关于 tlmgr 使用方法的简介*

Zhonghao Sun†

2020.4.20

*https://www.github.com/syvshc/tlmgr-intro-zh-cn

†syvshc@foxmail.com

前　言

这个文档是对 `texdoc tlmgr` 的翻译以及补充. 由于翻译水平与专业知识有限, 文档中有一些词汇和语句翻译的并不好, 如果您有更好的翻译或者纠正我的错误, 可以在该项目处提出 issue 或者 PR.

`tlmgr` 管理着 TEX Live 的安装, 包括软件包以及配置选项. 如果还没有安装 TEX Live, 可以参考 install-latex-guide-zh-cn 与 texlive-zh-cn 进行安装.

TEX Live 是由一些最高级别的**安装方案** (schemes) 构成的, 每一种方案中包含一些**集合** (collections) 和**软件包** (packages), 而每一个集合中也包含了一些软件包, 软件包中包含了一些可以看到的文件. 在安装 TEX Live 的时候, 我们可以选择在不同的级别来进行安装和管理. 比如在安装的时候可以只选则安装最小的安装方案 `minimal scheme (plain only)`, 以及一部分语言和功能的集合, 比如 `Chinese` 与 `LaTeX additional packages`

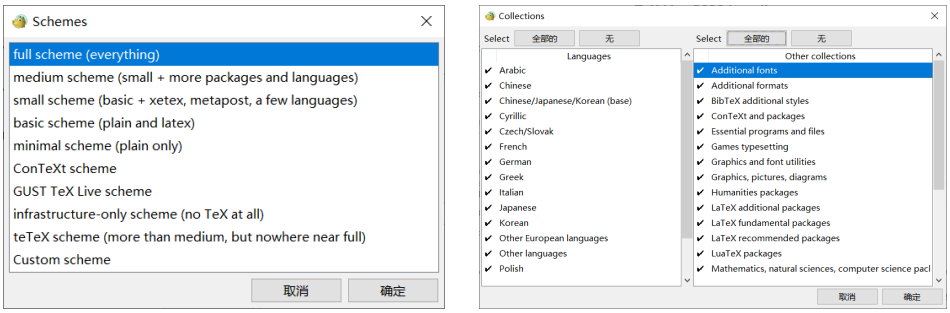

图 1: Schemes 与 collections

使用 `tlmgr` 管理这些安装方案, 集合和软件包是很有必要的, 但是 `tlmgr` 官方文档 过于冗长, 有很多一般用户接触不到的功能, 在这里将它简化, 拿出一些比较基础使用的命令来介绍.

`tlmgr` 也提供了图形界面: `tlshell`, `tlcockpit`, 在 Mac 上, 有 TEX Live Utility, 这些都需要启动另外的程序, 这篇简介只介绍命令行下的使用, 关于图形界面的使用可以看

https://tug.org/texlive/doc/tlmgr.html#GUI-FOR-TLMGR.

目录

1 基本格式与说明

1.1 tlmgr 命令的基本格式:

tlmgr [*global options*] <*action*> [*action-specific options*] [*operand*]

1.2 文档记号说明

(1) tlmgr 的**选项** (option) 分为全局选项与特定命令的选项, 它们一般以 - 或 -- 开头. 但是所有的选项可以在一条命令的任意位置, 按任意顺序调用. 一条命令中第一个不是选项的参数会称为这条命令的主要**操作** (action).

(2) 在所有的情况中, -option 与 --option 等价.

(3) 文档中被中括号 [arg] 框起来的为可选参数, 如 install [*option*],

(4) 文档中被尖括号 <arg> 框起来的为必选参数, 如 info <*pkg*>,

(5) 文档中被竖线 | 分隔的参数为 n 选 1, 如 -repository <*url*|*path*> 表示选项 -repository 后面可以选择远端地址 *url* 或本地位置 *path*,

(6) 文档中用斜体标出的参数 *arg* 表示参数的类型, 用直立体标出的参数 arg 表示参数需要直接填 arg, 比如 backup <*pkgs*|-all> 表示 backup 操作后可以跟参数 -all 或软件包类型的参数.

