C - Hello, World

- Level: Novice
- By Julien Barbier
- Weight: 1
- Your score will be updated as you progress.

Resources

Read or watch:

- Everything you need to know to start with C.pdf (*You do not have to learn everything in there yet, but make sure you read it entirely first*)
- Dennis Ritchie
- "C" Programming Language: Brian Kernighan
- Why C Programming Is Awesome
- Learning to program in C part 1
- Learning to program in C part 2
- Understanding C program Compilation Process
- Betty Coding Style
- Hash-bang under the hood (*Look at only after you finish consuming the other resources*)
- Linus Torvalds on C vs. C++ (*Look at only after you finish consuming the other resources*)

man or help:

- `gcc`
- `printf (3)`
- `puts`

- `putchar`

Learning Objectives

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

General

- Why C programming is awesome
- Who invented C
- Who are Dennis Ritchie, Brian Kernighan and Linus Torvalds
- What happens when you type `gcc main.c`
- What is an entry point
- What is `main`
- How to print text using `printf`, `puts` and `putchar`
- How to get the size of a specific type using the unary operator `sizeof`
- How to compile using `gcc`
- What is the default program name when compiling with `gcc`
- What is the official C coding style and how to check your code with `betty-style`
- How to find the right header to include in your source code when using a standard library function
- How does the `main` function influence the return value of the program

Requirements

C

- 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 repo, containing a description of the repository
- A `README.md` file, at the root of the folder of *this* project, containing a description of the project
- There should be no errors and no warnings during compilation
- You are not allowed to use `system`
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl

Shell Scripts

- Allowed editors: `vi`, `vim`, `emacs`
- All your scripts will be tested on Ubuntu 20.04 LTS
- All your scripts should be exactly two lines long (`$ wc -l file` should print 2)
- All your files should end with a new line
- The first line of all your files should be exactly `#!/bin/bash`

More Info

Betty linter

To run the Betty linter just with command `betty <filename>`:

- Go to the Betty repository
- Clone the repo to your local machine
- `cd` into the Betty directory
- Install the linter with `sudo ./install.sh`
- `emacs` or `vi` a new file called `betty`, and copy the script below:

```
#!/bin/bash
# Simply a wrapper script to keep you from having to use betty-style
# and betty-doc separately on every item.
# Originally by Tim Britton (@wintermanc3r), multiargument added by
# Larry Madeo (@hillmonkey)

BIN_PATH="/usr/local/bin"
BETTY_STYLE="betty-style"
BETTY_DOC="betty-doc"

if [ "$#" = "0" ]; then
    echo "No arguments passed."
    exit 1
fi

for argument in "$@" ; do
    echo -e "\n========== $argument =========="
    ${BIN_PATH}/${BETTY_STYLE} "$argument"
    ${BIN_PATH}/${BETTY_DOC} "$argument"
done
```

- Once saved, exit file and change permissions to apply to all users with `chmod a+x betty`
- Move the `betty` file into `/bin/` directory or somewhere else in your `$PATH` with `sudo mv betty /bin/`

You can now type `betty <filename>` to run the Betty linter!

Manual QA Review

It is your responsibility to request a review for your blog from a peer before the project's deadline. If no peers have been reviewed, you should request a review from a TA or staff member.

Quiz questions

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

Tasks

0. Preprocessor

mandatory

Write a script that runs a C file through the preprocessor and save the result into another file.

- The C file name will be saved in the variable `$CFILE`
- The output should be saved in the file `c`

```
julien@ubuntu:~/c/$ cat main.c
#include <stdio.h>


/**
 * main - Entry point
 *
 * Return: Always 0 (Success)
 */
int main(void)
{
    return (0);
}
julien@ubuntu:~/c/$ export CFILE=main.c
julien@ubuntu:~/c/$ ./0-preprocessor
julien@ubuntu:~/c/$ tail c
# 942 "/usr/include/stdio.h" 3 4


# 2 "main.c" 2




# 3 "main.c"
int main(void)
{
 return (0);
}
```

```
julien@ubuntu:~/c/$
```

Repo:

- GitHub repository: `holbertonschool-low_level_programming`
- Directory: `hello_world`
- File: `0-preprocessor`

Review your work Get a sandbox
0/5 pts
1. Compiler
mandatory
Write a script that compiles a C file but does not link.

