Αρχεία

Τα κανάλια stdin, stdout, stderr

•Κάθε φορά που ξεκινά η εκτέλεση ενός προγράμματος, ο υπολογιστής ανοίγει αυτόματα το **κανάλι καθιερωμένης εισόδου *stdin*** (standard input), το **κανάλι καθιερωμένης εξόδου *stdout*** (standard output) και το **κανάλι σφαλμάτων *stderr*** (standard errors). Γενικά αυτά τα κανάλια αναφέρονται στην κονσόλα αλλά το λειτουργικό σύστημα μπορεί να τα ανακατευθύνει σε κάποια άλλη συσκευή.

•Επειδή πρόκειται για δείκτες αρχείου το σύστημα I/O **με ενδιάμεση αποθήκευση (buffering)** μπορεί να χρησιμοποιεί αυτά τα κανάλια για να εκτελεί λειτουργίες I/O στην κονσόλα.

•Το ***stdin*** χρησιμοποιείται για ανάγνωση από την κονσόλα και τα ***stdout***, ***stderr*** για εγγραφή στην κονσόλα. Μπορούν να χρησιμοποιηθούν τα ***stdin***, ***stdout***, ***stderr*** ως δείκτες αρχείου σε οποιαδήποτε συνάρτηση χρησιμοποιεί μία μεταβλητή τύπου **FILE ***.

Τα κανάλια stdin, stdout, stderr

- Μπορούν να ανακατευθυνθούν τα κανάλια και να γράφονται τα μηνύματα λάθους σε αρχείο αντί να εμφανίζονται στην οθόνη.
- Με την εντολή ***program_name < filename*** ορίζεται ως κύρια είσοδος αντί του πληκτρολογίου το αρχείο **filename**.
- Με την εντολή ***program_name > filename*** ορίζεται ως κύρια έξοδος αντί για την οθόνη το αρχείο **filename**.
- Οι παραπάνω εντολές δίνονται από τη γραμμή διαταγής (command line). Η ***printf*** γράφει στο **stdout** και η ***scanf*** στο **stdin**.

Τα stdin, stdout, stderr δεν είναι μεταβλητές αλλά σταθερές και δεν μπορούν να αλλαχθούν. Όπως ο υπολογιστής δημιουργεί αυτόματα αυτούς τους δείκτες αρχείου στην αρχή του προγράμματος, έτσι και τους αποσύρει αυτόματα στο τέλος του προγράμματος. Δε θα πρέπει να κλείσουν αυτά τα κανάλια με παρέμβαση του χρήστη.

Ενδιάμεση μνήμη (buffer)

•Για ανάγνωση κι εγγραφή σε συσκευές εισόδου/εξόδου (input/output, I/O), όπως ο σκληρός δίσκος, τα λειτουργικά συστήματα χρησιμοποιούν ενδιάμεση μνήμη (buffers), η οποία είναι περιοχή της μνήμης όπου τα δεδομένα αποθηκεύονται προσωρινά πριν σταλούν στον τελικό τους στόχο. Έτσι επιταχύνονται τα προγράμματα γιατί ελαχιστοποιείται ο αριθμός των προσβάσεων στις I/O συσκευές.

•Οι μονάδες I/O επιτρέπουν στο λειτουργικό σύστημα να έχει πρόσβαση μόνο σε καθορισμένου μεγέθους τμήματα, τα ονομαζόμενα blocks, μεγέθους 512 ή 1024 bytes. Επομένως, ακόμη κι αν θέλουμε να διαβάσουμε μόνο ένα χαρακτήρα από ένα αρχείο, το λειτουργικό σύστημα διαβάζει όλο το μπλοκ στο οποίο βρίσκεται αποθηκευμένος ο χαρακτήρας. Έτσι, με τη χρήση του buffer, εάν χρειασθούμε άλλους χαρακτήρες από το ίδιο μπλοκ δεν επιστρέφουμε στη συσκευή αλλά τους διαβάζουμε από το buffer.

Δύο ευρείες τάξεις αρχείων

- **Δυαδικά αρχεία (binary files):**
  - Αποθηκεύουν **κάθε** τύπο δεδομένου: οριζόμενο από το χρήστη, char, int, float, double, string, data struct, κ.λ.π.
  - Συνήθως ΔΕΝ είναι αναγνώσιμα από τους συντάκτες (editors)
  - Συνήθως ΔΕΝ είναι φορητά (δεν ανοίγουν σε όλα τα μηχανήματα)

- **Αρχεία κειμένου (text files)**
  - Αποθηκεύουν μία ακολουθία (ένα 'ρεύμα - stream') από bytes χαρακτήρων.
  - Είναι αναγνώσιμα από τους συντάκτες (π.χ. αρχεία .h, .c)
  - Είναι φορητά σε κάθε υπολογιστή (σχεδόν)

- **Στα αρχεία κειμένου τα πάντα αποθηκεύονται ως ακολουθίες χαρακτήρων (τα *stdin*, *stdout* είναι ανοικτά ως κανάλια κειμένου).**

- **Στα δυαδικά αρχεία ο μεταγλωττιστής δεν κάνει μεταγλώττιση των bytes, απλά διαβάζει και γράφει bits, ακριβώς όπως αυτά εμφανίζονται.**

Παράδειγμα:

Ο αριθμός 12345 εγγράφεται ως αλφαριθμητικό σε αρχείο κειμένου, απαιτώντας 6 bytes (1 για κάθε χαρακτήρα κι ένα για το χαρακτήρα τερματισμού '\0'). Αντίθετα, σε ένα δυαδικό αρχείο εγγράφεται ως ακέραιος, απαιτώντας 4 bytes.

Άνοιγμα – κλείσιμο αρχείου

- **ΠΑΝΤΟΤΕ** **να ανοίγετε** ένα αρχείο πριν τη χρήση,
- **ΠΑΝΤΟΤΕ** **να το κλείνετε** όταν περατώνεται η χρήση του.

```
FILE *pF;    /* δήλωση ενός pointer σε μεταβλητή FILE */

pF = fopen("myfile.txt","r");     /* άνοιγμα αρχείου */

                /* . . . Διάφορες λειτουργίες . . .*/

fclose(pF);
```

Το όνομα αρχείου εισέρχεται ως string
(ο δείκτης δείχνει στον πρώτο χαρακτήρα του)

Προσδιοριστής string που ελέγχει το είδος της πρόσβασης

Άνοιγμα – κλείσιμο αρχείου

```
FILE *pF;    /* δήλωση ενός pointer σε μεταβλητή FILE */

pF = fopen("myfile.txt","r");     /* άνοιγμα αρχείου */

                    /* . . . Διάφορες λειτουργίες . . .*/

fclose(pF);
```

Επιστρεφόμενη τιμή: 'pointer-to-FILE' (ή NULL σε περίπτωση σφάλματος)

Άνοιγμα – κλείσιμο αρχείου

```
FILE *pF;/* δήλωση ενός pointer σε μεταβλητή FILE */

pF = fopen("myfile.txt","r"); /* άνοιγμα αρχείου */

                /* . . . Διάφορες λειτουργίες . . .*/

fclose(pF);
```

Κλείσιμο αρχείου: Επιστρέφει 0 όταν κλείνει σωστά ή EOF όταν υπάρχει σφάλμα

Άνοιγμα – κλείσιμο αρχείου

```
FILE *pF;    /* δήλωση ενός pointer σε μεταβλητή FILE */

pF = fopen("myfile.txt","r");     /* άνοιγμα αρχείου */

                /* . . . Διάφορες λειτουργίες . . .*/

fclose(pF);
```

