User Manual

Group 28

1. Requirement

- System requirements: the CVFS is best run on unix-based systems. Special characters might not print correctly in Windows console.

2. Get Started

- Step 1: Open **IntelliJ IDEA**
- Step 2: select the `Application` class
- Step 3: On the top right corner, from the drop-down menu, select `current file` and click then **run** button the start CVFS

Quick Start:

1. To avoid having to manually set up a directory, we have prepared a premade disk file at
   - `test/myDisk.ser` in Unix
   - `test\myDisk.ser` in Windows
2. Once you have CVFS up and running, type `load test/myDisk.ser`
3. You should see `load: loaded serialized disk from test/myDisk.ser`
4. Then, try to type `rList` (Section 3.12) and you should see the file hierachy:

```
>>>  load test/myDisk.ser
load: loaded serialized disk from test/myDisk.ser

~/
>>>  rList
.
├─root1/     size: 40
├─root2/     size: 586
│   ├─home3/     size: 318
│   │   ├─me1.txt     size: 72
│   │   ├─me3.txt     size: 114
│   │   └─me2.txt     size: 92
│   ├─home2/     size: 40
│   └─home1/     size: 188
│       └─HelloWorld.java size: 148
└─root3/     size: 40
-------------------------
Total Files     Total Size
10              666
```

,

5. Now, feel free to do whatever you want!

3. Commands Usage

3.1. `newDoc [docName] [docType] [docContent]` Supports Undo/Redo ✅

Creates a new document with specified name, type, and content.

- `docName`: new document name. must be $\leq$ 10 alphabet characters
- `docType`: new document type. must be one of {txt, java, html, css}
- `docContent`:
  - If `docContent` contains white space, user backtick (`).

    ```
    Example: newDoc HelloWorld txt `Hello World!`
    ```

  - Otherwise, directly write the content without backtick

    ```
    Example: newDoc HelloWorld txt HelloWorld!
    ```

Warning:
- Cannot create new document if insufficient disk space left.
- Cannot create new document if name already taken

3.2. `newDir [dirName]` Supports Undo/Redo ✅

Creates a new directory in the current working directory.

- `dirName`: new directory name. must be $\leq$ 10 alphabet characters

Warning:
- Cannot create new directory if remaining space < 40
- Cannot create new directory if name already taken

3.3. `delete [fileName]` Supports Undo/Redo ✅

Removes a file from the current working directory.

- `fileName`: existing file name. To check existing files, see Section 3.12

3.4. `rename [oldName] [newName]` Supports Undo/Redo ✅

Renames an existing file to a new name.

- `oldName`: existing file name. To check existing files, see Section 3.12
- `newName`: new file name. must be $\leq$ 10 alphabet characters

Warning
- Cannot rename file if `newName` already taken

- The file with `oldName` must be under the current directory. If not, use Section 3.8

3.5. `newSimplCri [criName] [attrName] [op] [value]` Supports Undo/Redo ✅

Creates a new simple criterion for file filtering based on name, size, or type.
- `criName`: name of criterion. must be exactly 2 alphabet characters
- `attrName`: name of targeting attribute. must be {name, size, type}

For name criterion
- `op`: must be *contains*
- `value`: must be written in double qoutes (e.g., "value")

For size criterion
- `op`: must be one of {<, >, <=, >=, ==, !=}
- `val`: must be numeric value **not** exceeding $2^{63} - 1 \approx 9 \times 10^{18}$

For type criterion
- `op`: must be *equals*
- `value`: must be one of {txt, java, html, css}

Warning:
- Cannot create new criterion if the two letter name already taken
- To check existing criterion, see Section 3.19

3.6. `newNegation [criName] [baseCriName]` Supports Undo/Redo ✅

Creates a new criterion that negates an existing criterion.
- `criName`: new criterion name. must be two letters
- `baseCriName`: existing criterion. To check existing criterion, see Section 3.19

Warning:
- Cannot create new criterion if the two letter name already taken

3.7. `newBinaryCri [criName] [baseCriName1] [logicOp] [baseCriName2]` Supports Undo/Redo ✅

