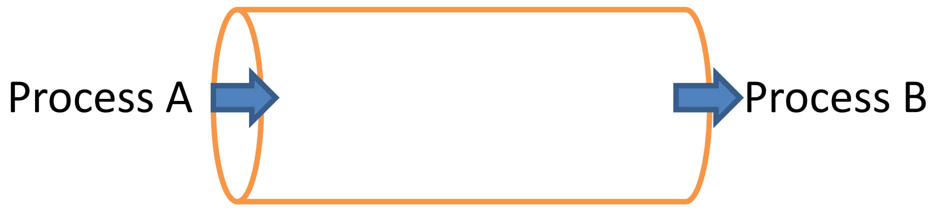
Message queue

- Inter process communication primitive
- Creates a permanent channel for communication



Create a message queue instance

int msgget(key_t key, int msgflg)

Message queue identifies

Name of the message queue

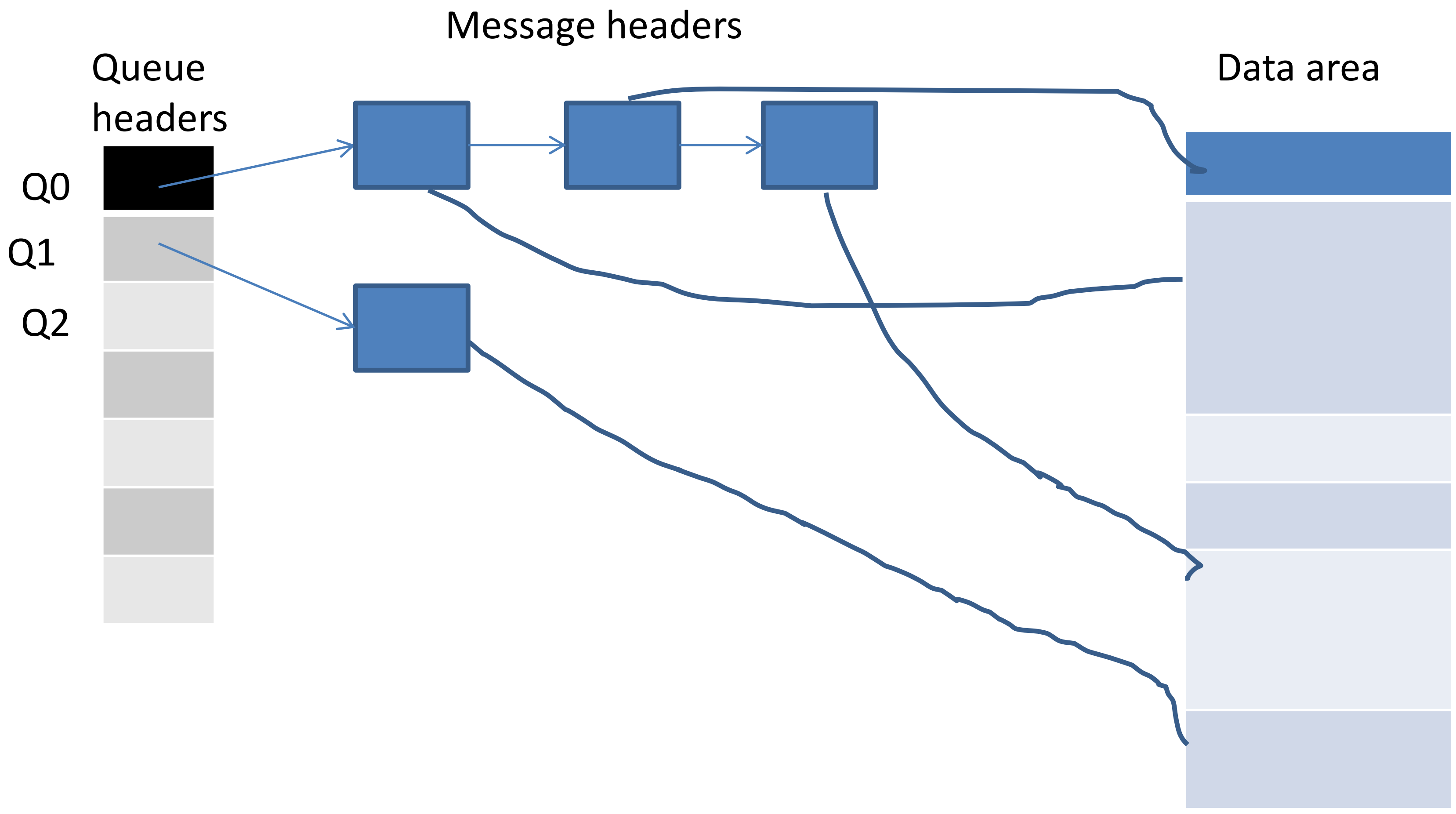
Flag (IPC_CREAT, IPC_EXCL, read, write permission)

```
int main()
{
        int msgid,len;
        key_t key;
        key=131;
        msgid=msgget(key,IPC_CREAT|0666);
        printf("\nq=%d",msgid);
}
```

