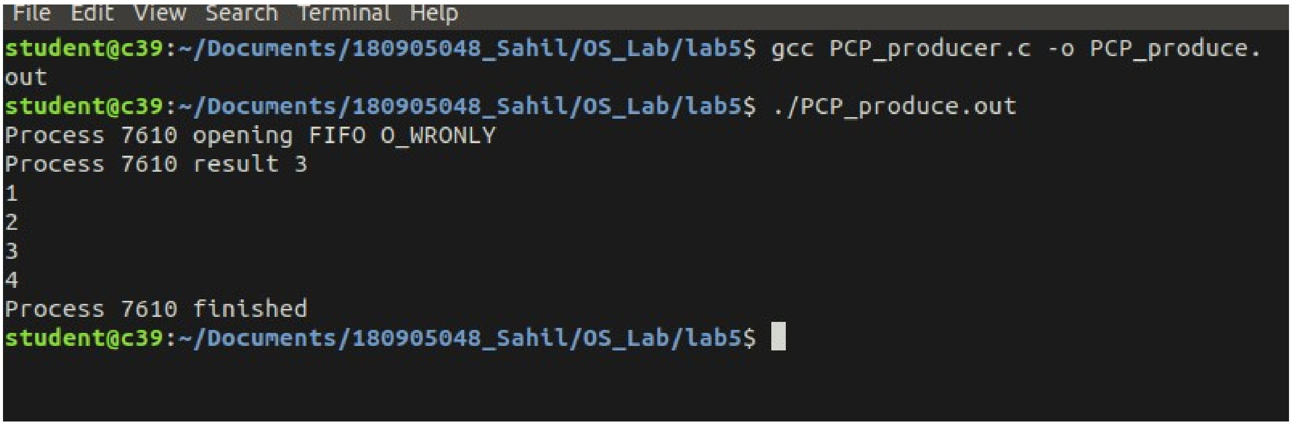
OS LAB 5

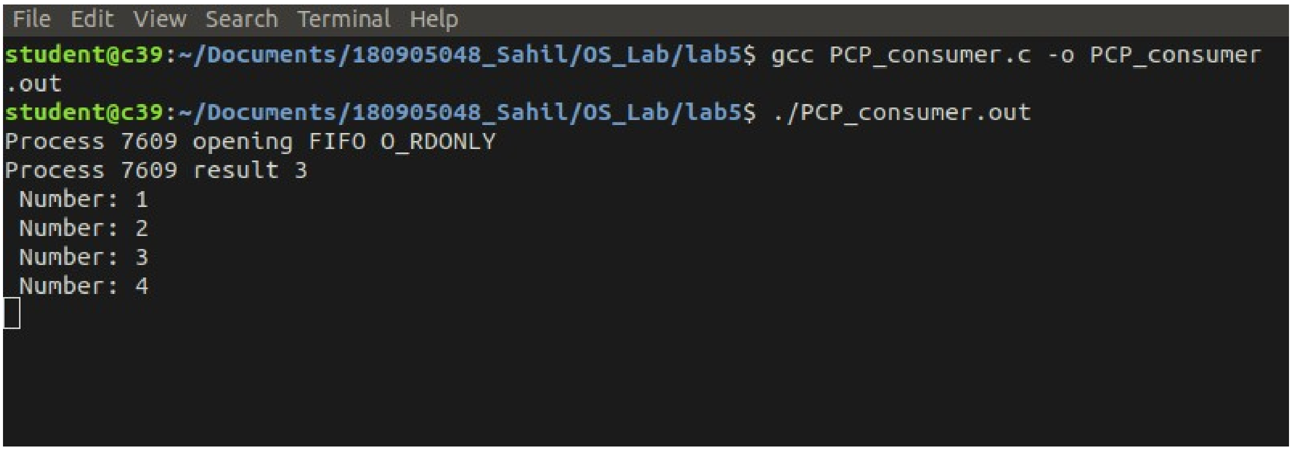
Sahil Saini Salaria
Reg No 180905048
Roll No 11C
Batch 5th
Lab OS

