LAB07

PIPES

P.PRIYADHARSHINI

2019103562

29-03-2021

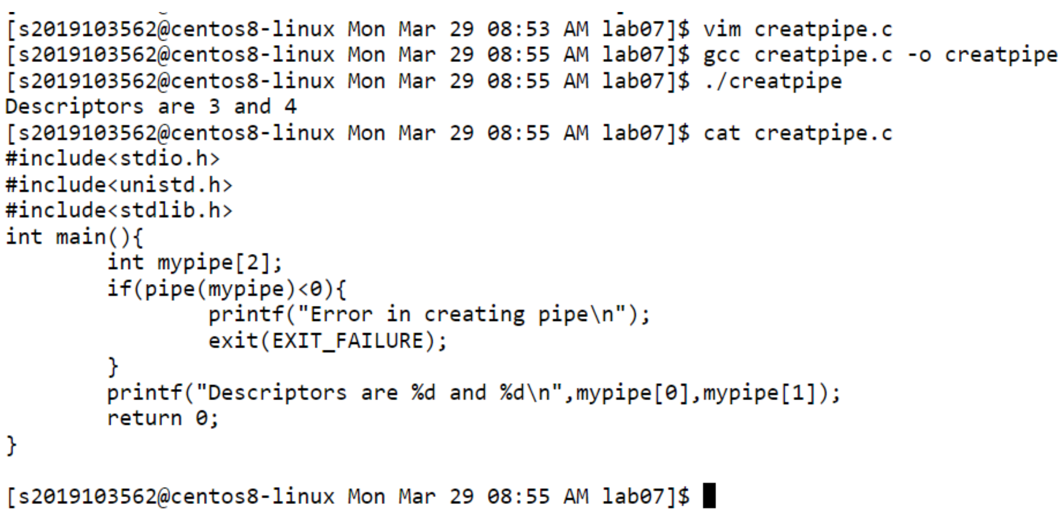
1.UNNAMED PIPES

2.TWO WAY COMMUNICATION

3.NAMED PIPES

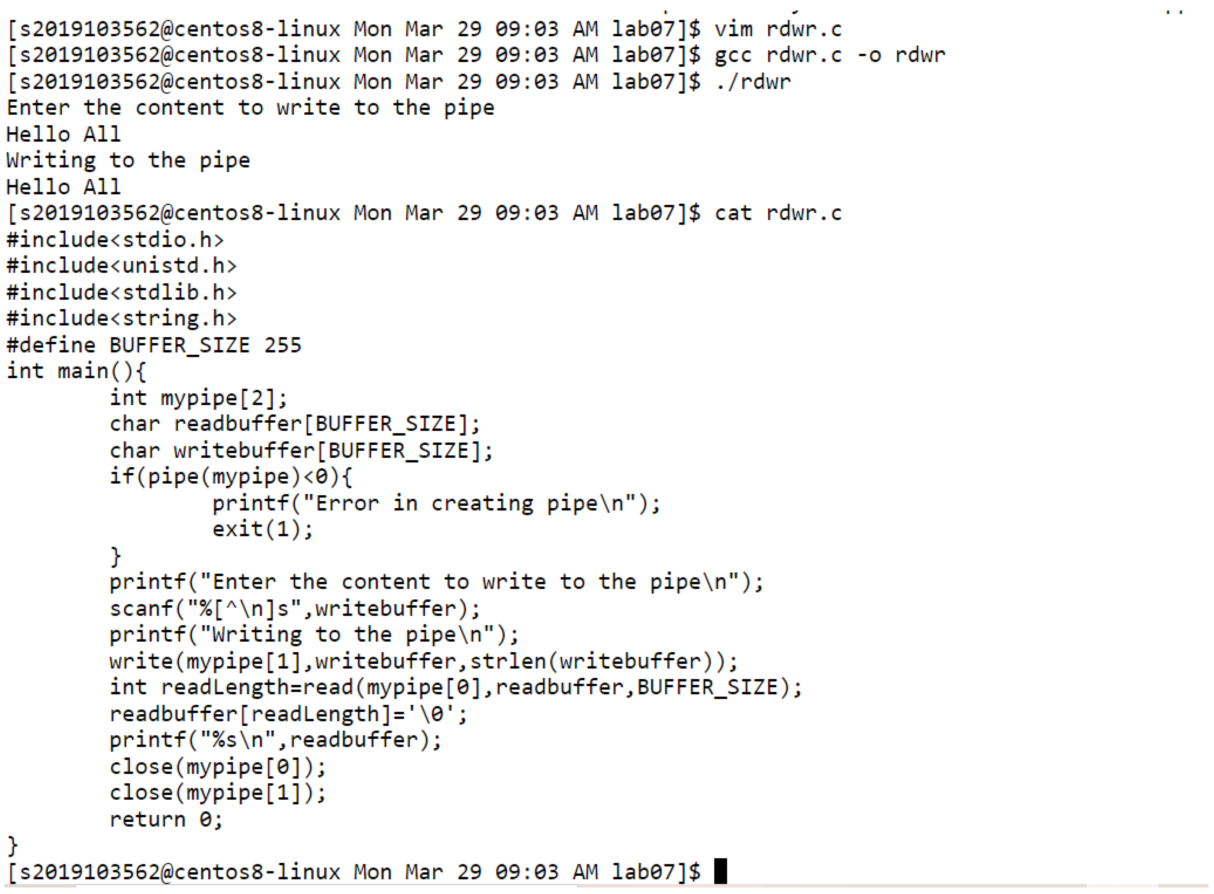
UNNAMED PIPES:

PIPE CREATION:



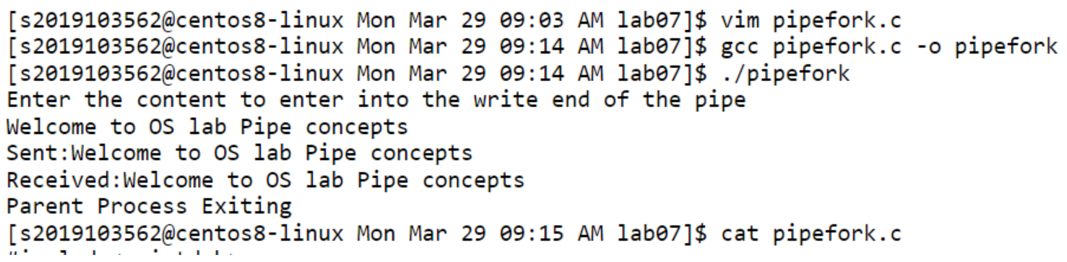
```
[s2019103562@centos8-linux Mon Mar 29 08:53 AM lab07]$ vim creatpipe.c
[s2019103562@centos8-linux Mon Mar 29 08:55 AM lab07]$ gcc creatpipe.c -o creatpipe
[s2019103562@centos8-linux Mon Mar 29 08:55 AM lab07]$ ./creatpipe
Descriptors are 3 and 4
[s2019103562@centos8-linux Mon Mar 29 08:55 AM lab07]$ cat creatpipe.c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main(){
        int mypipe[2];
        if(pipe(mypipe)<0){
                printf("Error in creating pipe\n");
                exit(EXIT_FAILURE);
        }
        printf("Descriptors are %d and %d\n",mypipe[0],mypipe[1]);
        return 0;
}

[s2019103562@centos8-linux Mon Mar 29 08:55 AM lab07]$ 
```

READ AND WRITE ENDS OF THE PIPE:

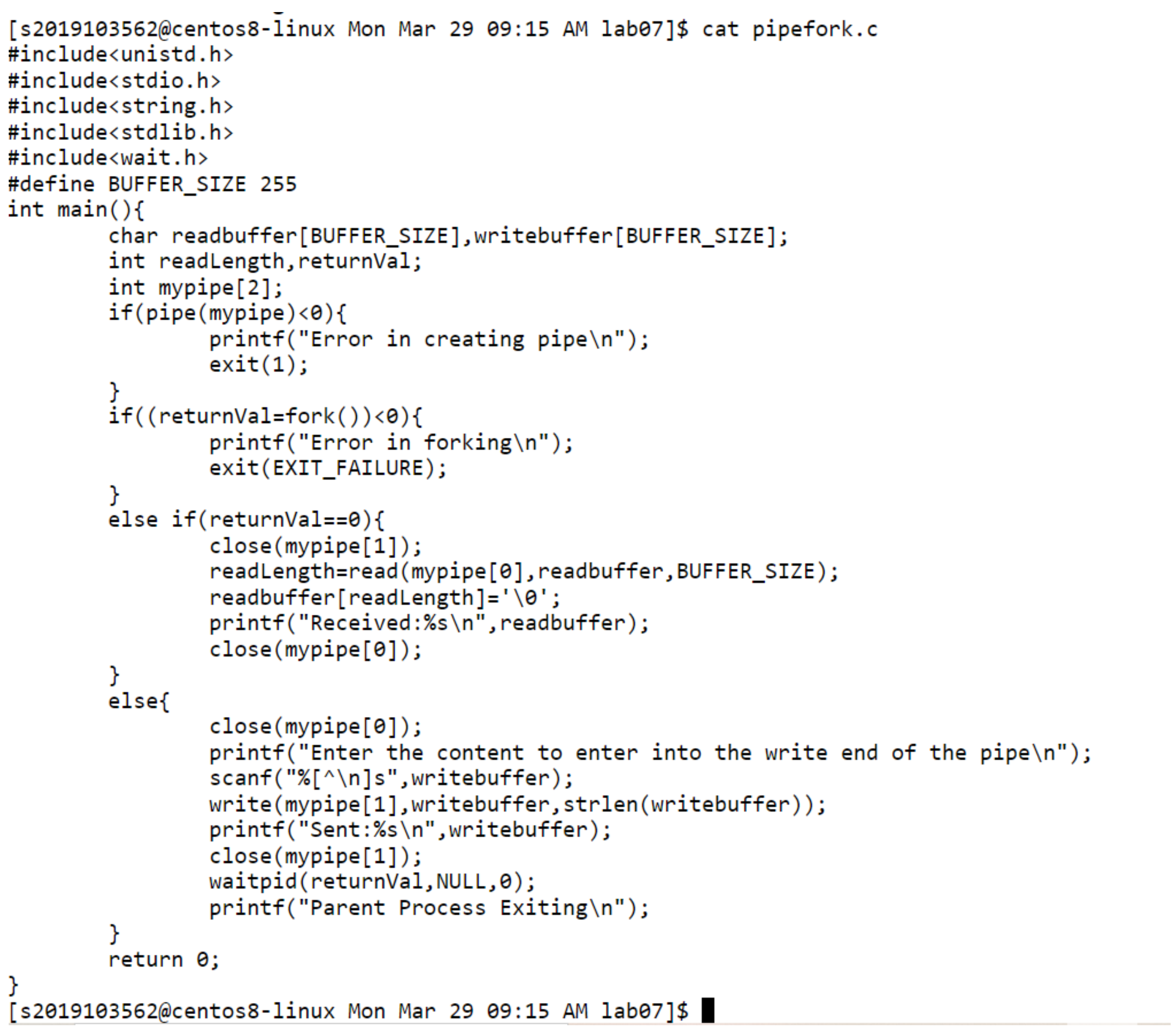


```
[s2019103562@centos8-linux Mon Mar 29 09:03 AM lab07]$ vim rdwr.c
[s2019103562@centos8-linux Mon Mar 29 09:03 AM lab07]$ gcc rdwr.c -o rdwr
[s2019103562@centos8-linux Mon Mar 29 09:03 AM lab07]$ ./rdwr
Enter the content to write to the pipe
Hello All
Writing to the pipe
Hello All
[s2019103562@centos8-linux Mon Mar 29 09:03 AM lab07]$ cat rdwr.c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<string.h>
#define BUFFER_SIZE 255
int main(){
        int mypipe[2];
        char readbuffer[BUFFER_SIZE];
        char writebuffer[BUFFER_SIZE];
        if(pipe(mypipe)<0){
                printf("Error in creating pipe\n");
                exit(1);
        }
        printf("Enter the content to write to the pipe\n");
        scanf("%[^\n]s",writebuffer);
        printf("Writing to the pipe\n");
        write(mypipe[1],writebuffer,strlen(writebuffer));
        int readLength=read(mypipe[0],readbuffer,BUFFER_SIZE);
        readbuffer[readLength]='\0';
        printf("%s\n",readbuffer);
        close(mypipe[0]);
        close(mypipe[1]);
        return 0;
}
[s2019103562@centos8-linux Mon Mar 29 09:03 AM lab07]$ 
```

PIPES USING FORK():

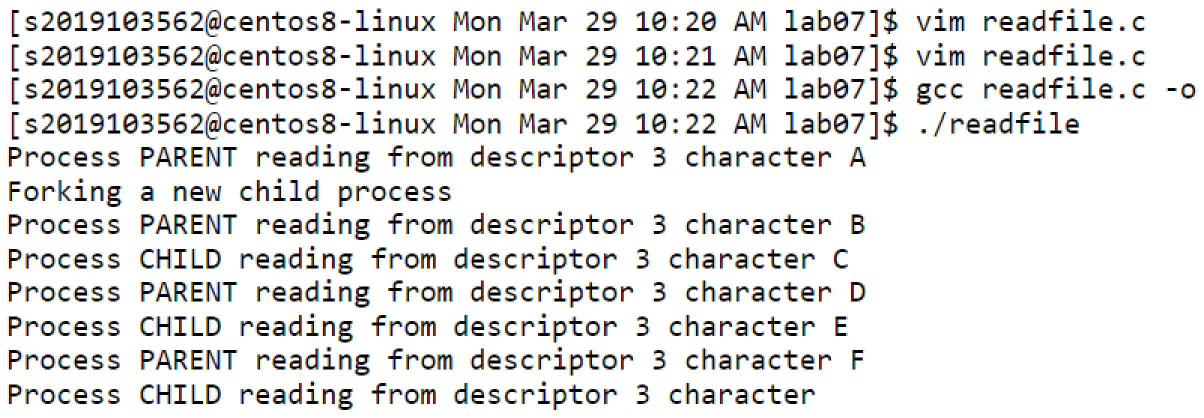


```
[s2019103562@centos8-linux Mon Mar 29 09:03 AM lab07]$ vim pipefork.c
[s2019103562@centos8-linux Mon Mar 29 09:14 AM lab07]$ gcc pipefork.c -o pipefork
[s2019103562@centos8-linux Mon Mar 29 09:14 AM lab07]$ ./pipefork
Enter the content to enter into the write end of the pipe
Welcome to OS lab Pipe concepts
Sent:Welcome to OS lab Pipe concepts
Received:Welcome to OS lab Pipe concepts
Parent Process Exiting
[s2019103562@centos8-linux Mon Mar 29 09:15 AM lab07]$ cat pipefork.c
```

