**3 Практическое программирование на языке C#**

**3.1 Постановка задачи**

Разработать приложение аналог эмулятора терминала Linux. Программа предназначена для взаимодействия с операционной системой посредством консольного ввода-вывода.

Основными функции программы являются:

* Отображение информации о текущем пользователе и рабочей папке;
* Использование команд;
* Возможность просмотра документации и вспомогательной информации по командам;
* Возможность выхода и очищение консоли.

**3.2 Описание программных модулей**

**Запуск программы**

Для запуска программы необходимо открыть файл Terminal.exe.

В окне терминала отображается информация о текущем пользователе, имени компьютера и текущая рабочая папка. (рисунок 3.1)

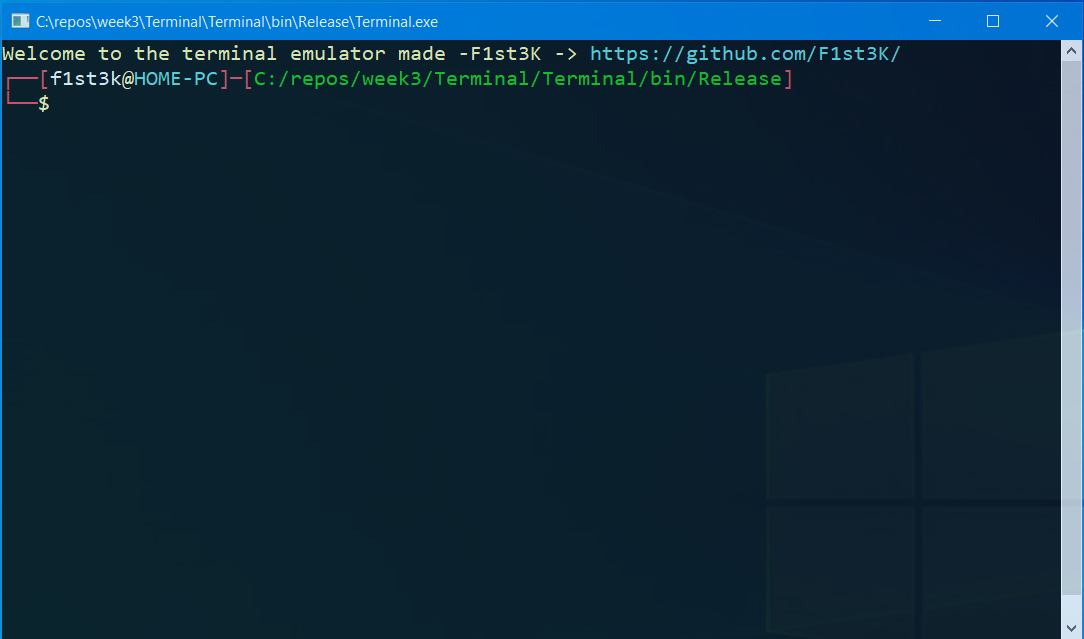


Рисунок 3.1 – Окно терминала

**Использование команд**

Для использования команды необходимо ввести ее название а далее через пробел вводить операторы и ключи которые вам необходимы, и нажать Enter. (Рисунок 3.2)

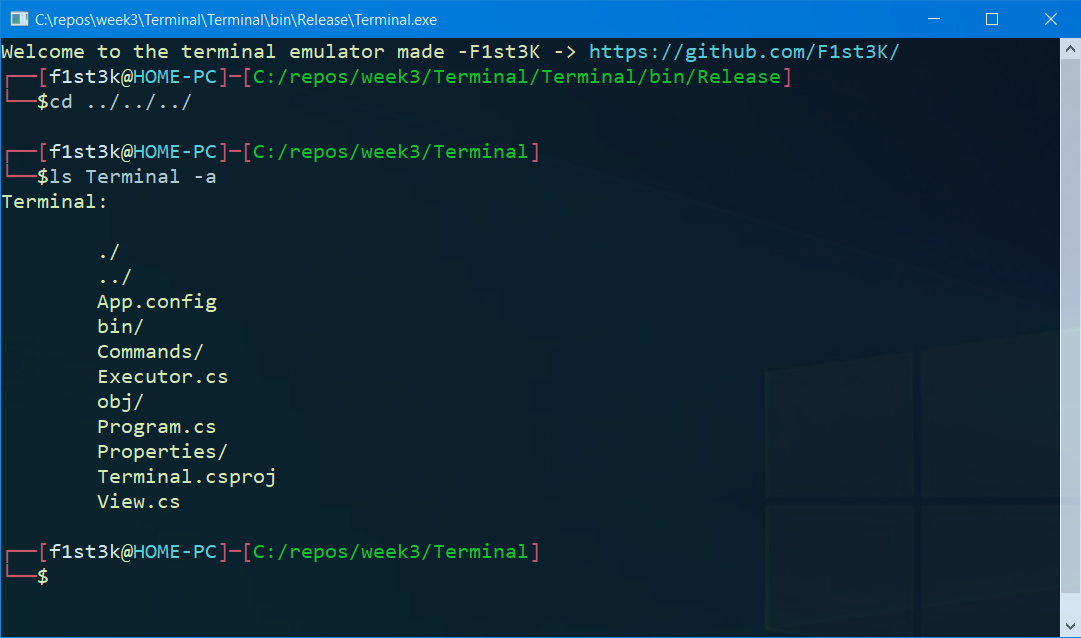


Рисунок 3.2- Использование команды ls

**Просмотр вспомогательной информации по команде**

Для просмотра вспомогательной информации по команде, необходимо ввести ее название а далее через пробел ввести ключ --help, и нажать Enter. (Рисунок 3.3)

