it2022120

Αναφορά άσκησης bonus: Fread και Read

Οδηγείες εκτέλεσης:

Για την εκτέλεση των προγραμμάτων χρειάζεται να δώσουμε σαν παραμέτρους (argv[1] , argv[2]) το όνομα του αρχείου που θέλουμε να διαβάσει το πρόγραμμα και στη συνέχεια ένα χαρακτήρα ‘S’ ή ‘s’ για να διαβάσει απο το αρχείο σειριακά, ενώ με ‘I’ ή ‘i’ για να διαβάσει με το εναλλάξ μοτίβο.

```
karatziask@karatziask:~/code/oper_sys$ time ./read testfile_1KB.txt s
```

Fread & Read:

Τα προγράμματα απαιτούν ένα αρχείο και ένα χαρακτήρα από το τερματικό. Σε περίπτωση που δεν τα λάβουν εμφανίζουν στη οθόνη μήνυμα με το αντίστοιχο error. Δημιουργείται μια λίστα απο χαρακτήρες με μέγεθος 1000 bytes. Ανάλογα με την είσοδο που δώσαμε στο τερματικό αν το argv[2] είναι ‘S’ ή ‘s’ τότε διαβάζει σειριακά 1000 bytes απο το αρχείο με την fread() ή την read() αντίστοιχα και τερματήζει. Ενώ αν στο τερματικό επιλέξουμε να διαβάσει απο το αρχείο εναλλάξ μετακινούμε τον κέρσορα συνεχώς στο σωστό σημείο και διαβάζουμε 1 byte κάθε φορά.

