DOSYALAR

Dosya Nedir?

Her programının ihtiyaç duyduğu ortak görevler kullanıcıdan girdiyi (input) okumak ve konsolda çıktıyı (output) göstermektir. Bu şekilde girdi ve çıktı işlemleri yaparsak, program çalıştığı sürece veriler var olur, program sonlandırıldığında o verileri tekrar kullanamayız. Bunun için elektrik kesildiğinde bellekteki veriler kaybolacağından harici hafıza ortamında (external memory) verileri saklamak gerekir.

Kısaca dosya (file), verilerin bir araya geldiği (collection of data) ikincil saklama ortamıdır (seconder storage). Bu ikincil ortam; Bir bilgisayar hafızası (memory) olabileceği gibi, verilerin elektrikler kesildiğinde kaybolmayacağı sabit disk (hard disc), ya da bir başka ortama/bilgisayara veri gönderen (send) veya alan (receive) modem ve benzeri bir cihaz (device) olabilir.

Standart Dosyalar

C dili programcıya, girdi ve çıktı kısaca IO (input output) için çeşitli işlevleri içeren başlık dosyaları (header file) sağlar ve tüm aygıtları (device) dosyalar olarak ele alınır. Bu nedenle, "ekran" gibi aygıtlar "dosyalar" ile aynı şekilde ele alınır.

Standart Dosya	Dosya	Aygıt
Standart Giriş	`stdin`	Klavye
Standart Çıktı	`stdout`	Ekran
Standart Hata	`stderr`	Kullanıcı Ekranı

Tablo 28. C Dilinde Standart Giriş Çıkış Dosyaları

Dosyalarla işlem yapmak için `stdio.h` başlık dosyası çeşitli fonksiyonlara sahiptir.

Biçimlendirilmemiş Karakter Giriş Çıkış Fonksiyonları

```
int getchar(void);
```

Yeni Satır (`'\n'`) tuşuna basmadan tek bir tuş vuruşunu/karakteri okur.

```
int putchar(int c);
```

Tek bir karakter yazar. Yazdırılacak karakterin ASCII kodu parametre olarak gönderilebilir.

```
#include <stdio.h>
int main() {
   char ch;
   printf("Bir karakter Giriniz: ");
   ch = getchar();
   puts("Girdiğiniz karakter: ");
   putchar(ch);
   return 0;
}
```

Biçimlendirilmemiş Dizgi Giriş Çıkış Fonksiyonları

Okuma ve yazma amacıyla dosyaya erişmek için önceden tanımlanmış bir FILE yapısı (struct) kullanır. Standart dosyalar da bu parametreye argüman olarak verilebilir.

```
char* fgets(char* str,int size,FILE* stream);
```

`stream` dosyasından `str` ile işaret edilen tampon belleğe (`buffer`), yeni satır veya dosya sonu (End of File- EOF) ile karşılaşılıncaya kadar bir satırı okur. Hata yoksa `str` değişkeni, varsa `EOF` ya da `NULL` geri döndürür.

```
fputs(const char* str,FILE* stream);
```

`str` dizgisini ve sonuna yeni satır karakteri koyarak `stream` dosyasına yazar. Hata yoksa pozitif bir değer döner, varsa `EOF` geri döndürür.

```
#include <stdio.h>
int main() {
   char adi[20];
   printf("Adınız: ");
   fgets(adi, sizeof(adi), stdin);
   fputs("Girdiğiniz Adınız", stdout);
   fputs(adi,stdout);
   return 0;
}
```

Biçimlendirilmiş Giriş Çıkış Fonksiyonları

```
int scanf(const char* format, ...);
```

Standart giriş dosyası `stdin` standart dosyasından girişi sağlanan biçime göre okur;

```
int printf(const char format, ...);
```