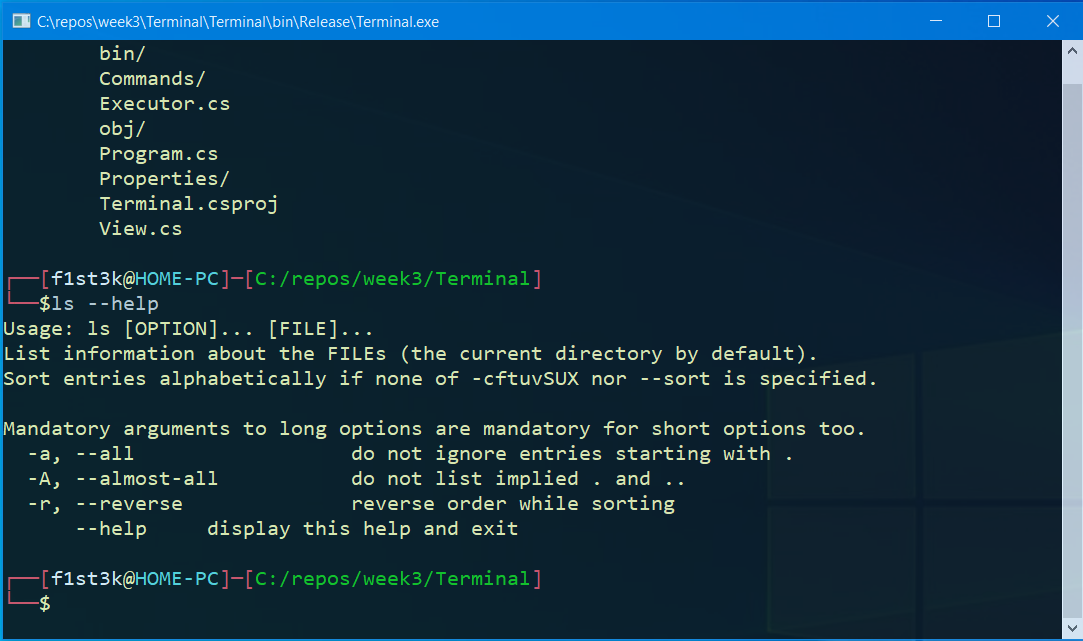


Рисунок 3.3 – Вспомогательная информация по команде ls

**Просмотр документации по команде**

Для просмотра документации по команде, необходимо ввести man а далее через пробел ввести название команды, и нажать Enter. (Рисунок 3.4) После чего начнется режим просмотра документации, перемещение осуществляется посредством стрелок вверх и вниз соответственно. (Рисунок 3.5) Что бы выйти из режима просмотра нажмите клавишу Q. (Рисунок 3.6)

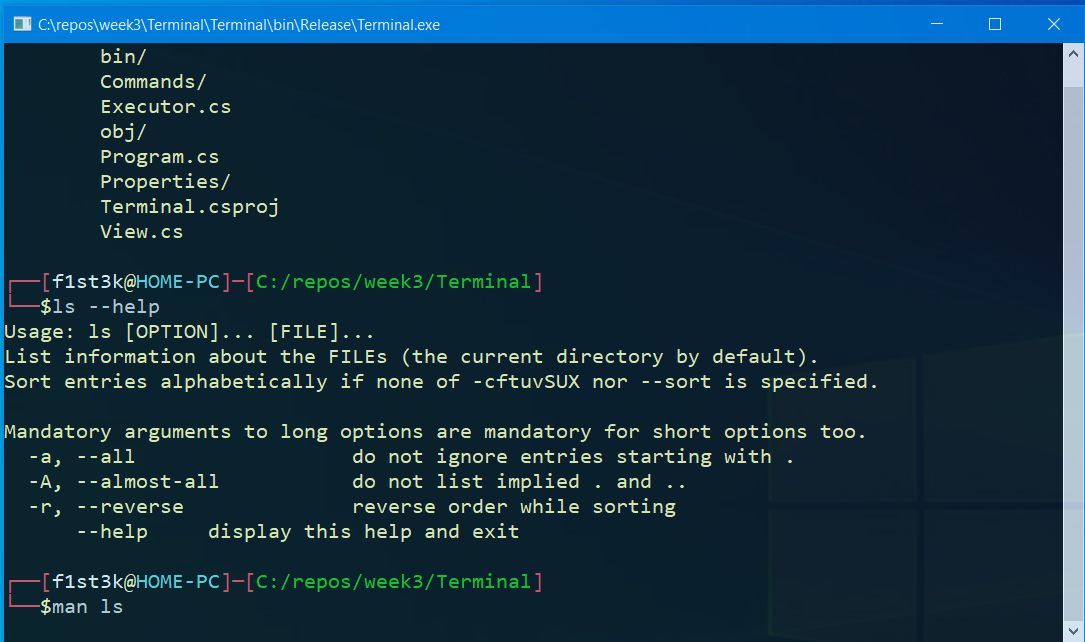


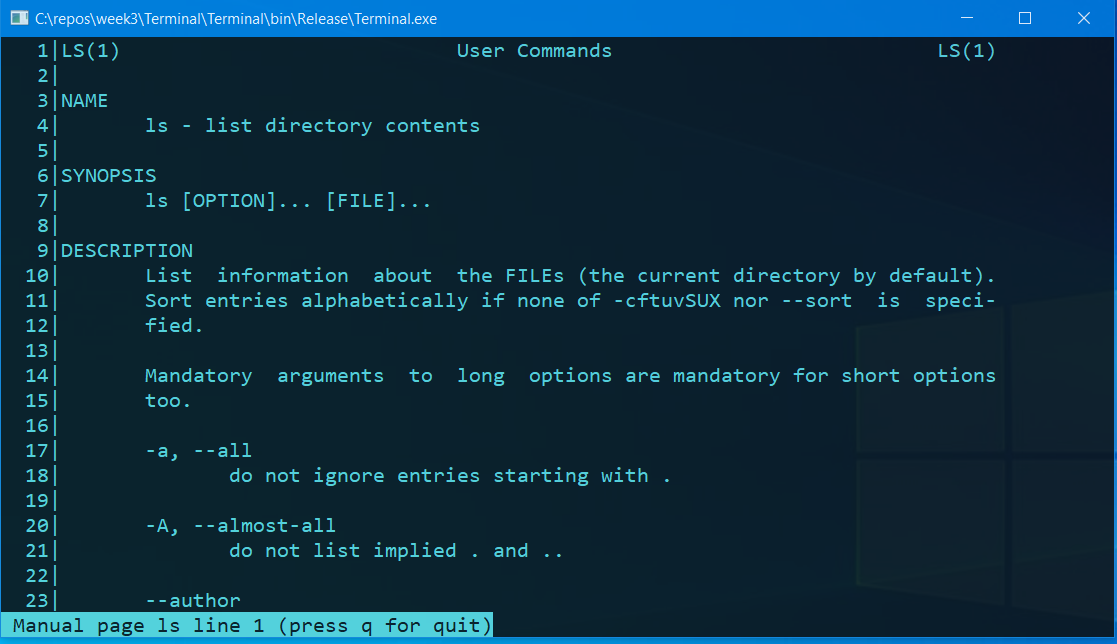
Рисунок 3.4 – Команда man

Рисунок 3.5 – Режим просмотра документации

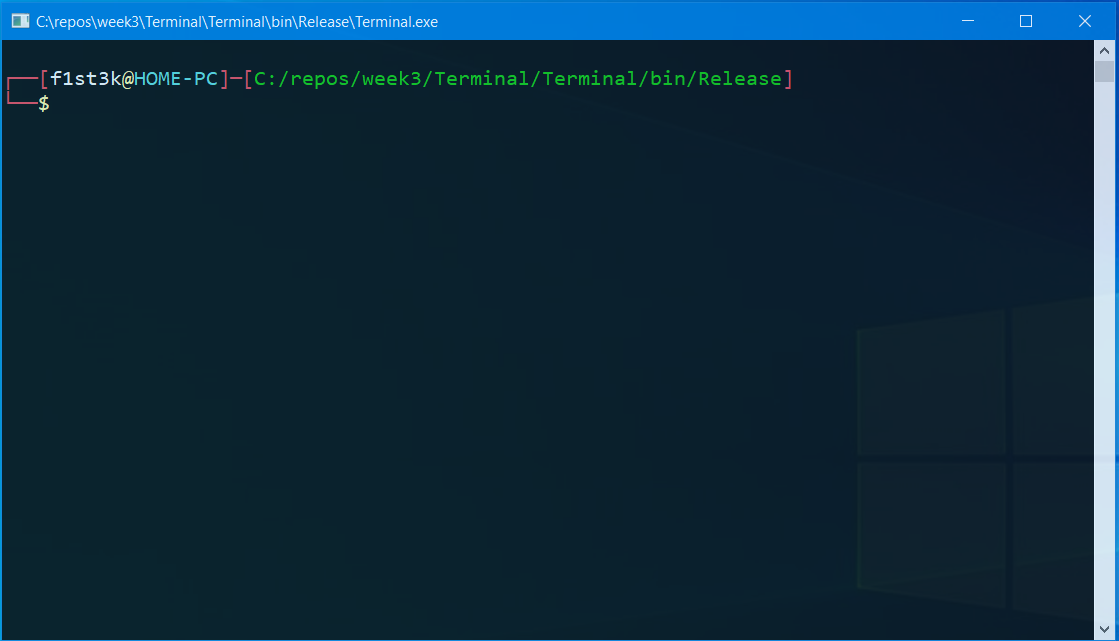


Рисунок 3.6 – Выход из режима просмотра документации

**Очищение консоли**

Для очистки консоли необходимо ввести команду clear, и нажать Enter. (Рисунок 3.7)

