DEPARTMENT OF INFORMATION TECHNOLOGY

IT 253 Operating Systems Lab

LAB3: 14/03/2023

Evaluation: 10 Marks

Objective

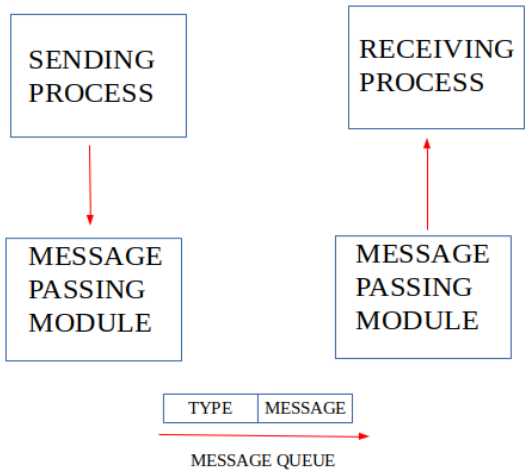
To understand the Inter process communication – Message Queues

Inter process communication (IPC) -Message Queue

1. Read these Basics for understanding IPC: message queue

A message queue is a linked list of messages stored within the kernel and identified by a message queue identifier. A new queue is created or an existing queue opened by **msgget()**. New messages are added to the end of a queue by **msgsnd()**. Every message has a positive long integer type field, a non-negative length, and the actual data bytes (corresponding to the length), all of which are specified to msgsnd() when the message is added to a queue. Messages are fetched from a queue by **msgrcv()**. We don't have to fetch the messages in a first-in, first-out order. Instead, we can fetch messages based on their type field. All processes can exchange information through access to a common system message queue. The sending process places a message (via some (OS) message-passing module) onto a queue which can be read by another process. Each message is given an identification or type so that processes can select the appropriate message. Process must share a common key in order to gain access to the queue in the first place.

- While pipes send unstructured byte streams, messages are sent as distinct units. When a message is retrieved from the queue, the process receives exactly one message in its entirety; there is no way to retrieve part of a message and leave the rest in the queue.
- Message queues use special identifiers, rather than file descriptors. As a result, message queues require special functions for sending and receiving data, rather than the standard file interface functions.
- Message queues have associated metadata that allows processes to specify the order in which messages are received. That is, message queues do not require or guarantee a first-in, first-out ordering.
- Message queues have kernel-level persistence and use special functions or utilities for removing them. Killing the process will not remove the message queue.



System calls used for message queues:

ftok(): is use to generate a unique key.

•**msgget():** either returns the message queue identifier for a newly created message queue or returns the identifiers for a queue which exists with the same key value.

•**msgsnd():** Data is placed on to a message queue by calling msgsnd().

•**msgrcv()**: messages are retrieved from a queue.

•**msgctl():** It performs various operations on a queue. Generally it is use to destroy message queue.

2. Execute the following program where a messagewrite program sends message to message queue. The messageread program reads one message at a time from message queue. Observe the result write your observation along with screenshots of result. [Marks : 4 Marks]

```c
// C Program for Message Queue (Writer Process)
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MAX 20

// structure for message queue
struct mesg_buffer {
        long mesg_type;
        char mesg_text[100];
} message;

int main()
{
        key_t key;
        int msgid;

        // ftok to generate unique key
        key = ftok("progfile", 65);

        // msgget creates a message queue
        // and returns identifier
        msgid = msgget(key, 0666 | IPC_CREAT);
        message.mesg_type = 1;

        printf("Write Data : ");
        fgets(message.mesg_text,MAX,stdin);

        // msgsnd to send message
        msgsnd(msgid, &message, sizeof(message), 0);

        // display the message
        printf("Data send is : %s \n", message.mesg_text);

        return 0;
}
```

Above program is msgwrite.c

compile as $gcc msgwrite.c -o ser

$./ser

Here server receives the message from user and sores in message queue.

```c
// C Program for Message Queue (Reader Process)
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>

// structure for message queue
struct mesg_buffer {
    long mesg_type;
    char mesg_text[100];
} message;

int main()
{
    key_t key;
    int msgid;

    // ftok to generate unique key
    key = ftok("progfile", 65);

    // msgget creates a message queue
    // and returns identifier
    msgid = msgget(key, 0666 | IPC_CREAT);

    // msgrcv to receive message
    msgrcv(msgid, &message, sizeof(message), 1, 0);

    // display the message
    printf("Data Received is : %s \n",
                    message.mesg_text);

    // to destroy the message queue
    msgctl(msgid, IPC_RMID, NULL);

    return 0;
}
```

Above
program is msgread.c

compile as $gcc msgread.c -o cli

$./cli

The client receives the earliest message one at a time from message queue.

3. The producer consumer problem can be considered through message queue. Write a program where message type 1 is used to send messages to user1 and message type 2 is used to send messages to user2. Show the results where two clients will access the same message queue but user1 receives with different message and user 2 receives separate message. [Marks : 6 Marks]
