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.