



Curriculum

SE Foundations ^

Average: 137.49% v

You have a captain's log due before 2024-04-21 (in 1 day)! Log it now!
(/captain_logs/5596018/edit)

0x15. C - File I/O

C

Syscall

⚙ Weight: 1

📅 Project over - took place from Sep 4, 2023 6:00 AM to Sep 5, 2023 6:00 AM☒ An auto review will be launched at the deadline

In a nutshell...

- **Auto QA review:** 43.0/43 mandatory & 0.0/9 optional
- **Altogether: 100.0%**
 - Mandatory: 100.0%
 - Optional: 0.0%
 - Calculation: $100.0\% + (100.0\% * 0.0\%) == 100.0\%$

Resources

Read or watch:

- File descriptors (/rltoken/Duva-9Fjyskt39R__Nnazg)
- C Programming in Linux Tutorial #024 - open() read() write() Functions (/rltoken/x05veqiLPSxXmJf9zTtCkQ)

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 (/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

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/alx-tools/Betty/blob/master/betty-style.pl>) and `betty-doc.pl` (<https://github.com/alx-tools/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/alx-tools/_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

Great! You've completed the quiz successfully! Keep going! ([Show quiz](#))

Tasks

0. Tread lightly, she is near

mandatory

Score: 100.0% (Checks completed: 100.0%)

Write a function that reads a text file and prints it to the `POSIX` standard output.

- Prototype: `ssize_t read_textfile(const char *filename, size_t letters);`
- where `letters` is the number of letters it should read and print
- returns the actual number of letters it could read and print
- if the file can not be opened or read, return `0`
- if `filename` is `NULL` return `0`
- if `write` fails or does not write the expected amount of bytes, return `0`



```
julien@ubuntu:~/0x15. File descriptors and permissions$ cat Requiescat
```

Requiescat
by Oscar Wilde

Tread lightly, she is near
Under the snow,
Speak gently, she can hear
The daisies grow.

All her bright golden hair
Tarnished with rust,
She that was young and fair
Fallen to dust.

Lily-like, white as snow,
She hardly knew
She was a woman, so
Sweetly she grew.

Coffin-board, heavy stone,
Lie on her breast,
I vex my heart alone,
She is at rest.

Peace, Peace, she cannot hear
Lyre or sonnet,
All my life's buried here,
Heap earth upon it.

```
julien@ubuntu:~/0x15. File descriptors and permissions$ cat 0-main.c
```

```
#include <stdio.h>
#include <stdlib.h>
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(int ac, char **av)
{
    ssize_t n;

    if (ac != 2)
    {
        dprintf(2, "Usage: %s filename\n", av[0]);
        exit(1);
    }
    n = read_textfile(av[1], 114);
    printf("\n(printed chars: %li)\n", n);
    n = read_textfile(av[1], 1024);
    printf("\n(printed chars: %li)\n", n);
    return (0);
}
```



```
}
(julien@ubuntu:~/0x15. File descriptors and permissions$ gcc -Wall -pedantic -Werror -Wextra
-std=gnu89 0-main.c 0-read_textfile.c -o a
julien@ubuntu:~/0x15. File descriptors and permissions$ ./a Requiescat
Requiescat
by Oscar Wilde

Tread lightly, she is near
Under the snow,
Speak gently, she can hear
The daisies grow.
(printed chars: 114)
Requiescat
by Oscar Wilde

Tread lightly, she is near
Under the snow,
Speak gently, she can hear
The daisies grow.

All her bright golden hair
Tarnished with rust,
She that was young and fair
Fallen to dust.

Lily-like, white as snow,
She hardly knew
She was a woman, so
Sweetly she grew.

Coffin-board, heavy stone,
Lie on her breast,
I vex my heart alone,
She is at rest.

Peace, Peace, she cannot hear
Lyre or sonnet,
All my life's buried here,
Heap earth upon it.

(printed chars: 468)
julien@ubuntu:~/0x15. File descriptors and permissions$
```


Repo:

- GitHub repository: [alx-low_level_programming](#)
- Directory: [0x15-file_io](#)
- File: [0-read_textfile.c](#)



 Done!

