File Processing Functions

Objectives

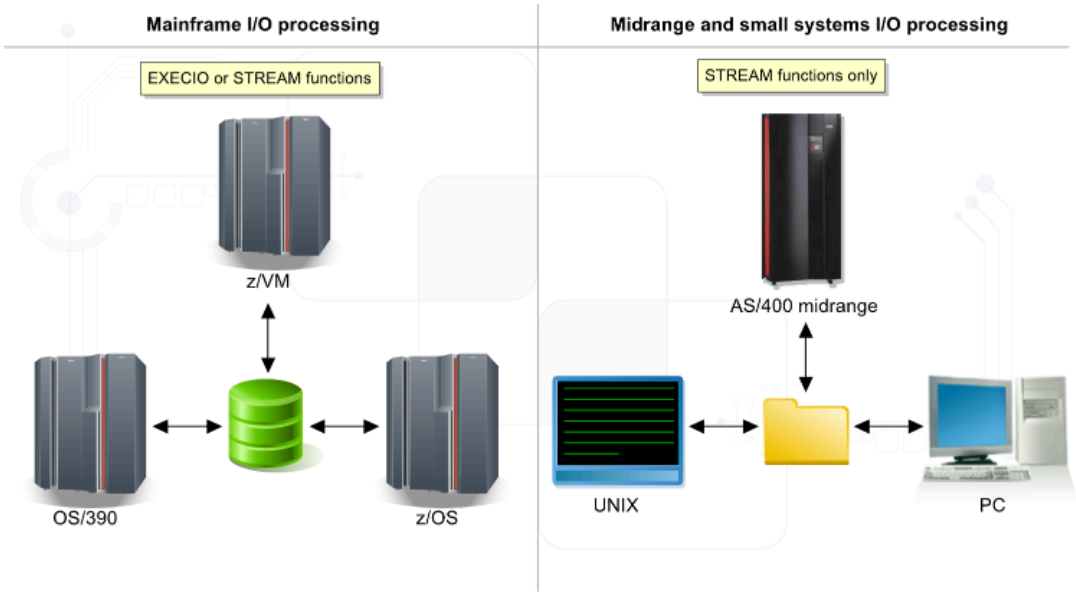
File Processing Functions

In this module, you will discover the built-in file processing functions in the REXX programming language.

You will also be introduced to the concept of streaming and the methods of performing character and line input/output (I/O) from files.

After completing this module, you will be able to:

- Identify the Stream I/O Facility and Functions
- Recognize Character-Oriented I/O
- Recognize Line-Oriented I/O



In earlier versions of mainframe systems, the EXECIO command was used by REXX to read from, and write to, files. This REXX command was unfortunately not compatible with files in other operating systems and instead, a common set of "streaming" I/O functions was developed for REXX on those systems. In support of this development, IBM now provides a Stream I/O for TSO/E REXX Function, which can be loaded to support the stream I/O concept on z/OS systems.

Some differences exist in the implementation of these functions in different environments. For platform-specific information, refer to the relevant manuals. Although they work in much the same way on all platforms, you will focus on the implementation of these functions in the PC environment.

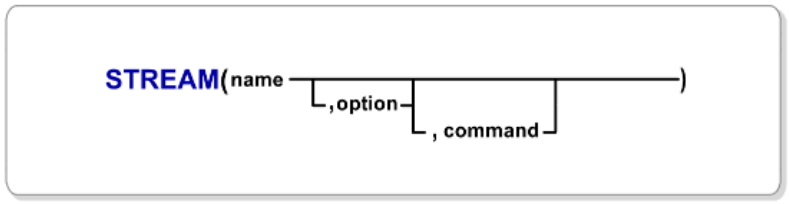
Standard streaming I/O functions	
STREAM()	The STREAM function is used to name, open, close, or query a stream.
CHARIN()	The CHARIN function reads characters from a specific stream.
CHAROUT()	The CHAROUT function writes characters to a specific stream.
CHARS()	The CHARS function determines if there are any more characters to be read in the stream.
LINEIN()	The LINEIN function reads lines or records from a specific stream.
LINEOUT()	The LINEOUT function writes lines or records to a specific stream.
LINES()	The LINES function determines if there are any more lines to be read in the stream.

A stream is a sequence of characters that come from a file, a program, or the keyboard. File processing is normally regarded as a persistent stream where the position in the stream can be changed. Data from a keyboard is considered to be transient because the data did not previously exist and must be processed as it is typed.

