# File I/O - Reading Data Files

For anyone who works with data -- scientists analyzing experimental data, linguists computing word distributions, or sports fans compiling statistics -- the ability to read data files is essential to making C programming a useful tool. Suppose I had had to type all of your Exam 1 and 2 scores into my source code in order to compute the basic statistics I showed in class. The process would be error-prone and inconvenient, and if I wanted to compute the same statistics for a different data set, I'd have to type all that new data in.

The grade data that I used for the example program consisted of two floating point numbers per line, separated by a comma. The first number was a particular student's Exam 1 score, and the second number was the same student's Exam 2 score. I wanted to read all the Exam 1 and 2 scores into to separate arrays, `ex1[]` and `ex2[]`, so that I could do things like compute the mean, median, min, and max scores as well as the grade distributions.

As discussed in class, there are three *big* steps to reading data from a file:

1. Open the file
2. Read the data
3. Close the file

We'll look at each of these steps in more detail, but first we need to address some "housekeeping" issues.

To you and me, a file is identified by its name. In the example, the grade data was stored in `grades.dat`. The computer needs to know the file name to find the file initially, but once it has found and opened the file, we will reference it with what is called a *file pointer*. Consider:

```
FILE *fp;
```

This line declares `fp` as a pointer to a `FILE` type. We will reference the file with the pointer `fp`. You can call your pointer whatever you want within the constraints on C variables names, and if you are working with multiple files, each will have to have it's own distinct file pointer name. For example, if my program reads one file an writes to a different file, I will probably have two file pointers, e.g.

```
FILE *in_fp;          // Pointer to input file
FILE *out_fp;         // Pointer to output file
```

At this point, it is okay if you just think of this as an incantation: to work with a file, you must declare a file pointer, and the syntax is "`FILE *<ptr_name>`" where `<ptr_name>` is whatever name you give to the file pointer.

As an aside, for those of you who are comfortable with the idea of a pointer, the use of "*" in the declaration of the file pointer is an example of a general C syntax. The "*" tells C that `fp` is a *pointer* to a variable of `FILE` type: without it, `fp` would *be* a variable of `FILE` type. This works with other types as well. For example "`int *num_ptr;`" declares `num_ptr` to be a pointer to a variable of type `int`.

## Opening the File

To open a file, we need to specify the file name and the *mode*. There are three possible values for the mode:

1. write ("w") - write to the file, destroying any existing file content
2. read ("r") - read from the file
3. append ("a") - write to the file, appending new data to the end of any existing file content

Since we are talking about reading data, we will use a mode value of "r".

**Example**: open the file `grades.dat` for reading

```
FILE *fp;
fp = fopen("grades.dat", "r");
```

Once the file is open, we will use fp to refer to it.

**Example**: suppose the file `temp_20140430.dat` is in a subdirectory `temp_data` of the current working directory Open the file for reading.

```
FILE *temp_fp;
temp_fp = fopen("temp_data/temp_20140430.dat", "r");
```

## Reading the Data

Before we can read the data in the file, we need to understand how it is formatted. In the grades example, each line of the data file consisted of two floating point numbers separated by a comma. The numbers on a line were the Exam 1 and Exam 2 scores for a particular student. Here's a small section of the data file:

```
71.25,78.44
57.50,78.67
74.50,84.67
56.00,52.00
96.00,97.33
78.75,95.33
81.75,91.33
```

Note that we can describe the content of a line with the format string "`%f,%f`" (a floating point number, a comma, and a second floating point number). This is important, because the function we will use to read the data, `fscanf()`, needs to know the format of our data so that it can read it correctly.

We will also need to tell the `fscanf()` function where to save the values that it reads from the file. In the example, I declaretwo `float` arrays, `ex1[]` and `ex2[]`, and want the data to be stored in those arrays.

Lastly, we need to tell `fscanf()` what file to use by passing it a file pointer. Here is the line from the sample code (in the code it is contained within a for-loop with control variable `i`):

```
fscanf(fp, "%f,%f", &ex1[i], &ex2[i]);
```

Note the use of "`&`" on the destination variables, just as with the `scanf()` function that we have used to read data from the keyboard.

## Closing the File

This is the easiest part. If our file has file pointer `fp`, then we close the file with:

```
fclose( fp );
```

**Example:** Suppose I want to read a file `temp_data.txt` consisting of the daily high temperature reading for a year. There are 365 lines in the file with each line containing a single floating-point number, the high temperature for the day (1st line - January 1, 2nd line - January 2, etc.). Write the C code to read the data in to an array.

```
#include <stdio.h>

#define NUMDAYS 365

int main() {
    float temps[NUMDAYS];
    int i;
    FILE *fp;

    /* Open the file */

    fp = fopen("temp_data.txt", "r");

    /* Read the data */

    for (i=0; i<NUMDAYS; i++) {
        fscanf(fp, "%f", &temps[i]);
    }

    /* Close the file */

    fclose( fp );

    return 0;
}
```

**Example:** I have a file `sales.txt` containing sales information for a small business over the last 180 days. Each line of the file is the sales information for a single day: the number of customers seen that day and the gross sales for the day. The two values on each line are separated by a tab character. Here's small sample from the data file

```
17   185.72
10   121.30
11   109.50
```

Write C code to read the data into two arrays, one holding the number of customers each day and the other the gross sales.

```
#include <stdio.h>

#define NUMDATA    180

int main () {
    int customers[NUMDATA];
    float sales[NUMDATA];
    int i;
    FILE *fp;

    /* Open the file */

    fp = fopen("sales.txt", "r");

    /* Read the data */

    for (i=0; i<NUMDATA; i++) {
        fscanf( fp, "%d\t%f", &customers[i], &sales[i] );
    }

    /* Close the file */

    fclose( fp );

    return 0;
}
```