Unidad 6 Archivos

Archivos

Estructura compuesta por datos almacenados en forma organizada en un **dispositivo de almacenamiento secundario**: discos rígidos, CD, pendrive, DVD

- **Ventajas del uso de archivos**
  - Datos pueden ser usados por distintos programas y en distintos procesos .
  - Capacidad de almacenamiento de memoria secundaria superior a memoria principal
  - El uso de archivos permite tener en memoria principal sólo aquella parte de datos que necesita ser accedida por el programa en un momento dado,
  - El tamaño de los archivos está limitado sólo por la cantidad de memoria secundaria disponible.
- **Desventajas**

  El acceso a los datos de un archivo es más lento que los datos residentes en memoria principal.

  tiempos mecánicos **Vs** tiempos electrónicos

  (orden de los milisegundos) (orden nanosegundos)

Organización de Archivos

- **Organización Secuencial :** las componentes se ubican en posiciones físicas contiguas en el soporte de almacenamiento

- **Organización Directa:** las componentes ocupan posiciones físicas relacionadas mediante una clave, que puede ser un dato de la componente o el resultado de un cálculo

Criterios de clasificación de archivos

- **Por la dirección del flujo de datos**

  Archivos de entrada
  Archivos de salida.
  Archivos de entrada / salida.

- **Según el tipo de valores permitidos a cada byte**

  Archivos de texto — **Con formato**
  Archivos Binarios — **Sin formato**

- **Según el tipo de acceso** (técnica utilizada para leer o grabar las componentes)

  Secuencial — **Por índice**
  Directa o aleatoria — **Dinámico**

- **Según la longitud de registro**

  Archivos de longitud variable
  Archivos de longitud constante

Archivos organizados secuencialmente con acceso secuencial

Declaración de un archivo en el lenguaje C

FILE *archi;

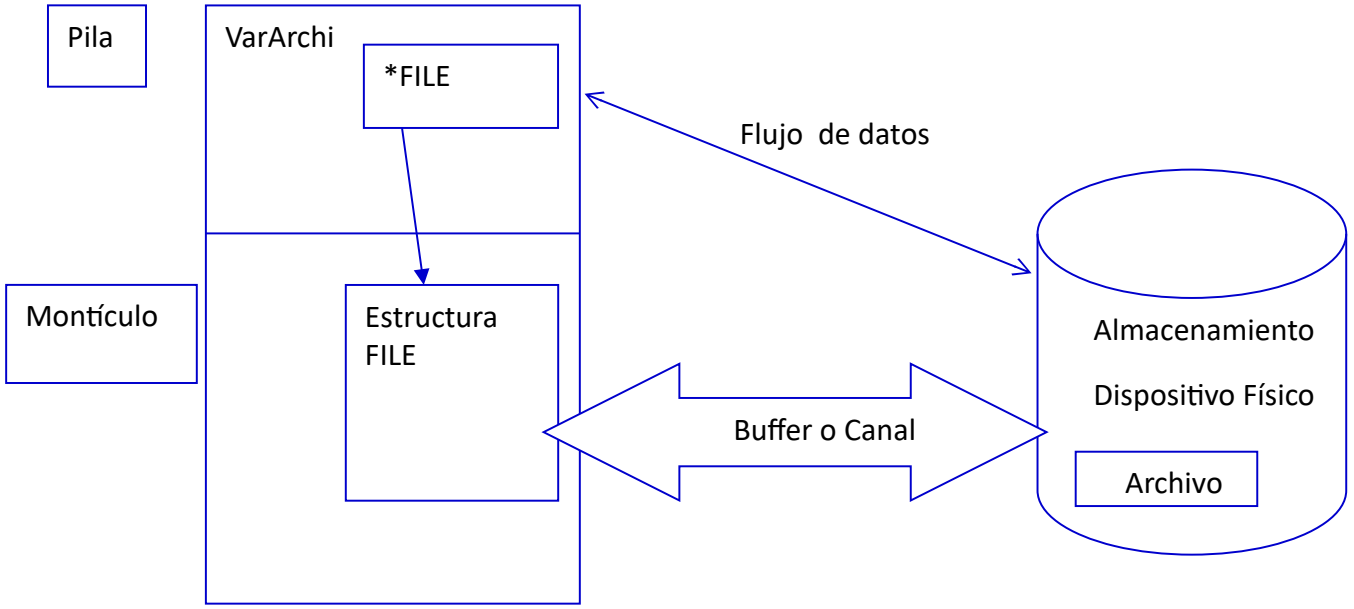
FILE: estructura de dato (**struct**) definida por C en la biblioteca "stdio.h".

