1. esplorazione della struttura *msqid_ds*

Scrivere un programma in cui è creata una coda di messaggi.

- Prima di effettuare qualsiasi operazione di invio di messaggi sulla coda, utilizzando la struttura di tipo *msqid_ds* associata alla coda, stampare la dimensione della coda (membro *msg_qbytes*), il numero di messaggi in coda (membro *msg_qnum*) e il tempo dell'ultima *msgsnd()* (deve essere 0, poiché non ci sono ancora stati invii sulla coda).
- Effettuare una *msgsnd()*, e verificare che il momento della *msgsnd()* è cambiato. Utilizzare la funzione *ctime()* di *time.h* per visualizzare il momento in cui è occorsa la *msgsnd()*.
- Prima di terminare il programma deve deallocare la coda.

2. costruzione di interfaccia alle system call *msgsnd()* e *msgrcv()*

Scrivere 2 programmi che funzionino da interfaccia alla *msgsnd()* e alla *msgrcv()*.

- Il primo, *intf_snd.c* (che fornisce un'interfaccia alla *msgsnd()*) deve creare una coda di messaggi: dopo la creazione della coda, entra in un ciclo di richiesta all'utente in cui a ogni interazione richiede all'utente di inserire un tipo di messaggio (un intero!) e una stringa. Tipo di messaggio e stringa saranno inseriti in un messaggio, e il messaggio inviato sulla coda. Scegliere una sequenza di caratteri che permetta di interrompere tale ciclo e terminare.
- Il secondo programma, *intf_rcv.c* (che fornisce un'interfaccia alla *msgrcv()*) deve connettersi alla stessa coda creata dal primo, domandare all'utente che tipo di messaggi desideri prelevare dalla coda, prelevarli e stamparli. Al termine della stampa dei messaggi, il processo deve rimuovere la coda.

3. invio e ricezione messaggi

Scrivere un programma che allochi una coda di messaggi tramite *msgget()*, generi un processo figlio e si metta quindi in attesa di un messaggio inviato dal figlio.

Il processo figlio deve collegarsi alla coda generata dal processo padre e inviare su di essa un messaggio costituito da due dati: una stringa (la stringa "saluti da") e il PID del processo figlio. Fatto questo il processo figlio termina. Ricevuto il messaggio atteso, il processo padre visualizza a video quando ricevuto, dealloca la coda e termina.

4. rimozione delle code

Verificare che la coda è stata deallocata usando il comando *ipcs* da terminale. Se non risulta deallocata, rimuoverla utilizzando *ipcrm*.

Architettura client-server

Implementare una semplice architettura client-server che rispetti queste specifiche.

Server:

1. Alloca una coda di messaggi e poi esegue un ciclo di attesa delle richieste dei processi clienti.
2. Per ogni richiesta ricevuta, esegue una *fork()* e passa al figlio la richiesta mentre lui si mette in attesa di una nuova richiesta.
3. Il figlio esamina la richiesta, produce una risposta e la invia al processo richiedente (nessun altro processo deve poter ricevere tale messaggio), quindi termina.
4. Se il messaggio ricevuto dal padre non è una richiesta ma un messaggio di fine, il padre dealloca la coda e termina.

Client:

1. Genera un numero di processi figli pari al numero che viene passato come argomento dall'utente all'atto dell'esecuzione (da linea di comando, usare *argc* e *argv*).
2. Esegue un ciclo di *wait()* in attesa della loro terminazione.
3. Ciascun figlio si aggancia a una coda di messaggi allocata dal server.
4. Se la coda non esiste il processo segnala la cosa all'utente e termina.
5. Se la coda esiste, il processo invia una richiesta (decidete voi quale informazione inserire nel messaggio) e si mette in attesa della risposta.
6. Ricevuta la risposta termina.
7. Quando tutti i figli sono terminati il processo padre informa il server che può deallocare la coda.