- The C file name will be saved in the variable `$CFILE`
- The output file should be named the same as the C file, but with the extension `.o` instead of `.c`.
  - Example: if the C file is `main.c`, the output file should be `main.o`

```
julien@ubuntu:~/c/$ export CFILE=main.c
julien@ubuntu:~/c/$ cat main.c
#include <stdio.h>


/**
 * main - Entry point
 *
 * Return: Always 0 (Success)
 */
int main(void)
{
    return (0);
}
julien@ubuntu:~/c/$ ./1-compiler
julien@ubuntu:~/c/$ ls
0-preprocessor  1-compiler   c            main.o
Makefile               100-intel      main.c  main.s
julien@ubuntu:~/c/$ cat -v main.o | head
^?ELF^B^A^A^@^@^@^@^@^@^@^@^@^A^@>^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^P^B^@^@^@^@^@^@^@^@^@^@@^@^@^@^@^@@^@^K^@^H^@UHM-^IM-eM-8^@^@^@^@]M-C^@GC
C: (Ubuntu 5.4.0-6ubuntu1~16.04.2) 5.4.0 20160609^@^T^@^@^@^@^@^@^@^AzR^@^A
x^P^A^[^L^G^HM-^P^A^@^@^\^@^@^@^\^@^@^@^@^@^@^@^K^@^@^@^@A^N^PM-^F^BC^M^FF^
L^G^H^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@^@^D^@M-qM
-^?^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^C^@^B^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^C^@^@^
```

```
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^E^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^C^@^F^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^D^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^H^@^@^@^R^@^A^@^@^@^@^@^@^@^@^@^K^@^@^@^@^@^@^@^@ma
in.c^@main^@^@^@^@ ^@^@^@^@^@^@^@^B^@^@^@^B^@^@^@^@^@^@^@^@^@^@^@^@.symtab^
@.strtab^@.shstrtab^@.text^@.data^@.bss^@.comment^@.note.GNU-stack^@.rela.e
h_frame^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^[^@^
@^@^A^@^@^@^F^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@@^@^@^@^@^@^@^@^K^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@!^@^@^@^A^@^@^@^C^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@K^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@^
@^@^@^@^@^@^@^@^@^@^@^@'^@^@^@^H^@^@^@^C^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@K^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@,
^@^@^@^A^@^@^@0^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@K^@^@^@^@^@^@^@5^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^A^@^@^@^@^@^@^@^A^@^@^@^@^@^@^@5^@^@^@^A^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@M-^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@J^@^@^@^A^@^@^@^B^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@M
-^@^@^@^@^@^@^@^@8^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^H^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@E^@^@^@^D^@^@^@@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@M- ^A^@^@^@^@^@^@^X^@^@^
@^@^@^@^@    ^@^@^@^F^@^@^@^H^@^@^@^@^@^@^@^X^@^@^@^@^@^@^@^Q^@^@^@^C^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@M-8^A^@^@^@^@^@^@T^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@^@^B^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@M-8^@^@^@^@^@^@^@M-X^@^@^@^@^@^@^@

^@^@^@^H^@^@^@^H^@^@^@^@^@^@^@^X^@^@^@^@^@^@^@  ^@^@^@^C^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@M-^P^A^@^@^@^@^@^@^M^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@julien@ubuntu:~/c/$
```

Repo:

- GitHub repository: `holbertonschool-low_level_programming`
- Directory: `hello_world`
- File: `1-compiler`

Review your work Get a sandbox
0/5 pts

2. Assembler

mandatory

Write a script that generates the assembly code of a C code and save it in an output file.

- The C file name will be saved in the variable `$CFILE`
- The output file should be named the same as the C file, but with the extension `.s` instead of `.c`.
    - Example: if the C file is `main.c`, the output file should be `main.s`

```
julien@ubuntu:~/c/$ export CFILE=main.c
julien@ubuntu:~/c/$ cat main.c
#include <stdio.h>


/**
 * main - Entry point
 *
 * Return: Always 0 (Success)
 */
int main(void)
{
    return (0);
}
julien@ubuntu:~/c/$ ./2-assembler
julien@ubuntu:~/c/$ ls
0-preprocessor  1-compiler  2-assembler c  main.c  main.s  Makefile
julien@ubuntu:~/c/$ cat main.s
    .file   "main.c"
    .text
    .globl  main
    .type   main, @function
main:
.LFB0:
    .cfi_startproc
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    movl    $0, %eax
    popq    %rbp
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE0:
    .size   main, .-main
    .ident  "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.2) 5.4.0 20160609"
    .section    .note.GNU-stack,"",@progbits
julien@ubuntu:~/c/$
```

Repo:

- GitHub repository: `holbertonschool-low_level_programming`
- Directory: `hello_world`
- File: `2-assembler`

Review your work Get a sandbox

0/5 pts

3. Name

mandatory

Write a script that compiles a C file and creates an executable named `cisfun`.

