Guide: How to Create a Portable Julia Pro Installation for Windows

RoyiAvital #1 October 14, 2019, 7:57pm

In this post I will describe how I created a portable installation of Julia Pro.

Portable Installation

In the past Windows applications used to keep all configuration files and data files within the folder of the installation of the application.

Modern Windows applications use different approach where the binary files of the program are kept in one folder (Usually inside `Program Files` folder) and settings are kept in (Usually):

1. The `%USERPROFILE%` folder (Usually `C:\Users\<UserName>`).
2. The `%APPDATA%` folder (Usually `C:\Users\<UserName>\AppData\Roaming`).
   3 Inside the Registry.

Of course some applications can use a combinations of those.

Portable installation means it uses none of those. It uses the classic old approach where the application has built in folder to keep all its configuration and it doesn't touch the Registry.

I like this concept. In my system besides Visual Studio and MATLAB non of the applications I use are integrated to the system. All of them are portable (Most of them come from **PortableApps**).

It allows me to format, install or do what ever I want with my `C` drive and all programs are untouched.
It also keeps the performance of Windows like on its installation day (Though for modern Windows (7 and above) it is like that in any case). For other it will allow putting it on a USB Drive / Network Drive and use it whenever they want on any computer.

Portable Julia Pro

Gladly, it seems that Julia Pro installation doesn't use the registry (Note: *Keep it that way guys, great choice!). It means it uses the Environment Variables to write data onto the drive. It mainly uses `%USERPROFILE%`.

So all we need is to install it to any folder we want and trick it to think `%USERPROFILE%` is located on the same folder it was installed. Easy to do…

Installation Process

1. Install Julia Pro according to its installation guide (Don't run the installer with Admin rights!). Chose *Shared Drive* option. Chose a folder of your choice to install it. Let's call it `JuliaPro` from now on.
2. Once installation is done go to `JuliaPro` folder and create a folder inside it called `Profile`.

3. Go to `%USERPROFILE%` folder and look for a folder named `.juliapro`. Move it into the folder `JuliaPro\Profile\`.
4. Locate the file `Juno.bat` in `JuliaPro\`. Edit it with any text editor. Add the following lines on its top just after the `@ECHO OFF` command:

```
if not exist "%CD%\Profile\ProgramData" mkdir "%CD%\Profile\ProgramData"
if not exist "%CD%\Profile\Public" mkdir "%CD%\Profile\Public"
if not exist "%CD%\Profile\AppData\Roaming" mkdir "%CD%\Profile\AppData\Roaming"
if not exist "%CD%\Profile\Documents" mkdir "%CD%\Profile\Documents"
if not exist "%CD%\Profile\AppData\Local\Temp" mkdir "%CD%\Profile\AppData\Local\Tem
set ALLUSERSPROFILE=%CD%\Profile\ProgramData
set APPDATA=%CD%\Profile\AppData
set HOMEPATH=%CD%\Profile
set ProgramData=%CD%\Profile\ProgramData
set Public=%CD%\Profile\Public
set TEMP=%CD%\Profile\AppData\Local\Temp
set TMP=%CD%\Profile\AppData\Local\Temp
set USERPROFILE=%CD%\Profile
set JULIADOC=%CD%\..\..\Documents\Julia
```

The above just means when we launch *Juno* it will be tricked to use the `Profile` folder we created.
5. Open Juno using `Juno.bat` and enjoy *Almost* portable Julia Pro installation.

I write *Almost* because it is portable in the manner it writes everything (Configuration, packages, etc…) to the `Profile` folder yet there is one thing missing. If we copy this folder to another path once we launch *Juno* it won't find Julia Binary Files. Why is that?

Well, in *Juno* (Atom) we have a preference which sets the Julia binary file path.
The variable is called `juliaPath` and it is inside `JuliaPro\.atom\config.CSON`.
By default (Also the only officially way supported) it uses absolute path for Julia binary file.

So we have 2 choices, either remember to change it manually when we copy the folder to another path or use non officially supported feature (According to **@pfitzseb**) - Use relative path.

So, let's say we installed JuliaPro on `D:\` drive, namely, `D:\JuliaPro`. Then `juliaPath` will have the value `juliaPath: "D:\\JuliaPro\\Julia-1.2.0\\bin\\julia.exe` (In this case we used Julia `1.2.0`).

