HTML to RTF .Net

(Multi-platform .Net library)

SautinSoft

Linux development manual

Table of Contents

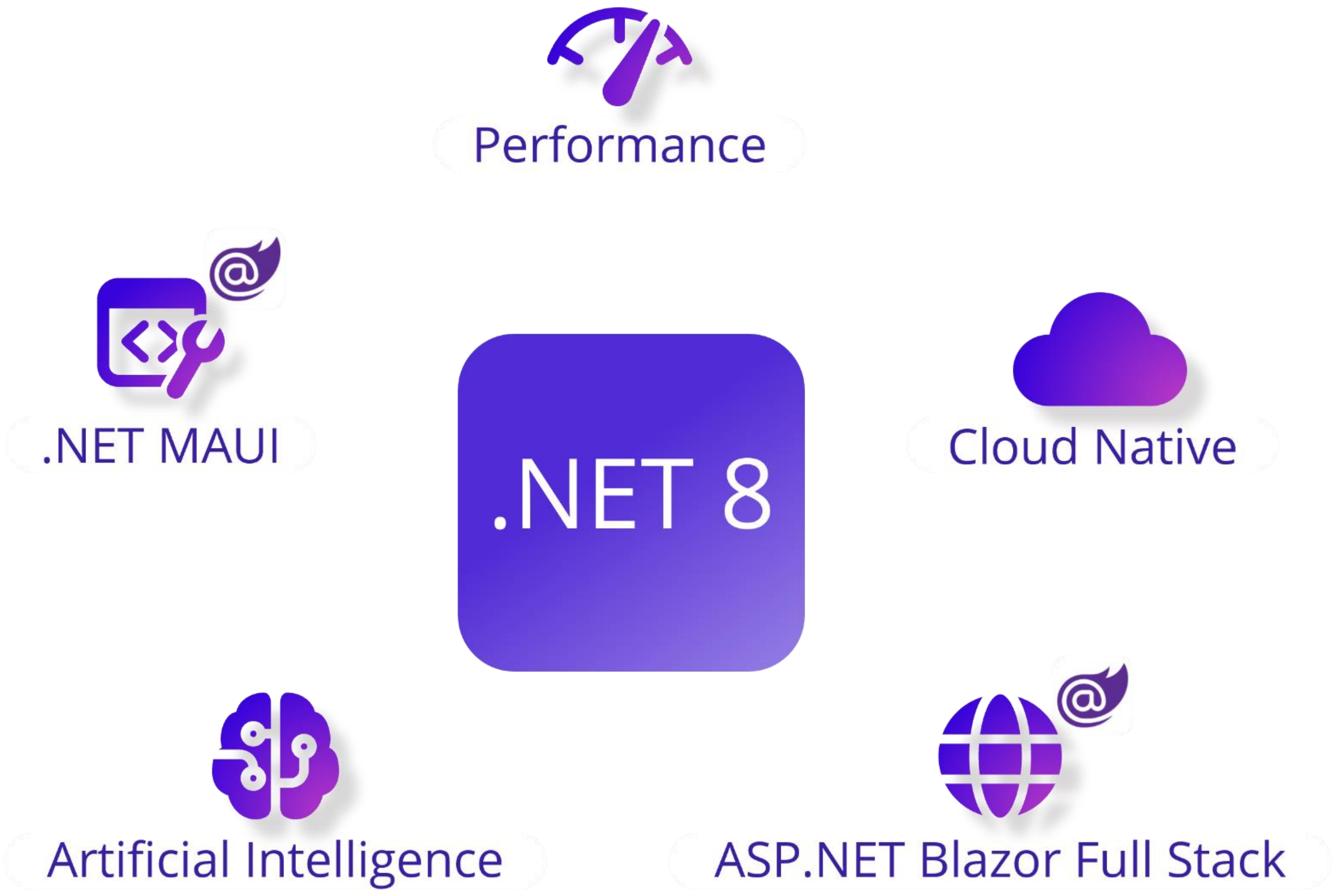
1. Preparing environment

In order to build multi-platform applications using .NET Core on Linux, the first steps are for installing in our Linux machine the required tools.

We need to install .NET Core SDK from Microsoft and to allow us to develop easier, we will install an advance editor with a lot of features, Visual Studio Code from Microsoft.

Both installations are very easy and the detailed description can be found by these two links:

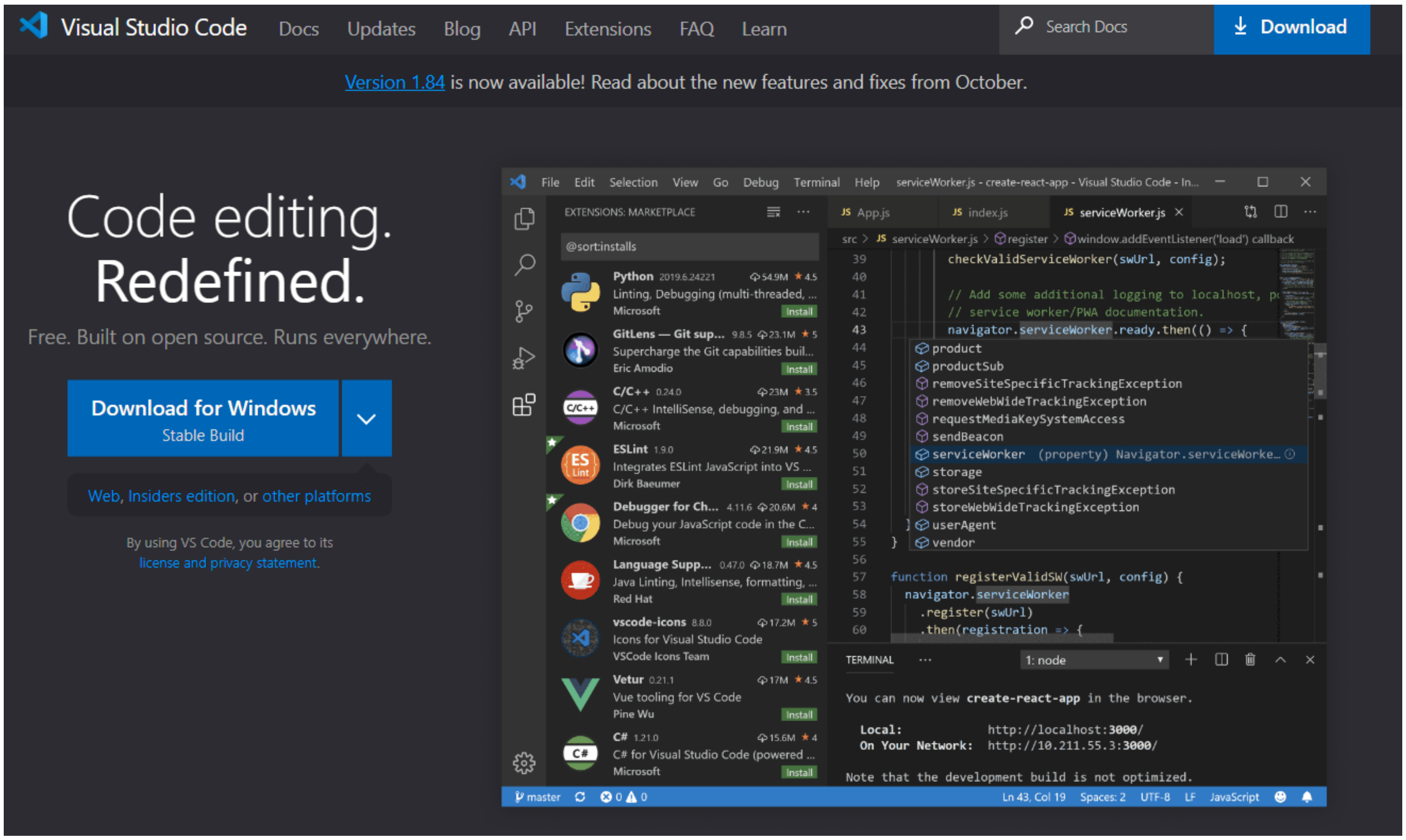
Install .NET Core SDK for Linux.