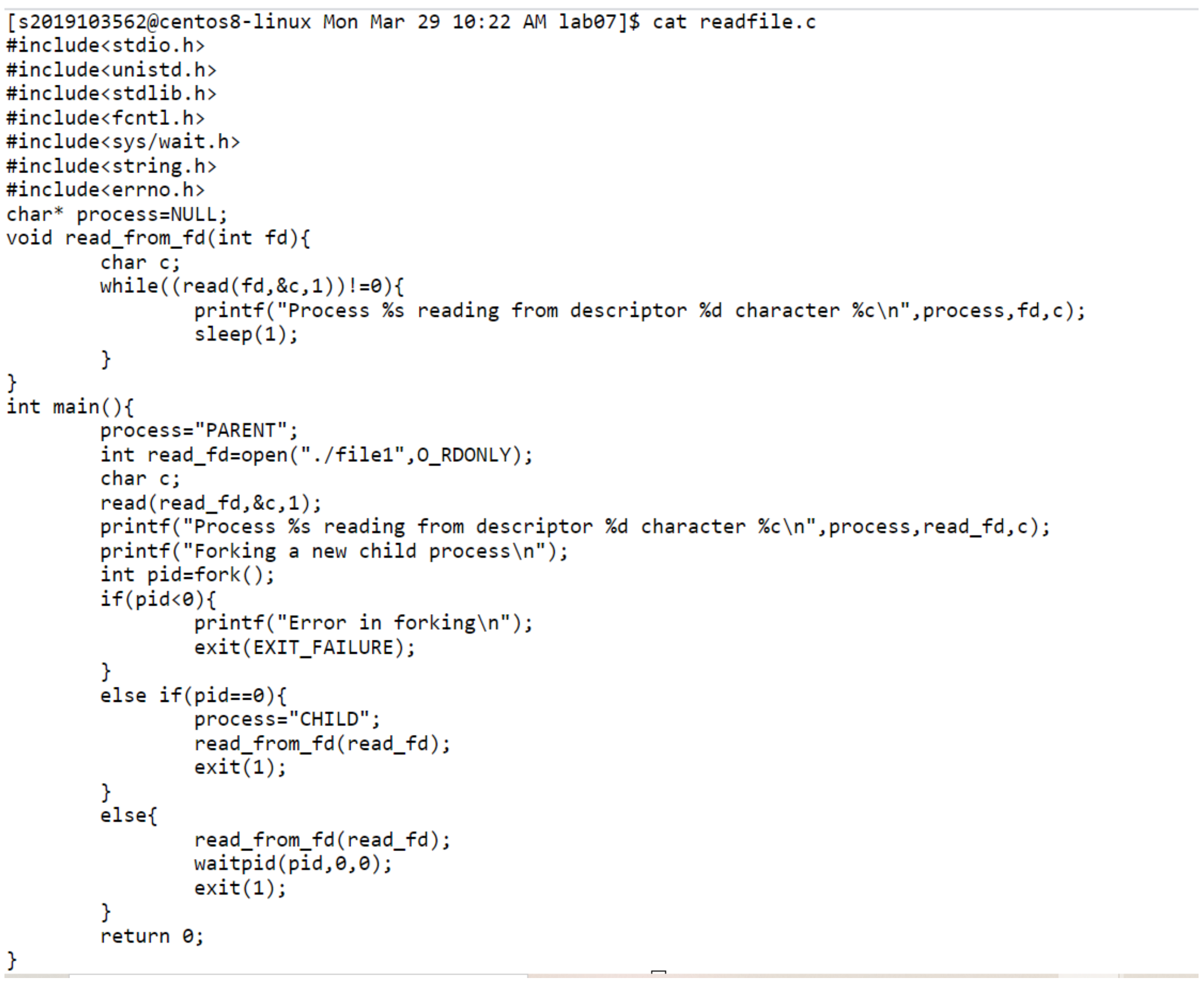


```
[s2019103562@centos8-linux Mon Mar 29 09:15 AM lab07]$ cat pipefork.c
#include<unistd.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<wait.h>
#define BUFFER_SIZE 255
int main(){
        char readbuffer[BUFFER_SIZE],writebuffer[BUFFER_SIZE];
        int readLength,returnVal;
        int mypipe[2];
        if(pipe(mypipe)<0){
                printf("Error in creating pipe\n");
                exit(1);
        }
        if((returnVal=fork())<0){
                printf("Error in forking\n");
                exit(EXIT_FAILURE);
        }
        else if(returnVal==0){
                close(mypipe[1]);
                readLength=read(mypipe[0],readbuffer,BUFFER_SIZE);
                readbuffer[readLength]='\0';
                printf("Received:%s\n",readbuffer);
                close(mypipe[0]);
        }
        else{
                close(mypipe[0]);
                printf("Enter the content to enter into the write end of the pipe\n");
                scanf("%[^\n]s",writebuffer);
                write(mypipe[1],writebuffer,strlen(writebuffer));
                printf("Sent:%s\n",writebuffer);
                close(mypipe[1]);
                waitpid(returnVal,NULL,0);
                printf("Parent Process Exiting\n");
        }
        return 0;
}
[s2019103562@centos8-linux Mon Mar 29 09:15 AM lab07]$ 
```

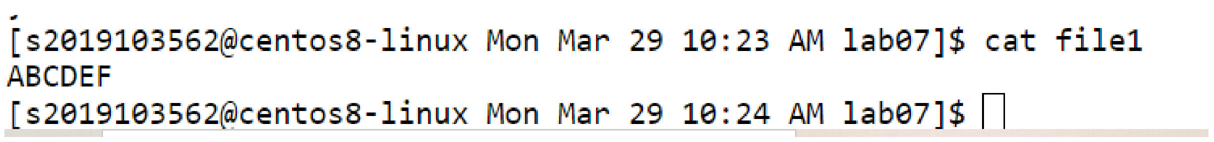
__READING CONTENTS OF A FILE:__



```
[s2019103562@centos8-linux Mon Mar 29 10:20 AM lab07]$ vim readfile.c
[s2019103562@centos8-linux Mon Mar 29 10:21 AM lab07]$ vim readfile.c
[s2019103562@centos8-linux Mon Mar 29 10:22 AM lab07]$ gcc readfile.c -o
[s2019103562@centos8-linux Mon Mar 29 10:22 AM lab07]$ ./readfile
Process PARENT reading from descriptor 3 character A
Forking a new child process
Process PARENT reading from descriptor 3 character B
Process CHILD reading from descriptor 3 character C
Process PARENT reading from descriptor 3 character D
Process CHILD reading from descriptor 3 character E
Process PARENT reading from descriptor 3 character F
Process CHILD reading from descriptor 3 character
```

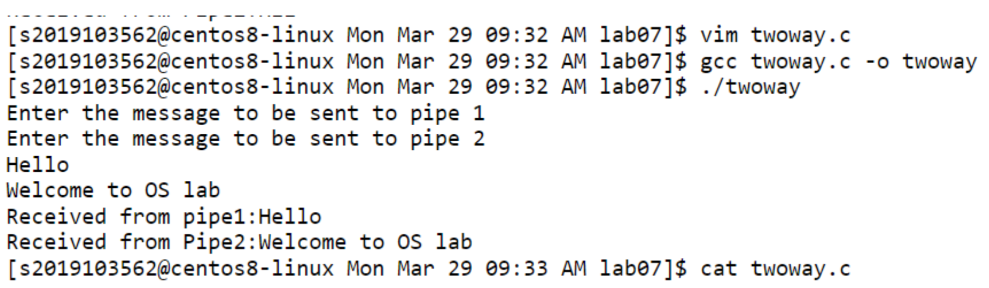


```
[s2019103562@centos8-linux Mon Mar 29 10:22 AM lab07]$ cat readfile.c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<fcntl.h>
#include<sys/wait.h>
#include<string.h>
#include<errno.h>
char* process=NULL;
void read_from_fd(int fd){
        char c;
        while((read(fd,&c,1))!=0){
                printf("Process %s reading from descriptor %d character %c\n",process,fd,c);
                sleep(1);
        }
}
int main(){
        process="PARENT";
        int read_fd=open("./file1",O_RDONLY);
        char c;
        read(read_fd,&c,1);
        printf("Process %s reading from descriptor %d character %c\n",process,read_fd,c);
        printf("Forking a new child process\n");
        int pid=fork();
        if(pid<0){
                printf("Error in forking\n");
                exit(EXIT_FAILURE);
        }
        else if(pid==0){
                process="CHILD";
                read_from_fd(read_fd);
                exit(1);
        }
        else{
                read_from_fd(read_fd);
                waitpid(pid,0,0);
                exit(1);
        }
        return 0;
}
```

