

PDF Vision .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 PDF" application	5

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.](#)

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

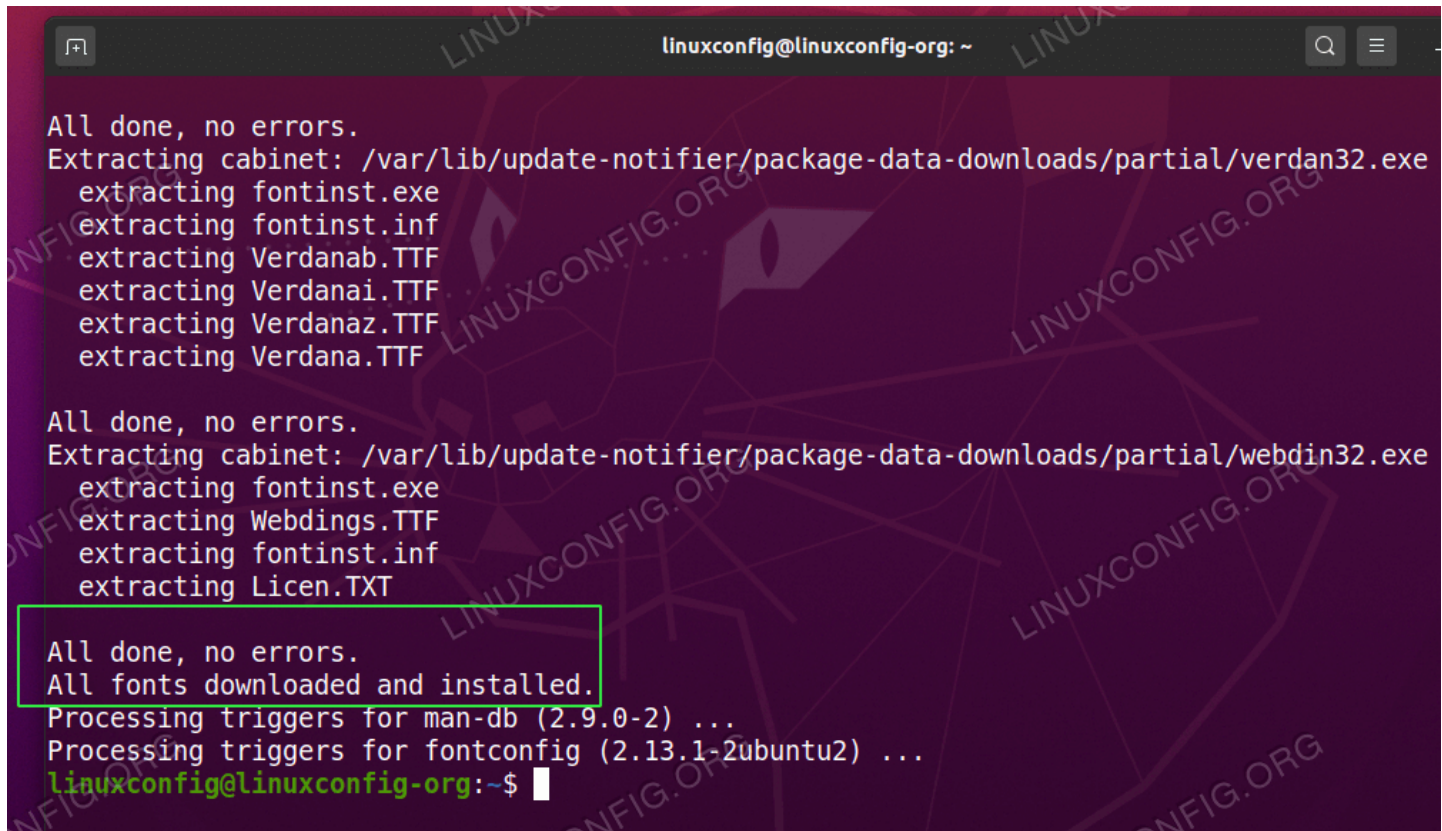
```
# /etc/rc.d/init.d/xfs restart
```

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
```



```
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 Verdana.TTF  
  extracting Verdana.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:

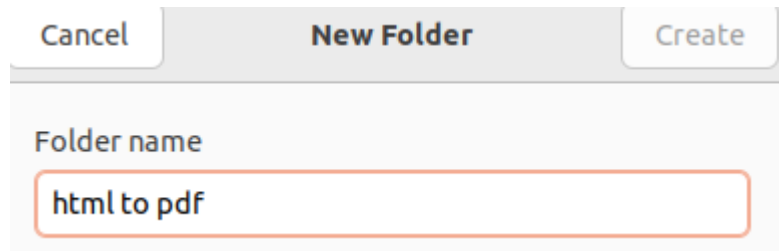
[Get Started with C# and Visual Studio Code](#)

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 2017.

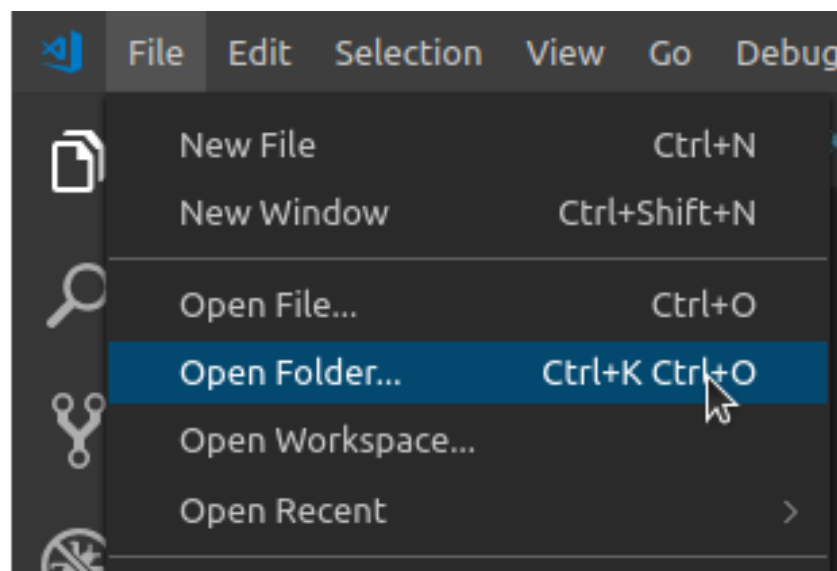
2. Creating “Convert HTML to PDF” application

Create a new folder in your Linux machine with the name *html to pdf*.