Producer

```
// Run Consumer code first then the Producer code
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <limits.h>
#include <sys/types.h>
#include <sys/stat.h>

#define FIFO_NAME "/tmp/my_fifo"
#define BUFFER_SIZE PIPE_BUF
#define TEN_MEG (1024 * 1024 * 10)

int main(int argc, char const *argv[])
{
    int pipe_fd;
    int res;
    int open_mode = O_WRONLY;
    int bytes_sent = 0;
    int buffer;

    if (access(FIFO_NAME, F_OK) == -1)
    {
        res = mkfifo(FIFO_NAME, 0777);
        if (res != 0)
        {
            fprintf(stderr, "Could not create fifo %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)
    {
        int count=0;
        while (count < 4)
        {
            scanf("%d",&buffer);
            res = write(pipe_fd, &buffer, sizeof(buffer));
            if (res == -1)
            {
                fprintf(stderr, "Write error on pipe \n");
                exit(EXIT_FAILURE);
            }
             count++;
        }
        (void)close(pipe_fd);
    }
    else
        exit(EXIT_FAILURE);

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

    return 0;
}


//Consumer

// Run Consumer code first then the Producer code
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <limits.h>

#define FIFO_NAME "/tmp/my_fifo"
#define BUFFER_SIZE PIPE_BUF

int main(int argc, char const *argv[])
{
    int pipe_fd;
    int res;
    int open_mode = O_RDONLY;
    int buffer;
    int bytes_read = 0;

    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)
    {
        res = read(pipe_fd, &buffer, sizeof(buffer));
        while(res > 0)
        {
            printf(" Number: %d \n",buffer);
            res = read(pipe_fd, &buffer, sizeof(buffer));
            bytes_read += res;
        }
        (void)close(pipe_fd);
    }
    else
        exit(EXIT_FAILURE);

    printf("Process %d finished %d bytes read \n", getpid(), bytes_read);
    exit(EXIT_FAILURE);
    return 0;
}
```



```
student@c39:~/Documents/180905048_Sahil/OS_Lab/lab5$ gcc PCP_producer.c -o PCP_produce.
out
student@c39:~/Documents/180905048_Sahil/OS_Lab/lab5$ ./PCP_produce.out
Process 7610 opening FIFO O_WRONLY
Process 7610 result 3
1
2
3
4
Process 7610 finished
student@c39:~/Documents/180905048_Sahil/OS_Lab/lab5$
```

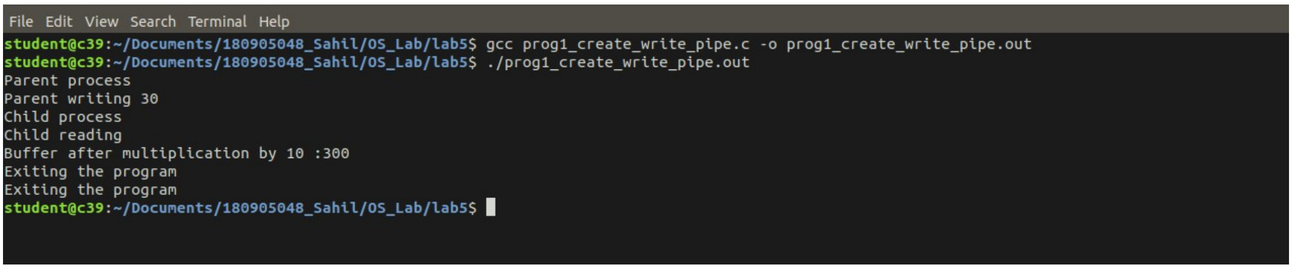


```
student@c39:~/Documents/180905048_Sahil/OS_Lab/lab5$ gcc PCP_consumer.c -o PCP_consumer
.out
student@c39:~/Documents/180905048_Sahil/OS_Lab/lab5$ ./PCP_consumer.out
Process 7609 opening FIFO O_RDONLY
Process 7609 result 3
 Number: 1
 Number: 2
 Number: 3
 Number: 4
```