Streams can also be processed by character, or by line or record. Different types of files on various different platforms indicate the end of a record or line in different ways. Sometimes it is better to process I/O by character so that REXX can recognize the end of a logical line. Functions are available to enable I/O processing in either mode.

Listed above are the standard streaming I/O functions that are available on most platforms.

The STREAM function



The STREAM function provides several functions for file processing, such as naming, opening, and closing files, and determining if a stream already exists. The STREAM function can often be bypassed by directly referring to the file from the CHARxxx or LINExxx functions. Error checking can be simplified by performing the STREAM function first.

The STREAM() function returns either `ERROR` or information depending on the option chosen. In the z/OS environment, STREAM() can also be used to allocate a data set.

Mouse-over each parameter for a brief description.

The STREAM function commands

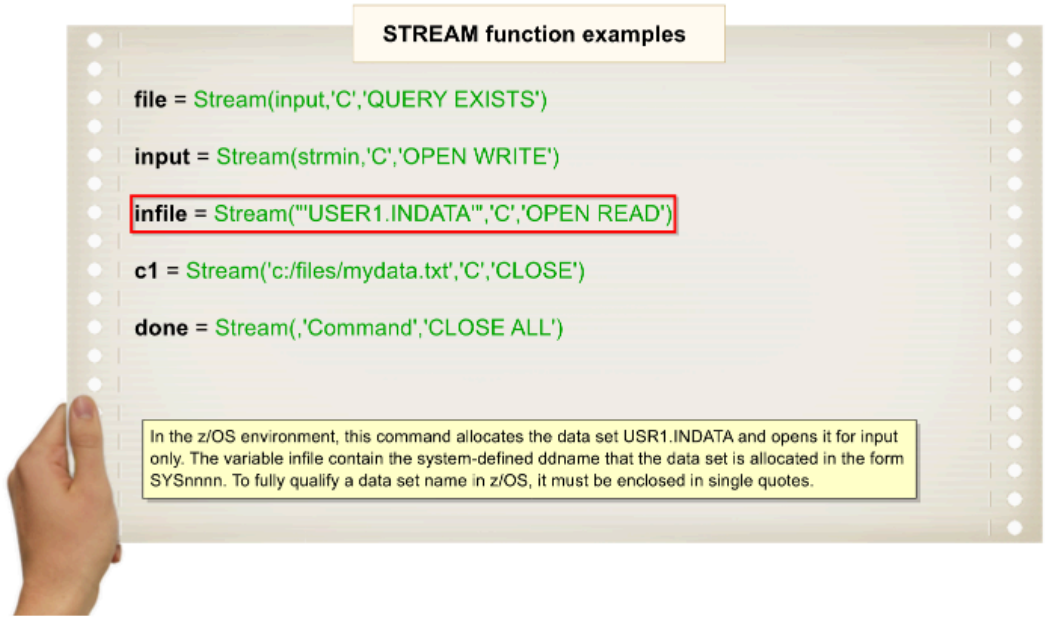
```
datafile = STREAM ( "MYFILE.TXT", "C", command )
```

OPEN	Opens the stream for input operations.
OPEN READ	Same as OPEN.
OPEN WRITE	Opens the stream for output operations.
CLOSE	Closes the named stream.
CLOSE ALL	Closes all streams opened in this program.
QUERY EXISTS	Queries existence of named stream and returns the data set or file name.
QUERY REFDATE	Queries date when named stream was last referenced - z/OS.
QUERY TIMESTAMP	Queries date when named stream was last referenced - PC only.
QUERY SERVICELEVEL	Queries the service level of the stream I/O function package - z/OS only.

