

HTML to RTF .Net

(Multi-platform .Net library)

[SautinSoft](#)

Linux development manual

Table of Contents

1. Preparing environment	2
1.1. Check the installed Fonts availability	3
2. Creating "Convert HTML to RTF/DOCX" app.....	5

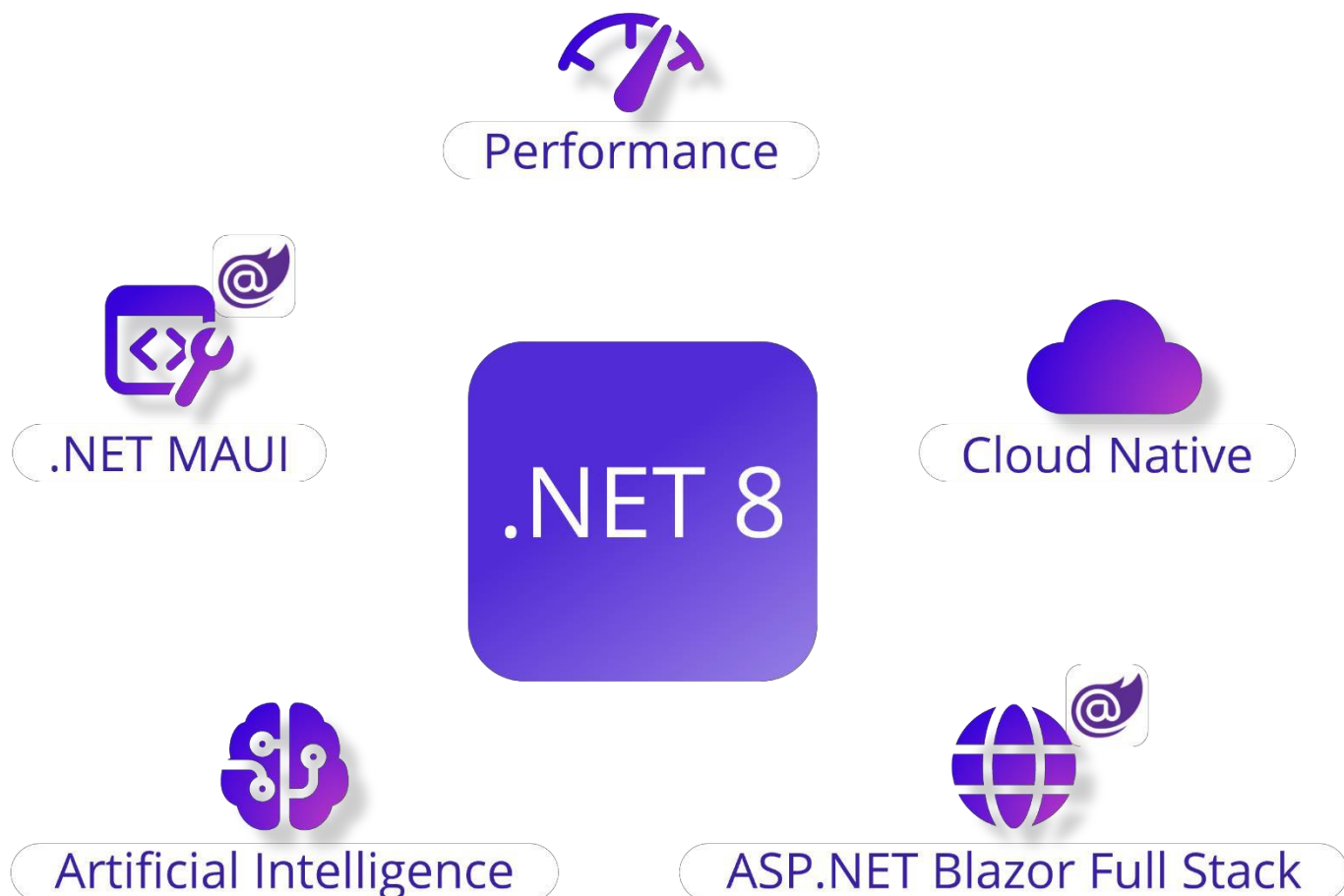
1. Preparing environment

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

We need to install .NET 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:

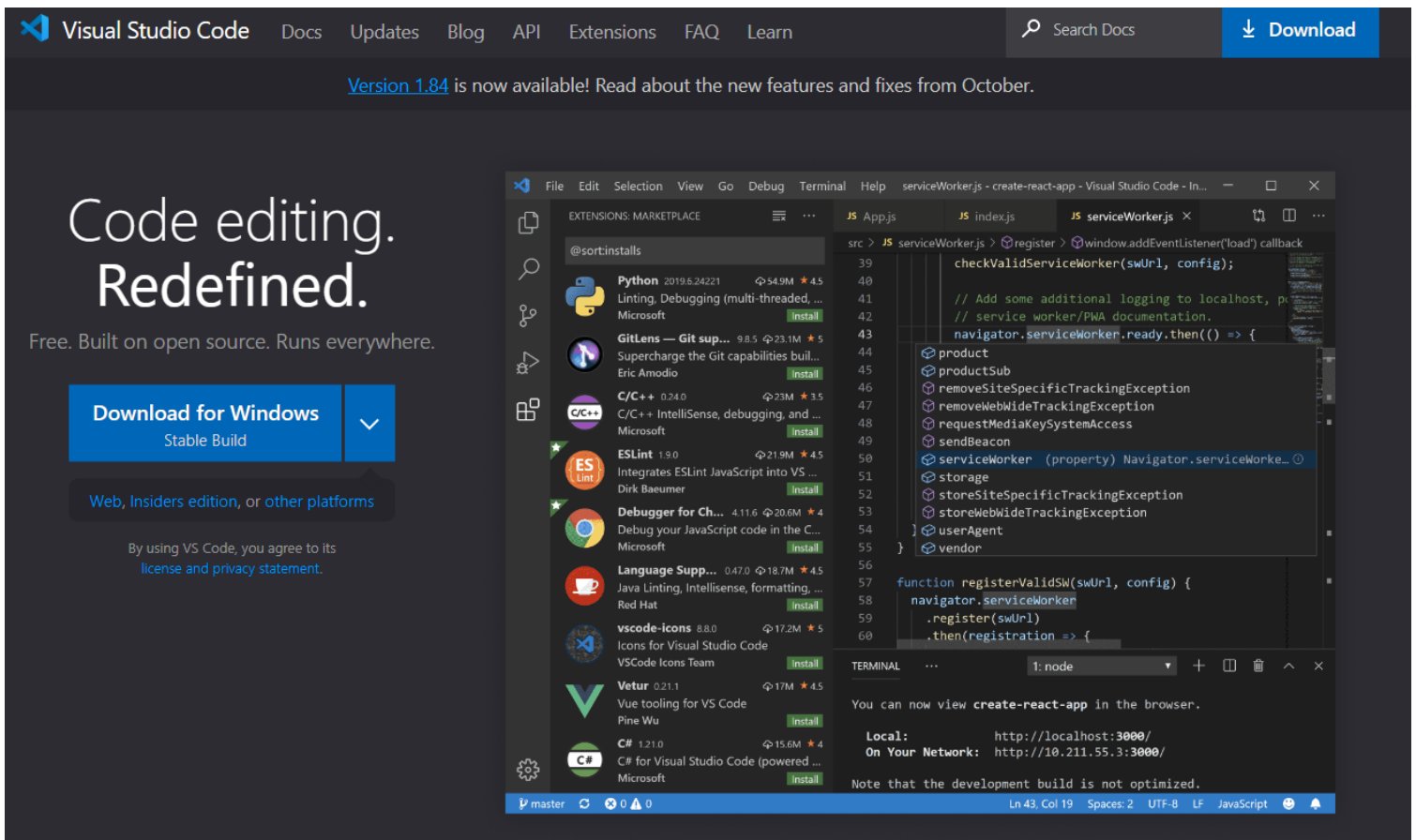
[Install .NET SDK for Linux.](#)



[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](#).



