

**File**

<b>COLLABORATORS</b>
----------------------

	<i>TITLE :</i> File		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		February 8, 2025	

<b>REVISION HISTORY</b>
-------------------------

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>File</b>	<b>1</b>
1.1	File V1.00 . . . . .	1
1.2	closefile . . . . .	1
1.3	createfile . . . . .	2
1.4	eof . . . . .	2
1.5	fileseek . . . . .	2
1.6	initfile . . . . .	3
1.7	loc . . . . .	3
1.8	lof . . . . .	3
1.9	openfile . . . . .	3
1.10	readbyte . . . . .	4
1.11	readfile . . . . .	4
1.12	readlong . . . . .	4
1.13	readstring . . . . .	4
1.14	readword . . . . .	5
1.15	usefile . . . . .	5
1.16	writebyte . . . . .	5
1.17	writelong . . . . .	5
1.18	writestring . . . . .	5
1.19	writeword . . . . .	6

# Chapter 1

## File

### 1.1 File V1.00

Pure Basic - File library V1.00

The file is the main way of storage for nowadays computers. With the Pure Basic you manage them in very simple and optimized way. Any number of files can be handled at the same time. This library use buffered functions to increase the writing/reading speed.

Commands summary in alphabetical order:

- CloseFile
- CreateFile
- Eof
- FileSeek
- InitFile
- Loc
- Lof
- OpenFile
- ReadByte
- ReadFile
- ReadLong
- ReadString
- ReadWord
- UseFile
- WriteByte
- WriteLong
- WriteString
- WriteWord

Example:

File demo

### 1.2 closefile

---

#### Syntax

```
CloseFile(#File)
```

#### Description

Close the specified #File and it can't be used anymore for later use. Closing a file ensure the buffer will be put effectively on the disk.

Note: on the program end, Pure Basic is smart enough to close all the unclosed files, so you don't need to do it yourself.

### 1.3 createfile

#### Syntax

```
Result = CreateFile(#File, FileName$)
```

#### Description

Open an empty file. If the file was existing, it open it and replace it by a blank one ! Careful. If 'Result' is not null, the file is effectively created and you get the AmigaOS pointer to the file (for advanced programmers). If Result is NULL, the creation has failed. It must be always tested, as performing operations on a non created file will cause severe crashes.

### 1.4 eof

#### Syntax

```
Result = Eof()
```

#### Description

Eof stands for 'End Of File'. It will return true if you have reached the end of the current file, NULL else.

### 1.5 fileseek

#### Syntax

```
FileSeek(NewPosition)
```

#### Description

It designed to change the current pointer position inside the file.

---

## 1.6 initfile

### Syntax

```
Result = InitFile(#NumMaxFiles)
```

### Description

With `init` all the file environnement for future use. You must call this function before any other file functions.

`#NumMaxFiles` = Number of maximum files you need to handle simultanely.

## 1.7 loc

### Syntax

```
Position = Loc()
```

### Description

Return the actual pointer position inside the file.

## 1.8 lof

### Syntax

```
Length = Lof()
```

### Description

`Lof` stands for 'Length Of File'. It will return the length of the current file.

## 1.9 openfile

### Syntax

```
Result = OpenFile(#File, FileName$)
```

### Description

It will open the designed file or create it if it doesn't exists. You can perform read and write on this file. If 'Result' is not null, the file is effectively opened and you get the AmigaOS pointer to the file (for advanced programmers). If Result is NULL, the creation has failed. It must be always tested, as performing operations on a non created file will cause severe crashes.

---

## 1.10 readbyte

Syntax

```
Number.b = ReadByte()
```

Description

Read one byte on the current opened file.

## 1.11 readfile

Syntax

```
Result = ReadFile(#File, FileName$)
```

Description

Open an existing file for read only operations. If 'Result' is not null, the file is effectively opened and you get the AmigaOS pointer to the file (for advanced programmers). If Result is NULL, the file is not found or can't be opened. It must be always tested, as performing operations on a non created file will cause severe crashes.

## 1.12 readlong

Syntax

```
Number.l = ReadLong()
```

Description

Read one long on the current opened file.

## 1.13 readstring

Syntax

```
Text.s = ReadString()
```

Description

Read one string on the current opened file.

---

## 1.14 readword

Syntax

```
Number.w = ReadWord()
```

Description

Read one word on the current opened file.

## 1.15 usefile

Syntax

```
UseFile(#File)
```

Description

It change the current used file to the given one.

## 1.16 writebyte

Syntax

```
WriteByte(Number)
```

Description

Write a byte number inside the current file. File must be opened with write feature (ie: not with ReadFile()).

## 1.17 writelong

Syntax

```
WriteLong(Number)
```

Description

Write a long number inside the current file. File must be opened with write feature (ie: not with ReadFile()).

## 1.18 writestring



### Syntax

`WriteString(Text$)`

### Description

Write a string inside the current file. File must be opened with write feature (ie: not with `ReadFile()`).

## 1.19 writeword

### Syntax

`WriteWord(Number)`

### Description

Write a word number inside the current file. File must be opened with write feature (ie: not with `ReadFile()`).

---