

#define Nitki 10

#define MIN\_GRANICA -10

#define MAX\_GRANICA 2500

#define BROEVI\_GENERIRANI 20

int \*mapped\_data;

int niza\_za\_broevi[200];

int niza\_Za\_kolku\_pati[200];

int N;

int brojac[200];

void \*prebaraj(void \*arg){//id na nitkata mi e arg

    int pomosna\_niza[20];

    int id = \*(int \*)arg;

        for(int j = id\*BROEVI\_GENERIRANI; j < id\*BROEVI\_GENERIRANI + BROEVI\_GENERIRANI; j++){

            int random\_broj = (rand() % (MAX\_GRANICA - MIN\_GRANICA + 1)) + MIN\_GRANICA;  //0-2510 + MIN\_GRANICA

            for(int k =0; k<N; k++){

                if(mapped\_data[k] == random\_broj){

                    pomosna\_niza[j] = random\_broj;

                    brojac[random\_broj]++;

                    break;

                }

            }

        }

        for(int j=0; j<200; j++){

            if(brojac[j] > 0){

                int najdeno=0;

                for(int k=0; k<200; k++){

                    if(niza\_za\_broevi[k] == pomosna\_niza[j]){

                        niza\_Za\_kolku\_pati[k] += brojac[pomosna\_niza[j]];

                        najdeno=1;

                        break;

                    }

                }

                if(!najdeno){

                        if(niza\_za\_broevi[k] == 0){

                            niza\_za\_broevi[k] = pomosna\_niza[j];

                            niza\_Za\_kolku\_pati[k] = brojac[pomosna\_niza[j]];

                            break;

                        }

                }

            }

        }

}

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

    if(argc < 2){ //ime na datoteka + 1

        printf("Nevaliden broj na argumenti");

        return EXIT\_FAILURE;

    }

    printf("Vnesi broj na elementi: ");

    scanf("%d", &N);

    int fd = open(argv[1], O\_RDONLY); // Отвори датотека во режим за читање

    if (fd == -1) {

        perror("Ne mozam da ja otvoram datotekata");

        return 1;

    }

    int golemina\_na\_datoteka = lseek(fd, 0, SEEK\_END); //ja zimam goleminata na datotekata (kon datotekata, pomestuvanje=0, od kade da pocne pomestuvanjeto)

    if(golemina\_na\_datoteka == -1){

        perror("lseek failed");

        close(fd);

        return -1;

    }

    lseek(fd,0,SEEK\_SET);

    mapped\_data = mmap(NULL, golemina\_na\_datoteka, PROT\_READ, MAP\_PRIVATE, fd, 0);

    if (mapped\_data == MAP\_FAILED) {

        perror("Ne mozam da ja mapiram datotekata");

        close(fd);

        return 1;

    }

    pthread\_t threads[10]; //inicijalizacija

    int arg[Nitki]; //argument na prebaraj

    for(int i=0; i<Nitki; i++){

        pthread\_create(&threads[i], NULL, prebaraj, (void \*)&arg[i]);//kreiram nitka,    ,    ,  id na nitka

    }

    //site nitki da zavrsat pred da prodolzam ponatamu

    for(int i=0; i<Nitki; i++){

        pthread\_join(threads[i], NULL);

    }

    pid\_t dete = fork(); //kreiram

    if(dete < 0){

        perror("FORK FAILED");

    }

    if(dete == 0){

        printf("Najdeni broevi i niven broj na pojavuvanja: ");

        for(int i=0; i<200; i++){

          printf("Brojot %d e pronajjden %d pati!", niza\_za\_broevi[i], niza\_Za\_kolku\_pati[i]);

        }

    }

    else{

        wait(NULL);

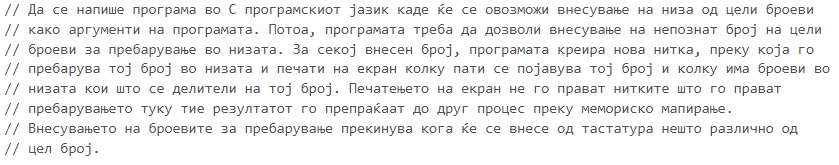
    }

    munmap(mapped\_data, golemina\_na\_datoteka);

    close(fd);

    return 0;

}



int niza\_broevi[100];

int \*mapped\_data; //ja zimam nizata od broevi

int dolzina\_na\_niza;

void \*prebaraj(void \*arg){

    int broj\_za\_prebaruvanje = \*(int \*)arg;

    int brojac\_za\_pojavuvanje=0;

    int brojac\_za\_deliteli=0;

