# Remote Reservation System

Sistema di prenotazione posti remoto.

# Specifica

Realizzazione di un sistema di prenotazione posti per una sala cinematografica. Un processo su una macchina server gestisce una mappa di posti per una sala cinematografica. Ciascun posto e' caratterizzato da un numero di fila, un numero di poltrona ed un FLAG indicante se il posto e' gia' stato prenotato o meno. Il server accetta e processa le richieste di prenotazione di posti da uno o piu' client (residenti, in generale, su macchine diverse). Un client deve fornire ad un utente le seguenti funzioni:

1. Visualizzare la mappa dei posti in modo da individuare quelli ancora disponibili.
2. Inviare al server l'elenco dei posti che si intende prenotare (ciascun posto da prenotare viene ancora identificato tramite numero di fila e numero di poltrona).
3. Attendere dal server la conferma di effettuata prenotazione ed un codice di prenotazione.
4. Disdire una prenotazione per cui si possiede un codice.

Si precisa che lo studente e' tenuto a realizzare sia il client che il server.

Il server deve poter gestire le richieste dei client in modo concorrente.

# Installazione

## Scaricare il codice sorgente

Posizionarsi nella cartella dove si vuole scaricare il codice e utilizzare wget

```
wget -c "https://github.com/ael-code/remote-reservation-system/archive/master.zip"
-O "remote-reservation-system.zip"
```

Decomprimere l'archivio

```
unzip remote-reservation-system.zip
```

Spostarsi nella cartella dove si trova il codice sorgente

```
cd remote-reservation-system-semaphores
```

### Compilazione

Il processo di compilazione e' stato facilitato tramite l'utilizzo di un Makefile. E' sufficente invocare

```
make
```

Verra' creata una cartella "bin" contenente i due eseguibili e i file intermedi utili alla compilazione (con estensione '.o')

# Esecuzione

## Server

Se si e' utilizzato il metodo di compilazione sopra descritto il file eseguibile relativo al server si trova in:

```
bin/rss-server
```

Il server necessita di due parametri obbligatori da passare tramite riga di comando e accetta eventuali parametri opzionali

*rss-server [OPTION...] rows columns*

rows e coloumns servono a definire il numero totale di posti che la sala puo' ospitare. I posti sono gestiti dal programma come una matrice e questi due parametri servono proprio a specificare le caratteristiche di quest matrice.

### Parametri opzionali

```
General options:
  -f, --file=FILE-NAME       Backup file
  -p, --port=PORT-NUM        Listening port
  -s, --pwd-length=LENGTH    length of password used to generate reservation keys [
default 8]

 Output options:
  -c, --colored-output       Colored output
  -v, --verbose              Verbose output

  -?, --help                 Give this help list
      --usage                Give a short usage message
```

### Esempi

Avviare un server che gestisce 25 posti e rimane in ascolto sulla porta 8080

```
rss-server -cv 5 5 -p 8080
```

Avviare un server che gestisce 10 posti, rimane in ascolto su una porta casuale e salva la sessione sul file "rss.bk"

```
rss-server -cv 5 2 -f 'rss.bk'
```

Ripristinare la sessione salvata sul file "rss.bk"

```
server -f rss.bk
```

# Client

Se si e' utilizzato il metodo di compilazione sopra descritto il file eseguibile relativo al client si trova in:

```
bin/rss-client
```

Il client necessita di due parametri obbligatori da passare tramite riga di comando e accetta eventuali parametri opzionali

*rss-client [OPTION...] hostname port*

- **hostname** indica l'ip della macchina remota dove sta girando il processo server
- **port** indica la porta sul quale e' in ascolto il processo server

## Parametri opzionali

```
Operations:
  -d, --delete=CODE          Request to revocate a reservation
  -r, --reserve              Reserve some seats

 Settings:
  -c, --colored-output       Colored output
  -v, --verbose              Verbose output

  -?, --help                 Give this help list
      --usage                Give a short usage message
```

## Esempi

Richiedi la situazione attuale dei posti al server

```
client -cv 127.0.0.1 1234
```

Effettua una prenotazione. In questo caso verra' avviata un'interazione con l'utente per l'immisione del numero e della posizione dei posti dsiderati.

```
client -cv 127.0.0.1 1234 -r
```

Richiedi la cancellazione di una prenotazione precedentemente effettuata. **CODE** e' il codice ricevuto dal server durante la prenotazione

```
client -cv 127.0.0.1 1234 -d 'CODE'
```

# Tips & Tricks

Elimina la cartella di compilazione e altri file temporanei (Attenzione cancella anche tutti i files con estensione ".bk")

```
make clean
```

Stampa le statistiche di questo progetto

```
make stat
```

# Troubleshooting

Se l'esecuzione del server restituisce questo errore:

> *semget in matrix_init(): invalid argument*

Vuol dire che il programma sta tentando di allocare un numero di semafori o una dimensione dei semafori maggiore di quella permessa dal kernel del tuo sistema operativo. (Solitamente 32 semafori di dimensione 250).

Per ovviare al problema si puo' modificare il file

```
/proc/sys/kernel/sem
```

Aumentando il primo valore (dimensione di ogni semaforo) e il terzo valore (numero di semafori totali)