Προγραμματισμός σε C

Αρχεία κειμένου

(Text files)

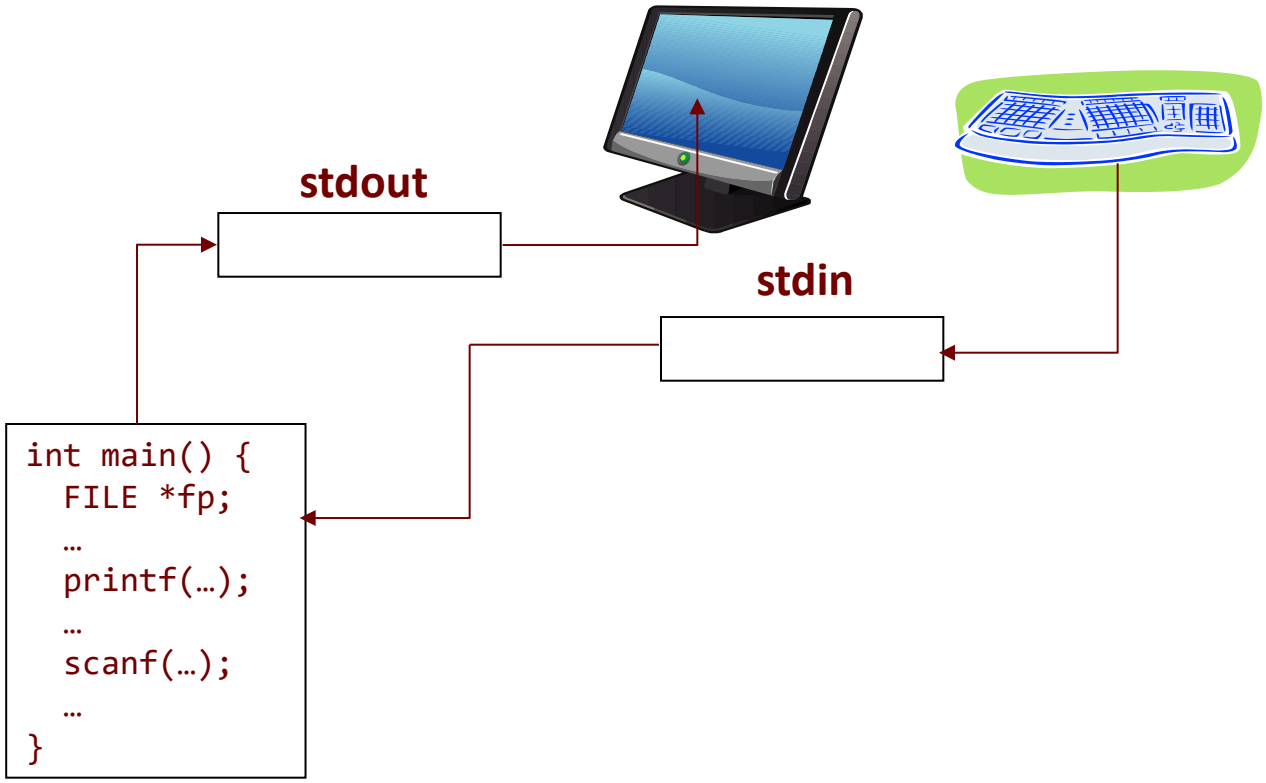
ΜΥΥ502

Εισαγωγή

- Στη C έχουμε ειδικές συναρτήσεις για να επεξεργαζόμαστε αρχεία κειμένου που αποθηκεύονται στο δίσκο
  - Τα αρχεία είναι σημαντικά για μόνιμη αποθήκευση "κειμένου"
  - Αποθηκεύουν "απεριόριστη" πληροφορία.
  - Μπορείτε να τα θεωρήσετε ως ένα μεγάλο string που φυλάσσεται στον δίσκο και όχι στη μνήμη
  - Όμως ΔΕΝ είναι σαν τα strings διότι π.χ.
    - Το μέγεθός του μόνο αυξάνεται, δεν αφαιρείται τίποτε…
    - Δεν μπορείς να πας άμεσα σε όποιο στοιχείο (χαρακτήρα) θέλεις – συνήθως ξεκινάς από την αρχή και προχωράς …
    - Επομένως δεν μπορείς να το χειριστείς με δείκτες.
- Προκειμένου να αποθηκευτεί / διαβαστεί πληροφορία σε αρχείο, απαιτείται η χρήση ***ρευμάτων*** εισόδου/εξόδου (***streams***)
