ACV-Dukang 手册

AWESOME-CV ACV-Dukang

基于Awesome-CV进行中文化适应及无侵入增强的面向 LaTeX 新人的快速开始项目

lnyk@me.com | lnyk | 衷心感谢Byungjin Park等童鞋的开源奉献

“执着而不计成本，不为索取只为陶醉” —— *Carl Zeiss*

偶然到访或是兴趣所致的你

二〇二二年七月一日

LaTeX, AWESOME-CV, ACV-Dukang, TEMPLATE
GITHUB, OPENSOURCE

ACV-Dukang
亲爱的朋友：

无论你是偶然访问到这里，还是正在搜索你感兴趣的项目，我在这里都表示由衷的感谢。

Awesome-CV-Dukang 项目，简称ACV-Dukang，是基于一款叫做Awesome-CV的 LaTeX 模板，进行中文化适应、额外提供许多新功能扩展，并面向入门使用者打包构建的快速开始项目。此篇文档旨在尽量以入门者的角度详细介绍ACV-Dukang项目涵盖的所有内容，比如使用、配置和附加功能，希望通过阅读本文，能与诸君分享更多知识和收获。

ACV-Dukang项目 Logo 来自互联网，如果冒犯请随时联系我纠正。生活不易，何以解忧？

ACV-Dukang 手册

最后：对你的关注，再次表示由衷的感谢！

总体介绍

Awesome-CV 本身设计非常优秀，但由于语言习惯等差别，其对中文环境的支持和适配需要用户自行调整很多东西，同时其原作者的本意主要是聚焦在构建“简历”模板，鄙人以为，如此漂亮的设计，理应延伸至文章甚至书籍创作领域。在这个层面上，需要更加宏观的对项目中每个部件的细节进行调整，并引入适合文章或书籍创作的功能模块，而且相对于漂亮的设计和丰富的功能，LaTeX 过高的使用门槛却一直阻挡在追求严谨排版和高质量输出，却不是特别习惯看似古怪的语法、复杂难记的命令环境和众多宏包的朋友面前，“劝退”很多有志之士踏进 LaTeX 的神奇世界。

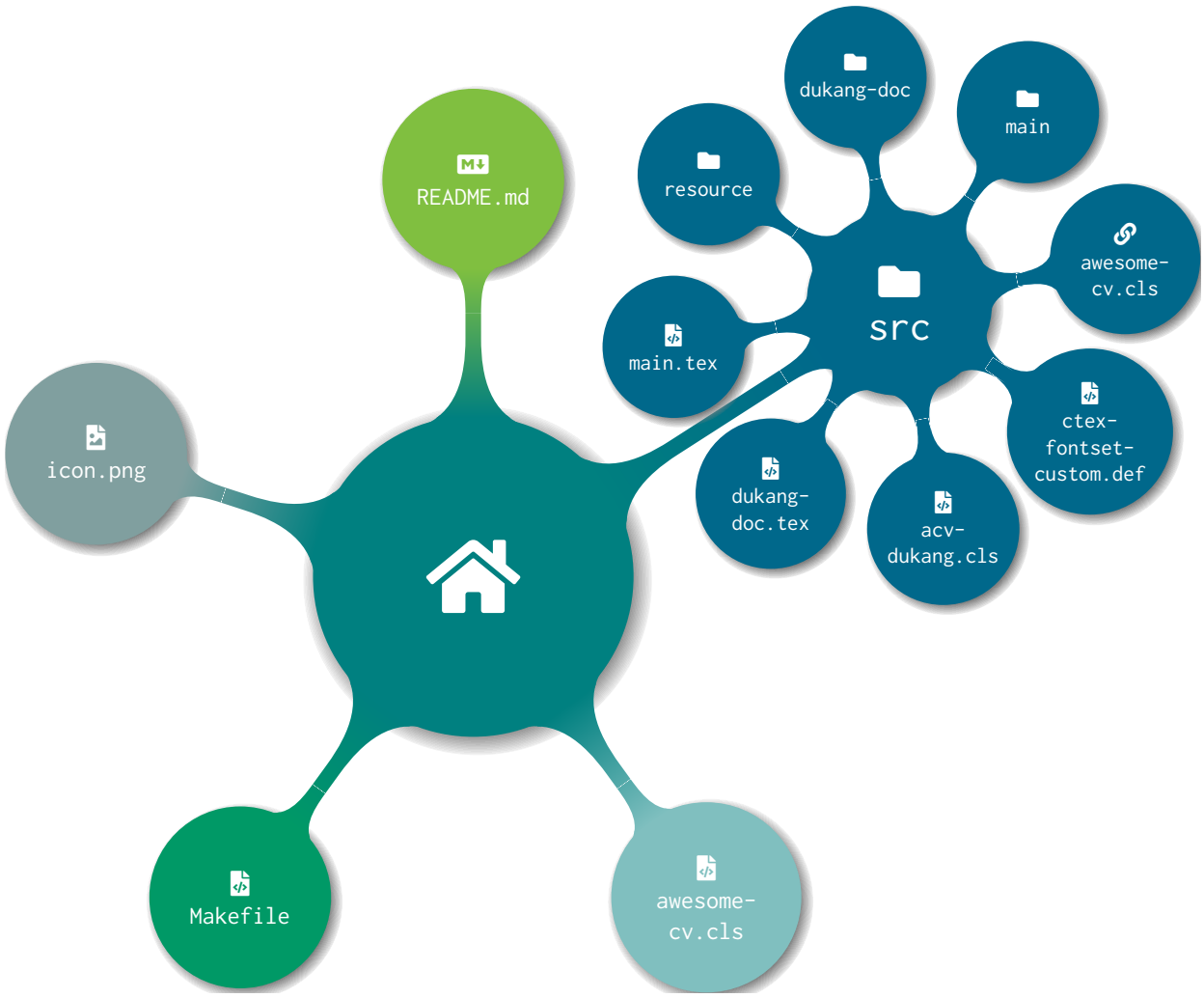
尝试使用 ctex 宏包对 Awesome-CV 进行中文环境适配，对相应的部件和元素进行调整，对一些代码上的设计进行改进，并增加一些封装好的开箱即用的实用部件，是ACV-Dukang项目的主要目的，将 LaTeX 方面的一些知识表述清楚，将维护该项目学习到的一些技巧分享出去，也是我的初衷。

