Module 9

Short Messages

Overview

- Objectives
- Relevance

Named Pipes

`mymkfifo.c`

```
#include <sys/types.h>
#include <sys/stat.h>

#define DIE(x) perror(x),exit(1)

main() {

  if(mknod("FIFO1",S_IFIFO | 0666,0) == -1)
    DIE("FIFO1");

  if(mkfifo("FIFO2",0666) == -1)
    DIE("FIFO2");
}
```

Named Pipes

`talker.c:`

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

main() {

  int pd, n;
  char msg[] = "Hi, Folks!";

  printf("Talker's here.\n");

  /* open the pipe for write */
  pd = open("./mypipe", O_WRONLY);
  if( pd == -1) {
    perror("open");
    exit(1);
  }

  n = write(pd, msg, strlen(msg) + 1 );
  if( n == -1) {
    perror("write");
    exit(1);
  }

  close(pd);
  return(0);
}
```

Named Pipes

`listener.c:`

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define BUFSIZE 80

main() {

  int pd, n = 0;
  char inmsg[BUFSIZE];
  char *introMessage = “Listener heard:”;

  /* open the pipe for read */
  pd = open("./mypipe", O_RDONLY);
```

```
  if( pd == -1) {
    perror("open");
    exit(1);
  }

  write(STDOUT_FILENO, introMessage, strlen(introMessage));

  while( (n = read(pd, inmsg, BUFSIZE) ) > 0 ) {
    write(STDOUT_FILENO, inmsg, n );
  }

  if( n == -1 ) {
    perror("read");
    exit(1);
  }

  write(STDOUT_FILENO,"\n", 1 );
  close(pd);
  return(0);
}
```

Message Queues

- `msgget()`– Gets messages
- `msgsnd()`– Sends messages
- `msgrcv()`– Receives messages
- `msgctl()`– Removes objects, get and set status, and control information

Message Queues

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFSIZE 80

main() {

  struct msgbuf *sendmsg;
  struct msgbuf *recvmsg;
  int msgqid;
  int rval = 0;

  sendmsg = (struct msgbuf *)
              malloc(sizeof(long)+BUFSIZE);
```

```
  recvmsg = (struct msgbuf*)
              malloc(sizeof(long)+BUFSIZE);

  if (sendmsg == NULL || recvmsg == NULL) {
    fprintf(stderr, "Out of virtual memory.\n");
    exit(1);
  }
  msgqid = msgget( IPC_PRIVATE, IPC_CREAT | 0600);

 if (msgqid == -1) {
   perror("msgget");
   exit(1);
  }
  system("ipcs");
  sendmsg->mtype = 1;
  (void)strcpy(sendmsg->mtext, "Test messages.");

  if(msgsnd(msgqid,sendmsg,strlen(sendmsg->mtext)+1,0) == -1){
    perror("msgsnd");
    rval = 1;
```

```
    /* "out" is a label near the end of main */
    goto out;
  }

  if (msgrcv(msgqid, recvmsg, BUFSIZE, 1L, 0) == -1) {
    perror("msgrcv");
    rval = 1;
    goto out;
  }
  printf("\nReceived message *%s*\n\n", recvmsg->mtext);

out:
  msgctl(msgqid, IPC_RMID, (struct msqid_ds *)NULL);
  system("ipcs");
  exit(rval);
}
```

Message Queues

```
$ mymsgqueues
IPC status from <running system> as of Sun Mar  7 13:22:11 MST 1999
T         ID      KEY        MODE        OWNER    GROUP
Message Queues:
q          0   0          --rw-------      mrh    staff
Shared Memory:
m          0   0x5001821c --rw-r--r--     root     root
Semaphores:

Received message *Test messages.*

IPC status from <running system> as of Sun Mar  7 13:22:11 MST 1999
T         ID      KEY        MODE        OWNER    GROUP
Message Queues:
Shared Memory:
m          0   0x5001821c --rw-r--r--     root     root
Semaphores
```

Exercise: Short Messages

- Objectives
- Tasks
- Discussion
- Solutions