Install VS Code for Linux.

Once installed VS Code, you need to install a C# extension to facilitate us to code and debugging:

Install C# extension.


Visual Studio Code
Docs
Updates
Blog
API
Extensions
FAQ
Learn
Search Docs
Download
Download for Windows
Stable Build
File Edit Selection View Go Debug Terminal Help
serviceWorker.js - create-react-app - Visual Studio Code - In...
EXTENSIONS: MARKETPLACE
@sort:installs
App.js
index.js
serviceWorker.js
src > serviceWorker.js > register > window.addEventListener('load') callback
TERMINAL
1: node
master
Ln 43, Col 19 Spaces: 2 UTF-8 LF JavaScript

Version 1.84 is now available! Read about the new features and fixes from October.

Code editing.
Redefined.

Free. Built on open source. Runs everywhere.

Web, Insiders edition, or other platforms

By using VS Code, you agree to its license and privacy statement.

Python 2019.6.24221 54.9M 4.5
Linting, Debugging (multi-threaded, ...
Microsoft Install

GitLens — Git sup... 9.8.5 23.1M 5
Supercharge the Git capabilities buil...
Eric Amodio Install

C/C++ 0.24.0 23M 3.5
C/C++ IntelliSense, debugging, and ...
Microsoft Install

ESLint 1.9.0 21.9M 4.5
Integrates ESLint JavaScript into VS ...
Dirk Baeumer Install

Debugger for Ch... 4.11.6 20.6M 4
Debug your JavaScript code in the C...
Microsoft Install

Language Supp... 0.47.0 18.7M 4.5
Java Linting, Intellisense, formatting, ...
Red Hat Install

vscode-icons 8.8.0 17.2M 5
Icons for Visual Studio Code
VSCode Icons Team Install

Vetur 0.21.1 17M 4.5
Vue tooling for VS Code
Pine Wu Install

C# 1.21.0 15.6M 4
C# for Visual Studio Code (powered ...
Microsoft Install

```
      checkValidServiceWorker(swUrl, config);

      // Add some additional logging to localhost, p
      // service worker/PWA documentation.
      navigator.serviceWorker.ready.then(() => {
        product
        productSub
        removeSiteSpecificTrackingException
        removeWebWideTrackingException
        requestMediaKeySystemAccess
        sendBeacon
        serviceWorker  (property) Navigator.serviceWorke...
        storage
        storeSiteSpecificTrackingException
        storeWebWideTrackingException
      } userAgent
  }     vendor

  function registerValidSW(swUrl, config) {
    navigator.serviceWorker
      .register(swUrl)
      .then(registration => {
```

```
You can now view create-react-app in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://10.211.55.3:3000/

Note that the development build is not optimized.
```

In next paragraphs we will explain in detail how to create simple console application. All of them are based on this VS Code guide:

Get Started with C# and Visual Studio Code

Not only is possible to create .NET Core applications that will run on Linux using Linux as a developing platform. It is also possible to create it using a Windows machine and any modern Visual Studio version, as Microsoft Visual Studio Community 2017.

1.1. Check the installed Fonts availability

Check that the directory with fonts "/usr/share/fonts/truetype" is exist.

Also check that it contains *.ttf files.

If you don't see this folder, make these steps:

1. Download the archive with *.ttf fonts: https://sautinsoft.com/components/fonts.tar
2. Uncompress the downloaded font's archive to a directory and add it to the font path, a list of directories containing fonts:

```
# tar xvzf
```