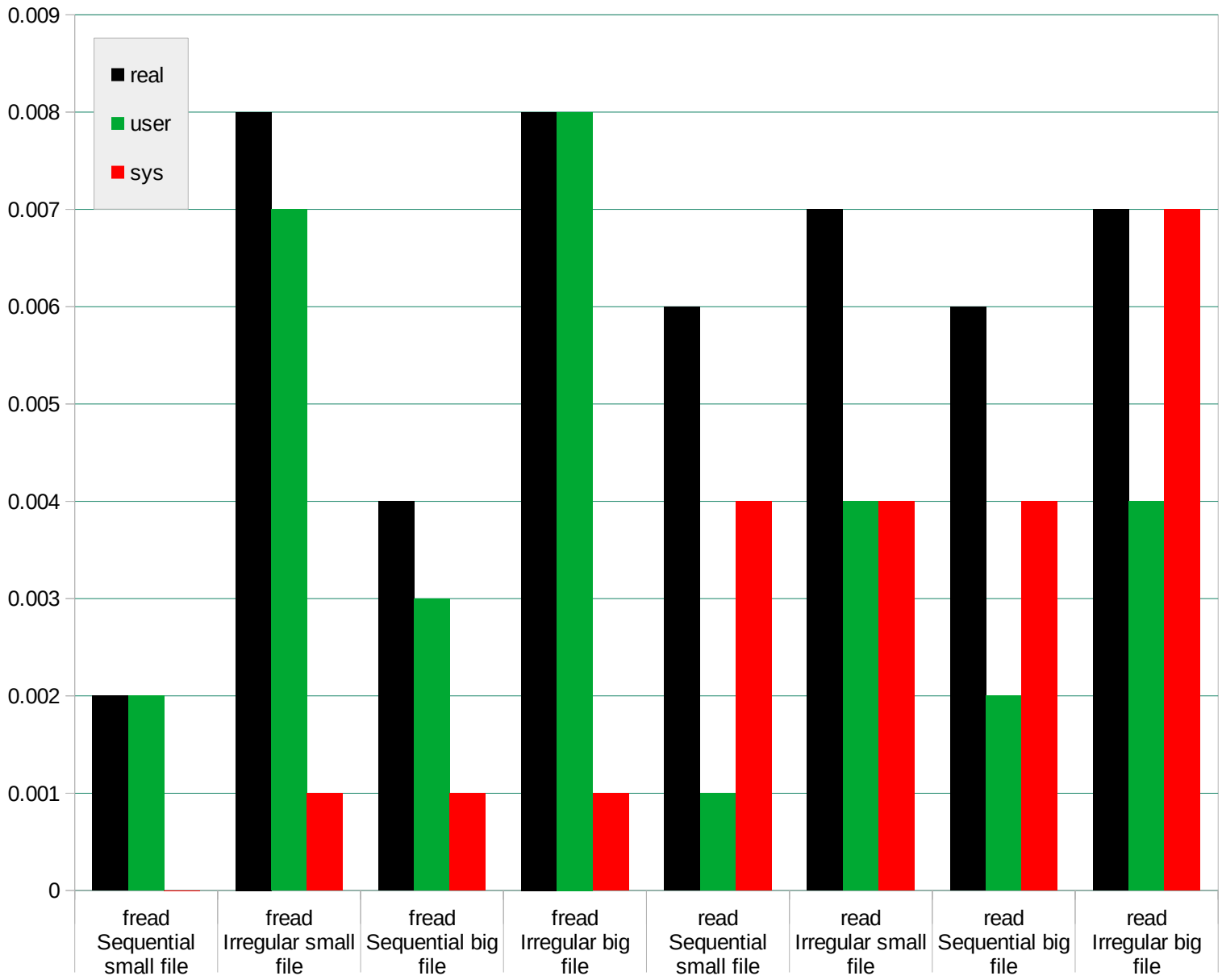
read.c

```c
#include<stdio.h>
#include<fcntl.h>
#include<unistd.h>
#include<stdlib.h>

int main(int argc,char **argv) {
    int fd,i=0;
    ssize_t size = 1000;

    if(argc < 2){
        printf("Not enough arguments!\n You have to add the \".txt\" file and method!\n");
    }

    fd = open(argv[1], O_RDONLY);

    if(fd == -1) {
        printf("Error opening file\n");
        exit(0);
    }

    if(size == -1) {
        printf("Error reading from file");
        close(fd);
        exit(0);
    }

    int j = 0;
    char buffer[size];
    lseek(fd, 0, SEEK_SET);
    if(argv[2][0] == 's' || argv[2][0] == 'S' ){
        for(int i = 0; i < size ; i++){
            read(fd, &buffer[i],1);
        }
    }else if(argv[2][0] == 'i' || argv[2][0] == 'I' ){
        for(int i = 1; i <= size/2; i++){
            lseek(fd, i-1, SEEK_SET);
            read(fd, &buffer[j], 1);
            lseek(fd, -i-1, SEEK_END);
            read(fd, &buffer[j+1], 1);
            j+=2;
        }
    }else {
        printf("Wrong arguments!\n");
    }
    close(fd);

    return 0;
}
```

