0x15. C - File I/O

C Syscall

By: Julien Barbier

Weight: 1

Project will start Oct 31, 2022 6:00 AM, must end by Nov 1, 2022 6:00 AM

was released at Oct 31, 2022 12:00 PM

An auto review will be launched at the deadline

Resources

Read or watch:

- File descriptors (https://alx-intranet.hbtn.io/rltoken/Duva-9Fjyskt39R__Nnazg)
- C Programming in Linux Tutorial #024 - open() read() write() Functions (https://alx-intranet.hbtn.io/rltoken/9Tmu01qEnA9q9khz3gqzJQ)

man or help:

- `open`
- `close`
- `read`
- `write`
- `dprintf`

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (https://alx-intranet.hbtn.io/rltoken/kQg2-u-cAYxh6oJz2TWHWw), **without the help of Google**:

General

- Look for the right source of information online
- How to create, open, close, read and write files
- What are file descriptors
- What are the 3 standard file descriptors, what are their purpose and what are their `POSIX` names
- How to use the I/O system calls `open`, `close`, `read` and `write`
- What are and how to use the flags `O_RDONLY`, `O_WRONLY`, `O_RDWR`
- What are file permissions, and how to set them when creating a file with the `open` system call

- What is a system call
- What is the difference between a function and a system call

(https://alx-intranet.hbtn.io/)

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl (https://github.com/holbertonschool/Betty/blob/master/betty-style.pl) and betty-doc.pl (https://github.com/holbertonschool/Betty/blob/master/betty-doc.pl)
- You are not allowed to use global variables
- No more than 5 functions per file
- The only C standard library functions allowed are `malloc`, `free` and `exit`. Any use of functions like `printf`, `puts`, `calloc`, `realloc` etc… is forbidden
- Allowed syscalls: `read`, `write`, `open`, `close`
- You are allowed to use _putchar (https://github.com/holbertonschool/_putchar.c/blob/master/_putchar.c)
- You don't have to push `_putchar.c`, we will use our file. If you do it won't be taken into account
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- The prototypes of all your functions and the prototype of the function `_putchar` should be included in your header file called `main.h`
- Don't forget to push your header file
- All your header files should be include guarded
- Tip: always prefer using symbolic constants (`POSIX`) vs numbers when it makes sense. For instance `read(STDIN_FILENO, ...` vs `read(0, ...`

Quiz questions

Question #0

What is the `unistd` symbolic constant for the standard input?

(https://alx-intranet.hbtn.io/)

- STDIN_FILENO
- STDOUT_FILENO
- STDERR_FILENO

Question #1

What is the `unistd` symbolic constant for the standard output?

- STDIN_FILENO
- STDOUT_FILENO
- STDERR_FILENO

Question #2

What is the `unistd` symbolic constant for the Standard error?

- STDIN_FILENO
- STDOUT_FILENO
- STDERR_FILENO

Question #3

What is the `oflag` used to open a file with the mode read only?

- `O_WRONLY`
- `O_RDONLY`
- `O_RDWR`

Question #4

What is the `oflag` used to open a file in mode read + write?

- `O_WRONLY`
- `O_RDONLY`
- `O_RDWR`

Question #5

What is the correct combination of `oflag` s used to open a file with the mode write only, create it if it doesn't exist and append new content at the end if it already exists?

- `O_WRONLY`
- `O_WRONLY | O_CREAT | O_EXCL`
- `O_WRONLY | O_CREAT | O_APPEND`
- `O_RDWR | O_CREAT | O_APPEND`

Question #6

is `open` a function or a system call? (select all valid answers)

- [ ] it's a function
- [ ] it's a system call
- [ ] it's a library call
- [ ] it's a function provided by the kernel
- [ ] it's a kernel routine

Question #7

What system call would you use to write to a file descriptor? (select all correct answers)

- printf
- fprintf
- write

Question #8

Without context, on Ubuntu 14.04 LTS, `write` is a … (please select all correct answers):

- [ ] executable
- [ ] system call
- [ ] library call
- [ ] game
- [ ] kernel routine

Question #9

What is the return value of the system call `open` if it fails?

- 0
- -1
- 98

Question #10
(https://alx-intranet.hbtn.io/)

Most of the time, on a classic, modern Linux system, what will be the value of the first file descriptor you will get after opening a new file with `open` (if `open` succeeds of course):

- [ ] 0
- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4
- [ ] 5
- [ ] 6

Question #11

why? #AlwaysAskWhy

- [ ] Because this will be the first opened file descriptor and in CS we start counting starting from `0`
- [ ] Because this will be the first opened file descriptor and we start counting starting from `1`
- [ ] Because this will be the second opened file descriptor for my process
- [ ] Because this will be the third opened file descriptor for my process
- [ ] Because most of the time, I will already have `stdin` (value 0), `stdout` (value `1`) and `stderr` (value 2) opened when my program starts executing.
- [ ] I don't care I never ask why, just let me access the tasks!

Question #12

Which of these answers are the equivalent of `O_RDWR` on Ubuntu 14.04 LTS? (select all correct answers):

- [ ] O_RDONLY
- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 1 << 1
- [ ] 3 & 2
- [ ] 3 | 2
- [ ] O_WRONLY
- [ ] (O_RDONLY + O_WRONLY)
- [ ] (O_RDONLY | O_WRONLY)

- [ ] (O_RDONLY & O_WRONLY)
- [ ] (O_RDONLY && O_WRONLY)
- [ ] (O_RDONLY << 1)
- [ ] (O_WRONLY << 1)
- [ ] 0

(https://alx-intranet.hbtn.io/)

Question #13

What happens if you try to write “Best” to the standard **input** on Ubuntu 14.04 LTS?

- ( ) Nothing
- ( ) Segmentation fault
- ( ) The text will be printed on the terminal but I can’t pipe it
- ( ) The text will be printed on the terminal on the standard output

Question #14

When I am using `O_WRONLY | O_CREAT | O_APPEND` -> the `|` are bitwise operators.

- ( ) True
- ( ) False

Submit answers

Please make sure to validate all quiz questions before moving on to project tasks