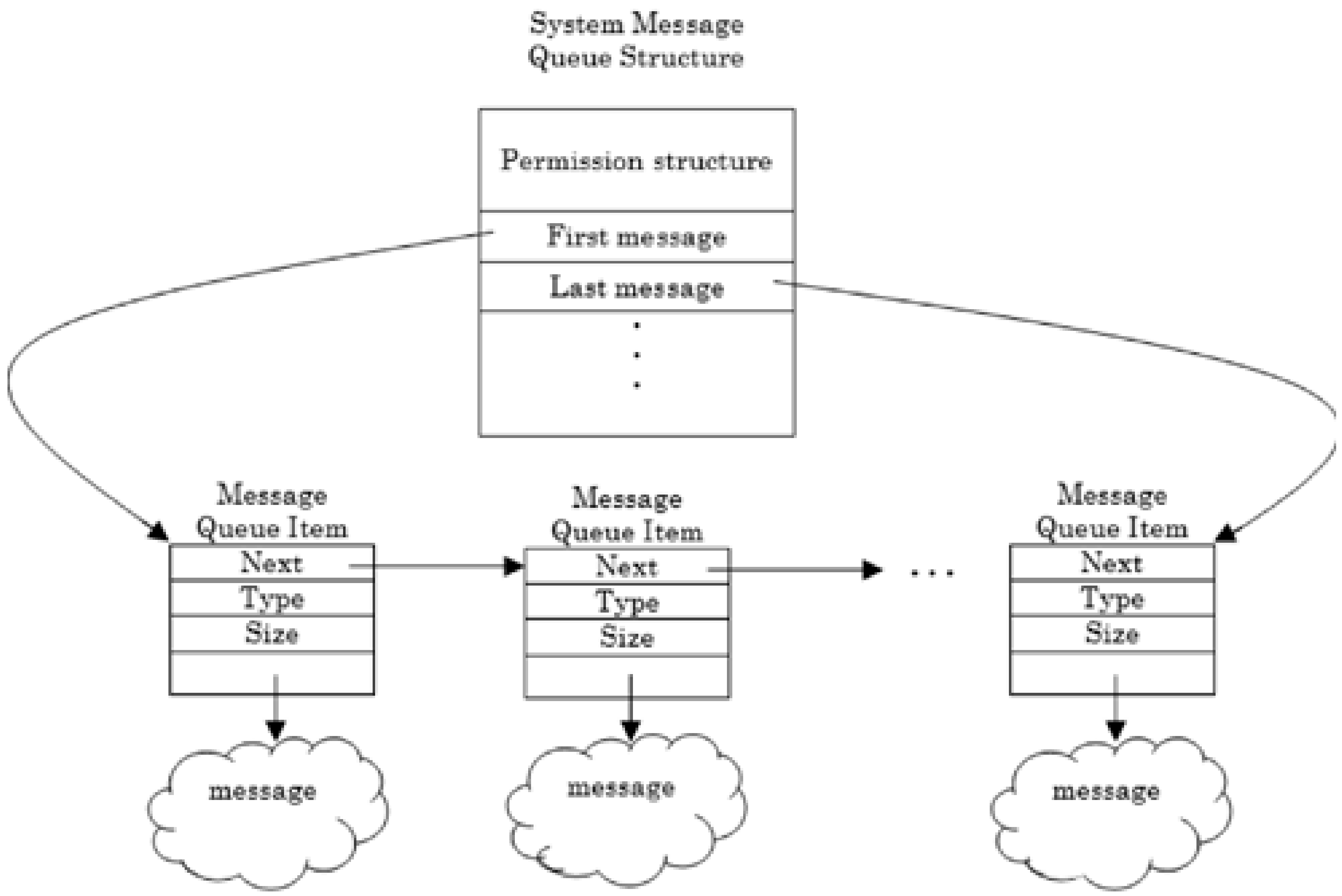
ipcs –q displays the message queue information in the system

```
Keys    MsqID   owner   permission      user bytes      messages
```

