Published: 05/15/68

Identification

I/O Switch
D. A. Levinson

Purpose

The primary function of the I/O Switch (IOSW) is to forward I/O System user calls (outer calls) with their argument lists to the proper entry-point of the proper outer module. In addition, the IOSW performs a variety of other functions either closely related to its switching function or a natural outcome of its position between outer modules in the iopath. These functions include call validation, locking and unlocking data bases, allocating and initializing data bases, etc., as described in detail below.

It is recommended that the appropriate overviews and summaries, especially BF.0, BF.2.10 and BF.2.20 be consulted for motivation and background material for the IOSW. For the reader interested only in what calls correspond to entry-points of the IOSW, the column of Table 1 headed "Call Name" is a list of the outer calls.

Forwarding Outer Calls

Before the IOSW can forward the user call it must determine:

1) the proper outer module to which to forward the call,

2) the proper entry-point of that module.

The Proper Module

The explanation of the determination of the proper module necessitates mentioning the salient points concerning the call index, the Entry Point Vector (EPV, see MSPM BF.2.15) and the Attach Table (AT, see MSPM BF.2.13).

The call index of an outer call is a number conventionally assigned to the call.

The Entry Point Vector for a given outer module is a table such that the ith entry is a (forced) link (in the sense of BD.7.01) to the entry-point in that module corresponding to the outer call with call index i.

The Attach Table associates with each ioname contained in it, the EPV for the proper outer module. It associates, by default, the EPV for the Not Founder (NF, see MSPM BF.2.12) with any ioname not contained in it.

When the IOSW receives control, it references the Attach Table to retrieve the pointer to the EPV associated with the ioname given as the first argument of the call. Hence, the IOSW "knows" the outer module to which to forward control by the ioname/outer-module correspondence embedded in the Attach Table.

The Proper Entry Point

When the IOSW receives control at a given entry-point, it sets a variable equal to the call index corresponding to the entry-point. The index gives the proper offset in the EPV for the link to the corresponding entry-point of the outer module. Hence, the IOSW "knows" the entry-point to which to forward the call by the entry-point/call-index association.

Other Functions

In addition to its switching function, the IOSW:

1) validates the callers' right to access the ioname;

2) locks and unlocks the switchpoint corresponding to the referenced ioname,

3) allocates and initializes transaction blocks for target outer modules;

4) holds transaction blocks until the caller has been guaranteed an opportunity to exercise his hold privilege;

5) initializes callee's per-ioname data-base (PIB);

Validity Checking

The ring number of a procedure issuing a detach call for an ioname "alpha" must not be greater than the value of the attach_ring_no item of the Attach Table entry for "alpha". The attach_ring_no is initialized at attach time to the number of the caller's containing ring. The attach_ring_no is alterable by a procedure, the number of whose containing ring is not greater than the current value of the attach_ring_no. The IOSW implements this validity check and rejects violating calls.

Locking and Unlocking

The I/O system provides for shared use of a frame "alpha" by any process of a group in which "alpha" is attached. The user must use the interprocess communication facility to properly sequence the processing of "alpha" among the various processes of the group. However, the I/O system provides the logic to protect against the chaos that would inevitably ensue as a result of two or more processes of a group operating on the same frame at the same time. More specifically, parallel processing of calls referencing the same frame "alpha" is inhibited by the I/O system by means of calls to the Locker (BQ.7.00). If the I/O system has initiated processing of an outer call referencing "alpha" and issued in a given process then, until the corresponding return, processing of other outer calls referencing "alpha" and issued in processes other than the given process, there is no attempt to delay their processing, because to do so, would prevent recursive calls to the outer module corresponding to "alpha" and such recursion is useful to the I/O system.

The IOSW implements this one-call-per-frame-per-group mode of processing as follows: When the IOSW receives a call referencing the ioname "alpha" but before it references "alpha"'s data bases, the IOSW calls the Locker (locker $wait) to lock the iosegment associated with "alpha" (IS(alpha)). When control is returned, the locker will have locked it to all other processes which request it in the same way, except the current one and return.

When the IOSW receives the return from the outer module to which it has forwarded the call referencing "alpha", it calls Locker (locker$reset) to unlock the IS(alpha).

Allocation and Initialization of Transaction Blocks

Before the IOSW forwards an outer call to the target module, the IOSW calls the Transaction Block maintainer (TBM) to allocate a Transaction Block (TB) for the use of the target outer module. As described in BF.2.20, the TB is used to develop the status string which is returned to the caller, and as a link to related transaction blocks, transaction block extensions and buffers associated with the given transaction. The TBM initializes the status string (bits 1-126) of the TB to zeros and places the index of the TB in the Transaction Block Segment in bits 127-124.

Transaction Block Holding

Unless specific action is taken, a TB is deallocated at an unpredictable time after an event which depends on whether or not the caller is an outer module: If the caller is an outer module, the event is that the outer module has received the return corresponding to the call and has itself returned (necessarily to the IOSW). If the caller is a non-outer-module, the event is that the non-outer-module has received the return corresponding to the call and has issued a next outer call (referencing any ioname) in the same process. A callee-outer-module may have an interest in holding a TB, for example, because an asynchronous transaction is not yet complete and the transaction block contains information, including pointers to other data bases, required for subsequent processing. Thus, there is in the TB a caller_hold_bit settable by the caller, as well as a callee_hold_bit settable by an outer module. In addition, there is an interim_hold_bit (for use by the IOSW) to guarantee that a given TB is not deallocated before the above-specified events which, in turn, guarantees that the caller has the opportunity to exercise his holding privilege before the block is deallocated.

To implement this holding strategy, the IOSW sets the interim_hold_bit on allocation of a TB, and resets it on receipt of the third return (necessarily from an outer module) which follows the return corresponding to the call for which the TB was allocated.

Initialization of the Target Module PIB

Before calling the target outer module, the IOSW checks for the existence of a last, optional argument, which, if present, is a pointer to the caller's PIB. The IOSW uses the pointer to copy the sync_event_id and the error_event_id from the caller's to the callee's PIB.

If required, the IOSW primes the PIB with the current values of the driving table and auxiliary outer module pointers.

Finally, if not already present, the optional, last argument is added to the argument list, and in either case, the IOSW sets its value to a pointer to the callee's PIB.