Dosya İşlemleri-1

Bu bölümde, veri dosyalarının C programları tarafından nasıl oluşturulduğunu, güncellendiğini ve işlendiğini inceleyelim. Hem sıralı erişimli hem de rastgele erişimli dosya işlemeyi dikkate alalım.

C, her dosyayı basitçe sıralı bir bayt akışı olarak görür (Şekil 11.1). Her dosya, ya bir dosya sonu işaretçisiyle ya da sistem tarafından yönetilen bir veri yapısında kaydedilen belirli bir bayt numarasıyla biter; bu, her platform tarafından belirlenir ve kullanıcıdan gizlenir.

Şekil 11.1 | C'nin n baytlık bir dosyanın görünümü.

Bir dosya açıldığında, onunla bir akış ilişkilendirilir. Program yürütme başladığında üç akış otomatik olarak açılır:

• standart giriş (klavyeden giriş alan),

• standart çıktı (çıktıyı ekranda gösteren) ve

• standart hata (ekranda hata mesajlarını gösteren)

Akışları, dosyalar ve programlar arasında iletişim kanalları sağlar. Örneğin, standart giriş akışı, bir programın klavyeden verileri okumasını sağlar. Standart çıkış akışı, bir programın verileri ekrana yazdırmasını sağlar.

DOSYA Yapısı

Bir dosyayı açmak, dosyayı işlemek için kullanılan bilgileri içeren bir FILE (Dosya) yapısına (<stdio.h> içinde tanımlanan) bir işaretçi döndürür. Her dizi öğesi, işletim sisteminin belirli bir dosyayı yönetmek için kullandığı bir dosya denetim bloğu (File Control Block-FCB) bilgisi içerir. Standart girdi, standart çıktı ve standart hata "stdin", "stdout" ve "stderr" kullanılarak değiştirilir.

Dosya İşleme fonksiyonu fgetc

Standart kütüphane, dosyalardan veri okumak ve dosyalara veri yazmak için birçok fonksiyon sağlar. fgetc fonksiyonu, getchar gibi, bir dosyadan bir karakter okur. fgetc fonksiyonu, bir karakterin okunacağı dosya için argüman olarak bir FILE (Dosya) işaretçisi alır. fgetc(stdin) çağrısı, standart girdi olan stdin'den bir karakter okur. Bu çağrı, getchar() çağrısına eşdeğerdir.

Dosya İşleme fonksiyonu fputc

fputc fonksiyonu, tıpkı putchar gibi, bir dosyaya bir karakter yazar. fputc fonksiyonu, argüman olarak yazılacak bir karakter ve karakterin yazılacağı dosya için bir işaretçi alır. fputc('a', stdout) fonksiyon çağrısı, 'a' karakterini standart çıktı olan stdout'a yazar. Bu çağrı putchar('a') ile eşdeğerdir.

Diğer Dosya İşleme Fonksiyonları

Standart girdiden veri okumak ve standart çıktıya veri yazmak için kullanılan diğer bazı fonksiyonlar, benzer şekilde adlandırılmış dosya işleme fonksiyonlarına sahiptir. Örneğin, fgets ve fputs fonksiyonları sırasıyla bir dosyadan bir satır okumak ve bir dosyaya bir satır yazmak için kullanılabilir.

Sırasal Erişimli Dosya Oluşturma

C bir dosyaya herhangi bir yapı dayatmaz. Bu nedenle, bir dosyanın kaydı gibi kavramlar C dilinin parçası değildir. Aşağıdaki örnek, kendi kayıt yapınızı bir dosyaya nasıl uygulayabileceğinizi gösterir.

Şekil 11.2, bir şirketin kredi müşterilerinin borçlu olduğu tutarları takip etmek için bir alacak hesabı sisteminde kullanılabilecek basit bir sıralı erişim dosyası oluşturur. Program her müşteri için bir hesap numarası, müşterinin adı ve müşterinin bakiyesini (yani müşterinin geçmişte aldığı mal ve hizmetler için şirkete borçlu olduğu tutar) alır. Her müşteri için elde edilen veriler, o müşteri için bir "kayıt" oluşturmaktadır. Hesap numarası, bu uygulamada kayıt anahtarı olarak kullanılır; dosya, hesap numarası sırasına göre oluşturulacak ve korunacaktır. Bu program, kullanıcının kayıtları hesap numarası sırasına göre girdiğini varsayar.

```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
   }
}
```

Şimdi bu programı inceleyelim. Satır 7, cfPtr'nin bir FILE yapısına işaretçi olduğunu belirtir. Bir C programı her dosyayı ayrı bir FILE (dosya) yapısıyla yönetir. Her açık dosya, dosyaya başvurmak için kullanılan FILE türünde ayrı olarak bildirilen bir işaretçiye sahip olmalıdır.

