Archivos

Principal material bibliográfico utilizado

- www.jorgesanchez.net
- Fundamentos de Programación C/C++ - Ernesto Peñaloza Romero.
- Lenguaje C – Adolfo Beltramo, Nélida Matas.

Archivos

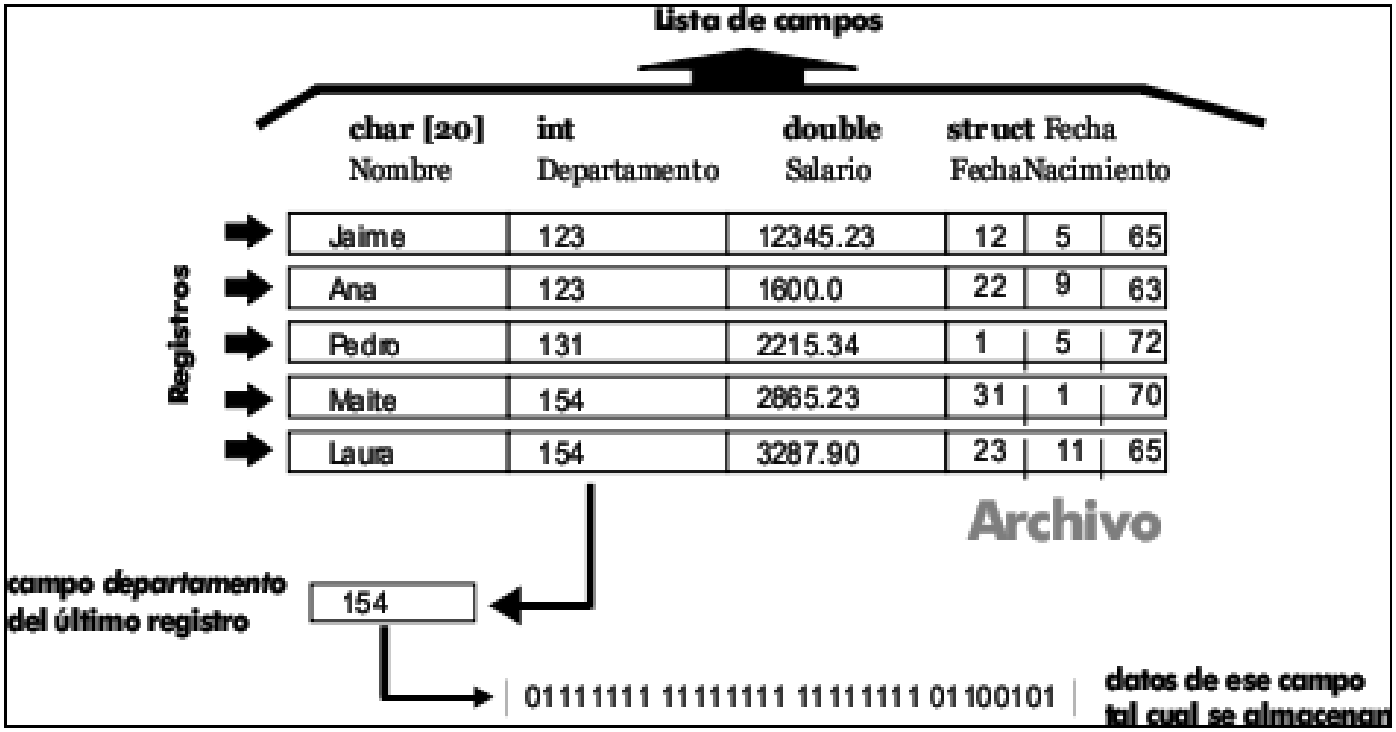
- Cuando trabajamos con datos en los programas, se trabajan en memoria volátil, por lo que luego se pierden.

- Por lo general, se desean utilizar datos que no se pierden, que se guardan o recuperan de un medio de almacenamiento permanente (ejemplo disco).