- The C file name will be saved in the variable `$CFILE`

```
julien@ubuntu:~/c/$ export CFILE=main.c
julien@ubuntu:~/c/$ cat main.c
#include <stdio.h>

/**
 * main - Entry point
 *
 * Return: Always 0 (Success)
 */
int main(void)
{
    return (0);
}
julien@ubuntu:~/c/$ ./3-name
julien@ubuntu:~/c/$ ls
0-preprocessor  1-compiler   3-name  cisfun  main.o  Makefile
100-intel       2-assembler  c       main.c  main.s
julien@ubuntu:~/c/$
```

Repo:

- GitHub repository: `holbertonschool-low_level_programming`
- Directory: `hello_world`
- File: `3-name`

Review your work Get a sandbox

0/5 pts

4. Hello, puts

mandatory

Write a C program that prints exactly `"Programming is like building a multilingual puzzle`, followed by a new line.

- Use the function `puts`
- You are not allowed to use `printf`
- Your program should end with the value `0`

```
julien@ubuntu:~/c/$ gcc -Wall -Werror -Wextra -pedantic -std=gnu89 4-puts.c
 && ./a.out

"Programming is like building a multilingual puzzle

julien@ubuntu:~/c/$ echo $?

0

julien@ubuntu:~/c/$
```

Repo:

- GitHub repository: `holbertonschool-low_level_programming`
- Directory: `hello_world`
- File: `4-puts.c`

Review your work Get a sandbox
0/7 pts

5. Hello, printf

mandatory

Write a C program that prints exactly `with proper grammar, but the outcome is a piece of art,`, followed by a new line.

- Use the function `printf`
- You are not allowed to use the function `puts`
- Your program should return `0`
- Your program should compile without warning when using the `-Wall` `gcc` option

```
julien@ubuntu:~/c/$ gcc -Wall -Werror -Wextra -pedantic -std=gnu89 5-print
f.c

julien@ubuntu:~/c/$ ./a.out

with proper grammar, but the outcome is a piece of art,

julien@ubuntu:~/c/$ echo $?

0

julien@ubuntu:~/c/$
```

Repo:

- GitHub repository: `holbertonschool-low_level_programming`
- Directory: `hello_world`
- File: `5-printf.c`

Review your work Get a sandbox
0/7 pts

6. Size is not grandeur, and territory does not make a nation

mandatory

Write a C program that prints the size of various types on the computer it is compiled and run on.

- You should produce the exact same output as in the example

- Warnings are allowed
- Your program should return `0`
- You might have to install the package `libc6-dev-i386` on your Linux (Vagrant) to test the `-m32` `gcc` option

```
julien@ubuntu:~/c/$ gcc 6-size.c -m32 -o size32 2> /tmp/32
julien@ubuntu:~/c/$ gcc 6-size.c -m64 -o size64 2> /tmp/64
julien@ubuntu:~/c/$ ./size32
Size of a char: 1 byte(s)
Size of an int: 4 byte(s)
Size of a long int: 4 byte(s)
Size of a long long int: 8 byte(s)
Size of a float: 4 byte(s)
julien@ubuntu:~/c/$ ./size64
Size of a char: 1 byte(s)
Size of an int: 4 byte(s)
Size of a long int: 8 byte(s)
Size of a long long int: 8 byte(s)
Size of a float: 4 byte(s)
julien@ubuntu:~/c/$ echo $?
0
julien@ubuntu:~/c/$
```

Repo:

- GitHub repository: `holbertonschool-low_level_programming`
- Directory: `hello_world`
- File: `6-size.c`

Review your work Get a sandbox
0/7 pts

Done with the mandatory tasks? Unlock 2 advanced tasks now!

Score

Your score will be updated as you progress.

Please review all the **manual checks** before you launch the project review.

Skip this project
Previous project