



ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ  
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΜΑΤΙΚΗΣ

# Κοκκινογέννης Βασίλειος

2<sup>η</sup> Εργασία στο μάθημα **Λειτουργικά Συστήματα**

Περιεχόμενα	
<b>Άσκηση 2</b>	<b>3</b>
Κώδικας	3
Τρόπος Εκτέλεσης	3
Ενδεικτικές εκτελέσεις (screenshots):	7
Βασική διεργασία η οποία ελέγχει τη δημιουργία των κατάλληλων διεργασιών και τον έλεγχο του προγράμματος	7
Screenshots	7
Συγχρονισμός των 2 διεργασιών εγγραφής ανάγνωσης	8
Screenshots	8
Διαχείριση σημάτων	8
Screenshots	8
Δημιουργία νημάτων και πέρασμα παραμέτρων	9
Screenshots	9
Συγχρονισμός νημάτων και σωστή διαδικασία μέτρησης αποτελέσματος	9
Screenshots	9
Ομαλή εκτέλεση προγράμματος, error handling, τεκμηρίωση	10
Screenshots	10
Γενικά Σχόλια/Παρατηρήσεις	10
Με δυσκόλεψε / δεν υλοποίησα	11
<b>Συνοπτικός Πίνακας</b>	<b>12</b>

# Άσκηση 2

## Κώδικας

Ο κώδικας της 2ης εργασίας που δημιουργήθηκε μαζί με τα σχόλια είναι:

```
#include <stdio.h>
#include <string.h>
#include <pthread.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <semaphore.h>
#include <signal.h>
#define RANDOM_STRING_LENGTH 2000 // Dhlonei to megethos tou tixaiau string
#define NTHREADS 4 // Dhlonei ton arithmo ton threads

int charfrequency[26] = {0}; // Pinakas int 26 theseon gia to oliko frequency arxikopoieitai me 0
pthread_mutex_t mymutex=PTHREAD_MUTEX_INITIALIZER; // Arxikopoihsh tou mutex
int myOffset = 0; // Metavlhth pou dhlonei to offset, einai global oste na mporei na fanei se ola ta threads

void * thread_func(void * args) // Leitourgia pou ektelei to kathe thread
{
    char partial[500]; // Pinakas char 500 theseon gia thn anagnosh apo arxeio
    int *argptr = args; // Pointer argptr pou deixnei sto argument pou tou pername otan to kaloume (dieuthinsi tou filedescriptor)

    int tempcharfrequency[26] = {0}; // Pinakas int 26 theseon gia to topiko frequency arxikopoieitai me 0

    pthread_mutex_lock(&mymutex); // Kleidonoume ton mutex gia na mhn exoume pollaplh tautoxronh prosvash apo ta alla threads (krishmh perioxh)
    lseek(*argptr, myOffset, SEEK_SET); // Apo to filedescriptor pou tou perasame san argument phgainei sthn arxh tou arxeiou kai metakinhei ton dhkth //kata offset theseis makria

    for (size_t i = 0; i < 500; i++) { // Arxikopoihsh tou pinaka partial me ""
        partial[i] = "";
    }

    read(*argptr,partial,strlen(partial)); // Diavazei apo to filedescriptor pou tou perasame san argument ta stoixeia tou arxeiou kai ta antigrafει // ston pinaka partial, mexri na diavasei 500 bytes

    for (int i = 0; i < 500; i++) { // Gemizei ton pinaka tempchatfrequency gia i apo 1 mexri 500 me vhma 1 me ton ekshs tropo:
```

```

    tempcharfrequency[partial[i] - 'a']++; // 0 xarakthras pou vrisketai ston pinaka partial sthn
    thesi i, afairontas tou to 'a' mporei na parei times
} // mono apo to 0 mexri to 26. Sinepos auksanoume thn
thesi tou pinaka tempcharfrequency pou antistoixei sto
// partial[i] - 'a' kata ena

// Xrhsimo gia debugging
/*
for (int i = 0; i < 26; i++) {
    printf("%c->%d ", i+49+'0', tempcharfrequency[i]);
}
printf("\n");
*/

for (int i = 0; i < 26; i++) { // Apo to 1 mexri to 26 me vhma 1
    charfrequency[i]+=tempcharfrequency[i]; // Prostheti ta topika apotelesmata tou
tempcharfrequency ston kirio pinaka charfrequency
}
myOffset += 500; // Metavaloume to offset kata 500 theseis
pthread_mutex_unlock(&mymutex); // Ksekleidonoume ton mutex afou teliosame thn epeksergia sthn
krishmh perioxh
pthread_exit (NULL); // Telionei h ektelesh tou thread
}

void sig_handler(int signum){ // Leitourgia pou kanei cleanup an dexthei shma sigint h sigterm

    printf("\nTerminating program..\n"); // Kanei print sto termatiko thn frash Terminating program..
    const char *semName = "myfilelockFork"; // Thetei to onoma tou shmaforou iso me myfilelockFork
    sem_t *sem_filelock; // Dhlonei thn kleidaria
    sem_close(sem_filelock); // Kleinei ton shmaforo (thn kleidaria)
    sem_unlink(semName); // Aposindei to onoma tou shmaforou apo ton shmaforo
    exit(0); // Kanei exit me epitixia
}

int main(){ // Leitourgia ths main

    const char *semName = "myfilelockFork"; // thetei to onoma tou shmaforou iso me myfilelockFork
    sem_t *sem_filelock; // Dhlonei thn kleidaria
    sem_filelock = sem_open(semName, O_CREAT, 0600, 1); // Anoigoume ton shmaforo me onoma semName
(myfilelockFork) kai thn parametro O_CREAT
// pou ton dhmiourgei MONO an den iparxei ,
me dikaiomata gia ton xrhsth kai timh 1

    int fd=open("data.txt", O_RDWR | O_CREAT, 0600); // Anoigoume to arxeio data.txt me parametro
readwrite kai thn parametro O_CREAT
// pou to dhmiourgei MONO an den iparxei kai
me dikaiomata gia ton xrhsth

    int pid=fork(); // Kanoume fork meta thn fd=open oste o file descriptor ths kirias diergasias
kai tou paidiou na einai koinos