Satır 10, program tarafından kullanılacak dosyayı "clients.txt" olarak adlandırır ve dosyayla bir "iletişim hattı" kurar. cfPtr dosya işaretçisine, fopen ile açılan dosya için FILE yapısına bir işaretçi atanır. fopen fonksiyonu iki argüman alır:

• bir dosya adı (dosyanın konumuna götüren yol bilgisini içerebilir) ve

• bir dosya açma modu.

Dosya açma modu "w", dosyanın yazılmak üzere açılacağını belirtir. Bir dosya yoksa ve yazmak için açılmışsa, fopen dosyayı oluşturur. Mevcut bir dosya yazılmak üzere açılırsa, dosyanın içeriği uyarı yapılmadan atılır. Programda, cfPtr dosya işaretçisinin NULL olup olmadığını belirlemek için if ifadesi kullanılır (yani, dosya mevcut olmadığı veya kullanıcının dosyayı açma izni olmadığı için açılmaz). NULL ise, program bir hata mesajı yazdırır ve sonlandırılır. Aksi takdirde, program girişi işler ve dosyaya yazar.

Program, kullanıcıdan her kayıt için çeşitli alanları girmesini veya veri girişi tamamlandığında dosya sonu girmesini ister. Aşağıdaki tablo, çeşitli bilgisayar sistemleri için dosya sonu girmek için tuş kombinasyonlarını listelemektedir.

İşletim sistemi	Tuş kombinasyonu
Linux/Mac OS X/UNIX	<Ctrl> d
Windows	<Ctrl> z ardından Enter'a basın

Satır 25, dosya sonu göstergesinin stdin için ayarlanıp ayarlanmadığını belirlemek için feof fonksiyonunu kullanır. Dosya sonu göstergesi, programa işlenecek başka veri olmadığını bildirir. feof fonksiyonunun argümanı, dosya sonu göstergesi (bu durumda stdin) için test edilen dosyanın bir işaretçisidir. Bu programda feof çağrısını içeren while ifadesi, dosya sonu göstergesi ile karşılaşıncaya kadar çalışmaya devam eder. Satır 26, client.txt dosyasına veri yazar. Veriler, dosyayı okumak için tasarlanmış bir program tarafından daha sonra alınabilir (bkz. Bölüm 11.4)., fprintf'in ayrıca verilerin yazılacağı dosya için bir argüman olarak bir dosya işaretçisi alması dışında fprintf fonksiyonu printf fonksiyonuna eşdeğerdir. fprintf fonksiyonu, aşağıdaki gibi dosya işaretçisi olarak stdout'u kullanarak verileri standart çıktıya verebilir:

