Assumptions - General

- If using tests in a different project, place the folder *testFiles* in the root folder in your repository.

Assumptions - Basic Functionality (BF)

echo - Following the linux shell implementation, the echo ignores all paired quotes (double or single). Eg. echo 1”abc”2 will display 1abc2.

If the echo arguments contain quotes, they must be in pairs, else an exception will be thrown.

Multiple adjacent spaces will be merged into one space unless if between quotes

exit - N.A.

paste - If any of the files are longer than the others, there will still be tabs to separate the remaining lines of the other ended files.

If a “-” is used as argument, they must have either pipes or IO-direction as a source of stdin to be read by the app.

There must be at least one argument, or an exception will be thrown.

rm - There must be at least one argument, or an exception will be thrown.

rm -r and rm -r -d are equivalent. Meaning, rm -r -d test.txt is perfectly valid even though “test.txt” is not a directory.

In order for deletion to occur, there must be both execute and write permission for the parent directory of file/directory.

If there is no execute permission on a folder, the shell program cannot open the folder, as is the case on Linux terminals. Hence, when deleting files in a directory which has no execute permission, “File not found” error message is printed.

If there are invalid files/directories among the list of files/directories, an error message will be printed, and the valid files/directories will be deleted.

sed - The replacement index has to be an integer more than 0.

Each of the arguments must be non-empty, except the replacement index, which will then have the default value 1.

If a special character is to be replaced, it has to be preceded by a backslash, i.e to replace $, use \$.

Double quotes cannot be replaced as it is resolved and removed by the parser.

call - N.A.

pipe - When a call command pipes into a call command that doesn't take input from stdin, no errors and execute as usual. Only the result of the last command will be displayed though.

When a call command that doesn't write to stdout pipes into a call command that does take input from stdin, the previous command is executed but the second will not.

sequence - If any of the commands raise exceptions, the other commands will not be affected and run as normal.

Assumptions - Extended Functionality 1 (EF1)

cd - Users are allowed to put more than 1 args. Similar to Linux shell implementation, it will cd according to the 1st args. Eg. "cd test result" will act as "cd test"

There must be at least one argument, or an exception will be thrown.

cp - If the destination doesn't exist, similar to Linux shell implementation, treat the destination argument as a file and create it. If that destination file specified is in a non-existent directory, throw error.

Examples where there is no directory/file named "nonexistant":

cp 1.txt nonexistant

creates new file "nonexistant"

cp 1.txt nonexistant/

"nonexistant/" is treated as "nonexistant". This behaviour is caused by Path.resolve. Creates new file "nonexistant"

cp 1.txt nonexistant/2.txt

Throws an error for no such directory

diff - cannot diff - -, diff dir file

Assume we have read permission on all files

No recursive diff of directories (eg. directories inside directories)

When using diff on two directories, files and directories will be sorted according to alphabetical order before being compared. Then, only the file/directory names will be compared. (The handout appears to imply this as well.) If the files have the same name, then it will compare their contents. For directories, we do not recursively check its contents. Instead, we only compare their names.

When using diff on two files, only identify the files as different if any of them have unique lines. So if two files have the same lines in different order, they are considered identical.

Duplicate lines do not affect the diff result. I.e. if a file contains 10 lines of "1" and another file contains just one line of "1", they are treated as the same as the line containing "1" is found in both files and is not unique to either file.

Assume all text files must end with ".txt", else treat as a binary file, which will always be treated with isSimple flag and won't print the differing lines.

Output of different files are of the format: (no space after ">"/"<")

<line1contentA

<line2contentA

>line1contentB

grep - N.A.

wc - if more than one arguments appear, it will display in the order: lines, words, bytes, no matter the order of the flags

globbing - Uses asterisk to form regex arguments

If there is no match to the regex or folder does not exist or is empty, the original ".../*" is returned.

