МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

# УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

# «ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

# ИМЕНИ П.М. МАШЕРОВА»

Факультет математики и информационных технологий

Кафедра информатики и информационных технологий

КУРСОВОЙ ПРОЕКТ

по дисциплине «Дополнительные главы информатики»

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ПОИСКА ИНФОРМАЦИИ В ПОЛЬЗОВАТЕЛЬСКИХ ФАЙЛАХ

Колебянов Виталий Романович,

студент 4 курса, группы 41з

Руководитель:

Кухарев А.В

Витебск, 2019

**СОДЕРЖАНИЕ**

|  |
| --- |
| ВВЕДЕНИЕ……………………………………………………………………..3  1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ………………………………………………..4  1.1. Назначение и область применения программного продукта…………...4 |
| 1.2. Требования к программе и программному продукту…………………...4  1.2.1. Требования к конфигурации электронно-вычислительных средств…4  1.2.2. Требования к информационной и программной совместимости…….4  1.2.3. Требования к языку программирования……………………………….5 |
| 2. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА…………………………6  2.1. Постановка задачи…………………………………………………………6  2.1.1. Источники входной информации………………………………………6  2.1.2. Выходная информация…………………………………………………..6  2.1.3. Создание программных модулей……………………………………….6  2.1.4. Описание алгоритма……………………………………………………..7  ЗАКЛЮЧЕНИЕ………………………………………………………………...8  СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ……………………..……9  ПРИЛОЖЕНИЕ А. Исходный текст программы …………………………..10  ПРИЛОЖЕНИЕ В. Демонстрационный пример........................……………47 |

ВВЕДЕНИЕ

В настоящий момент времени пользователи могут иметь у себя многочисленные библиотеки файлов, по которым необходимо производить поиск. Зачастую это трудно сделать, т.к. файлы могут иметь различный формат хранения данных (в том числе бинарный), и «обычные» файловые менеджеры не смогут искать информацию в таких файлах эффективно. Как раз для решения такой задачи разрабатывается данное веб-приложение.

Целью курсового проекта является разработка веб-приложения для поиска в пользовательских файлах, в котором пользователи смогут загружать свой файлы разных форматов, производить по ним поиск и скачивать их в изначальном варианте при необходимости.

Задачи курсового проекта:

1. Разработка веб-приложения;
2. Использование современных средств развертывания веб-приложений (Docker)
3. ТЕХНИЧЕСКОЕ ЗАДАНИЕ
   1. Назначение и область применения программного продукта

Данный программный продукт представляет собой веб-приложение.

Основным назначением программы является поиск в пользовательских файлах.

Программный продукт должен выполнять следующие функции:

* позволять пользователям загружать свои файлы;
* производить поиск по файлам;
* просматривать список всех загруженных файлов;
* скачивать файлы в изначальном виде при необходимости.
  1. Требования к конфигурации электронно-вычислительных средств
     1. Требования к конфигурации электронно-вычислительных средств пользователя веб-приложения

Для эксплуатации программы необходим персональный компьютер с установленным веб-броузером.

Список поддерживаемых веб-броузеров:

* Google Chrome (версия 23 и выше)
* Mozilla Firefox (версия 21 и выше)
* Microsoft Edge (версия 10 и выше)
* Safari (версия 6 и выше)
* Internet Explorer 11
  + 1. Требования к конфигурации электронно-вычислительных средств, где будет развернуто веб-приложение

OC: Windows, Linux, Mac OS

Предустановленное ПО: Docker (Docker for Windows для OC Windows)

* 1. Описание используемых программных средств и технологий
     1. Язык программирования C#

Данный программный продукт должен быть написан на языке программирования высокого уровня С# в инструментальной среде разработки приложений – Microsoft Visual Studio.

Выбор в качестве системы программирования языка программирования C# обусловлен тем, что С# – это универсальный объектно-ориентированный язык программирования со статической типизацией. Он является безопасным в отношении работы с памятью, а также имеет сборщик мусора.

* + 1. IDE Microsoft Visual Studio

Особенностями Microsoft Visual Studio являются:

* интегрированная среда разработки приложений (IDE) – служит для организации взаимодействия с программистом и включает в себя ряд окон, содержащих различные управляющие элементы. С помощью средств интегрированной среды разработчик может удобно проектировать интерфейсную часть приложения, а также писать программный код и связывать его с управляющими элементами;
* визуальная технология разработки программ – позволяет быстро создавать приложения путем размещения в форме стандартных компонентов. При этом соответствующий код программы автоматически генерируется Microsoft Visual Studio. Такая технология освобождает разработчика от рутинной работы по созданию пользовательского интерфейса и позволяет уделить больше внимания внутренней организации программы и обработке данных;
* компилятор может генерировать как 32-битные, так и 64-битные исполняемый файлы и библиотеки.
  + 1. Docker

Docker является современным средством для автоматизации развертывания приложений. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесён на любую Linux-систему с поддержкой cgroups в ядре, а также предоставляет среду по управлению контейнерами. Изначально использовал возможности LXC, с 2015 года применял собственную библиотеку, абстрагирующую виртуализационные возможности ядра Linux — libcontainer. С появлением ​Open Container Initiative начался переход от монолитной к модульной архитектуре.

2. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА

2.1. Постановка задачи

2.1.1. Источники входной информации

Источниками входной информации являться файлы и папки размещенные на диске.

2.1.2. Выходная информация

Выходной информацией являются:

* созданные и скопированные файлы и папки;
* отображаемое содержимое папок;
* диалоговые окна;
* сообщения об ошибках и успешных действиях.

2.1.3. Создание программных модулей

Для данной программы модулями программы будут являться файлы с расширением \*. сs .

Модули программы:

Startup – стартовый модуль, с которого начинается выполнение программы.

MainWindow – модуль главного окна программы.

KeyboardHelpButton – модуль кнопки, находящейся внизу программы.

Localizator – модуль для хранения локализируемых данных.

FileListPanel – модуль панели, отображающей содержимое папок.

ЗАКЛЮЧЕНИЕ

В ходе написания курсового проекта использовал среду разработки Microsoft Visual Studio.

«Файловый менеджер» соответствует исходному техническому заданию. На основе выполненных тестов можно сделать вывод, что программа отвечает эксплуатационным, технологическим требованиям.

