This tutorial provides instructions on how to install LaTeX and Visual Studio Code on different systems and configure them for use.

This is the British English, website version of the tutorial.

Install LaTeX

A recommended approach to install LaTeX is to install TeX Live. Full guide can be found on its official website.

A brief induction here:

- For **Windows**, it is recommended to download ISO Image. Mount the image and install via running the installer inside.
- For **RHEL Linux Distribution** (e.g. RHEL, CentOS Stream, Rocky Linux, AlmaLinux, etc.), it is also good to install via ISO Image.
- For **Debian Linux Distribution** (e.g. Debian, Ubuntu, Kubuntu, Xubuntu, Linux Mint, etc.), can just run `sudo apt install texlive-full` in terminal to install.
- For **MacOS & MacOSX**, it is recommended to install MacTeX, which has full content of TeX Live, and some extra content especially for Mac users.

Although TeX Live official recommends to install with online installer, but unless your internet environment is good enough, install with ISO Image can usually save your time.

Once you installed TeX Live, future update can be run with `tlmgr update --all` If you want cross-version (e.g. from TeX Live 2024 to TeX Live 2025), it is better to uninstall/remove old version and directly install the new version. In generall, TeX Live almost do not need update or upgrade.

Install LaTeX on Windows

Mount the ISO Image mentioned above by double-click it. Switch to Image path, and run the file `install-tl-Windows.bat`. Generally, just specify the installation location, and keep all other settings as default is enough.

Run following commands in terminal, check if they successfully show version detail to double-check wether installation is successful:

```
xelatex --version
pdflatex --version
lualatex --version
latexmk --version
```

install LaTeX in RHEL Linux Distributions

Mount the ISO Image mentioned above by double-click it. Switch to Image path, right click to open terminal there. Then, run following commands:

```
sudo perl ./install-tl --no-interaction
```

Type your password, and then the installer will start installation as adminstrator.

After installation, it is necessary to add TeX Live path to PATH for system and other software's use. One reliable apprach is to change environment variables of current user via `sudo nano ~/.bashrc`. Add new line at the end with these content:

```
export PATH=<TeX Live Installation Path>:$PATH
export MANPATH=/usr/share/man:<man Path of TeX Live>:$MANPATH
export INFOPATH=<info Path of TeX Live>:$INFOPATH
```

For instance, when installing on Inter x86-64 machine, the content should be:

```
export PATH=$PATH:/usr/local/texlive/2025/bin/x86_64-linux:$PATH
export MANPATH=/usr/share/man:/usr/local/texlive/2025/texmf-dist/doc/man:$MANPATH
export INFOPATH=/usr/local/texlive/2025/texmf-dist/doc/info:$INFOPATH
```

Then, run `source ~/.bashrc` to refresh environment variables, and install necessary perl packages with `sudo dnf install perl-core perl-Time-HiRes perl-Unicode-Normalize perl-LWP-Protocol-https`, check if they successfully show version detail to double-check wether installation is successful:

```
xelatex --version
pdflatex --version
lualatex --version
latexmk --version
```

Install LaTeX in Debian Linux Distributions

Run `sudo apt install texlive-full` in terminal to install. check if they successfully show version detail to double-check wether installation is successful:

```
xelatex --version
pdflatex --version
lualatex --version
latexmk --version
```

Install Visual Studio Code

Due to limitation of FlatPak (or Snap), it is the best to install text editors for codings and IDEs like Visual Studio Code (VSCode, VSC) and JetBrains products directly on Linux system. Thus, following induction will also guide how to install Visual Studio Code to Linux directly.

Install Visual Studio Code on Windows

There exists two different approaches to insall Visual Studio Code on Windows: via **Microsoft Store** and via **Installer**.

Install Visual Studio Code With Microsoft Store

If you have already logged in with your Microsoft account on your Windows machine, then install via **Microsoft Store** might be the simplist way:

1. Open Microsoft Store.
2. Search for Visual Studio Code.
3. Choose the Visual Studio Code with blue icon, or the green one (Insider version) if you would like to enjoy new features.
4. Click "Install" and wait for everything to be done!

Install Visual Studio Code With Installer

Install with installer is a traditional approach, which provides you some customisation to the Installation process.

1. Visit the download page of Visual Studio Code.
2. Choose the installer for Windows. If you want more customisation, choose "User Installer" or "System Installer" below, **Remember to choose the installer that maches your CPU architecture!**
3. Run the downloaded installer, install following its guidance.

Install Visual Studio Code in RHEL Linux Distributions

In RHEL distributions, you can follow following instructions to add VSCode to your software sources, for easier installation and updates.

