

Odin contributors
Oct 14, 2026

Reading a file is very easy to do in Odin, but sometimes you just want to read the file line-by-line. Below we will show some approaches for this.

Iterator Approach

Reading the entire file at once and then split line by line with an iterator.

This is the preferred approach in general on most modern machines.

```
package line_by_line

import "core:os"
import "core:strings"

read_file_by_lines_in_whole :: proc(filepath: string) {
    data, err := os.read_entire_file(filepath, context.allocator)
    if err != nil {
        // could not read file
        return
    }
    defer delete(data, context.allocator)

    it := string(data)
    for line in strings.split_lines_iterator(&it) {
        // process line
    }
}
```

Buffered IO Approach

Reading the file a chunk at a time and then copy a string from the buffer.

This is useful for when you have a general io.Reader which cannot read everything at once.

```
import "core:bufio"

read_file_by_lines_with_buffering :: proc(filepath: string) {
    f, ferr := os.open(filepath)
    if ferr != nil {
        // handle error appropriately
        return
    }
    defer os.close(f)

    r: bufio.Reader
    buffer: [1024]byte
    bufio.reader_init_with_buf(&r, {os.stream_from_handle(f)}, buffer[:])
    // NOTE: bufio.reader_init can be used if you want to use a dynamic backing buffer
    defer bufio.reader_destroy(&r)

    for {
        // This will allocate a string because the line might go over the backing
        // buffer and thus need to join things together
        line, err := bufio.reader_read_string(&r, '\n', context.allocator)
        if err != nil {
            break
        }
        defer delete(line, context.allocator)
        line = strings.trim_right(line, "\r")

        // process line
    }
}
```