Group 10: YANG Xikun, YANG Jinkun, REN Yixiao, Arda EREN
Part of Project Peregrine Falcon

COMP2021 Object-Oriented Programming

Group Project

COMP Virtual File System (CVFS)

USER MANUAL

Please run the `Application` class to start the CVFS application

COMP Virtual System (CVFS) is a virtual file system where you can create virtual disks and manage files within them. CVFS allows you to create criteria to filter specific files. You can also load/save the virtual disks and criteria files from/to your local file system.

This user manual helps you use CVFS correctly and achieve the best experience.

I. TERMS

1. ***Virtual disks*** refers to the disks in our system, they store files. A virtual disk has a ***size***.

2. ***Files*** refers to directories and documents. It is not allowed to have files with same names under one directory, even if their types are different. The type suffix (e.g., `.java`) is not included in the filename and will only be used for presenting, and you cannot use it.

3. ***Root directory*** refers to the root directory of the virtual disk. All files in the virtual disk should be under the root directory. The root directory is denoted as `$`.

4. ***Working directory*** refers to the current directory you are in.

5. ***Paths*** of files refer to the paths to the corresponding files. This may be relative or absolute. Different components of the path are split by `:`.

   In addition,

   `.` means to stay in the current directory,

   `..` means to go to the parent directory.

   For example, the following two paths are valid:

   `$:Main:Main.java` (you need to use `$:Main:Main`)

   `..:Main:Main.java` (you need to use `..:Main:Main`)

   In our system, you can only use paths when changing the working directory (see **III.8**).

6. ***Criteria*** refers to criteria to filter directories and documents. For example,

   ```
   size > 60
   ```

   is a criterion which filters all files with size greater than 60.

II. COMMANDS GUIDELINE

1. You need ***commands*** to do ***operations***.
    - Invalid commands cannot be converted into corresponding operations (e.g., wrong parameters).
    - Even if the commands are valid, the operations may fail to execute (e.g., view the content of a non-existent file).

2. The name of the commands, and the parameters of the commands, should be split by a spacebar (␣)[1]. For example,

```
commandName parameter1 parameter2
```

3. If you need to use spacebars in parameters, quote the whole parameter. You have to choices, use quote marks (`""`) or use backticks (`` ` ``). When your command is parsed, quote marks will be kept and backticks will be removed.

[1] Multiple spacebars are also allowed.

III. COMMAND LIST

1. Create a new virtual disk

`newDisk <diskSize>`

Where `<diskSize>` is the size of the virtual disk. It should be a positive appropriate integer.

WARNING: This operation would eject the mounted disk, if it exists. Please be sure you have saved the mounted disk. Currently, an invalid `newDisk` command will not cause the ejection, but this feature may be changed.

WARNING: This operation would remove the records of all file-related operations, including `newDir`, `newDoc`, `rename`, `delete` and `modify`, which affects undo and redo.

Example usage: `newDisk 50000`

```
admin@CVFS >> newDisk notANumber
Invalid command: Invalid disk size. Disk size should be a positive and appropriate integer.

admin@CVFS >> newDisk -1000
Invalid command: Invalid disk size. Disk size should be larger than 0.

admin@CVFS >> newDisk 10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
Invalid command: Invalid disk size. Disk size should be a positive and appropriate integer.

admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.
```

2. Load a virtual disk from the local file system

`load <path>`

Where `<path>` is the path (whatever relative path or absolute path) to the virtual disk file in your local file system.

WARNING: This operation would eject the mounted disk, if it exists. Please be sure you have saved the mounted disk. Currently, an invalid `load` command or invalid virtual disk file will not cause the ejection, but this feature may be changed.

WARNING: This operation would remove the records of all file-related operations, including `newDir`, `newDoc`, `rename`, `delete` and `modify`, which affects undo and redo.