  - ένα ρεύμα εξόδου μπορείτε να το φαντάζεστε σαν χώρο μνήμης στον οποίο το πρόγραμμα μόνο αποθηκεύει δεδομένα (χρησιμοποιώντας γνωστές συναρτήσεις – printf, fprintf, puts, fputs) τα οποία εν συνεχεία τα παραλαμβάνει το λειτουργικό και τα στέλνει στο δίσκο.
  - ένα ρεύμα εισόδου μπορείτε να το φαντάζεστε σαν χώρο μνήμης στον οποίο το λειτουργικό αποθηκεύει δεδομένα από το δίσκο και εν συνεχεία το πρόγραμμα μπορεί να παραλάβει αυτά τα δεδομένα για επεξεργασία χρησιμοποιώντας γνωστές συναρτήσεις (scanf, gets, fscanf, fgets…).

Ειδικά ρεύματα (streams)

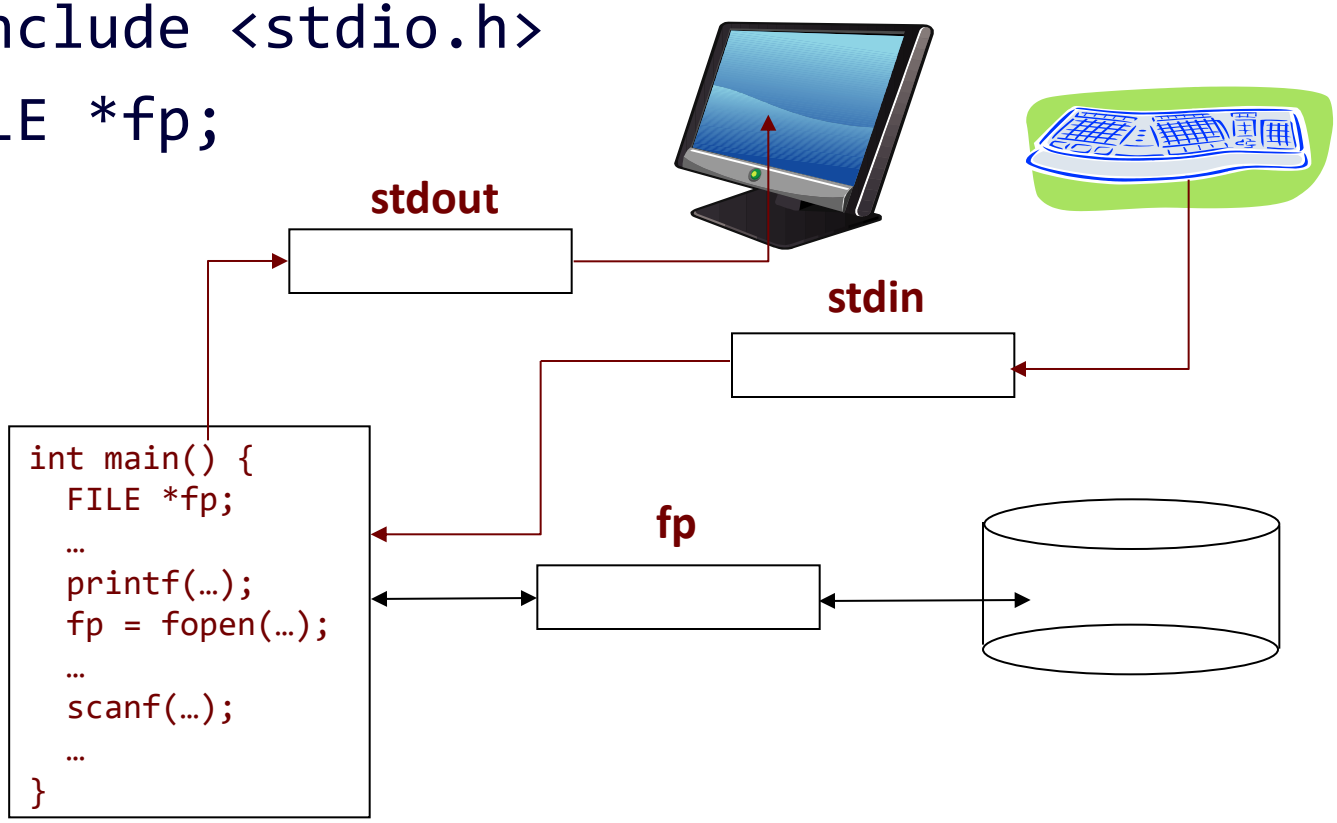
- **stdout** → γράφω σε ρεύμα που γίνεται flush στην **οθόνη**
- **stdin** → διαβάζω από ρεύμα στο οποίο γίνονται flush τα δεδομένα που περνάω από το **πληκτρολόγιο**