Q2

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<fcntl.h>
#include<unistd.h>
#include<error.h>
#include<sys/wait.h>
```

```
int main()
{
    int arr[2];

    // Creating the pipe
    int fd=pipe(arr);

    if(fd ==-1)
    {
        printf("Can't create a pipe\n");
        exit(1);
    }
    int f=fork();

    if(f==-1)
    {
        printf("Can't create a child process\n");
        exit(2);
    }
    else if (f==0)
    {
        printf("Child process\n");
        close(arr[1]);
        int buffer;
        printf("Child reading\n");

        // Reading from the pipe
        int r= read( arr[0] , &buffer, sizeof(buffer));
        buffer=buffer*10;
        printf("Buffer after multiplication by 10 :%d \n",buffer);
        close(arr[0]);
    }
    else
    {
        printf("Parent process\n");
        close(arr[0]);
        int buffer=30;
        printf("Parent writing %d \n", buffer);

        // Writing to the pipe
        int w=write(arr[1], &buffer, sizeof(buffer));
        close(arr[0]);
        wait(NULL);
    }

    printf("Exiting the program\n");
    return 0;


}
```

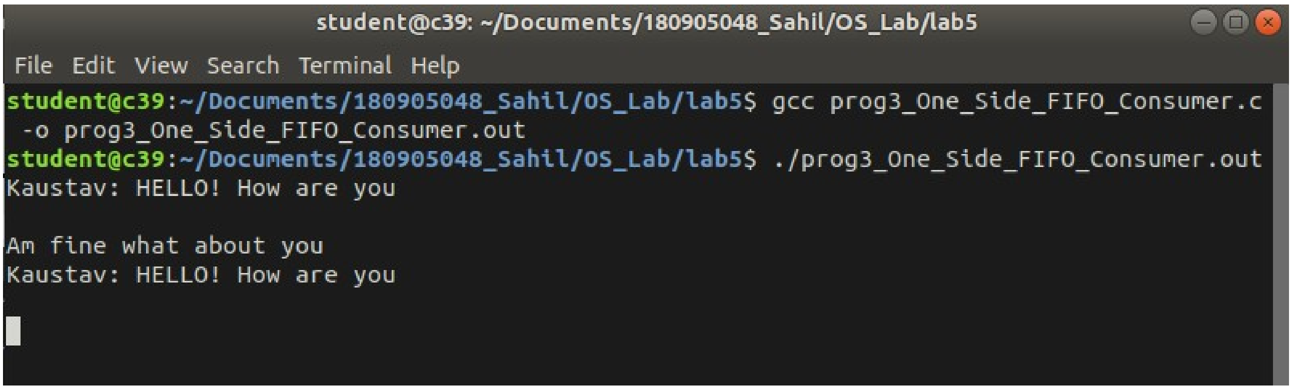


```
student@c39:~/Documents/180905048_Sahil/OS_Lab/lab5$ gcc prog1_create_write_pipe.c -o prog1_create_write_pipe.out
student@c39:~/Documents/180905048_Sahil/OS_Lab/lab5$ ./prog1_create_write_pipe.out
Parent process
Parent writing 30
Child process
Child reading
Buffer after multiplication by 10 :300
Exiting the program
Exiting the program
student@c39:~/Documents/180905048_Sahil/OS_Lab/lab5$ 
```

Q3

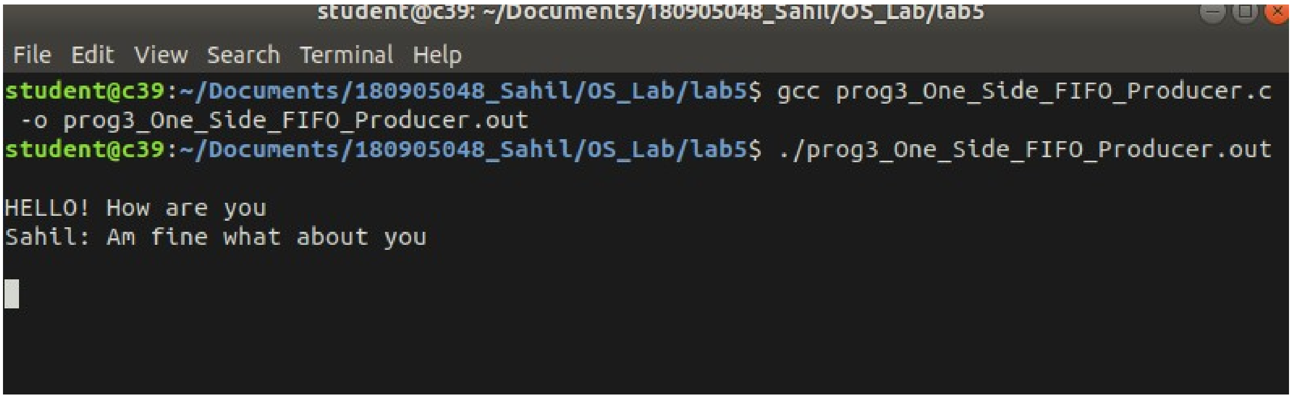
```
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<limits.h>
#include<fcntl.h>
#include<string.h>
#include<sys/types.h>
#include<signal.h>

int main()
{
    char * myfifo_file="./myfifo";

    int fd;

   if(access(myfifo_file, F_OK) == -1){