2 例子

在成功安装 TEX Live 后, 可以使用一些 `tlmgr` 上的常用配置.

`tlmgr option repository ctan`

告诉 `tlmgr` 它可以从一个附近的 CTAN 镜像去获取最近的更新.

`tlmgr update -list`

列出所有可以被更新的内容.

`tlmgr update -all`

更新全部的安装包.

`tlmgr update -self`

升级 `tlmgr` 本身.

`tlmgr info <pkgs>`

列出 `<pkgs>` 的详细信息, 比如它们的安装状态, 版本号, 介绍等等.

`tlmgr update –reinstall-forcibly-removed –all`

当使用 `tlmgr` 升级宏包到一半时因为意外终止, 运行该命令来继续没有完成的升级, 详见 `update` 的 `-reinstall-forcibly-removed` 选项.

3 全局选项 (GLOBAL OPTIONS)

`-help`, `-h`, `-?`

这些选项可以显示任何一个操作的参考文档.

`-version`

显示 TeX Live 发行版以及 `tlmgr` 本身的版本信息, 如果同时指定了 `-v` 选项, TeX Live Perl 模块的版本号也会被显示.

`-q`

抑致输出信息的生成.

`-v`

显示调试信息, 重复使用 `-v` 来显示更多的调试信息.

`-command-logfile` <*file*>

`tlmgr` 把所有程序 (`mktexlr`, `mtxrun`, `fmtutil`, `updmap`) 的输出都记录在了一个 log 文件中, 这个文件默认在 `TEXMFSYSVAR/web2c/tlmgr-commands.log`. 这个选项允许用户把日志存放在 <*file*> 中.

`-package-logfile` <*file*>

`tlmgr` 把所有对软件包的操作 (`install`, `remove`, `update`, failed updates, failed restores) 记录在一个 log 文件中, 这个文件默认在 `TEXMFSYSVAR/web2c/tlmgr.log`. 这个选项允许用户把日志存放在 <*file*> 中.

`-pause`

这个选项让 `tlmgr` 在退出之前等待用户输入. 可以有效地防止 Windows 10 中命令行窗口在运行后直接消失.

`-repository` <*url*|*path*>

用来临时修改当前命令的仓库位置, 可以选择远端位置 <*url*> 或者本地位置 <*path*>, 如果想更改默认仓库位置, 可以使用 `tlmgr option repository`, 见 `option`.

4 操作 (ACTIONS)

4.1 info

使用方法:

`tlmgr info [collections|schemes|pkgs]`.

如果不带参数, 将会列出所有软件包仓库中的软件包, 把哪些已经被安装的软件包加上前缀 `i`.

如果带参数 `collections` 或 `schemes`, 将会列出全部的集合或安装方案, 而不列出软件包, 同样地, 把已经安装的集合或安装方案加上前缀 `i`.

如果带了任意其他的参数, 将会依次列出 `<pkgs>` 中的每一个软件包的信息: 名称 (name), 分类 (category), 简短和详细的介绍 (short and long description), 软件包大小 (size), 安装状态 (status), 以及 TeX Live 中的版本号. 如果这个软件包没有在本地安装, 那么 `tlmgr` 将会在远端仓库去查找它.

对于普通的软件包 (除了集合与安装方案), 软件包大小是分 4 个部分显示的 (run/src/doc/bin). 对于集合, 显示的是所有直接**从属** (dependent) 于它的软件包的大小总和, 但是不包括从属的集合. 对于安装方案, 显示的是所有直接从属于它的集合与软件包的大小总和.

如果在本地与线上都没有完整的匹配结果, 那么 `tlmgr` 将会使用 `search` 操作寻找名称与它相关的软件包, 见 `search` 小节.

`info` 操作也会显示从 TeX Catalogue 获取的信息: 软件包版本 (package revision), 日期 (date), 以及许可证 (license). 不过这些信息, 尤其是软件包版本, 获得的信息只能用做参考, 这是由于不同部分的更新有时间差.

