# Esercizio E4.4

## Impostazione

Il processo `server` offre ai processi clienti la entry `richiesta(in int i)` dove l'intero `i` rappresenta l'indice del processo chiamante (valore compreso tra zero e `N-1`) Chiamando tale entry il generico cliente specifica il proprio indice al `server`, informazione che serve a quest'ultino per gestire la priorità fra i processi clienti. Infatti, subito dopo aver chiamato la precedente entry `richiesta`, il cliente $i_{esimo}$ chiama una seconda entry del `server`, la $i_{esma}$ componente dell'array di entry `risorsa[N](out int ris)` che restituisce, tramite il parametro `ris`, l'indice della risorsa che verrà allocata quando il `server` deciderà di accettare tale chiamata. Quando il server accetta la chiamata di `richiesta`, se ha risorse disponibili, accetta subito anche la entry `risorsa` corrispondente al processo chiamante; altrimenti il chiamante resta sospeso sulla chiamata di `risorsa`. In questo caso, quando successivamente una risorsa verrà rilasciata, il `server` potrà selezionare, fra i vari processi clienti bloccati, quello di indice più basso, e cioè quello a priorità più alta, svegliandolo mediante l'accettazione della corrispondente chiamate di `risorsa`.
Il `server` fornisce anche la entry `rilascio(in int ris)`che ogni cliente invoca per rilasciare la risorsa precedentemente allocata.

Per richiedere, usare e rilasciare una risorsa, il processo $P_i$ $(0 \leq i < N)$ segue il seguente schema:

```
server.richiesta(i);
server.risorsa[i](ris);
     <uso della risorsa di indice ris>;
server.rilascio(ris)l
```

dove `ris` denota una variabile di tipo `int` tramite la quale il cliente riceve dal server l'indice della risorsa da usare e, successivamente, da rilasciare.

## Soluzione

```
process server {
  entry richiesta(in int i);
  entry risorsa[N](out int ris);
  entry rilascio(in int ris)

  boolean libera[M]; /*indicatori necessari per sapere quale risorsa è libera e quale no*/
  int disponibili=M; /*contatore delle risorse libere */
  boolean bloccato[N]; /*indicatori necessari per sapere quale processo è bloccato e quale no*/
  int sospesi; /*contatore dei processi bloccati */
  int cliente;
  int rilasciata;

  { for(int i=0; i<N; i++) { bloccato[i]=false;}
    for(int j=0; j<M; j++) { libera[j]=true;}
  } /*inizializzazione*/

  do
     [] accept richiesta(in int i){cliente=i;} ->
          if(disponibili >0) { /*ci sono risorse disponibili*/
             accept risorsa[cliente](out int ris){
                   int k=0;
                   while(!libera[k]) k++;
                   ris=k;}
```

```
            libera[k]=false;
            disponibili--;
         else {/*il cliente resta sospeso sulla chiamata di risorsa*/
            sospesi++;
            bloccato[cliente]=true;}
    [] accept rilascio(in int ris){rilasciata=ris;} ->
            if(sospesi>0) { /*ci sono processi sospesi e uno può essere svegliato*/
            int b=0;
            while(!bloccato[b]) b++;
            bloccato[b]=false;
            sospesi--;
            accept risorsa[b](out int ris){ris=riasciata}}
         else { /*la risorsa rilasciata viene resa disponibile */
            libera[rilasciata]=true;
            disponibili++;}
   od
}
```