Published: 08/07/67

## Identification

The give-call facility
Michael J. Spier

## Purpose

The "give-call" facility is an extension of the Interprocess Communication Facility, allowing procedure to procedure calls to be made over process boundaries between member processes of a process group. Give-call allows calls to be made to arbitrary procedures (within certain limitations of ring access privileges and segment access rights) in the same way in which calls are made between procedures of the same process.

It is assumed that the reader is well acquainted with the Interprocess Communication Facility as described in MSPM sections BQ.6.00-07. In order to avoid misunderstanding it should be remembered that there is no conceptual relationship between the give-call facility and event-call-channels (even though both happen to have the word "call" as part of their name) but rather that give-call, as a user of Interprocess Communication, may or may not make use of such event channels.

## Introduction

It is sometimes necessary for some process to assume control of another process. A typical example is that of an overseer process which creates a working process and wants to control the latter's initiation. Another example is that of a process which wants to have some task (e.g., matrix inversion) executed for it in parallel by another process.

Give-call was designed to perform such calls over process boundaries and provides (within the limits of certain restrictions discussed in the implementation) the following services:

a. Allows the calling of an arbitrary procedure, known by its path name, within the called process.

b. Allows the called procedure to directly access the call arguments (within the calling process).

c. Provides appropriate handlers for condition signalling and abnormal returns.

d. Simulates a return from the called procedure by ending an event signal to the calling process over an agreed-upon event channel.

The above-stated may seem to be very complicated, but in fact give-calls are as straightforward and as easy to make as calls which are internal to a process, the only major difference being the simulated return, imposed by the parallel processing aspect of give-call.

To return to the above-mentioned example of an overseer process which wants to initiate a newly created working process, the latter is blocked in its Wait Coordinator, awaiting an event signal over an event channel dedicated to the give-call facility.

The overseer process ("calling process") issues a give-call to a procedure within the working process ("called procedure"). The give-call facility sends an event signal to the called process over the dedicated event channel, causing the called process to wake up in its Wait Coordinator and call the called procedure on behalf of the calling process.

The give-call facility is composed of two modules, named gvclf$give_call and gvclf$pass_call (both in ring 1), which are interposed between the Interprocess Communication Facility and the calling and called procedures respectively. The give_call procedure copies the necessary information associated with the interprocess call into a data base shared by both processes (known as the "give-call table" (gvclt)) together with an event id for future identification, and then sends an event signal to the called process over the dedicated event channel. The called process receives the event signal and transfers control to pass_call. Pass_call retrieves the information in the give-call table (by matching it with the received event id), generates a calling sequence to the called procedure and invokes the latter in the normal manner. Upon return, pass_call sends the calling process an event signal over an event channel (whose name it finds in the give-call table) in order to inform it of the completion of the give-call, and returns control to the Wait Coordinator.

Overview

The give-call facility is designed to allow interprocess calls to be made in the same manner as calls which are internal to a process. It imposes upon the caller a somewhat

different calling sequence (explained in the implementation), but accepts as called procedure any arbitrary procedure in the called process, allowing it to be completely unaware of whether the call came from inside or outside of the process it belongs to.

The following conventions must be followed in conjunction with the give-call facility:

a. Give-calls are legal only between member processes of a process group.

b. Give-calls can be made from and to procedures in rings 1-63 (ring 0 procedures do not use this facility).

c. Even though a give-call is handled by ring 1 procedures (give_call and pass_call), the actual call to the called procedure emanates from the ring on behalf of which the give-call was initiated.

d. The list of arguments for the called procedure must contain pointers to the arguments' symbol table entries (see MSPM section BD.1.00).

The give-call facility's primary task is that of "normalizing" the argument pointers when it passes the called procedure's argument list from the calling to the called process. It converts only the argument pointers in the argument list (including specifiers). No attempt is made to pry any deeper and convert process-dependent information (such as pointer or label variables). As a rule, only arguments recognized as legal by the ring-crossing mechanism (see BD.9.01-03) are validated and, -- as an additional restriction -- no process-dependent information should be used as a give_call argument.

Implementation

Procedure a of process A wants to issue a give-call to procedure b of process B. The calling procedure knows the called process' id (prcs_b_id), the called procedure's calling name (symbolic character string name of the form

"sqrt" or "function$sine") and its arguments (a ,a , ... ,a ). Process A creates an event channel (rtn_chn) over which it is going to be notified that the called procedure has returned, and then calls

```
gvclf$give_call("proc_b_name",prcs_b_id,rtn_chn,stat_rtn,
   a ,a , .. ,a )
```

where stat_rtn is a location, specified by the caller, int which give_call puts return status information.

