包管理器杂谈

2024-11-29

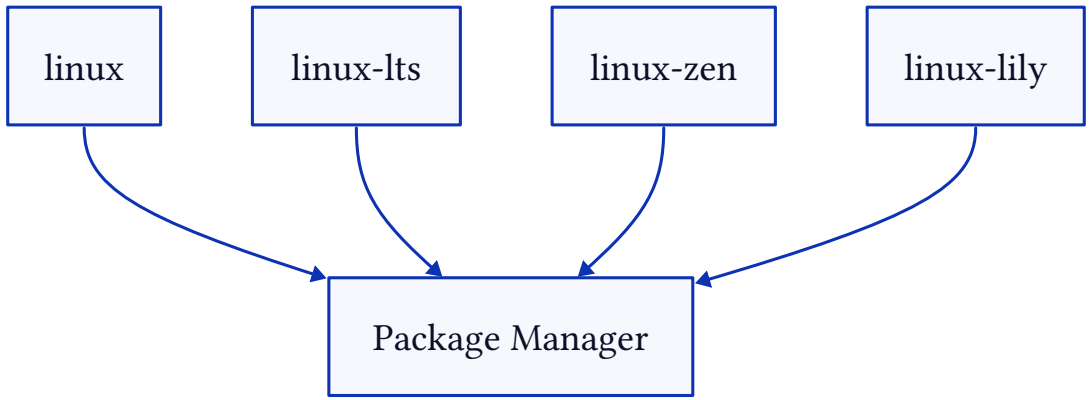
前言

使用体验

- Makefile 有多难用
- 直观感受：只需要知道名字，一行代码，安装并管理
- 环境变量
- 源，依赖
- 查找表
- …

地位

Package managers are at the core of Linux distributions. —— Jonathan Corbet, LWN.net



发行版和包管理器

- Arch Linux: *pacman*
- Debian: *apt*, *apt-get*
  - Ubuntu: += *Snap*
- CentOS: *yum*
- Fedora: *dnf*
- Gentoo: *Portage*
- NixOS: *Nix*
- ~~Windows: *Chocolatey*, *Scoop*~~
- …

