# File handling

Algorithms and programming (Paper 2) · revision notes

File handling lets a program store data permanently and read it back later. You need to open, read, write and close files, and create new ones, using the ERL commands. Expect short code questions and longer programs that read every line of a file.

### Key facts

- newFile("name.txt"): create a new file
- f.readLine(): read the next line (always a string)
- f.endOfFile(): True when there are no more lines
- Read every line: while NOT f.endOfFile() … endwhile
- f = open("name.txt"): open a file
- f.writeLine(data): write a line to the file
- f.close(): close the file when finished
- Files keep data permanently; variables are lost when the program ends

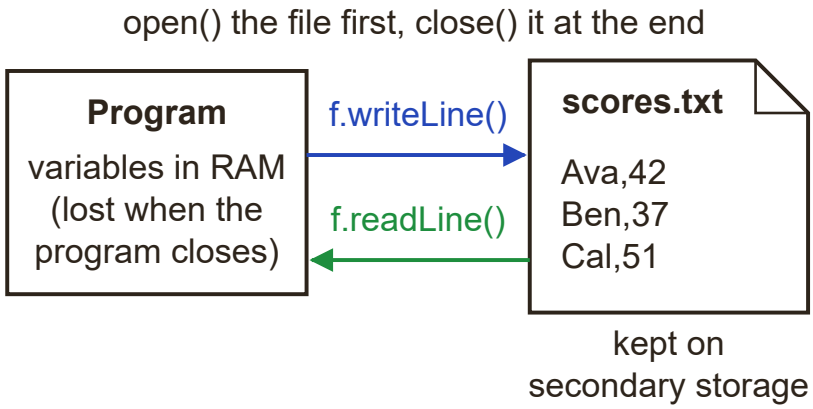
## Notes

### Why use files

- Variables are stored in RAM and are lost when the program closes.
- A file is saved on secondary storage, so the data is kept after the program ends and can be read back the next time it runs (for example high scores or settings).
- Files also let data be shared between programs.

## File commands in ERL

- `newFile("scores.txt")`: creates a new, empty file.
- `f = open("scores.txt")`: opens the file, so the program can use it through the variable f.
- `line = f.readLine()`: reads the next line from the file, as a string.
- `f.writeLine("Ava,42")`: writes a line of data to the file.
- `f.endOfFile()`: returns True when there are no more lines to read.
- `f.close()`: closes the file. Always close a file when you have finished with it, so the data is saved and the file can be used by other programs.




Writing to and reading from a file

## Reading every line

- Use a while loop so it works for any number of lines, even an empty file:

```
f = open("names.txt")
while NOT f.endOfFile()
   name = f.readLine()
   print(name)
endwhile
f.close()
```

- Each line is read as a string, so cast it (for example with int()) before doing arithmetic.

## Writing to a new file

- Create the file, open it, write each line, then close it:

```
newFile("log.txt")
f = open("log.txt")
f.writeLine("Game started")
f.close()
```

## How to answer each type of question

### Explain why data is stored in a file

2 marks **Grade 4**

1. Say what happens to variables when the program closes.
2. Say what the file allows: the data is kept and can be loaded next time.

**Example.** A game stores its high scores in a text file. Explain why the scores are stored in a file rather than only in variables. [2]

**Model answer.** Data stored in variables is lost when the program closes (1). A file keeps the scores permanently, so they can be read back the next time the game is played (1).

### Write code to read a file

4 to 5 marks **Grade 7**

1. Open the file.
2. Loop until the end of the file, reading one line each time.
3. Cast each line if it is a number and process it (total, count, compare).
4. Close the file, then output the result.

**Example.** The file `scores.txt` holds one whole-number score on each line. Write a program that reads every score and outputs the total. [5]

**Model answer.**

```
f = open("scores.txt")
total = 0
while NOT f.endOfFile()
   score = int(f.readLine())
   total = total + score
endwhile
f.close()
print(total)
```

Opens scores.txt (1); loops until the end of the file (1); reads a line each time and casts it to an integer (1); adds each score to a total set to 0 before the loop (1); closes the file and outputs the total (1).

### Write code to write to a file

3 to 4 marks **Grade 5**

1. Create the file with newFile() if it is new, then open it.
2. Write each item with writeLine(), inside a loop if there are several.
3. Close the file at the end.

**Example.** Write code that asks the user to enter 5 names and stores them in a new file called `team.txt`, one name per line. [4]

**Model answer.**

```
newFile("team.txt")
f = open("team.txt")
for i = 1 to 5
   name = input("Enter a name")
   f.writeLine(name)
next i
f.close()
```

Creates and opens team.txt (1); loop that runs 5 times with a name input inside it (1); writes each name to the file with writeLine (1); closes the file after the loop (1).

## Don’t lose marks

- Forgetting to close the file.
- Doing arithmetic on readLine() without casting it.
- Using a for loop with a guessed number of lines instead of checking endOfFile().
- Calling close() inside the loop, so the file is closed after the first line.
- Opening the file inside the loop, which starts reading from the beginning again each time.

## More tips

### Memory tricks

- File routine: Open, Read or Write, Close (O R/W C).
- Anything read from a file is a string, just like input().
- Unknown number of lines means a while NOT endOfFile() loop; never guess a fixed count.

### Exam technique

- Use the file name exactly as given in the question, in quotation marks.
- Every file answer should include open and close: they are easy marks.
- Put the output after the loop and after closing the file, unless the question asks for each line to be output.

## What each grade needs

What you need to be able to do, from the first marks up to the top grade.

Grade 4

### Explain why programs store data in files

Data in variables is lost when the program ends; a file keeps it for next time.

Grade 5

### Open, read and close a file

Use open(), readLine() and close() in the correct order.

Grade 5

### Create a file and write to it

Use newFile(), open() and writeLine(), then close the file.

Grade 6

### Read every line until end of file

Use a while loop with NOT file.endOfFile().

Grade 7

### Process data read from a file

Cast each line to the right type, then total, count or search it.

---

Free notes, Full packs, practice papers and mock exams at chhetriacademy.com. Written for this site; not taken from past papers. Stuck? 1-to-1 tutoring with a free 30-minute trial lesson.