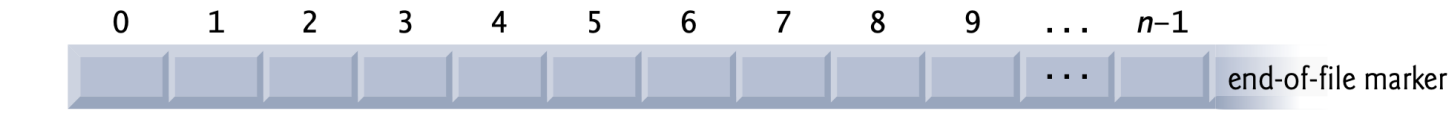
ΕΠΕΞΕΡΓΑΣΙΑ ΑΡΧΕΙΩΝ

- Λέμε *αρχείο* (File) τη συστηματική συγκέντρωση δεδομένων.
- Τα περισσότερα προγράμματα στη γλώσσα C, τα οποία διαπραγματεύονται είσοδο, αποθήκευση και επεξεργασία δεδομένων χρησιμοποιούν την έννοια των αρχείων.
- Για την επεξεργασία των αρχείων η γλώσσα C, συνοδεύεται από τις κατάλληλες συναρτήσεις εισόδου/εξόδου οι οποίες και αυτές βρίσκονται στο επικεφαλής αρχείο stdio.h.

Είδη αρχείων στη C

- Α. Τα *αρχεία κειμένου* (text file) όπου το περιεχόμενο είναι δομημένο σειριακά και σε γραμμές.
- Το τέλος ενός αρχείου κειμένου υποδηλώνεται με ένα ειδικό χαρακτήρα, τον EOF (End-Of-File), ο οποίος στα περισσότερα λειτουργικά συστήματα έχει αρνητική τιμή η οποία συνήθως ισούται με -1.
- Β. *Τα δυαδικά αρχεία* (binary file) όπου το περιεχόμενο είναι δομημένο σειριακά αλλά είναι κωδικοποιημένο σε δυαδική μορφή, τη γνωστή ως BCD (Binary Coded Decimal).

Αρχεία κειμένου



Επεξεργασία αρχείων στη γλώσσα C

Υπάρχουν δύο τρόποι ανάγνωσης και γραφής αρχείων

❶ **υψηλού επιπέδου είσοδος/έξοδος**

όπου κάθε ανάγνωση/εγγραφή γίνεται με ρυθμό ενός χαρακτήρα κάθε φορά

② **χαμηλού επιπέδου είσοδος/έξοδος** «UNIX LIKE»

Με τη μέθοδο αυτή, πρέπει να εκτελέσουμε κάθε ανάγνωση και γραφή με το χέρι, φροντίζοντας για τις προσωρινές μνήμες, τους μετρητές και τους δείκτες

- Οι συναρτήσεις οι οποίες χρησιμοποιούνται για την επεξεργασία αρχείων χαμηλού επιπέδου δεν βρίσκονται σε κάποιο επικεφαλής αρχείο ούτε αποτελούν μέρος των τυποποιημένων εκδόσεων της ANSI C (C89 και C99) αλλά αποτελούν μέρος των εκδόσεων POSIX (Portable Operating System Interface).
- *Παρατήρηση. Όταν λέμε είσοδο δεδομένων σε ένα αρχείο αυτό σημαίνει ότι μεταφέρονται τιμές από κάποιο άλλο περιφερειακό μέσο (πληκτρολόγιο, δίσκο, καταγραφική συσκευή κτλ.) προς αυτό το αρχείο, ενώ*
- *λέμε έξοδο των πληροφοριών από ένα αρχείο τη μεταφορά τιμών από το συγκεκριμένο αρχείο προς ένα άλλο περιφερειακό μέσο του υπολογιστή (οθόνη, δίσκο, εκτυπωτή κτλ.).*

Είσοδος/Εξόδος Υψηλού Επιπέδου

Υπάρχουν συνολικά 30 συναρτήσεις εισόδου/εξόδου

Οι 5 στοιχειώδεις συναρτήσεις της γλώσσας C είναι:

fopen( )	ανοίγει το αρχείο για να χρησιμοποιηθεί
putc( )	γράφει ένα χαρακτήρα στο αρχείο
getc( )	διαβάζει ένα χαρακτήρα από το αρχείο
fclose( )	κλείνει ένα αρχείο
fseek( )	εκτελεί πράξεις τυχαίας προσπέλασης σε δίσκο

