

Name-Sugandh Mishra
Reg-20204211
Sec- cse c

Motilal Nehru National Institute of Technology Allahabad Prayagraj
Distributed System (CS17201) B.Tech (CSE) – VII Sem Lab 1

1. Write a program to create two processes. First process takes a string and passes it to second process through a pipe. The second process concatenates the received string with another string without using string function and sends it back to the first process for printing.

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

```
#define BUFFER_SIZE 100
```

```
int main() {
    int pipe_fd[2];
    char input_string[BUFFER_SIZE];
    char received_string[BUFFER_SIZE];

    if (pipe(pipe_fd) == -1) {
        perror("pipe");
        exit(EXIT_FAILURE);
    }

    pid_t pid = fork();

    if (pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }

    if (pid == 0) {
        close(pipe_fd[1]);
        ssize_t bytes_read = read(pipe_fd[0], received_string, BUFFER_SIZE);

        if (bytes_read == -1) {
```

```

        perror("read");
        exit(EXIT_FAILURE);
    }

    close(pipe_fd[0]);

    strcat(received_string, " Concatenated");
    printf("Concatenated string::%s\n", received_string);
    exit(EXIT_SUCCESS);
} else {
    close(pipe_fd[0]);
    printf("Enter a string: ");
    fgets(input_string, BUFFER_SIZE, stdin);
    input_string[strcspn(input_string, "\n")] = '\0';

    ssize_t bytes_written = write(pipe_fd[1], input_string, strlen(input_string));

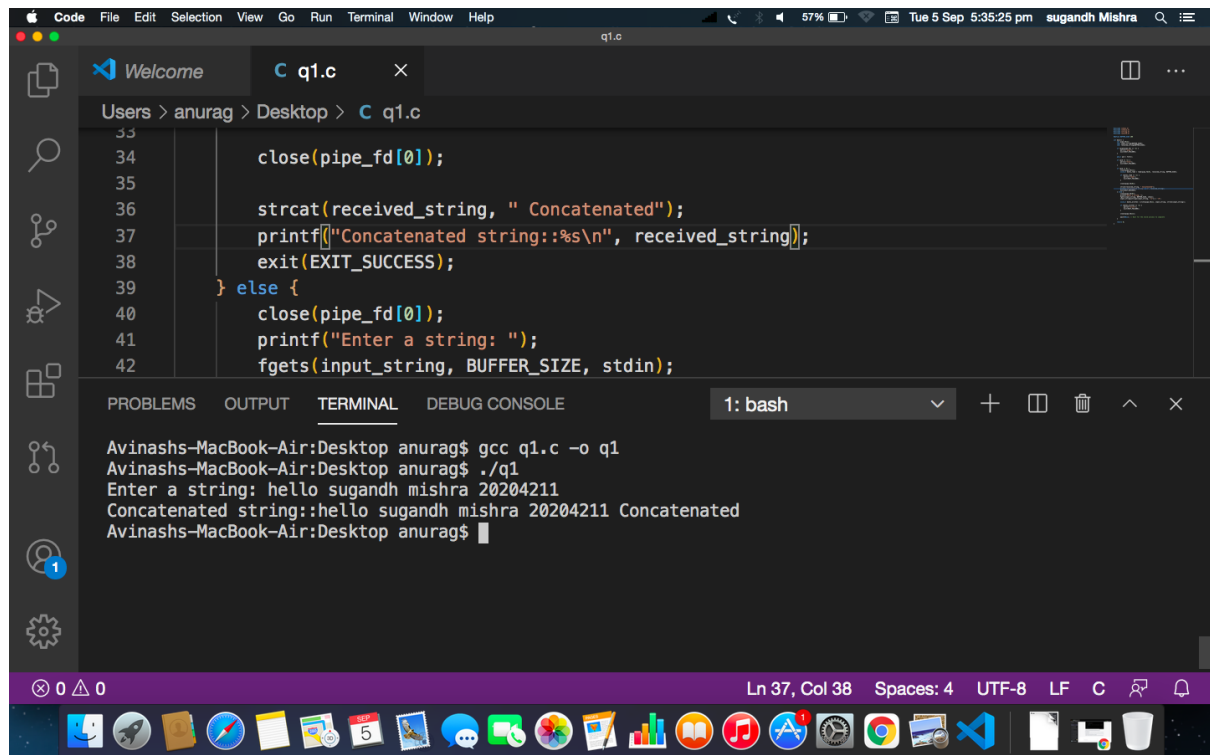
    if (bytes_written == -1) {
        perror("write");
        exit(EXIT_FAILURE);
    }

    close(pipe_fd[1]);

    wait(NULL); // Wait for the child process to complete
}

return 0;
}