- Con esta declaración, **en tiempo de compilación** se **habilita un espacio de memoria para una estructura de tipo FILE,** que queda referenciada a través de la variable declarada.
- La definición de esta estructura depende del compilador, pero en general mantiene información sobre:
  - la posición actual de lectura/escritura
  - el estado del archivo
  - un puntero a un buffer, área de memoria que sirve de almacenamiento intermedio entre el archivo físico y el programa que lo maneja.

A medida que se realizan operaciones sobre el archivo la estructura se actualiza.

Estructura FILE

```
typedef struct {
        short level;                    // nivel de ocupación del buffer
        unsigned flag;                  //  indicadores de control
        char fd;                        // descriptor del archivo
        char hold;                      //  carácter de ungetc()
        short bsize;                    //  tamaño del buffer
        unsigned char *buffer;          // puntero al buffer
        unsigned char *curp;            // posicion en curso
        short token;                    // se emplea para control
                }FILE;                  // tipo definido con el nombre FILE
```



- **En realidad, una variable de tipo FILE * representa un flujo de datos que se asocia con un dispositivo físico de entrada/salida (el archivo "real" estará almacenado en disco).**

Operaciones básicas en el manejo de archivos

- Crear y/o Abrir
- Grabar información.
- Recuperar información
- Modificar información
- Eliminar información
- Cerrar

Operaciones básicas : Creación y/o apertura

Se crea un manejador lógico o vínculo entre la variable tipo archivo y el archivo físico.

fopen: vincula una variable del programa con el archivo físico en el dispositivo externo

Formato: **FILE * fopen (char *nombre, char *modo);**

```
FILE *archi ;
archi = fopen("Datos.dat", "wb");
```

Parámetros:

nombre: cadena de caracteres que contiene un nombre de **archivo físico** .

modo: especifica **modo de acceso** en que se va utilizar el archivo, lectura, escritura o para agregar información, y **tipo de valores** permitidos a cada byte, de texto (**t**) o binario (**b**).

Resultado: puntero a FILE si la operación es exitosa, NULL caso contrario

Creación y/o apertura

Modo	Apertura
“r”	Sólo lectura: el archivo debe existir
“w”	Escritura: se crea uno nuevo o se sobrescribe si ya existe.
“a”	Añadir: se abre para escritura situándose el puntero al final del archivo. Si no existe, es creado.
“rb”	Lectura: abre un archivo binario
“wb”	Escritura: se crea un archivo binario nuevo o se sobrescribe si ya existe.
“ab”	Añadir: el archivo binario se abre para escritura. El puntero se sitúa al final . Si el archivo no existe, es creado.
“r+”	Lectura y/o escritura: **debiendo asegurar la existencia del mismo**.
“w+”	Lectura y escritura: se crea un archivo nuevo o se sobrescribe si existe.
“a+”	Añadir, lectura y/o escritura: el puntero se sitúa al final del archivo. Si no existe, es creado. Se dispone para lectura y escritura.

Ejemplo **FILE * archi ;** //declaración de la variable puntero a File

archi=fopen(“Datos.dat”, “w”); //apertura del archivo Datos para escritura

Operaciones básicas

Grabar información

Consiste en guardar una componente en el archivo físico abierto.

Recuperar información

- abrir archivo
- leer componente (*se almacena en memoria para su manipulación*)

Modificar información

Consiste en alterar una componente en un archivo físico.

Eliminar información

Proceso de quitar componentes de un archivo.

Operaciones básicas

Cerrar

Libera el vínculo que une el programa a través de la variable tipo archivo, con el archivo físico vinculado.

Función fclose **int fclose(FILE *archivo);**

Parámetro: un puntero a la estructura FILE del archivo que se quiere cerrar, NULL si se quiere cerrar todos los archivos abiertos.

Resultado: cero indica que el archivo ha sido correctamente cerrado, caso contrario indica que hubo un error.

