CS4218 Milestone 2 Assumptions

This document is hosted on HackMD:
https://hackmd.io/s/HJ3VqXrcz#

General

1. If there is an argument that is invalid for a command, an exception will be thrown and the command will not continue execution even if other arguments are valid unless otherwise stated in the application's assumptions.

2. Failed commands (eg. cat on non-existent file) will not terminate the shell; can continue to execute new commands.

3. File and folder names should not contain any new line characters and spaces unless enclosed by quotes.

Shell

Calling applications

1. Calling a non-existent commands will not terminate the app; new commands can be executed after pressing return key.

IO-redirection

1. Specifying multiple files for redirecting to input or output is done by using two or more instances of the same redirection operator eg. `cat < text1.txt < text2.txt` which will throw an exception

Quoting

1. Quotes and backticks in arguments must come in pairs unless arguments are enclosed by appropriate quotes. Otherwise, unexpected result or ShellException will be thrown.
2. Evaluating arguments with spaces in quoted variable will be treated as single argument. For example:

```
# cat will attempt to find file named: 'file1 file2' (including space)
cat `echo file1 file2`

# cat will read 'file1' and 'file2' separately
cat `echo file1` `echo file2`
```

Semicolon

1. We allow trailing `semicolon` even if there is no command follows it. For example the following are both valid commands:

```
# explicit semicolon
echo hello world;

# implicit semicolon
echo hello world
```

2. The `semicolon` will adhere to the default `quoting` rules. For example:

```
# will output: `echo hello; echo world`
echo '`echo hello; echo world`'

# will output: helloworld
echo `echo hello; echo world`
echo "`echo hello; echo world`"
```

Applications

ls

1. Options -d and -R can be used within the same execution, which will return only directories and subdirectories recursively

2. Options -d and -R cannot used as a single command line word (e.g -dR)

3. The order and the number of times options -d and -R appear does not matter

### cat

1. Cat will throw an exception for files that cannot be found but will continue execution until all arguments have been evaluated

2. Ignore all directory. This applies to globing whereas only files are displayed without

throwing exception.

echo

1. Only accepts input from arguments

exit

1. Exception is never thrown in System.exit
2. Use `exit` (lowercase) instead of `Exit` (likely a mistake in pdf specification)

mkdir

1. When an exception is thrown during the creation of folders, the command will terminate.

grep

1. The grep application can use single asterisk (*) as defined in specs. Client can technically make use of double asterisk (**) for deep recursve and single character matching (?), but the latter two are in beta and might throw exception.
2. When a base directory reference is not explicitly stated (eg: absolute path) then current directory will be assumed using dot syntax (.).

paste

1. If no files are specified, use stdin. (in project description)
2. When output is redirected, the file will end with a system-dependent line separator
3. Due to 1, mergeFileAndStdin method is left unimplemented and never called.

diff

1. If more than 2 files are specified, only the first 2 is taken and the rest are ignored.
2. Stdin will only be read if ‘-’ is specified in the input argument.
3. Duplicate options will be ignored.
4. Options and the 2 filenames can be specified in any order.
5. Diff on directory will not guarantee output in alphabetical order.
6. Simple diff on binary files are not guaranteed, but works most of the time.

cd

1. throws exception if path provided does not exist or is a file

sed

1. Regex cannot use operators that is used by shell such as ‘<’ and ‘|’
2. If replacement index is 0 then it does not replace anything.
3. If regex match occurence is less than replacement index then does not replace anything.
4. If replacement index is an invalid character or < 0, it will throw exception
5. Throws exception if input file is a directory or does not exist.
6. Throws exception when receive 2 input file as argument.

split

1. `split` uses Bijective base 26 algorithm to construct the filenames. This is different from the given specs because our implementation it will prepend a tilde `~` if there are more than 676 files created, as appose to appending `z`. This is because alphabet `z` is part of the enumeration. The tilde `~` is lower order than `z` in ascii table so the topological order will still be preserved.
2. The number of `~` prepended uses the following formula:
$$tildeCount = \log_{26} fileCount$$

cmp

1. flag options take priority over filename, ie file named ‘-c-l’ will be ignored and the string is treated as flag option unless there is use of relative path ‘./-c-l’
2. throw exception if the number of files to compare is not 2
3. throw exception if files are different, but no differences found up to EOF
4. multiple STDIN (’-’) in args does not count towards number of files
5. STDIN is always evaluated as the second argument