Published: 08/24/66

Identification

Wait Coordinator
J. H. Saltzer

Purpose

The Wait-Coordinator is a set of procedures designed to facilitate inter-process control communication via the Traffic Controller entries block and wakeup. The Wait-Coordinator has three functions, namely:

1. Handing out unique name tags to identify "events" corresponding to task completion, etc.

2. Waiting for a selected set of events to occur.

3. Notifying another process that an event of interest to it has occurred.

To accomplish these functions, the Wait-Coordinator maintains a per-process table of events of interest to the process.

In addition, the Wait-Coordinator does validity checking of requests by one process to notify another.

Introduction

An event is anything which is observed during the execution of some process and which is of interest to some other process, or perhaps some other procedure of the first process. Associated with an event may be an event-variable which has value "0"b before the event occurs and value "1"b afterwards. Also associated with an event may be an event-name, which is a unique identifier for that event. A typical pattern of communication between two processes is as follows: One process (process "A") establishes an event-name and then places a request for the other process (process "B") to do something, along with the event name and its own process identifier, in a data base common to both processes. The event in this case corresponds to the completion of the requested computation by the process "B". Process "A" can now go about its business doing other work, if desired, or it may inquire as to whether the event has happened yet, or it may wait for the event. Process "B", when finished with the requested computation, notes the event by calling the wait-coordinator, giving as arguments the event-name and the identification

of process "A". "B" might then block itself pending another request.

Event Identification

A process may establish a name for an event in one of two ways:

```
call set_event (event_name, process_id);
```

where

event_name is a bit string of length 70 bits in which set_event will store a unique event identification tag. (See BB.1).

process_id is the identification number of the process which will note the fact that the event has happened.

Set_event creates an event name and places it and process_id in a table of events. This table belongs to the process calling set_event, but it is accessible to any other process which notes events of interest to this process.

Alternatively, one may establish a name for an event by:

```
call set_event_cell(event_name,event_cell);
```

where

event_name is described above

event_cell is a pointer to an event cell. An event cell is a structure containing two variables: a bit string of length 1 which will be set to "1"b when the event is noted, and an integer (precision 63 bits) in which will be placed the calendar time at which the event is noted.

Set_event_cell creates an event name and enters it and the event_cell location in the table of events.

Set_event is used for most routine inter-process communication. Set_event_cell is used for special applications in which event-cell is located in an agreed upon common data base, (other than the event table) between the communicating processes. It is intended primarily for communication with system interrupt handler procedures, in which case event-cell is located in latched core storage. (Since these procedures cannot take missing-page faults at interrupt time, they cannot call note_event (see below) which accesses the paged event table.)

Event noting

A process may note that an event of interest to another process has occurred by

```
call note_event (event_name, process_id);
```

where

event_name is the unique identification number of the event whose happening is being noted.

process_id is the process which is interested in the event event_name.

Note-event will mark this event as "happened" in process_id's event table, and call wakeup (process_id) in the traffic controller.

Consequently, if a process has access to an event cell of another process, it can set that event cell to "1"b, with the appropriate calendar time, and call wakeup (process_id) by itself. This latter technique is the only technique available to certain supervisor procedures which cannot tolerate a page fault.

Event checking

A process may inquire as to whether or not certain events have happened yet in one of two ways:

```
z = inquire (event_name, event_time);

call wait (event_list, count);
```

where

event_name is an event name

event_list is an array of event names

count is an integer

inquire is a function whose value is "1"b or "0"b, depending on whether or not the event has happened yet.

event_time if inquire has value "1"b, event_time will contain the time that the event was noted.

If at least "count" of the events in event_list have happened, wait will return immediately. If fewer than "count" of

the events have happened, wait will call block in the Traffic Controller. Wait will not return until "count" events in the list have happened.

Discarding events

Whenever a process has no further interest in an event, it may have the event removed from the event list by

```
call reset_event (event-name);
```

Reset_event may be called whether or not the event has happened yet.

The event table

There is one event table for every process which uses the Wait-Coordinator. It is paged, and accessible to any process which may note events of interest to the table's owner. For each event which has been named by a process, there is an entry in the table containing:

1. Unique event-identification number. This number, created by set_event or set_event_cell, consists of the calendar time concatenated with the serial number of the processor which was used to name the event.

2. Event_cell switch. This switch, if on, indicates that item three is a pointer to an external event_cell, and items four and five are invalid. On the other hand, if the switch is off, item three points to item 5, and item 4 is a process_id.

3. Event_cell pointer. This is a pointer to the event_cell corresponding to this event.

4. Process_id. This is the identification number of the process which is allowed to note the event.

5. Event_cell. This is a structure, containing

    1. Event switch. If this switch is on, the event has happened.

    2. Event time. This is the calendar time at which event switch was turned on.