JEGYZŐKÖNYV

Operációs rendszerek BSc

2021 tavasz féléves feladat

Készítette: **Nyeste Ágoston**

Neptunkód: **WH85ZH**

Feladat leírása:

Írjon egy programot, ami egy másodfokú egyenlet megoldóképletét reprezentálja message queue IPC mechanizmus segítségével.
A művelethez szükséges adatokat egy bemeneti fájlból olvassa be, majd az adatokat és az eredményt adja vissza egy kimeneti fájlba.

Bemeneti fájl:

i (A megoldani kívánt egyenletek száma)
a b c

Kimeneti fájl:

a b c x y (Az a, b, c jelzi a bemeneti adatokat, az x, y pedig a kimeneti adatokat)
A feladat elkészítésének lépései:

main.c

1. Szükséges header állományok deklarálása: sys/ipc.h, sys/msg.h.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <sys/ipc.h>
#include <sys/msg.h>
```

2. Létrehozom a struktúrát a message queue-hoz.

```c
//Struktura a message queue-hoz
struct mesg_buffer {
    long mesg_type;
    double a;
    double b;
    double c;
    double root1;
    double root2;
} message;
```

3. Létrehoztam két tömböt és egy változót, a 3 elemű tömbben tárolom el az a, b, c értékeket. A 2 elemű tömbben a két megoldást. A változó tárolja el, hogy hány egyenlet van a txt-ben.

```c
double input[3]; // a, b, c
double output[2];// x, y
int egyenletek;
```

4. Megnyitom a bemenet .txt fájlt olvasásra.

```c
FILE *fp = fopen("bemenet.txt","r"); //File pointer a bemeneti file-hoz
if (fp < 0){
     perror("Hiba van a file-al");//Hiba kezeles
     exit(-1);
}
```

5. A fáljból kiolvasom az első sort és eltárolom a változóban.

```
fscanf(fp,"%d",&egyenletek);//Beolvassuk az egyenletek szamat
printf("A fileban talalhato egyenletek szama: %d \n",egyenletek);
```

6. A fáljból eltárolom az adatokat az input tömbben, és kiírom az adatokat. Meghívom a masodfokumegoldó nevő függvényt.

```
for(int i = 0; i < egyenletek; i++){
    for(int k = 0; k < 3; k++){
        fscanf(fp,"%lf",&input[k]);//Beolvassuk az adatokat
    }
    printf("%d. egyenlet: a = %.2lf, b=%.2lf, c=%.2lf\n",i + 1,input[0],input[1],input[2]);
    masodfokumegoldo(input);//Masodfoku egyenlet megoldo fuggveny
}
```

7. Létrehozom a változókat és a tömb elemeit elmentem. Kiszámolom a diszkriminánst.

```
double a, b, c, discriminant, root1, root2, realPart, imagPart;

a = input[0];
b = input[1];
c = input[2];

discriminant = b * b - 4 * a * c;
```

8. A diszkrimináns értéke szerint meghívom a kuldes függvényt vagy kiírom, hogy nincs megoldás.

```
if (discriminant > 0) {
    root1 = (-b + sqrt(discriminant)) / (2 * a);
    root2 = (-b - sqrt(discriminant)) / (2 * a);
    kuldes(a,b,c,root1,root2);
}

else if (discriminant == 0) {
    root1 = root2 = -b / (2 * a);
    kuldes(a,b,c,root1,root2);
}
else{
    printf("Nincs megoldas!\n");
}
```

9. A kuldes függvényben létrehozok egy message queuet. Az ftok generál egy saját kulcsot, az msgid csinál egy message queuet. Az üzenet típusát beállítom 1-re., a struktúra értékeit pedig a paraméterben kapott adatokra állítom. A msgsnd elküldi az üzenetet.

```
// message queue
key_t key;//kulcs az uzenet sorhoz
int msgid;
key = ftok("progfile", 65);// ftok generál egy saját kulcsot

msgid = msgget(key, 0666 | IPC_CREAT);// msgid csinál egy message queuet

message.mesg_type = 1;

message.a = a;
message.b = b;
message.c = c;
message.root1 = root1;
message.root2 = root2;

msgsnd(msgid, &message, sizeof(message), 0);//msgnd elkuldi az uzenetet
```

reciever.c

10. A reciever.c-ben is deklarálom a szükséges header állományokat.

11. Létrehozom a struktúrát a message queuehoz.

```
// struktura
struct mesg_buffer {
    long mesg_type;
    double a;
    double b;
    double c;
    double root1;
    double root2;
} message;
```

12. Az main.c-hez hasonlóan létrehozok egy kulcsot, ftok generál egy saját kulcsot, msgid létrehozza a message queuet. Ellenőrzöm ha az msgid = -1, akkor hiba történt.

```
    key_t key;//kulcs az uzenet sorhoz
    int msgid;
    key = ftok("progfile", 65);// ftok generál egy saját kulcsot

    msgid = msgget(key, 0666 | IPC_CREAT);// msgid csinál egy message queuet

    if(msgid == -1){
        perror("msgget");
        exit(1);
    }
```

13. Egy for ciklusban a msgrcv segítségével fogadom az üzeneteket, ezt is ellenőrzöm ha -1 akkor hiba történt. Meghívom a fileKiiras függvényt.

```
for(;;){

    if(msgrcv(msgid, &message, sizeof(message), 1, 0) == -1){//msgrcv fogadja az uzenetet
        perror("msgrcv");
        exit(1);
    }

    fileKiiras(message.a,message.b,message.c,message.root1,message.root2);// kiiras fileba

}
```

14. Létrehozom az eredmeny.txt fájlt, ellenőrzöm, hogy van e hiba a fájlal, kiírom bele az adatokat.

```
void fileKiiras(double a, double b, double c, double root1, double root2){
    FILE *file_to_write = fopen("eredmeny.txt","a");

    if (file_to_write < 0){
        perror("Hiba a file-al");
        exit(-1);
    }

    printf("Sikeres file kiiras!\n");

    fprintf(file_to_write,"Egyenlet: a = %.2lf, b=%.2lf, c=%.2lf = x1=%lf, x2=%lf\n",message.a,message.b,message.c,message.root1,message.root2);

    fclose(file_to_write);
}
```

15. Végül kitörlöm a message queuet. Az msgctl-el.

```
msgctl(msgid, IPC_RMID, NULL);// a message queue törlése
```

Console:

```
agoston@agostons-ubuntu:~/Desktop/masodfoku$ ./main
A fileban talalhato egyenletek szama: 5
1. egyenlet: a = 8.00, b=6.00, c=-2.00
Sikeres kuldes!
2. egyenlet: a = 2.00, b=-5.00, c=3.40
Nincs megoldas!
3. egyenlet: a = 6.00, b=2.00, c=11.20
Nincs megoldas!
4. egyenlet: a = -13.00, b=12.00, c=45.00
Sikeres kuldes!
5. egyenlet: a = 8.00, b=7.00, c=-2.00
Sikeres kuldes!
agoston@agostons-ubuntu:~/Desktop/masodfoku$ ./reciever
Sikeres file kiiras!
Sikeres file kiiras!
Sikeres file kiiras!
```

eredmenyek.txt:

```
1 Egyenlet: a = 8.00, b=6.00, c=-2.00 = x1=0.250000, x2=-1.000000
2 Egyenlet: a = -13.00, b=12.00, c=45.00 = x1=-1.455375, x2=2.378452
3 Egyenlet: a = 8.00, b=7.00, c=-2.00 = x1=0.226884, x2=-1.101884
```