// Да се напише програма во C која како аргумент добива име на непостоечка датотека.// Програмата најпрво креира дете процес. Родител процесот треба да генерира 1000 случајни // броеви со помош на функцијата rand() (од библиотеката stdlib.h) и истите ги запишува на екран // (кои со помош на редирекција се запишуваат во датотеката) и родител процесот завршува. Дете // процесот проверува дали може да ја отвори датотеката, и ако не може заспива 1 секунда. Потоа

// проверува пак, и заспива пак 1 секунда, се додека не успее да ја отвори датотеката. Чим успее // да ја отвори, чита 1000 броеви од тастатура редиректирани од датотеката) и ги сместува во // низа. Потоа, дете процесот, со помош на 4 нитки (секоја нитка пребарува по 250 броеви), го // наоѓа најголемиот број и го печати на екран.

//PROGRAMA

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <pthread.h>

#include <time.h>

#include <string.h>

#define dolzina\_niza 1000

#define broj\_nitki 4

int niza\_so\_broevi[dolzina\_niza];

int niza\_za\_max\_el\_lokalno[broj\_nitki];

void \*prebaraj(void \*arg){

    int indeks = \*(int\*)arg;

    int pocetok, kraj;

    if(indeks == 0){

        pocetok = 0;

        kraj = 249;

    }else if(indeks == 1){

        pocetok = 250;

        kraj = 499;

    }else if(indeks == 2){

        pocetok = 500;

        kraj = 749;

    }else if(indeks == 3){

        pocetok = 750;

        kraj = 999;

    }

    int max = niza\_so\_broevi[pocetok];

    for(int i=pocetok + 1; i<= kraj; i++){

        if(niza\_so\_broevi[i] > max){

            max  = niza\_so\_broevi[i];

        }

    }

    niza\_za\_max\_el\_lokalno[indeks] = max;

    return NULL;

}

int main(int argc, char \*argv[]){ //ARGC broj argumenti na komandna linija

    if(argc != 2){

        printf("Nema datoteka na sama komandna linija!");

        exit(1);

    }

    char \*ime\_datoteka = argv[1];

    pid\_t pid = fork(); //od ovoj moment imam 2 procesi koi se izvrsuvaat

    if(pid < 0){

        perror("ERROR");

        exit(1);

    }

    if(pid > 0){ //RODITEL PROCES

        srand(time(NULL));

        int ekran = dup(1); //privremeno

        for(int i=0; i<dolzina\_niza; i++){

            int random\_broj = rand() % 1000; // do kade sakam da odi

            dup2(ekran,1);

            printf("%d", random\_broj); //redirekcijata ja prai vo komandna linija,       SAMO ZA EKRANNNNNN

            //REDIREKCIJA BEZ GORNIOT PRINTF

            // dup2(FD, 1); //1 ekran, 0 tastatura

            // printf("....")

        }

        return 0; //tuka roditelot zavrsil

    }else if (pid == 0){//DETE PROCES

        FILE \*f = NULL;

        while((f = fopen(ime\_datoteka, "r")) == NULL){ //se dodeka ne moze da se otvori spij

            sleep(1);

        }

        for(int i=0; i<dolzina\_niza; i++){

            if(fscanf(f,"%d",&niza\_so\_broevi[i]) != 1){

                fprintf(stderr, "Greska pri citanje na broj! ");

                fclose();

                return 1;

            }

        }

        fclose(f);

        pthread\_t threads[broj\_nitki]; //ja cuva ID na sekoja nitka

        int indeks\_nitka[broj\_nitki]; //preku ova ja prakam nitkata na funkcijata

        for(int i=0; i<broj\_nitki; i++){

            indeks\_nitka[i] = i;

            pthread\_create(&threads[i], NULL, prebaraj, &indeks\_nitka[i]); //kreira nova nitka, sekoe prebaraj dobiva razlicen indeks

            //so &indeks\_nitka se praka na funkcijata prebaraj

        }

        for(int i=0; i<broj\_nitki; i++){

            pthread\_join(threads[i], NULL); //sekoja nitka cekam da zavrsi

        }

        int globalen\_maksimum = niza\_za\_max\_el\_lokalno[0];

        for(int i=0; i<broj\_nitki; i++){

            id(niza\_za\_max\_el\_lokalno[i] > globalen\_maksimum){

                globalen\_maksimum=niza\_za\_max\_el\_lokalno[i];

            }

        }

        printf("Najgolemiot element e: %d", globalen\_maksimum);

    }

}

//PROGRAMA 1

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <pthread.h>

#include <time.h>

#include <string.h>

#define KEY 12345

#define dolzina\_na\_niza 1000

#define maksimalen\_broj\_nitki 10

int \*shared\_memory;

int broj\_nitki;

int niza\_na\_vrednost\_na\_sekoja\_nitka[maksimalen\_broj\_nitki];

void \*prebaraj( void \*arg){ //LOGIKATA ZA SEKOJA NITKA

    int indeks = \*(int \*)arg;

    int broj = niza\_na\_vrednost\_na\_sekoja\_nitka[indeks];

    while(1){

        sleep(1);

        if(shared\_memory[dolzina\_na\_niza] == indeks){ //DALI E NITKATA IZBRANA

            int brojac = 0;

            for(int i=0; i<dolzina\_na\_niza; i++){

                if(shared\_memory[i] == broj){

                    brojac++;