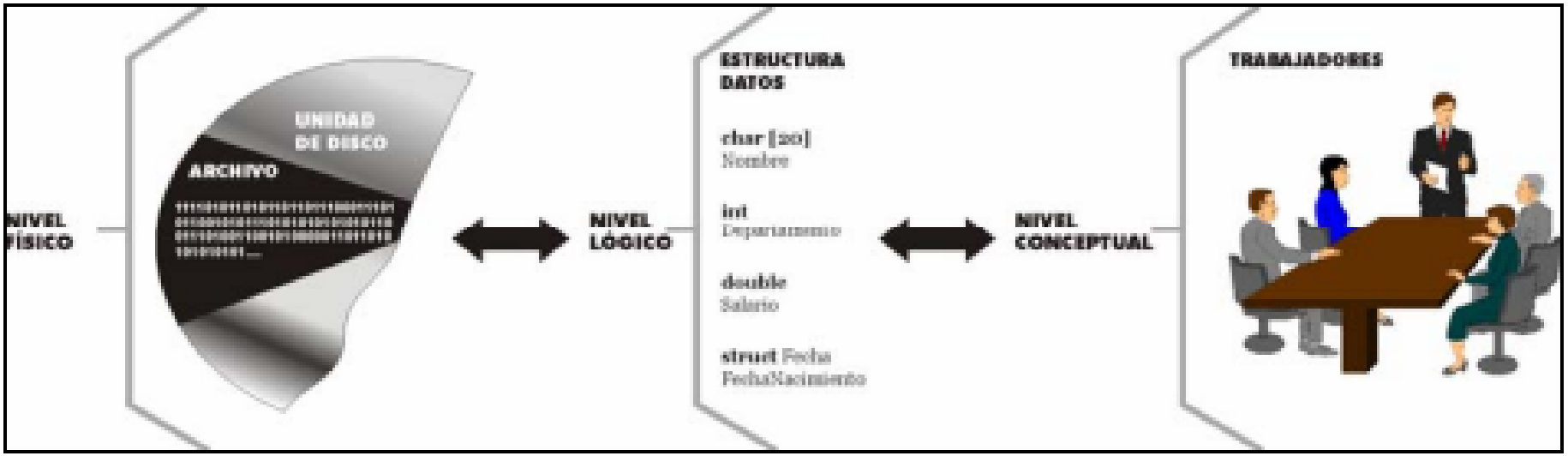
- Un archivo es una colección de datos relacionados entre si que se encuentran almacenados en un medio secundario, el cual puede retener la información aunque se retire la energía de alimentación del medio de almacenamiento.

- La comunicación con los archivos se realiza mediante corrientes de datos o secuencias, que proporcionan una interfaz independiente de los detalles de más bajo nivel, de los cuales se hará cargo el sistema operativo.

Archivos


Lista de campos
char [20]
Nombre
int
Departamento
double
Salario
struct Fecha
FechaNacimiento
Registros
Jaime 123 12345.23 12 5 65
Ana 123 1600.0 22 9 63
Pedro 131 2215.34 1 5 72
Maite 154 2865.23 31 1 70
Laura 154 3287.90 23 11 65
Archivo
campo departamento del último registro
154
01111111 11111111 11111111 01100101
datos de ese campo tal cual se almacenan

Archivos


NIVEL
FÍSICO
UNIDAD
DE DISCO
ARCHIVO
NIVEL
LÓGICO
ESTRUCTURA
DATOS
char [20]
Nombre
int
Departamento
double
Salario
struct Fecha
FechaNacimiento
NIVEL
CONCEPTUAL
TRABAJADORES

Archivos

- Los datos que conforman un archivo, a bajo nivel no dejan de ser 0 y 1, por lo que deben ser reinterpretados a enteros, char, etc.

- Los archivos pueden ser leídos en formato binario, o interpretados como estructuras, formándose registros y campos.

Archivos - Clasificación

POR EL TIPO DE CONTENIDO

- ARCHIVOS DE TEXTO: contienen información en forma de caracteres, al final de cada línea se coloca un identificador de fin de línea.

- ARCHIVOS BINARIOS: almacenan datos que no son interpretables a simple vista como texto.

POR LA FORMA DE ACCESO

- ARCHIVOS SECUENCIALES: el contenido se lee o se escribe de forma continua, para acceder a un dato debemos recorrer primero todos los que están antes de él. Por lo general los archivos de texto se accede de forma secuencial.