- Flatpak

数量统计

- https://repology.org/repositories/statistics/newest

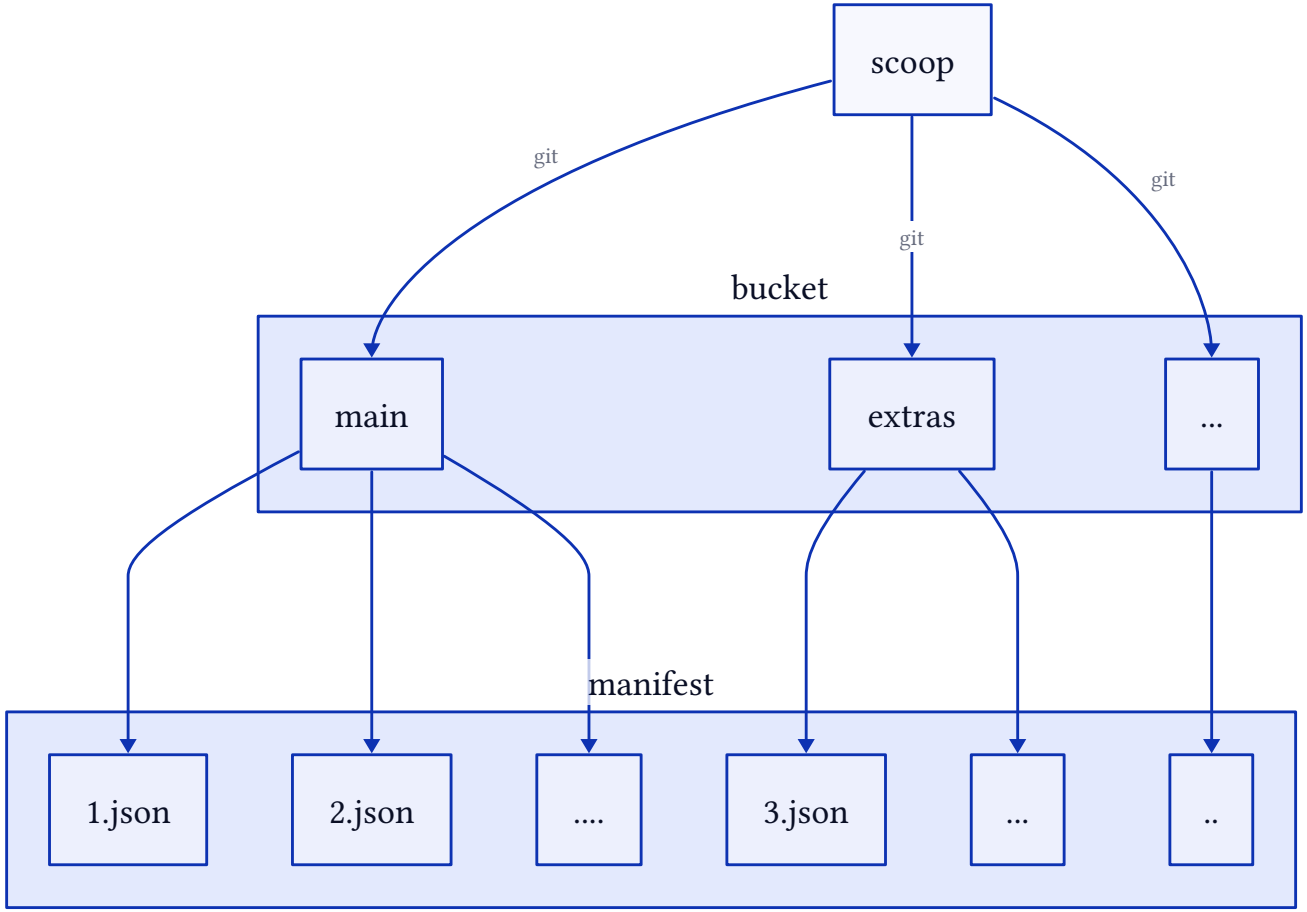
包管理器是如何工作的？

Scoop

github: ScoopInstaller/Scoop

- Scoop is a command-line installer for **Windows**. (PowerShell)
- 声明式

```json
{
    "version": "1.6.4",
    "description": "GIF encoder based on libimagequant (pngquant).",
    "homepage": "https://gif.ski",
    "license": "AGPL-3.0-or-later",
    "url": "https://gif.ski/gifski-1.6.4.zip",
    "hash": "dc97c92c9685742c4cf3de59ae12bcfcfa6ee08d97dfea26ea88728a388440cb",
    "pre_install": "if (!(Test-Path '$dir\\config')) { New-Item '$dir\\config' }",
    "bin": "gifski.exe",
    "checkver": "For Windows.*?gifski-([\\d.]+)\\.zip",
    "autoupdate": {
        "url": "https://gif.ski/gifski-$version.zip"
    }
}
```

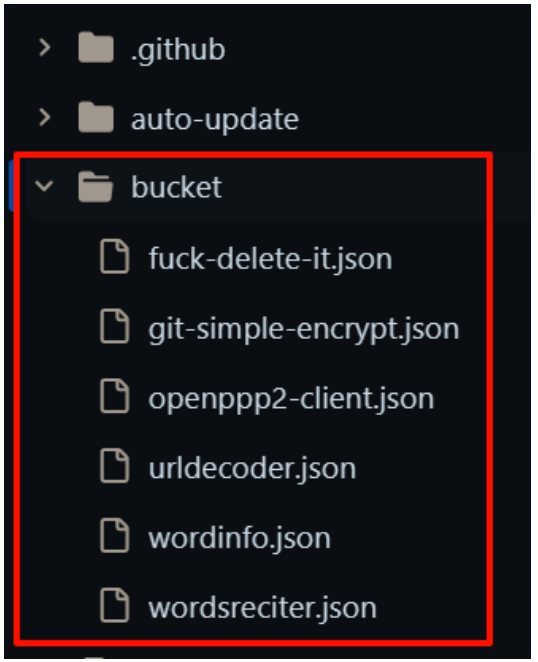

scoop
git
git
git
bucket
main
extras
...
manifest
1.json
2.json
....
3.json
...
..

Scoop

Found 1544 buckets.

