PDF .Net

(Multi-platform .Net library)

SautinSoft

Linux development manual

Table of Contents

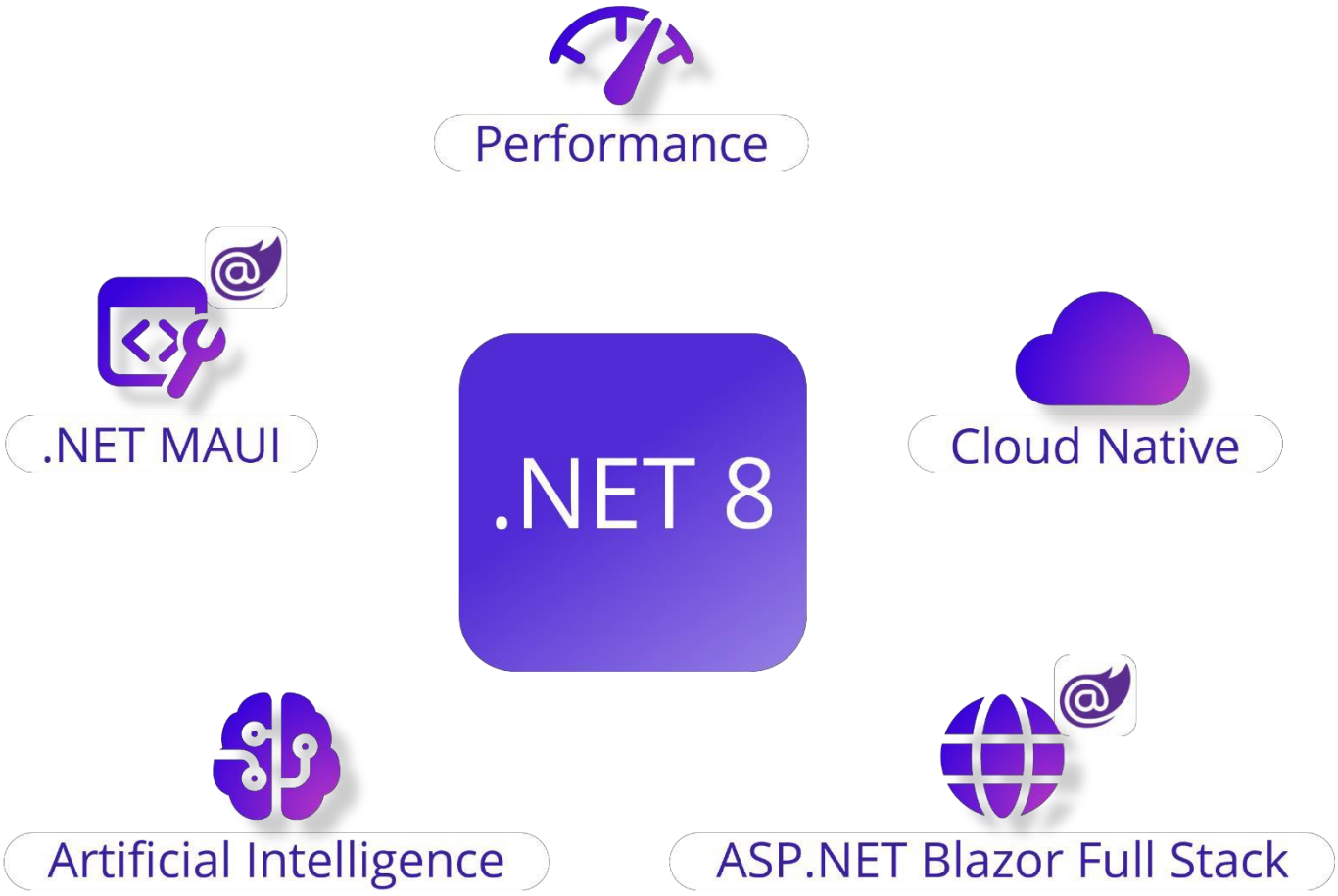
1. Preparing environment

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:

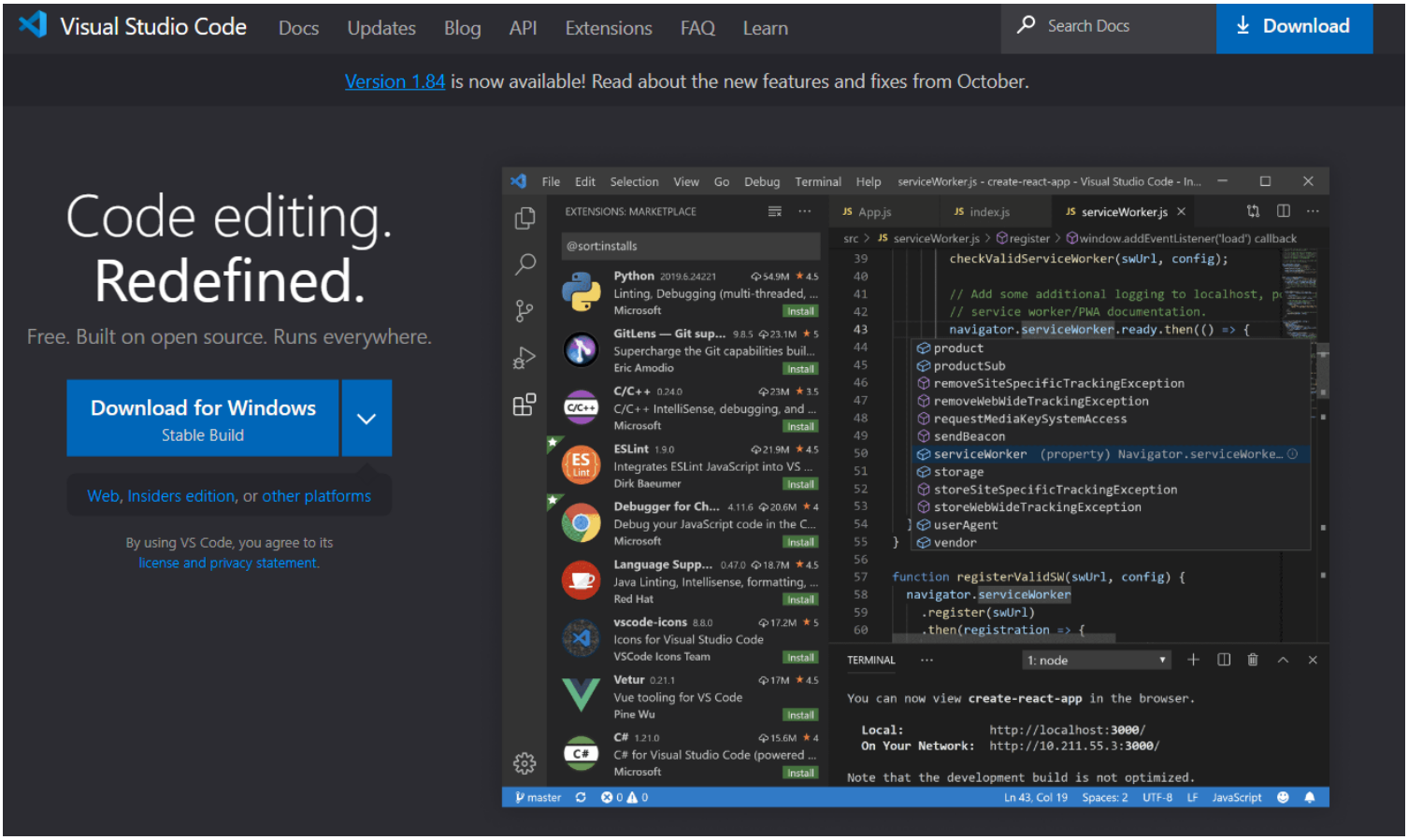
Install .NET SDK for Linux.