                }

            }

            printf("Nitkata %d go barase brojot %d i go najde %d pati! ", indeks, broj, brojac);

            shared\_memory[1001] = 1;

        }

    }

    return NULL;

}

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

    printf("Vnesi broj na nitki: ");

    scanf("%d", &broj\_nitki);

    if (broj\_nitki > maksimalen\_broj\_nitki){

        broj\_nitki = maksimalen\_broj\_nitki;

        return 1;

    }

    int shmid = shmget(KEY, sizeof(int) \* 1002, 0666 | IPC\_CREATE) // shmget(key, size, permissions | flags)

    // 1002 = 1000(NITKI) + 1(INDEKS KOJA NITKA DA RABOTI) + 1(FLAG DALI E ZAVRSENO)

    //IPC\_CREATE ako postoi ke ti go dade postoeckiot segment, a ako ne postoi ke sozdade nov

    if(shmid < 0){

        perror("Neuspesno mapiranje!");

        exit(1);

    }

    shared\_memory = (int \*)shmat(shmid, NULL, 0); //shared mem attach, shmat vraka void i zatoa jas imam (int \*)

    for(int i=0; i< broj\_nitki; i++){

        printf("Vnesi vrednost na samata nitka: ");

        scanf("%d", &niza\_na\_vrednost\_na\_sekoja\_nitka[i]);

    }

    pthread\_t threads[maksimalen\_broj\_nitki]; //ja cuva ID na sekoja nitka

    int indeks\_nitka[maksimalen\_broj\_nitki]; //preku ova ja prakam nitkata na funkcijata

    for(int i=0; i<broj\_nitki; i++){

        indeks\_nitka[i] = i;

        pthread\_create(&threads[i], NULL, prebaraj, &indeks\_nitka[i]); //kreira nova nitka, sekoe prebaraj dobiva razlicen indeks

        //so &indeks\_nitka se praka na funkcijata prebaraj

    }

    for(int i=0; i<broj\_nitki; i++){

        pthread\_join(threads[i], NULL); //sekoja nitka cekam da zavrsi

    }

    return 0;

}

//PROGRAMA 2

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <pthread.h>

#include <time.h>

#include <string.h>

#define KEY 12345

#define dolzina\_na\_niza 1000

#define maksimalen\_broj\_nitki 10

int \*shared\_memory;

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

    int shmid = shmget(KEY, sizeof(int) \* 1002, 0666 | IPC\_CREAT) // shmget(key, size, permissions | flags)

    // 1002 = 1000(NITKI) + 1(INDEKS KOJA NITKA DA RABOTI) + 1(FLAG DALI E ZAVRSENO)

    //IPC\_CREATE ako postoi ke ti go dade postoeckiot segment, a ako ne postoi ke sozdade nov

    if(shmid < 0){

        perror("Neuspesno mapiranje!");

        exit(1);

    }

    shared\_memory = (int \*)shmat(shmid, NULL, 0); //shared mem attach, shmat vraka void i zatoa jas imam (int \*)

    srand(time(NULL));

    for(int i=0; i<dolzina\_na\_niza; i++){ //od 0 do 999 site breovi

        printf("Vnesi vrednost za lokacija %d: ", i);

        scanf("%d", &shared\_memory[i]);

    }

    int izvrseni\_nitki[maksimalen\_broj\_nitki] = {0};

    int vkupno = 0;

    while(vkupno <= broj\_nitki){

        int indeks\_nitka = rand() % broj\_nitki;

        if (izvrseni\_nitki[indeks\_nitka] == 1){

            continue;

        }

        shared\_memory[1000] = indeks\_nitka; //i kazuvam na prog 1 koja nitka da raboti

        shared\_memory[1001] = 0; //za restartiranje

        printf("Aktivirana e nitkata %d. Cekame odgovor.... ", indeks\_nitka);

        while(shared\_memory[1001] == 0){ //momentalno se izvrsuva dr i ti treba da spies

            sleep(1);

        }

        printf("Nitkata %d zavrsi so rabota. ", indeks\_nitka);

        izvrseni\_nitki[indeks\_nitka] = 1;

        vkupno++;

    shmdt(shared\_memory);

    return 0;

    }

}

//PROGRAMA 1

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <pthread.h>

#include <time.h>

#include <string.h>

#include <fcntl.h>          // za open()

#include <sys/stat.h>       // za struct stat

#define KEY 12345

char \*shared\_memory; //bidejki zapisuvam tekst a ne brojki

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

    if(argc != 3){

        printf("Nema dovolno argumenti na komandna linija!");

        exit(1);

    }

    char \*ime\_datoteka = argv[1];

    int kolku\_pati = atoi(argv[2]); //bidejki ne e adresa tuku si e obicna vrednost ne mi treba int \*kolku\_pati

    FILE \*file\_open = fopen(ime\_datoteka, "r"); //se otvara datoteka koja e pratena kako argument

    if( file\_open == NULL){

        perror("Ne moze da se otvori datotekata!");

        return 1;

