CHAPTER IV: MESSAGE QUEUES

Unix IPC

Unix has three major IPC constructs to facilitate interaction between processes:

- Message Queues (this PowerPoint document)
  - permit exchange of data between processes
- Semaphores
  - can be used to implement critical-section problems; allocation of resources
- Shared Memory
  - an area of memory accessible by multiple processes.

IPC System Calls

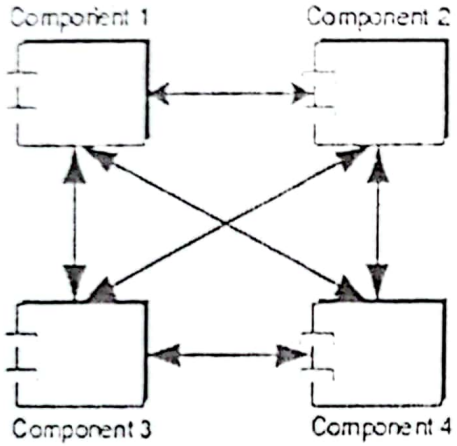
Functionality	Message Queue	Semaphore	Shared
Allocate IPC	***msgget***	***semget***	***shmget***
Access IPC	***msgsnd*** ***msgrcv***	***semop***	***shmat*** ***shmdt***
IPC Control	***msgctl***	***semctl***	***shmctl***

Message Queues

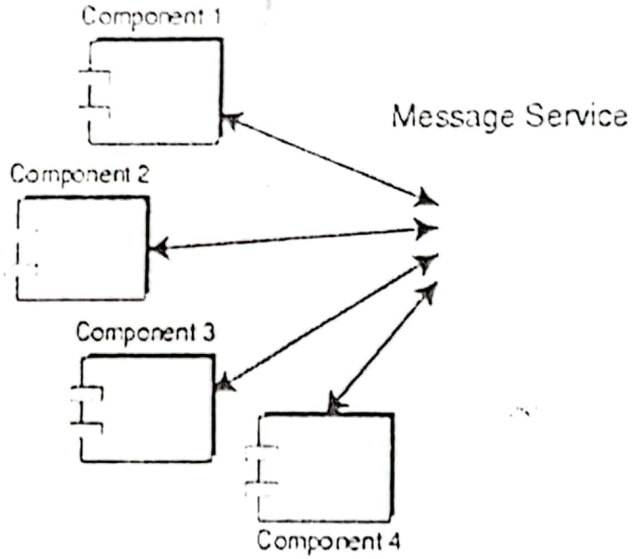
- Creating a Message Queue
- Message Queue Control
- Message Queue Operations
- IPC Call
- Client-Server Example

Messaging Methods

Peer to Peer Messaging



Centralized Messaging



Message Queues

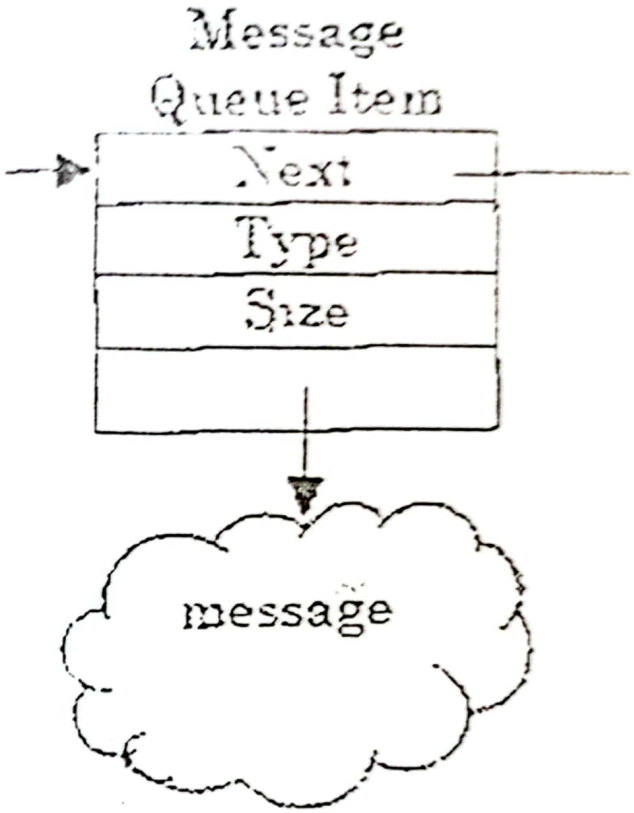
- One process establishes a message queue that others may access. Often a server will establish a message queue that multiple clients can access
- Features of Message Queues
  - A process generating a message may specify its type when it places the message in a message queue.
  - Another process accessing the message queue can use the message type to selectively read only messages of specific type(s) in a first-in-first-out manner.
  - Message queues provide a user with a means of multiplexing data from one or more producer(s) to one or more consumer(s).

