BQ.6.00-04,06 as well as with the ring protection mechanism, MSPM sections BD.9.00-01.

The Event Channel Table protection

A receiving process' Event Channel Table (by this term we refer to both segments) is located in the administrative ring. It is manipulated by the Event Channel Manager which resides in the same ring. All Event Channel Manager Modules (including set_event) reside in the administrative ring. A receiving process' Event Channel Table is known to and accessible by all processes which belong to the same Process Group at a ring 1 level of protection. The same table may be made accessible to a sending process that does not belong to the same Process Group as the receiving process, yet it will be made accessible to such a process at a ring 0 level of protection only. The only way a sending process may access a receiving process'

Event Channel Table is by calling set_event. Consequently, interprocess group communication must be handled by a ring 0 copy of the ring 1 set_event procedure.

A sending process' access to the Event Channel Table is subject to the File System's Access Control. We therefore may assume that only processes that have an agreement with a consenting receiving process will actually gain access to its Event Channel Table, and this in either ring 1 or ring 0 depending upon the sending/receiving processes' process-group relationship.

We must now consider the fact that a sending process is not interested in a collection of segments known as the Event Channel Table, but rather in a data structure within these segments that is the event channel.

The Event Channel Access Control

A sending process which has gained access to the Event Channel Table may, theoretically, write information into any event channel within that table. We say theoretically, because in order to do so he must be in possession of the event channel name which is a unique bit string. Still, that possibility exists, and consequently individual channels have to be protected from being accessed by unauthorized sending processes. This is done by checking the event channel's channel-access-list which contains the process-group-ids of the sending processes which are allowed to signal over that event channel. A convention is made by which a zero-value access control list implies that any process which knows that channel's event channel name may signal over it.

Event channels are thus doubly protected by both the unique event channel name and the channel-access-list.

To summarize the scheme of protection as discussed this far:

1. There is an overall Event Channel Table protection in conjunction with a sending process' Process Group affiliation. Interprocess group communication is carried out at a ring 1 level of protection, interprocess group communication at a ring 0 level of protection.

2. A sending process must have the receiving process' permission to access its Event Channel Table (File system's Access Control).

3. A receiving process must willingly allow a sending process to access a given event channel by

   a. Communicating to him the channel's event channel name, and by

   b. Putting him on the event channel's access list.

At this point, a receiving process' event channel is protected from unauthorized access by other processes. We must now determine which procedures may signal over an event channel when the process they belong to has been granted access to the event channel.

Ring number validation level

The request for the creation of an event channel (call ecm$create_ev_chn) is initiated by a procedure belonging to the receiving process. Such a procedure resides in a given protection ring. We make the convention that the newly created event channel must be protected -- within the receiving process -- by the creating procedure's validation ring number and -- within the sending processes -- by an arbitrarily specified signalling-ring-number.

For example, every process has a number of event channels dedicated to events which are an integral part of the Multics system and of great importance to the existence and proper functioning of the process. Such channels, dedicated to system events like I/O interrupts, regular or automatic logout, device hangup etc. must be protected from being signalled over by unauthorized procedures. Furthermore, some of these channels are used in order to signal events which are "disagreeable" to a system user. Such a user, not respecting system conventions, may try, from his own ring, to delete such event channels and thus sever important system control communications.

Event channel creation

Create_ev_chn (described in MSPM section BQ.6.04) can be directly called by any procedure within a receiving process. It gets the caller's validation ring number from sb|3 and stores it in the created event channel.

Any subsequent call to an ECM or IPGECM module (other than set_event) is validated in conjunction with this ring number. No procedure, calling from a lower privilege (outer) ring is allowed to access the event channel in any way.

(Note: If the creating procedure wants the event channel to be accessed from rings of lower privilege than the one it resides in, it has to set its validation ring number accordingly before invoking create_ev_chn.)

In its calling sequence, create_ev_chn has an argument called "signalling-ring" which may assume any of the values 0-63. The value of this ring number is arbitrarily determined by the creator and corresponds to the lowest privilege ring from which a sending process' procedure may access the channel (via set_event).

Set event

As mentioned in the introduction to this section, set_event (described in MSPM section BQ.6.04) exists in two copies. One -- named ecm$set_event -- resides in ring 1, the other -- named ipgecm$set_event -- resides in ring 0.

When a procedure (other than a ring 0 procedure) wants to signal the occurrence of an event to a receiving process, it calls ecm$set_event which checks the receiving process' id against a ring 1 list which contains all the ids of this group's member processes. It can thus determine whether this is an intra- or inter-process group communication.

If it is an intra-process group communication, it retrieves the event channel in the receiving process' event channel table, checks for a possible ring access violation and only if the caller had the right to request signalling over this channel does ecm$set_event proceed to the actual writing of an event indicator into the event channel.

If it is an inter-process group communication, set event knows that the receiver's event channel table is (by hardware) inaccessible to it. It therefore passes the call along (having assumed its caller's validation ring number) to ipgecm$set_event in ring 0. The ring 0 copy of set_event first checks to see whether or not the sending process' group is allowed to signal over this event channel (channel access list), then goes through the same checking and signalling process as does ecm$set_event.

Other IPGECM modules

Thus far, we have discussed the ring 0 copy of set_event. Two of the Event Channel Manager's procedures originally reside in ring 0. These procedures are dedicated to device signal communications and must reside in the hardcore ring for two reasons:

a. All device signal communications are, by definition (see MSPM section BQ.6.03) Inter-process Group Communications.

b. These procedures must have access to the Device Signal Table that is wired down in ring 0.

These procedures are described in detail in MSPM section BQ.6.07.

```
call IPGECM$link_dev_chn(ev_chn_name,dev_index)
```

effects the coupling of the specified event channel with a device signal channel, located in the Device Signal Table and identified by dev_index, then hooks the event channel up to the device-signal-channel-list in the Event Channel Table.

```
call IPGECM$unlink_dev_chn(dev_index)
```

looks up the device-signal-channel-list and uncouples the event channel associated with dev_index from the corresponding device signal channel.