Estructuras de datos: Archivos

Archivos secuenciales
Archivos aleatorios

1. Introducción.

Los programas de computadora procesan datos directamente en la memoria RAM. Sin embargo, al término de la aplicación los datos son eliminados por el sistema operativo, a menos que se guarden o almacenen en memoria auxiliar permanente (Discos, cintas, etc.). Mas tarde cuando la aplicación vuelve a ejecutarse, puede llamar a esta colección de datos y la reutilizará iterativamente para su gestión. Un **Fichero** o dispositivo de disco es una estructura de datos, pues almacena y organiza una colección de datos en memoria secundaria o permanente.

Acceso a dispositivo de disco:

Es la técnica de Organización y gestión de datos estructurados sobre dispositivos de almacenamiento secundario

Fichero:

Estructuras o Colecciones de datos almacenados en dispositivos de memoria auxiliar y permanente. Los datos son Escritos / Leidos desde ficheros y alojados en la memoria RAM solo para su procesamiento (Administrar información).

2. Características del proceso de Datos en Memoria Principal (RAM) y Secundaria.

Limitaciones.

El uso y presencia de los datos en la memoria esta restringido al tiempo que la computadora esta encendida y el programa en ejecución (acceso inmediato, aleatorio y temporal). Esto implica que los datos desaparecen de la memoria **RAM** (Random Access Memory), cuando la computadora se apaga o deja de ejecutarse la aplicación.

Soluciones.

Esta gran limitación es cubierta por los dispositivos de almacenamiento secundario permanente (Memoria externa o secundaria: Discos HDD, CD, Flash, etc), desde donde las colecciones de datos son recuperados para procesos ulteriores.

3. Definición de Fichero o archivo.

Las estructuras de datos, aplicadas al almacenamiento secundario de colecciones de datos se denominan *"Organización de Archivos".* Un fichero por tanto es. "Una estructura de datos, cuyos elementos básicos, los campos que pertenecen a un tipo de datos, se organizan linealmente en registros de un mismo tamaño, y estos a su vez en tablas de registros de longitud variable."

Jerarquía de la Organización de Archivos

- **Datos**
- **Campos**.
- **Registros**.
- **Ficheros**
- **Base de Datos**

4. Organización de los Archivos.

5. Procesamiento de archivos secuenciales.

En un archivo secuencial los datos se insertan en el dispositivo de almacenamiento en orden cronológico de llegada. (Inmediatamente a continuación del anterior).

Los archivos secuenciales terminan con una marca de "Final De Archivo: FDA" (End Of File: EOF en ingles). Cuando se agregan nuevos datos se añaden (append) en las marcas EOF.

Por contraparte, un Archivo secuencial también posee una marca de "Inicio De Archivo: IDA" (Begin Of File: BOF en ingles).

area BOF
1. registro
2. registro
…
n. registro
area EOF

Fig. 1. Estructura de un archivo secuencial (s).

6. Gestión de Archivos.

El acceso a dispositivo de disco, es un proceso que tiene los siguientes pasos:

Algoritmo. Acceso a dispositivo de disco

1. Definir el dispositivo y obtener un *identificador* de archivo o canal de comunicación.
2. Usar el *identificador* para: Escribir en el dispositivo o Leer desde el dispositivo.
3. Cerrar el canal de comunicación o *identificador* (El Sistema Operativo permite disponer entre 0 a 255 identificadores o canales simultáneos)

Paso 1. Para definir un dispositivo se usa la siguiente sintaxis y las siguientes funciones:

Sintaxis:

```
tipo
      archivo_<organizacion> de <tipo_dato>: <identificador_f>

var
      <identificador-f>:  var_tipoArchivo
```

Funciones

```
crear(var_tipoArchivo, nombreFisico)
abrir(var_tipoArchivo, <modo_acceso>, nombreFisico)
```

En donde:

tipoArchivo:	variable de de tipo archivo(FILE), declarado a partir del tipo identificador.
nombreFisico:	Nombre del dispositivo de disco de tipo cadena(Dependiendo del sistema operativo UNIX-Linux-DOS. Por ejemplo en DOS: [Unidad][Path][nombre[.ext]]).
<modo_acceso>:	Operaciones de acceso para Escritura(write) o para Lectura (read). El modo es unidireccional y único. El modo escritura por omisión sobre escribir, pero también agrega datos(append).