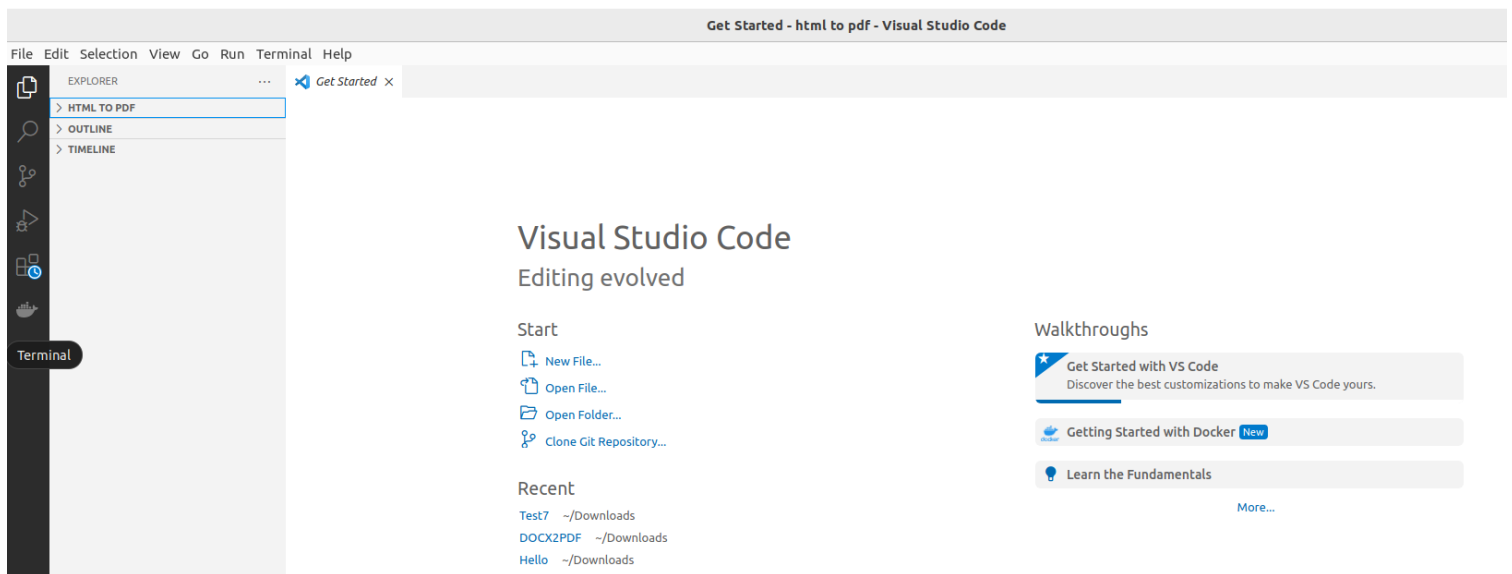
For example, let’s create the folder “*html to pdf*” on Desktop (Right click-> New Folder):



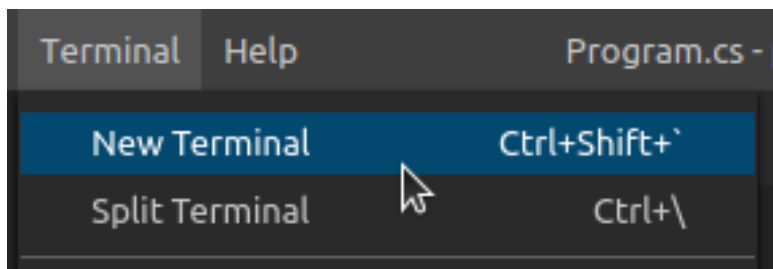
Open VS Code and click in the menu **File->Open Folder**. From the dialog, open the folder you’ve created previously:



Next you will see the similar screen:

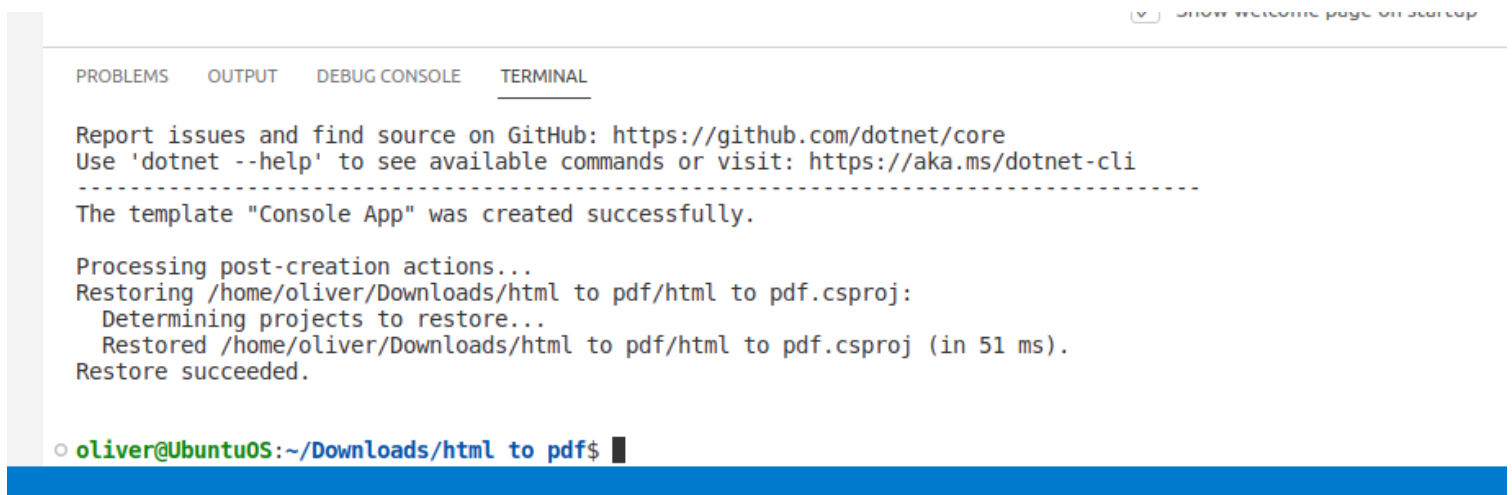


Now, open the integrated console – the Terminal: follow to the menu **Terminal -> New Terminal** (or press Ctrl+Shift+`):



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

 The template "Console App" was created successfully.

Processing post-creation actions...

Restoring /home/oliver/Downloads/html to pdf/html to pdf.csproj:

Determining projects to restore...

Restored /home/oliver/Downloads/html to pdf/html to pdf.csproj (in 51 ms).

Restore succeeded.

```

oliver@Ubuntu0S:~/Downloads/html to pdf$ dotnet run
Hello, World!
oliver@Ubuntu0S:~/Downloads/html to pdf$ █
  
```

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

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

We'll transform it into an application that will convert a html file to a pdf file.

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

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

The screenshot displays the NuGet package manager in VS Code. The search filter is set to 'sautinsoft'. The package 'SautinSoft.PdfVision' is selected and highlighted in blue. The detailed view on the right shows the following information:

- Version:** 6.2.11.16
- Description:** PDF Vision .Net provides API to convert HTML, ASPX, Images (Tiff, Jpeg, Png) to PDF Format. Furthermore, it gives you methods to split and merge PDF.
- Open in:** nuget.org
- Version:** 6.2.11.16
- Date Published:** Wednesday, November 16, 2022 (11/16/2022)
- Author(s):** SautinSoft
- License:** [View license](#)
- Project Url:** <https://www.sautinsoft.com/products/pdf-vision/index.php>
- Tags:** sautinsoft, pdfvision, sautinsoft.pdfvision, c#, .net, html, aspx, images, tiff, jpeg, convert, pdf, split, merge

The dependencies section lists the following packages:

- net7.0
 - System.Drawing.Common [4.7.0,)
 - Microsoft.Extensions.Logging [2.0.2,)
 - Newtonsoft.Json [10.0.3,)
- net6.0
 - System.Drawing.Common [4.7.0,)
 - Microsoft.Extensions.Logging [2.0.2,)
 - Newtonsoft.Json [10.0.3,)
- net5.0
 - System.Drawing.Common [4.7.0,)
 - Microsoft.Extensions.Logging [2.0.2,)
 - Newtonsoft.Json [10.0.3,)
- .NETStandard2.0
 - System.Drawing.Common [4.7.0,)
 - Microsoft.Extensions.Logging [2.0.2,)
 - Newtonsoft.Json [10.0.3,)
- .NETCoreApp2.0
 - System.Drawing.Common [4.7.0,)
 - Microsoft.Extensions.Logging [2.0.2,)

The terminal at the bottom shows the same output as the previous image:

```

-----
The template "Console App" was created successfully.

Processing post-creation actions...
Restoring /home/oliver/Downloads/html to pdf/html to pdf.csproj:
  Determining projects to restore...
  Restored /home/oliver/Downloads/html to pdf/html to pdf.csproj (in 51 ms).
Restore succeeded.

oliver@Ubuntu0S:~/Downloads/html to pdf$ dotnet run
Hello, World!
oliver@Ubuntu0S:~/Downloads/html to pdf$ █
  
```

Add these lines into the file "**html to pdf.csproj**":

