Files

Dr. V.Sireesha

Associate Professor

Dept. of CSE

File

- A **File** is a collection of records that can be accessed through the set of library functions. Record is nothing but collection of fields of related data items. These files are stored on the disk and can be accessed through file-handling functions provided by the C-standard library.
- Eg: Here a file called " STUDENT.DAT" which is having 5 students' records. And every record is collection of 4 fields.

RollNo	Name	Total Marks	Average
101	Pavan	450	70
102	Laxman	400	66
103	Dinesh	433	68
104	Simha	500	80
105	Sun	540	82

Types of I/O

- There are numerous library functions available for I/O. these can be classified into two broad categories:
- a) console I/O b) file I/O

- **Console I/O**:Functions to receive input from keyboard and write output to VDU (monitor). The screen and keyboard are called a console. Console I/O functions can be further classified into two categories – **formatted console I/O** and **unformatted console I/O**. The basic difference between them is that the formatted functions allow the input and output to be formatted as per requirements. The different console I/O functions are as follows:

console I/O

- **Formatted functions**: The input function scanf( ) and the output function printf( ) are called formatted console I/O functions. By using the format specifiers and escape sequence characters we can use these functions to read/print required format of information.

- **Unformatted functions:-** The input functions like getch( ), getche( ), getchar( ) and gets( ) are called unformatted input functions and putch( ),putchar( ), and puts( ) are unformatted output functions. Each and every function is having its own syntax and meaning.

FILE I/O

- **File I/O:** Sometimes it is necessary to store the data in a manner that can be later retrieved and displayed either in a part or in whole. This medium is usually a “file” on the disk. File I/O can be handled by using different functions.
- **Formatted functions**: The file input function fscanf( ) and the file output function fprintf( ) are called formatted file I/O functions.
- **Unformatted functions:** The input functions like getc( ), getw( ), and fread( ) are called unformatted file input functions and putc( ), putw( ), and fwrite( ) functions are unformatted file output functions. Each and every function is having its own syntax and meaning.

File streams

- File streams:- Stream is either reading or writing of data. The streams are designed to allow the user to access the files efficiently. A stream is a file or physical device like key board, printer, monitor, etc., The FILE object uses these devices. When a C program is started, the operating system is responsible for opening three streams: standard input stream (**stdin**), standard output stream (**stdout**), standard error(**stderr**).Normally the stdin is connected to the keyboard, the stdout and stderr are connected to the monitor.
- **Text files and Binary files:** C uses two types of files, text files and binary files.

Text files

- **Text files:-** Text file is a file consists of a sequence of characters divided into lines with each line terminated by a new line('\n'). A text file is writing using text stream. We can read and write text files using different input/output functions. Formatted input/output (scanf/printf), character input/output (getchar/putchar), and string input/output (gets/puts) functions. And these functions can work with only text files. A text file contains only textual information like alphabets, digits, and special symbols. In actuality the ASCII codes of these characters are stored in text files.
- Ex: A good example of text file is any C program file.

Binary files

- **Binary files:** A binary file is a collection of data stored in the internal format of the computer. Unlike text files, the data do not need to be reformatted as they are read and write rather, the data is stored in the file in the same format that they are stored in memory. Binary files are read and write using binary streams known as block input/output functions. Simply a binary file is merely a collection of bytes. This collection might be a compiled version of a C program or music data stored in a wave file or a picture stored in a graphic file.

Differences between Text and Binary files

The major characteristics of text files are:

- All data in a text file are human-readable graphic characters.
- Each line of data ends with a newline character.
- There is a special character called end-of-file(EOF) marker at the of the file.

The major characteristics of binary files are:

- Data is stored in the same format as that is stored in memory.
- There are no lines or new line characters.
- There is an end-of-file marker.

Basic operations on files

The file consists of large, amount of data, which can be read or modified depending on the requirement. The basic operations that can be performed on files are:

a) **Opening a file** : A file has to be opened before to read and write operations. This can be achieved through fopen() function.

syntax: FILE *fp;

fp=fopen(filename,mode);

Here is a pointer pointing to the file “filename”, which can be opened in specified mode.

The fopen() performs three tasks:

i)It searches the disk for opening the file .

ii)If file exists, it opens that file,if the file is not existing this function returns NULL in case of read mode. In case of write mode,it creates a new file and in case of append mode it opens that file for updating.