Разработанный программный продукт с включенными в него функциональными возможностями позволяет ускорить и улучшить автоматизацию процесса работы с текстовыми файлами.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Объектно-ориентированный анализ и проектирование с примерами пользователя/ Пер. с англ. Г.Г. Буг. – М. : ДМК, 2000. – 224 с.: ил.
2. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. – Введ. 81-07-01. – М. : Изд-во стандартов, 2001.
3. ГОСТ 19.701-80. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – Введ 81-07-01 – М. : Изд-во стандартов, 2001.
4. Зелковец М.В. Принципы разработки программного обеспечения/ М.В. Земковец. – Москва : Мир, 1982. – 400 с.: ил.
5. . Кинг Д.А. Создание эффективного программного обеспечения/ Д.А. Кинг. – М. : Мир, 1991. – 412 с.: ил.
6. Константайн Л.О. Разработка программного обеспечения/ Л.О. Константайн, Л. Локвуд. – СПб. : Питер, 2004. – 592 с.: ил.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ ТЕКСТ ПРОГРАММЫ

Startup.cs:

using System;

using Xwt;

using System.Reflection;

using pluginner.Toolkit;

namespace fcmd

{

static class Startup

{

[STAThread]

static void Main(string[] Commands)

{

string product\_version = Assembly.GetExecutingAssembly().GetName().Version.ToString();

#if DEBUG

try

{

switch (OSVersionEx.Platform)

{

case PlatformID.Win32NT:

Application.Initialize(ToolkitType.Wpf);

break;

case PlatformID.MacOSX:

Application.Initialize(ToolkitType.Cocoa);

break;

default:

Application.Initialize(ToolkitType.Gtk3);

break;

}

}

catch (Exception ex)

{

string errmsg = "The XWT could not be loaded:\n" + ex.InnerException.Message;

if(ex.InnerException.InnerException != null) {

errmsg+= "\n"+ex.InnerException.InnerException.Message;

}

Xwt.MessageDialog.ShowError(

errmsg + Environment.NewLine +

"The File Commander " + product\_version + " (" + (Environment.Is64BitProcess ? "x64" : "x86") + "-DEBUG) Startup Failure"

);

return;

}

#else

try

{

var toolkitType = OSVersionEx.GetToolkitType();

if (toolkitType == ToolkitType.Gtk) {

toolkitType = ToolkitType.Gtk3;

}

Application.Initialize(toolkitType);

}

catch (Exception ex) {

Xwt.MessageDialog.ShowError(

"The XWT could not be loaded:\n" + ex.InnerException.Message + Environment.NewLine +

"The File Commander " + product\_version + " (" + (Environment.Is64BitProcess ? "x64" : "x86") + ") Startup Failure"

);

return;

}

#endif

#if !DEBUG

try {

#endif

new MainWindow(Commands).Show();

Application.Run();

#if !DEBUG

}

catch (Exception ex)

{

string msg = "The File Commander has been crashed:\n" + ex.Message + "\n" + ex.StackTrace;

string inex = "";

if(ex.InnerException != null) inex = "\n Inner exception" + ex.InnerException.Message + "\n" + ex.StackTrace;

msg+= inex;

Xwt.MessageDialog.ShowError(

msg + Environment.NewLine +

"The File Commander " + product\_version + " (" + (Environment.Is64BitProcess ? "x64" : "x86") + ") Crash"

);

return;

}

#endif

}

}

}

MainWindow.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using System.Text;

using System.IO;

using pluginner.Toolkit;

using pluginner.Widgets;

namespace fcmd

{

partial class MainWindow : Xwt.Window

{

static string ProductVersion {

get { return Assembly.GetExecutingAssembly().GetName().Version.ToString(); }

}

Stylist stylist;

Xwt.Menu WindowMenu = new Xwt.Menu();

Xwt.MenuItem mnuFile = new Xwt.MenuItem { Tag = "mnuFile" };

Xwt.MenuItem mnuFileExit = new Xwt.MenuItem { Tag = "mnuFileExit" };

Xwt.MenuItem mnuHelp = new Xwt.MenuItem { Tag = "mnuHelp" };

Xwt.MenuItem mnuHelpHelpMe = new Xwt.MenuItem { Tag = "mnuHelpHelpMe" };

Xwt.MenuItem mnuHelpAbout = new Xwt.MenuItem { Tag = "mnuHelpAbout" };

Xwt.VBox Layout = new Xwt.VBox();

Xwt.HPaned PanelLayout = new Xwt.HPaned();

FileListPanel p1;

FileListPanel p2;

List<ListView2.ColumnInfo> LVCols = new List<ListView2.ColumnInfo>();

FileListPanel ActivePanel;

FileListPanel PassivePanel;

Xwt.HBox KeyBoardHelp = new Xwt.HBox();

KeyboardHelpButton[] KeybHelpButtons = new KeyboardHelpButton[6];

public MainWindow(string[] argv)

{

this.Title = "File Commander";

this.MainMenu = WindowMenu;

this.PaddingLeft = PaddingRight = PaddingTop = 0;

PaddingBottom = PaddingBottom/3;

MainMenu.Items.Add(mnuFile);

//MainMenu.Items.Add(mnuHelp);

mnuFile.SubMenu = new Xwt.Menu();

mnuFile.SubMenu.Items.Add(mnuFileExit);

mnuHelp.SubMenu = new Xwt.Menu();

mnuHelp.SubMenu.Items.Add(mnuHelpHelpMe);

mnuHelp.SubMenu.Items.Add(mnuHelpAbout);

TranslateMenu(MainMenu);

this.CloseRequested += MainWindow\_CloseRequested;

PanelLayout.KeyReleased += PanelLayout\_KeyReleased;

mnuFileExit.Clicked += (o, ea) => { this.Close(); };

mnuHelpAbout.Clicked += mnuHelpAbout\_Clicked;

Layout.PackStart(PanelLayout, true, Xwt.WidgetPlacement.Fill, Xwt.WidgetPlacement.Fill, 0, 0, 0, 0);

Layout.PackStart(KeyBoardHelp, false, Xwt.WidgetPlacement.End, Xwt.WidgetPlacement.Fill, 1, 3, 1, 2);

this.Content = Layout;

if (fcmd.Properties.Settings.Default.UserTheme != null)

{

if (fcmd.Properties.Settings.Default.UserTheme != "")

{

if (File.Exists(fcmd.Properties.Settings.Default.UserTheme))

stylist = new Stylist(fcmd.Properties.Settings.Default.UserTheme);

else

{

Xwt.MessageDialog.ShowError(Localizator.GetString("ThemeNotFound"), fcmd.Properties.Settings.Default.UserTheme);

Xwt.Application.Exit();

}

}

}

string BookmarksStore = null;

if (fcmd.Properties.Settings.Default.BookmarksFile != null && fcmd.Properties.Settings.Default.BookmarksFile.Length > 0)

{

BookmarksStore = File.ReadAllText(fcmd.Properties.Settings.Default.BookmarksFile, Encoding.UTF8);

}

PanelLayout.Panel1.Content = new FileListPanel(BookmarksStore, fcmd.Properties.Settings.Default.UserTheme, Properties.Settings.Default.InfoBarContent1, Properties.Settings.Default.InfoBarContent2); //Левая, правая где сторона? Улица, улица, ты, брат, пьяна!

