COMP 2021 Group 4 CVFS

In-Memory Virtual File System User Manual

Authors: Siu Yau Shing, Chow Kwan Ho

Contents

[Introduction: 3](#_Toc182838335)

[List of Commands 3](#_Toc182838336)

[Creating a New Disk 3](#_Toc182838337)

[Creating a New Document 3](#_Toc182838338)

[Creating a New Directory 3](#_Toc182838339)

[Deleting a File 4](#_Toc182838340)

[Renaming a File 4](#_Toc182838341)

[Changing the Working Directory 4](#_Toc182838342)

[Listing Files 4](#_Toc182838343)

[Recursively Listing Files 5](#_Toc182838344)

[Constructing Simple Criteria 5](#_Toc182838345)

[Checking if a File is a Document 5](#_Toc182838346)

[Constructing Composite Criteria 5](#_Toc182838347)

[Printing All Defined Criteria 6](#_Toc182838348)

[Searching for Files Based on a Criterion 6](#_Toc182838349)

[Recursively Searching for Files Based on a Criterion 6](#_Toc182838350)

[Saving the Working Virtual Disk 6](#_Toc182838351)

[Loading a Virtual Disk 7](#_Toc182838352)

[Terminating the System 7](#_Toc182838353)

[Troubleshooting 8](#_Toc182838354)

[Common Issues and Solutions 8](#_Toc182838355)

[Additional Resources 8](#_Toc182838356)

[FAQs: 8](#_Toc182838357)

[Technical Support: 8](#_Toc182838358)

[References: 8](#_Toc182838359)

# 

# Introduction:

This is a an in-memory Virtual File System (VFS), named the Comp VFS (CVFS), which is made in Java and utilizes OOP concepts to simulate a file system. The user opens application .java to start the program and use the file system. This is a group project made by Siu Yau Shing and Chow Kwan Ho.

The application uses the Model-View-Controller Pattern in Designing the CVFS. Simply run Application.Java and the program would start.

# List of Commands

## Creating a New Disk

**Command:** newDisk diskSize  
**Effect:** Creates a new virtual disk with the specified maximum size. The previous working disk, if any, is closed. The newly created disk is set to be the working disk of the system, and the working directory is set to be the root directory of the disk.

**Steps:**

1. Open the CLI tool.
2. Type newDisk followed by the desired disk size (e.g., newDisk 1024).
3. Press Enter.

## Creating a New Document

**Command:** newDoc docName docType docContent  
**Effect:** Creates a new document in the working directory with the specified name, type, and content.

**Steps:**

1. Ensure you are in the desired directory.
2. Type newDoc followed by the document name, type, and content (e.g., newDoc myfile txt "Hello World").
3. Press Enter.

## Creating a New Directory

**Command:** newDir dirName  
**Effect:** Creates a new directory in the working directory with the specified name.

**Steps:**

1. Ensure you are in the desired directory.
2. Type newDir followed by the directory name (e.g., newDir myfolder).
3. Press Enter.

## Deleting a File

**Command:** delete fileName  
**Effect:** Deletes an existing file with the specified name from the working directory.

**Steps:**

1. Ensure you are in the directory containing the file.
2. Type delete followed by the file name (e.g., delete myfile).
3. Press Enter.

## Renaming a File

**Command:** rename oldFileName newFileName  
**Effect:** Renames an existing file in the working directory from oldFileName to newFileName.

**Steps:**

1. Ensure you are in the directory containing the file.
2. Type rename followed by the old file name and the new file name (e.g., rename myfile newfile).
3. Press Enter.

## Changing the Working Directory

**Command:** changeDir dirName  
**Effect:** Changes the working directory to the specified directory name.

**Steps:**

1. Type changeDir followed by the directory name (e.g., changeDir myfolder).
2. If you want to move to the parent directory, type changeDir ...
3. Press Enter.

## Listing Files

**Command:** list  
**Effect:** Lists all the files directly contained in the working directory.

**Steps:**

1. Type list.
2. Press Enter.
3. View the list of files, including their names, types, and sizes.

## Recursively Listing Files

**Command:** rList  
**Effect:** Lists all the files contained in the working directory recursively.

**Steps:**

1. Type rList.
2. Press Enter.
3. View the hierarchical list of files with indentation indicating the level of each file.

## Constructing Simple Criteria

**Command:** newSimpleCri criName attrName op val  
**Effect:** Constructs a simple criterion that can be referenced by criName.

**Steps:**

1. Type newSimpleCri followed by the criterion name, attribute name, operator, and value (e.g., newSimpleCri cr1 name "contains” "report").
2. Press Enter.

