# std.stdio.File struct

This lesson teaches how to use the file struct from the std.stdio module.

WE'LL COVER THE FOLLOWING



## The `std.file` module #

The std.file module contains functions and types that are useful when working with the contents of directories. For example, `exists` can be used to determine whether a file or a directory exists on the file system. It takes a string as an argument, which specifies the name of the file.

```
import std.file; // ...
  if (exists(fileName)) {
      // there is a file or directory under that name
  } else {
      // no file or directory under that name
  }
```

Below is the code implementing the concepts discussed so far:

```
import std.stdio;
import std.string;
import std.file;
void main() {

    if (exists("student_records.txt")) {

        File file = File("student_records.txt", "r");

        while (!file.eof()){
            string line = strip(file.readln());
```

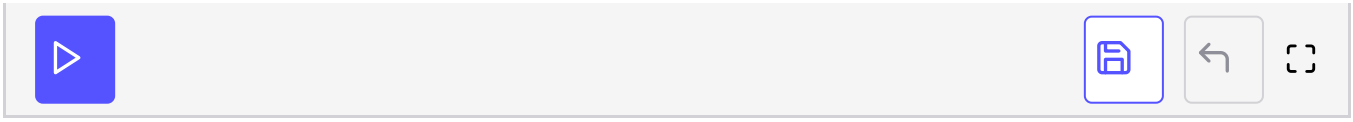
```
            writeln(line);
        }


        file.close();

        file.open("student_records1.txt", "w");
        file.writeln("Name  : ", "Bob");
        file.close();

        file.open("student_records1.txt", "r");
        while (!file.eof()) {
            string line = strip(file.readln());
            writeln(line);
        }
    }
    else {
        writeln("No such file");
    }

}
```



File handling in D

## std.stdio.File struct #

The `File` struct is included in the std.stdio module. We can create a variable of type `File` and specify the name of the file and desired access permissions or mode for this `File` type variable. It uses the same access mode parameters as used by the `fopen()` function of the C programming language:

| Mode | Definition |
|---|---|
| r | **read** access<br>the file is opened to be read from the beginning |
| r+ | **read and write** access<br>the file is opened to be read from and written at the beginning |
| w | **write** access<br>if the file does not exist, it is created as empty<br>if the file already exists, its contents are cleared |
| w+ | **read and write** access<br>if the file does not exist, it is created as empty<br>if the file already exists, its contents are cleared |
| a | **append** access<br>if the file does not exist, it is created as empty<br>if the file already exists, its contents are preserved and it is opened to be written at the end |
| a+ | **read and append** access<br>if the file does not exist, it is created as empty<br>if the file already exists, its contents are preserved and the file is opened to be read from the beginning and |

| | |
|---|---|
| | written at the end |

fopen mode characters

In some cases, a `b` can be added to the mode string as in `rb`. This may have an effect on platforms that support the binary mode, but it is ignored on all POSIX (Portable Operating System Interface) systems.

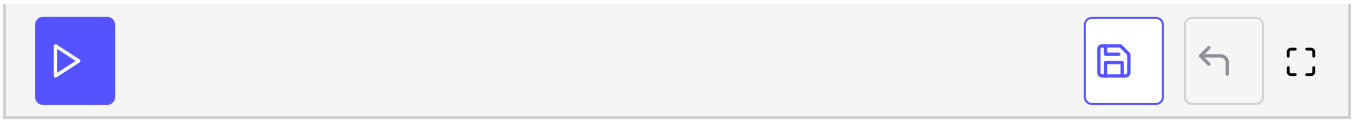
## Writing to a file

The file must have been opened in one of the write modes first:

```
import std.stdio;

void main() {
    File file = File("student_records", "w+");

    file.writeln("Name   : ", "Zafer");
    file.writeln("Number: ", 123);
    file.writeln("Class : ", "1A");
}
```



Writing to a file

As you remember from the strings lesson, the type of literals like `student_records` is string, consisting of immutable characters. For this reason, it is not possible to use `char[]` type mutable text to specify the file name. When needed, we can call the `.idup` property of the mutable `char[]` type to get an immutable copy. This can then be used to create the File type object.

The program above creates or overwrites the contents of a file named `student_records` in the directory that it has been created under (in the program's working directory).

> **Note:** File names can contain any character that is legal for that file system. For portability, use of only commonly supported ASCII characters is suggested.

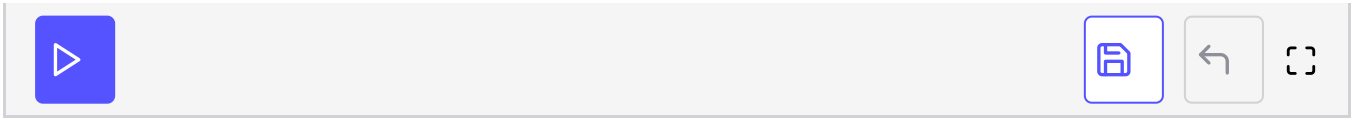
## Reading from a file

To read from a file the file must first have been opened in one of the read modes:

```
import std.stdio;
import std.string;

void main() {
    File file = File("student_records.txt", "r");

    while (!file.eof()) {
        string line = strip(file.readln());
        writeln("read line -> |", line);
    }
}
```



Reading from a file

The program above reads all of the lines of the file named `student_records` and writes those lines to its standard output.

---

In the next lesson, you will find a coding challenge based on file handling.