LAB-6

-Saloni Patel

Aim: Inter process communication:

Pipe system call:

#include int pipe(int pipefd[2]);

pipe() creates a pipe, a unidirectional data channel that can be used for inter process communication.

The array pipefd is used to return two file descriptors referring to the ends of the pipe.

- pipefd[0] refers to the read end of the pipe.
- pipefd[1] refers to the write end of the pipe.

Data written to the write end of the pipe is buffered by the kernel until it is read from the read end of the pipe.

On success, zero is returned.

On error, -1 is returned, and errno is set appropriately.

Close system call:

#include int close(int fd);

close() closes a file descriptor, so that it no longer refers to any file and may be reused. Any record locks held on the file it was associated with, and owned by the process, are removed (regardless of the file descriptor that was used to obtain the lock).

close() returns zero on success.

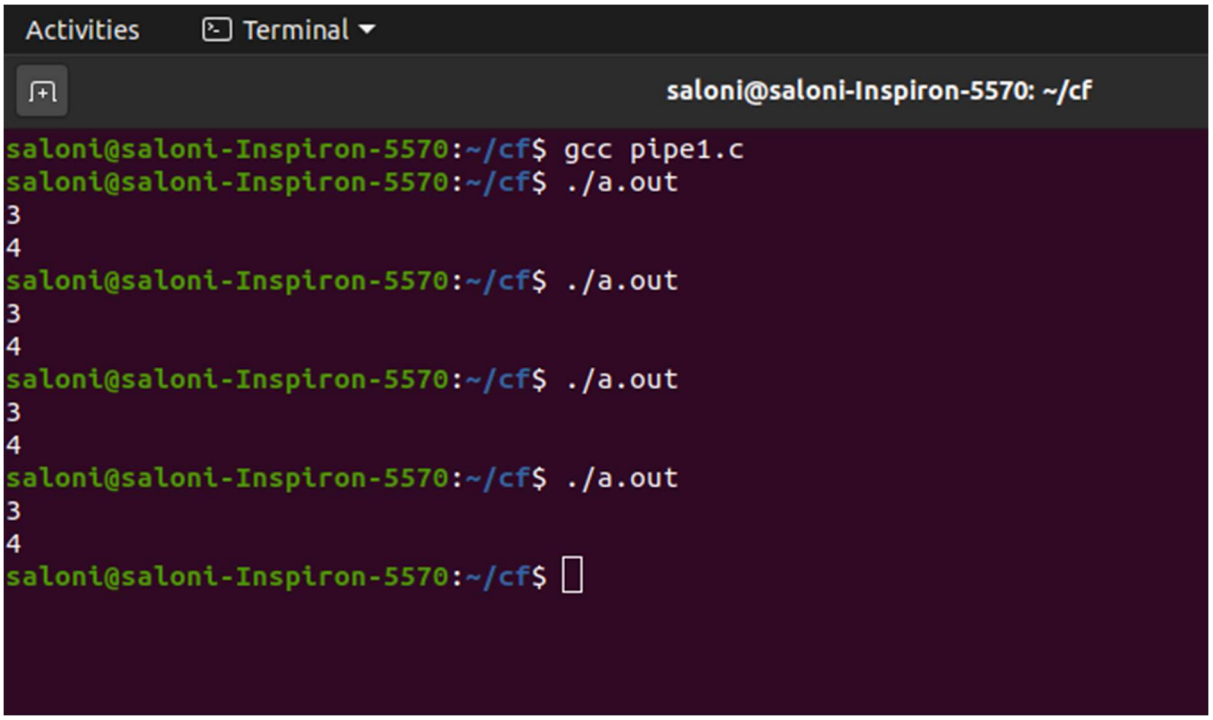
On error, -1 is returned, and errno is set appropriately.

1. **Write a program to create a pipe and print the values of pipe file descriptors.**

```
#include<stdio.h>
#include<unistd.h>
void main()
{
        int fd1[2],n,number[2];
        int n1=pipe(fd1);
//      int n2=pipe(fd2);
        if(n1==0)
        {
                printf("%d\n",fd1[0]);
                printf("%d\n",fd1[1]);
//              printf("%d\n",fd2[0]);
//          printf("%d\n",fd2[1]);

        }
        else{
```

```
                printf("error in pipe creation\n");
        }
}
```



