
# Computer System Engineering

**Week 5: Lab 4 (25 marks)**

**Objective:** Implement file operation in Shell Interface using Java

In Lab 1 we have implemented a Shell Interface using Java. In Lab 4 we will extend the Shell implementation with several file operation methods.

A collection of file operation functions are defined in **java.io.File** class. You can refer to the documentation of **java.io.File** class from the following link:

http://docs.oracle.com/javase/7/docs/api/java/io/File.html

In order to implement Q1 to Q4, you need to write code to handle different file operation commands, and also implement the following functions:

```
public static void Java_create(File dir, String name);

public static void Java_delete(File dir, String name);

public static void Java_cat(File dir, String name);

public static void Java_ls(File dir, String display_method, String
sort_method);

public static boolean Java_find(File dir, String name);

public static void Java_tree(File dir, int depth, String sort_method);
```

The first 3 functions are for Q1, and the last 3 functions are for Q2, Q3, Q4, respectively.

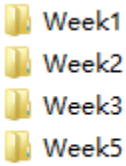
**Q1. Implement functions to create, delete, and display a file (10 marks)**

Use the starting code "FileOperation – starting code.java" to implement.

For Q1, there are 3 operations for single file: create, delete, and display.

**Create a file:** New files can be created and added to current directory.

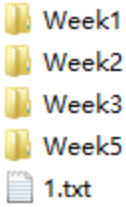
For example, now we have 4 folders under current directory:




Under Shell Interface, when we type in the following command:

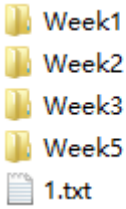
```
jsh>create 1.txt
```

A file called “1.txt” will be created under current directory:




**Delete a file:** When a file is no longer needed, we want to remove it from current directory.

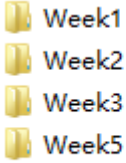
For example, now we have 4 folders and 1 file “1.txt” under current directory:




When we type in the following command:

```
jsh>delete 1.txt
```

The file “1.txt” will be deleted from current directory:




**Display a file:** When we want to see the content in a file, we use the display command.

By type in this command:

```
jsh>display test.txt
```

The content inside “test.txt” will be printed out:

```
Hello.
This is the content inside test.txt file.
```

**Suggested Function:**

| Function | Class which function belongs to |
|---|---|
| *.exists()* | *File* |
| *.createNewFile()* | *File* |
| *.readLine()* | *BufferedReader* |

**Test Command:**

*jsh> create a.txt*
*jsh> display a.txt*
*jsh> delete a.txt*

**Q2. Implement function to list a directory (5 marks)**

Continue to use the code you implemented in Q1.

Here we design a function to list files under current directory.

When we type in:

```
jsh>list
```

The files (including folders) will be printed out as follows:

```
Week1
Week2
Week3
Week5
```

We also add in the option to show property of files, e.g. file size, last modified time.

When we type in:

```
jsh>list property
```

The files (including folders) will be printed out as follows:

```
Week1       Size: 4096        Last Modified: Mon Jan 26 13:10:47 SGT 2015
Week2       Size: 4096        Last Modified: Sun Jan 18 21:09:22 SGT 2015
Week3       Size: 4096        Last Modified: Thu Feb 05 16:43:57 SGT 2015
Week5       Size: 0           Last Modified: Thu Feb 12 16:16:27 SGT 2015
```

The list function should be able to sort the files according to different property.

For example, when we type in:

```
jsh>list property time
```

The files (including folders) will be printed out as follows:

```
Week2       Size: 4096        Last Modified: Sun Jan 18 21:09:22 SGT 2015
Week1       Size: 4096        Last Modified: Mon Jan 26 13:10:47 SGT 2015
Week3       Size: 4096        Last Modified: Thu Feb 05 16:43:57 SGT 2015
Week5       Size: 0           Last Modified: Thu Feb 12 16:16:27 SGT 2015
```

The function for sorting file list based on different property (e.g. name, size, time) is provided in starting code:

```
private static File[] sortFileList(File[] list, String sort_method);
```

**Suggested Function:**

| Function | Class which function belongs to |
|---|---|
| *.lastModified()* | *File* |
| *.getName()* | *File* |
| *Date()* | *Date* |

**Test Command:**

*jsh> list*
*jsh> list property*
*jsh> list property time*

**Q3. Implement function to find files under current directory and subdirectories (5 marks)**

Continue to use the code you implemented in Q2.

