EOF define symbol and EOF system indicator flag

In C language, we have the constant define symbol EOF which often equals“-1” and the system indicator flag (variable) EOF attached to every file stream variable.

You can print the value of the constant define symbol EOF using the statement

printf("%d",EOF);

As shown in slides 27-28 of lecture 9, both functions fprintf and fscanf return the constant value EOF on a failure. We use hence this information to determine if any of the two actions failed.

In parallel, two system indicator flags(variables) EOF and ERROR are attached to every stream variable. The values of these flags can only be retrieved by the function feof and ferror, respectively.

The flag EOF for instance is intialised to zero or the false state, and will be set to true or 1 only when you attempt to read beyond the last value in your data file (the end of file will hence be identified).

In lect 9, we had this code

while(!**feof**(inFile))

fscanf(inFile, "%d",number)

printf("%d ", number);

}

With file data content 1 2 3

The function feof returns the EOF system indicator flag and not the EOF define symbol constant.

At the start, the flag is false and hence feof(inFile) is false, and thus “!**feof**(inFile)” is true.

When we first read 3, the EOF flag remains still false; thus the condition of the while remains true; the loop body is hence re-executed. Then, fscanf returns the constant value EOF as there is no more data to read from the file, the variable “number” is not assigned a new value, the system indicator EOF is set to true., the printf state prints the current value of number from the previous iteration, i.e. 3. In the next iteration of the loop, feof(inFile) returns true as the EOF flag is now true; subsequently“!feof(infile) becomes false; the loop is then exited.

That’s why I suggested to always check feof after scanf and not before.

The ERROR system indicator flag reports other failures, such hardware failure.

You would have noticed in lect 9 that we used the two statements

if(!feof(inFile))

if ferror(inFile)

each time we wanted to **read the entire file data**. This was done to ascertain that the system indicator EOF is set; if it were not “!feof(inFile)” would have been true indicating that not all the file data was read after exiting the while loop

The statement “if ferror(inFile)” allows to check if any other failure, such hardware failure, happened during the processing of the file