First, add repository and GPG key of VSCode to your system:

```
sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc &&
echo -e "[code]\nname=Visual Studio
Code\nbaseurl=https://packages.microsoft.com/yumrepos/vscode\nenabled=1\nautorefresh
=1\ntype=rpm-
md\ngpgcheck=1\ngpgkey=https://packages.microsoft.com/keys/microsoft.asc" | sudo tee
/etc/yum.repos.d/vscode.repo > /dev/null
```

If you are using newer system (for Fedora, Fedora 22 or newer), install VSCode with `dnf`:

```
dnf check-update
sudo dnf install code # code-insiders if you want VSCode Insider
```

If you are using older system, then you need to install with `yum`:

```
yum check-update
sudo yum install code # code-insiders if you want VSCode Insider
```

Install Visual Studio Code in Debian Linux Distributions

In Debian distributions, you can follow following instructions to add VSCode to your software sources, for easier installation and updates.

First, run following command in terminal to install GPG key:

```
sudo apt-get install wget gpg &&
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor >
microsoft.gpg &&
sudo install -D -o root -g root -m 644 microsoft.gpg
/usr/share/keyrings/microsoft.gpg &&
rm -f microsoft.gpg
```

Then, create a new source file with `sudo nano /etc/apt/sources.list.d/vscode.source`{.bash}. With following content, VSCode repository can be added to system software sources:

```
Types: deb
URIs: https://packages.microsoft.com/repos/code
Suites: stable
Components: main
Architectures: amd64,arm64,armhf
Signed-By: /usr/share/keyrings/microsoft.gpg
```

Finally, update software source cache, and install VSCode:

```
sudo apt install apt-transport-https && sudo apt update # Update software
source cache
sudo apt install code # code-insiders if you want VSCode Insider
```

Configuring Visual Studio Code

After installing VSCode, it can be found among the installed software menu of your machine. After setting up language, theme, account and synchronisation settings, we can moving forward to next step: **configuration especially for LaTeX**.

LaTeX Workshop

LaTeX Workshop is a classic extension for LaTeX on VSCode, prividing preview, compiling options management, autocomplete, Keywords highlighting, and many other features. It is the main reason choosing VSCode other than other editors to write and build LaTeX documents.

To install LaTeX Workshop, open the `Extension` option on the left (or keyboard shortcut `Ctrl+Shift+X`), Search for latex Workshop, and click `install`.

After installation, click the gear button and select `Settings` to enter its configuration part.

As a configuration template can be provided, configuration is not that complex. Append `tools` to content exists in search bar, and click `Edit in settings.json` of the first result. By paste following content to the end of the file, you can get basically good experience:

```
"latex-workshop.latex.autoBuild.run": "never",  // Do not automically compile
when saving
"latex-workshop.showContextMenu": true, // Enable LaTeX context menu
"latex-workshop.intellisense.package.enabled": true,    // Automacally complete
command and package based on the packages loaded
"latex-workshop.message.error.show": false,
"latex-workshop.message.warning.show": false,
"latex-workshop.latex.recipe.default": "lastUsed",  // Use last recipe used as
default recipe
"latex-workshop.view.pdf.internal.synctex.keybinding": "double-click",
// Command groups will be used
"latex-workshop.latex.tools": [
    {
        // Latexmk with default settings from extension
        "name": "latexmk",
        "command": "latexmk",
        "args": [
            "-synctex=1",
            "-interaction=nonstopmode",
            "-file-line-error",
            "-pdf",
            "-outdir=%OUTDIR%",
            "%DOCFILE%"
        ]
    },
    {
        // Compile completely based on .latexmkrc file
        "name": "latexmk_auto",
        "command": "latexmk",
        "args": []
```

```
},
{
    // Clean up based on .latexmkrc file
    "name": "latexmk_auto_clean",
    "command": "latexmk",
    "args": [
        "-c"
    ]
},
{
    // LuaLaTeX
    "name": "lualatex",
    "command": "lualatex",
    "args": [
        "-synctex=1",
        "-interaction=nonstopmode",
        "-file-line-error",
        "%DOCFILE%"
    ]
},
{
    // XeLaTeX
    "name": "xelatex",
    "command": "xelatex",
    "args": [
        "-synctex=1",
        "-interaction=nonstopmode",
        "-file-line-error",
        "%DOCFILE%"
    ]
},
{
    // PDFLaTeX
    "name": "pdflatex",
    "command": "pdflatex",
    "args": [
        "-synctex=1",
        "-interaction=nonstopmode",
        "-file-line-error",
        "%DOCFILE%"
    ]
},
{
    // Biber
    "name": "biber",
    "command": "biber",
```

```
        "args": [
            "%DOCFILE%"
        ]
    },
    {
        // BibTeX
        "name": "bibtex",
        "command": "bibtex",
        "args": [
            "%DOCFILE%"
        ]
    }
],
"latex-workshop.latex.recipes": [
    {
        // Compile completely based on .latexmkrc file settings
        "name": "LaTeXmk (auto)",
        "tools": [
            "latexmk_auto"
        ]
    },
    {
        // Clean up supportive files for the command above
        "name": "LaTeXmk Clean (auto)",
        "tools": [
            "latexmk_auto_clean"
        ]
    },
    // Different compilers with Biber, for documents with both bibliography and
table of contents
    {
        "name": "lualatex -> biber -> lualatex*2",
        "tools": [
            "lualatex",
            "biber",
            "lualatex",
            "lualatex"
        ]
    },
    {
        "name": "xelatex -> biber -> xelatex*2",
        "tools": [
            "xelatex",
            "biber",
            "xelatex",
            "xelatex"
```

```
        ]
    },
    {
        "name": "pdflatex -> biber -> pdflatex*2",
        "tools": [
            "pdflatex",
            "biber",
            "pdflatex",
            "pdflatex"
        ]
    },
    // Different compilers with BibTeX, for documents with both bibliography and
table of contents
    {
        "name": "lualatex -> bibtex -> lualatex*2",
        "tools": [
            "lualatex",
            "bibtex",
            "lualatex",
            "lualatex"
        ]
    },
    {
        "name": "xelatex -> bibtex -> xelatex*2",
        "tools": [
            "xelatex",
            "bibtex",
            "xelatex",
            "xelatex"
        ]
    },
    {
        "name": "pdflatex -> bibtex -> pdflatex*2",
        "tools": [
            "pdflatex",
            "bibtex",
            "pdflatex",
            "pdflatex"
        ]
    },
    // Run Compiler/Biber/BibTeX once
    {
        "name": "LuaLaTeX",
        "tools": [
            "lualatex"
        ]
```

```
    },
    {
        "name": "XeLaTeX",
        "tools": [
            "xelatex"
        ]
    },
    {
        "name": "PDFLaTeX",
        "tools": [
            "pdflatex"
        ]
    },
    {
        "name": "LaTeXmk",
        "tools": [
            "latexmk"
        ]
    },
    {
        "name": "Biber",
        "tools": [
            "biber"
        ]
    },
    {
        "name": "BibTeX",
        "tools": [
            "bibtex"
        ]
    }
],
```

