CMPT 300 D200
Assignment 4
myls command
Marks: 100 marks

Notes:

1. You can do this assignment individually or in a team of two. If you are doing it in a group, then only one submission per group is required.
2. You may submit multiple times until the deadline. Grade penalties will be imposed for late submissions (see the course outline for the details).
3. Always plan before coding.
4. All the codes in this lab must be done using C language only. No other languages should not be used.
5. Use function-level and inline comments throughout your code. We will not be specifically grading documentation. However, remember that you will not be able to comment on your code unless sufficiently documented. Take the time to document your code as you develop it properly.
6. We will be carefully analyzing the code submitted to look for plagiarism signs, so please do not do it! If you are unsure about what is allowed, please talk to an instructor or a TA.

Background

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.

Note: Mac-OS's ls is different; must develop under Linux.

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.

Ques 1. Create a program named **myls** (short for *"MY List"*)

1.1 Requirements

Your program should support below mentioned functionalities.

When calling the program, it will take the form:

```
./myls [options] [file list]
```

- `[options]`: Optional options 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.

Based on assignment created by Dr. Brian Fraser

1. **Options:** 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

2. **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.

3. **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.

4. **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!

Based on assignment created by Dr. Brian Fraser

5. **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).
   - 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'
     ```

6. **Error Handling**
   You must display a meaningful error and exit when:
   - User specified an unsupported option such as -a. It is undefined behavior (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;
     - print all the good files/folders, plus error messages for each bad one, in order (actual ls behavior).

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
```

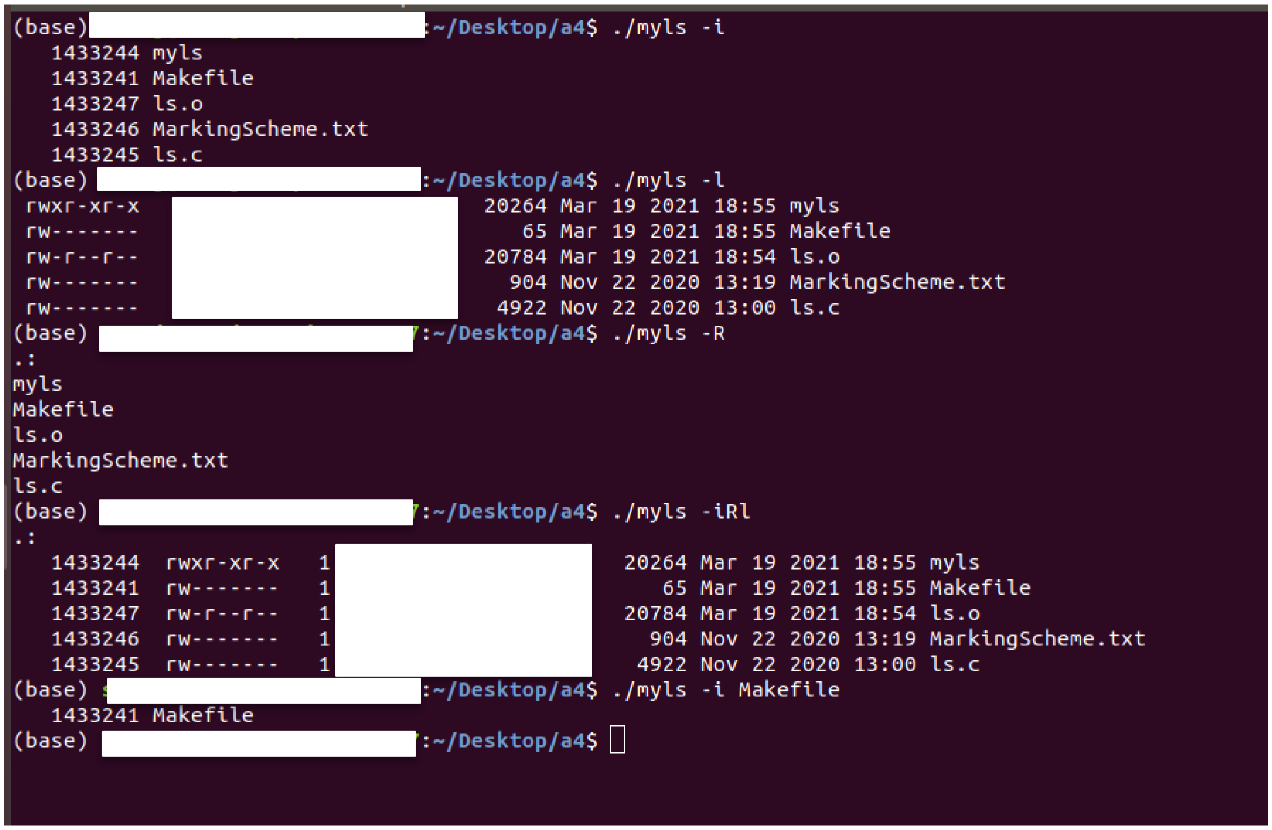
1.3 Hints

- 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!
- You must implement sorting yourself (any *O(n²)* or better algorithm OK).
  You may not copy-and-paste sorting code from another sort.

Sample output:



```
(base) [redacted]:~/Desktop/a4$ ./myls -i
   1433244 myls
   1433241 Makefile
   1433247 ls.o
   1433246 MarkingScheme.txt
   1433245 ls.c
(base) [redacted]:~/Desktop/a4$ ./myls -l
 rwxr-xr-x  [redacted]  20264 Mar 19 2021 18:55 myls
 rw-------                 65 Mar 19 2021 18:55 Makefile
 rw-r--r--              20784 Mar 19 2021 18:54 ls.o
 rw-------                904 Nov 22 2020 13:19 MarkingScheme.txt
 rw-------               4922 Nov 22 2020 13:00 ls.c
(base) [redacted]:~/Desktop/a4$ ./myls -R
.:
myls
Makefile
ls.o
MarkingScheme.txt
ls.c
(base) [redacted]:~/Desktop/a4$ ./myls -iRl
.:
   1433244  rwxr-xr-x   1 [redacted]  20264 Mar 19 2021 18:55 myls
   1433241  rw-------   1                65 Mar 19 2021 18:55 Makefile
   1433247  rw-r--r--   1             20784 Mar 19 2021 18:54 ls.o
   1433246  rw-------   1               904 Nov 22 2020 13:19 MarkingScheme.txt
   1433245  rw-------   1              4922 Nov 22 2020 13:00 ls.c
(base) [redacted]:~/Desktop/a4$ ./myls -i Makefile
   1433241 Makefile
(base) [redacted]:~/Desktop/a4$ 
```

Question 2. [2 marks]

Please specify how long you spend on it and what you learned. Any extra feature you would like to add to the myls? (This question is worth marks, so please do it!)

Test Cases

- Run the executable with all the combinations of iRl (-i, -l, -R, -lR, -iR, -il, -iRl) including without passing an of the options
- Run the executable by passing a file or directory along with the (iRl) options
- Place a subdirectory (with a few files) inside your original directory and test if the iRl options are working fine for the files inside the subdirectory.

Submission Instructions:

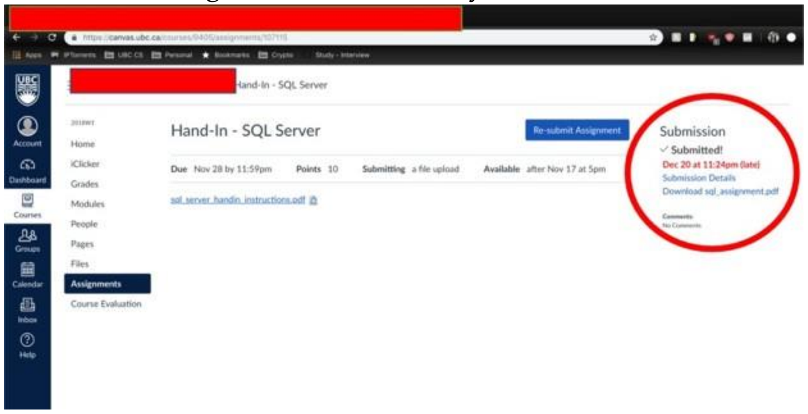
Submit a ZIP file of your assignment folder. Your submission is composed of the following

- **Source code**
    - Include your source code file(s) (.h and .c)

- **Makefile**: the Makefile provides the following functionality:
    - **all**: compiles your program (this is the default behavior), producing an executable file named same as the C file.
    - **clean**: deletes the executable file and any intermediate files (.o, specifically)

- **README.txt**: documentation
    - Include your name (and team member name) and assignment number at the top of the file
    - Write down any resources that you consulted (URLs).
    - Write down the name of the class members with whom you have discussed your solution.
    - Mention the claimed late days for the assignment.
    - The answer to Question 2.

After submitting the file on canvas, you should be able to see the below message:



[redacted]

https://canvas.ubc.ca/courses/9405/assignments/107115

[redacted] Hand-In - SQL Server

Hand-In - SQL Server

Due Nov 28 by 11:59pm Points 10 Submitting a file upload Available after Nov 17 at 5pm

sql_server_handin_instructions.pdf

Submission
✓ Submitted!
Dec 20 at 11:24pm (late)
Submission Details
Download sql_assignment.pdf
Comments
No Comments

Grading Criteria

- Correctness (96%): passes all of the (hidden) unit tests
- Makefile (2%): satisfies the make file requirements
- README.txt (2%): contains the required information

Based on assignment created by Dr. Brian Fraser