# 0x18. C - Dynamic libraries

C

By: Julien Barbier

Weight: 1

Project will start Dec 12, 2022 6:00 AM, must end by Dec 13, 2022 6:00 AM

was released at Dec 12, 2022 12:00 PM

An auto review will be launched at the deadline

## Resources

**Read or watch**:

- What is difference between Dynamic and Static library (Static and Dynamic linking) (/rltoken/XLLmLISlteUlxrLzNdm3_Q)
- create dynamic libraries on Linux (/rltoken/JEqzgE_pPe48rvbspGL-2g)
- Technical Writing (/rltoken/dAV47Y4lulj75aeSxpYHbQ)

## Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/wZXKCWgm5hGCD0ZKtZAOrQ), **without the help of Google**:

### General

- What is a dynamic library, how does it work, how to create one, and how to use it
- What is the environment variable `$LD_LIBRARY_PATH` and how to use it
- What are the differences between static and shared libraries
- Basic usage `nm`, `ldd`, `ldconfig`

### 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

## 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 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
- You are not allowed to use the standard library. Any use of functions like `printf`, `puts`, etc... is forbidden
- 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

## Bash

- Allowed editors: `vi`, `vim`, `emacs`
- All your scripts will be tested on Ubuntu 20.04 LTS
- All your files should end with a new line (why? (http://unix.stackexchange.com/questions/18743/whats-the-point-in-adding-a-new-line-to-the-end-of-a-file/18789))
- The first line of all your files should be exactly `#!/bin/bash`
- A `README.md` file, at the root of the folder of the project, describing what each script is doing
- All your files must be executable

# Tasks

## 0. A library is not a luxury but one of the necessities of life

mandatory

Create the dynamic library libdynamic.so containing all the functions listed below:

```
int _putchar(char c);
int _islower(int c);
int _isalpha(int c);
int _abs(int n);
int _isupper(int c);
int _isdigit(int c);
int _strlen(char *s);
void _puts(char *s);
char *_strcpy(char *dest, char *src);
int _atoi(char *s);
char *_strcat(char *dest, char *src);
char *_strncat(char *dest, char *src, int n);
char *_strncpy(char *dest, char *src, int n);
int _strcmp(char *s1, char *s2);
char *_memset(char *s, char b, unsigned int n);
char *_memcpy(char *dest, char *src, unsigned int n);
char *_strchr(char *s, char c);
unsigned int _strspn(char *s, char *accept);
char *_strpbrk(char *s, char *accept);
char *_strstr(char *haystack, char *needle);
```

If you haven't coded all of the above functions create empty ones, with the right prototype.
Don't forget to push your `main.h` file in your repository, containing at least all the prototypes of the above functions.

```
julien@ubuntu:~/0x18. Dynamic libraries$ ls -la lib*
-rwxrwxr-x 1 julien julien 13632 Jan  7 06:25 libdynamic.so
julien@ubuntu:~/0x18. Dynamic libraries$ nm -D libdynamic.so
0000000000000a90 T _abs
0000000000000aa9 T _atoi
0000000000202048 B __bss_start
                 w __cxa_finalize
0000000000202048 D _edata
0000000000202050 B _end
00000000000011f8 T _fini
                 w __gmon_start__
0000000000000900 T _init
0000000000000bd7 T _isalpha
0000000000000c04 T _isdigit
0000000000000c25 T _islower
0000000000000c46 T _isupper
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
                 w _Jv_RegisterClasses
0000000000000c67 T _memcpy
0000000000000caa T _memset
0000000000000ce9 T _putchar
0000000000000d0e T _puts
0000000000000d4a T _strcat
0000000000000dcf T _strchr
0000000000000e21 T _strcmp
0000000000000e89 T _strcpy
0000000000000eeb T _strlen
0000000000000f15 T _strncat
0000000000000fa5 T _strncpy
0000000000001029 T _strpbrk
000000000000109d T _strspn
0000000000001176 T _strstr
                 U write
julien@ubuntu:~/0x18. Dynamic libraries$ cat 0-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always EXIT_SUCCESS.
 */
int main(void)
{
    printf("%d\n", _strlen("My Dyn Lib"));
    return (EXIT_SUCCESS);
}
julien@ubuntu:~/0x18. Dynamic libraries$ gcc -Wall -pedantic -Werror -Wextra -L. 0-m
ain.c -ldynamic -o len
julien@ubuntu:~/0x18. Dynamic libraries$ ldd len
    linux-vdso.so.1 =>  (0x00007fff5d1d2000)
    libdynamic.so => not found
```

```
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f74c6bb9000)
    /lib64/ld-linux-x86-64.so.2 (0x0000556be5b82000)
julien@ubuntu:~/0x18. Dynamic libraries$ export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
julien@ubuntu:~/0x18. Dynamic libraries$ ldd len
    linux-vdso.so.1 =>  (0x00007fff41ae9000)
    libdynamic.so => ./libdynamic.so (0x00007fd4bf2d9000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fd4beef6000)
    /lib64/ld-linux-x86-64.so.2 (0x0000557566402000)
julien@ubuntu:~/0x18. Dynamic libraries$ ./len
10
julien@ubuntu:~/0x18. Dynamic libraries$
```

**Repo:**

- GitHub repository: `alx-low_level_programming`
- Directory: `0x18-dynamic_libraries`
- File: `libdynamic.so, main.h`