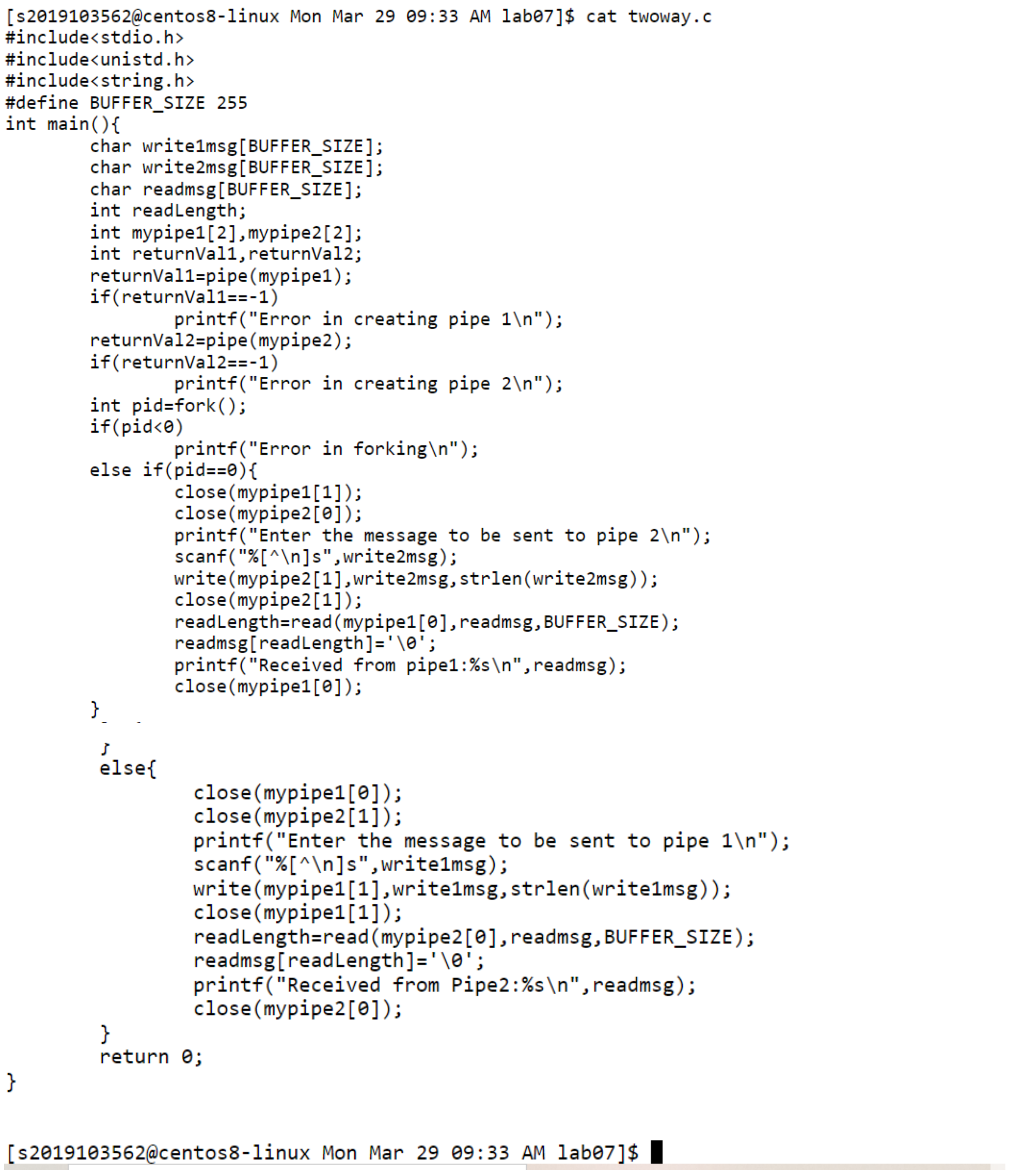


```
[s2019103562@centos8-linux Mon Mar 29 10:23 AM lab07]$ cat file1
ABCDEF
[s2019103562@centos8-linux Mon Mar 29 10:24 AM lab07]$
```

__TWO WAY COMMUNICATION IN PIPES:__

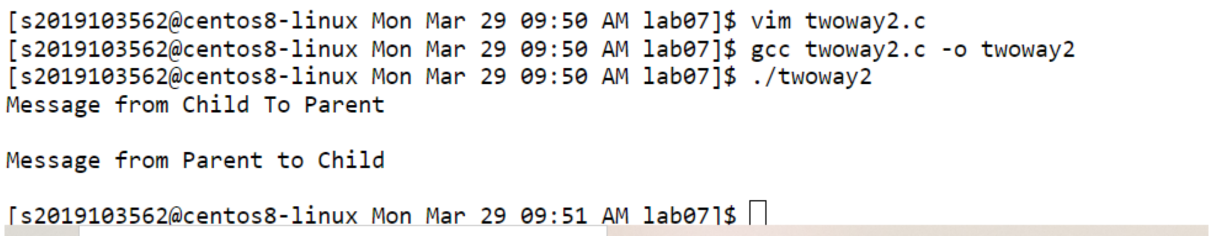


```
[s2019103562@centos8-linux Mon Mar 29 09:32 AM lab07]$ vim twoway.c
[s2019103562@centos8-linux Mon Mar 29 09:32 AM lab07]$ gcc twoway.c -o twoway
[s2019103562@centos8-linux Mon Mar 29 09:32 AM lab07]$ ./twoway
Enter the message to be sent to pipe 1
Enter the message to be sent to pipe 2
Hello
Welcome to OS lab
Received from pipe1:Hello
Received from Pipe2:Welcome to OS lab
[s2019103562@centos8-linux Mon Mar 29 09:33 AM lab07]$ cat twoway.c
```

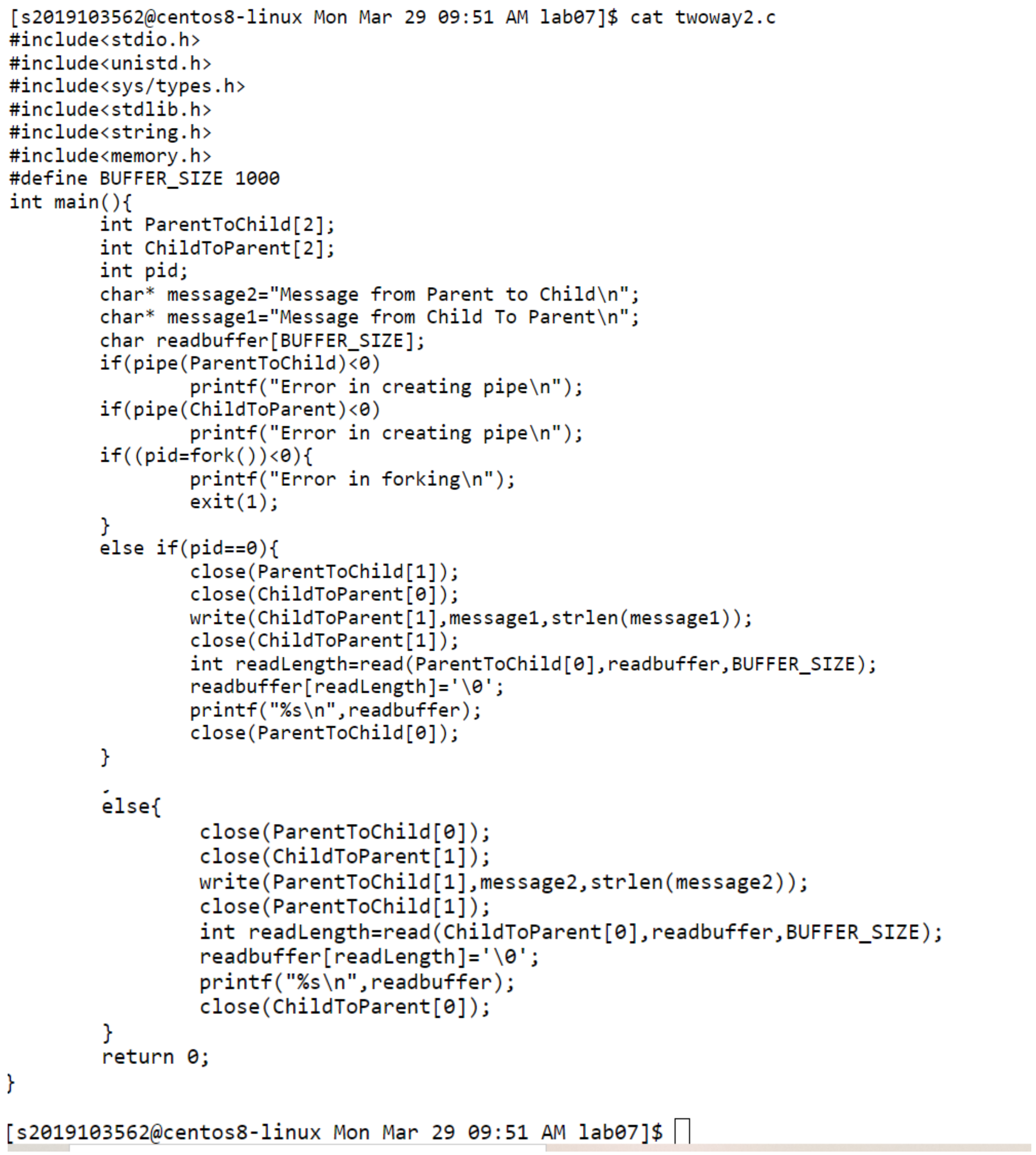


```
[s2019103562@centos8-linux Mon Mar 29 09:33 AM lab07]$ cat twoway.c
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#define BUFFER_SIZE 255
int main(){
        char write1msg[BUFFER_SIZE];
        char write2msg[BUFFER_SIZE];
        char readmsg[BUFFER_SIZE];
        int readLength;
        int mypipe1[2],mypipe2[2];
        int returnVal1,returnVal2;
        returnVal1=pipe(mypipe1);
        if(returnVal1==-1)
                printf("Error in creating pipe 1\n");
        returnVal2=pipe(mypipe2);
        if(returnVal2==-1)
                printf("Error in creating pipe 2\n");
        int pid=fork();
        if(pid<0)
                printf("Error in forking\n");
        else if(pid==0){
                close(mypipe1[1]);
                close(mypipe2[0]);
                printf("Enter the message to be sent to pipe 2\n");
                scanf("%[^\n]s",write2msg);
                write(mypipe2[1],write2msg,strlen(write2msg));
                close(mypipe2[1]);
                readLength=read(mypipe1[0],readmsg,BUFFER_SIZE);
                readmsg[readLength]='\0';
                printf("Received from pipe1:%s\n",readmsg);
                close(mypipe1[0]);
        }
        else{
                close(mypipe1[0]);
                close(mypipe2[1]);
                printf("Enter the message to be sent to pipe 1\n");
                scanf("%[^\n]s",write1msg);
                write(mypipe1[1],write1msg,strlen(write1msg));
                close(mypipe1[1]);
                readLength=read(mypipe2[0],readmsg,BUFFER_SIZE);
                readmsg[readLength]='\0';
                printf("Received from Pipe2:%s\n",readmsg);
                close(mypipe2[0]);
        }
        return 0;
}

[s2019103562@centos8-linux Mon Mar 29 09:33 AM lab07]$ █
```

