Ayush Goyal
190905522 CSE D Roll 62

Operating Systems Lab 5: IPC-1: PIPE, FIFO

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.

Code:

Producer code: "l5q1prod.c"

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.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_queue"
#define BUFFER_SIZE 1000

int main(int argc, char *argv[]){
	int pipe_fd, res, open_mode = O_WRONLY, 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);
	if(pipe_fd != -1){
		printf("Enter 4 numbers: \n");
		while(n<4){
			printf("%d : ", n+1);
			scanf("%s", buffer);
			res = write(pipe_fd, buffer, BUFFER_SIZE);
			if(res == -1){
				fprintf(stderr, "Write error on PIPE\n");
				exit(EXIT_FAILURE);
			}
			n++;
		}
		close(pipe_fd);
```

```
	}
	else{
		exit(EXIT_FAILURE);
	}
	printf("Process %d finished\n", getpid());
	exit(EXIT_SUCCESS);
	return 0;
}
```

Consumer code: “l5q1con.c”

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.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_queue"
#define BUFFER_SIZE 1000

int main(int argc, char *argv[]){
	int pipe_fd, res, open_mode = O_RDONLY, 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);
	int bytes_read = 0;
	if(pipe_fd != -1){
		do{
			printf("%d : ", n+1);
			res = read(pipe_fd, buffer, BUFFER_SIZE);
			printf("%s\n", buffer);
			bytes_read += BUFFER_SIZE;
			n++;
		}while(n<4);
		close(pipe_fd);
	}
	else{
		exit(EXIT_FAILURE);
	}
	printf("Process %d finished, %d bytes read.\n", getpid(), bytes_read);
	exit(EXIT_SUCCESS);
	return 0;
}
```

Output:

We first compile and run the producer code and then the consumer code. After which we get the option to enter the 4 numbers in the producer window required which are succesfully displayed in the consumer window as shown in the output below.

Producer terminal:

```
ugcse@pglab-cp: ~/Desktop/AyushGoyal_OSLab/Lab_5
ugcse@pglab-cp:~/Desktop/AyushGoyal_OSLab/Lab_5$ gcc l5q1prod.c -o producer1
ugcse@pglab-cp:~/Desktop/AyushGoyal_OSLab/Lab_5$ ./producer1
Process 4122 opening FIFO O_WRONLY
Enter 4 numbers:
1 : 5
2 : 23
3 : 123
4 : 54
Process 4122 finished
ugcse@pglab-cp:~/Desktop/AyushGoyal_OSLab/Lab_5$
```

Consumer terminal:

```
ugcse@pglab-cp: ~/Desktop/AyushGoyal_OSLab/Lab_5
ugcse@pglab-cp:~/Desktop/AyushGoyal_OSLab/Lab_5$ gcc l5q1con.c -o consumer1
ugcse@pglab-cp:~/Desktop/AyushGoyal_OSLab/Lab_5$ ./consumer1
Process 4121 opening FIFO O_RDONLY
ugcse@pglab-cp:~/Desktop/AyushGoyal_OSLab/Lab_5$ ./consumer1
Process 4134 opening FIFO O_RDONLY
1 : 5
2 : 23
3 : 123
4 : 54
Process 4134 finished, 4000 bytes read.
ugcse@pglab-cp:~/Desktop/AyushGoyal_OSLab/Lab_5$
```

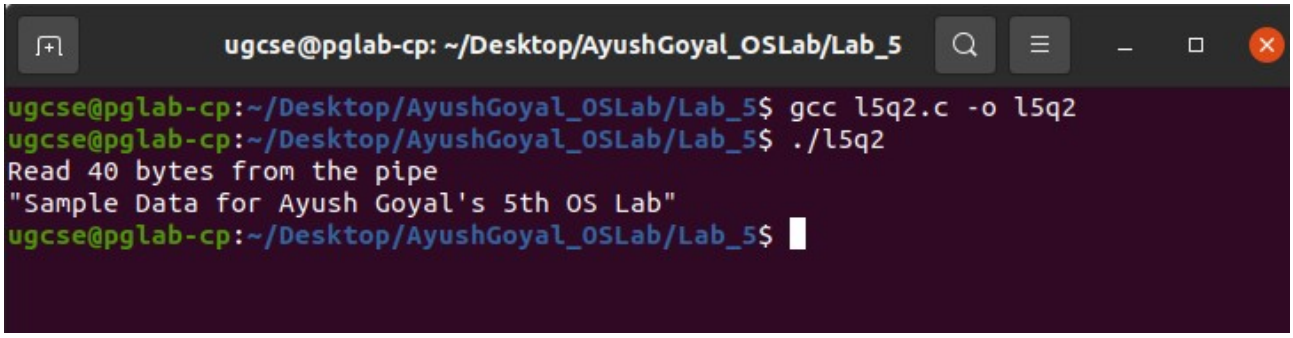
2. Demonstrate creation, writing to, and reading from a pipe.