3. Create a directory for new fonts

   ```
   # mkdir /usr/share/fonts/truetype
   ```

4. Move the uncompressed font files to the new font directory

   ```
   # mv *.ttf /usr/share/fonts/truetype
   ```

5. Navigate to the font directory

   ```
   # cd /usr/share/fonts/truetype
   ```

6. Create fonts.scale and fonts.dir

   ```
   # mkfontscale && mkfontdir
   # fc-cache
   ```

7. Add the new font directory to the X11 font path

   ```
   # chkfontpath --add /usr/share/fonts/truetype
   ```

8. Restart X font server

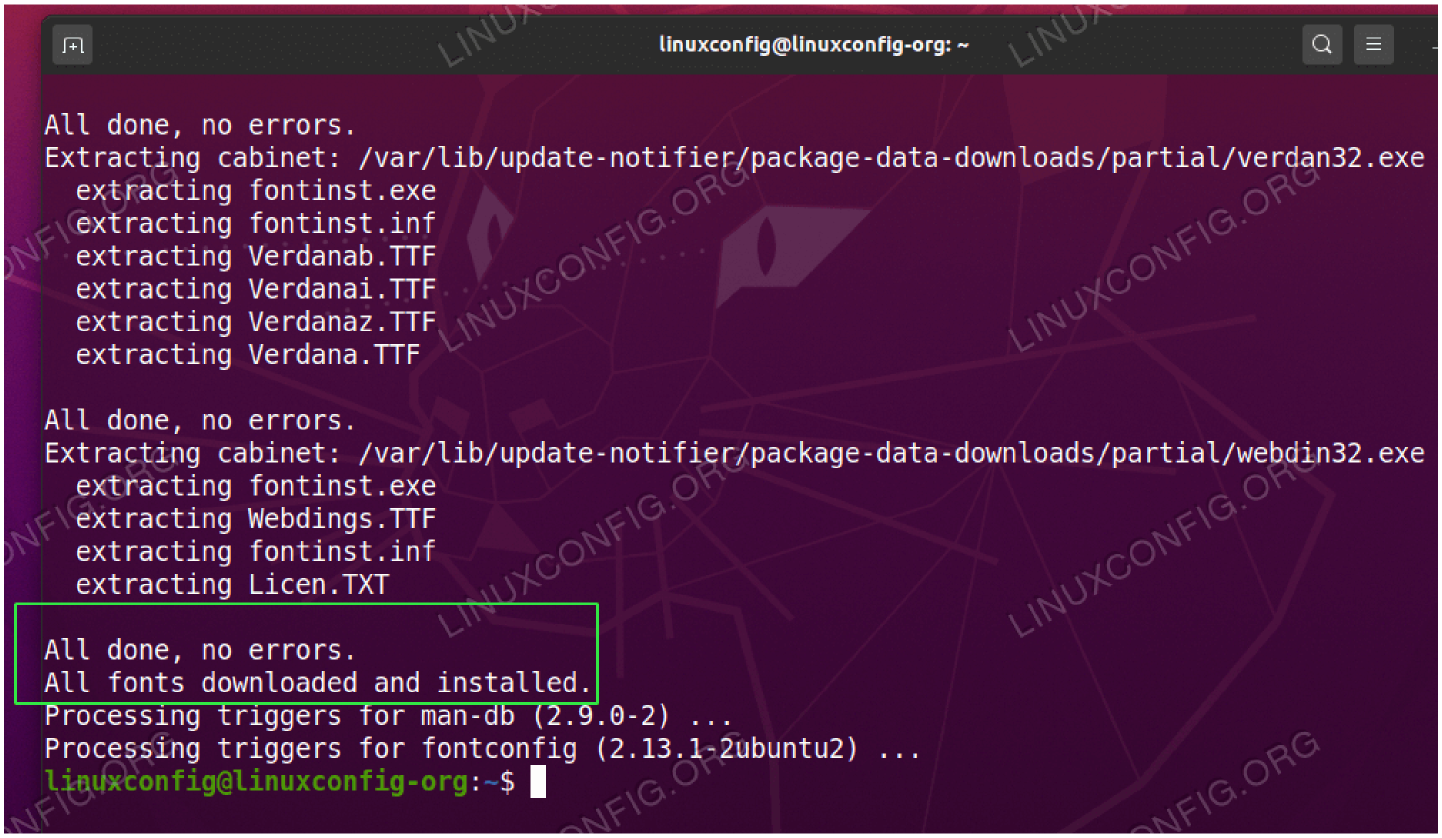
   ```
   # /etc/rc.d/init.d/xfs restart
   ```

You can verify the successful addition of the new path by running `chkfontpath` command or by listing X font server's /etc/X11/XF86Config file.

If you do not have root access, copy the *.ttf to ~/.fonts directory instead.

Or you may install “Microsoft TrueType core fonts” using terminal and command:

```
$ sudo apt install ttf-mscorefonts-installer
```



```
All done, no errors.
Extracting cabinet: /var/lib/update-notifier/package-data-downloads/partial/verdan32.exe
  extracting fontinst.exe
  extracting fontinst.inf
  extracting Verdanab.TTF
  extracting Verdanai.TTF
  extracting Verdanaz.TTF
  extracting Verdana.TTF

All done, no errors.
Extracting cabinet: /var/lib/update-notifier/package-data-downloads/partial/webdin32.exe
  extracting fontinst.exe
  extracting Webdings.TTF
  extracting fontinst.inf
  extracting Licen.TXT

All done, no errors.
All fonts downloaded and installed.
Processing triggers for man-db (2.9.0-2) ...
Processing triggers for fontconfig (2.13.1-2ubuntu2) ...
linuxconfig@linuxconfig-org:~$
```

Read more about TrueType Fonts and "How to install Microsoft fonts, How to update fonts cache files, How to confirm new fonts installation" .

With these steps, we will ready to start developing.

In next paragraphs we will explain in detail how to create simple console application. All of them are based on this VS Code guide:

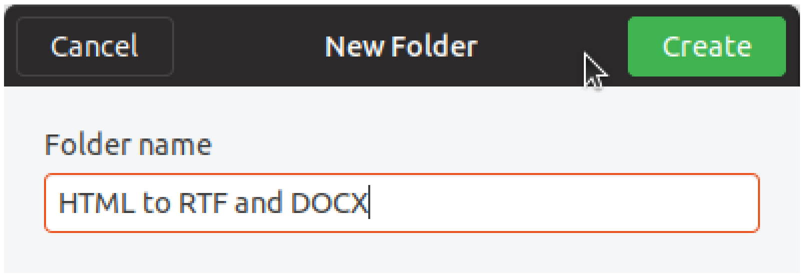
Get Started with C# and Visual Studio Code

Not only is possible to create .NET Core applications that will run on Linux using Linux as a developing platform. It is also possible to create it using a Windows machine and any modern Visual Studio version, as Microsoft Visual Studio Community 2022.

2. Creating “Convert HTML to RTF/DOCX” app

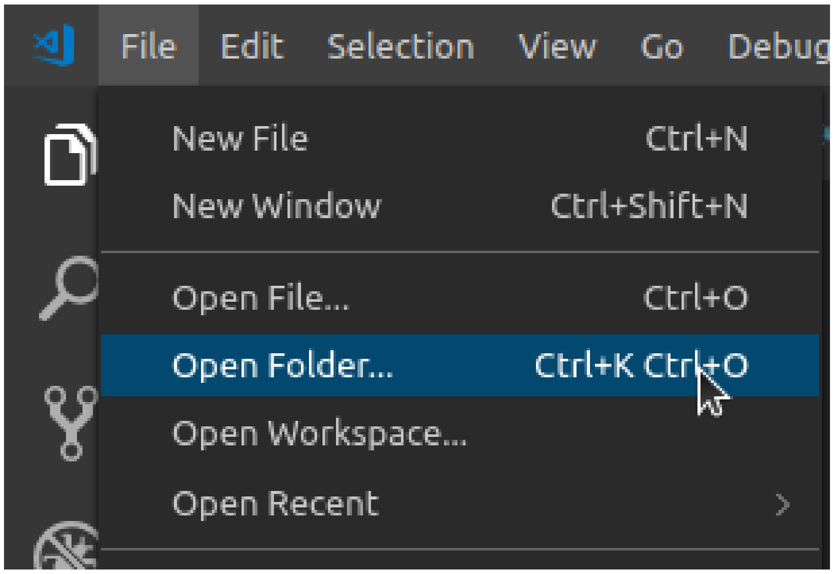
Create a new folder in your Linux machine with the name ***HTML to RTF and DOCX.***

For example, let’s create the folder “***HTML to RTF and DOCX***” on the Desktop (Right click-> New Folder):