If we replace this by `juliaPath: ".\\Julia-1.2.0\\bin\\julia.exe"` (Namely the relative path to `JuliaPro`) we will have a completely portable installation of Julia Pro.

Pay attention this is not officially supported and the Relative Path evaluation might stop one day :-).
Until then, Put Julia in your OneDrive and enjoy it everywhere :-).

Questions

Some questions for the Julia Pro packagers:
4. I set the following environment variables: `%ALLUSERSPROFILE%`, `%APPDATA%`, `%HOMEPATH%`, `%ProgramData%`, `%Public%`, `%TEMP%`, `%TMP%` and `%USERPROFILE%`. Is there any other variable Julia might use? Is there a way it override the user definitions of those?
5. Does Julia / Julia Pro use Registry entries and not only system variables?

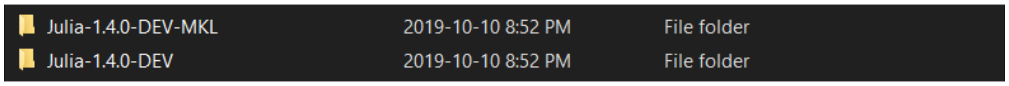
Julia Portable Mode o Standalone

Amin_Yahyaabadi #2 October 12, 2019, 10:18pm

I am not sure what you are trying to achieve. But you can grab one installation, and install it two times on different folders names:
I recommend using official binaries because JuliaPro doesn't always have the latest version of the packages (MKL for this case):

The Julia Language

Official website for the Julia programming language



Julia-1.4.0-DEV-MKL	2019-10-10 8:52 PM	File folder
Julia-1.4.0-DEV	2019-10-10 8:52 PM	File folder

The `Users/yourName/.julia/packages` is where packages are downloaded. This folder is shared between two different installations.

When you add MKL to one of those (**https://github.com/JuliaComputing/MKL.jl#to-install**), it downloads the package and then builds the files. So Julia-1.4-dev-MKL content is modified after installation.

You can install Atom:

A hackable text editor for the 21st Century

At GitHub, we're building the text editor we've always wanted: hackable to the core, but approachable on the first day without ever touching a config file. We can't wait to see what you build with it.

Then uber-juno package for Atom:

uber-juno

The official installer of Juno, the Julia IDE

You can change the Julia path for different installations doing this:
`Go to Packages > Julia > Settings and change "Julia Path" to point to the Julia binary.`

Alternatively, install Julia Pro and one Julia + MKL. Then point to the different path of Julia folders for two installations

RoyiAvital #3 October 12, 2019, 10:20pm

I'm not sure what you display but in my case even if they don't have the same location for packages (`.julia/packages/`) they both use MKL one I install on one of them.

So it has nothing to do with the path of the Julia binary but the path Julia use to look for packages (I might not be accurate as I'm not an expert on the way Julia handles things).

Look for the BLAS vendor on those two. Verify they are different.

Regarding the version of `MKL.jl`, it is not official package hence anyway it is being installed form the Git itself and not from a specific release of it.

Amin_Yahyaabadi #4 October 12, 2019, 10:28pm

I think you'd better use official binaries. It is simpler to have different installations. If works for me perfectly. JuliaPro is not really anything more than Atom+Julia

If JuliaPro is the same as usual official binaries: In the 2nd JuliaPro, `Go to Packages > Julia > Settings and change "Julia Path" to point to the Julia binary.` and point to the one which doesn't use MKL.

RoyiAvital #5 October 12, 2019, 11:14pm

@Amin_Yahyaabadi , Let me try to pinpoint the issue.
In my case not only they point to different Julia binary folder, they have different `Base.DEPOT_PATH`.
Yet, still something mix them to use the same BLAS although they shouldn't be able to be aware of each other (Remember, each have its Environment Variable pointing to different paths).

I suggest you have a look on your installation and verify that indeed you different BLAS vendor on each.

I don't want to use Julia's binaries.
I'm old fashioned on this and I want a package which coherent form the producer. Different people will

have different choice. Yet this is mine. Still it should work and something tricky I don't understand happening here.

Let's wait for an answer of someone who understand how Julia Pro sets some of its parameters. Specifically the `Path`. My indication (Since the folders are locked and I can't delete them) that one of the processes doesn't release the `convhost.exe` process. Which created a case when I launched them one after the other to "merge information". Just a speculation.
We'll have to patient until someone who is informed will explain.

