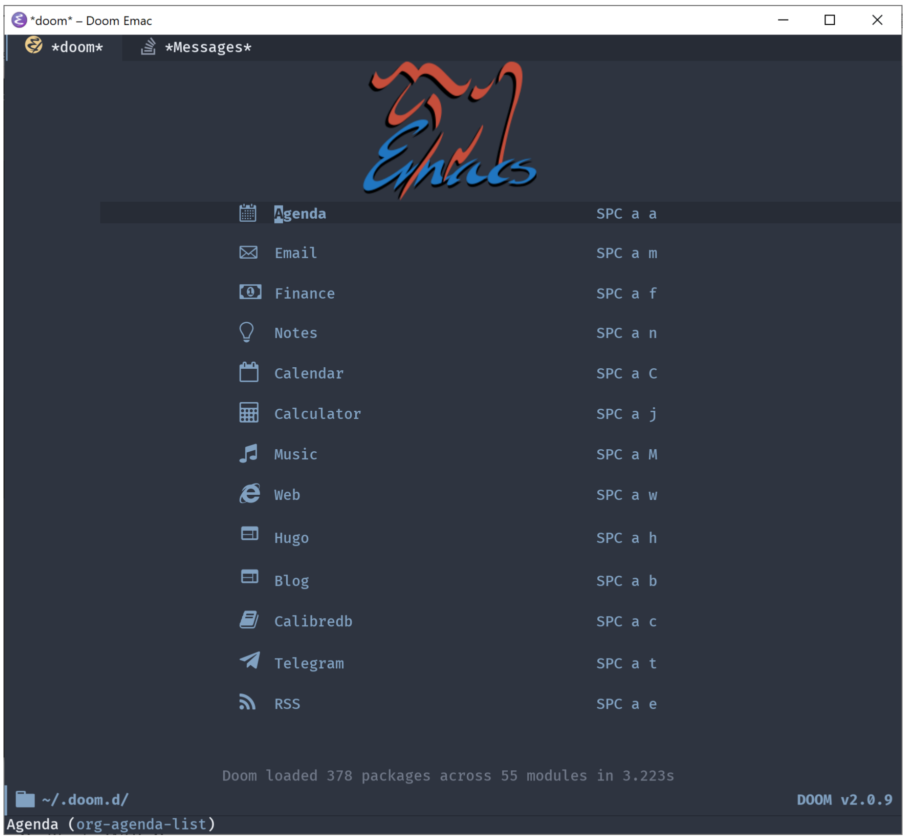
The ultimate Emacs hacking tutorial in Windows 10 WSL 2

INITIATED BY	WRITTEN BY	PRESENT	ABSENT	EXCUSED
PARTICIPANTS				

PROJECT	DATE October 1, 2020	DURATION

Contents


```
*doom* – Doom Emac
*doom*   *Messages*

Agenda          SPC a a
Email           SPC a m
Finance         SPC a f
Notes           SPC a n
Calendar        SPC a C
Calculator      SPC a j
Music           SPC a M
Web             SPC a w
Hugo            SPC a h
Blog            SPC a b
Calibredb       SPC a c
Telegram        SPC a t
RSS             SPC a e

Doom loaded 378 packages across 55 modules in 3.223s
~/.doom.d/                                   DOOM v2.0.9
Agenda (org-agenda-list)
```

Do you think Emacs's performance in windows is bad? Do you really want to use a native Speed Emacs in Windows? Do you hate the unnatural path transition between windows convention and Linux convention? Do you feel frustrated when you try to install and configure Emacs in WSL? Do you just want to taste the power of Emacs running in WSL 2?

This tutorial may help you :)

1 What is WSL?

Earlier on August 2, 2016, Microsoft released **Windows Subsystem for Linux** (WSL), enabling the native way to run Linux Tools in Windows 10 and Windows Server 2019.

In May 2019, WSL 2 was introduced, by importing the Real Linux Kernel through Hyper-V features (in a Virtual Machine Environment), providing the users with the full & immerse way to work with Linux under windows, with 20 times the read/write performances of WSL 1.

For Windows Emacs Users, here are some advantages/disadvantages for you to consider before switching the WSL 2:

1.1 Advantages

1. The performance of Magit is way faster than the GNU compiled original windows Emacs-27 binaries.
2. The Font Rendering is better.
3. No flickering.
4. interoperability between Windows and Linux.
5. Native OneDrive support.
6. Super fast boot up time for Emacs.

1.2 Disadvantages

1. The network configuration is a pain, workaround is available.
2. X11 may lost connection when network changes.

1.3 More Details

More details of the differences between WSL 1 and WSL 2, check `https://docs.microsoft.com/en-us/windows/wsl/compare-versions`.

2 Taste WSL 2

2.1 Install and enable WSL 2

2.1.1 Update Windows 10

The first step is to make sure you have updated to the latest Windows 10. For Windows 10 Versions 1903 & 1909 users, make sure the minor version number is **1049**, according to Microsoft's Devblog.