    }

    fseek(file\_open, 0, SEEK\_END); //ODI DO KRAJ NA DATOTEKATA

    int golemina = ftell(file\_open); //VAKA SE VRAKA VKUPNA GOLEMINA NA DATOTEKA

    rewind(file\_open); //pointer na pocetok od datoteka

    if(golemina == -1){

        perror("ERROR");

        close(file\_open);

        exit(1);

    }

    int shmid = shmget(KEY, golemina+100, 0666 | IPC\_CREAT); //ovie 100 gi dodavam za prvata linija sto treeba da go zapisam brojot na izvrsuvanja

    if(shmid < 0){

        perror("Neuspesno mapiranje!");

        exit(1);

    }

    shared\_memory = (char \*)shmat(shmid, NULL, 0); //shared mem attach, shmat vraka void i zatoa jas imam (int \*)

    if(shared\_memory == (char \*) -1){

        perror("Neuspesno povrzuvanje na shared memory!");

        exit(1);

    }

    //ZAPISUVA PRVO KOLKU\_PATI

    sprintf(shared\_memory,"//%d\n", kolku\_pati); //bidejki kolku\_pati e int vrednost zatoa koristime sprINTf

    //funkcijata sprintf() raboti kako printf()

    int offset= strlen(shared\_memory);

    char red[1024];

    while(fgets(red, sizeof(red), file\_open)){ //se cita linija i se smestuva vo red

        strcpy(shared\_memory+offset, red);

        offset += strlen(red);

    }

    fclose(file\_open);

    stmdt(shared\_memory);

    return 1;

}

// PROGRAMA 2

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <pthread.h>

#include <time.h>

#include <string.h>

#define KEY 12345

#define File\_C "datoteka.c"

#define Pom\_za\_izvrsuvanje "./datoteka"

char \*shared\_memory; //bidejki zimam tekst a ne broevi

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

    int shmid = shmget(KEY, 0, 0666 | IPC\_CREAT); //ovie 100 gi dodavam za prvata linija sto treeba da go zapisam brojot na izvrsuvanja

    if(shmid < 0){

        perror("Neuspesno mapiranje!");

        exit(1);

    }

    shared\_memory = (char \*)shmat(shmid, NULL, 0); //shared mem attach, shmat vraka void i zatoa jas imam (int \*)

    if(shared\_memory == (char \*) -1){

        perror("Neuspesno povrzuvanje na shared memory!");

        exit(1);

    }

    int broj\_izvrsuvanja;

    sscanf(shared\_memory, "//%d\n", &broj\_izvrsuvanja);

    char \*kod\_start = strchr(shared\_memory, '\n');

    if(!kod\_start){

        fprintf(stderr, "Kodot ne e validen!");

        exit(1);

    }

    FILE \*file\_za\_zapisuvanje = fopen(File\_C, "w");

    if(!file\_za\_zapisuvanje){

        perror("Ne moze da se otvori!");

        exit(1);

    }

    fputs(kod\_start+1, file\_za\_zapisuvanje); // +1 za da otide vo nov red

    fclose(file\_za\_zapisuvanje);

    char komanda[1024]; //ova mi e linijata od komandna linija  gcc datoteka.c -o datoteka

    sprintf(komanda, "gcc %s -o %s", File\_C, Pom\_za\_izvrsuvanje); //sprintf POLNI promenliva komandna

    if(system(komanda) != 0){

        fprintf(stderr, "ERROR");

        exit(1);

    }

    for (int i=0; i<broj\_izvrsuvanja; i++){

        pid\_t pid = fork();

        if(pid == 0){

            execl(Pom\_za\_izvrsuvanje, Pom\_za\_izvrsuvanje, NULL); //EXECL jja izvrsuva programata, execl(path\_do\_programa, ime\_vo\_argv0, NULL);

            perror("ERROR");

            exit(1);

        }

    }

    for(int i=0; i<broj\_izvrsuvanja; i++){ //gi cekam site deca

        wait(NULL);

    }

    return 0;

}

// PROGRAMA

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <pthread.h>

#include <time.h>

#include <string.h>

#include <fcntl.h>          // za open()

#include <sys/stat.h>       // za struct stat

#define broj\_procesi 5

#define max\_broj\_znaci 1000

#define KEY 12345

char \*mmap\_memory; //pokazuvacot vo mapmemorijata

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

     if(argc != 2){

        printf("Nema dovolno argumenti na komandna linija!");

        exit(1);

    }

    char \*datoteka = argv[1];

    int fd = open(datoteka, O\_RDONLY);

    if(fd == -1){

        perror("ERROR");

        exit(1);

    }

    int golemina = lseek(fd, 0, SEEK\_END); //odi do kraj

    lseek(fd, 0, SEEK\_SET); //vraka na pocetok

    if (golemina == 0){

        printf("Datotekata e prazna!");

        exit(1);

    }

    char \*mapirani\_podatoci = mmap(NULL, golemina, PROT\_READ, MAP\_SHARED,fd, 0); //void \*mmap(void \*addr, size\_t length, int prot, int flags, int fd, off\_t offset);

    if(mapirani\_podatoci == MAP\_FAILED){

        perror("ERROR");

        exit(1);