Basic operations on files

iii)It locates a file pointer pointing to the first character of the file.

Ex:
```
FILE    *fp;
fp = fopen(“sample.txt”,”r”);
if (fp==NULL)
printf(“file doesa not exist”);
else
 {
          …
          …
          …
  }
```

b) Reading/writing a file: Once the file is opened, the associated file pointer points to the starting of the file. If the file is opened in ***writing mode***, then we can write information by using different functions like:

putc ( ): Putting a character in to the file. It works with only character data type. One character at a time can write into a file.

Ex: char ch =’a’;   putc (ch, fp); **putw ( ):** putting or writing of an integer value to a file.

Ex:
```
int  x = 5;
putw(x,fp);
```

Basic operations on files

fprintf(): writing a group of values into a file.

Syntax: fprintf (file pointer, ”control string”, list of arguments);

Ex: fprintf (fp, ”%s %d %d ”, name, rno, marks);

If the file is opened in ***reading mode***, then we can read information from the file, i.e. using some functions like:

getc ( ): getting a character from the file, or reading the file information character by character at a time, upto the end of the file by using this function.

Ex:
```
char ch;
ch = getc (fp);
```

getw ( ): getting or reading integer value from a file.

Ex:
```
int x;
x = getw (fp);
```

fscanf ( ): This function is used to read the information from a file record-wise. It is used to read more values at a time.

Ex: fscanf(fp,”%s%d%d”, name, &no, &marks);

Among all the above different file i/o functions, fscanf() and fprintf() are called as formatted i/o functions. Printf and scanf are also called as formatted i/o functions.

NOTE: some compilers refer that:

putc() is same as fputc()

getc() is same as fgetc().

File Operations

c) **closing a file:** After reading/writing a file ,it is needed to close that file. fclose() is used to close a file. It closes only one file at a time.

fclose (fp):-close a file which is pointed by fp.

fcloseall ( ):-close all opened files at a time.

Different modes to open a file

Different modes to open a file: The tasks performed by fopen() function when a file is opened in each of these modes are as follows:

1.**"r" (Read) mode**: opens the file that already exists for reading only. If the file doesn't exist it returns NULL.

2.**"w"(write) mode**: File is opened for writing, if the file exists its contents are overwritten and if the file doesn't exists, then a new file is created. It returns NULL if it is unable to open the file.

f1=fopen("sample.txt", "w");

3.**"a"(append) mode**: searches for the file, if it exists then appending new contents at the end of the file.If a file doesn't exist then a file with a specified name is created and ready to get append or add.

f1 = fopen ("sample.txt", "a");

4."**r+" (Read & write mode):** This is for both reading and writing the data. If the file doesn't exist then it returns NULL.

