Published: 03/13/67

Identification

Summary of Calls of the Interprocess Communication Facility
B. A. Tague

Purpose

This section briefly summarizes the calls of the interprocess communication facility described in section BQ.6.00. Details of the implementation and use of the calls can be found in the sections BQ.6.02, and following. This document merely gives, for each module of the facility, the name and arguments of each entry, the protection rings in which the entry is accessible, and a very brief statement of what a call to the entry accomplishes.

Argument Definitions and Declarations

In the paragraphs that follow, the entry points to each module are given with their parameter lists. The PL/I declarations of each parameter that appears in those paragraphs is provided here for reference, along with a brief statement of what the parameter is. A knowledge of BQ.6.00 is presupposed.

channel_data_location bit(18)

This is a relative pointer to an entry in the event table or the interuser event table.

channel_id bit(72)

This is the unique identifier of an event channel.

channel_id_list array(*) bit(72)

This is an array of channel ids

channel_key array(2) bit(72)

This is an array consisting of a channel id and its receiving process id. It identifies a channel for the sending process.

channel_key_list array(*) bit(72)

This is an array of channel keys, two successive array positions holding each key. It should properly be a structure of an array of array(2) bit(72) arrays, but it is undesirable to use structures for arguments for efficiency reasons.

channel_mode bit(3)

This is a bit string identifying the channel mode as event queue (001), event count (010), or event time (100). The channel mode determines the kind of signalling possible over the channel.

channel_type bit(3)

This is a bit string identifying the channel type as intrauser (001), interuser (010), or device signal (100). The channel type determines the class of processes permitted to signal over the channel.

device_index fixed binary(17)

This is an index into the device signal table. It points to an entry in that table containing information about one device (GIOC channel, tape drive, etc.) attached to the system.

event_count fixed binary(17)

This is a count of the number of events signalled over an event channel operating in event count mode. The count is zero when the channel is created or after it is reset. It is increased by one for each event signalled.

event_id bit(72)

This is a unique identifier generated when an event is signalled over an event channel operating in event queue mode. It is added to the channel event queue by the sending process.

event_id_list array(*) bit(72)

This is an array of event ids generated when an array of event channels is signalled in a single call. If some of the channels are operating in event count or event time mode, the corresponding array positions may contain counts or times rather than event ids.

event_queue array(*) bit(72)

This is an array of event ids read out of an event channel queue. If the channel mode is event count or event time rather than event queue, the array will have size 1 and will contain the count or time read.

event_time bit(72)

This is a system clock reading. It is placed in a longer bit string than necessary for reasons of uniformity.

procedure_entry entry (bit(72))

This is an entry point to a procedure of one argument that is to be called as an event call. The argument is the channel id that signals the call.

receiving_process_id bit(72)

This is a unique identifier, the process id of the process that created an event channel.

reset_switch bit(1)

This is a bit which is one if the call in which it appears is to read and reset the event channel read. If the bit is zero, the channel is read, but not reset.

route bit(18)

This is a piece of data stored in the device signal table giving information about the physical address of the device. The interpretation of this data depends on the class of device.

sending_process_id bit(72)

This is the id of the process which is permitted to signal over an event channel. If it is zero, it is interpreted as either any process of the process group of the receiver (intrauser channel), or any process at all (interuser channel).

waking_channel_id bit(72)

This is the id of that event channel which causes return from a wait call.

wakeup_switch bit(1)

A bit that is one if the sender of an interprocess event signal wishes that a wakeup be sent to the receiving process. If the bit is zero, the event is signalled, but no wakeup is sent.

Event Channel Manager Entries

For additional details on the following calls, see section BQ.6.02. All of the entry points of this module are accessible in the user ring. The module itself resides in the administrative rin.

create_ev_chnl (channel_id, channel_type, channel_mode, sending_process_id)

This call causes an event channel to be set up with the calling process as receiving process.

index_ev_chnl (channel_id, channel_type, channel_data_location)

This call makes an entry in the event channel index for a channel created by a hard core procedure.

send_ev (Channel_key, event_id, wakeup_switch)

This call sends a signal over an event channel.

get_ev_q (channel_id, eventqueue, channel_mode, reset_switch)

This call reads out an event channel for the receiving process.

inquire (channel_id, event_count, reset_switch)

This call reads the number of events signalled over an event channel for the receiving process.

delete_ev_chnl (channel_id)

This call deletes an event channel for its receiving process.

ev_error

This function returns an error status string for the last call to the interprocess communication facility. See BQ.6.07 for details.

ev_chnl_info (channel_id, channel_data_location, channel_type, channel_mode, sending_process_id)

This call reads out channel data for the receiving process.

Interuser Event Table Manager Entries

For additional details on the following calls, see section BQ.6.03. All of the entry points to this module are accessible in the administrative ring, but not in the user ring. The module itself resides in the hard core ring. The entries are for the use of the event channel manager in controlling interuser event channels, and for the use of hard core ring procedures.

create_inter_chnl (channel_id, channel_data_location, channel_mode, sending_process_id)

This call is used by the event channel manager and hard core procedures to create interuser event channels

send_inter_ev (channel_key, event_id, wakeup_switch)

This call is used by the event channel manager and hard core procedures to signal over interuser event channels.

get_inter_ev (channel_data_location, event_queue, channel_mode, reset_switch)

This call is used by the event channel manager and hard core procedures of the receiving process to read out an interuser event channel.

delete_inter_chnl (channel_data_location)

This call is used by the event channel manager and hard core procedures of the receiving process to delete an event channel.

inter_chnl_info (channel_data_location, channel_mode, sending_process_id)

This call is used by the event channel manager and hard core procedures of the receiving process to read out interuser event channel data.

Device Signal Table Manager Entries

For additional details on the following calls, see section BQ.6.04. Aside from the get_dev_signal entry which is accessible in the administrative ring, all the entries to this module are accessible only to hard core procedures. The module itself resides in the hard core ring with some of its procedures in "wired down" core.

get_dev_signal (device_index, event_time, reset_switch)

This call is used by the event channel manager and hard core procedures to read device signal channels.

send_dev_signal (device_index, receiving_process_id, wakeup_switch)

This call is used by hard core procedures to signal over device signal channels.

set_auth (device_index, receiving_process_id)

This call is used by hard core procedures to assign a device signal channel to a receiiving process.

check_auth (device_index, receiving_process_id)

This call is used by hard core procedures to check that a receiving process has been assigned the device signal channel located by the device index.

set_route (device_index, route)

This call is used by hard core ring procedures to store certain routing information needed for device management.

get_route (device_index, route)

This call is used by hard core procedures to read out the information stored by the preceding call.

Wait Coordinator Entries

For additional details on the following calls, see section BQ.6.05. All of the entries of this module are accessible in the user ring. The module itself resides in the administrative ring.

wait (channel_id_list, waking_channel_id, event_queue)

This call is used to wait upon signals from event channels.

signal_ev (channel_key_list, event_id_list)

This call is used to signal events. It does not result in redundant wakeups to receiving processes that have more than one channel on the channel key list.

on_ev_call (procedure_entry, channel_id)

This call is used to set up an event call on an existing channel.

reset_ev_call (channel_id)

This call is used to reset the event call mechanism to enable it to receive another call over the channel.

delete_ev_call (channel_id)

This call is used to delete an event call. It does not delete the associated channel.