ScanLab-WSL2

Preloaded with Fsl, Matlab and Brainvoyager

Login: **scan**
Pwd: **scan123**

by Jorie van Haren

March 7, 2024

Contents

0.1 Short word

Below you will find instructions for installing and running a WSL2 (Linux subsystem for windows) Ubuntu distribution with a GUI interface. The installation comes fully prepackaged with FSL, FSLEYES, Conda, Matlab and BrainVoyager (relying on network licences for both Matlab and Brainvoyager). The full system runs using WSL2 and is, therefore, way more powerfull compared to a virtual machine (meaning we can fully capitalise on certain aspects of modern hardware, such as GPU-acceleration). After installation you can launch programs using either:

- Terminal commands (requires XLAUNCH) -
  i.e. type BrainVoyager, fsl, fsleyes, matlab etc.

- Use remote desktop connection to access a GUI interface -
  i.e. run windows msct, login to linux ip

Both methods work locally (on the machine with the WSL-instance installed), as well as remotely from any other machine (requiring a VPN-connection and a SSH-tunnel).

While the WSL instance comes preinstalled with many of our favourite programs, there are some steps you still may want to do to get the most out of the system. Firstly, the installation does not come with CUDA out of the box, doing so would hamper flexibility and heavily limit the amount of compatible devices this distribution can be installed on. Below you will find instructions to install CUDA yourself. Secondly, while Matlab is installed, it must be reactivated by either typing 'activate_matlab.sh' or by clicking the activate matlab shortcut on the desktop.

0.2 Starting an instance

After installation you have two possibilities approaches of running installed Linux programs. You can A. run them from the terminal (opening the program in a new window through XLAUNCH) or B. run them using the GUI Linux interface.

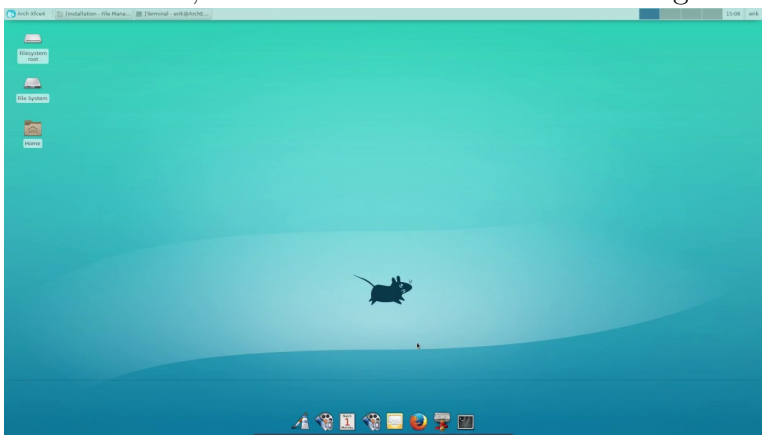
Using the terminal

1. In Windows, launch *'XLaunch'* - (Download)
   - In extra settings check *'Disable access control'*, and uncheck *'Native opengl'*
   - leave everything else default and run
2. Open Ubuntu by A. typing *'Ubuntu'* in Windows Powershell, or B. by opening Ubuntu app (search for Ubuntu)
3. Type the name of the program you want to run (case sensitive)

```
fsl
fsleyes
matlab
activate_matlab.sh
BrainVoyager
jupyter notebook
xdg-open /home/scan/
```

Using the GUI interface

Here we connect to a GUI (XFCE-4) interface of our Linux distro using a remote connection tool (here I use MSCT - Remote Desktop Connection, as it comes preinstalled on any modern windows machine, however other alternatives like TigerVNC are available).