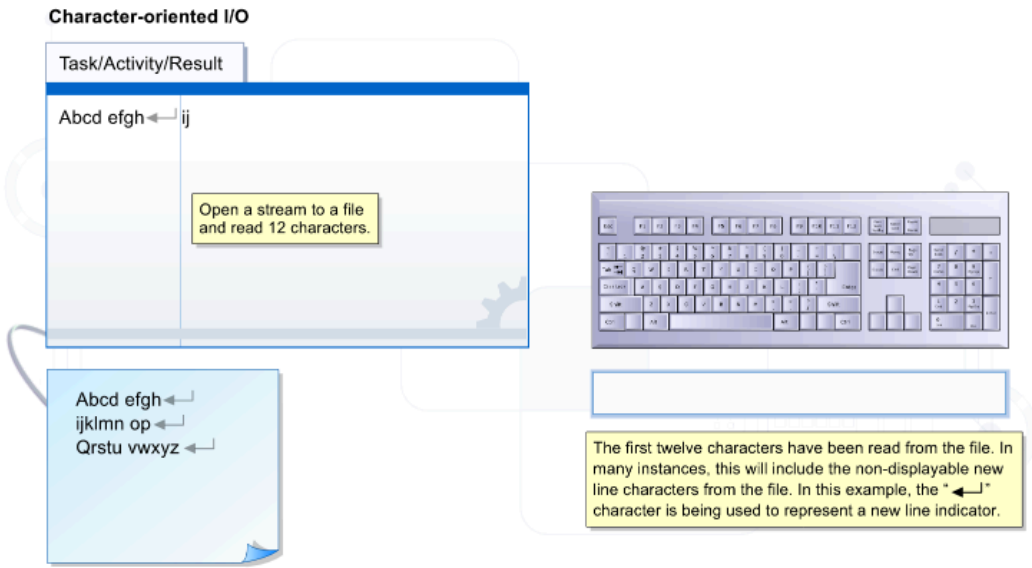
When the STREAM() function is used with the "C" option, several commands can be specified to perform processing of a stream. Most programming languages require files and streams to be opened before reading and writing, and closed after completion. Generally, only one type of action, read or write, can be performed on a stream during a single session before closing and reopening. Extreme care should be taken when reading and writing to the same stream in a single session because the results can be unpredictable.

The actual syntax of these commands varies depending on the platform. Check the appropriate REXX manual of the executing environment for other commands that can be used.

Click Play to see some of the STREAM function commands.



Mouse-over these examples of the STREAM function for descriptions of their results.

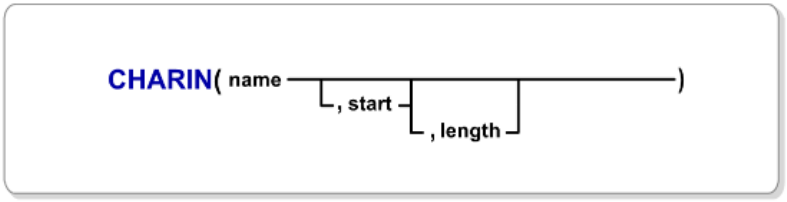


Character-oriented I/O enables data to be read or written to or from a stream character by character, or by sets of characters. For persistent streams, such as files, the start character position can be specified and moved each time a new operation is performed; however, for transient streams such as the keyboard, the operation always starts from the next character.

Note that some platforms and file types use special characters to indicate the end of a line or record. Depending on the implementation, this character may also be passed to the program as a readable character. It would then be up to the REXX code to identify it as the end of line. Processing of character-oriented I/O varies across platforms.

Click **Play** to see how character processing can be used.

The CHARIN function

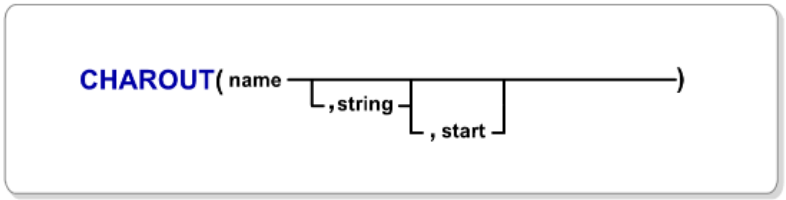


The CHARIN function is used to read characters one at a time or in groups from the keyboard, which is the default, or files. While the fully-qualified name of the file can be coded and will be automatically allocated and opened, it is standard practice to use a STREAM function first to name the stream and open it, and refer to the stream in the CHARIN function.

The result of the CHARIN function will be the characters read. If end-of-file is encountered, the result can differ depending on the system and implementation, but it will usually be null or cause a NOTREADY condition.

Mouse-over the CHARIN function syntax for a description of its parameters.