    }

    close(fd);

    //KREIRANJE NA SHARE MEMORIJA ZA DA CUVA VO NEA FLAG, INDEKS, KRAJ

    int shmid = shmget(KEY, sizeof(int) \* 3, 0666 | IPC\_CREAT); //PAZI SIZEOFF(CHAR)!!!!!!

    if(shmid < 0){

        perror("ERROR");

        exit(1);

    }

    int \*shared\_memory = (int \*)shmat(shmid, NULL, 0); //shared mem attach, shmat vraka void i zatoa jas imam (int \*)

    if(shared\_memory == (void \*) -1){

        perror("Neuspesno povrzuvanje na shared memory!");

        exit(1);

    }

    int \*flag = &shared\_memory[0]; //proces dali e vo kriticna sostojba, 0-sloboden pristap, 1-ima proces

    int \*indeks = &shared\_memory[1]; //deka citame od pocetok, pa ponatamu se menuva

    int \*kraj = &shared\_memory[2]; //koga kraj = 1 togash e signalizacija za kraj

    \*flag = \*indeks = \*kraj =0;

    for(int i=0; i<broj\_procesi; i++){

        pid\_t pid = fork();

        if(pid == 0){

            srand(time(NULL)); //sekogash koga imam random nesto koristi go ova

            while(1){

                if(\*kraj){

                    exit(1);

                }

                if(\*flag == 0){

                    \*flag=1;

                    int ostatok = golemina - \*indeks;

                    if(ostatok <= 0){

                        \*kraj = 1;

                        \*flag = 0;

                        exit(1);

                    }

                    int x = rand()%90 + 10;

                    if (x > ostatok) {

                        x = ostatok;

                    }

                    for(int j=0; j<x; j++){

                        putchar(mapirani\_podatoci[\*indeks+j]); //citam

                    }

                    putchar('\n');

                    \*indeks += x;

                    \*flag = 0;

                    sleep(2);

                }

            }

        }

    }

    for(int i=0; i<broj\_procesi; i++){

        wait(NULL);

    }

    munmap(mapirani\_podatoci, golemina);

    shmdt(shared\_memory);

    shmctl(shmid, IPC\_RMID, NULL);

    return 0;

}

// mmap() е системска функција која ти овозможува директно да ја мапираш содржината на датотека во меморија, така што можеш да пристапуваш до неа како до обична низа (char \*).

//KOGA KORISTAM \* NE SE MENUVA ADRESATA, TUKU SAMO VREDNOSTA NA TAA ADRESA

// Да се напише С програма која работи со процеси и меѓупроцесна комуникација. Главниот процес, како

// аргументи од командна линија, добива наредба со аргументи која треба да ја изврши. Потоа, преку

// тастатура се внесува колку пати треба да се изврши соодветната наредба Z. Главниот процес креира Z деца

// процеси колку што треба да се изврши наредбата, притоа секој еден од деца процесите ја извршува

// наредбата еднаш. Иако деца процесите се креираат и се извршуваат паралелно, сепак треба да се овозможи

// механизам каде деца процесите ќе се чекаат помеѓу себе т.е. ќе се извршуваат сериски. Главниот процес

// треба да брои колку време им требало на деца процесите да завршат со извршување. Комуникацијата

// помеѓу деца процесите и синхронизацијата за да се постигне сериско извршување да се изведе со помош на

// заедничка (shared) меморија.

// Пример:

// ./prva wc -c -l -d

//PROCES

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <pthread.h>

#include <time.h>

#include <string.h>

#include <fcntl.h>          // za open()

#include <sys/stat.h>       // za struct stat

#define KEY 12345

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

    if (argc < 2){

        printf("NEdovolen broj na argumenti!");

        exit(1);

    }

    char \*naredba = argv[1];

    int broj\_naredbi;

    printf("Vnesi broj za kolku pati da se izvrsi naredbata:  ");

    scanf("%d", &broj\_naredbi);

    int shmid = shmget(KEY, sizeof(int), 0666 | IPC\_CREAT);

    if(shmid < 0){

        perror("ERROR");

        exit(1);

    }

    int \*shared\_memory = (int \*)shmat(shmid, NULL, 0); //shared mem attach, shmat vraka void i zatoa jas imam (int \*)

    if(shared\_memory == (void \*) -1){

        perror("Neuspesno povrzuvanje na shared memory!");

        exit(1);

    }

    \*shared\_memory = 0; //PRVIOT PROCES KE BIDE NA RED

    int vreme = 0;

    for(int i=0; i<broj\_naredbi; i++){

        pid\_t pid = fork();

        if(pid == 0){ //DETE PROCES

            while(\*shared\_memory != i); //SAMO 1 PROCES KE VLEZE DRUGITE CEKAAT DODEKA NE IM DOJDE REDOT

            char \*args[argc]; //vaka se zimaat argumenti od komandna linija, T.E NIZA SO GOLEMINA KOLKU STO IMA BROOJ NA ARGUMENTI

            for(int j=1; j<argc; j++){

                args[j-1] = argv[j]; //vo ARGS ke si pocnat od 0,1,2,3 da se popolnuvaat, dodeka od ARGV se zimaat od 1,2,3 bidejki prviot e ime na programa