旧操作 `show` 与 `list` 已经被合并到了 `info` 操作中, 但是为了兼容性这两个操作仍然可用.

`info` 的特有选项:

`-list`

如果 `-list` 与一个软件包一起指定, 那么将会同时列出这个软件包包含的文件, 包括那些特定平台上的从属包. 如果与 `collections` 或者 `schemes` 一起指定, 那么 `-list` 将会以一个相似的方式输出依赖关系.

`-only-installed`

如果指定这个选项, 那么将不会使用安装源的信息, 只会显示本地已经安装的软件包, 集合或安装方案.

`-only-remote`

只列出远端仓库中的软件包, 下面是一条测试哪些软件包在某远端仓库中可用的命令:

```
tlmgr -repo <url> -only-remote info
```

注意 `-only-installed` 与 `-only-remote` 不能同时使用.

4.2 search

使用方法:

tlmgr search <*what*>.

默认状态下, tlmgr 会在所有本地安装的软件包中搜索名字, 短描述与长描述中是否含有参数 <*what*>, 其中 <*what*> 被解释为一个 (Perl) 正则表达式.

search 的特有选项:

-file

列出含有 <*what*> 的文件名 (含路径), 比如使用

tlmgr search -file amsmath

那么包含在文件夹 amsmath 下的文件也会被显示, 无论它们的文件名是否含有 amsmath.

-global

在安装介质 (installation medium) 的 TeX Live Database 中搜索, 而不是本地安装.

-word

严格匹配软件包的名字与描述 (并不是文件名). 比如使用

tlmgr search -word table

就不会匹配到含有 tables 的软件包, 除非它同时含有一个 table.

-all

匹配所有的项目: 软件包名, 长短描述与文件名 (含路径).

4.3 install

使用方法:

tlmgr install <*pkgs*>.

安装 <*pkgs*> 中的每一个软件包, 除非某软件包已经被安装过了. 默认状态下所有指定的的软件包的从属文件也会被同时安装. install 操作不会更改已经安装的软件包, 如果想使用最新版本的软件包, 可以使用 update 小节介绍的 update 操作.

install 的特有选项:

-dry-run

在终端显示将要执行的操作, 而不进行安装.

-reinstall

重新安装一个软件包 (包括对集合的依赖), 即使它看起来已经安装了, 比如它已经存在于 TLPDB (TEX Live Package Database) 中. 这个选项对于恢复在层级中不小心删除的软件包很有用.

重新安装时, 只遵循对普通包的依赖 (不遵循类别 Scheme 或 Collection 的依赖)[1].

-with-doc, -with-src

install-tl 给了一个 "不安装文档/源文件" 的选项, 但是我们不推荐使用这个选项. (默认状态下会安装所有的文件). 如果用了这个选项, 那么当用户想获得软件包的文档或者源文件的时候, 可以使用这两个选项与 –reinstall, 用 fontspec 宏包为例:

```
tlmgr install -reinstall -with-doc -with-src fontspec
```

-no-depends

不安装软件包的从属. (默认状态下, 安装一个软件包会保证它包含它所有的从属.)

注意: install 操作并不会自动向系统目录中添加新的**符号链接** (symlinks), 需要自行运行

```
tlmgr path add
```

如果您想使用这个功能并且想添加新的符号链接, 可以阅读 path 操作.

4.4 update

使用方法:

tlmgr update <-all|*pkgs*>

把参数 <*pkgs*> 中的宏包升级到安装源中最新的可用版本, 必须指定 <-all> 或至少一个软件包.

[1] When re-installing, only dependencies on normal packages are followed (i.e., not those of category Scheme or Collection).