Here we design a function to find files under current directory and its subdirectories. For example, the current directory is **C:\CSE_Lab\src**. When we type in the find command with a string:

```
jsh>find .java
```

The function will print out the entries of all files with ".java" in its name (as a substring in the name). The output is as follows:

```
C:\CSE_Lab\src\Week1\SimpleShell.java
C:\CSE_Lab\src\Week2\MergeSortThreaded.java
C:\CSE_Lab\src\Week2\MultiThread.java
C:\CSE_Lab\src\Week3\Bank.java
C:\CSE_Lab\src\Week3\BankImpl.java
C:\CSE_Lab\src\Week3\TestBank.java
C:\CSE_Lab\src\Week5\FileOperation.java
```

Note that you need to find files not only in current directory, but also in its subdirectories. So here you can implement a recursive function to find files in different level of subdirectories.

**Suggested Function:**

| Function | Class which function belongs to |
|---|---|
| *.contains()* | *String* |
| *.listFiles()* | *File* |

**Test Command:**

*jsh> find .java*
*jsh> find .xyz*

**Q4. Implement function to list subdirectories and files in a tree structure (5 marks)**

Continue to use the code you implemented in Q3.

In order to efficiently show files under different level of directories, we need a function to show files in tree structure. For example, when we type in tree command under directory **C:\CSE_Lab\src**:

```
jsh>tree
```

The output should be as follows:

```
Week1
  |-SimpleShell.java
Week2
  |-data
    |-input_1.txt
    |-input_2.txt
  |-MergeSortThreaded.java
  |-MultiThread.java
Week3
  |-Bank.java
  |-BankImpl.java
  |-TestBank.java
Week5
  |-FileOperation.java
```

Sometimes we only want to see several top levels of current directories. We should be able to control the maximum level of subdirectories to be shown. For example, when we type in tree command under directory **C:\CSE_Lab\src**:

```
jsh>tree 1
```

The output should be as follows:

```
Week1
Week2
Week3
Week5
```

As you can see here, we set the maximum level of subdirectories to be 1, so only the first level of subdirectories is shown.

Similar to the list command, we can also decide which file property we use to sort the file order to print. For example, when we type in tree command under directory **C:\CSE_Lab\src**:

```
jsh>tree 2 time
```

The output should be as follows:

```
Week2
  |-MergeSortThreaded.java
  |-MultiThread.java
  |-data
Week1
  |-SimpleShell.java
Week3
  |-Bank.java
  |-TestBank.java
  |-BankImpl.java
Week5
  |-FileOperation.java
```

The tree structure is listed based on time order. So the sequence is different from using name order.

**Suggested Function:**

| Function | Class which function belongs to |
|---|---|
| *.isDirectory()* | *File* |
| *.listFiles()* | *File* |

**Test Command:**

*jsh> tree*
*jsh> tree 2*
*jsh> tree 5*

The starting codes for Lab 4 can be found in eDimension:

FileOperation - starting code.java

After you finish all 4 questions, submit the java file (modified from "FileOperation – starting code.java") to eDimension before next Lab.

*File operation functions in **java.io.File** class:*

**Create a file:**

File file;

file.createNewFile();

**Delete a file:**

File file;

file.delete();

**List files:**

File dir;

File[] list = dir.listFiles();

**Get file property:**

File file;

file.getName(); //file name

file.length(); //file size

new Date(file.lastModified()); //file time

**Get file path:**

File file;

file.getAbsolutePath();

**Check whether a file is a directory(folder):**

File file;

file.isDirectory();

# Lab 4: C version

**Week 5: Lab 4 (25 marks)**

**Objective:** Implement file operation in Shell Interface using C

In Lab 1 we have implemented a Shell Interface using Java/C. In Lab 4 we will extend the Shell implementation with several file operation methods.

NOTE: In the C part, if you have any problem in any command, just type (man) followed by the command, the manual page for that command will help you to get more information and options about the command.

**For C:**

A collection of file operation functions are defined in **C**. You can refer to the documentation of file operation in the following link:

http://www.thegeekstuff.com/2012/07/c-file-handling/

**Q1. Implement functions to create, delete, and display a file (10 marks)**

Use the starting code "StartingMain.c" to implement.

For Q1, there are 3 operations for single file: create, delete, and display.

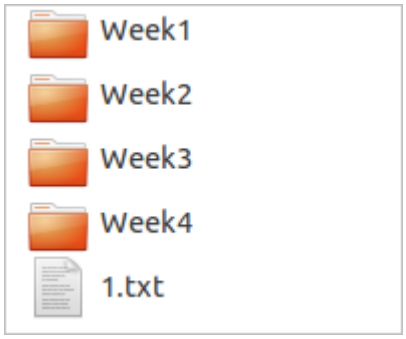
**Create a file:** New files can be created and added to current directory.

