Files

Prepared by: RNC Recario
rncrecario@gmail.com
Institute of Computer Science UPLB
Jan 2012

OBJECTIVES
At the end of the laboratory session, the student is expected to
- Define what a file is
- Identify and differentiate the two types of files
- Create a program using files

Files

So far we have already discussed atomic and non-atomic data types as well as static and dynamic data types. While the said data types are different and have different purposes for usage, they still have one thing in common. When the program terminates, the values stored in the memory, regardless of the data type and approach of employing data types, are deleted. Now, there are certain situations that the data needs to be preserved for the next run of a program. In these cases, files are used.

File, by definition, is a collection of bytes stored in the secondary memory of a computer. With files, data used and processed by a program can be stored and accessed again even if the program terminates.

In C, a file is accessed and manipulated through a declaration of a file pointer. The syntax is

```
      FILE *<filepointername>;
```

It should be noted that a file pointer should be assigned only to one file.

There are two types of files available. The first is the **text file** and the second one is the **binary file**. Essentially the two files are the same except that a text file takes extra processing and that it is human readable whereas a binary file is not.

File Operations and File Modes

A file has three operations applicable to files in general: file, file access and file close. A file access can be more elaborated as a file read, file write or file append. The type of access determines the mode.

Before we discuss modes in detail, here is the syntax for file open.

```
filepointer = fopen("<filename>","<mode>");
```

File opening using `fopen()` function allows you to well, open the file for file access. If the file is found, it returns an address. The following are the valid modes for your `fopen()` function.

"r" or "rt"	open a text file for reading
"w" or "wt"	open a text file for writing
"a" or "at"	append to a text file
"rb"	open a binary file for reading
"wb"	create a binary file for writing
"ab"	append to a binary file
"r+" or "r+t"	open a text file for read/write
"w+" or "w+t"	create a text file for read/write
"a+" or "a+t"	open or create a text file for read/write
"r+b"	open a binary file for read/write
"w+b"	create a binary file for read/write
"a+b"	open or create a binary file for read/write

With the different modes used, here are possible functions used to write/append and/or read a file.

Read	**Write/Append**	**Description**
`getc()`		Gets one character from the text file
	`putc()`	Writes one character into the text file
`fscanf()`		Reads a formatted set of values from the text file. This can be used for automatically formatting file data into `int`, `char`, `float`, etc.
	`fprintf()`	Writes a formatted set of data into the text file.
`fgets()`		Reads a block of characters/string (from the text file) with specific size `size_n` and puts it in a string.
	`fputs()`	Writes a string into a text file
`fread()`		Reads a block of data from the binary file
	`fwrite()`	Write a block of data from the binary file

An `fclose()` operation should follow after a file access to close (hence *`fclose()`*) the file and to make the changes permanent. The syntax for `fclose()` is

```
fclose(<filepointer>);
```

The function fclose is usually used in an `if-else` statement because `fclose()` has a return value to determine if the file has been properly closed or not. In most cases however, this is not done because we assume that the file is closed properly.

Exercise 8: Files

Prepared by: RNC Recario
rncrecario@gmail.com
Institute of Computer Science UPLB
Jan 2012

Using your previous exercise, extend it by adding files. Whenever the user exits, the program first writes the contents of the structure into a file named `<section>_exer8_<lastname>.txt`. You also have to preserve the count of students in the records. You can write/save it on the same text file or create a separate one (filename? *Kayo nang bahala!*)

Because of the extension, you are now allowed to add extra functions and variables.

Note: If deemed necessary, you can declare and use other variables but **do not** declare them GLOBALLY!

Once done, DO NOT forget to:

- FULLY document your work!
- Change the filename to appropriate name!!! Filename is `<section>_exer8_<lastname>.c`
- Send to your lab instructor's email OR follow the submission guideline stated for this exercise!

PS: STUDENTS WHO CHEAT WILL AUTOMATICALLY RECEIVE A GRADE OF 5.0.

Enjoy! ☺