Εάν θέλουμε ολόκληρη τη διαδρομή μέσα στο μέσο αποθήκευσης:
Π.χ. pF = fopen("c:\\teicm\\progrII\\myfile.txt","r");

Άνοιγμα – κλείσιμο αρχείου

- **Η *fopen* δεσμεύει τους απαραίτητους πόρους από το λειτουργικό σύστημα, δημιουργεί το κανάλι επικοινωνίας κι επιστρέφει στο πρόγραμμα που την κάλεσε ένα δείκτη, που δείχνει σε δομή τύπου FILE.**
- **Ο δείκτης, που δείχνει σε δομή τύπου FILE, είναι γνωστός ως διαχειριστής ή δείκτης αρχείου (file handler ή file descriptor). Χρησιμοποιείται για να κρατήσει την ταυτότητα του καναλιού που επιστρέφεται από την fopen.**
- **Η FILE ορίζεται στο <stdio.h>.**
- **Ένα από τα πεδία της δομής FILE είναι ο δείκτης θέσης αρχείου (file position indicator), ο οποίος δείχνει στο byte από όπου ο επόμενος χαρακτήρας πρόκειται να διαβασθεί ή όπου ο επόμενος χαρακτήρας πρόκειται να εγγραφεί.**

Άνοιγμα – κλείσιμο αρχείου

Παράμετροι προσδιορισμού του τρόπου πρόσβασης σε αρχεία κειμένου:

- **‘r’: άνοιγμα αρχείου για ανάγνωση. Ο δείκτης θέσης αρχείου βρίσκεται στην αρχή του κειμένου.**
- **‘w’: δημιουργία νέου αρχείου για εγγραφή. Εάν το αρχείο υπάρχει ήδη, το μέγεθός του θα μηδενισθεί και τα περιεχόμενα θα διαγραφούν. Ο δείκτης θέσης αρχείου τίθεται στην αρχή του αρχείου.**
- **‘a’: άνοιγμα υπάρχοντος αρχείου κειμένου, στο οποίο όμως μπορούμε να γράψουμε μόνο στο τέλος του αρχείου.**
- **‘r+’: άνοιγμα υπάρχοντος αρχείου κειμένου για ανάγνωση και εγγραφή. Ο δείκτης θέσης αρχείου τίθεται στην αρχή του αρχείου.**
- **‘w+’: δημιουργία νέου αρχείου για ανάγνωση και εγγραφή. Εάν το αρχείο υπάρχει ήδη, το μέγεθός του θα μηδενισθεί και τα περιεχόμενα θα διαγραφούν.**

Παράμετροι προσδιορισμού του τρόπου πρόσβασης σε αρχεία κειμένου (συνέχεια):

- **'a+': άνοιγμα υπάρχοντος αρχείου ή δημιουργία νέου σε append μορφή. Μπορούμε να διαβάσουμε δεδομένα από οποιοδήποτε σημείο του αρχείου, αλλά μπορούμε να γράψουμε δεδομένα μόνο στη θέση του δείκτη end-of-file.**

Οι προσδιοριστές για τα δυαδικά αρχεία μορφή είναι ίδιοι, με τη διαφορά ότι έχουν ένα *b* που τους ακολουθεί. Έτσι, για να ανοίξουμε ένα δυαδικό αρχείο και να διαβάσουμε, θα πρέπει να χρησιμοποιήσουμε τον προσδιοριστή *'rb'*.

Άνοιγμα – κλείσιμο αρχείου

Παρατηρήσεις:

1) Ο δείκτης ***FILE*** χειρίζεται κατά τρόπο αποκλειστικό το αρχείο

2) Στους δείκτες ***FILE*** δεν επιτρέπεται «αριθμητική δεικτών»!!!
Π.χ.

```
fclose(pF);     /* σωστό */
fclose(pF+1);   /* ΛΑΘΟΣ */
```

3) `fopen()`: δεσμεύει μνήμη. Εάν ξεχάσουμε να την απελευθερώσουμε με `fclose()` θα έχουμε **μεγάλη διαρροή μνήμης!!**

fprintf(): «Τύπωσε στο αρχείο»

- **f**ormatted **print** output, to **f**ile
- Ακριβώς οι ίδιοι μορφολογικοί κανόνες με εκείνους της ***printf()***

```
float flot;
int cnt, k;
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
char msg[40]="This is my song!\n";  /* string */

pF = fopen("testfile.txt","w");
cnt = fprintf(pF,"%s,yep!%d,%f,\n",msg,21,34.5);
fclose(pF);
```

Άνοιγμα αρχείου για εγγραφή

fprintf(): «Τύπωσε στο αρχείο»

```
int cnt;
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
char msg[40]="This is my song!\n";  /* string */

pF = fopen("testfile.txt","w");
cnt = fprintf(pF,"%s,yep!%d,%f,\n",msg,21,34.5);
fclose(pF);
```

Εγγραφή string, όπως ακριβώς στην *printf*

fprintf(): «Τύπωσε στο αρχείο»

```
int cnt;
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
char msg[40]="This is my song!\n";  /* string */

pF = fopen("testfile.txt","w");
cnt = fprintf(pF,"%s,yep!%d,%f,\n",msg,21,34.5);
fclose(pF);
```

Λίστα από pointers-to-items προς εγγραφή: string, int, float,…

fprintf(): «Τύπωσε στο αρχείο»

```
int cnt;
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
char msg[40]="This is my song!\n";  /* string */

pF = fopen("testfile.txt","w");
cnt = fprintf(pF,"%s,yep!%d,%f,\n",msg,21,34.5);
fclose(pF);
```

Επιστρεφόμενη τιμή:
Ο αριθμός των bytes που ενεγράφησαν στο αρχείο

fprintf(): «Τύπωσε στο αρχείο»

```
int cnt;
FILE *pF;          /* declare a pointer-to-FILE */
char msg[40]="This is my song!\n";  /* string */

pF = fopen("testfile.txt","w");
cnt = fprintf(pF,"%s,yep!%d,%f,\n",msg,21,34.5);
fclose(pF);
```

Τέλος εργασιών, κλείσιμο του αρχείου

fscanf(): «Διάβασε κείμενο από αρχείο»

- **f**ormatted **scan**ned input, from **f**ile
- Ακριβώς οι ίδιοι μορφολογικοί κανόνες με εκείνους της *scanf()*

```
float flot;
int cnt, k;
FILE *pF;  /* δήλωση ενός pointer-to-FILE */
char msg[40];
pF = fopen("testfile.txt","r");assert(pF!=NULL);
cnt = fscanf(pF,"%s %d %f",msg,&k,&flot);
fclose(pF);
```

Άνοιγμα αρχείου για ανάγνωση

fscanf(): «Διάβασε κείμενο από αρχείο»

```
float flot;
int cnt, k;
FILE *pF;  /* δήλωση ενός pointer-to-FILE */
char msg[40];

pF = fopen("testfile.txt","r"); assert(pF!=NULL);
cnt = fscanf(pF,"%s %d %f",msg,&k,&flot);
fclose(pF);
```

pointer-to-FILE

fscanf(): «Διάβασε κείμενο από αρχείο»

```
float flot;
int cnt, k;
FILE *pF;            /* δήλωση ενός pointer-to-FILE */
char msg[40];

pF = fopen("testfile.txt","r");assert(pF!=NULL);
cnt = fscanf(pF,"%s %d %f",msg,&k,&flot);
fclose(pF);
```

