Cairo University
Faculty of Computers & Information.
Operating Systems 1 Course
Third Year
2019/2020

Assignment #1
Command Line Interpreter

Purpose

An operating system interfaces with a user through a Command Line Interpreter (CLI). A CLI is a software module capable of interpreting textual commands coming either from the **user's keyboard** or from a **script file**. A CLI is often referred to as a shell.

Description

In this assignment, you will write a Command Line Interpreter (CLI) for your operating system. Your CLI should prompt the user to enter the input through the keyboard. After a sequence of characters is entered followed by a return, the string is parsed and the indicated command(s) executed. The user is then again prompted for another command.
Your program implements some built-in commands**; the list of required commands is listed below.** This means that your program must implement these commands directly by using the system calls that implement them. Do not use **exec** to implement any of these commands. The **`exit`** command is also a special case: it should simply cause termination of your program.

For this assignment, the following are essential features for your work

1. Your CLI should be written in Java
2. Your application should contain 2 major classes (Parser, Terminal).

```
// Interface for parser
public class Parser{
        String[] args; // Will be filled by arguments extracted by parse method
        String   cmd; // Will be filled by the command extracted by parse method

        // Returns true if it was able to parse user input correctly. Otherwise false
        // In case of success, it should save the extracted command and arguments
        // to args and cmd variables
        // It should also print error messages in case of too few arguments for a commands
        // eg. "cp requires 2 arguments"
        public boolean parse(String input);
        public String getCmd();
        public String[] getArguments();
}
// Interface for Termianl
public class Terminal{
        public void cp(String sourcePath, String destinationPath );
        public void mv(String sourcePath, String destinationPath);
        public void rm(String sourcePath);
        public void pwd();
        public void cat(String[] paths);
        // Add any other required command in the same structure…..
}
```

// Main implementation is up to you

3. Your CLI should be written in **Java** and as a task function (CLI commands maybe written as functions or tasks).

- All commands and parameters should be entered from the keyboard and **parsed** by your program, **verified**, and then **executed**. If the user enters wrong command or bad parameters the program should print some error messages. For example, if the user writes **mkdir**, the program should response by an error message as the command **mkdir** should have one parameter.
- Your program should handle different parameters for each command. For example, if the user writes **cd C:/** then the program should change to directory **C:/** in case of the current directory is **D:/.** On the other hand, if the user writes **cd** only then the program should change to default directory (defined in your program) which may be **D:/**
- Command parameters are either strings or quoted.
- You should implement the following commands: **clear, cd, ls, cp, mv, rm, mkdir, rmdir, cat, more, pwd.**
- Other commands should be implemented also:

a. **args** - list all parameters on the command line, numbers or strings for specific command. (eg. "args cp" should print "arg1: SourcePath, arg2: DestinationPath")
b. **date** - output current system date and time.
c. **help** - list all user commands and the syntax of their arguments. For example, if the user write **help** command, the program output should be like the following :
help

```
args : List all command arguments
date :  Current date/time
exit  : Stop all
```

- Redirecting should also be implemented (i.e. > and >>) to output the result of command to some file.
- The terminal should allow any "possible" combination of all the above commands using "**|**" pipe operator**.** For example, if the user enters **cd C:/ | pwd** the program should first change the current directory to **C:/** and then display to the user the current working directory which is **C:/.**
- You're required to handle paths using short paths (Relative to the current working directory) and full paths

Submission instructions:

1. **Submission deadline date 19/10 on Acadox**
2. **Discussion time will be arranged by your TA.**
3. **The assignment is submitted in group of maximum 3 students**.
4. **Total grade is 6 + 1 Bonus (Check grading criteria)**

Grading Criteria

Following Parser, Terminal Structure	5
Handling short paths and full paths	5
cd	5
ls	5
cp	5
cat	5
more	5
Pipe Operator	5
Redirect Operator >	5
Redirect Operator >>	5
mkdir	2.5
rmdir	2.5
mv	2.5
rm	2.5
args	2
date	2
help	2
pwd	2
clear	2
Total 70 scaled to 7 (6 Grades + 1 Bonus)	