SEC : A **IPC-1: PIPE, FIFO** **BATCH : A3**

REG: 190905513 **LAB 05** **NAME: MOHAMMAD DANISH EQBAL**

LAB EXERCISES:

1. Write a producer and consumer program in C using the FIFO queue. The producer should write a set of 4 integers into the FIFO queue and the consumer should display the 4 integers.

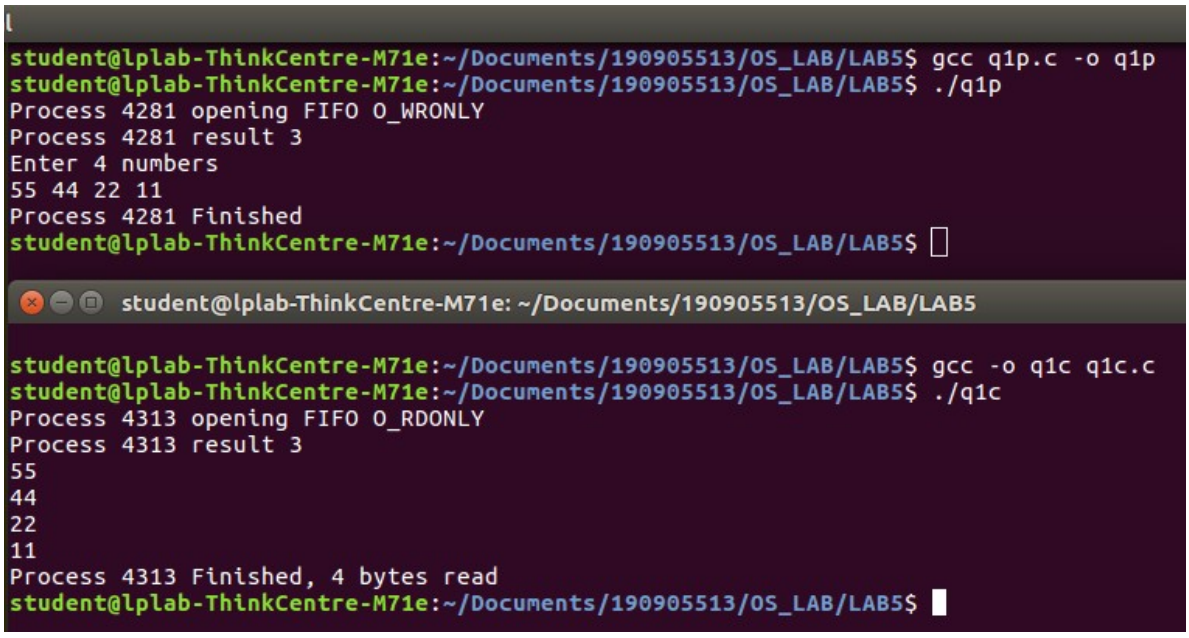
Program:

```
//Producer
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<limits.h>
#include<fcntl.h>
#include<sys/msg.h>
#include<sys/stat.h>
#include<string.h>

#define FIFO_NAME "my_fifo"
#define BUFFER_SIZE 1000


int main(int argc, char *argv[])
{
    int pipe_fd;
    int res;
    int open_mode=O_WRONLY;
    int n=0;
    char buffer[BUFFER_SIZE+1];

    if(access(FIFO_NAME,F_OK)==-1)
    {
        res=mkfifo(FIFO_NAME,0777);
        if(res!=0)
        {
            fprintf(stderr, "Could not create file
%s\n",FIFO_NAME );
            exit(EXIT_FAILURE);
        }
    }

    printf("Process %d opening FIFO O_WRONLY\n",getpid());
    pipe_fd=open(FIFO_NAME,open_mode);
    printf("Process %d result %d\n",getpid(),pipe_fd);
```

```
    if (pipe_fd!=-1)
    {
        printf("Enter 4 numbers\n");

        while(n<4)
        {
            scanf("%s",buffer);
            res=write(pipe_fd,buffer,BUFFER_SIZE);

            if(res==-1)
            {
                fprintf(stderr, "Write Error on Pipe\n");
                exit(EXIT_FAILURE);
            }
            n++;
        }
        (void)close(pipe_fd);
    }
    else
        exit(EXIT_FAILURE);

    printf("Process %d Finished\n",getpid() );
    exit(EXIT_SUCCESS);

}


//Consumer
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<limits.h>
#include<fcntl.h>
#include<sys/msg.h>
#include<sys/stat.h>
#include<string.h>

#define FIFO_NAME "my_fifo"
#define BUFFER_SIZE 1000

int main(int argc, char *argv[])
{
    int pipe_fd;
    int res;
    int open_mode=O_RDONLY;
    int n=0;
    char buffer[BUFFER_SIZE+1];
    memset(buffer,'\0',sizeof(buffer));

    printf("Process %d opening FIFO O_RDONLY\n",getpid());
```

```
        pipe_fd=open(FIFO_NAME,open_mode);
        printf("Process %d result %d\n",getpid(),pipe_fd);

        if (pipe_fd!=-1)
        {
            do
            {
                res=read(pipe_fd,buffer,BUFFER_SIZE);
                printf("%s\n",buffer );
                n++;
            }while(n<4);

            (void)close(pipe_fd);
        }

        else
            exit(EXIT_FAILURE);

        printf("Process %d Finished, %d bytes
read\n",getpid(),n );
        exit(EXIT_SUCCESS);
    }
```