Archivos organizados secuencialmente

Componentes ubicadas en **posiciones físicas contiguas en soporte** de almacenamiento.

Pueden accederse: **secuencialmente o en forma directa**

Características de los archivos secuenciales:

- Escritura de nuevos datos al final del archivo.
- Lectura de una componente concreta en forma secuencial desde el lugar donde está posicionado el puntero. *Necesidad de posicionarse al comienzo del archivo*
- Sólo se pueden abrir para lectura o escritura, nunca de los dos modos a la vez.

Archivos de caracteres

Archivos de texto

Archivos binarios

Archivos de Caracteres

Los archivos de caracteres son una secuencia de caracteres almacenados en un dispositivo de almacenamiento secundario e identificado por un nombre de archivo.

Grabar información

int **fputc** (int caracter, FILE *archivo);

```
fputc(c,archi);          //escritura en el archivo
```

Recuperar información

int **fgetc** (FILE *archivo);

```
c=fgetc(archi);   //lectura del caracter del archivo
```

Archivos de texto

Un archivo de texto contiene un conjunto de "líneas" de caracteres de longitud variable y están separadas por un salto de línea.

Forma 1

Grabar información

```
int fputs(const char *s, FILE *archivo);
        char texto[200];
        fputs (texto,archi);
```

Recuperar información

```
char *fgets(char *s, int n, FILE * archivo);
        char texto[200]
        fgets(texto,200, archi);
```

Archivos de textos

Forma 2

Recuperar información

int **fscanf** (FILE *fichero, const char *formato, argumento, ...);

int fscanf (*archivo*, *formato*, *argumento*, ...);

```
char buffer[50];
fscanf(fp, "%s" ,buffer);
```

Grabar información

int **fprintf** (FILE *archivo, const char *formato, argumento, ...);

int fprintf (*archivo*, *formato*, *argumento*, ...);

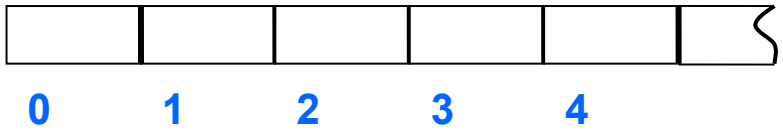
```
char buffer[50];
fprintf (fp, "%s", buffer);
```

Archivos sin formato

Estos archivos son tratados como un bloque de datos binarios (en bytes) que tan solo tienen significado si se conoce la estructura: distribución y tipos de datos

Acceso: secuencial o directo.

Almacenamiento:



Archivos sin formato

EJEMPLO 2

Generar un archivo de nombre “alumnos.dat”, que contenga la siguiente información correspondiente a los alumnos que cursan la materia Programación Procedural: Nombre , Número de registro y resultado de un parcial (A aprobado- R Reprobado). Para este archivo generado, se pide:

1- Realizar un listado que contenga el número de registro y nombre de todos los alumnos aprobados.

2- Dado el nombre de un alumno, indicar si está aprobado.

- Generar ➡ **FILE * fopen (char *nombre, char *modo);**
  Grabar componentes
- Recuperar información
  - Listar ➡ **Abrir para lectura**
    Recorrer el archivo: Lectura de componentes
  - Buscar ➡ **Abrir para lectura**
    Recorrer el archivo: Lectura de componentes

Archivos sin formato

Funciones de C para manipular archivos sin formato

- **Grabar información en el archivo**

Función fwrite

size_t fwrite(void *puntero, size_t tamaño, size_t nregistros, FILE *archivo);

parámetros :

- puntero a la zona de memoria **donde están almacenados** los datos a grabar.
- El tamaño de cada bloque.
- El número de bloques a grabar.
- Un puntero a FILE del archivo correspondiente.

valor de retorno: número de **bloques escritos.**

Archivos sin formato

Funciones de C para manipular archivos sin formato

- Recuperar información desde el archivo

Función fread

```
size_t fread(void *puntero, size_t tamaño, size_t numero, FILE *archivo);
```

parámetros :

- puntero a la zona de memoria donde se almacenarán los **datos recuperados del archivo**
- El tamaño de cada bloque.
- El número de bloques a leer.
- Un puntero a FILE del archivo correspondiente.

valor de retorno: número de **bloques leídos.**

Archivos sin formato

Funciones de C para manipular archivos sin formato

Función feof

```
int  feof ( FILE *archivo);
```

parámetro:

- Un puntero a FILE del archivo correspondiente**.**

valor de retorno:

- **cero** si no se ha alcanzado el fin de archivo
- **distinto de cero** si llegó al final

Cerrar archivo

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

parámetro:

- Un puntero a FILE del archivo correspondiente.

valor de retorno:

- **cero** si la apertura se realizó correctamente
- **distinto de cero** en caso de error

Detalles:
Cierra un fichero. Si hay datos en buffer pendientes de volcar, se vuelcan antes de cerrar. Un fichero abierto con "fopen" se cerrará también automáticamente al terminar el programa. Si se intenta cerra un fichero no abierto, se obtendrá una "violación de segmento"

Archivos sin formato

EJEMPLO 2: Generar un archivo de nombre “alumnos.dat”, que contenga la siguiente información correspondiente a los alumnos que cursan la materia Programación Procedural:

Nombre , Número de registro y resultado de un parcial (A aprobado- R Reprobado).

Para este archivo generado, se pide:

1- Realizar un listado que contenga el número de registro y nombre de todos los alumnos aprobados.

2- Dado el nombre de un alumno, indicar si está aprobado.

```
typedef struct
 { char nombre[30];
   int reg;
   char nota;
 } alumno;
```

```
int main(void)
 {
     FILE *archi;
     char nom[30];
     if ((archi=fopen("alumnos.dat","wb"))==NULL)
         printf("hay error1 \n");       /* Se abre para escritura */
     else
         { cargar(archi);
           fclose(archi);
          }
   }
```

Archivos sin formato

```
void cargar (FILE * archi)
{  alumno c;
   printf("\n ingrese el nombre del alumno (termina con FIN): ");
   gets(c.nombre);
   while( strcmp((c.nombre),"FIN"))
     {     printf("\n ingrese el número de registro: ");
           scanf("%d",&c.reg);
            printf("\n ingrese La Nota  (A=aprobado R=reprobado) : ");
           fflush(stdin);
            c.nota=getch();
            fwrite(&c,sizeof(c),1,archi);
            printf("\n ingrese el nombre del alumno (termina con FIN): ");
            gets(c.nombre);
       }
}
```

fwrite(&c, sizeof(c), 1, archi); toma los datos almacenados en la variable c, calcula el tamaño de esos datos con sizeof(c), y luego escribe esos datos en el archivo apuntado por archi. El número 1 indica que se está escribiendo un solo elemento de ese tamaño en el archivo. Por lo tanto, esta sentencia se utiliza para escribir un bloque de datos binarios en el archivo desde la dirección de memoria de la variable c.

Archivos sin formato

Recorrido del archivo y búsqueda

```
int main(void)
 {
    FILE *archi;
    char nom[30];
    if ((archi=fopen("alumnos.dat","rb"))==NULL)        /* Se abre para lectura*/
        printf("hay error1 \n");
    else
        {  mostrar(archi);
           printf("\n ingese nombre del alumno a buscar: "); gets(nom);
           buscar(archi, nom);
           fclose(archi);
        }
    getchar();
}
```

Archivos sin formato

Recorrido del archivo y búsqueda

```
void mostrar(FILE* xarchi )
{alumno c;
    rewind(xarchi);
    fread(&c,sizeof(c),1,xarchi);       //Se debe hacer una lectura anticipada

     while(feof(xarchi)==0 )
       {  if(c.nota=='A')
          printf("\n Nombre Alumno %s Registro %d nota %c",c.nombre,c.reg,c.nota);
          fread(&c,sizeof(c),1,xarchi);
       }
 }
```

Archivos sin formato

Recorrido del archivo y búsqueda

```
void buscar(FILE *xarchi,char nom[30])
{    alumno c;
      int b=0;

      rewind(xarchi);
      fread(&c,sizeof(c),1,xarchi);
      while((!feof(xarchi) )&&(b==0))
      {   if (strcmp(c.nombre,nom)==0)
           {
           printf("\n  Alumno %s Encontrado Registro %d nota %c",c.nombre
                                                              ,c.reg,c.nota);
           b=1;
              }
          else fread(&c,sizeof(c),1,xarchi);
        }
      if(b==0)
          printf("\n  Alumno %s NO se encontro",nom);
      getch();
  }
```

Archivos sin formato

Otra forma de Recorrido

```
void mostrar(FILE* xarchi )
{alumno c;
   rewind(xarchi);
   while(fread(&c,sizeof(c),1,xarchi))
   {
    if(c.nota=='A')
       printf("\n Nombre Alumno %s Registro %d nota %c",c.nombre,c.reg,c.nota);
    }
   }
```

fread devuelve el número de bloques leídos, por lo tanto leemos hasta que no hayan bloques.

