

HTML to RTF .Net

(Multi-platform .Net library)

[SautinSoft](#) 2002 - 2020

Linux development manual

Table of Contents

1. Preparing environment	2
2. Creating "Convert HTML to RTF/DOCX" app.....	4

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.](#)

Windows

Linux

macOS

.NET
Core

.NET Core 2.2

.NET Core is a cross-platform version of .NET for building websites, services, and console apps.

Build Apps ⓘ

Install .NET Core SDK

Run Apps ⓘ

Install .NET Core Runtime

[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.](#)

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

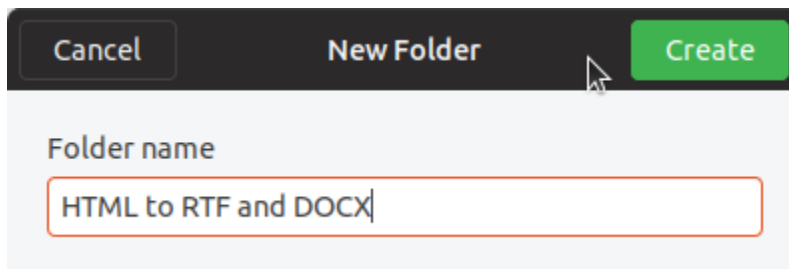
[Get Started with C# and Visual Studio Code](#)

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.

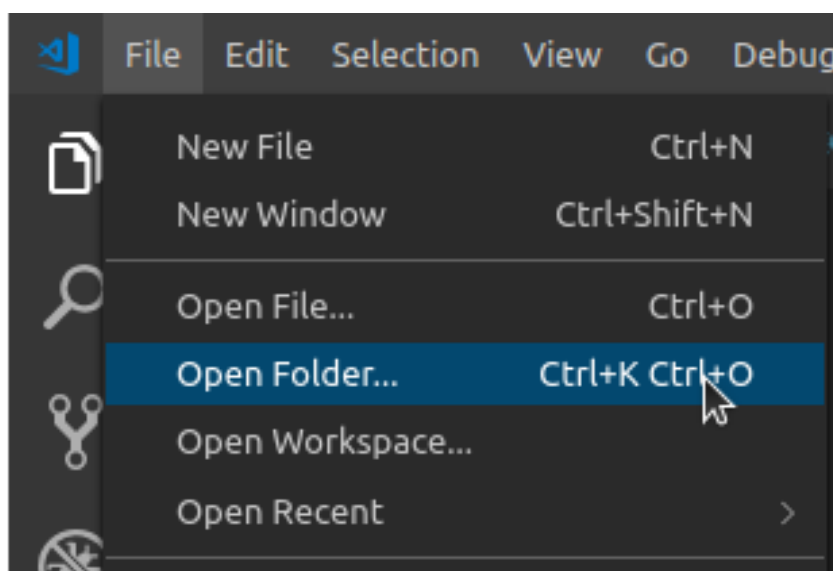
2. Creating “Convert HTML to RTF/DOCX” app

