Message queues

1. ftok - Link 1
    - `key_t ftok(const char *pathname, int proj_id)`
    - The ftok() function uses the identity of the file named by the given *pathname* (which must refer to an existing, accessible file)and the least significant 8 bits of *proj_id* (which must be nonzero) to generate a *key_t* type System V IPC key. The resulting value is the same for all pathnames that name the same file when the same value of *proj_id* is used.
2. msgget - Link 1
    - `int msgget(key_t key, int msgflg)`
    - The msgget() system call returns the message queue identifier associated with the value of the *key* argument. It may be used to obtain the identifier of a previously created message queue or create a new one. The *msgflag* field can be modified using bitwise operations to use the function differently. For example, `msgget(key, PERMS | IPC_CREAT)` will create a new message queue with the id *key* if it already does not exist.
3. msgrcv - Link 1(Highly recommended)
    - `ssize_t msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp, int msgflg)`
    - An appropriate explanation regarding the function can be found in the link mentioned above.
4. msgsend - Link 1(Highly recommended)
    - `int msgsnd(int msqid, const void *msgp, size_t msgsz, int msgflg)`
    - An appropriate explanation regarding the function can be found in the link mentioned above.
5. fgets - Link 1
    - `char *fgets (char *str, int n, FILE *stream)`
    - *str* is a pointer to an array of chars where the string read is copied. *n* is the maximum number of characters copied into *str*(including the terminating null character). **stream* is a pointer to a FILE object that identifies an input stream. The fgets() function returns a pointer to the string where the input is stored.
6. msgctl - Link 1, Link 2(Highly recommended)
    - `int msgctl(int msqid, int cmd, struct msqid_ds *buf)`
    - The *msgctl*() function shall provide message control operations as specified by *cmd.* Refer to Link 2, as mentioned above, for more information on what each value of *cmd* would do. Information about *buf* can be found in Link 1, but the field can be kept NULL or 0 depending on the purpose for which the *msgctl()* function is used.
7. sprintf - Link 1, Link 2
    - `int sprintf(char *str, const char *format, ...)`
    - sprintf stands for "string print." In C programming language, it is a file-handling function that sends formatted output to the string. Instead of printing on the console, the *sprintf()* function stores the output on the char buffer specified in sprintf.

Problem 0
Write a C program (reader.c) that creates a message queue and prints the first message from it. Additionally, write another program (writer.c) that sends a message to this queue.

Solution
1) reader.c

```
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <sys/types.h>

#define PERMS 0644

// Struct for the message
struct my_msgbuf {
  long mtype;
  char mtext[200];
};

int main() {
  struct my_msgbuf buf;
  int msqid;
  int len;
  key_t key;

  // Generate System V IPC key
  if ((key = ftok("writer.c", 'B')) == -1) {
     printf("Error in ftok");
     exit(1);
  }

  // Get the message queue ID using key
  if ((msqid = msgget(key, PERMS | IPC_CREAT)) == -1) {
     printf("Error in creating message queue");

     exit(1);
  }

  // Read the message
  if (msgrcv(msqid, &buf, sizeof(char) * 200, 0, 0) == -1) {
     printf("Error in receiving the message");
     exit(1);
  }
```

```
  printf("Reader: \"%s\"\n", buf.mtext);

  // Delete the message queue
  if (msgctl(msqid, IPC_RMID, NULL) == -1) {
     printf("Error in deleting the message queue");
     exit(1);
  }
  return 0;
}
```

2) writer.c

```
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <sys/types.h>

#define PERMS 0644

// Struct for the message
struct my_msgbuf {
  long mtype;
  char mtext[200];
};

int main() {
  struct my_msgbuf buf;
  int msqid;
  int len;
  key_t key;

  // Generate System V IPC key
  if ((key = ftok("writer.c", 'B')) == -1) {
     printf("Error in ftok");
     exit(1);
  }

  // Create the message queue and get its ID using key
  if ((msqid = msgget(key, PERMS)) == -1) {
     printf("Error in getting message queue ID. Did you run the
reader?");
     exit(1);
  }

  buf.mtype = 1;
```

```
  sprintf(buf.mtext, "%s", "Hi");
  len = strlen(buf.mtext);

  // Send the message
  if (msgsnd(msqid, &buf, len + 1, 0) == -1) {
     printf("Error in sending the message");
     exit(1);
  }

  return 0;
}
```