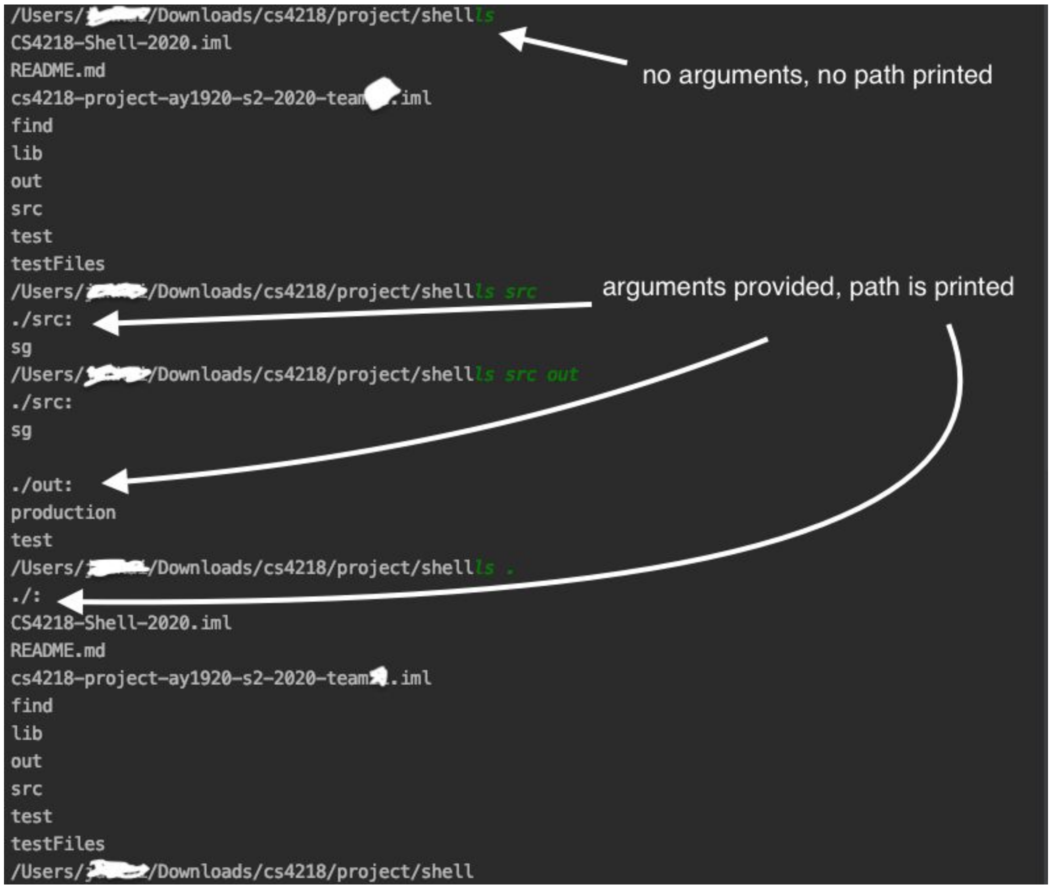
quoting - N.A

Assumptions - Extended Functionality 2 (EF2)

ls - If both recursive mode (-R) and folders-only mode (-d) are provided together in the command, folders-only mode will take priority and recursive mode will be ignored (similar to OSX).

Invalid path arguments such as those with special characters (i.e folder name "#$") will work if these paths actually exist. We assume they are illegal file/folder names that should not exist.

For given directory path arguments, the target directory path will be listed prior to the listing of its contents. This does not apply if no arguments are provided (e.g. the command “ls”). This is clarified to be acceptable.



```
/Users/[redacted]/Downloads/cs4218/project/shellls
CS4218-Shell-2020.iml
README.md
cs4218-project-ay1920-s2-2020-team[redacted].iml
find
lib
out
src
test
testFiles
/Users/[redacted]/Downloads/cs4218/project/shellls src
./src:
sg
/Users/[redacted]/Downloads/cs4218/project/shellls src out
./src:
sg

./out:
production
test
/Users/[redacted]/Downloads/cs4218/project/shellls .
./:
CS4218-Shell-2020.iml
README.md
cs4218-project-ay1920-s2-2020-team[redacted].iml
find
lib
out
src
test
testFiles
/Users/[redacted]/Downloads/cs4218/project/shell
```

no arguments, no path printed
arguments provided, path is printed

Figure 1: printed directory paths for ls

sort - Emulates sort in OSX terminal. In case-independent sorting (-f), comparison of equivalent converted lowercase strings will yield to their original order. For example, if “a” precedes “A” in the file, a case-independent sorting will still follow this order despite the fact that “A” should precede “a” in a non case-independent sort.

In numerical sorting (-n), the first word in a line is treated as the element for comparison in for a numerical sort. The first word here is defined as the first retrievable numerical string chunk from the string. A numerical chunk is defined as a real number (negative inclusive) in decimal form. If the numerical chunk is padded with 0s in the front, the zeros will be ignored (i.e 0099 is the same as 99). If there are no valid numerical chunks retrievable, it will be treated as a regular string. If the compared numerical chunks are equal, its remaining constituents will be recursively numerically compared.

The natural order of numerical sorting utilises the following priority (from smallest to largest), similar to numerical sorting in UNIX based shells:

1. Negative real numbers until -1
2. Non alphabetic characters (not numerical)
3. 0
4. Capitalised alphabetic characters (not numerical)
5. Non-capitalised alphabetic characters (not numerical)
6. Real numbers from 1 onwards

find - N.A.

mv - *mvSrcFileToDestFile* and *mvFilesToFolder* do not throw exceptions but instead returns any exceptions thrown in String format.

When no overwrite is used (-n), attempting to overwrite a file/folder with the same name will not yield any output or error (similar to OSX).

cut -

Index of character starts from 1.

If LIST is a list of a single number or 2 comma separated numbers

- Returns character at position of number or 2 characters at positions of 2 numbers
- If number or numbers passed in are larger than the length of the input string, an empty character is returned "" for that position

If LIST is a range of a 2 numbers:

- Returns substring from 1st number to 2nd number, inclusive
- if numbers passed in are longer than the length of the input string, an empty character is returned ""

cannot use stdin as input i.e cannot pass in '-' as input. Index of character starts from 1, 0 and negative numbers will throw errors.

IO-Redirection

Only one file can be specified per redirect argument

Eg. `command < [only one file argument] > [only one file argument]

Command Substitution -

Can only run one nested command per command string input

PMD Violations

- ApplicationRunner (ExcessiveMethodLength) - Long switch case for command handling
- EnvironmentUtils (ExcessiveMethodLength) - Long proper names to identify directories
- MvApplicationTest (ExcessiveMethodLength) - Long proper names to identify paths
- SortApplication:57 (CloseResource) - handled close resource but error persists
- SedApplicationTest, RmApplicationTest - Long proper names to identify type of test case
- ShortVariable

For excessive method lengths, it can be difficult to balance the length of the method and the readability of the code. We are advised to use descriptive names but at times, with the ExcessiveMethodLength constraint enforced by the PMD rules, it can be very difficult to do so. Hence, in these cases where the PMD rules reduce readability of method names, we have decided to favour readability over complying with those rules. For ShortVariable names, even if those variables were given more descriptive names, it would not add much value as they are dummy variables.