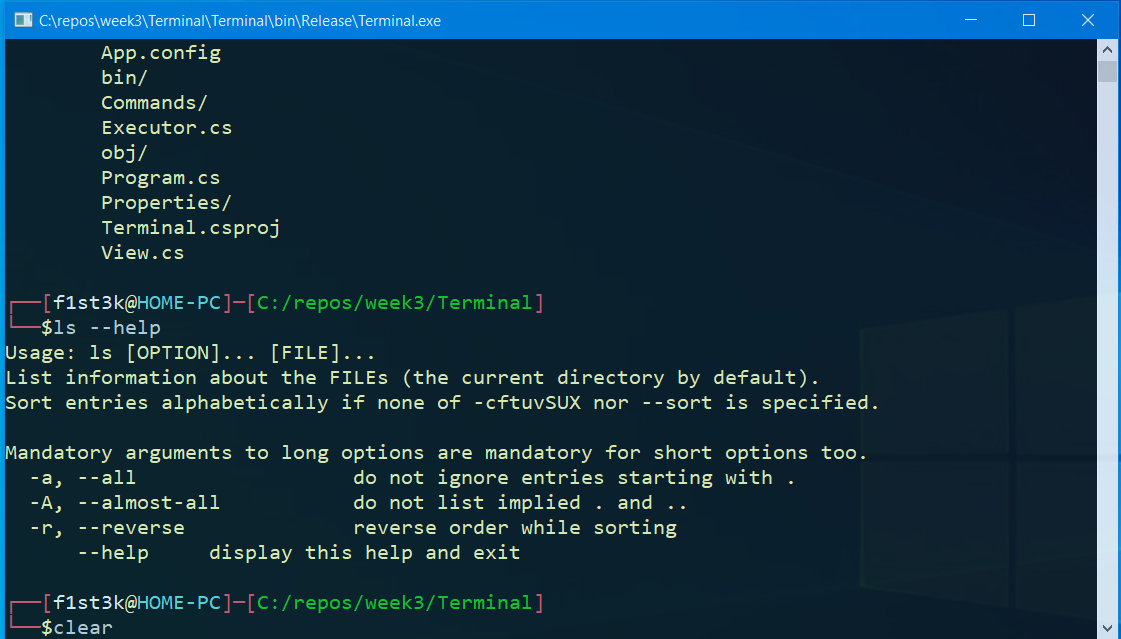


Рисунок 3.7 – Очищение консоли

**Выход из программы**

Для выхода из программы необходимо ввести команду exit, и нажать Enter. (Рисунок 3.8)

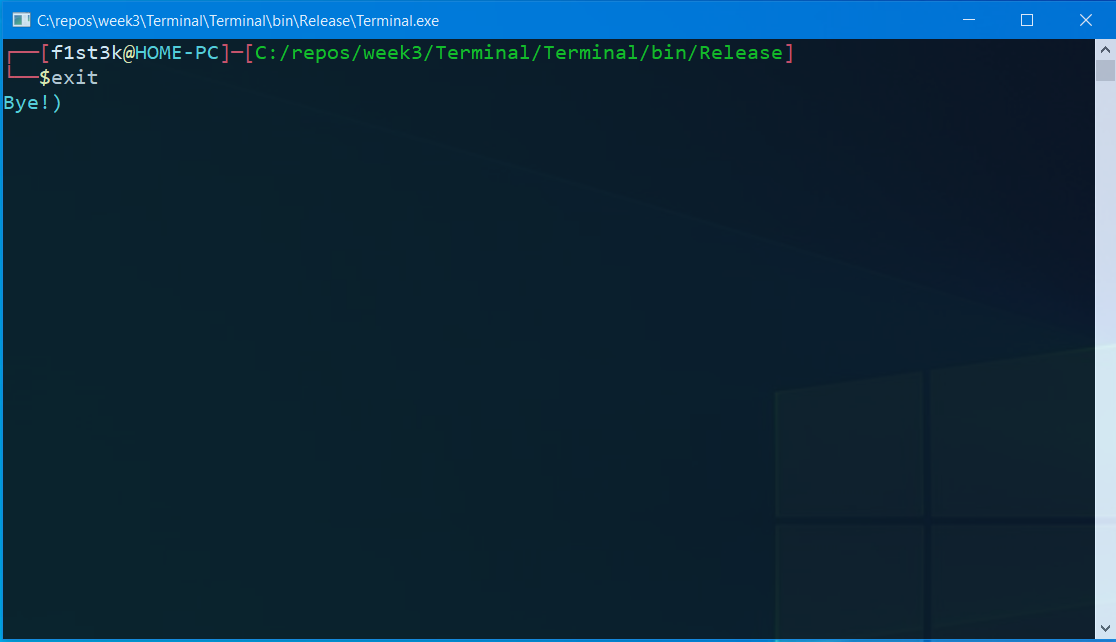


Рисунок 3.8 – Завершение работы программы

**3.4 Листинг программы**

**Program.cs** - точка входа в программу

namespace Terminal

{

/// <summary>

/// Class <c>Program</c> start point programm.

/// </summary>

class Program

{

//Run separate Task

static async System.Threading.Tasks.Task Main(string[] args)

{

await System.Threading.Tasks.Task.Run(() => Work());

}

//Main cycle

static void Work()

{

var view = new View();

var executor = new Executor();

view.Start();

while (true)

try

{

string[] commands = view.Run().Split(new char[] {' '}, System.StringSplitOptions.RemoveEmptyEntries);

if (commands.Length == 0)

continue;

switch (commands[0])

{

case "exit":

view.Stop();

return;

case "clear":

view.Clear();

continue;

default:

break;

}

if (executor.CommandIsExist(commands[0]))

view.Output(executor.RunCommand(commands));

else view.CommandNotFound(commands[0]);

}

catch

{

view.Output("Terminal: ooops..");

continue;

}

}

}

}

View.cs - Класс для вывода на консоль

using System;

using System.IO;

namespace Terminal

{

/// <summary>

/// Class <c>View</c> view console.