Çıktıyı standart çıktı dosyası `stdout` standart çıkış dosyasına sağlanan biçime göre üreterek yazar. Şimdiye kadar bu fonksiyonların kullanımını çokça gördük.

```
int fscanf(FILE* stream, const char* format, ...);
```

FILE yapısı ile belirtilen dosyadan girişi sağlanan biçime göre okur. `stdin` standart giriş dosyası olarak kullanılabilir.

```
int fprintf(FILE* stream, const char format, ...);
```

FILE yapısı ile belirtilen dosyaya sağlanan biçime göre çıktıyı üreterek yazar. `stsout` ve `stderr` standart çıkış dosyalarıdır. Bu fonksiyonların kullanımı `scanf` ve `printf` fonksiyonları ile aynıdır. Yalnızca hangi dosyanın kullanılacağına ilişkin ek bir `stream` parametresi eklenmiştir.

Standart Olmayan Dosyalar

Standart olan `stdin`, `stdout` ve `stderr` dosyaları her zaman veri alışverişine açıktır. Bu dosyalarla metinler üzerinden veri alışverişi yapılır. Standart olmayan dosyalar ise her zaman veri alışverişine açık değildir. Bu nedenle dosya ile işlem yapmadan önce dosyayı açmalı ve işimiz bittiğinde de kapatmalıyız.

Standart Olmayan Metin Dosyaları

Standart olmayan dosyaları açıp kapatmayı sağlayan iki fonksiyon vardır;

```
FILE *fopen(const char* file_name, const char* mode_of_operation);
```

Adı `file_name` ile verilen dosyayı; okumak (read), yazmak (write) ya da sonuna eklemek (append) için ya da bunlardan her ikisini yapmak için açar. Açılan dosyanın FILE yapısına (struct) ilişkin göstericiyi geri döndürür. Burada bir metin dosyasını aşağıdaki argümanları bir dizgi (string) olarak `fopen` fonksiyonuna argüman vererek çeşitli şekillerde işlem yapmak için açabiliriz. Bu argümanlar dosya açma modları olarak adlandırılır.

Modlar	Açıklaması
r	Dosyayı, çalışılan klasörde/dizinde arar. Dosyayı yalnızca okumak için açar. Dosya başarıyla açılırsa `fopen()` onu belleğe yükler ve içindeki ilk karakteri işaret eden bir gösterici ayarlar. Dosya açılamıyorsa `NULL` değerini döndürür.
w	Dosyayı, çalışılan klasörde/dizinde arar. Dosya zaten mevcutsa içeriğinin üzerine yazılır. Dosya mevcut değilse yeni bir dosya oluşturulur. Dosya açılamıyorsa `NULL` değerini döndürür. Yalnızca yazmak için yeni bir dosya oluşturur.
a	Dosyayı çalışılan klasörde/dizinde arar. Dosya başarıyla açılırsa `fopen()` onu belleğe yükler ve içindeki son karakteri işaret eden bir gösterici ayarlar. Dosya mevcut değilse yeni bir dosya

Modlar	Açıklaması
	oluşturulur. Dosya açılamıyorsa `NULL` değerini döndürür. Dosya yalnızca ekleme (append) yani dosyanın sonuna yazma amacıyla açılır.
r+	Dosyayı, çalışılan klasörde/dizinde arar. Dosyayı hem okumak hem de yazmak için açar. Başarıyla açılırsa, `fopen()` onu belleğe yükler ve içindeki ilk karakteri işaret eden bir gösterici ayarlar. Dosya açılamıyorsa `NULL` değerini döndürür.
w+	Dosyayı, çalışılan klasörde/dizinde arar. Dosya mevcutsa içeriğinin üzerine yazılır. Dosya mevcut değilse yeni bir dosya oluşturulur. Dosya açılamıyorsa `NULL` değerini döndürür. w ve `w+` arasındaki fark, `w+` kullanılarak oluşturulan dosyayı da okuyabilmemizdir.
a+	Dosyayı, çalışılan klasörde/dizinde arar. Dosya başarılı bir şekilde açılırsa `fopen()` onu belleğe yükler ve içindeki son karakteri işaret eden bir gösterici ayarlar. Dosya mevcut değilse yeni bir dosya oluşturulur. Dosya açılamıyorsa `NULL` değerini döndürür. Dosya okumaya ve eklemeye (dosyanın sonuna yazma) açılır.