Η συνάρτηση fopen ( )

Η συνάρτηση fopen( ) λειτουργεί σαν δύο συναρτήσεις :

❶ ανοίγει ένα αρχείο ώστε να μπορεί να χρησιμοποιηθεί

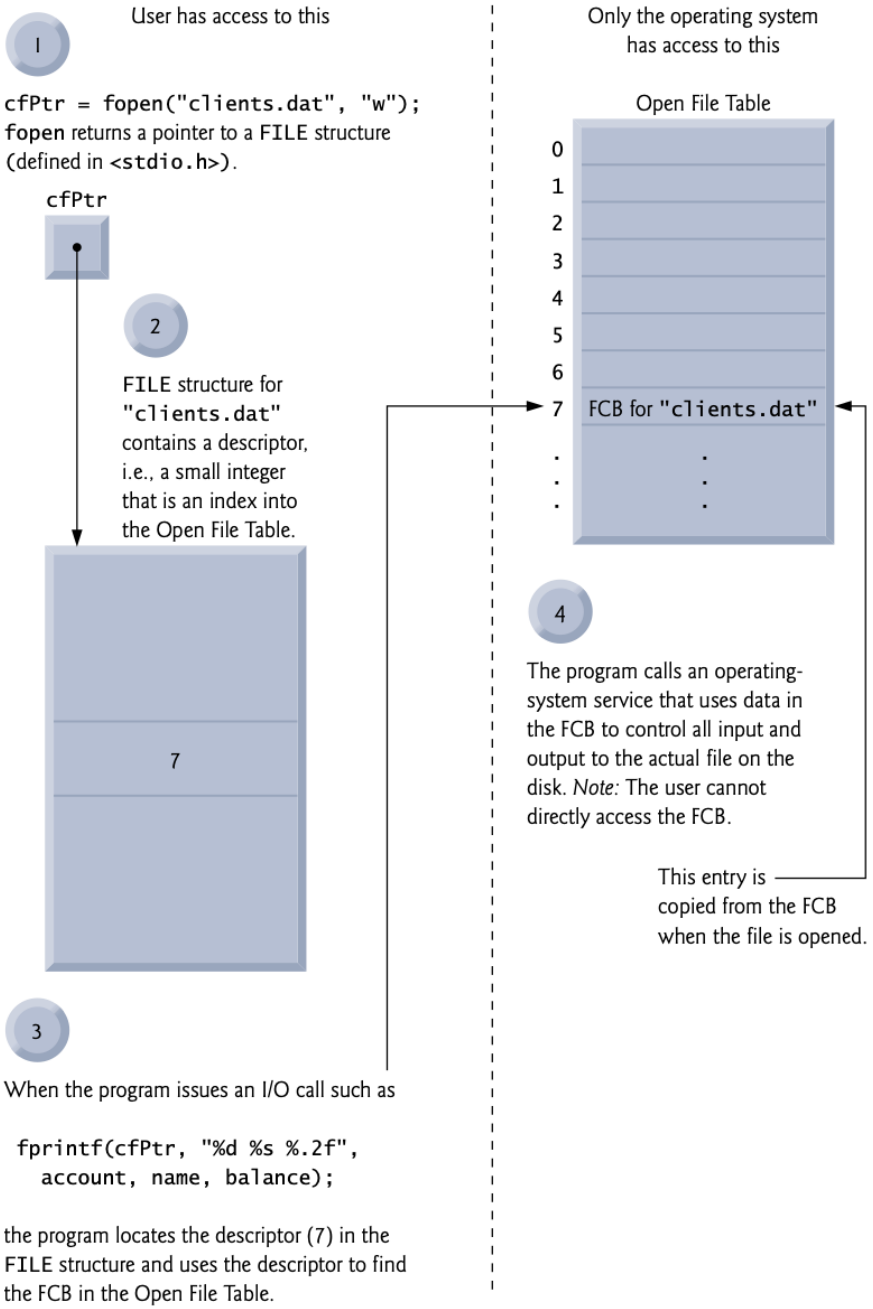
❷ επιστρέφει ένα δείκτη αρχείου

Ο γενικός τύπος της fopen( ) είναι:

```
FILE *fp;

fp = fopen("όνομα αρχείου", "τύπος") ;
```

fp είναι ο δείκτης (index) του αρχείου ο οποίος στην αρχή δείχνει την πρώτη θέση του αρχείου

τύπος είναι ένας χαρακτήρας που πρέπει να είναι:


User has access to this
1
cfPtr = fopen("clients.dat", "w");
fopen returns a pointer to a FILE structure (defined in <stdio.h>).
cfPtr
2
FILE structure for "clients.dat" contains a descriptor, i.e., a small integer that is an index into the Open File Table.
7
3
When the program issues an I/O call such as
fprintf(cfPtr, "%d %s %.2f", account, name, balance);
the program locates the descriptor (7) in the FILE structure and uses the descriptor to find the FCB in the Open File Table.
Only the operating system has access to this
Open File Table
0
1
2
3
4
5
6
7
FCB for "clients.dat"
4
The program calls an operating-system service that uses data in the FCB to control all input and output to the actual file on the disk. Note: The user cannot directly access the FCB.
This entry is copied from the FCB when the file is opened.

Τύποι ανοίγματος αρχείου

Mode	Description
`r`	Open an existing file for reading.
`w`	Create a file for writing. If the file already exists, *discard* the current contents.
`a`	Open or create a file for writing at the end of the file—i.e., write operations *append* data to the file.
`r+`	Open an existing file for update (reading and writing).
`w+`	Create a file for reading and writing. If the file already exists, *discard* the current contents.
`a+`	Open or create a file for reading and updating; all writing is done at the end of the file—i.e., write operations *append* data to the file.
`rb`	Open an existing file for reading in binary mode.
`wb`	Create a file for writing in binary mode. If the file already exists, discard the current contents.
`ab`	Append: open or create a file for writing at the end of the file in binary mode.
`rb+`	Open an existing file for update (reading and writing) in binary mode.
`wb+`	Create a file for update in binary mode. If the file already exists, discard the current contents.
`ab+`	Append: open or create a file for update in binary mode; writing is done at the end of the file.

- **r (read) για την ανάγνωση**
- **r +** Ανάγνωση/εγγραφή
- **w (write) για την εγγραφή**
- **w +** Ανάγνωση/εγγραφή
- **a (append) για την πρόσθεση εγγραφών στο τέλος του αρχείου**
- **a +** Ανάγνωση και πρόσθεση νέων εγγραφών μετά το τέλος του αρχείου
- **Αν θέλουμε να δημιουργήσουμε ένα δυαδικό αρχείο πρέπει να προσθέσουμε το χαρακτήρα b (binary) μετά τον τύπο δημιουργίας ενός αρχείου.**
- **Ο χαρακτήρας b μπορεί να τοποθετηθεί πριν ή μετά από το σύμβολο +.** Π.χ. **rb+ ή r+b**

Η συνάρτηση fopen (συνέχεια)

Η συνάρτηση fopen() συνήθως γράφεται ως εξής:

```
FILE *fp
if ((fp = fopen("test3.dat","r")) = = NULL)
 {
    puts("Το αρχείο αυτό δεν υπάρχει.\n");
    exit( ) ;
}
```

Η συνάρτηση fopen ( )

Ειδικές περιπτώσεις

- Η τιμή NULL που ισούται με το 0, χρησιμοποιείται επειδή δεν υπάρχει δείκτης αρχείου που θα μπορούσε να έχει την τιμή 0.
- Αν χρησιμοποιήσουμε την fopen( ) για να ανοίξουμε ένα αρχείο για γραφή, θα διαγραφεί κάθε αρχείο με το όνομα αυτό και θα δημιουργηθεί ένα καινούργιο.
- Προσοχή, γιατί τα δεδομένα του προηγούμενου αρχείου θα χαθούν οριστικά ☺

Η συνάρτηση putc( )

Η συνάρτηση putc( ) χρησιμοποιείται για:

- να γράφει χαρακτήρες σ' ένα αρχείο
- το οποίο έχει ανοίξει χρησιμοποιώντας τη συνάρτηση fopen( ) με τους τύπους w, w+, r+, a και a+.

Ο γενικός τύπος της συνάρτησης είναι:

putc (ch, fp);

- όπου fp είναι ο δείκτης του αρχείου που επιστρέφει η fopen( )
- ch είναι ο χαρακτήρας που παράγεται δηλ. γράφεται στο αρχείο.

