**Learning Outcome:** Students will have a solid understanding of Python socket programming along with the **socket, subprocess,** and **os** libraries.

## Lab 2: Remote File System Management

Your task in this lab is to write a Python socket program that allows the client to manage a file system in a remote server called FSM server.

Your program must support the following user commands:

1. **pwd** – print working directory
2. **ls** – list current directory contents
3. **mkdir** <arg> – make a directory
4. **write** <file path> – write a file
5. **cat** <file path> – print file contents
6. **cd** <dir> - change directory
7. **quit** – quit the program

The FSM client handles the user commands by interacting with the FSM server that manages the user's entire file system. The client program creates the illusion that the user thinks she manages the file system on her local machine. When the FSM server sends output to the client, the client displays the output to standard output.

Examples:

```
gahyun:client gahyunpark$ python3 FSMclient.py
Enter command: pwd
/Users/gahyunpark/CSEC201/week2-gxpics-1/Lab2/server

Enter command: write file1.txt
> Hello world!
> Bye.
>
file1.txt: saved.
Enter command: cat file1.txt
Hello world!
Bye.
Enter command: mkdir folder1
folder1: directory created
Enter command: ls
 file1.txt,FSMserver.py,folder1,
Enter command: cd folder1
Enter command: ls

Enter command: cd ../
Enter command: ls
 file1.txt,FSMserver.py,folder1,
```

```
Enter command: write folder1/file2.txt
> hi
> there
>
folder1/file2.txt: saved.
Enter command: quit
gahyun:client gahyunpark$
```

**Restrictions:**

1. Create a new directory named **Lab2** under your **Assignment02** directory. Create two sub-directories in Lab2 named **client** and **server**. Create **FSMclient.py** in the **client** directory and **FSMserver.py** in the **server** directory. Yes, the client and server programs must be located separately!

2. You will need to use the **socket**, **subprocess**, and/or **os** modules.

3. Your program must be able to handle invalid user commands. When the user enters an invalid command, your program should display a proper error message and prompt the user to enter a command.

Examples:

```
Enter command: foo
foo: invalid command
Enter command: cat a.txt
cat: a.txt: No such file or directory

Enter command: write folder3/file4.txt
folder3/file4.txt: directory does not exist
Enter command: cd folder3
folder3: dir does not exist
Enter command:
```

4. You may assume that directory or file names do not include blank space and that the file size is at most 4096 bytes.

5. Your final solution should be able to run on remote hosts. Everyone enrolled in CSEC 201 has been provided with RLES VMs. Run FSMserver on Kali Linux VM and FSMclient on Windows VM. You may need to do some extra work such as Python installation on Windows VM, modifying the host IP address in the client program to the IP address of your Kali VM, etc.

6. You may assume that the number of arguments provided by the user is always valid:

- Commands with no arguments: **pwd**, **ls**, **quit**
- Commands with one argument: **cat**, **write**, **cd**, **mkdir**
-

**List of invalid commands you will be tested:**

- Undefined commands
- Make a directory that exists already
- Cat a file that does not exists
- Write a file under a directory that does not exist
- Change directory to a directory that does not exist

**Deliverables:**

1. Upload your final solution FSM_server.py and FSM_client.py to your Assignment 02 GitHub repository before 7 PM, 9/8/2021.
2. You must acquire signoffs from one of the TAs or instructors during office hours before the deadline.

**Sign offs – Each signature is worth 1/N of your lab grade where N is the number of signatures**

- The student was able to use the socket module to implement basic client and server programs.

- The student was able to use the **os** and/or **subprocess** modules to run Linux commands.

- The student was able to handle invalid user commands.

- The student was able to properly manage all user commands.