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, you may install "Microsoft TrueType core fonts" using terminal and command:

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

```
linuxconfig@linuxconfig-org: ~  
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 Verdanz.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"](#) .

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

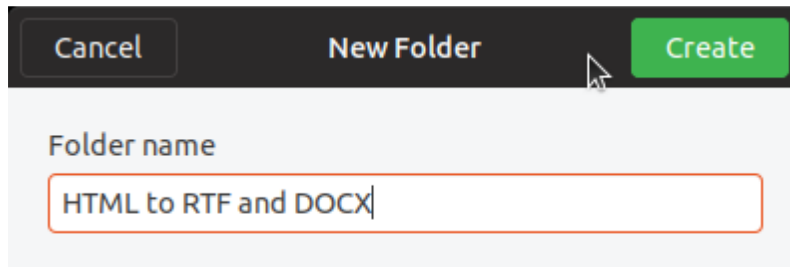
[Get Started with C# and Visual Studio Code](#)

Not only is possible to create .NET 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.

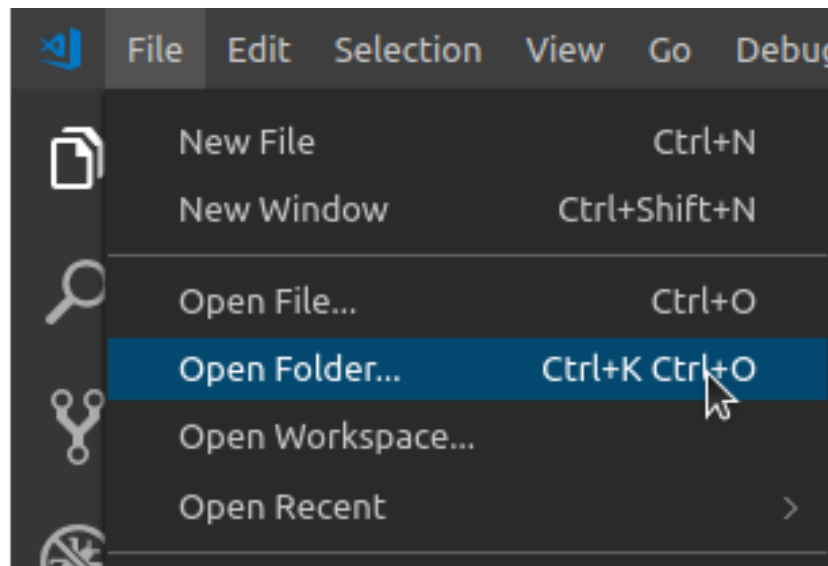
2. Creating “Convert HTML to RTF/DOCX” app

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