1. **Restart the xrdp service**

```
sudo service xrdp restart
```

2. **Login**
   - Windows search *'msct'* (Remote Desktop Connection)
   - Connect to **localhost:3390**
   - Login with username: **scan** and password: **scan123**
3. Many of the programs can be run by double clicking desktop shortcuts, alternatively you can use the terminal and type the program name or path

0.3 Installation instructions

Importing the WSL instance with GUI

Download the .tar file from Surfdrive. Open Windows Powershell (as administrator):

```
# make new directory (in this example in D:)
New-Item -Path "D:\WSL2GUI" -ItemType Directory
# import wsl distro from file (place in above created folder)
wsl --import Ubuntu "D:\WSL2GUI" "C:\Users\xx\Downloads\ubuntu-scan.tar"
# set default login account to scan
ubuntu config --default-user scan
```

Here I chose to import our WSL instance to the **D:** drive, mainly to allow for A. multiple users on one machine, and B. to be less constrained by diskspace. Feel free to change the destination to your heart's desire :).

If you only need Linux for some light programs (without GPU), you are good to go, see run instructions on previous page.

Download CUDA for Linux WSL [optional but recommend]

Within a Linux terminal (open Ubuntu or type Ubuntu in Powershell):

```
# install nvidia drivers
sudo apt install nvidia-driver-460

# install cuda
wget https://developer.download.nvidia.com/compute/cuda/11.2.0/
   local_installers/cuda_11.2.0_460.27.04_linux.run
sudo sh cuda_11.2.0_460.27.04_linux.run
```

You can then download and install CUDNN:

1. Download cudnn (version 11.2-linux-x64-v8.1.0.77)
2. Copy to ubunty home folder: \\wsl$ \Ubuntu\home\Downloads

```
cd /home/scan/Downloads/
tar -xzvf cudnn-11.2-linux-x64-v8.1.0.77.tgz
sudo cp /home/scan/Downloads/cuda/include/cudnn*.h /usr/local/cuda/
   include
sudo cp /home/scan/Downloads/cuda/lib64/libcudnn* /usr/local/cuda/lib64
sudo chmod a+r /usr/local/cuda/include/cudnn*.h /usr/local/cuda/lib64/
   libcudnn*
```

Optional: change password

This step is highly recommended, especially if your want to use the remote options described below. Within a Linux terminal (old password is scan123):

```
passwd
```

0.4 Remote access

Here I will describe some options to access our WSL instance remotely, circumventing the need to use a full remote desktop. Note that all instruction described here are fully optional but might be useful when, for example, working from home.

Terminal - Using XLaunch

This approach does not require a windows OpenSSH server to be running, it does require a WSL SSH server to be running (should come preinstalled but might need a restart using *'sudo service ssh start'* or *'sudo service ssh --full-restart'*).

1. Make sure you are using a VPN-connection to the university
2. In Windows Powershell (open as admin)

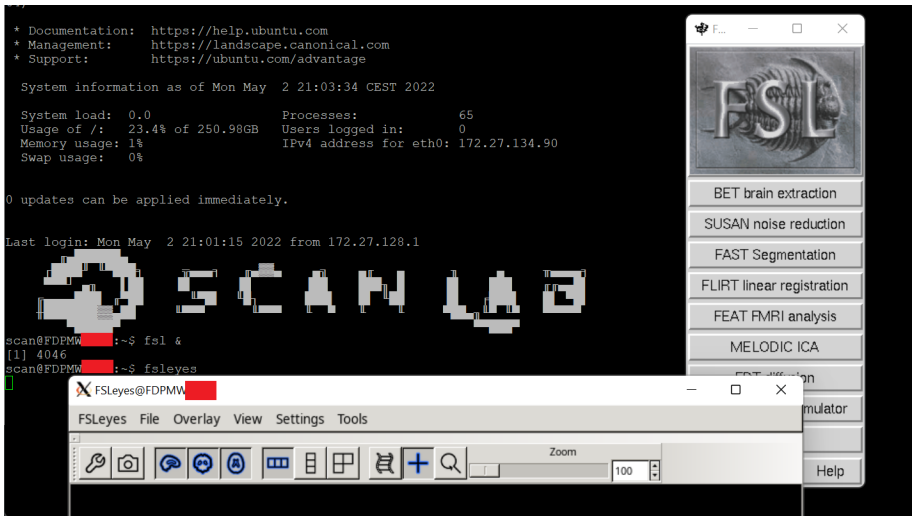
```
# add ip-address
netsh interface portproxy add v4tov4 listenaddress=0.0.0.0
    listenport=2222 connectaddress=localhost connectport=2222

# add port
netsh advfirewall firewall add rule name= Open Port 2222 for
    WSL2 dir=in action=allow protocol=TCP localport=2222
```