2.1.2 Install WSL 2

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
```

2.1.3 Enable WSL 2

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

2.1.4 Check WSL version

```
wsl -l -v
```

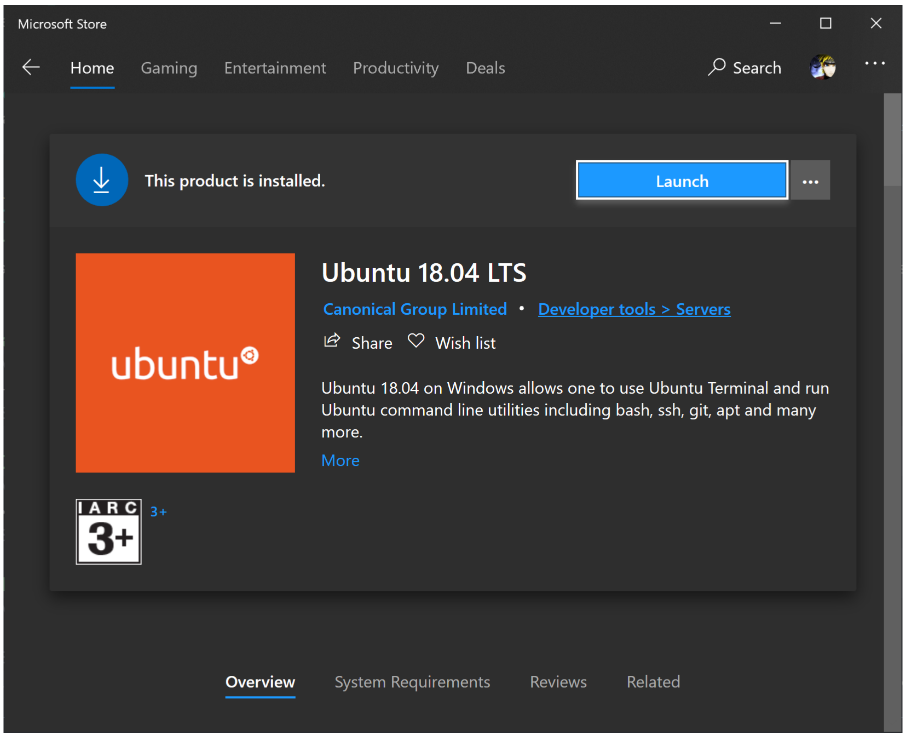
There are many people who may be not able to run the above commands, just make sure you have updated to the latest windows. `https://devblogs.microsoft.com/commandline/wsl-2-support-is-coming-to-windows-10-versions-1903-and-1909/`

2.1.5 Set default WSL version to 2

```
wsl.exe --set-default-version 2
```

All later WSL Distros are installed will be WSL 2. No worries, you can switch between WSL 1 and WSL 2 with just one command without pain, just do it~

2.1.6 Install Ubuntu 18.04 in Microsoft Store



This product is installed.

Ubuntu 18.04 LTS

Canonical Group Limited • Developer tools > Servers

Ubuntu 18.04 on Windows allows one to use Ubuntu Terminal and run Ubuntu command line utilities including bash, ssh, git, apt and many more.

2.1.7 Optional: Convert WSL 1 to WSL 2

If you already installed some Distros before but they are in WSL 1, no worries, it is very easy to switch from WSL 1 to WSL 2:

```
wsl.exe --set-version Ubuntu-18.04 2
```

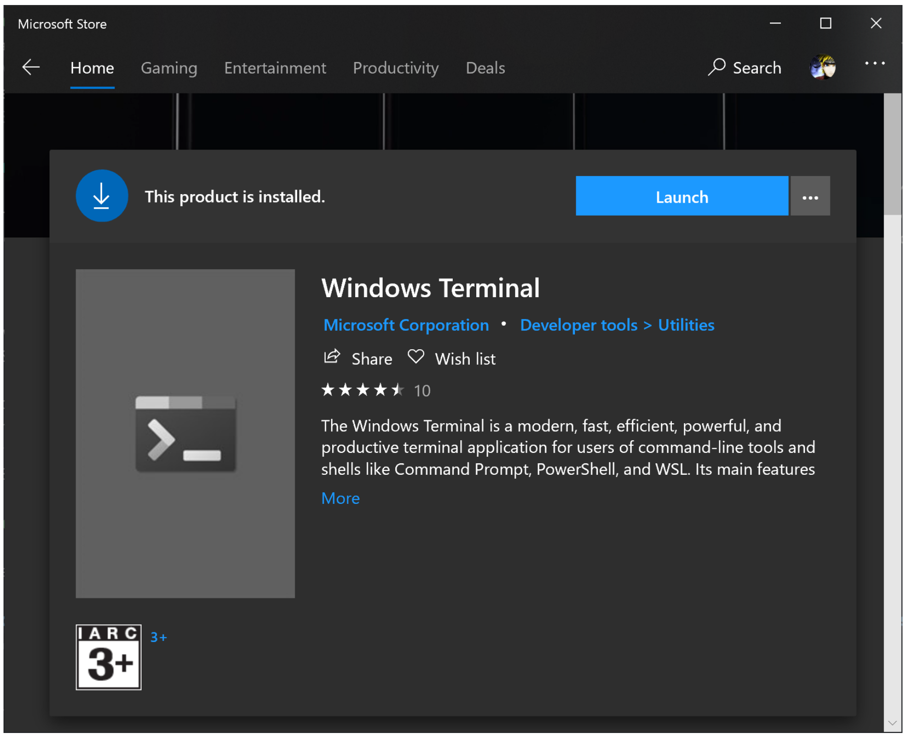
2.2 Choosing the terminal

The first step to talk to WSL 2 is via Terminal Applications. In macOS, we have iTerm2, how about Windows? You may wonder we could just use the default Ubuntu Terminal, but it lacks UTF8 and Unicode supports, and only a few configuration options. Apparently, it is not the best option. We just need a more powerful and modern terminal. Luckily, Microsoft brings us another great product - Windows Terminal, and it is ranked with **67.6k** stars in GitHub up to the time this article composed.

Windows Terminal has the following benefits compared with other terminals:

1. Windows Terminal is multi-tabs and multi-panels.
2. UTF8 & Unicode support.
3. Supports Command Prompt, PowerShell, and WSL.
4. GPU accelerated Text Rendering Engine.
5. Custom Themes, styles and configurations.
6. Modern and user-friendly.
7. Open Source and Official Support.

2.2.1 Install windows Terminal in Microsoft Store



The Windows Terminal is a modern, fast, efficient, powerful, and productive terminal application for users of command-line tools and shells like Command Prompt, PowerShell, and WSL. Its main features

2.2.2 Windows Terminal

Windows Terminal's setting is implemented in a json file - `settings.json`, every time you modify and save the file, it will take effects immediately, nice!

1. Add "cursorShape" and "fontFace" to the "defaults" section, it will apply on all tabs

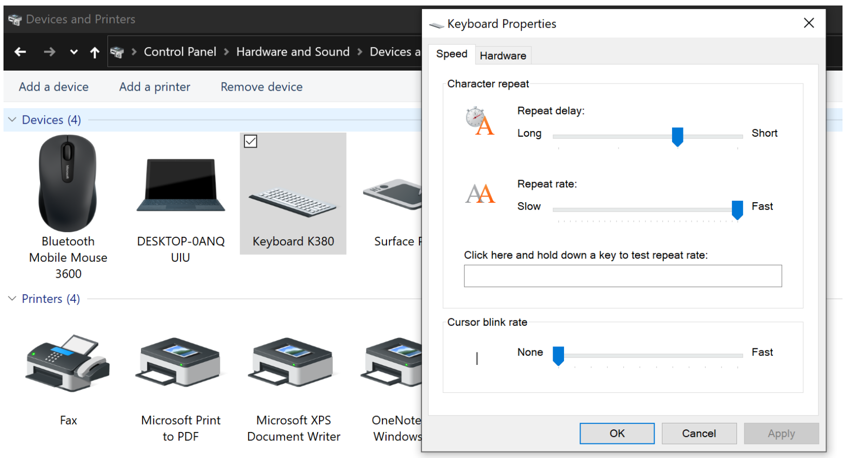
```
"defaults":
{
    // Put settings here that you want to apply to all profiles.
    "cursorShape": "filledBox" ,
    "fontFace": "JetBrains Mono"
}
```

2. Update the home directory