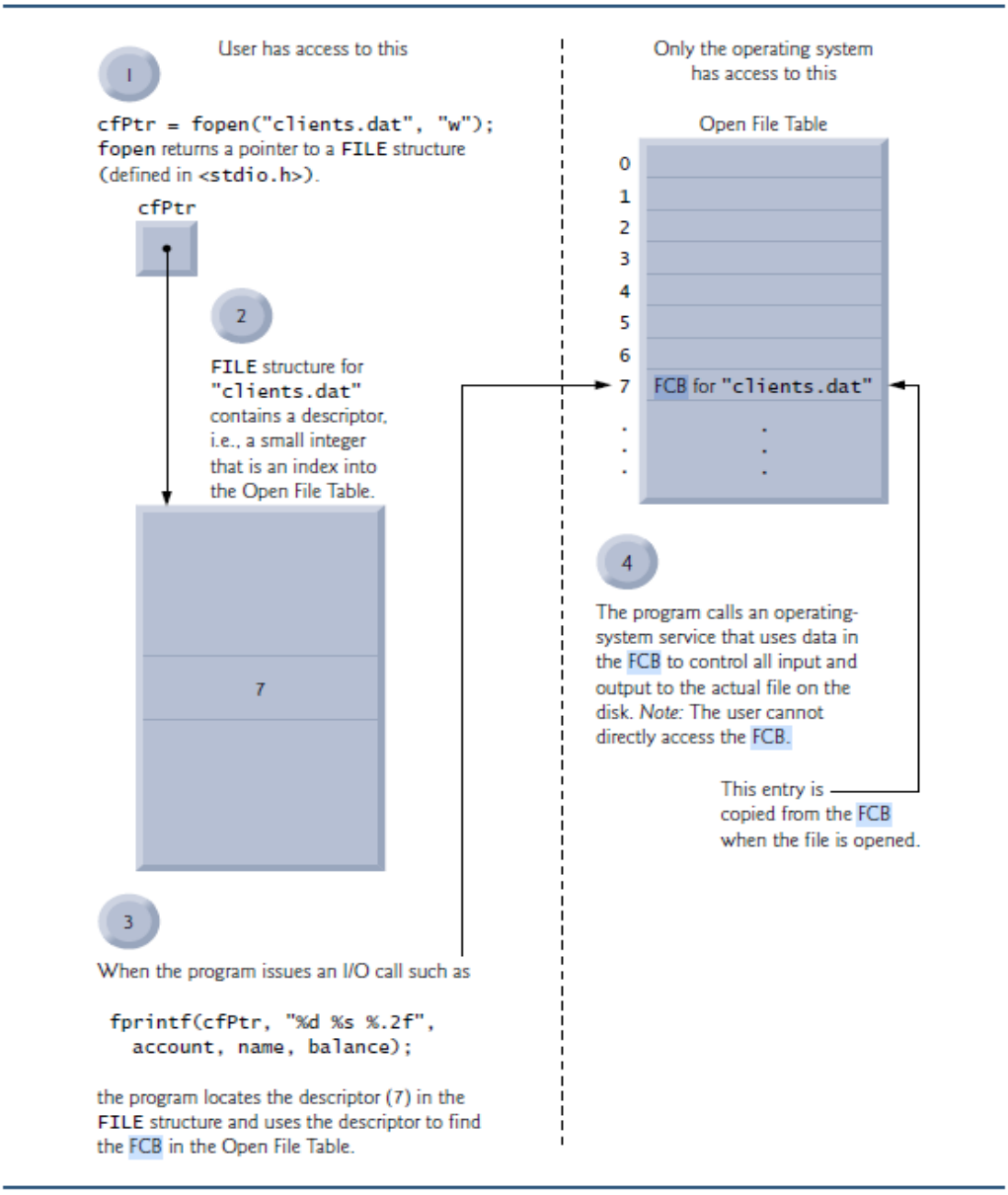
```
fprintf(stdout, "%d %s %.2f\n", account, name, balance);
```

Kullanıcı dosya sonuna girdikten sonra, program client.txt dosyasını fclose (satır 31) ile kapatır ve sonlandırır. fclose işlevi ayrıca (dosya adı yerine) dosya işaretçisini bağımsız değişken olarak alır. fclose işlevi açıkça çağrılmazsa, program yürütme sona erdiğinde işletim sistemi normalde dosyayı kapatır.

NOT: Bir dosyayı kapatmak, diğer kullanıcıların veya programların beklediği kaynakları serbest bırakabilir, bu nedenle, işletim sisteminin program sonlandırıldığında kapatmasını beklemek yerine, her dosyayı artık ihtiyaç kalmadığı anda kapatmak daha doğrudur.

