Materials: Configuration

ppppass

January 22, 2018

The information is updated on January 21, 2018.

1 Environments

It is suggested to use a Unix-like environment when programming even if you have a Windows system. Reasons includes:

(1) There are differences between Unix-like environments and Windows, including line feed (`\n` in Linux but `\n\r` in Windows), path separator (`/` in Linux and `\` in Windows) and so on.

(2) Some packages either do not work well or are hard to configure in Windows, especially large deep learning packages TensorFlow and PyTorch. On the contrary, there are thorough tutorials on configuration in Ubuntu (and maybe macOS).

(3) Tool chains are broken up in Windows. That is, Git, TeX Live, Python (or say Anaconda), MSYS2 (or PuTTY) all provide different environments in Windows, where an unified tool is provided in Linux environments.

(4) Getting accustomed to shells and terminals benefits because you may login to remote servers at some point.

However, there are no absolutely correct or easy way to setup a Unix-like environment for Windows users. Several known approaches to such environments for Windows users are listed below, together with the disadvantages.

(1) Buy an Apple computer with macOS. (Expensive)

(2) Switch to a Linux system. (Time-expensive with backups to be made)

(3) Set up a dual system. (Probably unstable, with corruptions related to the dual system have happened before)

(4) Install a virtual machine, and use the graphical interface. (Slow, especially the graphical interface, while giving up the graphical interface and only making use of terminals (something like `tty2`) is also a choice)

(5) Use a terminal emulator in Windows. (Not fully functional)

(6) Install a virtual machine and use a SSH client (such as PuTTY) to perform remote login into the virtual machine. (No programming support in windows)

(7) Install a virtual machine and use a terminal emulator in Windows to perform remote login into the virtual machine. (Troublesome with a huge tool chain)

(8) Use Ubuntu directly in Windows 10 from Microsoft Store. (Not popular)

For a virtual machine, VirtualBox is recommended, which is a free virtual machine program by Oracle. Note that VMWare is another famours virtual machine software, but there are license issues. Additionally, VirtualBox provides better support for Linux systems, while VMWare is better for Windows systems or heavy tasks.

Downloads of VirtualBox can be found this page. Documentation can be found here, where a `.pdf` User Manual is provided.

For Linux distributions, Ubuntu for desktops is a ready-to-use operating system for newcomers to Linux. Ubuntu 16.04.3 LTS is more stable but somehow backward, while Ubuntu 17.10 is on the contrary. By the way, Arch Linux is another Linux distribution with high flexibility, full wiki support and a wonderful rolling update mechanism, but its installation needs some knowledge about operating system, and is not recommended to beginners.

Downloads of Ubuntu can be found here. There are easy-to-understand instructions during installation.

Installation guide of Ubuntu on VirtualBox can be easily retrieved through search engine, and an example installation guide in Chinese is provided here. (Note that some advanced settings, after section 4.2 inclusive, are fully optional)

Ubuntu officially provides a tutorial *Install Ubuntu on Windows 10*.

For a terminal emulator in Windows, MSYS2 and Cygwin are two choices, which both provide some Linux utilities in Windows environment. (I (pppppass) have adopted MSYS2 and have no knowledge about Cygwin)

Downloads of MSYS2 can be found here and a small installation guide is also included. Further tutorials can also be found on Internet.

For Linux and Mac OS users, nothing is needed but just opening a terminal.

A Chinese introduction to Linux command lines is shown here. NIH also provides a course

Introduction to Linux to tell basic GNU and Linux concepts. The first 32 pages is adequate for a start in Linux.

SSH, Vim and Make, which are three important tools, are always included in Unix-like environments. If not, you may install it through your package manager.

To perform remote login, a SSH key is needed to be generated. This is described in Xuefeng Liao's *Git tutorial* in the section *Remote repositories*, as well as in *Git official tutorial* in section 4.3 *Git on the Server — Generating Your SSH Public Key*. If you want to permit remote logins into your system, a SSH service is also required, which is described here. The command `ssh-add` creates a daemon of related keys and frees people from repeating input the passphrase, which is described here and here.

A famous post in Stack Overflow describes how to exit the Vim editor, which puzzles thousands of sophisticated programmers. Complete introduction to Vim are given by this post and this website.