Bucket
Calinou/scoop-games
matthewjberger/scoop-nerd-fonts
niheaven/scoop-sysinternals
ScoopInstaller/Extras
ScoopInstaller/Java
ScoopInstaller/Main
ScoopInstaller/Nirsoft
ScoopInstaller/Nonportable
ScoopInstaller/PHP
ScoopInstaller/Versions
0b1000/bucket

```sh
scoop install git  # uninstall, search
scoop bucket add extras
scoop bucket add absx https://github.com/absxsfriends/scoop-bucket
scoop config proxy 127.0.0.1:<port>
```



缺点

- Apps are self-contained units

  They should keep their own copies of any libraries they need to run, and not rely on or interfere with any libraries outside their own install path.—— Scoop Wiki
- PowerShell

apt

- apt（Advanced Packaging Tool）（2014） 是 apt-get（1998） 的较新版本。apt 命令被设计为对用户更加友好的 apt-get 替代方案。
- debian 系维护一个 .deb 包，能够直接使用 dpkg -i 安装。
- apt 处理依赖并下载，交由 dpkg 安装。

.deb 包结构

```
staging
   ├─ DEBIAN
   └─ usr
       ├─ bin
       └─ share
           ├─ project1       #resources of your package
           ├─ applications   #shortcut
           ├─ pixmaps        #default icon
           ├─ icons
           │   └─ hicolor    #icons of various sizes
           ├─ doc
           │   └─ project1   #information about your package
           └─ man            #user manual
               └─ man1       #index
```

ref: Debian package structure

打包上传

- Once you become an **official developer**, you can upload the package to the Debian archive.——Debian Manuals
- Caddy 安装指南？

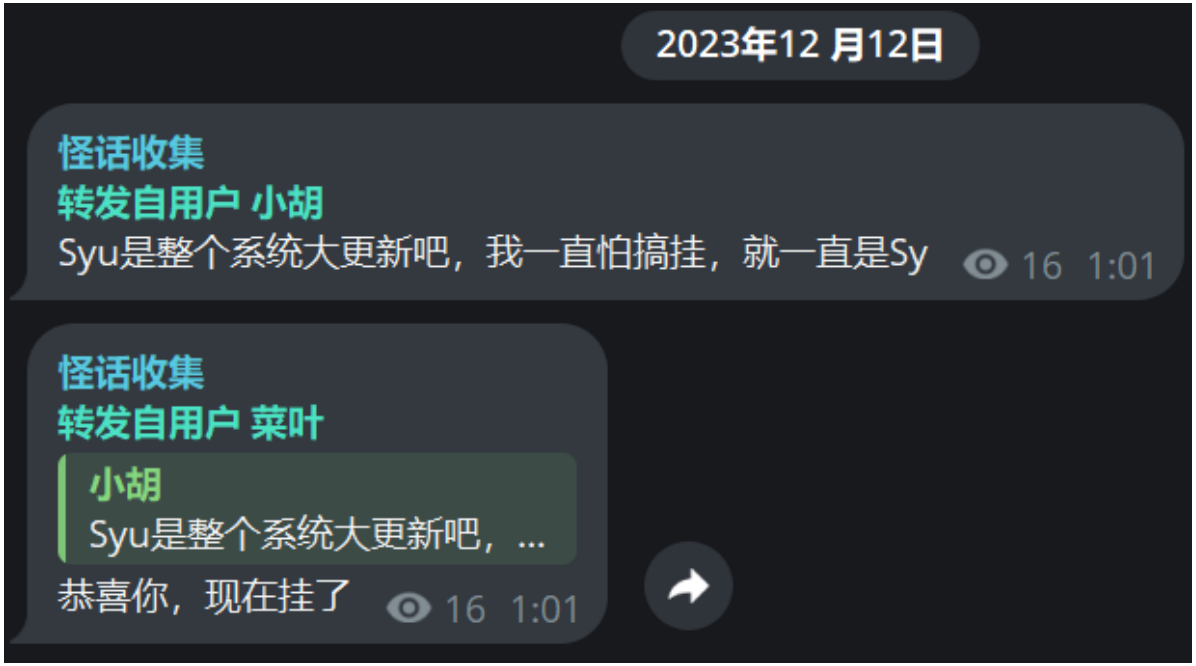
```sh
sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https curl
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | sudo gpg --dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' | sudo tee /etc/apt/sources.list.d/caddy-stable.list
sudo apt update
sudo apt install caddy
```