   Take care that you have set a secure password for your instance, since this will enable SSH access from anywhere within the VPN-network. Also: as a safety precaution, you will probably get an email asking you why you forwarded this port on your machine (since the university hired a company to monitor these kinds of commands on Windows machines.)
3. SSH to your WSL-instance - (for Windows use Putty)
   - Hostname: **FDPMWXXXX.wired.unimaas.local:2222**
   - Connections > SSH > X11: Enable X11 forwarding
   - Optional: Give saved session a name and save
   - For Mac you can just: (using own workstation number)

```
ssh -p 2222 -X scan@FDPMWXXXX.wired.unimaas.local
```

4. Login with the WSL credentials (default username: **scan** and password: **scan123**)
5. On windows open *'XLaunch'* (Download), on Mac run the **XQuartz.app**
   - In extra settings check *'Disable access control'*, and uncheck *'Native opengl'*
   - leave everything else default and run
6. Type the name of the program you want to run (case sensitive)



GUI Environment - Using MSCT

This approach requires your desktop to have an OpenSSH server running. Please follow instructions listed Here.

1. Make sure you are using a VPN-connection to the university
2. On the host PC, either:
   - Go to: **System → Device name → DNAME.wired.unimaas.local**
     - *e.g. FDPMWXXX.wired.unimaas.local*
   - Google: *my ip*
     - *e.g. 137.120.XXX.XX*
   - Command prompt: type *ipconfig*
     - *e.g. 137.120.XXX.XX*
3. SSH to your Windows machine - (for Windows use Putty)
   - Hostname: **FDPMWXXXX.wired.unimaas.local:22**
   - Connections > SSH > Tunnel: 3390:localhost:3390
   - Connections > Data: Auto login username: unimaas\pXXXXXXXX
   - Optional: Give session a name and save
   - For Mac you can just: (using own P-Number and workstation number)

```
ssh -N -L localhost:3390:localhost:3390 unimaas\pXNumberX@FDPMWXXXX
    .wired.unimaas.local
```

4. Login using your desktop password
5. Open **MSCT** (Remote Desktop Connection) → localhost:3390
   (or use a different remote client)
6. The program will open like any other GUI app

0.5 Problem Solving

Connection issues

In some instances windows can block traffic to and from the virtual network to your Windows environment. Here I will describe two common problems:

1. Localhost might not be forwarded from WSL to windows, meaning that you can only connect to the WSL ip-address (type *'ifconfig'* in WSL). To make sure we can connect to localhost instead, we will create (or alter if it already exists) a *.wslconfig* file (can make with notepad) in your *'%UserProfile%'* folder. Copy paste the following settings:

```
[wsl2]
localhostForwarding=true
localportForwarding=true
```

2. XLaunch / VcXsrv might be blocked by Windows Defender (circumventing you to open GUI apps from the terminal). Go to **Control Panel** > **System and security** > **Windows defender firewall** > **Advanced settings** > **Inbound rules** and make sure VcXsrv rules are *not* set to *block* (if it is, change it to *allow*).

Known bugs

After compiling the environment, and after changing it to my personal needs I noticed that within the *'brainvoyager'* environment for conda, I am using a incorrect version of *'tensorflow'*, namely version 2.2. This version is not compatible with the version of Cuda that could be installed on a WSL2 machine. Please upgrade tensorflow to version 2.5.0, in order to enable GPU acceleration (with CUDA version 11.2, and cuDNN version 8.1).

```
source activate brainvoyager
pip install --upgrade tensorflow==2.5.0
# follow instructions if any other packages have to be upgraded before
```

0.6 Other tricks

You are able to access all your Linux files by going to *wsl$* \ in Windows file explorer.