2. **Write a program to pass a message from parent process to child process through a pipe.**

```
#include<stdio.h>
#include<unistd.h>
#include <sys/types.h>
    #include <sys/wait.h>

void main()
{
    int fd[2],pid,nb,sta;
    int n;
    char msg[40];
    n=pipe(fd);
    if(n==-1)
    {
        printf("error in creation of pipe\n");
    }
    else
        {
        pid=fork();
        if(n==-1)
            printf("error in process creation\n");
        else if(pid>0)
        {
```

```
                close(fd[0]);
                nb=read(0,msg,sizeof(msg));
                        msg[nb-1]='\0';
                        write(fd[1],msg,sizeof(msg));
                printf("parent pid %d sending msg %s to child %d\n",getpid(),msg,pid);

        }
        else
        {
                close(fd[1]);
                        nb=read(fd[0],msg,sizeof(msg));
                printf("child pid %d receving msg %s from parent
%d\n",getpid(),msg,getppid());
        }
    }
}
```

3. **Write a program to pass file name from parent process to child process through a pipe, child process should pass the file contents to parent process and parent should print the contents.**

```
#include<stdio.h>
#include<unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <fcntl.h>
void main()
{
    int fd[2],pid,nb,sta,nf,fdr[2],nr,nfr;
    int n,f;
    char msg[256],content[256];
    n=pipe(fd);
```

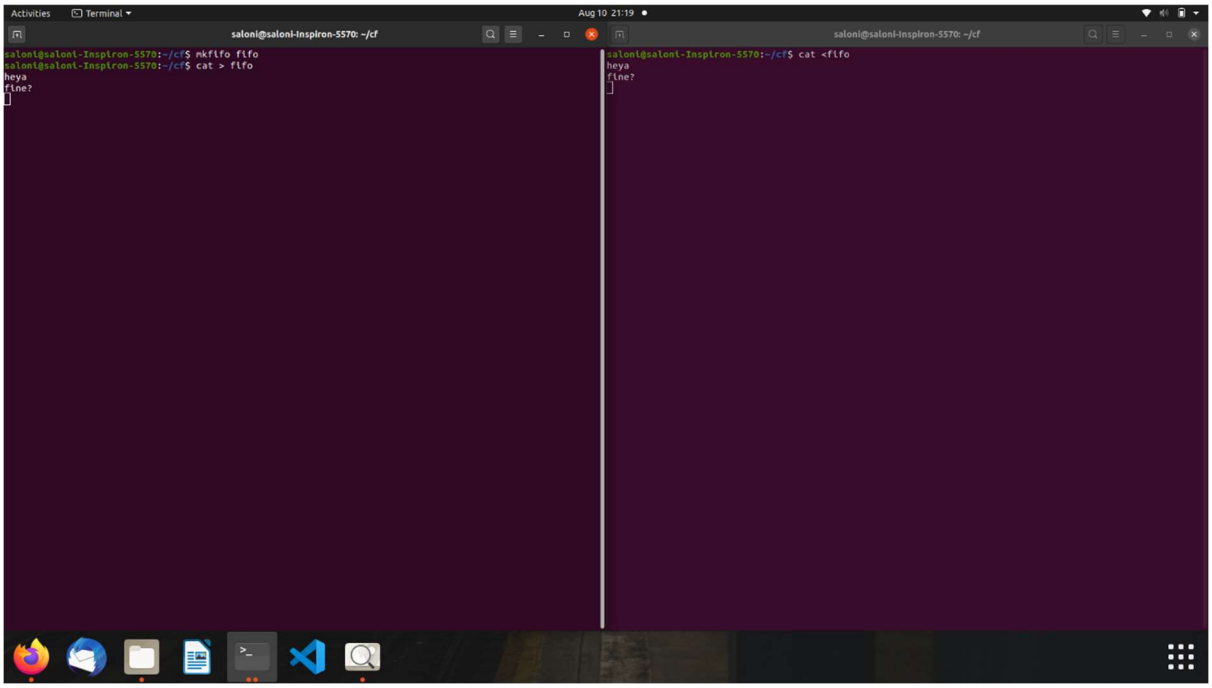
```
        nr=pipe(fdr);
    if(n==-1)
    {
        printf("error in pipe\n");
    }
    else{
        pid=fork();
        if(n==-1)
            printf("error in process creation\n");
        else if(pid>0)
        {
            close(fd[0]);
                    nb=read(0,msg,sizeof(msg));
            msg[nb-1]='\0';
            write(fd[1],msg,nb);
                    wait(&sta);
                    close(fdr[1]);
                    nfr=read(fdr[0],content,sizeof(content));
                    printf("%s",content);
                }
        else
        {
            close(fd[1]);
            nf=read(fd[0],msg,sizeof(msg));
                    f=open(msg,O_RDWR);
            nf=read(f,content,sizeof(msg));
//              write(1,&content,nf);
                    close(fdr[0]);
                    write(fdr[1],content,nf);

        }
    }
}
```

```
Activities   Terminal
                                  saloni@saloni-Inspiron-5570: ~/cf
saloni@saloni-Inspiron-5570:~/cf$ gcc pipe5.c
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
/home/saloni/cf/demo.txt
hello saloni..hoe r u?
saloni@saloni-Inspiron-5570:~/cf$ cat demo.txt
hello saloni..hoe r u?
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
/home/saloni/cf/dest.txt
hello saloni..hoe r u?
saloni@saloni-Inspiron-5570:~/cf$ cat dest.txt
hello saloni..hoe r u?
saloni@saloni-Inspiron-5570:~/cf$
```

