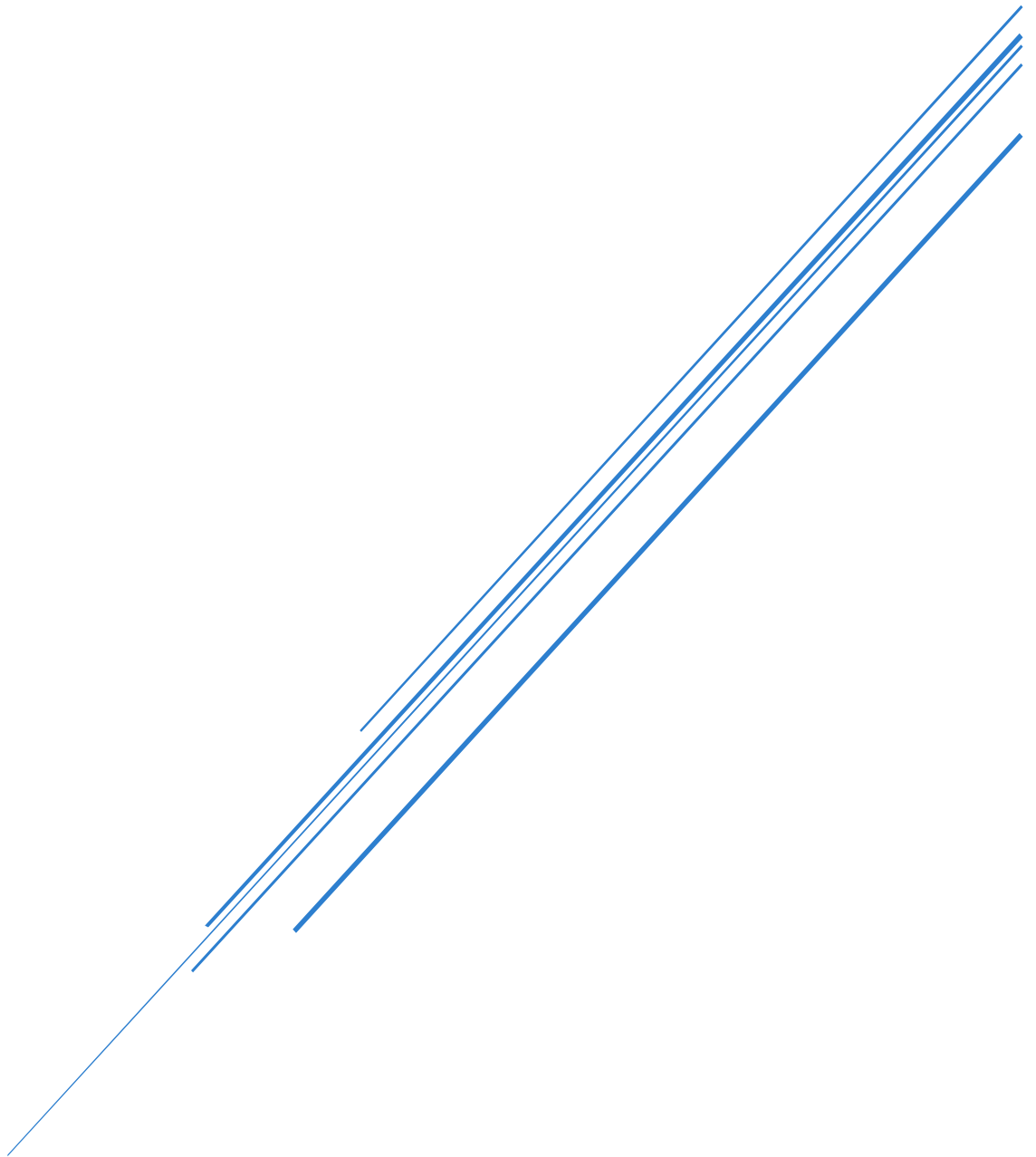


HOW TO USE WSL DISTRIBUTION ON WINDOWS

WSL Setting and Import



Masahiro Akizuki

Contents

Prerequisites.....	2
Install Ubuntu (Dummy).....	2
Why use Debian.....	5
Run Linux GUI apps on the Windows Subsystem for Linux.....	6
Import methods.....	6
Step 1: Import a Linux Distribution.....	6
Step 2: Change User (from root to elmeruser).....	7
How to use.....	7
External Tools.....	7
Salome or Salomemeca.....	7
Salome.....	7
Salomemeca.....	7
Gmsh.....	8
File Exchange with Windows.....	8