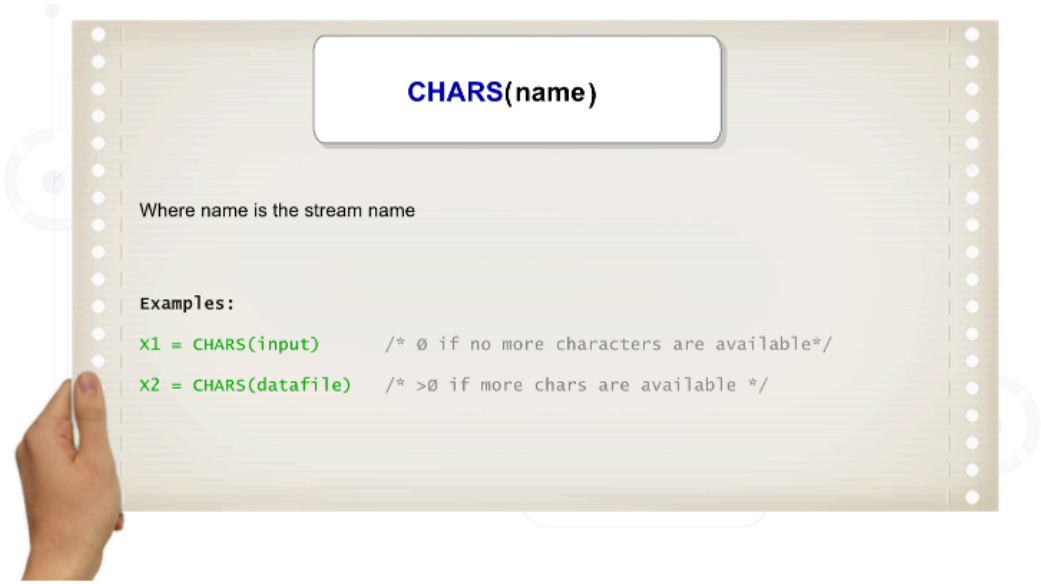
The CHAROUT function



The CHAROUT function is used to write characters to the terminal, which is the default, or a file. As with the CHARIN function, CHAROUT() usually refers to a stream that is created or opened for output by the STREAM function.

When writing to a file, the CHAROUT function may overwrite, append, or insert characters, depending on how the stream has been opened and the platform on which the program is running. Check the relevant manual for the available options.

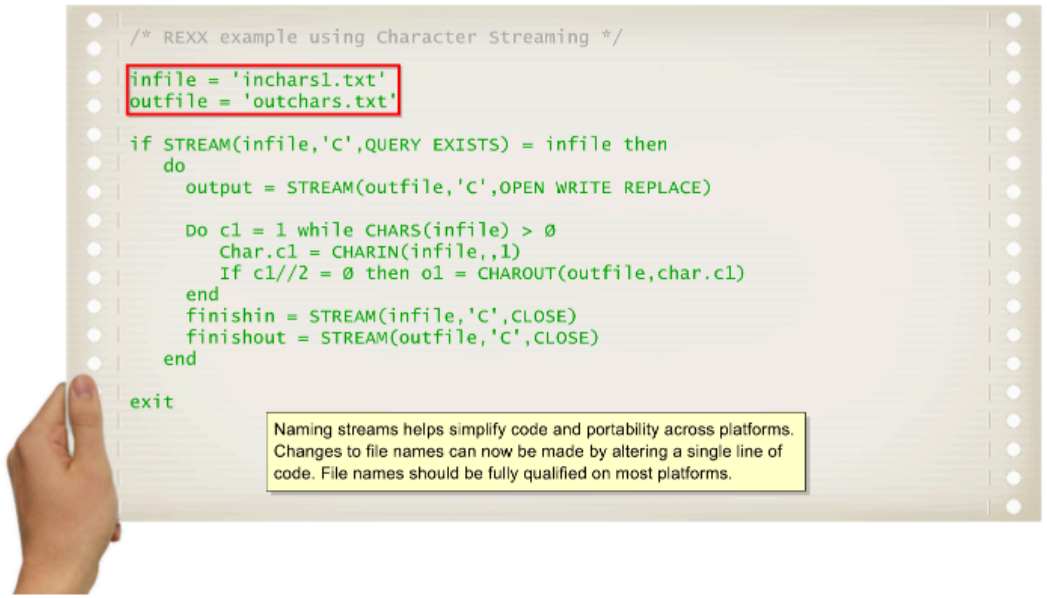
Mouse-over the CHAROUT syntax for a description of its parameters.



The CHARS function is used to determine if more characters are available in the stream. If no more characters are available, the value of "0" is returned. The returned value for transient streams is always "1" if more characters are available.

The value returned for persistent streams may be the number of characters remaining in the file, depending on the platform, the file, and the implementation of the stream functions.

