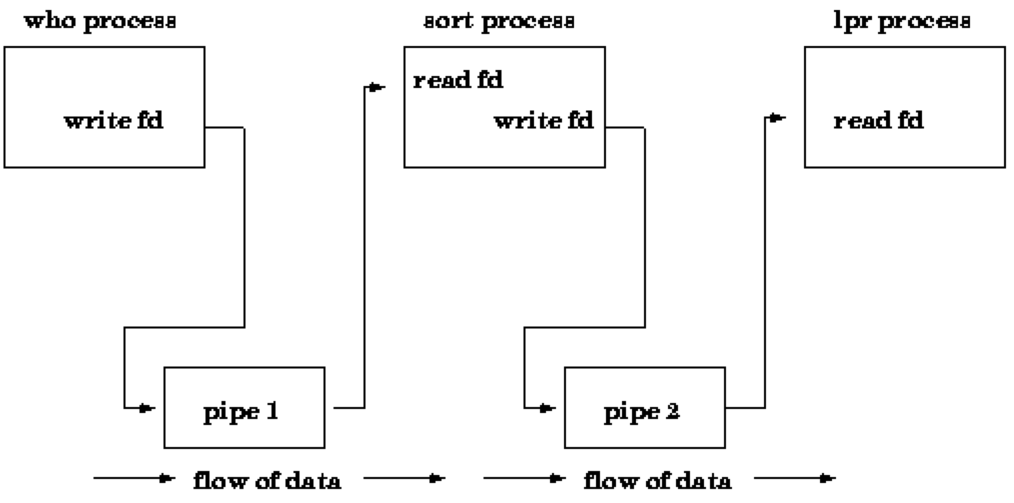
Practical 8 - Study of pipe() system call

- **Pipe()** System call

A Unix pipe() provides a one-way flow of data.

One way communication between processes.

It can use a pipe such that One process write to the pipe, and the other process reads from the pipe.

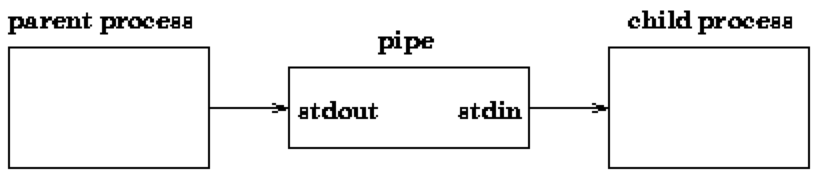


For example, if a Unix users issues the command

who | sort | lpr

then the Unix shell would create three processes with two pipes between them:

- **Pipe() with parent and child process.**



- **Pipe() system call**

  Syntax : int pipe(int *filedes*[2]);

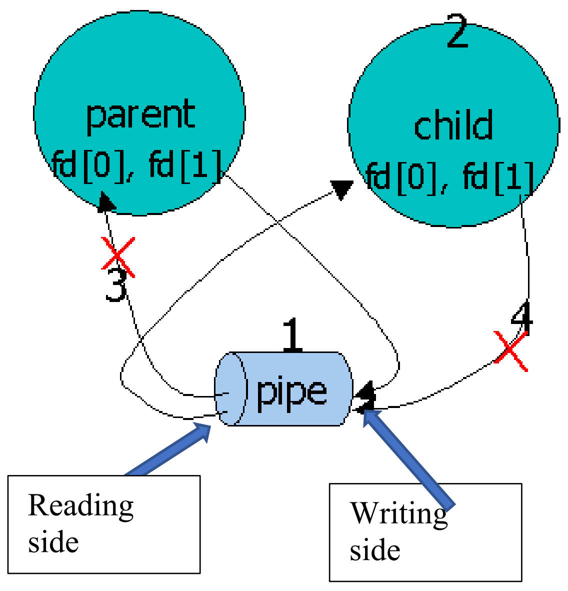
  - **pipe**() creates a pair of file descriptors, pointing to a pipe inode, and places them in the array pointed to by *filedes*.
  - filedes[0] is for reading,
  - filedes[1] is for writing.