罗嗦这么多，还是赶快让我们进入主题吧。

主体结构

ACV-Dukang的结构大致上分为三个逻辑部分，第一部分是来自 Awesome-CV 项目的文档类，定义了所生成文件的最重要的内容，第二部分是本项目新增的宏包、扩展内容、资源文件夹以及编译控制文件，第三部分是提供给使用者作为快速开始模板的文件，当然还有一些项目维护所涉及的例如 README、LICENSE、Git 相关文件等。

下图列举了项目主体结构和主要的文件所在位置，其中来自本项目的所有文本类文件，都有详细的备注和说明信息，如果想要快速开始，不妨按照下图找到这些文件，阅读里面的源代码，寻找自己感兴趣的部分。



项目其他子文件夹和一些辅助性的文件，比如 LOGO 等没有在上图中体现。

快速开始

如果你是 LaTeX 资深玩家，或者之前有过使用经验，很有可能已经拥有了一个随时可用的本地环境。确认你已经安装好了合适的 LaTeX 套件（比如 TexLive 2022 版本），对应操作系统的 GNU Make，Python 3 和 Pygments 库以及所需字体[①]之后，可以使用下面的命令快速体验ACV-Dukang的输出过程和简单用法，如果不确定你的本地环境是否符合要求，请暂时越过本章节，找到“编译使用详解”章节的“本地环境准备”部分跟着说明逐步进行。

打开控制台依次执行

```
# 进入项目根目录
cd latex-dukang

# 编译生成你的大作
make

# 编译生成本文档
make doc
```

整个编译的过程，会输出大量的编译信息和告警内容，如果看到 Warning 提示，先别着急，在 LaTeX 的世界里，有众多的叫做“宏包”的贡献者在共同的支撑着你的创作，并不是所有的告警信息都意味着你“哪里做错了”，有些只是“善意的提醒”，比如ACV-Dukang当前版本会有“未找到斜体字形”、“hbox overfull/underfull”等告警，虽然这类告警一定意味着“编译质量不怎么高”，但实际情形并不影响最终的输出结果，这些告警在后续的章节会有详细的说明。

回到上面的命令，如果执行顺利的话，第一个 make 执行完毕后，都会有类似下面的内容：

```
> Latexmk: All targets (src/main.xdv src/main.pdf) are up-to-date
```

这说明 main.tex 文件已经成功被编译输出到了 main.pdf，可以打开这个 PDF 文件看看实际的效果。第二个 make 是用来生成ACV-Dukang文档（也就是本文）的，也跑一跑编译吧，确保你看到下面的提示信息。

```
> Latexmk: All targets (src/dukang-doc.xdv src/dukang-doc.pdf) are up-to-date
```

如果顺利编译完成，你会在`./src/`文件夹中看到 main.pdf 和 dukang-doc.pdf 两个文件，这说明你已经可以正常使用ACV-Dukang进行创作了。

如果，你碰到了光标停在一个问号后面，画面信息不再滚动的情形，那直接就是“错误”而不是“告警”了，这时候需要你向上查看具体哪里出了问题，或者输入大写的“X”并回车退出编译过程。这时候一般是没有 main.pdf 或 dukang-doc.pdf 输出的，请仔细检查输出日志中的错误提示，尝试着解决问题，并在`> make cleanall`清理完所有临时文件之后再次尝试 make。

[①]如果你是资深玩家，请查阅 src/ctex-fontset-custom.def 文件并安装好对应字体，否则请首先阅读“编译使用详解”章节的“本地环境准备”部分。

结构设计

由于ACV-Dukang尽量使用非侵入式的方式与 Awesome-CV 进行集成，自然也沿用了其简洁方便的结构设计，除了总体编译文件、项目说明文件、项目图标以及文档类本体以外，其他的内容相关文件都放在`./src/`中。其中为了更方便的操作，部分文件使用了软链接[①]。

项目结构说明

```
ACV-Dukang
├── awesome-cv.cls ..... Awesome-CV 的类文件
├── icon.png ..... 项目图标
├── Makefile ..... 控制编译命令的 make 文件
├── README.md ..... 项目说明文件
└── src ..... 内容相关文件
    ├── awesome-cv.cls ..... Awesome-CV 类文件的软链接
    ├── ctex-fontset-custom.def ..... ctex 的自定义字体文件
    ├── acv-dukang.cls ..... ACV-Dukang 的主文件
    ├── dukang-doc.tex ..... 本文档的主文件
    ├── dukang-doc ..... 存放本文档的章节内容文件
    ├── main.tex ..... 从此文件入手开始创作
    ├── main ..... 存放 main.tex 引用的正文内容
    └── resource ..... 存放图形、表格、外部 pdf 或 tex 等资源文件
        ├── ctex-fontset-custom.def ..... ctex 自定义字体文件的软链接
        ├── Makefile ..... 对应 resource 文件夹的 make 文件
        ├── logo.png ..... 文档首页头信息中引用的图片文件
        └── r-*.tex ..... 所有以r-*.tex形式命名的文件都会支持联动编译控制
```

awesome-cv.cls

文档类

- 该文件属于 Awesome-CV 的文档类文件，ACV-Dukang主要也是通过对该文件进行重定义来达到中文支持的目的。
- 由于是无侵入方式，在没有发生重大变化的前提下，应该可以下载该文件的最新版本后直接覆盖来升级。