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



```
html to pdf.csproj  
1 <Project Sdk="Microsoft.NET.Sdk">  
2  
3 <PropertyGroup>  
4 <OutputType>Exe</OutputType>  
5 <TargetFramework>net7.0</TargetFramework>  
6 <RootNamespace>html_to_pdf</RootNamespace>  
7 <ImplicitUsings>enable</ImplicitUsings>  
8 <Nullable>enable</Nullable>  
9 </PropertyGroup>  
10  
11 <ItemGroup>  
12 <PackageReference Include="SautinSoft.PdfVision" Version="6.2.11.16" />  
13 </ItemGroup>  
14  
15 </Project>  
16
```

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

At the moment of writing this manual, the latest version of **sautinsoft.pdfvision** was 6.X.

But you may specify the latest version, to know what is the latest, follow:

<https://www.nuget.org/packages/sautinsoft.pdfvision>

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

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


```
Processing post-creation actions...
Restoring /home/oliver/Downloads/html to pdf/html to pdf.csproj:
  Determining projects to restore...
  Restored /home/oliver/Downloads/html to pdf/html to pdf.csproj (in 51 ms).
Restore succeeded.
```

```
● oliver@UbuntuOS:~/Downloads/html to pdf$ dotnet run
Hello, World!
● oliver@UbuntuOS:~/Downloads/html to pdf$ dotnet restore
  Determining projects to restore...
  All projects are up-to-date for restore.
○ oliver@UbuntuOS:~/Downloads/html to pdf$ █
```

Very important!

There are a lot of Linux varieties: Suse, Fedora, Debian, Ubuntu, etc.

In addition, there are cloud platforms: AWS Lambda, Google Cloud, Azure, Apex, etc.

Because our dll works with Graphics and using GDI+ API, you need to check, that your system has [System.Drawing.Common](#) is the graphics library which ships as part of .NET Core. On macOS and Linux, it uses [libgdiplus](#) as its back-end.

There are existing ways to install libgdiplus on macOS and Linux. On macOS, you can use the [mono-libgdiplus](#) Homebrew package; on Ubuntu Linux, you can install the [libgdiplus-dev](#) package.

🔗 Using libgdiplus on Ubuntu Linux

You can install libgdiplus on Ubuntu Linux using the Quamotion PPA. Follow these steps:

```
sudo add-apt-repository ppa:quamotion/ppa
sudo apt-get update
sudo apt-get install -y libgdiplus
```

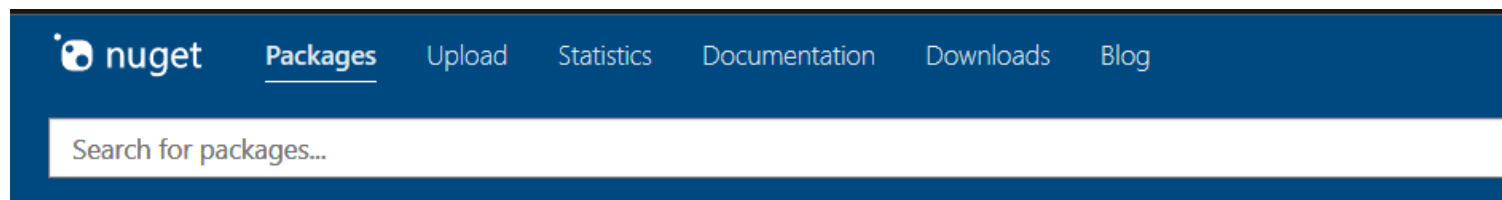
Using libgdiplus on macOS

On macOS, add a reference to the [runtime.osx.10.10-x64.CoreCompat.System.Drawing](#) package:

```
dotnet add package runtime.osx.10.10-x64.CoreCompat.System.Drawing
```

If you have installed LibGdiPlus' dll and it still doesn't work. Please add (update) this string in your project:

```
<PackageReference Include="runtime.osx.10.10-x64.CoreCompat.System.Drawing"
Version="5.23.62"/>
```



runtime.osx.10.10-x64.CoreCompat.System.Drawing 5.8.64

Allows you to use System.Drawing on OS X

Package Manager

.NET CLI

PackageReference

Paket CLI

```
PM> Install-Package runtime.osx.10.10-x64.CoreCompat.System.Drawing -Version 5.8.64
```



Good, now our application has the reference to **sautinsoft.pdfvision** package and we can write the code to convert html to pdf and other formats.