            }

            args[argc-1] = NULL; //za da znae deka zavrsuva mora null

            // char \*args[] = {"ls", "-l", "-a", NULL};

            // execvp("ls", args);

s

            execvp(args[0], args); // ako ova raboti, kodot posle nego ne se izvrsuva

            perror("Neuspesno izvrsuvanje");

            exit(1);

        }

    }

    for(int i=0; i<broj\_naredbi;  i++){

        wait(NULL);

        (\*shared\_memory)++; //dozvola za sleden proces i taka ke se poklopat so i

        vreme++;

        sleep(1);

    }

    // time\_t start\_time = time(NULL);

    // time\_t end\_time = time(NULL);

    printf("Vkupno vreme za izvrsuvanje %d: ", vreme);

    shmdt(shared\_memory);

    shmctl(shmid, IPC\_RMID, NULL);

    return 0;

}

//SO SIGNALIZACIJA

// PROGRAMA 1

#include <stdio.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <pthread.h>

#include <time.h>

#include <string.h>

#include <fcntl.h>          // za open()

#include <sys/stat.h>       // za struct stat

#define sinhronizacija\_memory 1

#define KEY\_memory 128

#define memory 1024

#define signal 0

char \*mapirana\_memory;

void otkluci(char \*kluc, char \*tekst, int p)

{

    int k = strlen(kluc);

    int t = strlen(tekst);

    int i, sum, z = 0;

    for (i = 0; i < t; i++)

    {

        if (p == 0)

        {

            sum = kluc[z] + tekst[i];

            if (sum > 255)

            {

                sum -= 255;

            }

        }

        else

        {

            sum = kluc[z] - tekst[i];

            if (sum < 0)

            {

                sum += 255;

            }

        }

        tekst[i] = sum;

        i++;

        z++;

        if (z == k)

            z = 0;

    }

}

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

{

    if (argc < 2)

    {

        printf("Nema dovolno argumenti na komandna linija!");

        exit(1);

    }

    char \*ime\_datoteka = argv[1];

    int fd = open(ime\_datoteka, O\_RDONLY);

    if (fd == -1)

    {

        perror("ERROR, ne moze da se otvori");

        exit(1);

    }

    int golemina = lseek(fd, 0, SEEK\_END); // odi do kraj

    lseek(fd, 0, SEEK\_SET);                // vraka na pocetok

    if (golemina == 0)

    {

        printf("Datotekata e prazna!");

        exit(1);

    }

    //1024+128+1 - realna golemina na datotekata

    mapirana\_memory = mmap(NULL, golemina, PROT\_READ, MAP\_SHARED, fd, 0); // void \*mmap(void \*addr, size\_t length, int prot, int flags, int fd, off\_t offset);

    if (mapirana\_memory == MAP\_FAILED)

    {

        perror("ERROR");

        exit(1);

    }

    pid\_t pid = fork();

    if (pid < 0)

    {

        printf("Ne moze da rabotis nitu so dete nitu so roditel!");

        exit(1);

    }

    else if (pid > 0)

    { // Roditelot

        int kluc\_fd = open("kluc.out", O\_RDWR);

        char orginalen\_kluc[KEY\_memory];

        read(kluc\_fd, orginalen\_kluc, KEY\_memory);

        close(kluc\_fd);

        while(1){

            if(mapirana\_memory[signal] == 1){

                char \*kluc\_od\_dete = mapirana\_memory + 1; //mora da pocne od +1 bidejki na 0 mi e staven bitot za signalizacija

                char \*tekst = mapirana\_memory + 1 + KEY\_memory; //go zimam klucot pa posle mozam da pristapam do teksot

                if(strcmp (orginalen\_kluc, kluc\_od\_dete) == 0){

                    printf("Isti kluc! ");

                    otkluci(orginalen\_kluc, tekst, 1); //otsifriraj

                    mapirana\_memory[signal] = 2;

                    sleep(10);

                    otkluci(orginalen\_kluc,tekst, 0); //sifriraj

                    mapirana\_memory[signal] = 0;

                    break;

                }else{

                    printf("Gresen kluc");

                    mapirana\_memory[signal] = 0;

                }

            }

            wait(NULL);

            munmap(mapirana\_memory, memory);

            close(fd);

        }

    }else if(pid == 0){ //dete prices

        for(int obid=1; obid<=3; obid++){

            printf("Dete cita od tastatura, vnesi kluc: ");

            char kluc[KEY\_memory];

            fgets(kluc, KEY\_memory, stdin); //STDIN citaj od standarden vlez - tastatura (scanf)

            kluc[strcspn(kluc, "\n")] = 0; //na ovoj nacin ja zimam samo sodrzinata bez \n

            memcpy(mapirana\_memory, kluc, strlen(kluc) + 1);

            mapirana\_memory[signal] = 1; //mu signaliziram

            if(mapirana\_memory[signal] == 2){

                printf("Uspesno otklucena!");

                break;

            }else{

                printf("Obidi se pak");

