Published: 07/19/67

Identification

The Initial Scheduler
A. Evans

Purpose

It is the scheduler's job in Multics to place the process for which it is operating into the ready list, the queue of processes which are ready to run. The process' position in this queue is dependent on its relative priority with respect to other processes in the queue. In addition, the scheduler sets the maximum execution "time" the process will be permitted to run when it comes to the head of the queue. Multics in its initial implementation uses a very primitive scheduler which operates on a simple round-robin basis. In general, a process to be scheduled is assigned a fixed amount of time and placed at the end of the ready list. An exception is made for a process which, because it has set interlocks on hardcore ring system-wide data bases, is entitled to temporary priority. Such a process is placed at the head of the ready list.

Preface

An overview of the operation of scheduler is given in Section BJ.4.00, with which familiarity is assumed. However, some simplifications are possible in this initial version. In particular, the priority of the process being scheduled is independent of all considerations of other processes in the system or of information about system loading. Either the process has set interlocks on critical data bases, in which case it goes at the head of the queue, or not, in which case it goes at the tail of the queue. Further, BJ.4.00 specifies calculating the priority before interlocking the data bases, to keep to a minimum the time the lock is set. The closest concept in this initial scheduler to the calculation of priority is the examination of the block_lock_count to test for temporary priority, and this test is so simple that it is done while the lock is set.

Details of Operation

In the following description, a reference such as "time_limit for the current process" refers to tc_data$apt.entry(i).time_limit, where i is the index of the entry in the active process table (APT) of the current process. The APT is described in detail, with its complete declaration, in Section BJ.1.01.

The ready list for the initial scheduler is discussed in Section BJ.4.05. The index in the APT of "the current process" is in the Process Data Block at pds$apt_index. (See BJ.1.04.)

The scheduler performs the following:

1. Set the time-limit item in the APT for the current process to x cycles, where x is a constant set by the system administrator.

2. Save the current processor mask and mask the processor against all interrupts.

3. Lock the current process' state switches (ready_sw and running_sw) by interlocking its state_lock. If these switches are already locked, it is necessary to wait in a loop until they are free.

4. Set the state to ready. That is, set ready_sw in the current entry to "1"b and running_sw to "0"b.

5. Lock the ready list by interlocking the ready_list_lock. If the ready list is already locked, it is necessary to wait in a loop until it is free.

6. Put the current process (i.e., the process being scheduled) onto the ready list. If the ready list is empty, the current process is merely inserted. Otherwise, its position in the ready list is dependent only on whether or not it has temporary priority, which is determined by examining pds$block_lock_count. If this quantity is non-zero, the process has interlocks set on common data bases and is entitled to temporary priority. Do one of (a), (b) or (c).

   a) The process is not entitled to temporary priority. For the current process, set ready_list_frwd to zero and ready_list_bkwd to the value now in ready_list_tail. In the entry pointed to by ready_list_tail, set ready_list_frwd to point to the current process. Finally, set ready_list_tail to the current process.

   b) The process is entitled to temporary priority and so is to be placed at the head of the ready list. For the current process, set ready_list_frwd to the

value currently in ready_list_head and ready_list_bkwd to zero. In the entry of the process pointed to by ready_list_head, set ready_list_bkwd to point to the current process. Finally, set ready_list_head to point to the current process.

c) The ready list is empty. Set ready_list_head and ready_list_tail to point to the current process, and set ready_list_frwd and ready_list_bkwd of the current process to zero.

7. Unlock the ready list, by setting ready_list_lock to zero.

8. Unlock the process state switches, by setting the process' state_lock to zero.

9. Restore the processor mask to the value saved in step No. 2, above.

10. Return to the caller.

Discussion

The quantity set in step 1 is the 24 bit value which will be loaded into the hardware interval "timer". This latter is a register which counts down one each time the processor accesses memory. When the count reaches zero, a timer runout fault occurs whose effect is ultimately to cause the running process to schedule itself for later execution and then to block. (See BK.3.07 for information about the interval timer and the immediate processing of the fault it creates, and BJ.3.04 for the remaining processing.) In the initial scheduler all processes are always assigned the same permitted execution at each scheduling operation. The quantity is in the Traffic Controller Data Block (see BJ.1.08) at tc_data$time_limit.

The masking and unmasking of the processor in steps 2 and 9 are done by the procedure master_mode_ut$set_mask, described in Section BK.5.01.

The interlocking in steps 3 and 5 is done, ultimately, with the 645 opcode stac -- store a-register conditionally.

In steps 3 and 5 there is the unusual procedure of a fully masked processor waiting in a loop for an interlock to be released. Such an action may be catastrophic, unless it is certain

(a) that the lock was set by a process currently executing in another processor, and

(b) that it will be released by that process before that process can be interrupted or go blocked.

(If the other process could block with the lock set, the next process to run might need the ready list. There would then be a fatal hangup.) In the present case, all necessary conditions are met. By convention, a process' state switches and the ready list are only locked by a processor that is fully masked, and the interlock is released before unmasking.

It should be clear that the processor must be fully masked while the scheduler is using the ready list. Any interrupt received will probably result in the initiation of a wakeup for some process -- the one responsible for the interrupt. But wakeup (see Section BJ.3.02) involves calling the scheduler, requiring access to the ready list. BJ.6 contains a detailed discussion of interlocks in the process exchange.

In step 6, the order of setting the various switches is not critical, since the processor is masked during this step.