EUPHORIA - FILES I/O

Using Euphoria programming language you can write programs that read and change file data on your floppy drive or hard drive, or create new files as a form of output. You can even access devices on your computer such as the printer and modem.

This chapter will cover all ithe basic I/O functions available in Euphoria. For more functions please refere to standard Euphoria documentation.

Printing to the Screen

The simplest way to produce output is using the *puts* statement where you can pass any string to be displayed on the screen. There is another method *printf* which can also be used in case you have to format a string using dynamic values.

These methods convert the expressions you pass them to a string and writes the result to standard output as follows −

```
#!/home/euphoria-4.0b2/bin/eui

puts(1, "Euphoria is really a great language, isn't it?" )
```

This would produce following result on your standard screen −

```
Euphoria is really a great language, isn't it?
```

Opening and Closing Files

Euphoria provides basic methods necessary to manipulate files by default. You can do your most of the file manipulation using open, close, printf, gets and getc methods.

The *open* Method

Before you can read or write a file, you have to open it using Euphoria's built-in *open* method. This function creates a file descriptor which would be utilized to call other support methods associated with it.

Syntax

```
integer file_num = open(file_name, access_mode)
```

Above method returns -1 in case there is an error in opening the given file name. Here is detail of the paramters −

- **file_name:** The file_name argument is a string value that contains the name of the file that you want to access.
- **access_mode:** The access_mode determines the mode in which the file has to be opened ie. read, write append etc. A complete list of possible values is given below in the table.

Here is a list of the different modes of opening a file:

Modes	Description
r	Opens a text file for reading only. The file pointer is placed at the beginning of the file.
rb	Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file.
w	Opens a text file for writing only. Overwrites the file if the file exists. If the file does not

	exist, creates a new file for writing.
wb	Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
u	Opens a file for both reading and writing. The file pointer will be at the beginning of the file.
ub	Opens a file for both reading and writing in binary format. The file pointer will be at the beginning of the file.
a	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
ab	Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.

Example

Following is the example to create a new text file in your current directory on your Linux −

```
#!/home/euphoria-4.0b2/bin/eui

integer file_num
constant ERROR = 2
constant STDOUT = 1

file_num = open("myfile,txt", "w")
if file_num = -1 then
    puts(ERROR, "couldn't open myfile\n")
else
     puts(STDOUT, "File opend successfully\n")
end if
```

If file opens successfully then it would create "myfile.txt" in your current directory and would produce following result −

```
File opend successfully
```

The *close* Method:

The close method flushes any unwritten information and closes the file, after which no more reading or writing can be done on the file.

Euphoria automatically closes a file when the reference object of a file is reassigned to another file. It is a good practice to use the close method to close a file.

Syntax

```
close( file_num );
```

Here passed parameter is the file descriptor which was received while opening a file.

Example

Following is example which would create a file as above and then would close it before existing the program −

```
#!/home/euphoria-4.0b2/bin/eui

integer file_num
constant ERROR = 2
```

```
constant STDOUT = 1

file_num = open("myfile.txt", "w")
if file_num = -1 then
    puts(ERROR, "couldn't open myfile\n")
else
     puts(STDOUT, "File opend successfully\n")
end if

if file_num = -1 then
    puts(ERROR, "No need to close the file\n")
else
     close( file_num )
     puts(STDOUT, "File closed successfully\n")
end if
```

This would produce following result −

```
File opend successfully
File closed successfully
```

Reading and Writing Files

Euphoria provides a set of access methods to make our lives easier while reading or writing a file either in text mode or binary mode. We would see how to use *printf* and *gets* methods to read and write files.

The *printf* Method

The *printf* method writes any string to an open file.

Syntax

```
printf(fn, st, x)
```

Here is detail of the paramters −

- **fn:** File descriptor recieved from open method.
- **st:** Format string where decimal or atom would be formatted using %d and string or sequence would be formatted using %s.
- **x:** If x is a sequence, then format specifiers from st are matched with corresponding elements of x. If x is an atom, then normally st will contain just one format specifier and it will be applied to x, however if st contains multiple format specifiers, each one will be applied to the same value x.

Example

Following example would open a file and would write name and age of a person in this file −

```
#!/home/euphoria-4.0b2/bin/eui

integer file_num
constant ERROR = 2
constant STDOUT = 1

file_num = open("myfile.txt", "w")
if file_num = -1 then
    puts(ERROR, "couldn't open myfile\n")
else
     puts(STDOUT, "File opend successfully\n")
end if

printf(file_num, "My name is %s and age is %d\n", {"Zara", 8})
```

```
if file_num = -1 then
    puts(ERROR, "No need to close the file\n")
else
     close( file_num )
     puts(STDOUT, "File closed successfully\n")
end if
```

The above method example creates *myfile.txt* file and would write given content in that file and finally it would close that file. If you would open this file, it would have following content.

```
My name is Zara and age is 8
```

The *gets* Method

The *gets* method read a string from an open file.

Syntax

```
gets(file_num)
```

Here passed parameter is file description return by the opend method. This method starts reading from the beginning of the file line by line. The characters will have values from 0 to 255. The atom -1 is returned on end of file.

Example

Let's take a file *myfile.txt* which we have created above.

```
#!/home/euphoria-4.0b2/bin/eui

integer file_num
object line

constant ERROR = 2
constant STDOUT = 1

file_num = open("myfile.txt", "r")
if file_num = -1 then
    puts(ERROR, "couldn't open myfile\n")
else
     puts(STDOUT, "File opend successfully\n")
end if

line = gets(file_num)

printf( STDOUT, "Read content : %s\n", {line})

if file_num = -1 then
    puts(ERROR, "No need to close the file\n")
else
     close( file_num )
     puts(STDOUT, "File closed successfully\n")
end if
```

This would produce following result –

```
File opend successfully
Read content : My name is Zara and age is 8

File closed successfully
```

File & Directory Related Methods

Euphoria provides a list of many methods which helps you in manipulating files. Thes methods are listed in Euphoria Library Routines