Published: 10/01/68
(Supersedes: BJ.5.01, 03/24/67;
BJ.5.01, 09/13/66)

Identification

The Scheduler
Robert L. Rappaport, Michael J. Spier, A. Evans

Purpose

This section describes the multi-level scheduling algorithm which is used by the Traffic Controller in order to multiplex the available processor time among ready processes.

Background

The reader is referred to the paper "An Experimental Time Sharing System" by F. J. Corbató, M. Merwin-Daggett and R. C. Daley which was presented at the 1962 Spring Joint Computer Conference and which describes the F. J. Corbató multi-level scheduling algorithm.

Introduction

The Traffic Controller maintains in the Active Process Table (APT) a threaded list of all ready processes, named the ready-list; it is a linear thread which goes through the APT and which is broken into subthreads, or queues. Typical of the ready-list (see BJ.1.02) is that it can be accessed sequentially as a whole (when selecting the next process to run), or partially by means of direct access to a given queue (when putting a process on the ready list). The number of queues in the ready list is fixed at system initialization time; queues are numbered from 1 (highest-priority) to q (number of queues, lowest-priority) and are accessed directly by their number.

Associated with every process in the system is a level number l which corresponds to the ready list queue into which this process' APT entry may currently be threaded.

Also associated with each process (and kept in the process' Definition Segment, PDS) are two level numbers named `lowest-level' and `highest-level' which delimit the lowest-priority and the highest-priority level numbers which the process can assume, respectively.

The Traffic Controller maintains in the Traffic Controller Data Block (TCDB) the following per-system items which are used in conjunction with the scheduler (see BJ.1.01):

time_quantum A process´ time allotment is a multiple of this value; in addition, the time-quantum corresponds to the span of uninterrupted execution during which a process is immune to pre-emption (excepting system pre-emption, see BJ.5.02).

level_coefficient This is an array of constants, set during system initialization time, where every entry corresponds to a level number and can be accessed by using the level number as an index into the array.

The value of level_coefficient(i) is 2**i

The scheduling algorithm

The scheduling algorithm always decrements the process´ priority in the ready-list by one (increments the current level number) making sure that the level number stays within the boundaries of lowest- and highest-level. When a process is engaged in dialogue with a human being, it is said to be `interacting´. In order to provide fast system response to human request, an interacting process indicates its state to the scheduler by calling it with the argument `interaction_switch´ set to `on´. The scheduler assigns it its `highest_priority´ level number, thus making it a prime candidate in the multi-process race for a processor. However, the interaction is expected to be brief, and the time-allotment associated with this level is small. When the time-allotment runs out, the process is denoted to the next-lowest level and its time-allotment is increased correspondingly. This goes on until the process stabilizes itself in its `lowest-priority´ level where it is assumed to engage in non-interacting computations and where it is given its largest time allotment. The algorithm for computing a time allotment as a function of time-quantum and level-number is:

time-allotment = time-quantum * 2**level

Following is the implemented scheduler, in PL/I language; symbols have been used above and are self-explanatory:

```
if level < highest_level then level=highest_level-1;
if interaction_switch then    level=highest_level-1;
level = level+1;
if level < lowest_level then  level=lowest_level;
time_allotment=time_quantum*level_coefficient(level);
```

The scheduler is invoked as follows:

```
      call scheduler(interaction_switch)
```

where interaction_switch is set to `on´ whenever the scheduler is called in behalf of an interacting process via subroutine block, and `off´ when the scheduler is called by the timer-runout interrupt handler.