Ρεύματα που κατευθύνονται στο δίσκο

- Για να χρησιμοποιήσω από ένα πρόγραμμα κάποιο αρχείο χρειάζομαι ένα ρεύμα εισόδου/εξόδου το οποίο το δημιουργώ **μέσω ενός δείκτη σε FILE.**

```
#include <stdio.h>
FILE *fp;
```



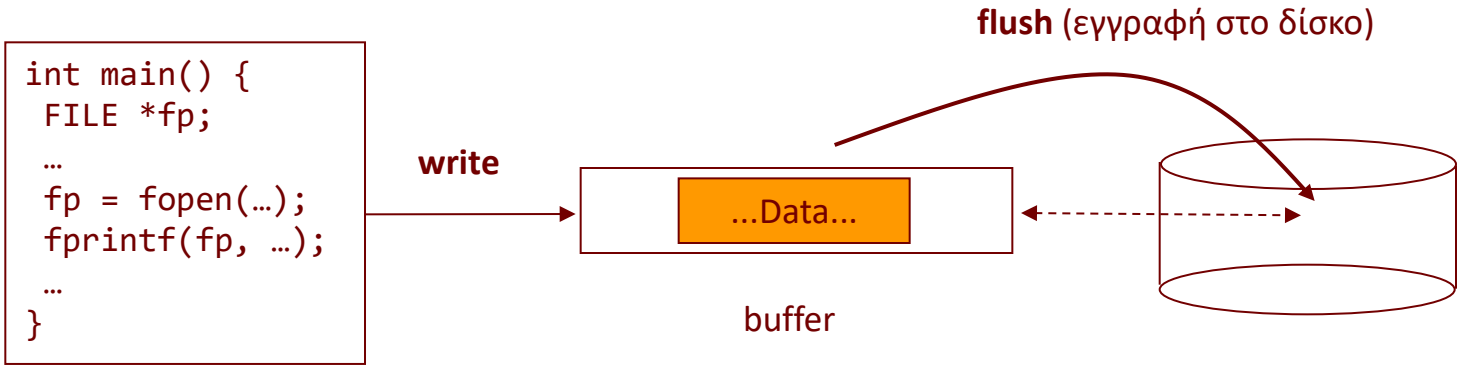
Αρχικοποίηση

- `FILE *fopen(char *name, char *mode);`
  - name: όνομα αρχείου
  - mode:
    - `"r"` για ανάγνωση (αν δεν ∃, η `fopen` επιστρέφει NULL)
    - `"w"` για εγγραφή (αν ∃ ήδη, τότε αδειάζουν τα περιεχόμενα)
    - `"a"` για επέκταση (αν ∃ ήδη, τα περιεχόμενα διατηρούνται)
  - Άλλες επιλογές για mode
    - Η πρόσθεση του + μετά από έναν χαρακτήρα σημαίνει ότι επιτρέπεται και η αντίθετη χρήση ταυτόχρονα
    - `"r+"` για ανάγνωση / εγγραφή στην αρχή (αν δεν ∃, επιστρέφει NULL)
    - `"w+"` για ανάγνωση / εγγραφή με απόρριψη (αν ∃ ήδη, άδειασμα)
    - `"a+"` για ανάγνωση / εγγραφή στο τέλος