Archivos sin formato

Lectura de varias componentes de un archivo

Utilizando el archivo "alumnos.dat, mostrar los 20 primeros alumnos del archivo ordenados alfabéticamente por nombre.

```
void listar_primeros(FILE* xarchi ,int cm)
{alumno c[m];
 int cant, i;
    rewind(xarchi);
    cant=fread( c,sizeof(alumno),cm,xarchi);  /* lee 20  Alumnos*/
    printf("Se leyeron %d registros \n",cant);
    ordena(c,cant);          // invoca la función para ordenar  arreglo
    for (i=0;i<cant; i++)         //muestra el arreglo ordenado
 printf("Nombre: %s Registro: %d \n", c[i].nombre,  c[i].reg);
 }
```

```
int main(void)
{ int m=20;
FILE *archi;
  :
  :
else

{listar_primeros(archi,m);

    fclose(archi);       }
}
```

Archivos organizados secuencialmente con acceso directo

Archivos organizados secuencialmente con acceso directo

- Lenguaje C, permite manipular archivos organizados en forma secuencial y accederlos en forma directa.
- El archivo tiene en las componente un **elemento llamado clave, que está en relación con la posición física del almacenamiento**. Esto permite la ubicación precisa de la componente en el archivo.

Cuando se accede en forma directa a un archivo de acceso directo, hay que tener en cuenta algunas características:

- La escritura de nuevos datos se realiza al final del archivo sin que se pierda la relación de la clave con la posición física de la componente.
- Para leer una componente concreta se utiliza la clave para ubicar dicha componente

Archivos organizados secuencialmente con acceso directo

Función fseek: Esta función sirve para ubicar el puntero en el archivo, para lectura o escritura en el lugar deseado.

```
int  fseek(FILE *archivo, long int desplazamiento, int origen);
```

parámetros

- **puntero** a FILE del archivo que se quiere acceder
- **valor** del desplazamiento , expresado en bytes (positivo o negativo).
- **punto** de origen desde el que se calculará el desplazamiento.

valor de retorno:

- cero si fue exitosa la función,
- distinto de cero si hubo error.

Archivos organizados secuencialmente con acceso directo

Función fseek: Posibles valores del parámetro **origen**

Nombre	Valor	Descripción
SEEK_SET	0	desplazamiento se cuenta desde el principio del archivo.(primer byte del archivo, desplazamiento cero)
SEEK_CUR	1	el desplazamiento se cuenta desde posición actual del puntero.
SEEK_END	2	el desplazamiento se cuenta desde el final del archivo.

Función fgetpos

Esta función se utiliza para obtener la posición actual del puntero del archivo. Dicha posición está dada en bytes.

int fgetpos (FILE *archivo, fpos_t *posicion);

parámetros

- **puntero** a FILE
- **la ubicación del registro dentro del archivo expresada en bytes.**

valor de retorno:

- cero si fue exitosa la función,
- distinto de cero si hubo error.

fpos_t *es un tipo de dato definido por el lenguaje que se utiliza para hacer referencia a las posiciones de los punteros de archivos.*

EJEMPLO 4:

El siguiente código genera un archivo de productos con los siguientes datos: código de producto, descripción, precio unitario y stock. (El código de producto está ordenado en forma ascendente y consecutiva a partir de 100).

```
typedef struct
{   int cod;
    char descrip[30];
    float pu;
    int  stock;
} prod;

int main(void)
{ FILE *archi;   int cod;
 if ((archi=fopen("producto.dat","w"))==NULL)          /* Se abre para escritura
*/
        printf("hay error1 \n");
 else   {
        cargar(archi);
        fclose(archi);
        mostrar(archi);
        fclose(archi);
      }
}
```

```
typedef struct
 {   int cod;
      char descrip[30];
      float pu;
       int  stock;
} prod;

void cargar (FILE * xarchi)
{ prod p;
 fpos_t x;
 printf("ingrese producto (termina con FIN)“);
 gets(p.descrip);
while( strcmp((p.descrip),"FIN"))
{
fseek(xarchi,0, SEEK_END);
 fgetpos(xarchi,&x);
 int cod=(int)(x/sizeof(prod))+100;
 printf("El codigo del nuevo producto es: %d",cod); p.cod=cod;
 printf("\n ingrese el stock : "); scanf("%d",&p.stock);
 printf("\n ingrese el precio unitario : "); scanf("%f",&p.pu);
fwrite(&p,sizeof(p),1,xarchi);
 printf("\n ingrese el nombre del producto (termina con FIN): "); fflush(stdin);
 gets(p.descrip);
}
}
```

```
int main(void)

{

 FILE *archi;
 int cod;
 if
((archi=fopen("producto.dat","wb"))==NU
LL)
printf("hay error1 \n");
else  {  cargar(archi);
          fclose(archi);
          mostrar(archi);
          fclose(archi);
        }

 }
```

```
typedef struct

{   int cod;
char descrip[30];
float pu;
int  stock;
} prod;


void mostrar(FILE* xarchi )
{
prod p;

if ((xarchi=fopen("producto.dat","rb"))==NULL)
    printf(" Error al abrir el archivo \n");

else
  { while( fread(&p, sizeof(p), 1, xarchi))
     printf("\n Nombre %s codigo %d Stock %d precio %f", p.descrip, p.cod,
p.stock, p.pu);
  }
}
```

```
int main(void)
    {

   FILE *archi;
    int cod;
    if
   ((archi=fopen("producto.dat","w‟b))==NULL
   )
   printf("hay error1 \n");
   else   {

        cargar(archi);
          fclose(archi);
          mostrar(archi);
           fclose(archi);
       }
```

EJEMPLO 5: A partir del archivo de productos generado se pide, realizar un algoritmo en C que permita:

- Agregar un nuevo producto (el código se debe generar en forma automática).
- Dado el código de producto mostrar su stock.
- Mostrar la información de todos los productos.

```
int main(void)
{ FILE *archi;
  int cod, opcion;
   If archi=fopen("producto.dat","a+"))==NULL)
      else  do
            { printf("\n 1. agregar un producto");
              printf("\n 2. mostrar stock de un producto");
              printf("\n 3. mostrar todos los productos");
              printf("\n 4. salir\n");
              scanf("%d", &opcion);
              switch(opcion)
                 {       case 1: agregar(archi); break;
                     case 2: { printf("\n Ingrese código de producto   a  mostrar stock ");
                               scanf("%d",&cod);
                               mostrarstock(archi,cod-100);      break; ;
                        case 3:  { mostrar(archi);     break;
                        case 4: break;
                  }
                        } while (opcion != 4);
        }
```

```
void agregar (FILE * xarchi)
{ prod p; fpos_t x;
  printf("\n ingrese el nombre del producto: ");
 fflush(stdin);
 gets(p.descrip);
 fseek(xarchi,0, SEEK_END);      // se posiciona  al final del archivo
 fgetpos(xarchi,&x);
 int cod=(int)(x/sizeof(prod))+100;
 printf("El codigo del nuevo producto es: %d",cod); p.cod=cod;
 printf("\n ingrese el stock : "); scanf("%d",&p.stock);
 printf("\n ingrese el precio unitario : ");
scanf("%f",&p.pu);
fwrite(&p,sizeof(p),1,xarchi);
}
```

```
void mostrarstock(FILE *xarchi, int pos)
{   prod p;
    rewind(xarchi);
    fseek(xarchi,  pos*sizeof(prod), SEEK_SET);
    fread(&p,sizeof(p),1,xarchi);
    printf("\n Nombre producto %s  stock %d",p.descrip, p.stock);
}
```

Modificación de una componente del archivo

Cuando se ingresa la ubicación de una componente

Actualizar el stock de un producto del archivo " producto.dat" cuyo código se ingresa por teclado.

```
void modistock(FILE *xarchi, int cod)
{
   prod p;
   fseek(xarchi,(cod-100)*sizeof(prod), SEEK_SET);
   fread(&p,sizeof(p),1,xarchi);
   printf("\n Nombre producto %s stock %d",p.descrip,p.stock);
   printf("\n Ingrese el nuevo stock"); scanf("%d",&p.stock);
   fseek(xarchi,  (cod-100)*sizeof(prod), SEEK_SET);  //Se ubica el puntero de nuevo
   fwrite(&p,sizeof(p),1,xarchi);                     //Se graba la componente en el archivo
}
```

