OS - Practical – 6 Part 2

Name: Rishabh Jain

Branch & Sem: CSE - IV

Sec: A

Roll no.:54  
  
**Aim:** Develop an application for Inter-Process Communicationusing pipes.

CODE:

**Program1**

#include <stdio.h>

int main()

{

    int pid, pip[2];

    char instring[20];

    pipe(pip);

    pid = fork();

    if (pid == 0)

    {

        write(pip[1], "Hi World!", 10);

    }

    else /\* parent : receives message from child \*/

    {

        read(pip[0], instring, 10);

        for (int i = 0; i <= 10; i++)

            printf("%c", instring[i]);

        printf("\n");

    }

}

OUTPUT:



**Program2**

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#define MSG\_LEN 64

int main()

{

    int result;

    int fd[2];

    char \*message = "Linux World!!!";

    char recvd\_msg[MSG\_LEN];

    result = pipe(fd); // Creating a pipe fd[0] is for reading andfd[1] is for writing

    if (result < 0)

    {

        perror("pipe ");

        exit(1);

    }

    // writing the message into the pipe

    result = write(fd[1], message, strlen(message));

    if (result < 0)

    {

        perror("write");

        exit(2);

    }

    // Reading the message from the pipe

    result = read(fd[0], recvd\_msg, MSG\_LEN);

    if (result < 0)

    {

        perror("read");

        exit(3);

    }

    printf("%s\n", recvd\_msg);

    return 0;

}

OUTPUT:



**Program3:**

#include <stdio.h>

#include <stdlib.h>

main()

{

    int pipefd[2], n, pid;

    char buff[100];

    pipe(pipefd);

    printf("\n readfd=%d", pipefd[0]);

    printf("\n writefd=%d", pipefd[1]);

    pid = fork();

    if (pid == 0)

    {

        close(pipefd[0]);

        printf("\n CHILD PROCESS SENDING DATA\n");

        write(pipefd[1], "hello world", 12);

    }

    Else

    {

        close(pipefd[1]);

        printf("PARENT PROCESS RECEIVES DATA\n");

        n = read(pipefd[0], buff, sizeof(buff));

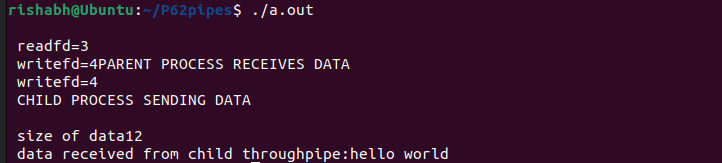
        printf("\n size of data%d", n);

        printf("\n data received from child throughpipe:%s\n", buff);

    }

}

**OUTPUT:**

****

**Program 4:**

CREATION OF A TWOWAY PIPE BETWEEN TWO PROCESS

#include <stdio.h>

#include <stdlib.h>

main()

{

    int p1[2], p2[2], n, pid;

    char buf1[25], buf2[25];

    pipe(p1);

    pipe(p2);

    printf("\n readfds=%d %d\n", p1[0], p2[0]);

    printf("\n writefds=%d %d\n", p1[1], p2[1]);

    pid = fork();

    if (pid == 0)

    {

        close(p1[0]);

        printf("\n CHILD PROCESS SENDING DATA\n");

        write(p1[1], "where is GEC", 25);

        close(p2[1]);

        read(p2[0], buf1, 25);

        printf(" reply from parent:%s\n", buf1);

        sleep(2);

    }

    Else

    {

        close(p1[1]);

        printf("\n parent process receiving data\n");

        n = read(p1[0], buf2, sizeof(buf2));

        printf("\n data received from child through pipe:%s\n", buf2);

        sleep(3);

        close(p2[0]);