```

```

if (pid!=0) { // An to pid einai tou patera

    signal(SIGINT,sig_handler); // Kaloume thn leitourgia sig_handler an dothei shma SIGINT
    signal(SIGTERM,sig_handler); // Kaloume thn leitourgia sig_handler an dothei shma SIGTERM

    //printf("hello from parent \n"); // Xrhsimo gia debugging

    sem_wait(sem_filelock); // A -> Kanei thn timh tou shmaforou ish me 0 (mpainei gia proth
fora edo, prohgooumenos h timh htan dhlomenh me 1)

    srand(getpid()); // Xrhsimo gia thn dhmiourgia tixeon xarakthron me vash to pid
    char buf[RANDOM_STRING_LENGTH + 1]; // Dhmiourgia pinaka char me onoma buf kai me 2001
theseis
    for (size_t i = 0; i < RANDOM_STRING_LENGTH; i++) { // Gia i apo 0 mexri 2000 me vhma 1
        buf[i] = "abcdefghijklmnopqrstuvwxyz"[rand() %26]; // Pairnei enan tixeo xarakthra apo
autous pou periexontai sta eisagogika kai ton topothetei sthn
    } // i thesh tou pinaka buf

    buf[RANDOM_STRING_LENGTH] = '\0'; // Sto telos tou pinaka prosthetoume to null terminator
gia na deiksoume oti telionei to string
    write(fd,buf,strlen(buf)); // Grafoume apo to fd sto arxeio data.txt ta periexomena tou
pinaka buf mexri na exoun graftei 2000 bytes

    //printf("Writing from process:%d\n", getpid()); // Xrhsimo gia debugging

    sleep(3); // Mikrh kathisterhsh gia na prolavei o xrhsths na pathsei ctrl c
    sem_post(sem_filelock); // B -> Kanei thn timh tou shmaforou ish me 1 (prohgooumenos h timh
htan 0)

    // Prepei na perimenoume to paidi gia na kanei cleanup o goneas (na kleisei to arxeio kai
ton shmaforo)
    int status; // Metavlhth pou dhlonei thn katastash pou vrisketai to paidi
    waitpid(pid,&status,0); // Perimenei mexri na teleiosei to paidi
    printf("Parent cleaning up..\n"); // Kanei print sto termatiko to parapano string
    close(fd); // Kleinei ton file descriptor
    sem_close(sem_filelock); // Kleinei ton shmaforo
    sem_unlink(semName); // Aposindeei to onoma tou shmaforou apo ton shmaforo

} else { // An to pid einai tou paidiou

    //printf("hello from child\n"); // Xrhsimo gia debugging
    sem_wait(sem_filelock); // C -> Kanei thn timh tou shmaforou ish me 0 (prohgooumenos h timh
htan 1)

    //printf("Reading from process:%d\n", getpid()); // Xrhsimo gia debugging

    int filedescriptors[4]; // Pinakas apo filedescriptors
    for (int i = 0; i < 4; i++) { // Gia i apo 0 mexri 4 me vhma 1
        filedescriptors[i] = dup(fd); // dhmiourgei tessera nea filedescriptors pou einai
antigrafa tou arxikou fd
    }

    pthread_t threads[NTHREADS]; // Dhlosh pinaka apo threads

