An Introduction to C++ Files

A **file** is a collection of information that is stored in secondary memory.

The need for files can be seen by noting that:

1) When a computer is powered off, information stored in main memory is lost. Programmers need a way of storing information in a more permanent way.

2) Main memory is limited and sometimes programmers need to deal with very large amounts of information.

3) Programmers and users need a way to share information. It is difficult to share information stored in main memory.

Files provide a solution to all of these problems.

The following operations are common when working with files at a *high level*, for example at the command prompt or while working in a text editor.

- Creating a new file
- Making a copy of a file
- Renaming a file
- Examining information in an existing file
- Appending (adding) information to the end of an existing file
- Modify selected information in a file
- Deleting a file

Programming languages allow programmers to work with files at a *low level* and generally support a number of file operations. This handout focuses on **sequential access files**, which are accessed in a way similar to other sequential devices, such as VCR tapes. C++ also has support for **random access files**, which allow direct access to file components, much like an array.

Happily, C++ file operations are very similar to the operations that we have already seen when working with standard input and output. In most cases, the "same" functions can be used.

A sequential file can be viewed as an ordered stream of bytes. To get data from a sequential access file, it is necessary to open the file, then start at the beginning of the file and read information from the file until the desired information is found. Similarly, to create a sequential file one opens the file and writes the entire stream of bytes to the file. An **ASCII** (**text**) file is an example of a sequential file. The C++ source files that you create in the IDE are also text files.

Writing to And/Or Creating a Sequential File

Creating a text file involves the following 6 steps:

1) Including the header file <`fstream`> in your source file.
2) Declaring a file variable. When using C++ stream I/O this is done by declaring a member of the `ofstream` (output file stream) class. The name used for the file variable is the programmer's choice, but names similar to those used in writing to standard output are recommended.
3) Opening the file for output. When the file is opened, it is necessary to supply a filename to be used by the operating system. This name can be a string constant of a string stored in a string variable. Note that the filename is not the same as the file variable name! If the file does not yet exist, then this will create one for you. If the file already exist, then you will write over and lose original information in the file.
4) Checking to make sure file was successfully opened.
5) Writing information to the file, using the << operator or member functions of `ostream`, such as `put` or `write`.
6) Closing the file.

Example: The program below creates a text file named "POEM.TXT" that contains some of the lines of *Mary Had a Little Lamb*. After the program finishes executing, a file named "POEM.TXT" will exist in the current working directory. The new file will contain four lines.

```
  #include <fstream>                  // 1)

  int main()
  {
    char Filename[] = "POEM.TXT";

    ofstream FileOut;                 // 2)

    FileOut.open(Filename);           // 3) Open file named Filename for output

    if ( FileOut.fail() )             // 4) Be sure to check for error!
    {
      cerr << "Sorry, cannot create " << Filename << endl;
      return 1;
    }

    FileOut  << "Mary had a little lamb." << endl       // 5)
             << "His fleece was white as snow." << endl
             << "And everywhere that Mary went," << endl
             << "The sheep was sure to go." << endl;

    FileOut.close();                  // 6)

    return 0;
}
```

Reading from an Existing Sequential File

If a file already exists, the following steps can be used to retrieve information that has been stored in the file.

1) Including the header file <`fstream`> in your source file.
2) Declaring a file variable, a member of the `ifstream` (input file stream) class that will be used to access data in the file.
3) Open the file for input. When the file is opened, a name must be supplied. This name must correspond to the operating system filename. It can be a string constant of a string stored in a string variable.
4) Checking to make sure file was successfully opened. If file does not exist you should not proceed, but display error message to user.
5) Reading information from the file, using the extraction operator >>, the `get` function, or any of the other appropriate input functions found in *istream*. The "same" functions that were used to read input from standard input can be used to read input from a file.
6) Closing the file.

Below is a program that reads the information in the file "POEM.TXT" and displays it on the user's screen. Note that in C++, EOF is not set to true until at least one char has been read. This is why a priming read is done before entering the while loop below.

```
#include <iostream>
#include <fstream>              // 1)
using namespace std;

int main()
{
  char Filename[] = "POEM.TXT", Ch;

  ifstream FileIn(Filename);   // Steps 2) and 3) above

  if (!FileIn)                 // 4)
  {
    cerr << "Sorry, cannot open " << Filename << endl;
    return 1;
  }

  FileIn.get(Ch);              // 5) Always do priming read.
  while (FileIn.good())        // while not EOF or any other problem
  {
    cout << Ch;                // send chars to stdout
    FileIn.get(Ch);            // read chars from file
  }

  FileIn.close();              // Always close file when done with it

  return 0;
}
```