Open Linux files from Windows applications.....	8
Paraview.....	8
VSCode.....	9
Exporting WSL.....	10

Prerequisites

You must be running Windows 10 version 2004 and higher (Build 19041 and higher) or Windows 11 to use the commands below.

Installed driver for vGPU (Driver for Windows)

Intel, AMD, NVIDIA

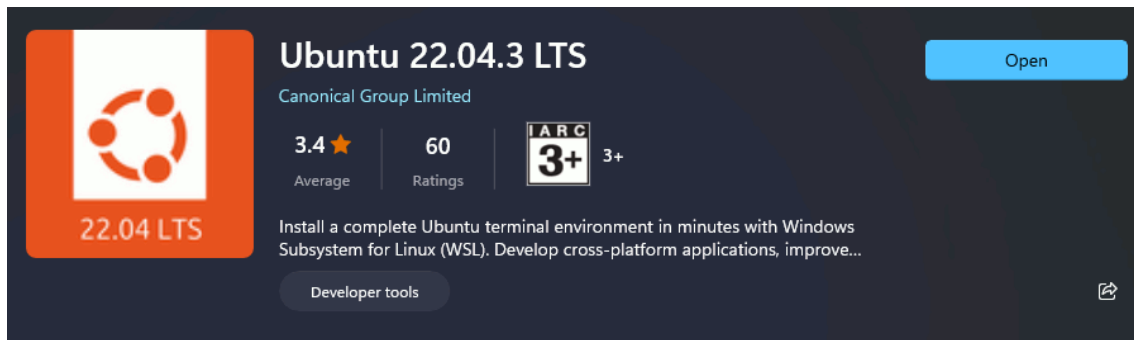
Install Ubuntu (Dummy)

Skip this step if your environment is already running WSL and GUI environment.

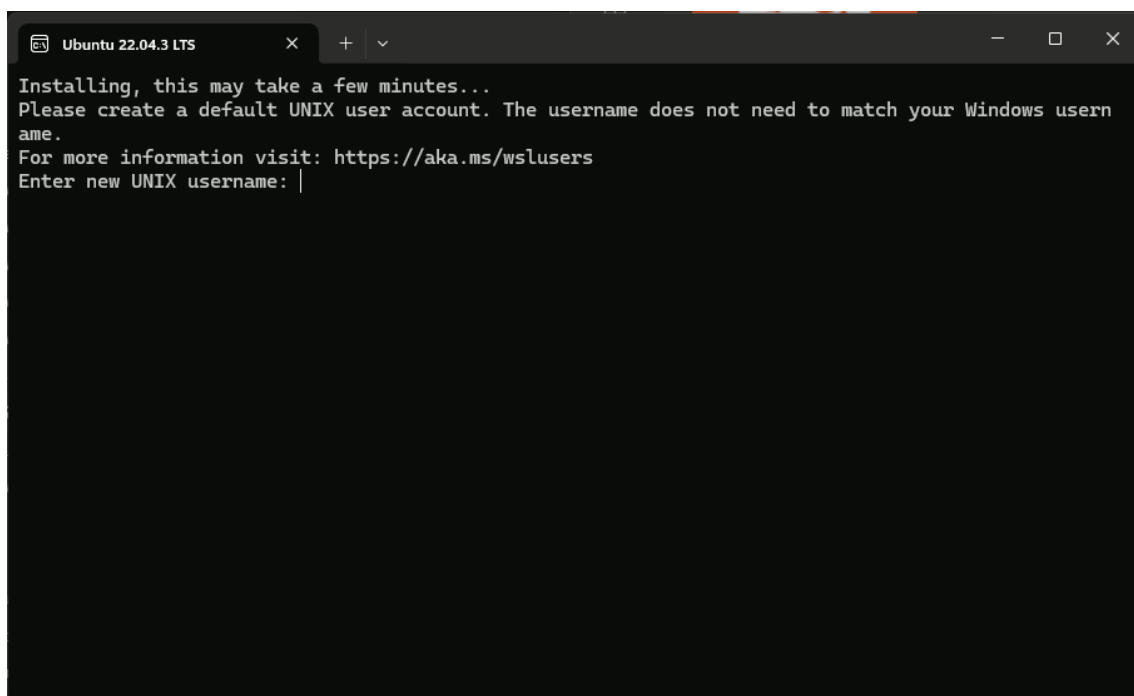
Install Ubuntu 22.04 LTS from the Microsoft Store.

