P1.c

1 #include<stdio.h>

2 main()

3 {

4 int fd[2];

5 if(pipe(fd)<0)

6 {

7 perror("pipe");

8 return;

9 }

10 if(fork()==0)

11 {

12 int i;

13 char b[20];

14 read(fd[0],b,sizeof(b));

15 for(i=0;b[i];i++)

16 b[i]=b[i]^32;

17 write(fd[1],b,strlen(b)+1);

18 }

19 else

20 {

21 char a[20];

22 printf("enter data...\n");

23 scanf("%s",a);

24 write(fd[1],a,strlen(a)+1);

25 sleep(1);

26 read(fd[0],a,sizeof(a));

27 printf("%s\n",a);

28 }

29 }

P2.c

1 //process got blocked once pipe is full

2 //finding the size of the pipe

3 #include<stdio.h>

4 main()

5 {

6 char ch='a';

7 int count=0;

8 int fd[2];

9 pipe(fd);

10 printf("writing into pipe..\n");

11 while(write(fd[1],&ch,1)>0)

12 {

13 count++;

14 printf("count=%d\n",count);

15 }

16 //printf("writing done..\n");

17 }

P3.c

1 //finding the size of the pipe using pipe2 system call

2 #include<stdio.h>

3 #include<fcntl.h>

4 main()

5 {

6 char ch='a';

7 int count=0,ret;

8 int fd[2];

9 pipe2(fd,O\_NONBLOCK);//O\_NON BLOCK when pipe is full process not blocked but

10 //write system call failed

11 printf("writing into pipe..\n");

12

13 while(ret=write(fd[1],&ch,1)>0)

14 count++;

15

16 printf("ret:%d\n",ret);

17

18 printf("writing done...\n");

19 printf("size of the pipe is=%d\n",count);

20 }

P4.c

1 /\* fifo - first-in first-out special file, named pipe

2 DESCRIPTION

3 A FIFO special file (a named pipe) is similar to a pipe, except that it is

4 accessed as part of the file system. It can be opened by multiple processes for reading or writing. When processes are exchanging data via the FI FO, the

5 kernel passes all data internally without writing it to the file

6 system.Thus,the FIFO special file has no contents on the file system;the file

7 system entry merely serves as a reference point so that processes

8 can access the pipe using a name in the file system.\*/

9

10 /\*The FIFO must be opened on both ends (reading and writing) before data can be passed. Normally, opening the FIFO blocks until the other end is opened

11 also\*/

12

13 /\*how to create named pipe

14 1.using mkfifo(commnad) or using mkfifo() function

15 2.using mknod function\*/

16 /\*

17 mkfifo - make a FIFO special file (a named pipe)

18 int mkfifo(const char \*pathname, mode\_t mode);

19 DESCRIPTION:mkfifo() makes a FIFO special file with name pathname. mode specifies the FIFO's permissions. It is modified by the process's umask in the

20 usual way: the permissions of the created file are (mode & ~umask)\*/

P5.c

1)

1 //wap to create the named pipe

2 #include<stdio.h>

3 #include<fcntl.h>

4 main()

5 {

6 int fd;

7 mkfifo("np1",0644);

8 perror("mkfifo");

9

10 printf("hello...\n");

11 fd=open("np1",O\_WRONLY);

12 if(fd<0)

13 {

14 perror("open");

15 return;

16 }

17 printf("by..\n");

18 printf("fd:%d\n",fd);

19 }

P6.c

2)

1 //wap to create the named pipe

2 #include<stdio.h>

3 #include<fcntl.h>

4 main()

5 {

6 int fd;

7 mkfifo("np1",0644);

8 perror("mkfifo");

9

10 printf("banglore..\n");

11 fd=open("np1",O\_RDONLY);

12 if(fd<0)

13 {

14 perror("open");

15 return;

16 }

17 printf("vector..\n");

18 printf("fd:%d\n",fd);

19 }