```
{
    "guid": "{c6eaf9f4-32a7-5fdc-b5cf-066e8a4b1e40}",
    "hidden": false,
    "name": "Ubuntu-18.04",
    "source": "Windows.Terminal.Wsl",
    "startingDirectory": "//wsl$/Ubuntu-18.04/home/damonchan"

}
```

3. Disable keyboard cursor blink Disable globally in keyboard setting:



2.3 zsh

I like to use zsh which provides more enhancement and configuration options compared with Bash.

```
sudo apt install zsh
chsh -s $(which zsh)
sh -c "$(curl -fsSL https://raw.githubusercontent.com/robbyrussell/oh-my-zsh/master/tools/install.sh)"
```

2.4 Fonts

Just create a soft link from Windows to Ubuntu:

```
ln -s /mnt/c/Windows/Fonts ~/.fonts
fc-cache -fv
```

2.5 OneDrive

Just link Windows OneDrive Root Directory to Ubuntu:

```
ln -s /mnt/c/Users/elecm/OneDrive ~/OneDrive
```

2.6 X11 Server (VcXsrv)

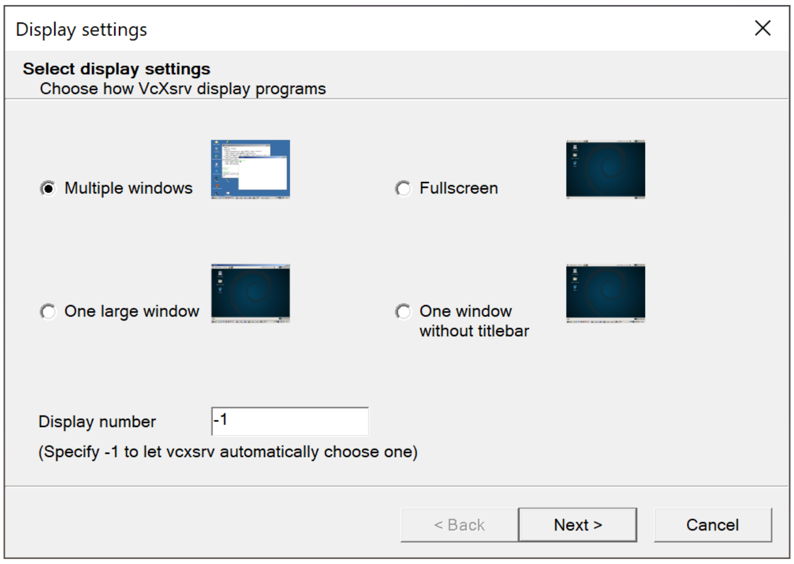
In this tutorial, we choose VcXsrv as X11 server.

2.6.1 Install VcXsrv

