Esercizio E4.1

Impostazione

Indichiamo con `R0` ed `R1` le due risorse gestite dal `server`. Sono previste due entry offerte dal `server` per consentire ai processi clienti di richiedre, rispettivamente, una risorsa o due risorse contemporaneamente (`richiesta1` e `richiesta2`). Dovendo implementare la strategia di allocazione che privilegia, all'atto di un rilascio, le richieste di due risorse rispetto alle richieste singole, le corrispondenti chiamate delle entry `richiesta1` e `richiesta2` vengono sempre accettate dal `server` mentre un processo cliente che ha effettuato una di tali chiamate si pone immediatamente in attesa invocando altre entry del `server` (la entry `risorsa` per le richieste singole e la entry `risorse` per le altre). Il processo `server` accetterà queste ulteriori chiamate, abilitando quindi il cliente a operare sulla, o sulle, risorse richieste rispettivamente, soltanto quando le corrispondenti condizioni logiche sono verificate. Con la entry `risorsa` il server restituisce anche l'indice (`0` oppure `1`) della particolare risorsa allocata. La entry `risorse`, viceversa, non ha bisogno di parametri poiché il cliente è abilitato ad operare su entrambe le risorse.
Per il rilascio delle risorse il `server` riserva le due entry `rilascio2()` e `rilascio1(in int r)` dove il parametro `r` denota la risorsa rilasciata (`r=0` se `R0` e `r=1` se `R1`).

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

```
server.richiesta1();
server.risorsa(ris);
    <uso della risorsa di indice ris >
sever.rilascio1(ris);
```

Per richiedere le due risorse contemporaneamente, un processo cliente segue il seguente schema:

```
server.richiesta2();
server.risorse();
    <uso delle due risorse>
sever.rilascio2();
```

Soluzione

```
process server {
  entry richiesta1(), richiesta2();
  entry risorsa(out int r), risorse();
  entry rilascio1(in int r), rilascio2();

  boolean libera[2]= {true, true}; /*indica lo stato di allocazione delle due risorse*/
  int bloccati1=0, bloccati2=0: /*contatori  che indicano il numero di clienti sospesi in attesa,
                          ripettivamente, di ottenere una risorsa o le due risorse contemporaneamente*/
  int i;

  do
     [] accept richiesta1(){} ->
         if(libera[0]) {  /*se è libera R0*/
                  libera[0]=false;
                  accept risorsa(out int r) {r=0;}}
         else if(libera[1]) {   /*se è libera R1*/
                  libera[1]=false;
                  accept risorsa(out int r) {r=1;}}
               else bloccati1++; /*nessuna risorsa è libera e il processo cliente resta sospeso*/
     [] accept richiesta2(){} ->
         if(libera[0]&&libera[1]) {  /*se sono libere entrambe le risorse*/
                  libera[0]=false;
                  libera[1]=false;
                  accept risorse(){}}
         else bloccati2++; /*altrimenti il processo cliente resta sospeso*/
     [] accept rilascio1(in int r){i=r];} ->
```

```
            if(libera[(i+1)%2]&&bloccati2>0) { /*la risorsa di indice r viene liberata. Se è libera
                  anche l'altra risorsa e se ci sono processi sospesi in attesa delle due risorse, uno di questi può
                  essere riattivato allocandogli le due risorse*/
                  libera[(i+1)%2]=false;
                  bloccati2--;
                  accept risorse(){}}
            else if(bloccat1>0) { /*altrimenti, se c'è almeno un processo sospeso in attesa di una
                              risorsa, la risorsa rilasciata gli può essere assegnata */
                        bloccati1--;
                        accept risorsa(out int r) {r=i;}}
                  else libera[i]=true; /*in caso contrario la risorsa viene resa disponibile*/
    [] accept rilascio2(){} ->
         if(bloccati2>0) {  /* se ci sono processi sospesi in attesa delle due risorse, uno di questi
                              può essere riattivato allocandogli le due risorse*/
                  bloccati2--;
                  accept risorse(){}}
         else if(bloccati1>0) {  /*altrimenti, se c'è almeno un processo sospeso in attesa di una
                              risorsa, gli può essere assegnata la prima risorsa, quella di indice 0*/
                  bloccati1--;
                  accept risorsa(out int r) {r=0;}
                  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=1;}
                  else libera[1]=true;/*altrimenti la risorsa di indice 1 viene resa disponibile  */
               else { /*se non ci sono processi sospesi in attesa di due risorse né in attesa di una risorsa, le
                              due risorse rilasciate vengono rese disponibili */
                  libera[0]=true;
                  libera[1]=true; }
  od
}
```