        printf("pipe does not exist\n");
        fd = mkfifo(myfifo_file, 0777);
        if(fd != 0){
            fprintf(stderr, "could not create fifo %s\n", myfifo_file);
            exit(EXIT_FAILURE);
        }
    }

    char arr1[100], arr2[100];

    while (1)
    {

         int f1 = open(myfifo_file, O_RDONLY);
        read(f1,arr1,sizeof(arr1));
        printf("Kaustav: ");
        puts(arr1);
        close(f1);

        int f2 = open(myfifo_file, O_WRONLY);
        fgets(arr2,sizeof(arr2),stdin);
        write(f2,arr2,sizeof(arr2));
        close(f2);;

    }

return 0;

}
```

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

int main()
{
    char * myfifo_file="./myfifo";

    int fd;

    if(access(myfifo_file, F_OK) == -1){
        printf("pipe does not exist\n");
        fd = mkfifo(myfifo_file, 0777);
        if(fd != 0){
            fprintf(stderr, "could not create fifo %s\n", myfifo_file);
            exit(EXIT_FAILURE);
        }
    }

    char arr1[100], arr2[100];

    while (1)
    {

        int f1 = open(myfifo_file, O_WRONLY);
        fgets(arr2,sizeof(arr2),stdin);
        write(f1,arr2,sizeof(arr2));
        close(f1);;

        int f2 = open(myfifo_file, O_RDONLY);
        read(f2,arr1,sizeof(arr1));
        printf("Sahil: ");
        puts(arr1);
        close(fd);


    }

    return 0;

}
```



student@c39: ~/Documents/180905048_Sahil/OS_Lab/lab5
File Edit View Search Terminal Help

```
student@c39:~/Documents/180905048_Sahil/OS_Lab/lab5$ gcc prog3_One_Side_FIFO_Consumer.c
 -o prog3_One_Side_FIFO_Consumer.out
student@c39:~/Documents/180905048_Sahil/OS_Lab/lab5$ ./prog3_One_Side_FIFO_Consumer.out
Kaustav: HELLO! How are you

Am fine what about you
Kaustav: HELLO! How are you

```



student@c39: ~/Documents/180905048_Sahil/OS_Lab/lab5
File Edit View Search Terminal Help