```

```

    for (int i = 0; i < NTHREADS; i++) { // Gia i apo 0 mexri 4 me vhma 1

        pthread_create(&threads[i], NULL, thread_func, &filedescriptors[i]); // Dhmiourgei neo
thread pou tha ektelei thn leitourgia thread_func kai
                                                                    // tou pernaei os
parametro thn dieuthinsi tou i-ostou filedescriptor
    }

    for (int i = 0; i < NTHREADS; i++){ // Gia i apo 0 mexri 4 me vhma 1
        pthread_join(threads[i], NULL); // Perimenei mexri na teliosoun ola ta threads kai
gnostopoiei to charfrequency
    }
    for (int i = 0; i < 26; i++) { // Gia i apo 0 mexri 26 me vhma 1
        printf("%c->%d ", i+49+'0', charfrequency[i]); // Ektiponei sto termatiko ton xarakthra
i(+49+'0' , metatroph int se char me vash ton pinaka ascii)
    }
                                                                    // kai dipla ektiponei to poses fores
emfanistike o xarakthras autos

    printf("\n"); // Prosthetei newline (afhnei mia grammh)

    sleep(3); // Mikrh kathisterhsh gia na prolavei o xrhsths na pathsei ctrl c
    for (int i = 0; i < 4; i++) { // Gia i apo 0 mexri 4 me vhma 1
        close (filedescriptors[i]); // Kleinei tous file descriptors pou einai antigrafa tou
arxikou fd
    }
    sem_post(sem_filelock); // D -> Kanei thn timh tou shmaforou ish me 1 (prohgoumenos h timh
htan 0)
    }
}

```

## Τρόπος Εκτέλεσης

Ο κώδικας εκτελείται με τον εξής τρόπο: Έχοντας ανοίξει το τερματικό εκτελούμε τις εντολές

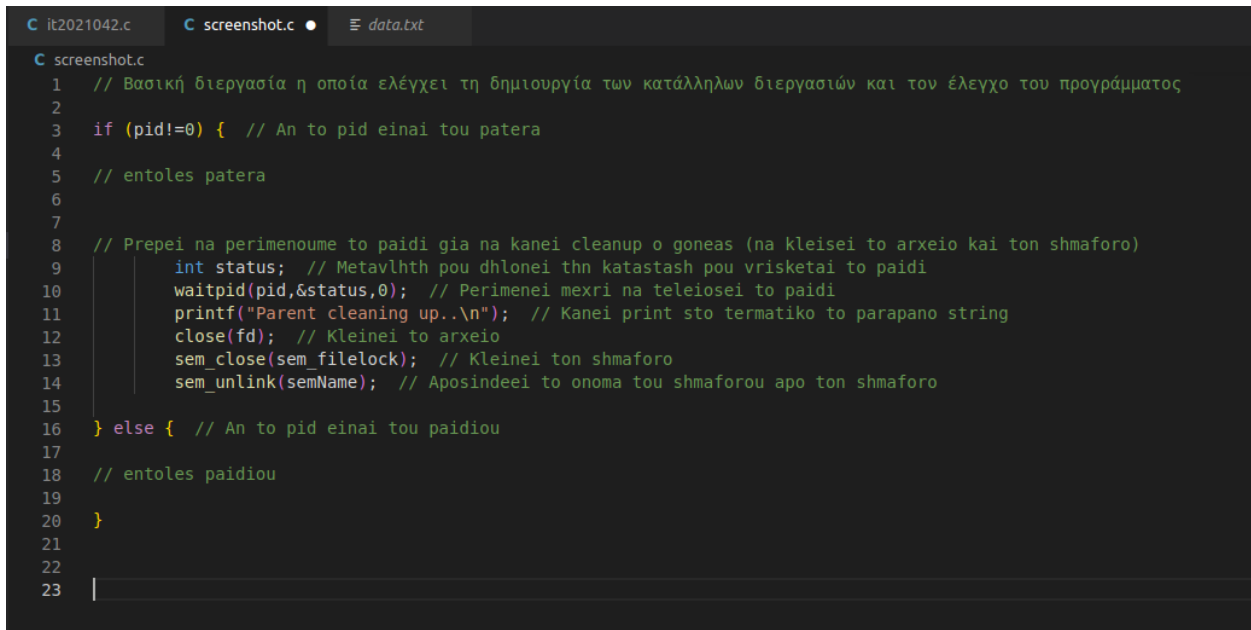
```
gcc -pthread it2021042.c
```

```
./a.out
```