Example usage: `load SampleVirtualDisk.ser`

```
admin@CVFS >> load NoSuchFile.ser
Operation cannot execute: Local file system error: NoSuchFile.ser (No such file or directory).

admin@CVFS >> load SampleVirtualDisk.ser
The virtual disk has been successfully loaded from: SampleVirtualDisk.ser and mounted.
```

3. Save a virtual disk to the local file system

`save <path>`

Where `<path>` is the path (whatever relative path or absolute path) to the virtual disk file in your local file system.

Example usage: `save SampleVirtualDisk.ser`

```
admin@CVFS >> save SampleVirtualDisk.ser
Operation cannot execute: No mounted virtual disk or available working directory.

admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.

admin@CVFS $ >> save SampleVirtualDisk.ser
The mounted virtual disk has been successfully saved to: SampleVirtualDisk.ser.
```

4. Create a new directory

`newDir <dirName>`

Where `<dirName>` is the name of the new directory. It should be numeral alphabetic and within 10 characters.

Example usage: `newDir Main`

```
admin@CVFS >> newDir Main
Operation cannot execute: No mounted virtual disk or available working directory.

admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.

admin@CVFS $ >> newDir TooLongNameTooLongName
Invalid command: The length of the filename cannot be more than 10, with provided length: 22.

admin@CVFS $ >> newDir Invalid_Name
Invalid command: The length of the filename cannot be more than 10, with provided length: 12.

admin@CVFS $ >> newDir Main
New directory Main has been created successfully and added into $.
```

5. Create a new document

`newDoc <docName> <docType> <docContent>`

Where,

- `<docName>` is the name of the new name. It should be numeral alphabetic and within 10 characters.
- `<docType>` is the type of the new name. It should be one of `txt`, `java`, `html` and `css`.
- `<docContent>` is the content of the new name.

Example usage: `newDoc Main java` `` `public class Main { }` ``

```
admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.

admin@CVFS $ >> newDoc Main invalid_type `public class Main { }`
Operation cannot execute: Cannot initialize the file: Invalid parameter(s) for initializing the document.

admin@CVFS $ >> newDoc Main java public class Main { }
Invalid command: Wrong number of parameters: 7.

admin@CVFS $ >> newDoc Main java `public class Main { }`
New document Main.java has been created successfully and added into $.
```

6. View the content of a document

`view <docName>`

Where `<docName>` is the name of the document to view. You can only view the document of the current working directory.

Example usage: `view Main` (Do not use `view Main.java`)

```
admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.

admin@CVFS $ >> newDoc Main java `public class Main { }`
New document Main.java has been created successfully and added into $.

admin@CVFS $ >> view NoSuchFile
Operation cannot execute: File not exists: "NoSuchFile".

admin@CVFS $ >> view Main
The content of Main.java:
public class Main { }
```

7. Rename a file

`rename <oldFileName> <newFileName>`

Where,

- `<oldFileName>` if the old file name (as well as the name of the file to rename).
- `<newFileName>` is the new name of that file.

Example usage: `rename Main MainSrc`

```
admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.

admin@CVFS $ >> newDoc Main java `public class Main { }`
New document Main.java has been created successfully and added into $.

admin@CVFS $ >> rename Main TooLongNameTooLongName
Invalid command: The length of the filename cannot be more than 10, with provided length: 22.

admin@CVFS $ >> rename Main Main2
The file "Main" has been successfully renamed to "Main2", now the full name is: "Main2.java".

admin@CVFS $ >> view Main
Operation cannot execute: File not exists: "Main".

admin@CVFS $ >> view Main2
The content of Main2.java:
public class Main { }
```

8. Remove a file

`delete <fileName>`

Where, `<fileName>` is the name of the file to delete.

Example usage: `delete Main`

```
admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.

admin@CVFS $ >> newDoc Main java `public class Main { }`
New document Main.java has been created successfully and added into $.

admin@CVFS $ >> delete NoSuchFile
Operation cannot execute: File not exists: "NoSuchFile".

admin@CVFS $ >> delete Main
The file "Main.java" has been removed successfully.

admin@CVFS $ >> view Main
Operation cannot execute: File not exists: "Main".
```

9. Modify the content of a file

`modify <docName> <newContent>`

Where,

- `<docName>` is the name of the document to modify.
- `<newContent>` is the new content of that file.