Ο δείκτης του αρχείου λέει στην putc( ) σε ποιό αρχείο και σε ποιο σημείο του να γράψει. Έτσι, γράφει στην αμέσως επόμενη θέση από την τελευταία εγγραφή

Η συνάρτηση getc( )

Η συνάρτηση getc( ) χρησιμοποιείται για

- να διαβάζει χαρακτήρες από ένα αρχείο
- που άνοιξε με χρήση της συνάρτησης fopen ( ) με τους τύπους r, r+, w+ και a+

Ο γενικός τύπος της συνάρτησης είναι:

```
char ch;

ch = getc( fp );
```

όπου fp είναι ένας δείκτης (index) αρχείου τύπου FILE που έχει επιστραφεί από την fopen( )

Η συνάρτηση fclose( )

Η συνάρτηση fclose( ) χρησιμοποιείται για να κλείνει ένα αρχείο που έχει ανοίξει μετά από κλήση στην fopen( )

Πρέπει να κλείνουμε όλα τα αρχεία πριν τελειώσει το πρόγραμμα

Η fclose( ) κάνει κάτι περισσότερο από το να ελευθερώνει απλώς το δείκτη του αρχείου.

Γράφει οποιαδήποτε δεδομένα δεν έχουν γραφεί στο δίσκο και κάνει ένα τυπικό κλείσιμο του αρχείου στο επίπεδο του λειτουργικού συστήματος

Ο γενικός τύπος της συνάρτησης fclose( ) είναι:

```
fclose(fp);
```

Παραδείγματα...

Ανοίγει ένα αρχείο για να γράψει τους χαρακτήρες που διαβάζει από το πληκτρολόγιο μέχρι να βρεί τον χαρακτήρα #

```
#include <stdio.h>
#define CR  13
#define LF   10
main( int argc, char* argv[]) {
    FILE *fp;
    char ch;
    if ( argc != 2)  {
        printf( "Ξεχάσατε να γράψετε το όνομα του αρχείου\n” );
        exit(0);
 }
```

```
if (( fp = fopen(argv[1], "w" )) = = NULL)
{
   printf("Δεν μπορεί να ανοίξει αυτό το αρχείο ίσως  δεν υπάρχει ελεύθερος
            χώρος στο δίσκο\n");
   exit(0);
 }
   do {
      ch=getchar( );
      if ( ch = = '\n' ) {
          putc( CR, fp );
          putc( LF, fp );
      }
       else putc(ch, fp);
    }   while ( ch != ' # ' );
    fclose( fp );
  }
```

fprintf()

```c
// Fig. 11.2: fig11_02.c
// Creating a sequential file
#include <stdio.h>

int main(void)
{
   FILE *cfPtr; // cfPtr = clients.txt file pointer

   // fopen opens file. Exit program if unable to create file
   if ((cfPtr = fopen("clients.txt", "w")) == NULL) {
      puts("File could not be opened");
   }
   else {
      puts("Enter the account, name, and balance.");
      puts("Enter EOF to end input.");
      printf("%s", "? ");

      unsigned int account; // account number
      char name[30]; // account name
      double balance; // account balance

      scanf("%d%29s%lf", &account, name, &balance);

      // write account, name and balance into file with fprintf
      while (!feof(stdin)  ) {
         fprintf(cfPtr, "%d %s %.2f\n", account, name, balance);
         printf("%s", "? ");
         scanf("%d%29s%lf", &account, name, &balance);
      }

      fclose(cfPtr); // fclose closes file
   }
}
```

```
Enter the account, name, and balance.
Enter EOF to end input.
? 100 Jones 24.98
? 200 Doe 345.67
? 300 White 0.00
? 400 Stone -42.16
? 500 Rich 224.62
? ^Z
```

fscanf()