Makefile

编译控制

- 控制编译过程、简化项目使用操作的 GNU Make 文件。
- 在 Awesome-CV 原有基础上，根据ACV-Dukang所引入的部分新功能进行了增强。
- 不建议修改此文件，除非你知道自己在做什么。

src/awesome-cv.cls 和 src/resource/ctex-fontset-custom.def

软链接

- 这两个文件都是 Linux 平台的相对位置软链接。
- 软链接使用`ln -rs`定义。

[①]使用 GNU Make 以及软链接，对 Windows 系统和使用 IDE 的用户来说可能需要做一些适应性调整，详情请阅读“编译使用详解”章节的“本地环境准备”部分。

src/ctex-fontset-custom.def 字体定义

- 配合 ctex 宏包的`fontset=custom`选项来使用，可以修改这个文件中的字体设定。
- 根据 ctex 的字体自定义设置，这里的`custom`对应 ctex-fontset-`custom`.def，可以根据需要创建自己的自定义文件。

src/acv-dukang.cls ACV-Dukang主文件

- 该文件是ACV-Dukang的文档类主文件，在文档中使用`\documentclass`进行引用。
- 所有修改以及增强的内容在源代码中都有详细的注释，想要更深入底层了解的话可以直接看该文件源码。
- 之所以没有使用 Doc 和 DocStrip 构建 Class 文件，是因为我不会🐶或者只是单纯的不想把简单问题复杂化，毕竟ACV-Dukang本身没有多少代码，内容也不复杂，况且已经配有详细的文档说明，没必要再生成一套“干货”，又不是通用宏包，不会提交给 CTAN🐶

src/dukang-doc.tex 和 src/dukang-doc/ ACV-Dukang说明文档

- ACV-Dukang说明文档（本文）主文件和存放章节内容文件的目录，同时也包含了用法示例。
- 由于受 Makefile 编译控制，dukang-doc.tex 的位置和文件名不要随便更改，但是 dukang-doc 文件夹中的内容文件都是由`\input{...}`引入到主文档中，所以只要对应好文件位置，这个文件夹是可以自由修改的。
- 由于上述文件只用于生成ACV-Dukang的说明文档（本文），与用户作品无关，所以不建议改动。

src/main.tex 和 src/main/ 用户作品文件

- 用户作品的起始模板文件，可以直接从该文件开始进行创作。
- 文件夹`./src/main/`与编译过程无关，仅用于存放起始模板文件所引用的内容文件，可以更改，甚至不用也行。
- 如果作品的篇幅较长，还是建议使用`\input{...}`的方式，可以保证主文档的内容干净整洁。

src/resource/ 资源文件夹

- 用来统一存放图形、表格、外部 pdf 或 tex 等资源文件，并支持部分文件的联动编译。
- Makefile 是与外层 Makefile 联动的编译控制文件，同时提供了部分针对资源目录中不同类型文件的编译命令。
- logo.png 是封面上部显示的图片，在主文档的导言区中使用`\photo{...}`进行定义。
- 该文件夹被 dukang-doc 和用户作品共用。
- 所有以`r-*.tex`形式命名的文件都会被联动编译和控制，本文档封面中 Tikz 图的源文件`r-arch.tex`就在这里。
- 不要更改该文件夹的位置，否则联动编译控制将找不到文件。

编译使用详解

在本章节里，我会用很短的篇幅快速介绍一下与ACV-Dukang运行相关的内容，包括本地环境准备、Makefile 及编译、选项设定和使用流程，有些内容比较重要，请尽量顺序阅读。

本地环境准备

在真正开始使用ACV-Dukang进行创作之前，要先准备好你的本地编译环境。

关于兼容性

我是使用 TexLive 2022 套件在 Ubuntu 20.04 平台上对ACV-Dukang进行开发维护和编译测试的，如果你的是 Windows 操作系统，或其他版本 TeX 套件，可能会遇到问题，虽然ACV-Dukang的 Makefile 已经针对 Windows 系统进行了设计，但对应 Windows 版本的 TexLive 套件，以及 GNU Make 和 Python 3 可能需要你花些功夫安装调试一番。目前还没有更加详细的测试结果，也衷心希望能够得到你的使用情况反馈。

关于字体

由于ACV-Dukang主要使用 $ctex$ 宏包进行中文环境底层支持，而 $ctex$ 默认提供的四套不同平台的字体方案都比较通用，没有个性化的设置，并且ACV-Dukang在 $ctex$ 的基础上增加了几个中文环境下的常用字体，因此需要采用 custom 方式进行设置，配置文件的相对位置在：`./src/ctex-fontset-custom.def`

完整列表如下：

字体名称	说明	短命令
Adobe 仿宋 Std	文档正文默认字体，大小为 10pt	*\fangsong*
Adobe 黑体 Std	**一般用于章节名称**	*\heiti*
Adobe 楷体 Std	脚注、代码块以及中文 Mono/Sans 环境会用到	*\kaishu*
方正小标宋简体	**主要用于章节标题等场景**	*\fzxbs*
Inconsolata	Footnote, Code block and English Mono/Sans Environmet	*\sffamily*
Adobe 宋体 Std	在文档中几乎不用，可以根据需要使用	*\songti*
幼圆	艺术创意等场景可能会用到	*\youyuan*
隶书	**诗词歌赋、文学环境比较合适**	*\lishu*

以上字体一部分是ACV-Dukang基本依赖，一部分是本文档所用字体，建议全部安装，或者根据需要自行修改字体配置文件。短命令的具体用法可以参考本文档源码。

参考配置

因为本地环境的构建可以有很多种选择，现在以我的配置为例：

1. Ubuntu 20.04
2. TexLive 2022
3. GNU Make 4.2.1
4. Python 3.8.10
5. Python Pygments

如果你用的是 Linux 操作系统，很有可能已经自带 GNU Make 和 Python 了，使用下面的命令查看一下版本号，确保版本别太旧：

```
# 这里推荐使用 Python3
pytho3 --version

# 查看 GNU Make 版本，越高越好
make --version
```

之所以需要 Python 支持，是因为ACV-Dukang部分组件使用了 $minted$ 引擎调用 $Pygments$ 库进行代码高亮渲染，如果缺少该库，这部分组件将不可用：

```
# 查看 Pygments 版本
pygmentize -V

# 安装或升级 Pygments 库
pip install --upgrade Pygments
```