表 1: update 的几种关系

```
tlmgr update -self       # 仅升级 tlmgr 本身
tlmgr update -self -all  # 升级 tlmgr 本身和全部可升级软件包
tlmgr upadte -force -all # 升级全部软件包但是不升级 tlmgr
                         # 最后一个有风险，不建议使用！
```

update 的特有选项:

`-all`

升级全部的软件包 (除了 tlmgr 本身). 如果把 tlmgr 也列出安装范围将会报错, 除非同时指定了 -force 与 -self 选项. (见下)

除了对已安装的软件包进行升级之外, 对本地安装的集合的升级 (默认是) 将它与服务器上的集合进行同步, 无论是新增还是移除.

类似地, 如果服务器上的一个本地已经安装的集合新增了一个软件包, 那么它将会被加入本地安装, 这被称作 "auto-install", 同时会在使用选项 -list 的时候被显示.

在集合从属软件包的检查中, 如果有一些被用户强制删除 (forcibly removed), 也就是用户使用对它们使用了 tlmgr remove -force 命令 (见 remove 小节). 如果想重新安装任何一个被强制删除的软件包, 要使用 -reinstall-forcibly-removed 选项.

重申一遍: 自动新增与移除是完全通过集合之间的比较完成的. 因此, 如果用户手动安装了一个稍后将被服务器移除的软件包, tlmgr 不会注意到它, 也不会在本地移除它. (尽量不要主动进行架构重组, 因为 TLPDB 不记录软件包的来源仓库[2])

如果用户想在最近的更新中忽略某些软件包, 可以使用下面的 -exclude 选项.

`-self`

如果有更新的 tlmgr 版本存在, 那么升级 tlmgr 本身, 即 tlmgr 的基础架构. 在 Windows 设备上这个选项也会升级 TeX Live 自带的 Perl 解释器.

如果这个选项与 -all 一起被指定, 那么 tlmgr 将会先升级自身, 如果成功了, tlmgr 将会自动重启并执行接下来的升级.

表 1 展示了 update 中 -self 与 -all 的关系.

`-dry-run`

在终端显示将要执行的操作, 而不进行安装. 这个选项有比 -list 更详细的显示内容.

`-list [pkgs]`

简略地列出即将被更新, 新增或移除的软件包, 但不做实质的更改. 如果同时指定了 -all, 那么将列出所有可用的更新. 如果同时指定了 -self 但是不指定 -all, 那么只会列出可更新的核心软件包 (tlmgr, TeX Live 的核心架构, Windows 上的 Perl 等等).

[2]It has to be this way, without major rearchitecture work, because the tlpdb does not record the repository from which packages come from.

如果不指定 -all 或 -self, 但是给出了具体的软件包 [pkgs], 那么将检查 [pkgs] 中的可更新软件包并列出.

如果不指定 -all 或 -self, 同时也没有给出 [pkg], 那么 tlmgr 就会假定使用了 -all, 也就是说 tlmgr update -list 与 tlmgr update -list -all 是一样的.

-exclude <pkg>

在更新程序中排除软件包 <pkg> 本身以及所有平台特定的软件包, 比如

```
tlmgr update -all -exclude a2ping
```

将不会升级 a2ping, a2ping.i386-linux, 以及任何其它的 a2ping.xxx 软件包.

如果这个选项指定了一个宏包, 程序将会在以下情况报错并退出: <pkg> 将被自动安装, <pkg> 将被自动移除, 或者 <pkg> 是被手动删除并需要被重新安装[3]的时候. -exclude 选项不支持这些情况.

-no-dependent

如果用户升级一个软件包, 正常情况下所有依赖的软件包在必要的情况下都会被升级. 这个选项会抑致这个行为.

-reinstall-forcibly-removed

在通常情况下 tlmgr 不会自动安装那些被用户强制删除的软件包, 比如被 remove -force 删除, 在之前的升级中被选项 -no-auto-install 禁止安装, 或者在升级操作中意外终止时正在升级的软件包.

-no-auto-remove [pkgs]

默认状态下, tlmgr 将会尝试删除存在于集合中但是在服务器中已经被删除的软件包. 这个选项会阻止这些删除操作, 如果同时指定 -all 则范围为全部软件包, 如果指定 [pkgs], 则范围为给定的软件包. 这可能导致 TEX 安装的不一致, 因为软件包的作者可能重命名软件包或者改变软件包的位置, 所以这个选项不推荐使用[4].

