using System;

using System.Collections.Generic;

class FileItem

{

public string Name;

public bool IsFolder;

public List<FileItem> Children = new List<FileItem>();

public FileItem Parent;

public FileItem(string name, bool isFolder, FileItem parent = null)

{

Name = name;

IsFolder = isFolder;

Parent = parent;

}

public void Print()

{

Console.WriteLine(IsFolder ? $"[{Name}]" : Name);

}

}

class FileManager

{

private FileItem current;

private FileItem root;

public FileManager()

{

root = new FileItem("C:", true);

current = root;

}

public void Run()

{

while (true)

{

Console.Write($"{GetPath()}> ");

string input = Console.ReadLine().Trim();

if (input == "exit") break;

try

{

ProcessCommand(input.Split(' '));

}

catch (Exception e)

{

Console.WriteLine($"Error: {e.Message}");

}

}

}

private string GetPath()

{

var path = new List<string>();

var temp = current;

while (temp != null)

{

path.Insert(0, temp.Name);

temp = temp.Parent;

}

return string.Join("/", path);

}

private void ProcessCommand(string[] parts)

{

switch (parts[0].ToLower())

{

case "mkdir":

CreateFolder(parts[1]);

break;

case "create":

CreateFile(parts[1]);

break;

case "rm":

Remove(parts[1], true);

break;

case "del":

Remove(parts[1], false);

break;

case "dir":

ShowContents();

break;

case "cd":

ChangeDir(parts[1]);

break;

default:

Console.WriteLine("Unknown command");

break;

}

}

private void CreateFolder(string name)

{

if (current.Children.Exists(x => x.Name == name))

throw new Exception("Already exists");

current.Children.Add(new FileItem(name, true, current));

Console.WriteLine($"Folder '{name}' created");

}

private void CreateFile(string name)

{

if (current.Children.Exists(x => x.Name == name))

throw new Exception("Already exists");

current.Children.Add(new FileItem(name, false, current));

Console.WriteLine($"File '{name}' created");

}

private void Remove(string name, bool isFolder)

{

var item = current.Children.Find(x => x.Name == name);

if (item == null) throw new Exception("Not found");

if (item.IsFolder != isFolder) throw new Exception(isFolder ? "Not a folder" : "Not a file");

if (item.IsFolder && item.Children.Count > 0)

{

Console.Write($"Delete folder '{name}' with contents? (y/n): ");

if (Console.ReadLine().ToLower() != "y") return;

}

current.Children.Remove(item);

Console.WriteLine($"Deleted '{name}'");

}

private void ShowContents()

{

if (current.Children.Count == 0)

{

Console.WriteLine("Empty");

return;

}

foreach (var item in current.Children)

item.Print();

}

private void ChangeDir(string name)

{

if (name == "..")

{

if (current.Parent != null)

current = current.Parent;

return;

}

if (name == "/")

{

current = root;

return;

}

var folder = current.Children.Find(x => x.Name == name && x.IsFolder);

if (folder == null) throw new Exception("Folder not found");

current = folder;

}

}

class Program

{

static void Main()

{

new FileManager().Run();

}

}