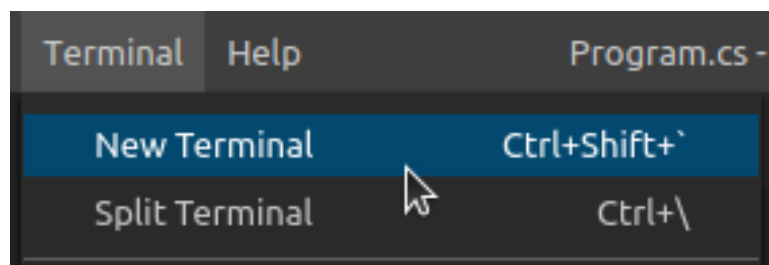
For example, let’s create the folder “**HTML to RTF and DOCX**” on Desktop (Right click-> New Folder):



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

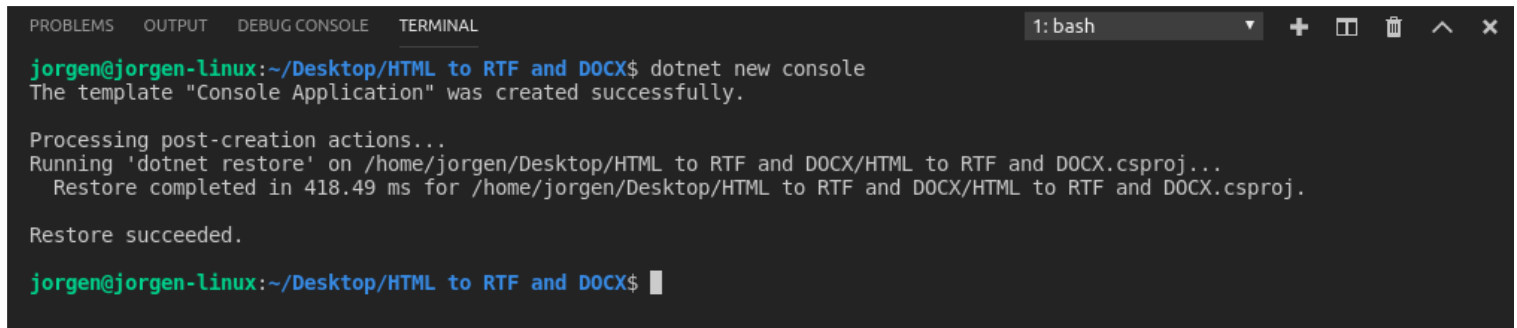


Now, open the integrated console – the Terminal: follow to the menu **Terminal -> New Terminal** (or press Ctrl+Shift+’):



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

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



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash
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.

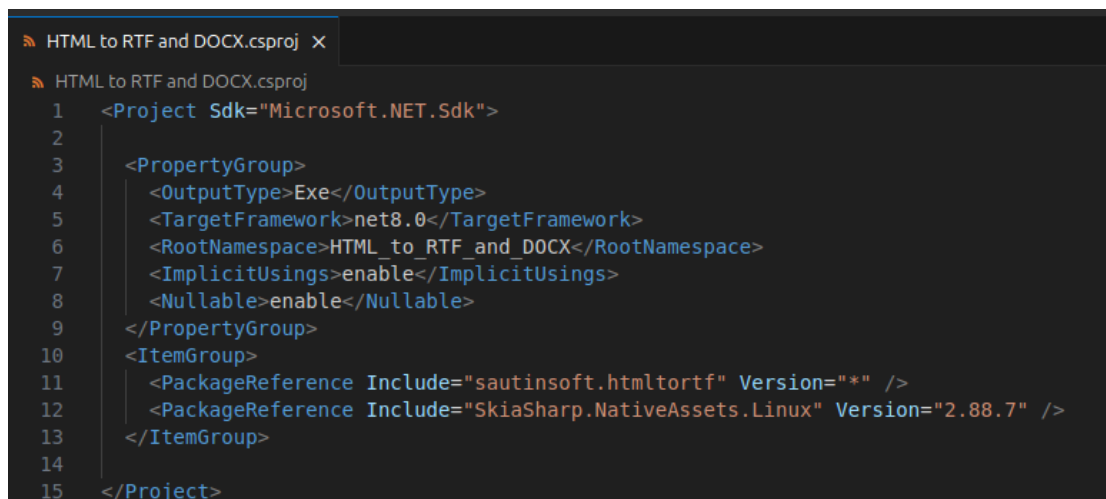
jorgen@jorgen-linux:~/Desktop/HTML to RTF and DOCX$
```

