Lab #6 Solutions: COMP 3000B (Operating Systems)

1. (2 pts) What is the right syntax for msgget() that will create a private message queue with access only to the owner? What standard functions are used to send and receive messages on this message queue?

   Ans: To create the message queue:

   ```
   int queue_id = msgget(IPC_PRIVATE, 0600);
   ```

   Messages can be sent and received with this queue using msgsnd() and msgrcv(), respectively.

2. (2 pts) Use 'mknod' to create a name pipe called 'pipe-test' in your home directory. Run "more pipe-test". What does more do? How could you change this behavior without sending more a (manual) signal?

   Ans: The more command should immediately block when opening the pipe; it does this because it is waiting for another process to connect to the queue to send data. You can get more to unblock by creating a process that will attach to the pipe, say by running "echo foo > pipe-test" or even "ls > pipe-test".

3. (2 pts) The most common way of writing data to a socket is using the write() system call defined like this: int write(int socket, char *buffer, int buflen). What values write( ) may return? If successful, does it mean all the date has arrived at the other end?

   Ans: On a socket, the write system call returns one of the following values:

0	The connection was closed by the remote host.
-1	The write system call was interrupted, or failed for some reason.
n	The write system call wrote 'n' bytes into the socket.

   The kernel buffers network transmissions before sending them on the wire. If the kernel's buffer storage hasn't been exceeded, then the write system call will normally return immediately, i.e. it will return before data has been transmitted. Thus, data may not have even left the computer when the write system call returns.

   (If the kernel's allowed outgoing socket buffer storage has been exceeded, then the write system call will block until room becomes available.)

4. (4 pts) The program "pipe.c" is using a pipe to communicate data between a process and its child process. The parent reads input from the user, and sends it to the child via a pipe. The child prints the received data to the screen. Please implement function "dochild" and "doparent" in "pipe.c" using some combinations of the system calls open, read, write, close, and exit.

   Ans: Here is a very simple, not too robust implementation of do_child() and do_parent():

```
void do_child(int data_pipe[]) {
   int c;      /* data received from the parent. */
   int rc;     /* return status of read().    */

   /* first, close the unneeded writable portion of the pipe. */
   close(data_pipe[1]);

   /* now enter a loop of reading data from the pipe, and printing it */
   while ((rc = read(data_pipe[0], &c, 1)) > 0) {
     putchar(c);
   }

   /* probably pipe was broken, or got EOF via the pipe. */
   exit(0);
}

void do_parent(int data_pipe[])
{
   int c;      /* data received from the user. */
   int rc;     /* return status of getchar(). */

   /* first, close the unneeded writable portion of the pipe. */
   close(data_pipe[0]);

   /* now enter a loop of read user input, and writing it to the pipe. */
   while ((c = getchar()) > 0) {

     /* write the character to the pipe. */
     rc = write(data_pipe[1], &c, 1);

     if (rc == -1) { /* write failed - notify the user and exit */
       perror("Parent: write error");
       close(data_pipe[1]);
       exit(1);
     }
   }

   /* probably got EOF from the user. */
   close(data_pipe[1]); /* close the pipe, to let the
                           child know we’re done. */
   exit(0);
}
```