Esercizio di Sincronizzazione tra Processi: Il Problema dei Lettori/Scrittori

Consideriamo le strutture dei processi:

LETTORI SCRITTORI

```
type lettori= process;          type scrittori= process;
begin                           begin
repeat                          repeat
    lettscritt.LRICH;               lettscritt.SRICH;
       <leggi>                      < scrivi >
    lettscritt.LRIL;                lettscritt.SRIL;
until false;                    until false;
end;                            end;
```

Usiamo due code separate, una per i processi lettori in attesa di accedere alla risorsa (codalett) ed una per i processi scrittori (codascritt); la risorsa stessa ha il proprio stato in un booleano occupato; e' inoltre necessario un contatore del numero di lettori in coda: nlettori.

1° Versione

```
type rw= monitor
var codalett, codascritt: condition;
   occupato : boolean; nlettori: integer;

procedure entry LRICH;
begin     if ( occupato and nlettori = 0 ) then
            {  un lettore deve sospendersi se la risorsa e' occupata da uno scrittore }
            codalett.wait;
        occupato := true; nlettori := nlettori + 1;
        { segnalazione dei prossimi lettori in coda, se questi non sono tutti segnalati dal processo
                        scrittore che rilascia }
        codalett.signal
end;
procedure entry LRIL ;
begin     nlettori:= nlettori - 1;
        if nlettori = 0 then begin occupato := false;
                             codascritt. signal;
                        end;
end;
procedure entry SRICH;
begin     if occupato then { uno scrittore si sospende se la risorsa e' occupata }
            codascritt. wait;
         occupato := true;
end;
```

```
procedure entry SRIL;
begin     if codascritt. queue then codascritt. signal
              { se ci sono altri scrittori in coda, segnalali }
          else
              if codalett.queue then { ci sono lettori, segnalali }
                  codalett.signal
                              { se si vuole invece far segnalare dallo scrittore tutti i lettori, e'
                              necessarioeseguire
          invece della signal singola:
                      while codalett.queue do codalett.signal  }

              else occupato := false;
end;
```

L'applicazione parallela e' costituita da:

```
 program LettorieScrittori;

    { le dichiarazioni di tipo viste inserite qui }
     var lettscritt      :  rw;
        s1, ...        :  scrittore; {assieme ad altri processi}
        l1, ...         :  lettore  ;  { con altri }
 begin end.
```

2° Versione

Consideriamo adesso le modalita' per evitare starvation:

- si deve rendere impossibile l'acquisizione senza limite dei processi lettori, anche se la risorsa e' in possesso di un certo numero di lettori, se c'e' almeno uno scrittore in attesa

- si deve impedire agli scrittori di passarsi la risorsa tra loro, non considerando richieste di lettori gia' accodate.

```
type rw= monitor
var codalett, codascritt: condition;
   occupato : boolean; nlettori: integer;

procedure entry LRICH;
begin     if ( occupato and nlettori = 0 ) OR ( CODASCRITT.QUEUE)
          then
            {un lettore deve sospendersi se la risorsa e' occupata da uno scrittore, ma anche se c'e'
                        almeno uno scrittore in coda }
            codalett.wait;
         occupata := true; nlettori := nlettori + 1;
         codalett.signal
end;
procedure entry LRIL ;
begin     nlettori:= nlettori - 1;
         if nlettori = 0 then begin occupato := false;
                              codascritt. signal,
                         end;
end;

procedure entry SRICH;
begin     if occupato then { uno scrittore si sospende se la        risorsae' occupata }
            codascritt. wait;
          occupata := true;
end;

procedure entry SRIL;
begin     if codalett. queue then codalett. signal
            { se ci sono lettori in coda, segnala il primo}
         else
            if codascritt. queue then
                { c'e' uno scrittore, segnalalo }
                codascritt.signal
            else occupato := false;
end;
```