## Ενδεικτικά κομμάτια κώδικα (screenshots):

*Βασική διεργασία η οποία ελέγχει τη δημιουργία των κατάλληλων διεργασιών και τον έλεγχο του προγράμματος*

### Screenshots



```
C it2021042.c  C screenshot.c •  data.txt
C screenshot.c
1  // Βασική διεργασία η οποία ελέγχει τη δημιουργία των κατάλληλων διεργασιών και τον έλεγχο του προγράμματος
2
3  if (pid!=0) { // An to pid einai tou patera
4
5  // entoles patera
6
7
8  // Prepei na perimenoume to paidi gia na kanei cleanup o goneas (na kleisei to arxeio kai ton shmaforo)
9      int status; // Metavlhth pou dhlonei thn katastash pou vrisketai to paidi
10     waitpid(pid,&status,0); // Perimenei mexri na teleiosei to paidi
11     printf("Parent cleaning up..\n"); // Kanei print sto termatiko to parapano string
12     close(fd); // Kleinei to arxeio
13     sem_close(sem_filelock); // Kleinei ton shmaforo
14     sem_unlink(semName); // Aposindei to onoma tou shmaforou apo ton shmaforo
15
16 } else { // An to pid einai tou paidiou
17
18 // entoles paidiou
19
20 }
21
22
23 |
```

## Συγχρονισμός των 2 διεργασιών εγγραφής ανάγνωσης

### Screenshots

```
C it2021042.c  C screenshot.c  data.txt
C screenshot.c
1 // Συγχρονισμός των 2 διεργασιών εγγραφής ανάγνωσης
2
3 if (pid!=0) { // An to pid einai tou patera
4
5     sem_wait(sem_filelock); // A -> Kanei thn timh tou shmaforou ish me 0 (mpainei gia proth fora edo, prohgooumenos h timh htan dhlomenh me 1)
6
7     // entoles patera
8
9     sem_post(sem_filelock); // B -> Kanei thn timh tou shmaforou ish me 1 (prohgooumenos h timh htan 0)
10
11 } else { // An to pid einai tou paidiou
12
13     sem_wait(sem_filelock); // C -> Kanei thn timh tou shmaforou ish me 0 (prohgooumenos h timh htan 1)
14
15     // entoles paidiou
16
17     sem_post(sem_filelock); // D -> Kanei thn timh tou shmaforou ish me 1 (prohgooumenos h timh htan 0)
18
19 }
20
21
22
```

## Διαχείριση σημάτων

### Screenshots

```
C it2021042.c  C screenshot.c  data.txt
C screenshot.c
1 // Διαχείριση σημάτων
2
3
4 void sig_handler(int signum){ // Leitourgia pou kanei cleanup an dexthei shma sigint h sigterm
5
6     printf("\nTerminating program..\n"); // Kanei print sto termatiko thn frash Terminating program..
7     const char *semName = "myfilelockFork"; // Thetei to onoma tou shmaforou iso me myfilelockFork
8     sem_t *sem_filelock; // Dhlonei thn kleidaria
9     sem_close(sem_filelock); // Kleinei ton shmaforo (thn kleidaria)
10    sem_unlink(semName); // Aposindeei to onoma tou shmaforou apo ton shmaforo
11    exit(0); // Kanei exit me epitixia
12 }
13
14
15
16 if (pid!=0) { // An to pid einai tou patera
17
18     signal(SIGINT,sig_handler); // Kaloume thn leitourgia sig_handler an dothei shma SIGINT
19     signal(SIGTERM,sig_handler); // Kaloume thn leitourgia sig_handler an dothei shma SIGTERM
20
21     // entoles patera
22
23 }
24
25
26
```