After download and install, open Ubuntu.





Next, input the username and password.



```
masahiro@Desktop-12th: ~  
Installing, this may take a few minutes...  
Please create a default UNIX user account. The username does not need to match your Windows username.  
For more information visit: https://aka.ms/wslusers  
Enter new UNIX username: masahiro  
New password:  
Retype new password:  
passwd: password updated successfully  
Installation successful!  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.146.1-microsoft-standard-WSL2 x86_64)  
  
* Documentation: https://help.ubuntu.com  
* Management:   https://landscape.canonical.com  
* Support:      https://ubuntu.com/advantage  
  
This message is shown once a day. To disable it please create the  
/home/masahiro/.hushlogin file.  
masahiro@Desktop-12th:~$
```

In my environment, I could not connect to the Internet, so I edited the following two files.

```
sudo nano /etc/wsl.conf
```

```
[network]
```

```
generateResolvConf = false
```

```
GNU nano 6.2 /etc/wsl.conf *  
  
[boot]  
systemd=true  
  
[network]  
generateResolvConf = false  
|  
  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location  
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

exit

run Terminal on Windows

wsl --shutdown

wsl -d ubuntu-22.04

sudo nano /etc/resolv.conf

nameserver 8.8.8.8 # 1.1.1.1

If "sudo apt update && sudo apt full-upgrade" works, the connection to the Internet is established.

A screenshot of a terminal window titled "masahiro@Desktop-12th: ~". Inside the terminal, the nano text editor is open, editing the file "/etc/resolv.conf". The editor's status bar at the top shows "GNU nano 6.2" and the file path. The content of the file is "nameserver 8.8.8.8 # 1.1.1.1". At the bottom of the terminal, there is a detailed menu bar with various keyboard shortcuts for nano editor functions: ^G Help, ^O Write Out, ^W Where Is, ^K Cut, ^T Execute, ^C Location, ^U Undo, ^M-A Set Mark, ^X Exit, ^R Read File, ^N Replace, ^U Paste, ^J Justify, ^_ Go To Line, ^M-E Redo, and ^M-G Copy. A small status indicator "[Read 1 line]" is also visible above the menu bar.

Why use Debian

