Author:
Abdussamed Guzey

Interprocess Communication Using IPC Sources

ABSTRACT

Processes running on the computer may need to communicate with each other for various reasons (speed, security, simplicity, etc.).This need can be provided by the IPC(Inter Process Communication) mechanism by the developers.IPC is a mechanism that enables communication between processes running on a computer.Signals,pipes,message queue,semaphores,shared memory are kind of IPC sources.In this section we will talk about M***essage Queues.***

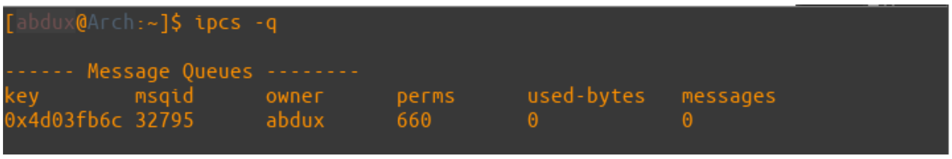
KEYWORDS

Message Queue,msgget,msgctl,msgsnd,msgrcv,key,message queue ID.

What is Message Queue?

Message Queueing is a method by which process can exchance or pass data using an interface to a system-managed queue of messages.The message queue stores the messages received by the processes in sequence (not interested in the content of the messages).Therefore, the sender and the recipient do not need to interact at the same time.This means that the message queue system is not synchronized.Like a mail box.In the message queue, messages are exchanged according to FIFO.Each message queue has key,message queue ID,owner and access permission.(Output of ipcs command shown Figure 0.1)

Figure 0.1



```
[abdux@Arch:~]$ ipcs -q

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
0x4d03fb6c 32795      abdux      660        0            0
```

Key:This argument generated by *ftok*[1] library function for use by *msgget* system call.And IPC_PRIVATE(value is 0), create a private message queue, with access only to the owner.

Msqid:This argument return by *msgget* system call.Processes access same message queue as this ID.

[1] See also linux man pages

How to create a message queue?

A message queue is generated by *msgget* system call.*msgget* have two parameters *key* and *msgflag*.The key is used to find the same message queue ID.And *msgflag* can be thought of as an order for creating a message queue and assigning permissions.

Implementation:

int *msgget(**key_t** key,**int** msgflag);*

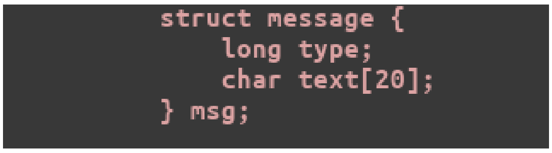
Returns message queue ID on success.
Returns -1 on failure.

Msgflags:
IPC_CREAT→Create the queue if it doesn't already exist in the kernel.
IPC_EXCL→When used with IPC_CREAT, fail if queue already exists.

We have a message queue. So how do we send and receive messages?

Msgsnd system call is used to send messages using the message queue.*msgsnd* have four parameters.The message queue id to send,message,message size and *msgflag.*Message must be a structure.An example message format shown Figure 0.2

Figure 0.2



```
struct message {
    long type;
    char text[20];
} msg;
```

Implementation:

int *msgsnd(**int** msqid,**const void** *msgp,**size_t** msgsz,**int** msgflag);*

Returns 0 on success.
Returns -1 on failure.

Msgrcv system call is used to receive messages using the message queue.msgrcv have five parameters.The message queue id to receive,message,message size,message type,*msgflag.*

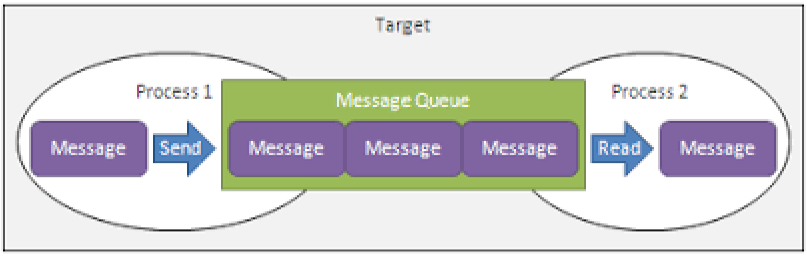
Implementation:

ssize_t *msgrcv(**int** msqid,**void** *msgp,**size_t** msgsz,**long** msgtyp,**int** msgflag);*

You can also look at the C code in this link.

A visual example send and receive message shown Figure 0.3

Figure 0.3



Message queue status and remove it

System call used to edit and remove the message queue is *msgctl*.It have three parameters.Message queue ID,command,message queue id datastrucutre.This datastructure includes some informations.(Shown Figure 0.4)

Implementation:

***int** msgctl(**int** msqid,**int** cmd,**struct msqid_ds** *buf);*

Returns 0 on success.
Returns -1 on failure.