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

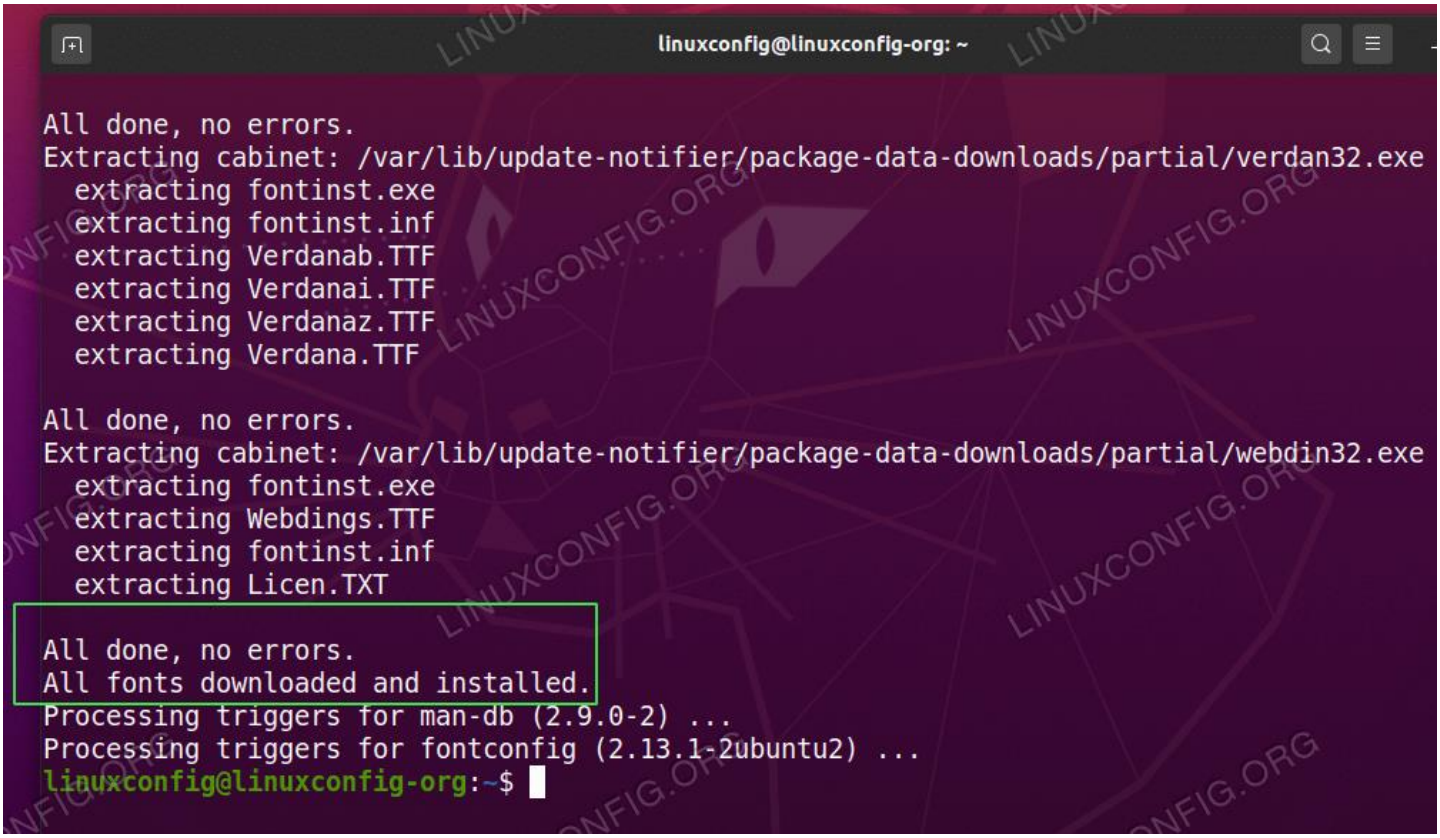
Check that the directory with fonts "/usr/share/fonts/truetype" is