Amin_Yahyaabadi #6 October 13, 2019, 5:24am

> **Problem with 2 Different Installations of Julia Pro Interfering One with Each Other**
>
> @Amin_Yahyaabadi , Let me try to pinpoint the issue. In my case not only they point to different Julia binary folder, they have different Base.DEPOT_PATH. Yet, still something mix them to use the same BLAS although they shouldn't be able to be aware of each other (Remember, each have its Environment Variable pointing to different paths). I suggest you have a look on your installation and verify that indeed you different BLAS vendor on each. I don't want to use Julia's binaries. I'm old fashi…

For my case, given that I use official binaries, different installations have different `BLAS.vendor()`.

Yes, I prefer Julia Pro users answer this. Move your thread to Tooling and JuliaPro section of the forum to get more answer.

pfitzseb #7 October 14, 2019, 8:11am

Ok, here's what I tried:

1. Install JuliaPro (with shared drive option) in `D:\\JuliaPro-1.2.0-1-mkl`
2. Install JuliaPro (normal user install, but shouldn't matter) in `D:\\JuliaPro-1.2.0-1`
3. Start the MKL JuliaPro, install MKL.jl, restart Julia processes a couple of times and check that

```
julia> using LinearAlgebra

julia> BLAS.vendor()
:mkl

julia> normpath(Sys.BINDIR, "..", "lib", "julia")
"D:\\Programme\\JuliaPro-1.2.0-1-mkl\\Julia-1.2.0\\lib\\julia"
```

4. Start the other JuliaPro install and check that

```
julia> using LinearAlgebra

julia> BLAS.vendor()
:openblas64
```

Checking the changed date of both `sys.dlls` (`D:\\JuliaPro-1.2.0-1-mkl\Julia-1.2.0\lib\julia\sys.dll` and `D:\\JuliaPro-1.2.0-1\Julia-1.2.0\lib\julia\sys.dll`) shows that the MKL one was updated correctly.

Note that I didn't mess around with any env vars etc, so both installations had MKL.jl *available*. That doesn't really matter though, because MKL.jl installation modifies your *Julia installation* – it's not purely a package (and also can't really be uninstalled).

RoyiAvital #8 October 14, 2019, 9:06am

@pfitzseb , appreciate the effort to check this.

I found the mistake I made and fixed it. Now it works on my system as well.
I will update the post with the mistake I made and will create a guide - How to Create a Portable Installation of Julia Pro on Windows.

Thank You.

1 Like

RoyiAvital #9 April 1, 2020, 10:41pm

@pfitzseb ,
I have downloaded `JulaiPro 1.4.0-1`.

I tried doing the same as I did for `Julia 1.2.0-1`.
All worked except for setting:

```
juliaPath: ".\\Julia-1.4.0\\bin\\julia.exe"
```

Any reason it doesn't work on this version?
I tried entering completely garbage paths and got a message and got `The system cannot find the path specified.`.
So the path `.\\Julia-1.4.0\\bin\\julia.exe` is correct but for some reason it doesn't let Julia start.

It gives (Indeed it gives the same message twice):

```
Julia has exited.
Press Enter to start a new session.
```

```
Julia has exited.
Press Enter to start a new session.
```

Any idea?

Update

After conversation of **@pfitzseb**, he has created a fix - **https://github.com/JunoLab/atom-julia-client/pull/711**.

Not sure when it will be available in JuliaPro.

JohnDoe #10 April 2, 2020, 2:25am

I tried to download the file **https://github.com/JunoLab/atom-julia-client/blob/master/lib/misc/paths.coffee** and save and replace the old one at `\JuliaPro\.atom\packages\julia-client\lib\misc\`, but it still doesn't work, showing the same error message of `Julia has exited. Press Enter to start a new session.`

I hope this can be trivially fixed after they update the `julia-client` package on **atom.io**, so that we can uninstall the package and re-install the updated version within atom. Otherwise we have to wait until Julia Computing updates the entire JuliaPro, which could take much longer than community maintained packages.

RoyiAvital #11 May 29, 2020, 9:31am

In order to set the Juno Window Size programitically edit `init.coffee` in `.atom` folder with:

```
remote = require('remote')
remote.getCurrentWindow().setBounds({ x: 10, y: 10, width: 1600, height: 1350 })
```

Where x and y are the coordinates of the top left point of the window where `0, 0` is the top left of the screen.