

دانشگاه صنعتي امیرکبیر  
(پلی تکنیک تهران)

دانشكده مهندسی برق

آزمایشگاه سیستم عامل

آزمایش۴

استفاده از مکانیزم هاي ارتباط بین فرآیند ها

نگارش

علی بابالو

پویا شریفی

استاد راهنما

مهندس کیخا

خرداد ماه 1402

بخش اول - Reader Writer :

برای برقراری ارتباط ۲ فرایند از طریق حافظه مشترک، دو برنامه یکی برای read و یکی برای write مینویسیم. ابتدا به توضیح برنامهwrite میپردازیم. در این برنامه ابتدا یک کلید با استفاده از ftok می­سازیم و یک حافظه مشترک با آن کلید ساخته میشود سپس این حافظه مشترک در قالب \*int یعنی در واقع یک آرایه به طول 2 به متغیر nums نسبت داده میشود و در نهایت دو عدد از ورودی گرفته شده و مقدار هر یک در یکی از دوخانه آرایه nums نوشته میشود. به این صورت دو عدد در داخل حافظه مشترک ثبت میشود.

#include <sys/ipc.h>

#include <sys/shm.h>

#include <stdio.h>

int main()

{

    key\_t key;

    key = ftok("/mnt/c/Users/98912/Desktop/uni/OS LAB/EXP 4/Q1", 'R');

    // shmget returns an identifier in shmid

    int shmid = shmget(key, sizeof(int) \* 2, 0666|IPC\_CREAT);

    // shmat to attach to shared memory

    int \*nums = (int\*) shmat(shmid,(void\*)0,0);

    printf("Write Data : ");

    scanf("%d", &nums[0]);

    printf("Write Data : ");

    scanf("%d", &nums[1]);

    printf("Write Data : ");

    scanf("%d", &nums[2]);

    printf("Data written in memory: %d, %d, %d\n", nums[0], nums[1], nums[2]);

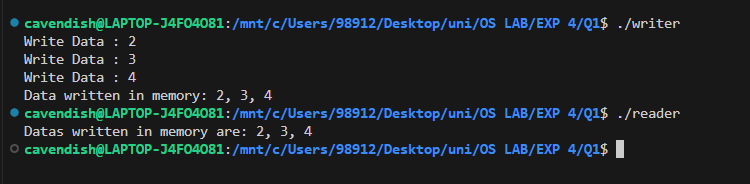
    //detach from shared memory

    shmdt(nums);

    return 0;

}

در قسمت reader نیز مجددا همانند کد قبلی یک حافظه مشترک با همان کلید قسمت writer ساخته میشود که چون کلید این حافظه با کلید حافظه برنامه قبلی یکسان است این دو حافظه به یک محل اشاره می کنند. سپس این حافظه مشترک در قالب \*int به متغیر nums نسبت داده میشود و در نهایت به کمک printf مقادیر موجود در آن حافظه نوشته می­شوند.



بخش دوم – server client :

کد مربوط به کالینت و سرور که به کمک حافظه مشترک نوشته شده و در دستور کار وجود دارد را اجرا میکنیم و خروجی را در زیر مشاهده می­کنید:

A picture containing screenshot, text

Description automatically generated

بخش سوم – ارتباط با کمک pipeline :

برای ایجاد این ارتباط یک پایپ الین میسازیم، میدانیم که هر پایپ الین دوdescriptors به ما میدهد که یکی برای نوشتن در پایپ الین و دیگری برای خواندن از آن است. پردازش رشته 6 سپس به کمک دستور ()frok یک پردازه فرزند میسازیم و در آن ابتدا descriptors مربوط به خواندن را میبندیم زیرا از طریق پردازه فرزند فقط میخواهیم در پایپ الین بنویسیم ، سپس به کمک دستور write رشته ای را که به عنوان آرگومان به برنامه داده شده است را در پایپ الین مینویسیم. در پردازه پدر ابتدا صبر میکنیم کار فرزند که نوشتن در پایپ الین باشد تمام شود سپس descriptors مربوط به نوشتن را بسته و به کمک دستور read و descriptors مربوط به آن محتوای موجود در پایپ الین را میخوانیم و در pipelineText میریزیم، در نهایت نیز روی رشته پیمایش کرده و با توجه به کد اسکی هر کاراکتر تشخیص میدهیم که کاراکتر حروف بزرگ است یا کوچک و آن را تغییر میدهیم و نتیجه نهایی را چاپ میکنیم.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/wait.h>

#include <unistd.h>

int main(int argc, char \*argv[])

{

    if (argc == 1)

    {

        printf("Enter some srting after command!");

        return 1;

    }

    // create two pipes:

    // - descriptors used by parent to write to child

    int descriptors[2];

    // read:[0] - write:[1]

    if (pipe(descriptors) != 0)  /\* || not && \*/

    {

        fprintf(stderr, "pipes failed!\n");

        return 1;

    }

    // fork() child process

    int child = fork();

    if (child < 0)

    {

        fprintf(stderr, "fork failed!");

        return 1;

    }

    else if (child != 0)    /\* parent \*/

    {

        while (wait(NULL) > 0);

        // close unwanted pipe (write)

        close(descriptors[1]);

        // read from pipe

        char pipelineText[100];

        int n = read(descriptors[0], pipelineText, sizeof(pipelineText) - 1);

        pipelineText[n] = '\0';

        printf("Parent: read from pipe '%s'\n", pipelineText);

        close(descriptors[0]);

        for (int i = 0; pipelineText[i]!='\0'; i++)

        {

            if(pipelineText[i] >= 'a' && pipelineText[i] <= 'z')

            {

                pipelineText[i] = pipelineText[i] - 32;

        }

            else if(pipelineText[i] >= 'A' && pipelineText[i] <= 'Z')

            {

                pipelineText[i] = pipelineText[i] + 32;

        }

        }

        printf("Parent: after processing: %s\n",pipelineText);

    }

    else

    {

        // close unwanted pipe

        close(descriptors[0]);

        // write from terminal to pipe

        printf("Child: writing to pipe '%s'\n", argv[1]);

        write(descriptors[1], argv[1], strlen(argv[1]));

        close(descriptors[1]);

    }

    return 0;

}

خروجی در زیر قابل مشاهده است:

