Esercizio sulle Semafori e Memoria Condivisa

- In una sala di un museo, è ammessa la presenza contemporanea di al massimo 5 persone.
- Quando una persona vuole entrare, verifica innanzi tutto che ci sia posto disponibile. Se la sala è piena se ne va, e riproverà più tardi.
- Se c'è posto in sala, la persona entra, rimane nella sala per una quantità di tempo casuale, e poi esce.

Esercizio sulle Semafori e Memoria Condivisa

- costruite un comando *entra* che simuli la situazione appena descritta. In particolare, entra deve fare le seguenti cose:
- la prima volta che viene eseguito, *entra* alloca un segmento di memoria condivisa contenente un array di 5 interi, e un semaforo che servirà a gestire l'accesso mutuamente esclusivo per tutte le operazioni da eseguire sul segmento di memoria condivisa

Esercizio sulle Semafori e Memoria Condivisa

- Il comando *entra* cerca la prima entry libera nell'array, e vi scrive dentro il suo pid (le entry a 0 sono libere, le entry contenti un valore maggiore di zero contengono i pid di processi che rappresentano persone attualmente nella sala)
- Se *entra* trova che l'array è pieno, stampa su terminale il contenuto dell'array, avvisa l'utente di riprovare più tardi e termina.

Esercizio sulle Semafori e Memoria Condivisa

- Dopo essere riuscito a registrarsi nell'array, *entra* stampa su terminale il contenuto corrente dell'array, e attende per un tempo random compreso tra 1 e 10 secondi (per simulare il tempo necessario a visitare la sala del museo)
- Dopo l'attesa di 1-10 secondi, *entra* segnala su terminale che sta per uscire dalla sala: rimette a zero la entry contentente il suo pid e termina

Esercizio sulle Semafori e Memoria Condivisa

- Prevedete la possibilità che il comando *entra* venga eseguito con un argomento opzionale:

  $> entra termina
- In questo caso, *entra* deve rimuovere il segmento di memoria condivisa, il semaforo, e terminare.

Alcune osservazioni:

- Per sapere come fanno le esecuzioni di *entra* successive alla prima, ad usare le risorse allocate dalla prima esecuzione di *entra*, guardate i lucidi relativi all'esercizio sulle Code di Messaggi
- L'accesso mutuamente esclusivo all'array deve essere strettamente limitato al tempo necessario per scrivere e cancellare il pid del processo. In altre parole, mentre una istanza di *entra* è in attesa che trascorra il tempo di 1-10 secondi, altre istanze di *entra* possono essere lanciate e accedere all'array (ossia deve essere effettivamente possibile per più "persone" accedere contemporaneamente alla sala del museo)

Alcune osservazioni:

- Per provare il funzionamento di entra, è sufficiente prevedere uno script contenente un numero sufficiente di chiamate a *entra*. Ad esempio con un for (è un costrutto della shell che vedremo più avanti):

```
for i in 1 2 3 4 5 6 7 8 9 10
   do
   entra
done
```

shell script

- E verificare che gli output delle varie chiamate siano corretti

Una possibile variante:

- Gli output su terminale di più chiamate consecutive di *entra* possono essere difficili da leggere.
- Modificate *entra* in modo che scriva tutti i messaggi di stampa su un file *entra.log*.
- Anche l'accesso a *entra.log* deve essere in mutua esclusione, e quindi regolato da un secondo semaforo prelevato dalla prima esecuzione di entra.

Una possibile variante:

- Potete fare in modo che alcune stampe di *entra* vengano inviate sia su file che su terminale, ad esempio quelle che segnalano l'ingresso e l'uscita dalla sala. Su terminale potrebbe quindi apparire qualcosa del tipo (il numero è il pid del processo *entra* che esegue la stampa):

  415: entro nella sala

  516: entro nella sala

  651: sala piena, riprova più tardi

  415 esco dalla sala

  715: entro nella sala

  .....