Archivos/ Ficheros C++

Archivos en C++

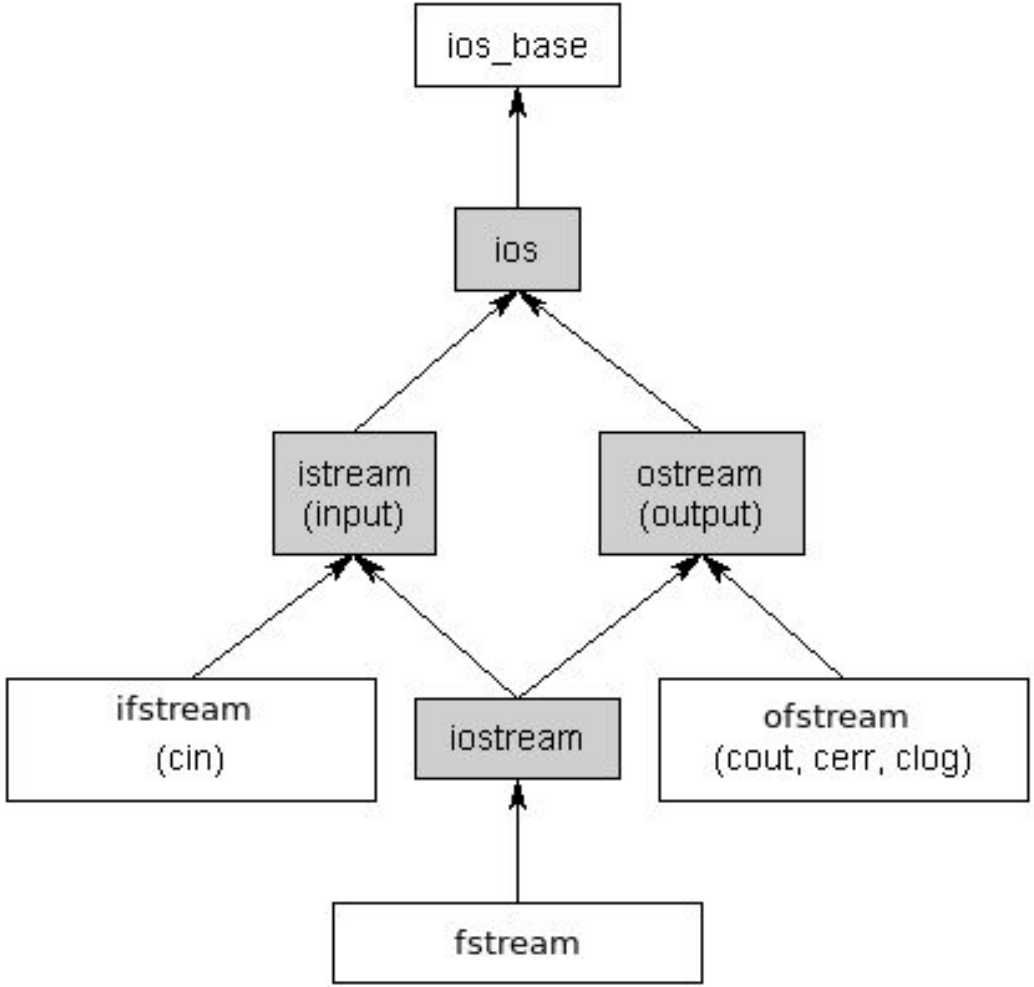
Los archivos o ficheros son la forma en la que C++ permite el acceso al disco.

Todos los procesos tienen abiertos, por defecto, los archivos 0(entrada), 1(salida) y 2(salida de errores), de manera que en C++ se corresponden con los objetos cin, cout y cerr.

De estos últimos, el primero pertenece a la clase ifstream, que a su vez desciende de istream (flujo de entrada). Los dos últimos pertenecen a la clase ofstream, que desciende de la clase ostream (flujo de salida)

Archivos en C++

Una jerarquía aproximada puede verse a continuación.



Archivos en C++

Lo bueno es que es posible manejar los flujos de la misma manera que se manejan cout y cin, es decir, con los operadores de inserción y de extracción.

Archivo del sistema operativo	Clase	Objeto
0	ifstream	cin
1	ofstream	cout
2	ofstream	cerr

Archivos de texto

Los archivos de texto son los más sencillos de manejar, pues, como ya se ha mencionado, para trabajar con ellos se emplean los operadores de inserción y extracción que ya se conocen de la consola.

