//SO PRINTF I SCANF

/\*Да се напише C програма за комуникација помеѓу три процеси. Процесите комуницираат преку pipe комуникација, така што првиот процес комуницира со вториот процес преку еден pipe, додека пак вториот комуницира со третиот процес преку втор pipe. Првиот процес чита цели броеви од тастатура и број по број ги испраќа на вториот процес. Вториот процес, ги чита броевите, број по број, пресметува просек на моментално пристигнатите броеви, така што за секој нареден број се наново пресметува просекот вклучувајќи го и новиот број. Просекот на дотогаш внесените броеви го испраќа на третиот процес и чека нов број. Третиот процес, ги чита просек по просек и на екран го печати првиот просек, а секој нареден просек печати колава е разликата од претходниот просек.\*/

/\*Пример: Прв процес чита 12, втор процес пресметува 12, трет процес печати 12

Потоа, прв процес чита 8, втор процес пресметува 10, трет процес печати 10-12=-2

Потоа, прв процес чита 5, втор процес пресметува 25/3 = 6, трет процес печати 6-10=-4 И така натаму\*/

#include <stdio.h>

#include <unistd.h>  // за pipe, fork, dup2

#include <sys/types.h>

#include <stdlib.h>  // за exit

int main(){

    int pipe1[2], pipe2[2]; //imam 2 cefki, i dvete mi trebaat za zapisuvanje/citanje

    pipe(pipe1);

    pipe(pipe2);

    //pipe1[0] - cita od pip

    //pip1[1] - zapisuva vo pipe

    pid\_t dete1;

    pid\_t dete2;

    dete1=fork();

    if(dete1>0){ //Sum vo roditelot - glavniot proces

        close(pipe1[0]);

        dup2(pipe1[1], 1); //od standarden vlez da zimam podatoci i da idat na pipe

        close(pipe1[1]);

        int broj;

        printf("Vnesi broj od tastatura! ");

        while(scanf("%d", &broj) == 1){

            printf("%d\n", broj); //printf sega izleguva na pipe1

        }

        exit(0);

    }else if(dete1==0){

        dete2 = fork();

        if(dete2 > 0){ //proces 2 koj e roditel na proces 3

            close(pipe1[1]);

            dup2(pipe1[0], 0);

            close(pipe1[0]);

            //scanf mi e nasocen kon pipe1

            close(pipe2[0]);

            dup2(pipe2[1],1);

            close(pipe2[1]);

            //tuka veke printf mi e nasocen kon pipe2;

            int broj;

            int suma = 0;

            int brojac = 0;

            while(scanf("%d", &broj) == 1){

                brojac++;

                suma+=broj;

                int prosek = suma/brojac;

                printf("%d", prosek);

            }

            exit(0);

        }else if(dete2 == 0){//ova znaci deka e proces 3 koj e dete na proces 2

            //0-cita

            //1-zapisuva

            close(pipe2[1]);

            dup2(pipe2[0], 0);

            close(pipe2[0]);

            //ovde scanf mi doaga od pipe2

            //printf si ide na ekran

            int broj;

            int prethodnik;

            int flag = 0;

            int razlika = 0;

            while(scanf("%d", &broj) == 1){

                if(flag == 0){

                    printf("%d", broj);

                    flag = 1;

                }else{

                    razlika = broj - prethodnik;

                    printf("%d", razlika);

                }

                prethodnik = broj;

            }

            exit(0);

        }

    }

    return 0;}

//ISTA ZADACA KAKO PIPE1

//SAMO KORISTIME READ I WRITE

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main() {

    int pipe1[2]; // pipe помеѓу родител и процес 2

    int pipe2[2]; // pipe помеѓу процес 2 и процес 3

    if (pipe(pipe1) == -1) {

        perror("pipe1 error");

        exit(1);

    }

    if (pipe(pipe2) == -1) {

        perror("pipe2 error");

        exit(1);

