Programming Assignment 3

Introduction

- **Deadline : 2021.12.08**

- **You have two days for late submission (~2021.12.10)**
  - **25%** deduction per day

- **Submit both source code and Makefile**
  - You will not get a point if your makefile do not build an executable program
  - Command “make” should generate an executable file with name “skku_sh”

Problem 1 (100pt)

■ **Description**

1. Write a program that simulate Linux shell environment.
   - You don’t need to generate real files or directories
   - The program just simulates operations on virtual directories and files
   - root is the highest-level directory in the hierarchy

2. The program simulates 10 Linux commands.
   - mkdir, touch, echo, ls, tree, cat, mv, cp, rmdir, rm
   - You need to implement 10 commands

3. Do not spend your time on input validations

   - Assume that the test cases will always follow the valid input format stated in each command detail

   - For example, test cases always meet below conditions

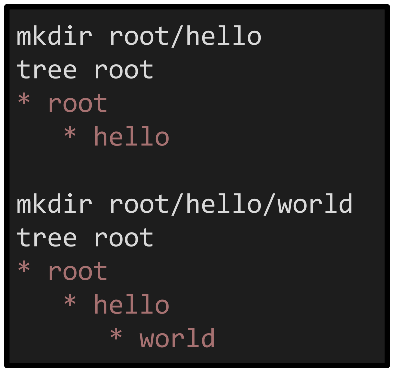
     * The directory and file names only consist of pure alphabets ([a-z A-Z])

     * No whitespace in directory and file names.

Problem 1 (100pt)

■Command #1 : mkdir (10pt)

- Command name : mkdir (this command only works with directories)
- Input Format : ***command path/directoryName*** (i.e., mkdir root/path/to/dir)
- Output Format : None
- Explanation : If there is no directory with the given name in the input path, make a new directory with that name.
- Exception handling : If a directory with the given name already exists in the specified path, print ***"directory already exists"***
- Concrete Program Examples



```
mkdir root/hello
tree root
* root
   * hello

mkdir root/hello/world
tree root
* root
   * hello
      * world
```

make directory

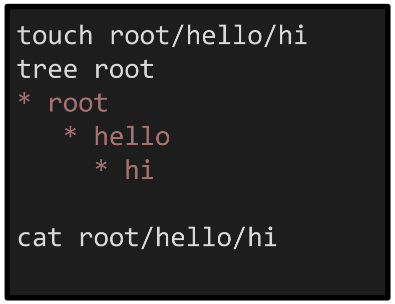
Can check the result via command tree (page 8)

tree output

Problem 1 (100pt)

■Command #2 : touch (10pt)

- Command name : touch (this command only works with files)
- Input Format : ***command path/fileName*** (i.e.,touch root/path/to/file
- Output Format : None
- Explanation : If there is no file with the same name in the given path, create an empty file.
- Exception Handling : If a file with the given name already exists in the specified path, print ***"file already exists"***
- Concrete Program Examples



```
touch root/hello/hi
tree root
* root
   * hello
     * hi

cat root/hello/hi
```

make an empty file named "hi" (assume that root/hello directory already exists)

Can check the result via command tree (page 8)

tree output

no output since 'hi' is an empty file -> cat (page 9)

Problem 1 (100pt)

■ Command #3 : echo (10pt)

- Command name : echo (this command only works with files)
- Input Format : ***command path/fileName content***

  (i.e., echo root/path/to/file hello world)

  => ***content*** is the value that will be stored in the given file.

  => In this example, ***content*** is "hello world"
- Output Format :
  - Does not print anything if a new file is created with the given name and content.
  - If the content of an already existing file is overwritten (include empty file), print "Content updated!"
- Exception Handling : None
- Explanation : If there is a file with the given name in the specified path, *overwrite* the file content with the value provided as ***content.*** If there is no file with the given name, *create a new file* with the value provided as ***content*** in the specified path.

Problem 1 (100pt)

■Command #3 : echo (10pt) cont.