/// </summary>

internal sealed class View

{

//Start message

public void Start()

{

WriteColor("Welcome to the terminal emulator made -F1st3K -> ", ConsoleColor.Yellow);

WriteColor("https://github.com/F1st3K/\n", ConsoleColor.Cyan);

}

//View information

public string Run()

{

WriteColor("┌──[", ConsoleColor.Red);

WriteColor(Environment.UserName, ConsoleColor.White);

WriteColor("@", ConsoleColor.Yellow);

WriteColor(Environment.UserDomainName, ConsoleColor.Cyan);

WriteColor("]─[", ConsoleColor.Red);

WriteColor(Directory.GetCurrentDirectory().Replace("\\", "/"), ConsoleColor.Green);

WriteColor("]\n└──", ConsoleColor.Red);

WriteColor("$", ConsoleColor.Yellow);

return Console.ReadLine();

}

//Output console

public void Output(string value)

{

WriteLineColor(value, ConsoleColor.Yellow);

}

//Clear console

public void Clear()

{

Console.Clear();

}

//Error message

public void CommandNotFound(string command)

{

WriteLineColor(command + ": command not found", ConsoleColor.Red);

}

//Stop message

public void Stop()

{

WriteLineColor("Bye!)", ConsoleColor.Cyan);

}

//Write color console

private void WriteColor(string value, ConsoleColor color)

{

Console.ForegroundColor = color;

Console.Write(value);

Console.ResetColor();

}

//write line color console

private void WriteLineColor(string value, ConsoleColor color)

{

Console.ForegroundColor = color;

Console.WriteLine(value);

Console.ResetColor();

}

}

}

Executor.cs - Класс исполнитель команд

using System;

using System.Collections.Generic;

using Terminal.Commands;

namespace Terminal

{

/// <summary>

/// Class <c>Executor</c> execute commands.

/// </summary>

internal sealed class Executor

{

//Diction containe all commands

private Dictionary<string, Func<string[], string>> \_listCommands = new Dictionary<string, Func<string[], string>>()

{

{ "ls", new ListFiles("ls").Run},

{ "cd", new ChangeDirectory("cd").Run},

{ "pwd", new PrintWorkingDirectory("pwd").Run},

{ "arch", new Architecture("arch").Run},

{ "mkdir", new MakeDirectory("mkdir").Run},

{ "rmdir", new RemoveDirectory("rmdir").Run},

{ "cat", new Concatenate("cat").Run},

{ "head", new Head("head").Run},

{ "tail", new Tail("tail").Run},

{ "date", new Date("date").Run},

{ "rm", new Remove("rm").Run},

{ "touch", new Touch("touch").Run},

{ "ps", new Processes("ps").Run},

{ "kill", new Kill("kill").Run},

{ "wc", new WordCount("wc").Run},

{ "man", new Manual("man").Run},

{ "cp", new Copy("cp").Run},

{ "uname", new Uname("uname").Run},

{ "du", new DiskUsage("du").Run},

{ "df", new DiskFree("df").Run},

};

public bool CommandIsExist(string value)

{

return \_listCommands.ContainsKey(value);

}

//Run command in dictionary

public string RunCommand(string[] values)

{

if (values.Length < 1)

throw new ArgumentException("values is no null");

\_listCommands.TryGetValue(values[0], out var command);

string[] arguments = new string[values.Length - 1]; ;

for (int i = 1, j = 0; i < values.Length; i++, j++)

arguments[j] = values[i];

return command.Invoke(arguments);

}

}

}

Command.cs - Базовый класс для остальных команд

using System;

using System.Collections.Generic;

using System.IO;

using System.Reflection;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>Command</c> base class for commands.

/// </summary>

internal class Command

{

//Current key on run command

private protected List<string> \_keys;

//Current values on run command

private protected List<string> \_values;

//Method start command

private protected Func<string> \_command;

//Name in man

private protected string \_name;

//Possible keys

public List<string> PossibleKeys { get; private protected set; }

public Command(string name)

{

\_keys = new List<string>();

\_values = new List<string>();

PossibleKeys = new List<string>();

\_name = name;

PossibleKeys.Add("--help");

}

//Start command

public string Run(string[] arguments)

{

SeparateArguments(arguments);

var check = CheckKeys();

if (check != String.Empty)

return check;

if (\_keys.Contains("--help"))

try

{

var start = Directory.GetParent(Assembly.GetExecutingAssembly().Location);

return File.ReadAllText(start + "\\help\\" + \_name + ".help");

}

catch

{

return "No such help information";

}

return \_command.Invoke();

}

//Separate value without keys

private protected void SeparateArguments(string[] args)

{

\_keys.Clear();

\_values.Clear();

foreach (var arg in args)

{

if (arg.ToCharArray()[0] == '-')

\_keys.Add(arg);

else \_values.Add(arg);

}

}

//Check valid keys

private protected string CheckKeys()

{

foreach (var key in \_keys)

{

if (PossibleKeys.Contains(key) == false)

return "Invalid key " + key;

}

return String.Empty;

}

}

}

Architecture.cs - команда arch

using System;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>Architecture</c> print architecure.

/// </summary>

internal class Architecture : Command

{

public Architecture(string name) : base(name)

{

\_command = Start;

}

private string Start()

{

string output = string.Empty;

output += Environment.Is64BitOperatingSystem ? "x86\_64" : "x86";

return output;

}

}

}

ChangeDirectory.cs - команда cd

using System.IO;

namespace Terminal.Commands

{

internal class ChangeDirectory : Command

{

/// <summary>

/// Class <c>ChangeDirectory</c> change current dirrectory.

/// </summary>

public ChangeDirectory(string name) : base(name)

{

\_command = Start;

}

private string Start()

{

string output = string.Empty;

string current = Directory.GetCurrentDirectory();

if (\_values.Count > 1)

return "cd: too many arguments";

if (\_values.Count <= 0)

{

Directory.SetCurrentDirectory("\\");

return output;

}

try

{

if (Path.IsPathRooted(\_values[0]))

Directory.SetCurrentDirectory(\_values[0]);

else

Directory.SetCurrentDirectory(current + "\\" + \_values[0]);

}

catch

{

return "cd:\t" + \_values[0] + ": No such file or directory";

}

return output;

}

}

}

Concatenate.cs - команда cat

using System;

using System.Collections.Generic;

using System.IO;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>Concatenate</c> output and input files.

