

OPERATING SYSTEMS ASSIGNMENT - 5

Question 1 :

Parent sets up a string which is read by child, reversed there and read back the parent

Logic :

Create two file descriptors one for the parent and one for the child. The original string is written onto the parent file descriptor for which the write end is open. The child reads the string from the same pipe . It calculates the length of the string and runs a reverse loop to create a new string containing the reversed string. This new string is written over the second pipe for which the child has the writing permission and is read by the parent and displayed.

Code :

```
/*
 * @Author: prateek
 * @Date: 2020-10-21 00:39:25
 * @Last Modified by: prateek
 * @Last Modified time: 2020-10-21 01:33:12
 */

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

#define READ_END 0
#define WRITE_END 1

int main(int argc, char* argv[])
{
    char write_message[10] = "Helloworld";
    char read_message[10];
    char reversed_message[10];

    int fd1[2], fd2[2];
```

```
pid_t pid;
if(pipe(fd1) == -1)
{
    fprintf(stderr, "Some error occurred\n");
    return 1;
}
if(pipe(fd2)==-1)
{
    fprintf(stderr, "Some error occurred\n");
    return 2;
}

pid = fork();
if(pid>0)
{

    close(fd1[READ_END]);
    close(fd2[WRITE_END]);
    write(fd1[WRITE_END],write_message,sizeof(write_message) +1);

    read(fd2[READ_END],read_message,sizeof(read_message));
    printf("Parent-->Value received by the child :
%s\n",read_message);

    close(fd1[WRITE_END]);
    close(fd2[READ_END]);

}

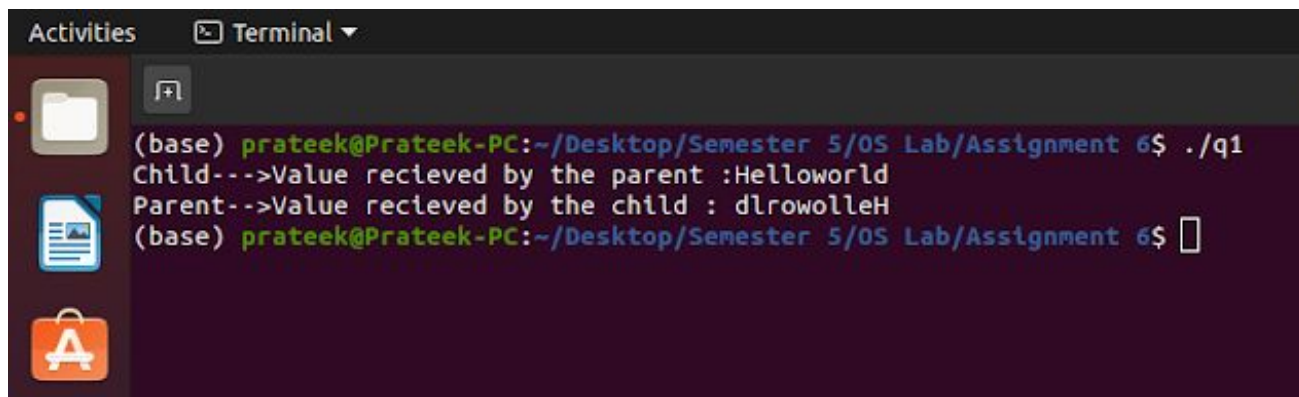
else if(pid==0)
{

    close(fd1[WRITE_END]);
    close(fd2[READ_END]);

    read(fd1[READ_END],read_message,sizeof(read_message));
    printf("Child--->Value received by the parent
:%s\n",read_message);
    int j=0;
    int i=sizeof(read_message)/sizeof(read_message[0]) -1;
    for(;i>=0;i--)
    {
        reversed_message[j]=read_message[i];
```

```
        j++;  
    }  
  
    write(fd2[WRITE_END], reversed_message, sizeof(reversed_message)+1);  
  
    close(fd1[READ_END]);  
    close(fd2[WRITE_END]);  
  
    }  
    else  
    {  
        fprintf(stderr, "Forking failure\n");  
    }  
  
}
```