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
```



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:

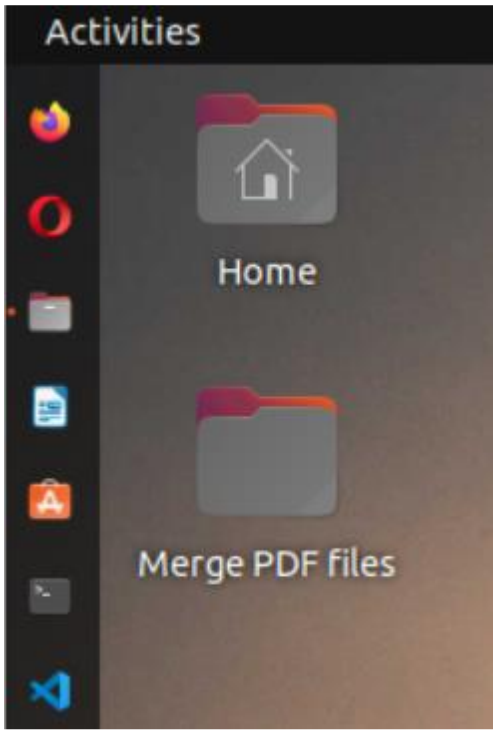
[Get Started with C# and Visual Studio Code]

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.

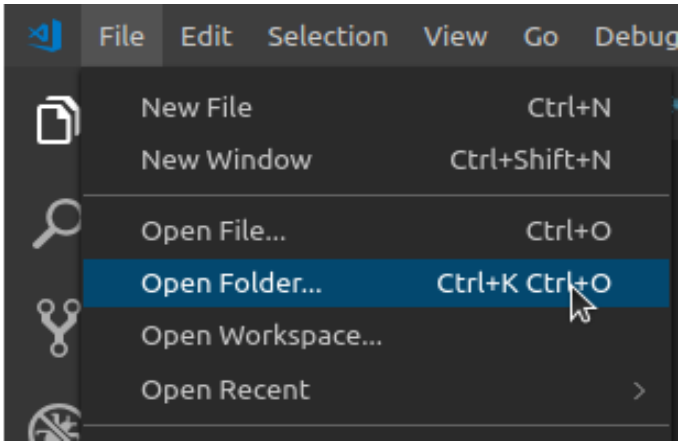
2. Creating “Merge PDF files” application

Create a new folder in your Linux machine with the name ***Merge PDF Files.***

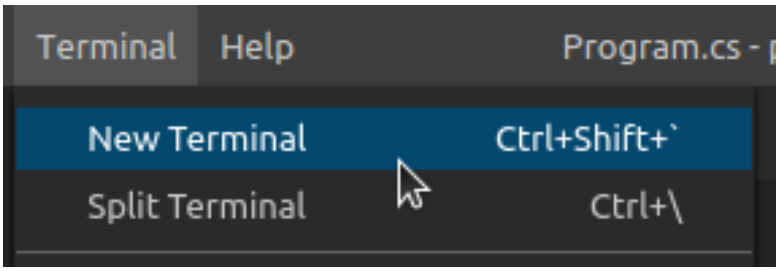
For example, let’s create the folder “***Merge PDF Files***” on Desktop (Right click-> New Folder):



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  PORTS  POLYGLOT NOTEBOOK

developer@defaultpc:~/Desktop/Merge PDF files$ dotnet new console
The template "Console App" was created successfully.

Processing post-creation actions...
Restoring /home/developer/Desktop/Merge PDF files/Merge PDF files.csproj:
  Determining projects to restore...
  Restored /home/developer/Desktop/Merge PDF files/Merge PDF files.csproj (in 69 ms).
Restore succeeded.


developer@defaultpc:~/Desktop/Merge PDF files$
```

Now we are going to convert 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.pdf*** assembly using Nuget or the library SautinSoft.Pdf.dll with additional references.

In order to do it, follow to the ***Explorer*** and open project file "***Merge PDF Files.csproj"*** :

In the first case (NuGet):

Merge PDF files.csproj

```
1  <Project Sdk="Microsoft.NET.Sdk">
2
3    <PropertyGroup>
4      <OutputType>Exe</OutputType>
5      <TargetFramework>net8.0</TargetFramework>
6      <RootNamespace>Merge_PDF_files</RootNamespace>
7      <ImplicitUsings>enable</ImplicitUsings>
8      <Nullable>enable</Nullable>
9    </PropertyGroup>
10
11   <ItemGroup>
12     <PackageReference Include="sautinsoft.pdf" Version="*" />
13     <PackageReference Include="HarfBuzzSharp.NativeAssets.Linux" Version="*" />
14   </ItemGroup>
15
16 </Project>
```