### Δημιουργία νημάτων και πέρασμα παραμέτρων

## Screenshots

```
C it2021042.c C screenshot.c E data.txt
```

```
C screenshot.c  
1 // Δημιουργία νημάτων και πέρασμα παραμέτρων  
2  
3  
4 void * thread_func(void * args) // Leitourgia pou ektelei to kathe thread  
5  
6  
7 if (pid!=0) { // An to pid einai tou patera  
8  
9     // entoles patera  
10  
11 } else { // An to pid einai tou paidiou  
12  
13 pthread_t threads[NTHREADS]; // Dhlosh pinaka apo threads  
14  
15 for (int i = 0; i < NTHREADS; i++) { // Gia i apo 0 mexri 4 me vhma 1  
16  
17 pthread_create(&threads[i], NULL, thread_func, &filedescriptors[i]); // Dhmiourgei neo thread pou tha ektelei thn leitourgia thread_func kai  
18 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  
19 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  
20 } // tou pernaei os parametro thn dieuthinsi tou i-ostou filedescriptor  
21  
22 for (int i = 0; i < NTHREADS; i++){ // Gia i apo 0 mexri 4 me vhma 1  
23 | pthread_join(threads[i],NULL); // Perimenei mexri na teliosoun ola ta threads kai gnostopoiei to charfrequency  
24 }  
25 }  
26  
27  
28
```

## Συγχρονισμός νημάτων και σωστή διαδικασία μέτρησης αποτελέσματος

## Screenshots

```

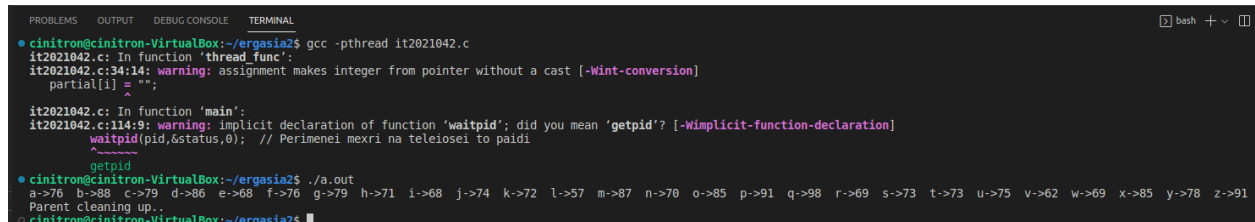
C it2021042.c C screenshot.c ≡ data.txt
C screenshot.c
1 // Συγχρονισμός νημάτων και σωστή διαδικασία μέτρησης αποτελέσματος
2
3
4 void * thread_func(void * args) // Leitourgia pou ektelei to kathe thread
5 {
6     char partial[500]; // Pinakas char 500 theseon gia thn anagnosh apo arxeio
7     int *argptr = args; // Pointer argptr pou deiknei sto argument pou tou pername otan to kaloume (dieuthinsi tou filedescriptor)
8
9     int tempcharfrequency[26] = {0}; // Pinakas int 26 theseon gia to topiko frequency arxikopoleitai me 0
10
11     pthread_mutex_lock(&mymutex); // Kleidonoume ton mutex gia na mhn exoume pollaplh tautoxronh provsash apo ta alla threads (krishmh perioxh)
12     lseek(*argptr, myOffset, SEEK_SET); // Apo to filedescriptor pou tou perasame san argument phgainai sthn arxh tou arxeiou kai metakinhei ton dhkth
13     //kata offset theseis makria
14
15     for (size_t i = 0; i < 500; i++) { // Arxikopoihsh tou pinaka partial me ""
16         partial[i] = "";
17     }
18
19     read(*argptr, partial, strlen(partial)); // Diavazei apo to filedescriptor pou tou perasame san argument ta stoixeia tou arxeiou kai ta antigrafei
20     // ston pinaka partial, mexri na diavasei 500 bytes
21
22     for (int i = 0; i < 500; i++) { // Gemizei ton pinaka tempcharfrequency gia i apo 1 mexri 500 me vhma 1 me ton ekshs tropo:
23         tempcharfrequency[partial[i] - 'a']++; // 0 xarakthras pou vrisketai ston pinaka partial sthn thesi i, afairontas tou to 'a' mporei na parei times
24     } // mono apo to 0 mexri to 26. Sinepos auksanoume thn thesi tou pinaka tempcharfrequency pou antistoixei sto
25     // partial[i] - 'a' kata ena
26
27     for (int i = 0; i < 26; i++) { // Apo to 1 mexri to 26 me vhma 1
28         charfrequency[i] += tempcharfrequency[i]; // Prostheti ta topika apotelesmata tou tempcharfrequency ston kirio pinaka charfrequency
29     }
30     myOffset += 500; // Metavaloume to offset kata 500 theseis
31     pthread_mutex_unlock(&mymutex); // Ksekleidonoume ton mutex afou teliosame thn epeksgerasia sthn krishmh perioxh
32     pthread_exit (NULL); // Telionei h ektelesh tou thread
33 }
34
35 // sthn main
36
37 for (int i = 0; i < 26; i++) { // Gia i apo 0 mexri 26 me vhma 1
38     printf("%c-%d ", i+9+'0', charfrequency[i]); // Ektiponei sto termatiko ton xarakthra i+(49+'0', metatroph int se char me vash ton pinaka ascii)
39 } // kai dipla ektiponei to poses fores emfanistike o xarakthras autos
40
41