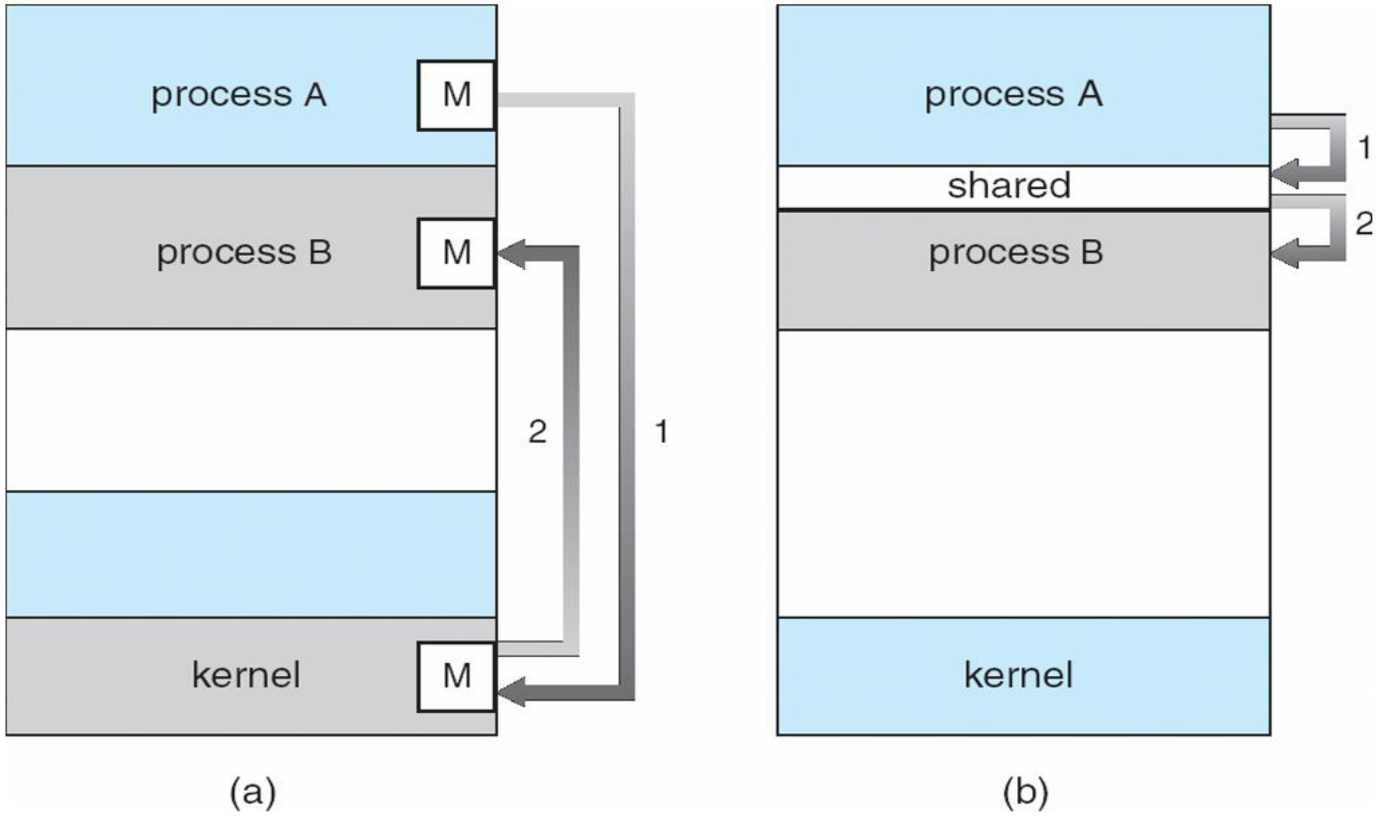
Kernel view



Kernel view


System Message Queue Structure
Permission structure
First message
Last message
Message Queue Item
Next
Type
Size
message
Message Queue Item
Next
Type
Size
message
. . .
Message Queue Item
Next
Type
Size
message