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

```
Program help
NuGet Gallery Program.cs
Program.cs > {} html_to_pdf > html_to_pdf.Program > ConvertHtmlFileToPdfFile()
1 using System;
2 using System.IO;
3 using SautinSoft.PdfVision;
4
5 namespace html_to_pdf
6 {
7     0 references
8     class Program
9     {
10         0 references
11         static void Main(string[] args)
12         {
13             ConvertHtmlFileToPdfFile();
14         }
15         1 reference
16         public static void ConvertHtmlFileToPdfFile()
17         {
18             string inpFile = Path.GetFullPath(@"..\..\Sample.html");
19             string outFile = new FileInfo("Result.pdf").FullName;
20             // Local chromium will be downloaded into this directory.
21             // This takes time only at the first startup.
22             string chromiumDirectory = new DirectoryInfo(@"..\..\..\..\..\Local Chromium\").FullName;
23
24             PdfVision v = new PdfVision();
25             // v.Serial = "123456789";
26             HtmlToPdfOptions options = new HtmlToPdfOptions()
27             {
28                 ChromiumBaseDirectory = chromiumDirectory,
29                 PageSetup = new PageSetup()
30                 {
31                     PaperType = PaperType.Letter,
32                     Orientation = Orientation.Portrait,
33                     PageMargins = new PageMargins()
34                     {
35                         Left = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter),
36                         Top = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter),
37                         Right = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter),
38                         Bottom = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter)
39                     }
40                 },
41                 PrintBackground = true,
42                 Scale = 1
43             };
44
45             try
46             {
47                 v.ConvertHtmlToPdf(inpFile, outFile, options);
48                 // Open the result for demonstration purposes.
49                 System.Diagnostics.Process.Start(new System.Diagnostics.ProcessStartInfo(outFile) { UseShellExecute = true });
50             }
51             catch { }
52         }
53     }
54 }
```

The code:

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

namespace Sample
{
    class Program
    {
```

```

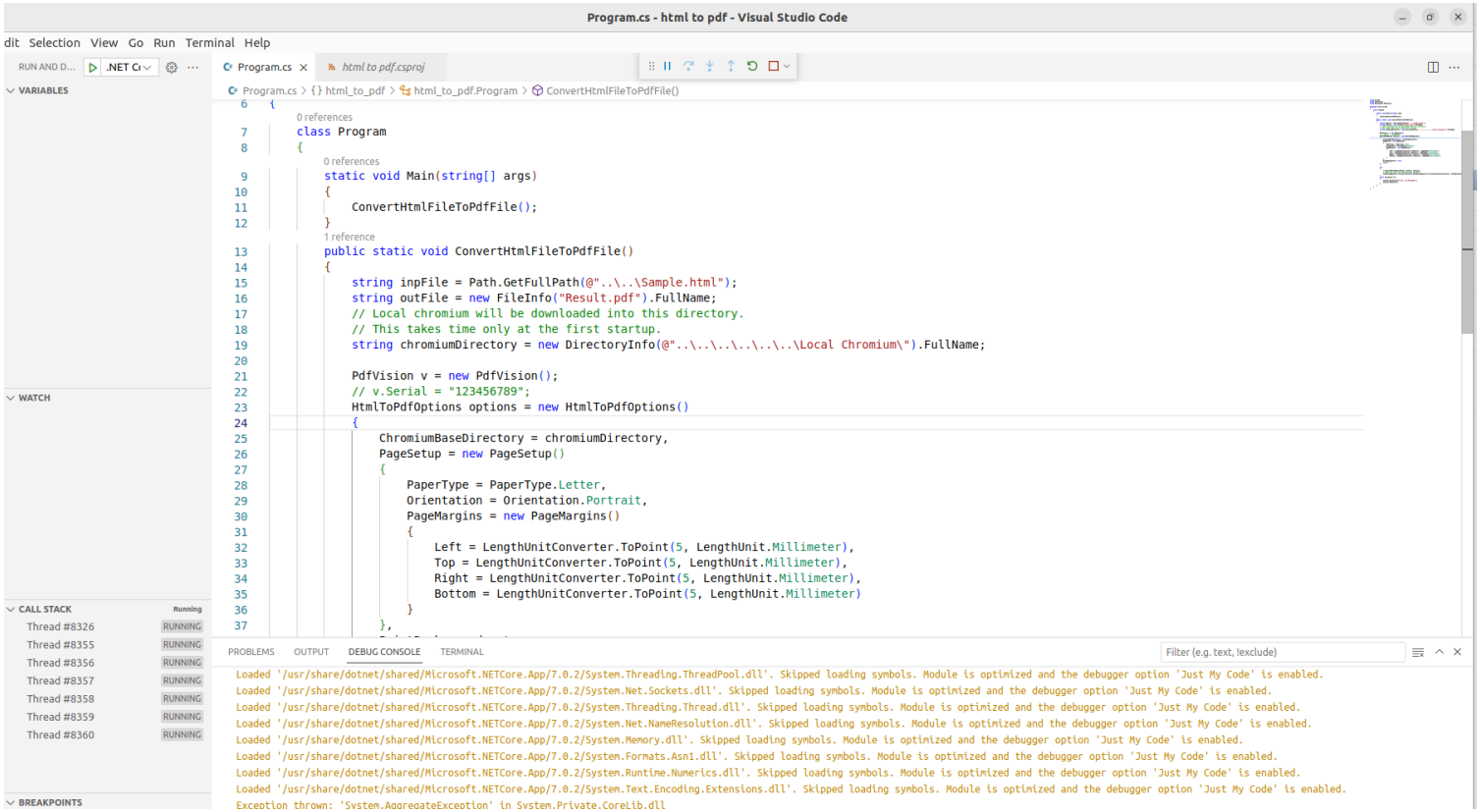
static void Main(string[] args)
{
    ConvertHtmlFileToPdfFile();
}
public static void ConvertHtmlFileToPdfFile()
{
    string inpFile = Path.GetFullPath(@"..\..\Sample.html");
    string outFile = new FileInfo("Result.pdf").FullName;
    // Local chromium will be downloaded into this directory.
    // This takes time only at the first startup.
    string chromiumDirectory = new DirectoryInfo(@"..\..\..\..\..\..\Local
Chromium\").FullName;

    PdfVision v = new PdfVision();
    // v.Serial = "123456789";
    HtmlToPdfOptions options = new HtmlToPdfOptions()
    {
        ChromiumBaseDirectory = chromiumDirectory,
        PageSetup = new PageSetup()
        {
            PaperType = PaperType.Letter,
            Orientation = Orientation.Portrait,
            PageMargins = new PageMargins()
            {
                Left = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter),
                Top = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter),
                Right = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter),
                Bottom = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter)
            }
        },
        PrintBackground = true,
        Scale = 1
    };

    try
    {
        v.ConvertHtmlToPdf(inpFile, outFile, options);
        // Open the result for demonstration purposes.
        System.Diagnostics.Process.Start(new
System.Diagnostics.ProcessStartInfo(outFile) { UseShellExecute = true });
    }
    catch (Exception ex)
    {