```

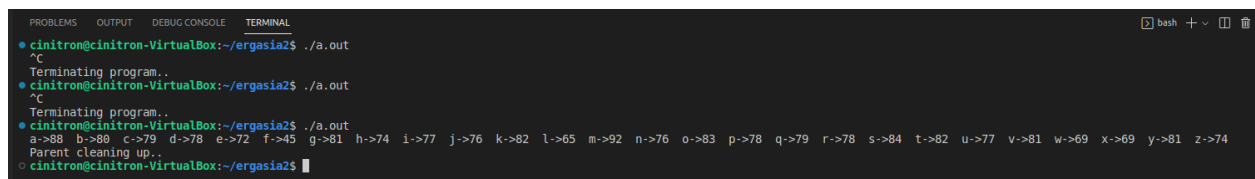
## Ενδεικτικές εκτελέσεις (screenshots):

Ομαλή εκτέλεση προγράμματος, *error handling*, τεκμηρίωση

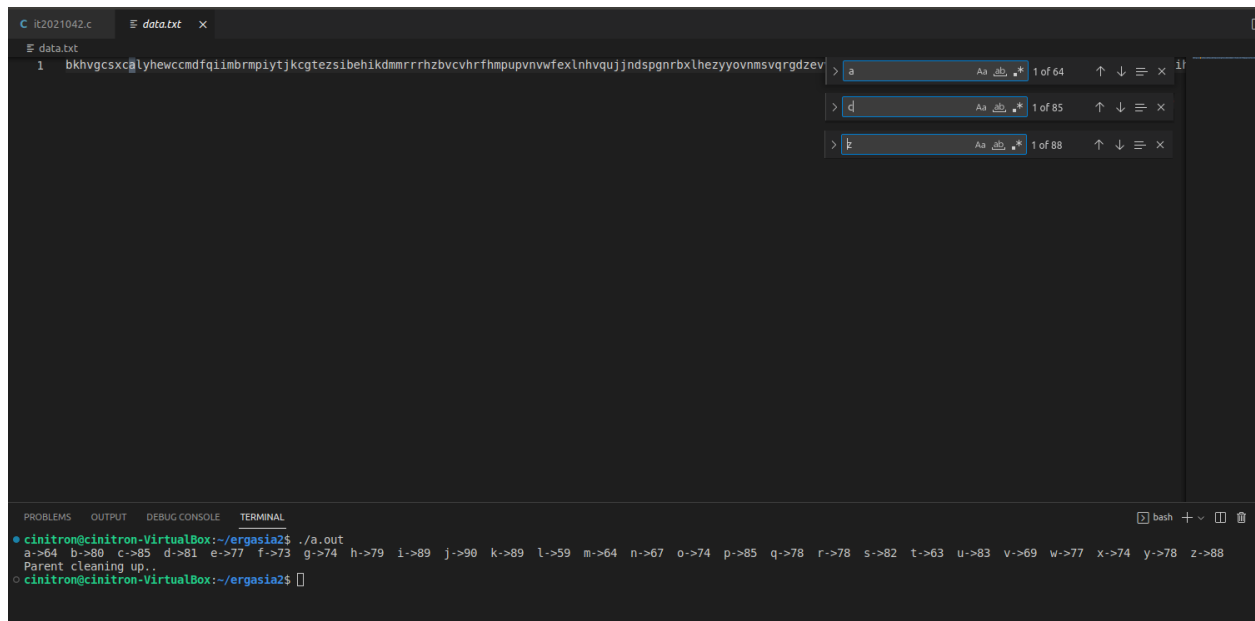
### Screenshots



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
cinitron@cinitron-VirtualBox:~/ergasia2$ gcc -pthread it2021042.c
it2021042.c: In function 'thread_func':
it2021042.c:34:14: warning: assignment makes integer from pointer without a cast [-Wint-conversion]
    partial[i] = "";
               ^
it2021042.c: In function 'main':
it2021042.c:114:9: warning: implicit declaration of function 'waitpid'; did you mean 'getpid'? [-Wimplicit-function-declaration]
    waitpid(pid,&status,0); // Perimenei mexri na teleiosei to paidi
    ^~~~~~
    getpid
cinitron@cinitron-VirtualBox:~/ergasia2$ ./a.out
a->76 b->88 c->79 d->86 e->68 f->76 g->79 h->71 i->68 j->74 k->72 l->57 m->87 n->70 o->85 p->91 q->98 r->69 s->73 t->73 u->75 v->62 w->69 x->85 y->78 z->91
Parent cleaning up..
cinitron@cinitron-VirtualBox:~/ergasia2$
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
cinitron@cinitron-VirtualBox:~/ergasia2$ ./a.out
^C
Terminating program..
cinitron@cinitron-VirtualBox:~/ergasia2$ ./a.out
^C
Terminating program..
cinitron@cinitron-VirtualBox:~/ergasia2$ ./a.out
a->88 b->80 c->79 d->78 e->72 f->45 g->81 h->74 i->77 j->76 k->82 l->65 m->92 n->76 o->83 p->78 q->79 r->78 s->84 t->82 u->77 v->81 w->69 x->69 y->81 z->74