Output :

A screenshot of a Linux terminal window. The title bar shows 'Activities' and 'Terminal'. On the left is a sidebar with icons for a file manager, a terminal window, and an application store. The terminal text shows a user running a program in a directory path. The output shows two lines of data being exchanged between a parent and child process.

```
(base) prateek@Prateek-PC:~/Desktop/Semester 5/OS Lab/Assignment 6$ ./q1  
Child-->Value recieved by the parent :Helloworld  
Parent-->Value recieved by the child : dlrowolleH  
(base) prateek@Prateek-PC:~/Desktop/Semester 5/OS Lab/Assignment 6$
```

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.

Logic :

Create two file descriptors one for the parent and one for the child. The original string is written onto the parent file descriptor for which the write end is open. The child reads the string from the same pipe and writes it over the pipe for the parent. The parent reads this new string from the child and creates a new string concatenating both the string one setup by the parent and the other one setup by the child. This new concatenated string is then written over the pipe again and is read by the child and displayed.

For the sake of simplicity the first string that the parent sets up is : **Hello** and the string setup by the child is : **World**

Code :

```
/*
 * @Author: prateek
 * @Date: 2020-10-21 01:22:44
 * @Last Modified by: prateek
 * @Last Modified time: 2020-10-21 01:54:20
 */

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

#define READ_END 0
#define WRITE_END 1

int main(int argc, char* argv[])
{
    char str1[6] = "Hello";
    char str3[10];
    char str4[5];
    int r, w;
```

```
int fd1[2], fd2[2];
pid_t pid;
if (pipe(fd1) == -1)
{
    fprintf(stderr, "Some error occurred\n");
    return 1;
}
if (pipe(fd2) == -1)
{
    fprintf(stderr, "Some error occurred\n");
    return 2;
}
pid = fork();
if (pid > 0)
{
    close(fd1[READ_END]);
    close(fd2[WRITE_END]);
    r = read(fd2[READ_END], str4, sizeof(str4));
    int k = 0;
    for (int i = 0; i < 5; i++)
    {
        str3[i] = str1[i];
    }
    for (int j = 5; j < 10; j++)
    {
        str3[j] = str4[k];
        k++;
    }

    w = write(fd1[WRITE_END], str3, sizeof(str3)+1);

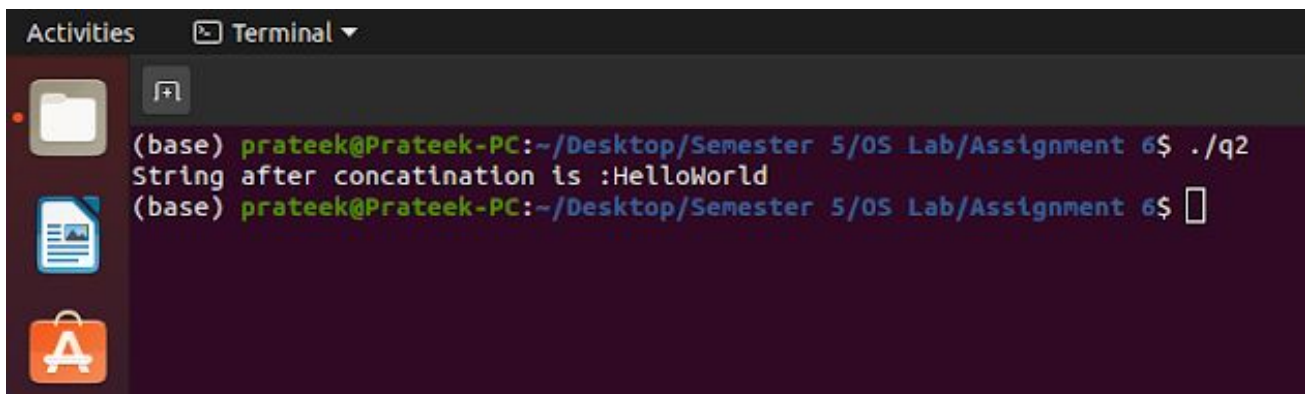
    close(fd1[WRITE_END]);
    close(fd2[READ_END]);
}

else if (pid == 0)
{
    close(fd1[WRITE_END]);
    close(fd2[READ_END]);
    char str2[5] = "World";
    w = write(fd2[WRITE_END], str2, sizeof(str2) + 1);
    r = read(fd1[READ_END], str3, sizeof(str3));
}
```

```
        printf("String after concatenation is :%s\n", str3);

        close(fd1[READ_END]);
        close(fd2[WRITE_END]);
    }
    else
    {
        fprintf(stderr, "Forking failure\n");
    }
}
```