Ανάγνωση string, όπως ακριβώς στην *scanf*, χωρίς &

fscanf(): «Διάβασε κείμενο από αρχείο»

```
float flot;
int cnt, k;
FILE *pF;     /* δήλωση ενός pointer-to-FILE */
char msg[40];

pF = fopen("testfile.txt","r");assert(pF!=NULL);
cnt = fscanf(pF,"%s %d %f",msg,&k,&flot);
fclose(pF);
```

Λίστα από pointers-to-items προς ανάγνωση: int, float,…

fscanf(): «Διάβασε κείμενο από αρχείο»

```
float flot;
int cnt, k;
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
char msg[40];

pF = fopen("testfile.txt","r");assert(pF!=NULL);
cnt = fscanf(pF,"%s %d %f",msg,&k,&flot);
fclose(pF);
```

Επιστρεφόμενη τιμή: Ο αριθμός των στοιχείων που ανεγνώσθησαν. Σε περίπτωση σφάλματος θα επιστραφεί το μηδέν ή το EOF

fscanf(): «Διάβασε κείμενο από αρχείο»

```
float flot;
int cnt, k;
FILE *pF;  /* δήλωση ενός pointer-to-FILE */
char msg[40];

pF = fopen("testfile.txt","r");
cnt = fscanf(pF,"%s %d %f",msg,&k,&flot);
fclose(pF);
```

Τέλος εργασιών, κλείσιμο του αρχείου

Προσοχή: Παρόλο που η χρήση των *fprintf()*, *fscanf()* είναι συχνά ο πιο εύκολος τρόπος για να γράφουμε ή να διαβάζουμε μία συλλογή δεδομένων σε ένα αρχείο δίσκου, δεν είναι πάντοτε και ο πιο αποτελεσματικός. Επειδή γράφουμε φορμαρισμένα δεδομένα ASCII, όπως θα εμφανίζονταν στην οθόνη κι όχι δυαδικά, κάνουμε περισσότερα πράγματα σε κάθε κλήση και καταλαμβάνουμε περισσότερο χώρο. Έτσι, εάν μας ενδιαφέρει η ταχύτητα ή το μέγεθος του αρχείου, θα πρέπει πιθανώς να χρησιμοποιήσουμε τις συναρτήσεις *fread()* και *fwrite()*.

Ανάγνωση αρχείου: fread()

- Αντιγράφει bytes από το **αρχείο** στη μνήμη
- **Θα πρέπει να έχετε εξασφαλίσει** επαρκή μνήμη!

```
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
int buf[40];       // προσωρινή αποθήκευση
int n,cnt;
                   /* άνοιγμα αρχείου για ανάγνωση */
   pF = fopen("myfile.qzk","r");
   if (pF == NULL) exit(-1);       /*(σφάλμα!) */
   n = 32;
   cnt = fread(buf, sizeof(int), n, pF);
```

Άνοιγμα αρχείου για ανάγνωση
(σε περίπτωση σφάλματος: έξοδος)
Εναλλακτικά: `assert(pF!=NULL)`

Ανάγνωση αρχείου: fread()

```
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
int buf[40];
int n,cnt;
                   /* άνοιγμα αρχείου για ανάγνωση */
   pF = fopen("myfile.qzk","r");
   if (pF==NULL) exit(-1);          /*(σφάλμα!) */
   n = 32;
   cnt = fread(buf, sizeof(int), n, pF);
```

Δείκτης που δείχνει στον buffer. Διατηρεί τουλάχιστον 'n' στοιχεία οιουδήποτε είδους. (στην παρούσα περίπτωση ακέραιοι)

Ανάγνωση αρχείου: fread()

```
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
int buf[40];
int n,cnt;
            /* άνοιγμα αρχείου για ανάγνωση    */
   pF = fopen("myfile.qzk","r");
   if (pF==NULL) exit(-1);          /*(σφάλμα!) */
   n = 32;
   cnt = fread(buf, sizeof(int), n, pF);
```

το μέγεθος των προς ανάγνωση στοιχείων (σε bytes)
(στην παρούσα περίπτωση ακέραιοι)

Ανάγνωση αρχείου: fread()

```
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
int buf[40];
int n,cnt;
                   /* άνοιγμα αρχείου προς ανάγνωση */
   pF = fopen("myfile.qzk","r");
   if (pF==NULL) exit(-1);          /*(σφάλμα!) */
   n = 32;
   cnt = fread(buf, sizeof(int), n, pF);
```

Ο αριθμός των προς ανάγνωση στοιχείων

Ανάγνωση αρχείου: fread()

```
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
int buf[40];
int n,cnt;
                   /* άνοιγμα αρχείου για ανάγνωση   */
   pF = fopen("myfile.qzk","r");
   if (pF==NULL) exit(-1);          /*(σφάλμα!) */
   n = 32;
   cnt = fread(buf, sizeof(int), n, pF);
```

pointer-to-FILE που δείχνει στο προς ανάγνωση αρχείο

Ανάγνωση αρχείου: fread()

```
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
int buf[40];
int n,cnt;
                   /* άνοιγμα αρχείου για ανάγνωση   */
   pF = fopen("myfile.qzk","r");
   if (pF==NULL) exit(-1);          /*(σφάλμα!) */
   n = 32;
   cnt = fread(buf, sizeof(int), n, pF);
   if (cnt!=n) return(-1);          /*(σφάλμα!) */
```

Επιστρεφόμενη τιμή: Αριθμός στοιχείων που ανεγνώσθησαν επιτυχώς

Εγγραφή σε αρχείο: fwrite()

- Αντιγράφει bytes από τη μνήμη στο **αρχείο**
- Η σύνταξη και η χρήση είναι **αντίστοιχες** με την *fread()* !

```
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
int buf[40];
int n,cnt;
                   /* άνοιγμα αρχείου για εγγραφή   */
   pF = fopen("myfile.qzk","w");

   n = 32;
   cnt = fwrite(buf, sizeof(int), n, pF);
   if(cnt!=n) exit(-1);            /*(σφάλμα!) */
```

Άνοιγμα αρχείου για εγγραφή
(σε περίπτωση σφάλματος, έξοδος)

Εγγραφή σε αρχείο: fwrite()

```
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
int buf[40];
int n,cnt;
                   /* άνοιγμα αρχείου για εγγραφή */
   pF = fopen("myfile.qzk","w");

   n = 32;
   cnt = fwrite(buf, sizeof(int), n, pF);
   if (cnt!=n) exit(-1);            /*(σφάλμα!) */
```

Δείκτης που δείχνει στον buffer. Διατηρεί τουλάχιστον 'n' στοιχεία οιουδήποτε είδους. (στην παρούσα περίπτωση ακέραιοι)

Εγγραφή σε αρχείο: fwrite()

```
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
int buf[40];
int n,cnt;
                   /* άνοιγμα αρχείου για εγγραφή */
   pF = fopen("myfile.qzk","w");

   n = 32;
   cnt = fwrite(buf, sizeof(int), n, pF);
   if (cnt!=n) exit(-1);          /*(σφάλμα!) */
```

το μέγεθος των προς εγγραφή στοιχείων (σε bytes)
(στην παρούσα περίπτωση ακέραιοι)

Εγγραφή σε αρχείο: fwrite()

```
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
int buf[40];
int n,cnt;
                   /* άνοιγμα αρχείου για εγγραφή */
   pF = fopen("myfile.qzk","w");

   n = 32;
   cnt = fwrite(buf, sizeof(int), n, pF);
   if (cnt!=n) exit(-1);            /*(σφάλμα!) */
```

