Task 1 report

In this Task we implemented terminal that fulfils simple tasks such as navigating between directories, checking entities in directories, viewing and setting environment variables and accessing external commands from the main terminal with the use of FORK to run different processess at the same time with the benefit to run some in the background. Finally, we also made some functionality to execute commands written in a file directly to the terminal with the option to print an output file.

GitHub: https://github.com/yesso2004/Operating-sys

In case of failure to open the file, below is the code:

```
#include <iostream>
#include <string>
#include <unistd.h>
#include <dirent.h>
#include <stdlib.h>
#include <vector>
#include <sys/wait.h>
#include <fstream>

using namespace std;

void Split(vector<string> &arg, string cmd)
{
string args = "";

for (char character : cmd)
{
if (character == ' ')
{
if (!args.empty())
```

```
{
arg.push_back(args);
args.clear();
}
}
else
{
args += character;
}
}

if (!args.empty())
{
arg.push_back(args);
}
}

void cd(string Path)
{
if (chdir(Path.c_str()) != 0)
{
perror("Failed to change Directory");
}
else
{
cout << "Directory Changed to : " << Path << endl;
}
}

void ListDirectory(string Path)
{
DIR *dir = opendir(Path.c_str());
if (dir == nullptr)
{
perror("Failed to open directory");
return;
}

struct dirent *Entity;
Entity = readdir(dir);

while (Entity != NULL)
{
```

```
cout << Entity->d_name << endl;
Entity = readdir(dir);
}

closedir(dir);
}

void EnviromentVariable()
{
extern char **environ;

for (char **env = environ; *env != NULL; env++)
{
cout << *env << endl;
}
}

void SetEnvVar(string VarName, string VarValue)
{
if (VarName.empty() || VarValue.empty())
{
cout << "Environment Parameters are empty" << endl;
return;
}

if (setenv(VarName.c_str(), VarValue.c_str(), 1) == 0)
{
cout << "Environment Variable " << VarName << " is set to " << VarValue << endl;
}
else
{
perror("Failed to change Environment Variable");
}
}

void Print(string Input)
{
cout << Input << endl;
}

void HelpMenu()
{
cout << "cd" << '\t' << "Change the directory" << endl;
```

```
cout << "pwd" << '\t' << "print the current directory" << endl;
cout << "dir" << '\t' << "List of all entities in directory" << endl;
cout << "environ" << '\t' << "List of all environment variables" << endl;
cout << "set" << '\t' << "Alter or create a new environment variable with its value" << endl;
cout << "echo" << '\t' << "Print in the terminal" << endl;
cout << "help" << '\t' << "List of all commands with description" << endl;
}

void Pause()
{
cout << "Press Enter to continue... ";
cin.ignore();
cin.get();
}

void ExternalCommands(string Command)
{
vector<string> args;
Split(args, Command);

bool background = false;
if (!args.empty() && args.back() == "&")
{
background = true;
args.pop_back();
}

pid_t id = fork();
if (id == 0)
{
char *argv[args.size() + 1];
for (int i = 0; i < args.size(); ++i)
{
argv[i] = &args[i][0];
}
argv[args.size()] = NULL;

execvp(argv[0], argv);
perror("Not a valid Command");
exit(1);
}
else if (id > 0)
{
```

```
if (!background)
{

wait(NULL);
}
}
else
{
perror("Failed to fork");
}
}

void ExecuteFile(string FileName, string &Path)
{
ifstream file(FileName);
string Command;
ofstream outputFile;
bool saveToFile = false;

if (!file)
{
cout << "Failed to access file" << endl;
return;
}

cout << "Do you want to save output in a file (y/n): ";
char Answer;
cin >> Answer;
cin.ignore();

if (Answer == 'y' || Answer == 'Y')
{
string OutputFileName;
cout << "Enter output file name: ";
getline(cin, OutputFileName);

outputFile.open(OutputFileName, ios::app);
if (!outputFile)
{
cout << "Failed to open output file!" << endl;
return;
}
saveToFile = true;
```

```
}

while (getline(file, Command))
{
if (!Command.empty())
{
cout << "Executing Command: " << Command << endl;
if (saveToFile)
outputFile << "Executing Command: " << Command << endl;
}

if (Command == "quit")
{
if (saveToFile)
outputFile.close();
exit(0);
}
else if (Command == "cd")
{
string NewPath;
if (getline(file, NewPath))
{
cd(NewPath);
Path = NewPath;
if (saveToFile)
outputFile << "Directory changed to: " << Path << endl;
}
}
else if (Command == "pwd")
{
cout << Path << endl;
if (saveToFile)
outputFile << Path << endl;
}
else if (Command == "dir")
{
ListDirectory(Path);
}
else if (Command == "environ")
{
if (saveToFile)
{
streambuf *consoleBuffer = cout.rdbuf();
cout.rdbuf(outputFile.rdbuf());
```

```
EnviromentVariable();
cout.rdbuf(consoleBuffer);
}
else
{
EnviromentVariable();
}
}
else if (Command == "set")
{
string VarName, VarValue;
cout << "Enter Variable to Change: ";
getline(cin, VarName);
cout << "Enter a value: ";
getline(cin, VarValue);

Pause();
SetEnvVar(VarName, VarValue);

if (saveToFile)
outputFile << "Environment Variable " << VarName << " set to " << VarValue << endl;
}
else if (Command == "echo")
{
string Input;
getline(cin, Input);
cout << Input << endl;
if (saveToFile)
outputFile << Input << endl;
}
else if (Command == "help")
{
if (saveToFile)
{
streambuf *consoleBuffer = cout.rdbuf();
cout.rdbuf(outputFile.rdbuf());
HelpMenu();
cout.rdbuf(consoleBuffer);
}
else
{
HelpMenu();
}
}
```

```
else
{
if (saveToFile)
{
streambuf *consoleBuffer = cout.rdbuf();
cout.rdbuf(outputFile.rdbuf());
ExternalCommands(Command);
cout.rdbuf(consoleBuffer);
}
else
{
ExternalCommands(Command);
}
}
}

if (saveToFile)
outputFile.close();
}

int main(int argc, char *argv[])
{

string InitialPath = "/home/ahmed/Desktop/";
string Command;
string Path = InitialPath;

if (argc == 2)
{
string batchfile = argv[1];
ExecuteFile(batchfile, Path);
return 0;
}

cd(InitialPath);

while (true)
{
cout << Path << ": ";
cin >> Command;

if (Command == "quit")
{
```

```
exit(0);
}
else if (Command == "cd")
{
cin.ignore();
cout << "Enter a Directory: ";
getline(cin, Path);
cd(Path);
}
else if (Command == "pwd")
{
cout << Path << endl;
}
else if (Command == "dir")
{
ListDirectory(Path);
}
else if (Command == "environ")
{
EnviromentVariable();
}
else if (Command == "set")
{
cin.ignore();
string VarName;
cout << "Enter Variable to Change: ";
getline(cin, VarName);
string VarValue;
cout << "Enter a value: ";
getline(cin, VarValue);

Pause();

SetEnvVar(VarName, VarValue);
}
else if (Command == "echo")
{
cin.ignore();
string Input;
getline(cin, Input);

Print(Input);
}
```

```
else if (Command == "help")
{
HelpMenu();
}
else if (Command == "execute")
{
cin.ignore();
string FileName;
cout << "Enter a file to extract and execute commands: ";
getline(cin, FileName);

ExecuteFile(FileName, Path);
}
else
{
ExternalCommands(Command);
}
}

return 0;
}
```