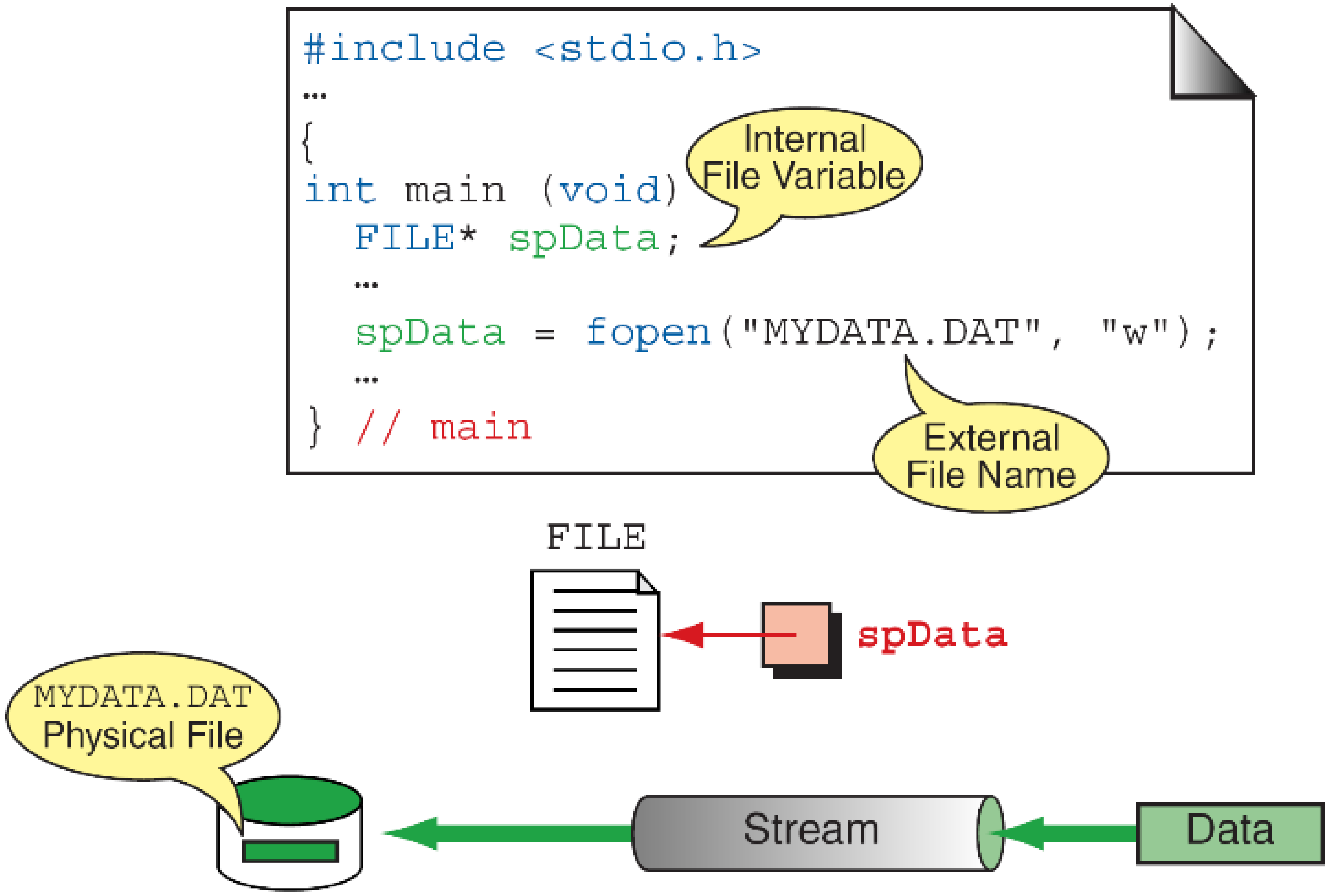
5."**a+" (Append & Read) mode**: The file can be read as well as data can be appended.

The two modes of binary files are:

"rb" (read binary) mode: Binary file is opened for reading only.

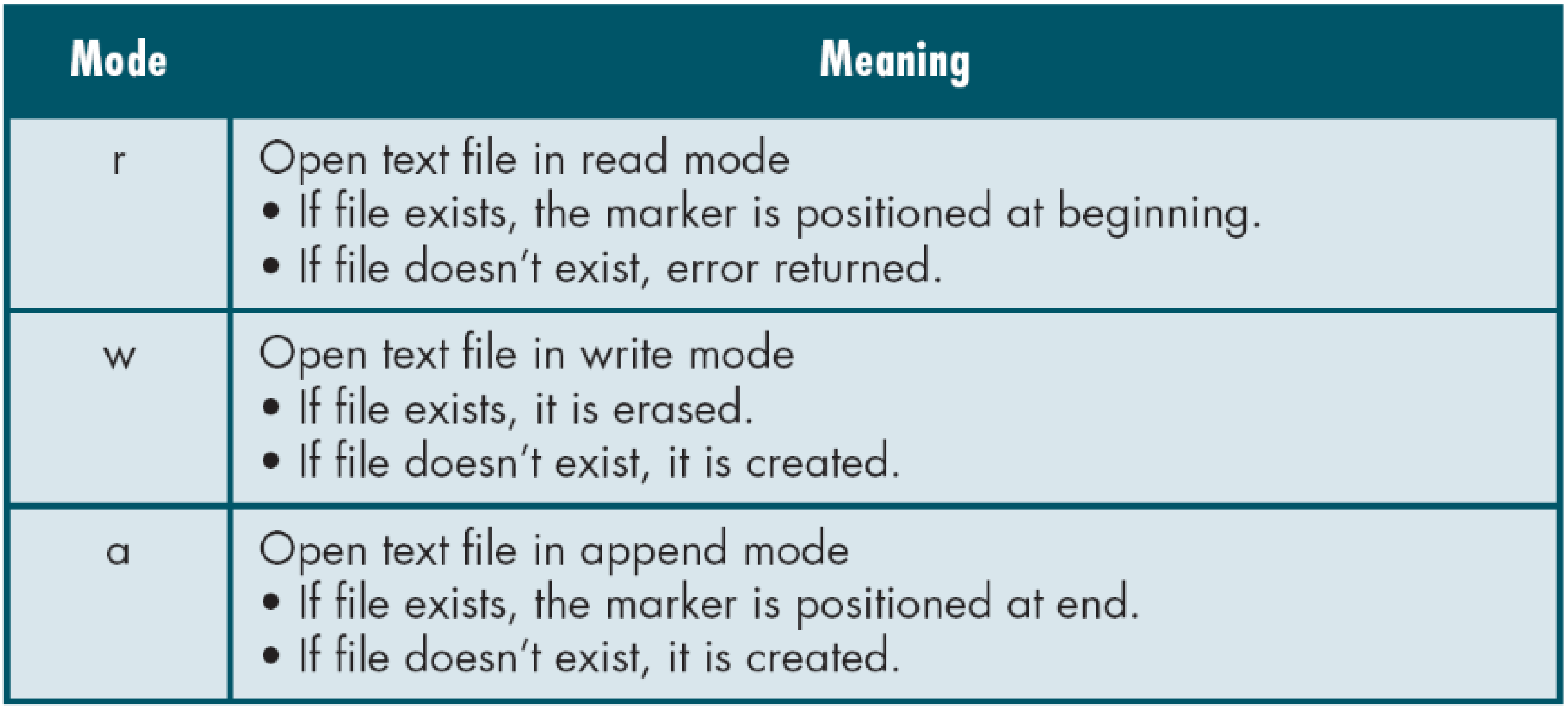
"wb" (write binary) mode: Binary file opened for writing only.