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

        if(mapped\_data[j] == broj\_za\_prebaruvanje){

            brojac\_za\_pojavuvanje++;

        }

        if(broj\_za\_prebaruvanje % mapped\_data[j] == 0){

            brojac\_za\_deliteli++:

        }

    }

   mapped\_data[0]=broj\_za\_prebaruvanje;

   mapped\_data[1] = brojac\_za\_pojavuvanje;

   mapped\_data[2] = brojac\_za\_deliteli;

}

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

    if(argc < 2){ //niza + 1

        printf("Nevaliden broj na argumenti");

        return EXIT\_FAILURE;

    }

    dolzina\_na\_niza = argc - 1;

    printf("Elementi od nizata: ");

    for(int i=1; i<argc; i++){

        mapped\_data[i-1] = (atoi[i]);

    }

    int fd=open("dat.txt",O\_RDWR | O\_CREAT |O\_TRUNCATE,0x666); //kreirame

        // datoteka koja kje ni bide dostapna preku mmap

        if(fd==-1){

         printf("Ne uspea fd");

         return EXIT\_FAILURE;

        }

    mapped\_data=mmap(NULL,4000,PROT\_READ,|PROT\_WRITE,MAP\_SHARED,fd,0);

     if(mapped\_data==MAP\_FAILED){

     printf("Ne uspea mmap");

     return -1;

    }

while(true){

    int broj;

    printf("Vnesi element sto sakas za prebaruvanje vo nizata: ");

    scanf("%d", broj);

    pthread\_t threads;

    pthread\_create(&threads, NULL, prebaraj, &broj);//mesto indeks, brojot go vnesuvam

    pthread\_join(threads, NULL);

    printf("Brojot %d e najden %d pati i ima %d deliteli", mapped\_data[0], mapped\_data[1], mapped\_data[2]);

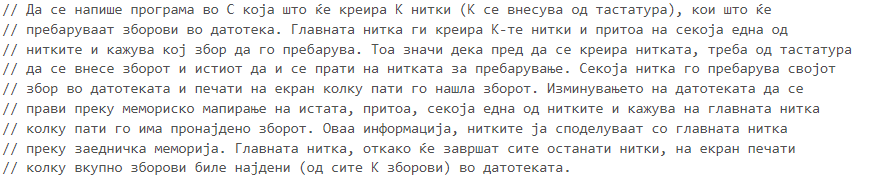
    munmap(mapped\_data);

    close(fd);

    return 0;

}

}



#define MAX\_DOLZINA\_ZBOR 100

#define MAX\_NITKI

int K;

char \* mapped\_data;

char niza\_zborovi[MAX\_NITKI][MAX\_DOLZINA\_ZBOR]; //max zborovi so dolzina 100

int \*rezultat;

void \*prebaraj(void \*arg){

    int indeks = \*(int \*)arg;

    char \*zbor = niza\_zborovi[indeks];

    int brojac;

    char \*pokazuvac = mapped\_data;

    while(pokazuvac = strstr(pokazuvac, zbor) != NULL){

        brojac++;

        pokazuvac+=strlen(zbor);

    }

    rezultat[indeks] = brojac;

}

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

{

    printf("Vnesi broj za K: ");

    scanf("%d", &K); //broj na nitki - brojot na zborovite

    for(int i=0; i<K; i++){

        printf("Vnesi zbor za nitka: ");

        scanf("%d", niza\_zborovi[i]);

    }

    int fd = open(argv[1], O\_RDONLY); // Отвори датотека во режим за читање

    if (fd == -1) {

        perror("Ne mozam da ja otvoram datotekata");

        return 1;

    }

    int golemina\_na\_datoteka = lseek(fd, 0, SEEK\_END); //ja zimam goleminata na datotekata (kon datotekata, pomestuvanje=0, od kade da pocne pomestuvanjeto)

    if(golemina\_na\_datoteka == -1){

        perror("lseek failed");

        close(fd);

        return -1;

    }

    lseek(fd,0,SEEK\_SET);

    mapped\_data = mmap(NULL, golemina\_na\_datoteka, PROT\_READ, MAP\_PRIVATE, fd, 0); //cel tekst od datoteka kako niza od karakteri

    if (mapped\_data == MAP\_FAILED) {

        perror("Ne mozam da ja mapiram datotekata");

        close(fd);

        return 1;