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

```
Merge PDF files.csproj
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net8.0</TargetFramework>
    <RootNamespace>Merge_PDF_files</RootNamespace>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Pkcs11Interop" Version="5.1.2" />
    <PackageReference Include="Portable.BouncyCastle" Version="1.9.0" />
    <PackageReference Include="SkiaSharp" Version="2.88.7" />
    <PackageReference Include="SkiaSharp.HarfBuzz" Version="2.88.7" />
    <PackageReference Include="HarfBuzzSharp.NativeAssets.Linux" Version="2.88.7" />
    <PackageReference Include="Svg.Skia" Version="1.0.0.18" />
    <PackageReference Include="System.IO.Packaging" Version="4.5.0" />
    <PackageReference Include="System.Net.Http" Version="4.3.4" />
    <PackageReference Include="System.Text.Encoding.CodePages" Version="4.5.0" />
    <PackageReference Include="System.Xml.XPath.XmlDocument" Version="4.7.0" />
    <PackageReference Include="Tesseract" Version="5.2.0" />
    <PackageReference Include="Tesseract.Data.English" Version="4.0.0" />
    <Reference Include="SautinSoft.Pdf">
      <HintPath>/YourPathToDll/SautinSoft.Pdf.dll</HintPath>
    </Reference>
  </ItemGroup>

</Project>
```

At once as we've added the package references, we have to save the "***Merge PDF Files.csproj***" and restore the added packages.

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

```
developer@defaultpc:~/Desktop/Merge PDF files$ dotnet restore
  Determining projects to restore...
  All projects are up-to-date for restore.
developer@defaultpc:~/Desktop/Merge PDF files$
```

Good, now our application has all the references and we can write the code to convert Merge PDF Files formats.

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

```
Program.cs
using SautinSoft.Pdf;

namespace Sample
{
    0 references
    class Sample
    {
        0 references
        static void Main(string[] args)
        {
            //SautinSoft.Pdf.PdfDocument.SetLicense("1234567890");
            string[] inpFiles = new string[] {
                Path.GetFullPath(@"/home/alex/Desktop/MergeFile01.pdf"),
                Path.GetFullPath(@"/home/alex/Desktop/MergeFile02.pdf"),
                Path.GetFullPath(@"/home/alex/Desktop/MergeFile03.pdf")};
            string outFile = @"/home/alex/Desktop/Merged.pdf";
            using (var pdf = new PdfDocument())
            {
                // Merge multiple PDF files into single PDF.
                foreach (var inpFile in inpFiles)
                {
                    using (var source = PdfDocument.Load(inpFile))
                    {
                        foreach (var page in source.Pages)
                            pdf.Pages.AddClone(page);
                    }
                }
                pdf.Save(outFile);
            }
            // Show the result PDF document.
            System.Diagnostics.Process.Start(new System.Diagnostics.ProcessStartInfo(outFile) { UseShellExecute = true });
        }
    }
}
```

The code:

```
using SautinSoft.Pdf;

namespace Sample
{
    class Sample
    {
        static void Main(string[] args)
        {
            // SautinSoft.Pdf.PdfDocument.SetLicense("1234567890");
            string[] inpFiles = new string[] {
                Path.GetFullPath(@"/home/alex/Desktop/MergeFile01.pdf"),
                Path.GetFullPath(@"/home/alex/Desktop/MergeFile02.pdf"),
                Path.GetFullPath(@"/home/alex/Desktop/MergeFile03.pdf")};
            string outFile = @"/home/alex/Desktop/Merged.pdf";
            using (var pdf = new PdfDocument())
            {
                // Merge multiple PDF files into single PDF.
                foreach (var inpFile in inpFiles)
                {
                    using (var source = PdfDocument.Load(inpFile))
                    {
                        foreach (var page in source.Pages)
                            pdf.Pages.AddClone(page);
                    }
                }
                pdf.Save(outFile);
            }
            // Show the result PDF document.
            System.Diagnostics.Process.Start(new System.Diagnostics.ProcessStartInfo(outFile)
            { UseShellExecute = true });
        }
    }
}
```