Example usage: `modify Main `public class Main { private int a; }``

```
admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.

admin@CVFS $ >> newDoc Main java `public class Main { }`
New document Main.java has been created successfully and added into $.

admin@CVFS $ >> modify NoSuchFile `public class Main { private int a; }`
Operation cannot execute: File not exists: "NoSuchFile".

admin@CVFS $ >> modify Main `public class Main { private int a; }`
The content has been successfully modified.

admin@CVFS $ >> view Main
The content of Main.java:
public class Main { private int a; }
```

10. Change the working directory

`changeDir <dirName>`

Where `<dirName>` is the **path** to the new working directory.

Example usages:

`changeDir $`

`changeDir ..`

`changeDir ..:Main`

`changeDir .:Main`

`changeDir Main`

```
admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.

admin@CVFS $ >> newDir Main1
New directory Main1 has been created successfully and added into $.

admin@CVFS $ >> newDir Main2
New directory Main2 has been created successfully and added into $.

admin@CVFS $ >> changeDir Main1
The directory was successfully changed into: $:Main1.

admin@CVFS $:Main1 >> changeDir ..:Main2
The directory was successfully changed into: $:Main2.

admin@CVFS $:Main2 >> changeDir $
The directory was successfully changed into: $.
```

11. List files *directly* in the working directory.

`list`

This will list files *directly* contained in the working directory.

Example usage: `list`

```
admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.

admin@CVFS $ >> newDir Main1
New directory Main1 has been created successfully and added into $.

admin@CVFS $ >> newDir Main2
New directory Main2 has been created successfully and added into $.

admin@CVFS $ >> changeDir Main1
The directory was successfully changed into: $:Main1.

admin@CVFS $:Main1 >> newDir src
New directory src has been created successfully and added into Main1.

admin@CVFS $:Main1 >> list
$:Main1:
src (40)
Report: 1 files, with total size 40.

admin@CVFS $:Main1 >> changeDir $
The directory was successfully changed into: $.

admin@CVFS $ >> list
$:
Main1 (80)
Main2 (40)
Report: 2 files, with total size 120.
```

12. List all files *recursively* in the working directory

`rList`

This will list *all* files *recursively* contained in the working directory.

Example usage: `rList`

```
admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.

admin@CVFS $ >> newDir Main1
New directory Main1 has been created successfully and added into $.

admin@CVFS $ >> newDir Main2
New directory Main2 has been created successfully and added into $.

admin@CVFS $ >> changeDir Main1
The directory was successfully changed into: $:Main1.

admin@CVFS $:Main1 >> newDir src
New directory src has been created successfully and added into Main1.

admin@CVFS $:Main1 >> rList
$:Main1
└──src (40)
Report: 1 files, with total size 40.

admin@CVFS $:Main1 >> changeDir $
The directory was successfully changed into: $.

admin@CVFS $ >> rList
$
├──Main1 (80)
│   └──src (40)
└──Main2 (40)
Report: 3 files, with total size 120.
```

13. Create a new simple criterion

```
newSimpleCri <criName> <attrName> <op> <val>
```

Where,

- `<criName>` is the name of the new simple criterion. It is only two English letters.
- `<attrName>` is the main attribute (or, type) of the criterion.
- `<op>` is the operator.
- `<val>` is the value.