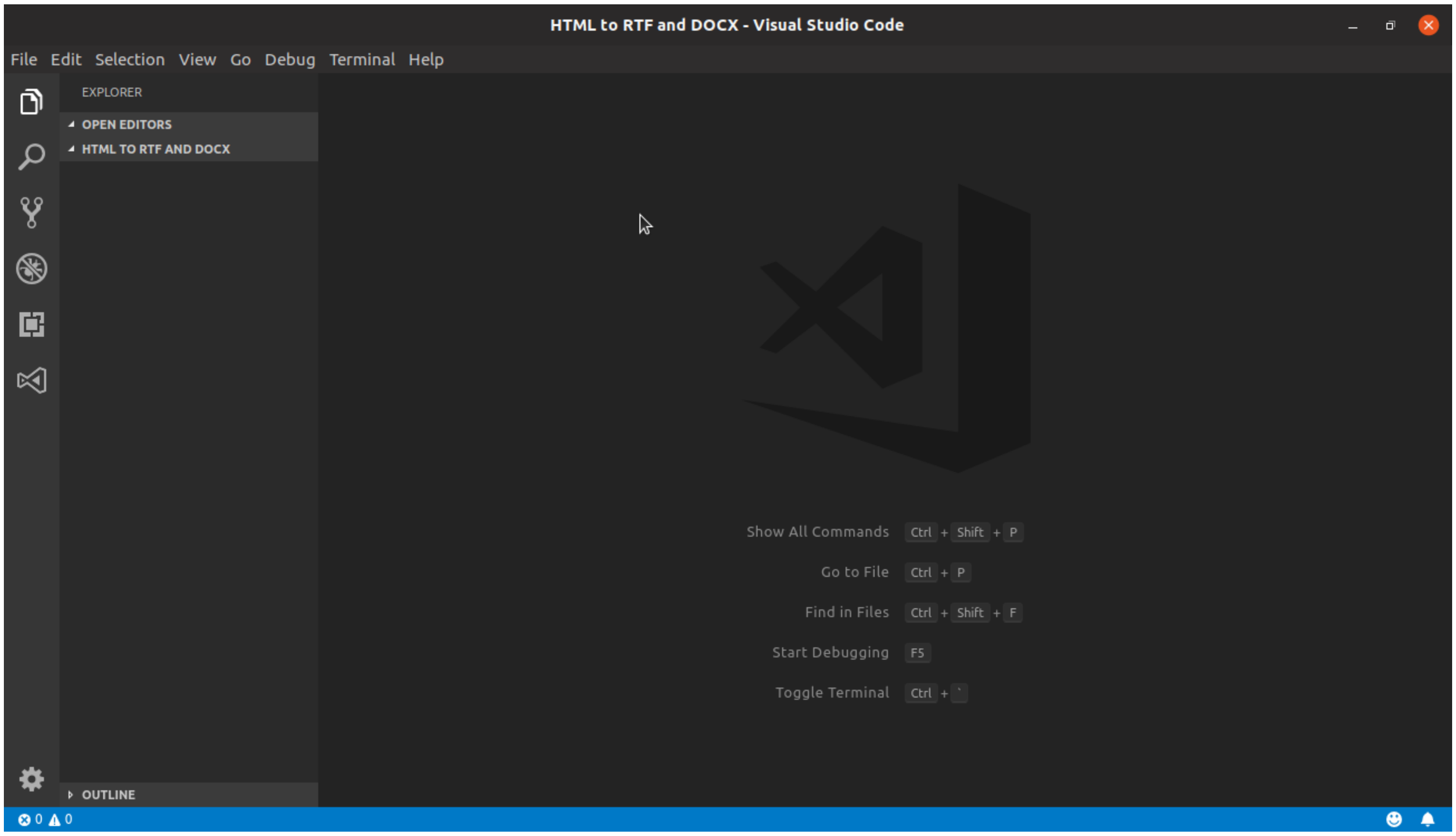


Folder name: HTML to RTF and DOCX

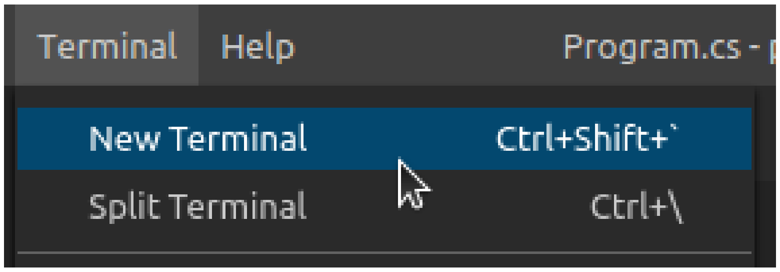
Open VS Code and click in the menu ***File->Open Folder***. From the dialog, open the folder you’ve created previously:



Next you will see the similar screen:

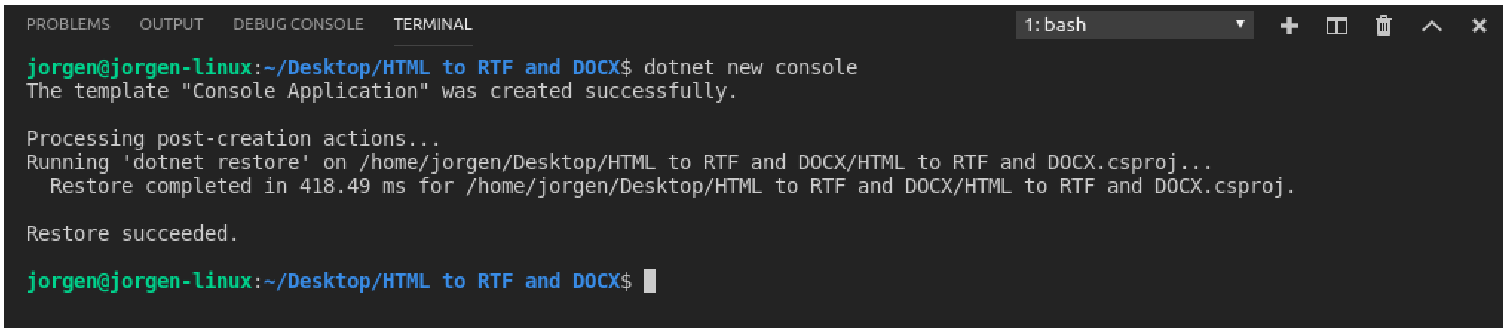

HTML to RTF and DOCX - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
EXPLORER
OPEN EDITORS
HTML TO RTF AND DOCX
OUTLINE
Show All Commands Ctrl + Shift + P
Go to File Ctrl + P
Find in Files Ctrl + Shift + F
Start Debugging F5
Toggle Terminal Ctrl + `

Now, open the integrated console – the Terminal: follow to the menu ***Terminal -> New Terminal*** (or press Ctrl+Shift+'):



