message queue

- A process may send or receive data with a predefined format from another process
- Four system calls on message queues
  - msgget ( ) – allocates the queue where message objects are saved
  - msgrcv ( ) – retrieves a message in FIFO fashion according to the mtype value of the message. When a message to retrieve is not in the queue, the function may halt the execution of the process until the correct message is available
  - msgsnd ( ) – puts a message in the queue. When the queue is full, the function may halt the execution of the process until space is available for the new message
  - msgctl ( ) – deallocates the queue when the processes terminate (may use for various other operations also)

msgQ_A.cpp

```
1   /*
2   This is a simple illustration of the use of:
3               ftok, msgget, msgsnd, msgrcv
4
5   Program A will use a message queue created by Program B.
6
7   Program A sends the first message and reads the reply. Program A
8   also sends two "fake" messages to the msgQ.
9
10  This implementation does not contain error handling routines.

1   */
2   #include <sys/ipc.h>
3   #include <sys/msg.h>
4   #include <iostream>
5   using namespace std;
```

msgQ_A.cpp

```
1   int main( ) {
2           int qid = msgget(ftok(".",'u'), 0);
3
4           // declare my message buffer
5           struct buf {
6                   long mtype; // required
7                   char greeting[50]; // mesg content
8           }; buf msg;
9           int size = sizeof(msg)-sizeof(long);
10
1           msg.mtype = 111;            // set message type mtype = 111
2           strcpy(msg.greeting, "Fake message");
3           msgsnd(qid, (struct msgbuf *)&msg, size, 0);
4
5           msg.mtype = 113;            // set message type mtype = 113
6           strcpy(msg.greeting, "Another fake");
7           msgsnd(qid, (struct msgbuf *)&msg, size, 0);
```

msgQ_A.cpp

```
1           // prepare my message to send
2           strcpy(msg.greeting, "Hello there");
3
4           cout << getpid( ) << ": sends greeting" << endl;
5           msg.mtype = 117;              // set message type mtype = 117
6           msgsnd(qid, (struct msgbuf *)&msg, size, 0); // sending
7
8           msgrcv(qid, (struct msgbuf *)&msg, size, 314, 0); // why not 117?
9           cout << getpid( ) << ": gets reply" << endl;
10          cout << "reply: " << msg.greeting << endl;
1           cout << getpid( ) << ": now exits" << endl;
2
3           msg.mtype = 117;
4           msgsnd(qid, (struct msgbuf *)&msg, size, 0);
5
            exit(0);
    }
```

msgQ_B.cpp

```
1   /*
2   This is a simple illustration of the use of:
3               ftok, msgget, msgsnd, msgrcv, and wait
4
5   Program B creates a message queue to be shared with Program A.
6   Program B receives a message and reply to Program A. Also, Program B
7   clears both messages from the queue.
8
9   This implementation does not contain error handling routines
10
1   */
2   #include <sys/ipc.h>
3   #include <sys/msg.h>
4   #include <iostream>
5   using namespace std;
```

msgQ_B.cpp

```
1   int main( ) {
2           // create my msgQ with key value from ftok()
3           int qid = msgget(ftok(".','u'), IPC_EXCL|IPC_CREAT|0600);
4
5           // declare my message buffer
6           struct buf {
7                   long mtype; // required
8                   char greeting[50]; // mesg content
9           }; buf msg;
10          int size = sizeof(msg)-sizeof(long);
1
2           msgrcv(qid, (struct msgbuf *)&msg, size, 117, 0); // read real mesg
3                                                             // don't read "fake" mesg
4           cout << getpid( ) << ": gets message" << endl;
5           cout << "message: " << msg.greeting << endl;
```

msgQ_B.cpp

```
1           strcat(msg.greeting, " and Adios.");
2           cout << getpid( ) << ": sends reply" << endl;
3           msg.mtype = 314; // only reading mesg with type mtype = 314
4           msgsnd(qid, (struct msgbuf *)&msg, size, 0);
5           cout << getpid( ) << ": now exits" << endl;
6
7           msgrcv(qid, (struct msgbuf *)&msg, size, -112, 0);
8           msgrcv(qid, (struct msgbuf *)&msg, size, 0, 0);
9
10          msgrcv(qid, (struct msgbuf *)&msg, size, 117, 0);
1
2           //now safe to delete mesg queue
3           msgctl(qid, IPC_RMID, NULL);
4
5           exit(0);
    }
```

To execute the programs in lab

A) Compile: g++ msgQ_A.cpp -o a.out

B) Compile: g++ msgQ_B.cpp -o b.out

C) First execute: b.out

D) Open another terminal window to execute: a.out

Online man pages: man <function> (e.g., man msgrcv)