/// </summary>

internal class Concatenate : Command

{

public Concatenate(string name) : base(name)

{

PossibleKeys.Add("->");

PossibleKeys.Add("-b");

PossibleKeys.Add("-T");

\_command = Start;

}

public static string[] Read(string path)

{

string current = string.Empty;

if (Path.IsPathRooted(path) == false)

current = Directory.GetCurrentDirectory() + "\\";

if (File.Exists(current + path) == false)

throw new ArgumentException("No such file or directory ");

return File.ReadAllLines(current + path);

}

private string Start()

{

if (\_keys.Contains("->"))

return WriteFile();

else

{

if (\_values.Count <= 0)

return OnlineMode();

return ReadFile();

}

}

private string OnlineMode()

{

while (true)

{

var line = Console.ReadLine();

if (line == "^exit")

break;

line = ChangeView(new string[] { line })[0];

Console.WriteLine(line);

}

return string.Empty;

}

private string ReadFile()

{

var output = string.Empty;

foreach (var filename in \_values)

{

try

{

var lines = Read(filename);

lines = ChangeView(lines);

foreach (var line in lines)

output += "\n" + line;

}

catch

{

return "cat: " + filename.Replace("\\", "/") + ": No such file or directory ";

}

}

return output;

}

private string[] ChangeView(string[] lines)

{

for (int i = 0; i < lines.Length; i++)

{

if (\_keys.Contains("-T"))

lines[i] = lines[i].Replace(" ", "^I");

if (\_keys.Contains("-b"))

lines[i] = $"{(i+1), 4} |" + lines[i];

}

return lines;

}

private string WriteFile()

{

if (\_values.Count <= 0)

return "syntax error near unexpected token `newline'";

var listLines = new List<string>();

while (true)

{

var line = Console.ReadLine();

if (line == "^exit")

break;

listLines.Add(line);

}

var lines = ChangeView(listLines.ToArray());

string current = string.Empty;

foreach (var filename in \_values)

{

if (Path.IsPathRooted(filename) == false)

current = Directory.GetCurrentDirectory() + "\\";

File.WriteAllLines(current + filename, lines);

}

return string.Empty;

}

}

}

Copy.cs - команда copy

using System;

using System.IO;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>Copy</c> copy files and directoryes.

/// </summary>

internal class Copy : Command

{

public Copy(string name) : base(name)

{

PossibleKeys.Add("-f");

PossibleKeys.Add("-v");

PossibleKeys.Add("-i");

PossibleKeys.Add("-n");

\_command = Start;

}

private string Start()

{

string output = string.Empty;

try

{

string current = string.Empty;

if (Path.IsPathRooted(\_values[\_values.Count-1]) == false)

current = Directory.GetCurrentDirectory() + "\\";

string destFile = current + \_values[\_values.Count - 1];

if (\_values.Count == 2 && File.Exists(current + \_values[0]))

{

if (File.Exists(destFile))

{

if (\_keys.Contains("-n"))

return output;

if (\_keys.Contains("-i") && Confirmation("cp: overwrite " + destFile + "?") == false)

return output;

}

File.Copy(current + \_values[0], destFile, true);

if (\_keys.Contains("-v"))

output += "\n" + \_values[0] + " -> " + destFile;

return output;

}

if (Directory.Exists(destFile) ||

Directory.Exists(Directory.GetParent(destFile).FullName))

{

for (int i = 0; i < \_values.Count - 1; i++)

{

current = string.Empty;

if (Path.IsPathRooted(\_values[i]) == false)

current = Directory.GetCurrentDirectory() + "\\";

if (File.Exists(current + \_values[i]))

{

CopyFile(current + \_values[i], destFile);

if (\_keys.Contains("-v"))

output += "\n" + \_values[i] + " -> " + destFile;

}

else if (Directory.Exists(current + \_values[i]))

{

CopyDirectory(current + \_values[i], destFile);

if (\_keys.Contains("-v"))

output += "\n" + \_values[i] + " -> " + destFile;

}

else

{

throw new ArgumentException(\_values[i] + ": No such file or directory ");

}

}

}

else

throw new ArgumentException(\_values[\_values.Count - 1] + ": No such directory ");

}

catch (Exception ex)

{

return "cp: " + ex.Message;

}

return output;

}

private void CopyDirectory(string path, string dest)

{

var dir = (path).Split(new string[] { "\\" }, StringSplitOptions.RemoveEmptyEntries);

var nameDir = dir[dir.Length - 1];

Directory.CreateDirectory(dest +"\\" + nameDir);

var files = Directory.GetFileSystemEntries(path);

for (int i = 0; i < files.Length; i++)

{

if (File.Exists(files[i]))

CopyFile(files[i], dest + "\\" + nameDir);

else if (Directory.Exists(files[i]))

CopyDirectory(files[i], dest + "\\" + nameDir);

else continue;

}

}

private void CopyFile(string path, string dest)

{

var f = (path).Split(new string[] { "\\" }, StringSplitOptions.RemoveEmptyEntries);

var fileName = f[f.Length - 1];

if (dest.Substring(dest.Length - 1) != "\\")

dest += "\\";

if (File.Exists(dest + fileName))

{

if (\_keys.Contains("-n"))

return;

if (\_keys.Contains("-i") && Confirmation("cp: overwrite " + fileName + "?") == false)

return;

}

File.Copy(path, dest + fileName, true);

}

private bool Confirmation(string value)

{

while (true)

{

Console.Write(value);

var key = Console.ReadLine();

if (key == "y")

return true;

if (key == "n")

return false;

}

}

}

}

Date.cs - команда date

using System;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>Date</c> show date in custom format.

/// </summary>

internal class Date : Command

{

public Date(string name) : base(name)

{

\_command += Start;

}

private string Start()

{

string output = string.Empty;

if (\_values.Count <= 0)

return DateTime.Now.ToLongDateString() +" "+ DateTime.Now.ToLongTimeString();

else

{

foreach (var date in \_values)

{

if (date.ToCharArray()[0] != '+')

return "date: invalid date " + date;

return DateTime.Now.ToString(date.Replace("+", string.Empty).Replace("%", " "));

}

}

return output;

}

}

}

DiskFree.cs - команда df

using System;

using System.Collections.Generic;

using System.IO;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>DiskFree</c> show information for disks.

/// </summary>

internal class DiskFree : Command

{

public DiskFree(string name) : base(name)

{

PossibleKeys.Add("-a");

PossibleKeys.Add("-k");

PossibleKeys.Add("-H");

\_command = Start;

}

private string Start()

{

string output = string.Empty;

var drives = new List<DriveInfo>();

if (\_keys.Contains("-a"))

{

drives.AddRange(DriveInfo.GetDrives());

}

else

{

if (\_values.Count <= 0)

\_values.Add(Directory.GetCurrentDirectory());

foreach (var file in \_values)

try

{

var directory = new DirectoryInfo(file);

drives.Add(new DriveInfo(directory.FullName.Substring(0, 1)));

}

catch (Exception ex)

{

output += "\n du: " + file + ": " + ex.Message;

}

}

output += $"{"Name",6} {"Total",12} {"Used",12} {"Avaliable",12} {"Use",3}% Mounted on";

foreach (var d in drives)

{

var total = d.TotalSize;

var used = d.TotalSize - d.AvailableFreeSpace;

var free = d.AvailableFreeSpace;

var percent = (float)used / (float)total \* 100;

string unit = "";

if (\_keys.Contains("-k"))

{

total = total / 1024;

used = used / 1024;

free = free / 1024;

unit = "KB";

}

else if (\_keys.Contains("-H"))

{

total = total / (1024 \* 1024 \* 1024);

used = used / (1024 \* 1024 \* 1024);

free = free / (1024 \* 1024 \* 1024);

unit = "GB";

}

output += $"\n{d.Name, 6} {total + unit, 12} {used + unit,12} {free + unit,12} {Math.Truncate(percent),3}% {d.DriveFormat}";

}

return output;

}

}

}

DiskUsage.cs - команда du

using System;

using System.IO;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>DiskUsage</c> show usage memory directoryes and files.