```
choco install vcxsrv
```

2.6.2 Configure VcXsrv

1. Start XLunch (VcXsrv)
    - Multiple windows -> Display number: -1 -> Next



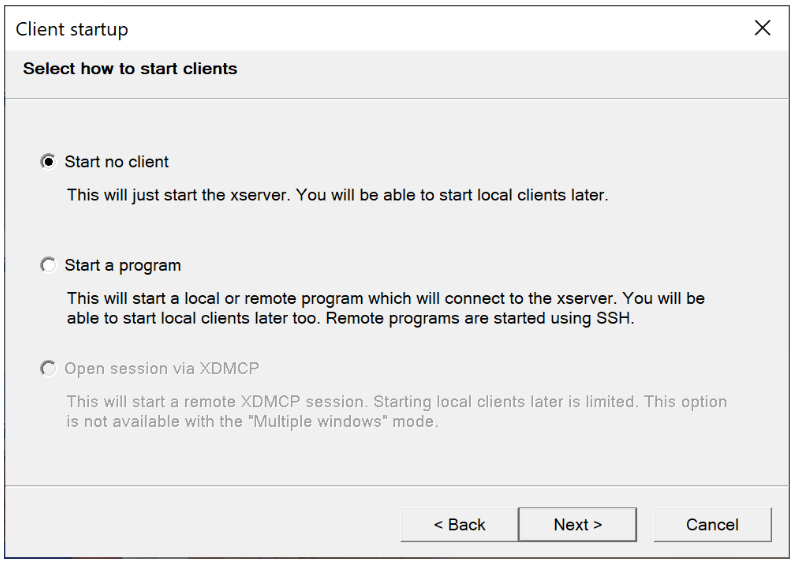
Display settings

Select display settings
Choose how VcXsrv display programs

(● Multiple windows) ○ Fullscreen
○ One large window ○ One window without titlebar

Display number: -1
(Specify -1 to let vcxsrv automatically choose one)

    - Start no client



Client startup

Select how to start clients

(●) Start no client

This will just start the xserver. You will be able to start local clients later.

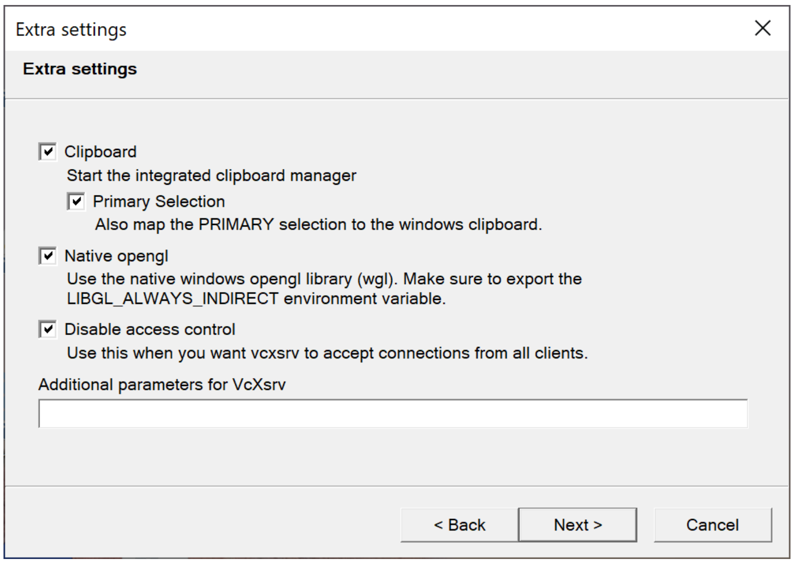
( ) Start a program

This will start a local or remote program which will connect to the xserver. You will be able to start local clients later too. Remote programs are started using SSH.

( ) Open session via XDMCP

This will start a remote XDMCP session. Starting local clients later is limited. This option is not available with the "Multiple windows" mode.

- Tick Clipboard, Primary Selection, Native opengl, Disable access control (If you use `socat` to do X11 forwarding, Access control can be enabled, check 2.6.6 below) -> Next



Extra settings

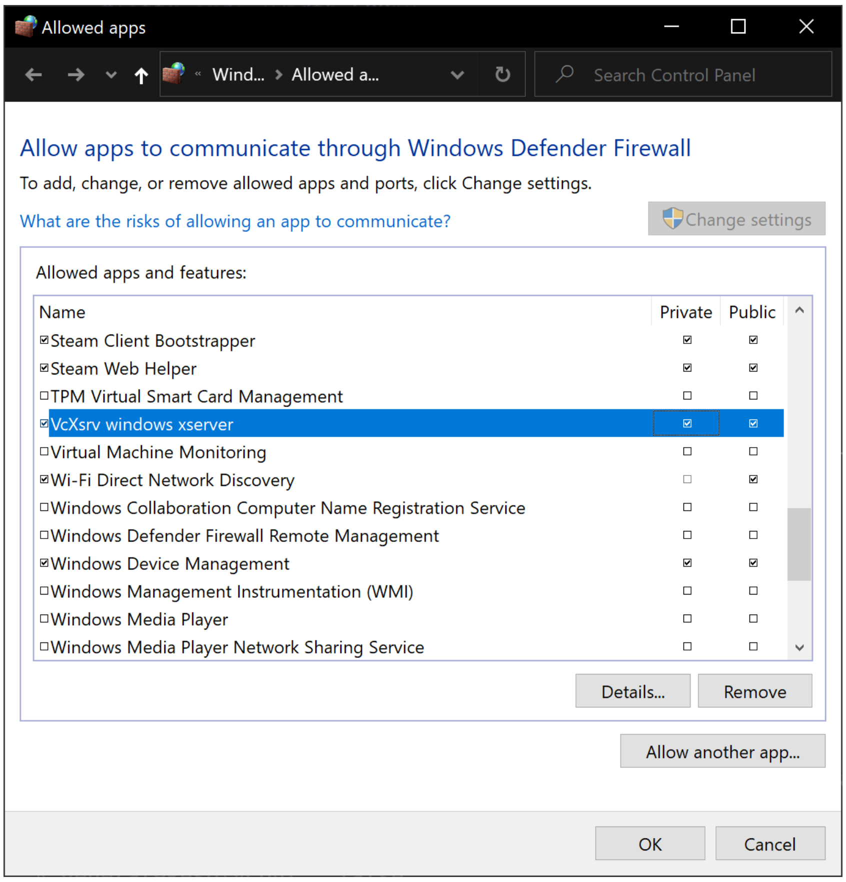
Extra settings

[✓] Clipboard
Start the integrated clipboard manager
  [✓] Primary Selection
  Also map the PRIMARY selection to the windows clipboard.
[✓] Native opengl
Use the native windows opengl library (wgl). Make sure to export the LIBGL_ALWAYS_INDIRECT environment variable.
[✓] Disable access control
Use this when you want vcxsrv to accept connections from all clients.

Additional parameters for VcXsrv

- Save configuration PS:
  - Press "Win + R" -> insert `shell:startup` -> Press Enter
  - Save to startup can start X11 server when system boots
- Finish

2. allow both Private and Public network in windows firewall setting `Control Panel\System and Security\Windows Defender Firewall\Allowed apps`



Allow apps to communicate through Windows Defender Firewall

To add, change, or remove allowed apps and ports, click Change settings.

What are the risks of allowing an app to communicate?

Allowed apps and features:

Name	Private	Public
☑ Steam Client Bootstrapper	☑	☑
☑ Steam Web Helper	☑	☑
☐ TPM Virtual Smart Card Management	☐	☐
☑ VcXsrv windows xserver	☑	☑
☐ Virtual Machine Monitoring	☐	☐
☑ Wi-Fi Direct Network Discovery	☐	☑
☐ Windows Collaboration Computer Name Registration Service	☐	☐
☐ Windows Defender Firewall Remote Management	☐	☐
☑ Windows Device Management	☑	☑
☐ Windows Management Instrumentation (WMI)	☐	☐
☐ Windows Media Player	☐	☐
☐ Windows Media Player Network Sharing Service	☐	☐

2.6.3 Disable `/etc/resolv.conf` generation

We disable `/etc/resolv.conf` generation, so that we can use custom name servers which points to google. `vi /etc/wsl.conf`