Check your code

 Get a sandbox

QA Review

1. Under the snow

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Create a function that creates a file.

- Prototype: `int create_file(const char *filename, char *text_content);`
- where `filename` is the name of the file to create and `text_content` is a `NULL` terminated string to write to the file
- Returns: `1` on success, `-1` on failure (file can not be created, file can not be written, write "fails", etc...)
- The created file must have those permissions: `rw-----` . If the file already exists, do not change the permissions.
- if the file already exists, truncate it
- if `filename` is `NULL` return `-1`
- if `text_content` is `NULL` create an empty file



```

julien@ubuntu:~/0x15. File descriptors and permissions$ cat 1-main.c
#include <stdio.h>
#include <stdlib.h>
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(int ac, char **av)
{
    int res;

    if (ac != 3)
    {
        dprintf(2, "Usage: %s filename text\n", av[0]);
        exit(1);
    }
    res = create_file(av[1], av[2]);
    printf("-> %i\n", res);
    return (0);
}
julien@ubuntu:~/0x15. File descriptors and permissions$ gcc -Wall -pedantic -Werror -Wextra
-std=gnu89 1-main.c 1-create_file.c -o b
julien@ubuntu:~/0x15. File descriptors and permissions$ ./b hello world
-> 1)
julien@ubuntu:~/0x15. File descriptors and permissions$ ls -l hello
-rw----- 1 julien julien 5 Dec  3 14:28 hello
julien@ubuntu:~/0x15. File descriptors and permissions$ cat hello
worldjulien@ubuntu:~/0x15. File descriptors and permis$

```

Repo:

- GitHub repository: alx-low_level_programming
- Directory: 0x15-file_io
- File: 1-create_file.c

☒ Done!

2. Speak gently, she can hear

Score: 100.0% (Checks completed: 100.0%)



Write a function that appends text at the end of a file.

- Prototype: `int append_text_to_file(const char *filename, char *text_content);`

- where `filename` is the name of the file and `text_content` is the NULL terminated string to add at the end of the file
- Return: 1 on success and -1 on failure
- Do not create the file if it does not exist
- If `filename` is NULL return -1
- If `text_content` is NULL, do not add anything to the file. Return 1 if the file exists and -1 if the file does not exist or if you do not have the required permissions to write the file

```
julien@ubuntu:~/0x15. File descriptors and permissions$ cat 2-main.c
#include <stdio.h>
#include <stdlib.h>
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(int ac, char **av)
{
    int res;

    if (ac != 3)
    {
        dprintf(2, "Usage: %s filename text\n", av[0]);
        exit(1);
    }
    res = append_text_to_file(av[1], av[2]);
    printf("-> %i\n", res);
    return (0);
}
julien@ubuntu:~/0x15. File descriptors and permissions$ echo -n Hello > hello
julien@ubuntu:~/0x15. File descriptors and permissions$ ls -l hello
-rw-rw-r-- 1 julien julien 5 Dec  3 14:48 hello
julien@ubuntu:~/0x15. File descriptors and permissions$ gcc -Wall -pedantic -Werror -Wextra
-std=gnu89 2-main.c 2-append_text_to_file.c -o c
julien@ubuntu:~/0x15. File descriptors and permissions$ ./c hello " World!
> "
-> 1)
julien@ubuntu:~/0x15. File descriptors and permissions$ cat hello
Hello World!
julien@ubuntu:~/0x15. File descriptors and permissions$
```

Repo:

- GitHub repository: [alx-low_level_programming](#)
- Directory: [0x15-file_io](#)
- File: [2-append_text_to_file.c](#)



[\(7\) Done!](#)[Check your code](#)[> Get a sandbox](#)[QA Review](#)

3. cp

mandatory

Score: 100.0% (Checks completed: 100.0%)

Write a program that copies the content of a file to another file.