Makefile 及编译

ACV-Dukang使用 Makefile[①]将几个最常用的 LaTeX 编译相关操作定义成了对应的控制台命令，用以对编译、输出、清理等过程进行简化处理，以及联动 resource 文件夹下的 Makefile 文件。

Makefile

```
# 默认使用 xelatex 进行编译
# 由于部分命令使用了 minted 包，需要打开-shell-escape
CC = latexmk -xelatex -shell-escape
# 源码根目录
BASE_DIR = src
# 项目文档的主文件
DOC_SRC = $(BASE_DIR)/dukang-doc.tex
# 所有需要自动清理的文件
CLEANDIRS = . $(BASE_DIR) $(foreach d, resource tex dukang-doc, $(BASE_DIR)/$(d))
CLEANFILES = $(foreach x, *.pyg *.listing *.xdv *.aux *.log *.fls *.dvi *.fdb_latexmk .auctex*
 ↪  dist _minted*, $(x))

# 定义删除时候使用的命令，用于同时支持 Windows 和 Linux
ifdef SystemRoot
  RM = del /Q
else
  RM = rm -fr
endif

# 以下是定义的所有编译命令
.PHONY: main doc all clean cleanall

make: main

main: $(BASE_DIR)/main.tex
  make -C $(BASE_DIR)/resource/ -f Makefile all
  $(CC) -output-directory=$(BASE_DIR) $<

doc: $(BASE_DIR)/dukang-doc.tex
  make -C $(BASE_DIR)/resource/ -f Makefile all
  $(CC) -output-directory=$(BASE_DIR) $<
```

[①]前半部分的指令都有注释进行说明，GNU Make 的用法也不在本文的讨论范围之内，感兴趣的朋友可以自行学习。

```
31
32 resource:
33   make -C $(BASE_DIR)/resource/ -f Makefile all
34
35 all: main doc
36
37 dist: doc
38   mkdir -p doc
39   cp $(BASE_DIR)/dukang-doc.pdf doc/
40   make cleanall
41
42 clean:
43   @for d in $(CLEANDIRS); \
44   do \
45     for f in $(CLEANFILES); \
46     do \
47       $(RM) $$d/$$f; \
48     done \
49   done
50
51 cleanall: clean
52   -@$(RM) $(BASE_DIR)/*.pdf
53   make -C $(BASE_DIR)/resource/ -f Makefile cleanall
```

以下所有 make 命令，都必须在项目根目录（就是你能看到 README.md 的地方）下执行。

make	自动编译输出 main.tex，等同于 make main	编译输出
make main	同上	编译输出
make doc	自动编译输出ACV-Dukang文档，也就是本文档的输出命令	编译输出
make resource	使用 resource 文件夹下的 Makefile 编译输出所有支持的资源文件，等同于进入 resource 文件夹下进行 make 或 make all	关联编译
make all	一次性编译 main.tex 和ACV-Dukang文档，等同于 make main && make doc	编译输出
make dist	主要用于执行ACV-Dukang本身发布前的准备工作：新建 doc 目录，生成文档及清理，普通使用者用不到该命令	项目维护
make clean	自动清理所有临时文件和文件夹，包括主目录和所有子目录	自动清理
make cleanall	在自动清理所有临时文件和文件夹的基础上，还会删除掉所有主目录下的 PDF 文件，并联动 resource 文件夹下所有生成的资源 PDF	自动清理

make cleanall 只会删除符合资源文件命名规则的 PDF 文件，其他文件不受影响。

文档选项及设定

ACV-Dukang提供了用于立刻开始创作的初始文件，位置是 src/main.tex，该文件导言区[1]部分的设定与本文档源码

[1]相信你知道什么是导言区，如果不是太清楚，还是抓紧自学一下吧，或者简单的认为，在大多数情况下，\begin{document} 前面的部分，就是导言区。

src/dukang-doc.tex 完全相同，只需要根据需要修改一些文档名称、作者姓名、首页需要哪些字段、是否需要首行缩进等信息和选项，就可以正式开始为你的大作添加正文了。

打开 main.tex 之后，首先会看到导言区一堆设定和备注信息，大多数情况下，阅读这些备注信息就足够掌握如何修改了，这里对所有选项进行深入说明。

`\documentclass[12pt, a4paper, final]{awesome-cv-dukang}` 基本配置

- 正式引入 Awesome-CV 文档类。
- 这里的字号设定（12pt）基本没什么卵用，因为几乎每个文档组件都定义了自己的字体风格。

`\geometry{left=1.4cm, top=.8cm, right=1.4cm, bottom=1.8cm, footskip=14.5pt}` 基本配置

- 使用 geometry 宏包定义纸张的页边距以及页脚距离

`\colorlet{awesome}{awesome-red}` *Awesome-CV*

Awesome-CV 的颜色设定 必选

- 可以指定 Awesome-CV 预制好的几个配色集，包括 awesome-emerald, awesome-skyblue, awesome-red, awesome-pink, awesome-orange, awesome-nephritis, awesome-concrete, awesome-darknight
- 也可以使用`\definecolor`指定自己喜欢的颜色，总共有 awesome, darktext, text, graytext, lighttext, sectiondivider 这几个颜色名称可供定义。
- ACV-Dukang提供的所有增强组件都可以根据颜色设定进行风格自适应哦

`\setbool{acvSectionColorHighlight}{true}` *Awesome-CV*

指定是否使用配色凸显章节标题后紧跟的分割线 必选

- 如果设定为 true，责章节名称后面的长分割线会有颜色，否则为黑色。

`\renewcommand{\acvHeaderSocialSep}{\quad\textbar\quad}` *Awesome-CV*

封面头部 Logo 右侧社交媒体帐号之间的分隔符定义 可选

- 默认为管道符：`< 空格 >|< 空格 >`

个人信息部分 *Awesome-CV*

该部分用来定义一些个人信息或文档信息 部分可选

- `\photo[rectangle,noedge,left]{./src/resource/dukang-logo}`用于定义首页的 Logo 图片，文件扩展名默认为.png，可用的裁剪选项为 circle（圆形）和 rectangle（正方形），可用的边框选项为 edge（有边框）和 noedge（无边框），可用的位置选项为 left（靠左）和 right（靠右）。
- 该部分除了`\name`和`\dukangPDFTitle`是必选的以外，其他设定不需要的均可以注释掉，首页中相应的部分会自适应。
- 由于`\name`在 Awesome-CV 文档类中有多处引用，所以必须指定，不能删掉。但对于正文来说，仅用在首页顶部第一行大标题及信件环境结尾，不出现在其他位置。
- `\dukangPDFTitle`用来生成 PDF 文件书签中的主标题，所以必须指定，不能删掉。