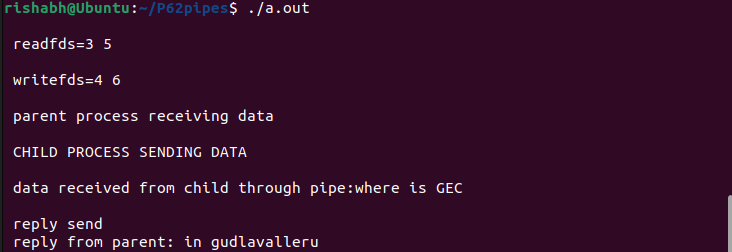
        write(p2[1], " in gudlavalleru", 25);

        printf("\n reply send\n");

    }

}

OUTPUT:



Program 5:

#include <stdio.h>

#include <unistd.h>

#include <string.h>

main()

{

    int p1[2], p2[2], p3[2], p4[2];

    int i, j = 0, k = 0, l = 0;

    char r[10], s[10], t[10], u[10];

    printf("\t PROCESS 1.ENTER THE STRING");

    scanf("%s", r);

    pipe(p1);

    pipe(p2);

    write(p1[1], r, sizeof(r));

    write(p2[1], r, sizeof(r));

    int a = fork();

    if (a == 0)

    {

        printf("\n\t PROCESS 2:it splits the given string\n");

        read(p1[0], r, sizeof(r));

        int n = strlen(r);

        for (i = 0; i < n / 2; i++)

        {

            s[i] = r[i];

        }

        for (i = n / 2; i <= n; i++)

        {

            t[j++] = r[i];

        }

        pipe(p3);

        pipe(p4);

        write(p3[1], s, sizeof(s));

        write(p4[1], t, sizeof(t));

        int b = fork();

        if (b == 0)

        {

            printf("p4 %d\t", getpid());

            printf("p2 %d\n", getppid());

            read(p3[0], s, sizeof(s));

            printf("\t PROCESS 4:sub string \t %s \t", s);

            printf("no of char=%d \n", strlen(s));

        }

        else

        {

            int c = fork();

            if (c == 0)

            {

                printf("p5 %d\t", getpid());

                printf("p2 %d\n", getppid());

                read(p4[0], t, sizeof(t));

                printf("\t PROCESS 5:sub string \t %s \t", t);

                printf("no of char=%d \n", strlen(t));

            }

            else

            {

                wait();

                printf("p2 %d\t", getpid());

                printf("p1 %d\n", getppid());

            }

        }

    }

    else

    {

        wait();

        int d = fork();

        if (d == 0)

        {

            printf("p3 %d\t", getpid());

            printf("p1 %d\n", getppid());

            read(p2[0], r, sizeof(r));

            for (i = strlen(r) - 1; i >= 0; i--)

            {

                u[l++] = r[i];

            }

            for (i = 0; i < strlen(r); i++)

            {

                if (u[i] == r[i])

                    k++;

                else

                    continue;

            }

            if (k == strlen(r))

                printf("\t PROCESS 3: the given string is palindrome\n");

            else

                printf("\t PROCESS 3: the given string is not palindrome\n");

        }

        else

        {

            printf("p1 %d\t", getpid());

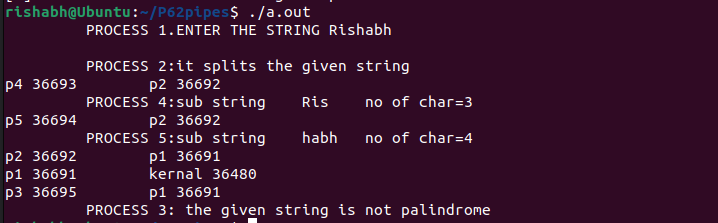
            printf("kernal %d\t\n", getppid());

        }

    }

}

OUTPUT:



**Program 6:**

#include <stdio.h>

#include <unistd.h>

#include <string.h>

#include <sys/wait.h>

int main()

{

    int p1[2], p2[2], p3[2], p4[2];

    int i, j = 0, k = 0, l = 0;

    char r[10], s[10], t[10], u[10];

    printf("ENTER THE STRING : ");

    scanf("%s", r);

    pipe(p1);

    pipe(p2);

    write(p1[1], r, sizeof(r));

    write(p2[1], r, sizeof(r));

    int a = fork();

    if (a == 0)