- Usage: `cp file_from file_to`
- if the number of argument is not the correct one, exit with code 97 and print `Usage: cp file_from file_to`, followed by a new line, on the `POSIX` standard error
- if `file_to` already exists, truncate it
- if `file_from` does not exist, or if you can not read it, exit with code 98 and print `Error: Can't read from file NAME_OF_THE_FILE`, followed by a new line, on the `POSIX` standard error
 - where `NAME_OF_THE_FILE` is the first argument passed to your program
- if you can not create or if write to `file_to` fails, exit with code 99 and print `Error: Can't write to NAME_OF_THE_FILE`, followed by a new line, on the `POSIX` standard error
 - where `NAME_OF_THE_FILE` is the second argument passed to your program
- if you can not close a file descriptor, exit with code 100 and print `Error: Can't close fd FD_VALUE`, followed by a new line, on the `POSIX` standard error
 - where `FD_VALUE` is the value of the file descriptor
- Permissions of the created file: `rw-rw-r--`. If the file already exists, do not change the permissions
- You must read 1,024 bytes at a time from the `file_from` to make less system calls. Use a buffer
- You are allowed to use `dprintf`

```
julien@ubuntu:~/0x15. File descriptors and permissions$ gcc -Wall -pedantic -Werror -Wextra
-std=gnu89 3-cp.c -o cp
julien@ubuntu:~/0x15. File descriptors and permissions$ cat incitatus
Why you should think twice before putting pictures on social media.
(What you always wanted to know about @Incitatus)
#PrivacyAware
http://imgur.com/a/Mq1tc
julien@ubuntu:~/0x15. File descriptors and permissions$ ./cp incitatus Incitatus
julien@ubuntu:~/0x15. File descriptors and permissions$ ls -l Incitatus
-rw-rw-r-- 1 julien julien 158 Dec  3 15:39 Incitatus
julien@ubuntu:~/0x15. File descriptors and permissions$ cat Incitatus
Why you should think twice before putting pictures on social media.
(What you always wanted to know about @Incitatus)
#PrivacyAware
http://imgur.com/a/Mq1tc
julien@ubuntu:~/0x15. File descriptors and permissions$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x15-file_io`
- File: `3-cp.c`



[\(7\) Done!](#)[Check your code](#)[>_ Get a sandbox](#)[QA Review](#)

4. elf

#advanced

Score: 0.0% (Checks completed: 0.0%)

Write a program that displays the information contained in the `ELF` header at the start of an `ELF` file.

- Usage: `elf_header elf_filename`
- Displayed information: (no less, no more, do not include trailing whitespace)
 - Magic
 - Class
 - Data
 - Version
 - OS/ABI
 - ABI Version
 - Type
 - Entry point address
- Format: the same as `readelf -h` (*version 2.26.1*)
- If the file is not an `ELF` file, or on error, exit with status code `98` and display a comprehensive error message to `stderr`
- You are allowed to use `lseek` once
- You are allowed to use `read` a maximum of 2 times at runtime
- You are allowed to have as many functions as you want in your source file
- You are allowed to use `printf`

man `elf` , `readelf`



```
julien@ubuntu:~/0x15. File descriptors and permissions$ gcc -Wall -pedantic -Werror -Wextra  
-std=gnu89 100-elf_header.c -o elf_header
```

```
julien@ubuntu:~/0x15. File descriptors and permissions$ ./elf_header ubuntu64
```

ELF Header:

```
Magic:  7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00  
Class:                                     ELF64  
Data:                                       2's complement, little endian  
Version:                                   1 (current)  
OS/ABI:                                    UNIX - System V  
ABI Version:                               0  
Type:                                       EXEC (Executable file)  
Entry point address:                       0x400600
```

```
julien@ubuntu:~/0x15. File descriptors and permissions$ readelf --version
```

GNU readelf (GNU Binutils for Ubuntu) 2.26.1

Copyright (C) 2015 Free Software Foundation, Inc.

This program is free software; you may redistribute it under the terms of
the GNU General Public License version 3 or (at your option) any later version.

This program has absolutely no warranty.

