An introduction to working with files.

The most popular form of data files are the sequential files. This means that the data items are not packed per record and therefore there is no record structure in those files. All items are placed in the file as is. When building a file like that, Liberty BASIC will automatically place a combination of two bytes CHR$(13)+CHR$(10) after every chunk of data you place in the file. This means that every time you issue a - print #handle, - command, that LB will places these two bytes 0D 0A (hexadecimal for 13 and 10 decimal) at the rear.
Here I am building a sequential file called “testfile.txt”. We use the output (output something to a file) as the purpose here. Both item strings are 13 characters long. I did this deliberately.

```
Open “testfile.txt” for output as #m
Print #m, “First... item”
Print #m, “Second.. item”
Close #m
```

You can see the contents by opening the file with your Windows notepad program. The Windows program “Explorer” will know from the extension “.txt” that you need to open this file with notepad. Give it the following command

```
Run “explorer.exe testfile.txt”
```

To see the 0D 0A bytes you need a hex viewer program. But here is a small listing to view every byte in a file in hex format. So, now you see what you get.

```
filedialog DefaultDir$,"*.*",filename$
if filename$ = "" then close #m : end
open filename$ for random as #m len = 1
field #m, 1 as a$

for w = 1 to lof(#m)
get #m, w
print right$("0"+dechex$(asc(a$)),2);" ";
next w

close #m
```

Of course you may print everything you want and in every way you want to a file.
By this I mean that you may place a comma after every word like this: Print #m, “word”;”,”;
This would create a CSV (comma separated values) and that’s the file format that Microsoft uses with EXCEL. You can concatenate the strings and so on.
Every time you open a file with the purpose to output to it or to input from it, then your file-pointer starts at byte 0.
You can change the file pointer with the SEEK command. The seek command works with OUTPUT, INPUT and APPEND purpose opened files as well.
This is not described in any manual or text book. The seek command was invented to use with Binary files. Binary files I will explain later on.
So here I will open the testfile again to write to it at a “random” selected place.

```
Open “testfile.txt” for output as #m
Print #m, “First... item”
Print #m, “Second.. item”
Seek #m, 0
Print #m, “Eerste”;
Close #m

Run “explorer.exe testfile.txt”
```

Run your explorer program to view the contents of your file now, and don’t be surprised to see.
Eerste.. item
Second.. item
By the way, “eerste” is Dutch for “first”. Trust me, it sounds like First without the F. So we wrote “First” to the file and changed it to “Eerste” after all.
Why did I make my items 13 bytes long? Well, in my case I chose for a “record” length of 13 bytes. Now I can changed (overwrite) the 13 bytes of each item.
As long as you don’t exceed the length of a written item, you can overwrite it’s contents.
Although it looks promising, there is nothing to gain here. Random access file have a record structure with fielding’s to accomplish the same as I did here.
Here’s how this testfile would look in RAF (Random access file) style.

```
Open “testfile.txt” for random as #m len = 13
Field #m, 13 as a$

a$ = “First... item”
put #m, 1

a$ = “Second.. item”
```

```
put  #m, 2

a$ =  "Eerste.. item"
put #m,1

a$ =  "Extra... item"
put #m,3


Close #m
Run "explorer.exe testfile.txt"
```

There are some troubles with RAF programming. You can't replace the len argument in the open statement by a variable.
This can't be done within the field command either.
Let's open the "testfile.txt" again, to read all item from it. Now we will need the input (input something from a file) purpose of opening a file.

```
Open "testfile.txt" for input as #m
input #m, a$
input #m, b$
Close #m
```

You can also get the data from the file by using a line input command in stead of what I used here.
Line input will read the data until it meats a CHR$(13)+CHR$(10), while input will read until it meats a comma or the return + linefeed combination.
There is no special thrill with the other commands in this area.
They are:
1. The Loc(# function to display the file-pointer position (location in the file) at that moment .
2. The Eof(# function which gives a true or false status depending if the "end of file" has been reached.
3. The lof(# function which gives you the "length of the file".

Liberty BASIC has some special features.
This is how we read and display the contents of a sequential file most of the time.

```
Open "testfile.txt" for input as #m
While eof(#m) = 0
    Line input #m, a$
    Print a$
Wend
Close #m
```

But you can read the file this way too.

```
Open “testfile.txt” for input as #m
a$ = Input$( #m,lof(#m))
Close #m
Print a$
```

Marvelous that special INPUT$(# function that you won’t find anywhere else.
The special function INPUT$(even comes in handy when we need INKEY$,
because INKEY$ is the same as INPUT$(1) when we work in the mainwin world.

Here is a link to the accompanying lesson.
Lesson on seeking in sequential files

Gordon Rahman