            }

        }

        exit(1);

    }

}

#include <pthread.h>

#include <stdio.h>

#include <unistd.h>

#include <signal.h>

#include <sys/wait.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define max\_linii\_vo\_datoteka 1000

#define max\_linii\_procitani 10

#define max\_nitki 100

char \*ime\_datoteka;

int broj\_mali\_bukvi[max\_nitki]; //kreiram niza globalna

int broj\_golemi\_bukvi[max\_nitki]; //kreiram niza globalna za sekoja nitka

int redovi =0;

int broj\_nitki =0;

void \*prebaraj(void \*arg){

    int indeks = \*(int\*)arg; //od listata nitki jas rabotam so 1 nitka

    int start\_linija = indeks\*max\_linii\_procitani;

    int end\_linija = indeks\*max\_linii\_procitani + max\_linii\_procitani;

   if(end\_linija > redovi){

    end\_linija = redovi;

   }

   FILE \*file\_open = fopen(ime\_datoteka, "r"); //MORAM PAK DA JA OTVORAM BIDEJKI NE DOZVOLUVA SITE NITKI DA CITAAT ISTOVREMENO

    if(file\_open == NULL){

        perror("Datotekata ne moze da se otvori!");

        exit(1);

    }

    char linija[1024];

    int momentalno = 0;

    int br\_mal = 0;

    int br\_golemi =0;

    //so ova go postavuvam na pocetna pozicija vo samata datoteka

    while(momentalno < start\_linija && fgets(linija, sizeof(linija), file\_open)){

        momentalno++;

    }

    while(momentalno < end\_linija && fgets(linija, sizeof(linija), file\_open)){

        momentalno++;

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

            if(isupper((unsigned char)linija[i])){

                br\_golemi++;

            }else if(islower((unsigned char)linija[i])){

                br\_mal++;

            }

        }

    }

    fclose(file\_open);

    printf("Golemi bukvi za nitkata so indeks %d ima %d!", indeks,br\_golemi);

    printf("Mali bukvi za nitkata so indeks %d ima %d!", indeks,br\_mal);

    broj\_golemi\_bukvi[indeks] = br\_golemi;

    broj\_mali\_bukvi[indeks] = br\_mal;

    return NULL;

}

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

    if(argc < 2){

        printf("Nema dovolno argumeni od komandna linija!");

        exit(1);

    }

    ime\_datoteka = argv[1];

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

    if(file\_open == NULL){

        perror("Datotekata ne moze da se otvori!");

        exit(1);

    }

    //SO OVA JA DOBIVAM GOLEMINATA VO BAJTI

    fseek(file\_open, 0, SEEK\_END); // odi do kraj i vraka 0 ako e uspesno

    long golemina = ftell(file\_open); //vraka pozicija

    fseek(file\_open, 0 SEEK\_SET); //vraka na pocetok

    if (golemina == 0)

    {   printf("Datotekata e prazna!");

        exit(1);

    }

    //DA PROCITAM BROJ NA REDOVI OD DATOTEKA

    redovi = 0;

    char linija[1024];

    while(fgets(linija, sizeof(linija), file\_open) != NULL){

        redovi++;

    }

    int ostatok = redovi % 10;

    if( ostatok != 0){

        //kreirame redovi / 10 + 1(tuka ke bide ostatokot od liniite)

        broj\_nitki = redovi / 10 + 1;

    }else{

        //tocno ke se podelat site redovi/10 i ke si imaat site po 10 linii

        broj\_nitki = redovi / 10;

    }

    pthread\_t threads[broj\_nitki]; //ja cuva ID na sekoja nitka

    int indeks\_nitka[broj\_nitki]; //preku ova ja prakam nitkata na funkcijata

    for(int i=0; i<broj\_nitki; i++){

            indeks\_nitka[i] = i;

            pthread\_create(&threads[i], NULL, prebaraj, &indeks\_nitka[i]); //kreira nova nitka, sekoe prebaraj dobiva razlicen indeks

            //so &indeks\_nitka se praka na funkcijata prebaraj

    }

    for(int i=0; i<broj\_nitki; i++){

            pthread\_join(threads[i], NULL); //sekoja nitka cekam da zavrsi

    }

    int vkupno\_golemi =0;

    int vkupno\_mali = 0;

    for(int i=0; i<broj\_nitki; i++){

        vkupno\_golemi += broj\_golemi\_bukvi[i];

        vkupno\_mali += broj\_mali\_bukvi[i];

    }

    printf("Vkupno golemi bukvi ima %d!", vkupno\_golemi);

    printf("Vkupno mali bukvi ima %d!", vkupno\_mali);

    return 0;

}

#include <stdio.h>

#define golemina\_na\_niza 100

int niza[golemina\_na\_niza];

int N;

int K[500];

int indeksi\_nitki[500];

void \*prebaraj(void \* arg){

    int indeks = \*(int\*)arg;

    int vrednost\_K = K[indeks]; //slucaen broj z anitkava

    for(int i=0; i < vrednost\_K; i++){

        int broj\_ran = rand() % 100;

        if(indeks <= N/2){

            niza[broj\_ran] += 1;