Attributes of Message Queues

- A conceptual view of a message queue, from *Interprocess Communications in Unix*:
  - The attributes of each element on the queue:
    - long integer ***type***;
    - ***size*** of the data portion of the message (can be zero);

* A message queue element then has one more field:
  - ***data*** (if the length is greater than zero)

* Message Structure



Message Queue Structure

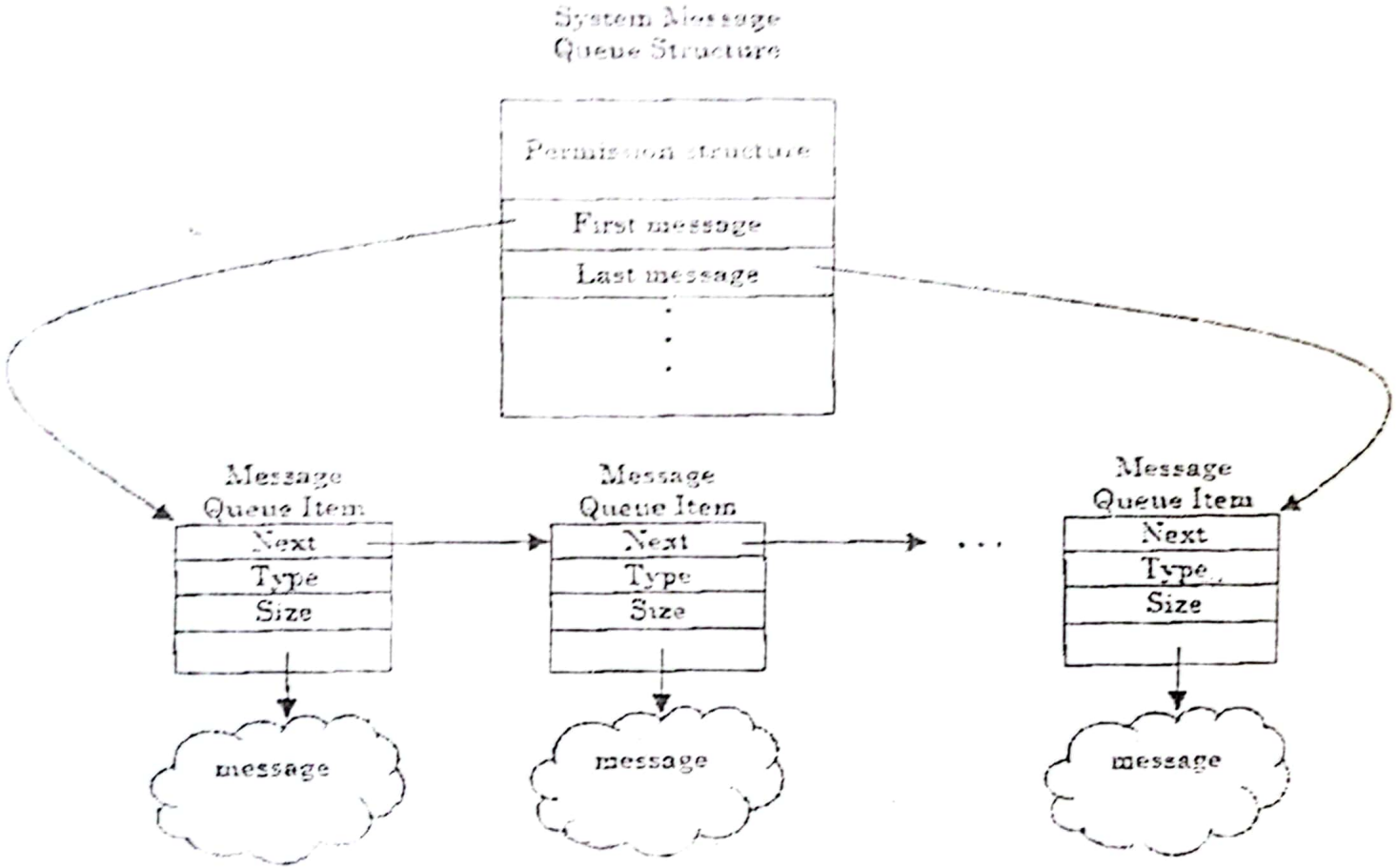


Figure 6.5 A message queue with *N* items.