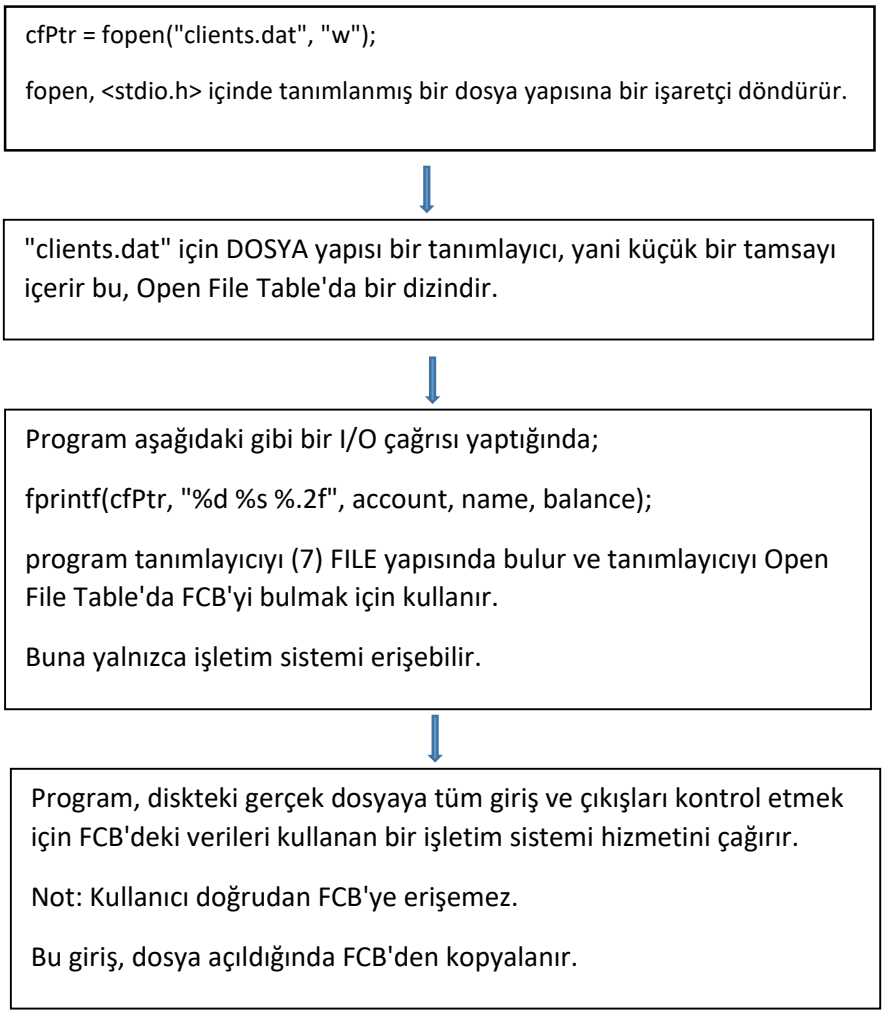
Şekil 11.2'deki program için örnek yürütmede, kullanıcı beş hesap için bilgi girer, ardından veri girişinin tamamlandığını belirtmek için dosya sonunu girer. Örnek yürütme, veri kayıtlarının dosyada gerçekte nasıl göründüğünü göstermez. Dosyanın başarıyla oluşturulduğunu doğrulamak için bir sonraki bölümde dosyayı okuyan ve içeriğini yazdıran bir program inceleyeceğiz.

Şekil 11.4, FILE işaretçileri, FILE yapıları ve FCB'ler (File Control Blocks) arasındaki ilişkiyi göstermektedir. "clients.txt" dosyası açıldığında, dosya için bir FCB belleğe kopyalanır. Şekil, fopen tarafından döndürülen dosya işaretçisi ile işletim sistemi tarafından dosyayı yönetmek için kullanılan FCB arasındaki bağlantıyı gösterir. Programlar hiçbir dosyayı, bir dosyayı veya birkaç dosyayı işleyemez. Bir programda kullanılan her dosya, fopen tarafından döndürülen farklı bir dosya işaretçisine sahip olacaktır. Dosya açıldıktan sonraki tüm dosya işleme fonksiyonları, uygun dosya işaretçisi ile dosyaya başvurmalıdır.



Şekil 11.4 | FILE işaretçileri, FILE yapıları ve FCB'ler arasındaki ilişki.

Blok şemalarla açıklayacak olursak;



Dosya Açma Modları

Dosyalar, Aşağıdaki tabloda özetlenen çeşitli modlardan birinde açılabilir. Tabloda her dosya açma modu, bir ikili (binary) moda sahiptir.

Mod	Açıklama
r	Okumak için mevcut bir dosyayı açar
w	Yazmak için bir dosya oluşturur. Dosya zaten varsa, mevcut içeriği yok sayar.
a	Dosyanın sonuna yazmak için bir dosya açar veya oluşturur; yani, write işlemleri dosyaya veri ekler.
r+	Güncelleme için mevcut bir dosyayı açar (okuma ve yazma).
w+	Okumak ve yazmak için bir dosya oluşturur. Dosya zaten varsa, geçerli içeriği yok sayar.
a+	Okumak ve güncellemek için bir dosya açar veya oluşturur; tüm yazma işlemi dosyanın sonunda yapılır - yani yazma işlemleri dosyaya veri ekler.
rb	İkili (binary) modda okumak için mevcut bir dosyayı açar.
wb	İkili (binary) modda yazmak için bir dosya oluşturur. Dosya zaten varsa, mevcut içeriği yok sayar.

ab	Ekleme: ikili (binary) modda dosyanın sonuna yazmak için bir dosya açar veya oluşturur.
rb+	Güncelleme (okuma ve yazma) için mevcut bir dosyayı ikili modda açar.
wb+	İkili modda güncelleme için bir dosya oluşturur. Dosya zaten varsa, geçerli içeriği yok sayar.
ab+	Ekleme: ikili modda güncelleme için bir dosya açar veya oluşturur; dosyanın sonuna yazma işlemi yapılır.