- ARCHIVOS DE ACCESO DIRECTO: se puede acceder directamente a cualquier parte del archivo conociendo su posición en el mismo. Dicha posición se suele indicar en bytes.

Archivos

- Para la programación, un archivo no es mas que una corriente, flujo (stream) de bits, bytes que pose un final (marca de fin de archivo).

- Para poder trabajar con un arhivo en C, se asocia este a un flujo, que es el elemento que permite leer / escribir datos de/en el archivo.

- En C un archivo puede ser cualquier cosa, por ejemplo una impresora. Asociamos el flujo al archivo cuando abrimos el archivo.

Archivos - Apertura

- ESTRUCTURA FILE: en el archivo stdio.h se define la estructura FILE.

  La secuencia de acceso a un archivo debe contener esta cabecera. Todo programa que desee trabajar con archivos debe contener un puntero *FILE para cada archivo que se desee leer o escribir.

- FUNCION fopen: para abrir los archivos, devuelve un puntero de tipo FILE.

  En caso que la apertura falle devuelve NULL.

```
FILE *fopen(const char *nombreArchivo, const char *modo)
```

- FUNCION fclose: cierra el archivo previamente abierto.

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

Archivos - Apertura

modo	significado
"r"	Abre un archivo para lectura de archivo de textos (el archivo tiene que existir)
"w"	Crea un archivo de escritura de archivo de textos. Si el archivo ya existe se borra el contenido que posee.
"a"	Abre un archivo para adición de datos de archivo de textos
"rb"	Abre un archivo para lectura de archivos binarios (el archivo tiene que existir)
"wb"	Crea un archivo para escritura de archivos binarios (si ya existe, se descarta el contenido)
"ab"	Abre un archivo para añadir datos en archivos binarios
"r+"	Abre un archivo de texto para lectura/escritura en archivos de texto. El archivo tiene que existir
"w+"	Crea un archivo de texto para lectura/escritura en archivos de texto. Si el archivo tenía datos, estos se descartan en la apertura.
"a+"	Crea o abre un archivo de texto para lectura/escritura. Los datos se escriben al final.
"r+b"	Abre un archivo binario para lectura/escritura en archivos de texto
"w+b"	Crea un archivo binario para lectura/escritura en archivos de texto. Si el archivo tiene datos, éstos se pierden.
"a+b"	Crea o abre un archivo binario para lectura/escritura. La escritura se hace al final de el archivo.

Archivos de texto

- FUNCION fgetc: sirve para leer caracteres de forma secuencial. Devuelve el siguiente carácter, ó EOF (fin de archivo) cuando alcanzó a este.

```
int fgetc(FILE *pArchivo);
```

- FUNCION fputc: escribe un carácter en el archivo.

```
int putc(int carácter, FILE *pArchivo);
```

- FUNCION feof:devuelve verdadero si se llego al final del archivo.

```
int feof(FILE *pArchivo)
```

- FUNCION fgets: lee una cadena de carateres. Lee la cadena de caracteres hasta que llegue a un salto de línea ó hasta que supere la longitud indicada.

```
char *fgets(char *texto, int longitud, FILE *pArchivo)
```

Archivos de texto

- FUNCION fputs: escribe una cadena en el archivo.

```
int fputs(const char texto, FILE *pArchivo)
```

Archivos de texto

```c
#include <stdio.h>
#include <conio.h>
int main(){
    FILE *archivo;
    char c=0;
    archivo=fopen("c:\\prueba.txt","r+");
    if(archivo!=NULL) { /* Apertura correcta */
        while(c!=EOF){ /* Se lee hasta llegar al final */
            c=fgetc(archivo);
            putchar(c);
        }
       fclose(archivo);
    }
    else  printf("Error");
}
```

Archivos de texto

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

int main(){
    FILE *archivo;
    char c=0;
    archivo=fopen("c:\\prueba.txt","r+");
    if(archivo!=NULL) {
        while(!feof(archivo)){
            c=fgetc(archivo);
            putchar(c);
        }
    }
    else{
        printf("Error");
    }
    fclose(archivo);
    return 0;
}
```

Archivos de texto