```
[network]
generateResolvConf = false
```

2.6.4 Setup custom name servers and point to google

```
vi /etc/resolv.conf
```

```
# This file was automatically generated by WSL. To stop automatic generation of this file, add the followi
# [network]
# generateResolvConf = false
nameserver 8.8.8.8
nameserver 8.8.4.4
```

2.6.5 Export DISPLAY and $LIBGL_{ALWAYSINDIRECT}$ settings to `~/.zshrc`

```
vi ~/.zshrc
```

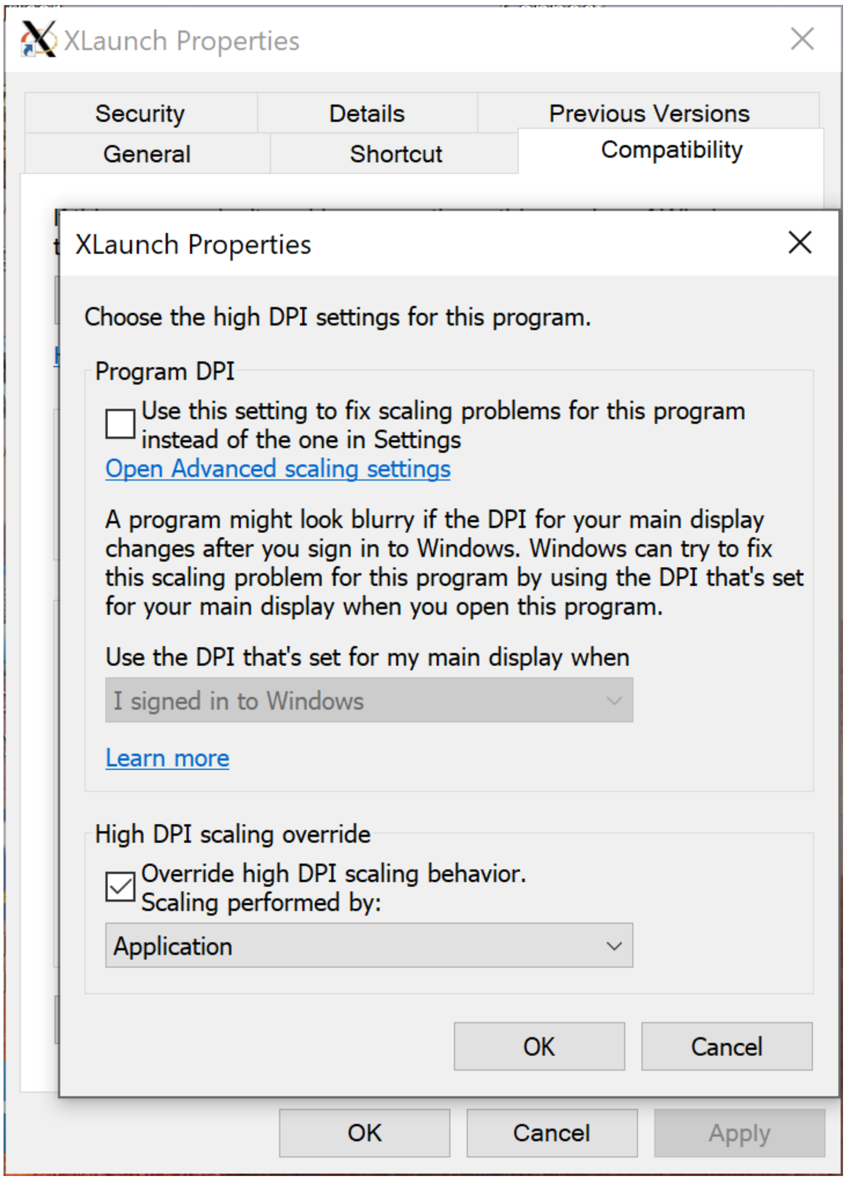
```
export DISPLAY=$(ip route | awk '{print $3; exit}'):0
export LIBGL_ALWAYS_INDIRECT=1
```

2.6.6 Other Methods

There is another way to enable the `X11` forwarding through socat, check this github issue. It is more safe but has some performance lost.

2.6.7 HiDPI

1. Right click `XLaunch` (VcXsrv) -> Compatibility -> Change high DPI settings -> Tick Override high DPI scaling Behavior, Application.


XLaunch Properties

Choose the high DPI settings for this program.

Program DPI

☐ Use this setting to fix scaling problems for this program instead of the one in Settings

Open Advanced scaling settings

A program might look blurry if the DPI for your main display changes after you sign in to Windows. Windows can try to fix this scaling problem for this program by using the DPI that's set for your main display when you open this program.

Use the DPI that's set for my main display when: I signed in to Windows

Learn more

High DPI scaling override

☑ Override high DPI scaling behavior. Scaling performed by: Application

2. Add the following statement to `.zshrc`