Code:

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

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

int main(int argc, char const *argv[]){
	int n, fd[2];
	char buf[1025], *data = "Sample Data for Ayush Goyal's 5th OS Lab";
	pipe(fd);
	write(fd[1], data, strlen(data));
	n = read(fd[0], buf, 1024);
	if(n >= 0){
		buf[n] = 0;
		printf("Read %d bytes from the pipe\n\"%s\"\n", n, buf);
	}
	else
		perror("Read");
	exit(0);
}
```

Output:



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

Code:

First Program: “l5q3p1.c”

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.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_queue"
#define BUFFER_SIZE 10000

int main(int argc, char const *argv[]){
	int pipe_fd, res, open_mode1 = O_WRONLY, open_mode2 = O_RDONLY, 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 files%s\n", FIFO_NAME);
			exit(EXIT_FAILURE);
		}
	}

	printf("Creating a program to communicate with another program through FIFO:\n");
	printf("Currently in Program 1 and starting communication with Program 2...\n");
	while(1){
		pipe_fd = open(FIFO_NAME, open_mode2);

		printf("\nText from the Program 2: ");
		res = read(pipe_fd, buffer, BUFFER_SIZE);
		if(strstr(buffer, "exit") != NULL)
			break;
		printf("%s\n", buffer);
		close(pipe_fd);

		pipe_fd = open(FIFO_NAME, open_mode1);

		printf("\nEnter the text to send to Program 2: ");
		fgets(buffer, BUFFER_SIZE, stdin);
		res = write(pipe_fd, buffer, BUFFER_SIZE);

		close(pipe_fd);
	}
	close(pipe_fd);
	printf("Process %d finished.\n", getpid());
	exit(EXIT_SUCCESS);
}
```

Second Program : “l5q3p2.c”

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.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_queue"
#define BUFFER_SIZE 10000

int main(int argc, char const *argv[]){
	int pipe_fd, res, open_mode1 = O_WRONLY, open_mode2 = O_RDONLY, 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 files%s\n", FIFO_NAME);
			exit(EXIT_FAILURE);
		}
	}

	printf("Currently in Program 2 and starting communication with Program 1...\n");
	while(1){
		pipe_fd = open(FIFO_NAME, open_mode1);

		printf("\nEnter the text to send to Program 1: ");
		fgets(buffer, BUFFER_SIZE, stdin);
		res = write(pipe_fd, buffer, BUFFER_SIZE);

		close(pipe_fd);

		pipe_fd = open(FIFO_NAME, open_mode2);

		printf("\nText from the Program 1: ");
		res = read(pipe_fd, buffer, BUFFER_SIZE);
		if(strstr(buffer, "exit") != NULL)
			break;
		printf("%s\n", buffer);

		close(pipe_fd);
	}
	close(pipe_fd);
	printf("Process %d finished.\n", getpid());
	exit(EXIT_SUCCESS);
}
```

Output:

I have compiled and executed Program 1 and then compiled and executed Program 2 concurrently. We can see in the output below that the messages are being inter communicated between both the terminals.

Terminal for Program 1:

```
ugcse@pglab-cp: ~/Desktop/AyushGoyal_OSLab/Lab_5
ugcse@pglab-cp:~/Desktop/AyushGoyal_OSLab/Lab_5$ gcc l5q3p1.c -o prog1
ugcse@pglab-cp:~/Desktop/AyushGoyal_OSLab/Lab_5$ ./prog1
Creating a program to communicate with another program through FIFO:
Currently in Program 1 and starting communication with Program 2...

Text from the Program 2: Hello i am Ayush from 2nd Prog


Enter the text to send to Program 2: I am Sam from Prog1

Text from the Program 2: Are you lonely?