Give_call has the name of one of process B's event channels (giv_call_chn) which it knows to be dedicated to the give-call Facility. Furthermore, process A has a single event-call channel (giv_call_rtn) dedicated to returns from called procedures and associated with the give_call procedure (not to be confused with rtn_chn).

(Note: There is only one (if any) giv_call_chn and giv_call_rtn channel per process.)

When called in the above-described manner, give_call goes through the following steps:

a. It separates its own (first four) arguments from the called procedure's arguments.

b. It opens a new entry in the give-call table into which it puts a newly-generated event id (for table entry identification) and giv_call_rtn. It also stores rtn_chn in this table entry, although not for immediate use.

c. It asks the Segment Management Module for procedure b's path-name, which it stores in the table entry together with procedure a's validation ring number, obtained from sb|3.

d. It reconstructs a new argument list from the remaining (called procedure's) arguments and invokes a module which converts each pointer into a path-name/relative-pointer pair, to be stored in the table entry.

e. It then calls set_event and sends process B an event signal over its giv_call_chn, specifving the same event id which was stored in the give-call table entry.

f. Give_call returns to its caller.

Process B receives (in one of the two ways described below) the event signalled over its giv_call_chn and, knowing the event channel to be associated with the give-call facility, transfers control to pass_call. This transfer of control depends upon giv_call_chn's receiving type. If it is an event-call channel having pass_call as its associated procedure, the event will be detected and control automatically transferred to pass_call when process B executes in the Wait Coordinator. This is the usual way of receiving give-call event signals.

It is also possible for giv_call_chn to have the event-wait type attribute and to have one of process B's procedures (other than procedure b) wait for an event to be signalled over it. In that case, upon return from the Wait Coordinator, this procedure calls

```
gvclf$pass_call(gvclt_ptr,ev_ind)
```

where gvclt_ptr is a pointer to the process' give-call table and ev_ind is the event indicator returned by the Wait Coordinator.

Pass_call goes through the following steps:

a. It finds the current give-call table entry by matching the received event id with the stored one.

b. It calls the conversion procedure which reconverts the path-name/relative-pointer pairs into a standard Multics list of arguments.

c. The called procedure's path-name is converted into a pointer.

d. Pass_call generates a calling sequence to the called procedure. This calling sequence is generated in a segment which resides in the calling procedure's ring. Pass_call declares itself as handler for access violation and unclaimed signals (see MSPM section BD.9.01) and invokes the generated calling sequence. This mechanism insures that the actual call to procedure b will always emanate from the calling procedure's ring, providing the called procedure with actual hardware protection.

Upon return from the called procedure, the generated calling sequence returns to pass_call, which does the following:

e. It deletes the generated calling sequence.

f. It puts return status information into the current give-call table entry (for example, information associated with intercepted access violation or unclaimed signals).

g. It then sends the calling process an event signal over giv_call_rtn specifying the same event id used before (for table entry re-identification).

h. Pass_call returns to its caller.

The calling process receives the event signal over giv_call_rtn and transfers control to give_call (remember that giv_call_rtn is an event-call channel associated with give_call), which finds the current table entry, retrieves the returned status information and deletes the entry. It then sends the calling procedure an event signal over rtn_chn (note: in this case the event is signalled and received within the same process) and returns to the Wait Coordinator.

Protection

The ring protection problems associated with the give-call facility seem at first to be rather complex. The whole Facility is nothing but a mechanism which is invoked to re-issue a call on behalf of a procedure which resides in a given protection ring. The call is re-issued from the administrative ring which has considerable access privileges. The Facility could thus be used by some "unfriendly" procedure as a means of circumventing the system's protection mechanism. This possibility is excluded by having pass_call invoke the called procedure from the calling procedure's ring. Now another problem arises, namely, if an access violation happens to occur, it is the called (rather than the calling) process which "suffers punishment." It is for this reason that pass_call declares itself as handler for access-violation conditions. If such conditions are intercepted, pass_call signals them to the calling process, which may then invoke its own appropriate handlers.

## Initial Implementation

In the Initial Multics implementation, give_call is restricted as follows:

a. It may only be used by administrative ring procedures, although possibly to make outward calls to user ring procedures.

b. PL/I signals may not be invoked by the called procedure.

c. No arguments may be passed.