SISTEMI OPERATIVI

ESERCIZIO N. 1 del 4 SETTEMBRE 2002

Un **database** permette di effettuare due tipi di **richieste**: **normali** e di **amministrazione**. Le seconde hanno priorità sulle prime, e possono essere eseguite solo se non ci sono altre richieste (di qualsiasi tipo) in esecuzione. Le richieste normali possono invece eseguire contemporaneamente. Quando arriva una richiesta normale, deve dare la precedenza alle eventuali richieste di amministrazione che sono in attesa.
Si implementi una soluzione usando il costrutto monitor per modellare il **database** e i processi per modellare le **richieste** e si descriva la sincronizzazione tra i processi. Nella soluzione si massimizzi l'utilizzo delle risorse. Si discuta se la soluzione proposta può presentare starvation e in caso positivo per quali processi, e si propongano modifiche e/o aggiunte per evitare starvation.

```
program Database

type      tipo = (normale, amm);  { tipo di richiesta }

type richiesta = process (t: tipo)
begin
      repeat
            d.richiedi (t);
            <esegui la richiesta >
            d.rilascia (t);
      until false
end

type dbms = monitor

{ variabili del monitor }
var  running : array[tipo] of integer;
      { numero delle richieste in esecuzione }
      coda : array[tipo] of condition;
      { code su cui sospendere le richieste }

procedure entry richiedi (t: tipo)
begin
      { se è normale e c'è una amm in esecuzione o in coda }
      if t = normale and
            (running[am] > 0 or coda[amm].queue) then
            { sospensione }
            coda[t].wait;
      { se è amm e c'è una richiesta in esecuzione }
      if t = amm and
            (running[amm] > 0 or running[normale] > 0) then
            { sospensione }
            coda[t].wait;
      { occupo la risorsa }
      running[t]++;
end
```

```
procedure entry rilascia (t: tipo)
begin
      { rilascio le risorse }
      running [t] --;
      { se è amm }
      if t = amm
      begin
            { dà la precedenza ad una amm }
            if coda[amm].queue then
                  coda[amm].signal;
            else
                  while coda[normale].queue do
                        coda[normale].signal;
      end
      else { è normale }
      begin
            { se non ci sono più normali in esecuzione }
            if running [normale] = 0 then
                  { risveglia una amm (se c'è) }
                  coda[amm].signal;
      end
end

begin { inizializzazione delle variabili }
      running[normale] := 0;
      running[amm] := 0;
end

var   d: dbms; { il nostro monitor }
      n1, n2, ... : richiesta (normale);
      a1, a2, ... : richiesta (amm);

begin end.
```

Starvation

La soluzione proposta presenta starvation perché le richieste di amministrazione possono ritardare l'esecuzione delle richieste normali in modo indefinito.

Si può ridurre il ritardo imponendo un contatore per le richieste amministrative, alternando la priorità ogni tot di richieste amministrative eseguite.