Tablo 29. Metin Dosyası Açma Modları

Aşağıdaki fonksiyon ise açık olan dosyayı kapatır.

```
int fclose(FILE* stream);
```

Aşağıda verilen örnekte görüldüğü gibi aynen konsola yazar gibi yazılması gerekenleri `fprintf()` fonksiyonu ile aynen dosyaya yazılmıştır. Kullanımı `printf()` ile aynıdır sadece ilk parametresi dosya göstericisidir. Dosya `w+` modunda açıldığından, program her çalıştığında içeriği silerek yeniden aynı şeyleri yazacaktır.

```
#include <stdio.h>
int main()
{
    FILE* dosyaGostericisi;
    // "metin.txt" dosyası yoksa oluşturulur.
    // Varsa üzerine yazılır. Hem razma hem okuma modunda dosya açılıyor.
    dosyaGostericisi = fopen("metin.txt", "w+");
    fprintf(dosyaGostericisi, "Adi Soyadi;Yaşı;Cinsiyeti\n");
    fprintf(dosyaGostericisi, "%s;%d;%c\n", "Ilhan OZKAN", 50,'E');
    fprintf(dosyaGostericisi, "%s;%d;%c\n", "Yagmur OZKAN", 45,'K');
    fclose(dosyaGostericisi);
    return 0;
}
```

Oluşan metin dosyası olan `metin.txt` içeriği aşağıda verilmiştir;

```
Adi Soyadi;Yaşı;Cinsiyeti
Ilhan OZKAN;50;E
Yagmur OZKAN;45;K
```

Aynı dosyası her bir satırı tam olarak okuyan `fgets()` fonksiyonu ile okuyabiliriz;

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define BIRSATIRDAKIENFAZLAKARAKTER 80
int main()
{
    char *dosyaAdi="metin.txt";
    char satir[BIRSATIRDAKIENFAZLAKARAKTER] = {0};
    unsigned int satirNumarasi = 0;
    FILE *dosyaGostericisi = fopen(dosyaAdi, "r");
    if (!dosyaGostericisi)
    {
        perror(dosyaAdi);
        return EXIT_FAILURE;
    }
    while (fgets(satir, BIRSATIRDAKIENFAZLAKARAKTER, dosyaGostericisi))
    {
```

```
        printf("satir[%06d]: %s", ++satirNumarasi, satir);
        if (satir[strlen(satir) - 1] != '\n')
            printf("\n");
    }
    if (fclose(dosyaGostericisi))
    {
        return EXIT_FAILURE;
        perror(dosyaAdi);
    }
}
/* Program Çıktısı:
satir[000001]: Adi Soyadi;Yasi;Cinsiyeti
satir[000002]: Ilhan OZKAN;50;E
satir[000003]: Yagmur OZKAN;45;K
*/
```

Bunların yanında `perror()` fonksiyonu, programınızda neyin yanlış gittiğini anlamanıza ve hata ayıklamanıza yardımcı olan standart hata akışına (stderr) açıklayıcı bir hata mesajı yazdırmak üzere tasarlanmıştır;