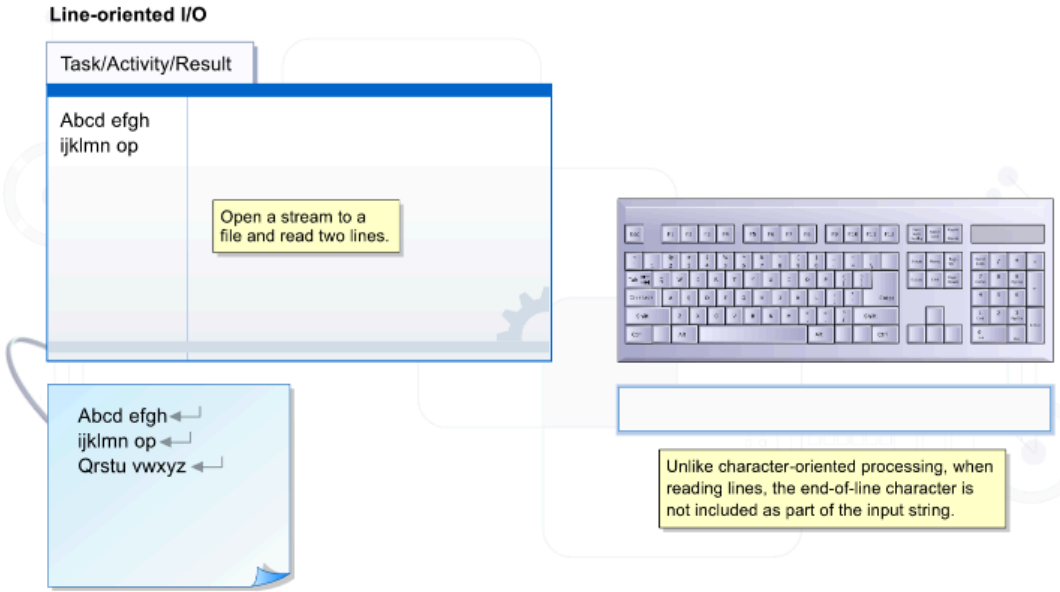
The CHARS function has the stream name as the only parameter.



This example shows how character-oriented I/O works in the PC environment.

Remember that the use and results of these functions differ for each platform and each implementation. Always consult the relevant manuals before using them.

Mouse-over each section of the code for a description.



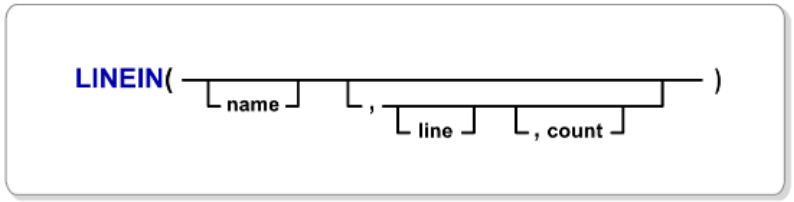
Line-streaming functions enable data to be read from a file or keyboard line by line, or record by record where applicable.

Note that some files use ASCII characters to indicate end-of-line so the LINE functions and CHAR functions occasionally produce different results for the same file.

When reading lines from the keyboard, the Enter key always indicates end-of-line and end-of-line characters are not returned as part of the line.

Click Play to see how line processing can be used.

The LINEIN function

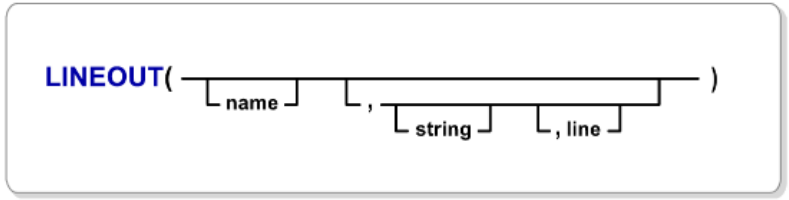


Like the CHARIN function, the LINEIN function opens and extracts a line or record at a time from a stream. Most implementations of streaming functions handle end-of-line and end-of-file processing adequately, but occasionally there are problems.

Sometimes there is confusion about whether an end-of-file condition should be treated as end-of-line, and the LINEIN function does not return the last line. This is entirely up to the implementer of the function on the system. Check the relevant documentation to discover the consequences of end-of-file conditions.

Mouse-over the LINEIN function syntax for a description of its parameters.

