CSE 316

Operating System Assignment

Problem no: 25

Name: Rittike Ghosh

Student ID: 11810568

Section: K18SJ

Email: rittike.11810568@lpu.in

GitHub link: https://github.com/RittikeGhosh/IPC_with_c_problem_25

Problem Statement Q25:

Design a program using concepts of inter-process communication ordinary pipes in which one process sends a string message to a second process, and the second process reverses the case of each character in the message and sends it back to the first process.

For example,

If the first process sends the message "Hi There"

The second process will return "hI tHERE".

This will require using two pipes, one for sending the original message from the first to the second process and the other for sending the modified message from the second to the first process. You can write this program using either UNIX or Windows pipes.

Problem Analysis:

According to the given problem statement, we need to implement the concept of Inter-Process Communication (IPC) with pipes. At first we have take a string as a input from the user in the first process, then we need to pass this input value to the next process through pipe, where the second process will swap the letter case of the whole sting, and then it will return the modified string back to the first process, and then then finally the first process will print the modified string as the output.

Algorithm:

1. Create 2 pipes
2. String1 = read (“Enter the string”)
3. Create child process with fork ()
4. Pass the string to the child process and wait for the child process to finish
5. Inside child process swapCase () and return it to parent process with other pipe
6. String2 = receive modified string from the child process
7. Display the modified
8. exit ()

Complexity:

Let the number of characters in the string be *n*. Therefore, there has to be n comparisons.

So, the *Complexity:* ***O(n)***

Code:

Main Function

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <string.h>
#define MAX_LENGTH 1000

void caseSwap(char* str);

int main()
{
    system("clear");
    printf("##########################################################\n");
    printf("#                                                        #\n");
    printf("#                        WELCOME                         #\n");
    printf("#      This Program illustrates the working of IPC       #\n");
    printf("#        The program will transform letter cases         #\n");
    printf("#            e.g Hello World! --> hELLO wORLD!           #\n");
    printf("#                                                        #\n");
    printf("##########################################################\n\n\n");

    char p1_buffer[MAX_LENGTH];  // for input and pass to the second program
    char p2_buffer[MAX_LENGTH];  // for reading from first process and process text
    char outputBuffer[MAX_LENGTH];  // for reading from second process and output string

    int p1_pipeFd[2], p2_pipeFd[2];  // fd for the pipe : [0:read, 1:write]

    int err_code1 = pipe(p1_pipeFd);     // creating first/parent prog pipe
    int err_code2 = pipe(p2_pipeFd);  // creatint second/child prog pipe

    // error check
    if (err_code2 == -1 || err_code1 == -1)
    {
        printf("Error Occured ! process aborted ...");
        return 0;
    }


    printf("=> Executing First program[pid:%d]  \n", getpid());
    // string input
    printf("Enter the String[MAX_LENGTH = 1000] : ");
    scanf("%[^\n]%*c", p1_buffer);

    int fork_id = fork();
    // process p1
    if(fork_id > 0)
    {
        // delete unwanted pipe channels
        close(p1_pipeFd[0]);
        close(p2_pipeFd[1]);

        printf("*(Passing string to the next process[pid:%d])\n", fork_id);
        // write the string to the pipe p1_pipeFd[1]
        write(p1_pipeFd[1], p1_buffer, sizeof(p1_buffer));
        printf("-------------------------------------------------\n\n");
        wait(NULL);  // wait for the child process to finish

```

```
59          sleep(1);
60          printf("=> Back to first process[pid:%d]\n", getpid());
61          printf("[Reading the data in the read channel of process 2 pipe]\n");
62          // read from the p2_pipeFd[0] which is comming from second process
63          int size = read(p2_pipeFd[0], outputBuffer, sizeof(outputBuffer));
64          printf("String returned by the second process[pid:%d]: %s\n", fork_id, outputBuffer);
65          printf("-----------------------------------------------------\n");
66
67          // final output...
68          sleep(1);
69          printf("\nFinally, (...)\n=>\t| %s ---> %s |\n\n", p1_buffer, outputBuffer);
70      }
71      else
72      {
73          sleep(1);
74          // closing unwanted pipe channels
75          close(p1_pipeFd[1]);
76          close(p2_buffer[0]);
77
78          printf("=> Executing second process[%d]\n", getpid());
79          printf("[Reading the data in the read channel of process 1 pipe]\n");
80          // reading from the p1_pipeFd[0]
81          read(p1_pipeFd[0], p2_buffer, sizeof(p2_buffer));
82          printf("String recieved from the first process [%d] : %s \n", getppid(), p2_buffer);
83
84          //case transform
85          caseSwap(p2_buffer);
86
87          printf("Processed String is : %s\n", p2_buffer);
88          printf("*(Passing the processed data back to first process[%d])\n", getppid());
89          write(p2_pipeFd[1], p2_buffer, sizeof(p2_buffer));
90          printf("-----------------------------------------------------\n");
91      }
92
93      printf("(process id: %d finished)\n\n", getpid());
94      return 0;
95  }
96
```

Swap Case Function

```
97  void caseSwap(char* str)
98  {
99      for(int i = 0; str[i] != '\0'; i++)
100     {
101         if(str[i] >= 'A' && str[i] <= 'Z')
102             str[i] += 32;
103         else if(str[i] >= 'a' && str[i] <= 'z')
104             str[i] -= 32;
105     }
106 }
107
```

Output:

```
############################################################
#                                                          #
#                         WELCOME                          #
#      This Program illustrates the working of IPC         #
#        The program will transform letter cases           #
#            e.g Hello World! --> hELLO wORLD!             #
#                                                          #
############################################################


=> Executing First program[pid:64]
Enter the String[MAX_LENGTH = 1000] : Coding is AwESoME :-)
*(Passing string to the next process[pid:67])
-----------------------------------------------------

=> Executing second process[67]
[Reading the data in the read channel of process 1 pipe]
String recieved from the first process [64] : Coding is AwESoME :-)
Processed String is : cODING IS aWesOme :-)
*(Passing the processed data back to first process[64])
-----------------------------------------------------
(process id: 67 finished)

=> Back to first process[pid:64]
[Reading the data in the read channel of process 2 pipe]
String returned by the second process[pid:67]: cODING IS aWesOme :-)
-----------------------------------------------------

Finally, (...)
=>      | Coding is AwESoME :-) ---> cODING IS aWesOme :-) |

(process id: 64 finished)
```