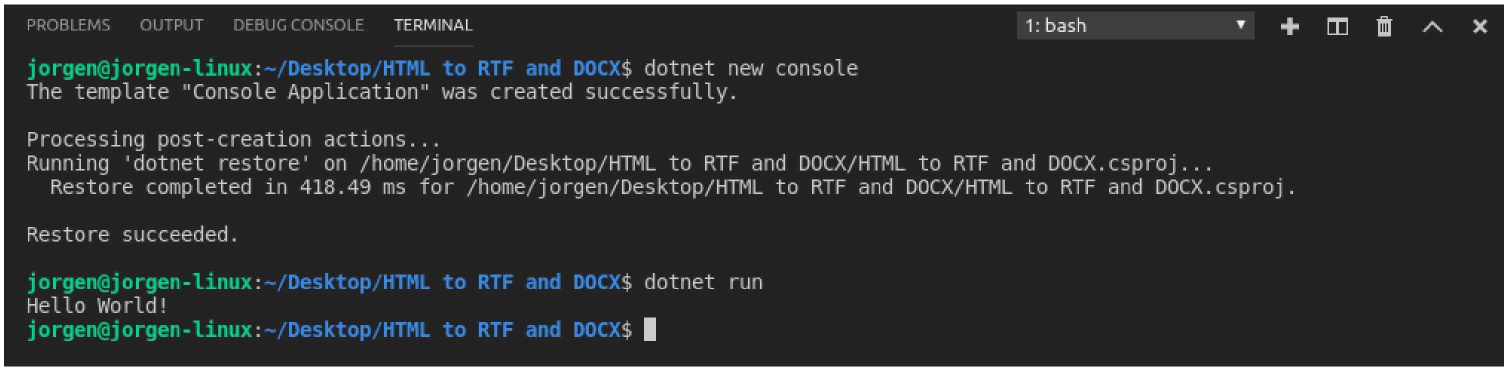
Create a new console application, using ***dotnet*** command.

Type this command in the Terminal console: ***dotnet new console***



```
jorgen@jorgen-linux:~/Desktop/HTML to RTF and DOCX$ dotnet new console
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on /home/jorgen/Desktop/HTML to RTF and DOCX/HTML to RTF and DOCX.csproj...
  Restore completed in 418.49 ms for /home/jorgen/Desktop/HTML to RTF and DOCX/HTML to RTF and DOCX.csproj.

Restore succeeded.

jorgen@jorgen-linux:~/Desktop/HTML to RTF and DOCX$
```

A new simple ***Hello world!*** console application has been created. To execute it, type this command: ***dotnet run***



```
jorgen@jorgen-linux:~/Desktop/HTML to RTF and DOCX$ dotnet new console
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on /home/jorgen/Desktop/HTML to RTF and DOCX/HTML to RTF and DOCX.csproj...
  Restore completed in 418.49 ms for /home/jorgen/Desktop/HTML to RTF and DOCX/HTML to RTF and DOCX.csproj.

Restore succeeded.

jorgen@jorgen-linux:~/Desktop/HTML to RTF and DOCX$ dotnet run
Hello World!
jorgen@jorgen-linux:~/Desktop/HTML to RTF and DOCX$ 
```

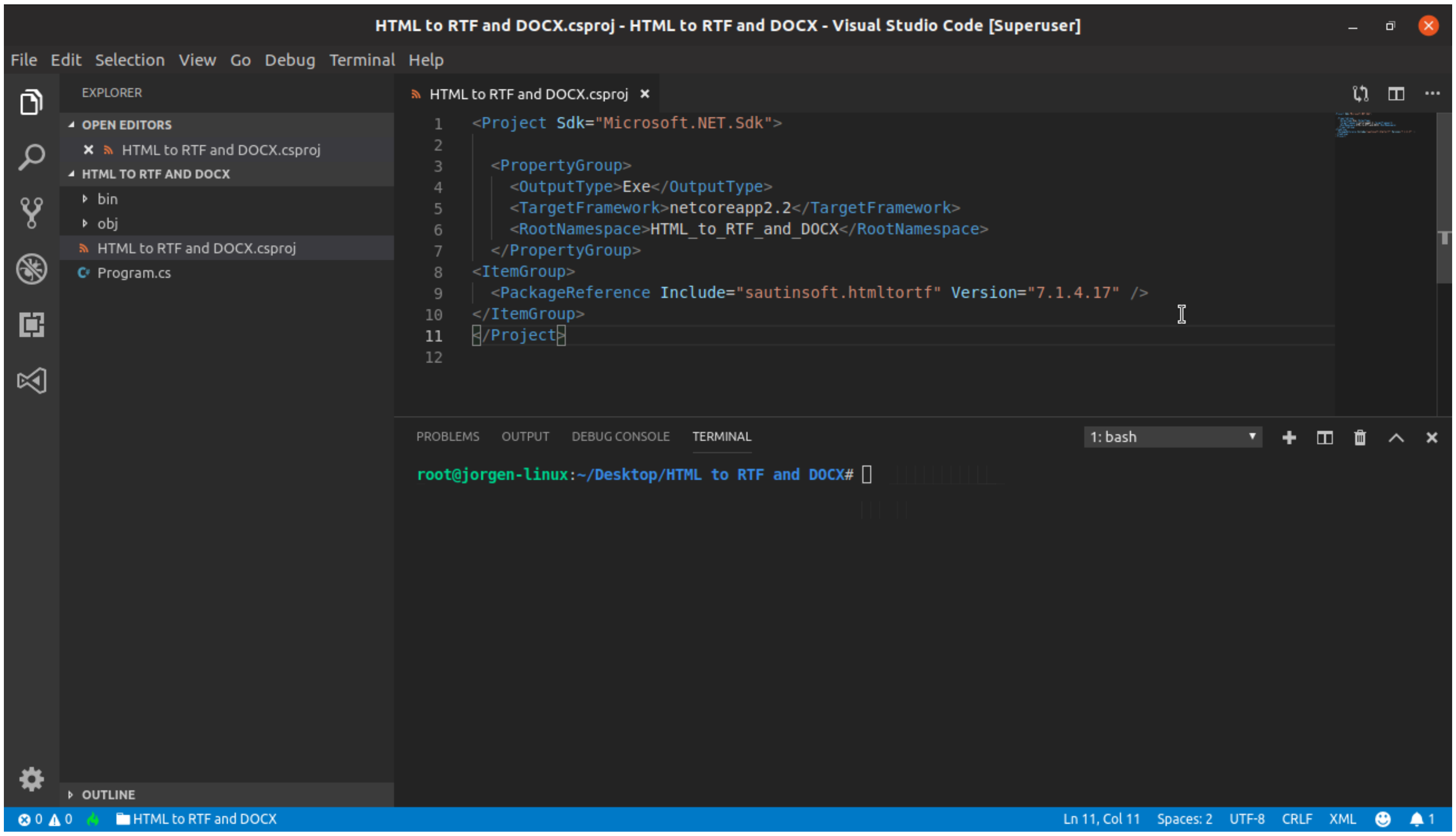
You can see the typical "Hello world!" message.

Now we are going to convert this simple application into something more interesting.

We'll transform it into an application that will convert html file to rtf and docx files.

