Lesson 7

Files

Class Objectives

In this lesson, you will learn:

- A simple definition of what files are in the C language
- The operations that can be performed on files
- How to do those operations
- How to write C programs that use files

Important Notes:

- **You should type in all the programs in this handout, and run them more than once with different data**
- **You should read, and understand everything in this handout, the material in it forms the basis of the quizzes**
- **If you don't understand something; ask me to explain.**

What are files?

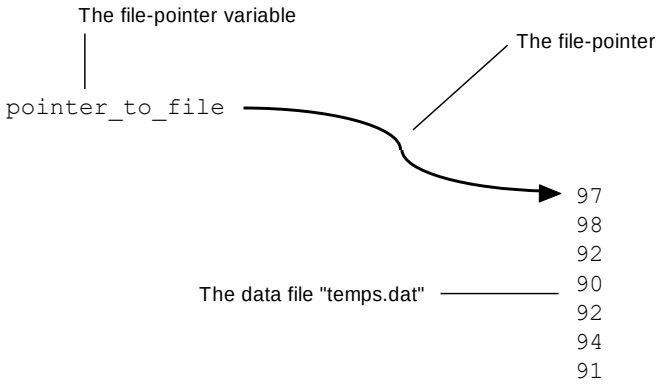
A simple definition of a file in the C language, is that they are special data types (outside the program) that are a series of ASCII characters, with the last character of the data type being the special end-of-file character "`EOF`".

File pointers

To access a file we need to use a *file-pointer*. A file-pointer is a special variable type for use with files.

It may be useful to think of a file-pointer as a piece of string, the program holds one end of the string (as a variable name), and the other end is attached to the open file. This is how the program gets access to the file.

In all of the programs in this handout, the file-pointer is named "`pointer_to_file`". Below, is a pictorial representation of the file-pointer and the file "`temps.dat`".



NOTE: There must be a file-pointer for each open file in a program.

File operations

Just as we can perform certain operations on variables of type `int`, such as addition, subtraction, multiplication, and division, files also have a set of operations that can be performed on them. They are:

open (Files must be opened before they can be used)
read (Once opened, we can read data from a file)
write (Once opened, we can write data to a file for later use)
close (When we are finished using a file in a program, we must close it)

Opening a file

To open a file, we must use the file opening function `fopen()`, this attaches the file-pointer (our piece of string) to the file. The function `fopen()` takes two parameters, and its return type is a file-pointer. Horton, page 431 shows the function prototype:

```
FILE *fopen(char *name, char* mode);
```

Don't worry about what those `*` characters mean right now, these file functions are easier to use than they look. The word `FILE` must be in capitals. The first parameter in a call of `fopen()` is the name of the file to be opened, the second is the file mode.

File mode

File mode is how you tell the program what you are going to do with the file while it's open.

There are three file modes:

- **"r"** (open for reading data from the file)
- **"w"** (open for writing data to the file)
- **"a"** (open for adding data to the end of the file)

In the first program we will look at (`read_write1.c`), the variable that holds the name of the file is declared as a character array variable:

```
char file_name[15];
```

The file-pointer variable is declared as a variable of type `FILE`:

```
FILE *pointer_to_file;
```

To "read" this line, we need to do it backwards (left to right), and in three steps:

1. The variable `pointer_to_file`
2. "Points to" `*`
3. A variable of type `FILE`

Opening a file for reading

In the program (`read_write1.c`) we open the program for reading with the line of code:

```
pointer_to_file = fopen(file_name,"r");   /* open file for reading */
```

It is the "`pointer_to_file =`" that attaches the file-pointer to the file. From that point on, when we want to refer to the file, we just use: `pointer_to_file`

Reading from a file

Some standard input/output functions, and file input/output functions.

Standard Input/Output Functions		File Input/Output Functions	
`printf()`	Prints formatted parameters to the screen	`fprintf()`	Prints formatted parameters to a file
`scanf()`	Reads input from the keyboard to first whitespace	`fscanf()`	Reads input from a file to first whitespace
`gets()`	Reads a line of input from the keyboard (doesn't keep newline character)	`fgets()`	Reads a line of input from a file (keeps newline character)
`getchar()`	Reads a character from the keyboard	`fgetc()`	Reads a character from a file
`putchar()`	Writes a character to the screen	`fputc()`	Writes a character to a file

In the program `read_write1.c`, we use the file equivalent of the `gets()` to read a line of text. The `fgets()` has the form:

```
fgets(line_of_text,75,pointer_to_file);
```

The `fgets()` has three parameters:

1. The name of the character array for storing the line of text
2. The maximum number of characters to be read from each line of the file
3. The name of the file-pointer

Closing a file

In the program `read_write1.c`, when the `EOF` character is read, the `while()` loop terminates, and the line:

```
fclose(pointer_to_file);   // close file
```

closes the file.