- *ΣΗΜΕΙΩΣΗ: τα ειδικά ρεύματα stdout και stdin είναι πάντα αρχικοποιημένα αυτόματα όταν ξεκινά ένα πρόγραμμα.*

Αρχικοποίηση

- Η συνάρτηση fopen επιστρέφει έναν δείκτη (file pointer) σε ένα **ρεύμα** που συσχετίζεται με το αρχείο.
  - Άρα δεν μπορεί κανείς να προσβεί το αρχείο άμεσα μέσω του δείκτη (π.χ. δεν γίνεται, `*fp = 'a';`).
- Αν δεν υπάρχει αρχείο και ανοίγω με mode "w" ή "a", τότε δημιουργείται νέο αρχείο
- Αν κάτι πάει στραβά, η συνάρτηση επιστρέφει NULL



Βασικές Συναρτήσεις

- `int fscanf(FILE *fp, char *format, ...)`
  - Λειτουργεί όπως η `scanf`
  - Επιστρέφει αριθμό στοιχείων που διαβάστηκαν
  - Επιστρέφει **EOF** σε περίπτωση τέλους αρχείου ή λάθους
- `int fprintf(FILE *fp, char *format, ...)`
  - Λειτουργεί όπως η `printf` μόνο που γράφει στο fp
  - Επιστρέφει αριθμό χαρακτήρων που γράφτηκαν
  - Επιστρέφει < 0 σε περίπτωση λάθος

- Η scanf διαβάζει από τον ειδικό δείκτη αρχείου stdin, άρα:

```
scanf("%d", &i);  ≡  fscanf(stdin, "%d", &i);
```

- Η printf γράφει στον ειδικό δείκτη αρχείου stdout, επομένως:

```
printf("%d", i);  ≡  fprintf(stdout, "%d", i);
```

fgets

- `char *fgets(char *line, int maxline, FILE *fp);`
  - Διαβάζει το πολύ ***maxline-1*** χαρακτήρες
  - Αν βρει πρώτα αλλαγή γραμμής ('\n') διαβάζει μέχρι εκεί και η αλλαγή γραμμής ('\n') τοποθετείται στο line
  - Το line τερματίζει με το χαρακτήρα τερματισμού '\0'
  - Η συνάρτηση επιστρέφει **NULL** όταν τελειώσει το αρχείο ή συμβεί σφάλμα
- `char *gets(char *s);`
  - Η gets διαβάζει 1 γραμμή από το stdin ενώ αντικαθιστά τον τερματικό χαρακτήρα νέας γραμμής με ΄\0΄.
  - Αν size of line < μέγεθος της γραμμής που διαβάζω από το stdin , τότε έχω undefined behavior
  - Συνιστάται η χρήση της fgets αντί της gets, όπως έχουμε πει

- `int fputs(char *s, FILE *fp);`
  - Γράφει το string s στο fp
  - Η συνάρτηση επιστρέφει τον αριθμό των χαρακτήρων που έγραψε και EOF σε περίπτωση λάθους
- `int puts(char *s);`
  - Γράφει το string s μαζί με τον χαρακτήρα αλλαγής γραμμής στο stdout

Κλείσιμο αρχείου

- `int fclose(FILE *fp);`
  - Κλείνει το αρχείο (γίνεται flush ο buffer)
  - Επιστρέφει 0 σε περίπτωση επιτυχίας
  - Επιστρέφει EOF σε περίπτωση λάθους
  - Το σύστημα εκτέλεσης κλείνει όλα τα ανοικτά αρχεία με τον τερματισμό του κυρίου προγράμματος. Αποτελεί καλή προγραμματιστική πρακτική, παρ' όλα αυτά, να κλείνουμε ρητά τα αρχεία που ανοίγουμε.

- Μπορεί να εκτελεστεί η εντολή fclose(stdout), μετά όμως δεν μπορώ να γράψω – η fprintf(stdout,...) θα επιστρέφει **EOF.**

Λεπτομέρειες

- Ο δείκτης `FILE *fp`, δεν μας δίνει άμεση πρόσβαση στα περιεχόμενα του αρχείου (π.χ. ΔΕΝ ΜΠΟΡΟΥΜΕ ΝΑ ΚΑΝΟΥΜΕ ΑΡΙΘΜΗΤΙΚΗ ΜΕ ΑΥΤΟΝ). Άρα δεν μπορούμε να γράψουμε στο αρχείο με εντολές τύπου:

```
*fp = 'a'; /* Ο δείκτης δεν δείχνει στα περιεχόμενα */
strcpy(fp, "abc");     /* Το ίδιο */
```

- Τα περιεχόμενα τα διαβάζουμε / γράφουμε **ΜΟΝΟ** μέσω των fgets/fputs/fscanf/fprintf.

