uls

Track C

January 14, 2020

Contents

Engage

DESCRIPTION

Yooo!!!

The `ls` command is one of the important commands that you use the most while working in the shell.
Are you interested to learn about the implementation of this command?

This challenge is developed you to implement your own `uls` :)
An in-depth understanding of this utility will help you learn how the filesystem works in the operating system. Also you'll learn how to interact programmatically with the filesystem using `C`.
Good luck, The One ;-)

BIG IDEA

Data storage, data persistence and data management.

ESSENTIAL QUESTION

How does OS interacts with the filesystems?

CHALLENGE

Recode the system's utility `ls`.

ucode

uls | Track C **> 2**

Investigate

GUIDING QUESTIONS

We invite you to find answers to the following questions. This will help you realize why you are here and what you want to get from it. And most importantly, how to move forward.

Ask your neighbor on the right, left, or behind you, and discuss the following questions together. You can walk around the campus and find yourself a comfortable place to do it.

We encourage you to ask as many personal and contextual questions as possible. Note down your discussion.

- What is the file?
- What is the directory?
- What is device files?
- Why do you need symbolic links?
- What is the filesystem?
- What is the UNIX filesystem?
- Do you familiar with conventional directory layout on MacOS?
- What is command-line flags (options) and why are they useful?
- Have you tried to use all the flags available in `ls`?
- Why does the filesystem require file access control lists (ACL)?
- What is the role of the extended file attributes in the filesystem?

GUIDING ACTIVITIES

These are only a set of example activities and resources. The goal is to develop a solution. Do not forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

1. Meet the team.
2. Read the story to the end and carefully study this task.
3. Read and understand the man of standard `ls` utility. This is your main reference for this challenge.
4. Experiment with standard `ls`. Learn its features. There you'll find out additional information that is not listed in the man.
5. Read the documentation about every component of the solution.
6. Collaborate with other students to investigate challenge in-depth. Learn from others and share your knowledge.
7. Clone your git repository, that is issued on the challenge page.
8. Distribute tasks between all team members.
9. Try to implement thoughts in code.

ucode

uls | Track C **> 3**

10. Check the work and check if it works exactly like a standard `ls` utility.
11. Have fun :)

ANALYSIS

You need to analyze all the collected information before you start.

- Be attentive to all statements of the story.
- Perform only those tasks that are given in the story.
- You can proceed to `Act: Creative` only after you have completed all requirements in `Act: Basic`.
- The challenge must be performed in `C`.
- Your challenge must have the next structure:
  - `src` directory contains source files `.c`;
  - `obj` directory contains object files `.o`. It must be created only during compilation;
  - `inc` directory contains header files `.h`;
  - `libmx` directory contains source files of your library including its `Makefile`. You must use it.
  - `Makefile` that compiles the library `libmx` firstly and then compiles and builds `uls`.
- Makefile must be written in accordance to Auditor.
- You should submit only files required to complete the task in the required directories and nothing else. Garbage shall not pass.
- You should compile C files with clang compiler and use these flags: `-std=c11 -Wall -Wextra -Werror -Wpedantic`.
- Your program must manage memory allocations correctly. Memory which is no longer needed must be released otherwise the task is considered as incomplete.
- Usage of forbidden functions is considered as cheat and your challenge will be failed.
- You must complete tasks according to the rules specified in the `Auditor`.
- Your challenge will be checked and graded by students. The same as you. `Peer-to-Peer (P2P) learning`.
- Also, your challenge will pass automatic evaluation which is called `Oracle`.
- Got a question or you do not understand something? Ask the students or just Google that.
- Use your brain and follow the white rabbit to prove that you are the Chosen one!!!

Act: Basic

ALLOWED FUNCTIONS

`write, malloc, free, perror, strerror, exit, opendir, readdir, closedir, stat, lstat, getpwuid, getgrgid, listxattr, getxattr, time, ctime, readlink, ioctl, isatty, acl_get_file, acl_to_text, acl_free`

BINARY

`uls`

DESCRIPTION

In this part you must recode the system's utility `ls`.

- Usage `usage: uls [-l] [file ...]`.
- Implement a base work of this command. List directory contents without flags.
- Implement processing of `file` operands for files and directories.
- Implement `-l` flag which is the one of the most useful flags.
- Implement the view of extended file attributes and access-control lists (ACL).
- Implement error handling like in original `ls`. Output `uls` as a program name instead of `ls` where it is needed.
- The solution must deal with the multicolumn output format when the option `-l` isn't specified.

