You have a captain's log due before 2022-10-09 (in 2 days)! Log it now!
(/captain_logs/1077316/edit)

0x0A. C - argc, argv

C

By: Julien Barbier

Weight: 1

Project over - took place from Sep 30, 2022 6:00 AM to Oct 1, 2022 6:00 AM

An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 24.05/37 mandatory & 5.85/9 optional
- **Altogether: 107.25%**
  - Mandatory: 65.0%
  - Optional: 65.0%
  - Calculation: 65.0% + (65.0% * 65.0%) == **107.25%**

Resources

Read or watch:

- Arguments to main (/rltoken/Jip_nI4tv2ybQZ-jV3fqJg)
- argc and argv (/rltoken/31aLwv8qsXuiUZrOk9Djqg)
- What does argc and argv mean? (/rltoken/A0pzqsIB6Z3Y3OV3hJQ6Tw)
- how to compile with unused variables (/rltoken/MkOUE1ndq1UAx9Erk-AVbg)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/DBgGt1BaQ75AkikI88WbEw), **without the help of Google**:

General

(/)

- How to use arguments passed to your program
- What are two prototypes of `main` that you know of, and in which case do you use one or the other
- How to use `__attribute__((unused))` or `(void)` to compile functions with unused variables or parameters

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 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
- You are allowed to use the standard library

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Hide quiz)

Question #0

What is `argc` ?

- [✓] The number of command line arguments

- [ ] A flag set to 1 when command line arguments are present

(/)

- [✓] The size of the `argv` array
- [ ] The length of the first command line argument

Question #1

What is `argv` ?

- [ ] An array containing the program compilation flags
- [✓] An array containing the program command line arguments
- [✓] An array of size `argc`

Question #2

What is `argv[0]`

- ○ NULL
- ○ It does not always exist
- ○ The first command line argument
- [✓ The program name]

Question #3

What is `argv[argc]` ?

- [✓ NULL]
- ○ It does not always exist
- ○ The first command line argument
- ○ The program name
- ○ The last command line argument

Question #4

In the following command, what is `argv[2]` ?

```
$ ./argv My School is fun
```

- ○ NULL
- ○ ./argv
- ○ My

[✓ School]

(/)

My School

is

fun

is fun

My School is fun

Question #5

In the following command, what is `argv[2]` ?

```
$ ./argv "My School" "is fun"
```

NULL

./argv

My

School

My School

is

fun

[✓ is fun]

My School is fun

Question #6

In the following command, what is `argv[2]` ?

```
$ ./argv "My School is fun"
```

[✓ NULL]

./argv

My

School

My School

is

fun

is fun

My School is fun
(/)

Tasks

0. It ain't what they call you, it's what you answer to

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a program that prints its name, followed by a new line.

- If you rename the program, it will print the new name, without having to compile it again
- You should not remove the path before the name of the program

```
julien@ubuntu:~/0x0A. argc, argv$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 0-w
hatsmyname.c -o mynameis
julien@ubuntu:~/0x0A. argc, argv$ ./mynameis
./mynameis
julien@ubuntu:~/0x0A. argc, argv$ mv mynameis mynewnameis
julien@ubuntu:~/0x0A. argc, argv$ ./mynewnameis
./mynewnameis
julien@ubuntu:~/0x0A. argc, argv$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0A-argc_argv`
- File: `0-whatsmyname.c`

Done! Help Check your code >_ Get a sandbox QA Review

1. Silence is argument carried out by other means

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a program that prints the number of arguments passed into it.

- Your program should print a number, followed by a new line

(/)

```
julien@ubuntu:~/0x0A. argc, argv$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 1-a
rgs.c -o nargs
julien@ubuntu:~/0x0A. argc, argv$ ./nargs
0
julien@ubuntu:~/0x0A. argc, argv$ ./nargs hello
1
julien@ubuntu:~/0x0A. argc, argv$ ./nargs "hello, world"
1
julien@ubuntu:~/0x0A. argc, argv$ ./nargs hello, world
2
julien@ubuntu:~/0x0A. argc, argv$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0A-argc_argv`
- File: `1-args.c`

Done! Help Check your code >_ Get a sandbox QA Review

2. The best argument against democracy is a five-minute conversation with the average voter

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a program that prints all arguments it receives.

