```
#include      <sys/types.h>
#include      <sys/ipc.h>
#include      <sys/msg.h>

#define MSGKEY      75

struct msgform {
      long      mtype;
      char      mtext[256];
};

main()
{
      struct msgform msg;
      int msgid, pid, *pint;

      msgid = msgget(MSGKEY, 0777);

      pid = getpid();
      pint = (int *) msg.mtext;
      *pint = pid;      /* copy pid into message text */
      msg.mtype = 1;

      msgsnd(msgid, &msg, sizeof(int), 0);
      msgrcv(msgid, &msg, 256, pid, 0);      /* pid is used as the msg type */
      printf("client: receive from pid %d\n", *pint);
}
```

Figure 11.6. A Client Process

A process can receive messages of a particular type by setting the *type* parameter appropriately. If it is a positive integer, the kernel returns the first message of the given type. If it is negative, the kernel finds the lowest type of all messages on the queue, provided it is less than or equal to the absolute value of *type*, and returns the first message of that type. For example, if a queue contains three messages whose types are 3, 1, and 2, respectively, and a user requests a message with type −2, the kernel returns the message of type 1. In all cases, if no messages on the queue satisfy the receive request, the kernel puts the process to sleep, unless the process had specified to return immediately by setting the *IPC_NOWAIT* bit in *flag*.

Consider the programs in Figures 11.6 and 11.8. The program in Figure 11.8 shows the structure of a *server* that provides generic service to *client* processes. For instance, it may receive requests from client processes to provide information from a database; the server process is a single point of access to the database, making consistency and security easier. The server creates a message structure by setting

```
#include        <sys/types.h>
#include        <sys/ipc.h>
#include        <sys/msg.h>

#define MSGKEY        75
struct msgform
{
        long mtype;
        char mtext[256];
} msg;
int msgid;

main()
{
        int i, pid, *pint;
        extern cleanup();

        for (i = 0;  i < 20;  i++)
                signal(i, cleanup);
        msgid = msgget(MSGKEY, 0777 | IPC_CREAT);

        for (;;)
        {
                msgrcv(msgid, &msg, 256, 1, 0);
                pint = (int *) msg.mtext;
                pid = *pint;
                printf("server: receive from pid %d\n", pid);
                msg.mtype = pid;
                *pint = getpid();
                msgsnd(msgid, &msg, sizeof(int), 0);
        }
}

cleanup()
{
        msgctl(msgid, IPC_RMID, 0);
        exit();
}
```

Figure 11.8. A Server Process

Messages are formatted as type-data pairs, whereas file data is a byte stream. The *type* prefix allows processes to select messages of a particular type, if desired, a feature not readily available in the file system. Processes can thus extract messages of particular types from the message queue in the order that they arrive, and the kernel maintains the proper order. Although it is possible to implement a message