Output :

A screenshot of a Linux terminal window. The title bar shows 'Activities' and 'Terminal'. On the left is a sidebar with icons for a file manager, a terminal window, and an application store. The terminal text shows a user running a command in a directory: `(base) prateek@Prateek-PC:~/Desktop/Semester 5/OS Lab/Assignment 6$./q2`. The output of the program is `String after concatination is :HelloWorld`. The prompt then changes to `(base) prateek@Prateek-PC:~/Desktop/Semester 5/OS Lab/Assignment 6$` with a cursor.

```
(base) prateek@Prateek-PC:~/Desktop/Semester 5/OS Lab/Assignment 6$ ./q2
String after concatination is :HelloWorld
(base) prateek@Prateek-PC:~/Desktop/Semester 5/OS Lab/Assignment 6$
```

The original strings were **‘Hello’** and **‘World’**.

Question 3 :

Substring generation at child end of a string setup at parent process end.

Logic :

Create two file descriptors one for the parent and one for the child. The parent sets up a string and also asks the user to input two indexes. The first one being the start index and the other one the ending index. As in most of the programming languages the substring function is exclusive and does not include the ending index the same has been implemented. The child reads the string setup by the parent and creates a new string on the basis of the start and end index. This new string is then written over a pipe from which the parent reads and displays the substring.

For the sake of simplicity the first string that the parent sets up is : **HelloWorld** and we assume the user inputs correct values of start and end index.