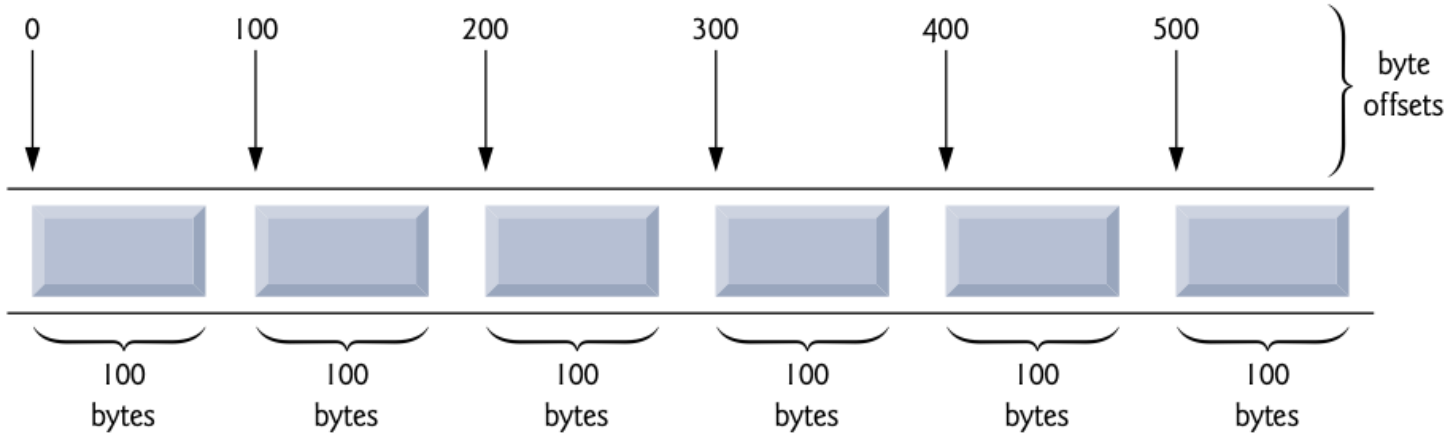
```
Account   Name          Balance
100       Jones           24.98
200       Doe            345.67
300       White            0.00
400       Stone          -42.16
500       Rich           224.62
```

```c
// Fig. 11.6: fig11_06.c
// Reading and printing a sequential file
#include <stdio.h>

int main(void)
{
   FILE *cfPtr; // cfPtr = clients.txt file pointer

   // fopen opens file; exits program if file cannot be opened
   if ((cfPtr = fopen("clients.txt", "r")) == NULL) {
      puts("File could not be opened");
   }
   else { // read account, name and balance from file
      unsigned int account; // account number
      char name[30]; // account name
      double balance; // account balance

      printf("%-10s%-13s%s\n", "Account", "Name", "Balance");
      fscanf(cfPtr, "%d%29s%lf", &account, name, &balance);

      // while not end of file
      while (!feof(cfPtr)  ) {
         printf("%-10d%-13s%7.2f\n", account, name, balance);
         fscanf(cfPtr, "%d%29s%lf", &account, name, &balance);
      }

      fclose(cfPtr); // fclose closes the file
   }
}
```

Αρχεία τυχαίας προσπέλασης

- Η γλώσσα C, δεν διαθέτει ειδικές εντολές ή συναρτήσεις για την επεξεργασία αρχείων τυχαίας (random access) ή άμεσης προσπέλασης (direct access)
- Όλα τα αρχεία είναι σειριακά (sequential)
- Τα αρχεία τυχαίας ή άμεσης προσπέλασης είναι πολύ χρήσιμα και βοηθούν στην αποτελεσματική επεξεργασία των αρχείων
- Μπορούμε να μεταβούμε άμεσα σε μια συγκεκριμένη εγγραφή του αρχείου και στη συνέχεια να την επεξεργαστούμε χωρίς αυτό να σημαίνει ότι το αρχείο είναι άμεσης προσπέλασης με την αυστηρή έννοια του όρου

Αρχεία τυχαίας προσπέλασης



Η συνάρτηση fseek( )

Ο γενικός τύπος της συνάρτησης fseek( ) είναι:

fseek (fp, μήκος, αφετηρία) ;

όπου

fp είναι ένας δείκτης αρχείου που επιστρέφει με την κλήση στην fopen( )

μήκος είναι ο αριθμός των byte από την αφετηρία, τα οποία προσδιορίζουν την τρέχουσα θέση

αφετηρία μπορεί να είναι

- το 0 για την έναρξη του αρχείου
- το 1 για την τρέχουσα θέση
- το 2 για το τέλος του αρχείου

Η συνάρτηση fseek( )

Η fseek( ) χρησιμοποιείται για να προσδιορίσει την τρέχουσα θέση (συγκεκριμένο byte) σ' ένα αρχείο

Π.χ. Για την ανάγνωση του 234^{ου} χαρακτήρα ενός αρχείου πρέπει να γράψουμε

```
FILE *fp;
if(( fp=fopen("test2.dat","r")) = = NULL)
 {
  printf("Δεν μπορεί να ανοίξει το αρχείο,
                άγνωστο όνομα\n");
  exit(1);
 }
fseek(fp,233,0);
 getc(fp);
```