/// </summary>

internal class DiskUsage : Command

{

public DiskUsage(string name) : base(name)

{

PossibleKeys.Add("-a");

PossibleKeys.Add("-c");

PossibleKeys.Add("-h");

\_command = Start;

}

private string Start()

{

string output = string.Empty;

if (\_values.Count <= 0)

\_values.Add(Directory.GetCurrentDirectory());

long total = 0;

foreach (var file in \_values)

try

{

string current = string.Empty;

if (Path.IsPathRooted(\_values[\_values.Count - 1]) == false)

current = Directory.GetCurrentDirectory() + "\\";

output += RecursiveEntry(current + file);

total += DirSize(new DirectoryInfo(current + file));

}

catch (Exception ex)

{

output += "\n du: "+ file + ": " + ex.Message;

continue;

}

if (\_keys.Contains("-c"))

{

var size = \_keys.Contains("-h") ? (total / 1024).ToString() + "KB" : total.ToString();

output += $"\n {size,6}\ttotal";

}

return output;

}

private string RecursiveEntry(string path)

{

string output = string.Empty;

if (Directory.Exists(path))

{

var d = new DirectoryInfo(path);

string size = \_keys.Contains("-h") ? (DirSize(d) / 1024).ToString() + "KB" : DirSize(d).ToString();

output += $"\n {size,6}\t" +

$"{(path).Replace(Directory.GetCurrentDirectory() + "\\", string.Empty).Replace("\\", "/")}";

foreach (var directory in d.GetDirectories())

{

output += RecursiveEntry(directory.FullName);

}

if (\_keys.Contains("-a"))

foreach (var directory in d.GetFiles())

{

output += RecursiveEntry(directory.FullName);

}

}

else if (File.Exists(path))

{

string size = \_keys.Contains("-h") ? (File.ReadAllBytes(path).Length / 1024).ToString() + "KB" : File.ReadAllBytes(path).Length.ToString();

output += $"\n {size,6}\t" +

$"{(path).Replace(Directory.GetCurrentDirectory() + "\\", string.Empty).Replace("\\", "/")}";

}

return output;

}

public static long DirSize(DirectoryInfo d)

{

long Size = 0;

// Add file sizes.

FileInfo[] fis = d.GetFiles();

foreach (FileInfo fi in fis)

{

try

{

Size += fi.Length;

}

//Данное исключение делается для пропуска папок к которым нет доступа

catch (UnauthorizedAccessException)

{

;

}

}

// Add subdirectory sizes.

DirectoryInfo[] dis = d.GetDirectories();

foreach (DirectoryInfo di in dis)

{

try

{

Size += DirSize(di);

}

//Данное исключение делается для пропуска папок к которым нет доступа

catch (UnauthorizedAccessException)

{

;

}

}

return (Size);

}

}

}

Head.cs - команда head

using System;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>Head</c> show 10 start lines for file.

/// </summary>

internal class Head : Command

{

public Head(string name) : base(name)

{

PossibleKeys.Add("-n");

PossibleKeys.Add("-v");

\_command = Start;

}

private string Start()

{

if (\_values.Count <= 0)

return OnlineMode();

var output = string.Empty;

var count = 10;

if (\_keys.Contains("-n"))

{

if (int.TryParse(\_values[\_values.Count - 1], out count) == false)

return "head: invalid number of lines: " + \_values[\_values.Count - 1];

\_values.RemoveAt(\_values.Count - 1);

}

foreach (var filename in \_values)

{

try

{

if (\_keys.Contains("-v"))

output += "\n" + "==>\t" + filename + "\t<==";

var lines = Concatenate.Read(filename);

for (int i = 0; i < (count > lines.Length ? lines.Length : count); i++)

{

output += "\n" + lines[i];

}

}

catch (Exception ex)

{

return "head: " + filename.Replace("\\", "/") + ": " + ex.Message;

}

}

return output;

}

public static string OnlineMode()

{

while (true)

{

var line = Console.ReadLine();

if (line == "^exit")

break;

}

return string.Empty;

}

}

}

Kill.cs - команда kill

using System;

using System.Diagnostics;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>Kill</c> kill process by pid.

/// </summary>

internal class Kill : Command

{

public Kill(string name) : base(name)

{

PossibleKeys.Add("-l");

PossibleKeys.Add("-v");

\_command += Start;

}

private string Start()

{

string output = string.Empty;

try

{

if (\_keys.Contains("-l"))

return ListSignal();

if (\_values.Count <= 0)

throw new Exception("too many arguments");

foreach (var pid in \_values)

{

if (int.TryParse(pid, out var id) == false)

throw new Exception(pid + ": syntax error");

var p = Process.GetProcessById(id);

p.Kill();

if (\_keys.Contains("-v"))

output += "\n " + p.ProcessName + ": killed";

}

}

catch(Exception ex)

{

output += "kill: " + ex.Message;

}

return output;

}

private string ListSignal()

{

return " 1) SIGHUP 2) SIGINT 3) SIGQUIT 4) SIGILL 5) SIGTRAP\n" +

" 6) SIGABRT 7) SIGEMT 8) SIGFPE 9) SIGKILL 10) SIGBUS\n" +

"11) SIGSEGV 12) SIGSYS 13) SIGPIPE 14) SIGALRM 15) SIGTERM\n" +

"16) SIGURG 17) SIGSTOP 18) SIGTSTP 19) SIGCONT 20) SIGCHLD\n" +

"21) SIGTTIN 22) SIGTTOU 23) SIGIO 24) SIGXCPU 25) SIGXFSZ\n" +

"26) SIGVTALRM 27) SIGPROF 28) SIGWINCH 29) SIGPWR 30) SIGUSR1\n" +

"31) SIGUSR2 32) SIGRTMIN 33) SIGRTMIN+1 34) SIGRTMIN+2 35) SIGRTMIN+3\n" +

"36) SIGRTMIN+4 37) SIGRTMIN+5 38) SIGRTMIN+6 39) SIGRTMIN+7 40) SIGRTMIN+8\n" +

"41) SIGRTMIN+9 42) SIGRTMIN+10 43) SIGRTMIN+11 44) SIGRTMIN+12 45) SIGRTMIN+13\n" +

"46) SIGRTMIN+14 47) SIGRTMIN+15 48) SIGRTMIN+16 49) SIGRTMAX-15 50) SIGRTMAX-14\n" +

"51) SIGRTMAX-13 52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9\n" +

"56) SIGRTMAX-8 57) SIGRTMAX-7 58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4\n" +

"61) SIGRTMAX-3 62) SIGRTMAX-2 63) SIGRTMAX-1 64) SIGRTMAX ";

}

}

}

ListFiles.cs - команда ls

using System.Collections.Generic;

using System.IO;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>ListFiles</c> show entry files in directory.