    {

        printf("PROCESS 2\n");

        printf("SPLIT THE STRING\n");

        read(p1[0], r, sizeof(r));

        int len = strlen(r);

        for (int i = 0; i < len / 2; i++)

            s[i] = r[i];

        for (int i = len / 2; i < len; i++)

            t[j++] = r[i];

        pipe(p3);

        pipe(p4);

        write(p3[1], s, sizeof(s));

        write(p4[1], s, sizeof(s));

        int b = fork();

        if (b == 0)

        {

            printf("p4 --> %d\t", getpid());

            printf("p2 --> %d\n", getppid());

            read(p3[0], s, sizeof(s));

            printf("PROCESS 4\nSub string --> %s \n", s);

            printf("no of char = %ld \n", strlen(s));

        }

        else

        {

            int c = fork();

            if (c == 0)

            {

                printf("p5 --> %d\t", getpid());

                printf("p2 --> %d\n", getppid());

                read(p4[0], t, sizeof(t));

                printf("PROCESS 5\nSub string --> %s \n", t);

                printf("no of char = %ld \n", strlen(t));

            }

            else

            {

                wait(NULL);

                printf("p2 --> %d\t", getpid());

                printf("p1 --> %d\n", getppid());

            }

        }

    }

    else

    {

        wait(NULL);

        int d = fork();

        if (d == 0)

        {

            printf("p3 %d\t", getpid());

            printf("p1 %d\n", getppid());

            read(p2[0], r, sizeof(r));

            for (i = strlen(r) - 1; i >= 0; i--)

            {

                u[l++] = r[i];

            }

            for (i = 0; i < strlen(r); i++)

            {

                if (u[i] == r[i])

                    k++;

                else

                    continue;

            }

            if (k == strlen(r))

printf("PROCESS 3\nThe given string is

palindrome\n");

else

printf("PROCESS 3\nThe given string is not

palindrome\n");

        }

        else

        {

            printf("p1 %d\t", getpid());

            printf("PARENT %d\t\n", getppid());

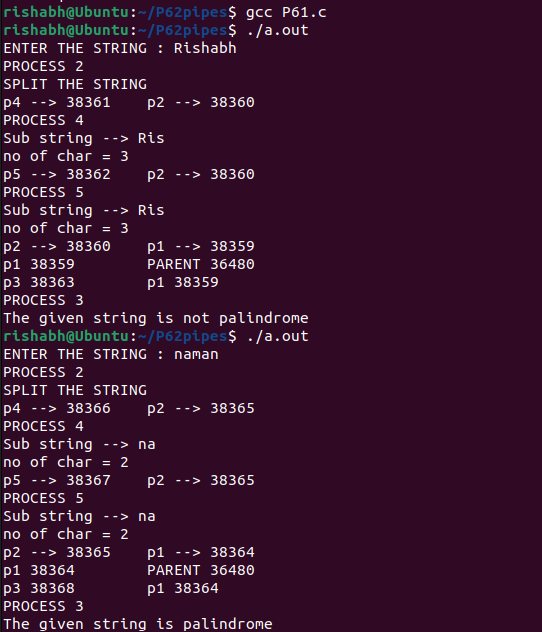
        }

    }

    return 0;

}

**OUTPUT:**

****

**Program- 7:**

Write a C program to implement the following game. The parent programP first creates two pipes, and then spawns two child processes CandD. One of the two pipes is meant for communications between P andC, andthe other for communications between P and D. Now, a loop runs asfollows. In each iteration (also called round), P first randomly choosesone of the two ags: MIN and MAX (the choice randomly varies fromone iteration to another). Each of the two child processes C and Dgeneratesarandom positive integer and sends that to P via its pipe. P reads thetwointegers; let these be c and d. If P has chosen MIN, then the childwhosent the smaller of c and d gets one point. If P has chosen MAX, thenthesender of the larger of c and d gets one point. If c = d, then this roundisignored. The child process who first obtains ten points wins the game. When the game ends, P sends a user-defined signal to both CandD, andthe child processes exit after handling the signal (in order to knowwhowas the winner). After C and D exit, the parent process P exits. Duringeach iteration of the game, P should print appropriate messages (likeP'schoice of the ag, the integers received from C and D, which child gets thepoint, the current scores of C and D) in order to let the user knowhowthegame is going on. Name your program childsgame.c .