        }else if(indeks > N/2){

            niza[broj\_ran] -= 1;

        }

    }

return NULL;

}

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

    printf("Vnesi vrednost za N: ");

    scanf("%d", &N);

    pid\_t dete = fork();

    if(dete < 0){

        printf("Ne se aktivira nikoj od procesite!");

        exit(1);

    }

    if(dete == 0){

        for(int i=0; i<golemina\_na\_niza; i++){

            niza[i] = 0;

        }

        pthread\_t threads[N]; //ja cuva ID na sekoja nitka

        int indeks\_nitka[N]; //preku ova ja prakam nitkata na funkcijata

        for(int i=0; i<N; i++){

                K[i] = rand() % 500 + 1; //najmnogu 500

                indeks\_nitka[i] = i;

                pthread\_create(&threads[i], NULL, prebaraj, &indeks\_nitka[i]); //kreira nova nitka, sekoe prebaraj dobiva razlicen indeks

                //so &indeks\_nitka se praka na funkcijata prebaraj

        }

        for(int i=0; i<N; i++){

            pthread\_join(threads[i], NULL); //sekoja nitka cekam da zavrsi

        }

        int promeni = 0;

        for(int i=0; i<golemina\_na\_niza; i++){

            if(niza[i] != 0){

                promeni++;

            }

        }

        printf("Vrednosta ja imaat promeneto %d nitki", promeni);

        exit(0);

    }

    wait(NULL);

    return 0;

}

#BITNAAAAAA

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>     //za pid\_t

#include <sys/types.h>

#include <time.h>

#include <string.h>

#include <fcntl.h>          // za open()

#include <sys/stat.h>       // za struct stat

#define golemina\_niza 1000

#define broj\_nitki 10

#define broj\_procesi 10

int niza[golemina\_niza];

void \*prebaraj(void \*arg){

    int indeks = \*(int\*)arg; //koja nitka ja zimam

    int start = indeks \* 100;

    int end = (indeks+1) \*100;

    int brojac=0;

    for(int i=start; i<end; i++){

        if(niza[i] == 0){

            brojac++;

        }

    }

    printf("Nitkata so id %d go pronajde brojot nula %d pati", indeks, brojac);

    return NULL;

}

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

    for(int i=0; i<golemina\_niza; i++){

        niza[i] = rand() % 1000; //tuka ke imam broj od 0 do 999

    }

    int vreme1;

    int vreme2;

    srand(time(NULL));

    pid\_t dete1 = fork();

    if(dete1 < 0){

        printf("Ne raboti, ni deteto ni glavnat programa!");

    }

    clock\_t start\_vreme, end\_vreme;

    if(dete1 == 0){

        start\_vreme = clock();

        for(int i=0; i<broj\_procesi; i++){

            int brojac = 0;

            pid\_t proces = fork();

            if(proces == 0){

                int start = i\*100; //od 0 ...................900

                int end = (i+1)\*100; // do 99  ................999

                for(int j=start; j<end; j++){ //od kaj do kaj da ide niz niza

                    if(niza[j] == 0){

                        brojac++;

                    }

                }

                printf("Procesot so reden broj %d ja pronasol 0, %d pati", i+1, brojac);

                exit(0);

            }

        }

        exit(0);

    }

    for(int i=0; i<broj\_procesi; i++){

            wait(NULL);

    }

    clock\_t end\_vreme = clock();

    vreme1 = end\_vreme-start\_vreme;

    pid\_t dete2 = fork();

    if(dete2<0){

        printf("Nema da raboti nitu deteto nitu glavnata programa!");

        exit(1);

    }

    clock\_t start\_vreme = clock();

    if(dete2 == 0){

        pthread\_t threads[broj\_nitki]; //ja cuva ID na sekoja nitka

        int indeks\_nitka[broj\_nitki]; //preku ova ja prakam nitkata na funkcijata

        for(int i=0; i<broj\_nitki; i++){

            indeks\_nitka[i] = i;

            pthread\_create(&threads[i], NULL, prebaraj, &indeks\_nitka[i]); //kreira nova nitka, sekoe prebaraj dobiva razlicen indeks

            //so &indeks\_nitka se praka na funkcijata prebaraj

        }

        for(int i=0; i<broj\_nitki; i++){

            pthread\_join(threads[i], NULL); //sekoja nitka cekam da zavrsi

        }

        exit(1);

    }

    wait(NULL);

    clock\_t end\_vreme = clock();

    vreme2=end\_vreme-start\_vreme;

    if(vreme1> vreme2){

        printf("Pobrzo e prebaruvanjeto so nitki, i toa za vreme %d", vreme2);

    }else if(vreme2 > vreme1){

        printf("Pobrzo e prebaruvanjeto so procesi i toa zavrsuva za %d", vreme1);

    }else{

        printf("Izvrsuvanjeto na dvata nacina e za isto vreme i toa iznesuva vreme1!");

    }

    return 0;

}

//PROGRAMA 1

#include <stdio.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <pthread.h>

#include <time.h>

#include <string.h>

#include <fcntl.h>          // za open()

#include <sys/stat.h>       // za struct stat

#define KEY 1234

#define memorija 1024

char \*shared\_memory;