- All arguments should be printed, including the first one
- Only print one argument per line, ending with a new line

```
julien@ubuntu:~/0x0A. argc, argv$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 2-a
rgs.c -o args
julien@ubuntu:~/0x0A. argc, argv$ ./args
./args
julien@ubuntu:~/0x0A. argc, argv$ ./args You can do anything, but not everything.
./args
You
can
do
anything,
but
not
everything.
julien@ubuntu:~/0x0A. argc, argv$
```

(/)
Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0A-argc_argv`
- File: `2-args.c`

Done! Help Check your code >_ Get a sandbox QA Review

3. Neither irony nor sarcasm is argument

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a program that multiplies two numbers.

- Your program should print the result of the multiplication, followed by a new line
- You can assume that the two numbers and result of the multiplication can be stored in an integer
- If the program does not receive two arguments, your program should print `Error`, followed by a new line, and return `1`

```
julien@ubuntu:~/0x0A. argc, argv$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 3-m
ul.c -o mul
julien@ubuntu:~/0x0A. argc, argv$ ./mul 2 3
6
julien@ubuntu:~/0x0A. argc, argv$ ./mul 2 -3
-6
julien@ubuntu:~/0x0A. argc, argv$ ./mul 2 0
0
julien@ubuntu:~/0x0A. argc, argv$ ./mul 245 3245342
795108790
julien@ubuntu:~/0x0A. argc, argv$ ./mul
Error
julien@ubuntu:~/0x0A. argc, argv$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0A-argc_argv`
- File: `3-mul.c`

Done! Help Check your code >_ Get a sandbox QA Review

4. To infinity and beyond (/)

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a program that adds positive numbers.

- Print the result, followed by a new line
- If no number is passed to the program, print `0`, followed by a new line
- If one of the number contains symbols that are not digits, print `Error`, followed by a new line, and return `1`
- You can assume that numbers and the addition of all the numbers can be stored in an `int`

```
julien@ubuntu:~/0x0A. argc, argv$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 4-a
dd.c -o add
julien@ubuntu:~/0x0A. argc, argv$ ./add 1 1
2
julien@ubuntu:~/0x0A. argc, argv$ ./add 1 10 100 1000
1111
julien@ubuntu:~/0x0A. argc, argv$ ./add 1 2 3 e 4 5
Error
julien@ubuntu:~/0x0A. argc, argv$ ./add
0
julien@ubuntu:~/0x0A. argc, argv$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0A-argc_argv`
- File: `4-add.c`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

5. Minimal Number of Coins for Change

#advanced

Score: 65.0% (*Checks completed: 100.0%*)

Write a program that prints the minimum number of coins to make change for an amount of money.

- Usage: `./change cents`
- where `cents` is the amount of cents you need to give back
- if the number of arguments passed to your program is not exactly `1`, print `Error`, followed by a new line, and return `1`
- you should use `atoi` to parse the parameter passed to your program
- If the number passed as the argument is negative, print `0`, followed by a new line
- You can use an unlimited number of coins of values 25, 10, 5, 2, and 1 cent

```
julien@ubuntu:~/0x0A. argc, argv$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 100
-change.c -o change
julien@ubuntu:~/0x0A. argc, argv$ ./change
Error
julien@ubuntu:~/0x0A. argc, argv$ ./change 10
1
julien@ubuntu:~/0x0A. argc, argv$ ./change 100
4
julien@ubuntu:~/0x0A. argc, argv$ ./change 101
5
julien@ubuntu:~/0x0A. argc, argv$ ./change 13
3
julien@ubuntu:~/0x0A. argc, argv$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0A-argc_argv`
- File: `100-change.c`

Done! Help Check your code >_ Get a sandbox QA Review

Copyright © 2022 ALX, All rights reserved.