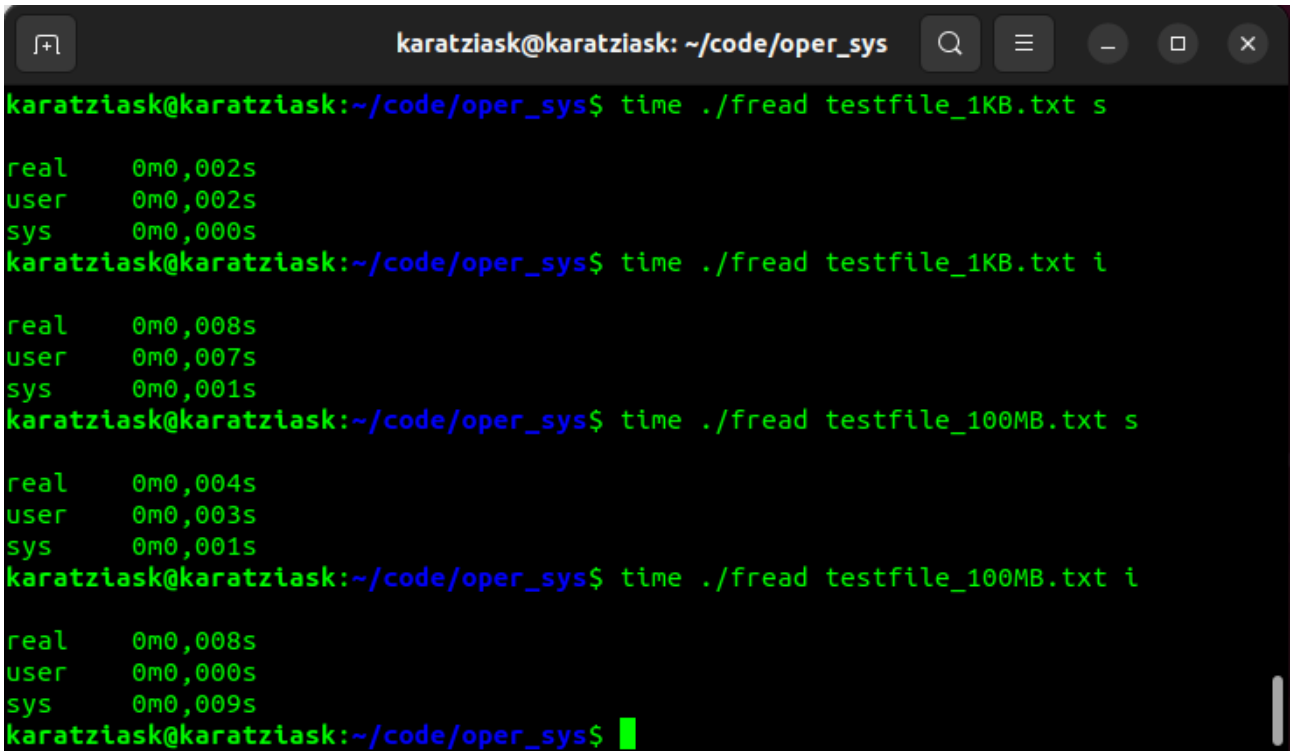
fread.c

```c
#include<stdio.h>
#include<stdlib.h>

#define size 1000

int main (int argc,char **argv){

    if(argc < 2 ){
        printf("Not enough arguments!\n You have to add the \".txt\" file!\n");
    }

    FILE *fl = fopen(argv[1],"r");
    if(fl == NULL){
        printf("Error opening file!\n");
        exit(0);
    }

    char buffer[size];
    int j = 0;

    rewind(fl);
    int i=0;

    if(argv[2][0] == 'S' || argv[2][0] == 's'){
        for(int i=0; i < size ; i++){
            fread(&buffer[i], 1, 1, fl);
        }
    }else if(argv[2][0] == 'I' || argv[2][0] == 'i' ){
        for(int i = 1; i <= (size+1)/2; i++){
            fseek(fl, i-1, SEEK_SET);
            fread(&buffer[j],1,1,fl);
            fseek(fl, -i-1, SEEK_END);
            fread(&buffer[j+1], 1, 1, fl);
            j+=2;
        }
    }else {
        printf("Wrong arguments!\n");
    }

    fclose(fl);
}
```