Modificación de una componente del archivo

Cuando se ingresa un dato específico de una componente (no la ubicación).

```
void modistock_pr_nombre(FILE *xarchi,char n[30])
{
   prod p; fpos_t x;
   rewind(xarchi);
  fread(&p,sizeof(p),1,xarchi);
  while (!feof(xarchi) && strcmp(p.descrip,n))
        fread(&p,sizeof(p),1,xarchi);
  if(!feof(xarchi))
       {  printf("\n Nombre producto %s stock %d",p.descrip,p.stock);
          printf("\n Ingrese el nuevo stock");
         scanf("%d",&p.stock);
         fseek(xarchi,  - sizeof(p), SEEK_CUR);
         fwrite(&p,sizeof(p),1,xarchi);
         fclose(xarchi);
       }
}
```

Eliminar información del archivo

Para eliminar registros:

- Ubicarse en el registro que se desea eliminar o buscarlo
- Una vez que la componente está en memoria, se marca
- Grabar componente marcada
- Crear un archivo auxiliar con registros sin marcas
- Borrar el archivo original
- Renombrar el archivo auxiliar

rename

Cambia el nombre a un archivo físico de un dispositivo de almacenamiento.

Formato:

int rename(const char* viejo_nombre, const char* nuevo_nombre);

parámetros :

una cadena viejo_nombre y otra cadena nuevo_nombre.

valor de retorno

- cero si la función se ejecutó con éxito
- distinto de cero si hubo error.

remove

Elimina un archivo físico de un dispositivo de almacenamiento.

Formato:

int remove(const char *nombre_archivo);

parámetro : el nombre del archivo físico que se quiere eliminar.

.

valor de retorno cero si la función se ejecutó con éxito

distinto de cero si hubo error.

Se tiene un archivo **clientes.dat** con los siguientes datos: nombre del cliente y número de cuenta. Se pide eliminar del archivo los clientes cuyo número de cuenta se ingresa por teclado, el ingreso finaliza con número de cuenta igual a cero.

```
#include <stdio.h>
typedef struct
     { char nombre[30];
        int cuenta;
     }  cliente;
void main(void)
  {  FILE *archi, *auxi;
     if ((archi=fopen("clientes.dat","r+"))!=NULL)
        {   mostrar(archi);
            marcar(archi);
           auxi=fopen("auxiliar.dat","wb");
           compactar(archi,auxi);
            fclose (archi);
           fclose (auxi);
           remove("clientes.dat");
           rename("auxiliar.dat","clientes.dat");
        }
  }
```

```
void marcar(FILE *xarchi)
{ cliente c;   int nro,b;
   printf("\n Ingrese un número de cuenta a eliminar: ");
   scanf("%d",&nro);
  while(nro!=0)
   {    b=0;
        fseek(xarchi, 0, SEEK_SET);
        while((b==0)&& fread(&c,sizeof(c),1,xarchi))
              if(nro==c.cuenta)   b=1;
          // fin del while anidado
       if(b==1)
             {    fseek(xarchi,- sizeof(c),SEEK_CUR);
                  c.cuenta= -1;
                  fwrite(&c,sizeof(c),1,xarchi);
                  printf(" Cliente %s  fue marcado para borrarse",c.nombre);
              }
      else  printf("\n La cuenta no existe");
      printf("\n Ingrese un número de cuenta a eliminar: ");
       scanf("%d",&nro);
   }
}
```

```
void compactar(FILE *xarchi,FILE *xauxi)
{
      cliente c;
      fseek(xarchi, 0, SEEK_SET);
      fread(&c,sizeof(c),1,xarchi);
      while( !feof(xarchi))
                  {
            if (c.cuenta!=-1)
                fwrite(&c,sizeof(c),1,xauxi);
            fread(&c,sizeof(c),1,xarchi);
                  }
}
```

Actividad

¿Como puede determinar la longitud o cantidad de registros de un archivo?