/// </summary>

internal class ListFiles : Command

{

public ListFiles(string name) : base(name)

{

PossibleKeys.Add("-a");

PossibleKeys.Add("-r");

PossibleKeys.Add("-A");

\_command = Start;

}

private string Start()

{

string output = string.Empty;

if (\_values.Count <= 0)

{

output = List(Directory.GetCurrentDirectory());

}

else

{

foreach (var value in \_values)

{

output += List(value);

}

}

return output;

}

private string List(string path)

{

string current = Path.GetFullPath(path).Replace("\\", "/");

var names = current.Split(new string[] {"/"}, System.StringSplitOptions.RemoveEmptyEntries);

string output = names[names.Length-1] + ": \n";

var list = new List<string>();

try

{

if (Path.IsPathRooted(path))

list.AddRange(Directory.GetFileSystemEntries(path));

else

list.AddRange(Directory.GetFileSystemEntries(Directory.GetCurrentDirectory()+ "\\" + path));

}

catch

{

return "cd:\t" + \_values[0] + ": No such file or directory";

}

for (int i = 0; i < list.Count; i++)

{

if (Directory.Exists(list[i]))

list[i] += "\\";

list[i] = list[i].Replace("\\", "/");

list[i] = list[i].Replace(current, string.Empty);

if (list[i].ToCharArray()[0] == '/')

list[i] = list[i].Substring(1);

}

if (\_keys.Contains("-a") == false && \_keys.Contains("-A") == false)

for (int i = 0; i<list.Count;i++)

if (list[i].ToCharArray()[0] == '.')

list.RemoveAt(i);

if (\_keys.Contains("-a") == true && \_keys.Contains("-A") == false ||

\_keys.Contains("-a") == true && \_keys.Contains("-A") == true &&

\_keys.IndexOf("-a") > \_keys.IndexOf("-A"))

{

list.Insert(0, "../");

list.Insert(0, "./");

}

if (\_keys.Contains("-r"))

list.Reverse();

foreach (var file in list)

{

output += "\n\t" + file;

}

return output + "\n";

}

}

}

MakeDirectory.cs - команда mkdir

using System.IO;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>MakeDirectory</c> create new directory.

/// </summary>

internal class MakeDirectory : Command

{

public MakeDirectory(string name) : base(name)

{

PossibleKeys.Add("-p");

PossibleKeys.Add("-v");

\_command = Start;

}

private string Start()

{

string output = string.Empty;

if (\_values.Count <= 0)

return "mkdir: missing operand";

string current = string.Empty;

foreach (var value in \_values)

{

if (Path.IsPathRooted(value) == false)

current = Directory.GetCurrentDirectory() + "\\";

if (\_keys.Contains("-p") == false &&

Directory.Exists(Directory.GetParent(current + value).FullName) == false)

return "mkdir: cannot create directory " + value.Replace("\\", "/") + ": No such file or directory";

Directory.CreateDirectory(current + value);

if (\_keys.Contains("-v"))

output += "\nmkdir: created directory " + value.Replace("\\", "/");

}

return output;

}

}

}

Manual.cs - команда man

using System;

using System.IO;

using System.Linq;

using System.Reflection;

using System.Threading;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>Manual</c> viewer manual documentation for commands.

/// </summary>

internal class Manual : Command

{

public Manual(string name) : base(name)

{

\_command = Start;

}

private string Start()

{

var bufer = Console.BufferHeight;

try

{

if (\_values.Count <= 0)

return "What manual page do you wand?\nFor example, try 'man man'.";

var file = Directory.GetParent(Assembly.GetExecutingAssembly().Location) + "\\manual\\" + \_values[0] + ".man";

if (File.Exists(file) == false)

return "No manual entry " + \_values[0];

var lines = File.ReadAllLines(file);

bool run = true;

int currentLine = 0;

Console.BufferHeight = Console.WindowHeight;

Console.CursorVisible = false;

Console.Clear();

Render(lines, currentLine, Console.WindowHeight, \_values[0]);

while (run)

{

var key = Console.ReadKey(true).Key;

switch (key)

{

case ConsoleKey.Q: run = false;

continue;

case ConsoleKey.J:

case ConsoleKey.UpArrow: currentLine -= currentLine > 0 ? 1 : 0;

break;

case ConsoleKey.K:

case ConsoleKey.DownArrow: currentLine += currentLine < lines.Length - Console.WindowHeight ? 1 : 0;

break;

default:

break;

}

Render(lines, currentLine, Console.WindowHeight, \_values[0]);

Thread.Sleep(20);

}

}

catch (Exception ex)

{

return "man: " + ex.Message;

}

Console.Clear();

Console.ResetColor();

Console.CursorVisible = true;

Console.BufferHeight = bufer;

return string.Empty;

}

private void Render(string[] lines, int start, int maxLengh, string name)

{

var output = string.Empty;

string spase = String.Concat(Enumerable.Repeat(" ", Console.WindowWidth));

spase = String.Concat(Enumerable.Repeat(spase + "\n", Console.WindowHeight));

Console.Write(spase);

Console.SetCursorPosition(0, 0);

for (int i = start; i < start + maxLengh-1; i++)

{

if (i > lines.Length-1)

{

output += "\n";

continue;

}

output += $"{i + 1, 4}|{lines[i]} \n";

}

Console.ForegroundColor = ConsoleColor.Cyan;

Console.Write(output);

Console.BackgroundColor = ConsoleColor.Cyan;

Console.ForegroundColor = ConsoleColor.Black;

Console.Write($" Manual page {name} line {start + 1} (press q for quit)");

Console.BackgroundColor = ConsoleColor.Black;

Console.SetCursorPosition(0, 0);

}

}

}

PrintWorkingDirectory.cs - команда pwd

using System.IO;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>PrintWorkingDirectory</c> show current directory.

/// </summary>

internal class PrintWorkingDirectory : Command

{

public PrintWorkingDirectory(string name) : base(name)

{

PossibleKeys.Add("-L");

PossibleKeys.Add("-P");

\_command = Start;

}

private string Start()

{

string output = string.Empty;

output += Directory.GetCurrentDirectory();

return output;

}

}

}

Processes.cs - команда ps

using System;

using System.Collections.Generic;

using System.Diagnostics;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>Processes</c> show procesess by pid.

/// </summary>

internal class Processes : Command

{

public Processes(string name) : base(name)

{

PossibleKeys.Add("-A");

PossibleKeys.Add("-e");

PossibleKeys.Add("-p");

\_command = Start;

}

private string Start()

{

string output = string.Empty;

try

{

Process[] ps;

if (\_keys.Contains("-p") == false && \_values.Count > 0)

throw new Exception("missing operands");

if (\_keys.Count > 0)

ps = Process.GetProcesses();

else

ps = new Process[] { Process.GetCurrentProcess() };

if (\_keys.Contains("-p"))

ps = SearchProcesses(ps, \_values.ToArray());

var view = ViewProcesses(ps);

foreach (var p in view)

{

output += "\n" + p;

}

}

catch (Exception ex)

{

output += "ps: " + ex.Message;

}

return output;

}

private Process[] SearchProcesses(Process[] ps, string[] values)

{

var listProcesses = new List<Process>();

foreach (var pid in values)

{

if (int.TryParse(pid, out var id) == false)

throw new Exception("process id syntax error");

foreach (var p in ps)

{

if (p.Id == id)

listProcesses.Add(p);

}

}

return listProcesses.ToArray();

}

private string[] ViewProcesses(Process[] ps)

{

var output = new List<string>();

foreach (var p in ps)

try

{

output.Add($"{p.Id, 6} {"?", 6} {p.TotalProcessorTime.ToString(@"hh\:mm\:ss"), 9} {p.ProcessName}");

}

catch

{

continue;

}

output.Sort();

output.Insert(0, $"{"PID",6} {"TTY",6} {"TIME",9} {"CMD"}");

return output.ToArray();

}

}

}

Remove.cs - команда rm

using System;

using System.IO;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>Remove</c> remove file or directory.