    }

    pid\_t pid1 = fork();

    if (pid1 > 0) {

        // Родителски процес - чита од тастатура и праќа во pipe1

        close(pipe1[0]); // Затвори читање од pipe1 - не ни треба

        close(pipe2[0]); // Не користи pipe2

        close(pipe2[1]); // Не користи pipe2

        int broj;

        printf("Внеси цели броеви (Ctrl+D за крај):\n");

        while (scanf("%d", &broj) == 1) {

            if (write(pipe1[1], &broj, sizeof(int)) != sizeof(int)) {

                perror("Грешка при пишување во pipe1");

                break;

            }

        }

        close(pipe1[1]); // Затвори го pipe1 кога завршуваш

        wait(NULL); // Чекај детето

        exit(0);

    } else if (pid1 == 0) {

        pid\_t pid2 = fork();

        if (pid2 > 0) {

            // Процес 2 - чита од pipe1, пресметува просек, праќа во pipe2

            close(pipe1[1]); // Затвори пишување во pipe1

            close(pipe2[0]); // Затвори читање од pipe2

            int broj;

            int suma = 0;

            int brojac = 0;

            while (read(pipe1[0], &broj, sizeof(int)) == sizeof(int)) {

                brojac++;

                suma += broj;

                int prosek = suma / brojac;

                if (write(pipe2[1], &prosek, sizeof(int)) != sizeof(int)) {

                    perror("Грешка при пишување во pipe2");

                    break;

                }

            }

            close(pipe1[0]);

            close(pipe2[1]);

            wait(NULL); // Чекај детето

            exit(0);

        } else if (pid2 == 0) {

            // Процес 3 - чита од pipe2 и печати на екран

            close(pipe2[1]); // Затвори пишување во pipe2

            close(pipe1[0]); // Не користи pipe1

            close(pipe1[1]); // Не користи pipe1

            int broj;

            int prethoden\_prosek = 0;

            int prvi\_prosek = 1;

            while (read(pipe2[0], &broj, sizeof(int)) == sizeof(int)) {

                if (prvi\_prosek) {

                    printf("%d\n", broj);

                    prvi\_prosek = 0;

                } else {

                    int razlika = broj - prethoden\_prosek;

                    printf("%d\n", razlika);

                }

                prethoden\_prosek = broj;

            }

            close(pipe2[0]);

            exit(0);

        }

    } return 0;}

// Да се напише програма во C програмскиот јазик за броење на зборови во датотека. // Програмата добива име на влезна датотека како аргумент од командна линија. Процесот чита // по 128 знаци од влезната датотека, притоа на секое второ читање ги праќа податоците до друг // процес кој ќе го врши броењето на зборовите (скокнува по едно читање т.е. ги праќа вторите 128 // знаци, па четвртите 128 знаци и т.н.). Како метод на комуникација помеѓу двата процеси да се // користи концепт на цевки, додека пак за броење на зборовите може да се користи наредбата

// „wc -w“ од Shell.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/wait.h>

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

    if(argc < 2){

        fprintf("Nema dovolno adrugmenti od komandna linija!");

        exit(1);

    }

    char \*ime\_datoteka = argv[1];

    pid\_t dete1;

    int pipe1[2];

    pipe(pipe1);

    if (pipe(pipe1) == -1) {

        perror("pipe1 error");

        exit(1);

    }

    //0-cita //1- zapisuva

    FILE \*file\_open = fopen(ime\_datoteka, "r");

    if(file\_open == NULL){

        perror("Greska pri otvaranje!");

        exit(1);

    }

    dete1 = fork();

    if(dete1>0){ //jas sum vo roditelot - glavniot proces

        close(pipe1[0]);

        dup2(pipe1[1], 1);

        close(pipe1[1]);

        //printf sega odi na pipe

        //scanf si ostanuva kaj so si e od datoteka

        int c;

        int flag = 0;

        int end = flag+128;

        int iteracija = 1;

        while(( c = fgetc (file\_open)) != EOF){ //FGETC -cita eden znak od datoteka izbrana

            //vo C mi stoi karakterot

            flag++;

            if(iteracija % 2 == 0){

                printf("%c", c);

