```
# Lab 5 - network (socket), multi-threading, system
# Names: Trien Bang Huynh and Marcel Gunadi
# server.py

import socket
import os
import pickle
import threading
import sys

HOST = "localhost"
PORT = 5641


CLIENTS, TIME_OUT = int(sys.argv[1]), int(sys.argv[2])

try:
    assert 0 < CLIENTS < 5, "Number of clients should be from 1 - 4"
    assert 2 < TIME_OUT < 31, "Time out should be from 3 - 30 seconds"
except AssertionError as e:
    print(e)
    raise SystemExit


def processRequest(clientInput):
    '''
    A function which processes client input and return objects for sending back to
the client
    '''
    if clientInput[0] == "g":
        return f'Current directory: {clientInput[1]}\n'

    elif clientInput[0] == "c":
        # clientInput[2] is current dir of client
        os.chdir(clientInput[2])
        # clientInput[1] is new dir of client
        newPath = os.path.join(os.getcwd(), clientInput[1])
        try:
            os.chdir(newPath)
            return ("New path: " + os.getcwd() + "\n", os.getcwd())
        except OSError as e:
            return ("Invalid path\n", clientInput[2])

    elif clientInput[0] == "l":
        os.chdir(clientInput[1])  # clientInput[1] is current dir of client
        tempMsg = "Directories and files found under " + os.getcwd() + "\n"
        if len(os.listdir()) == 0:
            return "Empty directory\n"
        for file in os.listdir():
            tempMsg += file + '\n'
        return tempMsg

    elif clientInput[0] == "f":
        os.chdir(clientInput[2])  # clientInput[2] is current dir of client
        # clientInput[1] is new file from client
        if not os.path.exists(clientInput[1]):
            open(clientInput[1], 'w').close()
            return "File created in " + os.getcwd() + "\n"
        else:
            return "File already exists\n"


def processThreading(s, serverDir):
    '''
    A function which creates thread(s) for each client connected to the server and
provoke the process client input function
    '''
    s.settimeout(TIME_OUT)
    try:
        (conn, addr) = s.accept()
        print(f"Connecting to client at port: {PORT}")
        print(f"sending {serverDir}")
        conn.send(pickle.dumps(serverDir))

    except socket.timeout:
        print("Time out. No accept any more client connection.")
        return

    while True:
        fromClient = pickle.loads(conn.recv(1024))
        if fromClient[0] == 'q':
            break
        print("From client:", addr)
        print("Received:", fromClient)

        mesg = processRequest(fromClient)
        conn.send(pickle.dumps(mesg))


with socket.socket() as s:
    s.bind((HOST, PORT))
    print("Server is up, hostname:", HOST, "port:", PORT)
    s.listen()

    serverDir = os.getcwd()

    threads = []

    for i in range(CLIENTS):
        t = threading.Thread(target=processThreading, args=(s, serverDir))
        threads.append(t)
        t.start()

    for t in threads:
        t.join()
```