

# OPERATING SYSTEMS PRACTICE

## (ASSIGNMENT 5)

**Name** – Mridul Harish

**Roll number** – CED18I034

**Question 1** : Parent sets up a string which is read by child, reversed there and read back the parent.

**Code :-**

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

#define read_end 0
#define write_end 1

int main(int argc, char const *argv[])
{
    int pipefd1[2]; int pipefd2[2];
    int status1; int status2;

    char string1[12] = "MridulHarish";
    char string2[12]; char string3[12];

    status1 = pipe(pipefd1);
    if(status1 == -1)
    {
        printf("Pipe failed\n");
        return 0;
    }

    status2 = pipe(pipefd2);
    if(status2 == -1)
    {
        printf("Pipe failed\n");
        return 0;
    }
```

```

pid_t pid = fork();

if(pid > 0)
{
    //parent pipe
    close(pipefd1[read_end]);
    close(pipefd2[write_end]);

    printf("The original string is %s\n",string1);
    write(pipefd1[write_end], string1, sizeof(string1)+1);

    read(pipefd2[read_end], string2, sizeof(string2));
    printf("The reversed string is %s\n", string2);
}
else
{
    //child pipe
    close(pipefd1[write_end]);
    close(pipefd2[read_end]);

    read(pipefd1[read_end], string2, sizeof(string2));

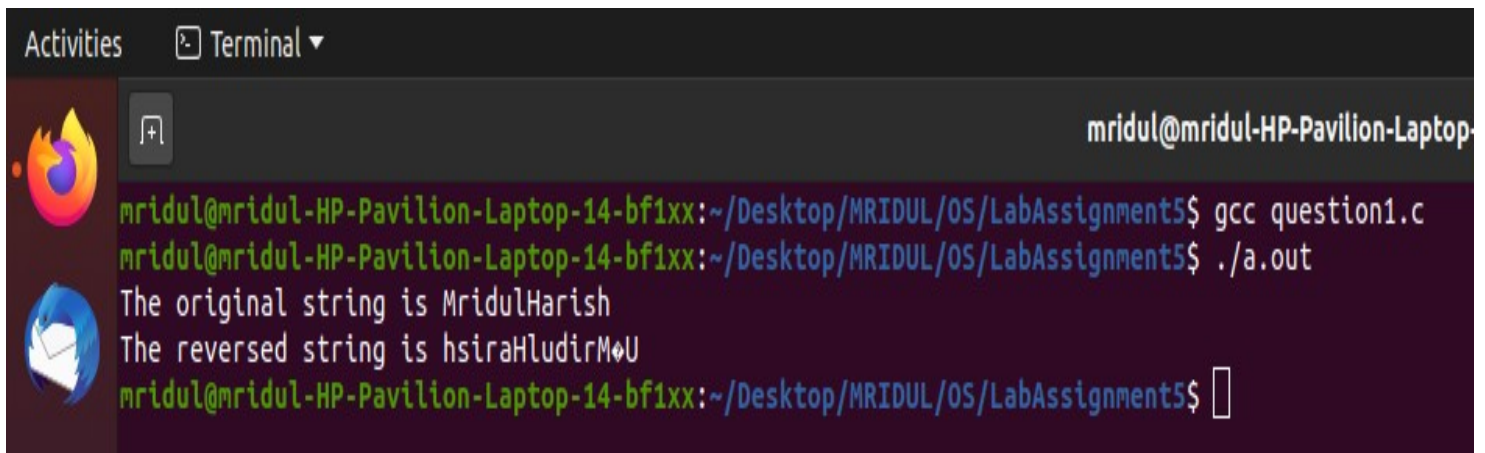
    int j = -1; int i = (sizeof(string2)/sizeof(string2[0])) - 1;

    for(; i >= 0; i = i-1)
    {
        j = j+1;
        string3[j] = string2[i];
    }

    write(pipefd2[write_end], string3, sizeof(string3)+1);
}
return 0;
}

```

## Output :-