Program (read_write1.c)

```
/*
  Program read_write1.c
  Written by: Joe Dorward
  Date: 04/28/00

  This program will open a file and print its contents to the screen
*/

#include <stdio.h>

void main(void)
{
char file_name[15],  // variable for storing the file name
     line_of_text[75];

FILE *pointer_to_file;   // the file pointer variable


  printf("Filename to use: ");   // ask user for filename
  gets(file_name);


  pointer_to_file = fopen(file_name,"r");   // open file for reading


  fgets(line_of_text,75,pointer_to_file);   // read first line from file


  while (!feof(pointer_to_file))   // while not end-of-file
  {
    printf("%s",line_of_text);
    fgets(line_of_text,75,pointer_to_file);   // read another line from file
  }


  fclose(pointer_to_file);   // close file
}
```

You can test this program, by typing the name of any filename at the prompt.

Try using the files:

```
temps.dat
gold.dat
words.txt
class_list.dat
```

Program (temps3.c)

This next program is a file reading version of program temps2.c, that reads its data from the file temps.dat

```
/*
  Program temps3.c
  Written by: Joe Dorward
  Date: 04/26/00

  This program is based on temps2.c, it reads temperatures from a file, and
  puts them into an array.

  The program then scans the array for the highest, and low temperature, and
  prints them out.
*/

#include <stdio.h>

void main(void)
{
char file_name[15];  // variable for storing the file name

FILE *pointer_to_file;   // the file pointer variable

int element_number,      // declare a loop counter
    highest_temperature = 0,
    lowest_temperature = 200,
    daily_high_temperatures[35];   // declare an array


  printf("Enter the temperatures filename: ");  // ask user for filename
  gets(file_name);

  pointer_to_file = fopen(file_name,"r");   // open file for reading

  // ============================================================================
  // load data into the array

  for (element_number = 1; element_number <= 30; element_number++)
  {
  fscanf(pointer_to_file,"%d",&daily_high_temperatures[element_number]);
  }

  fclose(pointer_to_file);   // close file

  // ============================================================================
  // loop through array and find highest and lowest temperature

  for (element_number = 1; element_number <= 30; element_number++)
  {
  // find highest temperature
  if (daily_high_temperatures[element_number] > highest_temperature)
  {
    highest_temperature = daily_high_temperatures[element_number];
  }

  // find lowest temperature
  if (daily_high_temperatures[element_number] < lowest_temperature)
  {
    lowest_temperature = daily_high_temperatures[element_number];
  }
  }
  // ============================================================================
  // print highest temperature

  printf("\n This week's highest high temperature was: ");
  printf("%d degrees\n",highest_temperature);

  // print lowest temperature
```

```
  printf(" This week's lowest high temperature was: ");
  printf("%d degrees\n\n",lowest_temperature);
}
```

Writing to a file

So far, we have only written programs that read from files, and print the contents of those files to the screen. This next program opens the file in "a" mode so that we can add a name to the data file class_list.dat. In the call of the function fopen() we have named the file literally with "class_list.dat", so the program doesn't need to ask us for the name of the file. When you write a program, and you want to fix the name of the data file, you specify its name this way. So:

```
pointer_to_file = fopen("class_list.dat","a");   // open for adding
```

opens the file for adding. Then the program asks for a student name. The lines:

```
fprintf(pointer_to_file,"%s",student_name);
fputc('\n',pointer_to_file);   // start newline in the file
```

adds the name to the file, and adds a newline. The file is closed with:

```
fclose(pointer_to_file);   // close file
```

Next, the file is re-opened for reading, and (as we have seen before) the lines:

```
fgets(student_name,25,pointer_to_file);   // read first name from file

while (!feof(pointer_to_file))   // while not end-of-file
{
  printf("%s",student_name);
  fgets(student_name,25,pointer_to_file);   // read name from file
}
```

reads, and prints the contents of the file to the screen. Each time you run this program you will add another name to the file.

You can look at the file class_list.dat, by opening it using Notepad.

Program (`class_list1.c`)

NOTE: You do not need a file named `class_list.dat` before you run this program, if one by that name doesn't exist, it will be created. Once created, however, that file will have names added to it.

```
/*
  Program class_list1.c
  Written by: Joe Dorward
  Date: 04/26/00


  This program will open the file "class_list.dat",
  and allow the user to add a name to the file
*/

#include <stdio.h>

void main(void)
{
char student_name[25];

FILE *pointer_to_file;   // the file pointer variable

  pointer_to_file = fopen("class_list.dat","a");   // open for adding

  printf("\nPlease enter the student\'s name: ");
  gets(student_name);

  fprintf(pointer_to_file,"%s",student_name);
  fputc('\n',pointer_to_file);   // start newline in the file

  fclose(pointer_to_file);   // close file
  // ========================================================================
  pointer_to_file = fopen("class_list.dat","r");   // open for reading

  fgets(student_name,25,pointer_to_file);   // read first name from file

  while (!feof(pointer_to_file))   // while not end-of-file
  {
    printf("%s",student_name);
    fgets(student_name,25,pointer_to_file);   // read name from file
  }

  fclose(pointer_to_file);   // close file
}
```