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

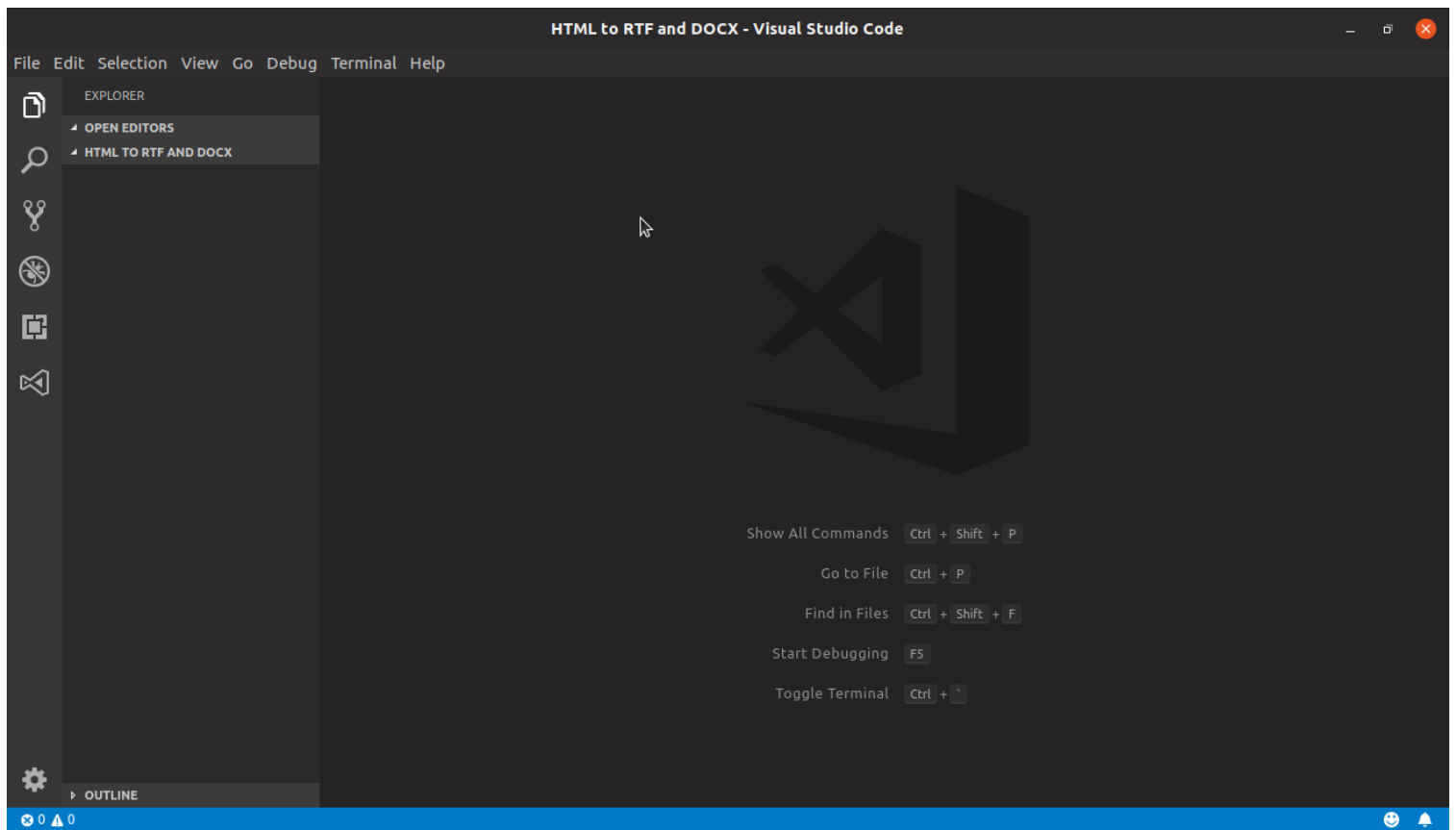
For example, let’s create the folder “**HTML to RTF and DOCX**” on the Desktop (Right click-> New Folder):



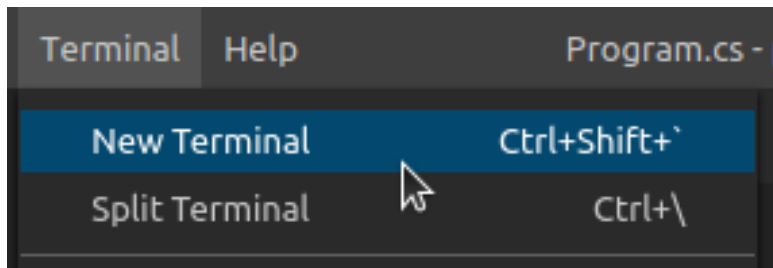
Open VS Code and click in the menu **File->Open Folder**. From the dialog, open the folder you’ve created previously:



Next you will see the similar screen:

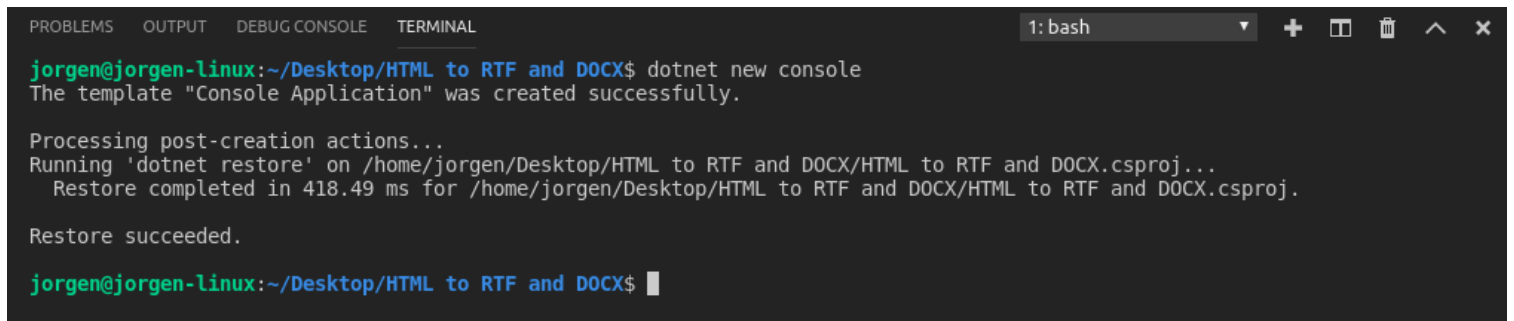


Now, open the integrated console – the Terminal: follow to the menu **Terminal** -> **New Terminal** (or press Ctrl+Shift+`):

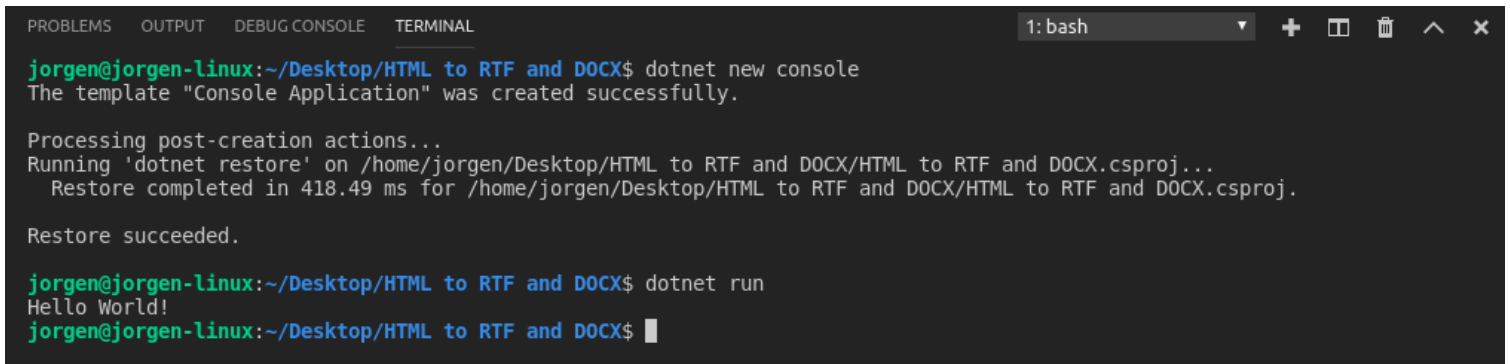


Create a new console application, using **dotnet** command.