snap

- Ubuntu 的预装包管理器，/snap
- 守护进程：snapd
- 每个软件包里带有所有依赖
- loop mount，~~btrfs-convert~~

```
root in 🌐 iZj6cffqt5550r30xxrp5vZ in ~ on  master [?] took 41s
❯ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0    7:0    0 63.9M  1 loop /snap/core20/2264
loop2    7:2    0   87M  1 loop /snap/lxd/27948
loop3    7:3    0   87M  1 loop /snap/lxd/28373
loop4    7:4    0 39.1M  1 loop /snap/snapd/21184
loop5    7:5    0 38.7M  1 loop /snap/snapd/21465
loop6    7:6    0 63.9M  1 loop /snap/core20/2318
vda    252:0    0   40G  0 disk
├─vda1 252:1    0    1M  0 part
├─vda2 252:2    0  200M  0 part /boot/efi
└─vda3 252:3    0 39.8G  0 part /var/lib/containers/storage/overlay
```

- pacman（package manager）是 Arch Linux 的官方仓库包管理器。
- 激进，滚动更新，不允许部分更新。



- 依赖的可用性由官方仓库保证。

缺点

- 抽象的参数 (S, Sy, Syu, Syyu, Syyuu; Qtdq, Qtdm; Rns, Rsc)
- 过于严格的审查与缺陷
  - error: xxx: signature from “xxx” is unknown trust
  - 自 2023 年 12 月初 archlinux-keyring 中删除了一个退任的 master key (https://gitlab.archlinux.org/archlinux/archlinux-keyring/-/issues/246) 导致 farseerfc 的 key 的信任数不足，由 GnuPG 的 web of trust 推算为 marginal trust，从而不再能自动信任 archlinuxcn-keyring 包的签名。
- do not support search name only (pacman -Ss ‘^vim-’)

- AUR（Arch User Repository）是 Arch Linux 的用户仓库，由用户自由上传
- 所有包使用 PKGBUILD 描述；AUR Helper 处理依赖并下载 PKGBUILD，交由 makepkg 构建安装。

```
1  # Maintainer: ab5_x <lxl66566@gmail.com>
2
3  pkgname=tdl-bin
4  pkgver=0.17.0
5  pkgrel=1
6  pkgdesc="A Telegram downloader/tools written in Golang"
7  arch=("x86_64" "aarch64" "armv7h")
8  url="https://github.com/iyear/tdl"
9  license=("AGPL-3.0-or-later")
10 depends=()
11 provides=("tdl")
12
```

```
13 source_x86_64=("$pkgname-x86_64::https://github.com/iyear/tdl/releases/download/v$pkgver/tdl_Linux_64bit.tar.gz")
14 source_aarch64=("$pkgname-aarch64::https://github.com/iyear/tdl/releases/download/v$pkgver/tdl_Linux_arm64.tar.gz")
15 source_armv7h=("$pkgname-armv7h::https://github.com/iyear/tdl/releases/download/v$pkgver/tdl_Linux_armv7.tar.gz")
16
17 sha256sums_x86_64=('2d9ac6d36530ba08da44572447120691f5487443a1eb65be189850ddaa6d6c7d')
18 sha256sums_aarch64=('2c34a9255ae7a79a6cac0c74dd72e79d539813f4f39c4f904f940b44b2b30bb5')
19 sha256sums_armv7h=('ea7a126b120682e8130dbe87e07e790c2d9cc5bc41c8ba464cd27d3b5e5f7062')
20
21 package() {
22         cd "$srcdir/"
23         install -Dm755 tdl -t "${pkgdir}/usr/bin/"
24         install -Dm644 LICENSE -t "${pkgdir}/usr/share/licenses/$pkgname/LICENSE"
25 }
```

缺点

- 安全问题
- 没有通用提交 patch/PR 的办法，只能用户自行维护
  - 可以标记过期

- 通用包管理器
- 包描述：函数式 Nix 语言
- hash 存储，ex. /nix/store/l5rah62vpsr3ap63xmk197y0s1l6g2zx-simgrid-3.22.2
  - 隔离
  - 回滚
  - 可复现
- cache：不需要再分 -bin, -git
- NixOS：基于 Nix 的操作系统