```c
#include <stdio.h>
#include <conio.h>

int main(){
    FILE *archivo;
    char texto[2000];
    archivo=fopen("c:\\prueba2.txt","r");
    if(archivo!=NULL) {
        fgets(texto,2000,archivo);
        while(!feof(archivo)){
           puts(texto);
           fgets(texto,2000,archivo);
        }
        fclose(archivo);
    }
    else{
        printf("Error en la apertura");
    }
    return 0;
}
```

Archivos de texto

- FUNCION fprintf: idem printf pero para escritura en archivos de texto.

- FUNCION fscanf: idem scanf pero para lectura de datos de archivos de texto.

Archivos de texto

```c
#include <stdio.h>
int main(){
    int n=1; /*Número del empleado*/
    char nombre[80];
    double salario;
    FILE *pArchivo;

    pArchivo=fopen("c:\\prueba3.txt","w");
    if(pArchivo!=NULL){
        do{
            printf("Introduzca el número de empleado: ");
            scanf("%d",&n);
       /*Solo seguimos si n es positivo, en otro caso
       entenderemos que la lista ha terminado */
            if(n>0){
              printf("Introduzca el nombre del empleado: ");
               scanf("%s",nombre);
              printf("Introduzca el salario del empleado: ");
               scanf("%lf",&salario);
               fprintf(pArchivo,"%d\t%s\t%lf\n",
                                  n,nombre,salario);
            }
        }while(n>0);
        fclose(pArchivo);
    }
}
```

Archivos de texto

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

int main(){
    int n=1;
    char nombre[80];
    double salario;
    FILE *pArchivo;
    pArchivo=fopen("c:\\prueba3.dat","r");
    if(pArchivo!=NULL){
        while(!feof(pArchivo)){

fscanf(pArchivo,"%d\t%s\t%lf\n",&n,nombre,&salario);
          printf("%d\t%s\t%lf\n",n,nombre,salario);
        }
        fclose(pArchivo);
        return 0;
    }
}
```

Archivos de texto

- FUNCION fflush: vacía el buffer del archivo indicado, si no se indica archivo, vacía todos los buffers de archivos abiertos.

- FUNCION rewind: vuelve el indicador de posición al inicio del archivo, lo que se escriba o lea luego de su ejecución será lo que este al principio del archivo.

```
void rewind(FILE *pArchivo);
```

Archivos binarios

- FUNCION fwrite: permite escribir en un archivo binario

```
size_t fwrite(void *búfer, size_t bytes, size_t cuenta,FILE *p)
```

size_t bytes: representa tamaño de datos en bytes. Suele ser usado con sizeof().

bufer: puntero a la dirección de memoria que contiene el dato que se desea escribir.

size_t cuenta: cuantos elementos de size_t bytes se escribirán.

p: puntero al archivo que se desea escribir.

La función devuelve el nro. de elementos escritos que debería coincidir con el parámetro cuenta.

Archivos binarios

```c
#include <conio.h>
#include <stdio.h>

typedef struct {
  char nombre[25];
  int edad;
}Persona;

int main(){
  Persona per[25];
  int i=0;
  FILE *pArchivo;

  pArchivo=fopen("C:\\datos.dat","wb");
  if(pArchivo!=NULL){
    do{
      fflush(stdin); /* Se vacía el búfer de teclado */
      printf("Introduzca el nombre de la persona: ");
      gets(per[i].nombre);
      if(strlen(per[i].nombre)>0){
        printf("Introduzca la edad");
        scanf("%d",&(per[i].edad));
        fwrite(&per[i],sizeof(Persona),1,pArchivo);
        i++;
      }
    }while(strlen(per[i].nombre)>0 && i<=24);
    fclose(pArchivo);
  }
  else{
    printf("Error en la apertura del archivo");
  }
}
```

Archivos binarios

- FUNCION fread: permite leer de un archivo binario

```
size_t fread(void *búfer, size_t bytes, size_t
                                        cuenta,FILE *p)
```

La lectura es secuencial y para controlar hasta donde leeremos utilizamos feof().

size_t bytes: representa tamaño de datos en bytes. Suele ser usado con sizeof().

bufer: puntero a la posición del archivo que contiene el dato que se desea leer.

size_t cuenta: cuantos elementos de size_t bytes se leerán.

p: puntero al archivo que se desea leer

La función devuelve el nro. de elementos leídos que debería coincidir con el parámetro cuenta.