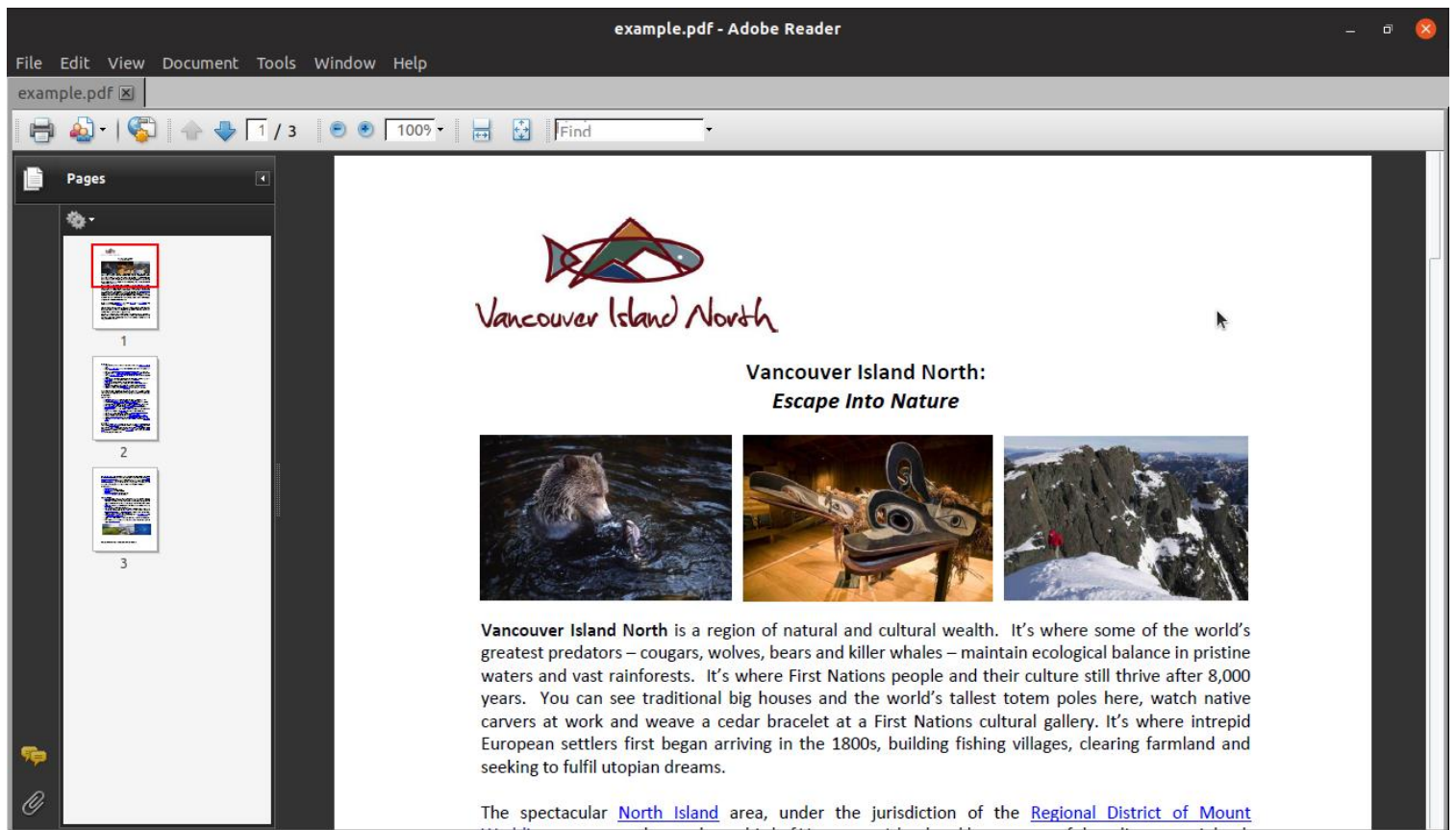
```

```
        Console.WriteLine($"Error: {ex.Message}");
        Console.ReadLine();
    }
}
}
```



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

If we open this file in the default PDF Viewer, we'll its contents:



Launch our application and convert the "sample.html" into "result.pdf", type the command:

dotnet run

Well done! You have created the "HTML to PDF" 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!