社交媒体信息部分

Awesome-CV

该部分用来定义社交媒体帐号或联系方式

部分可选

- 该部分有若干社交媒体选项，可以根据需要进行定义，不需要的可以注释掉。
- 上面定义的分隔符`\acvHeaderSocialSep`就是用来分割这些帐号的。
- 至少要保留一条，否则编译出错！

cvletter 环境基本信息

Awesome-CV

cvletter 环境一般用于定义首页的内容

必选

- 由于使用了 *ctex* 宏包，`\today`默认为大写中文日期格式，比如二〇二二年七月一日。
- 该部分的定义一个都不能少！

dukang 导言区设定部分

ACV-Dukang

此部分包含ACV-Dukang及相关宏包提供的若干增强设定

必选

- `\setbool{dukangParIndent}{true}`，由于 Awesome-CV 大多数风格都使用了组件化（自定义命令或环境）来实现，没有使用 chapter/section 等标准结构，这直接导致了在引入 ctex 宏包进行中文化的时候，需要对每个组件进行单独的设定，比如首行缩进两字符对于有些组件要么不起作用，要么显示错乱，这里提供一个全局开关，会自动根据组件的具体情况有选择的开启首行缩进，以达到风格统一、显示美观的效果。
- `\setbool{dukangBookmarkLeadingNumber}{true}`，Awesome-CV 当前版本并不支持给输出的 PDF 文件按照文档结构自动添加书签（导航栏），ACV-Dukang提供了这方面的支持，这个全局开关用来指定所添加的书签标题前，是否包含阿拉伯数字的章节编号。
- `\hypersetup`，该部分用来为生成的 PDF 文件提供若干属性字段。

Awesome-CV 文档区设定部分

Awesome-CV

该部分设定出现在文档区，也就是`\begin{document}` 和`\end{document}` 之间。

可选

- `\makecvheader[R]`，这不是页眉！Awesome-CV 没有页眉。这是首页包括 Logo 在内的抬头（Header）部分，可以注释掉，首页布局会自动从 cvletter 环境开始。可用选项用来控制对齐方向，L 标识左对齐，C 标识居中对齐，R 标识右对齐。都要大写！
- `\makecvfooter`，这个是每页的页脚，分为左中右三个部分，每个部分都可以留空，但必须保留大括号`{}`

以上是ACV-Dukang当前版本设定部分的详细说明，需要格外注意的地方都有颜色高亮，右边颜色高亮的标签说明该选项来自哪个部分。并且，假如在修改的过程中不小心把 main.tex 搞乱了也没关系，可以随时打开 dukang-doc.tex 查看正确的配置，或者干脆把除了正文以外的所有内容复制回来，重新设定一下，就又可以开始创作了。这也是为什么我推荐用`\input{...}`把文档正文章节和 main.tex 主文件分开的原因。

温馨提示

无论何时，dukang-doc.tex 都是你值得参考的示例文档，文档本身和其内容章节文件（特别是源代码）尽量涵盖到了ACV-Dukang的全部功能，包括设定和功能模块等，随时可以回来查看。

使用流程

ACV-Dukang的编译控制文件 Makefile 提供了适合下面几种场景的编译流程，相信总有一个适合你。首先，对于一般使用来说，需要做的步骤很简单：

修改main.tex → 添加内容 → > make ✓

这样在 src 文件夹下就得到了 main.pdf，同时在 resource 文件夹下，如果有符合命名规则的资源文件，也会被联动编译，并生成对应的 PDF 文件。ACV-Dukang当前版本资源文件的命名规则是`r-*.tex`，符合这个规则的.tex 文件都会被自动编译和控制。

如果想要保留编译之后的 PDF，同时把项目目录清理干净的话：

修改main.tex → 添加内容 → `> make` → `> make clean` ✓

最后，如果希望只保留源代码[1]，把其他临时文件连同编译出来的东西一同干掉的话：

修改main.tex → 添加内容 → `> make` → `> make cleanall` ✓

> 如果在编译过程中出现问题，强制退出编译过程之后想要再次编译，最好先执行`> make cleanall`一遍，清理完所有临时文件之后再开始，否则编译很可能会出错无法继续下去。

[1] 比如用于提交源代码，归档，或者你就是个纯粹的代码强迫症患者❤️

组件使用详解

组件共分为两类，一类是 Awesome-CV 原生定义的组件，用于结构化排版文档，另一类是ACV-Dukang为 Awesome-CV 添加的自定义组件[①]可以在作品中使用。以上两种组件，如果你是一路看到这里，相信已经见过它们中的绝大多数了。

由于语言环境不同，许多 Awesome-CV 的原生组件在进行中文化的过程中，需要对一些细节进行处理，ACV-Dukang尽量以无侵入（重定义）的方式在不碰类文件的情况下，对这些原生组件进行了修改和部分增强。同时，ACV-Dukang增加了许多自定义组件，有的是方便引入资源的，有的是生成表格的，有些提供了代码高亮，有些生成小按钮风格，它们都有一个共同的能力，就是可以随着文档定义的 Awesome-CV 主色调自动适应配色。