```
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS/LabAssignment5$ gcc question1.c
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS/LabAssignment5$ ./a.out
The original string is MridulHarish
The reversed string is hsirahLudirMou
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS/LabAssignment5$
```

## Explanation :-

Here we use the basic pipeline setup of 2 pipes i.e. a parent and a child pipe. We declare three strings and a string is sent from the parent to the child through the pipe. At the child end's of the pipe, the string is read and then we reverse the strings using for loops and stores the string. Then this string is sent from the child to the parent's end of the pipe and displayed.

**Question 2 :** Parent sets up string 1 and child sets up string 2. String 2 concatenated to string 1 at parent end and then read back at the child end.

## Code :-

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

#define read_end 0
#define write_end 1

int main(int argc, char const *argv[])
{
    int pipefd1[2]; int pipefd2[2];
    int status1; int status2;

    char string1[5] = "Hello";
```

```
char string2[5] = "There";  
char string3[10];
```

```
status1 = pipe(pipefd1);  
if(status1 == -1)  
{  
    printf("Pipe failed\n");  
    return 0;  
}
```

```
status2 = pipe(pipefd2);  
if(status2 == -1)  
{  
    printf("Pipe failed\n");  
    return 0;  
}
```

```
pid_t pid = fork();  
if(pid == 0)  
{  
    //child pipe  
    close(pipefd1[read_end]);  
    close(pipefd2[write_end]);  
    write(pipefd1[write_end], string2, sizeof(string2) + 1);  
  
    read(pipefd2[read_end], string3, sizeof(string3));  
    printf("The concatenated string is %s\n", string3);  
}  
else  
{  
    //parent pipe  
    close(pipefd1[write_end]);  
    close(pipefd2[read_end]);  
    read(pipefd1[read_end], string2, sizeof(string2));  
  
    //concatenation  
    int k = 0;  
    for(int i = 0; i < 5; i = i+1)  
    {
```

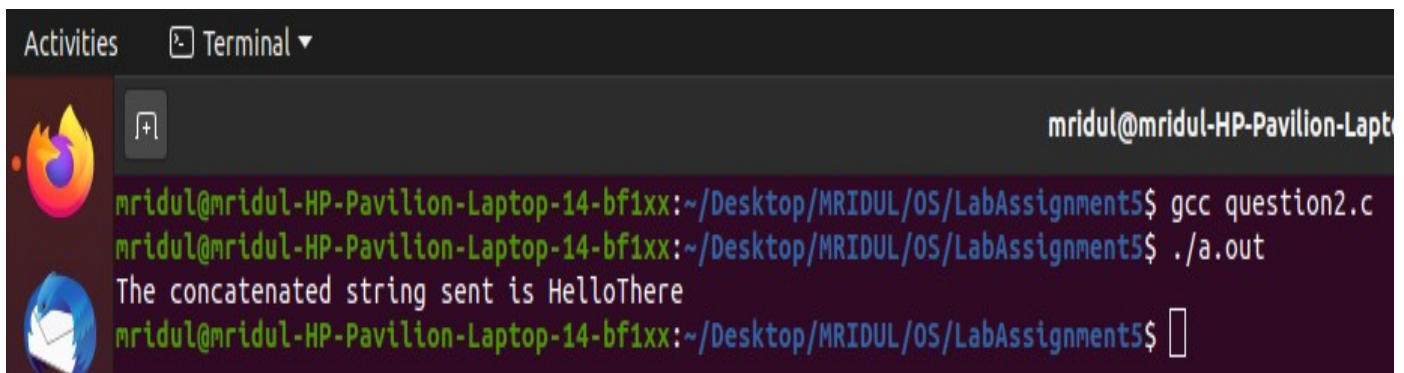
```

        string3[i] = string1[i];
    }

    for(int j = 5; j < 10; j = j+1)
    {
        string3[j] = string2[k];
        k = k+1;
    }
    printf("full is = %s\n", string3);
    write(pipefd2[write_end], string3, sizeof(string3)+1);
}
return 0;
}