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

    if(argc != 2){

        printf("Nema dovolno argumenti na komandna linija!");

        exit(1);

    }

    char \*naredba = argv[1];

    //SHARED MEMORIJA

    int shmid = shmget(KEY, memorija, 0666 | IPC\_CREAT);

    if(shmid >= 0){

        int \*shared\_memory = (int \*)shmat(shmid, NULL, 0); //shared mem attach, shmat vraka void i zatoa jas imam (int \*)

        if(shared\_memory == (void \*) -1){

            perror("Neuspesno povrzuvanje na shared memory!");

            exit(1);

        }

        strcpy(shared\_memory, naredba);

        printf("Naredbata e zapisana vo shared memory!");

        shmdt(shared\_memory);

    }else{

        //MAPIRANA MEMORIJA

        int fd = open("dat.txt", O\_RDWR | PROT\_WRITE); //deka zapisuvam mora vaka

        if(fd == -1){

            perror("Ne moze da se otvori datotekata!");

            exit(1);

        }

        int golemina = lseek(fd, 0, SEEK\_END); //odi do kraj

        lseek(fd, 0, SEEK\_SET); //vraka na pocetok

        if (golemina == 0){

            printf("Datotekata e prazna!");

            exit(1);

        }

        char \*mapirani\_podatoci = mmap(NULL, golemina, PROT\_READ, MAP\_SHARED,fd, 0); //void \*mmap(void \*addr, size\_t length, int prot, int flags, int fd, off\_t offset);

        if(mapirani\_podatoci == MAP\_FAILED){

            perror("Neuspesno mapiranje na datotekata");

            exit(1);

        }

        if(mapirani\_podatoci == NULL){

            printf("Greska: Ne mozat da se prikacat vo adresen prostor!");

            exit(1);

        }

        strcpy(mapirani\_podatoci, naredba);

        printf("Naredbata e zapisana vo mmap memorija!");

        munmap(mapirani\_podatoci, memorija);

        close(fd);

        return 0;

    }

}

//PROGRAMA 2

#include <stdio.h>

#define KEY 1234

#define memorija 1024

#define max\_procesi 200

#define max\_pati 6

char \*shared\_memory;

char \*mmap\_memory;

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

    if(argc != 2){

        printf("Nema dovolno argumenti na komandna linija!");

        exit(1);

    }

    char \*naredba = argv[1];

    int znamence = 0;

    //SHARED MEMORIJA

    int shmid = shmget(KEY, memorija, 0666 | IPC\_CREAT);

    if(shmid >= 0){

        shared\_memory = (char \*)shmat(shmid, NULL, 0); //shared mem attach, shmat vraka void i zatoa jas imam (int \*)

        if(shared\_memory == (void \*) -1){

            perror("Neuspesno povrzuvanje na shared memory!");

            exit(1);

        }

    }else{

        //MAPIRANA MEMORIJA

        int fd = open("dat.txt", O\_RDONLY);

        if(fd == -1){

            perror("Ne moze da se otvori datotekata!");

            exit(1);

        }

        int golemina = lseek(fd, 0, SEEK\_END); //odi do kraj

        lseek(fd, 0, SEEK\_SET); //vraka na pocetok

        if (golemina == 0){

            printf("Datotekata e prazna!");

            exit(1);

        }

        char \*mapirani\_podatoci = mmap(NULL, golemina, PROT\_READ, MAP\_SHARED,fd, 0); //void \*mmap(void \*addr, size\_t length, int prot, int flags, int fd, off\_t offset);

        if(mapirani\_podatoci == MAP\_FAILED){

            perror("Neuspesno mapiranje na datotekata");

            exit(1);

        }

        if(mapirani\_podatoci == NULL){

            printf("Greska: Ne mozat da se prikacat vo adresen prostor!");

            exit(1);

        }

        for(int p=0; p<max\_procesi; p++){

            pid\_t pid = fork();

            if(pid == 0){

                for(int  i=0; i<max\_pati; i++){

                    if(shared\_memory && strlen(shared\_memory) > 0){

                        strcpy(naredba, shared\_memory);

                        znamence = 1;

                        break;

                    }else if(mapirani\_podatoci && strlen(mapirani\_podatoci) > 0){

                        strcpy(naredba, mapirani\_podatoci);

                        znamence=1;

                        break;

                    }

                    sleep(10);

                }

                if(znamence != 1){

                    printf("Ne e zapisano uste!");

                    if(shared\_memory){

                        shmdt(shared\_memory);

                    }

                    if(mapirani\_podatoci){

                        munmap(mapirani\_podatoci, memorija);

                    }

                    if(fd >= 0 ){

                        close(fd);

                    }

                    exit(1);

                }

                printf("Procesot ja izvrsuva naredbata!!!!!");

                char \*args[10];

                execvp(args[0], naredba);

                perror("Neuspesno izvrsuvanje");

                if(shared\_memory){

                        shmdt(shared\_memory);

                    }

                    if(mapirani\_podatoci){

                        munmap(mapirani\_podatoci, memorija);

                    }

                    if(fd >= 0 ){

                        close(fd);

                    }

            }

        }

        return 0;

    }

}