msqid_ds Structure

```
struct msqid_ds {
   struct ipc_perm  msg_perm; /*operation permission struct */
   struct msg  *msg_first;  /* pointer to first message on q*/
   struct msg  *msg_last;  /* point to last message on q */
   ulong          msg_cbytes; /* current # bytes on q */
   ulong          msg_qnum;  /* # of message on q */
   ulong          msg_qbytes; /* max # of bytes on q */
   pid_t          msg_lspid;    /* pid of last msgsnd */
   pid_t          msg_lrpid;   /* pid of last msgrcv */
   time_t         msg_stime;   /* last msgsnd time  */
  .....
};  /* total 17 members */
```

ipc_perm Structure

```
* struct ipc_perm {
        uid_t           uid;    /*owner's user ID */
        gid_t           gid;    /* owner's group ID */
        uid_t           cuid;   /* creator's user ID */
        gid_t           cgid;   /* creator's group ID */
        mode_t          mode;   /* access modes */
        ulong           seg;    /* slot usage sequence number */
        key_t           key;    /* key */
        long            pad[4];         /* reserve area */
    };
* Struct msqid_ds {
    struct      ipc_perm        msg_perm; .....};
```

msg structure

```
struct msg {
    struct msg    *msg_next;  /* pointer to next message on q */
    long           msg_type;  /*  message type */
    ushort         msg_ts;    /*  message text size */
    short          msg_spot;  /* message text map address */
};
```

msgget System Call

Create a message queue.

- Includes: <sys/types.h> <sys/ipc.h> <sys/msg.h>
- Command: ***int msgget(key_t key, int msgflg);***
  - Returns: Success: message queue identifier ;

    Failure: -1;
  - Arguments:
    - ***key***: to be specified directly by the user or generated using ftok. We will use a function getuid() to generate unique, consistent message queues for each person
    - ***msgflg***: IPC_CREAT, IPC_EXCL or permission value.

Example of generating a message queue: Qcreate.cpp

msgsnd System Call (Message Queue Operation)

- Function: to place (send) message in the message queue.
- Include: <sys/types.h> <sys/ipc.h> <sys/msg.h>
- Summary:

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

  - Returns: Success: 0; Failure: -1; Sets errno: Yes
  - Arguments

    –*int msgid*: valid message queue identifier

    –*const void *msgp*: address of the message to be sent

    –*size_t msgsz*: the size of the message to be sent.

    –*int msgflg*: Two possible values:

    - 0: Block, if the message queue is full
    - IPC_NOWAIT : don't wait if message queue is full

msgrcv System Call (Message Queue Operation)

- Function: to retrieve message from the message queue.
  - Include <sys/types.h> <sys/ipc.h> <sys/msg.h>
  - *int msgrcv (int msqid, void *msgp,*
    size_t msgsz, long msgtyp, int msgflg);
    - Return: Success: number of bytes actually received;
      Failure: -1; Sets errno: Yes
- Arguments:
  - *int msqid*: the message queue identifier.
  - *void *msgp*: a point to received message location (structure).
  - *size_t msgsz*: the maximum size of the message in bytes.
  - *long msgtype*: the type of the message to be retrieved.
  - *int msgflg*: to indicate what action should be taken.
    - 0: error if size of message exceeds msgsz
    - MSG_NOERROR: if size of message exceeds msgsz, accept msgsz bytes
    - IPC_NOWAIT: return –1 with errno set to ENOMSG

Type of Message (*msgrcv* System Call)

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

- *long msgtype*: the type of the message to be retrieved.
  - Actions for *msgrcv* as indicated by *msgtyp* value

msgtyp value	action
0	return first (oldest) message on queue
> 0	return first message with type equal to msgtyp
< 0	return the first message with lowest type less than or equal to msgtyp

msgctl System Call - Message Queue Control

Function

Ownership and access permissions, established when the message queue was created, can be examined and modified using the *msgctl* system call.

Include

< sys/types.h> <sys/ipc.h> < sys/msg.h>

Command:

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

Return

Success: 0; Failure: -1; Sets errno: Yes

Remove a Message Queue in a Program

Command:

msgctl (msqid, IPC_RMID, (struct msqid_ds *) 0);

- To remove the message queue with key *msqid*.
- You must be the owner

ipcs & ipcrm Command

- *% ipcs – display ipc structures active in system*

```
T   ID   KEY        MODE       OWNER   GROUP   QBYTES
Message Queues:
q   50   0X67028a01  -Rrw-rw----  gray    other   4096
Shared Memory facility not in system
T   ID   KEY        MODE       OWNER   GROUP   NSEMS
Semaphores:
s   0    0X000187cf  --ra-ra-ra-  root    root    2
s   1    0X000187ce  --ra-ra-ra-  root    root    1
```

* *% ipcrm -q 50 (ipc remove)*

-q tells icprm that a message queue (ID 50) is to be removed.