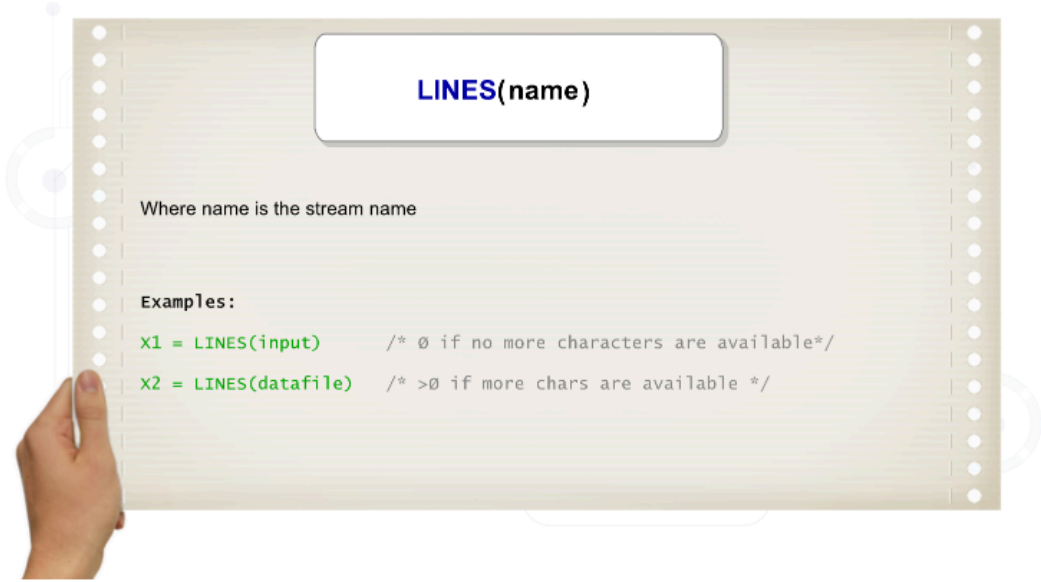
The LINEOUT function



The LINEOUT function enables complete lines to be written to a file or the terminal display.

When writing to a file, the LINEOUT function may overwrite, append, or insert lines, depending on how the stream has been opened and the platform on which the program is running. Check the relevant manual for the available options.

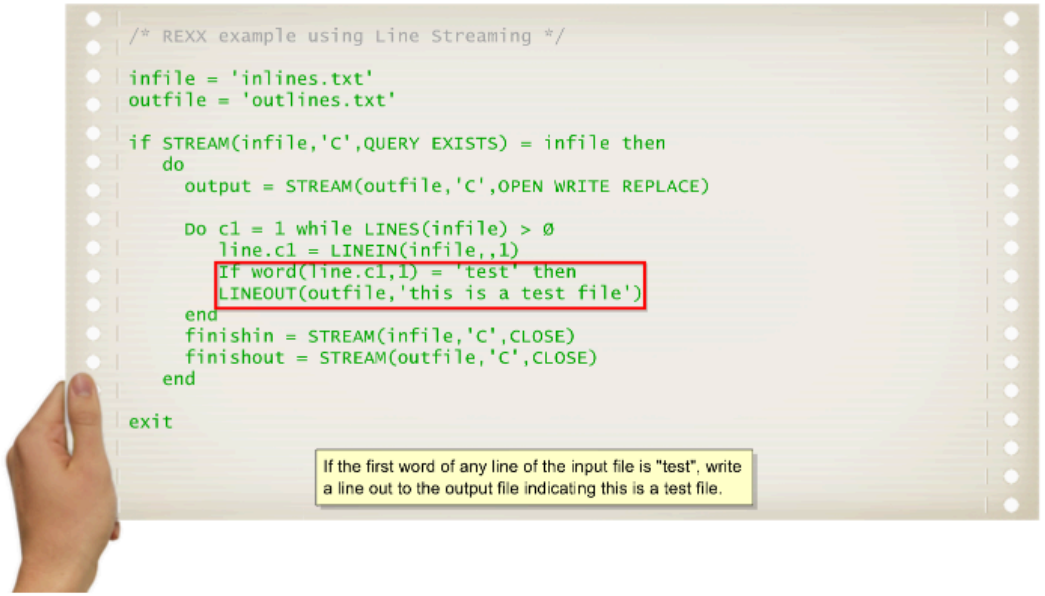
Mouse-over the LINEOUT syntax for a description of its parameters.



The LINES function is used to determine if more lines are available in the file. If no more lines are available, the value of "0" will be returned. The returned value for transient streams is always "1" if more lines are available.

The value returned for persistent streams may be the number of lines remaining in the file, depending on the platform, the file, and the implementation of the stream functions.

The LINES function has the stream name as the only parameter.



This example shows how line-oriented I/O works in the PC environment.

Remember that the use and results of these functions differ for each platform and each implementation. Always consult the relevant manuals before using them.

Mouse-over the code for a description of each section.