Now we are going to modify this simple application 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 or the library SautinSoft.HtmlToRtf.dll with additional references.

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

In the first case (NuGet):



```
HTML to RTF and DOCX.csproj x
HTML to RTF and DOCX.csproj
1 <Project Sdk="Microsoft.NET.Sdk">
2
3   <PropertyGroup>
4     <OutputType>Exe</OutputType>
5     <TargetFramework>net8.0</TargetFramework>
6     <RootNamespace>HTML_to_RTF_and_DOCX</RootNamespace>
7     <ImplicitUsings>enable</ImplicitUsings>
8     <Nullable>enable</Nullable>
9   </PropertyGroup>
10  <ItemGroup>
11    <PackageReference Include="sautinsoft.htmltortf" Version="*" />
12    <PackageReference Include="SkiaSharp.NativeAssets.Linux" Version="2.88.7" />
13  </ItemGroup>
14
15 </Project>
```

In the second case (SautinSoft.HtmlToRtf.dll):

```
HTML to RTF and DOCX.csproj X
HTML to RTF and DOCX.csproj
1  <Project Sdk="Microsoft.NET.Sdk">
2
3    <PropertyGroup>
4      <OutputType>Exe</OutputType>
5      <TargetFramework>net8.0</TargetFramework>
6      <RootNamespace>HTML_to_RTF_and_DOCX</RootNamespace>
7      <ImplicitUsings>enable</ImplicitUsings>
8      <Nullable>enable</Nullable>
9    </PropertyGroup>
10   <ItemGroup>
11     <PackageReference Include="System.Text.Encoding.CodePages" Version="4.5.0" />
12     <PackageReference Include="System.IO.Packaging" Version="4.5.0" />
13     <PackageReference Include="Svg.Skia" Version="1.0.0.18" />
14     <PackageReference Include="SkiaSharp" Version="2.88.7" />
15     <PackageReference Include="SkiaSharp.NativeAssets.Linux" Version="2.88.7" />
16     <Reference Include="SautinSoft.HtmlToRtf">
17       <HintPath>/YourPathToDll/SautinSoft.HtmlToRtf.dll</HintPath>
18     </Reference>
19   </ItemGroup>
20
21 </Project>
```

At once as we’ve added the package references, we have to save the “**HTML to RTF and DOCX.csproj**” and restore the added packages.

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

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash
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 all the references 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
File Edit Selection View Go Run Terminal Help
EXPLORER
HTML to RTF and DOCX
  bin
  obj
  HTML to RTF and DOCX.csproj
  HTML to RTF and DOCX.sln
  Program.cs
Program.cs X
1 namespace HTML_to_RTF_and_DOCX
2 {
3     0 references
4     class Program
5     {
6         0 references
7         static void Main(string[] args)
8         {
9             SautinSoft.HtmlToRtf h = new SautinSoft.HtmlToRtf();
10            string htmlFile = @"/home/jorgen/Desktop/sample.html";
11            string rtfFile = Path.ChangeExtension(htmlFile, ".rtf");
12            string docxFile = Path.ChangeExtension(htmlFile, ".docx");
13            if (h.Convert(htmlFile, rtfFile, new SautinSoft.HtmlToRtf.HtmlConvertOptions() { OutputFormat = SautinSoft.HtmlToRtf.OutputFormat.Rtf }))
14                System.Console.WriteLine("To Rtf ok!");
15            if (h.Convert(htmlFile, docxFile, new SautinSoft.HtmlToRtf.HtmlConvertOptions() { OutputFormat = SautinSoft.HtmlToRtf.OutputFormat.Docx }))
16                System.Console.WriteLine("To Docx ok!");
17        }
18    }
19 }
```

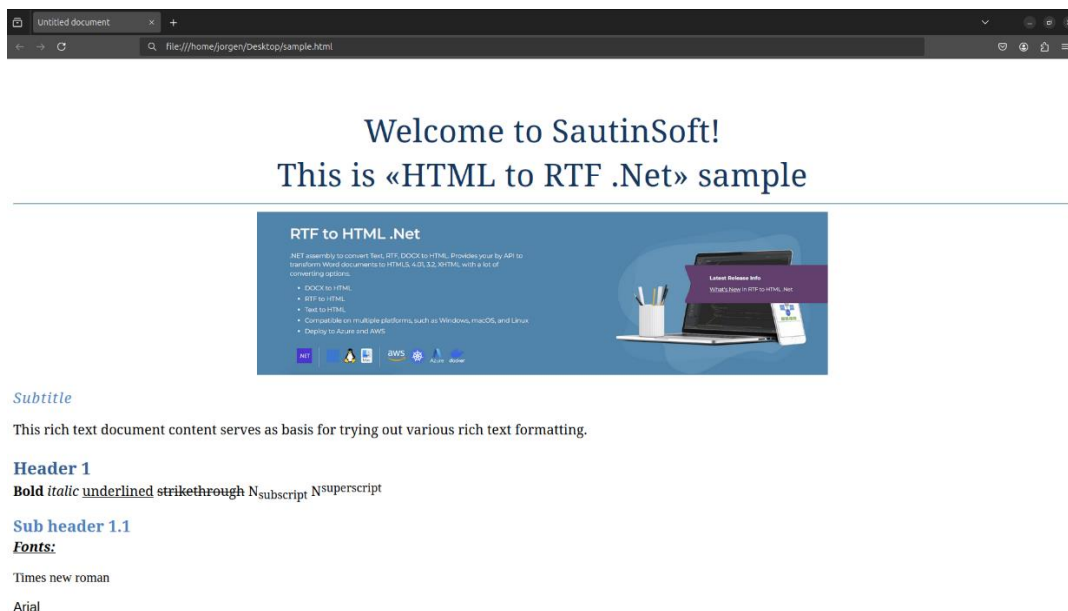
The code:

```
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.Convert(htmlFile, rtfFile, new SautinSoft.HtmlToRtf.HtmlConvertOptions() {
OutputFormat = SautinSoft.HtmlToRtf.OutputFormat.Rtf }))
                System.Console.WriteLine("To Rtf ok!");
            if (h.Convert(htmlFile, docxFile, new SautinSoft.HtmlToRtf.HtmlConvertOptions() {
OutputFormat = SautinSoft.HtmlToRtf.OutputFormat.Docx }))
                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 see its content:

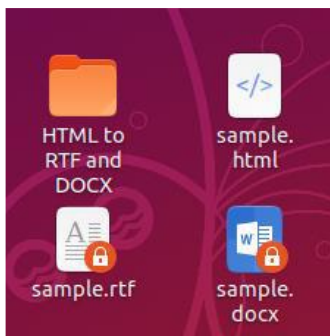


Launch our application and convert the "sample.html" into "sample.rtf" and "sample.docx", type the command: **dotnet run**

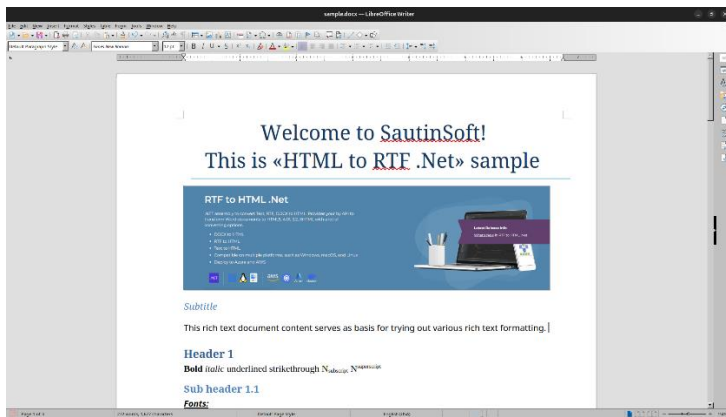
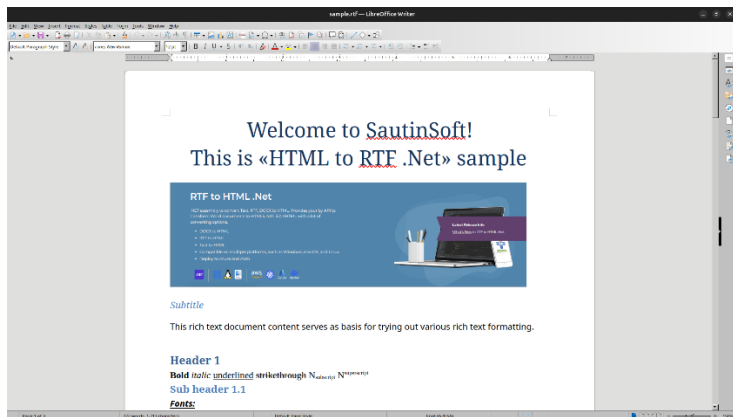
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
root@jorgen-linux:~/Desktop/HTML to RTF and DOCX# dotnet run
To Rtf ok!
To Docx ok!
root@jorgen-linux:~/Desktop/HTML to RTF and DOCX#
```

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:



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!