

B.M.S. College of Engineering

P.O. Box No.: 1908 Bull Temple Road,

Bangalore-560019

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING



Course-Unix System Programming

Course Code-23IS3AEUSP

AY 2024-25

Report on Unix System Programming Project

First In First Out

Submitted By:

Devansh Agrawal	1BM23IS077
Hamzah Ahmad	1BM23IS100
Imaddudin Dakaray	1BM23IS110
Jaydev Adhikari	1BM23IS115

Submitted To:

Rudramurthy V C

Assistant Professor

B.M.S. College of Engineering

P.O. Box No.: 1908 Bull Temple Road,

Bangalore-560019

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING



CERTIFICATE

Certified that the Project has been successfully presented at B.M.S College Of Engineering by **Devansh Agrawal, Hamzah Ahmad, Imaddudin Dakaray and Jaydev Adhikari** bearing USN: **1BM23IS077, 1BM23IS100, 1BM23IS110 and 1BM23IS115** in partial fulfilment of the requirements for the III Semester degree in Bachelor of Engineering in Information Science & Engineering of Visvesvaraya Technological University, Belgaum as a part of project for the Course– Unix System Programming Course Code – 23IS3AEUSP during academic year 2024-2025.

Faculty Name – Rudramurthy V C

Designation – Assistant Professor
Department of ISE, BMSCE.

Contents

Chapter No.	Chapters	Page No.
	Abstract	
1	Introduction	5
2	Problem Statement	6
3	API's and Process Control used	7
4	Implementation	10
5	Result	13
6	Conclusion and References	14

ABSTRACT

This project investigates the implementation of a **teacher-student interaction system** utilizing **named pipes (FIFOs)** for inter-process communication (IPC) in a Unix environment. The primary goal is to simulate a dynamic real-time quiz system where a teacher interacts with two students by exchanging questions and answers through IPC mechanisms.

The implementation comprises three key components: **teacher.c**, **student1.c**, and **student2.c**. The teacher process sends questions to both students, collects their responses, evaluates their correctness, and updates scores accordingly. Named pipes are employed to enable seamless communication between processes while maintaining isolation.

This report explores the design considerations, such as FIFO creation, process control, and error handling, alongside challenges like sequential dependencies in responses. It also discusses the practical applications of such systems in learning environments, emphasizing their scalability and adaptability to real-world scenarios.

In conclusion, this project demonstrates the utility of named pipes for IPC in Unix-based systems, showcasing their role in creating efficient, interactive systems for educational and training purposes.

CHAPTER 1

INTRODUCTION

1.1 Named Pipes and IPC

Named pipes, also known as FIFOs, are a method for inter-process communication that allows processes to exchange data in a first-in-first-out order. Unlike anonymous pipes, named pipes persist in the filesystem, enabling unrelated processes to communicate effectively.

1.2 Objectives

The project aims to simulate a teacher-student interaction system to:

- Facilitate question-answer exchanges between processes.
- Evaluate students' responses in real-time.
- Maintain scores and handle end conditions dynamically.

1.3 Requirements

- Input Files:
 - **questions.txt**: Contains quiz questions.
 - **answers.txt**: Contains corresponding correct answers.
- FIFOs: Named pipes **/tmp/myfifo**, **/tmp/myfifo1**, **/tmp/myfifo3**, and **/tmp/myfifo4**.
- Programs:
 - **teacher.c**: Manages questions and evaluations.
 - **student1.c** and **student2.c**: Receive questions and send responses.

CHAPTER 2

DEFINING THE PROBLEM STATEMENT

2.1 Problem Description

Design a real-time quiz system that:

- Uses named pipes for communication between processes.
- Allows students to respond to questions and receive evaluations.
- Records scores in a file for later review.

2.2 Requirements

1. Communication:
 - Bi-directional communication using FIFOs.
2. Operations:
 - Send and receive questions and answers.
 - Evaluate correctness and update scores dynamically.
3. Error Handling:
 - Ensure smooth operation with appropriate error messages.

CHAPTER 3

API AND PROCESS CONTROL USED

3.1 File API

File APIs were essential for managing the interaction between the application and external files (**questions.txt**, **answers.txt**, and **marks.txt**). The following functions were utilized:

fopen:

Opens files in the required mode (**r** for reading and **w** for writing). It is used to access the question and answer files as well as the output file where scores are stored.

Example: `FILE *f_questions = fopen("questions.txt", "r");`

fgets:

Reads a line from a file into a buffer. This function is used to read each question and its corresponding correct answer line-by-line.

Example: `fgets(buffer, sizeof(buffer), f_questions);`

fprintf:

Writes formatted output to a file. This is used to save the final scores of both students and the unanswered question count into **marks.txt**.

Example: `fprintf(f1, "Student 1: %d\n", mark1);`

fclose:

Closes an opened file and releases associated resources, ensuring no memory leaks.

Example: `fclose(f_questions);`

These APIs ensure efficient file handling for input and output operations, forming the backbone of the system's file interaction.