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

## 1. Without libraries what have we? We have no past and no future

mandatory

Create a script that creates a dynamic library called `liball.so` from all the `.c` files that are in the current directory.

```
julien@ubuntu:~/0x18. Dynamic libraries$ ls *.c
abs.c   isalpha.c  islower.c  memcpy.c  putchar.c  strcat.c  strcmp.c  strlen.c   st
rncpy.c  strspn.c
atoi.c  isdigit.c  isupper.c  memset.c  puts.c     strchr.c  strcpy.c  strncat.c  st
rpbrk.c  strstr.c
julien@ubuntu:~/0x18. Dynamic libraries$ ./1-create_dynamic_lib.sh
julien@ubuntu:~/0x18. Dynamic libraries$ nm -D --defined-only liball.so
0000000000000a90 T _abs
0000000000000aa9 T _atoi
0000000000202048 B __bss_start
0000000000202048 D _edata
0000000000202050 B _end
00000000000011f8 T _fini
0000000000000900 T _init
0000000000000bd7 T _isalpha
0000000000000c04 T _isdigit
0000000000000c25 T _islower
0000000000000c46 T _isupper
0000000000000c67 T _memcpy
0000000000000caa T _memset
0000000000000ce9 T _putchar
0000000000000d0e T _puts
0000000000000d4a T _strcat
0000000000000dcf T _strchr
0000000000000e21 T _strcmp
0000000000000e89 T _strcpy
0000000000000eeb T _strlen
0000000000000f15 T _strncat
0000000000000fa5 T _strncpy
0000000000001029 T _strpbrk
000000000000109d T _strspn
0000000000001176 T _strstr
julien@ubuntu:~/0x18. Dynamic libraries$
```

**Repo:**

- GitHub repository: `alx-low_level_programming`
- Directory: `0x18-dynamic_libraries`
- File: `1-create_dynamic_lib.sh`

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

## 2. Let's call C functions from Python

#advanced

I know, you're missing C when coding in Python. So let's fix that!

Create a dynamic library that contains C functions that can be called from Python. See example for more detail.

(/)

```
julien@ubuntu:~/0x18$ cat 100-tests.py
import random
import ctypes

cops = ctypes.CDLL('./100-operations.so')
a = random.randint(-111, 111)
b = random.randint(-111, 111)
print("{} + {} = {}".format(a, b, cops.add(a, b)))
print("{} - {} = {}".format(a, b, cops.sub(a, b)))
print("{} x {} = {}".format(a, b, cops.mul(a, b)))
print("{} / {} = {}".format(a, b, cops.div(a, b)))
print("{} % {} = {}".format(a, b, cops.mod(a, b)))
julien@ubuntu:~/0x16. Doubly linked lists$ python3 100-tests.py
66 + -76 = -10
66 - -76 = 142
66 x -76 = -5016
66 / -76 = 0
66 % -76 = 66
julien@ubuntu:~/0x18$ python3 100-tests.py
-34 + -57 = -91
-34 - -57 = 23
-34 x -57 = 1938
-34 / -57 = 0
-34 % -57 = -34
julien@ubuntu:~/0x18$ python3 100-tests.py
-5 + -72 = -77
-5 - -72 = 67
-5 x -72 = 360
-5 / -72 = 0
-5 % -72 = -5
julien@ubuntu:~/0x18$ python3 100-tests.py
39 + -62 = -23
39 - -62 = 101
39 x -62 = -2418
39 / -62 = 0
39 % -62 = 39
julien@ubuntu:~/0x18$
```

**Repo:**

- GitHub repository: `alx-low_level_programming`
- Directory: `0x18-dynamic_libraries`
- File: `100-operations.so`

## 3. Code injection: Win the Giga Millions!

#advanced

I bought a ticket for the Giga Millions and chose these numbers: 9, 8, 10, 24, 75 + 9. If you could run two commands on the same server where the Giga Millions program runs, could you make me win the Jackpot?

- Our mole got us a copy of the program, you can download it here (https://github.com/holbertonschool/0x18.c). Our mole also gave us a piece of documentation:

```
/* Giga Millions program
  * Players may pick six numbers from two separate pools of numbers:
  * - five different numbers from 1 to 75 and
  * - one number from 1 to 15
  * You win the jackpot by matching all six winning numbers in a drawing.
  * Your chances to win the jackpot is 1 in 258,890,850
  *
  * usage: ./gm n1 n2 n3 n4 n5 bonus
```

- You can't modify the program `gm` itself as Master Sysadmin Sylvain (MSS) always checks its MD5 (/rltoken/njnwPTMpc1-RSp5sVEAyfg) before running it
- The system is an `Linux Ubuntu 16.04`
- The server has internet access
- Our mole will be only able to run two commands from a shell script, without being detected by MSS
- Your shell script should be maximum 3 lines long. You are not allowed to use `;`, `&&`, `||`, `|`, `` ` `` (it would be detected by MSS), and have a maximum of two commands
- Our mole has only the authorization to upload one file on the server. It will be your shell script
- Our mole will run your shell script this way: `mss@gm_server$ . ./101-make_me_win.sh`
- Our mole will run your shell script from the same directory containing the program `gm`, exactly 98 seconds before MSS runs `gm` with my numbers: `./gm 9 8 10 24 75 9`
- MSS will use the same terminal and session than our mole
- Before running the `gm` program, MSS always check the content of the directory
- MSS always exit after running the program `gm`
- TL;DR; This is what is going to happen

```
mss@gm_server$ . ./101-make_me_win.sh
mss@gm_server$ rm 101-make_me_win.sh
mss@gm_server$ ls -la
. .. gm
mss@gm_server$ history -c
mss@gm_server$ clear
mss@gm_server$ ls -la
. .. gm
mss@gm_server$ md5sum gm
d52e6c18e0723f5b025a75dea19ef365  gm
mss@gm_server$ ./gm 9 8 10 24 75 9
--> Please make me win!
mss@gm_server$ exit
```

Tip: `LD_PRELOAD`

**Repo:**

- GitHub repository: `alx-low_level_programming`
- Directory: `0x18-dynamic_libraries`
- File: `101-make_me_win.sh`

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