Restricciones en el acceso a dispositivos.

Al ejecutar la instrucción **abrir**() se pueden encontrar los siguientes errores: (retorna NULL)

- Archivo no encontrado en el dispositivo especificado (Nombre de archivo o identificador de dispositivo erróneo)
- Archivo ya esta en uso para alguna otra aplicación.
- Errores de hardware.

Ejemplos comentados:

a) Ejemplo de acceso a dispositivo.

Algoritmo. Definir acceso a archivo secuencial.

```
tipo
      archivo_s de <tipo_dato>: <identificador_f>
var
      <identificador_f>: var_archivo

inicio
      crear(var_archivo, 'nombreArchivo')
      abrir(var_archivo, modo, 'nombreArchivo') //e- modo escritura o
                                                //a- modo añadir, y
                                                //l- modo lectura
      si (var_archivo = NULL) entonces
            escribir('Error: no se puede abrir archivo')
      fin_si
fin
```

Paso 2. Usar el identificador para Escribir / Leer en el dispositivo se usan una de las siguientes funciones:

Sintaxis:

```
var
      <tipo_dato>: variableDatos
```

Operación escritura.

```
escribirArchivo(var_tipoArchivo, variableDatos)
```

Operación de lectura.

```
variableDatos ← leerArchivo(var_tipoArchivo)
```

b) Ejemplo de acceso a dispositivo en modo Escritura

Algoritmo. Definir archivo secuencial para escritura

```
tipo
      archivo_s de <tipo_dato>: <identificador_f>
var
      <identificador_f>: var_archivo
      <tipo_dato>: variableDatos

inicio
      crear(var_archivo, 'datos1')
      abrir(var_archivo, e, 'datos1')     //e- modo escritura, o
                                          //a- modo añadir
      si (var_archivo = NULL) entonces
            escribir('Error: no se puede abrir archivo')
      fin_si
      //asignar valor a variableDatos
      variableDatos ← valor
      //y almacenar en disco
      escribirArchivo(var_archivo, variableDatos)
fin
```

c) Ejemplo de acceso a dispositivo en modo Lectura

Algoritmo. Definir archivo secuencial para lectura

```
tipo
      archivo_s de <tipo_dato>: <identificador_f>
var
      <identificador_f>: var_archivo
      <tipo_dato>: variableDatos

inicio

      abrir(var_archivo, l, "datos1")     //l- modo lectura
      si (var_archivo = NULL) entonces
            escribir("Error: no se puede abrir archivo")
      si_no
            variableDatos ← leerArchivo(var_archivo)
            escrbir("Dato leído: ", variableDatos)
      fin_si
fin
```

Paso 3. Para cerrar el canal de comunicación o identificador de archivo se usa la siguiente función:

cerrar(var_tipoArchivo)

d) Ejemplo de acceso a dispositivo en modo Agregar y cerrar

Algoritmo. Definir archivo secuencial en modo añadir y cerrar identificador

```
tipo
      archivo_s de <tipo_dato>: <identificador_f>
var
      <identificador_f>: varArchivo
      <tipo_dato>: variableDatos

inicio
      crear(varArchivo, "datos2")
      abrir(varArchivo, a, "datos2")      //a- modo añadir

      //validar definición de archivo
      si (varArchivo = NULL) entonces
            escribir("Error: no se puede abrir archivo")
      si_no
            //asignar valor a variableDatos
            variableDatos ← valor
            escribirArchivo(varArchivo, variableDatos)
      fin_si
      //cerrar canal de comunicacion
      cerrar(varArchivo)
fin
```

RESUMEN

Operaciones Básicas.

Las operaciones básicas e importantes, que se permiten en un archivo secuencial son:

- Crear un dispositivo: Create
- Definir acceso a dispositivo: Open
- Escribir su contenido: Write
- Añadir al final de su contenido: Append,
- Consultar su contenido: Read
- Cerrar el dispositivo: Close, y
- Borrar o eliminar el dispositivo. Delete

Entre muchas otras.

Problemas resueltos

Problema 1. Algoritmo de acceso secuencial: Crear archivo para almacenar caracteres.