3.2 Process Control

Process control involves the creation, management, and synchronization of communication channels between the teacher and student processes. This was achieved using **Named Pipes (FIFOs)**:

1. Named Pipes (FIFOs):

FIFOs enable inter-process communication by providing a persistent, first-in-first-out communication channel. The following FIFO-related functions were used:

➤ **mkfifo**: Creates a named pipe with specified permissions (e.g., **0666** for read/write access to all users).

Example: **mkfifo("/tmp/myfifo", 0666);**

➤ **open**: Opens the FIFO in either **O_RDONLY** (read-only) or **O_WRONLY** (write-only) mode.

Example: **fd = open("/tmp/myfifo", O_WRONLY);**

➤ **read**: Reads data from the FIFO into a buffer. This is used to receive questions and answers.

Example: **read(fd, buffer, sizeof(buffer));**

➤ **write**: Writes data from a buffer into the FIFO. This is used to send questions and answers.

Example: **write(fd, buffer, strlen(buffer) + 1);**

➤ **close**: Closes the FIFO after each operation, releasing system resources.

Example: **close(fd);**

2. **Synchronization:**

Synchronization was critical to ensure proper sequencing of communication:

The teacher writes a question and waits for responses from both students. Each student waits to receive the question before sending their answer. This ensures a structured interaction between processes.

3. **Error Handling:**

Proper error handling is implemented for file and FIFO operations to avoid crashes or undefined behavior:

Example:

```
if (f_questions == NULL) {  
    perror("Error opening file");  
    return 1;  
}
```

4. **Console Interaction:**

fgets is used for student input (answers), and **printf** is used to display questions and responses, aiding debugging and real-time interaction.

CHAPTER 4

IMPLEMENTATION

4.1 Teacher Process (**teacher.c**)

The teacher process acts as the central controller of the system. It performs the following functions:

1. Reading Input Files:

Opens **questions.txt** and **answers.txt** to fetch the questions and their corresponding correct answers.

```
FILE *f_questions = fopen("questions.txt", "r");
```

2. Sending Questions to Students:

Sends each question through FIFOs (**/tmp/myfifo** and **/tmp/myfifo1**) to the two student processes.

```
fd      =      open("/tmp/myfifo",      O_WRONLY);write(fd,  
question, strlen(question) + 1);  
close(fd);
```

3. Receiving Student Responses:

Reads responses sent by both students through FIFOs (**/tmp/myfifo3** and **/tmp/myfifo4**).

```
fd_ts1 = open("/tmp/myfifo3", O_RDONLY);read(fd_ts1,  
student1_response, sizeof(student1_response));  
close(fd_ts1);
```

4. Evaluating Answers:

Compares student responses with the correct answer. Updates the respective scores or increments the count of unanswered questions.

```
if (strcmp(student1_response, correct_answer) == 0)
    mark1++;
```

5. Writing Scores to File:

Saves the final scores into **marks.txt**.

```
FILE *f1 = fopen("marks.txt", "w");
fprintf(f1, "Student 1: %d\n", mark1);
fclose(f1);
```

4.2 Student Processes (**student1.c**, **student2.c**)

Each student process performs the following tasks:

1. Receiving Questions:

Read the question sent by the teacher via FIFOs.

```
fd = open("/tmp/myfifo1", O_RDONLY);
read(fd, question, sizeof(question));
close(fd);
```

2. Inputting Responses:

Reads user input for the answer via **fgets**.

```
fgets(answer, sizeof(answer), stdin);
```

3. Sending Responses:

Writes the response back to the teacher via FIFOs.

```
fd_st = open("/tmp/myfifo4", O_WRONLY);  
write(fd_st, answer, strlen(answer) + 1);  
close(fd_st);
```

4.3 Interaction Flow

1. The teacher sends a question to both students.
2. The students respond with their answers.
3. The teacher evaluates the responses and updates the scores.
4. The interaction repeats for all questions until the "END" condition is met.

This structured approach ensures real-time communication and evaluation between processes.