First of all, we need to add the package reference to the ***sautinsoft.htmltortf*** assembly using Nuget.

In order to do it, follow to the ***Explorer*** and open project file "***HTML to RTF and DOCX.csproj"*** within VS Code to edit it:



```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp2.2</TargetFramework>
    <RootNamespace>HTML_to_RTF_and_DOCX</RootNamespace>
  </PropertyGroup>
<ItemGroup>
  <PackageReference Include="sautinsoft.htmltortf" Version="7.1.4.17" />
</ItemGroup>
</Project>
```

```
root@jorgen-linux:~/Desktop/HTML to RTF and DOCX# 
```

Add these lines into the file "***HTML to RTF and DOCX.csproj***":

```
<ItemGroup>
     <PackageReference Include="sautinsoft.htmltortf" Version="*" />
</ItemGroup>
```

Add this library, it will also download some of the dependencies. If the conversion does not work, then add dependencies manually:

```
<PackageReference Include="System.IO.Packaging" Version="4.4.0" />
<PackageReference Include="System.Text.Encoding.CodePages" Version="4.5.0" />
<PackageReference Include="System.Xml.XPath.XmlDocument" Version="4.3.0" />
<PackageReference Include="SkiaSharp" Version="2.88.7" />
<PackageReference Include="Svg.Skia" Version="1.0.0.18" />
<PackageReference Include="SkiaSharp.NativeAssets.Linux" Version="2.88.6" />
```

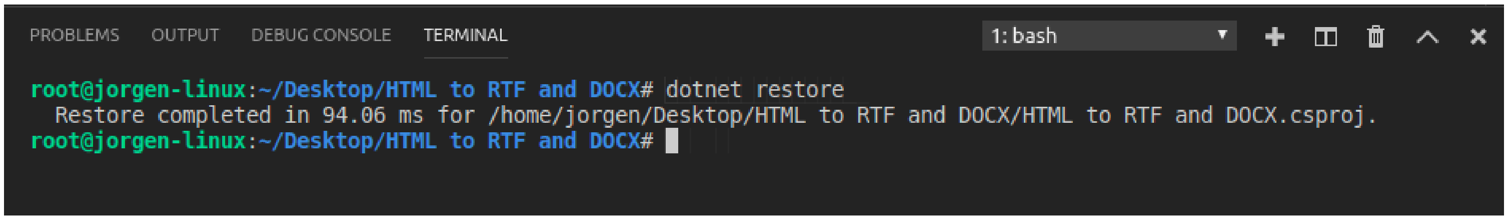
It's the reference to ***sautinsoft.htmltortf*** package from Nuget.

At the moment of writing this manual, the latest version of ***sautinsoft.htmltortf*** was 2024.X. But you may specify the latest version, to know what is the latest, follow:

https://www.nuget.org/packages/sautinsoft.htmltortf/

At once as we've added the package reference, we have to save the "***HTML to RTF and DOCX.csproj"*** and restore the added package.

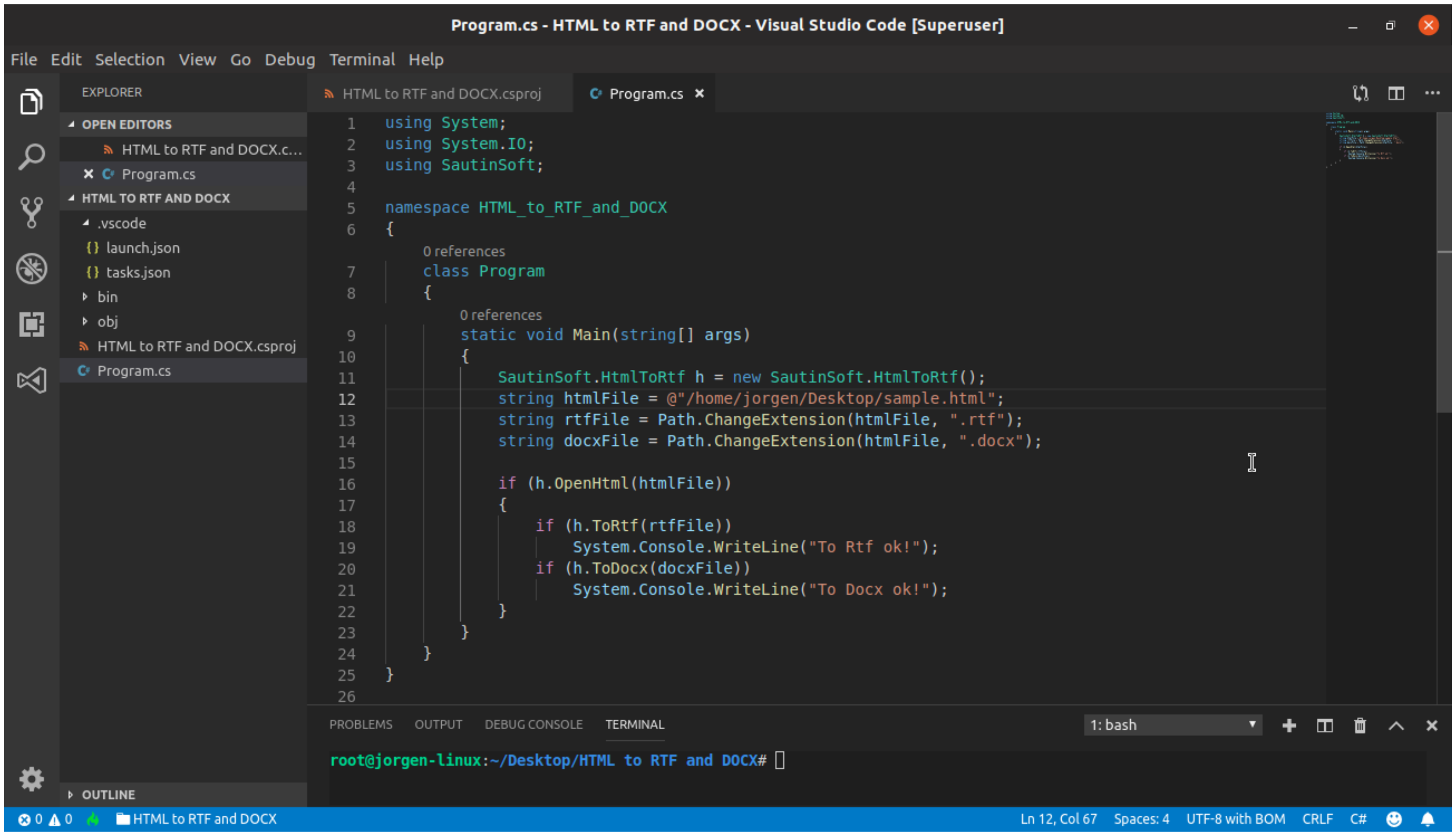
Follow to the ***Terminal*** and type the command: ***dotnet restore***



```
root@jorgen-linux:~/Desktop/HTML to RTF and DOCX# dotnet restore
  Restore completed in 94.06 ms for /home/jorgen/Desktop/HTML to RTF and DOCX/HTML to RTF and DOCX.csproj.
root@jorgen-linux:~/Desktop/HTML to RTF and DOCX# 
```

Good, now our application has the reference to ***sautinsoft.htmltortf*** package and we can write the code to convert html to rtf and docx formats.

