**Report 8.6**

8.6:

Program 1: Message Receiver

---------------------------

Program 1 acts as the message receiver. It creates a message queue, waits to receive a message from Program 2, and then displays the received data on the console. Here's a breakdown of the program's flow:

1. It includes the necessary header files for interprocess communication and defines constants for message size, key file path, ID, and return values.

2. The program defines a structure `msgbuff` to hold the message type and text.

3. It declares variables for message queue ID, key, and the `msqid\_ds` structure for message queue control.

4. The program prints a start message to indicate its execution has begun.

5. It generates a key using the `ftok` function with a key file path and ID.

6. The program attempts to acquire the message queue ID using `msgget`. If successful, it displays the obtained message queue ID.

7. It waits to receive a message using `msgrcv` and stores the received message in the `message` structure.

8. The program displays the received message on the console.

9. It deletes the message queue using `msgctl` with the `IPC\_RMID` command.

10. Finally, the program prints an end message and exits.

Program 2: Message Sender

-------------------------

Program 2 acts as the message sender. It prompts the user to enter an arbitrary character string, sends the message to Program 1 using a message queue, and displays a success message. Here's a breakdown of the program's flow:

1. It includes the necessary header files for interprocess communication and defines constants for message size, key file path, ID, and return values.

2. The program defines a structure `msgbuff` to hold the message type and text.

3. It declares variables for message queue ID, key, and the `msqid\_ds` structure for message queue control.

4. The program prints a start message to indicate its execution has begun.

5. It generates a key using the `ftok` function with a key file path and ID.

6. The program attempts to acquire the message queue ID using `msgget`. If successful, it displays the obtained message queue ID.

7. It prompts the user to enter an arbitrary character string.

8. The program sets the message type to 1, copies the entered message into the `message` structure, and sends the message using `msgsnd`.

9. If the message sending is successful, it displays a success message.

10. Finally, the program prints an end message and exits.

By executing Program 1 and Program 2 in sequence, you establish an interprocess communication channel using a message queue. Program 1 receives messages from Program 2 and displays the received data, while Program 2 sends a message to Program 1 after prompting for user input.

This is the result: (because my laptop is not update so I borrow my friend to run the code)

Ảnh có chứa văn bản, phần mềm, Phần mềm đa phương tiện, ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động