NOTE: While opening the file in text mode we can use either "r" or "rt"/ "w" or "wt", but since text mode is the default mode we usually drop the "t".



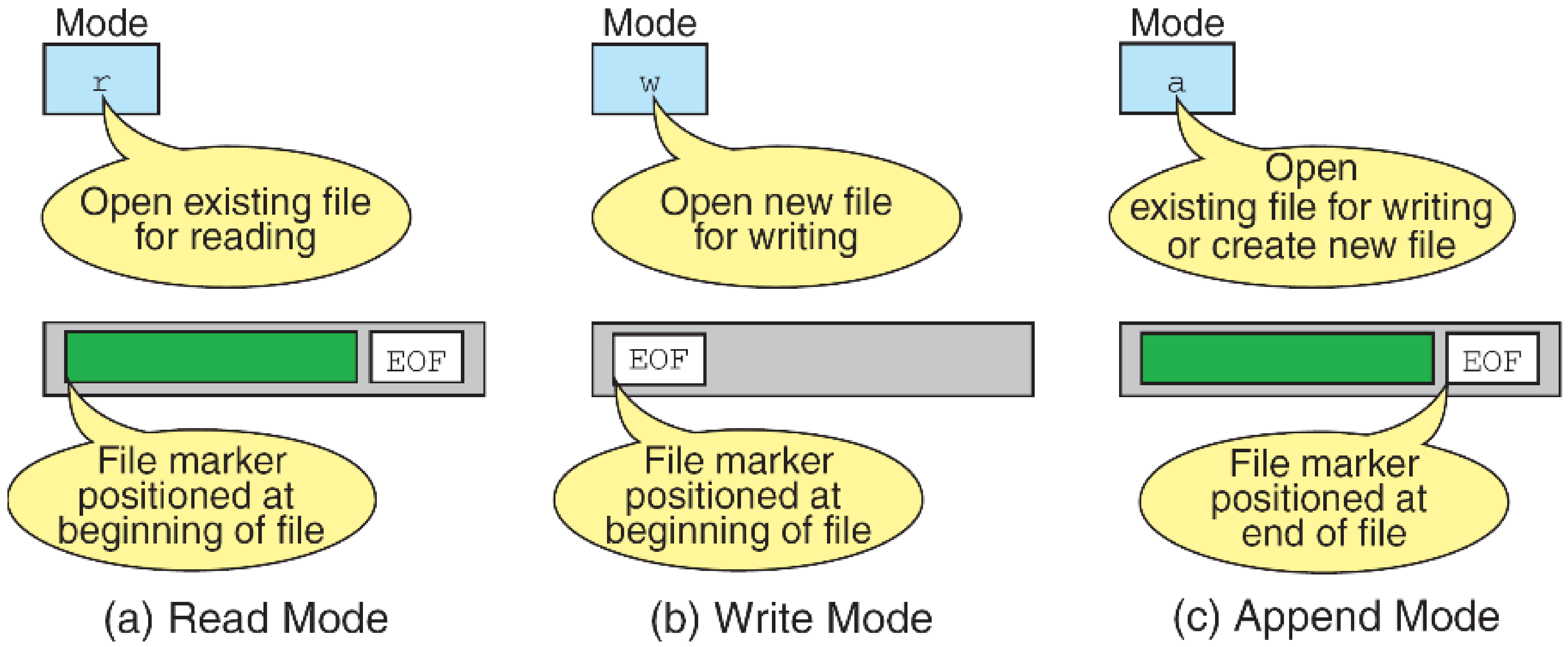
```
#include <stdio.h>
...
{
int main (void)
   FILE* spData;
   ...
   spData = fopen("MYDATA.DAT", "w");
   ...
} // main
```