    }

    pthread\_t threads[MAX\_NITKI];

    int nitka\_id[MAX\_NITKI];

    for(int i=0; i<K; i++){

        nitka\_id[i] = i;

        pthread\_create(&threads[i], NULL, prebaraj, &nitka\_id[i]);

    }

    for (int i = 0; i < K; i++) {

        pthread\_join(threads[i], NULL); // Чекај нитките

    }

    int vkupno\_najdeni = 0;

    for (int i = 0; i < K; i++) {

        vkupno\_najdeni += rezultat[i];

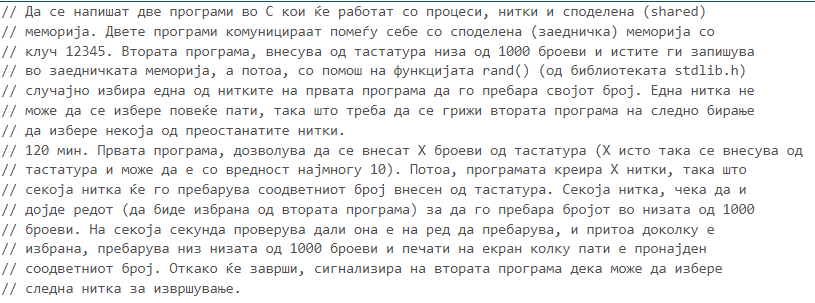
    }

    printf("Vkupno najdeni zborovi: %d\n", vkupno\_najdeni); // Испечати вкупно најдени зборови

    munmap(mapped\_data, golemina\_na\_datoteka);

    close(fd);

}



//PROGRAMA 1

#define MAX\_VREDNOST 10

#define SHM\_KEY 1234

#define GOLEMINA\_NIZA 1000

int X;

int \*mapped\_data; //pokazuvac kon memorijata

int broj\_za\_prebaruvanje[MAX\_VREDNOST];

int nitki\_id[MAX\_VREDNOST];

int \*prebaraj(void \*arg){

    int indeks = \*(int \*)arg;   //go zimam indeksot na nitkata

    int broj = broj\_za\_prebaruvanje[indeks]; //zemi go brojot

    while(1){ //celo vreme raboti mesto true -> 1

        sleep(1); //ceka sekunda pred da vidi dali e izbrana

        if(mapped\_data[0] != indeks){ //ako ovaa nitka ne e izbrana prodolzi so cekanje

            continue;

        }else if(mapped\_data[0] == indeks){//izbrana e nitkata so nekoj indeks

            int brojac=0;

            for(int i=2; i<GOLEMINA\_NIZA; i++){  //mapped\_data[0] = id na izbrana nitka, mapped\_data[1] go cuva brojot na nitki, mapped\_data[2] do mapped\_data[1000] site drugi broevi

                if(mapped\_data[i]==broj){

                    brojac++;

                }

            }

            printf("Nitkata %d, go najde brojot %d, %d pati", mapped\_data[0], broj, brojac);

            mapped\_data[0] = -1; //deka moze da se bira druga datoteka

        }

    }

}

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

    int shmid=shmget(1234,1001,IPC\_CREAT | 0x666);

    mapped\_data=(int \*) shmat(shmid,NULL,0);

    mapped\_data[0]=-1; //niedna nitka ne e zemena

    printf("Vnesi vrednost za X - vrednosti: ");

    scanf("%d", &X);

    if(X>MAX\_VREDNOST){

        X=MAX\_VREDNOST;

    }

    mapped\_data[1]=X;

    pthread\_t threads[MAX\_VREDNOST];

    int nitka\_id[MAX\_VREDNOST];

    for(int i=0; i<X; i++){

        nitka\_id[i] = i;

        pthread\_create(&threads[i], NULL, prebaraj, &nitka\_id[i]);

    }

    for (int i = 0; i < X; i++) {

        pthread\_join(threads[i], NULL); // Чекај нитките

    }

    shmdt(mapped\_data);

    shmet(shmid,IPC\_RMID,0);

    return 0;

}

//PROGRAMA 2

//generira slucajni broevi 1000 i gi zapisuva vo spodelenata memorija

//pritoa izbira slucajna nitka za izvrsuvanje

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <sys/wait.h>

#define MAX\_VREDNOST 10

#define SHM\_KEY 1234

#define GOLEMINA\_NIZA 1000

int \*mapped\_data;

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

    int shmid=shmget(1234,1001,IPC\_CREAT|0x666);

    if(shmid==-1){

        printf("Nepravilno shmget");

        return 0;

