Estructuras de Almacenamiento físico

Almacenamiento físico y archivos

Definición de almacenamiento físico y su relación con el almacenamiento en memoria

Hasta ahora venimos trabajando con almacenamiento en memoria, es decir, es un almacenamiento al cual se accede más rápidamente pero que no se mantiene de manera permanente.

En cambio, el almacenamiento físico tiene características distintas. Este tipo de almacenamiento guarda la información en archivos que se guardan en el disco. Esto hace que la información se guarde de manera permanente pero es más lenta que la memoria.

Tipos de archivos

Diferencia entre archivos de texto y archivos binarios

Ahora pasemos a la diferencia entre archivos de texto y archivos binarios. Los archivos de texto almacenan información en forma de caracteres legibles, utilizando un conjunto de caracteres específico, como ASCII o Unicode. Por otro lado, los archivos binarios almacenan datos en su forma binaria, es decir, secuencias de bits. Esto nos permite almacenar cualquier tipo de datos, como números, imágenes, videos o estructuras complejas. En nuestro caso lo vamos a usar especiamente guardando registros.

	Archivos de texto	Archivos binarios
Ventajas	Legibles por humanos	Almacenan datos estructurados y complejos
	Fáciles de editar con editores de texto	Manipulación eficiente de datos
	Ocupan menos espacio	Procesamiento más rápido
Desventajas	Limitaciones en la representación de datos	No legibles por humanos
	Menos eficientes en tiempo de lectura/escritura	Requieren programas específicos para interpretación y manipulación

¿Cómo crear o abrir un archivo?

La función `fopen` se utiliza para abrir un archivo en C, y los parámetros que se le pasan determinan el modo de apertura del archivo. El primer parámetro es el nombre del archivo. El segundo parámetro de `fopen` es una cadena de caracteres que indica el modo de apertura.

Ejemplo

```
fopen("archivo.dat", "rb")
```

En el caso de `"rb"`, "r" indica que el archivo se abrirá en modo de solo lectura, y "b" indica que el archivo se abrirá en modo binario. Aquí hay una explicación más detallada de los modos de apertura:

- `"r"`: Abre el archivo en modo de lectura. El archivo debe existir previamente. El puntero de posición se coloca al principio del archivo.
- `"w"`: Abre el archivo en modo de escritura. Si el archivo no existe, se crea. Si el archivo ya existe, su contenido se elimina. El puntero de posición se coloca al principio del archivo.

- `"a"`: Abre el archivo en modo de adjuntar. Si el archivo no existe, se crea. Si el archivo ya existe, los nuevos datos se añaden al final. El puntero de posición se coloca al final del archivo.
- `"b"`: Abre el archivo en modo binario. Esto significa que el archivo se leerá o escribirá en su forma binaria original, sin realizar conversiones. Es importante utilizar este modo para archivos binarios, como imágenes o archivos estructurados.
- `"t"`: Abre el archivo en modo de texto. Este es el modo predeterminado si no se especifica "b". Los caracteres de nueva línea se pueden convertir automáticamente según la plataforma.

También está el símbolo `+` que permite tanto la lectura como la escritura.

Puedes combinar diferentes modos según tus necesidades. Por ejemplo, `"rb"` se utiliza para abrir un archivo existente en modo de lectura binaria. Si deseas abrir un archivo en modo de escritura binaria, puedes utilizar `"wb"`.

Aquí hay una descripción de los modos de apertura más comunes utilizados en la función `fopen`:

- `"rb"`: Apertura en modo de lectura binaria.
- `"wb"`: Apertura en modo de escritura binaria.
- `"ab"`: Apertura en modo de agregado binario.
- `"r+"`: Apertura en modo de lectura/escritura. El archivo debe existir.
- `"w+"`: Apertura en modo de lectura/escritura. Si el archivo existe, su contenido se trunca. Si no existe, se crea uno nuevo.
- `"a+"`: Apertura en modo de lectura/escritura. Los datos se escriben al final del archivo sin truncar su contenido existente. Si el archivo no existe, se crea uno nuevo.

¿Cómo escribir un archivo?

Vamos a usar la función:

```
fwrite(lo_que_vamos_a_guardar, tamaño, cantidad, variable_archivo);
```

Ejemplo:

```
fwrite(estudiantes, sizeof(Estudiante), 3, archivo);
```

```
#include <iostream>
#include <cstring>

using namespace std;

struct Estudiante {
    int codigo;
    char nombre[50];
    int edad;
};

Estudiante dameEstudiante(int codigo) {
    Estudiante estudiante;

    estudiante.codigo = codigo;

    cout << "Ingrese el nombre del estudiante: ";
    cin.getline(estudiante.nombre, sizeof(estudiante.nombre));

    cout << "Ingrese la edad del estudiante: ";
    cin >> estudiante.edad;
    cin.ignore(); // Ignorar el salto de línea pendiente en el búfer


    return estudiante;
}


int main() {
    Estudiante estudiantes[3];

    for (int i = 0; i < 3; i++) {
        cout << "Ingrese los datos del estudiante " << i + 1 << ":" << endl;
        estudiantes[i] = dameEstudiante(i + 1);
    }

    FILE* archivo = fopen("estudiantes_binarios.dat", "wb");
    if (archivo != NULL) {
        fwrite(estudiantes, sizeof(Estudiante), 3, archivo);
        fclose(archivo);
        cout << "Archivo creado exitosamente." << endl;
    } else {
        cout << "No se pudo crear el archivo." << endl;
```

```
    }

    return 0;
}
```

¿Cómo leer un archivo?