TWO WAY COMMUNICATION PARENT AND CHILD PROCESSES:

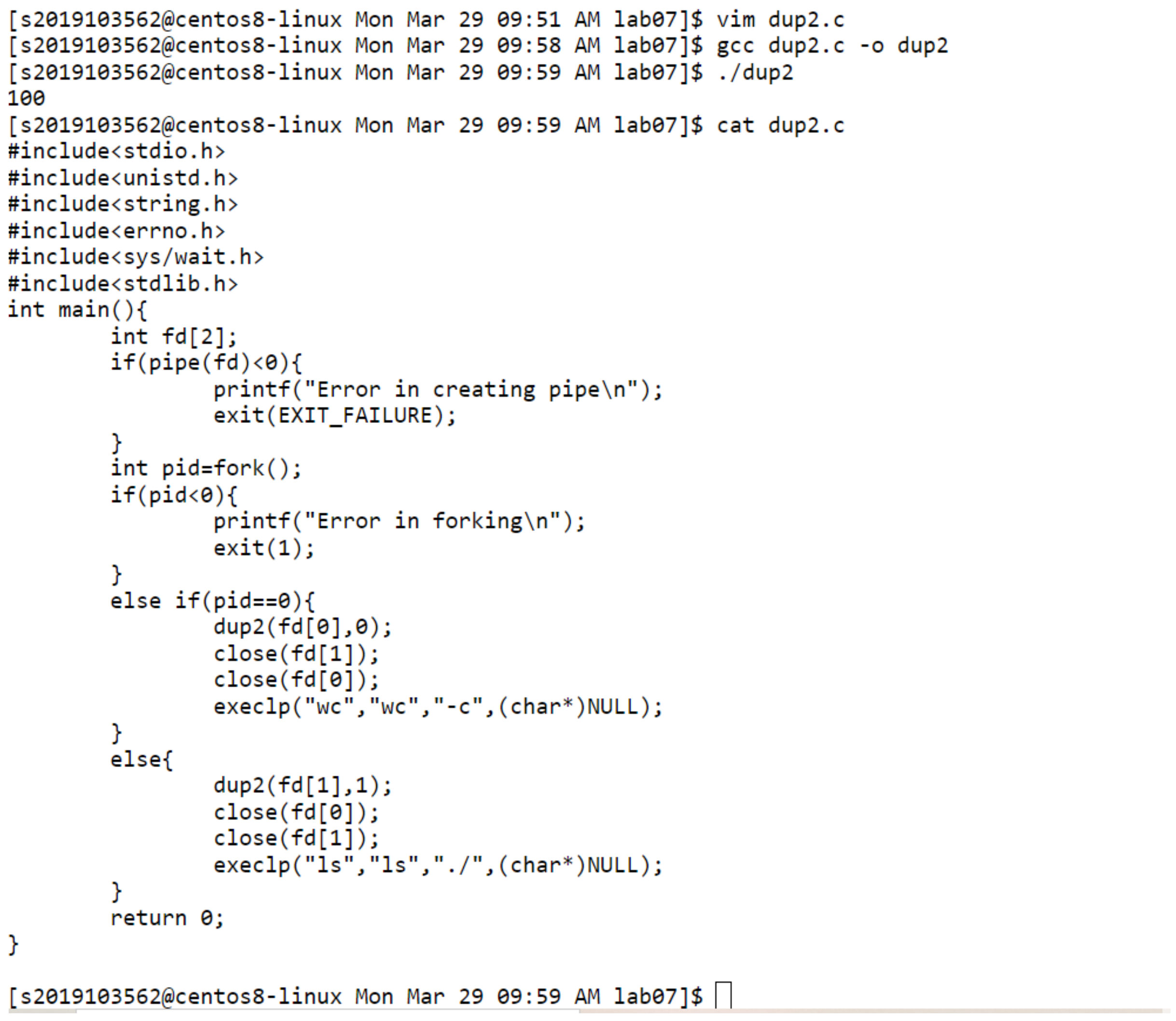


```
[s2019103562@centos8-linux Mon Mar 29 09:50 AM lab07]$ vim twoway2.c
[s2019103562@centos8-linux Mon Mar 29 09:50 AM lab07]$ gcc twoway2.c -o twoway2
[s2019103562@centos8-linux Mon Mar 29 09:50 AM lab07]$ ./twoway2
Message from Child To Parent

Message from Parent to Child

[s2019103562@centos8-linux Mon Mar 29 09:51 AM lab07]$ 
```



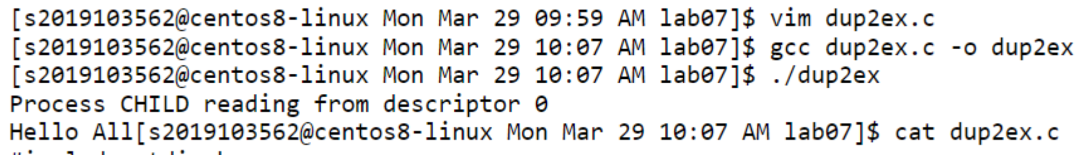
```
[s2019103562@centos8-linux Mon Mar 29 09:51 AM lab07]$ cat twoway2.c
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<stdlib.h>
#include<string.h>
#include<memory.h>
#define BUFFER_SIZE 1000
int main(){
        int ParentToChild[2];
        int ChildToParent[2];
        int pid;
        char* message2="Message from Parent to Child\n";
        char* message1="Message from Child To Parent\n";
        char readbuffer[BUFFER_SIZE];
        if(pipe(ParentToChild)<0)
                printf("Error in creating pipe\n");
        if(pipe(ChildToParent)<0)
                printf("Error in creating pipe\n");
        if((pid=fork())<0){
                printf("Error in forking\n");
                exit(1);
        }
        else if(pid==0){
                close(ParentToChild[1]);
                close(ChildToParent[0]);
                write(ChildToParent[1],message1,strlen(message1));
                close(ChildToParent[1]);
                int readLength=read(ParentToChild[0],readbuffer,BUFFER_SIZE);
                readbuffer[readLength]='\0';
                printf("%s\n",readbuffer);
                close(ParentToChild[0]);
        }

        else{
                close(ParentToChild[0]);
                close(ChildToParent[1]);
                write(ParentToChild[1],message2,strlen(message2));
                close(ParentToChild[1]);
                int readLength=read(ChildToParent[0],readbuffer,BUFFER_SIZE);
                readbuffer[readLength]='\0';
                printf("%s\n",readbuffer);
                close(ChildToParent[0]);
        }
        return 0;
}

[s2019103562@centos8-linux Mon Mar 29 09:51 AM lab07]$
```