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <time.h>

#include <unistd.h>

int childCScore = 0;

int childDScore = 0;

int main()

{

    int pipe1[2], pipe2[2];

    int result1 = pipe(pipe1);

    int result2 = pipe(pipe2);

    int c, d, num1, num2;

    if (result1 == -1 || result2 == -1)

    {

        printf("Pipe Failed !!\n");

        exit(1);

    }

    int childA, childB;

    childA = fork();

    if (childA == -1)

    {

        printf("Fork Failed.\n");

        exit(1);

    }

    else if (childA == 0)

    {

        // CHILD PROCESS C

        printf("Child Process C.\n");

        close(pipe1[0]);

        close(pipe2[1]);

        srand(time(NULL));

        while (childCScore < 10)

        {

            num1 = rand() % 100 + 1;

            write(pipe1[1], &num1, sizeof(int));

        }

        close(pipe1[1]);

    }

    childB = fork();

    if (childB == -1)

    {

        printf("Fork Failed.\n");

        exit(1);

    }

    else if (childB == 0)

    {

        // CHILD PROCESS D

        printf("Child Process D.\n");

        close(pipe1[1]);

        close(pipe2[0]);

        sleep(1);

        srand(time(NULL));

        while (childDScore < 10)

        {

            num2 = rand() % 100 + 1;

            write(pipe2[1], &num2, sizeof(int));

        }

        close(pipe2[1]);

    }

    // PARENT PROCESS

    close(pipe1[1]);

    close(pipe2[1]);

    int roundNo = 1;

    srand(time(NULL));

    while (childCScore < 10 && childDScore < 10)

    {

        int choice = rand() % 2;

        printf("Round Number: %d\n", roundNo);

        if (choice == 0)

            printf("Parent Process chose MIN\n");

        else

            printf("Parent Process chose MAX\n");

        read(pipe1[0], &num1, sizeof(int));

        read(pipe2[0], &num2, sizeof(int));

        c = num1;

        d = num2;

        printf("Child C chose the number: %d\n", c);

        printf("Child D chose the number: %d\n", d);

        if ((choice == 0 && c < d) || (choice == 1 && c > d))

        {

            childCScore++;

            printf("Child C got the point.\n");

        }

        else if ((choice == 0 && c > d) || (choice == 1 && c < d))

        {

            childDScore++;

            printf("Child D got the point.\n");

        }

        else

            printf("Round Ignored.\n");

        printf("Child C score is %d\n", childCScore);

        printf("Child D score is %d\n\n", childDScore);

        roundNo++;

    }

    if (childCScore == 10)

        printf("Winner-C\n");

    else if (childDScore == 10)

        printf("Winner-D\n");

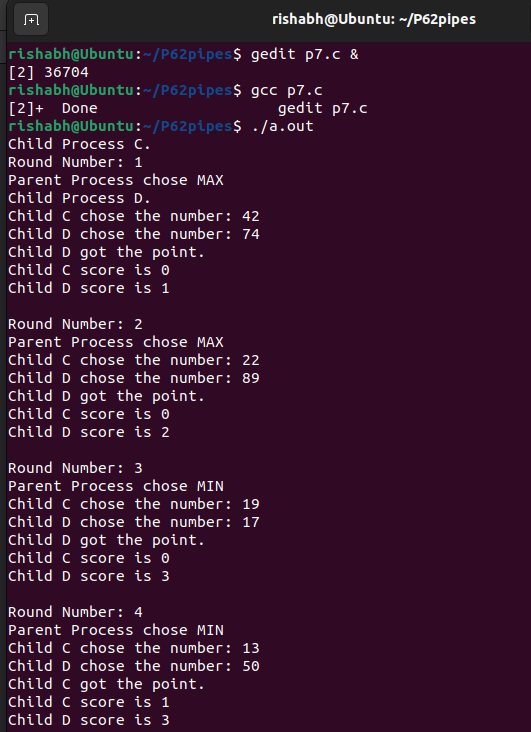
    close(pipe1[0]);

