Assignment 4: Unix ls (LiSt) command[1]

Notes

- This assignment is to be done **individually or in pairs**. Do not share your code or solution, do not copy code found online, ask all questions on Piazza discussion forum (see webpage).
  - You may use code provide by the instructor, or in the guides/videos provided by the instructor (such as instructor's YouTube videos).
  - You may follow any guides you like online as long as it is providing you information on how to solve the problem, as opposed to a solution to the assignment.
  - If receiving help from someone, they must be helping you debug **your** code, not sharing their code or writing it for you.
  - We will be carefully analyzing the code submitted to look for signs of plagiarism so please don't do it! If you are unsure about what is allowed please talk to an instructor.
- You may not resubmit code you have previously submitted for any course or offering.
- Instructors and TAs may conduct followup interviews with students to discuss their implementation and design in order to verify the originality of the assignment.

1. Implement the Unix `ls` command

The Unix/Linux ls command is used to list files and directories in the file system. For this assignment you will be implementing a version of ls which supports only a limited set of options. When submitting, ensure your code compiles and executes correctly on CSIL Linux machines.

By completing this assignment you will:

1. Understand what an *inode* is, and understand its role in the Unix file system.
2. Know how to use system calls to navigate the Unix file system from a user-level program.
3. Understand how files and directories are organized and stored in Unix.

1.1 Requirements

- Create a program named **`myls`** (short for "*MY LiSt*")

  When calling the program, it will take the form:
  `./myls [options] [file list]`

  - `[options]`: Optional options[2] from the list below. May be specified in any order or grouping, such as "" (none), "`-i`", "`-i -l -R`", "`-iRl`" "`-R -li`", "`-lR`"…
  - `[file list]`: Optional space separated list of paths to files or directories to displayed.
- Implement the following options (you need *not* support any other options, and you need *not* support the long version of the option names):
  - `-i`: Print the index number of each file
  - `-l`: Use a long listing format
  - `-R`: List subdirectories recursively

1 Based on an assignment created by Harinder Khangura.
2 No, that's not a typo. :)

- **Sort**
  Sort the files and directories lexicographically.
  - When printing directories recursively (-R option), do a depth-first traversal of the directories, entering sibling directories in lexicographical order.
- **Date Format**
  - When using the -l option you must print out date information in the following format:
    `mmm dd yyyy hh:mm`

    For example:
    ```
    Oct  2 2021 14:52
    Sep 30 2021 00:01
    ```
  - Specifically:

`mmm`	Initial three characters of month name; first letter capitalized.
`dd`	Two digit day of month; pad with space (‘ ‘) for single digit day.
`yyyy`	Four digit year number; pad with spaces (‘ ‘) if less than 4 digits.
`hh`	Two digit hour (24 hour clock); pad with zero (‘0’) if less than 2 digits.
`mm`	Two digit minute; pad with zero (‘0’) if less than 2 digits.
- **Special Paths**
  You must support the following paths with special meaning (`x` and `y` are arbitrary names):
  - `/` Root directory (when at the front of path of the path such as `/usr`)
  - `/x/y` Absolute path to a file or directory
  - `/xy/` Absolute path to a directory
  - `x/y` Relative path to a file or directory
  - `x/y/` Relative path to a directory
  - `~` User’s home folder (can be used such as `~` or `~/test/this/out`)
  - `.` Current directory
  - `..` Up one directory (can be used such as `../` or `../../oh/my/../really/..`)
  - `*` Wildcard (such as *.c); most of this work will be done by the shell!
- **Simplifications vs built-in `ls` command**
  - When *not* using `-l`, do not display output in multiple columns. i.e., just display a single column of file/directory names.
  - Do not print the “`Total 123456`” line at the top of the output when using the `-l` option.
  - All options (if any) must be specified before any files/directories (if any).
  - Date format as specified above.[1]

1 The real ls command omits the year of the date is less than a year in the past and replaces the time with the year if more than a year in the past.

- You do *not* need to support quoted strings as arguments:
  ```
  $ myls "thank goodness we don't/have to support this/"
  $ myls 'it is like a "gift" not to do this!.txt'
  $ myls 'seemed like a good idea.c'
  ```

- **Error Handling**
  You must display a meaningful error and exit when:
  - User specified an unsupported option such as `-a`. It is undefined behaviour (do anything you like) if the user specifies an argument such as "`--version`".
  - User specified a nonexistent file/directory.
    If the user provided multiple files/directories you may either:
    - print just an error message and exit;
    - print the good files/folders until the bad one, print an error message and then exit; or
    - print all the good files/folders, plus error messages for each bad one, in order (actual `ls` behaviour).

1.2 Testing

- Use the built-in `ls` command to compare your output to a working implementation. To get the `ls` output into a single column, use (a "–one" as the option):
  ```
  $ ls -1
  ```
- Test all permutations of the possible options, including no options at all.
- Test listing edge cases:
  - on a file; on an empty directory; on a very large directory
  - `ls` on different file types (normal, symbolic)
- Many Linux/Unix systems create an automatic alias for ls so that it automatically uses some options.
  - To see your current alias (if any) run:
    ```
    $ alias ls
    ```
  - To temporarily clear your alias for the current terminal, run:
    ```
    $ unalias ls
    ```
- Example commands (numbered for easy reference)

```
1. $ myls
2. $ myls -lR
3. $ myls /
4. $ myls ~
5. $ myls ~/
6. $ myls / /usr
7. $ myls -R -il ~/cmpt300 ~/tmp /usr
8. $ myls ..
9. $ myls -R ../hello
```

1.3 Hints

- Initially concentrate on getting the listing for a directory correct; later worry about recursion.
- Focus on extracting the proper information and just printing it. Once you have that all done work on making the format match the `ls` command.
- Consult the provided file `infodemo.c` to see what calls to use to get actual group and user names.
- Consult the tutorials on the assignments page of the course website for information on UNIX file system structure.
- Don't forget to test your code on directories with symbolic links in them.

1.4 Restrictions

- You may *not* use 3rd party libraries. For example, you must write the path processing, file listing, and directory recursion code yourself.
- You may not copy other peoples (or open source) code.
- Your code should make use of a modular design; try breaking your code into meaningful modules and reasonable functions!
- Your code must cleanly pass execution under Valgrind.
- You must implement sorting yourself (any $O(n^2)$ or better algorithm OK).
  You may not copy-and-paste sorting code from another sort.

2. Deliverables

In CourSys, create a group for your submission. Even if you worked on your own, you need to be in a group (of 1 in this case) in order to submit. Submit the following to CourSys:

1. `as4.tar.gz:` Zip file of your project folder

   Required files:
   - `makefile`: Should have a default target to build all, and a `clean` target to remove build products.
   - your `.h` / `.c` files

   To create this archive, from inside your source code folder, execute:

```
$ make clean
$ tar -czvf as4.tar.gz *
```

Please remember that all submissions will automatically be compared for unexplainable similarities.