Deadlock Prevention: Wait/Wound/Die Algorithms

The algorithms below, by Rosenkrantz (1978),
prevent the *No Preemption* condition.

Assign each process a global timestamp when it starts.
No two processes should have same timestamp.
Basic idea: "When one process is about to block waiting for a resource that another process is using, a check is made to see which has a larger timestamp (i.e. is younger)."
(Tanen95)

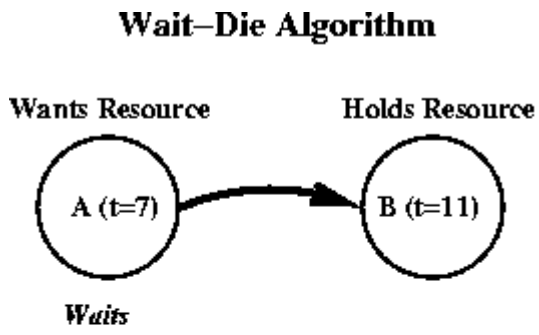
Somehow put timestamps on each process,
providing the creation time of each process.
Suppose a process needs a resource
already owned by another process.

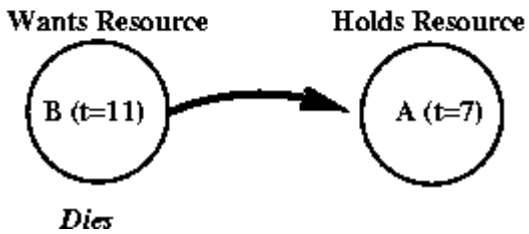
- Determine relative ages of both processes.
- Decide if waiting process should
  - ***Preempt***,
  - ***Wait***,
  - ***Die***, or
  - ***Wound***

  the process that owns the resource.

The two different algorithms by Rosenkrantz (1978)
use these ideas, as explained below:

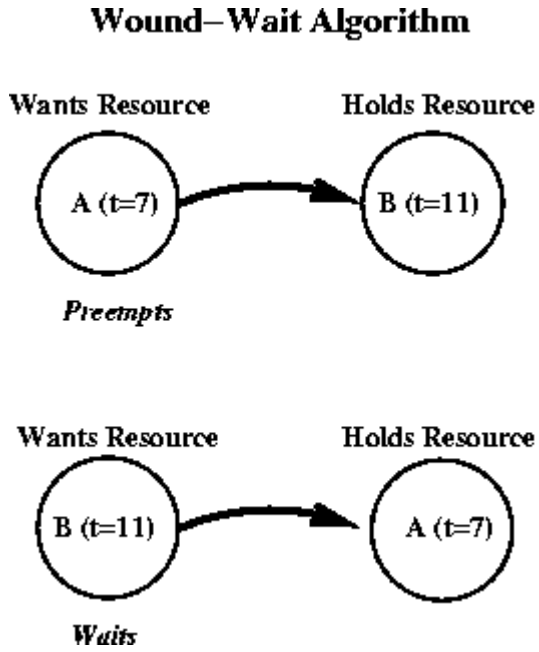
1. The ***Wait-Die*** algorithm:
   Allow wait only if waiting process is older.
   Since timestamps increase in any chain of waiting processes,
   cycles are impossible.





The Wait-Die algorithm kills the younger process.
When the younger process restarts
and requests the resource again,
it may be killed once more.
This is the less efficient of these two algorithms.

2. The ***Wound-Wait*** algorithm:
Otherwise allow wait only if waiting process is younger.
Here timestamps decrease in any chain of waiting process,
so cycles are again impossible.
It is wiser to give older processes priority.



The Wound-Wait algorithm preempts the younger process.
When the younger process re-requests resource,
it has to wait for older process to finish.
This is the better of the two algorithms.

Note: To avoid *starvation*,
a process should **not** be assigned
a new timestamp each time it restarts.

Comments: