

# OS Lab 5

Reg No: 180905416  
Name: Paawan Kohli

**Q1. 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.**

**producer.c:**

```
#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() {
    if (access(FIFO_NAME, F_OK) == -1) {
        int status = mkfifo(FIFO_NAME, 0777);

        if (status != 0) {
            printf("Could not create fifo %s\n", FIFO_NAME);
            exit(EXIT_FAILURE);
        }
    }

    printf("Enter 4 integers: ");
    int buffer[BUFFER_SIZE + 1];
    for (int i = 0; i < 4; i++) {
        scanf("%d", &buffer[i]);
    }

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

    int bytes_sent = 0;

    if (pipe_fd != -1) {
        while (bytes_sent < TEN_MEG) {

            int status = write(pipe_fd, buffer, BUFFER_SIZE);

            if (status == -1) {
                fprintf(stderr, "Write error on pipe\n");
                exit(EXIT_FAILURE);
            }

            bytes_sent += status;
        }
        close(pipe_fd);
    }
    else {
        exit(EXIT_FAILURE);
    }
}
```

```

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

```

#### consumer.c:

```

#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

int main() {
    int pipe_fd = open(FIFO_NAME, O_RDONLY);
    int bytes_read = 0;

    if (pipe_fd != -1) {
        int buffer[BUFFER_SIZE + 1];
        memset(buffer, '\0', sizeof(buffer));

        int status;

        do {
            status = read(pipe_fd, buffer, BUFFER_SIZE);
            bytes_read += status;
        } while (status > 0);

        close(pipe_fd);

        for (int i = 0; i < 4; i++) {
            printf("%d ", buffer[i]);
        }
        printf("\n");
    }
    else {
        exit(EXIT_FAILURE);
    }

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

```

Terminal Window	Command	Output
Left (prod)	<code>./prod</code>	Process 6374 opening FIFO O_WRONLY Enter 4 integers: 1 2 3 4 Process 6374 result 3 Process 6374 finished
Right (cons)	<code>./cons</code>	Process 6375 opening FIFO O_RDONLY Process 6375 result 3 1 2 3 4 Process 6375 finished, 10485760 bytes read

## Q2. Demonstrate creation, writing to and reading from a pipe.

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

char msg1[] = "this is message #1";
char msg2[] = "this is message #2";
char msg3[] = "this is message #3";

int main() {
    // creating a pipe
    int fd[2];
    if (pipe(fd) == -1) {
        printf("Pipe failed\n");
        exit(EXIT_FAILURE);
    }

    pid_t pid = fork();

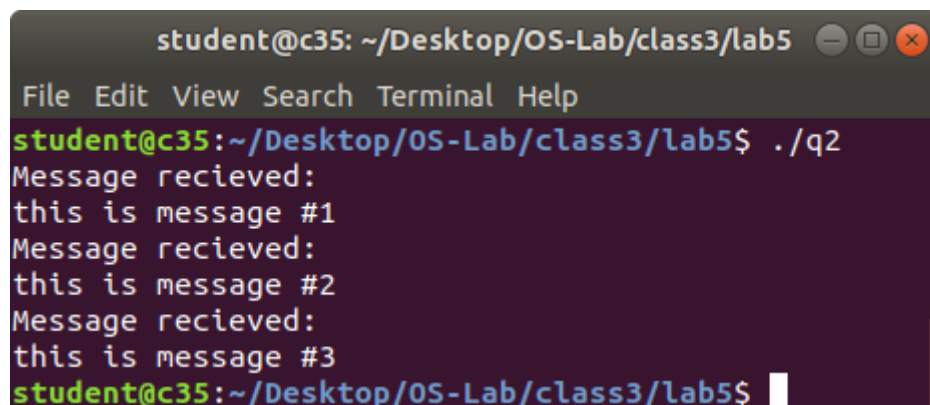
    if (pid == -1) {
        printf("Fork failed\n");
        exit(EXIT_FAILURE);
    }

    // parent writes to the pipe
    if (pid != 0) {
        sleep(1);
        write(fd[1], msg1, 100);
        sleep(1);
        write(fd[1], msg2, 100);
        sleep(1);
        write(fd[1], msg3, 100);

        wait(NULL);
    }
    // child reads from the pipe
    else {
        char buff[100];

        for (int i = 0; i < 3; ++i) {
            printf("Message recieved: \n");
            read(fd[0], buff, 100);
            printf("%s\n", buff);
        }

        exit(0);
    }
}
```

A terminal window titled 'student@c35: ~/Desktop/OS-Lab/class3/lab5' with standard window controls. The terminal shows the execution of the program './q2'. The output consists of three lines, each preceded by 'Message recieved:'. The messages are 'this is message #1', 'this is message #2', and 'this is message #3'. The prompt 'student@c35:~/Desktop/OS-Lab/class3/lab5\$' is visible at the bottom.