    }

    mapped\_data=(int \*) shmat(shmid,NULL,0);

    srand(time(NULL));

    printf("Generiram 1000 broevi vo shared mem");

    for(int i=2; i<GOLEMINA\_NIZA; i++){

        mapped\_data[i]=rand()%1000;

    }

    int broj\_nitki = mapped\_data[1];

    int izbrani\_nitki[broj\_nitki];

    for(int i=2; i<broj\_nitki; i++){

        while (mapped\_data[0] != -1) //ceka da zavrsi

        {

            sleep(1);

        }

        int random\_indeks=rand()%broj\_nitki;

        while(izbrani\_nitki[random\_indeks] == 1){ //prethodno bila, zemi nekoja druga sega

            random\_indeks=rand()%broj\_nitki;

        }

        izbrani\_nitki[random\_indeks] = 1;

        mapped\_data[0]=random\_indeks;

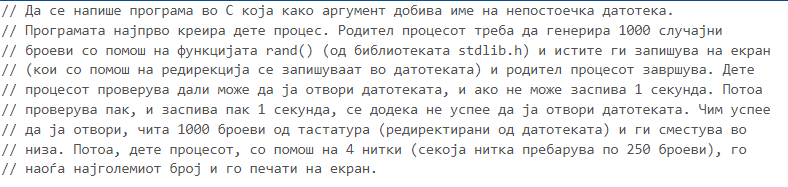
        printf("Ja izbrav nitkata so id: %d", random\_indeks);

    }

    shmdt(mapped\_data);

    return 0;

}



#define NITKI 4;

int niza\_broevi[1000];

int max\_broj=0;

void \*prebaraj(void \*arg){

    int indeks = \*(int \*)arg;

    int start,end;

    if(indeks == 0){

        start=0;

        end=249;

    }else if(indeks == 1){

        start=250;

        end=499;

    }else if(indeks == 2){

        start =500;

        end = 749;

    }else if(indeks == 3){

        start = 750;

        end = 999;

    }

    int lokalen\_maksimum=niza\_broevi[start];

    for(int i=start; i<end; i++){

        if(niza\_broevi[i] > lokalen\_maksimum){

            lokalen\_maksimum = niza\_broevi[i];

        }

    }

    if(lokalen\_maksimum>max\_broj){

        max\_broj = lokalen\_maksimum;

    }

}

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

    if(argc < 2){ //ime na datoteka + 1

        printf("Nevaliden broj na argumenti");

        return EXIT\_FAILURE;

    }

    ime\_datoteka = argv[1];

    pid\_t dete = fork(); //kreiram

    if(dete < 0){

        perror("FORK FAILED");

    }

    if(dete == 0){

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

    while(fd == -1) {

        perror("Ne mozam da ja otvoram datotekata");

        sleep(1);

    }

    while (i<1000 && read(fd,&niza\_broevi[i],sizeof(int))>0){

        i++;

    }

    pthread\_t threads[4]; //inicijalizacija

    int nitki\_indeks[4]; //argument na prebaraj

    for(int i=0; i<4; i++){

        nitki\_indeks[i] = i;

        pthread\_create(&threads[i], NULL, prebaraj, &nitki\_indeks[i]);//kreiram nitka,    ,    ,  id na nitka

    }

    //site nitki da zavrsat pred da prodolzam ponatamu

    for(int i=0; i<4; i++){

        pthread\_join(threads[i], NULL);

    }

    printf("Najgolem broj e %d", max\_broj);

    }

    else{//vo roditel sum

        int fd = open(argv[1], O\_RDONLY); // Отвори датотека во режим за читање

        if (fd == -1) {

            perror("Ne mozam da ja otvoram datotekata");

            return 1;

        }

        srand(time(NULL));

        for(int i=0; i<1000; i++){

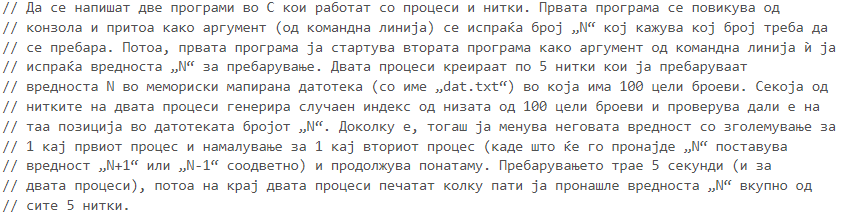
            printf("%d", rand()%1000);

        }

        return 0;