Make is a automatic build tool, which turn complicated build commands into a simple command `make`. Tutorials on Make can be easily found on the Internet, among which Yifeng Ruan's *Tutorial on Make* and *Make (software)* on Wikipedia give useful information.

2 Anaconda

It is strongly recommended to install and use Anaconda in a Unix-like environment, which organizes different environments among versions of packages, and free people from version conflicts.

Anaconda can be directly downloaded from official website. During the installation, not to add the path of Anaconda into `~/.bashrc` is preferred, in order to avoid overriding Python utilities in the original system. However, in this way, you have to explicitly use `source some/path/to/anaconda/bin/activate` to activate the Anaconda.

A great introduction to the usage of Anaconda is given in Anaconda's User Guide. The section *Getting started* shows basic usage of the command `conda`, and the section *Cheet sheet* gives a list of `conda` commands. A Chinese guide to Anaconda is given in this post.

You may turn to Anaconda Cloud to search for non-standard package, e.g. MOSEK, Gurobi, CVXPY and many more.

3 Jupyter Notebook

It is strongly recommended to install and use Jupyter Notebook in Anaconda, which provides an interactive notebook environment base on web pages. Jupyter Notebook is widely used in tutorials, where live codes and visualization are important.

In Anaconda, you can install jupyter notebook by `conda install jupyter`. Before installation, please check the Anaconda environment you are in to avoid version conflicts.

You may consult *Running the Notebook* to run a Jupyter Notebook. For public access, you may generate a config file as shown in *Configuration Overview*, and then set password as *Running a Notebook Server*. Using SSL for encryption takes some time and is not much necessary is you run the Notebook in local network.

An offical tutorial of Jupyter Notebook *Jupyter Notebook Quickstart* is also provided. Only the *Running the Notebook* section is in necessity. A brief tutorial on remote access of Jupyter Notebook in Chinese is provided in this post.

A complete introduction to IPython and Jupyter Notebook is given here. Jupyter Notebook is also introduced here and here , while this post and this post are Chinese translations.

IPython, an interactive and enhanced Python prompt is also useful when testing code snippets. IPython can be easily installed in Anaconda environment by `conda install ipython`, and is already installed if you have installed Jupyter Notebook. An introduction to IPython *Introduction IPython* is provided, while magic commands are mentioned in the section *Magic functions*. These magic functions are also available in Jupyter Notebook, where two of the most important ones are `%matplotlib notebook` (show interactive plots) and `%matplotlib inline` (show inline static plots).

Jupyter Notebook is a versatile tool for interactive Python programming, and further information can be found in Jupyter Notebook document and IPython document.

4 Editors and IDEs

Some recommended editors and IDEs are listed below. Please help expand this section by making pulls and pull requests.

Text editors:

(1) Vim, which is a text editors provided in most Unix-like systems, and therefore may be the unique text editor available in command lines. Vim are quite “abstract” and filled with shortcuts, which makes it hard to learn but efficient to use once learned.

(2) Nano, which is another command line text editor in Unix-like systems. Nano is more "intuitive" but less capable than Vim.

(3) Notepad++, which is a text editor in Windows system. Notepad++ is light-weighted, and useful in handling data files. However, Notepad++ users suffer from its auto completion, which makes it less capable for coding.

Integrated text editor for coding:

(1) Atom, which is based on open source community. Its plug-in features contributes to its functionality, which also caused a very slow loading speed. Atom have a great compatibility with Git.

(2) Visual Studio Code, which is also a open source community-based project with plug-ins. Developed by Microsoft, VS Code have great code completion and prompting mechanism to codes. However, because of the age, VS Code is more immature than Atom and suffers from complicated adaption process in different environments.

Integrated text editor for TEX-ing:

(1) TeXworks, which is a TEXeditor shipped with TeX Live. TeXworks is light-weighted and capable for typesetting small documents. The documentation in English lies in the path `.../tlpkg/texworks/texworks-help/TeXworks-manual/en/TeXworks-manual-en.pdf`, and information about spell check dictionaries can be found here.

Note that some integrated editors, namely Atom and VS Code, both support TEXplug-ins and provide a great environment to typeset with latex.