使用

```sh
❯ sudo nix-channel --add https://nixos.org/channels/nixpkgs-unstable
❯ sudo nix-channel --update
❯ sudo nix-env -i fish
❯ which fish
/nix/var/nix/profiles/default/bin/fish
❯ nix-shell -p cowsay
[nix-shell:~]# cowsay 456
```

总结

1. 打包：开发者编写 manifest 并测试，集中上传
2. 构建：根据 manifest 构建为 package（服务端构建 / 客户端构建）
   - 依赖、源
3. 安装：交给安装器
4. 卸载（补充）：
   - 移除整个目录
   - 追踪表

包管理器实践

init-script

github: lxl66566/init-script

- 服务器一键脚本
- 安装列表：sudo, wget, curl, rsync, btop, lsof, ncdu, tldr, podman, fzf, make, paru, trojan, base, python-requests, python-pip, pipx, bpm, trojan-go, caddy, hysteria2, fd, mcfly, zoxide, fish, starship, cargo, sd, ripgrep, eza, yazi, neovim, fastfetch, zellij, bat, xh, openppp2, atuin

初代

```python
@log
@mycache_once(name="install")
def install_zoxide():
    match distro():
        case "a":
            pacman("zoxide")
        case _:
            if distro() == "d" and version() < 11:
                rc_sudo(
                    "curl -sS https://raw.githubusercontent.com/ajeetdsouza/zoxide/main/install.sh | bash"
                )
            else:
                basic_install("zoxide")
```

V2

```python
class Package:
    """
    `pm_name`: 一个函数，根据当前架构与包管理器返回系统包名。
    `level`: 优先级，数字越大优先级越高。0 默认不安装，1 在配置充足的系统上安装，2 必定安装。
    `pre_install_fun`：自定义安装前函数，返回 None | bool。如果未设置或返回 False，使用 `pm_install` 安装，返回 None 不安装，否则使用 `install_fun` 安装。
    `install_fun`: 自定义安装函数。如果返回 False，改为使用 pm_install 安装。
    `post_install_fun`: 自定义安装后函数，主要进行一些配置。无论使用 pm_install 还是 install_fun 安装，都会执行这个函数。
    """

    @install_once(name="install")
    def install(self):
        # install dependencies
        for i in self.depends:
            p = packages_list.get(i)
            assert p, f"找不到依赖包 {i}"
            if not SetCache("package_installed").in_set(p.name):
                p.install()
            else:
```

init-script

```
19                log.debug(f"依赖 {p.name} 已经安装，跳过安装")
20
21        cut()
22        print(f"""开始安装 {colored(self.name, "green")}...""")
23
24        name = getattr(self, "pm_name", lambda: self.name)()
25        if not name:
26            name = self.name
27
28        """
29        调用函数，如果函数有且只有一个参数，则将 self 作为参数传入，否则不传。
30        """
31        pre_ret = self._call_with_param_0_or_1(
32            getattr(self, "pre_install_fun", lambda: False)
33        )
34
35        if pre_ret is None:
36            log.warning(f"""{colored(self.name, 'yellow')} 不满足安装条件，安装取消.""")
37            return
38
39        if not pre_ret and check_package_exists(name):
```

init-script

```
40              pm_install(name)
41          else:
42              assert hasattr(
43                  self, "install_fun"
44              ), "跳过了系统包安装，并且找不到自定义安装函数。这可能是您的平台不受支持，或者包管理器版本过低，请开 issue 报告"
45              fun = getattr(self, "install_fun")
46              self._call_with_param_0_or_1(fun)
47
48          self._call_with_param_0_or_1(getattr(self, "post_install_fun", lambda: None))
49
50          SetCache("package_installed").append_set(self.name)
51          print(f"""{colored(self.name, "green")} 安装完成.""")
```