Análisis:
El dispositivo a escribir corresponde a una estructura de tipo carácter
Especificaciones de E/S
Entrada: caracteres
Salida: dispositivo de disco de tipo caracter

Pseudocodigo

```
Algoritmo Escribir caracteres en disco.
tipo
  archivo_s de caracter: identificador_fc
var
      identifiacodr_fc: fc    // canal de comunicacion
      caracter:   letra

inicio
      letra ← 'a' //inicializar var

      //paso1: definir dispositivo
      crear(fc, "datos1.txt")
      abrir(fc, e, "datos1.txt")

      //validar definición de archivo
      si (fc = NULL) entonces
            escribir("Error: no se puede abrir archivo")
      si_no
            //paso2: escribir caracter
            escribirArchivo(fc, letra)
      fin_si
      //paso 3: cerrar canal de comunicacion
      cerrar(fc)
      //explore el sistema de archivos de la PC y verifique el archivo
fin
```

Problema 2. Algoritmo de acceso secuencial: Leer archivo que almacena caracteres.

Análisis:
El dispositivo a leer corresponde a una estructura de tipo carácter
Especificaciones de E/S
Entrada: dispositivo de disco de tipo caracter
Salida: caracteres

Pseudocodigo

```
Algoritmo leer caracteres desde disco.
tipo
  archivo_s de caracter: identificador_fc
var
      identifiacodr_fc: fc    // canal de comunicacion
      caracter:   letra
```

```
inicio

      //paso1: definir dispositivo
      abrir(fc, 1, 'datos1.txt')    //1-  modo lectura
      //validar identificador de archivo
      si (fc = NULL) entonces
            escribir('Error: no se puede abrir archivo')
      si_no
            //paso2: leer caracter
            letra ← leerArchivo(fc)
            escribir(letra)
      fin_si

      //paso 3: cerrar canal de comunicacion
      cerrar(fc)
      //explore el sistema de archivos de la PC y verifique.
fin
```

Comprobación de algoritmos.

2. Codigo C++: Modo Source File

```
/*Name: File_EscribirChar.cpp
  Author: J.Medianero.A
  Date: 25/09/08 09:42
  Description: Define archivo en modo escritura
               de tipo caracter
*/

#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char* argv[]){
    FILE *pNumFile;
    char letra = 'a';

    pNumFile = fopen("Letras.txt", "w");    //crea un fichero de escritura
                                            //si ya existe, lo crea de nuevo
    //pNumFile = fopen("Letras.txt", "a"); //abre o crea un fichero para
                                            //añadir datos al final del mismo
    if( pNumFile == NULL ) {
        printf("Error de acceso");
        system("PAUSE");
        exit(0); //abandonar aplicacion
    }else{
        fputc(letra, pNumFile);
        fclose(pNumFile);
        //system("NotePad Letras.txt");
    }

    system("PAUSE");
    return EXIT_SUCCESS;
}//fin main
```

```
/*Name: File_LeerChar_v1.cpp
  Author: J.Medianero.A
  Date: 25/09/08 09:42
  Description: Define archivo en modo lectura
               de tipo caracter
*/
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char* argv[]){
    FILE *pNumFile;
    char letra;

    pNumFile = fopen("Letras.txt", "r");    //abre un fichero en modo lectura
                                            //si no exite retorna NULL
    if( pNumFile == NULL ) {
        printf("Error de acceso...");
        cin.get();
        exit(0); //abandonar aplicacion
    }else{
        letra = fgetc(pNumFile);
        printf("%c", letra);
    }
    fclose(pNumFile);
    //system("NotePad Letras.txt");
    system("PAUSE");
    return EXIT_SUCCESS;
}//fin main



/*Name: File_LeerChar_v2.cpp
  Author: J.Medianero.A
  Date: 02/01/10 09:42
  Description: Define archivo en modo lectura de tipo carácter.
  NOTA: Si la escritura de archivo esta definido en modo "a" (append), los
  datos se acumulan en el archivo. Para leer todos los datos guardados, debe
  modificar la rutina de lectura, implementando un bucle controlado por
  agotamiento (hasta encontrar el final del archivo -EOF: End Of File-)
*/
//define prototipo: leerArchivoDisco
procedimiento leerArchivoDisco(cadena: nombreArchivo)
var
      identificador_fc: fc
      caracter:   letra
inicio
      abrir(fc, l, nombreArchivo)
      si(fc = null)entonces
            escribir("Error de acceso")
      si_no
            repetir
                  letra ← leerArchivo(fc)
                  escribir(letra)
            hasta_que( EOF )
      fin_si
      cerrar(fc)  //cerrar canal de comunicacion
fin
```