- Concrete Program Examples

```
echo root/hello/bye BYE 2021
tree root
* root
   * hello
      * bye

cat root/hello/bye
BYE 2021

echo root/hello/bye HI 2022
Content updated!

cat root/hello/bye
HI 2022
```

make a file named “bye” with content “BYE 2021” (assume that root/hello directory already exists)

Can check the result via command tree (page 8)

tree output

print the content of the file -> cat (page 9)

file is overwritten

the content of the file is updated -> cat (page 9)

Problem 1 (100pt)

■Command #4 : ls (10pt)

- Command name : ls (this command only works with directories)

- Input Format : ***command path/directoryName***

  (i.e., ls root/path/to/dir)

- Output Format : refer to the screenshot in "Concrete Program Examples" (file/directory should be separated by '\n')

- Explanation : Print all files and directories in that directory.

- Exception handling:
  - If there is no directory with the given name in the specified path, print *"no such directory".*

Problem 1 (100pt)

■Command #4 : ls (10pt)

- Concrete Program Examples

```
mkdir root/hello/pa3
tree root
* root
   * hello
      * pa3
```

make a directory named “pa3” (assume that the root/hello directory already exists)
Can check the result via command **tree** (page 8)
tree output

```
ls root/hello
pa3
```

show all the files/directories in “hello”

```
ls root/hello/pa3
```

no output since there is no file/directory in pa3

```
touch root/hello/pa4
ls root/hello
pa3
pa4
```

Create file named “pa4” using touch command
show all the files/directories in “hello”
the order of file/directory: oldest file/directory comes first, newest file is printed at last

Problem 1 (100pt)

■ Command #5 : tree (10pt)

- Command name : tree (This command only works with directories)
- Input Format : ***command path/directory*** (i.e., tree root/path/to/dir)
- Output Format : refer to the screenshot in “Concrete Program Examples”

  (3 space for directory, 2 space for file)

- Explanation :
  - Print all files and directories under the given directory, and
  - The program should recursively print directories and files in subdirectories as well.

- Exception handling: If there is no directory with the given name, print *“no such directory”*

Problem 1 (100pt)

■Command #5 : tree (10pt)

- Concrete Program Examples

```
tree root
* root
   * hello
      * world
     * greetings

tree root/hello
* hello
   * world
  * greetings

tree root/hello/greetings
no such directory
```

show all the files/directories in the ‘root’ directory

tree output

‘world’ is a directory -> 3 space before *

‘greetings’ is a file -> 2 space before *

the order of file/directory: oldest file/directory comes first, newest file is printed at last

Problem 1 (100pt)

■Command #6 cat (10pt)

- Command name : cat (this command only works with files)
- Input Format : ***command path/fileName*** (i.e., cat root/path/to/file)
- Output Format : print the content of the given file
- Explanation : If there is a file with the given name in the specified path, print the content of that file.
- Exception handling: If there is no file with the given name in the specified path, print *"no such file"*

Problem 1 (100pt)

■Command #6 cat (10pt)

- Concrete Program Examples

```
echo root/hello/bye BYE 2021
tree root
* root
   * hello
     * bye

cat root/hello/bye
BYE 2021

cat root/hello/foo
no such file
```

make a file named “bye” with text “BYE 2021” (assume that the root/hello directory already exists)

tree output

print the content of the file

Error because there is no file named “foo”

Problem 1 (100pt)

■ Command #7 mv (10pt)

- Command name : mv (this command works with both files and directories)

- Input Format :

command source_path/(fileName||directoryName) destination_ path/directoryName

(i.e., mv root/path/to/src_dir root/path/to/dst_dir)

- Output Format : None

- Explanation :
  - move source file or directory to the destination directory.
  - All the resources such as files and directories in the given source directory must also be moved to the destination directory.

- Exception handling:
  - If there is no such file or directory in the given source path, print *“no such file or directory”*
  - If there is no such directory in the given destination path, print *“no such file or directory”*

Problem 1 (100pt)

■Command #7 mv (10pt)

- Concrete Program Examples

```
tree root
* root
   * hello
      * world
     * pa3

mv root/hello/world root
tree root
* root
   * hello
     * pa3
   * world
```

(shown directories/files already exists)

tree output
- **directory: root, hello, world**
- **file: pa3**

move directory “world” to “root”

```
mv root/hello/pa3 root/world
tree root
* root
   * hello
   * world
     * pa3

mv root/hello/pa4 root
no such file or directory
```

move file “pa3” to “world” dir

There is no “pa4” file.
An error message is shown.