PanelLayout.Panel2.Content = new FileListPanel(BookmarksStore, fcmd.Properties.Settings.Default.UserTheme, Properties.Settings.Default.InfoBarContent1, Properties.Settings.Default.InfoBarContent2);

p1 = PanelLayout.Panel1.Content as FileListPanel;

p2 = PanelLayout.Panel2.Content as FileListPanel;

var openFileHandler = new pluginner.TypedEvent<string> (Panel\_OpenFile);

p1.OpenFile += openFileHandler;

p2.OpenFile += openFileHandler;

p1.FS = new base\_plugins.fs.localFileSystem();

p2.FS = new base\_plugins.fs.localFileSystem();

p1.GotFocus += (o, ea) => SwitchPanel(p1);

p2.GotFocus += (o, ea) => SwitchPanel(p2);

for (int i = 1; i < KeybHelpButtons.Length; i++)

{

KeybHelpButtons[i] = new KeyboardHelpButton { CanGetFocus = false };

KeyBoardHelp.PackStart(KeybHelpButtons[i], true, Xwt.WidgetPlacement.Fill, Xwt.WidgetPlacement.Fill, 0, -6, 0, -3);

}

KeybHelpButtons[1].Clicked += (o, ea) => { this.PanelLayout\_KeyReleased(this, new Xwt.KeyEventArgs(Xwt.Key.F5, Xwt.ModifierKeys.None, false, 0)); };

KeybHelpButtons[1].Tag = 5;

KeybHelpButtons[2].Clicked += (o, ea) => { this.PanelLayout\_KeyReleased(this, new Xwt.KeyEventArgs(Xwt.Key.F6, Xwt.ModifierKeys.None, false, 0)); };

KeybHelpButtons[2].Tag = 6;

KeybHelpButtons[3].Clicked += (o, ea) => { this.PanelLayout\_KeyReleased(this, new Xwt.KeyEventArgs(Xwt.Key.F7, Xwt.ModifierKeys.None, false, 0)); };

KeybHelpButtons[3].Tag = 7;

KeybHelpButtons[4].Clicked += (o, ea) => { this.PanelLayout\_KeyReleased(this, new Xwt.KeyEventArgs(Xwt.Key.F8, Xwt.ModifierKeys.None, false, 0)); };

KeybHelpButtons[4].Tag = 8;

KeybHelpButtons[5].Clicked += (o, ea) => { this.PanelLayout\_KeyReleased(this, new Xwt.KeyEventArgs(Xwt.Key.F10, Xwt.ModifierKeys.None, false, 0)); };

KeybHelpButtons[5].Tag = 10;

Localizator.LocalizationChanged += (o, ea) => Localize();

Localize();

this.Width = fcmd.Properties.Settings.Default.WinWidth;

this.Height = fcmd.Properties.Settings.Default.WinHeight;

char[] Policies = fcmd.Properties.Settings.Default.SizeShorteningPolicy.ToCharArray();

if (Properties.Settings.Default.Panel1URL.Length != 0) {

p1.LoadDir (Properties.Settings.Default.Panel1URL, ConvertSDP (Policies [0]), ConvertSDP (Policies [1]), ConvertSDP (Policies [2]));

} else {

p1.LoadDir ("file://" + System.IO.Directory.GetCurrentDirectory (), ConvertSDP (Policies [0]), ConvertSDP (Policies [1]), ConvertSDP (Policies [2]));

}

if (Properties.Settings.Default.Panel2URL.Length != 0) {

p2.LoadDir (Properties.Settings.Default.Panel2URL, ConvertSDP (Policies [0]), ConvertSDP (Policies [1]), ConvertSDP (Policies [2]));

} else {

p2.LoadDir ("file://" + System.IO.Directory.GetCurrentDirectory (), ConvertSDP (Policies [0]), ConvertSDP (Policies [1]), ConvertSDP (Policies [2]));

}

switch (fcmd.Properties.Settings.Default.LastActivePanel)

{

case 1:

p1.ListingView.SetFocus();

ActivePanel = p1; PassivePanel = p2;

if(argv.Length == 1) p1.LoadDir(argv[0]);

break;

case 2:

p2.ListingView.SetFocus();

ActivePanel = p2; PassivePanel = p1;

if (argv.Length == 1) p2.LoadDir(argv[0]);

break;

default:

p1.ListingView.SetFocus();

ActivePanel = p1; PassivePanel = p2;

if (argv.Length == 1) p1.LoadDir(argv[0]);

break;

}

#if DEBUG

Console.WriteLine(@"DEBUG: MainWindow initialization has been completed.");

#endif

}

private void Localize()

{

TranslateMenu(this.MainMenu);

for (int i = 1; i < KeybHelpButtons.Length; i++)

{

int num = (int)KeybHelpButtons[i].Tag;

KeybHelpButtons[i].FKey = "F" + num;

KeybHelpButtons[i].Text = Localizator.GetString("FCF" + num);

}

LVCols.Clear();

LVCols.Add(new ListView2.ColumnInfo { Title = "", Tag = "Icon", Width = 16, Visible = true });

LVCols.Add(new ListView2.ColumnInfo { Title = "URL", Tag = "Path", Width = 0, Visible = false });

LVCols.Add(new ListView2.ColumnInfo { Title = Localizator.GetString("FName"), Tag = "FName", Width = 100, Visible = true });

LVCols.Add(new ListView2.ColumnInfo { Title = Localizator.GetString("FSize"), Tag = "FSize", Width = 50, Visible = true });

LVCols.Add(new ListView2.ColumnInfo { Title = Localizator.GetString("FDate"), Tag = "FDate", Width = 50, Visible = true });

LVCols.Add(new ListView2.ColumnInfo { Title = "Directory item info", Tag = "DirItem", Width = 0, Visible = false });

p1.ListingView.SetColumns(LVCols);

p2.ListingView.SetColumns(LVCols);

}

private void mnuViewWithFilter\_Clicked(object sender, EventArgs e)

{

string Filter = @"\*.\*";

InputBox ibx = new InputBox(Localizator.GetString("NameFilterQuestion"), Filter);

Xwt.CheckBox chkRegExp = new Xwt.CheckBox(Localizator.GetString("NameFilterUseRegExp"));

ibx.OtherWidgets.Add(chkRegExp, 0, 0);

if (!ibx.ShowDialog()) return;

Filter = ibx.Result;

if (chkRegExp.State == Xwt.CheckBoxState.Off)

{

Filter = Filter.Replace(".", @"\.");

Filter = Filter.Replace("\*", ".\*");

Filter = Filter.Replace("?", ".");

}

try

{

System.Text.RegularExpressions.Regex re = new System.Text.RegularExpressions.Regex(Filter);

List<pluginner.DirItem> GoodItems = new List<pluginner.DirItem>();

foreach (pluginner.DirItem di in ActivePanel.FS.DirectoryContent)

{

if (re.IsMatch(di.TextToShow))

GoodItems.Add(di);

}

ActivePanel.LoadDir(

ActivePanel.FS.CurrentDirectory,

ActivePanel.CurShortenKB,

ActivePanel.CurShortenMB,

ActivePanel.CurShortenGB

);

ActivePanel.StatusBar.Text = string.Format(Localizator.GetString("NameFilterFound"), Filter, GoodItems.Count);

}

catch (Exception ex)

{

Xwt.MessageDialog.ShowError(Localizator.GetString("NameFilterError"), ex.Message);

}

}

private void mnuNavigateReload\_Clicked(object sender, EventArgs e)

{

ActivePanel.LoadDir();

}

private void Panel\_OpenFile(string data)