下面分两个部分分别对 Awesome-CV 原生组件和ACV-Dukang自定义组件进行详细介绍。

Awesome-CV 原生组件

Awesome-CV 没有使用 LaTeX 传统的 chapter/section 组织结构，而是完全自定义了自己的组件用于支撑文档结构。这样做对于简历类型的文档当然更加灵活，但是如果想要用来进行文章或书籍的创作，就有些不够用了。而且对于中文环境排版来说，我们有着更加复杂的习惯和要求，比如首行缩进、行间距、断句、对齐、字体等，这些是 Awesome-CV 原生环境装进中文时一定会遇到的问题，虽然一部分能够被 ctex 宏包自动修正，但由于没有 chapter/section 等结构，面对非标准化的自定义环境和命令，ctex 宏包的强大能力也无处施展。因此，ACV-Dukang在这方面着重下了一番功夫，对绝大部分 Awesome-CV 原生组件进行了调整，并修复了一些我感觉像 bug 的地方[②]，比如某些组件不能紧挨着，某些组件调用顺序不对会搞乱行间距或编译错误等等。

下面我们逐一列出这些经过修改和增强之后的 Awesome-CV 原生组件。

cvletter	信件环境，主要用于首页的信封效果	环境
lettersection	包含在 cvletter 环境中的段落元素	命令
cvparagraph	正文段落，ACV-Dukang调整了正文格式，该组件基本没必要再用了	环境
cvsection	章节定义，对应一级标题	命令
cvsubsection	章节命令，对应二级标题	命令
cvsubsubsection	章节命令，对应三级标题	命令
cventries	带主副标题的列表环境	环境
cventry	包含在 cventries 环境中的列表元素	命令
cvhonors	带风格定义的三列列表环境	环境
cvhonor	包含在 cvhonors 环境中的列表行	命令
cvskills	带风格定义的两列列表环境	环境
cvskill	包含在 cvskills 环境中的列表行	命令

cvletter & lettersection

主要用于首页信件环境的风格定义，也就是看似一封信的显示效果。由于`\lettersection`的显示效果与`\cvsection`类似，且只用在信件环境中，扩展意义不大，所以尽管在 Awesome-CV 的示例文件中使用了，但ACV-Dukang推荐直接在`\cvletter`中书写正文，或者包含`\cvsection`部分。

使用方法可参考`./src/dukang-doc/00-cover.tex`

[①]所谓组件，其本质上只是对一些宏包功能的再封装，以达到方便使用的目的，真正要感谢的是那些宏包的作者。

[②]其实应该也不算 bug，只是修改完之后在使用方面会更方便灵活

cvparagraph

这是段落的环境封装，在正文中用的不多，因为ACV-Dukang已经定义好了正文段落的样式，这个 cvparagraph 环境可以不用，直接书写正文就好。

cvsection & cvsubsection & cvsubsubsection

都是 Awesome-CV 自定义的章节命令，由于没有 chapter，所以 cvsection 就是一级标题，其实对于简历风格的模板来说，这样设计也是合情合理的，只不过用来创作文章或书籍的话，章节层级就要分明一些。

ACV-Dukang修改了这些命令的样式，增加了决定是否启用首行缩进的开关，在文档的主文件 main.tex 中可以指定，具体可以参考“dukang 导言区设定部分”的备注说明。同时，原生环境并不带 PDF 书签能力，ACV-Dukang为这几个命令增加了该功能，并可以通过全局开关设置是否显示书签标题前的章节编号。

cventries & cventry

cventries 环境包含若干 cventry 命令，ACV-Dukang使用可变参数重新封装了 Awesome-CV 的原生 cventry，现在该组件共包含五个部分，命令格式为：

```
\cventry[第一行左侧][第一行右侧][第二行左侧][第二行右侧]{正文}
```

参数中除“正文”以外都可选，但需要按顺序给定，否则无法判断是第几个参数，例如只想显示第二行右侧的文字，需要将前面三个参数都标记出来并留空，有内容的参数后面的空参数可以不标记，例如：

```
\cventry[][][][第二行右侧]{正文文字}
```

或者

```
\cventry[][第一行右侧]{正文文字}
```

cvhonors & cvhonor

cvhonors 环境包含若干 cvhonor 命令，其本质是表格。ACV-Dukang重新封装了原生组件，现在 cvhonors 环境有两个可选参数，格式为：

```
\begin{cvhonors}<*>[LLR]\cvhonor...\end{cvhonors}
```

其中第一个参数星号“*”为底色开关，带星号标识启用奇偶行底色，不带星号为无底色，切换底色会少许改变外边距，组件会自动进行调整。第二个参数“LLR”的三个字母表示接下来所有 cvhonor 组件左中右三部分的对齐方式，L 表示左对齐，C 标识居中对齐，R 表示右对齐，不给出该参数的话，默认为“LLR”。

cvhonor 命令分为三个必选参数，可以留空但不能不写，格式为：

```
\cvhonor{左}{中}{右}
```

cvskills & cvskill

cvskills 环境包含若干 cvskill 命令，其本质是两列的表格。这个组件与 cvhonors 风格基本相同，前者是三列表格，这个环境是两列，参数默认为“RL”，即第一列右对齐，第二列左对齐，具体用法是：

```
\begin{cvskills}<*>[RL]\cvskill...\end{cvskills}
```

cvskill 命令分为两个必选参数，可以留空但不能不写，格式为：

```
\cvskill{左}{右}
```

以上是ACV-Dukang当前版本进行过优化的所有 Awesome-CV 文档组件，其实其原生组件也大致就这么多了，下一小节我们看一下来自ACV-Dukang的自定义组件。

ACV-Dukang 自定义组件

在 Awesome-CV 原生组件的基础上，ACV-Dukang将几个比较强大的宏包封装成了一些较为通用的自定义组件，一是更加灵活使用方便，二是丰富了 ACV 在“简历”场景以外的功能，这些组件适合应用在特别是科技类或 IT 类文章和书籍场景中。先看一下ACV-Dukang自定义组件的全家福：

dkbutton	Inline 风格的代码小盒子，可选两种显示风格	命令
dkresource	引入外部图片或 PDF 的命令	命令
dkcode	可以对大段代码进行高亮显示的环境，可选两种显示风格	环境
dkcodefile	读取外部文件并将内容进行代码高亮显示的环境，可选两种显示风格	命令
dkcomment	可完全自定义显示的备注框，同样有两种显示风格可选	环境
dkcodebox	另一种风格的 Inline 代码小盒子，可切换是否显示命令提示符	命令
dkmeasure	打印长度数值，主要用于在构建页面时对长度变量进行测量	命令
dirtree	显示文件夹树状结构的宏包命令，ACV-Dukang引入未封装，直接使用	命令