- Το αρχείο είναι σαν **ΤΑΙΝΙΑ** ΑΠΟ ΚΑΣΕΤΑ. Όταν διαβάζουμε ή γράφουμε κάτι, **ΠΡΟΧΩΡΑΕΙ Η ΤΑΙΝΙΑ ΑΠΌ ΜΟΝΗ ΤΗΣ**.
  - Επομένως, δεν προχωράμε εμείς τον δείκτη fp
  - Υπάρχει κλήση που μας επιστρέφει στην αρχή, ή σε κάποιο άλλο σημείο της "ταινίας".

Παράδειγμα 1 – fgets/fputs (διάβασμα γραμμή-γραμμή)

```
#include <stdio.h>

int main() {
   FILE *infile, *outfile;
   char buf[10];

   if ((infile = fopen("original.txt", "r")) == NULL)
         return 1;
   if ((outfile = fopen("copy.txt", "w")) == NULL)
         return 1;
   while (fgets(buf, 10, infile) != NULL) {
        fputs(buf, outfile);
   }
   fclose(infile);
   fclose(outfile);
   return 0;
}
```

Παράδειγμα 2 – fscanf/fprintf (διάβασμα λέξη-λέξη)

```
#include <stdio.h>

int main() {
   FILE *infile, *outfile;
   char buf[81];

   if ((infile = fopen("original.txt", "r")) == NULL)
         return 1;
   if ((outfile = fopen("copy.txt", "w")) == NULL)
         return 1;
   while (fscanf(infile, "%s", buf)  != EOF) {
        fprintf(outfile, "%s", buf); /* Όλες οι λέξεις κολλητά */
        fprintf(stdout, "%s", buf);  /* Εμφάνιση και στην οθόνη */
   }
   fclose(infile);
   fclose(outfile);
   return 0;
}
```

Παράδειγμα 3 – fflush/fclose

```
int main() {
   FILE *fp;
   fp = fopen("file", "w"); /* Should check for NULL */
   fprintf(fp, "%s", "xxx");
   while(1) ;
   return 0;
}
```

```
$./a.out
^C
```

- Η εκτέλεση της εντολής "cat file" κατά την ώρα εκτέλεσης του προγράμματος δεν θα δείξει δεδομένα
- Το ίδιο θα συμβεί και μετά τον μη ομαλό τερματισμό του προγράμματος (με Control-C)
- Για την εγγραφή των δεδομένων πρέπει να προστεθεί η **`fflush(fp);`** πριν το while loop, η οποία ***ολοκληρώνει ότι εγγραφές έχουν γίνει μέχρι εκείνη τη στιγμή στον δίσκο***.
- Η `fclose(fp);` κάνει ακριβώς το ίδιο (μόνο που επιπλέον κλείνει και το αρχείο).

Παράδειγμα 4 – Εύρεση μέγιστης λέξης

```
#include <stdio.h>
#include <string.h>

int main() {
   FILE *infile;
   char buf[101], maxWord[101];
   int maxLength = 0;

   if ((infile=fopen("testWords.ascii", "r")) == NULL) return 1;

   strcpy(maxWord, "");
   while (fscanf(infile, "%s", buf) != EOF) {
        if (strlen(buf)>maxLength) {
                strcpy(maxWord, buf);
                maxLength = strlen(buf);
        }
   }
   fprintf(stdout, "Max string is: %s with length %d\n",
           maxWord, maxLength);
   return 0;
}
```