Άμεση ανάγνωση του 3ου και του 8ου χαρακτήρα

```
#include <stdio.h>
 void main( )
{   FILE *fp;
    char value;
    if((fp=fopen("datain.txt","r")) == NULL)
      {    printf("Δεν μπορεί να ανοίξει το αρχείο\n");
           exit(1);     }
//Εντοπισμός της 3ης θέσης από την αρχή του αρχείου
    fseek(fp,2,0);
    value = fgetc(fp);  // Ανάγνωση του χαρακτήρα
      printf("Τιμή του χαρακτήρα = %c \n", value);
//Εντοπισμός της 8ης θέσης από την αρχή του αρχείου
      fseek(fp,4,1);       //   ή  fseek(fp,7,0)
    value = fgetc(fp);  // Ανάγνωση του χαρακτήρα
      printf("Τιμή του χαρακτήρα = %c \n", value);
}
```

Πρόγραμμα το οποίο ανοίγει το αρχείο που θα ορίσει ο χρήστης κατά την εκτέλεση του προγράμματος και εμφανίζει το περιεχόμενό του.

```
#include <stdio.h>
   main( argc, argv)
    int argc;
   char *argv[ ];

     {   FILE *fp;
         char ch;
         if( argc != 2)
{ printf("Ξεχάσατε να γράψετε το όνομα
                  του αρχείου\n");
     exit(0);
 }
```

```
if(( fp = fopen(argv[1], "r" )) = = NULL)
    { printf("Δεν μπορεί να ανοίξει το αρχείο\n");
          exit(0);
     }
       ch=getc( fp );  /*ανάγνωση ενός χαρακτήρα*/
           while ( ch != EOF )    /*έλεγχος*/
        {  putchar( ch );   /*εμφάνιση στην οθόνη*/
            ch=getc( fp );
         }
          fclose( fp );
     }
```

Τα αρχεία stdin, stdout και stderr

Αυτά τα αρχεία έχουν σαν σημείο αναφοράς την οθόνη.

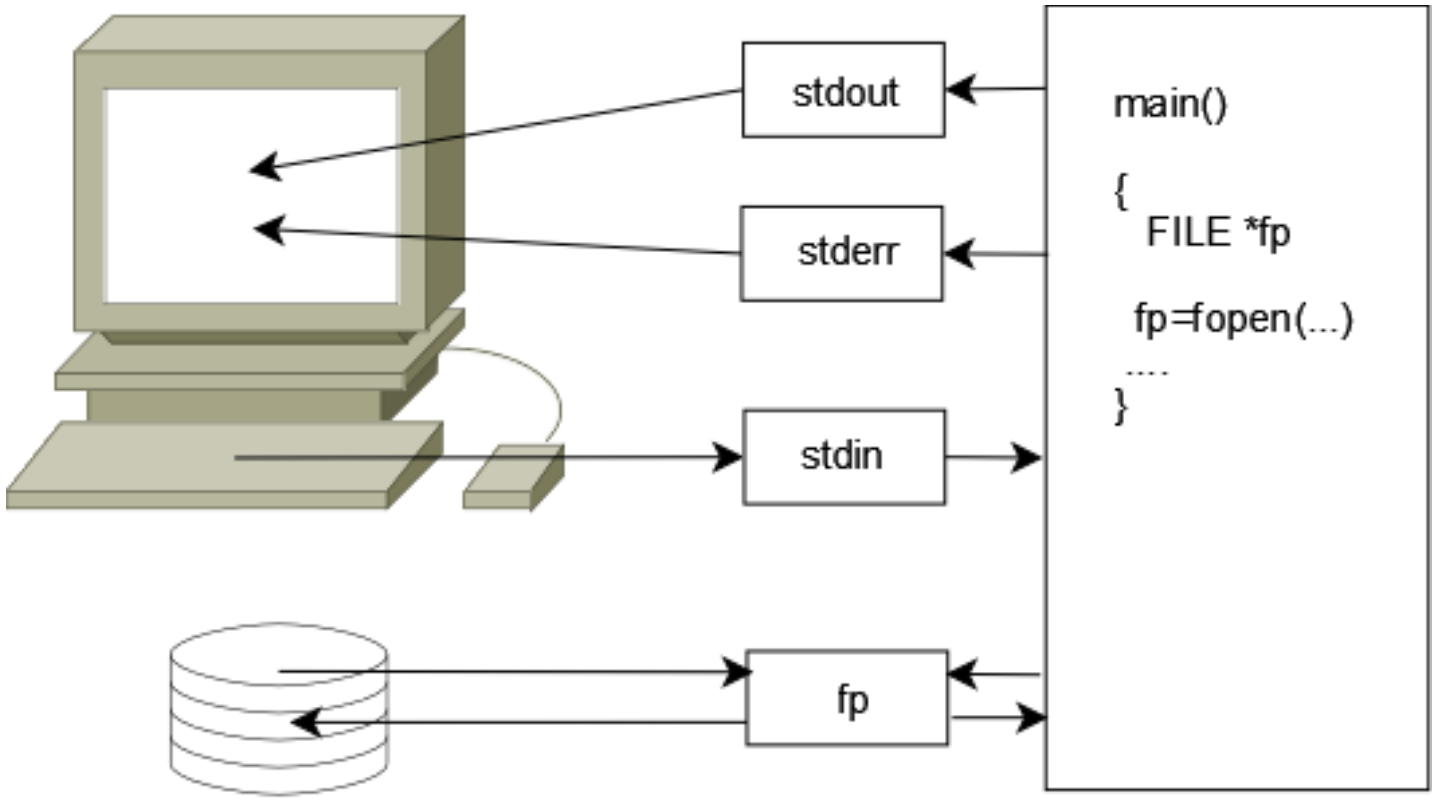
Χρησιμοποιούν την οθόνη του υπολογιστή σαν ένα αρχείο.