Archivos de entrada

El típico recorrido de un archivo de texto desconocido (al igual que uno binario), se hace con lectura adelantada. Así, si en el directorio donde se ejecuta la aplicación hay un archivo llamado entrada.txt, se puede volcar por pantalla de la siguiente forma, utilizando la clase ifstream:

Archivos de entrada

```
int main(){
 string s;
 ifstream f( "salida.txt" );
 if ( f.is_open() ) {
 getline( f, s );
 while( !f.eof() ) {
 cout << s << endl;
 getline( f, s ); }}
 else cerr << "Error de apertura del archivo." << endl; }
```

Archivos de salida

Un archivo de salida de texto se maneja con la clase ofstream

Archivos de salida

```
int main()
{
 ofstream f( "salida.txt" );
 if ( f.is_open() ) {
 f << "hola " << endl;
 f << 5 << endl;
 }
 else cerr << "Error de apertura del archivo." << endl;
}
```

Para leer y volcar por pantalla este archivo que acabamos de generar, se haría lo siguiente:

```
int main()

{ string s;

 ifstream f( "salida.txt" );

 if ( f.is_open() ) {

        getline( f, s ); cout << s << endl;

         getline( f, s );cout << atof( s.c_str() ) << endl; }

 else cerr << "Error de apertura del archivo." << endl;}
```

Archivos binarios

Los archivos binarios se tratan con la clase fstream.

En este caso, debemos especificar si se desea entrada y salida, o sólo entrada o sólo salida.

Uno de los usos más típicos es utilizar el archivo como una pequeña base de datos, utilizando un registro (struct) como referencia. Recordemos que los registros deben tener todos el mismo tamaño.

Archivos binarios

Así, suponiendo la struct Persona:

```
const unsigned int MaxNombre = 50;
const unsigned int MaxTelefono = 15;
struct Persona {
 char nombre[MaxNombre];
 int edad;
 char telefono[MaxTelefono];
};
```

Archivos binarios

```
void volcar(ostream & o, const Persona &p)


{
        o << nombre << ':' << edad << endl
          << "Tlf:" << telefono << ':' << endl
        ;
}
```

Cuando se deseen guardar registros de este tipo en un archivo, se puede hacer de la manera siguiente:

```
Persona p1;

 Persona p2;

 strcpy( p1.nombre, "Baltasar" );

 strcpy( p1.telefono, "988387028" );

 p1.edad = 33;

 strcpy( p2.nombre, "Pedro" );

 strcpy( p2.telefono, "988387018" );

 p1.edad = 33;
```

Cuando se deseen guardar registros de este tipo en un archivo, se puede hacer de la manera siguiente:

```
ofstream f( "datos.bin", ios::binary );
 if ( f.is_open() ) {
        f.write( (char *) &p1, sizeof( Persona ) );
        f.write( (char *) &p2, sizeof( Persona ) );
 } else cout << "Error de apertura de archivo." << endl;
}
```

Es posible realizar una lectura de cada registro de la siguiente forma:

```
Persona p;
 ifstream f( "datos.bin", ios::binary );
if ( f.is_open() ) {
        f.read( (char *) &p, sizeof( Persona ) );
        while( !f.eof() ) {
                volcar( cout, p );f.read( (char *) &p, sizeof( Persona ) );}
 } else cout << "Error de apertura de archivo." << endl;}
```

ATENCIÓN

Los objetos de la clase string no se pueden utilizar, pues guardan la cadena en el heap, y no dentro de sí mismos. De hecho, no puede utilizarse para archivos ningún objeto que no sea autocontenido, es decir, que no sea simple.

ATENCIÓN

Un archivo binario generado con una máquina o un compilador distinto, del que generó el programa que leerá el archivo pueden hacer que el archivo sea ilegible. Así, si por ejemplo se crea un archivo binario de estructuras Persona en una máquina con Windows XP, y compilador GNU gcc (por ejemplo, mediante el uso de Codeblocks), y se pretende leer ese mismo archivo con un programa creado mediante la misma máquina y Visual Studio, lo más probable es que la lectura no sea correcta, debido a formas distintas entre ambos compiladores de lidiar con la alineación en memoria (en pocas palabras, una estructura o una clase debe ocupar un tamaño que sea múltiplo del tamaño de la palabra de la máquina). En cuanto, a distintas máquinas, una SPARC leerá datos binarios utilizando su propia arquitectura MSB (most significant byte, primero el byte más significativo), mientras una máquina x86 utilizará la LSB (least significant byte, o primero el byte menos significativo). De ahí que, en la actualidad, hayan surgido formatos multiplataforma como el XML.

Bibliografía

http://jbgarcia.webs.uvigo.es/asignaturas/TO/cursilloCpp/14_archivos.html