这些组件都有自己的用法和适合的使用环境，以及一些注意事项，其中最重要的一条共性原则是：必选参数都使用大括号{}指定，可选参数都是方括号[]指定，由于 LaTeX 没有所谓“命名参数”的概念，因此所有的可选参数必须从右至左省略，举个例子，参数 $1,2,3,4$，要想指定参数 3 的内容，必须连同 $1,2$ 一起给出，否则引擎无法判断你到底给出了第几个参数。

dkbutton

首先介绍这个小按钮，是因为在本文档中它用的最多，几乎随处可见，调用格式为：

```
\dkbutton<*>[color]{text}
```

\dkbutton{这里有个小按钮} & \dkbutton*{这里有圆形的小按钮}

color 默认为自动适应在主文档中指定的 Awesome-CV 的主配色，也可以自行指定，比如：

\dkbutton[blue]{蓝色小按钮} & \dkbutton*[green]{绿色圆形小按钮}

在使用方面，除了尽量以行内方式使用外几乎没有限制，同一行中别用太多就行，否则影响自动断行。当然如果想要一个超大的按钮也是可以的，只是推荐把它放在单独一个段落中，比如：

\dkbutton*{一个巨大的圆形按钮}

dkresource

这个组件的作用是将外部资源（比如另一个 PDF 文档或 PNG 图片）引入当前位置居中显示，并可手动指定宽度自动缩放，调用格式为：

```
\dkresource<[caption]>{file}<[width_factor]><[htbp!]>
```

caption	可选参数，资源下方的名称或标签。
file	必选参数，需要引入的资源，可以是图片、PDF 文件或其它 $figure$ 环境支持的文件类型，只需给出路径及名称，无需加扩展名或后缀，相对路径的默认起点为 `./src/`.
width_factor	可选参数，相对于整个行宽的占比，取值为 0-1，默认为 0.9 倍行宽。
htbp!	可选参数，用来控制资源在页面的浮动显示位置，其中 h 表示当前位置或代码所处的上下文位置，t 表示 Top（页面顶端），b 表示 Bottom（页面底部），p 表示 Page（单独成页），! 表示在判定位置时忽略限制选项，默认为 htb.

使用示例：

```
\dkresource[这里有个酒坛子]{resource/dukang-logo}[0.25]
```

[1] resource 文件夹的联动编译功能最主要目的也是为了支持 dkresource.

这里有个酒坛子

关于浮动体的说明

dkresource 实际上是基于 LaTeX *figure* 环境的简单再封装，让人比较晕的是 *htbp*! 浮动体控制参数的意义，简单理解的话就是，在生成 PDF 文件的时候，LaTeX 编译器会根据浮动体控制参数来自动判断并确定浮动体在页面中的显示位置，这里的浮动体指的就是 *dkresource* 所引入的资源，排版位置的选取与参数里符号的顺序并无关系，编译器总是以 $h \to t \to b \to p$ 的顺序来检查参数。

!的作用是忽略限制，这里的默认限制总共有两条，超出该限制会强制将浮动体拖入下一页再判断，这两条分别是：

- 个数：除 p 参数（单独成页）外，默认每页不超过 3 个浮动体，其中顶部 t 不超过 2 个，底部 b 不超过 1 个。
- 空间占比：默认顶部 t 不超过页面高度的 70%，底部 b 不超过 30%.

dkcode

这个组件通过封装 *tcolorbox* 宏包，使用 *minted* 引擎对计算机语言代码进行高亮显示，支持中文并调整行距字体显得更加美观，同时支持组件标题栏风格、代码高亮风格以及主配色的手动指定。用法为：

```
\begin{dkcode}<*>{lang}<[title]><[color]><[minted_style]>...\end{dkcode}
```

特别说明

dkcode 环境以及 *dkcodefile* 命令默认使用 *tcolorbox* 的 *minted* 高质量引擎进行代码渲染，这意味着你需要提前准备好 Python 和 Pygments 库，详细安装说明在之前“编译使用详解”章节的“本地环境准备”小节中已经进行了阐述，遇到问题请随时检查本地环境是否满足编译条件。

*	可选参数，不带星号为普通标题栏，带星号为瘦标题栏，两者只是标题风格不同。
lang	必选参数，指定 *minted* 引擎支持的语言名称，例如 $bash, python3, java, c, go$ 等，运行`pygmentize -L lexers`命令查看所有支持的语言名称。
title	可选参数，代码块的标题栏内容，如果没有该参数，则不显示标题栏，并自动忽略 * 参数。
color	可选参数，指定主色调或省略进行自适应，默认为 *awesome*
minted_style	可选参数，指定 *minted* 引擎预定义的高亮显示风格，运行`pygmentize -L styles`命令查看所有预定义风格，默认为 tango.

使用示例：

带标题栏默认颜色

```
\begin{dkcode}*{latex}[带标题栏默认颜色]
  ...
```

带瘦标题栏蓝色 vim 风格

```
\begin{dkcode}{latex}[带瘦标题栏蓝色 vim 风格][blue][vim]
  ...
```

```
\begin{dkcode}{latex}
  % 无标题
```

dkcodefile

这个组件的功能和设定与\dkcode类似，都是代码高亮块，风格也一样，只不过多了一个必选参数，就是从外部文件中读取内容，比较适合多行代码，或者联动编译[①]。具体用法是：

\dkcodefile<*>{file}{lang}<[title]><[color]><[minted_style]>

*	可选参数，不带星号为普通标题栏，带星号为瘦标题栏，两者只是标题风格不同。
file	必选参数，指定外部文件的位置，可以使用相对路径，起点跟 *dkresource* 的 *file* 参数一样，都是./src/.
lang	必选参数，指定 *minted* 引擎支持的语言名称，例如 $bash, python3, java, c, go$ 等，运行pygmentize -L lexers命令查看所有支持的语言名称。
title	可选参数，代码块的标题栏内容，如果没有该参数，则不显示标题栏，并自动忽略 * 参数。
color	可选参数，指定主色调或省略进行自适应，默认为 *awesome*
minted_style	可选参数，指定 *minted* 引擎预定义的高亮显示风格，运行pygmentize -L styles命令查看所有预定义风格，默认为 tango.

