

Analysis of Aging techniques

As a solution to Starvation; a phenomenon associated with Priority based scheduling algorithms; with the help of WIN32 API

Priority scheduling is a commonly used scheduling algorithm where each process is assigned a priority and the processes are executed in increasing order of their priority inside the Ready-Queue, which can be implemented by means of a Priority Queue data structure. Priority is assigned on the base of time and space requirements as well as any other resource requirements that may exist. When two processes of equal priority exist they are discriminated on the basis of arrival time, giving preference to the one who arrived first, just like in a normal Queue.

Traditional Priority scheduling algorithms define fixed priority values for each of its process, and can often be found arising in situations that require the execution of an urgent/ important process over some other processes that are considered to be less important at the moment. Though this approach is reasonable, simple and often found suitable for certain applications, a major problem that occurs with priority scheduling is indefinite blocking or starvation.

Starvation is a phenomenon, wherein a process is denied access to required resources or even CPU execution for an undesirable range of time. A process that is ready to run but waiting for the CPU can be considered blocked. A priority scheduling algorithm can leave some low-priority processes waiting indefinitely.

*“Rumor has it that, when they shut down the IBM 7094 at MIT in 1973, they found a low-priority process that had been submitted in 1967 and had not yet been run.”*

Aging is an OS related technique applied during process scheduling as a solution to starvation as well as indefinite blocking that may occur along with starvation. Aging involves gradual dynamic increases in the priorities of processes which are waiting in the Ready-Queue.

Though there are many different approaches for executing aging techniques, they are all based on the same principle of Gradual Dynamic Updating of priorities for the processes that are waiting in the Ready-Queue in accordance with their arrival time. The main objective behind dynamic updating of priorities is to ensure that, processes which have been initialised with a lower priority value can reach the CPU for execution with minimum waiting time.

Aging:

for(pp = proc; pp < proc+NPROC; pp++)

{

if (pp->prio != MAX)

pp->prio++;

if (pp->prio > curproc->prio)

reschedule();

}

[This code is taken almost verbatim from 6th Edition Unix, circa.]

NOTE:

Here, a higher integer value represents higher priority

Problem:

O(n). Every process is examined on each schedule() call

OUTPUT:

