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



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



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

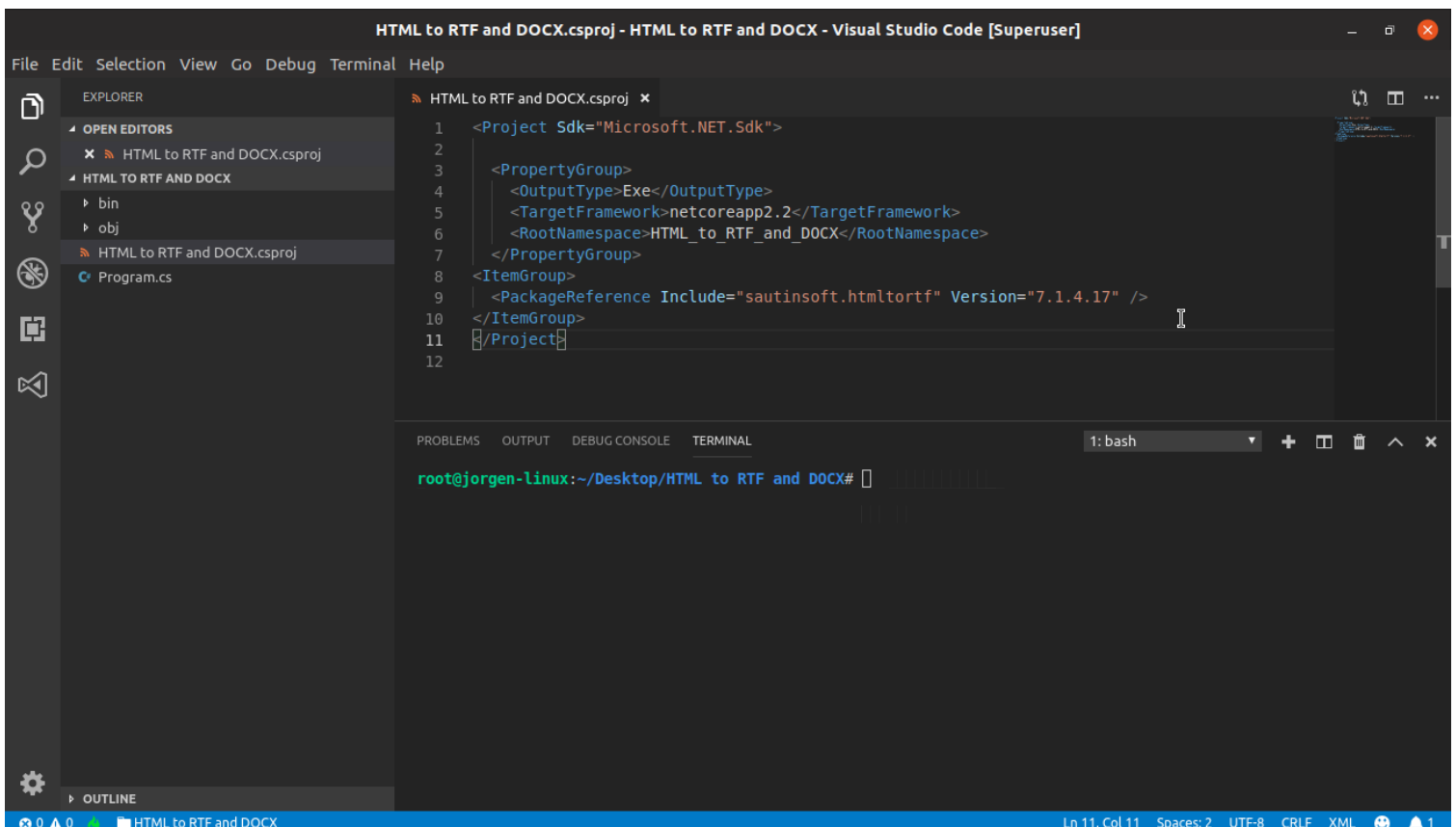
You can see the typical “Hello world!” message.