- Two file descriptors are returned by pipe system call - fildes[0] and fildes[1], and they are both open for reading and writing respectively.
- A read from fildes[0] accesses the data written to fildes[1] on a first-in-first-out (FIFO) basis
- When a pipe is used in a Unix/Linux command line, the first process is assumed to be writing to stdout(standard output file) and the second is assumed to be reading from stdin (standard input file)

// program for parent process writing into pipe and child process reading from pipe.

```
int main( void ) {
   int n, fd[2];
   int pid;
   char line[MAXLINE];

   if (pipe(fd) < 0 )       // 1: pipe created
      perror( "pipe error" );
   else if ( (pid = fork( ) ) < 0 )      // 2: child forked
      perror( "fork error" );
   else if ( pid > 0 ) {    // parent
      close( fd[0] );       // 3: parent's fd[0] closed
      write( fd[1], "hello world\n", 12 );
   } else {                 // child
      close( fd[1] );       // 4: child's fd[1] closed
      n = read( fd[0], line, MAXLINE );
      write( 1, line, n );
   }
   exit( 0 );
}
```



• Program 1
//This program shows working of pipe() system call

```
#include<stdio.h>
#include<unistd.h>

int main()
{
        int n,fd[2];
        char buf[100];

        pipe(fd);

        printf("Writing to the pipe...\n");
        write(fd[1],"ABC",3);
        printf("Reading from the pipe...\n");
        n= read(fd[0],buf,100); // reading from pipe
        //write(STDOUT_FILENO,buf,n);
       printf("%s",buf);
exit(0);
 }
```

- **Program:2**

```
//This program shows use of pipe() system call
//Parent writes to the pipe and child reads from the pipe

#include<unistd.h>
#include<stdio.h>
#include<sys/stat.h>

int main()
{
        int n,fd[2];
        char buf[100];

        pipe(fd);

        switch(fork())
        {
                case -1:
                        printf("Fork error...\n");
                        exit(1);
                case 0:
                        close(fd[1]);
                        n=read(fd[0],buf,100);   // read the data into inputline of pipe
                        write(STDOUT_FILENO,buf,n); // similar to printf , display the
                                                        message on standard output
                        close(fd[0]);
                        break;
                default:
                        close(fd[0]);
                        write(fd[1],"writing to pipe\n",16);  // writing data into pipe
                        close(fd[1]);
        }
        exit(0);
}
```

Program 3

```
//Child enter the number array into pipe and parent process read array from pipe


#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
int main()
{
        int n,fd[2];
     int i,n1;
     int *arr ,*arr1;

    pipe(fd);
```

```
      arr = (int *) malloc(10*sizeof(int));
      arr1 = (int *) malloc(10*sizeof(int));


        switch(fork())
        {
                case -1:
                        printf("Fork error...\n");
                        exit(1);
                default :
                        sleep(1);
                        close(fd[1]); //close outputline
                        read(fd[0],arr1,sizeof(int)*10); // read number from
                                                       //    inputline
                 for(i=1;i<10;i++)
                   {   printf("%d",arr1[i]);
                    }
                    close(fd[0]);
                    break;
                case 0:
                        close(fd[0]);
                        for(i=1;i<10;i++)
                        arr[i]= i;

                  write(fd[1],arr,sizeof(int)*10);  // Write data to the output line of pipe

                        close(fd[1]);
                        break;
        }
        exit(0);
}
```

Exercise

1	WAP in which Child process pass number (for e.g 10) into pipe and Parent read number form pipe and print 1….N (for e.g 1 2 3 4 5 6 7 8 9 10)
2	WAP in which Parent process writes String into pipe and child process read string from pipe and find vowels form that string.
3	WAP in which takes two numbers form user and Child process write two numbers(take an array of two integer element) into pipe and Parent process will read two numbers and print the sum.