Problem

My hard drive filled up today at home. I need to quickly purge some crap before my OS crashes.

Then in the office I see the NAS sitting under my desk has the red message "exceeding threshold".

Solution

To get the biggest bang for the buck, your app needs to **identify** the biggest offenders, which could be individual files or entire directory subtrees. For example, a single mp4 file of a movie could be 5 GB. A directory of mp3 files of songs could be 6 GB even though no single mp3 is more than 50 MB.

⚠YOUR APP MUST NOT DELETE ANY FILE. It should just identify candidates the user can choose to manually delete.

Requirements

Print the directory tree with sizes, including the size of directories.

Help Text (help.txt)

Usage: ***node bigstuff*** *[arguments]*

-p, --path *value, the relative or absolute path name of a file or directory. Default is -p .*

-s, --sort *alpha | exten | size*

alpha *sorts alphabetically (ascending)*

exten *sorts by extension alphabetically (ascending)*

size *sorts by size (descending)*

Default is unsorted

-m, --metric*, sizes displayed as KB, MB, GB, and TB instead of bytes.*

-t, --threshold *min, only displays files and folders of at least minimum size.*

-b, --blocksize*, displays the actual storage sizes on disk.*

-h, --help *prints this usage screen. Ignores all other arguments.*

More Details

- Start with a subdir.
- File sizes must display commas as needed, e.g. 124,000,000 (not 124000000).
- Directory names are displayed with a trailing **/**.
- Directory and files names use relative, not absolute paths.
- Only allowed native modules are **fs**.
- Required to use **filesize** for metric. It is the only allowed 3rd party module.
- Use **console.group** for indentation.
- Use only fs synchronous calls.
- Copy the Help Text above into **help.txt**. Your code will read and display from help.txt as needed.

Out of Scope

- No Sad Path
- Don't start at root of a drive
- No Globbing
- No RegEx
- No hidden/system/linked; just normal files and directories
- No interactivity

FAQ

Q1. Can I do this in loops? Do I need recursion?
A1. Recursion can do everything iteration does. Iteration can *mostly* do what recursion does. We think of them interchangeably, and consequently as a personal preference, because mostly our code works in one direction. What if we need to work in the opposite direction?

see https://repl.it/@10248629/PossiblePeriodicGame#index.js

Walking trees (like the file system) is miserable with iteration. You end up cheating by using your own stack instead of the system stack. DON'T. Learn recursion. You need its power.

Q2. Why do I need an intermediate data structure? Why can't I print it in one pass as I'm traversing?
A2. Think of your starting directory, the root of your subtree. It's always the first line in the answer, displaying the sum size of all the files in its subtree. How can you print the sum before you have all the file sizes to add together to get the sum?

Q3. I read that async is always a best practice in JavaScript, and for node, in particular.

A3. Generally, not always. "Best Practice" is not a best practice in every circumstance.

We'll watch a short video in class that clarifies the proper (and improper) use of "Best Practices".

If you're asked why you did something and you can't explain the context, i.e. the WHY behind a best practice, saying "It's a best practice" is not justification; it's a dogmatic statement.

Q9. OK, so what's wrong with doing it asynchronously?
A9. It'll work. It's just not necessary and not helpful for this one case, it'll very likely confuse you by adding work.

Q10. I don't understand. Directories don't have sizes. How can the program show a size for each directory?
A10. You're right. The OS doesn't store a size for a directory. You have to calculate it by summing the children sizes. The Solution section and Q2/A2 mention it.

Q11. So with -b or --blocks, we need to display how many blocks?
A11. No, the display number is always an integer or a metric approximation. Let's look at a single file of one byte.

defaults	**-m**	**-b**	**-b -m**
1	1*	4,096	4KB*
4,100	4.1KB*	8,192	8KB*

*whatever **filesize** module gives you

Q12. fs.statSync(path) gives me *size*, *blksize*, and *blocks*. Do I have to do my own block calculations?
A12. statSync's blksize and blocks are not what you think they are. DON'T USE THEM.

Q13. So we should start with a binary tree?
A13. No. I started with it in class to remind you of what you already knew (or should have known). I showed how a multiway tree is almost identical to a binary tree. Instead of *left* and *right*, you just use *children*.

Q14. What happened to Q4 to Q8?
A14. They rambled and they offended some students.

Q15. Do I have to use the filesize module? Can't I do that code myself?
A15. Yes, you must use **filesize**. Even if you do the code correct for this project, something is coming up for which I promise your code will fail.

Q16. Can I use path module? It's included in node.
A16. No. You don't need it. Always use "/" for the path separator.

Q17. Isn't the path separator in Windows "\"?
A17. Windows also accepts "/". Always use "/".