SISTEMI OPERATIVI

ESERCIZIO N. 1 del 5 aprile 2001

Una **centrale elettrica** produce una quantità **Q** di energia elettrica al secondo. Ogni utente fa richiesta di una determinata quantità di corrente al secondo (minore di Q) e svolge il suo lavoro solo quando tale quantità gli viene fornita; terminato il lavoro, annulla la richiesta. Gli utenti si dividono in **normali** e **urgenti**: i secondi, che hanno priorità sui primi, sono utenti che gestiscono strutture critiche tipo ospedali. Inoltre, periodicamente un **tecnico** svolge un lavoro di manutenzione che richiede che nessun utente stia usufruendo della corrente.
Si implementi una soluzione usando il costrutto monitor per modellare la **centrale elettrica** e i processi per modellare gli **utenti** e il **tecnico** e si descriva la sincronizzazione tra i processi. Nella soluzione si massimizzi l'utilizzo della risorsa. 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 CentraleElettrica

const     Q = ...; { quantità massima di corrente }
type      tipo = (normale, urgente);  { tipo di utente }
type      quantità = 1..Q;  { quantità di corrente }

type utente = process (t: tipo, q: quantità)
begin
      repeat
            centrale.richiedi (t, q);
            < effettua il lavoro >
            centrale.rilascia (n);
      until false
end

type tecnico = process
begin
      repeat
            centrale.inizio_man;
            < effettua la manutenzione >
            centrale.fine_man;
            < altri incarichi >
      until false
end
```

```
type centrale_elettrica = monitor

{ variabili del monitor }
var  manutenzione : boolean;
     { manutenzione in corso? }
     Qdisponibile : integer;
     { corrente disponibile }
     n_utenti : integer;
     { numero di utenti }
     coda_tecnico : condition;
     { coda su cui sospendere il tecnico }
     coda : array[tipo] of condition;
     { code su cui sospendere gli utenti }
     in_coda : array[tipo] of integer;
     { numero di utenti sospesi per tipo }

procedure entry richiedi (t: tipo, q: quantità)
begin
     { se non c'è abbastanza corrente}
     while Qdisponibile < q
       or manutenzione { o c'è la manutenzione }
       { o ci sono degli utenti urgenti in coda }
       or (t = normale and coda[urgente].queue) do
     begin
          in_coda[t] ++;
          coda[t].wait; { sospensione }
          in_coda[t] --;
     end;
     { occupo le risorse }
     Qdisponibile := Qdisponibile – q;
     n_utenti := n_utenti + 1 ;
end
```

```
procedure entry rilascia (q: quantità)
var i, s : integer;
begin
      { rilascio le risorse }
      Qdisponibile := Qdisponibile + q;
      n_utenti := n_utenti – 1 ;
      if n_utenti = 0 then { se sono l'ultimo utente }
            { risveglio il tecnico (se sta aspettando) }
            coda_tecnico.signal;
      { risveglio tutti gli utenti in attesa }
      { prima quelli urgenti }
      s := in_coda[urgente];
      for  i := 1 to s do
            coda[urgente].signal;
      { poi quelli normali }
      s := in_coda[normale];
      for  i := 1 to s do
            coda[normale].signal;
end

procedure entry inizio_man
begin
      { se ci sono degli utenti }
      if n_utenti <> 0 then
            { il tecnico si sospende }
            coda_tecnico.wait;
      { occupo le risorse }
      manutenzione := true;
end
```

```
procedure entry fine_man
var s: integer;
begin
        { rilascio le risorse }
        manutenzione := false;
        { risveglio tutti gli utenti in attesa }
        { prima quelli urgenti }
        s := in_coda[urgente];
        for  i := 1 to s do
                coda[urgente].signal;
        { poi quelli normali }
        s := in_coda[normale];
        for  i := 1 to s do
                coda[normale].signal;
end

begin { inizializzazione delle variabili }
        n_utenti := 0;
        manutenzione := false;
        Qdisponibile := Q
end

var  centrale: centrale_elettrica; { il nostro monitor }
        un1, un2, ... : utente (normale, k);
        uu1, uu2, ... : utente (urgente, j);
        t: tecnico;

begin end.
```

Starvation

La soluzione proposta presenta starvation in due casi.
Il primo caso si verifica quando gli utenti normali vengono sempre superati da quelli urgenti. Si può risolvere introducendo un contatore che, dopo un numero prefissato di utenti urgenti, dia la precedenza agli utenti normali.
Il secondo caso si ha quando gli utenti non lasciano mai entrare il tecnico. Si può risolvere introducendo un contatore come nel caso precedente, oppure evitando di fare entrare utenti se il tecnico è in attesa.

Considerazioni

In **rilascia** potrei risvegliare gli utenti solo se il tecnico non ha iniziato la manutenzione, per evitare di risvegliare processi che comunque si risospendono.
Allo stesso modo, in **rilascia** e in **fine_man**, potrei risvegliare gli utenti normali solo se non ci sono utenti urgenti in coda, per evitare di risvegliare processi che comunque si risospendono.
Il numero di utenti deve essere mantenuto da un contatore, perché bisogna sapere chi è l'ultimo che lascia la centrale, mentre per il tecnico è sufficiente un booleano perché è uno solo.