CS-362L Operating System Lab 06

Assignment 6

Inter-process Communication Using Message Queue.

Objectives: To understand what is inter-process communication and implement it using message queues and pipes.

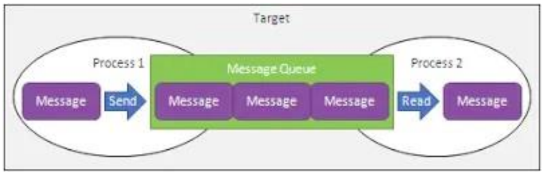
Processing steps:

What is inter-process communication?

IPC (inter-process communication) provides a way for the processes to communicate with each other. Processes executing concurrently in the operating system might be either independent processes or cooperating processes. A process is independent if it cannot be affected by the other processes executing in the system.

Message Queue

Message queues are one of the inter-process communication mechanism which allows processes to exchange data in the form of messages between two **unrelated** processes. Two (or more) processes can exchange information via access to a common system message queue. 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. The queue works on the principle of **FIFO** First In First Out. The first message inserted in the queue is the first one to be retrieved.



Step 1: Type of message

For sending data and deciding type of message, **struct** is used. The syntax is as follow:

```
struct msgque {
long msgtype;
char msgtext [size ];
};
```

Step 2: Create message queue

To create a new message queue or connect to the existing message queue **msgget()** system call is used. The syntax of the call is:

msgget(key_t key, int msgflg);

1. key_t key: **ftok()** is used to generate unique key.

2. msgflg: 0777 | IPC CREAT
3. Return Type: An integer value specify message queue identifier (msgid). On Error: return -1 with an error no specified below:

   - EACCESS (permission denied)
   - ENOENT (Queue does not exist)

Step 3: Set the type of message
In the main program, create an instance of the struct as follow:
struct msgque message;

Next, set the type of message for receiver end to receive.
message.msgtype = 1

Step 4: Send message to queue
Get message from user on command line using **gets** function.
To send message to queue, **msgsnd()** function is used. The syntax is as follow:

msgsnd(int msqid, &message, size_t message, int msgflg);

- msgid: id of the message queue.
- &message: address of the instance of struct.
- size_t: specifies the length of the message in bytes.
- msgflg: value set to 0.
- Return Value: returns 0 on success. On error: return -1.

Step 5: Receive message from queue
To receive message from queue, **msgrcv()** function is used. The syntax is as follow:

msgrcv(int msqid, &message, size_t message , long msgtype, int msgflg);

- msgid: id of the message queue.
- &message: address of the instance of struct.
- size_t: specifies the length of the message in bytes.
- msgtype: the message type set while sending.
- msgflg: value set to 0.

- Return Value: returns 0 on success. On error: return -1.

Step 6: Remove/Destroy queue
The **msgctl()** function alters the permissions and other characteristics of a message queue and performs the control operation. The syntax is as follow:
msgctl(int msqid, int cmd, struct msqid ds *buf);
The cmd argument is specified below:

- IPC STAT: Place information about the status of the queue in the data structure pointed to by buf.
- IPC RMID: Remove the message queue specified by the msqid argument.
- Return Value: returns 0 on success. On error: return -1.

Conclusion: At the end of this lab, student will be able to implement inter-process communication using message queues.

Class Activity

1. Write a C program to implement inter-process communication between two processes using message queue.

Exercise

1. Implement date time server through message queues.
   Hint:
   - The program creates a message queue then it forks.
   - The child process is the consumer.
   - The parent process is the producer.
   - The producer continues to generate a date string and sends it in a message to the consumer.
   - The message type for a date message is one

References:

- ”Inter-Process Communication”
  http://www.chandrashekar.info/articles/
  linux-system-programming/introduction-to-linux-ipc-mechanims.html