CMP SCI 4760 **Operating Systems**
Assignment # 1 Due Date: September 14, 2020

Important: Please do all assignments on `hoare`

Linux System Calls and Library Functions

Purpose

The goal of this homework is to become familiar with the environment in hoare while practising system calls. You will also demonstrate your proficiency in the use of `perror` and `getopt` in this submission.

Task

Implement the task specified in Exercise 5.39 (p. 180) in your text by Robbins/Robbins. Do only questions 1–4 in the assignment, that is, you do not have to do `breadthfirstapply`. Your executable will be called `mydu`.

This exercise deals with the implementation of the Unix application known as `du`. The command `du` displays the size of subdirectories of the tree rooted at the directories/files specified on the command-line arguments. If called with no argument, the `du` utility uses the current working directory. Experiment with the `du` command on `hoare` and also read its man page.

Invoking the solution

Your solution will be invoked using the following command:

```
mydu [-h]
mydu [-a] [-B M | -b | -m] [-c] [-d N] [-H] [-L] [-s] <dir1> <dir2> ...
```

-a Write count for all files, not just directories.

-B M Scale sizes by `M` before printing; for example, `-BM` prints size in units of 1,048,576 bytes.

-b Print size in bytes.

-c Print a grand total.

-d N Print the total for a directory only if it is `N` or fewer levels below the command line argument.

-h Print a help message or usage, and exit

-H Human readable; print size in human readable format, for example, 1K, 234M, 2G.

-L Dereference all symbolic links. By default, you will not dereference symbolic links.

-m Same as `-B 1048576`.

-s Display only a total for each argument.

When you use `perror`, please print some meaningful error messages. The format for error messages should be:

```
mydu: Error: Detailed error message
```

where `mydu` is actually the name of the executable (`argv[0]`) and should be appropriately modified if the name of executable is changed without a need for recompilation.

It is required for this project that you use version control, a `Makefile`, and a `README`. Your `README` file should consist, at a minimum, of a description of how I should compile and run your project, any outstanding problems that it still has, and any problems you encountered. Your `Makefile` should use suffix-rules or pattern-rules and have an option to clean up object files.

Suggested implementation steps

1. Set up your git account, if you have not already done so. You must periodically check your code into the git repository.
2. Create your Makefile.
3. Write code to parse options and receive the command parameters. Study `getopt(3)`, if you do not know how to do it. The page also has an example to guide you.
4. Follow steps 1 to 4 in the exercise in the book.
5. Create the `README` file.

Criteria for success

You are free to use your own data structures and algorithms to perform the depth-first search. Your code should give proper results if the directory is empty, or if the directory does not contain any more subdirectories. Your code should properly account for the file size, possibly by using `stat(2)` family of functions. *Do not use `system(3)` function for anything*.

Grading

1. *Overall submission: 25 pts*. Program compiles and upon reading, seems to be able to solve the assigned problem.
2. *Command line parsing: 10 pts*. Program is able to parse the command line appropriately, assigning defaults as needed; issues help if needed.
3. *Use of perror: 5 pts*. Program outputs appropriate error messages, making use of `perror(3)`.
4. *Makefile: 5 pts*. Must use suffix rules or pattern rules. You'll receive only 2 points for Makefile without those rules.
5. *README: 5 pts*. Must address any special things you did, or if you missed anything.
6. *Conformance to specifications: 40 pts*. Program works correctly and meets all of the specifications. Each option will be worth 2 points.
7. *Code readability: 10 pts*. The code must be readable, with appropriate comments. Author and date should be identified.

Submission

Create your programs in a directory called *username*.1 where *username* is your user name on hoare. Once you are done with developing and debugging, *remove the executables and object files*, and issue the following commands:

```
% cd
% chmod 755 ~
% ~sanjiv/bin/handin cs4760 1
% chmod 700 ~
```

Do not copy and paste those commands from the PDF *of the assignment. Type in the commands.*

Do not forget `Makefile` (with suffix or pattern rules), your versioning files, and `README` for the assignment. If you do not use version control, you will lose 10 points. I want to see the log of how the program files are modified. Therefore, you should use some logging mechanism and let me know about it in your `README`. You must check in the files at least once a day while you are working on them. Omission of a `Makefile` (with suffix rules) will result in a loss of another 10 points, while `README` will cost you 5 points. I do not like to see any extensions on `Makefile` and `README` files.

Before the final submission, perform a `make clean` and keep the latest source checked out in your directory.

You do not have to hand in a hard copy of the project. Assignment is due by 11:59pm on the due date.