PROGRAMMING METHODOLOGY

Lab 9: File I/O

1 Introduction

Before this lab, all inputs to your program are from keyboard, using ***scanf*** function, and output the screen, using ***printf*** function. In this lab, we'll introduce another way to interact with your program by using file input/output.

2 File I/O

A file represents a sequence of bytes, regardless of it being a text file or a binary file. C programming language provides access on high level functions as well as low level (OS level) calls to handle file on your storage devices. We'll go through the important calls for file management.

Opening File

You can use the **fopen()** function to create a new file or to open an existing file. This call will initialize an object of the type **FILE**, which contains all the information necessary to control the stream. The prototype is as follows:

```
FILE *fopen( const char * filename, const char * mode );
```

Here, filename is a string literal, which represents the path to your input file. And mode is an access mode and can have one of the following values:

Mode	*Description*
r	Opens an existing text file for reading purpose.
w	Opens a text file for writing. If it does not exist, then a new file is created. Here your program will start writing content from the beginning of the file.
a	Opens a text file for writing in appending mode. If it does not exist, then a new file is created. Here your program will start appending content in the existing file content.
r+	Opens a text file for both reading and writing.
w+	Opens a text file for both reading and writing. It first truncates the file to zero length if it exists, otherwise creates a file if it does not exist.
a+	Opens a text file for both reading and writing. It creates the file if it does not exist. The reading will start from the beginning but writing can only be appended.

In case you want to handle a binary file, you need to insert character **'b'** after your access mode, *i.e.*, **"rb"**, **"a+b"**.

Closing a File

To close a file, use the **fclose()** function. The **fclose()** function returns zero on success, or EOF if there is an error in closing the file. This function flushes any data still pending in the buffer to the file, closes the file, and releases any memory used for the file. The EOF is a constant defined in the header file **<stdio.h>**.

```
int fclose( FILE *fp );
```

Writing a File

To write data to an output file, we use **fputc()** function, which writes the character value of the argument *c* to the output stream referenced by *fp*. It returns the written character written on success otherwise EOF if there is an error.

```
    int fputc( int c, FILE *fp );
int fputs( const char *s, FILE *fp);
```

The function **fputs()** writes the string *s* to the output stream referenced by *fp*. It returns a non-negative value on success, otherwise EOF is returned in case of any error.

The program below will illustrate how-to write data to a file.

```
1   #include <stdio.h>
2
3   int main()
4   {
5     FILE *fp;
6
7     fp = fopen("C:\\output.txt", "w+");
8     fputs("Faculty of Information Technology\n", fp);
9     fputs("Ton Duc Thang University", fp);
10    fclose(fp);
11
12    return 0;
13  }
```

Reading a File

To read a single character from a file, we use **fgetc()** function, which reads a character from the input file referenced by *fp*. The return value is the character read, or in case of any error, it returns EOF.

```
int fgetc( FILE * fp );
int *fgets( char *buf, int n, FILE *fp );
```

The functions **fgets()** reads up to *n* - 1 characters from the input stream referenced by *fp*. It copies the read string into the buffer *buf*, appending a null character to terminate the string.

The program below will illustrate how-to read string data from a file.

```
1   #include <stdio.h>
2
3   int main()
4   {
5     FILE *fp;
6     char buff[255];
7
8     fp = fopen("C:\\input.txt", "r");
9
10    fgets(buff, 255, (FILE*)fp);
11    printf("%s", buff );
12
13    fgets(buff, 255, (FILE*)fp);
14    printf("%s", buff );
15
16    fclose(fp);
17
18    return 0;
19  }
```

3 Exercises

1. Read the input data (`input01.txt`) contains 20 integer numbers, then write to new file the sum of them.
2. Read the input data (`input02.txt`) contains 5 strings, then write to new file the length of each string.
3. Write a program to illustrate a search program, the program's scenario is as follows:
   - Print to the console: `"enter a filename:"`.
   - Read a string from the console for the filename.
   - Print a menu of the algorithms you implemented, having each represented by a letter or number.
   - Print to the console: `"choose one of these search algorithms:"`.
   - Read a letter or number indicating the algorithm.
   - Print to the console: `"enter a value to search for:"`.
   - Read an integer from the console as your target value, then print the position or print `"not found"`.

4 Resource

All lab resources are available on course homepage **[5]**.

5 Reference

[1] Brian W. Kernighan & Dennis Ritchie (1988). *C Programming Language, 2nd Edition*. Prentice Hall.

[2] Paul Deitel & Harvey Deitel (2008). *C: How to Program, 7th Edition*. Prentice Hall.

[3] *C Programming Tutorial* (2014). Tutorials Point.

[4] *C Programming* (2013). Wikibooks.

[5] *Course homepage* – http://it.tdt.edu.vn/~dhphuc/teaching/cs501042/