Ο αριθμός των προς εγγραφή στοιχείων

Εγγραφή σε αρχείο: fwrite()

```
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
int buf[40];
int n,cnt;
                   /* άνοιγμα αρχείου για εγγραφή */
   pF = fopen("myfile.qzk","w");

   n = 32;
   cnt = fwrite(buf, sizeof(int), n, pF);
   if (cnt!=n) exit(-1);              /* (σφάλμα!) */
```

pointer-to-FILE που δείχνει στο προς εγγραφή αρχείο

Εγγραφή σε αρχείο: fwrite()

```
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
int buf[40];
int n,cnt;
                   /* άνοιγμα αρχείου για εγγραφή */
   pF = fopen("myfile.qzk","w");

   n = 32;
   cnt = fwrite(buf, sizeof(int), n, pF);
   if (cnt!=n) exit(-1);          /*(σφάλμα!) */
```

Επιστρεφόμενη τιμή: Αριθμός στοιχείων που ενεγράφησαν επιτυχώς

Αλφαριθμητικά και fwrite()

Σφάλμα: αμελούμε να γράψουμε τον τερματιστή του string

– Θυμηθείτε ότι η **strlen(msg)** **ΔΕ** μετρά το μηδενικό χαρακτήρα '\0', ο οποίος τελειώνει το αλφαριθμητικό!!

```
int cnt;
FILE *pF;          /* δήλωση ενός pointer-to-FILE */
char msg[40]="This is my song!\n";  /* string */

  pF = fopen("music.mdi","w");
  cnt= fwrite(msg, sizeof(char), strlen(msg),pF);
```

ΣΦΑΛΜΑ! - ατελής εγγραφή string -
Δε γράφει στο αρχείο το χαρακτήρα τερματισμού

Αλφαριθμητικά και fwrite()

```
int cnt;
FILE *pF;     /* δήλωση ενός pointer-to-FILE */
char msg[40]="This is my song!\n"; /* string */

  pF = fopen("music.mdi","w");
  cnt= fwrite(msg, sizeof(char),1+strlen(msg),pF);
```

Το σφάλμα διορθώνεται με την προσθήκη μίας μονάδας στη strlen(), ώστε να περιληφθεί ο χαρακτήρας τερματισμού του αλφαριθμητικού

Αλφαριθμητικά και fread()

Αρχείο κειμένου (text file) == Αρχείο αποτελούμενο αποκλειστικά από αλφαριθμητικά

- Τα αρχεία κειμένου είναι πιο επίφοβα από τα αριθμητικά αρχεία
  - Ποιο είναι το μέγιστο μήκος αλφαριθμητικού μέσα στο αρχείο;
  - Δεν το γνωρίζουμε έως ότου διαβάσουμε το αρχείο

Οι τροποποιήσεις των `scanf`, `printf` επιτελούν όλο το έργο!

putc(): «Τύπωσε χαρακτήρα στο αρχείο»

- **Πρωτότυπο της συνάρτησης:**

int putc(int ch, FILE *pF);

όπου pF ο δείκτης αρχείου που επιστρέφεται από την fopen και ch είναι ο προς εγγραφή χαρακτήρας. Για ιστορικούς λόγους ο ch ονομάζεται int αλλά χρησιμοποιεί μόνο ένα byte, το byte χαμηλής τάξης (*ανοίξτε το stdio.h για να το επιβεβαιώσετε*).

- **Εάν η λειτουργία της συνάρτησης επιτύχει, επιστρέφεται ο χαρακτήρας που γράφτηκε. Αν αποτύχει, θα επιστρέψει το EOF (End Of File, *τέλος αρχείου*, ακέραιος με τιμή -1).**

getc(): «Διάβασε χαρακτήρα από αρχείο»

•Είναι συμπληρωματική της putc(). Πρωτότυπο της συνάρτησης:

```
int getc(FILE *pF);
```

όπου pF ο δείκτης αρχείου που επιστρέφεται από την fopen. Για ιστορικούς λόγους η getc() επιστρέφει έναν ακέραιο αλλά τα bytes υψηλής τάξης είναι μηδέν, άρα μόνο το byte χαμηλής τάξης περιέχει πληροφορία.

•Η συνάρτηση επιστρέφει ένα EOF όταν ο υπολογιστής φθάσει στο τέλος του αρχείου. Έτσι, για να διαβάσουμε ένα αρχείο κειμένου έως το σημάδι τέλους αρχείου, μπορούμε να χρησιμοποιήσουμε τον ακόλουθο κώδικα:

```
ch = getc(pF);
while (ch!=EOF){ ch=getc(pF);}
```

Ανάγνωση/εγγραφή ενός χαρακτήρα

```
FILE *pFin,*pFout;
char k;

pFin  = fopen("src.txt","r");
pFout = fopen("dest.txt","w");
if (pFin == NULL) return(-1);
k = getc(pFin); /* διάβασε τον πρώτο χαρακτήρα ως int */
while(k!=EOF)
{
    putc(k, pFout);
    k = getc(pFin);
}
fclose(pFin); fclose(pFout);
```

Ανοίγει ένα αρχείο για ανάγνωση (**r**ead) κι ένα για εγγραφή (**w**rite)

Ανάγνωση/εγγραφή ενός χαρακτήρα

```
FILE *pFin,*pFout;
char k;

pFin  = fopen("src.txt","r");
pFout = fopen("dest.txt","w");
if (pFin == NULL) return(-1);
k = getc(pFin); /* διάβασε τον πρώτο χαρακτήρα ως int */
while(k!=EOF)
{
    putc(k, pFout);
    k = getc(pFin);
}
fclose(pFin); fclose(pFout);
```

Διαβάζει τον πρώτο χαρακτήρα. Η **getc()** επιστρέφει ***int*** ώστε να μπορεί να γίνουν έλεγχος για **EOF (EOF=-1)**

Ανάγνωση/εγγραφή ενός χαρακτήρα

```
FILE *pFin,*pFout;
char k;

pFin  = fopen("src.txt","r");
pFout = fopen("dest.txt","w");
if (pFin == NULL) return(-1);
k = getc(pFin); /* διάβασε τον πρώτο χαρακτήρα ως int */
while(k!=EOF)
{
    putc(k, pFout);
    k = getc(pFin);
}
fclose(pFin); fclose(pFout);
```

Εάν ο **pFin** έχει ένα **char**, ο οποίος ΔΕΝ ΕΙΝΑΙ **EOF**, τότε …

Ανάγνωση/εγγραφή ενός χαρακτήρα

```
FILE *pFin,*pFout;
char k;

pFin  = fopen("src.txt","r");
pFout = fopen("dest.txt","w");
if (pFin == pFin) return(-1);
k = getc(pFin); /* διάβασε τον πρώτο χαρακτήρα ως int */
while(k!=EOF)
{
    putc(k, pFout);
    k = getc(pFin);
}
fclose(pFin); fclose(pFout);
```

… Τότε να αντιγραφεί στο **pFout** και να ληφθεί ο επόμενος **char**, …

Ανάγνωση/εγγραφή ενός χαρακτήρα

```
FILE *pFin,*pFout;
char k;

pFin  = fopen("src.txt","r");
pFout = fopen("dest.txt","w");
if (pFin == NULL) return(-1);
k = getc(pFin); /* διάβασε τον πρώτο χαρακτήρα ως int */
while(k!=EOF)
{
    putc(k, pFout);
    k = getc(pFin);
}
fclose(pFin); fclose(pFout);
```

Όταν τελικά βρεθεί **EOF**, περατώνεται η διαδικασία: κλείνουν τα αρχεία.