Vamos a usar la función:

```
fread(&variable_donde_guardamos, tamaño, cantidad, variable_archivo);
```

Ejemplo

```
fread(&estudiante, sizeof(Estudiante), 1, archivo)
```

```
#include <iostream>
#include <cstring>

using namespace std;

struct Estudiante {
    int codigo;
    char nombre[50];
    int edad;
};

int mostrarEstudiantes() {
    FILE* archivo = fopen("estudiantes_binarios.dat", "rb");
    if (archivo != NULL) {
        Estudiante estudiante;
        while (fread(&estudiante, sizeof(Estudiante), 1, archivo) == 1) {
            cout << "Estudiante encontrado:" << endl;
            cout << "Código: " << estudiante.codigo << endl;
            cout << "Nombre: " << estudiante.nombre << endl;
            cout << "Edad: " << estudiante.edad << endl;
        }
        fclose(archivo);
    } else {
        cout << "No se pudo abrir el archivo para lectura." << endl;
    }
    return 0;
}

int main() {
```

```
    mostrarEstudiantes();
    return 0;
}
```

¿Cómo buscar en un archivo?

Secuencial:

```
#include <iostream>
#include <cstring>

using namespace std;

struct Estudiante {
    int codigo;
    char nombre[50];
    int edad;
};

int buscarEstudiante(const char* nombreBuscado) {
    FILE* archivo = fopen("estudiantes_binarios.dat", "rb");
    if (archivo != NULL) {
        Estudiante estudiante;
        while (fread(&estudiante, sizeof(Estudiante), 1, archivo) == 1) {
            if (strcmp(estudiante.nombre, nombreBuscado) == 0) {
                cout << "Estudiante encontrado:" << endl;
                cout << "Código: " << estudiante.codigo << endl;
                cout << "Nombre: " << estudiante.nombre << endl;
                cout << "Edad: " << estudiante.edad << endl;
                fclose(archivo);
                return 1;
            }
        }
        fclose(archivo);
    } else {
        cout << "No se pudo abrir el archivo para lectura." << endl;
    }
    return 0;
}

int main() {
    char nombre[50];
    cout << "Ingrese el nombre del estudiante a buscar: ";
    cin >> nombre;

    if (!buscarEstudiante(nombre)) {
        cout << "Estudiante no encontrado." << endl;
    }

    return 0;
}
```

Directa

Vamos a usar la función:

```
fseek(variable_archivo, offset, origen);
```

- `variale_archivo`: Puntero al archivo en el que se realizará el desplazamiento del puntero.
- `offset`: Número de bytes a desplazarse, relativo a la posición indicada por `origin`.
- `origen`: Valor que especifica desde dónde se contará el desplazamiento. Puede ser una de las siguientes constantes:
  - `SEEK_SET` (desde el principio del archivo)
  - `SEEK_CUR` (desde la posición actual del puntero)
  - `SEEK_END` (desde el final del archivo).

Ejemplo:

```
fseek(archivo, (codigoBuscado - 1) * sizeof(Estudiante), SEEK_SET);
```

```
#include <iostream>
#include <cstring>

using namespace std;

struct Estudiante {
    int codigo;
    char nombre[50];
    int edad;
};

int buscarEstudiante(int codigoBuscado) {
    FILE* archivo = fopen("estudiantes_binarios.dat", "rb");
    if (archivo != NULL) {
        Estudiante estudiante;
        fseek(archivo, (codigoBuscado - 1) * sizeof(Estudiante), SEEK_SET);
        if (fread(&estudiante, sizeof(Estudiante), 1, archivo) == 1) {
```

```
            cout << "Estudiante encontrado:" << endl;
            cout << "Código: " << estudiante.codigo << endl;
            cout << "Nombre: " << estudiante.nombre << endl;
            cout << "Edad: " << estudiante.edad << endl;
            fclose(archivo);
            return 1;
        }
        fclose(archivo);
    } else {
        cout << "No se pudo abrir el archivo para lectura." << endl;
    }
    return 0;
}

int main() {
    int codigo;
    cout << "Ingrese el código del estudiante a buscar: ";
    cin >> codigo;

    if (!buscarEstudiante(codigo)) {
        cout << "Estudiante no encontrado." << endl;
    }

    return 0;
}
```

¿Cómo editar datos de un archivo?

```
#include <iostream>
#include <string>
#include <cstring>

using namespace std;

struct Estudiante {
    int codigo;
    char nombre[50];
    int edad;
};

void editarEdadEstudiante(int codigoBuscado, int nuevaEdad) {
    FILE* archivo = fopen("estudiantes_binarios.dat", "rb+");
    if (archivo != NULL) {
        Estudiante estudiante;
        bool encontrado = false;
        while (!encontrado && fread(&estudiante, sizeof(Estudiante), 1, archivo) == 1) {
            if (estudiante.codigo == codigoBuscado) {
                encontrado = true;
                estudiante.edad = nuevaEdad;
                fseek(archivo, -sizeof(Estudiante), SEEK_CUR);
                fwrite(&estudiante, sizeof(Estudiante), 1, archivo);
```

```
                cout << "Edad actualizada exitosamente." << endl;
            }
        }
        if (!encontrado) {
            cout << "No se encontró un estudiante con ese código." << endl;
        }
        fclose(archivo);
    } else {
        cout << "No se pudo abrir el archivo para lectura y escritura." << endl;
    }
}

int main() {
    int codigoBuscado, nuevaEdad;
    cout << "Ingrese el código del estudiante a buscar: ";
    cin >> codigoBuscado;
    cout << "Ingrese la nueva edad: ";
    cin >> nuevaEdad;

    editarEdadEstudiante(codigoBuscado, nuevaEdad);

    return 0;
}
```