Dup2:

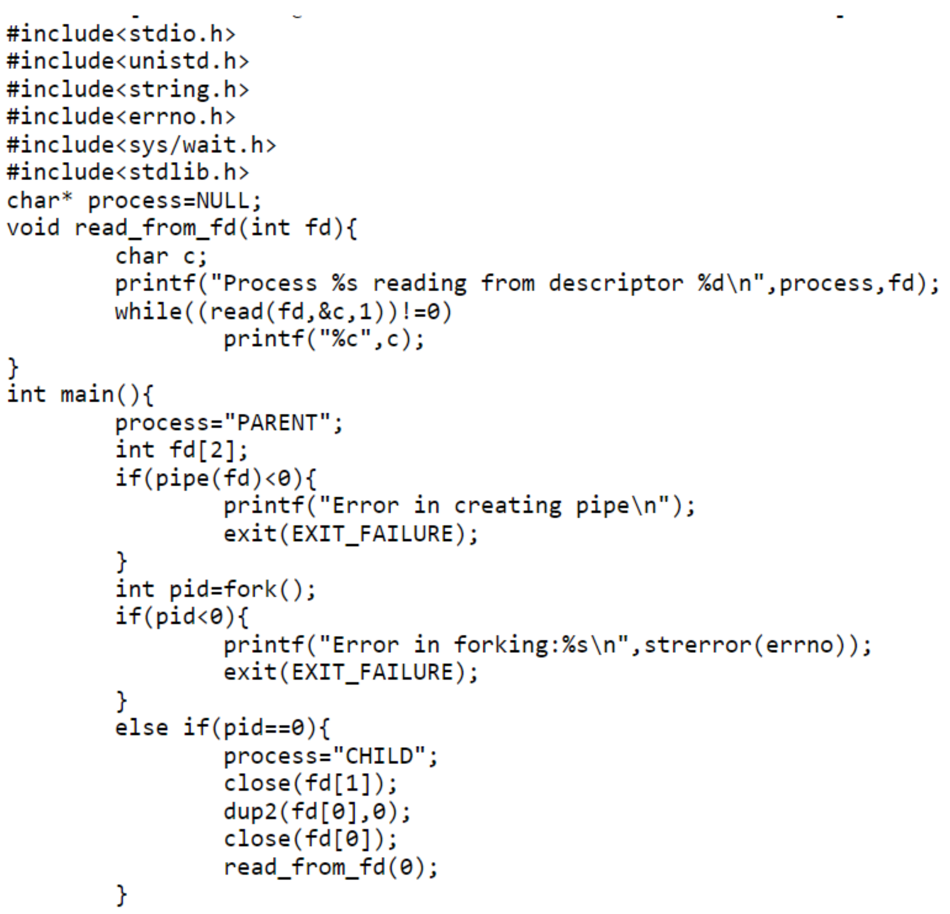


```
[s2019103562@centos8-linux Mon Mar 29 09:51 AM lab07]$ vim dup2.c
[s2019103562@centos8-linux Mon Mar 29 09:58 AM lab07]$ gcc dup2.c -o dup2
[s2019103562@centos8-linux Mon Mar 29 09:59 AM lab07]$ ./dup2
100
[s2019103562@centos8-linux Mon Mar 29 09:59 AM lab07]$ cat dup2.c
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<errno.h>
#include<sys/wait.h>
#include<stdlib.h>
int main(){
        int fd[2];
        if(pipe(fd)<0){
                printf("Error in creating pipe\n");
                exit(EXIT_FAILURE);
        }
        int pid=fork();
        if(pid<0){
                printf("Error in forking\n");
                exit(1);
        }
        else if(pid==0){
                dup2(fd[0],0);
                close(fd[1]);
                close(fd[0]);
                execlp("wc","wc","-c",(char*)NULL);
        }
        else{
                dup2(fd[1],1);
                close(fd[0]);
                close(fd[1]);
                execlp("ls","ls","./",(char*)NULL);
        }
        return 0;
}

[s2019103562@centos8-linux Mon Mar 29 09:59 AM lab07]$ 
```

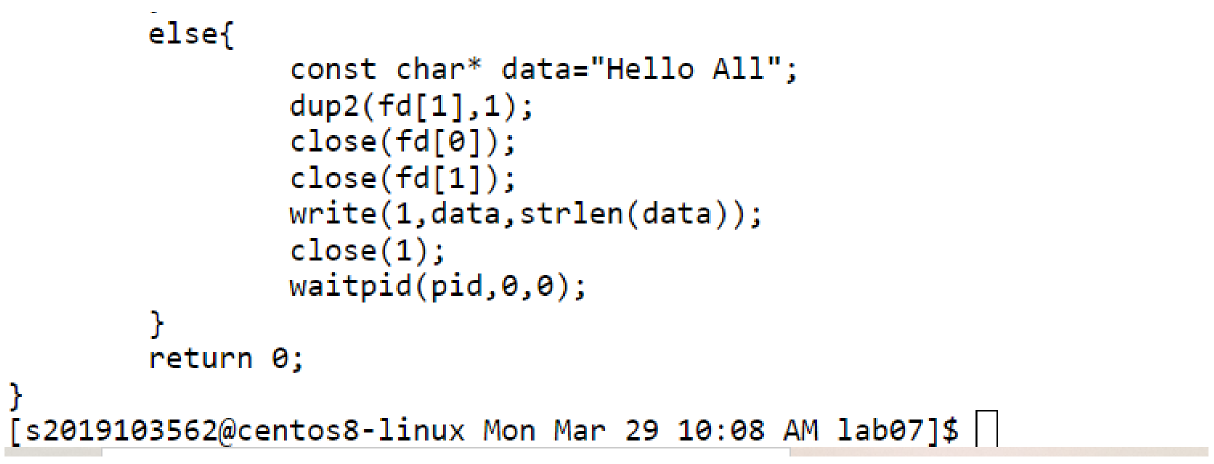
WRITE AND READ USING DUP2:



```
[s2019103562@centos8-linux Mon Mar 29 09:59 AM lab07]$ vim dup2ex.c
[s2019103562@centos8-linux Mon Mar 29 10:07 AM lab07]$ gcc dup2ex.c -o dup2ex
[s2019103562@centos8-linux Mon Mar 29 10:07 AM lab07]$ ./dup2ex
Process CHILD reading from descriptor 0
Hello All[s2019103562@centos8-linux Mon Mar 29 10:07 AM lab07]$ cat dup2ex.c
```



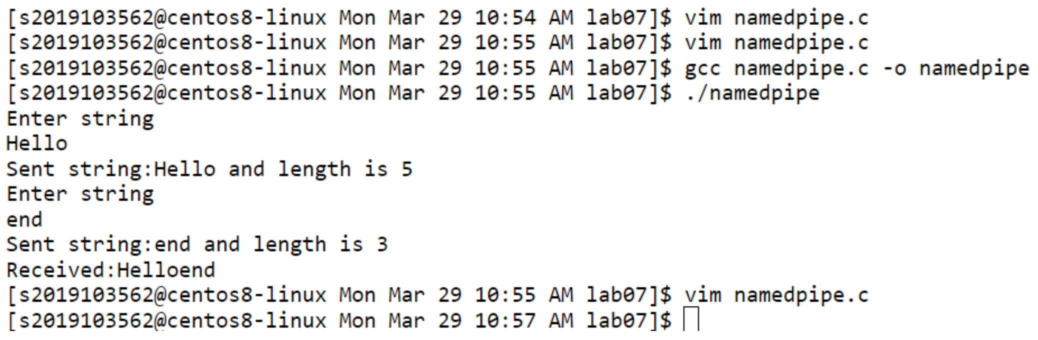
```
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<errno.h>
#include<sys/wait.h>
#include<stdlib.h>
char* process=NULL;
void read_from_fd(int fd){
        char c;
        printf("Process %s reading from descriptor %d\n",process,fd);
        while((read(fd,&c,1))!=0)
                printf("%c",c);
}
int main(){
        process="PARENT";
        int fd[2];
        if(pipe(fd)<0){
                printf("Error in creating pipe\n");
                exit(EXIT_FAILURE);
        }
        int pid=fork();
        if(pid<0){
                printf("Error in forking:%s\n",strerror(errno));
                exit(EXIT_FAILURE);
        }
        else if(pid==0){
                process="CHILD";
                close(fd[1]);
                dup2(fd[0],0);
                close(fd[0]);
                read_from_fd(0);
        }
```