```
julien@ubuntu:~/0x15. File descriptors and permissions$ readelf -h ubuntu64
```

ELF Header:

```
Magic:  7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00  
Class:                                     ELF64  
Data:                                       2's complement, little endian  
Version:                                   1 (current)  
OS/ABI:                                    UNIX - System V  
ABI Version:                               0  
Type:                                       EXEC (Executable file)  
Machine:                                   Advanced Micro Devices X86-64  
Version:                                   0x1  
Entry point address:                       0x400600  
Start of program headers:                  64 (bytes into file)  
Start of section headers:                  6936 (bytes into file)  
Flags:                                     0x0  
Size of this header:                        64 (bytes)  
Size of program headers:                   56 (bytes)  
Number of program headers:                  9  
Size of section headers:                   64 (bytes)  
Number of section headers:                 31  
Section header string table index: 28
```

```
julien@ubuntu:~/0x15. File descriptors and permissions$ ./elf_header /lib/ld-linux.so.2
```

ELF Header:

```
Magic:  7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00  
Class:                                     ELF32  
Data:                                       2's complement, little endian  
Version:                                   1 (current)  
OS/ABI:                                    UNIX - System V  
ABI Version:                               0  
Type:                                       DYN (Shared object file)  
Entry point address:                       0xac0
```

```
julien@ubuntu:~/0x15. File descriptors and permissions$ readelf -h /lib/ld-linux.so.2
```

ELF Header:

```
Magic:  7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
```



```

Class: ELF32
(/)Data: 2's complement, little endian
Version: 1 (current)
OS/ABI: UNIX - System V
ABI Version: 0
Type: DYN (Shared object file)
Machine: Intel 80386
Version: 0x1
Entry point address: 0xac0
Start of program headers: 52 (bytes into file)
Start of section headers: 145756 (bytes into file)
Flags: 0x0
Size of this header: 52 (bytes)
Size of program headers: 32 (bytes)
Number of program headers: 7
Size of section headers: 40 (bytes)
Number of section headers: 24
Section header string table index: 23

```

julien@ubuntu:~/0x15. File descriptors and permissions\$./elf_header netbsd32

ELF Header:

```

Magic: 7f 45 4c 46 01 01 01 02 00 00 00 00 00 00 00
Class: ELF32
Data: 2's complement, little endian
Version: 1 (current)
OS/ABI: UNIX - NetBSD
ABI Version: 0
Type: EXEC (Executable file)
Entry point address: 0x80484c0

```

julien@ubuntu:~/0x15. File descriptors and permissions\$./elf_header sortix32

ELF Header:

```

Magic: 7f 45 4c 46 01 01 01 53 00 00 00 00 00 00 00
Class: ELF32
Data: 2's complement, little endian
Version: 1 (current)
OS/ABI: <unknown: 53>
ABI Version: 0
Type: EXEC (Executable file)
Entry point address: 0x80484c0

```

julien@ubuntu:~/0x15. File descriptors and permissions\$./elf_header solaris32

ELF Header:

```

Magic: 7f 45 4c 46 01 01 01 06 01 00 00 00 00 00 00
Class: ELF32
Data: 2's complement, little endian
Version: 1 (current)
OS/ABI: UNIX - Solaris
ABI Version: 1
Type: EXEC (Executable file)
Entry point address: 0x8052400

```

julien@ubuntu:~/0x15. File descriptors and permissions\$./elf_header sparc32

ELF Header:

```

Magic: 7f 45 4c 46 01 02 01 00 00 00 00 00 00 00 00
Class: ELF32

```



```
Data:                2's complement, big endian
(/)Version:          1 (current)
OS/ABI:              UNIX - System V
ABI Version:         0
Type:                EXEC (Executable file)
Entry point address: 0x10d20
julien@ubuntu:~/0x15. File descriptors and permissions$
```


Repo:

- GitHub repository: alx-low_level_programming
- Directory: 0x15-file_io
- File: 100-elf_header.c

☐ Done?

Check your code

Ask for a new correction

 Get a sandbox

QA Review

Copyright © 2024 ALX, All rights reserved.