Follow to the ***Explorer***, open the ***Program.cs***, remove all the code and type the new:


Program.cs - HTML to RTF and DOCX - Visual Studio Code [Superuser]
File Edit Selection View Go Debug Terminal Help
EXPLORER
OPEN EDITORS
HTML to RTF and DOCX.c...
Program.cs
HTML TO RTF AND DOCX
.vscode
launch.json
tasks.json
bin
obj
HTML to RTF and DOCX.csproj
Program.cs
OUTLINE
HTML to RTF and DOCX.csproj
Program.cs
0 references
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: bash
0 0 HTML to RTF and DOCX
Ln 12, Col 67 Spaces: 4 UTF-8 with BOM CRLF C#

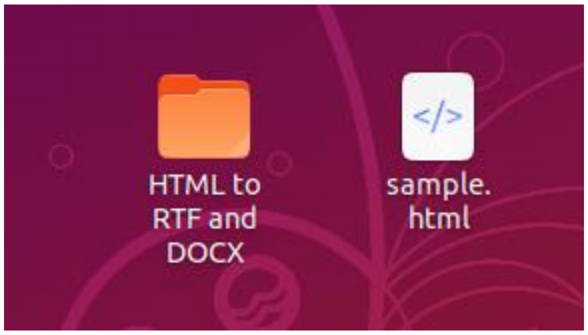
```
using System;
using System.IO;
using SautinSoft;

namespace HTML_to_RTF_and_DOCX
{
    class Program
    {
        static void Main(string[] args)
        {
            SautinSoft.HtmlToRtf h = new SautinSoft.HtmlToRtf();
            string htmlFile = @"/home/jorgen/Desktop/sample.html";
            string rtfFile = Path.ChangeExtension(htmlFile, ".rtf");
            string docxFile = Path.ChangeExtension(htmlFile, ".docx");

            if (h.OpenHtml(htmlFile))
            {
                if (h.ToRtf(rtfFile))
                    System.Console.WriteLine("To Rtf ok!");
                if (h.ToDocx(docxFile))
                    System.Console.WriteLine("To Docx ok!");
            }
        }
    }
}
```

```
root@jorgen-linux:~/Desktop/HTML to RTF and DOCX#
```

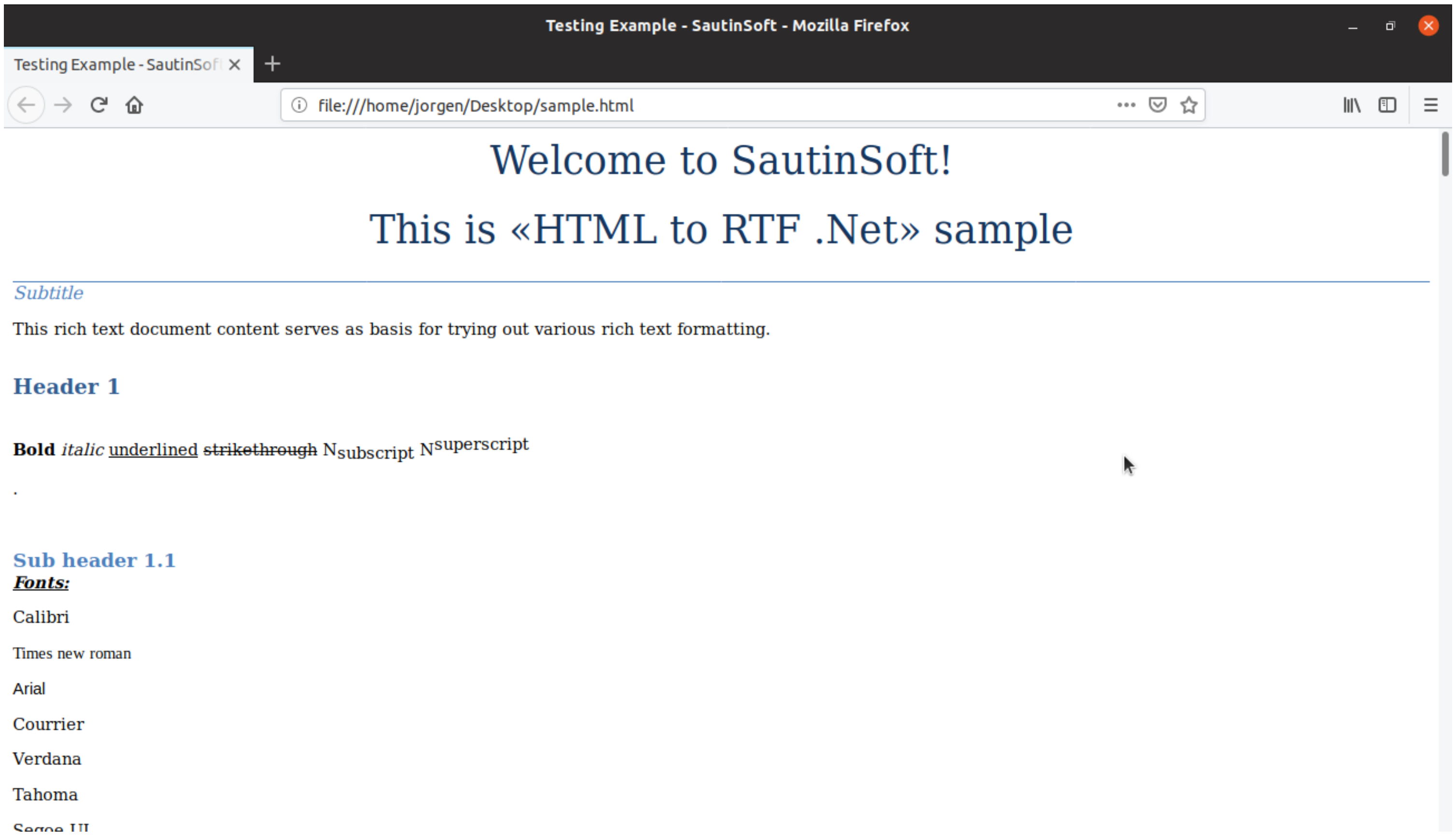
The new code:

```
using System;
using System.IO;
using SautinSoft;

namespace HTML_to_RTF_and_DOCX
{
    class Program
    {
        static void Main(string[] args)
        {
            SautinSoft.HtmlToRtf h = new SautinSoft.HtmlToRtf();
            string htmlFile = @"/home/jorgen/Desktop/sample.html";
            string rtfFile = Path.ChangeExtension(htmlFile, ".rtf");
            string docxFile = Path.ChangeExtension(htmlFile, ".docx");

            if (h.OpenHtml(htmlFile))
            {
                if (h.ToRtf(rtfFile))
                    System.Console.WriteLine("To Rtf ok!");
                if (h.ToDocx(docxFile))
                    System.Console.WriteLine("To Docx ok!");
            }
        }
    }
}
```

To make tests, we need an input HTML document. For our tests, let's place the HTML file with the name "sample.html" at the Desktop.