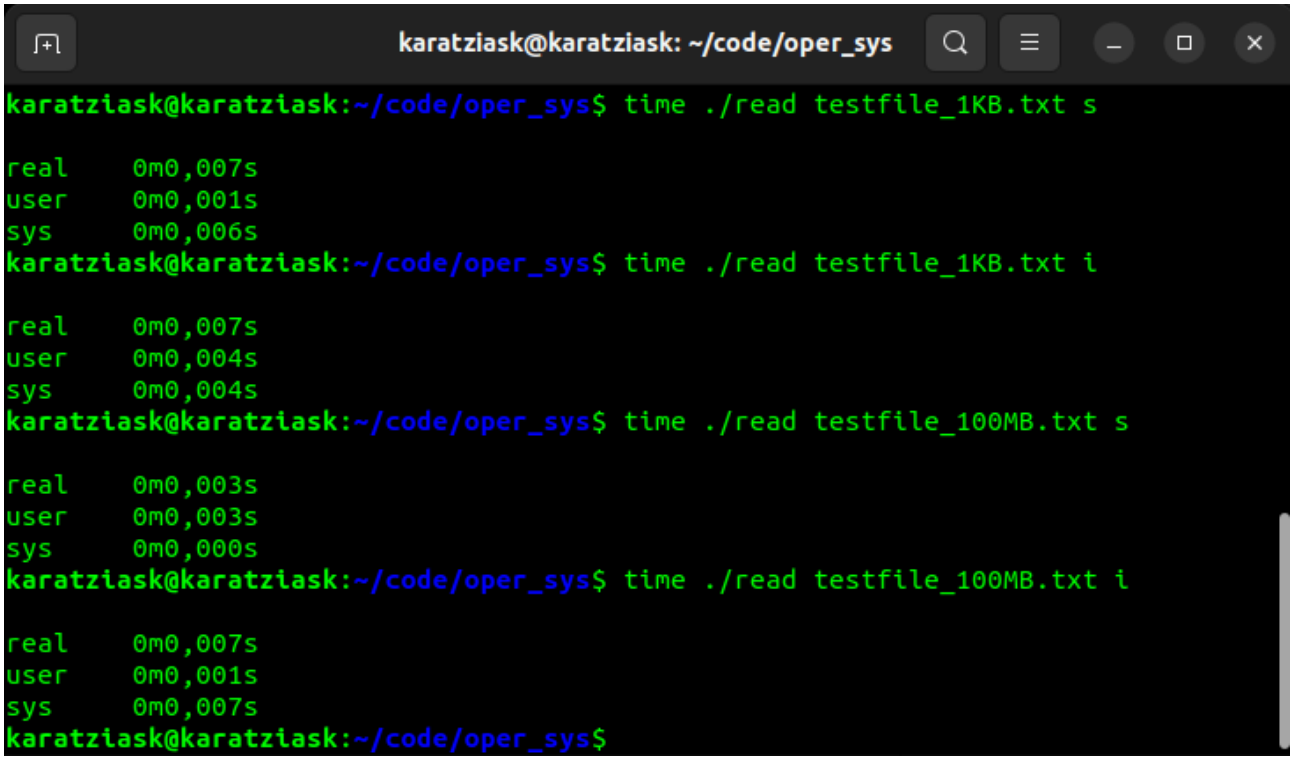
Γραφική παράσταση χρόνων:



Τερματικό εκτέλεσης Fread:



Τερματικό εκτέλεσης Read:



Χρόνοι εκτέλεσης:

Παρατηρούμε στους χρόνους εκτέλεσης:

- Με την χρήση της fread ο χρόνος που χρειάζεται για να διαβάσει απο το αρχείο εναλλάξ είναι μεγάλύτερος από αυτό που διαβάζει σειριακά ανεξαιρέτως του μεγέθους του αρχείου. Αυτό οφείλεται λόγω των εκτελέσεων των fseek. Επίσης κατά βάση χρησημοποιείται το user mode.
- Με την χρήση της read παρατηρούμε ότι ο χρόνος εκτέλεσης αφορά κυρίος το kernel mode λόγω της χρήσης system call (read). Ο χρόνος των user mode όταν διαβάζουμε το αρχείο εναλλάξ είναι αυξημένος σε σχέση με την σειριακή αναγνωση λόγω της lseek που χρησιμοποιούμε.
- Γενικά έχουμε μεγαλύτερο χρόνο στην εκτέλεση όταν διαβάζουμε το αρχείο εναλλάξ παρά σειριακά και αυτό οφείλεται στις μετακινήσεις που κάνουμε του κέρσορα. Δεν φάνηκε μεγάλη διαφορά όταν διαβάζαμε μέγάλο η μικρό αρχείο.