{

if (data.StartsWith("file://") && System.IO.File.Exists(data.Replace("file://", "")))

{

try

{

System.Diagnostics.Process proc = new System.Diagnostics.Process();

proc.StartInfo.FileName = data.Replace("file://", "");

proc.StartInfo.UseShellExecute = true;

proc.Start();

}

catch (Exception ex)

{

Xwt.MessageDialog.ShowMessage(ex.Message);

}

}

}

private void ShowDebugInfo (object sender, EventArgs e)

{

System.Configuration.Configuration confLR = System.Configuration.ConfigurationManager.OpenExeConfiguration(System.Configuration.ConfigurationUserLevel.PerUserRoamingAndLocal);

System.Configuration.Configuration confR = System.Configuration.ConfigurationManager.OpenExeConfiguration(System.Configuration.ConfigurationUserLevel.PerUserRoaming);

System.Configuration.Configuration confEXE = System.Configuration.ConfigurationManager.OpenExeConfiguration(System.Configuration.ConfigurationUserLevel.None);

Xwt.Dialog Fcdbg = new Xwt.Dialog();

Fcdbg.Buttons.Add(Xwt.Command.Close);

Fcdbg.Buttons[0].Clicked += (o, ea) => { Fcdbg.Hide(); };

Fcdbg.Title = "FC debug output";

string txt = "" +

"===THE FILE COMMANDER, VERSION " + ProductVersion + (Environment.Is64BitProcess ? " 64-BIT" : " 32-BIT") + "===\n" +

Environment.CommandLine + " @ .NET fw " + Environment.Version + (Environment.Is64BitOperatingSystem ? " 64-bit" : " 32-bit") + " on " + Environment.MachineName + "-" + Environment.OSVersion + " (" + OSVersionEx.Platform + " v" + Environment.OSVersion.Version.Major + "." + Environment.OSVersion.Version.Minor + ")\n" +

"The current drawing toolkit is " + Xwt.Toolkit.CurrentEngine.GetSafeBackend(this) + "\n" +

"\nCONFIGuration files:\n---------\n" +

"Local: " + confLR.FilePath + " (exists? " + b2s(confLR.HasFile) + ")\n" +

"Roaming: " + confR.FilePath + " (exists? " + b2s(confR.HasFile) + ")\n" +

"Overall: " + confEXE.FilePath + " (exists? " + b2s(confEXE.HasFile) + ")\n" +

"\nPanel debug:\n---------\n" +

"The active panel is: " + ((ActivePanel == p1) ? "LEFT\n" : "RIGHT\n") +

"The passive panel is: " + ((ActivePanel == p2) ? "LEFT\n" : "RIGHT\n") +

"They are different? " + b2s(ActivePanel != PassivePanel) + " (should be yes)\n" +

"The LEFT filesystem: " + p1.FS + " at \"" + p1.FS.CurrentDirectory + "\"\n" +

"The RIGHT filesystem: " + p2.FS + " at \"" + p2.FS.CurrentDirectory + "\"\n" +

"Filesystems are same by type? " + b2s(p1.FS.GetType() == p2.FS.GetType()) + ".\n" +

"Filesystems are identically? " + b2s(p1.FS == p2.FS) + " (should be no).\n" +

"\nTheme debug:\n---------\n" +

"Using external theme? " + b2s(!string.IsNullOrEmpty(fcmd.Properties.Settings.Default.UserTheme)) + "\n" +

"Theme's cascade style sheet file: \"" + fcmd.Properties.Settings.Default.UserTheme + "\"\n\nIf you having some troubles, please report this to https://github.com/atauenis/fcmd bug tracker or http://atauenis.ru/phpBB3/viewtopic.php?f=4&t=211 topic. \nThe End.";

Xwt.RichTextView rtv = new Xwt.RichTextView();

rtv.LoadText(txt, new Xwt.Formats.PlainTextFormat());

Xwt.ScrollView sv = new Xwt.ScrollView(rtv);

Fcdbg.Content = sv;

Fcdbg.Width = 500;

Fcdbg.Run();

}

private string b2s(bool b)

{

return (b == true) ? "YES" : "NO";

}

private void mnuHelpAbout\_Clicked(object sender, EventArgs e)

{

System.Configuration.Configuration conf = System.Configuration.ConfigurationManager.OpenExeConfiguration(System.Configuration.ConfigurationUserLevel.PerUserRoamingAndLocal);

string AboutString = string.Format(

Localizator.GetString("FileCommanderVer"),

"File Commander",

ProductVersion,

"\nhttps://github.com/atauenis/fcmd",

conf.FilePath,

Environment.OSVersion,

Environment.Version + (Environment.Is64BitProcess ? " x86-64" : " x86")

);

Xwt.MessageDialog.ShowMessage(AboutString);

}

private void MainWindow\_CloseRequested(object sender, Xwt.CloseRequestedEventArgs args)

{

Properties.Settings.Default.WinHeight = this.Height;

Properties.Settings.Default.WinWidth = this.Width;

Properties.Settings.Default.Panel1URL = p1.FS.CurrentDirectory;

Properties.Settings.Default.Panel2URL = p2.FS.CurrentDirectory;

Properties.Settings.Default.LastActivePanel = (ActivePanel == p1) ? (byte)1 : (byte)2;

Properties.Settings.Default.Save();

Xwt.Application.Exit();

}

private void PanelLayout\_KeyReleased(object sender, Xwt.KeyEventArgs e)

{

#if DEBUG

FileListPanel p1 = (PanelLayout.Panel1.Content as FileListPanel);

FileListPanel p2 = (PanelLayout.Panel2.Content as FileListPanel);

Console.WriteLine("KEYBOARD DEBUG: " + e.Modifiers + "+" + e.Key + " was pressed. Panels focuses: " + (ActivePanel == p1) + " | " + (ActivePanel == p2));

#endif

if (e.Key == Xwt.Key.Return) return;

string URL1;

if (ActivePanel.ListingView.SelectedRow > -1)

{ URL1 = ActivePanel.GetValue(ActivePanel.dfURL); }

else

{ URL1 = null; }

pluginner.IFSPlugin FS1 = ActivePanel.FS;

string URL2;

if (PassivePanel.ListingView.SelectedRow > -1)

{ URL2 = PassivePanel.GetValue(PassivePanel.dfURL); }

else

{ URL2 = null; }

pluginner.IFSPlugin FS2 = PassivePanel.FS;

switch (e.Key)

{

case Xwt.Key.NumPadAdd:

string Filter = @"\*.\*";

InputBox ibx\_qs = new InputBox(Localizator.GetString("QuickSelect"), Filter);

Xwt.CheckBox chkRegExp = new Xwt.CheckBox(Localizator.GetString("NameFilterUseRegExp"));

ibx\_qs.OtherWidgets.Add(chkRegExp, 0, 0);

if (!ibx\_qs.ShowDialog()) return;

Filter = ibx\_qs.Result;

if (chkRegExp.State == Xwt.CheckBoxState.Off)

{

Filter = Filter.Replace(".", @"\.");

Filter = Filter.Replace("\*", ".\*");

Filter = Filter.Replace("?", ".");