Problem 1 (100pt)

■ Command #8 cp (10pt)

- Command name : cp (this command works with both files and directories)

- Input Format :

command source_path/(fileName||directoryName) destination_ path/directoryName

(i.e., cp root/path/to/src_dir root/path/to/dst_dir)

- Output Format : None

- Explanation :
  - ➢ Copy the source file or directory to the destination directory
  - ➢ All the resources such as files and directories in the given source directory must also be copied to the destination directory.

- Exception handling:
  - ➢ If there is no such file or directory in the given source path, print *"no such file or directory"*
  - ➢ If there is no such directory in the given destination path, print *"no such file or directory"*

Problem 1 (100pt)

■ Command #8 cp (10pt)

- Concrete Program Examples

```
tree root
* root
   * hello
      * world
        * pa4
     * pa3
```

(shows the directories and files in the root directory)

tree output
- directory: root, hello, world
- file: pa3 pa4

```
cp root/hello/world root
tree root
* root
   * hello
      * world
        * pa4
     * pa3
   * world
     * pa4
```

copy directory “world” to “root”

```
cp root/hello/pa3 root/world
tree root
* root
   * hello
      * world
        * pa4
     * pa3
   * world
     * pa4
     * pa3
```

copy file “pa3” to root/world directory

tree output

Problem 1 (100pt)

■Command #9 rmdir (10pt)

- Command name : rmdir (this command only works with directories)
- Input Format : ***command path/directoryName*** (i.e., rmdir root/path/to/dir)
- Output Format : None
- Explanation :
  - remove the given directory
  - Should recursively remove directories and files in subdirectories as well
- Exception handling: If there is no directory with the given name, print *“no such directory”*

Problem 1 (100pt)

■Command #9 rmdir (10pt)

- Concrete Program Examples

```
tree root
* root
   * hello
      * world
        * pa4
     * pa3

rmdir root/hello/world
tree root
* root
   * hello
     * pa3
```

(shows the directories and files in the root directory)

tree output

- **directory: root, hello, world**
- **file: pa3 pa4**

remove directory “world”

-> world, pa4 are removed

```
rmdir root/hello
tree root
* root
```

remove directory hello

Problem 1 (100pt)

■Command #10 rm (10pt)

- Command name : rm (this command only works with files)
- Argument Format : ***command path/fileName*** *(*i.e., rm root/path/to/file)
- Output Format : None
- Explanation : remove the given file if the file exists in the specified path.
- Exception handling: If there is no such file, print *“no such file”*
- Concrete Program Examples

```
tree root
* root
   * hello
      * world
        * pa4
     * pa3

rm root/hello/world/pa4
tree root
* root
   * hello
      * world
     * pa3
```

(shows the directories and files in the root directory)
tree output
- **directory: root, hello, world**
- **file: pa3 pa4**

remove file “pa4”
-> pa4 is removed

```
rm root/hello
no such file
```

remove file “hello”
-> no file with name “hello” in root directory
(“Hello” exists in root directory but it is directory not a file.)

Problem 1 (100pt)

■Evaluation

- 10 points for each command.
  - ➢ Total 10*10 = 100 points
  - ➢ You must implement all 10 commands.
  - ➢ We recommend you implement “ls”, “cat”, and “tree” commands first and use them to debug your program.
  - ➢ The concrete program examples in each command(page 4-20) are independent to each other.

Problem 1 (100pt)

■Restriction

- You can only use <iostream>, <string>, <sstream>, <iomanip> library. 0 point if you use other libraries
- 0 point if we cannot compile your program using the given Makefile.
  - Your Makefile should make an executable file in the same directory as your source code. (or 0 point)
  - The executable file name should be "skku_sh"
  - Don't modify folder name. (0 point)
  - Do not modify the given template code. (0 point)
  - The program terminate if user enters "exit" command
  - The program starts with printing root directory. (This code is given in the template)

```
* root
  * hello
```

  - The maximum number of resource that the one directory can hold is 8. (check out the Directory.h, maxcount_ member variable)

Problem 1 (100pt)

■Submission Files

- Makefile
- main.cc
- File.h
- File.cc
- Entry.h
- Entry.cc
- Directory.h
- Directory.cc

Directory.h, Entry.h, File.h file contains the class definition.

* The class definition only includes member variables and declaration of member functions