Archivos binarios

```
#include <conio.h>
#include <stdio.h>
typedef struct {
  char nombre[25];
  int edad;
}Persona;
int main(){
  Persona aux;
  FILE *pArchivo;
  pArchivo=fopen("C:\\datos.dat","rb");
  if(pArchivo!=NULL){
    /* Se usa lectura adelantada, de otro modo
    el último dato sale repetido */
    fread(&aux,sizeof(Persona),1,pArchivo);
    while(!feof(pArchivo)){
      printf("Nombre: %s, Edad: %d\n",aux.nombre,
aux.edad);
      fread(&aux,sizeof(Persona),1,pArchivo);
    }
    fclose(pArchivo);
  }
  else{
    printf("Error en la apertura del archivo");
  }}
```

Archivos – acceso directo

Hasta ahora hemos visto archivos que los manejamos de forma secuencial, es decir para llegar a una posición determinada debemos pasar por todas las anteriores..

En el acceso de forma directa, debemos tener forma de llegar hasta una posición sin pasar por las anteriores. Para poder hacer esto es necesario mover el indicador de posición al lugar que queremos ir a leer ó escribir.

- FUNCION fseek: mueve el indicador de posición del archivo.

```
int fseek(FILE * pArchivo, long bytes, int origen)
```

La función coloca el indicador a la posición determinado por origen y lo mueve la cantidad de bytes que se indiquen.

Existen ciertas constantes predefinidas y útiles para origen:

SEEK_SET : principio del archivo.

SEEK_CUR: posición actual.

SEEK_END: final del archivo.

La función devuelve cero si se pudo recolocar el indicador correctamente.

Archivos – acceso directo

```
typedef struct {
  char nombre[25];
  int edad;
}Persona;

int main(){
  Persona aux;
  FILE *pArchivo;

  pArchivo=fopen("C:\\datos.dat","rb");
  if(pArchivo!=NULL){
    fseek(pArchivo,3*sizeof(Persona),SEEK_SET);
    fread(&aux,sizeof(Persona),1,pArchivo);
    printf("%s, %d años",aux.nombre,aux.edad);
    fclose(pArchivo);
  }
}
```

Archivos – acceso directo

- FUNCION ftell: obtiene el valor actual del indicador de posición del archivo.
- En un archivo binario es el numero de byte actual en el cual esta situado el indicador del archivo.

```
long ftell(FILE *pArchivo)
```

Si ftell falla da como resultado -1.

Archivos – acceso directo

```
/* Estructura de los registros almacenados en
el archivo*/
typedef struct{
    int n;
    int nombre[80];
    double saldo;
}Movimiento;
```

```
int main(){
    FILE *f=fopen("movimientos3.bin","rb");
    int nReg;/*Guarda el número de registros*/

    if(f!=NULL){
        fseek(f,0,SEEK_END);
        nReg=ftell(f)/sizeof(Movimiento);
        printf("N° de registros en el archivo =
%d",nReg);
        getch();
    }
    else{
        printf("Error en la apertura del archivo");
    }
}
```

Programa que averigua la cantidad de registros del archivo

Archivos binarios – acceso directo

- FUNCION fgetpos: almacena en el puntero posición, la posición actual del indicador del archivo
- FUNCION fsetpos: coloca el indicador del archivo en la posición almacenada por fgetpos.
- Si resultan bien, devuelven 0.

```
int fgetpos(FILE *pArchivo, fpos_t *posicion);
int fsetpos(FILE *pArchivo, fpos_t *posicion);
```

```
FILE *f=fopen("prueba.bin","rb+");
fpos_t posicion;
if(f!=NULL){
   ..... /*operaciones de lectura o de manipulación*/
   fgetpos(f,&posicion); /*Captura de la posición
                                          actual*/
   ... /*operaciones de lectura o manipulación */
   fsetpos(f,&posicion); /*El cursor vuelve a la posición
                              capturada */
```