对于初学者来说容易犯的一个错误

dkcode 是环境，有开始标记\begin和结束标记\end，dkcodefile 是命令，只有一行，没有开始和结束标记。

具体应用示例在这里就不展示了，感兴趣的话可以参考本文档“编译与使用详解”一节的源码，位置在：./src/dukang-doc/03-usage.tex

dkcomment

该组件样式风格与 *dkcode* 和 *dkcodefile* 类似，只不过不带代码高亮能力，而是多了一个可以指定标题栏（如果有标题的话）图标的参数，主要用来当作备注栏，用法为：

\begin{dkcomment}<*><[title]><[color]><[fontawesome]>...\end{dkcomment}

*	可选参数，不带星号为普通标题栏，带星号为瘦标题栏，两者只是标题风格不同。
title	可选参数，代码块的标题栏内容，如果没有该参数，则不显示标题栏，并自动忽略 * 参数。
color	可选参数，指定主色调或省略进行自适应，默认为 *awesome*
fontawesome	可选参数，*FontAwesome* 宏包的图标代码或名称，默认为 $\backslash faTheaterMasks$

关于 *FontAwesome* 宏包以及图标代码，可以访问Font Awesome官网，或者如果你的套装里有安装texdoc，可

[①] 本文档“编译使用详解”一节，就是在编译时使用 dkcodefile 直接读取的真实 Makefile 文件。

以使用命令`> texdoc fontawesome5`直接打开本地文档查阅。

使用示例：

瘦标题栏备注框

```
\begin{dkcomment}*[瘦标题栏备注框]
...
\end{dkcomment}
瘦标题栏风格是不带图标的
```

默认标题栏备注框，指定颜色和图标

```
\begin{dkcomment}[默认标题栏备注框，指定颜色和图标][green][\faTree]
...
\end{dkcomment}
指定颜色green和图标\faTree
```

```
\begin{dkcomment}
...
\end{dkcomment}
省略标题参数的话，就没有标题栏
```

一个发挥想象力的用法❤

代码在这里

```
\begin{dkcomment}[%
  \foreach \x in {39,36,...,3}{%
    \fontsize{\x}{1em}\faTree%
  }%
  \foreach \x in {3,6,...,39}{%
    \fontsize{\x}{1em}\faTree%
  }]%
  [cyan]%
  [\fontsize{42pt}{1em}\faTree]%
  一个发挥想象力的用法
\end{dkcomment}
```

上面的 *foreach* 循环控制命令需要用到 *pgffor* 宏包，ACV-Dukang已经帮你引用好了。
行尾的% 表示忽略后面的空格，意思就是告诉 LaTeX 引擎本行和下面的行是同一行。

dkcodebox

这个组件模拟行内的短命令行，类似于 dkbutton，但是可选开关有命令提示符，使用星号 * 控制，带星号表示有命令提示符和关键字高亮。用法为：

```
\dkcodebox<*>{text}
```

例如（开始作死）：

\dkcodebox{sudo rm -rf /} ➔ sudo rm -rf /

\dkcodebox*{sudo shutdown 0 -P} ➔ > sudo shutdown 0 -P

dkmeasure

呐，这个组件就比较有意思了，其本质是对 *layouts* 宏包里面几个命令的简单封装，当你想实际打印检查某个值，或者换算单位时，作为调试使用的话非常方便，比如表格的某个列到底有多宽，当前行距是多少，1em 等于多少 pt 等。用法为：

\dkmeasure<*><[unit]>{macro}

*	可选参数，带 * 显示被测目标名称，不带 * 只显示测量或转换结果。
unit	可选参数，用于指定长度单位，可以是 cm, mm, pt 等，默认为 cm.
macro	必选参数，需要打印的宏或命令，例如\textwidth, \baselineskip, 1em, 20pt 等。

例如：

\dkmeasure*[cm]{\textwidth} ➔ \textwidth > 18.19742cm

\dkmeasure*[mm]{\baselineskip} ➔ \baselineskip > 6.39647mm

\dkmeasure{1em} ➔ 0.35141cm

更直观一些的应用场景，比如打印 *tcolorbox* 环境每个部分的宽度：

\linewidth > **12.89816cm**

\linewidth > 5.94914cm | \linewidth > 5.94914cm

或者表格每个单元格的尺寸：

第一列	第二列	第三列
那啥来着？ \linewidth > 4.99928cm	执着而不急成本，不为索取只为陶醉 \linewidth > 5.99915cm	ACV-Dukang \linewidth > 4.99928cm

甚至某句话的宽度：

\newlength{\x}\settowidth{\x}{再这么熬夜下去就要变成熊猫了！ }\dkmeasure{\x}

5.27115cm

dirtree

针对这个宏包，ACV-Dukang并没有作任何封装工作，因为其本身功能就已经做的足够好了，之所以出现在这里，是我觉得非常有必要向你推荐。具体用法可以参考本文档“结构设计”章节的源码。

以上所有组件，基本可以满足日常创作，特别是科技类和 IT 类作品的创作需要，当然我会持续不断完善和扩展ACV-Dukang的功能，相信后续组件会更加丰富。

更新与维护

ACV-Dukang目前是基于Awesome-CV 2022-04-28 Master版本进行的开发，在Awesome-CV没有太大变动的情况下，应该能够通过下载最新版awesome-cv.cls覆盖旧版来直接升级，当然我也会在不断推进ACV-Dukang的同时随时跟进Awesome-CV的变化情况，并且做适应性的调整，直至代码贡献度提高到一定程度后，会考虑将ACV-Dukang重新封装成为一个独立的LaTeX启动项目。