```
student@c35: ~/Desktop/OS-Lab/class3/lab5
File Edit View Search Terminal Help
student@c35:~/Desktop/OS-Lab/class3/lab5$ ./q2
Message recieved:
this is message #1
Message recieved:
this is message #2
Message recieved:
this is message #3
student@c35:~/Desktop/OS-Lab/class3/lab5$
```

**Q3. Write a C program to implement one side of FIFO.**

```
#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>
#include <sys/wait.h>

#define FIFO_NAME "/tmp/myfifo"
#define BUFFER_SIZE 20

int main() {

    int mode;
    printf("Enter 1 for write first, 2 for read first: ");
    scanf("%d", &mode);

    if (mode == 1) {
        if (access(FIFO_NAME, F_OK) == -1) {
            int status = mkfifo(FIFO_NAME, 0777);
            if (status != 0) {
                printf("Could not create fifo %s\n", FIFO_NAME);
                exit(EXIT_FAILURE);
            }
        }
    }

    int pipe_fd;

    while (mode != 0) {
        if (mode == 1) {

            char buffer1[BUFFER_SIZE + 1];
            printf("Enter msg: ");
            scanf("%s", buffer1);

            pipe_fd = open(FIFO_NAME, O_WRONLY);
            if (pipe_fd != -1) {

                int status = write(pipe_fd, buffer1, BUFFER_SIZE);

                if (status == -1) {
                    printf("Error! Can't write to pipe\n");
                    exit(EXIT_FAILURE);
                }

                close(pipe_fd);
                mode = 2;

                if (strcmp(buffer1, "quit") == 0) {
                    mode = 0;
                }
            }
        }
        else {
            printf("Can't open the pipe\n");
            exit(EXIT_FAILURE);
        }
    }
}
```

```

else if (mode == 2) {

    char buffer2[BUFFER_SIZE + 1];
    memset(buffer2, '\0', sizeof(buffer2));

    pipe_fd = open(FIFO_NAME, O_RDONLY);

    if (pipe_fd != -1) {
        int status;
        do {
            status = read(pipe_fd, buffer2, BUFFER_SIZE);
        } while (status > 0);

        close(pipe_fd);
        mode = 1;
        printf("%s \n", buffer2);

        if (strcmp(buffer2, "quit") == 0) {
            mode = 0;
        }
    }
    else {
        printf("Can't open the pipe\n");
        exit(EXIT_FAILURE);
    }
}

exit(EXIT_SUCCESS);
}

```

The image shows two terminal windows side-by-side, both titled 'student@c35: ~/Desktop/OS-Lab/class3/lab5'. Both windows have a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'.

The left terminal window shows the following interaction:

```

student@c35:~/Desktop/OS-Lab/class3/lab5$ ./q3
Enter 1 for write first, 2 for read first: 1
Enter msg: hey
yo
Enter msg: dog
cat
Enter msg: bye
quit
student@c35:~/Desktop/OS-Lab/class3/lab5$

```

The right terminal window shows the following interaction:

```

student@c35:~/Desktop/OS-Lab/class3/lab5$ ./q3
Enter 1 for write first, 2 for read first: 2
fihey
Enter msg: yo
dog
Enter msg: cat
bye
Enter msg: quit
it
student@c35:~/Desktop/OS-Lab/class3/lab5$

```

**Q4. Write a C program for reading and writing binary files in C.**

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

int main() {
    // open a binary file in wb mode
    FILE *fptr = fopen("hey.bin", "wb");

    if (fptr == NULL) {
        printf("Error! opening file");
        exit(1);
    }

    int n = 5;
    int num[3];

    num[0] = 34;
    num[1] = 21;
    num[2] = 65;

    fwrite(num, sizeof(num), 1, fptr);
    fclose(fptr);

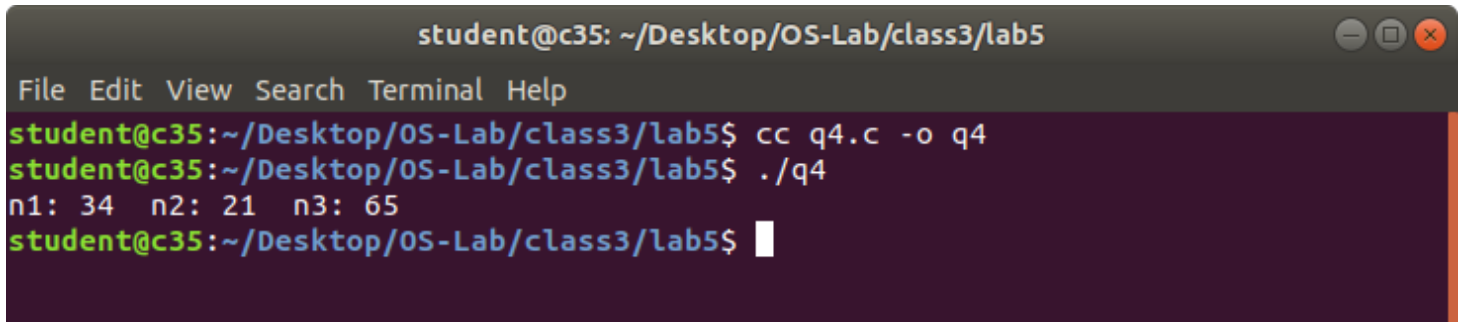
    // open the same binary file in rb mode
    FILE* fptr2 = fopen("hey.bin", "rb");

    if (fptr2 == NULL) {
        printf("Error! opening file");
        exit(1);
    }

    fread(num, sizeof(num), 1, fptr2);
    fclose(fptr2);

    printf("n1: %d\tn2: %d\tn3: %d\n", num[0], num[1], num[2]);

    return 0;
}
```



A terminal window titled "student@c35: ~/Desktop/OS-Lab/class3/lab5" with standard window controls. The terminal shows the following commands and output:

```
student@c35:~/Desktop/OS-Lab/class3/lab5$ cc q4.c -o q4
student@c35:~/Desktop/OS-Lab/class3/lab5$ ./q4
n1: 34  n2: 21  n3: 65
student@c35:~/Desktop/OS-Lab/class3/lab5$
```