```



The screenshot shows a macOS desktop with a Visual Studio Code editor open. The editor has a tab for 'q1.c' and shows the following C code:

```
33
34     close(pipe_fd[0]);
35
36     strcat(received_string, " Concatenated");
37     printf("Concatenated string::%s\n", received_string);
38     exit(EXIT_SUCCESS);
39 } else {
40     close(pipe_fd[0]);
41     printf("Enter a string: ");
42     fgets(input_string, BUFFER_SIZE, stdin);
```

Below the editor, the 'TERMINAL' tab is active, showing the following commands and output:

```
Avinashs-MacBook-Air:Desktop anurag$ gcc q1.c -o q1
Avinashs-MacBook-Air:Desktop anurag$ ./q1
Enter a string: hello sugandh mishra 20204211
Concatenated string::hello sugandh mishra 20204211 Concatenated
Avinashs-MacBook-Air:Desktop anurag$
```

The status bar at the bottom of the editor shows 'Ln 37, Col 38', 'Spaces: 4', 'UTF-8', 'LF', and 'C'.

2. Develop a program in which the parent process sends two matrices to its child process through a pipe and the child process returns the sum of the matrices to the parent through a pipe. The parent should print the result.

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

```
#define ROWS 3
#define COLS 3
```

```
void sendMatrix(int pipe_fd[2], int matrix[ROWS][COLS]) {
    close(pipe_fd[0]);
    write(pipe_fd[1], matrix, sizeof(int) * ROWS * COLS);
    close(pipe_fd[1]);
}
```

```
void receiveMatrix(int pipe_fd[2], int result[ROWS][COLS]) {
    close(pipe_fd[1]);
    read(pipe_fd[0], result, sizeof(int) * ROWS * COLS);
    close(pipe_fd[0]);
}
```

```
int main() {
```

```

int matrix1[ROWS][COLS] = {{3, 2, 3}, {4, 1, 6}, {2, 8, 3}};
int matrix2[ROWS][COLS] = {{4, 4, 7}, {6, 5, 4}, {3, 2, 1}};
int result[ROWS][COLS] = {0};

int pipe_parent_to_child[2];
int pipe_child_to_parent[2];

if (pipe(pipe_parent_to_child) == -1 || pipe(pipe_child_to_parent) == -1) {
    perror("pipe");
    exit(EXIT_FAILURE);
}

pid_t pid = fork();

if (pid == -1) {
    perror("fork");
    exit(EXIT_FAILURE);
}

if (pid == 0) {
    int child_result[ROWS][COLS] = {0};

    receiveMatrix(pipe_parent_to_child, matrix1);
    receiveMatrix(pipe_parent_to_child, matrix2);

    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            child_result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }

    sendMatrix(pipe_child_to_parent, child_result);
    exit(EXIT_SUCCESS);
} else {
    sendMatrix(pipe_parent_to_child, matrix1);
    sendMatrix(pipe_parent_to_child, matrix2);

    receiveMatrix(pipe_child_to_parent, result);

    printf("Sum of Matrices:\n");
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            printf("%d\t", result[i][j]);
        }
    }
}

```

```

        printf("\n");
    }
}

return 0;
}

```

The screenshot shows a macOS desktop with Visual Studio Code open. The editor displays a C program named `q2.c` which defines two 3x3 matrices, `matrix1` and `matrix2`, and a `result` array. The program is compiled and executed in the terminal. The output shows the sum of the two matrices, with the first two rows being identical (7, 6, 10 and 10, 10, 10) and the third row being (5, 10, 4).

```

Users > anurag > Desktop > C q2.c
19
20 int main() {
21     int matrix1[ROWS][COLS] = {{3, 2, 3}, {4, 1, 6}, {2, 8, 3}};
22     int matrix2[ROWS][COLS] = {{4, 4, 7}, {6, 5, 4}, {3, 2, 1}};
23     int result[ROWS][COLS] = {0};
24
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 1: bash
7      6      10
10     10     10
10     10     10
Avinashs-MacBook-Air:Desktop anurag$ ./q2
Sum of Matrices:
7      6      10
10     10     10
10     10     10
Avinashs-MacBook-Air:Desktop anurag$ gcc q2.c -o q2
Avinashs-MacBook-Air:Desktop anurag$ ./q2
Sum of Matrices:
7      6      10
10     6      10
5      10     4
Avinashs-MacBook-Air:Desktop anurag$

```