Interprocess communication



Message queue

Shared memory

msqid structure

```
/* one msqid structure for each queue on the system */
struct msqid_ds {
   struct ipc_perm msg_perm;
   struct msg *msg_first;  /* first message on queue */
   struct msg *msg_last;   /* last message in queue */
   time_t msg_stime;       /* last msgsnd time */
   time_t msg_rtime;       /* last msgrcv time */
   time_t msg_ctime;       /* last change time */
   ushort msg_cbytes;    /*current number of bytes*/
   ushort msg_qnum;      /*current number of messages*/
   ushort msg_qbytes;      /* max number of bytes on queue */
   ushort msg_lspid;       /* pid of last msgsnd */
   ushort msg_lrpid;       /* last receive pid */
};
```

```
struct ipc_perm {
key_t key;
ushort uid; /* user euid and egid */
ushort gid;
ushort cuid; /* creator euid and egid */
ushort cgid;
ushort mode; /* access modes see mode flags belo
*/
ushort seq; /* slot usage sequence number */ };
```

Message header

```
Struct msg
{       struct msg *msg_next
        long msg_type
        short msg_ts       //test size
        short msg_spot      //map address
    }
```

Message control

1. Display state of a msg queue
2. Set the parameters
3. Remove the msg queue

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

Message queue ID

IPC_STAT: status of the queue
IPC_SET: sets parameters
IPC_RMID: removes

Displays/sets the state

```
ipcrm –q <id>
```

Display state

```
int qid;
struct msqid_ds qstat;
qid=msgget((key_t)131,IPC_CREAT);
if(qid==-1)
{
		perror("msg failed\n");
		exit(1);
}
if(msgctl(qid,IPC_STAT,&qstat)<0)
{
		perror("msgctl failed");
		exit(1);
}
printf("\n%d msg in q",qstat.msg_qnum);
printf("last msg send by process %d",qstat.msg_lspid);
printf("last msg receved by process %d",qstat.msg_lrpid);
printf("current number of bytes on queue %d",qstat.msg_cbytes);
printf("max number of bytes %d",qstat.msg_qbytes);
```

qstat.msg_perm.cuid
qstat.msg_perm.cuid

qstat.msg_perm.mode=>**octal**

qstat.msg_stime
qstat.msg_rtime

⇩

time_t=> use ctime()

Set state

```
int main()
{

        int qid;
        struct msqid_ds qstat;
        qid=msgget((key_t)131,IPC_CREAT);
        if(qid==-1)
        {
                perror("msg failed\n");
                exit(1);
        }
        if(msgctl(qid,IPC_STAT,&qstat)<0)
        {
                perror("msgctl failed");
                exit(1);
        }
        printf("\n%d msg in q",qstat.msg_qnum);
        printf("last msg send by process %d",qstat.msg_lspid);
        printf("last msg receved by process %d",qstat.msg_lrpid);
        printf("current number of bytes on queue %d",qstat.msg_cbytes);
        printf("max number of bytes %d",qstat.msg_qbytes);
}
```

```
qstatus.msg_qbytes=5120
qstatus.msg_perm.mode=0644
msgctl(qid,IPC_SET,&qstatus);
```

Remove

```
int main()
{

        int qid;
        struct msqid_ds qstat;
        qid=msgget((key_t)131,IPC_CREAT);
        if(qid==-1)
        {
                perror("msg failed\n");
                exit(1);
        }
        if(msgctl(qid,IPC_STAT,&qstat)<0)
        {
                perror("msgctl failed");
                exit(1);
        }
        printf("\n%d msg in q",qstat.msg_qnum);
        printf("last msg send by process %d",qstat.msg_lspid);
        printf("last msg receved by process %d",qstat.msg_lrpid);
        printf("current number of bytes on queue %d",qstat.msg_cbytes);
        printf("max number of bytes %d",qstat.msg_qbytes);
}
```

msgctl(qid, IPC_RMID, NULL)