## Checking if a File is a Document

**Criterion name:** IsDocument  
**Effect:** Evaluates to true if and only if the file is a document.

**Steps:**

1. Use this criterion in search commands to filter documents.

## Constructing Composite Criteria

**Command:** newNegation criName1 criName2  
**Effect:** Constructs a composite criterion that is the negation of an existing criterion.

**Steps:**

1. Type newNegation followed by the new criterion name and the existing criterion name (e.g., newNegation cr2 cr1).
2. Press Enter.

**Command:** newBinaryCri criName1 criName3 logicOp criName4  
**Effect:** Constructs a composite criterion using a logical operator.

**Steps:**

1. Type newBinaryCri followed by the new criterion name, first criterion, logical operator, and second criterion (e.g., newBinaryCri cr3 cr1 && cr2).
2. Press Enter.

## Printing All Defined Criteria

**Command:** printAllCriteria  
**Effect:** Prints out all the defined criteria.

**Steps:**

1. Type printAllCriteria.
2. Press Enter.
3. View the list of all defined criteria.

## Searching for Files Based on a Criterion

**Command:** search criName  
**Effect:** Lists all the files directly contained in the working directory that satisfy the specified criterion.

**Steps:**

1. Type search followed by the criterion name (e.g., search cr1).
2. Press Enter.
3. View the list of files that match the criterion.

## Recursively Searching for Files Based on a Criterion

**Command:** rSearch criName  
**Effect:** Lists all the files contained in the working directory that satisfy the specified criterion recursively.

**Steps:**

1. Type rSearch followed by the criterion name (e.g., rSearch cr1).
2. Press Enter.
3. View the hierarchical list of files that match the criterion.

## Saving the Working Virtual Disk

**Command:** save path  
**Effect:** Saves the working virtual disk into a file at the specified path.

**Steps:**

1. Type save followed by the file path (e.g., save /path/to/diskfile).
2. Press Enter.

## Loading a Virtual Disk

**Command:** load path  
**Effect:** Loads a virtual disk from a file at the specified path and makes it the working virtual disk.

**Steps:**

1. Type load followed by the file path (e.g., load /path/to/diskfile).
2. Press Enter.

## Terminating the System

**Command:** quit  
**Effect:** Terminates the execution of the system.

**Steps:**

1. Type quit.
2. Press Enter.

**Undoing and Redoing Commands**

**Supported Commands:** newDoc, newDir, delete, rename, changeDir, newSimpleCri, newNegation

**Steps:**

1. To undo a command, type undo.
2. Press Enter.
3. To redo a command, type redo.
4. Press Enter.

• Screenshots: For each command, consider including screenshots to visually demonstrate the

results under different inputs. Capture screenshots that showcase the system’s response or

output when specific commands are executed. Include captions or annotations to provide

additional context or explanations for each screenshot.

• Step-by-Step Instructions: Provide clear and concise step-by-step instructions for users to

follow when using each command. Break down the process into logical steps, ensuring that

users can easily understand and replicate the actions.

# Troubleshooting

## Common Issues and Solutions

**Issue:** Command not recognized.  
**Solution:** Ensure you are typing the command correctly and using the correct syntax.

**Issue:** Disk size exceeded.  
**Solution:** Check the current disk usage and ensure the new file or directory fits within the remaining space.

**Issue:** File or directory not found.  
**Solution:** Verify the file or directory name and ensure it exists in the current working directory.

# Additional Resources

## FAQs:

**Q1: What file types are supported by the CVFS?**  
**A1:** The CVFS supports documents of types txt, java, html, and css.

**Q2: What are the limitations of the file names?**  
**A2:** Only digits and English letters are allowed in file names. Each file name may have at most 10 characters, and all the documents and directories have unique names.

**Q3: Can the document type be the document name?**  
**A3:** No, document types are not part of document names.

**Q4: How large are the size of the documents and directories?**  
**A4:** the size of a document is calculated as 40 + content.length\*2, and the size of a directory is calculated as 40 plus the total size of its contained files.

**Q4: What commands can be undone and redone?**  
**A4:** the commands newDoc, newDir, delete, rename, changeDir, newSimpleCri, and newNegation.

## Technical Support:

For any further questions, contact the email [23115372d@connect.polyu.hk](mailto:23115372d@connect.polyu.hk) or at +852 XXXXXXXX

## References:

<https://en.wikipedia.org/wiki/Virtual_file_system>

<https://en.wikipedia.org/wiki/Object-oriented_programming>

<https://www.polyu.edu.hk/comp/docdrive/ug/subject/COMP2021.pdf>