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:



```
HTML to RTF and DOCX.csproj - 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.csproj
HTML TO RTF AND DOCX
bin
obj
HTML to RTF and DOCX.csproj
Program.cs
HTML to RTF and DOCX.csproj
1 <Project Sdk="Microsoft.NET.Sdk">
2
3 <PropertyGroup>
4 <OutputType>Exe</OutputType>
5 <TargetFramework>netcoreapp2.2</TargetFramework>
6 <RootNamespace>HTML_to_RTF_and_DOCX</RootNamespace>
7 </PropertyGroup>
8 <ItemGroup>
9 <PackageReference Include="sautinsoft.htmltortf" Version="7.1.4.17" />
10 </ItemGroup>
11 </Project>
12
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash
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="7.1.4.17" />
</ItemGroup>
```

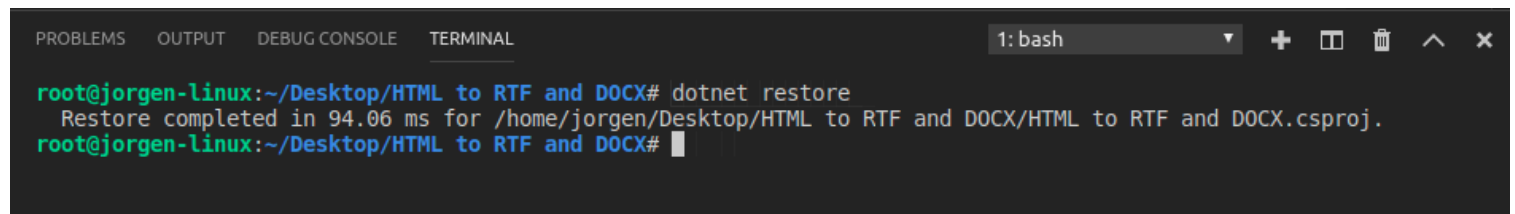
It's the reference to **sautinsoft.htmltortf** package from Nuget.

At the moment of writing this manual, the latest version of **sautinsoft.htmltortf** was 7.1.4.17. 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**

A screenshot of a terminal window with a dark background. The terminal title bar shows '1: bash' and standard window control icons. The terminal content shows a prompt 'root@jorgen-linux:~/Desktop/HTML to RTF and DOCX#' followed by the command 'dotnet restore'. The output is 'Restore completed in 94.06 ms for /home/jorgen/Desktop/HTML to RTF and DOCX/HTML to RTF and DOCX.csproj.' followed by another prompt 'root@jorgen-linux:~/Desktop/HTML to RTF and DOCX#' with a cursor.

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:

```
1 using System;
2 using System.IO;
3 using SautinSoft;
4
5 namespace HTML_to_RTF_and_DOCX
6 {
7     0 references
8     class Program
9     {
10        0 references
11        static void Main(string[] args)
12        {
13            SautinSoft.HtmlToRtf h = new SautinSoft.HtmlToRtf();
14            string htmlFile = @"~/home/jorgen/Desktop/sample.html";
15            string rtfFile = Path.ChangeExtension(htmlFile, ".rtf");
16            string docxFile = Path.ChangeExtension(htmlFile, ".docx");
17
18            if (h.OpenHtml(htmlFile))
19            {
20                if (h.ToRtf(rtfFile))
21                    System.Console.WriteLine("To Rtf ok!");
22                if (h.ToDocx(docxFile))
23                    System.Console.WriteLine("To Docx ok!");
24            }
25        }
26    }
27 }
```

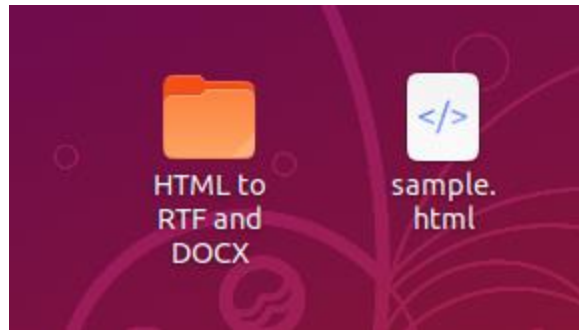
The new code:

```
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!");
            }
        }
    }
}
```