```
#include <stdio.h> /* perror(), fopen(), fputs() and fclose() */
#include <stdlib.h> /* EXIT_* macroları için */
int main(int argc, char **argv)
{
    int e = EXIT_SUCCESS;
    char *path = "output.txt";
    FILE *file = fopen(path, "w");
    if (!file)
    {
        perror(path);
        return EXIT_FAILURE;
    }
    if (fputs("Dosyaya Yazılacak Metin....\n", file) == EOF)
    {
        perror(path);
        e = EXIT_FAILURE;
    }
    if (fclose(file))
    {
        perror(path);
        return EXIT_FAILURE;
    }
    return e;
}
```

Ayrıca aşağıda verilen program `popen()` aracılığıyla bir program ya da servisi çalıştırır ve işlemden gelen tüm standart çıktıyı okur ve bunu standart çıktı olan konsola yansıtır;

```
#include <stdio.h>
void print_all(FILE *stream)
{
    int c;
    while ((c = getc(stream)) != EOF)
    putchar(c);
}
int main(void)
{
    FILE *stream;
    /* call netstat command. netstat is available for Windows and Linux */
    if ((stream = popen("dir", "r")) == NULL)
        return 1;
    print_all(stream);
    pclose(stream);
```

```
    return 0;
}
```

Standart Olmayan İkili Dosyalar

Değişkenlerimizi her zaman insan gözünün okuyacağı metne çevirerek dosyalara kaydetmeyiz. Onun yerine daha az yer kaplaması açısından bellekte saklandığı şekliyle dosyaya hızlıca yazarız. Ancak bu durumda dosyaya, bellekte ikili olarak tutulan verileri ikili olarak yazarız ve insan gözüyle okunamazlar.

Standart olmayan ikili dosyalar da metin dosyaları gibi açılıp kapatılır. İkili dosyaları da açmak için dosya modları farklıdır;

Modlar	Açıklaması
`rb`	Dosyayı çalışılan klasörde/dizinde arar. İkili dosyayı okuma modunda açar. Dosya mevcut değilse, `fopen()` işlevi `NULL` değerini döndürür.
`wb`	İkili dosya yazma modunda açılır. Gösterici dosyanın başlangıcına ayarlanır ve içeriklerin üzerine ikili olarak yazılır. Dosya mevcut değilse yeni bir dosya oluşturulur.
`ab`	İkili dosya, ekleme modunda açılır. Dosya göstericisi, dosyadaki son karakterden sonra ayarlanır. Verilen dosya isminden bir dosya yoksa yeni bir dosya oluşturulur.
`rb+`	İkili dosya okuma ve yazma modunda açılır. Dosya mevcut değilse, `open()` işlevi `NULL` değerini döndürür.
`wb+`	İkili dosya okuma ve yazma modunda açılır. Dosya mevcutsa içeriğin üzerine yazılır. Dosya mevcut değilse oluşturulacaktır.
`ab+`	İkili dosya okuma ve ekleme modunda açılır. Dosya mevcut değilse bir dosya oluşturulacaktır.

Tablo 30. İkili Dosya Açma Modları

İkili dosyalara veri yazmak ve okumak için kullanılan fonksiyonlar aşağıda verilmiştir;

```
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
```

Bu fonksiyon, `ptr` ile işaret edilen diziden verilen `stream` dosya göstericisi üzerinden dosyadan veri okur.

- `nmemb`, `size` ile verilen uzunluktan kaç adet okunacağını belirtir.
- Genellikle ikili dosyaları okumak için kullanılır ancak metin dosyaları için de kullanılabilir.
- Bir okuma hatası veya dosya sonu (`EOF`) oluşursa `nmemb`'den az olabilen, başarıyla okunan öğelerin sayısını döndürür. `size` veya `nmemb` sıfırsa, `fread` sıfır döndürür ve `ptr` tarafından işaret edilen belleğin içerikleri değişmez.