    }

}



//PROGRAMA 1

#define KOLKU\_BROEVI 100

int N;

int \*mapped\_data;

void \*prebaruvaj(void \*arg){

    int id = \*(int \*)arg;

    int brojac=0;

    for(int i=0; i<5; i++){

    int random\_indeks = rand() % 100;

   // pthread\_mutex\_lock(&mutex);

    if(mapped\_data[random\_indeks] == N){

        mapped\_data[random\_indeks] = N+1;

        brojac++;

    }

   // pthread\_mutex\_unlock(&mutex);

    }

    printf("Nitkata go najde brojot %d, %d pati! ", N, brojac);

}

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

    N = atoi(argv[1]); //koj broj go prebaruvam

    if(argc < 2){ //ime na datoteka + 1

        printf("Nevaliden broj na argumenti");

        return EXIT\_FAILURE;

    }

    pid\_t dete = fork();

    if(dete < 0){

        perror("Ne moze da se kreira proces");

        return EXIT\_FAILURE;

    }

    if(dete == 0){

        execl("./programa2", "programa2", argv[1], NULL);  // Startuvanje na programata 2

        perror("execl failed");

        return EXIT\_FAILURE;

    }

    int fd = open("dat.txt", O\_RDONLY); // Отвори датотека во режим за читање

    if (fd == -1) {

        perror("Ne mozam da ja otvoram datotekata");

        return 1;

    }

    int golemina\_na\_datoteka = lseek(fd, 0, SEEK\_END); //ja zimam goleminata na datotekata (kon datotekata, pomestuvanje=0, od kade da pocne pomestuvanjeto)

    if(golemina\_na\_datoteka == -1){

        perror("lseek failed");

        close(fd);

        return -1;

    }

    lseek(fd,0,SEEK\_SET);

    mapped\_data = mmap(NULL, golemina\_na\_datoteka, PROT\_READ, MAP\_PRIVATE, fd, 0);

    if (mapped\_data == MAP\_FAILED) {

        perror("Ne mozam da ja mapiram datotekata");

        close(fd);

        return 1;

    }

    pthread\_t threads[5]; //inicijalizacija

    int indeks\_nitka[5]; //argument na prebaraj

    for(int i=0; i<5; i++){

        indeks\_nitka[i] = i;

        pthread\_create(&threads[i], NULL, prebaraj, &indeks\_nitka[i]);//kreiram nitka,    ,    ,  id na nitka

    }

    //site nitki da zavrsat pred da prodolzam ponatamu

    for(int i=0; i<5; i++){

        pthread\_join(threads[i], NULL);

    }

    munmap(mapped\_data, golemina\_na\_datoteka);

    close(fd);

    wait(NULL); //cekanje za 2 proces

    return 0;

}

//PROGRAMA 2

#define KOLKU\_BROEVI 100

int \*mapped\_data;

void \*prebaruvaj(void \*arg){

    int id = \*(int \*)arg;

    int brojac=0;

    for(int i=0; i<5; i++){

    int random\_indeks = rand() % 100;

    if(mapped\_data[random\_indeks] == N){

        mapped\_data[random\_indeks] = N-1;

        brojac++;

    }

    }

    printf("Nitkata go najde brojot %d, %d pati! ", N, brojac);

}

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

    if(argc < 2) {

        printf("Nevaliden broj na argumenti\n");

        return EXIT\_FAILURE;

    }

    int N = atoi(argv[1]);

    int fd = open("dat.txt", O\_RDWR);  // Отвори ја датотеката

    if (fd == -1) {

        perror("Ne mozam da ja otvoram datotekata");

        return 1;

    }

    int golemina\_na\_datoteka = lseek(fd, 0, SEEK\_END);

    lseek(fd, 0, SEEK\_SET);

    mapped\_data = mmap(NULL, golemina\_na\_datoteka, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);

    if (mapped\_data == MAP\_FAILED) {

        perror("Ne mozam da ja mapiram datotekata");

        close(fd);

        return 1;

    }

    pthread\_t threads[5];  // 5 нитки

    int indeks\_nitka[5]; //argument na prebaraj

    for(int i=0; i<5; i++){

        indeks\_nitka[i] = i;

        pthread\_create(&threads[i], NULL, prebaraj, &indeks\_nitka[i]);//kreiram nitka,    ,    ,  id na nitka

    }

    for(int i = 0; i < 5; i++) {

        pthread\_join(threads[i], NULL);

    }

    munmap(mapped\_data, golemina\_na\_datoteka);  // Освободување на меморија

    close(fd);  // Затворање на датотеката

    return 0;

}