Παράδειγμα 4 – Β' έκδοση, με χρήση stdin

```
/* Άμεση πληκτρολόγηση γραμμών κειμένου και τερματισμός με τον συνδυασμό Control-
   D (^D),  που σηματοδοτεί το EOF */
#include <stdio.h>
#include <string.h>

int main() {
   char buf[101], maxWord[101];
   int maxLength = 0;

   strcpy(maxWord, "");
   while (fscanf(stdin, "%s", buf) != EOF) {
        if (strlen(buf)>maxLength) {
                strcpy(maxWord, buf);
                maxLength = strlen(buf);
        }
   }
   fprintf(stdout, "Max string is:%s with length %d\n",
           maxWord, maxLength);
   return 0;
}
```

Παράδειγμα 5 – fgets/sscanf (διάβασμα γραμμών + sscanf)

```
#include <stdio.h>

int main() {
   FILE *infile;
   int k;
   char s1[20], s2[20];
   float f;
   char buf[81];

   if ((infile = fopen("testSscanf.txt", "r")) == NULL) return 1;
   while (fgets(buf, 81, infile) != NULL) {
        puts(buf);     /* show original line */
        sscanf(buf, "%d %s %s %f", &k, s1, s2, &f);
        printf("%d\t%s\t%s\t%f\n", k, s1, s2, f);
   }
   fclose(infile);
   return 0;
}
```

```
$cat testSscanf.txt
2 minutes to 12.00
4 days to 1.0 breakdown
3 months to 5
$./a.out
2 minutes to 12.00

2       minutes to      12.000000
4 days to 1.0 breakdown

4       days       to      1.000000
3 months to 5

3       months  to      5.000000
```

Παράδειγμα 6 – μίνι grep

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]) {
    FILE *infile;
    char buf[101];

    if ((infile = fopen("testWords.ascii", "r"))==NULL) {
           return 1;
    }

    if (argc != 2) {
          printf("Usage: %s searchString\n", argv[0]);
          return 1;
    }

    while (fgets(buf, 101, infile) != NULL) {
          if (strstr(buf, argv[1]) != NULL)
                printf("%s", buf);          /* No "%s\n" here ?? */
    }
    fclose(infile);
    return 0;
}
```

Παράδειγμα 7 – Εκτύπωση λέξεων (fgets + strtok)

```
#include <stdio.h>
#include <string.h>

int main() {
   FILE *infile;
   int c;
   char buf[81];
   char *p, tokens[]=" \n\t";

   if ((infile = fopen("testWords.ascii", "r"))==NULL) return 1;

   while (fgets(buf, 81, infile) != NULL) {
        c = buf[strlen(buf)-1];
        p = strtok(buf, tokens); /* εκτύπωση λέξεων γραμμής στην οθόνη */
        while (p) {
            printf("%s ", p); /* ένα κενό μεταξύ των λέξεων */
            p = strtok(NULL, tokens);
        }
        if (c=='\n') printf("\n");
   }
   fclose(infile);
   return 0;
}
```

Παράδειγμα 8 – Μετρητής λέξης

```
#include <stdio.h>
#include <string.h>

void CheckDiomedes(char *X, int *counter) {
        char *c = strstr(X, "Diomhdh");

        while (c) {
            (*counter)++;
            c += strlen("Diomhdh");
            c  = strstr(c, "Diomhdh");
        }
        printf("%s %d\n", X, *counter);
}

int main() {
        FILE *infile;
        char buf[81];
        int nOcc = 0;

        if ((infile = fopen("iliada.txt", "r"))==NULL)
            return 1;
        while (fscanf(infile, "%s", buf) != EOF)
            CheckDiomedes(buf, &nOcc);
        printf("Diomhdh: %d\n", nOcc);
        fclose(infile);
        return 0;
}
```

Παράδειγμα 9 – Ένωση αρχείων

```
/* Παράδειγμα εκτέλεσης: $./a.out f1.txt f2.txt bothfiles.txt */
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
   FILE *inp, *outp;
   char line[200];
   int i;