Assignment:

1. **Using command:**



2. **Code implementation:**

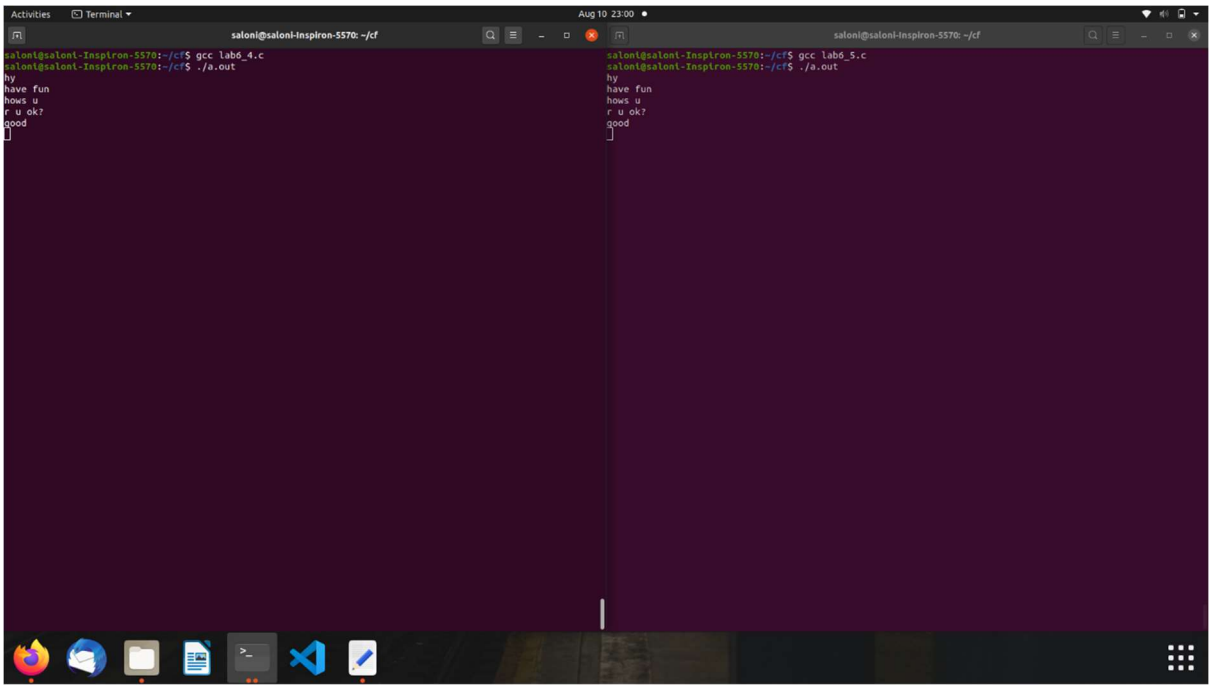
- **File for reading input of user,**

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

```
#include <sys/types.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <sys/stat.h>
void main()
{
        int f,nb;
        char fname[256]="test3",msg[256];
        mkfifo(fname,0666);
        f=open(fname,O_WRONLY);
        while(1)
        {
                nb=read(0,msg,sizeof(msg));
                        write(f,msg,nb);
        }
}
```

- **File for writing output to console,**

```
#include<stdio.h>
#include<unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <sys/stat.h>
void main()
{
        int nb,f;
        char fname[256]="test3",msg[256];
        while(1)
        {
                f=open(fname,O_SYNC);
                nb=read(f,msg,sizeof(msg));
                write(1,msg,nb);
                close(f);
        }
}
```


Activities
Terminal
Aug 10 23:00
saloni@saloni-Inspiron-5570: ~/cf
saloni@saloni-Inspiron-5570:~/cf$ gcc lab6_4.c
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
hy
have fun
hows u
r u ok?
good
saloni@saloni-Inspiron-5570: ~/cf
saloni@saloni-Inspiron-5570:~/cf$ gcc lab6_5.c
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
hy
have fun
hows u
r u ok?
good