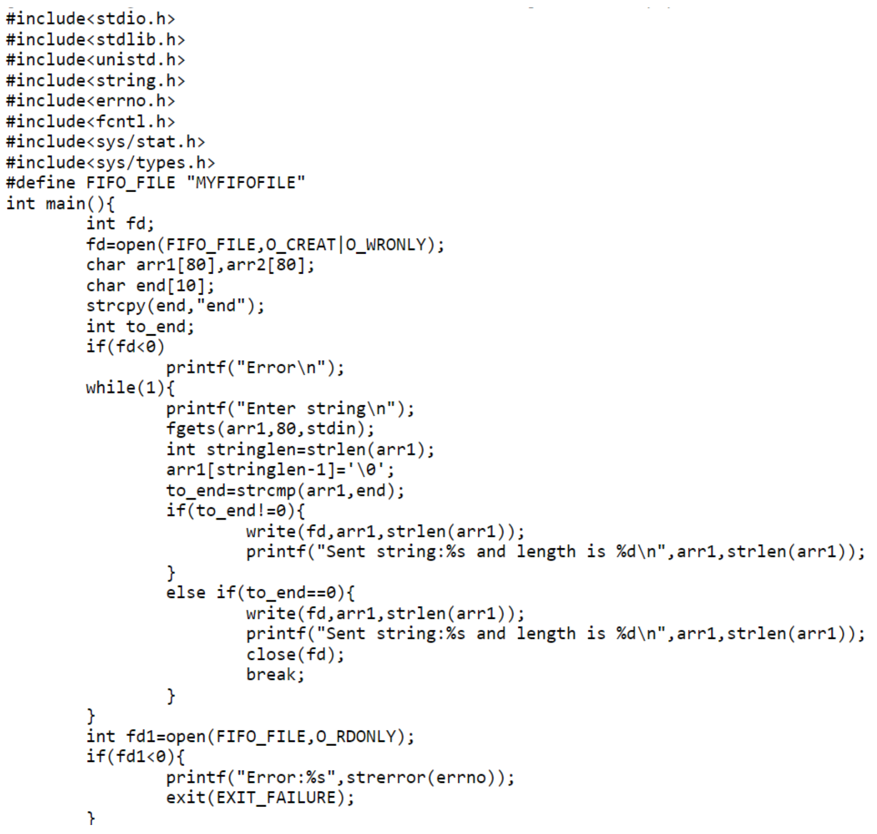
```
        else{
                const char* data="Hello All";
                dup2(fd[1],1);
                close(fd[0]);
                close(fd[1]);
                write(1,data,strlen(data));
                close(1);
                waitpid(pid,0,0);
        }
        return 0;
}
[s2019103562@centos8-linux Mon Mar 29 10:08 AM lab07]$
```

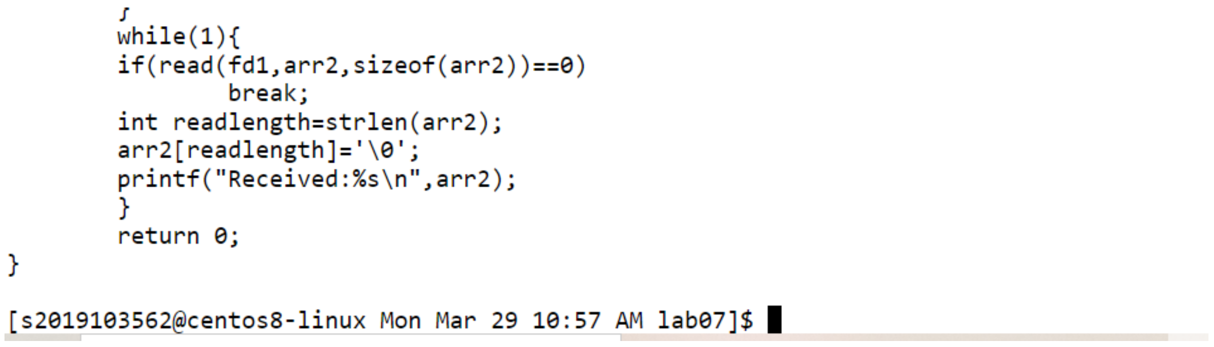
__NAMEDPIPES:__

__SERVER CLIENT(SINGLE PROGRAM):__



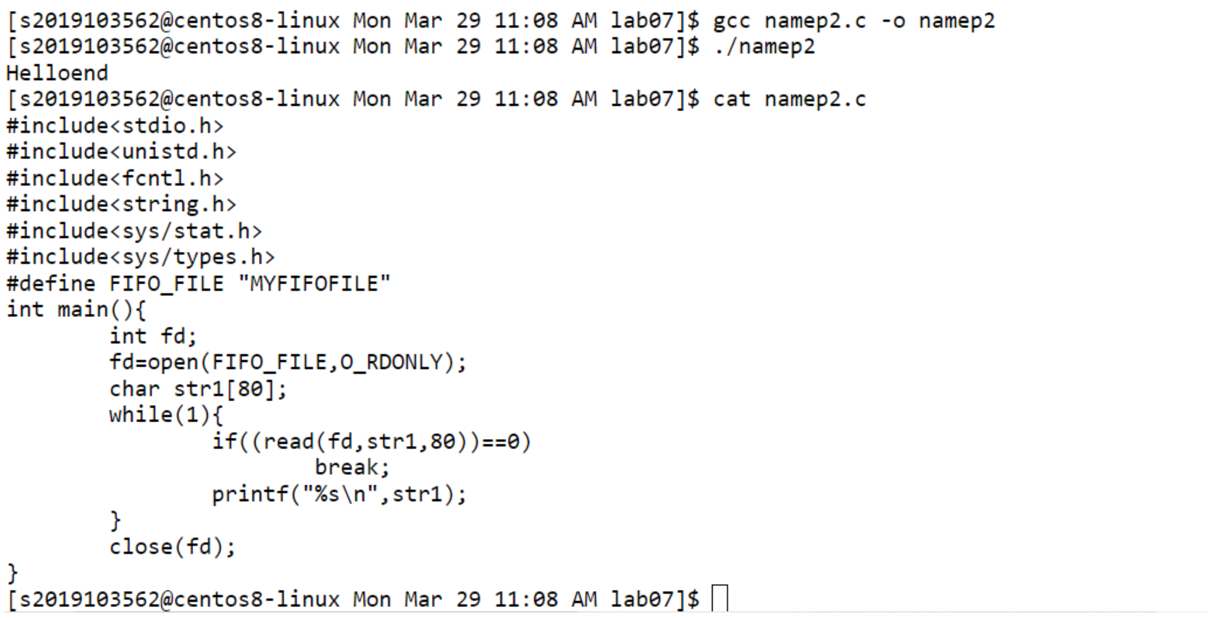
```
[s2019103562@centos8-linux Mon Mar 29 10:54 AM lab07]$ vim namedpipe.c
[s2019103562@centos8-linux Mon Mar 29 10:55 AM lab07]$ vim namedpipe.c
[s2019103562@centos8-linux Mon Mar 29 10:55 AM lab07]$ gcc namedpipe.c -o namedpipe
[s2019103562@centos8-linux Mon Mar 29 10:55 AM lab07]$ ./namedpipe
Enter string
Hello
Sent string:Hello and length is 5
Enter string
end
Sent string:end and length is 3
Received:Helloend
[s2019103562@centos8-linux Mon Mar 29 10:55 AM lab07]$ vim namedpipe.c
[s2019103562@centos8-linux Mon Mar 29 10:57 AM lab07]$ 
```