}

try

{

System.Text.RegularExpressions.Regex re = new System.Text.RegularExpressions.Regex(Filter);

int Count = 0;

foreach (ListView2Item lvi in ActivePanel.ListingView.Items)

{

if (re.IsMatch(lvi.Data[1].ToString())) {

ActivePanel.ListingView.Select(lvi);

Count++;

}

}

ActivePanel.StatusBar.Text = string.Format(Localizator.GetString("NameFilterFound"), Filter, Count);

}

catch (Exception ex)

{

Xwt.MessageDialog.ShowError(Localizator.GetString("NameFilterError"), ex.Message);

}

return;

case Xwt.Key.NumPadSubtract:

string Filter\_qus = @"\*.\*";

InputBox ibx\_qus = new InputBox(Localizator.GetString("QuickUnselect"), Filter\_qus);

Xwt.CheckBox chkRegExp\_qus = new Xwt.CheckBox(Localizator.GetString("NameFilterUseRegExp"));

ibx\_qus.OtherWidgets.Add(chkRegExp\_qus, 0, 0);

if (!ibx\_qus.ShowDialog()) return;

Filter\_qus = ibx\_qus.Result;

if (chkRegExp\_qus.State == Xwt.CheckBoxState.Off)

{

Filter\_qus = Filter\_qus.Replace(".", @"\.");

Filter\_qus = Filter\_qus.Replace("\*", ".\*");

Filter\_qus = Filter\_qus.Replace("?", ".");

}

try

{

System.Text.RegularExpressions.Regex re = new System.Text.RegularExpressions.Regex(Filter\_qus);

int Count\_qus = 0;

foreach (ListView2Item lvi in ActivePanel.ListingView.Items)

{

if (re.IsMatch(lvi.Data[1].ToString()))

{

ActivePanel.ListingView.Unselect(lvi);

Count\_qus++;

}

}

}

catch (Exception ex)

{

Xwt.MessageDialog.ShowError(Localizator.GetString("NameFilterError"), ex.Message);

}

return;

case Xwt.Key.F3:

if (URL1 == null)

return;

if (!FS1.FileExists(URL1))

{

Xwt.MessageDialog.ShowWarning(string.Format(Localizator.GetString("FileNotFound"), ActivePanel.GetValue(ActivePanel.dfDisplayName)));

return;

}

VEd V = new VEd();

if (e.Modifiers == Xwt.ModifierKeys.None)

{ V.LoadFile(URL1, FS1, false); V.Show(); }

else if(e.Modifiers == Xwt.ModifierKeys.Shift)

{ V.LoadFile(URL1, FS1, new base\_plugins.ve.PlainText(), false); V.Show(); }

return;

case Xwt.Key.F4:

if (URL1 == null)

return;

if (!FS1.FileExists(URL1))

{

Xwt.MessageDialog.ShowWarning(string.Format(Localizator.GetString("FileNotFound"), ActivePanel.GetValue(ActivePanel.dfDisplayName)));

return;

}

VEd E = new VEd();

if (e.Modifiers == Xwt.ModifierKeys.None)

{ E.LoadFile(URL1, FS1, true); E.Show(); }

else if(e.Modifiers == Xwt.ModifierKeys.Shift)

{ E.LoadFile(URL1, FS1, new base\_plugins.ve.PlainText(), true); E.Show(); }

return;

case Xwt.Key.F5:

if (URL1 == null)

return;

Cp();

return;

case Xwt.Key.F6:

if (URL1 == null)

return;

Mv();

return;

case Xwt.Key.F7:

InputBox ibx = new InputBox(Localizator.GetString("NewDirURL"), ActivePanel.FS.CurrentDirectory + Localizator.GetString("NewDirTemplate"));

if (ibx.ShowDialog()) MkDir(ibx.Result);

return;

case Xwt.Key.F8:

if (URL1 == null)

return;

Rm();

return;

case Xwt.Key.F10:

Xwt.Application.Exit();

return;

}

#if DEBUG

Console.WriteLine("KEYBOARD DEBUG: the key wasn't handled");

#endif

e.Handled = true;

}

private void SwitchPanel(FileListPanel NewPanel)

{

if (NewPanel == ActivePanel) return;

PassivePanel = ActivePanel;

ActivePanel = NewPanel;

#if DEBUG

string PanelName = (NewPanel == p1) ? "LEFT" : "RIGHT";

Console.WriteLine("FOCUS DEBUG: The " + PanelName + " panel (" + NewPanel.FS.CurrentDirectory + ") got focus");

#endif

AssemblyName an = Assembly.GetExecutingAssembly().GetName();

this.Title = string.Format(

"{0} - {1}",

"FC",

ActivePanel.FS.CurrentDirectory

);

PassivePanel.UrlBox.BackgroundColor = Xwt.Drawing.Colors.LightBlue;

ActivePanel.UrlBox.BackgroundColor = Xwt.Drawing.Colors.DodgerBlue;

}

private FileListPanel.SizeDisplayPolicy ConvertSDP(char SizeDisplayPolicy)

{

switch (SizeDisplayPolicy.ToString())

{

case "0":

return FileListPanel.SizeDisplayPolicy.DontShorten;

case "1":

return FileListPanel.SizeDisplayPolicy.OneNumeral;

case "2":

return FileListPanel.SizeDisplayPolicy.TwoNumeral;

default:

return FileListPanel.SizeDisplayPolicy.OneNumeral;

}

}

private void TranslateMenu(Xwt.Menu mnu)

{

try

{

foreach (Xwt.MenuItem currentMenuItem in mnu.Items)

{

if (currentMenuItem.GetType() != typeof(Xwt.SeparatorMenuItem))

{

currentMenuItem.Label = Localizator.GetString("FC" + currentMenuItem.Tag);

TranslateMenu(currentMenuItem.SubMenu);

}

}

}

catch { }

}

}

}

KeyboardHelpButton.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using pluginner.Toolkit;

using Xwt;

using mucss;

namespace fcmd

{

class KeyboardHelpButton : Widget

{

HBox hb = new HBox();

Label lblF = new Label();

Label lblD = new Label();

Stylist s = new Stylist(fcmd.Properties.Settings.Default.UserTheme);

string fkey = "", text = "";

public KeyboardHelpButton(string Style = "KeyboardHelp")

{

this.Content = hb;

hb.PackStart(lblF);

hb.PackStart(lblD, true);

lblF.Text = FKey;

lblD.Text = Text;

s.Stylize(this, Style);

s.Stylize(lblF, Style+"F");

s.Stylize(lblD, Style+"Descr");

var eventHandler = new EventHandler<ButtonEventArgs> ((o, ea) => {

var handler = this.Clicked;

if (handler != null) {

handler (this, ea);

}

});

lblF.ButtonPressed += eventHandler;

lblD.ButtonPressed += eventHandler;

}

public string FKey {

get { return fkey; }

set {

fkey = value;

lblF.Text = fkey;

}

}

public string Text {

get { return text; }

set {

text = value;

lblD.Text = text;

lblD.Visible = (Text.Length) > 0 ? true : false;