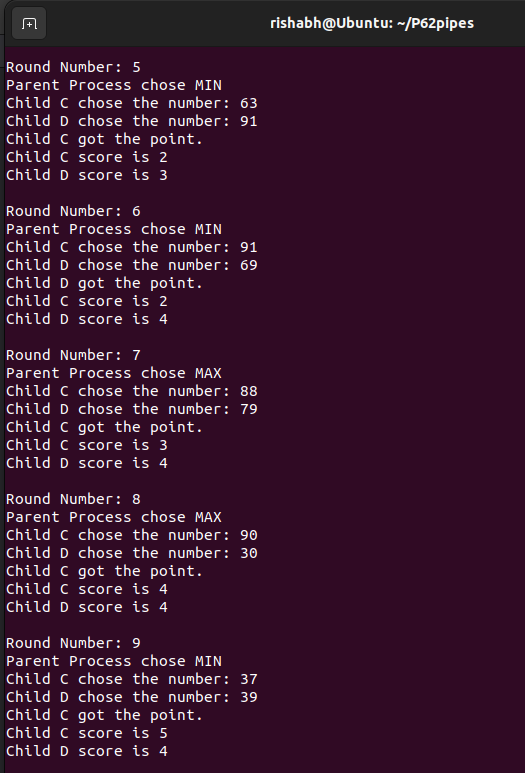
    close(pipe2[0]);

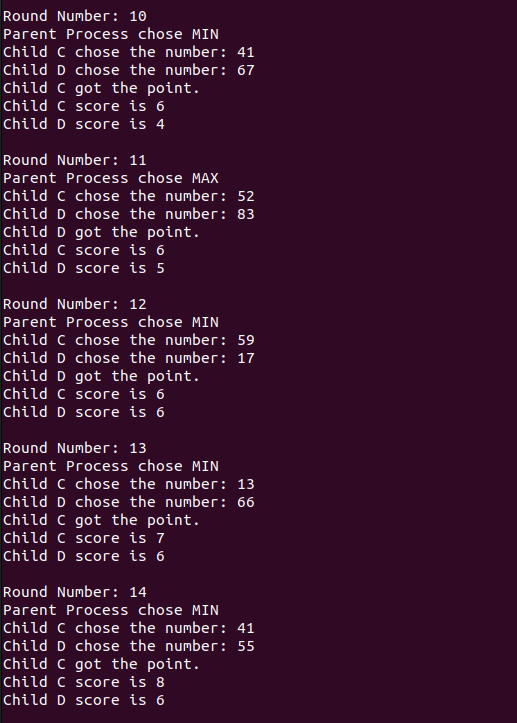
    return 0;

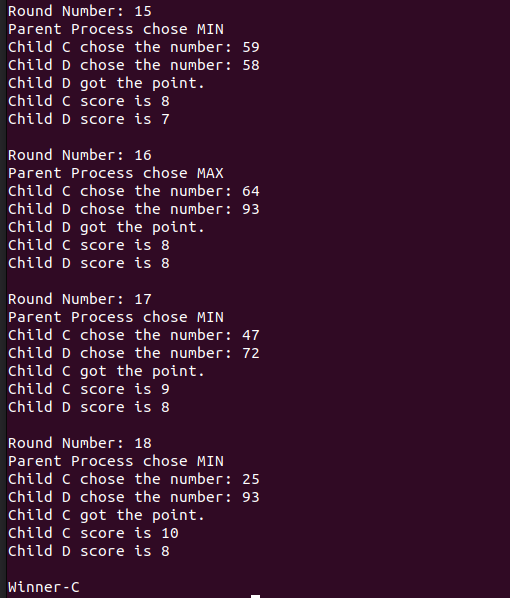
}

OUTPUT:

****

****

****

****

**Result:** In this practical we learned Linux C programs for Inter-Process Communication using pipes.