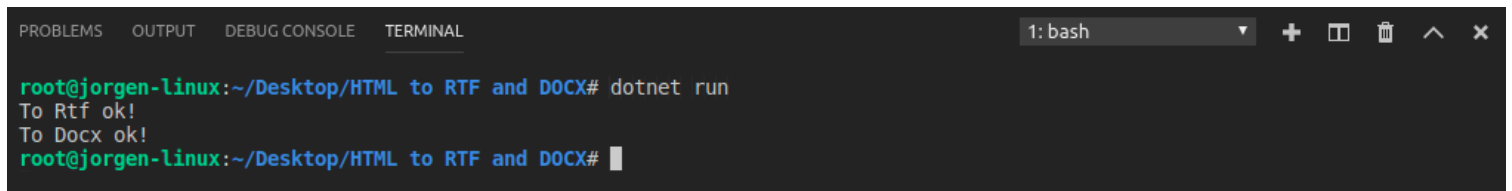
To make tests, we need an input HTML document. For our tests, let's place the HTML file with the name "sample.html" at the Desktop.



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

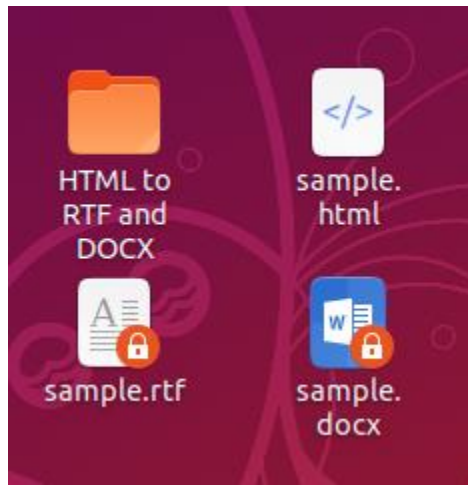


Launch our application and convert the "sample.html" into "sample.rtf" and "sample.docx", type the command: **dotnet run**

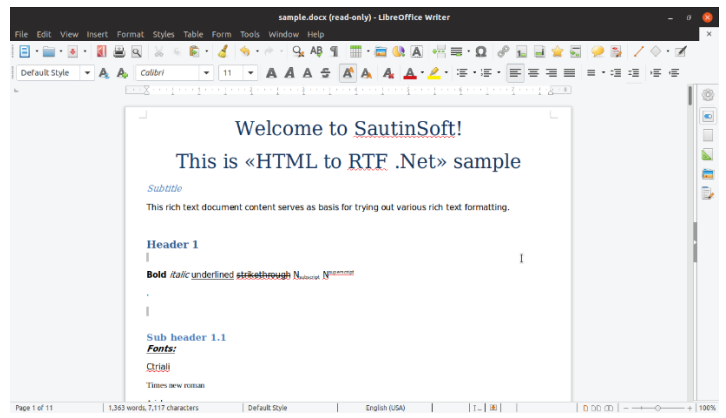
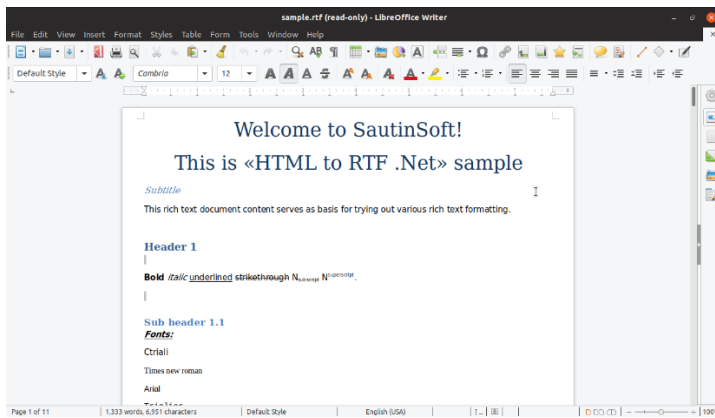


If you see the messages "To Rtf ok!" and "To Docx ok!", everything is fine and we can check the results produced by the [HTML to RTF .Net](#) library.

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.