CONSOLE OUTPUT

```
>./uls -z | cat -e
uls: illegal option -- z
usage: uls [-l] [file ...]
>./uls xxx
uls: xxx: No such file or directory
>echo $?
1
>./uls
dir1  dir2  dir3  file1 file2 file3 file4 file5
>./uls | cat -e
dir1$
dir2$
dir3$
file1$
file2$
file3$
file4$
file5$
>./uls dir1 dir2 file1 file2 file3
file1 file2 file3

dir1:
A.txt  E.txt  I.txt  M.txt  Q.txt  U.txt  Y.txt
B.txt  F.txt  J.txt  N.txt  R.txt  V.txt  Z.txt
C.txt  G.txt  K.txt  O.txt  S.txt  W.txt
D.txt  H.txt  L.txt  P.txt  T.txt  X.txt

dir2:    # empty directory
>./uls -l | cat -e
total 0
drwxr-xr-x  4 neo  staff  128 May 17 15:15 dir1$
drwxr-xr-x  2 neo  staff   64 May 17 14:57 dir2$
drwxr-xr-x  2 neo  staff   64 May 17 14:57 dir3$
-rw-r--r--  1 neo  staff    0 May 17 14:56 file1$
-rw-r--r--  1 neo  staff    0 May 17 14:56 file2$
-rw-r--r--  1 neo  staff    0 May 17 14:56 file3$
-rw-r--r--  1 neo  staff    0 May 17 14:56 file4$
-rw-r--r--  1 neo  staff    0 May 17 14:56 file5$
>
...    # find out more examples by yourself
```

FOLLOW THE WHITE RABBIT

```
man ls
man 4 tty
```

ucode

uls | Track C **> 6**

Act: Creative

DESCRIPTION

You can proceed to this part only after you have completed all requirements above.

Continue development of `uls` with various flags. The flags you want. It must be your responsible decision which flags to choose from. Of course, you must choose from those flags that are present in the original utility.

You must implement at least 5 flags.
Implement additional flags for higher score!
Add all implemented flags to the usage output as in standard `ls` utility.

Of course we have something for you and we will tell a little more about some flags:

- `-R` - this flag will force you to completely rewrite your code if you do not consider it during the initial development of program architecture. We highly recommend to implement this flag because it will boost your brain as well as it is good evaluated.
- `-a`, `-A` - are interesting flags to implement as standalone as well as in combination with the flag `-R`. Believe us, there is something to pay attention to and you need to think about how to solve these issues!
- `-G` - make you program colored and fancy.
- `-h`, `-@`, `-e`, `-T` etc - flags that are used in combinations with other flags for example with `-l`.
- `-1`, `-C` etc - some simple flags. Note that if you will choose an easy way to pass the challenge, 5 simple flags maybe not enough to validate the challenge with a positive mark.
- `-r`, `-t`, `-u`, `-c`, `-S` etc - various sorting flags. All of them are interesting, but not all give a lot of points. Investigate their behaviour!
- Read the man for additional cool flags.

Do not forget to optimize your algorithms! Compare the speed of execution of your program with the original one. Think about making improvements in your program to speed it up.

Please do not consider adding other advanced features except those flags that original ls provides. That is not the purpose of this challenge.

Share

PUBLISHING

The final important and integral stage of your work is its publishing. This allows you to share your challenges, solutions, and reflections with a local and global audience.

During this stage, you will find how to get a global assessment. You will get representative feedback. As a result, you get the maximum experience from the work you have done.

What you can create to disseminate information

- Text post, summary from reflection.
- Charts, infographics or any other ways to visualize your information.
- Video of your work, reflection video.
- Audio podcast. You can record a story with your experience.
- Photos from ucode with small post.

Example techniques

- Canva - a good way to visualize your data.
- QuickTime - easy way to record your screen, capture video, or record audio.

Example ways to share your experience

- Facebook - create a post that will inspire your friends.
- YouTube - upload a video.
- GitHub - share your solution.
- Telegraph - create a post. This is a good way to share information in a Telegram.
- Instagram - share a photos and stories from ucode. Don't forget to tag us :)

Share what you learned with your local community and the world. Use #ucode and #CBLWorld on social media.