Π.χ. η γνωστή συνάρτηση putchar( ) θα μπορούσε να ορισθεί σαν:

```
putchar( c )
char c;
{
   putc(c, stdout);
}
```

Γενικά, το αρχείο stdin χρησιμοποιείται για να διαβάζει από την οθόνη μέσω του πληκτρολογίου, και τα stdout και stderr για να γράφουν στην οθόνη.

Γραφική απεικόνιση λειτουργίας των αρχείων



Οι συναρτήσεις fprintf( ) και fscanf( )

Ο γενικός τύπος της fprintf( ) είναι:

fprintf(fp, "χαρακτήρες ελέγχου", λίστα παραμέτρων);

και της fscanf( ) είναι:

fscanf(fp, "χαρακτήρες ελέγχου", λίστα παραμέτρων);

όπου fp είναι ένας δείκτης αρχείου που έχει επιστραφεί από μία κλήση στη συνάρτηση fopen( )

Οι συναρτήσεις fprintf( ) και fscanf( ) λειτουργούν ακριβώς όπως οι συναρτήσεις printf( ) και scanf( ) με τη μόνη διαφορά ότι κατευθύνουν την έξοδό τους στο αρχείο που έχει ορισθεί από το fp.

Τυπικό παράδειγμα χρήσης αρχείου

Διαβάζει ένα κείμενο και έναν αριθμό, τα σώζει σε αρχείο, το κλείνει και αφού ανοίξει το ίδιο αρχείο εμφανίζει τις τιμές που σώθηκαν.

```
main( )
{
    FILE *fp;
    char s[80];
     int t;
     if ( fp=fopen("test3","w")) = = NULL)
{  printf("Δεν μπορεί να ανοίξει το αρχείο,
              ίσως δεν υπάρχει χώρος\n");
          exit(0);
}
```

```
        fscanf(stdin,"%s%d", s, &t);
        fprintf(fp,"%s  %d"; s, t);
        fclose(fp);
        if((fp=fopen("test3","r")) = = NULL)
{   printf("Δεν μπορεί να ανοίξει το αρχείο
                    ή δεν υπάρχει το αρχείο\n");
          exit(0);
}
         fscanf(fp,"%s%d",s,&t);
        fprintf(stdout,"%s  %d",s,t);
       fclose(fp);
    }
```

ΑΣΚΗΣΕΙΣ

Α. Γράψτε ένα πρόγραμμα που να χρησιμοποιεί είσοδο/έξοδο υψηλού επιπέδου, το οποίο θα αντιγράφει ένα αρχείο σ' ενα άλλο.

Να χρησιμοποιηθούν παράμετροι γραμμής εντολής για τον ορισμό των αρχείων.

```
#include <stdio.h>
   main(argc, argv)
    int argc;
    char *argv[ ];
{
 char ch;
 FILE *fp1,*fp2;
   if((fp1=fopen(argv[1], "r"))==0)
    {
      printf("Δεν μπορώ ν' ανοίξω το αρχείο %s\n", argv[1]);
       exit (0);
     }
```

συνεχίζεται...

```
    if((fp2=fopen(argv[2], "w")) == 0
  {
   printf("Δεν μπορώ ν' ανοίξω το αρχείο %s \n", argv[2]);
    exit(0);
  }
     while((ch=getc(fp1))!=EOF)
                putc(ch,fp2);
      fclose(fp1);
      fclose(fp2);
}
```