Removes the message queue

```
ipcrm –q <id>
```

Sending message

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

Queue ID

Message content

Message size

Flag
0,
IPC_NOWAIT

```
Struct message
{
            long mtype;
            char mtext[15];
}
```

Kernel checks

- Sending process has write permission
- Msg length does not exceed
- Queue has space
- Type is positive

The **msgsnd**() system call appends a copy of the message pointed to by *msgp* to the message queue whose identifier is specified by *msqid*.

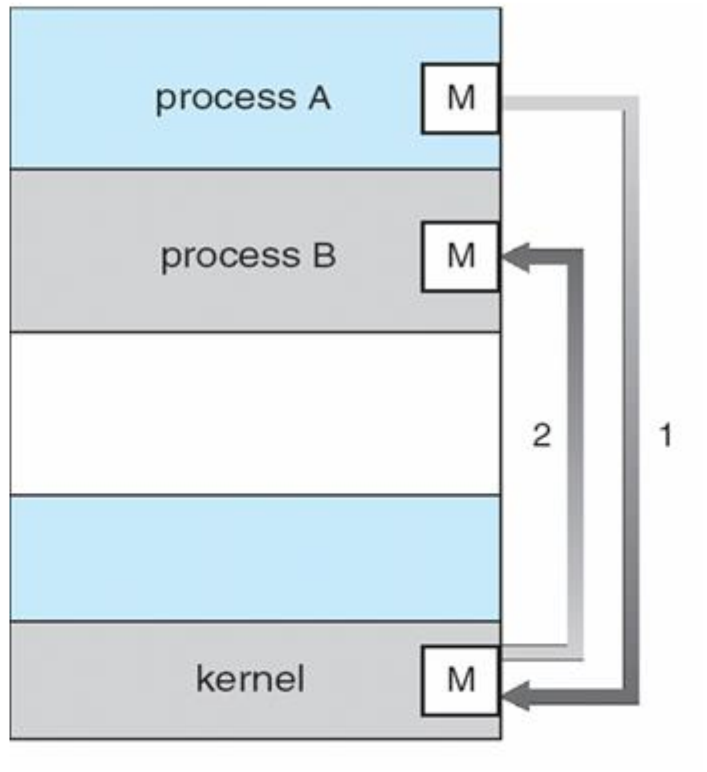
Sending message

```
struct message
{
		long mtype;
		char mtext[15];
};
int main()
{
int msgid,len;
key_t key;
struct message msg;
key=131;
msgid=msgget(key,IPC_CREAT|0666);
printf("\nq=%d",msgid);
strcpy(msg.mtext,"hello world\n");          //User memory space
msg.mtype=1;                                //User memory Space
len=strlen(msg.mtext);
if(msgsnd(msgid,&msg,len,0)==-1) //User to Kernel memory space
{
		printf("error\n");
		exit(1);
}
```

type

Message that you want to send.
Choose the size whatever you want.

Interprocess communication



Message queue

Receiving message

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

Msg Queue ID

Message content

Msg size

The **msgrcv**() system call removes a message from the queue specified by *msqid* and places it in the buffer pointed to by *msgp*.

Flag
MSG_NOERROR => If actual message length is greater than msgsz, receive the message with **Truncation**
Else, return without receiving-> error
If no message, wait

IPC_NOWAIT

IPC_EXCEPT

Type:
If x=0, first message in the queue is retrieved
x>0, first message with type x will be retrieved
x<0 ??

Receiving message

```
struct message
{
      long mtype;
      char mtext[15];
};
int main()
{             int msgid,len=20;
              key_t key;
struct message buff;

key=131;
msgid=msgget(key,IPC_CREAT|0666);
printf("\nq=%d",msgid);


if(msgrcv(msgid,&buff,len,0,0)==-1)            //Kernel to user memory space
{
      perror("msgrv failed\n");
      exit(1);
}
   printf("\nmsg received %s",buff.mtext);    //User memory space

}
```