For example, now we have 4 folders under current directory:


Under Shell Interface, when we type in the following command:

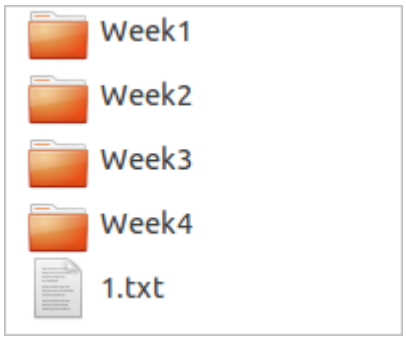
```
csh>create 1.txt
```

A file called "1.txt" will be created under current directory:




**Delete a file:** When a file is no longer needed, we want to remove it from current directory.

For example, now we have 4 folders and 1 file “1.txt” under current directory:




When we type in the following command:

```
csh>delete 1.txt
```

The file “1.txt” will be deleted from current directory:


**Display a file:** When we want to see the content in a file, we use the display command.

By type in this command:

```
csh>display test.txt
```

The content inside “test.txt” will be printed out:

```
Hello.
This is the content inside test.txt file.
```

**Suggested Function:**

| Function | Usage | Needed headers |
|---|---|---|
| main() | *the core of every program and it is required in each c program* | `#include <stdio.h>`<br>`#include<stdlib.h>` |
| *fget* | *To take input from user* | |
| *system* | *Execute command* | |

**NOTE: for remove and display options check the manual page in Linux.**

*man cat*
*man rm*

| Function | Usage | Header |
|---|---|---|
| *FILE *fp;*<br>*fopen=(file.txt,"w");* | *Create writable file with (file.txt) as a name* | `#include <stdio.h>` |
| *rm* | *Remove command* | |
| *cat* | *Display command* | |

**Test Command:**

*csh> create a.txt*
*csh> display a.txt*
*csh> delete a.txt*

**Q2. Implement function to list a directory (5 marks)**

Continue to use the code you implemented in Q1.

Here we design a function to list files under current directory, it should work as the ls command in Linux.

When we type in:

```
csh>list
```

The files (including folders) will be printed out as follows:

```
Week1
Week2
Week3
Week5
```

We also add in the option to show property of files, e.g. file size, last modified time.

When we type in:

```
csh>list property
```

The files (including folders) will be printed out as follows:

```
Week1       Size: 4096        Last Modified: Mon Jan 26 13:10:47 SGT 2015
Week2       Size: 4096        Last Modified: Sun Jan 18 21:09:22 SGT 2015
Week3       Size: 4096        Last Modified: Thu Feb 05 16:43:57 SGT 2015
Week5       Size: 0           Last Modified: Thu Feb 12 16:16:27 SGT 2015
```

The list function should be able to sort the files according to different property.

For example, when we type in:

```
csh>list property time
```

The files (including folders) will be printed out as follows:

```
Week2       Size: 4096        Last Modified: Sun Jan 18 21:09:22 SGT 2015
Week1       Size: 4096        Last Modified: Mon Jan 26 13:10:47 SGT 2015
Week3       Size: 4096        Last Modified: Thu Feb 05 16:43:57 SGT 2015
Week5       Size: 0           Last Modified: Thu Feb 12 16:16:27 SGT 2015
```

The function for sorting file list based on different property (e.g. name, size, time), to understand all the options you need to check the help function for ls:

ls -- help

**Suggested Function:**

| Coomand | Usage | Header |
|---|---|---|
| *ls –S -l* | *Sort by file size* | |
| *ls –t -l* | *Sort by file last modified time* | |
| *Ls -l* | *Sort by file name* | |
| *strtock* | *Parsing the command* | `#include<string.h>` |
| *chdir* | *Change directory* | `#include<unistd.h>` |
| *strcmp* | *Compare two strings* | |
| *strcpy* | *Copy between strings* | |
| *strstr* | *Locate a substring in another string* | |

**Test Command:**

*csh> list*
*csh> list property*
*csh> list property time*

**Q3. Implement function to find files under current directory and subdirectories (5 marks)**

Continue to use the code you implemented in Q2.