使用

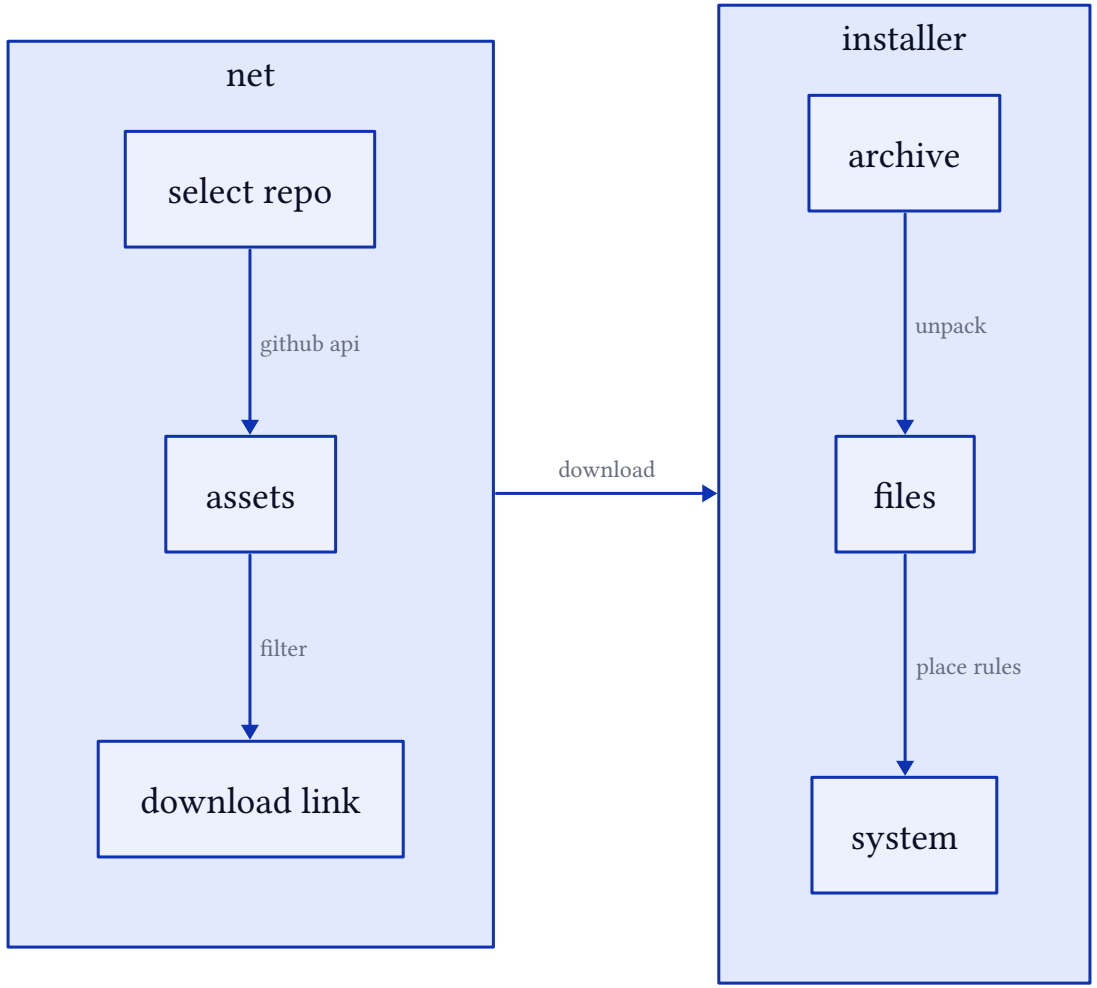
```python
packages_list.add(
    Package(
        "zoxide",
        2,
        pre_install_fun=lambda: distro() == "d" and version() < 11,
        install_fun=lambda: rc_sudo(
            "curl -sS https://raw.githubusercontent.com/ajeetdsouza/zoxide/main/install.sh | bash"
        ),
        post_install_fun=lambda: fish_add_config("zoxide init fish | source"),
        depends=["fish", "sudo"],
    )
)
```

- 不用 nix 的原因？
  - 当时不知道，~~现在不会用~~
  - nix 安装后需要手动重启 shell

github: lxl66566/bpm

- 从 Github Release 中安装 binary
  - 无需处理依赖
- 无需打包，依照既定规则安装

```
1  install (i)          Install packages.
2  remove (r)           Remove packages.
3  update (u)           Update packages.
4  info                 Info package.
5  alias                Alias package. (Windows only; Linux use shell alias instead.)
```


net
select repo
github api
assets
filter
download link
download
installer
archive
unpack
files
place rules
system