LAB 5

Juhi Mehta
190905412
Roll No: 55
Batch B3

1) Producer Consumer using named pipes

Code:

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

#define FIFO_NAME "my_fifo"

int main(int argc, char * argv[])
{
	int pipe_fd;
	int res;
	int buff[4];

	printf("I am the producer.\n");
	printf("Enter 4 elements: \n");
	for(int i=0;i<4;i++)
		scanf("%d",&buff[i]);

	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);
		}
	}
	pipe_fd = open(FIFO_NAME, O_WRONLY);
	if (pipe_fd > 0)
	{
```

```
                int n = write(pipe_fd, buff, sizeof(buff));
                if (n == -1)
                {
                        fprintf(stderr, "Write Error on Pipe\n");
                        exit(EXIT_FAILURE);
                }
                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 <fcntl.h>
#include <sys/msg.h>
#include <sys/stat.h>
#include <string.h>

#define FIFO_NAME "my_fifo"

int main(int argc, char * argv[])
{
        int pipe_fd;
        int res;
        int buff[4];

        printf("I am the consumer.\n");

        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);
                }
        }
        pipe_fd = open(FIFO_NAME, O_RDONLY);
        if (pipe_fd > 0)
        {
                int n = read(pipe_fd, buff, sizeof(buff));
                if (n == -1)
                {
                        fprintf(stderr, "Read Error on Pipe\n");
                        exit(EXIT_FAILURE);
                }
```

```
                printf("%d bytes read\n", n);
                for(int i=0;i<4;i++)
                                printf("%d ",buff[i]);
                        printf("\n");
                close(pipe_fd);
        }
        else
        exit(EXIT_FAILURE);
        printf("Process %d Finished\n", getpid());
        exit(EXIT_SUCCESS);
}
```

Output:

```
Student@dblab-hp-21:~/Desktop/190905412/OS/Lab_5$ gcc -o producer producer.c
Student@dblab-hp-21:~/Desktop/190905412/OS/Lab_5$ ./producer
I am the producer.
Enter 4 elements:
1 2 3 4
Process 5017 Finished
Student@dblab-hp-21:~/Desktop/190905412/OS/Lab_5$
```

```
Student@dblab-hp-21:~/Desktop/190905412/OS/Lab_5$ gcc -o consumer consumer.c
Student@dblab-hp-21:~/Desktop/190905412/OS/Lab_5$ ./consumer
I am the consumer.
16 bytes read
1 2 3 4
Process 5025 Finished
Student@dblab-hp-21:~/Desktop/190905412/OS/Lab_5$
```

2) Read 4 integers in child process and write in parent (using printf)

Code:

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

int main(int argc, char *argv[])
{
        int pfd[2];
        int buff[4];
        int buff1[4];
        pid_t cpid;
        printf("Enter 4 elements: \n");
        for(int i=0;i<4;i++)
                scanf("%d",&buff[i]);
```

```
	if(argc !=1)
	{
		fprintf(stderr, "Usage: %s <pathname>\n", argv[0]);
		exit(EXIT_FAILURE);
	}

	else
	{
		if(pipe(pfd)==-1)
		{
			perror("pipe error");
			exit(EXIT_FAILURE);
		}
		cpid = fork();
		if(cpid == -1)
		{
			perror("Fork error");
			exit(EXIT_FAILURE);
		}
		if(cpid == 0)
		{
			close(pfd[1]);
			int n= read(pfd[0],buff1,sizeof(buff1));
			printf("Output as read by child is: \n");
			for(int i=0;i<4;i++)
				printf("%d ",buff1[i]);
			printf("\n");
			close(pfd[0]);
			exit(EXIT_SUCCESS);
		}
		else
		{
			close(pfd[0]);
			write(pfd[1],buff,sizeof(buff));
			close(pfd[1]);
			wait(NULL);
			exit(EXIT_SUCCESS);
		}
	}
}
```

Output:

```
Student@dblab-hp-21:~/Desktop/190905412/OS/Lab_5$ gcc -o l5q1 l5q1.c
Student@dblab-hp-21:~/Desktop/190905412/OS/Lab_5$ ./l5q1
Enter 4 elements:
1 2 3 4
Output as read by child is:
1 2 3 4
Student@dblab-hp-21:~/Desktop/190905412/OS/Lab_5$
```

3) Implement 1 side of FIFO

Code:

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

#define FIFO_NAME "my_fifo"

int main(int argc, char * argv[])
{
	int pipe_fd;
	int res;
	int buff[4];

	printf("I am p1.\n");

	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);
		}
	}
	pipe_fd = open(FIFO_NAME, O_WRONLY);
	if (pipe_fd > 0)
	{
```

```
            int n = write(pipe_fd, buff, sizeof(buff));
            if (n == -1)
            {
                  fprintf(stderr, "Write Error on Pipe\n");
                  exit(EXIT_FAILURE);
            }
            close(pipe_fd);
      }
      else
      exit(EXIT_FAILURE);
      printf("Process %d Finished\n", getpid());
      exit(EXIT_SUCCESS);
}
```