Here we design a function to find files under current directory and its subdirectories. For example, the current directory is **C:\CSE_Lab\src**. When we type in the find command with a string:

```
csh>find .txt
```

The function will print out the entries of all files with ".txt" in its name (as a substring in the name). The output is as follows:

```
C:\CSE_Lab\src\Week1\SimpleShell.txt
C:\CSE_Lab\src\Week2\MergeSortThreaded.txt
C:\CSE_Lab\src\Week2\MultiThread.txt
C:\CSE_Lab\src\Week3\Bank.txt
C:\CSE_Lab\src\Week3\BankImpl.txt
C:\CSE_Lab\src\Week3\TestBank.txt
C:\CSE_Lab\src\Week5\FileOperation.txt
```

Note that you need to find files not only in current directory, but also in its subdirectories. So here you can implement a recursive function to find files in different level of subdirectories, or you can check **find –help command**

**Example:**

find –name '*.txt'

**Suggested Function:**

| Function | Usage | header |
|---|---|---|
| *find* | *Search for specific name or string* | |
| *atoi* | *Convert string to integer* | `#include<ctype.h>` |

**Test Command:**

*csh> find .txt*
*csh> find .xyz*

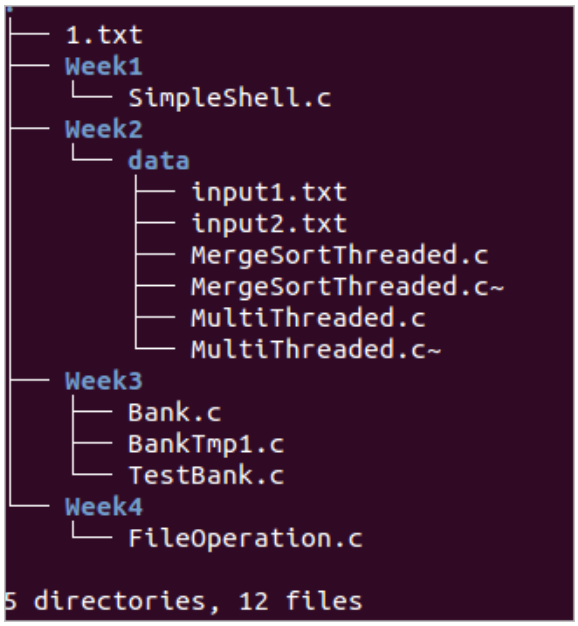
**Q4. Implement function to list subdirectories and files in a tree structure (5 marks)**

Continue to use the code you implemented in Q3.

In order to efficiently show files under different level of directories, we need a function to show files in tree structure. For example, when we type in tree command under directory **C:\CSE_Lab\src**:

```
csh>tree
```

The output should be as follows:




- In Linux, there is a command called (tree), install it if you do not already have it, then check the manual document for it.

  sudo apt-get install tree

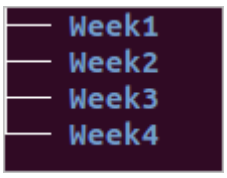
  tree –help

  man tree

- Sometimes we only want to see several top levels of current directories. We should be able to control the maximum level of subdirectories to be shown. For example, when we type in tree command under directory **C:\CSE_Lab\src**:

  ```
  jsh>tree 1
  ```

  The output should be as follows:

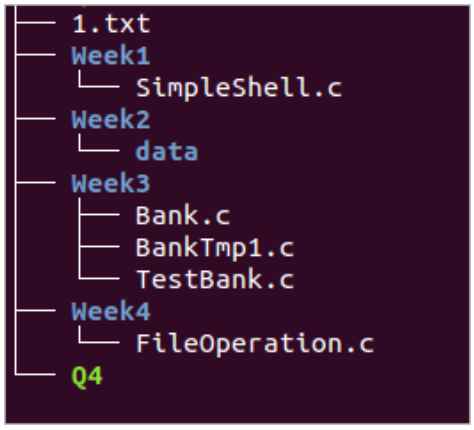
  
  

As you can see here, we set the maximum level of subdirectories to be 1, so only the first level of subdirectories is shown.

Similar to the list command, we can also decide which file property we use to sort the file order to print. For example, when we type in tree command under directory **C:\CSE_Lab\src**:

```
jsh>tree 2 time
```

The output should be as follows:




The tree structure is listed based on time order. So the sequence is different from using name order.

Check the sort option in tree command, examples:

tree --sort=name

tree --sort=ctime

tree --sort=size

(-- : two dashes without space)

**Suggested Function:**

| command | Class which function belongs to |
|---|---|
| *tree* | *List content of directories in a tree-like format* |
| *.listFiles()* | *File* |

**Test Command:**

*csh> tree*
*csh> tree 2*
*csh> tree 5*

The starting codes in C for Lab 4 can be found in eDimension:

```
osboxes@osboxes:~/Desktop/lab4$ gcc StartingMain.c -o StartingMain -g
osboxes@osboxes:~/Desktop/lab4$ ./StartingMain
csh>
```