   if ((outp = fopen(argv[argc-1], "w")) == NULL) return 1;
   for (i = 1; i < argc -1; i++) {
        if ((inp = fopen(argv[i], "r")) == NULL) continue;
        while (fgets(line, 200, inp) != NULL)
            fputs(line, outp);
        fclose(inp);
   }
   fclose(outp);
   return 0;
}
```

Θέση σε αρχείο

- Η δομή `FILE` διαχειρίζεται τη **θέση** του αρχείου στην οποία βρισκόμαστε αυτή τη στιγμή
  - Η αρίθμηση είναι από το 0 (αρχή του αρχείου, «πριν» τον πρώτο χαρακτήρα του αρχείου) μέχρι N, όπου N είναι το μέγεθος του αρχείου (θέση αμέσως μετά τον τελευταίο χαρακτήρα).
- Η θέση αυτή περιγράφεται από έναν ακέραιο long int.
  - Ουσιαστικά περιγράφει πόσους χαρακτήρες απέχουμε από την αρχή του αρχείου.

- Όπως γνωρίζουμε, αν καλέσουμε μια συνάρτηση εισόδου-εξόδου (π.χ. fscanf/fprintf), ***η τρέχουσα θέση αλλάζει αυτόματα μετά την ολοκλήρωσή της*** (προχωρά η "ταινία").

- Για να μάθουμε την τρέχουσα θέση:
  - `long ftell(FILE *fp);`

    Επιστρέφει την τρέχουσα θέση στο αρχείο
- Μπορούμε να την τροποποιήσουμε με συγκεκριμένες εντολές

fseek / rewind

- `int fseek(FILE *fp, long offset, int pos);`
  - Η fseek *ξεκινάει από την θέση* ***pos*** και προσθέτει ή αφαιρεί **offset** θέσεις
  - Δυνατές τιμές για την pos
    - SEEK_CUR: τρέχουσα θέση στο αρχείο
    - SEEK_SET: αρχή αρχείου
    - SEEK_END: τέλος αρχείου

- `void rewind(FILE *fp);`
  - Τρέχουσα θέση = αρχή του αρχείου
  - `rewind(fp)` ≡ `fseek(fp, 0L, SEEK_SET);`

- Οι συναρτήσεις εγγραφής και ανάγνωσης (π.χ. fputs, fgets) αλλάζουν την τρέχουσα θέση.

Παραδείγματα fseek

- `fseek(fp, 0L, SEEK_SET)`
  - Στην αρχή του αρχείου
- `fseek(fp, 0L, SEEK_END)`
  - Στο τέλος του αρχείου (*μετά* τον τελευταίο χαρακτήρα)
- `fseek(fp, (long) -1, SEEK_END)`
  - Πριν τον τελευταίο χαρακτήρα. Η επόμενη ανάγνωση θα διαβάσει τον τελευταίο χαρακτήρα.
- `fseek(fp, ftell(fp), SEEK_SET)`

  `fseek(fp, 0L, SEEK_CUR)`
  - Μείνε εδώ που είσαι (ισοδύναμα)

- Κατά το άνοιγμα του αρχείου με:
  - `FILE *fp = fopen("file", "r+");`

    Η αρχική θέση θα είναι στην **αρχή** του αρχείου
  - `FILE *fp = fopen("file", "a+");`

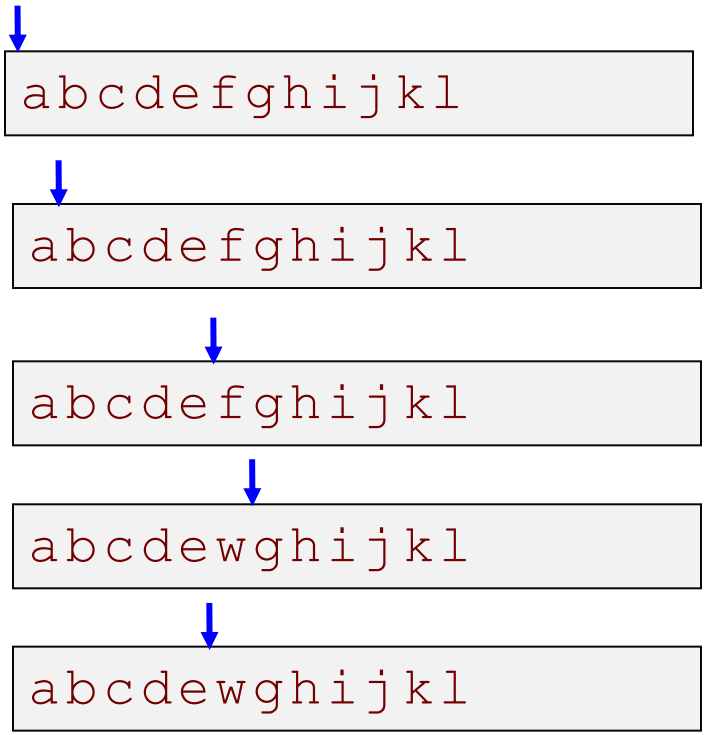
    Η αρχική θέση θα είναι στο **τέλος** του αρχείου

Παράδειγμα fseek