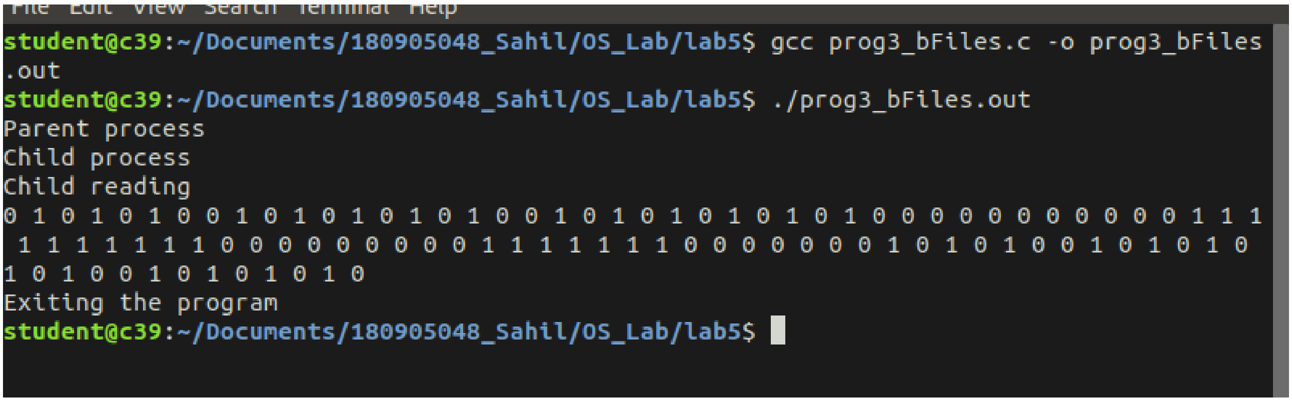
```
student@c39:~/Documents/180905048_Sahil/OS_Lab/lab5$ gcc prog3_One_Side_FIFO_Producer.c
 -o prog3_One_Side_FIFO_Producer.out
student@c39:~/Documents/180905048_Sahil/OS_Lab/lab5$ ./prog3_One_Side_FIFO_Producer.out

HELLO! How are you
Sahil: Am fine what about you

```

Q4

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<fcntl.h>
#include<unistd.h>
#include<error.h>
#include<sys/wait.h>

int main()
{
```

```
int arr[2];

// Creating the pipe
int fd=pipe(arr);


if(fd ==-1)
{
    printf("Can't create a pipe\n");
    exit(1);
}
int f=fork();

if(f==-1)
{
    printf("Can't create a child process\n");
    exit(2);
}
else if (f==0)
{
    printf("Child process\n");
    close(arr[1]);
    char buffer[4096];
    memset(buffer, '\0', sizeof(buffer));
    printf("Child reading\n");

    // Reading from the pipe
    int r= read( arr[0] , buffer, sizeof(buffer));
    FILE * fw= fopen("pro3_bin2.bin", "wb+");
    int w= fwrite(buffer,sizeof(buffer),100,fw);
    for (int i = 0; i < 100; i++)
    {
        printf("%c ", buffer[i]);
    }
    close(arr[0]);
    return 0;
}
else
{
    printf("Parent process\n");
    close(arr[0]);
    char buffer[4096];
    memset(buffer, '\0', sizeof(buffer));
    FILE *f1=fopen("pro3_bin.bin","rb");
    int r = fread(buffer,sizeof(buffer),100,f1);

    // Writing to the pipe
    int w=write(arr[1], buffer, sizeof(buffer));
    close(arr[0]);
    wait(NULL);
}
```

```
    printf("\nExiting the program\n");
    return 0;


}
```



```
student@c39:~/Documents/180905048_Sahil/OS_Lab/lab5$ gcc prog3_bFiles.c -o prog3_bFiles
.out
student@c39:~/Documents/180905048_Sahil/OS_Lab/lab5$ ./prog3_bFiles.out
Parent process
Child process
Child reading
0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0
1 0 1 0 0 1 0 1 0 1 0 1 0
Exiting the program
student@c39:~/Documents/180905048_Sahil/OS_Lab/lab5$ 
```

//END///