Noted that,

- When `<attrName>` is `name`, `<op>` must be `contains`, and `<val>` must be a string in the double quote. For example,

  ```
  newSimpleCri aa name contains "Main"
  ```

- When `<attrName>` is `type`, `<op>` must be `equals`, and `<val>` must be a string in the double quote. For example,

  ```
  newSimpleCri bb type equals "java"
  ```

- When `<attrName>` is `size`, `<op>` can be one of `>, <, >=, <=, ==, !=`, and `<val>` must be an integer. For example,

  ```
  newSimpleCri cc size > 60
  ```

This command helps you create criteria to filter file. View **III.18** and **III.19**.

```
admin@CVFS >> newSimpleCri aa name contains Main
Operation cannot execute: Invalid criterion parameter: Main.

admin@CVFS >> newSimpleCri aa name contains "Main"
The new criterion has been created successfully: aa: name contains "Main".

admin@CVFS >> newSimpleCri bb type equal "java"
Operation cannot execute: Invalid criterion parameter: equal.

admin@CVFS >> newSimpleCri bb type equals "java"
The new criterion has been created successfully: bb: type equals "java".

admin@CVFS >> newSimpleCri cc size <> 60
Operation cannot execute: Invalid criterion parameter: <>.

admin@CVFS >> newSimpleCri cc size > 60
The new criterion has been created successfully: cc: size > 60.
```

14. Create a new negation criterion

A ***negation criterion*** is the negation of an existing criterion. For example, if criterion `cc` represents for `size > 60`, then the negation of cc means `!(size > 60)`.

```
newNegation <criName1> <criName2>
```

Where,

- `<criName1>` is the name of the new simple criterion. It is only two English letters.
- `<criName2>` is the name of another criterion (to negate to).

Example usage: `newNegation dd cc`

```
admin@CVFS >> newSimpleCri cc size > 60
The new criterion has been created successfully: cc: size > 60.

admin@CVFS >> newNegation dd xx
Operation cannot execute: Criterion not exists: "xx".

admin@CVFS >> newNegation dd cc
The new criterion has been created successfully: dd: !cc.
```

15. Create a new binary criterion

A ***binary criterion*** is the criterion of the logic operation applied to two existing criteria. For example, if criterion `aa` represents for `name contains "Main"`, and `cc` represents for `size > 60`, then the binary AND criterion means `(name contains "Main") && (size > 60)`.

```
newBinaryCri <criName1> <criName3> <logicOp> <criName4>
```

Where,

- `<criName1>` is the name of the new simple criterion. It is only two English letters.
- `<criName3> <criName4>` are the name of the existing criteria.
- `<logicOp>` is the logic operation for the two existing criteria.

Noted that `<logicOp>` should be one of `&&` (logical AND) or `||` (logical OR).

Example usage: `newBinaryCri ee aa && cc`

```
admin@CVFS >> newSimpleCri aa name contains "Main"
The new criterion has been created successfully: aa: name contains "Main".

admin@CVFS >> newSimpleCri cc size > 60
The new criterion has been created successfully: cc: size > 60.

admin@CVFS >> newBinaryCri ee aa && cc
The new criterion has been created successfully: ee: aa && cc.

admin@CVFS >> newBinaryCri ff aa | cc
Operation cannot execute: Invalid criterion parameter: |.

admin@CVFS >> newBinaryCri ff aa || cc
The new criterion has been created successfully: ff: aa || cc.
```

16. Remove a criterion

`deleteCri <criName>`

Where, `<criName>` is the name of the crirterion to delete.

Noted you cannot remove the criterion if another criteria depend on it. This keeps the integrity and stability.

Example usage: `deleteCri aa`

```
admin@CVFS >> newSimpleCri aa name contains "Main"
The new criterion has been created successfully: aa: name contains "Main".

admin@CVFS >> newSimpleCri cc size > 60
The new criterion has been created successfully: cc: size > 60.

admin@CVFS >> newBinaryCri ee aa && cc
The new criterion has been created successfully: ee: aa && cc.

admin@CVFS >> deleteCri aa
Operation cannot execute: The criterion cannot be deleted. Possibly because other criterion depend on this criterion.

admin@CVFS >> deleteCri ee
The criterion has been removed successfully: ee: aa && cc

admin@CVFS >> deleteCri aa
The criterion has been removed successfully: aa: name contains "Main"
```

17. Print all criteria

`printAllCriteria`

This will list all existing criteria.

