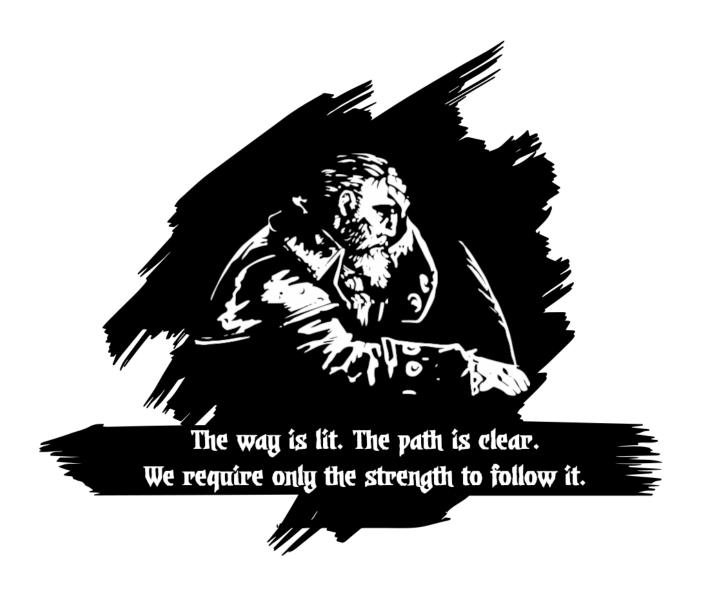


EXERCISES — cat

version #7be580532266ed398481e31366afcc24b1950c2a



Copyright

This document is for internal use at EPITA (website) only.

Copyright © 2022-2023 Assistants <assistants@tickets.assistants.epita.fr>

The use of this document must abide by the following rules:

- ▶ You downloaded it from the assistants' intranet.*
- ▶ This document is strictly personal and must **not** be passed onto someone else.
- ▷ Non-compliance with these rules can lead to severe sanctions.

Contents

1	Introduction	
2	Assignment	_

^{*}https://intra.assistants.epita.fr

File Tree

```
cat/
    Makefile (to submit)
    cat.c (to submit)
```

Makefile

· cat: Produce the cat binary

Authorized functions: You are only allowed to use the following functions

- fclose(3)
- fdopen(3)
- ferror(3)
- fflush(3)
- fgetc(3)
- fopen(3)
- fputc(3)

Authorized headers: You are only allowed to use the functions defined in the following headers

- err.h
- errno.h
- assert.h
- · stddef.h
- · err.h
- string.h

Compilation: Your code must compile with the following flags

• -std=c99 -pedantic -Werror -Wall -Wextra -Wvla

Main function: Required

1 Introduction

The aim of this exercise is to reproduce cat(1) by using the C standard I/O library and its streams. The stdio library provides a simple and efficient buffered stream I/O interface.

2 Assignment

The behavior and the output of your cat program must be identical to cat(1). However, you only have to handle these options:

- -n: number the output lines, starting at 1. Indeed, numbers and outputs are separated by *a tabulation*. You must follow the cat(1) alignment.
- -E: display a dollar sign at the end of each line.
- -: if an argument is a single dash ('-') or if there no argument, cat reads from the standard input.

The error output and return values have to be the same as cat(1) if any error occurs (such as non-existing files).

Be aware that the cat(1)'s man defines explicitly what it does.

Tips

When using stdio functions, if an error occurs, the errno(3) variable is associated with an error number.

While reading a file, we expect you to read it character by character with fgetc(3) and therefore write it with fputc(3).

```
42sh$ ./cat -E test1
This a test from the file 'test1'.$
Blabla blabla blabla...$
42sh$ ./cat -E test2
'test2' is also a test!$
42sh$ echo 'Test from stdin' | ./cat -E -n test1 - test2
1 This a test from the file 'test1'.$
2 Blabla blabla blabla...$
3 Test from stdin$
4 'test2' is also a test!$
```

Tips

To be able to read from the standard input or write to the standard output, you can use stdin(3) and stdout(3) variables. Be aware that when you send an EOF to stdin(3), it will be closed, thus you can reopen it with fdopen(3) and with the file descriptors 0 for stdin and 1 for stdout¹.

Do not forget to flush the standard output at the end of your program.

The way is lit. The path is clear. We require only the strength to follow it.

¹ If you don't like magic numbers, STDIN_FILENO and STDOUT_FILENO expand to these magic values.