```
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
```

Bu fonksiyon, `ptr` ile işaret edilen dizideki verileri belirtilen `stream` dosya göstericisi üzerine yazar.

- `nmemb`, `size` ile verilen uzunluktan kaç adet yazılacağını belirtir.
- Genellikle ikili dosyaları okumak için kullanılır ancak metin dosyaları için de kullanılabilir.
- Bu fonksiyon başarıyla yazılan toplam öğe sayısını döndürür. Bu sayı `nmemb`'den azsa, bir hata oluşmuştur veya dosya sonuna ulaşılmıştır.

Aşağıda verilen örnekte görüldüğü üzere dosyaya yazılan yapılar aynen bellekte olduğu gibi yazılır. Dosya `wb+` modunda açıldığından, program her çalıştığında içeriği silerek yeniden aynı şeyleri yazacaktır.

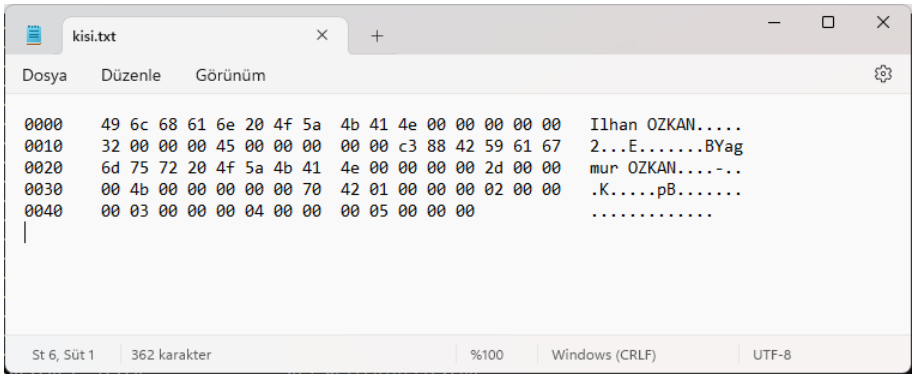
```
#include <stdio.h>
struct kisi {
    char adiSoyadi[16];
    int yas;
    char cinsiyet;
    float kilo;
};
typedef struct kisi Kisi;
int main() {
```

```
    FILE* dosyaGostericisi;
    Kisi kisi1={"Ilhan OZKAN",50,'E',100.0};
    Kisi kisi2={"Yagmur OZKAN",45,'K',60.0};
    int dizi[5]={1,2,3,4,5};
    // "kisi.bin" dosyası yoksa oluşturulur.
    // Varsa üzerine yazılır. Hem yazma hem okuma modunda dosya açılıyor.
    dosyaGostericisi = fopen("kisi.bin", "wb+");
    fwrite(&kisi1, sizeof(Kisi),1,dosyaGostericisi);
    fwrite(&kisi2, sizeof(Kisi),1,dosyaGostericisi);
    fwrite(dizi,sizeof(dizi),1,dosyaGostericisi);
    fwrite(dizi,sizeof(int),5,dosyaGostericisi);
    fclose(dosyaGostericisi);
    return 0;
}
```

Oluşan `kisi.bin` dosyasına ilk önce iki adet kişi yapısı kaydedilmiştir. Sonrasında 5 tamsayıdan oluşan dizi kaydedilmiştir. Dosya bellekte saklandığı şekliyle dosyaya kaydedildiğinden içeriği ikili olarak okunabilir. İkili dosyaları açan görüntüleyiciler ile açılıp görülebilir. Bir ikili dosyayı onaltılık rakamlara çevirip metin olarak görüntülemek için Windows ortamında aşağıdaki komut verilebilir;

```
C:\Users\ILHANOZKAN>certutil -encodehex kisi.bin kisi.txt
Input Length = 77
Output Length = 367
CertUtil: -encodehex command completed successfully.
```

Daha sonrasında dönüştürülen `kisi.txt` dosyası herhangi bir metin editörüyle açılıp okunabilir.



Şekil 24. Programın Oluşturduğu İkili Dosya İçeriği

Dosyaya hep aynı veri tipi (data type) yazılacağı gibi, birbirinden farklı olabilen yapı (struct) ve veri tipleri de yazılabilir. Burada önemli olan, yazıldığı sırada bu öğelerin okunmasıdır.