Output:



2. Demostrate creation, writing to, and reading from a pipe.

Program:

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

```
#include<sys/msg.h>
#include<string.h>

int main(int argc, char *argv[])
{
   int  n;
   int fd[2];
   char buf[1025];
   char *data="190905513 MOHAMMAD DANISH EQBAL B.TECH LATERAL ENTRY!";
   pipe(fd);
   write(fd[1],data,strlen(data));

   if(n=read(fd[0],buf,1024)>=0)
   {
      buf[n]=0;
      printf("Read %d bytes from pipe\"%s\"\n",n,buf);
   }

   else
      perror("Read");
      exit(0);
}
```

Output:

```
@lplab-ThinkCentre-M71e: ~/Documents/190905513/OS_LAB/LAB5
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB5$ gcc q2.c -o q2
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB5$ ./q2
Read 1 bytes from pipe"1"
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB5$ ./q2
Read 1 bytes from pipe"1"
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB5$
```

3. Write a C program to implement one side of FIFO.

Program:

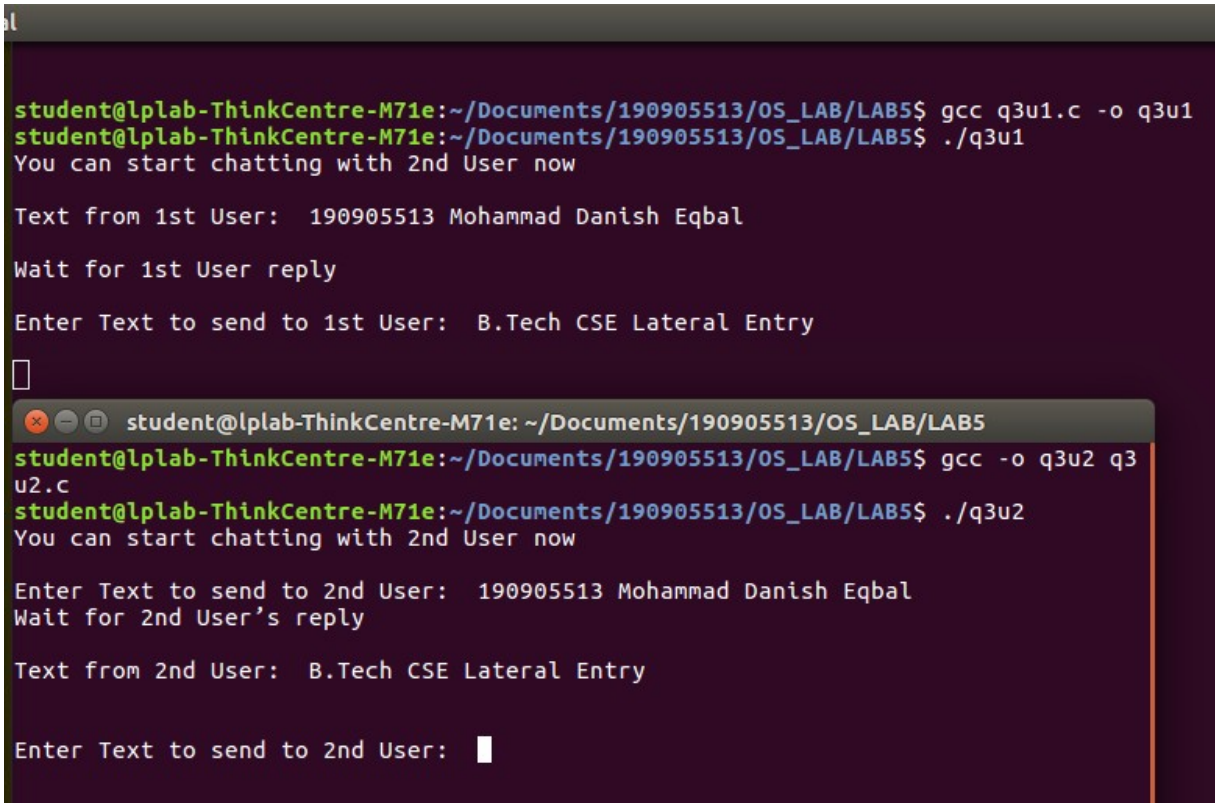
```
//1st User
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<limits.h>
#include<fcntl.h>
#include<sys/msg.h>
#include<sys/stat.h>
#include<string.h>


#define FIFO_NAME "my_fifo"
#define BUFFER_SIZE 10000


int main(int argc, char *argv[])
```

```
    {
        int pipe_fd;
        int res;
        int open_mode1=O_WRONLY;
        int open_mode2=O_RDONLY;
        int n=0;
        char buffer[BUFFER_SIZE+1];

        if(access(FIFO_NAME,F_OK)==-1)
        {
            res=mkfifo(FIFO_NAME,0777);

            if(res!=0)
            {
                fprintf(stderr, "Could not create file
%s\n",FIFO_NAME);
                exit(EXIT_FAILURE);
            }
        }

        printf("You can start chatting with 2nd User now\n");

        while(1)
        {
            pipe_fd=open(FIFO_NAME,open_mode2);

            printf("\nText from 1st User:  ");
            res=read(pipe_fd,buffer,BUFFER_SIZE);
            printf("%s\n",buffer );
            close(pipe_fd);

            printf("Wait for 1st User reply\n");
            pipe_fd=open(FIFO_NAME,open_mode1);