-no-auto-install [pkgs]

默认状态下, tlmgr 会自动安装服务器上的新软件包, 与指定选项 -all 的效果相同. 这个选项阻止了所有的自动安装, 如果同时指定 -all 则范围为全部软件包, 如果指定 [pkgs], 则范围为给定的软件包.

此外, 在 -no-auto-install 选项下完成升级之后, 那些应该被自动安装的软件包会被认为成**被用户强制删除的**. 所以, 如果 foobar 是唯一一个服务器上新增的软件包, 那么

```
tlmgr update -all -no-auto-install
```

等价于

```
tlmgr upadte -all
tlmgr remove -force foobar
```

同样地, 因为软件包的作者可能重命名软件包或者改变软件包的位置, 所以这个选项不推荐使用.

[3] reinstallation of a forcibly removed package.

[4] This can lead to an inconsistent TeX installation, since packages are not infrequently renamed or replaced by their authors. Therefore this is not recommended.

`-backup`, `-backupdir <dir>`

这两个选项会在**升级之前**创建软件包的备份, 也就是说会备份升级之前最新的软件包, 如果不指定这两个选项, 那么将不会创建备份. 如果指定了 `-backupdir` 同时 `<dir>` 是一个可写且有足够空间的路径, 那么 `tlmgr` 将会在这个路径下创建备份文件. 如果仅指定了 `-backup`, 那么备份将会被创建在 `option` 小节中设置的位置, 默认在 TeX Live 根目录下的 `texlive/<year>/tlpkg/backups` 中.

注: tlmgr 文档 原文为: If neither option is given, no backup will made. 但是 `tlmgr` 在 TeX Live2010 以后将自动备份关键词 `autobackup` 默认设置为 1, 也就是说, 即使不使用 `-backup` 选项, 也会自动将软件包备份在默认位置, 方便用户使用 `restore` 操作回滚. 更新日志见 texlive-en

`-force`

强制升级所有软件包, 不包括 `tlmgr` 本身, 除非同时指定了 `-self` 选项. 不推荐.

同样地, `update -list` 将会无视这个选项按它本身该有的样子显示.

注意: 如果本地安装的软件包的版本高于服务器端的时候 (比如设置的镜像站过时了), `tlmgr` 会忽略这个软件包而不进行降级.

4.5 restore

使用方法:

```
restore [pkg] [revision]
restore <-all>
```

这个操作将从之前的备份中回滚软件包版本.

如果指定了 `<-all>`, 那么将会从可找到的备份中回滚所有的软件包版本.

如果不指定 [*pkg*] 与 [*revision*], 那么将列出所有软件包的可用备份的版本.

如果指定了 [*pkg*] 但是没有指定 [*revision*], 那么就列出软件包 [*pkg*] 的所有可用的备份版本, 比如使用 `tlmgr restore fancyhdr`, 那么将会得到如下形式的输出

```
Available backups for fancyhdr: xxxxx (yyyy-MM-dd HH:mm)
```

其中 `xxxxx` 为 `fancyhdr` 可用备份的版本号.

当且仅当同时指定 [*pkg*] 与一个可用的版本号 [*revision*] 时, `tlmgr` 会从备份中将 [*pkg*] 回滚到版本 [*revision*].

`restore` 的特有选项:

`-all`

尝试从从可找到的备份中回滚所有的软件包版本. 其他的非选项 (non-option) 的参数, 如 [*pkg*] 将不可用.

`-backupdir <dir>`

指定一个供 `tlmgr` 获取备份的路径, 如果不使用这个选项, 那么 `tlmgr` 就会使用从 TLPDB 的配置文件.

`-dry-run`

在终端显示将要执行的操作, 而不进行回滚.

`-force`

强制进行回滚[5].

`-json`

将输出打印为 JSON 样式.

[5]原文为: Don't ask questions.

