



b_ls

As simple as listing the files in a directory.

Gaetan gaetan@42.us.org

Summary: In short: This project will make you recode the command “ls”

Contents

I	Foreword	2
II	Introduction	4
III	Goals	5
IV	General instructions	6
V	Mandatory part	8
VI	Bonus part	9
VII	Turn-in and peer-evaluation	10

Chapter I

Foreword

Here is the recipe for “Choucroute” Sauerkraut Alsatian style:

- Ingredient (4 people)
 - 1kg of sauerkraut (pickled shredded cabbage)
 - 350g of smoked bacon
 - 350g of pig back
 - 1tspofpigfat
 - 2 garlic cloves
 - 1 bay leaf
 - 10 juniper bays
 - 1 onion picked with 2 cloves
 - 25cl of Alsatian white wine “Riesling”
 - 350g of potatoes
 - 4 Frankfurters
- Recipe
 - Rinse the sauerkraut under cold water, press it to remove the water and pour half in a pot.
 - Layer the smoked bacon and the pig back meat, cover them with the rest of the sauerkraut.
 - Add the pig fat, garlic (*unpeeled!), the juniper, bay leaves and the onion.
 - Pour the white wine on top and let it simmer covered for 1h.
 - Add the potatoes and cook an additional 50 minutes.

- Add the sausages and let it cook again for 10 minutes.
- Serve with a reasonable quantity of beer.



This project is easier if you do it after you have already eaten a "Choucroute" Alsatian style.

Chapter II

Introduction

The ls command is one of the first commands you have learned to use with shell. It is also one you are using the most. Perhaps you have already asked yourself how is this function coded? Thanks to this project, you will soon find out.

To recode ls and some of its options will allow you to find out how to interact with the file system using C. After all, you already know how to open, read, write and close a file. But, what about the directories? Special files? Rights, dates or sizes of the files?

Chapter III

Goals

The project b_ls opens the path to the Unix branch of the sphere system. For the first time, you will have to face the one libc functions that will allow you to do other things than just read or write on a file descriptor (this is to simplify of course). You will discover a sub-system of functions of operating system's API, the associated data structures, as well as the management of memory allocation and the associated data.

b_ls is also a great opportunity to think about the structure of your code before you even start writing your code. b_ls' bad reputation is due to students discovering too late in the game that their (lack of) initial design is preventing them from finishing their project without refactorizing a great part of their code. I admit that it can be frustrating...

To conclude, b_ls is another opportunity to add to your libft new practical functions. Browsing a file and identifying directories is quite common in programming. Remember that you will have to do it on many future occasions. Improving your libft today will save you time tomorrow.

Chapter IV

General instructions

- This project will only be evaluated by actual human beings. You are therefore free to organize and name your files as you wish, although you need to respect some requirements listed below.
- The executable file must be named b_ls.
- You must submit a Makefile. That Makefile will have to compile the project and must contain the usual rules. It can only recompile the program if necessary.
- If you are clever, you will use your library for your b_ls, submit also your folder libft including its own Makefile at the root of your repository. Your Makefile will have to compile the library, and then compile your project.
- Your project must be written in accordance with the Norm.
- You have to handle errors in a sensitive manner. In no way can your program quit in an unexpected manner (Segmentation fault, bus error, double free, etc). If you are unsure, handle the errors like ls.
- Your program cannot have memory leaks.
- You'll have to submit at the root of your folder, a file called author containing your username followed by a newline.

```
$>cat -e author  
xlogin$
```

- Within your mandatory part you are allowed to use the following functions:
 - write
 - opendir
 - readdir
 - closedir

- `stat`
- `lstat`
- `getpwuid`
- `getgrgid`
- `listxattr`
- `getxattr`
- `time`
- `ctime`
- `readlink`
- `malloc`
- `free`
- `perror`
- `strerror`
- `exit`
- `printf.`

- You are allowed to use other functions to carry out the bonus part as long as their use is justified during your defence. For example, to use `tcgetattr` is justified in certain case. Be smart!
- You can ask your questions on the forum.

Chapter V

Mandatory part

- You must recode the system's command ls.
- Its behavior must be identical to the original ls command with the following variations:
 - Amongst the numerous options available, we are asking you to create the following: -l, -a, -r and -t.
 - You do not have to deal with the multiple column format for the exit when the option -l isn't in the arguments.
 - You are not required to deal with ACL and extended attributes.
 - The overall display, depending on each option, must stay as identical as possible to the system command. We will be cordial when grading either the padding or the pagination, but no information can be missing.



man ls

Chapter VI

Bonus part

We will look at your bonuses if and only if your mandatory part is EXCELLENT. This means that your must complete the mandatory part, beginning to end, and your error management must be flawless, even in cases of twisted or bad usage. If that's not the case, your bonuses will be totally IGNORED. Find below a few ideas of interesting bonuses you could create. Some could even be useful. You can, of course, invent your own, which will then be evaluated by your evaluators according to their own taste.

- Management of ACL and extended attributes.
- Management of the columns without the option -l. (man 4 tty)
- Management of options -R, -u, -f, -g, -d, ...
- Management of views in colors (Similar to option -G)
- Optimization of your code (What is the response time of your ls on a BIG ls -lR for example?)

Chapter VII

Turn-in and peer-evaluation

Turn your work in using your `GiT` repository, as usual. Only work present on your repository will be graded in defense.