Lecture 33 — Files & Streams

J. Zarnett
jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

August 27, 2016

Acknowledgments: W.D. Bishop

Files and File Systems

Files permanently store large amounts of data.

Operating systems implement a filesystem to manage files stored on a hard disk (or other types of storage media).

Filesystems typically support the following:

- Creating new files
- Reading from a file
- Writing to a file
- Appending to existing files
- Renaming existing files
- Deleting existing files
- Checking for the existence of a file

Streams

File I/O and console I/O are very similar.

We have used `cout` (an output stream) and `cin` (an input stream) extensively.

Streams are an abstraction for data flow that can be applied to both file I/O and console I/O.

A stream denotes the flow of data from a source to a destination.

File I/O Fundamentals

The file I/O classes are defined in the `fstream` library.

Output streams are created using `ofstream`.

Input streams are created using `ifstream` class.

The `ofstream` and `ifstream` classes provide sequential access to text files: each line must be written or read in sequence.

Files are assumed to store text data.

File Output Example

This code outputs the text “Hello World” to a file named “output.txt”.

```
ofstream outStream;
string filename = "output.txt";
outStream.open( filename );

outStream << "Hello World" << endl;

outStream.close();
```

File Input Example

```
ifstream inStream;
inStream.open("input.txt");

int a;
int b;
int c;

inStream >> a >> b >> c;

cout << a << endl;
cout << b << endl;
cout << c << endl;

inStream.close();
```

[In class demo: output of this program.]

Inputting Formatted Data

The input stream attempts to make sense of whatever it has been fed.

But perhaps it is not that simple and we will read in a line of text that we need to then interpret and split.

There are a couple of things to consider:

- How will the data be used, stored, manipulated, etc.?
- What errors should be detected, corrected, ignored, etc.?

Reading Data Example

Given the following class declaration:

```
class Employee
{
  private:
    ulong identifier;
    string surname;
    string givenName;
    double salary;
};
```

How would you build a program to read this data from a file and display the data on the console?

Step 1: File Format

Before coding anything, consider the format of the input file.

Let's assume a valid input file resembles the following:

```
3 Phaneuf Dion 6500000
81 Kessel Phil 6000000
12 Connolly Tim 5500000
8 Komisarek Mike 5500000
24 Liles John-Michael 4550000
```

This is an easy file to parse since we can use the space characters as our way of splitting a line into its component parts.

Step 2: Updating `Employee`

Next step: to make importing of data efficient, we should add a constructor to `Employee`.

```
Employee::Employee( ulong id, string sname,
                    string gname, double s )
{
    identifier = id;
    surname = sname;
    givenName = gname;
    salary = s;
}
```

Questions to Ponder

How would you deal with input data that includes spaces?

You could use commas or another special character to delimit fields.
...but how would this change your parsing...?

What would you do if you needed to read a file containing a fixed number of records into an array of records?

- Read the file twice: once to determine the number of records and once to store them?
- Store the number of records at the top of the input file?
- Set a maximum number of entries?
- Re-size the array when necessary?
- Use a collection?

Field Delimiter Example

What happens when we try to split `line` below with a space as the delimiter?

```
string line = "A,B;C D;E,F";
```

We get two strings as output:

"A,B;C"

"D;E,F"

Field Delimiter Example 2

What happens when we try to split `line` below with a comma as a delimiter?

```
string line = "A,B;C D;E,F";
```

We get three strings as output:

"A"
"B;C D;E"
"F"

Field Delimiter Example 3

What happens when we try to split `line` below with a semicolon as a delimiter?

```
string line = "A,B;C D;E,F";
```

We get three strings as output:

"A,B"
"C D"
"E,F"

Field Delimiter Example 4

What happens when we try to split `line` below using all three??

```
string line = "A,B;C D;E,F";
```

We get six strings as output:

"A"
"B"
"C"
"D"
"E"
"F"

Terminology Review

In our discussion of file I/O, we have introduced a number of important database terms:

1. A file is form of long-term, persistent data storage
2. A database file is a collection of records
3. A record is a collection of fields separated by delimiters
4. A field is a group of bits (or characters)
5. A delimiter is a special character used to separate fields

A detailed discussion of databases is beyond the scope of this course.