Example usage: `printAllCriteria`

```
admin@CVFS >> newSimpleCri aa name contains "Main"
The new criterion has been created successfully: aa: name contains "Main".

admin@CVFS >> newSimpleCri bb type equals "java"
The new criterion has been created successfully: bb: type equals "java".

admin@CVFS >> newSimpleCri cc size > 60
The new criterion has been created successfully: cc: size > 60.

admin@CVFS >> newNegation dd cc
The new criterion has been created successfully: dd: !cc.

admin@CVFS >> newBinaryCri ee aa && cc
The new criterion has been created successfully: ee: aa && cc.

admin@CVFS >> printAllCriteria
There are 6 criteria in the working directory:
IsDocument
aa: name contains "Main"
bb: type equals "java"
cc: size > 60
dd: !cc
ee: aa && cc
```

18. Search files *directly* in the working directory

`search <criName>`

This will list files *directly* contained in the working directory that satisfy the criterion.

Example usage: `search aa`

```
admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.

admin@CVFS $ >> newDir Main
New directory Main has been created successfully and added into $.

admin@CVFS $ >> changeDir Main
The directory was successfully changed into: $:Main.

admin@CVFS $:Main >> newDoc Main java `public class Main { }`
New document Main.java has been created successfully and added into Main.

admin@CVFS $:Main >> newSimpleCri aa name contains "Main"
The new criterion has been created successfully: aa: name contains "Main".

admin@CVFS $:Main >> search aa
These file(s) satisfy the criterion: aa: name contains "Main":
Main.java (82)
Report: 1 files, with total size 82.

admin@CVFS $:Main >> changeDir $
The directory was successfully changed into: $.

admin@CVFS $ >> search aa
These file(s) satisfy the criterion: aa: name contains "Main":
Main (122)
Report: 1 files, with total size 122.
```

19. Search all files *recursively* in the working directory

`rSearch <criName>`

This will list *all* files *recursively* contained in the working directory that satisfy the criterion.

Example usage: `rSearch aa`

```
admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.

admin@CVFS $ >> newDir Main
New directory Main has been created successfully and added into $.

admin@CVFS $ >> changeDir Main
The directory was successfully changed into: $:Main.

admin@CVFS $:Main >> newDir src
New directory src has been created successfully and added into Main.

admin@CVFS $:Main >> newDoc Info txt `Information: TBD`
New document Info.txt has been created successfully and added into Main.

admin@CVFS $:Main >> changeDir $
The directory was successfully changed into: $.

admin@CVFS $ >> rSearch IsDocument
These file(s) satisfy the criterion: IsDocument:
$:Main:Info.txt (72)
Report: 1 files, with total size 72.
```

20. Load criteria from the local file system

`loadCri <path>`

Where `<path>` is the path (whatever relative path or absolute path) to the criteria file in your local file system.

WARNING: This operation would delete all existing criteria. Please be sure you have saved the criteria. Currently, an invalid `load` command or invalid criteria file will not cause the deletion, but this feature may be changed.

WARNING: This operation would remove the records of all file-*un*related operations, including `newSimpleCri`, `newNegation`, `newBinaryCri` and `deleteCri`, which affects undo and redo.

Example usage: `loadCri sampleCriteria.ser`

```
admin@CVFS >> printAllCriteria
There are 1 criteria in the working directory:
IsDocument

admin@CVFS >> loadCri sampleCriteria.ser
The criteria has been successfully loaded from: sampleCriteria.ser.

admin@CVFS >> printAllCriteria
There are 2 criteria in the working directory:
IsDocument
aa: name contains "Main"
```

21. Save criteria to the local file system

`saveCri <path>`

Where `<path>` is the path (whatever relative path or absolute path) to the criteria file in your local file system.

Example usage: `saveCri sampleCriteria.ser`

```
admin@CVFS >> newSimpleCri aa name contains "Main"
The new criterion has been created successfully: aa: name contains "Main".

admin@CVFS >> saveCri sampleCriteria.ser
All criteria has been successfully saved to: sampleCriteria.ser.
```

22. Undo

`undo`

This will undo the latest operation.

