there are no accessible programs, or if the worst program in core is as good as the best accessible program, then

(1) if the best program in core is not hung, it will be run;

(2) if the best program in core is hung, the swapper will wait for a change in the status or location of some program. Whenever an interrupt routine recognizes a change in a user program's queue level, status, or location, it notifies the Swapper, which then begins a new evaluation.

If the Swapper has given control to a program in core, its job is obviously completed. If, on the other hand, the Swapper initiates a swap, the Swapper routine is immediately restarted from the beginning for another evaluation. In this case, the set of accessible programs is smaller (because of the set of programs made inaccessible by the initiation of the swap) so that a series of evaluations will always result in either running a program located in a user core or waiting for a change in some program's status or location (e.g., the completion of a swap).

In addition to the organizational philosophy described above, a number of special techniques were adopted to improve the performance of the system. Some are merely coding tricks used to save core space or processing time; others are modifications of the philosophy to handle specific problems. The rest of this Section describes the four most important of these techniques.

A. Queue Counter

The rules for finding a "best" program are, first, to find all