```
export GDK_SCALE=2
```

2.6.8 Bigger Cursor

Supposed VcXsrv is installed in `C:\Program Files\VcXsrv\`,

1. Install big-cursor

```
sudo apt install big-cursor
```

2. Rename `C:\Program Files\VcXsrv\fonts\misc\cursor.pcf.gz` to `C:\Program Files\VcXsrv\fonts\misc\cursor-small.pcf.gz`
3. Copy `/usr/share/fonts/X11/misc/big-cursor.pcf.gz` from WSL to as `C:\Program Files\VcXsrv\fonts\misc\cursor.pcf.gz`

2.7 Setup Emacs

Let's dive into the meat of Compiling and installing the latest **Emacs 27.1**!

2.7.1 install dependencies

```
sudo apt install -y autoconf automake autotools-dev bsd-mailx build-essential \
    diffstat gnutls-dev imagemagick libasound2-dev libc6-dev libdatrie-dev \
    libdbus-1-dev libgconf2-dev libgif-dev libgnutls28-dev libgpm-dev libgtk2.0-dev \
    libgtk-3-dev libice-dev libjpeg-dev liblockfile-dev liblqr-1-0 libm17n-dev \
    libmagickwand-dev libncurses5-dev libncurses-dev libotf-dev libpng-dev \
    librsvg2-dev libsm-dev libthai-dev libtiff5-dev libtiff-dev libtinfo-dev libtool \
    libx11-dev libxext-dev libxi-dev libxml2-dev libxmu-dev libxmuu-dev libxpm-dev \
    libxrandr-dev libxt-dev libxtst-dev libxv-dev quilt sharutils texinfo xaw3dg \
    xaw3dg-dev xorg-dev xutils-dev zlib1g-dev libjansson-dev libxaw7-dev \
    libselinux1-dev libmagick++-dev libacl1-dev gir1.2-javascriptcoregtk-4.0 \
    gir1.2-webkit2-4.0 libenchant1c2a libglvnd-core-dev libicu-le-hb-dev \
    libidn2-0-dev libjavascriptcoregtk-4.0-dev liboss4-salsa2 libsoup2.4-dev \
    libsystemd-dev libwebkit2gtk-4.0-dev libx11-xcb-dev libxcb-dri2-0-dev \
    libxcb-dri3-dev libxcb-glx0-dev libxcb-present-dev libxshmfence-dev \
    x11proto-composite-dev x11proto-core-dev x11proto-damage-dev \
    x11proto-fixes-dev
```

2.7.2 Download, compile and install

```
cd ~
wget https://ftp.gnu.org/pub/gnu/emacs/emacs-27.1.tar.gz
tar -xzvf emacs-27.1.tar.gz
cd emacs-27.1
./configure
make
sudo make install
rm ~/emacs-27.1.tar.gz
```

2.7.3 Install doom

```
git clone --depth 1 https://github.com/hlissner/doom-emacs ~/.emacs.d
~/.emacs.d/bin/doom install
```

2.7.4 Start Emacs

```
emacs
```

Please notice:

1. The first time to start Emacs may need some times (Every time the first time to start Emacs after system boots, it also needs some time at times), font setting, `x11` checking, sound checking etc, please wait a moment. Normally it will finish within one or two minute.
2. After refreshing font setting through `fc-cache -fv`, Emacs will take some time to configure the font setting, but it will only conduct one time.
3. The second time to start Emacs will resume to the normal startup time.

2.8 Fix WSL$_{\text{INTEROP}}$ issue

When you start Emacs, it will create another `interop` file, it make us can not start windows programs in Emacs, we can make it to use the one same `interop` file with the terminal.

Add the following to `.zshrc`

```
# fix interop
fix_wsl2_interop() {
for i in $(pstree -np -s $$ | grep -o -E ’[0-9]+’); do
if [[ -e "/run/WSL/${i}_interop" ]]; then
export WSL_INTEROP=/run/WSL/${i}_interop
fi
done
}

~/.emacs.d/bin/doom env > /dev/null 2>&1
```

2.9 Optional: Setup Python Development Environment

2.9.1 pyenv

```
sudo apt-get install git gcc make openssl libssl-dev libbz2-dev libreadline-dev libsqlite3-dev libffi-dev
git clone https://github.com/pyenv/pyenv.git ~/.pyenv
echo ’export PYENV_ROOT="$HOME/.pyenv"’ >> ~/.zshrc
echo ’export PATH="$PYENV_ROOT/bin:$PATH"’ >> ~/.zshrc
echo -e ’if command -v pyenv 1>/dev/null 2>&1; then\n  eval "$(pyenv init -)"\nfi’ >> ~/.zshrc
pyenv install 3.7.9
pyenv global 3.7.9
```

2.9.2 mypyls

mspyls can be installed by typing `M-x lsp-install-server RET mspyls`.

2.10 Optional: Setup Node.js Development Environment

2.10.1 node

Install nvm

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash
```

Add to .zshrc

```
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This loads nvm bash_completion
```

```
nvm install --lts
nvm use --lts
npm i -g javascript-typescript-langserver
.emacs.d/bin/doom sync
```

