```cpp
#include<iostream>
#include<bits/stdc++.h>
#include<filesystem>

using namespace std;
namespace fs = filesystem;

//commands to impmlment

//ls
void ls(string dir){
    //listing the files using
    for(auto &p: fs::directory_iterator(dir)){
        // cout<<p.path()<<endl;
        string path = p.path();
        string filename = path.substr(path.find_last_of("/")+1); //getting the
filename as the last pointer
        cout<<filename<<endl;
    }
}

//to return the files in current directory as a vector of strings
vector<string> lsr(string dir){
    //listing the files using
    vector<string> ans;
    for(auto &p: fs::directory_iterator(dir)){
        // cout<<p.path()<<endl;
        string path = p.path();
        string filename = path.substr(path.find_last_of("/")+1); //getting the
filename as the last pointer
        ans.push_back(filename);
    }
    return ans;
}

//cd
void cd(string &currpath, vector<string> command){
    string newpath = currpath + "/" + command[1];
    //handle error
    if(fs::exists(newpath)){
        currpath = newpath;
        cout<<"Current path: "<<currpath<<endl;
        ls(currpath);
    }
    else{
        cout<<"Error: No such file or directory\n";
    }
```

```
}

//error handling
void remove_last(string &dir){
    int i = dir.size()-1;
    while(dir[i]!='/'){
        i--;
    }
    dir = dir.substr(0, i);
    cout<<"Current path: "<<dir<<endl;
}

void go_home(string &dir){
    // used to travel to /home/username
    int i = 0;
    int count=0;
    while(count<3){
        i++;
        if(dir[i]=='/'){
            count++;
        }
    }
    dir = dir.substr(0, i);
    cout<<"Current path: "<<dir<<endl;
}

//touch
void touch(string &dir, string filename){
    string newfile = dir + "/" + filename;
    //if file already exists
    if(fs::exists(newfile)){
        cout<<"Error: File already exists\n";
        return;
    }
    ofstream file(newfile);
    file.close();
    cout<<"File created\n";
}

//history
void hist(vector<string> &command_history){
    for(auto i: command_history){
        cout<<i<<endl;
    }
}


//copy
void cp(string &dir , string filename , string newdir){
    //check for old file
```

```
    if(!fs::exists(dir + "/" + filename)){
        cout<<"Error: No such file or directory\n";
        return;
    }
    if(!fs::exists(newdir)){
        cout<<"Error: No such file or directory\n";
        return;
    }
    string oldfile = dir + "/" + filename;
    string newfile = newdir + "/" + filename;
    fs::copy(oldfile, newfile);
    cout<<"File copied\n";
}




int main(){

    //taking input from the shell and storing it in a vector of strings
    string input;
    vector<string> command_history ;
    //string exitcommand = "";
    string currpath = fs::current_path();
    cout<<"Current path: "<<currpath<<endl;


    while(true){

        //taking input from the shell prompt and storing it in a vector of strings
        cout<<"$ ";
        getline(cin, input);
        vector<string> command;


        //splitting the string into words and storing it's history in a vector
        stringstream ss(input);
        command_history.push_back(ss.str());
        string word;

        //checking for commands
        while(ss>>word){
            command.push_back(word);
        }

        if(command[0] == "exit"){
            cout<<"exiting the shell\n";
            break;
        }
```

```
        else if(command[0]=="hist"){
            hist(command_history);
        }

        else if(command[0]=="clear_hist"){
            command_history.clear();
        }

        else if (command[0]=="ls"){
            if(command.size() >1){
                cout<<"Error: ls command does not take these many arguments\n";
            }
            //ls command function call
            else{
                ls(currpath);
            }
        }

        else if(command[0]=="cd"){
            if(command[1]==".."){
                remove_last(currpath);
            }
            else if(command[1]=="~"){
                go_home(currpath);
            }
            else{
                cd(currpath,command);
            }
        }

        else if(command[0]=="touch"){
            if(command.size() >2){
                cout<<"Error: touch command does not take these many arguments\n";
            }
            else{
                touch(currpath,command[1]);
            }
        }

        else if(command[0]=="cp"){
            if(command.size() >3){
                cout<<"Error: cp command does not take these many arguments\n";
            }
            else{
                string newdir = currpath + "/" + command[2];
                cp(currpath,command[1],newdir);
            }
        }

        else if(command[0] == "clear")
```

```
        {
            cout<<"\033[2J\033[1;1H";
        }

        else{
            //command not found -> system call
            string command_string = "sh -c ";
            for(auto i: command){
                command_string += i + " ";
            }
            int returnCode = system(command_string.c_str());

            // Check if the command was executed successfully
            if (returnCode == 0) {
                std::cout << "Command executed successfully." << std::endl;
            } else {
                std::cout << "Command execution failed or returned non-zero: " <<
returnCode << std::endl;
            }

        }

        //clearing the command vector for the next command
        command.clear();
    }
    return 0;
}
```

-
    -
    -
    -
    -
    -
    -
    -
    -
    -