File Open Results

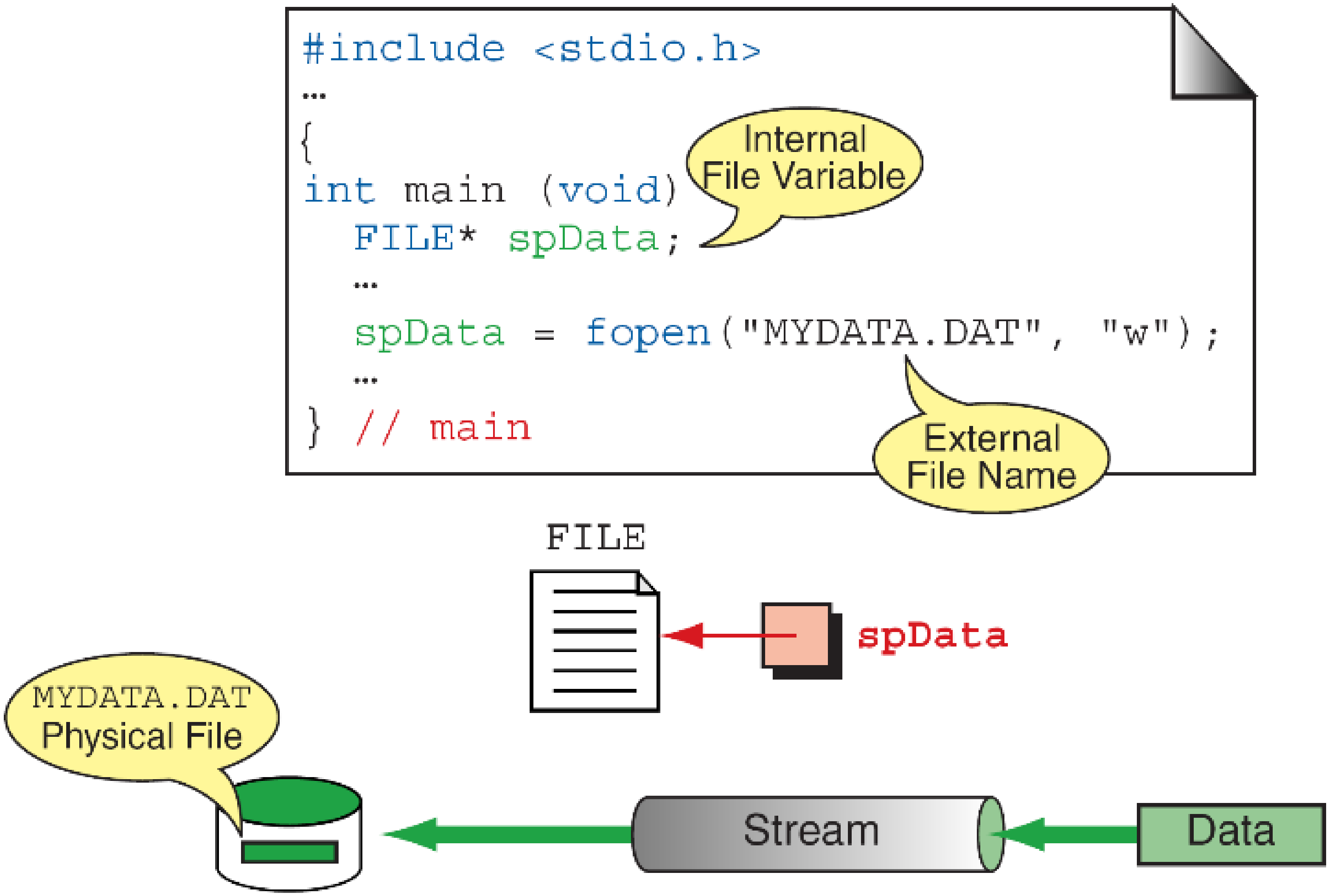


Mode	Meaning
r	Open text file in read mode • If file exists, the marker is positioned at beginning. • If file doesn't exist, error returned.
w	Open text file in write mode • If file exists, it is erased. • If file doesn't exist, it is created.
a	Open text file in append mode • If file exists, the marker is positioned at end. • If file doesn't exist, it is created.

Table 7-1 Text File Modes

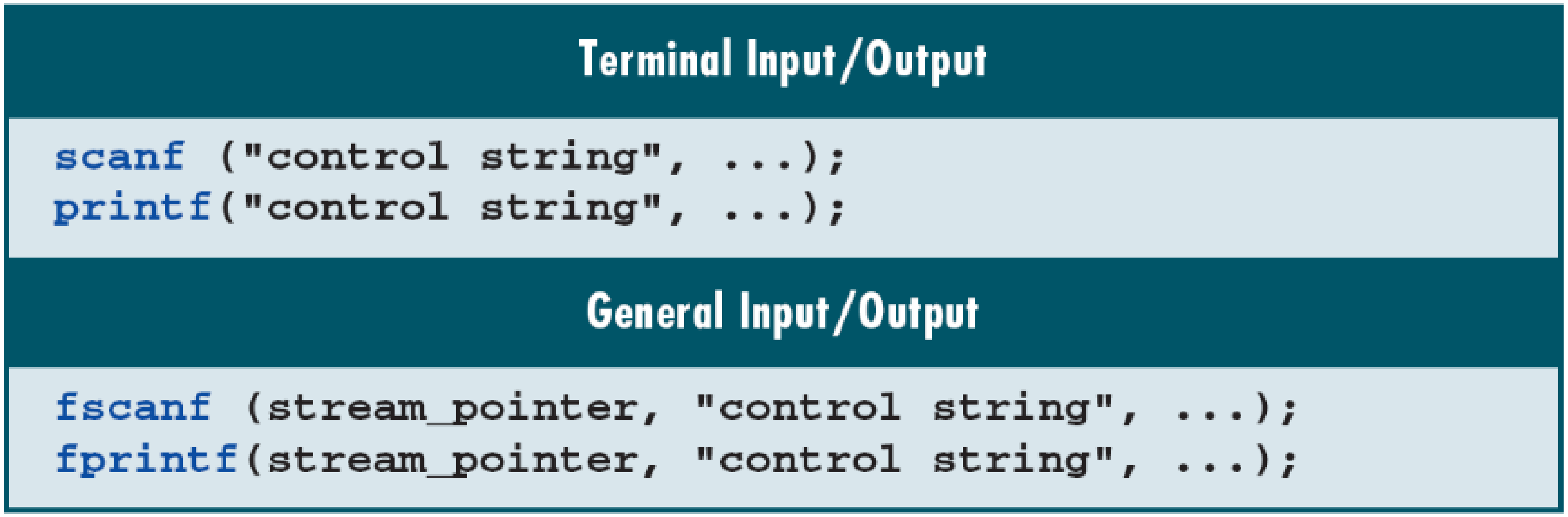

Mode
r
Open existing file for reading
EOF
File marker positioned at beginning of file
(a) Read Mode
Mode
w
Open new file for writing
EOF
File marker positioned at beginning of file
(b) Write Mode
Mode
a
Open existing file for writing or create new file
EOF
File marker positioned at end of file
(c) Append Mode

File-Opening Modes



```
#include <stdio.h>
...
{
int main (void)
   FILE* spData;
   ...
   spData = fopen("MYDATA.DAT", "w");
   ...
} // main
```

File Open Results



Terminal Input/Output

```
scanf ("control string", ...);
printf("control string", ...);
```

General Input/Output

```
fscanf (stream_pointer, "control string", ...);
fprintf(stream_pointer, "control string", ...);
```

Table 7-2 Formatting Functions

Note

scanf reads from *stdin*;
fscanf reads from a user-specified stream.