2.11 Optional: Setup Rust Development Environment

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
rustup component add rust-src
sudo curl -L https://github.com/rust-analyzer/rust-analyzer/releases/latest/download/rust-analyzer-linux -
sudo chmod +x /usr/bin/rust-analyzer
cargo install cargo-check
rustup component add clippy-preview
```

2.12 Optional: Install other useful packages

2.12.1 apt

With apt

```
sudo apt install calibre sqlite3 pandoc
```

2.12.2 pdf-tools

```
sudo apt install libpng-dev zlib1g-dev libpoppler-glib-dev libpoppler-private-dev
M-x pdf-tools-install
```

2.12.3 rga

```
sudo apt install build-essential pandoc poppler-utils ffmpeg
wget https://github.com/phiresky/ripgrep-all/releases/download/v0.9.6/ripgrep_all-v0.9.6-x86_64-unknown-li
tar -zxvf ripgrep_all-v0.9.6-x86_64-unknown-linux-musl.tar.gz
cd ripgrep_all-v0.9.6-x86_64-unknown-linux-musl
sudo cp rga /usr/bin
sudo cp rga-preproc /usr/bin
```

2.12.4 Rime

1. Install and setup Rime

   ```
   sudo apt install git build-essential cmake libboost-all-dev libgoogle-glog-dev libleveldb-dev libmari
   cd /usr/src/gtest
   sudo cmake CMakeLists.txt
   sudo make
   #copy or symlink libgtest.a and libgtest_main.a to your /usr/lib folder
   ```

```
sudo cp *.a /usr/lib
cd ~/.emacs.d/librime
make
sudo make install
sudo apt-get install ibus-rime
```

PS: It is better to not share Rime User folder between windows and WSL, it may cause troubles.

In Emacs

```
M-x rime-compile-module
```

2. plum

```
curl -fsSL https://git.io/rime-install | bash
# rime_dir="$HOME/.rime" bash rime-install
```

2.12.5 sdcv

```
sudo apt install stardict sdcv
```

2.12.6 telega

```
sudo apt install gperf
git clone https://github.com/tdlib/td.git
cd td
mkdir build && cd build && cmake ../
make
sudo make install
git clone https://github.com/zevlg/telega.el
cd telega.el
make && make install
```

2.13 An .zshrc example

```
# If you come from bash you might have to change your $PATH.
# export PATH=$HOME/bin:/usr/local/bin:$PATH

# Path to your oh-my-zsh installation.
export ZSH="/home/damonchan/.oh-my-zsh"

# Set name of the theme to load --- if set to "random", it will
# load a random theme each time oh-my-zsh is loaded, in which case,
# to know which specific one was loaded, run: echo $RANDOM_THEME
# See https://github.com/ohmyzsh/ohmyzsh/wiki/Themes
ZSH_THEME="robbyrussell"

# Set list of themes to pick from when loading at random
# Setting this variable when ZSH_THEME=random will cause zsh to load
# a theme from this variable instead of looking in $ZSH/themes/
# If set to an empty array, this variable will have no effect.
# ZSH_THEME_RANDOM_CANDIDATES=( "robbyrussell" "agnoster" )

# Uncomment the following line to use case-sensitive completion.
# CASE_SENSITIVE="true"

# Uncomment the following line to use hyphen-insensitive completion.
# Case-sensitive completion must be off. _ and - will be interchangeable.
# HYPHEN_INSENSITIVE="true"

# Uncomment the following line to disable bi-weekly auto-update checks.
# DISABLE_AUTO_UPDATE="true"

# Uncomment the following line to automatically update without prompting.
# DISABLE_UPDATE_PROMPT="true"

# Uncomment the following line to change how often to auto-update (in days).
# export UPDATE_ZSH_DAYS=13

# Uncomment the following line if pasting URLs and other text is messed up.
# DISABLE_MAGIC_FUNCTIONS="true"

# Uncomment the following line to disable colors in ls.
# DISABLE_LS_COLORS="true"

# Uncomment the following line to disable auto-setting terminal title.
# DISABLE_AUTO_TITLE="true"

# Uncomment the following line to enable command auto-correction.
# ENABLE_CORRECTION="true"

# Uncomment the following line to display red dots whilst waiting for completion.
# COMPLETION_WAITING_DOTS="true"

# Uncomment the following line if you want to disable marking untracked files
# under VCS as dirty. This makes repository status check for large repositories
# much, much faster.
# DISABLE_UNTRACKED_FILES_DIRTY="true"

# Uncomment the following line if you want to change the command execution time
# stamp shown in the history command output.
# You can set one of the optional three formats:
# "mm/dd/yyyy"|"dd.mm.yyyy"|"yyyy-mm-dd"
# or set a custom format using the strftime function format specifications,
# see 'man strftime' for details.
# HIST_STAMPS="mm/dd/yyyy"
# Would you like to use another custom folder than $ZSH/custom?
# ZSH_CUSTOM=/path/to/new-custom-folder

# Which plugins would you like to load?
# Standard plugins can be found in $ZSH/plugins/
# Custom plugins may be added to $ZSH_CUSTOM/plugins/
# Example format: plugins=(rails git textmate ruby lighthouse)
# Add wisely, as too many plugins slow down shell startup.
plugins=(git)

source $ZSH/oh-my-zsh.sh

# User configuration

# export MANPATH="/usr/local/man:$MANPATH"

# You may need to manually set your language environment
# export LANG=en_US.UTF-8

# Preferred editor for local and remote sessions
# if [[ -n $SSH_CONNECTION ]]; then
#   export EDITOR='vim'
# else
#   export EDITOR='mvim'
# fi