Enter the text to send to Program 2: Yes exit
^C
ugcse@pglab-cp:~/Desktop/AyushGoyal_OSLab/Lab_5$
```

Terminal for Program 2:

```
ugcse@pglab-cp: ~/Desktop/AyushGoyal_OSLab/Lab_5
ugcse@pglab-cp:~/Desktop/AyushGoyal_OSLab/Lab_5$ gcc l5q3p2.c -o prog2
ugcse@pglab-cp:~/Desktop/AyushGoyal_OSLab/Lab_5$ ./prog2
Currently in Program 2 and starting communication with Program 1...

Enter the text to send to Program 1: Hello i am Ayush from 2nd Prog

Text from the Program 1: I am Sam from Prog1


Enter the text to send to Program 1: Are you lonely?

Text from the Program 1: Process 5703 finished.
ugcse@pglab-cp:~/Desktop/AyushGoyal_OSLab/Lab_5$
```

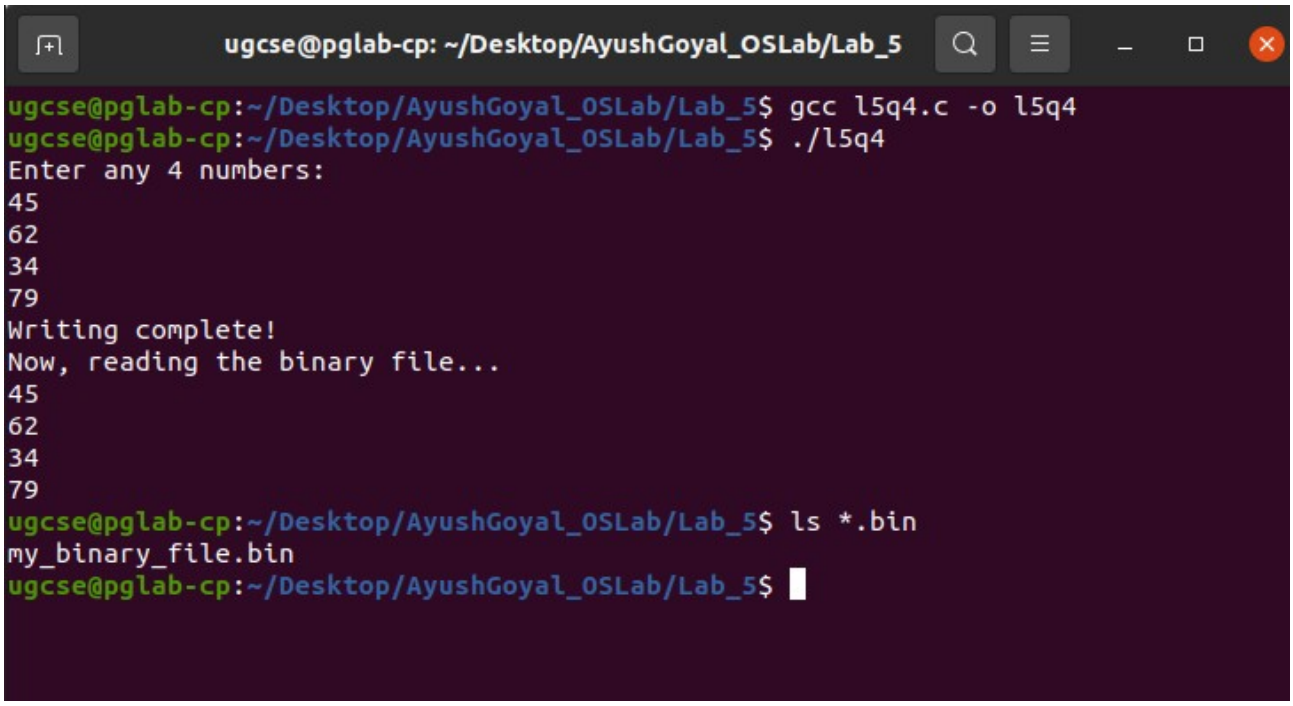
4. Write a C program reading and writing binary files in C.

Code:

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

int main(){
        FILE *fout;
        int num = 0;
```

```
	fout = fopen("my_binary_file.bin", "wb+");

	printf("Enter any 4 numbers: \n");
	for(int i=0;i<4;i++){
		scanf("%d", &num);
		fwrite(&num, sizeof(int), 1, fout);
	}

	printf("Writing complete!\n");
	fclose(fout);

	printf("Now, reading the binary file...\n");

	fout = fopen("my_binary_file.bin", "rb");

	for(int i=0;i<4;i++){
		fread(&num, sizeof(int), 1, fout);
		printf("%d\n", num);
	}
	fclose(fout);

	return 0;
}
```

Output:



THE END