Ayrıca C11, w, w+, wb veya wb+ modlarının sonuna bir x ekleyerek belirttiğiniz özel yazma modu sağlar. Özel yazma modunda, dosya zaten varsa veya oluşturulamıyorsa fopen başarısız olur. Bir dosyayı özel yazma modunda açmak başarılıysa ve temeldeki sistem özel dosya erişimini destekliyorsa, dosya açıkken yalnızca sizin programınız erişebilir. (Bazı derleyiciler ve platformlar özel yazma kipini desteklemez.) Bir dosyayı herhangi bir kipte açarken bir hata oluşursa, fopen NULL değerini döndürür.

NOT:

- Var olmayan bir dosyayı okumak için açmak bir hata verir.
- Dosyaya uygun erişim hakları verilmeden (bu işletim sistemine bağlıdır) bir dosyayı okumak veya yazmak için açmak bir hata verir.
- Boş alan olmadığında bir dosyayı yazmak için açmak bir çalışma zamanı hatasıdır.
- Bir dosyanın güncelleme modunda ("r+") açılması gerekirken yazma modunda ("w") açılması, dosyanın içeriğinin atılmasına neden olur.
- Bir dosyanın içeriğinin değiştirilmemesi gerekiyorsa yalnızca okumak için açılmalıdır. (güncelleme için değil). Bu, dosya içeriğinin istenmeden değiştirilmesini önler.

Sıralı Erişim Dosyasından Veri Okuma

Veriler, gerektiğinde işlenmek üzere alınabilmesi için dosyalarda saklanır. Önceki bölümde, sıralı erişim için bir dosyanın nasıl oluşturulacağı gösterildi. Bu bölümde, verilerin bir dosyadan sıralı olarak nasıl okunacağını gösterir.

Şekil 11.6, Şekil 11.2'deki program tarafından oluşturulan "clients.txt" dosyasındaki kayıtları okur ve içeriklerini yazdırır. 7. satır, cfPtr'nin bir FILE işaretçisi olduğunu gösterir. Satır 10, "clients.txt" dosyasını okumak ("r") için açmaya çalışır ve başarılı bir şekilde açılıp açılmadığını belirler (yani, fopen NULL döndürmez). Satır 19, dosyadan bir "kayıt" okur. fscanf fonksiyonu, scanf fonksiyonuna eşdeğerdir, ancak fscanf, verilerin okunduğu dosya için bir argüman olarak bir dosya işaretçisi alır. Bu ifade ilk kez yürütüldükten sonra, account, name ve balance dosyadaki ilk değeri alacaktır. İkinci fscanf ifadesi (satır 24) her yürütüldüğünde, program dosyadan başka bir kaydı okur ve hesap, ad ve bakiye yeni değerler alır. Program dosyanın sonuna geldiğinde dosya kapatılır (27. satır) ve program sonlandırılır.

```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
   }
}
```

Dosya Konumu İşaretçisini Sıfırlama

Bir dosyadan sıralı olarak veri almak için, bir program normalde dosyanın başından okumaya başlar ve istenen veri bulunana kadar tüm verileri ardışık olarak okur. Bir programın yürütülmesi sırasında bir dosyadaki verilerin birkaç kez (dosyanın başından itibaren) sıralı olarak işlenmesi istenebilir. Aşağıdaki ifade, dosyadaki okunacak veya yazılacak bir sonraki baytın sayısını gösteren bir programın dosya konumu işaretçisinin, cfPtr tarafından işaret edilen dosyanın başına (yani bayt 0) yeniden konumlandırılmasına neden olur.

```
rewind(cfPtr);
```

Dosya konumu işaretçisi gerçekten bir işaretçi değildir. Bunun yerine dosyada bir sonraki okuma veya yazma işleminin gerçekleşeceği baytı belirten bir tamsayı değeridir. Buna bazen dosya ofseti denir. Dosya konumu işaretçisi, her dosyayla ilişkili FILE yapısının bir üyesidir.

Kredi Sorgulama Programı

Şekil 11.7'deki program, bir kredi yöneticisinin bakiyesi sıfır olan (yani borcu olmayan müşteriler), kredi bakiyesi olan müşteriler (yani şirketin borçlu olduğu müşteriler) ve borç bakiyesi olan müşterilerin (yani, alınan mal ve hizmetler için şirkete borcu olan müşteriler) listelerini almasına olanak tanır. Alacak bakiyesi negatif bir tutardır; borç bakiyesi pozitif bir miktardır.