/// </summary>

internal class Remove : Command

{

public Remove(string name) : base(name)

{

PossibleKeys.Add("-f");

PossibleKeys.Add("-r");

PossibleKeys.Add("-v");

\_command = Start;

}

private string Start()

{

string output = string.Empty;

if (\_values.Count <= 0)

return "rm: missing operand";

foreach (var file in \_values)

{

string current = Directory.GetCurrentDirectory() + "\\";

try

{

if (Path.IsPathRooted(\_values[0]))

current = string.Empty;

if (Directory.Exists(current + file))

{

if (\_keys.Contains("-r"))

Directory.Delete(current + file, true);

else Directory.Delete(current + file);

}

else if (File.Exists(current + file))

{

File.Delete(current + file);

}

else throw new Exception("No such file or directory");

if (\_keys.Contains("-v"))

output += "\nremoved " + file;

}

catch (Exception ex)

{

if (\_keys.Contains("-f"))

continue;

output += "\nrm: cannot remove " + file + ": " + ex.Message;

break;

}

}

return output;

}

}

}

RemoveDirectory.cs - команда rmdir

using System.IO;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>RemoveDirectory</c> remove directory.

/// </summary>

internal class RemoveDirectory : Command

{

public RemoveDirectory(string name) : base(name)

{

PossibleKeys.Add("-p");

PossibleKeys.Add("-v");

\_command = Start;

}

private string Start()

{

string output = string.Empty;

if (\_values.Count <= 0)

return "rmdir: missing operand";

string current = string.Empty;

foreach (var value in \_values)

{

if (Path.IsPathRooted(value) == false)

current = Directory.GetCurrentDirectory() + "\\";

if (Directory.Exists(current + value) == false)

return "rmdir: failed to remove " + value.Replace("\\", "/") + ": No such file or directory ";

try

{

Directory.Delete(current + value);

if (\_keys.Contains("-v"))

output += "\nrmdir: removing directory " + value.Replace("\\", "/");

if (\_keys.Contains("-p"))

{

var dir = Directory.GetParent(current + value).FullName;

while (dir + "\\" != current)

{

Directory.Delete(dir);

if (\_keys.Contains("-v"))

output += "\nrmdir: removing directory " + dir.Replace(current, string.Empty).Replace("\\", "/");

dir = Directory.GetParent(dir).FullName;

}

}

}

catch

{

output += "\nrmdir: failed to remove " + value.Replace("\\", "/") + ": Directory not empty";

}

}

return output;

}

}

}

Tail.cs - команда tail

using System;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>Tail</c> show last 10 lines for file.

/// </summary>

internal class Tail : Command

{

public Tail(string name) : base(name)

{

PossibleKeys.Add("-n");

PossibleKeys.Add("-v");

\_command = Start;

}

private string Start()

{

if (\_values.Count <= 0)

return OnlineMode();

var output = string.Empty;

var count = 10;

if (\_keys.Contains("-n"))

{

if (int.TryParse(\_values[\_values.Count - 1], out count) == false)

return "head: invalid number of lines: " + \_values[\_values.Count - 1];

\_values.RemoveAt(\_values.Count - 1);

}

foreach (var filename in \_values)

{

try

{

if (\_keys.Contains("-v"))

output += "\n" + "==>\t" + filename + "\t<==";

var lines = Concatenate.Read(filename);

for (int i = lines.Length - 1; i >= (lines.Length - count >= 0 ? lines.Length - count : 0); i--)

{

output += "\n" + lines[i];

}

}

catch (Exception ex)

{

return "tail: " + filename.Replace("\\", "/") + ": " + ex.Message;

}

}

return output;

}

public static string OnlineMode()

{

while (true)

{

var line = Console.ReadLine();

if (line == "^exit")

break;

}

return string.Empty;

}

}

}

Touch.cs - команда touch

using System;

using System.IO;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>Touch</c> create file or update his time used.

/// </summary>

internal class Touch : Command

{

public Touch(string name) : base(name)

{

\_command = Start;

}

private string Start()

{

string output = string.Empty;

if (\_values.Count <= 0)

return "touch: missing operand";

foreach (var file in \_values)

{

string current = Directory.GetCurrentDirectory() + "\\";

try

{

if (Path.IsPathRooted(\_values[0]))

current = string.Empty;

var f = File.Open(current + file, FileMode.OpenOrCreate);

f.Close();

File.SetLastWriteTime(current + file, DateTime.Now);

}

catch (Exception ex)

{

if (\_keys.Contains("-f"))

continue;

output += "\nrm: cannot remove " + file + ": " + ex.Message;

break;

}

}

return output;

}

}

}

Uname.cs - команда uname

using System;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>Uname</c> show information from pc.

/// </summary>

internal class Uname : Command

{

public Uname(string name) : base(name)

{

PossibleKeys.Add("-p");

PossibleKeys.Add("-s");

PossibleKeys.Add("-n");

\_command = Start;

}

private string Start()

{

string output = string.Empty;

if (\_keys.Count <= 0)

\_keys.Add("-s");

if (\_keys.Contains("-p"))

output += "\t"+(Environment.Is64BitOperatingSystem ? "x64" : "x32");

if (\_keys.Contains("-n"))

output += "\t" + Environment.UserDomainName;

if (\_keys.Contains("-s"))

output += "\t" + Environment.OSVersion.VersionString;

return output;

}

}

}

WordCount.cs - команда wc

using System;

using System.IO;

namespace Terminal.Commands

{

/// <summary>

/// Class <c>WordCount</c> show count word, bytes or char in file.

/// </summary>

internal class WordCount : Command

{

public WordCount(string name) : base(name)

{

PossibleKeys.Add("-m");

PossibleKeys.Add("-c");

PossibleKeys.Add("-l");

\_command += Start;

}

private string Start()

{

string output = string.Empty;

if (\_keys.Count <= 0)

{

\_keys.Add("-m");

\_keys.Add("-c");

\_keys.Add("-l");

}

int totalLines = 0;

int totalBytes = 0;

int totalChars = 0;

foreach (var file in \_values)

{

try

{

string current = string.Empty;

if (Path.IsPathRooted(file) == false)

current = Directory.GetCurrentDirectory() + "\\";

if (File.Exists(current + file) == false)

throw new ArgumentException("No such file or directory ");

if (\_keys.Contains("-l"))

{

var lines = File.ReadAllLines(current + file).Length;

output += $"{lines, 6} ";

totalLines += lines;

}

if (\_keys.Contains("-c"))

{

var bytes = File.ReadAllBytes(current + file).Length;

output += $"{bytes,6} ";

totalBytes += bytes;

}

if (\_keys.Contains("-m"))

{

var chars = File.ReadAllText(current + file).Length;

output += $"{chars,6} ";

totalChars += chars;

}

output += file + "\n";

}

catch (Exception ex)

{

output += "cat: " + file.Replace("\\", "/") + ": " +ex.Message + "\n";

continue;

}

}

if (\_values.Count > 1)

output += $"{totalLines,6} {totalBytes,6} {totalChars,6} Total";

return output;

}

}

}