After these settings, basic configuration for LaTeX Workshop is done. When you finished writing your LaTeX file, you only need to turn to sidebar of LaTeX Workshop and choose and run your compiling recipe at the first time, then you can easily compile your document by simply clicking the `build` button on the right top side of the editing window of your `.tex` file (or use keyboard shortcut `Ctrl+Alt+B`).

Advance Technique: Latexmk

As what is shown in recipe settings: if your document has a table of contents, you need to compile it twice with same compiler to have a PDF with everything correct; if your document has a bibliography, you need to compile it first, and process references with BibTeX or Biber, then compile it another time (or twice if it also has a table of contents). For more complex feature, you may even want to run some programme or code during compiling process. All these needs make recipes more and more complex and not that easy to use.

So here comes the latexmk. It is a Perl script that you only need to run it once, and it will handle all other things for you. This is a tutorial page I found on Internet showing how to directly use latexmk.

Moreover, latexmk can work with `.latexmkrc` file to customise running parameters. And this is why I recommend to add `Latexmk (auto)` and `Latexmk Clean (auto)` settings to our LaTeX Workshop settings. The former will build your document completely based on the `.latexmkrc` file in your directory (if not exists, then reference to user default one or even the system default one), and the latter can correspondingly do clean-up.

Following are some introduction to several common-used `.latexmkrc` settings. The `.latexmkrc` file used for this document can also help you understand how to write such file.

Output And Compiler

Define whether and how to generate PDF file with `$pdf_mode`:

- 0: Do not generate PDF file
- 1: Use PDFLaTeX
- 2: Use PS2PDF
- 3: Use DVIPDF
- 4: Use LuaLaTeX
- 5: Use XeLaTeX

PDFLaTeX is the most simple one and can run very quick; XeLaTeX and LuaLaTeX works well with UTF-8. These three are mostly used options.

For reference tools, use `$bibtex_use`

- 0: Do not run BibTeX and Biber, will not clean `.bbl` file
- 1: Run BibTeX or Bibier only when `.bib` file exists, will not clean `.bbl` file.
- 1.5: Run BibTeX or Bibier only when `.bib` file exists, and clean `.bbl` file only when `.bib` file exists

- 2: Run BibTeX or Bibier only when `.bib` file exists, and clean `.bbl` file no matter `.bib` file exists or not

For Biber, a universal setting be like this: `biber = "biber %O %S";`

Main File

Main file(s) that need to be compiled can be set with `@default_files`, and file(s) need to be ignored can be set with `@default_excluded_files`. Here is one version for reference:

```
@default_files = ("build-en-GB.tex", "build-zh-CN.tex");
@default_excluded_files = ();
```

Output Directory And Clean-up Settings

The output directory can be set with `$out_dir`. Final output PDF and other supportive files generated during compile will all be placed there.

Define what file extensions should be cleaned with `$out_dir`. Keep certain supportive files on purpose might be helpful if your document is a paper that need to be published to a journal.