            printf("\nEnter Text to send to 1st User:  ");
            fgets(buffer,BUFFER_SIZE,stdin);
            res=write(pipe_fd,buffer,BUFFER_SIZE);

            close(pipe_fd);


        }

        (void)close(pipe_fd);


        printf("Process %d Finished\n",getpid());
        exit(EXIT_SUCCESS);
    }
```

```
//2nd User
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<limits.h>
#include<fcntl.h>
#include<sys/msg.h>
#include<sys/stat.h>
#include<string.h>
#define FIFO_NAME "my_fifo"
#define BUFFER_SIZE 10000

int main(int argc, char *argv[])
{
    int pipe_fd;
    int res;
    int open_mode1=O_WRONLY;
    int open_mode2=O_RDONLY;
    int n=0;
    char buffer[BUFFER_SIZE+1];

    if(access(FIFO_NAME,F_OK)==-1)
    {
        res=mkfifo(FIFO_NAME,0777);
        if(res!=0)
        {
            fprintf(stderr, "Could not create file
%s\n",FIFO_NAME );
            exit(EXIT_FAILURE);
        }
    }

    printf("You can start chatting with 2nd User now\n");

    while(1)
    {
        pipe_fd=open(FIFO_NAME,open_mode1);
        printf("\nEnter Text to send to 2nd User:  ");
        fgets(buffer,BUFFER_SIZE,stdin);
        res=write(pipe_fd,buffer,BUFFER_SIZE);

        close(pipe_fd);

        printf("Wait for 2nd User's reply\n");
        pipe_fd=open(FIFO_NAME,open_mode2);

        printf("\nText from 2nd User:  ");
        res=read(pipe_fd,buffer,BUFFER_SIZE);
        printf("%s\n",buffer );
        close(pipe_fd);
```

```
        }
        (void)close(pipe_fd);


        printf("Process %d Finished\n",getpid() );
        exit(EXIT_SUCCESS);
    }
```

Output:



4. Write a C program reading and writing a binary files in C.

Program:

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

int main()
{

    FILE* fptr;
    int num=0;
    fptr=fopen("demo.bin","wb+");

    printf("Enter 7 numbers : \n");
```

```
    for(int i=0;i<7;i++)
    {

        scanf("%d",&num);
        fwrite(&num,sizeof(int),1,fptr);
    }

    printf("Writing done!\n");
    fclose(fptr);

    fptr=fopen("demo.bin","rb");

    for(int i=0;i<7;i++)
    {

        fread(&num,sizeof(int),1,fptr);
        printf("%d\n",num);
    }

}
```

Output: