```c
//share.h
// keys for identifying the message queues
#define INKEY 0xaffedead
#define OUTKEY 0xbeeffeed
// maximum message length
#define MAXSIZE 32767

//msg_server
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <fcntl.h>
#include <errno.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <unistd.h>
#include "share.h"

int end = 0;

void sigint_handler(void *in){
  end = 1;
}

void rot13(char *buf){
  int idx=0;
  char c;

  c = buf[idx];
  while(c) {
    if (isalpha(c))
      buf[idx] = (toupper(c) <= 'M') ? c+13 : c-13;
    c = buf[++idx];
  }
  return;
}

int main(int argc, char* argv[]){
  int input_id, output_id, ret;
  ssize_t rcvlength;
  char *msgbuf;

  msgbuf = malloc(MAXSIZE);
  if (!msgbuf) {
    perror("malloc();");
    exit(EXIT_FAILURE);
  }

  ret = (int) signal(SIGINT, (__sighandler_t) sigint_handler);
  if (ret == -1) {
    free(msgbuf);
    perror("signal()");
    exit(EXIT_FAILURE);
  }

  /* Create two Message Queues */
  input_id = msgget(INKEY, IPC_CREAT | S_IRUSR | S_IWUSR | S_IWGRP | S_IWOTH);
  if (input_id == -1) {
    free(msgbuf);
    perror("msgget() input");
    exit(EXIT_FAILURE);
  }
  output_id = msgget(OUTKEY, IPC_CREAT | S_IRUSR | S_IWUSR | S_IRGRP |
                   S_IROTH);
```

```c
  if (output_id == -1) {
    free(msgbuf);
    perror("msgget() output");
    exit(EXIT_FAILURE);
  }

  printf("msgserver started; pid %d\n", getpid());

  while (end != 1) {
    rcvlength = msgrcv(input_id, msgbuf, MAXSIZE, 0, MSG_NOERROR);
    if (rcvlength == -1) {
      if (errno == EINTR) { /* signal received; leave loop */
        continue;
      }
      else {   /* 'normal' error; exit */
        perror("msgrcv()");
        free(msgbuf);
        exit(EXIT_FAILURE);
      }
    }

    rot13(msgbuf);  /* 'process' the received message */

    ret = msgsnd(output_id, msgbuf, rcvlength, 0);
    if (ret == -1) {
        perror("msgsnd()");
        free(msgbuf);
        exit(EXIT_FAILURE);
    }
  }

  ret = msgctl(input_id, IPC_RMID, NULL);
  if (ret == -1) {
    perror("msgctl(), removing input queue");
  }
  ret = msgctl(output_id, IPC_RMID, NULL);
  if (ret == -1) {
    perror("msgctl, removing output queue");
  }

  free(msgbuf);
  printf("msgserver, pid %d, ended.", getpid());
  exit(EXIT_SUCCESS);
}

//msg_client

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include "share.h"

#define MAXLINE 2048

char pline[MAXLINE], cline[MAXLINE];

int main(int argc, char* argv[]){
  int iid, oid, ret;

  if ((iid = msgget(INKEY, 0)) == -1) {
    perror("client: msgget() input");
    exit(EXIT_FAILURE);
  }
  if ((oid = msgget(OUTKEY, 0)) == -1) {
    perror("client: msgget() output");
    exit(EXIT_FAILURE);
  }

  while (fgets(pline, MAXLINE, stdin)) {
    if ((ret = msgsnd(iid, pline, strlen(pline), 0)) == -1) {
      perror("client: msgsnd()");
    }
    if ((ret = msgrcv(oid, cline, MAXLINE, 0, 0)) == -1) {
      perror("client: msgrcv()");
    }
    else {
      printf("%s", cline);
    }
  }
  exit(EXIT_SUCCESS);
}
```