Message Queues

Page No.:
Date: / /20

~~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 (+)ve ~~[illegible]~~ long integer type field, a non-negative length, & the actual data bytes (corresponding to length), all of which are specified to msgsnd when the message is added to a queue.

* Message 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.

Page No.:
Date: / /20

* Each queue has the following msqid_ds structure associated with it :-

```
struct msqid_ds {
        struct ipc_perm   msg_perm;
        msgqnum_t    msg_qnum; // no. of messages on queue
        msglen_t    msg_qbytes; // max no. of bytes on queue
        pid_t       msg_lspid; // Pid of last msgsnd()
        pid_t       msg_lrpid; // Pid of last msgrcv()
        time_t      msg_stime; // last-msgsnd() time
        time_t      msg_rtime; // last-msgrcv() time
        time_t      msg_ctime; // last-change time
        :
        :
};
```

This structure defines the current status of queue.

Page No.:
Date: / /20

ipcPerm Structure with each IPC structure. This structures defines the permissions & owner and includes at least the following members :-

```
Struct ipc_perm {
        uid_t  uid;  // owners effective userid
        gid_t  gid;  // owners eff. group id
        uid_t  cuid; // Creator's effective userid
        gid_t  cgid; // Creator's effective groupid
        mode_t mode; // access modes
        :
        :
}
```

→ when a new queue is created, the following members of the msqid_ds structure are initialized:-

→ msg_qnum = 0, msg_lspid = 0,
msg_stime = 0, msg_rtime = 0

→ msg_ctime is set to the current time

→ msg_qbytes is set to system limit.

NOPC Page No.: Date: / /20

(I) msgget to either open an existing queue or create a new queue.

```
#include <sys/msg.h>
int msgget (key_t key, int flag);
```

Return → message queue ID if OK, 1 on error

(II) The msgctl function performs various operations on queue.

```
#include <sys/msg.h>
int msgctl (int msqid, int cmd, struct msqid_ds *buf)
```

→ Return : 0 if OK, 1 on error

when, cmd = IPC_STAT (fetch the msqid_ds structure for this queue storing it in the structure pointed to by buf).

cmd = IPC_SET (copy the some field from the structure pointed by the msqid_ds)

cmd = IPC_RMID (Remove the message queue from the system & any data still on queue. This removal is immediate

NDPC
Page No.:
Date: / /20

III Data is placed onto a message queue by calling msgsnd.

```
#include <sys/msg.h>
int msgsnd(int msqid, const void* ptr,
              size_t nbytes, int flag )
```

Return: 0 if OK, -1 on error

→ each message is composed of a positive long integer type field, a non-negative length (nbytes), & the actual data types (corresp. to length)

→ Messages are always placed at the end of queue.

```
struct mymesg {
        long   mtype;        // positive message type
        char mtext[512];     // message data, of length nbytes
};
```

→ The ptr argument is then a pointer to a mymesg structure.

→ The message type can be used by the receiver to fetch messages in an order than first-in, first-out.

* → When msgsnd returns successfully, the msqid_ds structure associated with the message queue is updated to indicate the Process ID that made the call (msg_lspid), the time that the call was made (msg_stime), and that one more message is on the queue (msg_qnum).

(IV) Messages are recieved from a queue by msgrcv.

```
#include <sys/msg.h>
ssize_t msgrcv(int msqid void *~~[illegible]~~ptr,
                    size_t nbytes, long type, int flag);
```

Returns; Size of data portion of message if OK, 1 on error.

about type argument :-

* type == 0 ⇒ The first message on the queue is returned

* type > 0 ⇒ The first message on the queue whose message type equals type is returned.

* type < 0 ⇒ The first message on the queue whose message type is the lowest value less than or equal to the absolute value of type is returned.

* When msgrcv succeeds, the kernel updates the msqid_ds structure associated with the message queue to indicate the caller's process ID (msg_lrpid), the time of call (msg_lrpid), & that one less message is on the queue (msg_qnum).

→ Note

- → Once message is read by msgrcv, it will automatically remove from queue.
- → There may be more than one messages of same type.