Preferred Networks Intern Screening 2017 Coding Task for Back-end Development

Changelog

- 2017-5-12 : Initial version

Notice

- Please use one of the following programming languages:
- C, C++, Python, Ruby, Go, Java
- Please use only standard libraries.
- It is not allowed to call (existing) `ls` command or other programs by using methods or system calls such as `exec`, `system`, etc.
- You do not need to implement behavior for all possible corner cases, but it is preferable to describe such cases in the report.
- Execution environment is assumed as the following Linux distribution and CPU:
- Ubuntu 16.04/x86_64

Things to submit

- Please submit the code for problems and a report describing how to run the code (e.g., execution environment, compiler, and other necessary steps).
- Both the code and the report can either be divided into separate files or not.
- Please include following description in your report.
- What functionality is not implemented or which corner cases did you not cover? ("Where does your program show a different result?")
- What did you find difficult?
- Please write your code so that it will be easy to read for other people. Also, please make sure that it is easy to re-run your program. As for the report, please make sure that it is concise and easy to understand.
- Please make execution of the code reproducible.
- Please name the programming language with version used to develop the program in your report.
- Please include commands such as Makefile neccessary to build and run the program.

How to submit

Submit files should be uploaded to Google drive. After uploading them, please fill in the share URL on the submission form. See the following URL for how to

upload files to Google drive:

- Submission form: https://docs.google.com/a/preferred.jp/forms/d/e/1FAIpQLSd_zC_XT2dHM-yRO9WQ-YuRU0sx2HeQIep-NBoqMWpN_j8KNw/viewform
- How to upload files : https://www.preferred-networks.jp/wp-content/uploads/2017/04/intern2017_GoogleUpload_3.pdf

Inquiry

If you have any questions about the problems, please also send these to intern2017@preferred.jp (the same address as used for applying to the internship)

Task description

Task 1

In this task, you will implement a `myls` command similar to the `ls` command provided by Linux coreutils. Options to be implemented for the `myls` command are shown below. In short, running `myls` with `-al` option has to display the same information as coreutils `ls`.

c.f., http://unix.stackexchange.com/questions/103114/what-do-the-fields-in-ls-al-output-mean

- Options to be implemented
- `-a`
- `-l`
- `-al` or `-la` or `-a -l` (these options interpret both of `-a` and `-l`.)
- Notes
- When listing directory contents, do not display any files or directories whose names start with dot (`.`) unless the `-a` option is given.
- Symbolic links and hard links should be displayed as in coreutils `ls`.
- When multiple files and directories are given as parameters, display all of them.
- Displays an error message if files or directories with given names do not exist.
- The display uses English.
- This need not have exactly the same behaviors, error messages, etc. as the existing `ls` of coretuils, and error handlings also need not be complete. You do not have to read the code of coreutils.
  - You also need not consider return codes of errors or successes.
- You do not need to use colored highlights depending on terminals.
- You do not need to consider the width of users' terminals (because it can take much time and energy).
- To check your implementation, consider the commands below and compare them with those of `myls`.

- `ls`
- `ls -la .`
- `ls -la ..`
- `ls -la /proc/self`
- `ls -la /proc/self/`

Task 2

Pick two options explained in the `ls` manpage (except `--version` and `--help`) and add them to the `myls` command implemented in Task 1.

For example, these options could be

- `-R`
- `-s`
- `-S`

If there is not enough time to complete the tasks, please submit code and report as far as you were able to implement/write them.