Creates a new criterion by combining two existing criteria with a logical operator.
- `criName`: new criterion name. must be two letters
- `baseCriName`: existing criterion. To check existing criterion, see Section 3.19
- `logicOp`: the logical operator. Must be one of { &&, || }

Warning:
- Cannot create new criterion if the two letter name already taken

3.8. `changeDir [dirName / ..]` Supports Undo/Redo ✅
Changes the current working directory.
- `dirName`: must be an existing directory directly under the current directory
- to see all directories, use Section 3.12

3.9. `newDisk [diskSize]` Cannot Undo/Redo ❌
Initializes a new virtual disk with specified maximum size.
- `diskSize`: the maximum space of the new virtual diks

Warning:
- `diskSize` must be a numeric value **not** exceeding 10^9

3.10. `rSearch [criName]` Cannot Undo/Redo ❌
Prints a list of all the files (including in subdirectories) that satisfies the given criterion.

- `criName`: name of existing criterion. To check all criterions, use Section 3.19

3.11. `search [criName]` Cannot Undo/Redo ❌
Prints a list of all the files (**not** including in subdirectories) that satisfies the given criterion.

- `criName`: name of existing criterion. To check all criterions, use Section 3.19

3.12. `rList` Cannot Undo/Redo ❌
Prints an ASCII tree of all the file (including subdirectories) of the current working directory

3.13. `list` Cannot Undo/Redo ❌
Prints an ASCII tree of all the file (**not** including subdirectories) of the current working directory

3.14. `quit` Cannot Undo/Redo ❌
Quits the CVFS and ends the process

3.15. `undo` Cannot Undo/Redo ❌
Reverses the most recent reversible operation.

Warning
- Cannot undo when there are no new actions to reverse

3.16. `redo` Cannot Undo/Redo ❌
Reapplies the most recently undone operation.

Warning
- Cannot redo when there are NO immediately preceding `undo`(s)
- `redo` can **ONLY** be performed immediately after `undo`(s). If other instructions were executed in between, the undone instructions can never be redone.

3.17. `save [path]` Cannot Undo/Redo ❌
Saves the current state of the virtual file system to a file on the local machine.
- `path`: an absolute or relative path on the local machine. Supports both UNIX and Windows file path

Tips:
- If file path contains whitespace, wrap the entire path with backtick (`)
- Example: save `/path with/a lot of/space between`

3.18. `load [path]` Cannot Undo/Redo ❌
Loads a previously saved state of the virtual file system from a file.
- `path`: an absolute or relative path on the local machine. Supports both UNIX and Windows file path

Tips:
- If file path contains whitespace, wrap the entire path with backtick (`)
- Example: load `/path with/a lot of/space between`

3.19. `printAllCriteria` Cannot Undo/Redo ❌

prints all criterias of the current disk.

4. Troubleshoot

Q: How to input file path / document content with whitespace?

A: The CVFS prioritize backtick (`) over whitespace as command argument separators. Therefore, if you wish to have an argument contain whitespace, simply wrap the entire argument with backticks; it will automatically be treated as a single piece.

Q: Why can't I redo a previously undone action?

A: `redo` is **only** allowed immediately after `redo`(s), because `redo` in arbitrary context can be dangerous. Consider the following example:

$$\text{newDir A} \to \text{newDoc B} \to \text{undo} \to \text{rename A B}$$

After undoing "newDoc", executing "rename" creates a potential conflict. If we attempt to `redo` the `newDoc` operation:

$$\text{newDir A} \to \text{newDoc B} \to \text{undo} \to \text{rename A B} \ (\to \text{redo?})$$

This sequence would create a critical conflict since directory A has been renamed to B. Attempting to `redo` and restore document B would result in a name collision with the renamed directory. For this reason, `redo()` operations should only be permitted immediately after an `undo()`

Q: Why can't I save the virtual disk?

A: If you are attempting to save to C: drive on windows is most likely that CVFS does not have read or write permission to C: drive. To workaround this, please try another save location. In general, the error message of CVFS provides good enough guidance and explanation to the error cause.

Q: Why can't does it say invalid file format?

A: Ensure that you are using the write path format for your OS. For Unix-like systems (e.g., Linux or Mac) directories are separated by forward slash (`/`), where as in Windows it is backslash (\)