Commands[2] for *msgctl*;

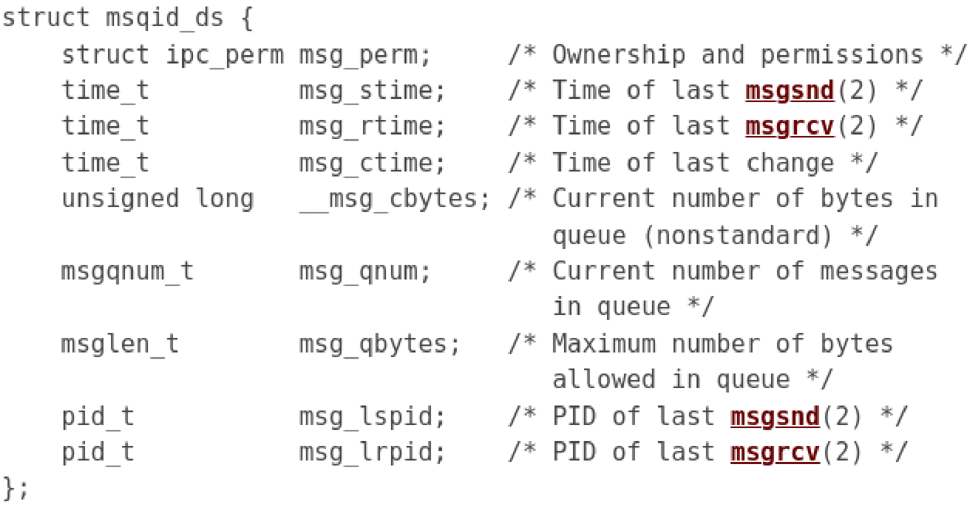
IPC_STAT→ Retrieves the msqid_ds structure for a queue, and stores it in the address of the buf argument.

IPC_SET→Sets the value of the ipc_perm member of the msqid_ds structure for a queue. Takes the values from the buf argument.

[2] Detailed information see also man pages for msgctl

IPC_RMID→Removes the queue from the kernel.

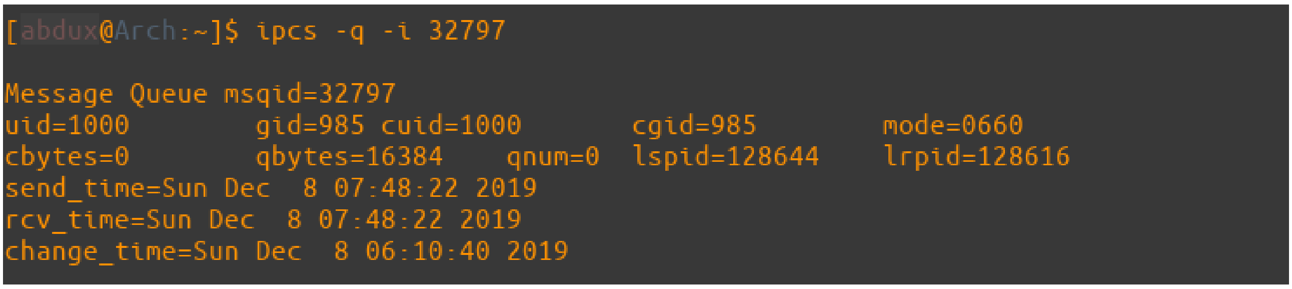
Message Queue ID datastructure **Figure 0.4**



```
struct msqid_ds {
    struct ipc_perm msg_perm;     /* Ownership and permissions */
    time_t          msg_stime;    /* Time of last msgsnd(2) */
    time_t          msg_rtime;    /* Time of last msgrcv(2) */
    time_t          msg_ctime;    /* Time of last change */
    unsigned long   __msg_cbytes; /* Current number of bytes in
                                     queue (nonstandard) */
    msgqnum_t       msg_qnum;     /* Current number of messages
                                     in queue */
    msglen_t        msg_qbytes;   /* Maximum number of bytes
                                     allowed in queue */
    pid_t           msg_lspid;    /* PID of last msgsnd(2) */
    pid_t           msg_lrpid;    /* PID of last msgrcv(2) */
};
```

and details an actually message queue using ipcs command in linux shown Figure 0.5

Figure 0.5



```
[abdux@Arch:~]$ ipcs -q -i 32797

Message Queue msqid=32797
uid=1000        gid=985 cuid=1000       cgid=985        mode=0660
cbytes=0        qbytes=16384    qnum=0  lspid=128644    lrpid=128616
send_time=Sun Dec  8 07:48:22 2019
rcv_time=Sun Dec  8 07:48:22 2019
change_time=Sun Dec  8 06:10:40 2019
```

Thank you for reading.

REFERENCES

https://medium.com/@Mohitdtumce/what-is-message-queue-b5468ff6db50
https://www.slideshare.net/anil_pugalia/ipc-5349082
https://www.geeksforgeeks.org/ipc-using-message-queues/

https://linux.die.net/man/2/msgctl
https://www.cs.kent.edu/~ruttan/sysprog/lectures/shmem/ipc_notes.html
https://searchapparchitecture.techtarget.com/definition/message-queueing
https://pubs.opengroup.org/onlinepubs/007904875/functions/msgrcv.html
https://www.tldp.org/LDP/lpg/node27.html#SECTION00742000000000000000

Books:
Interprocess Communications in Linux By John Shapley Gray ,Chapter 6, Section 6.1-4