}

}

public event EventHandler Clicked;

}

}

Localizator.cs:

using System;

using System.Collections.Generic;

using fcmd.Properties;

namespace fcmd

{

public static class Localizator {

static Localizator() {

LoadLanguage(Settings.Default.Language);

}

public static event EventHandler LocalizationChanged;

static Dictionary<string, string> Localization = new Dictionary<string, string>();

private static string CurrentDictionary;

public static string GetString(string Key) {

if(Localization == null) throw new InvalidOperationException("No localization file loaded!");

try {

return Localization[Key];

}

catch (Exception ex) { Console.WriteLine(@"WARNING: Locale string is not found for key: {0} ({1})", Key, ex.Message); return Key; }

}

public static void LoadLanguage(string URL, bool UseCache = true) {

URL = URL.Trim();

if (UseCache && CurrentDictionary == URL) return;

if (URL.StartsWith("(internal)")) {

ParseLangFile(Resources.lang\_RusUI.Split(new[] { '\n' }, StringSplitOptions.RemoveEmptyEntries));

}

else {

ParseLangFile(System.IO.File.ReadAllLines(URL));

}

CurrentDictionary = URL;

}

private static void ParseLangFile(IEnumerable<string> LangFile)

{

foreach (string UIFRow in LangFile)

{

try

{

string[] Parts = UIFRow.Split('=');

if(Parts.Length != 2) continue;

if (!System.Text.RegularExpressions.Regex.IsMatch(UIFRow, @"^\S\*=.\*")) continue;

Localization[Parts[0]] = Parts[1].Replace("{n}", Environment.NewLine).TrimEnd();

}

catch (Exception ex) {

Console.WriteLine(@"An error occured when parsing the language file. The invalid string is ""{0}"". It caused an error of type {1}.", UIFRow, ex.Message);

}

}

LocalizationChanged?.Invoke(null, EventArgs.Empty);

}

}

}

FileListPanel.cs:

using System;

using System.Collections.Generic;

using System.IO;

using System.Text.RegularExpressions;

using pluginner;

using pluginner.Toolkit;

using pluginner.Widgets;

using Xwt;

using Xwt.Drawing;

namespace fcmd

{

public class FileListPanel : Table

{

public int dfIcon = 0;

public int dfURL = 1;

public int dfDisplayName = 2;

public int dfSize = 3;

public int dfChanged = 4;

public int dfDirItem = 5;

public IFSPlugin FS;

public LightScroller DiskBox = new LightScroller();

public HBox DiskList = new HBox();

public List<Button> DiskButtons = new List<Button>();

EventHandler goRootDelegate = null;

EventHandler goUpDelegate = null;

public TextEntry UrlBox = new TextEntry();

public ListView2 ListingView = new ListView2();

public HBox QuickSearchBox = new HBox();

public TextEntry QuickSearchText = new TextEntry();

public Label StatusBar = new Label("Information bar");

public Table StatusTable = new Table();

public ProgressBar StatusProgressbar = new ProgressBar();

TextEntry CLIoutput = new TextEntry { MultiLine = true, ShowFrame = true, Visible = false, HeightRequest = 50 };

TextEntry CLIprompt = new TextEntry();

public event TypedEvent<string> Navigate;

public event TypedEvent<string> OpenFile;

public SizeDisplayPolicy CurShortenKB, CurShortenMB, CurShortenGB;

private string SBtext1, SBtext2;

private Stylist s;

public FileListPanel(string BookmarkXML = null, string CSS = null, string InfobarText1 = "{Name}", string InfobarText2 = "F: {FileS}, D: {DirS}")

{

s = new Stylist(CSS);

SBtext1 = InfobarText1;

SBtext2 = InfobarText2;

BuildUI(BookmarkXML);

DiskBox.Content = DiskList;

DiskBox.CanScrollByY = false;

DefaultColumnSpacing = 0;

DefaultRowSpacing = 0;

string fontFamily = fcmd.Properties.Settings.Default.UserFileListFontFamily;

ListingView.FontForFileNames = String.IsNullOrWhiteSpace(fontFamily) ? Font.SystemFont : Font.FromName(fontFamily);

Add(DiskBox, 0, 0, 1, 1, true, false, WidgetPlacement.Fill);

Add(UrlBox, 0, 1, 1, 1, true, false, WidgetPlacement.Fill);

Add(ListingView, 0, 2 , 1, 3, false, true);

Add(QuickSearchBox, 0, 3 , 1, 3);

Add(StatusBar, 0, 4, 1, 3);

Add(StatusProgressbar, 0, 5, 1, 3);

Add(CLIoutput, 0, 6 , 1, 3);

Add(CLIprompt, 0, 7 , 1, 3);

WriteDefaultStatusLabel();

CLIprompt.KeyReleased += CLIprompt\_KeyReleased;

QuickSearchText.GotFocus += (o, ea) => { OnGotFocus(ea); };

QuickSearchText.KeyPressed += QuickSearchText\_KeyPressed;

QuickSearchBox.PackStart(QuickSearchText, true, true);

QuickSearchBox.Visible = false;

}

void QuickSearchText\_KeyPressed(object sender, KeyEventArgs e)

{

if (e.Key == Key.Escape)

{

QuickSearchText.Text = "";

QuickSearchBox.Visible = false;

ListingView.AllowedToPoint.Clear();

return;

}

ListingView.Sensitive = false;

ListingView.AllowedToPoint.Clear();

foreach (ListView2Item lvi in ListingView.Items)

{

if(lvi.Data[1].ToString().StartsWith(QuickSearchText.Text)) {

ListingView.AllowedToPoint.Add(lvi.RowNo);

}

}

ListingView.Sensitive = true;

if (ListingView.AllowedToPoint.Count > 0) {

if (ListingView.SelectedRow < ListingView.AllowedToPoint[0]

||

ListingView.SelectedRow > ListingView.AllowedToPoint[ListingView.AllowedToPoint.Count-1]

)

{

ListingView.SelectedRow = ListingView.AllowedToPoint[0];

ListingView.ScrollToRow(ListingView.AllowedToPoint[0]);

}

}

}

void CLIprompt\_KeyReleased(object sender, KeyEventArgs e)

{

if (e.Key == Key.Return) {

if (Regex.Match(CLIprompt.Text, "cd|chdir|md|rd|del|deltree|move|copy|cls").Success)

{

CLIprompt.Text = "";

return;

}

CLIoutput.Visible = true;

string stdin = CLIprompt.Text;

CLIprompt.Text = "";

CLIoutput.Text += stdin;

FS.CLIstdinWriteLine(stdin);

}

}

void FS\_CLIpromptChanged(string data)

{

CLIprompt.PlaceholderText = data;

}

void FS\_CLIstdoutDataReceived(string data)

{

Application.Invoke(delegate

{

CLIoutput.Text += "\n" + data;

});

}

public void BuildUI(string BookmarkXML = null)

{

UrlBox.ShowFrame = false;

UrlBox.GotFocus += (o, ea) => { OnGotFocus(ea); };

UrlBox.KeyReleased += UrlBox\_KeyReleased;