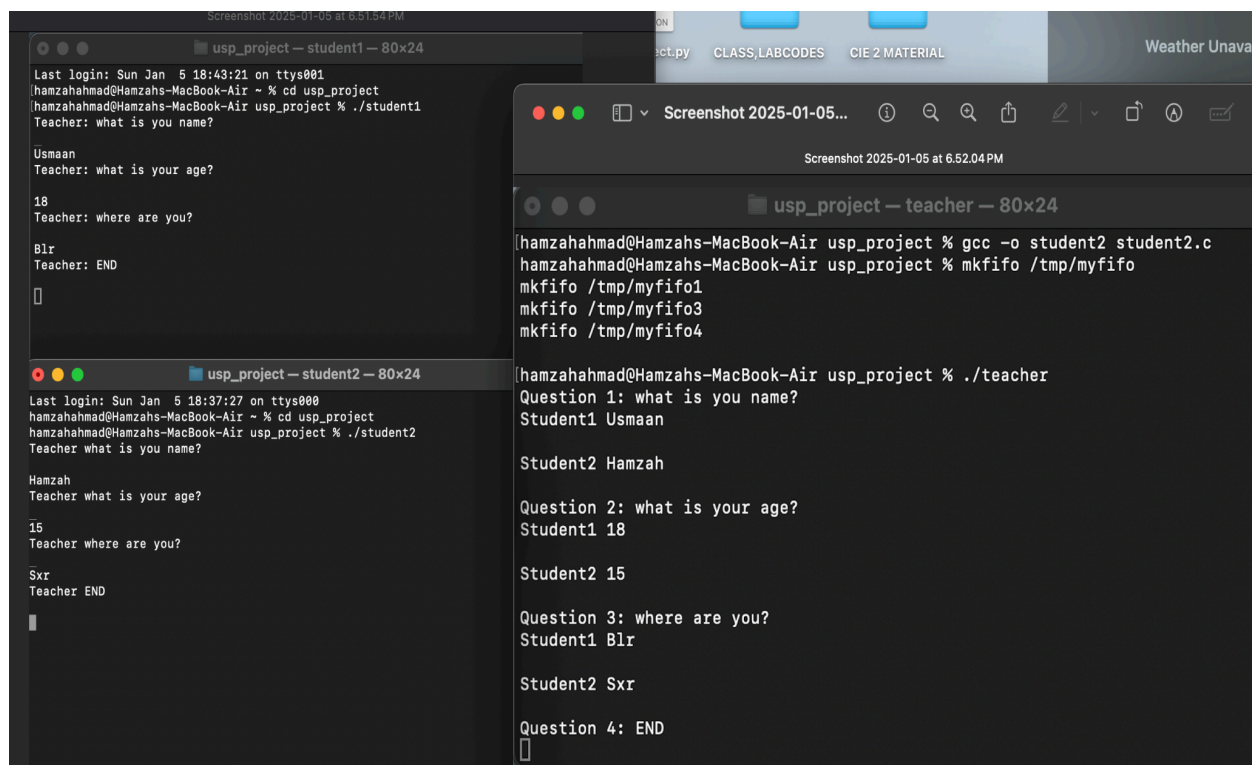
CHAPTER 5

RESULT

Observations:

1. Successfully transmitted questions to students via FIFOs.
2. Correct answers were evaluated, and scores were updated dynamically.
3. Sequential dependencies were noted in student responses.

Output



```
ScreenShot 2025-01-05 at 6:51:54 PM
usp_project — student1 — 80x24
Last login: Sun Jan 5 18:43:21 on ttys001
hamzahahmad@Hamzahs-MacBook-Air ~ % cd usp_project
hamzahahmad@Hamzahs-MacBook-Air usp_project % ./student1
Teacher: what is your name?
Usmaan
Teacher: what is your age?
18
Teacher: where are you?
Blr
Teacher: END
[]

ScreenShot 2025-01-05 at 6:52:04 PM
usp_project — student2 — 80x24
Last login: Sun Jan 5 18:37:27 on ttys000
hamzahahmad@Hamzahs-MacBook-Air ~ % cd usp_project
hamzahahmad@Hamzahs-MacBook-Air usp_project % ./student2
Teacher what is your name?
Hamzah
Teacher what is your age?
15
Teacher where are you?
Sxr
Teacher END
[]

ScreenShot 2025-01-05 at 6:52:04 PM
usp_project — teacher — 80x24
hamzahahmad@Hamzahs-MacBook-Air usp_project % gcc -o student2 student2.c
hamzahahmad@Hamzahs-MacBook-Air usp_project % mkfifo /tmp/myfifo
mkfifo /tmp/myfifo1
mkfifo /tmp/myfifo3
mkfifo /tmp/myfifo4
hamzahahmad@Hamzahs-MacBook-Air usp_project % ./teacher
Question 1: what is your name?
Student1 Usmaan
Student2 Hamzah
Question 2: what is your age?
Student1 18
Student2 15
Question 3: where are you?
Student1 Blr
Student2 Sxr
Question 4: END
[]
```

CHAPTER 6

CONCLUSION

The project effectively demonstrates the application of named pipes for inter-process communication in Unix-based systems. The teacher-student quiz system highlights the potential of IPC mechanisms for creating interactive and educational software solutions. While the implementation faces challenges such as sequential dependencies, it serves as a robust foundation for further development in adaptive learning systems and real-time training applications.

Future work may explore multithreading and enhanced concurrency to improve performance and responsiveness.

References

- [Named Pipe or FIFO with example C program - GeeksforGeeks](#)
- [FIFO \(First-In-First-Out\) approach in Programming - GeeksforGeeks](#)
- TextBook:Advanced programming in the UNIX environment
- [What are Named Pipes or FIFO in Linux/Unix systems? \(tutorialspoint.com\)](#)
- [Named pipes in C program \(unix\) - Stack Overflow](#)