# Compilation flags
# export ARCHFLAGS="-arch x86_64"

# Set personal aliases, overriding those provided by oh-my-zsh libs,
# plugins, and themes. Aliases can be placed here, though oh-my-zsh
# users are encouraged to define aliases within the ZSH_CUSTOM folder.
# For a full list of active aliases, run `alias`.
#
# Example aliases
# alias zshconfig="mate ~/.zshrc"
# alias ohmyzsh="mate ~/.oh-my-zsh"

#export DISPLAY=:0
export DISPLAY=$(ip route | awk '{print $3; exit}'):0
export LIBGL_ALWAYS_INDIRECT=1

export PYENV_ROOT="$HOME/.pyenv"
export PATH="$PYENV_ROOT/bin:$PATH"
if command -v pyenv 1>/dev/null 2>&1; then
  eval "$(pyenv init -)"
fi
```

```
alias em="emacsclient -nw"

#export NVM_DIR="$HOME/.nvm"
#[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
#[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This loads nvm bash_completion
export GDK_SCALE=2

# fix interop
fix_wsl2_interop() {
for i in $(pstree -np -s $$ | grep -o -E '[0-9]+'); do
if [[ -e "/run/WSL/${i}_interop" ]]; then
export WSL_INTEROP=/run/WSL/${i}_interop
fi
done
}

~/.emacs.d/bin/doom env > /dev/null 2>&1
```

2.14 Export WSL (Backup WSL)

wsl ships with a –export option for users to do export the WSL distro to a tar file which can be imported to other machines:

```
wsl --export Ubuntu-18.04 Ubuntu-18.04_20200905
```

3 An Emacs Shortcut

Create two file in desktop: `Emacs.sh` and `Emacs.bat`

3.1 Emacs.sh

```
cd ~
export DISPLAY=$(ip route | awk '{print $3; exit}'):0
export LIBGL_ALWAYS_INDIRECT=1

export PYENV_ROOT="$HOME/.pyenv"
export PATH="$PYENV_ROOT/bin:$PATH"
if command -v pyenv 1>/dev/null 2>&1; then
    eval "$(pyenv init -)"
fi

alias em="emacsclient -nw"

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This loads nvm bash_completion
# export GDK_SCALE=2
emacs
```

3.2 Emacs.bat

```
@echo off
wsl ./emacs.sh
```

3.3 Double click `Emacs.bat` to launch Emacs.

4 Some Emacs hacking ideas

It is very easy to use Emacs to interactive with Windows’s programs, such as browsing the URL with Chrome, open the PDF file with Acrobat Reader DC, open the current file with default program, launch `explorer.exe`, etc. Here are some ideas:

4.1 Browser URL with default browser

```
(defun wsl-browse-url-xdg-open (url &optional ignored)
  (interactive (browse-url-interactive-arg "URL: "))
  (shell-command-to-string (concat "explorer.exe " url)))
(advice-add #’browse-url-xdg-open :override #’wsl-browse-url-xdg-open)
```

4.2 If you a calibredb user, you can add the following advice to open the PDF/EPUB with windows default programs

```
;; calibredb
(defun wslcalibredb-open-with-default-tool (filepath)
  (shell-command-to-string
   (concat "cd " (shell-quote-argument (file-name-directory (expand-file-name filepath))) " && "
           (concat "cmd.exe /C start ’’ \"${@//&/^&}\" " (shell-quote-argument (file-name-nondirectory fil
(advice-add #’calibredb-open-with-default-tool :override #’wslcalibredb-open-with-default-tool)
```

4.3 Open the file with windows default programs or reveal it in explorer

```
;;;###autoload

(defmacro wsl--open-with (id &optional app dir)
  ‘(defun ,(intern (format "wsl/%s" id)) ()
     (interactive)
     (wsl-open-with ,app ,dir)))

(defun wsl-open-with (&optional app-name path)
  "Send PATH to APP-NAME on WSL."
  (interactive)
  (let* ((path (expand-file-name
                (replace-regexp-in-string
                 "’" "\\’"
                 (or path (if (derived-mode-p ’dired-mode)
                              (dired-get-file-for-visit)
                            (buffer-file-name)))
                 nil t)))
         (command (format "%s ‘wslpath -w %s‘" (shell-quote-argument app-name) path)))
    (shell-command-to-string command)))

(wsl--open-with open-in-default-program "explorer.exe" buffer-file-name)
(wsl--open-with reveal-in-explorer "explorer.exe" default-directory)
```

```
M-x wsl/open-in-default-program
M-x wsl/reveal-in-explorer
```

5 References

- WSL 2 Support is coming to Windows 10 Versions 1903 and 1909 | Windows Comman...
- Ubuntu on WSL 2 Is Generally Available | Ubuntu
- `https://github.com/microsoft/WSL/issues/5336`
- `https://en.wikipedia.org/wiki/Windows_Subsystem_for_Linux`
- `https://docs.microsoft.com/en-us/windows/wsl/wsl2-index`
- `https://docs.microsoft.com/en-us/windows/wsl/compare-versions`
- `https://github.com/microsoft/terminal/issues/1379`
- `https://github.com/microsoft/terminal`
- `https://docs.microsoft.com/en-us/windows/terminal/customize-settings/profile-settings`
- `https://stackoverflow.com/questions/61110603/how-to-set-up-working-x11-forwarding-on-wsl2`
- `https://superuser.com/questions/1196399/how-do-i-set-the-size-of-the-x-mouse-pointer-in-the-windows-`
- `https://github.com/hubisan/emacs-wsl`
- `https://stackoverflow.com/questions/27022373/python3-importerror-no-module-named-ctypes-when-using-va`
- `https://docs.microsoft.com/en-us/windows/nodejs/setup-on-wsl2`
- `https://github.com/microsoft/WSL/issues/5065`
- `https://docs.microsoft.com/en-us/windows/wsl/interop`