Esercizio E1.4

Impostazione

Due sono i problemi da risolvere nel realizzare la mailbox. Il primo consiste nel consentire ad ogni processo (sia al mittente che al ricevente) di iniziare la funzione corrispondente (`invio` o `ricezione`) e di bloccarsi dopo aver inserito o estratto alcuni caratteri dal buffer prima di terminare la funzione stessa. Ciò accade, ad esempio, se il mittente invia un messaggio di dimensione superiore a 10 caratteri (dimensione del buffer). In questo caso il processo che si sospende all'interno della funzione dovrà riprendere la sua esecuzione quando sarà possibile. Per questo motivo la soluzione prevede l'uso di *semafori condizione* utilizzati con la tecnica del *passaggio del testimone*.
Il secondo problema riguarda la necessità di evitare che messaggi inviati da mittenti diversi vengano mescolati nel buffer. Ciò potrebbe accadere se un mittente, sospeso nel proprio invio dopo aver inserito alcuni caratteri nel buffer, prima di riprendere la sua esecuzione venisse anticipato da un altro mittente che, iniziando il proprio invio, mescolerebbe i caratteri del suo messaggio con quelli del precedente. Per risolvere questo problema è sufficiente gestire un semaforo di mutua esclusione tra mittenti diversi.
Le due funzioni di invio e ricezione possono essere specificate mediante le seguenti **region**:

```
void invio(char messaggio[]):
          region mailbox <<
                              do when(cont<10) <inserisci un carattere nel buffer>;
                              while(<non sono stati inseriti tutti i caratteri del messaggio>);
                            >>
void ricezione(char messaggio[]):
          region mailbox <<
                              do when(cont>0) <estari un carattere dal buffer>;
                              while(<non sono stati estratti tutti i caratteri del messaggio>);
                            >>
```

La funzione `invio` è una **region** in quanto opera sulla `mailbox` che è condivisa tra tutti i mittenti, di cui solo uno alla volta può essere attivo sulla `mailbox`. Inoltre `invio`, eseguita dal mittente attivo, condivide la `mailbox` col ricevente che esegue `ricezione` che, quindi, è essa stessa una **region**.

Soluzione

```
class mailbox {
   char buffer[10]; /* vettore circolare*/
   int ultimo=0; /* rappresenta l'indice dell'elemento di buffer in cui inserire il prossimo carattere*/
   int  primo=0; /* rappresenta l'indice dell'elemento di buffer da cui prelevare il prossimo
                    carattere*/
   int cont=0; /* contatore degli elementi pieni del buffer*/
   semaphore mutex=1; /*semaforo di mutua esclusione tra il ricevente e il mittente attivo*/
   mutex_mittenti=1; /*semaforo di mutua esclusione tra mittenti diversi*/
   semaphore non_pieno=0; /* semaforo condizione associato alla condizione (cont<10) */
   semaphore non_vuoto=0; /*semaforo condizione associato alla condizione (cont>0) */
   boolean mit_sospeso=false; /* indica se il mittente è sospeso. Ciò accade quando, avendo
                                 trovato la condizione (cont<10) falsa, si è bloccato in attesa di
                                 poter terminare il suo invio*/
   boolean ric_sospeso=false; /* indica se il ricevente è sospeso. Ciò accade quando, avendo
                                 trovato la condizione (cont>0) falsa, si è bloccato in attesa di
                                 poter terminare la sua ricezione*/

   public void invio(char messaggio[]) {
      int i=0;
```

```
   P(mutex_mittenti); /*un solo mittente alla volta può eseguire invio */
   P(mutex); /*il mittente attivo entra in competizione col ricevente*/
   do
   {  if(cont==10) { /*buffer pieno. Il mittente si sospende e attiva il ricevente, se
                       sospeso, passandogli il diritto a operare in mutua esclusione*/
         mit_sospeso=true;
         if (ric_sospeso) V(priv_ric); else V(mutex);
         P(priv_mit);
         mit_sospeso=false;
         }
      buffer[ultimo]=messaggio[i];
      ultimo=(ultimo+1)%10;
      cont++;
      i++ }
   while(messaggio[i-1]!='\0');
   V(mutex);
   V(mutex_mettenti);
}

public void ricezione(char messaggio[]) {
   int i=0;
   P(mutex); /*il ricevente attivo entra in competizione col mittente */
   do
   {  if(cont==0) { /*buffer vuoto. Il ricevente si sospende e attiva il mittente, se
                      sospeso, passandogli il diritto a operare in mutua esclusione*/
         ric_sospeso=true;
         if (mit_sospeso) V(priv_mit); else V(mutex);
         P(priv_ric);
         ric_sospeso=false;
         }
      messaggio[i]=buffer[primo];
      primo(primo+1)%10;
      cont--;
      i++ }
   while(messaggio[i-1]!='\0');
   V(mutex);
}
```