```

### **Output :-**



```

Activities  Terminal ▾
mridul@mridul-HP-Pavilion-Lapt
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS/LabAssignment5$ gcc question2.c
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS/LabAssignment5$ ./a.out
The concatenated string sent is HelloThere
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS/LabAssignment5$ 

```

### **Explanation :-**

Here we use the similar basic pipeline setup of 2 pipes i.e. parent pipe and child pipe. The second string from the child is sent to the parent through the pipe and the first string is already at the parent pipe. The concatenation process is done using 2 for loops and the new string is obtained and displayed. parent and child processes are distinguished using `pid==0` for child process.

**Question 3** : Substring generation at child end of a string setup at parent process end.

### **Code :-**

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

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

#define read_end 0
#define write_end 1

int main(int argc, char const *argv[])
{
    int pipefd1[2]; int pipefd2[2];
    int status1; int status2;

    char string1[12] = "MridulHarish";
    char string2[12];
    char string3[12];
    memset(string3, 0, sizeof(string3));

    status1 = pipe(pipefd1);
    if(status1 == -1)
    {
        printf("Pipe failed\n");
        return 0;
    }

    status2 = pipe(pipefd2);
    if(status2 == -1)
    {
        printf("Pipe failed\n");
        return 0;
    }

    pid_t pid = fork();
    if(pid > 0)
    {
        //parent pipe
        close(pipefd1[read_end]);
        close(pipefd2[write_end]);

        printf("The original string is %s\n", string1);
    }
}
```

```

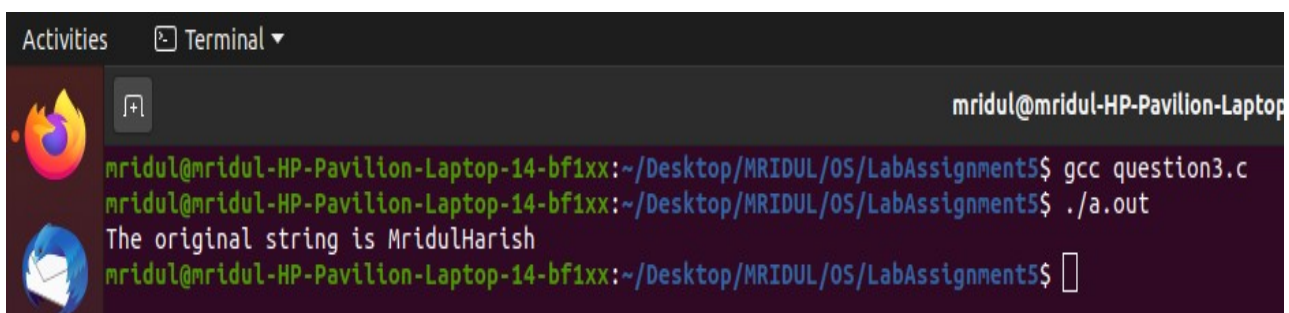
        write(pipefd1[write_end], string1, sizeof(string1)+1);

        read(pipefd2[read_end], string2, sizeof(string2));
        printf("The substring by the child is %s\n", string2);
    }
    else
    {
        //child pipe
        close(pipefd1[write_end]);
        close(pipefd2[read_end]);

        read(pipefd1[read_end], string2, sizeof(string2));
        for(int i = 0; i < 3; i = i+1)
        {
            string3[i] = string2[3+i];
        }
        write(pipefd2[write_end], string3, sizeof(string3)+1);
    }
    return 0;
}

```

### **Output :-**