Parent cleaning up..
cinitron@cinitron-VirtualBox:~/ergasia2$
```



```
C it2021042.c data.txt
data.txt
1 bkhvgcsxc@lyhewccmdfqiimbriptytkcgtezsibehikdmrrrhzbvcvhrfhpupvnnvfxlnhvqujjndspgnrbxlhezyovnmvsqrgdzev
> a Aa 1 of 64
> d Aa 1 of 85
> k Aa 1 of 88
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
cinitron@cinitron-VirtualBox:~/ergasia2$ ./a.out
a->64 b->80 c->85 d->81 e->77 f->73 g->74 h->79 i->89 j->90 k->89 l->59 m->64 n->67 o->74 p->85 q->78 r->78 s->82 t->63 u->83 v->69 w->77 x->74 y->78 z->88
Parent cleaning up..
cinitron@cinitron-VirtualBox:~/ergasia2$
```

## Γενικά Σχόλια/Παρατηρήσεις

Με δυσκόλεψε / δεν υλοποίησα

Υλοποίησα μερικώς τη διαχείριση σημάτων. Πιο συγκεκριμένα, Αν ληφθεί σήμα SIGINT ή SIGTERM ΔΕΝ ρωτάει τον χρήστη αν όντως θέλει να τερματίσει το πρόγραμμα, απλά το τερματίζει.

# Συνοπτικός Πίνακας

2η Εργασία		
Λειτουργία	Υλοποιήθηκε (ΝΑΙ/ΟΧΙ/ΜΕΡ ΙΚΩΣ)	Συνοπτικές Παρατηρήσεις
Βασική διεργασία η οποία ελέγχει τη δημιουργία των κατάλληλων διεργασιών και τον έλεγχο του προγράμματος	<b>ΝΑΙ</b>	
Συγχρονισμός των 2 διεργασιών εγγραφής ανάγνωσης	<b>ΝΑΙ</b>	
Διαχείριση σημάτων	<b>ΜΕΡΙΚΩΣ</b>	ΔΕΝ ρωτάει τον χρήστη αν όντως θέλει να τερματίσει το πρόγραμμα, απλά το τερματίζει.
Δημιουργία νημάτων και πέρασμα παραμέτρων	<b>ΝΑΙ</b>	
Συγχρονισμός νημάτων και σωστή διαδικασία μέτρησης αποτελέσματος	<b>ΝΑΙ</b>	
Ομαλή εκτέλεση προγράμματος, error handling, τεκμηρίωση	<b>ΝΑΙ</b>	