Noted only `newDoc`, `newDir`, `delete`, `rename`, `modify`, `changeDir`, `newSimpleCri`, `newNegation` and `newBinary` supports undo.

Example usage: `undo`

```
admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.

admin@CVFS $ >> undo
Invalid command: No operation to undo.

admin@CVFS $ >> newDir Main
New directory Main has been created successfully and added into $.

admin@CVFS $ >> undo
The file "Main" has been removed successfully.

admin@CVFS $ >> undo
Invalid command: No operation to undo.
```

23. Redo

`redo`

This will redo the latest undid operation.

Noted only `newDoc`, `newDir`, `delete`, `rename`, `modify`, `changeDir`, `newSimpleCri`, `newNegation` and `newBinary` supports redo.

Example usage: `redo`

```
admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.

admin@CVFS $ >> newDir Main1
New directory Main1 has been created successfully and added into $.

admin@CVFS $ >> undo
The file "Main1" has been removed successfully.

admin@CVFS $ >> redo
The file Main1 has been put back to the directory $ successfully.

admin@CVFS $ >> undo
The file "Main1" has been removed successfully.

admin@CVFS $ >> newDir Main2
New directory Main2 has been created successfully and added into $.

admin@CVFS $ >> redo
Invalid command: No operation to redo.
```

24. Quit the system

`quit`

WARNING: This operation would eject the mounted virtual disk and delete all criteria. Please be sure you have saved the resources. Currently, an invalid `quit` command will not cause the ejection and deletion, but this feature may be changed.

Example usage: `quit`

```
admin@CVFS >> newDisk 50000
New disk with size 50000 has been created and mounted successfully, and the previous disk, if exists, has been ejected.

admin@CVFS $ >> newDir Main
New directory Main has been created successfully and added into $.

admin@CVFS $ >> quit

Process finished with exit code 0
```

IV. TROUBLESHOOTING

Below, we list common error reports and provide troubleshooting solutions.

- **[In `newDisk`]** Invalid disk size. Disk size should be a positive and appropriate integer.
  - Make the disk size is an appropriate positive integer.
- **[In `newDoc`, `newDir`, `rename`, `delete`, `view`, `modify`]** No mounted virtual disk or available working directory.
  - Mount a virtual disk first before doing such operations.
- **[In `newDoc`, `newDir`, `rename`]** Cannot initialize the file: Invalid parameter(s) for initializing the directory.
  - File names must be within 10 characters, and include only letters and numbers.
  - Document type should be one of `txt`, `java`, `html` and `css`.
- **[In `newDoc`, `newDir`, `rename`]** The length of the filename cannot be more than 10, with provided length: …
  - File names must be within 10 characters.
- **[In `newDir`, `newDoc`, `rename`]** Duplicated filename "…".
  - Use another new and unique name.
- **[In `rename`, `delete`, `view`, `modify`]** File not exists: "…".
  - Make sure the file you want to operate on exists.
- **[In `newDir`, `newDoc`, `modify`]** Virtual disk out of space to save the file: … out of ….
  - Save this virtual disk and create a new virtual disk with larger size.
- **[In `newSimpleCri`, `newNegation`, `newBinaryCri`]** Invalid Criterion name.
  - Criterion names must be two English letters.
- **[In `newSimpleCri`, `newNegation`, `newBinaryCri`]** Incorrect Criterion parameters.
  - Please see **III.13**, **III.14** and **III.15** to make sure the criteria parameters are correct.
- **[In `deleteCri`]** The criterion cannot be deleted. Possibly because other criterion depend on this criterion.

  - Recheck the operation. Use `printAllCriteria` to check all existing criteria and dependencies. If you insist to delete that criterion, please remove those that depend on it first.

- **[In `newSimpleCri`, `newNegation`, `newBinaryCri`, `deleteCri`]** Criterion not exists: "…".
  - Make sure the criterion you want to operate on exists.
- **[In `load`, `save`, `loadCri`, `saveCri`]** Local file system error: …
  - Make sure the file name is valid, the file exists, and was created by CVFS.

Nov 20, 2024