```
#include <stdio.h>
int main() {
   FILE *fp;
   char line[200];

   fp = fopen("file1.txt", "r+");
   fgets(line, 2, fp);
   fseek(fp, 4, SEEK_CUR);
   fputs("w", fp);
   fseek(fp, -1, SEEK_CUR);
   fgets(line, 200, fp);
   fprintf(stdout, "%s", line);
   return 0;
}
```

```
$./a.out
wghijkl
$cat file1.txt
abcdewghijkl
```



Παράδειγμα ftell

```
#include <stdio.h>
int main() {
   FILE *fp;
   char line[200];

   fp = fopen("file2.test", "r+");
   printf("%ld\n", ftell(fp));
   fseek(fp, ftell(fp)+4, SEEK_SET);
   fputs("w", fp);
   fseek(fp, ftell(fp)-1, SEEK_SET);
   fgets(line, 200, fp);
   fprintf(stdout, "%s", line);
   printf("%ld\n", ftell(fp));
   fseek(fp, 4, SEEK_SET);
   fgets(line, 200, fp);
   fprintf(stdout, "%s", line);
   fclose(fp);
   return 0;
}
```

```
$cat file2.test
abcdefghijkl
$./a.out
0
wfghijkl
13
wfghijkl
$cat file2.test
abcdwfghijkl
```

Προσθήκη δεδομένων ("a+")

```
#include <stdio.h>

int main() {
   FILE *fp;
   char buf[81] = "22 hours ago";

   fp = fopen("testMode.txt", "a+");
   if (fp == NULL) return 1;
   fputs(buf, fp);     /* γράφει στο τέλος του αρχείου */

   rewind(fp);
   while (fgets(buf, 81, fp) != NULL)
        fputs(buf, stdout);   /* εκτύπωση στην οθόνη */
   fclose(fp);
   return 0;
}
```

```
$ cat testMode.txt
2 minutes to 12.00
4 days to 1.0 breakdown
3 months to 5
$ ./a.out
...
$ cat testMode.txt
2 minutes to 12.00
4 days to 1.0 breakdown
3 months to 5
22 hours ago
```

Προσθήκη δεδομένων ("r+")

```
#include <stdio.h>

int main() {
   FILE *fp;
   char buf[81] = "22 hours ago";

   fp = fopen("testMode.txt", "r+");
   if (fp == NULL) return 1;
   fputs(buf, fp);     /* γράφει στη τρέχουσα θέση, δηλαδή στην
                          αρχή του αρχείου */
   rewind(fp);
   while (fgets(buf, 81, fp) != NULL)
        fputs(buf, stdout);   /* εκτύπωση στην οθόνη */
   fclose(fp);
   return 0;
}
```

```
$ cat testMode.txt
2 minutes to 12.00
4 days to 1.0 breakdown
3 months to 5
$ ./a.out
...
$ cat testMode.txt
22 hours ago 12.00
4 days to 1.0 breakdown
3 months to 5
```

Προσθήκη δεδομένων ("w+")

```
#include <stdio.h>

int main() {
   FILE *fp;
   char buf[81] = "22 hours ago";

   fp = fopen("testMode.txt", "w+");
   if (fp == NULL) return 1;
   fputs(buf, fp);     /* γράφει στην αρχή του αρχείου, όλα τα
                          προηγούμενα δεδομένα χάνονται */
   rewind(fp);
   while (fgets(buf, 81, fp) != NULL)
        fputs(buf, stdout);   /* εκτύπωση στην οθόνη */
   fclose(fp);
   return 0;
}
```

```
$ cat testMode.txt
2 minutes to 12.00
4 days to 1.0 breakdown
3 months to 5
$ ./a.out
...
$ cat testMode.txt
22 hours ago
```