BookmarkTools bmt = new BookmarkTools(BookmarkXML, "QuickAccessBar");

bmt.DisplayBookmarks(

DiskList,

(url => NavigateTo(url)),

s

);

bmt = new BookmarkTools(BookmarkXML);

foreach (Button b in DiskButtons)

{

s.Stylize(b);

}

s.Stylize(DiskBox);

s.Stylize(UrlBox);

s.Stylize(ListingView);

s.Stylize(QuickSearchBox);

s.Stylize(CLIoutput, "TerminalOutput");

s.Stylize(CLIprompt, "TerminalPrompt");

s.Stylize(StatusTable);

ListingView.KeyReleased += ListingView\_KeyReleased;

ListingView.GotFocus += (o, ea) => { OnGotFocus(ea); };

ListingView.PointerMoved += ListingView\_PointerMoved;

ListingView.SelectionChanged += ListingView\_SelectionChanged;

ListingView.PointedItemDoubleClicked += pointed\_item => { OpenPointedItem(); };

ListingView.EditComplete += ListingView\_EditComplete;

StatusBar.Wrap = WrapMode.Word;

}

void ListingView\_EditComplete(EditableLabel el, ListView2 lv)

{

string Url1 = FS.CurrentDirectory + FS.DirSeparator + ListingView.PointedItem.Data[dfDisplayName];

string Url2 = FS.CurrentDirectory + FS.DirSeparator + el.Text;

try {

if(FS.DirectoryExists(Url1))

FS.MoveDirectory(Url1, Url2);

else

FS.MoveFile(Url1, Url2);

StatusBar.Text = ListingView.PointedItem.Data[dfDisplayName] + " → " + el.Text;

}

catch(Exception ex) {

MessageDialog.ShowWarning(ex.Message);

el.Text = ListingView.PointedItem.Data[dfDisplayName].ToString();

}

}

void ListingView\_SelectionChanged(List<ListView2Item> data)

{

WriteDefaultStatusLabel();

}

void ListingView\_PointerMoved(ListView2Item data)

{

WriteDefaultStatusLabel();

}

void UrlBox\_KeyReleased(object sender, KeyEventArgs e)

{

if (e.Key == Key.Return)

{

LoadDir(UrlBox.Text);

}

}

void ListingView\_KeyReleased(object sender, KeyEventArgs e)

{

if (e.Key == Key.Return && ListingView.SelectedRow > -1)

{

OpenPointedItem();

return;

}

if ((int)e.Key < 65000)

{

QuickSearchText.Text += e.Key.ToString();

QuickSearchBox.Visible = true;

QuickSearchText.SetFocus();

return;

}

if(Utilities.GetXwtBackendName() == "WPF")

ListingView.OnKeyPressed(e);

}

void OpenPointedItem()

{

NavigateTo(ListingView.PointedItem.Data[dfURL].ToString());

}

private void NavigateTo(string url, int? ClearHistory = null)

{

if (!url.Contains("://")) {

NavigateTo(FS.CurrentDirectory + FS.DirSeparator + url);

}

if (ClearHistory != null) {

}

try

{

if (FS.DirectoryExists(url))

{

var navigate = Navigate;

if (navigate != null) {

navigate(url);

} else {

Console.WriteLine("WARNING: the event FLP.Navigate was not handled by the host");

}

LoadDir(url);

return;

}

else

{

var openFile = OpenFile;

if (openFile != null) {

openFile(url);

} else {

Console.WriteLine("WARNING: the event FLP.OpenFile was not handled by the host");

}

}

}

catch (PleaseSwitchPluginException)

{

throw;

}

catch (Exception ex)

{

ListingView.Sensitive = true;

ListingView.Cursor = CursorType.Arrow;

MessageDialog.ShowError(ex.Message);

Console.WriteLine(ex.Message + "\n" + ex.StackTrace);

WriteDefaultStatusLabel();

}

}

public void LoadDir(string URL, SizeDisplayPolicy ShortenKB, SizeDisplayPolicy ShortenMB, SizeDisplayPolicy ShortenGB)

{

CurShortenKB = ShortenKB; CurShortenMB = ShortenMB; CurShortenGB = ShortenGB;

if (FS == null) throw new InvalidOperationException("No filesystem is binded to this FileListPanel");

FS.CLIstdoutDataReceived += FS\_CLIstdoutDataReceived;

FS.CLIstderrDataReceived += (stderr) => { CLIoutput.Text += "\n" + stderr; Utilities.ShowWarning(stderr); };

FS.CLIpromptChanged += FS\_CLIpromptChanged;

if (FS.CurrentDirectory == null) {

FS.StatusChanged += FS\_StatusChanged;

FS.ProgressChanged += FS\_ProgressChanged;

}

if (URL == "." && FS.CurrentDirectory == null) {

LoadDir(

"file://"+Directory.GetCurrentDirectory(),

ShortenKB,

ShortenMB,

ShortenGB

);

return;

}

ListingView.Cursor = CursorType.Wait;

ListingView.Sensitive = false;

string oldCurDir = FS.CurrentDirectory;

try

{

FS.CurrentDirectory = URL;

UrlBox.Text = URL;

ListingView.Clear();

UrlBox.Text = URL;

string updir = URL + FS.DirSeparator+"..";

string rootdir = FS.GetMetadata(URL).RootDirectory;

uint counter = 0;

const uint per\_number = ~(((~(uint)0) >> 10) << 10);

IEnumerable<DirItem> dis = FS.DirectoryContent;

foreach (DirItem di in dis)

{

List<Object> Data = new List<Object>();

List<Boolean> EditableFileds = new List<bool>();

Data.Add(di.IconSmall ?? Image.FromResource("fcmd.Resources.image-missing.png")); EditableFileds.Add(false);

Data.Add(di.URL); EditableFileds.Add(false);

Data.Add(di.TextToShow); EditableFileds.Add(true);

if (di.TextToShow == "..")

{

Data.Add("<↑ UP>"); EditableFileds.Add(false); EditableFileds[2] = false;

Data.Add(FS.GetMetadata(di.URL).LastWriteTimeUTC.ToLocalTime()); EditableFileds.Add(false);

updir = di.URL;

}

else if (di.IsDirectory)

{

Data.Add("<DIR>"); EditableFileds.Add(false);

Data.Add(di.Date); EditableFileds.Add(false);

}

else

{

Data.Add(KiloMegaGigabyteConvert(di.Size, ShortenKB, ShortenMB, ShortenGB)); EditableFileds.Add(false);

Data.Add(di.Date); EditableFileds.Add(false);

}

Data.Add(di);

ListingView.AddItem(Data, EditableFileds, di.URL);

if ((++counter & per\_number) == 0) {

Application.MainLoop.DispatchPendingEvents();

}

}

goUpDelegate = (o, ea) => { LoadDir(updir); };

goRootDelegate = (o, ea) => { LoadDir(rootdir); };

}

catch (Exception ex)

{

if (ex is pluginner.PleaseSwitchPluginException)

{

pluginfinder pf = new pluginfinder();

FS = pf.GetFSplugin(URL);