Ubuntu is useful when using NVIDIA's GPU computing.

However, I prefer Debian for CPU computing for the following reasons.

- Ubuntu recommend the Ubuntu Pro. Ubuntu Pro costs \$500.00 per year for commercial use and is free for up to 5 computers for personal use. I appreciate Debian's philosophy of being "freely available for everyone".
- Ubuntu is basically Debian Sid (Unstable) packages compiled from source code. Frequently used apps are fine, but I find it unreliable for apps that are used infrequently. On the other hand, Debian has been enough testing before going into Stable.

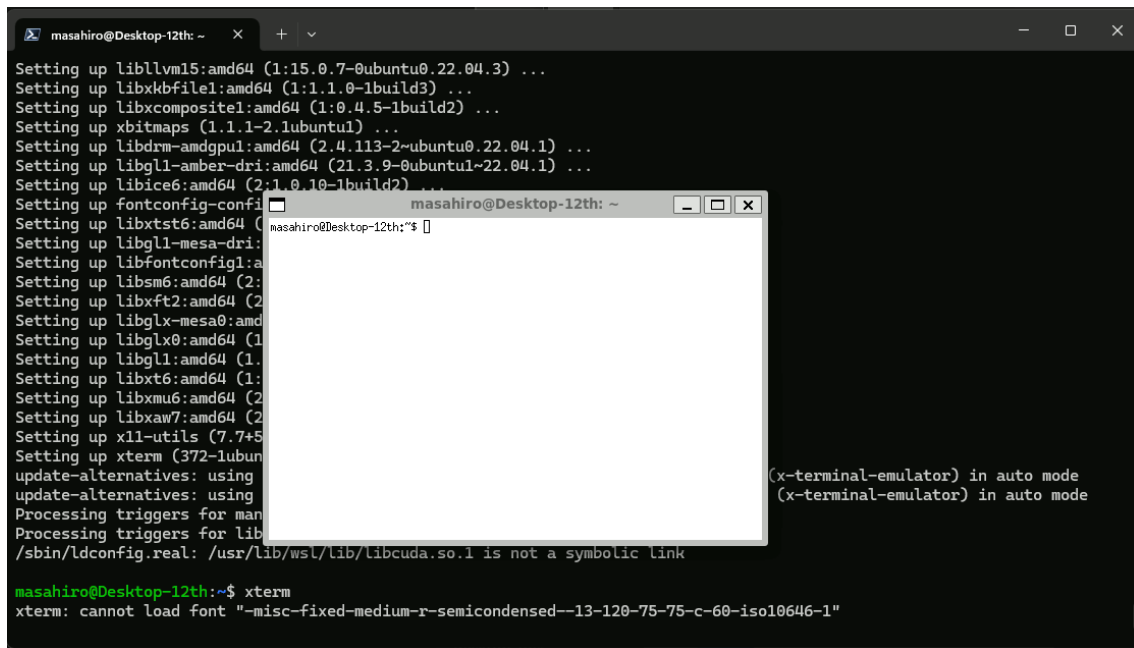
Run Linux GUI apps on the Windows Subsystem for Linux

Windows Subsystem for Linux (WSL) now supports running Linux GUI applications (X11 and Wayland) on Windows in a fully integrated desktop experience.

Run Linux GUI apps.

```
sudo apt install xterm
```

```
xterm
```



The screenshot shows a terminal window titled 'masahiro@Desktop-12th: ~'. The terminal output displays the installation of various Linux packages, including 'libllvm15:amd64', 'libxkbfile1:amd64', 'libxcomposit1:amd64', 'libxmaps (1.1.1-2.1ubuntu1)', 'libdrm-amdgpu1:amd64', 'libgl1-amd64', 'libice6:amd64', 'fontconfig-config', 'libxtst6:amd64', 'libgl1-mesa-dri', 'libfontconfig1:amd64', 'libsm6:amd64', 'libxft2:amd64', 'libglx-mesa0:amd64', 'libglx0:amd64', 'libgl1:amd64', 'libxt6:amd64', 'libxmu6:amd64', 'libxaw7:amd64', 'x11-utils (7.7+5)', and 'xterm (372-1ubuntu1)'. The output also shows the update of alternatives and the processing of triggers. A warning message is displayed: '/sbin/ldconfig.real: /usr/lib/wsl/lib/libcudart.so.1 is not a symbolic link'. The user then runs the command 'xterm', and the terminal displays 'xterm: cannot load font "-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso10646-1"'. A small window titled 'masahiro@Desktop-12th: ~' is also visible in the background.

Import methods

Step 1: Import a Linux Distribution

Open a command prompt or PowerShell window.

Run the following command to import the desired Linux distribution (replace elmer_debian12 with your preferred name and adjust the paths accordingly):

