How to use a pipe

A pipe is a system call that creates a unidirectional communication link between two file descriptors. The *pipe* system call is called with a pointer to an array of two integers. Upon return, the first element of the array contains the file descriptor that corresponds to the output of the pipe (stuff to be read). The second element of the array contains the file descriptor that corresponds to the input of the pipe (the place where you write stuff). Whatever bytes are sent into the input of the pipe can be read from the other end of the pipe.

Example

This is a small program that gives an example of how a pipe works. The array of two file descriptors is fd[2]. Whatever is written to fd[1] will be read from fd[0].

```
/*
	The simplest example of pipe I could think of
	Paul Krzyzanowski
*/

#include <stdlib.h>
#include <stdio.h>	/* for printf */
#include <string.h>	/* for strlen */

int
main(int argc, char **argv)
{
	int n;
	int fd[2];
	char buf[1025];
	char *data = "hello... this is sample data";

	pipe(fd);
	write(fd[1], data, strlen(data));
	if ((n = read(fd[0], buf, 1024)) >= 0) {
		buf[n] = 0;	/* terminate the string */
		printf("read %d bytes from the pipe: \"%s\"\n", n, buf);
	}
	else
		perror("read");
	exit(0);
}
```

Compile this program via:

```
gcc -o pipe-simple pipe-simple.c
```

If you don't have gcc, You may need to substitute the gcc command with cc or another name of your compiler.

Run the program:

```
./pipe-simple
```

Developing Application using Inter Process communication (using shared memory, pipes or message queues)

AIM:

To write a program for developing Application using Inter Process communication with pipes.

ALGORITHM:

1. Start the program.
2. Read the input from parent process and perform in child process.
3. Write the date in parent process and read it in child process.
4. Fibonacci Series was performed in child process.
5. Stop the program.

PROGRAM:

```
#include<stdio.h>
#include<unistd.h>
#include<sys/ipc.h>
#include<sys/uio.h>
#include<sys/types.h>
main()
{
int pid,pfd[2],n,a,b,c;
if(pipe(pfd)==-1)
{
printf("\nError in pipe connection\n");
exit(1);
}
pid=fork();
if(pid>0)
{
printf("\nParent Process");
printf("\n\n\tFibonacci Series");
printf("\nEnter the limit for the series:");
scanf("%d",&n);
close(pfd[0]);
write(pfd[1],&n,sizeof(n));
close(pfd[1]);
exit(0);
}
else
{
close(pfd[1]);
read(pfd[0],&n,sizeof(n));
printf("\nChild Process");
a=0;
b=1;
close(pfd[0]);
printf("\nFibonacci Series is:");
printf("\n\n%d\n%d",a,b);
while(n>2)
{
c=a+b;
printf("\n%d",c);
a=b;
b=c;
n--;
}
}
}
```

OUTPUT:

```
[root@localhost ~]# ./a.out
Parent Process
Fibonacci Series
Enter the limit for the series:5
Child Process Fibonacci Series is: 01123
```

RESULT:

Thus the program for developing Application using Inter Process communication with pipes is written and successfully executed.

INTER PROCESS COMMUNICATION USING PIPE - SORTING - SYSTEMS LAB

Program:

```
#include<stdio.h>
main()
 {
     int i,j,n,n1;
     int pid;
     int a[5],b[5],c[5];
     int p1[2],p2[2];
      pipe(p1);
```

```
    pipe(p2);
    pid=fork();
    if(pid==0)
    {
    printf("enter the limit \n");
            fflush(stdin);
           scanf("%d",&n);

               close(p1[0]);
        write(p1[1],&n,sizeof(int));
   close(p2[1]);
   read(p2[0],b,5*sizeof(int));
    for(i=0;i<n;i++)
        printf("%d\t",b[i]);


        }
        else
        {
         close(p1[1]);
   read(p1[0],&n1,sizeof(int));
   printf("enter the numnbers");
             for(i=0;i<n1;i++)
               scanf("%d",&a[i]);

    for(i=0;i<n1;i++)
    for(j=0;j<n1-i-1;j++)
     if(b[j]>b[j+1])
     {
      int t=b[j];
      b[j]=b[j+1];
      b[j+1]=t;
     }

   close(p2[0]);
   write(p2[1],a,5*sizeof(int));
  }
 }

#include<stdio.h>
main()
```

```
{
      int i,j,n,n1;
      int pid;
      int a[5],b[5],c[5];
      int p1[2],p2[2];
       pipe(p1);
      pipe(p2);
      pid=fork();
      if(pid==0)
      {
      printf("enter the limit \n");
            fflush(stdin);
           scanf("%d",&n);

               close(p1[0]);
        write(p1[1],&n,sizeof(int));
    close(p2[1]);
    read(p2[0],b,5*sizeof(int));
      for(i=0;i<n;i++)
        printf("%d\t",b[i]);


        }
        else
        {
         close(p1[1]);
    read(p1[0],&n1,sizeof(int));
    printf("enter the numnbers");
             for(i=0;i<n1;i++)
               scanf("%d",&a[i]);

      for(i=0;i<n1;i++)
      for(j=0;j<n1-i-1;j++)
       if(b[j]>b[j+1])
       {
         int t=b[j];
         b[j]=b[j+1];
         b[j+1]=t;
        }

    close(p2[0]);
    write(p2[1],a,5*sizeof(int));
  }
 }
```

Output:

```
 nn@ubuntu:~$ gcc pb1.c
nn@ubuntu:~$ ./a.out
enter the limit 
5
enter the numnbers5
4
3
2
1
5    4    3    2    1
nn@ubuntu:~$ gcc pb1.c
```

INTER PROCESS COMMUNICATION USING PIPE- FIBONACCI - SYSTEMS LAB - C

Program:

```
#include<stdio.h>
main()
{
    int pid;
    int p1[2],p2[2];
    pipe(p1);
    pipe(p2);
    int b,n,i,f1,f2;int ar[30],br[30];
    pid=fork();
    if(pid==0)
    {
        printf("enter count:");
        fflush(stdin);
        scanf("%d",&n);
        close(p1[0]);
        write(p1[1],&n,4);
        
        close(p2[1]);
        read(p2[0],br,30*sizeof(int));
        printf("\nFibonacci:\n");
        for(i=0;i<n;i++)
            printf("%d\n",br[i]);

    }
    else if(pid>0)
    {
    close(p1[1]);
    read(p1[0],&b,4);
    //printf("count is:%d",b);

    f1=0,f2=1;
    ar[0]=0;
    ar[1]=1;
    int i;
    for(i=2;i<b;i++)
    {
        int f3=f1+f2;
        f1=f2;
        f2=f3;
        ar[i]=f3;
    }
    close(p2[0]);
    write(p2[1],ar,30*sizeof(int));
    }
}
```

Output:

```
nn@ubuntu:~$ gcc pp.c
nn@ubuntu:~$ ./a.out
enter count:5
nn@ubuntu:~$ 
Fibonacci:
0
1
1
2
3
```