Published: 07/19/67

Identification

The Ready List-Initial Implementation
A. Evans

Purpose

The ready list is a data base listing the processes which are ready to run. The scheduler (see BJ.4.00) places a process into the ready list in the place appropriate for it. The dispatcher (see BJ.4.02) selects for running that process which is at the top of the ready list. Quit (see BJ.3.03) will remove a process from the ready list, if it is called on behalf of a ready process. Finally, swap_dbr (see BJ.5.01) actually removes from the head of the ready list the process about to execute.

Implementation

There is no independent data base known as the "ready list" -- instead, it is implemented as two threads through the Active Process Table (APT). The APT is discussed in detail in Section BJ.1.01, which includes its complete declaration. The part of the APT declaration relevant to the present discussion is

```
dcl 1 tc_data$apt external,
    ...
    2 ready_list_lock bit (36),
    2 ready_list_head fixed,
    2 ready_list_tail fixed,
    ...
    2 entry (200),
      3 process_id bit (36),
      ...
      3 ready_list_frwd fixed,
      3 ready_list_bkwd fixed,
      ...
```

The first process on the ready list is the one whose index is in

```
tc_data$apt.ready_list_head
```

That is, the process identification of the next process to run is

```
tc_data$apt.entry(tc_data$apt.ready_list_head).process_id
```

For any process with subscript i, the index of the next process in the ready list is in

```
tc_data$apt.entry(i).ready_list_frwd
```

For the last process in the ready list, this entry is zero. In a similar way, the entry

```
tc_data$apt.entry(i).ready_list_bkwd
```

points to the previous process in the ready list, and

```
tc_data$apt.ready_list_tail
```

points to the last process in the ready list. For the first process in the ready list, the backward pointer is zero.

As a special case, an empty ready list is indicated by both ready_list_head and ready_list_tail being zero.

Example

To put the process whose index is i at the head of a non-empty ready list, the scheduler executes code equivalent to the following:

```
tc_data$apt.entry(i).ready_list_frwd =
  tc_data$apt.ready_list_head;
tc_data$apt.entry(i).ready_list_bkwd = 0;
tc_data$apt.entry(tc_data$apt.ready_list_head)
  .ready_list_bkwd = i;
tc_data$apt.ready_list_head = i;
```

Discussion

It should be clear that both threads are needed, even though the example just shown does not need them. For example, the removal of a process from the ready list in the quit module requires both pointers to permit patching everything together properly.

Interlocking the Ready List

The ready list is a system-wide data base, accessible to all processes and to all processors. Only chaos could result if more than one process at a time were trying to change its various pointers, so it is necessary to adapt an interlocking strategy to prevent such an occurrence. One of the items in the APT is ready_list_lock, a bit string long enough to hold a process id. If this lock is zero, the ready list is available for use; otherwise the lock is set to the process identification of the process using the ready list. For a complete discussion of interlocks in the process exchange, see BJ.6.