```
wsl --import elmer_debian12 c:\¥elmer_debian12 .¥ ElmerWSL_with_ElmerVM_Scripts.tar
```

(¥ means backslash)

For more details, you can refer to the official documentation: Use Custom Distro with WSL.

<https://learn.microsoft.com/en-us/windows/wsl/use-custom-distro>

<https://learn.microsoft.com/en-us/windows/wsl/basic-commands>

Step 2: Change User (from root to elmeruser)

Edit the `/etc/wsl.conf` file within Elmer Debian WSL distribution.

Add the following section to set the default user to elmeruser:

```
[user]
default=elmeruser
```

How to use

Commands: “ElmerSolver”, “ElmerGrid”, “ElmerGUI”, “paraview”, “mpirun”, “ElmerSolver_mpi”

The username for the WSL is "elmeruser" and the password "elmerfem".

I recommend changing the password using the “passwd” command.

External Tools

Except for those familiar with Linux, the Windows versions of these two apps are recommended.

Salome or Salomemeca

Salome

https://www.salome-platform.org/?page_id=2430

Salomemeca

<https://code-aster-windows.com/download/>

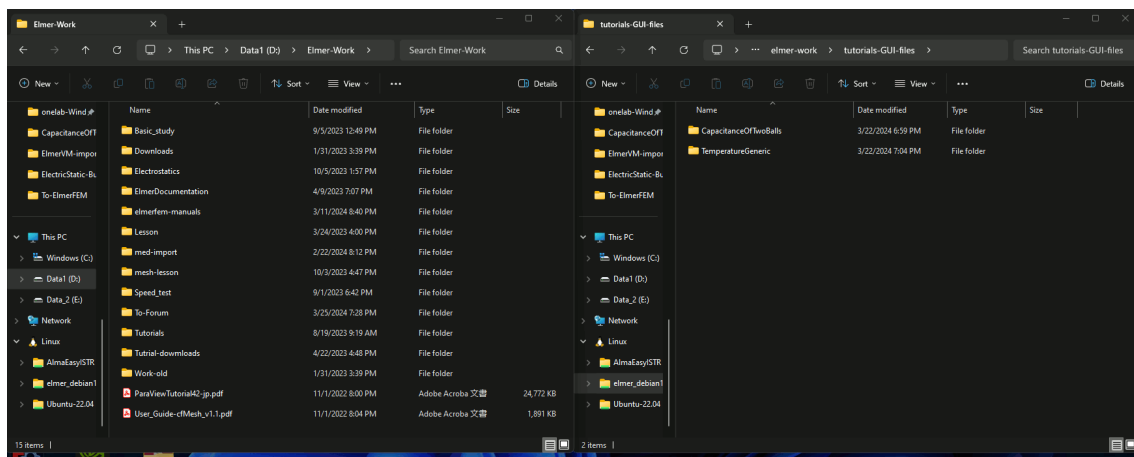
Gmsh

Gmsh included with Onelab (Gmsh + GetDP) is newer. Just unzip it and it will work.

<https://onelab.info/>

File Exchange with Windows

You can easily manipulate files by launching two Windows Explorers.

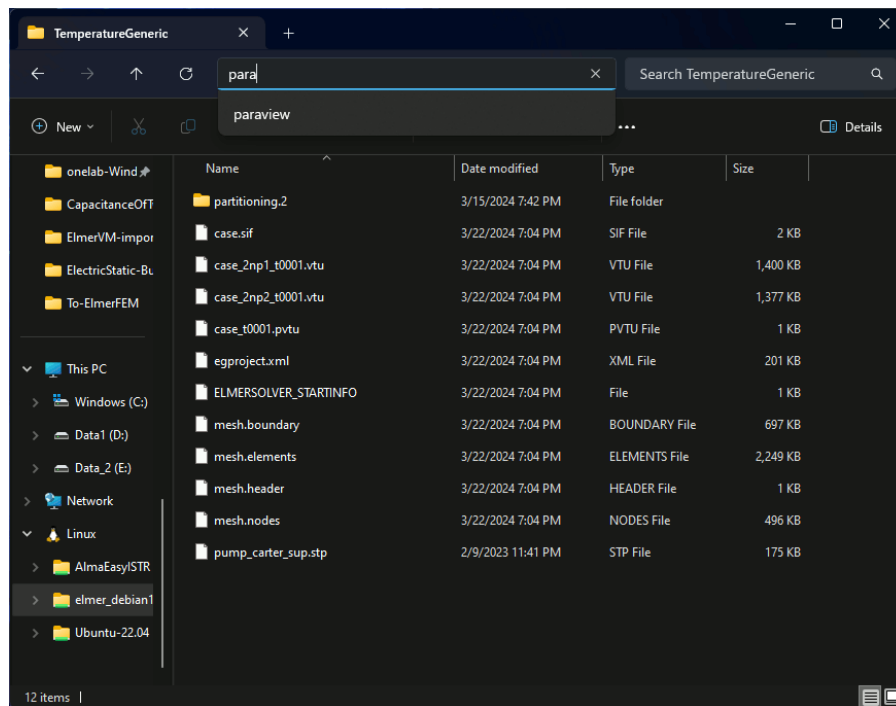


Open Linux files from Windows applications

Salome is not supported.

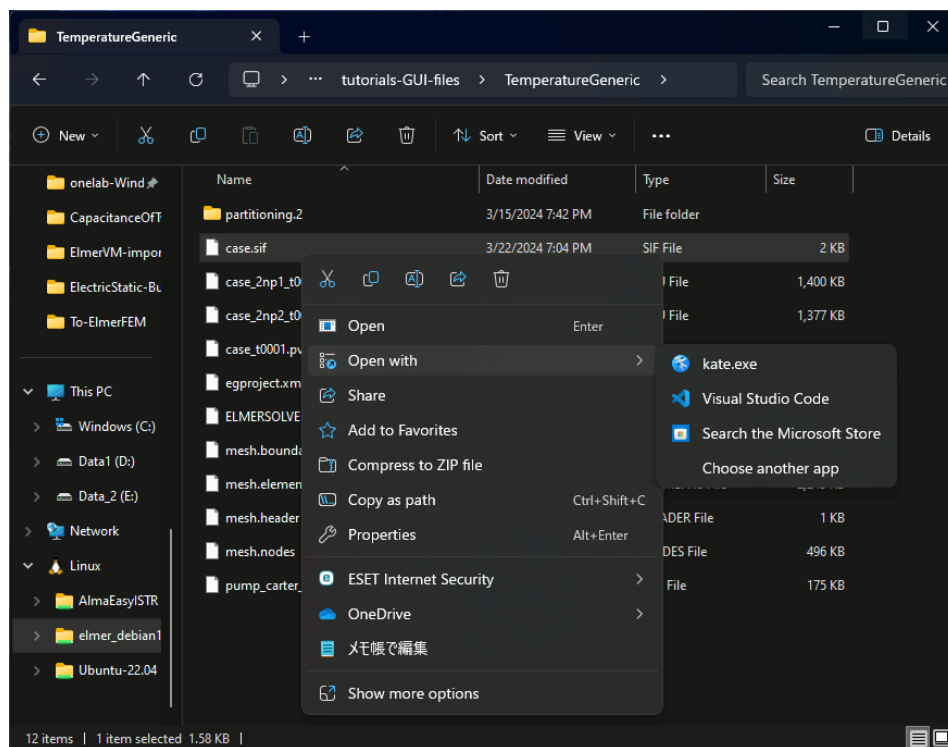
Paraview

Select the folder you want to open the Linux distribution from Explorer and type paraview at the top.



VSCode

Select the “case.sif” you want to open the Linux distribution from Explorer, right-click and open with Visual Studio Code.



Exporting WSL

You can export WSL by the following commands.

You can improve the WSL distribution I made and share a better version easily.

`wsl --export <WSL Image Name> <Export file>`

<https://4sysops.com/archives/export-and-import-windows-subsystem-for-linux-wsl/>

(EOF)