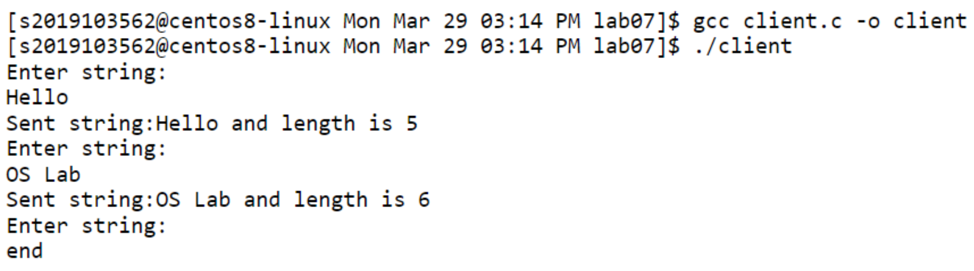
```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#include<errno.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<sys/types.h>
#define FIFO_FILE "MYFIFOFILE"
int main(){
        int fd;
        fd=open(FIFO_FILE,O_CREAT|O_WRONLY);
        char arr1[80],arr2[80];
        char end[10];
        strcpy(end,"end");
        int to_end;
        if(fd<0)
                printf("Error\n");
        while(1){
                printf("Enter string\n");
                fgets(arr1,80,stdin);
                int stringlen=strlen(arr1);
                arr1[stringlen-1]='\0';
                to_end=strcmp(arr1,end);
                if(to_end!=0){
                        write(fd,arr1,strlen(arr1));
                        printf("Sent string:%s and length is %d\n",arr1,strlen(arr1));
                }
                else if(to_end==0){
                        write(fd,arr1,strlen(arr1));
                        printf("Sent string:%s and length is %d\n",arr1,strlen(arr1));
                        close(fd);
                        break;
                }
        }
        int fd1=open(FIFO_FILE,O_RDONLY);
        if(fd1<0){
                printf("Error:%s",strerror(errno));
                exit(EXIT_FAILURE);
        }

        while(1){
        if(read(fd1,arr2,sizeof(arr2))==0)
                break;
        int readlength=strlen(arr2);
        arr2[readlength]='\0';
        printf("Received:%s\n",arr2);
        }
        return 0;
}

[s2019103562@centos8-linux Mon Mar 29 10:57 AM lab07]$ 
```

<u>SERVER(SEPARATE PROGRAM):</u>

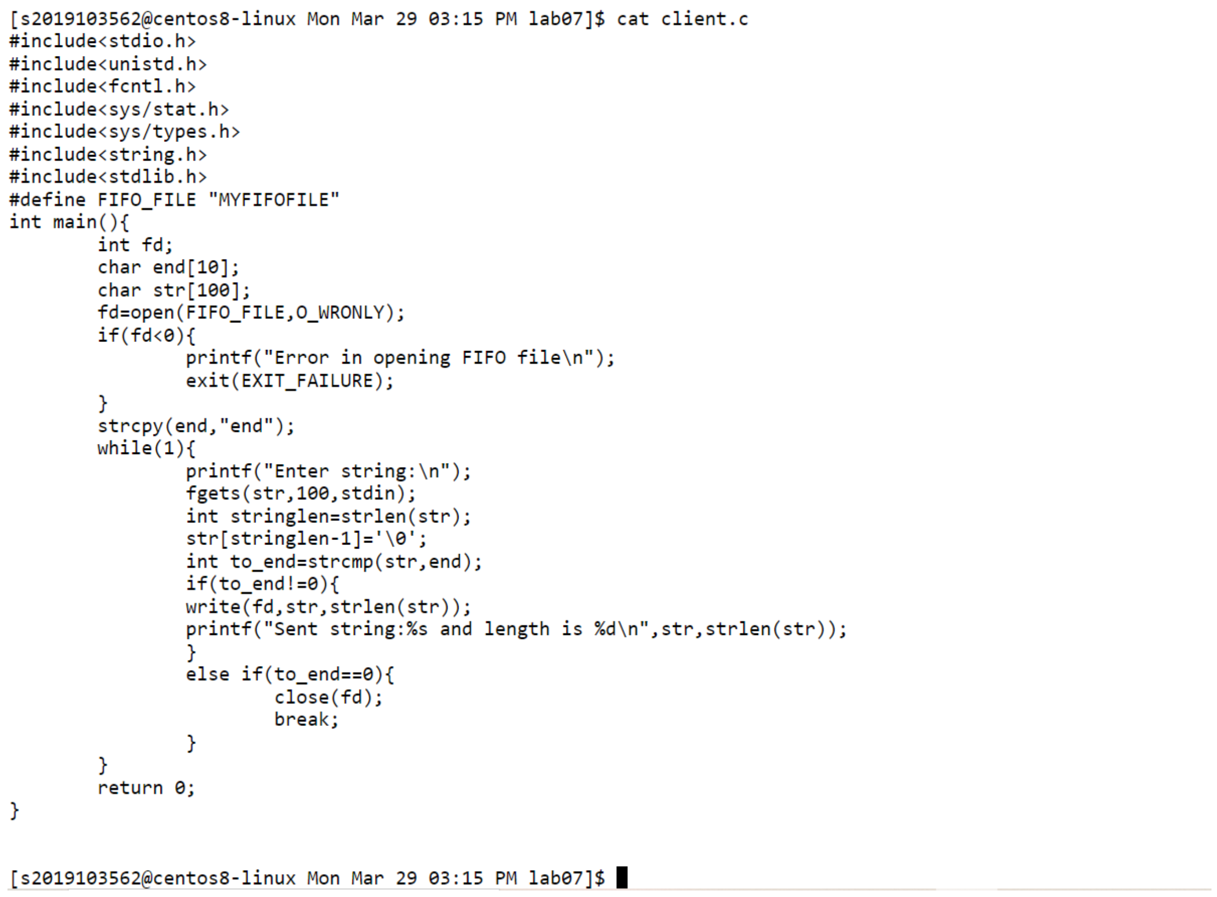


```
[s2019103562@centos8-linux Mon Mar 29 11:08 AM lab07]$ gcc namep2.c -o namep2
[s2019103562@centos8-linux Mon Mar 29 11:08 AM lab07]$ ./namep2
Helloend
[s2019103562@centos8-linux Mon Mar 29 11:08 AM lab07]$ cat namep2.c
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<string.h>
#include<sys/stat.h>
#include<sys/types.h>
#define FIFO_FILE "MYFIFOFILE"
int main(){
        int fd;
        fd=open(FIFO_FILE,O_RDONLY);
        char str1[80];
        while(1){
                if((read(fd,str1,80))==0)
                        break;
                printf("%s\n",str1);
        }
        close(fd);
}
[s2019103562@centos8-linux Mon Mar 29 11:08 AM lab07]$ 
```

<u>CLIENT(SEPARATE PROGRAM):</u>



```
[s2019103562@centos8-linux Mon Mar 29 03:14 PM lab07]$ gcc client.c -o client
[s2019103562@centos8-linux Mon Mar 29 03:14 PM lab07]$ ./client
Enter string:
Hello
Sent string:Hello and length is 5
Enter string:
OS Lab
Sent string:OS Lab and length is 6
Enter string:
end
```



```
[s2019103562@centos8-linux Mon Mar 29 03:15 PM lab07]$ cat client.c
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<string.h>
#include<stdlib.h>
#define FIFO_FILE "MYFIFOFILE"
int main(){
        int fd;
        char end[10];
        char str[100];
        fd=open(FIFO_FILE,O_WRONLY);
        if(fd<0){
                printf("Error in opening FIFO file\n");
                exit(EXIT_FAILURE);
        }
        strcpy(end,"end");
        while(1){
                printf("Enter string:\n");
                fgets(str,100,stdin);
                int stringlen=strlen(str);
                str[stringlen-1]='\0';
                int to_end=strcmp(str,end);
                if(to_end!=0){
                write(fd,str,strlen(str));
                printf("Sent string:%s and length is %d\n",str,strlen(str));
                }
                else if(to_end==0){
                        close(fd);
                        break;
                }
        }
        return 0;
}

[s2019103562@centos8-linux Mon Mar 29 03:15 PM lab07]$ █
```