Ανάγνωση/εγγραφή ενός χαρακτήρα

Τα προηγούμενα μπορούν να επιτευχθούν με τις **fread/fwrite**, όχι μόνο με ένα χαρακτήρα αλλά **με οιονδήποτε αριθμό χαρακτήρων**!

```
FILE *pFin,*pFout;
char buf[100];                          /* buffer χαρακτήρων */
int cnt;

pFin  = fopen("src.txt","r");
pFout = fopen("dest.txt","w");
if (pFin==NULL) return(-1); /*ERR!*/
cnt = fread(buf,sizeof(char),100,pFin);
while (cnt == 100)
{
    fwrite(buf,sizeof(char),100,pFout);
    cnt=fread (buf,sizeof(char),100,pFin );
}
if(cnt!=0) fwrite(buf,sizeof(char),cnt,pFout);
fclose(pFin); fclose(pFout);
```

Ανοίγει ένα αρχείο για ανάγνωση (**r**ead) κι ένα για εγγραφή (**w**rite)

//fread_fwrite_chars

Ανάγνωση/εγγραφή ενός χαρακτήρα

```
FILE *pFin,*pFout;
char buf[100];                    /* buffer χαρακτήρων */
int cnt;

pFin  = fopen("src.txt","r");
pFout = fopen("dest.txt","w");
if (pFin==NULL) return(-1);
cnt = fread(buf,sizeof(char),100,pFin);
while (cnt == 100)
{
    fwrite(buf,sizeof(char),100,pFout);
    cnt=fread (buf,sizeof(char),100,pFin );
}
if(cnt!=0) fwrite(buf,sizeof(char),cnt,pFout);
fclose(pFin); fclose(pFout);
```

Η **fread()** επιστρέφει τον αριθμό των χαρακτήρων που ανεγνώσθησαν.

Ανάγνωση/εγγραφή ενός χαρακτήρα

```
FILE *pFin,*pFout;
char buf[100];                    /* buffer χαρακτήρων */
int cnt;

pFin  = fopen("src.txt","r");
pFout = fopen("dest.txt","w");
if (pFin==NULL) return(-1);
cnt = fread(buf,sizeof(char),100,pFin);
while (cnt == 100)
{
    fwrite(buf,sizeof(char),100,pFout);
    cnt=fread (buf,sizeof(char),100,pFin );
}
if(cnt!=0) fwrite(buf,sizeof(char),cnt,pFout);
fclose(pFin); fclose(pFout);
```

Εάν γεμίσει ο buffer, τότε, …

Ανάγνωση/εγγραφή ενός χαρακτήρα

```
FILE *pFin,*pFout;
char buf[100]; /* buffer χαρακτήρων */
int cnt;

pFin  = fopen("src.txt","r");
pFout = fopen("dest.txt","w");
if (pFin==NULL) return(-1);
cnt = fread(buf,sizeof(char),100,pFin);
while (cnt == 100)
{
    fwrite(buf,sizeof(char),100,pFout);
    cnt=fread (buf,sizeof(char),100,pFin );
}
if(cnt!=0) fwrite(buf,sizeof(char),cnt,pFout);
fclose(pFin); fclose(pFout);
```

…, Τότε γράφουμε **ολόκληρο το buffer** στο αρχείο εγγραφής και ακολούθως επαναλαμβάνουμε την ανάγνωση …

Ανάγνωση/εγγραφή ενός χαρακτήρα

```
FILE *pFin,*pFout;
char buf[100];                    /* buffer χαρακτήρων */
int cnt;

pFin  = fopen("src.txt","r");
pFout = fopen("dest.txt","w");
if (pFin==NULL) return(-1);
cnt = fread(buf,sizeof(char),100,pFin);
while (cnt == 100)
{
        fwrite(buf,sizeof(char),100,pFout);
    cnt=fread (buf,sizeof(char),100,pFin );
}
if(cnt!=0) fwrite(buf,sizeof(char),cnt,pFout);
fclose(pFin); fclose(pFout);
```

*Γράφει τα **περιεχόμενα του buffer** που μπορεί να παρέμειναν*

Ανάγνωση/εγγραφή ενός χαρακτήρα

```
FILE *pFin,*pFout;
char buf[100];                /* char buffer    */
int cnt;

pFin  = fopen("src.txt","r");
pFout = fopen("dest.txt","w");
if (pFin==NULL) return(-1);
cnt = fread(buf,sizeof(char),100,pFin);
while (cnt == 100)
{
        fwrite(buf,sizeof(char),100,pFout);
    cnt=fread (buf,sizeof(char),100,pFin );
}
if(cnt!=0) fwrite(buf,sizeof(char),cnt,pFout);
fclose(pFin); fclose(pFout);
```

Τέλος εργασιών, κλείσιμο των αρχείων

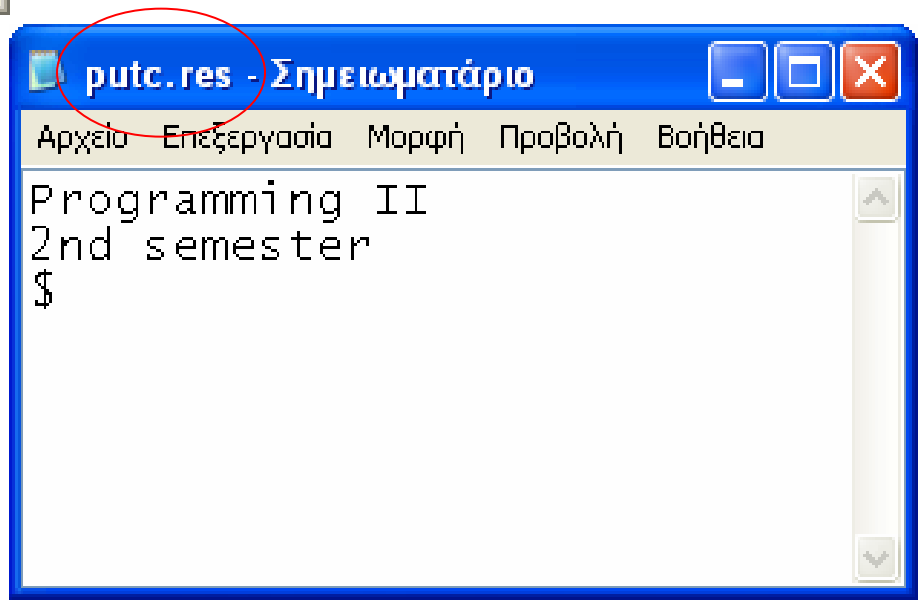
Παράδειγμα: **Να καταστρωθεί πρόγραμμα , το οποίο διαβάζει χαρακτήρες από το πληκτρολόγιο και τους γράφει σε αρχείο, έως ότου πληκτρολογήσουμε το σύμβολο του δολαρίου ($).**

```
#include<stdio.h>
main() {
  FILE *pF; char ch;
  pF=fopen( "putc.res","w" );
    do {
      ch=getchar();
      putc(ch,pF);
    } //end of do
    while (ch!='$');
  fclose( pF );
} //end of main
```

//example_of_putc

Αποτέλεσμα:


C:\temp\Project1.exe
Start writting:
Programming II
2nd semester
$


putc.res - Σημειωματάριο
Αρχείο Επεξεργασία Μορφή Προβολή Βοήθεια
Programming II
2nd semester
$