Οι συναρτήσεις remove( ) και rename( )

Η γλώσσα C, διαθέτει τη συνάρτηση remove( ) η οποία επιτρέπει τη διαγραφή ενός αρχείου ή ενός καταλόγου κατά τη διάρκεια της εκτέλεσης ενός προγράμματος

Π.χ.

```
FILE *fp;
fp = fopen("test3.dat","r")
int index=remove(fp);
printf("%d ", index);
```

Αν η τιμή της μεταβλητής index είναι διάφορη του μηδενός αυτό σημαίνει ότι η διαγραφή του αρχείου test3.dat είναι ανεπιτυχής

- **Η αποτυχία διαγραφής ενός αρχείου ή ενός καταλόγου μπορεί να οφείλεται σε διάφορους λόγους, με πιο σημαντικούς:**

- **Το αρχείο έχει προστασία εγγραφής**
- **Ο κατάλογος περιέχει αρχεία**
- **Το μέσο αποθήκευσης δεν επιτρέπει διαγραφές (είναι μόνο ανάγνωσης όπως ένας οπτικός δίσκος)**
- **Το αρχείο είναι ανοικτό και χρησιμοποιείται από άλλο πρόγραμμα**
- **Το αποθηκευτικό μέσο φιλοξενίας του αρχείου έχει αφαιρεθεί**

Οι συναρτήσεις remove( ) και rename( )

Αν θέλουμε να αλλάξουμε το όνομα ενός αρχείου ή ενός καταλόγου, αυτό μπορεί να γίνει εύκολα με τη βοήθεια των διαδικασιών αλλαγής ονόματος των αρχείων του λειτουργικού συστήματος

(εντολή Rename)

Π.χ.

```
FILE *fp;
fp = fopen("test3.dat","r")
int index=rename("test3.dat" , "test4.dat") ;
printf("%d ", index);
```

- **Η ανεπιτυχής μετονομασία ενός αρχείου ή ενός καταλόγου μπορεί να οφείλεται σε διάφορους λόγους με πιο σημαντικούς:**
- **Προσπάθεια αντικατάστασης ενός καταλόγου από ένα αρχείο**
- **Ο κατάλογος δεν είναι κενός αλλά περιέχει αρχεία**
- **Το αρχείο έχει προστασία εγγραφής**

Η συνάρτηση ungetc( )

Η συνάρτηση ungetc( ) χρησιμοποιείται για να επιστραφεί ο δείκτης του αρχείου πίσω κατά μία θέση δηλαδή, μπορεί να θεωρηθεί σαν μια συνάρτηση η οποία αναιρεί προσωρινά το αποτέλεσμα της χρήσης της συνάρτησης getc( ) στο αρχείο

Ο γενικός τύπος της συνάρτησης ungetc( ) είναι:

ungetc(c, fp);

c, είναι μια μεταβλητή τύπου char που χρησιμοποιείται για την προσωρινή αποθήκευση του χαρακτήρα ο οποίος έχει αναγνωστεί προηγουμένως από μια κλήση στη συνάρτηση getc( ).

Παράδειγμα

Έστω ότι θέλουμε να υπολογίζεται το πλήθος της εμφάνισης των εντολών for μέσα σε ένα αρχείο το οποίο περιέχει τον πηγαίο κώδικα ενός κοινού προγράμματος σε γλώσσα C.

```
#include <stdio.h>
#include <string.h>
 main()
   {  FILE *f;
      int i, number;
      int counter = 0;
      char c;
      char tmp[3];
```

```
// Έλεγχος ύπαρξης του αρχείου
f = fopen("test.txt", "r");
number = fileno(f);
if (ferror(f))
{  puts("Το αρχείο αυτό δεν υπάρχει.\n");
exit(1) ;  }
// Διαβάζουμε μέχρι το τέλος του αρχείου
while (!feof(f))
{  c = getc(f);
if (c == 'f') // έλεγχος του γράμματος f
{  ungetc(c, f); // επιστροφή μια θέση πίσω
```

```
// ανάγνωση των επόμενων 3 χαρακτήρων
for (i = 0; i < 3; ++i)
        tmp[i] = getc(f);
// έλεγχος ύπαρξης της εντολής for
   if (!strcmp(tmp, "for"))
       ++counter;    // μετρητής των εντολών for
  }
}
fclose(f);
      printf("Το πλήθος των εντολών for είναι: %d\n", counter);
}
```