LoadDir(URL, ShortenKB, ShortenMB, ShortenGB);

}

else if (ex is NullReferenceException)

{

MessageDialog.ShowWarning(ex.Message, ex.StackTrace + "\nInner exception: " + ex.InnerException.Message ?? "none");

LoadDir(oldCurDir, ShortenKB, ShortenMB, ShortenGB);

}

else

{

MessageDialog.ShowWarning(ex.Message);

LoadDir(oldCurDir, ShortenKB, ShortenMB, ShortenGB);

}

}

if (ListingView.Items.Count > 0)

{ ListingView.SelectedRow = 0; ListingView.ScrollerIn.ScrollTo(0, 0); }

ListingView.SetFocus();

ListingView.Sensitive = true;

ListingView.Cursor = CursorType.Arrow;

}

private void FS\_StatusChanged(string data)

{

if (data.Length == 0)

WriteDefaultStatusLabel();

else

StatusBar.Text = data;

}

private void FS\_ProgressChanged(double data)

{

if (data > 0 && data <= 1) {

StatusProgressbar.Visible = true;

StatusProgressbar.Fraction = data;

}

else

{

StatusProgressbar.Visible = false;

}

}

public void LoadDir()

{

LoadDir(FS.CurrentDirectory);

}

public void LoadDir(string URL)

{

LoadDir(URL, CurShortenKB, CurShortenMB, CurShortenGB);

}

private string KiloMegaGigabyteConvert(long Input, SizeDisplayPolicy ShortenKB, SizeDisplayPolicy ShortenMB, SizeDisplayPolicy ShortenGB)

{

double ShortenedSize;

if (Input > 1099511627776) return (Input / 1099511627776) + " TB";

if (Input > 1073741824)

{

ShortenedSize = Input / 1073741824;

switch (ShortenGB)

{

case SizeDisplayPolicy.OneNumeral:

return string.Format("{0:0.#} GB", ShortenedSize);

case SizeDisplayPolicy.TwoNumeral:

return string.Format("{0:0.##} GB", ShortenedSize);

}

}

if (Input > 1048576)

{

ShortenedSize = Input / 1048576;

switch (ShortenMB)

{

case SizeDisplayPolicy.OneNumeral:

return string.Format("{0:0.#} MB", ShortenedSize);

case SizeDisplayPolicy.TwoNumeral:

return string.Format("{0:0.##} MB", ShortenedSize);

}

}

if (Input > 1024)

{

ShortenedSize = Input / 1024;

switch (ShortenKB)

{

case SizeDisplayPolicy.OneNumeral:

return string.Format("{0:0.#} KB", ShortenedSize);

case SizeDisplayPolicy.TwoNumeral:

return string.Format("{0:0.##} KB", ShortenedSize);

}

}

return Input + " B";

}

public enum SizeDisplayPolicy

{

DontShorten = 0, OneNumeral = 1, TwoNumeral = 2

}

public T GetValue<T>(int Field) {

return (T)ListingView.PointedItem.Data[Field];

}

public string GetValue(int Field) {

return (string)ListingView.PointedItem.Data[Field];

}

private void AddSysDrives()

{

foreach (DriveInfo di in DriveInfo.GetDrives())

{

string d = di.Name;

Button NewBtn = new Button(null, d);

NewBtn.Clicked += (o, ea) => { NavigateTo("file://" + d); };

NewBtn.CanGetFocus = false;

NewBtn.Style = ButtonStyle.Flat;

NewBtn.Margin = -3;

NewBtn.Cursor = CursorType.Hand;

NewBtn.Sensitive = di.IsReady;

if (di.IsReady)

{

NewBtn.TooltipText = di.VolumeLabel + " (" + di.DriveFormat + ")";

}

switch (di.DriveType)

{

case DriveType.Fixed:

NewBtn.Image = Image.FromResource(GetType(), "fcmd.Resources.drive-harddisk.png");

break;

case DriveType.CDRom:

NewBtn.Image = Image.FromResource(GetType(), "fcmd.Resources.drive-optical.png");

break;

case DriveType.Removable:

NewBtn.Image = Image.FromResource(GetType(), "fcmd.Resources.drive-removable-media.png");

break;

case DriveType.Network:

NewBtn.Image = Image.FromResource(GetType(), "fcmd.Resources.network-server.png");

break;

case DriveType.Ram:

NewBtn.Image = Image.FromResource(GetType(), "fcmd.Resources.emblem-system.png");

break;

case DriveType.Unknown:

NewBtn.Image = Image.FromResource(GetType(), "fcmd.Resources.image-missing.png");

break;

}

if (d.StartsWith("A:")) NewBtn.Image = Image.FromResource(GetType(), "fcmd.Resources.media-floppy.png");

if (d.StartsWith("B:")) NewBtn.Image = Image.FromResource(GetType(), "fcmd.Resources.media-floppy.png");

if (d.StartsWith("/dev")) NewBtn.Image = Image.FromResource(GetType(), "fcmd.Resources.preferences-desktop-peripherals.png");

if (d.StartsWith("/proc")) NewBtn.Image = Image.FromResource(GetType(), "fcmd.Resources.emblem-system.png");

if (d == "/") NewBtn.Image = Image.FromResource(GetType(), "fcmd.Resources.root-folder.png");

s.Stylize(NewBtn);

DiskList.PackStart(NewBtn);

}

}

private void AddLinuxMounts()

{

if (Directory.Exists(@"/mnt"))

{

foreach (string dir in Directory.GetDirectories(@"/mnt/"))

{

Button NewBtn = new Button(null, dir.Replace("/mnt/", ""));

NewBtn.Clicked += (o, ea) => { NavigateTo("file://" + dir); };

NewBtn.CanGetFocus = false;

NewBtn.Style = ButtonStyle.Flat;

NewBtn.Margin = -3;

NewBtn.Cursor = CursorType.Hand;

NewBtn.Image = Image.FromResource(GetType(), "fcmd.Resources.drive-removable-media.png");

s.Stylize(NewBtn);

DiskList.PackStart(NewBtn);

}

}

else AddSysDrives();

}

private void WriteDefaultStatusLabel()

{

StatusProgressbar.Visible = false;

if(ListingView.SelectedItems.Count<1)

StatusBar.Text = MakeStatusbarText(SBtext1);

else

StatusBar.Text = MakeStatusbarText(SBtext2);

}

private string MakeStatusbarText(string Template)

{

string txt = Template;

if (ListingView.PointedItem != null) {

DirItem di = (DirItem)ListingView.PointedItem.Data[dfDirItem];

txt = txt.Replace("{FullName}", di.TextToShow);

txt = txt.Replace("{AutoSize}", KiloMegaGigabyteConvert(di.Size, CurShortenKB, CurShortenMB, CurShortenMB));

txt = txt.Replace("{Date}", di.Date.ToShortDateString());

txt = txt.Replace("{Time}", di.Date.ToLocalTime().ToShortTimeString());

txt = txt.Replace("{SelectedItems}", ListingView.SelectedItems.Count.ToString());

}

return txt;

}

}

}