Code :

```
/*
 * @Author: prateek
 * @Date: 2020-10-21 22:23:01
 * @Last Modified by: prateek
 * @Last Modified time: 2020-10-21 23:01:29
 */

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

#define READ_END 0
#define WRITE_END 1

int main(int argc, char* argv[])
{
    char str1[10] = "HelloWorld";
    int n=0,m=0;
    printf("The string is : %s . Please enter the starting and ending
```

```
index accordingly.\n",str1);
printf("Enter starting index :");
scanf("%d",&n);
printf("Enter ending index :");
scanf("%d",&m);
char str2[m-n];
memset(str2,0,sizeof(str2));

int r, w;

int fd1[2], fd2[2];
pid_t pid;
if (pipe(fd1) == -1)
{
    fprintf(stderr, "Some error occurred\n");
    return 1;
}
if (pipe(fd2) == -1)
{
    fprintf(stderr, "Some error occurred\n");
    return 2;
}
pid = fork();
if (pid > 0)
{
    close(fd1[READ_END]);
    close(fd2[WRITE_END]);
    w = write(fd1[WRITE_END], str1, sizeof(str1)+1);
    r = read(fd2[READ_END],str2,sizeof(str2));
    printf("The substring is %s\n", str2);

    close(fd1[WRITE_END]);
    close(fd2[READ_END]);

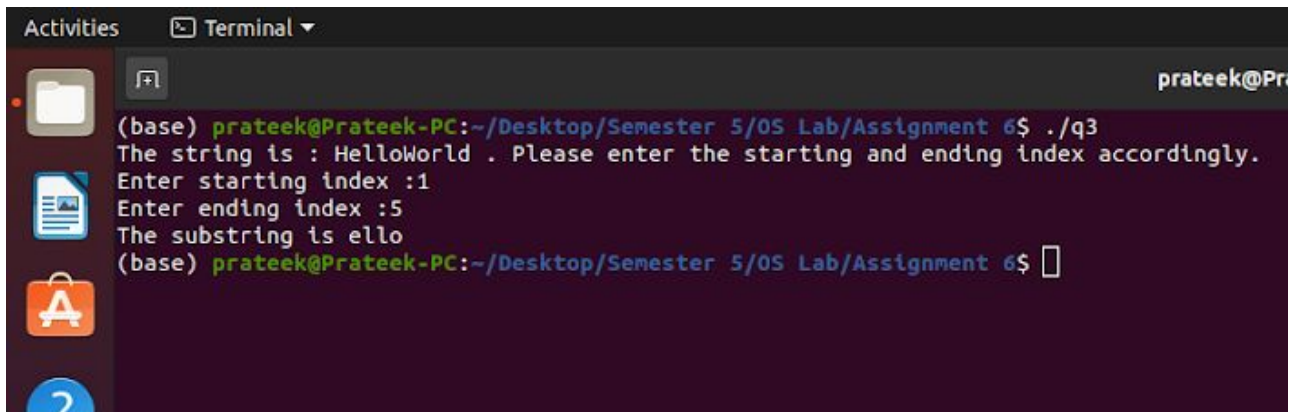
}

else if (pid == 0)
{
    close(fd1[WRITE_END]);
    close(fd2[READ_END]);
    r = read(fd1[READ_END], str1, sizeof(str1));
    for(int i=0;i<m-n;i++)
```



```
        {
            str2[i] = str1[n+i];
        }
        w = write(fd2[WRITE_END], str2, sizeof(str2) + 1);
        close(fd1[READ_END]);
        close(fd2[WRITE_END]);
    }
    else
    {
        fprintf(stderr, "Forking failure\n");
    }
}
```

Output :

A screenshot of a Linux terminal window. The title bar shows 'Activities' and 'Terminal'. The terminal content shows a user running a program './q3' in a directory '~/Desktop/Semester 5/OS Lab/Assignment'. The program prompts for a starting and ending index. The user enters '1' for the starting index and '5' for the ending index. The program outputs 'The substring is ello'. The prompt '(base) prateek@Prateek-PC:~/Desktop/Semester 5/OS Lab/Assignment 6\$' is visible at the bottom.

```
(base) prateek@Prateek-PC:~/Desktop/Semester 5/OS Lab/Assignment 6$ ./q3
The string is : HelloWorld . Please enter the starting and ending index accordingly.
Enter starting index :1
Enter ending index :5
The substring is ello
(base) prateek@Prateek-PC:~/Desktop/Semester 5/OS Lab/Assignment 6$
```

Question 4 :

String reversal and palindrome check using pipes / shared memory.

Logic :

Create two file descriptors one for the parent and one for the child. The parent sets up a string and writes it over the pipe from which the child can read. The child reads this original string and then creates a new string using a reverse for loop to create the reverse of the original string. This reversed new string is then written onto the pipe from which the parent can read. The parent reads this reversed string and compares it with the original string. If they are the same then it outputs palindrome else it does not.

