

配置 Visual Studio Code

安装完 VSCode 后，即可在已安装的程序中找到 VSCode。在配置好语言、主题、账号和同步等后，即可进入下一步：针对 LaTeX 的配置。

LaTeX Workshop

LaTeX Workshop 是 Visual Studio Code 上颇为经典的一款扩展，提供了预览、编译选项管理、自动完成、关键词高亮等功能。这个插件也是我们选择 VSCode 而不是其他工具来编写和编译 LaTeX 文档的主要原因。

要安装 LaTeX Workshop，只需要打开左侧的扩展选项（或按快捷键 `Ctrl+Shift+X`），搜索 `LaTeX Workshop`，点击安装即可。

安装完成后，点击齿轮图标进入管理菜单，选择“设置”进入其配置界面。

配置部分并不复杂。在搜索栏中已有的内容后添加 `tools`，在弹出的第一个栏目下点击“在 `settings.json` 中编辑”，然后粘贴下述内容即可获得较为通用的配置：

```
"latex-workshop.latex.autoBuild.run": "never", // 不在保存时自动编译
"latex-workshop.showContextMenu": true, // 启用 LaTeX 上下文菜单
"latex-workshop.intellisense.package.enabled": true, // 根据加载的包，自动完成命令或包
"latex-workshop.message.error.show": false,
"latex-workshop.message.warning.show": false,
"latex-workshop.latex.recipe.default": "lastUsed", // 默认将上次使用的编译方法作为默认编译动作
"latex-workshop.view.pdf.internal.synctex.keybinding": "double-click",
// 编译时调用的指令组
"latex-workshop.latex.tools": [
  {
    // 采用插件预设值的 Latexmk
    "name": "latexmk",
    "command": "latexmk",
    "args": [
      "-synctex=1",
      "-interaction=nonstopmode",
      "-file-line-error",
      "-pdf",
      "-outdir=%OUTDIR%",
      "%DOCFILE%"
    ]
  },
  {
    // 全部根据 .latexmkrc 文件设置进行编译的 Latexmk
    "name": "latexmk_auto",
    "command": "latexmk",
    "args": []
  },
  {
    // 根据 .latexmkrc 文件设置进行辅助文件清理
  }
]
```

```

    "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": [
{
    // 完全根据 .latexmkrc 文件进行编译
    "name": "Latexmk (auto)",
    "tools": [
        "latexmk_auto"
    ]
},
{
    // 与上一个对应，用于清理输出目录的辅助文件
    "name": "Latexmk Clean (auto)",
    "tools": [
        "latexmk_auto_clean"
    ]
},
// 各个编辑器和 Biber 组合，用于含目录和参考文献的场景
{
    "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"
    ],
},
// 各个编译器和 BibTeX 的组合，用于含目录和参考文献的场景
{
    "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"
    ],
},
// 单一编译器命令
{
    "name": "LuaLaTeX",
    "tools": [
        "lualatex"
    ],
},
{
    "name": "XeLaTeX",
    "tools": [
        "xelatex"
    ]
}

```

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

```

如此，在编写好你的 LaTeX 文档后，只需转到 LaTeX Workshop 的侧边栏页面，执行一次编译指令，后续点击 `.tex` 文件编辑窗口右上角的构建运行按钮（`Ctrl+Alt+B`），即可一键编译为 PDF 了。

进阶技巧 : Latexmk

就如同前文中设置 Recipe 时所展现的一样，如果文章有目录，你需要连续用同一个编译器编译两次才能获得正确的 PDF；而如果文章有参考文献，还需要先编译一次，再经过 Bibier 或 BibTeX，再由同一个编译器编译一次（如果文章含目录，则需要再编译两次）才可以。有时，为应对复杂的功能需求，甚至还需要在编译时运行一些外部程序。随着需求越来越复杂，显然只是配置 Recipe 会让选项越来越多且不一定实用。

Latexmk 是一个 Perl 脚本，它可以为你处理所有这些事情。[这里](#)是我发现的一个介绍页面，介绍了如何直接使用 latexmk。

Latexmk 还可以配合对应的 `.latexmkrc` 文件来管理运行选项。这也是之前的配置文件中我们添加了 `Latexmk (auto)` 和 `Latexmk Clean (auto)` 这两项的原因。前者将另 latexmk 完全根据你的项目目录的（如果没有，则会参考当前用户的或者系统的）`.latexmkrc` 文件中的设置来编译生成文档；后者则用于对应的执行清理辅助文件的工作。

以下是关于 `.latexmkrc` 中常用的几种设置的介绍。可以参考本项目中采用的 `.latexmkrc` 来进一步了解各个设置的具体用法。

设置输出和编译器

使用 `$pdf_mode` 设置是否输出 PDF 文件：

- 0：不生成 PDF
- 1：使用 PDFLaTeX 编译
- 2：使用 PS2PDF 编译
- 3：使用 DVIPDF 编译
- 4：使用 LuaLaTeX 编译
- 5：使用 XeLaTeX 编译

其中，PDFLaTeX 生成很快，XeLaTeX 和 LuaLaTeX 对 UTF-8 支持更好，都是比较常用的选项。

对于参考文献工具，使用 `$bibtex_use` 选项来控制。

- 0：不使用 BibTeX 和 Bibier，也不会清除 `.bb1` 文件
- 1：仅当 `.bib` 文件存在时使用 BibTeX 或 Bibier，不会清除 `.bb1` 文件
- 1.5：仅当 `.bib` 文件存在时使用 BibTeX 或 Bibier，仅当 `.bib` 文件存在时清除 `.bb1` 文件
- 2：仅当 `.bib` 文件存在时使用 BibTeX 或 Bibier，无论 `.bib` 文件是否存在，都会清除 `.bb1` 文件

此外，对于 Bibier，一般使用如下的参数配置：`biber = "biber %0 %S";`

设置主文件

可以通过 `@default_files` 设置需要编译的主文件。需要忽略的文件则可通过 `@default_excluded_files` 设置。一个可供参考的范例如下：

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

输出路径和清理

可以通过 `$out_dir` 设置输出文件所在的目录。编译中产生的辅助文件也会放在那里。

通过 `$out_dir` 设置需要清理的中间文件。选择性的保留中间文件对需要发表到期刊的论文可能很有帮助。