Program bir menü görüntüler ve kredi yöneticisinin dört seçenekten birini girmesine izin verir:

• Seçenek 1, sıfır bakiyeli hesapların bir listesini üretir.

• Seçenek 2, kredi bakiyesi olan hesapların bir listesini üretir.

• Seçenek 3, borç bakiyesi olan hesapların bir listesini üretir.

• Seçenek 4, programın yürütülmesini sonlandırır.

Bu tür sıralı dosyadaki veriler, diğer verileri yok etme riski olmadan değiştirilemez. Örneğin, "white" adının "Worthington" olarak değiştirilmesi gerekiyorsa, eski adın üzerine yazılamaz. white için kayıt dosyaya şu şekilde yazılmıştır;

```
300 White 0.00
```

Kayıt, yeni ad kullanılarak dosyadaki aynı konumdan başlayarak yeniden yazılırsa;

```
300 Worthington 0.00
```

Yeni kayıt, orijinal kayıttan daha büyüktür (daha fazla karakter içerir). "Worthington"daki ikinci "o"nun ötesindeki karakterler, dosyadaki bir sonraki sıralı kaydın başlangıcının üzerine yazılacaktır. Buradaki sorun, fprintf ve fscanf kullanan biçimlendirilmiş giriş/çıkış modelinde alanların - ve dolayısıyla kayıtların - boyutunun değişebilmesidir.

Bu nedenle, fprintf ve fscanf ile sıralı erişim genellikle yerinde kayıtları güncellemek için kullanılmaz. Bunun yerine, dosyanın tamamı genellikle yeniden yazılır.

```c
// Fig. 11.7: fig11_07.c
// Credit inquiry program
#include <stdio.h>

// function main begins program execution
int main(void)
{
   FILE *cfPtr; // clients.txt file pointer

   // fopen opens the file; exits program if file cannot be opened
   if ((cfPtr = fopen("clients.txt", "r")) == NULL) {
      puts("File could not be opened");
   }
   else {

      // display request options
      printf("%s", "Enter request\n"
         " 1 - List accounts with zero balances\n"
         " 2 - List accounts with credit balances\n"
         " 3 - List accounts with debit balances\n"
         " 4 - End of run\n? ");
      unsigned int request; // request number
      scanf("%u", &request);

      // process user's request
      while (request != 4) {
         unsigned int account; // account number
         double balance; // account balance
         char name[30]; // account name

         // read account, name and balance from file
         fscanf(cfPtr, "%d%29s%lf", &account, name, &balance);

         switch (request) {
            case 1:
               puts("\nAccounts with zero balances:");

               // read file contents (until eof)
               while (!feof(cfPtr)) {
                  // output only if balance is 0
                  if (balance == 0) {
                     printf("%-10d%-13s%7.2f\n",
                        account, name, balance);
                  }

                  // read account, name and balance from file
                  fscanf(cfPtr, "%d%29s%lf",
                     &account, name, &balance);
               }

               break;
```

```
52              case 2:
53                 puts("\nAccounts with credit balances:\n");
54
55                 // read file contents (until eof)
56                 while (!feof(cfPtr)) {
57                    // output only if balance is less than 0
58                    if (balance < 0) {
59                       printf("%-10d%-13s%7.2f\n",
60                          account, name, balance);
61                    }
62
63                    // read account, name and balance from file
64                    fscanf(cfPtr, "%d%29s%lf",
65                       &account, name, &balance);
66                 }
67
68                 break;
69              case 3:
70                 puts("\nAccounts with debit balances:\n");
71
72                 // read file contents (until eof)
73                 while (!feof(cfPtr)) {
74                    // output only if balance is greater than 0
75                    if (balance > 0) {
76                       printf("%-10d%-13s%7.2f\n",
77                          account, name, balance);
78                    }
79
80                    // read account, name and balance from file
81                    fscanf(cfPtr, "%d%29s%lf",
82                       &account, name, &balance);
83                 }
84
85                 break;
86           }
87
88           rewind(cfPtr); // return cfPtr to beginning of file
89
90           printf("%s", "\n? ");
91           scanf("%d", &request);
92        }
93
94        puts("End of run.");
95        fclose(cfPtr); // fclose closes the file
96     }
97  }
```