Draft for approval
Published: 5/27/66

Identification

Locking and Blocking in the Basic File System
C.A.Cushing; M.R.Thompson

Purpose

Most of the file system procedures rely on shared data bases. In order to avoid confusion, a procedure must be able to use a stable (i.e., one that no other process is currently modifying) copy of the data base. Since only one copy of a common data base exists, it is necessary that one process be able to prohibit all other processes from using the data base (i.e., lock the base) when this process is modifying it. It is also necessary that a process which wishes to read a stable copy can be sure that no other process is currently modifying the base. The method and details of the locking and checking procedures are discussed in section BG.18.02.

When a process finds that it is unable to use a data base in the manner that it desires, it normally wants to go blocked and wait until the data base is free. The freeing of the data base is called an event for which the process is waiting. When such an event occurs, the process that knows of it must initiate a procedure that will notify all the processes waiting for the event and cause them to be re-scheduled. The waiting and notifying procedures are described in BG.18.01.

Processor Masking

Once a data base has been locked for one process, no other process can use that data until the first process is finished with it and unlocks the data base. Thus, a process may wish to insure that it will not lose the processor for any indefinite period during the time that it has a data base locked. A privileged utility masking routine is available (see Section BK.1.04)for this process that will set the processor mask to a new value and return the previous value of the mask. This mask can ave one of two values; if it is ON, any interrupt that causes the processor to be given to another process for an indefinite period of time is masked (for example the timer-runout interrupt), if the mask is OFF, no interrupts are masked. The processor should be masked for only a short period of time (less than 1 millesecond). The call to the masking routine is made from the procedure that calls the locking routine rather than from the locking routine itself.

In accordance with the general Multics policy that masking should be kept to a minimum, the permission to use the masking routine must be granted by a system administrator. Thus, the processor will be masked only when a process has a data base locked that is essential to efficient system operation and when the duration of the masking time can be shown to be less than one millisecond. A data base is essential to efficient system operation if many processes must use this data base whenever they are running. An example of an essential data base might be the core map. If a process had the core map locked and lost the processor for an indefinite length of time, it is possible that most of the other processes currently running on the system would become blocked waiting for the core map to become available. Eventually, of course, the original process will reach the top of the scheduling queue and run again, this time finally unlocking the core map. However, meanwhile quite a bit of time has been spent blocking and rescheduling other processes. This time would have been saved if the processor had been masked while the core map was locked.