PRINCIPI DI SISTEMI OPERATIVI

ESERCIZIO del 28 NOVEMBRE 2003

Un **distributore di benzina** ha a disposizione **P pompe** e una cisterna da **L litri**. Le automobili arrivano al distributore e richiedono un certo numero di litri di benzina (diverso per ogni automobile e minore di L). La cisterna è rifornita da una autobotte che la riempie fino alla capacità massima e solo se nessuna automobile sta facendo benzina, avendo comunque la priorità sulle automobili. La automobili possono fare benzina solo se c'è una pompa libera, se la quantità di benzina richiesta è disponibile nella cisterna e se l'autobotte non sta riempiendo la cisterna. Dopo aver fatto benzina, liberano la pompa utilizzata.
Si implementi una soluzione usando il costrutto monitor per modellare il **distributore di benzina** e i processi per modellare le **automobili** e **l'autobotte** e si descriva la sincronizzazione tra i processi. Nel rispettare i vincoli richiesti, si cerchi di massimizzare 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 DistributoreBenzina

const     P = ...; { numero di pompe }
const     L = ...; { capacità della cisterna }

type automobile = process (l: 1..L)
begin
      repeat
            d.richiedi (l);
            <fai benzina >
            d.rilascia;
      until false
end

type autobotte = process
begin
      repeat
            d.rifornisci;
            < vai a prendere altra benzina >
      until false
end

type distributore = monitor

{ variabili del monitor }
var  benzdisp: 0..L;
      { benzina disponibile }
      pompedisp: 0..P;
      { pompe disponibili }
      codaAM : condition;
      { coda su cui sospendere le automobili }
      codaAB : condition;
      { coda su cui sospendere l'autobotte }
      sospesi : integer;
      { numero di automobili sospese }
```

```
procedure entry richiedi (l: 1..L)
begin
{ se non ci sono pompe disponibili, o se non c'è sufficiente
benzina, o se l'autobotte è in attesa }
      while (pompedisp = 0) or (benzdisp < l)
                  or codaAB.queue do
      begin
            { sospensione }
            sospesi ++;
            codaAM.wait;
            sospesi --;
      end
      { acquisizione delle risorse }
      pompedisp--;
      benzdisp := benzdisp - l;
end

procedure entry rilascia
begin
      { rilascio delle risorse }
      pompedisp ++ ;
      { se non ci sono automobili che fanno benzina }
      if pompedisp = P then
            { risveglio l'autobotte }
            codaAB.signal;
      { risveglio tutte le automobili in coda }
      s := sospesi;
      for i := 0 to s do
            codaAM.signal;
end
```

```
procedure entry rifornisci
begin
{ se ci sono macchine che stanno facendo benzina }
if pompedisp < P then
      { sospensione }
      codaAB.wait;
      { riempie la cisterna }
      benzdisp := L;
      { risveglio tutte le automobili in coda }
      s := sospesi;
      for i := 0 to s do
            codaAM.signal;
end

begin { inizializzazione delle variabili }
      benzdisp := L;
      pompedisp := P;
end

var  d: distributore; { il nostro monitor }
      a1, a2, ... : automobile (j);
      ab: autobotte;

begin end.
```

Starvation

Nella soluzione proposta può esserci starvation per le automobili che hanno bisogno di una grande quantità di benzina.
Si può risolvere o utilizzando code con diversa priorità o code differenziate per quantità di benzina richiesta.

Nota

In un caso reale, la cisterna contiene molti più litri di quanto possano chiedere le automobili, quindi la starvation è praticamente impossibile.

VERSIONE CON DUE PROCEDURE PER L'AUTOBOTTE

```
program DistributoreBenzina

const     P = ...; { numero di pompe }
const     L = ...; { capacità della cisterna }

type automobile = process (l: 1..L)
begin
      repeat
            d.richiedi (l);
            <fai benzina >
            d.rilascia;
      until false
end

type autobotte = process
begin
      repeat
            d.entra;
            <riempi la cisterna >
            d.esci;
      until false
end

type distributore = monitor

{ variabili del monitor }
var  benzdisp: 0..L;
      { benzina disponibile }
      pompedisp: 0..P;
      { pompe disponibili }
      riempimento: boolean;
      { l'autobotte sta riempiendo la cisterna }
      codaAM : condition;
      { coda su cui sospendere le automobili }
```

```
        codaAB : condition;
        { coda su cui sospendere l'autobotte }
        sospesi : integer;
        { numero di automobili sospese }

procedure entry richiedi (l: 1..L)
begin
{ se non ci sono pompe disponibili, o se non c'è sufficiente
benzina, o se l'autobotte sta riempiendo la cisterna o è in
attesa }
        while (pompedisp = 0) or (benzdisp < l)
                or riempimento or codaAB.queue do
        begin
                { sospensione }
                sospesi ++;
                codaAM.wait;
                sospesi --;
        end
        { acquisizione delle risorse }
        pompedisp--;
        benzdisp := benzdisp - l;
end

procedure entry rilascia
begin
        { rilascio delle risorse }
        pompedisp ++ ;
        { se non ci sono automobili che fanno benzina }
        if pompedisp = P then
                { risveglio l'autobotte }
                codaAB.signal;
        { risveglio tutte le automobili in coda }
        s := sospesi;
        for i := 0 to s do
                codaAM.signal;
end
```

```
procedure entry entra
begin
{ se ci sono macchine che stanno facendo benzina }
if pompedisp < P then
        { sospensione }
        codaAB.wait;
        { acquisizione delle risorse }
        riempimento = true;
end

procedure entry esci
var s, i: integer;
begin
        { rilascio delle risorse }
        riempimento := false ;
        { riempie la cisterna }
        benzdisp := L;
        { risveglio tutte le automobili in coda }
        s := sospesi;
        for i := 0 to s do
                codaAM.signal;
end

begin { inizializzazione delle variabili }
        benzdisp := L;
        pompedisp := P;
        riempimento = false;
end

var  d: distributore; { il nostro monitor }
        a1, a2, ... : automobile (j);
        ab: autobotte;

begin end.
```