

Real-time Scheduling

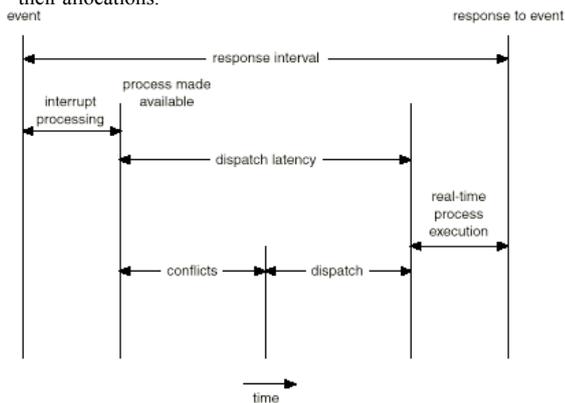
Hard real-time systems – required to complete a critical task within a guaranteed amount of time.

Soft real-time computing – requires that critical processes receive priority over less *important* ones.

When processes are submitted they indicate their CPU requirements.

The scheduler may reject the process if the requirement cannot be met.

But very important processes can force other processes to relinquish their allocations.



Real-time scheduling

Periodic and Sporadic processes

Periodic

activate regularly between fixed time intervals
used for polling, monitoring and sampling
predetermined amount of work every period

Sporadic

event driven – some external signal or change
used for fault detection, change of operating modes

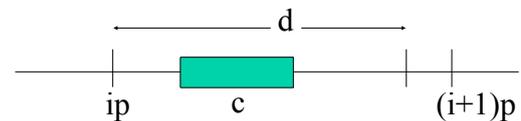
(c, p, d)

c – computation time (worst case)

p – period time

d – deadline

$$c \leq d \leq p$$



Periodic processes

Period and Deadline are determined by the system requirements (often the same).

Computation time is found through analysis, measurement or simulation.

When the computation is complete the process is blocked until the next period starts.

Sometimes it doesn't matter if the deadline extends beyond the period or the period can change depending on system load.

Sporadic processes

(c, p, d) still applies

c and d have the obvious meaning

p is the minimum time between events

aperiodic processes

p = 0

events can happen at any time, even simultaneously

timing can no longer be deterministic but there are ways of handling this

- statistical methods, we design to satisfy average response times
- if it is rare that the system has timing faults then special cases can be included in the handling code

Cyclic executives (CEs)

Handles periodic processes.

Sporadic processes can be converted into equivalent periodic processes or they can be ignored (if they take only a little time to handle).

Pre-scheduled – a feasible execution schedule is calculated before run time.

The cyclic executive carries out this schedule.

It is periodic.

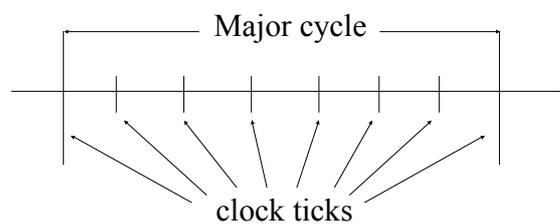
Highly predictable – non-preemptible

Inflexible, difficult to maintain.

CE schedule

Major schedule – cyclic allocation of processor blocks which satisfies all deadlines and periods.

Minor cycle (or frame) – major schedules are divided into equal size frames. Clock ticks only occur on the frame boundaries.



CE example

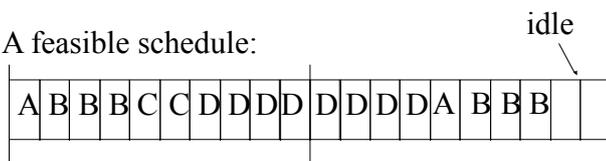
Periodic processes:

$A = (1, 10, 10)$, $B = (3, 10, 10)$, $C = (2, 20, 20)$
 $D = (8, 20, 20)$, D is divided into two sections of 4 units.

Major cycle time is 20 (smallest possible value we can use in this case). LCM of periods.

Frame time – can be 10, GCD of periods.

A feasible schedule:



Scheduling with priorities

Scheduling decisions are made:

- when a process becomes ready
- when a process completes its execution
- when there is an interrupt

Priorities can cause schedules to not be feasible.

$A = (1, 2, 2)$ better priority

$B = (2, 5, 5)$ worse priority

This is feasible (without preemption), but if the priorities are reversed it is not.

Still priorities are almost always used

- fixed – determined before execution
- dynamic – change during execution

Priority allocation

Fixed

Rate monotonic (RM) – the shorter the period the higher the priority.

Least compute time (LCT) – the shorter the execution time the higher the priority (shortest job first)

Dynamic

Shortest completion time (SCT) – shortest job first with preemption. But this time we have *good* information on the execution time requirement.

Earliest deadline first (EDF) – the process with the closest deadline has the highest priority.

Least slack time (LST) – the process with the least slack time has the highest priority.

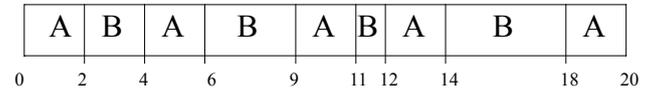
Slack time is the amount of time to the process's deadline minus the amount of time the process still needs to complete.

Calculating schedules

Calculate a schedule for the following two processes using EDF and SCT.

$A = (2, 4, 4)$ $B = (5, 10, 10)$

EDF



SCT

Same as above

What about LST?

Same as above until time 17.

Theory

For static priorities

RM is an optimal scheduling policy

If the CPU usage is $< \ln 2 \approx 0.69$ RM will always find a schedule.

For dynamic priorities

EDF and LST are optimal

But these are only true for single processors.

“The most practical policy for multiprocessors is to pre-assign processes to CPUs using some heuristic, and then to schedule each one independently.”

Also theory assumes complete knowledge – non-preemptible resources, precedence constraints, interrupt and context switching times all need to be taken into account (see the diagram on slide 1).

Before next time

Read from the textbook

5.1 Background

5.2 Critical-Section Problem

5.3 Peterson's Solution

5.4 Synchronization Hardware

5.5 Mutex Locks

5.6 Semaphores

5.7.1 The Bounded-Buffer Problem

Reference

Real-Time Systems and Software, Alan C. Shaw, Wiley