Συμπληρωματικό παράδειγμα: **Να καταστρωθεί πρόγραμμα , το οποίο διαβάζει οποιοδήποτε αρχείο ASCII και εμφανίζει το περιεχόμενό του στην οθόνη.**

```
#include<stdio.h>
main() {
  FILE *pF;
  char ch;
  pF=fopen( "putc.res","r" ); //Produced in the example of putc
  if (pF==NULL) printf( "\t\tFILE ERROR: Exit program\n" );
  else {
    printf( "\t\t\tPRESS ANY KEY TO START\n" ); ch=getc(pF);
    while (ch!=EOF) {
      putchar(ch);
      ch=getc(pF);
    } //end of while
  } //end of else
  fclose( pF );
} //end of main
```

//example_of_getc

Αποτέλεσμα:

putc.res - Σημειωματάριο

Αρχείο Επεξεργασία Μορφή Προβολή Βοήθεια

```
Programming II
2nd semester
$
```

C:\temp\Project1.exe

```
                                PRESS ANY KEY TO START


Programming II
2nd semester
$
                                PRESS ANY KEY TO FINISH_
```

Παράδειγμα:

- **Να καταστρωθεί πρόγραμμα , το οποίο να δίνει τον αριθμό των λέξεων που περιέχονται σε ένα αρχείο ASCII.**
- **Το πρόγραμμα θα πρέπει να χειρίζεται τους λευκούς χαρακτήρες (κενά, νέες γραμμές, στηλοθέτες) ως πραγματικούς χαρακτήρες. Δηλαδή, εάν υπάρχει μία συμβολοσειρά από κενά ή χαρακτήρες επιστροφής, το πρόγραμμα τους διαβάζει και αναμένει για τον πρώτο πραγματικό (μη λευκό) χαρακτήρα. Όλο αυτό το μετρά ως λέξη. Κατόπιν διαβάζει τους πραγματικούς χαρακτήρες έως την εμφάνιση του επόμενου λευκού χαρακτήρα.**
- **Μία μεταβλητή (σημαία) θα πρέπει να ελέγχει κατά πόσον το πρόγραμμα βρίσκεται στο μέσον μίας λέξης ή στο μέσον κάποιου κενού.**

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h> //για την exit()

main()
{
  FILE *fptr;
  char ch,string[81];
  int white=1;    //σημαία λευκού χαρακτήρα
  int count=0;    //μετρητής λέξεων

  if ((fptr=fopen("file.txt","r")) == NULL)
  {
    printf( "ERROR: can't open file" );
    exit(1);
  } //end of if
```

//word_counter

```
while ((ch=getc(fptr))!=EOF) //ανάγνωση χαρακτήρων έως EOF
{
  switch(ch)
  {
    case ' ':          //έλεγχος για λευκούς χαρακτήρες
    case '\t':
    case '\n':
      white++;
      break;
    default:  //μη λευκοί χαρακτήρες, μέτρηση λέξεων
      if (white)
      {
        white=0;
        count++;
      } //end of if
    break;
  } //end of switch
}  //end of while
fclose( fptr );
printf( "The file contains %d words\n",count );
} //end of main
```

Κοινή έξοδος για τις τρεις *case*

Αποτέλεσμα:

file.txt - Σημειωματάριο

Αρχείο Επεξεργασία Μορφή Προβολή Βοήθεια

```
TEI of Serres
Deprtament of Inoformatics and Communications
Course: Programming II
```

C:\temp\Project1.exe

```
The file contains 11 words
                    PRESS ANY KEY TO FINISH
```

Ανάγνωση/εγγραφή γραμμή ανά γραμμή

```
char buf[100];
FILE *pFin,*pFout;
. . .

    while(fgets(buf,100,pFin) != NULL)
    {
        fputs(buf,pFout);
    }
```

Η κλήση **rtn = fgets(pBuf,max,pFile)**:

- θα διαβάσει ένα αλφαριθμητικό από το **pFile** και θα το αποδώσει στο **buf**,
- θα σταματήσει μετά τη ***νέα-γραμμή*** **'\n'** ή τον 99ο **char**
- θα τοποθετήσει ακολούθως στο **buf** το ***null*** **'\0'**
- θα επιστρέψει **pBuf** σε περίπτωση επιτυχίας ή ***NULL*** εάν ο **pFile** άδειος

Ανάγνωση/εγγραφή γραμμή ανά γραμμή

Η ίδια εργασία γίνεται με τις **fscanf()**, **fprintf()**:

```
char buf[100];
FILE *pFin,*pFout;
int cnt;
. . .
    cnt = fscanf(pFin," %99[^\n]",buf);
    while ((cnt!=EOF) && (cnt!=0))
    {
        fprintf(pFout,"%s\n",buf);
        cnt = fscanf(pFin," %99[^\n]",buf);
    }
```

Η κλήση **rtn=fscanf(pFile,"%99[^\n]",buf)**:

- θα διαβάσει ένα αλφαριθμητικό από το **pFile** και θα το αποδώσει στο **buf**,
- θα σταματήσει μετά τη ***νέα-γραμμή*** **'\n'** ή τον 99ο **char**
- θα τοποθετήσει ακολούθως στο **buf** το ***null*** **'\0'**
- θα επιστρέψει τον αριθμό των μετατροπών ή το **EOF** όταν φθάσει στο τέλος του **pFile**.

Ανάγνωση/εγγραφή γραμμή ανά γραμμή

Η ίδια εργασία γίνεται με τις **fscanf(), fprintf()**:

```
char buf[100];
FILE *pFin,*pFout;
int cnt;
. . .
    cnt = fscanf(pFin," %99[^\n\0Q]",buf);
    while ((cnt!=EOF) && (cnt!=0))
    {
        fprintf(pFout,"%s\n",buf);
        cnt = fscanf(pFin," %99[^\n]",buf);
    }
```

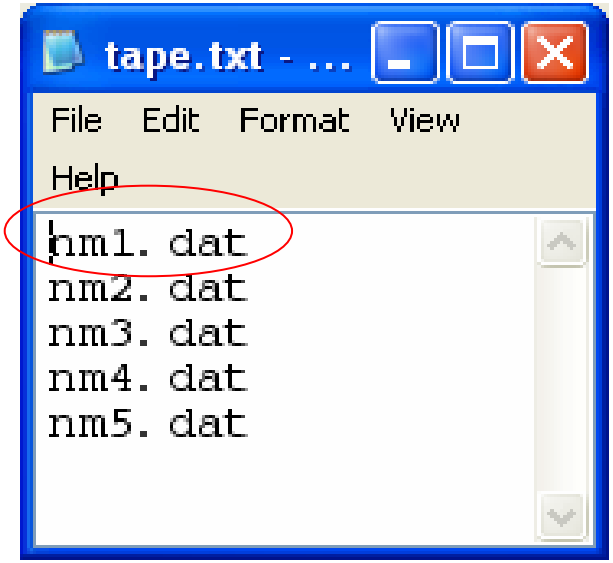
Τα περιεχόμενα γραμμής ορίζονται από τον προγραμματιστή: Η γραμμή τελειώνει είτε μετά την πρώτη εμφάνιση των χαρακτήρων **\n**, **\0**, **Q** ή όταν φθάσουμε στον **99**ο χαρακτήρα.

Ανάγνωση/εγγραφή γραμμή ανά γραμμή

