Esercizio E4.3

Impostazione

Per consentire la realizzazione della priorità specificata nel testo, il `server` offre due entry con le quali ogni cliente specifica il tipo della propria richiesta: il cliente invoca la entry `ric1()` per chiedere al `server` una qualunque delle tre risorse gestite o la entry `ric2()` se desidera avere due risorse contemporaneamente. Subito dopo aver chiamato o la entry `ric1` o la `ric2`, il clente invoca una ulteriore entry del server: la entry `risorsa(out int r)` nel primo caso o la entry `risorse(in int r1, in int r2)` nell'altro. Il `server` dopo aver accettato la `ric1` o la `ric2` verifica se la, o le risorse, richieste sono disponibili e in quelsto caso accetta immediatamente la seconda entry chiamata dal cliente restituendo tramite i parametri `r` oppure `r1` e `r2`, gli indici delle risorse allocate. Se, viceversa, le risorse non sono disponibili il `server` non accetta la seconda chiamata e il cliente resta sospeso.
All'atto di un rilascio, che il cliente specifica invocando una delle entry `ril1(in int r)` o `ril2(in int r1, in int r2)` rispettivamente, il `server` verifica se con le risorse rilasciate è possibile svegliare, se ce ne sono, uno o più dei processi clienti in attesa secondo l'ordine di priorità indicato nel testo.

Per richiedere una risorsa, un processo cliente segue il seguente schema (dove `ris` denota una variabile intera):

```
server.ric1();
server.risorsa(ris);
    <uso della risorsa di indice ris >
server.ril1(ris);
```

Per richiedere due risorse contemporaneamente, un processo cliente segue il seguente schema (dove `ris1` e `ris2` denotno variabili intere):

```
server.ric2();
server.risorse(ris1, ris2);
    <uso delle due risorse di indice ris1 e ris2>
sever.rilascio2(ris1, ris2);
```

Soluzione

```
process server {
  entry ric1(), ric2();
  entry risorsa(out int r), risorse(out int r1, out int r2);
  entry ril1(in int r), ril2(in int r1, in int r2);

  boolean libera[3]= {true, true, true}; /*indica lo stato di allocazione delle tre risorse*/
  int disponibili=3: /*contatore  del numero di riserse disponibili*/
  int bloccati1=0, bloccati2=0: /*contatori  che indicano il numero di clienti sospesi in attesa,
                                  ripettivamente, di ottenere una risorsa o due risorse contemporaneamente*/
  int i, j, k;

  do
     [] accept ric1(){} ->
          if(disponibili>0) { /*se ci sono risorse libere*/
             j=0;
             while(!libera[j]) j++ /*viene cercata una delle risorse libere*/
             accept risorsa(out int r) {r=j;}
             libera[j]=false;
             disponibili--;
          }
```

```
     else bloccati1++; /*nessuna risorsa è libera e il processo cliente resta sospeso*/
[] accept ric2(){} ->
     if(disponibili>1) { /*se ci sono almeno due risorse libere*/
        j=0;
        while(!libera[j]) j++
        libera[j]=false;
        k=j+1;
        while(!libera[k]) k++
        libera[k]=false;
        accept risorse(out int r1, out int r2){r1=j;r2=k;}
        disponibili=disponibili-2;
     }
     else bloccati2++; /*le risorse libere non sono sufficienti: il processo resta sospeso*/
[] accept ril1(in int r){j=r;} ->
        if(bloccat1>0) { /*la risorsa di indice r viene liberata. Se ci sono processi
                           sospesi in attesa di una risorsa, uno di questi può essere riattivato*/
              bloccati1--;
              accept risorsa(out int r) {r=j;}
        }
        else if(disponibili>0&&bloccati2>0){ altrimenti se è disponibile una
                           ulteriore risorsa e ci sono processi sospesi in attesa di due risorse, uno di
                           questi può essere riattivato*/
                k=0;
                while(!libera[k]) k++
                libera[k]=false;
                disponibili--;
                accept risorse(out int r1, out int r2){r1=j;r2=k;}
             }
             else { /*in caso contrario la risorsa viene resa disponibile*/
                libera[j]=true;
                dispobnibili++;
             }
[] accept ril2(in int r1, in int r2){j=r1; k=r2;} ->
     if(bloccati1>0) {  /* se c'è almeno un processo sospeso in attesa di una
                           risorsa, gli può essere assegnata una delle due risorse riasciate*/
          bloccati1--;
          accept risorsa(out int r) {r=j;}
          if(bloccati1>0) { /*e se ci sono altri processi sospesi in attesa di una risorsa
                           un altro processo può essere riattivato assegnandogli l'altra risorsa */
              bloccati1--;
              accept risorsa(out int r) {r=k;}
          }
          else libera[k]=true;/*altrimenti la risorsa di indice k viene resa disponibile*/
          disponibili++;
     }
     else if(bloccati2>0) {  /*altrimenti,  se ci sono processi sospesi in attesa di due risorse,
                           uno di questi può essere riattivato allocandogli le due risorse*/
              bloccati2--;
              accept risorse(out int r1, out int r2){r1=j; r2=k;}
          }
          else { /*se non ci sono processi sospesi le risorse rilasciate vengono rese disponibili */
```

```
                libera[j]=true;
                libera[k]=true;
                disponibili=disponibili+2;
            }
  od
 }
```