Programming Assignment #1

Due : 28th Oct. (Friday), 11:59 PM

1. Introduction

In this assignment, you will implement six simple commands in a Linux mini shell.

2. Problem specification

The Objective of PA1 is to implement six Linux commands in C.

The function specification of command is explained in the below. In general, the execution result of commands is required to be the same as the output normally given by the same commands in Linux. You can execute each command in your Linux environment to confirm its output.

In this assignment package, the file *PA1_template.c file* is provided, and you need to implement the functions in that file such as *ls, head, tail, mv, rm, and* pwd to complete the corresponding commands.

2.1 ls

```
>> ls dir_path
>> ls dir_path -al
```

Function specification:

```
int ls(char *dir_path, char *option){
}
```

If the function is executed normally, 0 is returned, otherwise, -1 is returned. If the *-al* option is entered, the option argument is *"-al"*. Otherwise, NULL.

'-al' option should print the output in the following.

```
Permissions / Number of hardlinks /File owner / File group / File size /
File name
```

The output files should be the same as when you put *ls -al* command in Linux.

```
>> ls .
minishell PA1.c a.out test_file.txt cp.c cp mv mv.c PA1_template.c

>> ls . -al
-rwxrwxr-x 1 mjyoo2 mjyoo2 13552 minishell
-rw-rw-r-- 1 mjyoo2 mjyoo2 5198 PA1.c
drwxrwxr-x 2 mjyoo2 mjyoo2 4096 .
-rwxrwxr-x 1 mjyoo2 mjyoo2 8648 a.out
-rw-rw-r-- 1 mjyoo2 mjyoo2   71 test_file.txt
-rw-rw-r-- 1 mjyoo2 mjyoo2  311 cp.c
-rwxrwxr-x 1 mjyoo2 mjyoo2 8480 cp
-rwxrwxr-x 1 mjyoo2 mjyoo2 8520 mv
-rw-rw-r-- 1 mjyoo2 mjyoo2  329 mv.c
drwxrwxr-x14 mjyoo2 mjyoo2 4096 ..
-rw-rw-r-- 1 mjyoo2 mjyoo2 1337 PA1_template.c
```

2.2 head

```
>> head -n K file_path
```

Function specification:

```
int head(char *file_path, char *line){

}
```

If the function is executed normally, 0 is returned, otherwise, -1 is returned. In the **line** argument, (K >0) on the command is input.

The result of using the head command on test_file.txt, which is a text file from line 1 to line 10, is as follows.

```
>> head -n 5 test_file.txt
line 1
line 2
line 3
line 4
line 5
```

2.3 tail

```
>> tail -n K file_path
```

Function specification:

```
int tail(char *file_path, char *line){

}
```

If the function is executed normally, 0 is returned, otherwise, -1 is returned. In the **line** argument, (K >0) on the command is input.

The result of using the tail command on test_file.txt, which is a text file from line 1 to line 10, is as follows.

```
>> tail -n 5 test_file.txt
line 6
line 7
line 8
line 9
line 10
```

2.4 mv

```
>> mv file_path1 file_path2
```

Function specification:

```
int mv(char *file_path1, char *file_path2){

}
```

If the function is executed normally, 0 is returned, otherwise, -1 is returned.

2.5 mkdir

```
>> mkdir dir_path
```

Function specification:

```
int mk_dir(char *dir_path){

}
```

If the function is executed normally, 0 is returned, otherwise, -1 is returned.

2.6 rmdir

```
>> rmdir dir_path
```

Function specification:

```
int rm_dir(char *dir_path){

}
```

If the function is executed normally, 0 is returned, otherwise, -1 is returned. If given directory is not empty, then it should print "***rmdir: failed to remove 'dir_path': Directory is not empty***".

```
root@f7b2aaa334a0:/workspace# mkdir test
root@f7b2aaa334a0:/workspace# vim test/a.py
root@f7b2aaa334a0:/workspace# rmdir test
rmdir: failed to remove 'test': Directory not empty
```

3. Restriction

- The length of file path will be no larger than 200 bytes.
- If the argument name is *file_path*, it is a valid path only if it is a path to file. For *dir_path*, it is a valid path only if it is a path to directory.
- *Head, tail, mv and mkdir* commands will not be tested with any directory in the path. (If a directory is given in the path for those commands, then simply print 'Error: invalid path'.). If the path is invalid, print the following.

```
>> ERROR: invalid path
```

- Including defined error condition (Ex. *Invalid path or rmdir: "Directory is not empty"*), If the command is executed abnormally, this statement should be printed as follows.

```
>> ERROR: The command is executed abnormally
```

This has implemented on the template so that it is automatically output when the function returns -1.

- exec functions and system functions are forbidden.
- For file I/O, using only Unix I/O functions are allowed.

- It is prohibited to use any header file other than the header file already used in the template.

4. Hand in instruction

- You should submit the codes with "student_id.tar.gz".
- Your submission should include `Makefile file`. Please Check if the make command works properly before submission.
    - If your codes are not compiled properly, **you will get no score (or you will get some penalties).**
    - The name of the executable file should be 'minishell'.

5. Logistics

- **Assignment should be done individually.**
- If the assignment is delayed, 10% will be deducted immediately after the deadline, and an additional 10% will be deducted every 24 hours. Delayed submissions will not be accepted after 7 days of submission deadline.
- **Copying assignments will be punished in accordance with the lab's own regulations, and there will be significant disadvantages.**