```

mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS/LabAssignment5$ gcc question3.c
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS/LabAssignment5$ ./a.out
The original string is MridulHarish
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS/LabAssignment5$

```

### **Explanation :-**

Again with the basic pipe setup we have generated an arbitrary substring from the original string given. The string is sent from the parent to the child through the pipe where the substring generation takes place.

**Question 4 :** String reversal and palindrome check using pipes / shared memory.

## **Code :-**

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

#define read_end 0
#define write_end 1

int main(int argc, char const *argv[])
{
    int pipefd1[2]; int pipefd2[2];
    int status1; int status2;

    char string1[24] = "mridulharishhsirahludirm";
    char string2[24];
    memset(string2, 0, sizeof(string2));
    char string3[24];
    memset(string3, 0, sizeof(string3));

    status1 == pipe(pipefd1);
    if(status1 == -1)
    {
        printf("Pipe failed\n");
        return 0;
    }

    status2 = pipe(pipefd2);
    if(status2 == -1)
    {
        printf("Pipe failed\n");
        return 0;
    }

    pid_t pid = fork();
    if(pid > 0)
    {
        //parent pipe
```



```

close(pipefd1[read_end]);
close(pipefd2[write_end]);

printf("The original string is %s\n", string1);
write(pipefd1[write_end], string1, sizeof(string1)+1);

read(pipefd2[read_end], string2, sizeof(string2));
printf("The reversed string is %s\n", string2);

int j = 0; int check = 1;
for(int i = 0; i <= 9; i = i+1)
{
    if(string2[i] != string1[i])
    {
        check = 0;
        break;
    }
    j = j+1;
}
if(check == 0)
{
    printf("It is not a palindrome\n");
}
else
{
    printf("It is a palindrome\n");
}
}
else
{
    //child pipe
    close(pipefd1[write_end]);
    close(pipefd2[read_end]);
    read(pipefd1[read_end], string2, sizeof(string2));

    int j = -1; int i = sizeof(string2)/sizeof(string2[0]) - 1;
    for(; i >= 0; i = i-1)
    {
        j = j+1;
    }
}

```

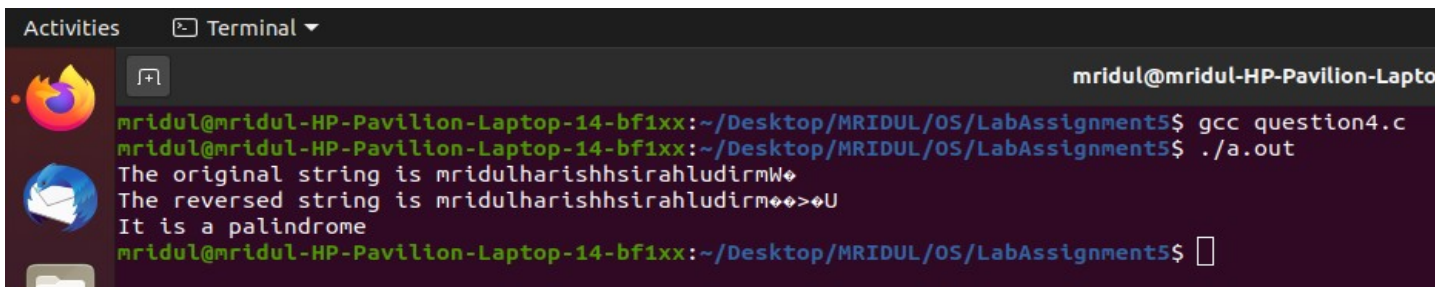
```

        string3[j] = string2[i];
    }

    write(pipefd2[write_end], string3, sizeof(string3)+1);
}
return 0;
}

```

### **Output :-**



```

mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS/LabAssignment5$ gcc question4.c
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS/LabAssignment5$ ./a.out
The original string is mridulharishhsirahludirmW
The reversed string is mridulharishhsirahludirmW
It is a palindrome
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS/LabAssignment5$ 

```

### **Explanation :-**

Similar pipe setup is used. The reversal of string from the first question is used here. As parent sends the string to the child through the pipe, the string is reversed at the child's end, sent back to the parent, checked with the original string and then output is displayed.