4.6 remove

使用方法:

```
tlmgr remove <pkgs>
```

移除 <*pkgs*> 中给出的软件包. 如果要移除一个集合, 那么将移除这个集合下的所有软件包. 除非指定了 -no-depends, 入市不移除这个集合下的任何集合. 然而, 当移除一个软件包的时候, 它的从属永远不会被移除.

remove 的特有选项:

-dry-run

在终端显示将要执行的操作, 而不进行实质的删除.

-no-depends

不移除从属软件包.

-all

卸载整个 TEX Live, 会询问用户是否删除, 除非指定了 -force 选项. 该命令可以在 Linux 下正常工作. 在 Windows 下, tlmgr remove -all 会返回

```
Please use "Add/Remove Programs" from the Control Panel to removing
TeX Live!
```

如果在 Windows 下卸载 TEX Live. 可以在命令行中运行

kpsewhich -var-value TEXMFROOT

查看 $TEXMFROOT 的路径, 然后运行 $TEXMFROOT\tlpkg\installer\uninst.bat 来进行卸载.

-force

默认状态下, 移除一个软件包或者一个集合的从属集合是不允许的. 使用这个选项可以无条件地铁软件包, 要谨慎使用.

如果使用 tlmgr remove -force 移除了一个软件包, 那它在 tlmgr update -list 中被认为是 forcibly removed.

4.7 option

使用方法:

```
tlmgr option
tlmgr option help
tlmgr option <key>
tlmgr option <key> [value]
```

(1) 第一种形式列出当前的全局 TeX Live 设置, 带有一个简短的介绍, 每一种设置里括号中的词就是修改这个设置需要的 key, 比如要修改备份文件的路径可以使用 `tlmgr option backupdir [dir]`.

```
Directory for backups (backupdir): tlpkg/backups
```

(2) 第二种形式与第一种很像, 它会在 `tlmgr option` 的基础上列出可以被设置但是当前没有值的选项,

(3) 第三种形式是列出 <*key*> 对应的值,

(4) 第四种形式将 <*key*> 的值设置为 [*value*]

表 2 中是一些可用的 <*key*> 值. 使用 `tlmgr option help` 来获得更详细的列表

表 2: `tlmgr option` 的部分可用选项

autobackup	升级操作是否备份,
backupdir	备份文件的位置,
repository	默认的软件包仓库,
docfiles	安装文档文件,
srcfiles	安装源文件,
sys_bin	通过 path 操作链接到的可执行文件的目录
sys_man	通过 path 操作链接到的手册文件的目录
sys_info	通过 path 动作链接到的信息文件的目录
desktop_integration	仅 Windows: 创建开始菜单的快捷方式
fileassocs	仅 Windows: 改变文件关联
multiuser	仅 Windows: 为所有用户安装

一个常见 `option` 的用法是它可以在从 DVD 安装以后, 永久地改变安装的设置来通过互联网获得更新. 比如, 用户可以使用

```
tlmgr option repository ctan
```

来从一个附近的 CTAN 镜像来获得更新.

但是中国大陆用户可能无法获取到实际 “附近” 的镜像站, 很多时候需要用户手动指定大陆地区的源. 详细信息见 中国大陆地区的源 一节.

`docfiles` 与 `srcfiles` 分别控制着每个软件包的文件组 (文档, 源文件)[6] 这两个 <*key*> 的默认值都是 `1`, 如果用户的硬盘空间不足, 或者只是想做一个最小的安装测试, 那么每一个都可以被修改成 `0`, 这样就不会再下载这些相关的文件了.

[6] The `docfiles` and `srcfiles` keys control the installation of their respective file groups documentation, sources; grouping is approximate) per package.

`sys_bin`, `sys_man`, `sys_info` 选项是在 Unix 系统上使用的, 控制着对于可执行文件, 信息文件, 手册页面链接的生成. 详细信息见 path 操作

其余的选项控制着 Windows 安装的表现.