```
char buf[100];
FILE *pFin,*pFout;
int cnt;
. . .
    cnt = fscanf(pFin," %99[^\n\0Q]",buf);
    while ((cnt!=EOF) && (cnt!=0))
    {
        fprintf(pFout,"%s\n",buf);
        cnt = fscanf(pFin," %99[^\n]",buf);
    }
```

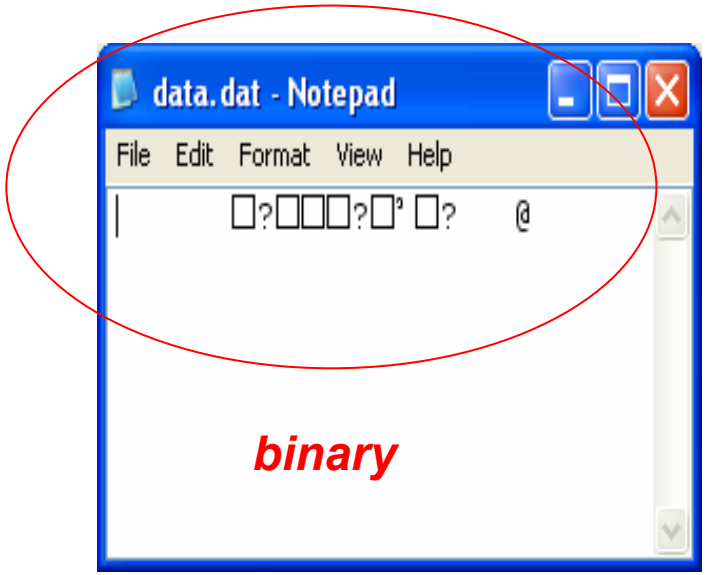
Η μορφή του αρχείου εξόδου ορίζεται από τον προγραμματιστή: Θέσε ένα **\n** στο τέλος κάθε γραμμής.

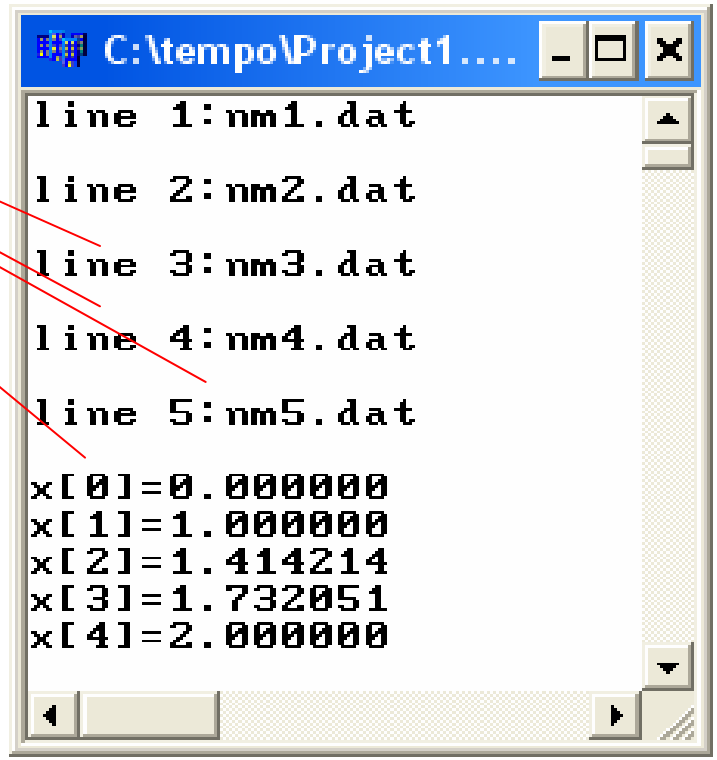
```
#include<conio.h>
#include<stdio.h>
#include<math.h>
#define name "tape.txt"
main()
{
  FILE *f1;
  char pchar[16];    int i;    float x[5];

  f1=fopen(name,"w");
  for (i=0;i<5;i++) fprintf(f1,"nm%d.dat\n",i+1);
  fclose(f1);

  f1=fopen(name,"r");   //fgets(): ανάγνωση γραμμή προς γραμμή
  for (i=0;i<5;i++) printf("line %d:%s\n",i+1,fgets(pchar,15,f1));
  fclose(f1);
  f1=fopen("data.dat","w");
  for (i=0;i<5;i++) x[i]=sqrt(i);
  fwrite(x,sizeof(x),1,f1);
  fclose(f1);   //Το f1 γίνεται δυαδικό αν και δεν μπήκε το "wb"
```



```
f1=fopen("data.dat","rb");
  for (i=0;i<5;i++)
  {
    fread(&x[i],sizeof(float),1,f1);
    printf("x[%d]=%f\n",i,x[i]);
  }
  fclose(f1);
}
```

Επιπλέον νέα γραμμή





Κατασκευή αλφαριθμητικών

Εάν θέλουμε το πρόγραμμα να κατασκευάζει αλφαριθμητικά με μεικτή χρήση λέξεων κι αριθμών και να είναι περισσότερο ευέλικτο σε ό,τι αφορά τις λειτουργίες εισόδου/εξόδου, τότε χρησιμοποιούμε τις ***sscanf()*** και ***sprintf()***. Αποτελούν τις αντίστοιχες των ***scanf()*** και ***sprintf()*** για την περίπτωση των **strings**.

Τυχαία προσπέλαση δυαδικού αρχείου

Σειριακή προσπέλαση: για να βρεθεί ένα δεδομένο σε ένα αρχείο πρέπει να προσπελασθούν πρώτα όλα τα προηγούμενά του. Επιπρόσθετα, για να ενημερωθεί μία εγγραφή του αρχείου (π.χ. ένα όνομα), πρέπει να διαβασθεί όλο το αρχείο, να γίνει η αλλαγή και κατόπιν να ξαναγραφεί.

Η σειριακή προσπέλαση δεν ενδείκνυται σε μεγάλα αρχεία ή σε αρχεία που προσπελαύνονται και τροποποιούνται συχνά. Για αυτές τις περιπτώσεις η C παρέχει τη δυνατότητα τυχαίας προσπέλασης (random access), με την οποία υπάρχει πρόσβαση σε οιοδήποτε σημείο ενός αρχείου.

Τυχαία προσπέλαση δυαδικού αρχείου

Η τυχαία προσπέλαση στηρίζεται στο γεγονός ότι κάθε ανοικτό αρχείο έχει έναν **δείκτη θέσης αρχείου**, ο οποίος καθορίζει σε ποιο σημείο του αρχείου θα γίνει ανάγνωση ή εγγραφή. Η θέση αυτή δίνεται ως αριθμός bytes από την αρχή του αρχείου και είναι μία μεταβλητή της δομής FILE. Όταν το αρχείο ανοίγει για ανάγνωση η θέση αυτή είναι **0**. Όταν ανοίγει για προσάρτηση είναι το τέλος του αρχείου. Καθορίζοντας το δείκτη θέσης αρχείου μπορούμε να έχουμε προσπέλαση σε οιοδήποτε σημείο του αρχείου. Αυτή είναι η έννοια της τυχαίας προσπέλασης.

Η συνάρτηση fseek()

Η συνάρτηση ***fseek()*** αποτελεί το εργαλείο για την εκτέλεση λειτουργιών τυχαίας ανάγνωσης και εγγραφής. Ορίζεται στο ***stdio.h*** και έχει το ακόλουθο πρωτότυπο:

int fseek(FILE *fptr, long offset, int origin)

όπου

1) ***fptr*** είναι ένας δείκτης αρχείου που επιστρέφεται από την ***fopen()***.
2) ***offset*** είναι ο αριθμός των bytes που δηλώνουν την απόσταση της νέας θέσης από το ***origin***.
3) ***origin*** είναι το σημείο αφετηρίας και μπορεί να έχει μία από τις τρεις ακόλουθες τιμές:

Αφετηρία	**Όνομα**
αρχή του αρχείου	**SEEK_SET (0)**
τρέχουσα θέση	**SEEK_CUR (1)**
τέλος αρχείου	**SEEK_END (2)**

Η συνάρτηση fseek()

Για να βρεθεί π.χ. το **offset** από την τρέχουσα θέση του, το **origin** θα πρέπει να λάβει την τιμή **SEEK_CUR**. Σε περίπτωση επιτυχίας η ***fseek()*** επιστρέφει **0**, ενώ εάν αποτύχει επιστρέφει **μη μηδενική** τιμή.

Θα πρέπει να σημειωθεί ότι ο δείκτης θέσης αρχείου επανατοποθετείται στην αρχή με τη συνάρτηση

rewind(fptr);

Παράδειγμα:

Για τη διαχείριση των στοιχείων των φοιτητών ορίζεται ο πίνακας ***student_list[size]*** με στοιχεία τύπου δομής ***Student***:

```
typedef struct
{
        int AM;
        int year;
        char firstname[15];
        char lastname[30];
}       Student;
```

Για τη διαχείριση μεμονωμένων φοιτητών ορίζεται η μεταβλητή ***svar***, επίσης τύπου δομής ***Student***.

Ζητείται να γραφούν συναρτήσεις που να επιτελούν τα ακόλουθα:

save_data(): Αποθήκευση των δεδομένων του πίνακα ***student_list*** στο αρχείο ***students.dat***.

read_data(): Ανάγνωση των δεδομένων από το αρχείο και αποθήκευσή τους στον πίνακα ***student_list***.

read_student(): Προσπέλαση ενός συγκεκριμένου φοιτητή στο αρχείο και αποθήκευση των στοιχείων του σε μία μεταβλητή τύπου ***Student***.

save_student(): Αποθήκευση των στοιχείων ενός συγκεκριμένου φοιτητή στο αρχείο.

save_data()

Η συνάρτηση καλείται στη ***main()*** ως εξής:

```
save_data(fptr, student_list);
```

και έχει το ακόλουθο σώμα:

```
void save_data( FILE *fp, Student  list[ ] )  //ή Student
*list
{
        int i;
        rewind(fp);
        for ( i=0; i<size; i++ )
                fwrite(&list[i],sizeof(Student),1,fp);
}
```

Αντί του βρόχου for θα μπορούσε να γραφεί:

```
fwrite(list,sizeof(Student),size,fp);
```

read_data()

Η συνάρτηση καλείται στη ***main()*** ως εξής:

```
read_data(fptr, student_list);
```

και έχει το ακόλουθο σώμα:

```
void read_data( FILE *fp, Student *list )
{
        rewind(fp);
        fread(list,sizeof(Student),size,fp);
}
```

read_student()

Η συνάρτηση καλείται στη ***main()*** ως εξής:

read_student(fptr, &svar, i);

όπου ***svar*** είναι μία μεταβλητή τύπου ***Student*** (στη θέση της θα μπορούσε να είναι η ***&Student_list[i]***), ***i*** είναι η θέση του στον πίνακα ***student_list***. Το σώμα της είναι το ακόλουθο:

```
void read_student( FILE *fp, Student *s, int k )
{
        fseek(fp,k*sizeof(Student),SEEK_SET);
        fread(s,sizeof(Student),1,fp);
}
```

save_student()

Η συνάρτηση καλείται στη ***main()*** ως εξής:

```
save_student(fptr, &student_list[i], i);
```

όπου ***i*** είναι ο αύξων αριθμός του στον πίνακα ***student_list***. Το σώμα της είναι το ακόλουθο:

```
void save_student( FILE *fp, Student *s, int k )
{
        fseek(fp,k*sizeof(Student),SEEK_SET);
        fwrite(s,sizeof(Student),1,fp);
}
```

Η συνάρτηση ***save_student()*** χρησιμοποιείται όταν έχουμε κάνει αλλαγές στα στοιχεία ενός φοιτητή και θέλουμε να ενημερώσουμε την εγγραφή του στο αρχείο.

Η συνάρτηση ***read_student()*** χρησιμοποιείται για να φέρουμε τα στοιχεία του i-στού φοιτητή από το αρχείο και πιθανώς να τα αλλάξουμε αργότερα.

Παράδειγμα:

Στο πρόγραμμα που ακολουθεί διαβάζονται χαρακτήρες από το πληκτρολόγιο και εγγράφονται σε δυαδικό αρχείο, το οποίο λόγω της φύσης των περιεχομένων του είναι αναγνώσιμο με τους συντάκτες κειμένου. Ως συνθήκη τερματισμού θεωρείται η ανάγνωση του δολαρίου (*$*).

Ακολούθως με χρήση της συνάρτησης

void read_character(FILE *fp, char *s, int k)

διαβάζεται ο 14ος χαρακτήρας, χρησιμοποιώντας τυχαία προσπέλαση του αρχείου.

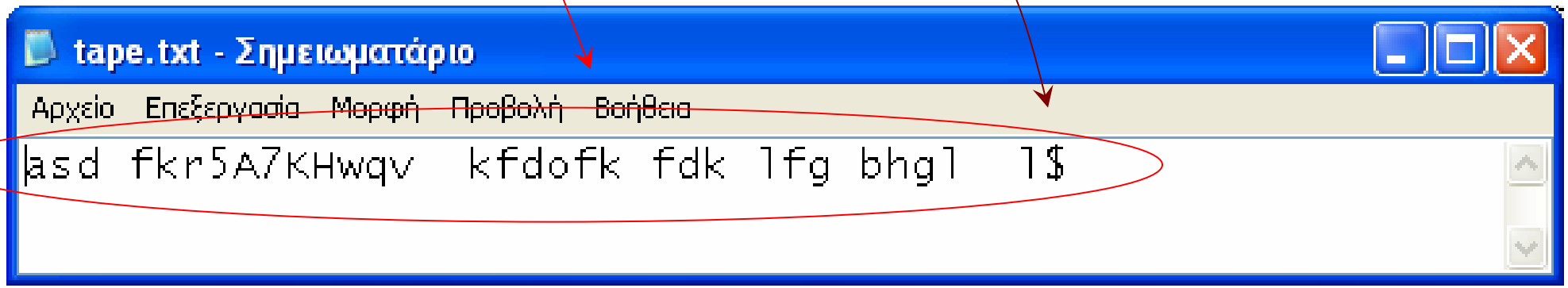
Η συνάρτηση

void save_character(FILE *fp, char *s, int k)

τοποθετεί στη 14η θέση του αρχείου το χαρακτήρα *!*.

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

#define name "tape.txt"

void read_character(FILE *fp, char *s, int k )
{
  fseek(fp,k*sizeof(char),SEEK_SET);
  fread(s,sizeof(char),1,fp);
}
void save_character(FILE *fp, char *s, int k )
{
  fseek(fp,k*sizeof(char),SEEK_SET);
  fwrite(s,sizeof(char),1,fp);
}
```

```
main() {
  FILE *f1;
  char ch;

  f1=fopen(name,"wb");
  do     {
    ch=getche();
    fwrite(&ch,sizeof(char),1,f1);
  }        while (ch!='$');
  fclose(f1);
```



```
f1=fopen(name,"rb+");
  read_character(f1,&ch,13);
  printf("\n\nThe 14th character is: %c\n",ch);
  ch='!';
  save_character(f1,&ch,13);
  fclose(f1);
}
```

C:\temp\Project1.exe

```
asd fkr5A7KHwqv  kfdofk fdk lfg bhgl  l$

The 14th character is: q
```

tape.txt - Σημειωματάριο

Αρχείο Επεξεργασία Μορφή Προβολή Βοήθεια

```
asd fkr5A7KHw!v  kfdofk fdk lfg bhgl  l$
```