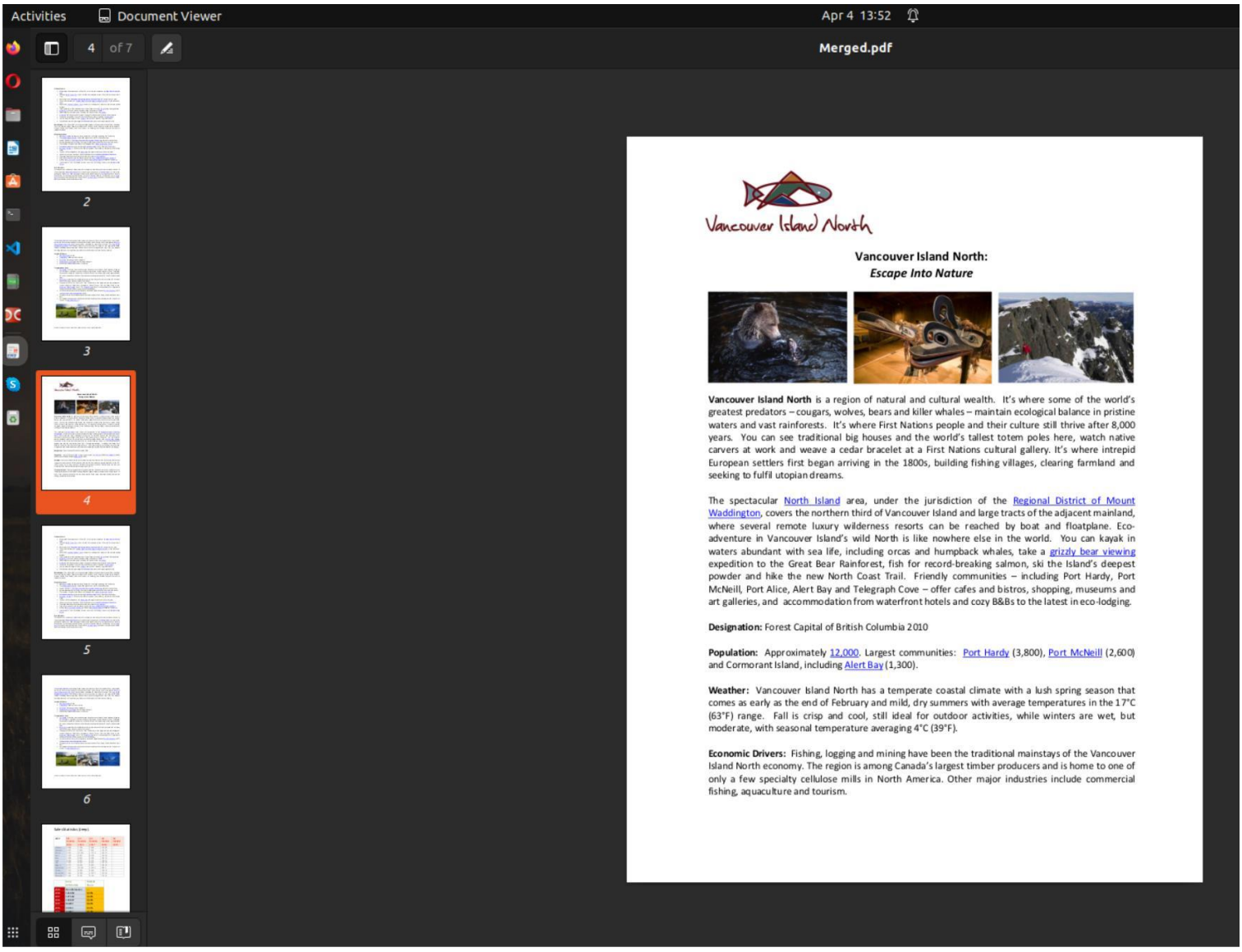
To make tests, we need an input PDF's documents.

Launch our application and merge PDF files, type the command: ***dotnet run***

If you don't see any exceptions, everything is fine and we can check the result produced by the PDF .Net library.

The new file “Merged.pdf” has to appear on the Desktop:

If we open this file in the default PDF Viewer, we’ll see its content:



Well done! You have created the “Merge PDF files” 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!