- 如果设置了 `desktop_integration`, 那么一些软件包将会把文件安装在开始菜单的 `tlmgr gui` 中, 比如文档等等.
- 如果设置了 `fileassocs`, 将会建立 Windows 下的文件关联. 见 `postaction` 操作.
- 如果设置了 `multiuser`, 那么对注册表和菜单的调整将针对系统上的所有用户, 而不仅仅是当前用户.

这三个选项默认都是开启的.

4.8 backup

使用方法:

```
tlmgr backup <pkgs|-all>
```

如果没有指定 -clean 选项, 那么这个操作会在默认位置, 为 <*pkgs*> 中的每一个软件包创建一个备份文件, 如果使用了 -all 选项, 那么将创建全部软件包的备份.

备份文件的位置由 -backupdir <*dir*> 指定, 前提是 <*dir*> 存在且可写. 如果没有指定 -backupdir, 那么就会使用 TLPDB 设置中的 backupdir 路径. 如果二者皆空, 那么就不会创建备份. 关于 backupdir 的信息可以参考 update 操作的 -backupdir 选项.

如果指定了 -clean 选项, 那么将会删除备份文件, 而不是创建备份. 这个选项有一个可选的整数值 N 来指定清理时保留的备份数量. 如果没有指定 N, 那么将会使用 autobackup 的值, 它在 TLPDB 中的默认值为 1, 如果二者皆空, 那么将会报错.

backup 的特有选项:

-backupdir <*dir*>

临时盖 TLPDB 中 backupdir 的值, 参数 <*dir*> 必须指定, 这是备份文件存放的路径, 它必须要存在且可写.

-all

如果没有指定 -clean 选项, 那么创建一个 TeX Live 安装过的所有软件包的备份, 这会消耗大量的存储空间与时间, 如果指定了 -clean 选项, 所有的备份将被删除.

-clean[=N]

删除备份目录 backupdir 中的旧备份, 而不创建备份. 可选整数参数 N 会临时覆盖 TLPDB 中的 autobackup 的值, 如果使用这个选项, 那么必须要指定 -all 或者一列软件包.

-dry-run

在终端显示将要执行的操作, 而不进行备份.

A 中国大陆地区的源

如果要更换大陆地区的源, 需要使用[7]

```
tlmgr option repository <url>/tlnet
```

需要将 <*url*> 更换为表 3 中的 TeX Live 源地址.

表 3: 大陆地区目前可用的源 (名称按拼音排序)

镜像站名称	TeX Live 源地址
阿里云	https://mirrors.aliyun.com/CTAN/systems/texlive/
北京交通大学	https://mirror.bjtu.edu.cn/ctan/systems/texlive/
北京理工大学	https://mirrors.bit.edu.cn/CTAN/systems/texlive/
北京外国语大学	https://mirrors.bfsu.edu.cn/CTAN/systems/texlive/
重庆大学	https://mirrors.cqu.edu.cn/CTAN/systems/texlive/
东莞理工学院	https://mirrors.dgut.edu.cn/CTAN/systems/texlive/
哈尔滨工业大学	https://mirrors.hit.edu.cn/CTAN/systems/texlive/
华为云	https://mirrors.huaweicloud.com/CTAN/systems/texlive/
华中科技大学	http://mirrors.hust.edu.cn/CTAN/systems/texlive/
兰州大学	https://mirror.lzu.edu.cn/CTAN/systems/texlive/
南京大学	https://mirrors.nju.edu.cn/CTAN/systems/texlive/
清华大学	https://mirrors.tuna.tsinghua.edu.cn/CTAN/systems/texlive/
上海交通大学	https://mirrors.sjtug.sjtu.edu.cn/ctan/systems/texlive/
上海科技大学	https://mirrors.geekpie.club/CTAN/systems/texlive/
腾讯云	https://mirrors.cloud.tencent.com/CTAN/systems/texlive/
中国科学技术大学	https://mirrors.ustc.edu.cn/CTAN/systems/texlive/

[7]这一节的内容来自 install-latex-guide-zh-cn