If we open this file in the default HTML browser, we'll its contents:


Testing Example - SautinSoft - Mozilla Firefox
Testing Example - SautinSof
file:///home/jorgen/Desktop/sample.html

Welcome to SautinSoft!

This is «HTML to RTF .Net» sample

Subtitle

This rich text document content serves as basis for trying out various rich text formatting.

Header 1

Bold *italic* underlined ~~strikethrough~~ N$_{\text{subscript}}$ N$^{\text{superscript}}$

.

Sub header 1.1

Fonts:

Calibri

Times new roman

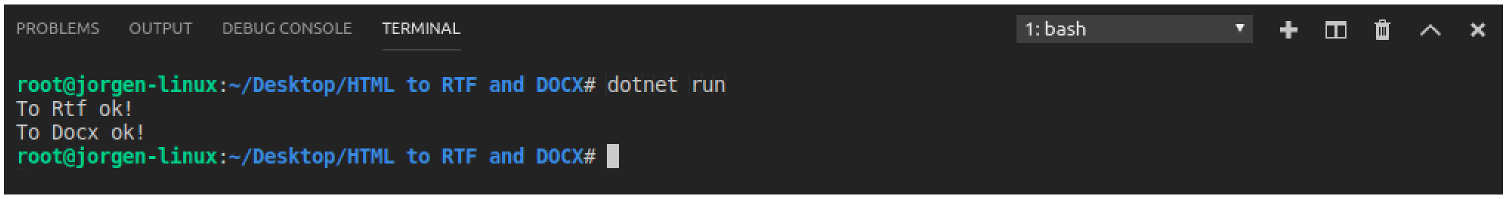
Arial

Courrier

Verdana

Tahoma

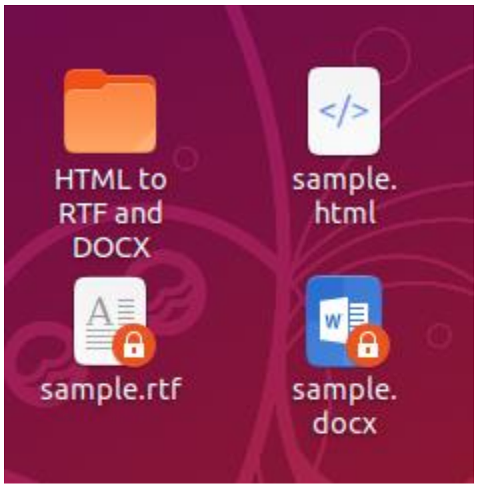
Launch our application and convert the "sample.html" into "sample.rtf" and "sample.docx", type the command: ***dotnet run***



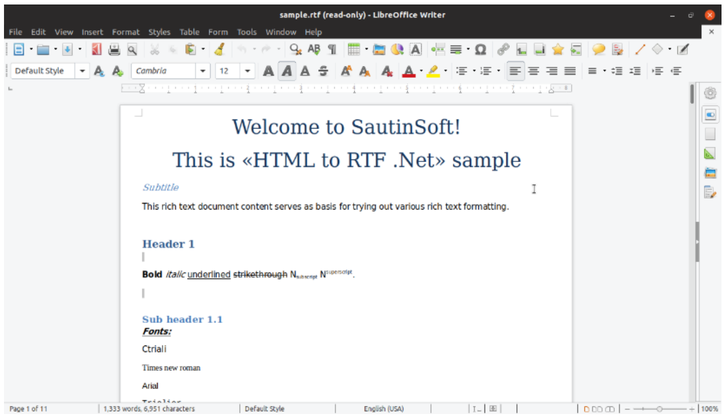
```
root@jorgen-linux:~/Desktop/HTML to RTF and DOCX# dotnet run
To Rtf ok!
To Docx ok!
root@jorgen-linux:~/Desktop/HTML to RTF and DOCX# 
```

If you see the messages "To Rtf ok!" and "To Docx ok!", everything is fine and we can check the results produced by the HTML to RTF .Net library.

The new files "sample.rtf" and "sample.docx" have to appear on the Desktop:



Open the results in LibreOffice:


sample.rtf (read-only) - LibreOffice Writer
File Edit View Insert Format Styles Table Form Tools Window Help
Default Style
Cambria
12

Welcome to SautinSoft!

This is «HTML to RTF .Net» sample

Subtitle

This rich text document content serves as basis for trying out various rich text formatting.

Header 1

Bold *italic* underlined ~~strikethrough~~ $N_{subscript}$ $N^{superscript}$.

Sub header 1.1

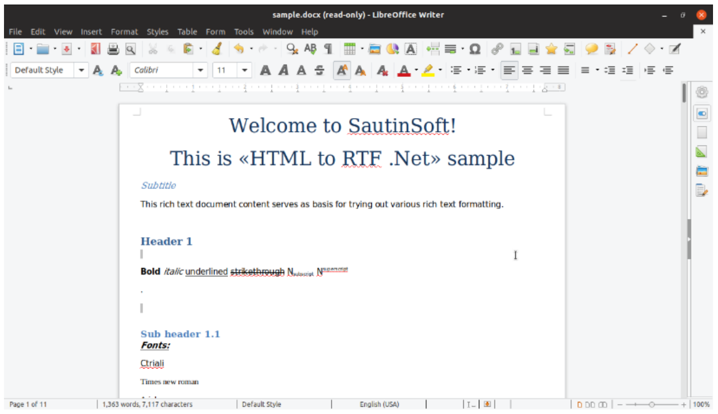
Fonts:

Ctriali

Times new roman

Arial

Page 1 of 11 | 1,333 words, 6,951 characters | Default Style | English (USA) | 100%


sample.docx (read-only) - LibreOffice Writer
File Edit View Insert Format Styles Table Form Tools Window Help
Default Style
Calibri
11

Welcome to SautinSoft!

This is «HTML to RTF .Net» sample

Subtitle

This rich text document content serves as basis for trying out various rich text formatting.

Header 1

Bold *italic* underlined ~~strikethrough~~ $N_{subscript}$ $N^{superscript}$

Sub header 1.1

Fonts:

Ctriali

Times new roman

Page 1 of 11 | 1,363 words, 7,117 characters | Default Style | English (USA) | 100%

Well done! You have created the “HTML to RTF/DOCX” application under Linux!

If you have any troubles or need extra code, or help, don’t hesitate to ask our SautinSoft Team at support@sautinsoft.com.