Code :

```
/*
 * @Author: prateek
 * @Date: 2020-10-21 23:02:23
 * @Last Modified by: prateek
 * @Last Modified time: 2020-10-21 23:27:14
 */

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

#define READ_END 0
#define WRITE_END 1

int main(int argc, char* argv[])
{
    char str1[7] = "ABCDcba";
    char str2[7];
    char str3[7];

    int r, w;

    int fd1[2], fd2[2];
    pid_t pid;
```

```
if (pipe(fd1) == -1)
{
    fprintf(stderr, "Some error occurred\n");
    return 1;
}
if (pipe(fd2) == -1)
{
    fprintf(stderr, "Some error occurred\n");
    return 2;
}
pid = fork();
if (pid > 0)
{
    close(fd1[READ_END]);
    close(fd2[WRITE_END]);
    w = write(fd1[WRITE_END], str1, sizeof(str1)+1);
    r = read(fd2[READ_END], str2, sizeof(str2));
    printf("The reversed string by the child is '%s'\n", str2);

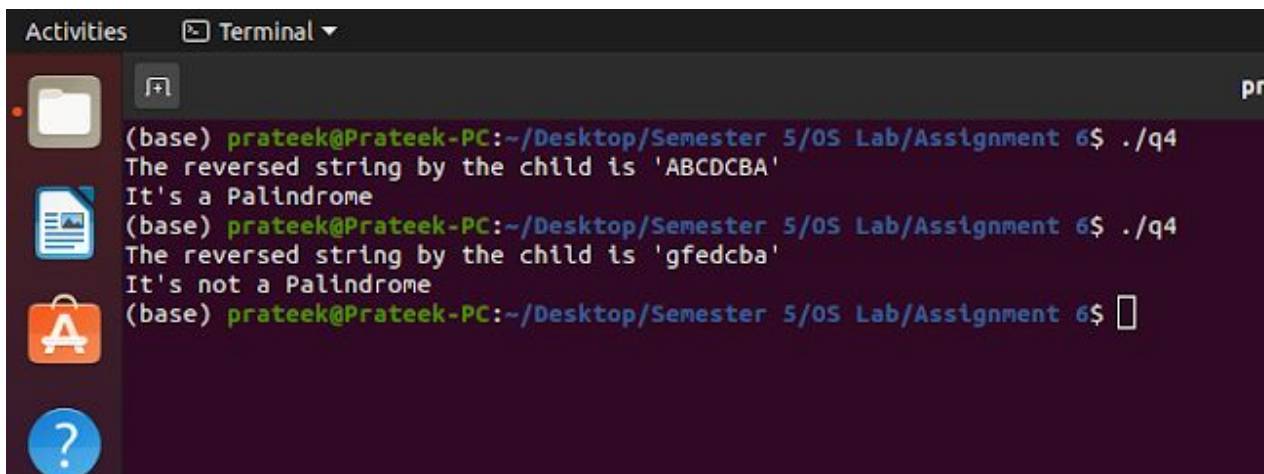
    int flag = 1;
    //printf("%ld    %s    %s", strlen(str1), str1, str2);
    for (int i = 0; i < strlen(str2); i++)
    {
        if (str2[i] != str1[i])
        {
            flag = 0;
            break;
        }
    }

    if (flag == 0)
    {
        printf("It's not a Palindrome\n");
    }
    else
    {
        printf("It's a Palindrome\n");
    }

    close(fd1[WRITE_END]);
    close(fd2[READ_END]);
}
```

```
else if (pid == 0)
{
    close(fd1[WRITE_END]);
    close(fd2[READ_END]);
    r = read(fd1[READ_END], str2, sizeof(str2));
    int j = 0;
    int i = sizeof(str2) / sizeof(str2[0]) - 1;
    for (; i >= 0; i--)
    {
        str3[j] = str2[i];
        j++;
    }
    write(fd2[WRITE_END], str3, sizeof(str3) + 1);
}
else
{
    fprintf(stderr, "Forking failure\n");
}
}
```

Output :



```
Activities  Terminal ▾
pr
(base) prateek@Prateek-PC:~/Desktop/Semester 5/OS Lab/Assignment 6$ ./q4
The reversed string by the child is 'ABCD CBA'
It's a Palindrome
(base) prateek@Prateek-PC:~/Desktop/Semester 5/OS Lab/Assignment 6$ ./q4
The reversed string by the child is 'gfedcba'
It's not a Palindrome
(base) prateek@Prateek-PC:~/Desktop/Semester 5/OS Lab/Assignment 6$
```