            }

            if(flag+1 == end){

                flag = end;

                end = flag + 128;

                iteracija++;

            }

        }

        fclose(file\_open);

        exit(1);

    }else if(dete1 == 0){

        close(pipe1[1]);

        dup2[pipe1[0], 0];

        close(pipe1[0]);

        //scanf mi e od pipe

        //printf si ide kaj saka

        int c;

        int brojac= 0;

        int in\_word = 0; //deka ne sum vo zbor;

        while((c=getchar()) != EOF){ //GETCHAR = GETC -cita od vlez

            if(!isspace(c)){

                if(!in\_word){

                    brojac++;

                    in\_word=1;

                }

            }else{

                in\_word = 0;

            }

        }

        printf("Vkupno zborovi ima %d", brojac);

        // execlp("wc", "wc", "-w", NULL);

        // perror("Neuspesno izvresno!");

        exit(1);

    }  return 0; }

/\*.Да се напише програма во С програмскиот јазик за pipeline извршување на процеси. На почеток,

програмата дозволува внесување на вредност Х од тастатура, притоа треба да се креираат Х процеси кои

што ќе комуницираат преку pipeline (Х не повеќе од 10). Наредбите се без аргументи и се внесуваат од

тастатура, така што, првата внесена наредба од тастатура е првиот процес, втората наредба е вториот

процес и т.н. Секој нареден процес, комуницира со следниот преку pipeline, така што чита од

претходниот процес, печати во следниот процес (првиот процес – наредба чита од тастатура, додека

последниот процес печати на екран).

2.Бонус: Да се внесе во првиот процес дополнителна вредност Y која, доколку има вредност 0, тогаш

pipeline е стандарден, но, доколку Y е различно од 0, тогаш секоја наредба од pipeline-от, исто така да

печати и на екран и во десниот pipeline (слично на shell наредбата tee).

Пример:

./zadaca

3

ls

cat

wc

Резултатот треба да ја реализира следнава shell наредба: ls | cat | w\*/

#include <sys/types.h>      // За типот pid\_t

#include <stdio.h>          // За printf и scanf

#include <unistd.h>         // За fork, pipe, dup2, execlp

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

    pid\_t dete;             // PID за секој дете-процес

    int X, i;               // X е бројот на процеси, i е бројач

    int p[10][2];           // Маса за максимум 10 pipe-ови (секој има читање и запишување)

    char n[10][30];         // Маса за максимум 10 наредби (секоја со најмногу 30 букви)

    printf("Vnesi go X: ");     // Побарај од корисникот број на процеси

    scanf("%d", &X);            // Сними го тој број во X

    for(i = 0; i < X; i++) {

        printf("Vnesi naredba %d: ", i + 1); // Побарај наредба за секој процес

        scanf("%s", n[i]);                  // Сними ја наредбата во n[i]

    }

    for(i = 0; i < X - 1; i++) {

        pipe(p[i]);         // Креирај pipe p[i]

        //ito znaci btojot na pipe-ot

        dete = fork();      // Направи дете-процес

        if(dete > 0)        // Ако сме родител (враќа PID > 0)

            break;          // Прекини го циклусот – родителот излегува

        // Детето продолжува и создава следен процес

    }

    if(i != 0) {

        close(p[i - 1][1]);             // Ако не сме првиот процес, затвори го делот за запишување на претходниот pipe

        dup2(p[i - 1][0], 0);           // Поврзи го стандардниот влез со читање од претходниот pipe

    }

//1-pisuvanje

//0-citanje

    if(i != X - 1) {

        close(p[i][0]);                 // Ако не сме последниот процес, затвори го делот за читање на тековниот pipe

        dup2(p[i][1], 1);               // Поврзи го стандардниот излез со запишување во следниот pipe

    }

    execlp(n[i], n[i], NULL);           // Изврши ја наредбата n[i] (заменува го процесот со таа команда)

    printf("neuspeshno izvrshuvanje na naredba %s\n", n[i]); // Ако execlp не успее

    return 0; // Заврши ја програмата

}