

Missing Class Before College

Package Manger, Shell, Vscod and Markdown

Zhaojiacheng Zhou

September 8, 2025

TechJI

Outline

TechJI Introduction

Package Manager

Shell

VS Code

Markdown

Recommendation

TechJI Introduction

Who are we?

- Fans of Computer Science (but not focs)
- Host tech related workshop like linux install party, git, bash, reflow ...



机械键盘
DIY



Reflow
Create your own
electrical cat



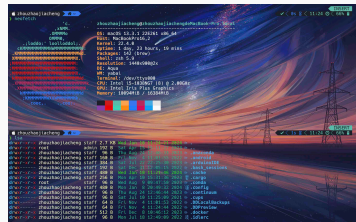
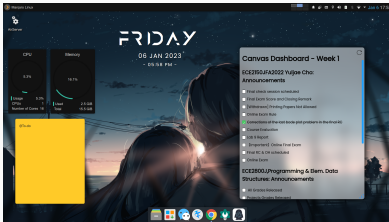
Git
"Handle everything
with speed and
efficiency"



Bash
An SH-Compatible
Shell

Who are we?

- Conduct a bunch of open source project development like course selection community, dancing party software, canvas helper...
- Post online tutorial like terminal beautify...



Package Manager

Introduction to package manager

- A package manager or package management system (PMS) is a collection of software tools that automates the process of installing, upgrading, configuring, and removing computer programs for a computer in a consistent manner.

Problem solved

- **Dependency Hell:** Different software packages require different, and sometimes conflicting, versions of the same shared libraries. Package managers solve this by managing and allowing for the coexistence of multiple library versions.
- **Manual Installation:** Package managers eliminate the need for users to manually download, compile, and install software, which can be a complex and time-consuming process.
- **Synchronization Issues:** They ensure that the list of installed software is always consistent and up-to-date with a central database, preventing conflicts and missing prerequisites that could arise from manual interventions.

How It Works

A package manager operates by interacting with three key components:

- **Packages:** Packages are the fundamental units of a PMS. A package is a file that contains the application, its necessary files, and metadata like the name, version, and dependencies.
- **Repositories:** These are centralized locations or servers where packages are stored. A package manager downloads packages from these repositories.
- **Local Database:** The package manager maintains a local database on the user's system. This database keeps a record of all installed packages, their versions, and their dependencies.

Examples

- Windows: winget, scoop, chocolatey
- MacOS: homebrew, macport
- Linux: pacman, apt, dnf ...

- The mirror/source config of package defines where your package manager fetch remote packages
- It can be customized to improve download speed and availability
- There are many good quality source like tsinghua, ustc...

Exercise: vscode installation

- Windows users: use winget to install vscode
- MacOS users: download homebrew and install vscode
- Linux users: you should now how to do so

You can install vsodium if you value your privacy since it is open source and no one will steal your data and code :)

Exercise: vscode installation (Windows)

1. Check **USTC Mirror** and change your source
2. Proof read `winget -help`
3. Run the following command to install vscode

```
winget install --location  
↪ <path-you-want-to-install>  
↪ Microsoft.VisualStudioCode
```

Exercise: vscode installation (Macos)

1. Download homebrew from [tsinghua mirror](#)
2. Run the following script to install homebrew

```
xcode-select --install
export HOMEBREW_BREW_GIT_REMOTE="https://mirrors.tuna.tsinghua.edu.cn/git/homebrew/brew.git"
export HOMEBREW_CORE_GIT_REMOTE="https://mirrors.tuna.tsinghua.edu.cn/git/homebrew/homebrew-core.git"
export HOMEBREW_INSTALL_FROM_API=1
export HOMEBREW_API_DOMAIN="https://mirrors.tuna.tsinghua.edu.cn/homebrew-bottles/api"
export HOMEBREW_BOTTLE_DOMAIN="https://mirrors.tuna.tsinghua.edu.cn/homebrew-bottles"
git clone --depth=1 https://mirrors.tuna.tsinghua.edu.cn/git/homebrew/install.git brew-install
/bin/bash brew-install/install.sh
rm -rf brew-install
```

Exercise: vscode installation (Macos)

For apple silicon CPU user run following command

```
test -r ~/.bash_profile && echo 'eval
↳ "$(/opt/homebrew/bin/brew shellenv)'" >>
↳ ~/.bash_profile
test -r ~/.zprofile && echo 'eval
↳ "$(/opt/homebrew/bin/brew shellenv)'" >>
↳ ~/.zprofile
```

Exercise: vscode installation (Macos)

For long term substitution of mirror, run following command, also see [this website](#)

```
export HOMEBREW_CORE_GIT_REMOTE="https://mirrors.tuna.tsinghua.edu.cn/git/homebrew/homebrew-core.git"
for tap in core cask command-not-found; do
brew tap --custom-remote "https://mirrors.tuna.tsinghua.edu.cn/git/homebrew/homebrew-${tap}.git"
done
brew update
```


Exercise: vscode installation (Macos)

- Proof read `brew -help`
- Run the following command to install vscode

```
brew install --cask visual-studio-code
```

Exercise: vscode installation (Linux)

1. Check [vscode official website](#) and download
2. For Ubuntu users, I don't recommend you to use snap

Shell

Introduction to shell

- A shell is a command-line interpreter that provides a user interface for accessing an operating system's services.
- It allows users to execute commands, manage files, and run programs through text-based inputs.
- Consider it as a direct communication channel between you and your computer's operating system.

Types of shells

- **PowerShell (Windows):** A task automation and configuration management framework from Microsoft.
- **Bash (Linux/Mac):** The Bourne Again Shell, the default shell on most Linux distributions and older macOS versions.
- **Zsh (Mac/Linux):** An extended version of Bash with additional features like better auto-completion and theme support.

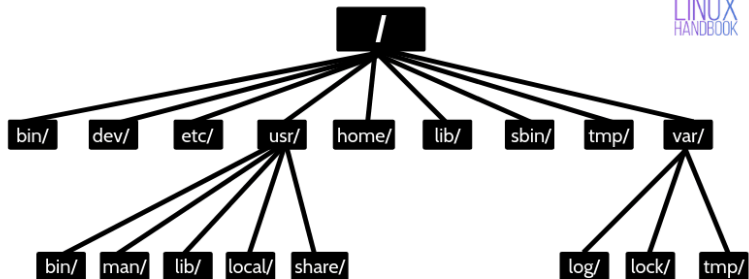
Why zsh?

- Enhanced auto-completion for commands, file paths, and options
- Better customization options with themes and plugins
- Spelling correction and approximate completion
- macOS has made zsh the default shell since Catalina (10.15)

WSL installation

- For MacOS and Linux users, you can take a rest
- For windows users, check the wsl installation manual [wsl.pdf](#)

File organization in Linux



File organization in Linux

- **/**: Root directory - the base of the entire file system
- **/home**: User home directories (your personal files)
- **/etc**: System configuration files
- **/usr**: User programs and support files
- **/var**: Variable data like logs, databases, websites
- **/tmp**: Temporary files
- **/bin**: Essential command binaries
- **/lib**: Essential shared libraries and kernel modules
- **/dev**: Device files
- **/proc**: Process information and system information

For more detailed information, check [this wiki](#)

Basic bash commands

- **pwd**: Print working directory - shows your current location in the file system
- **ls**: List directory contents (files and folders)
- **cd**: Change directory - navigate between folders
- **mkdir**: Create a new directory
- **touch**: Create an empty file or update file timestamps
- **cp**: Copy files or directories
- **mv**: Move or rename files or directories
- **rm**: Remove files or directories
- **cat**: Display file contents
- **echo**: Print text or variables to the terminal
- **>**: Redirect stdout to overwrite file
- **»**: Redirect stdout to append file

Practical examples

```
# Navigate to your home directory
```

```
cd ~
```

```
# List files in long format
```

```
ls -l
```

```
# Create a new directory
```

```
mkdir my_project
```

```
# Navigate into the directory
```

```
cd my_project
```

```
# Create a new file
```

```
touch README.md
```

```
# Copy a file
```

```
cp README.md README_copy.md
```

Practical examples

```
cd ~/my_project
# redirect stdout into files
echo "foo" >foo
echo "barr" >bar
```

```
# concatenate two files
cat foo bar
```

```
# Go to parent path
cd ..
```

```
# Dangerous!!! You'd better use project like
↳ trash-cli
rm -rf my_project
```

Environment Variable

- We know that, ls is at /bin/ls, but we can directly use ls command.
- This is the benefit of PATH environment variable.
- The PATH define where OS finds the executable files.
- You can use **echo \$PATH** to check your current PATH

Environment Variables

- What are environment variables?
- Environment variables, often called **ENVs**, are dynamic values that play a crucial role in controlling the behavior of programs and processes in Linux and other operating systems.

Environment Variables (Examples)

```
F00="foo"
echo $F00
set # display all the ENVs(global as well as local)
env # display all the global ENVs

# local variable won't show, don't use " here
bash -c 'echo $F00'

export F00="foo" # define a global variable

# global variable will show
bash -c 'echo $F00'

unset F00
```

Environment Variables (Examples)

How to set up proxy in shell

`export`

↪ `http_proxy="(http)|(socks5)://127.0.0.1:<port>"`

`export https_proxy="(http)|(socks5)://127.0.0.1:<p`

↪ `ort>"`

`export`

↪ `all_proxy="(http)|(socks5)://127.0.0.1:<port>"`

`curl -i www.google.com # test with google`

VS Code

Introduction to VS Code

- Visual Studio Code is a “free”, “open-source” code editor developed by Microsoft
- Supports debugging, syntax highlighting, intelligent code completion, snippets, and embedded Git
- Highly extensible through a vast marketplace of plugins and extensions
- Available for Windows, macOS, and Linux
- Built with Electron framework and written in TypeScript

Key features

- **Extensions:** Thousands of extensions to add new languages, themes, and tools
- **LSP:** Support many lsp for syntax highlighting and intellisense
- **Customizable:** Highly customizable interface and keybindings
- **Integrated terminal:** Built-in terminal for running commands
- **Git integration:** Built-in Git support for version control
- **Copilot integration:** Smart code completion that understands your code context

Extension marketplace

- Access thousands of extensions through the Extensions view (Ctrl+Shift+X)
- Categories include:
 - Programming languages (Python, JavaScript, Java, C++, etc.)
 - Linters and formatters
 - Themes and icon packs
 - Productivity tools
 - Fun extensions

Productivity shortcuts

- **Quick Open:** Ctrl+P (Cmd+P on Mac) to quickly open files
- **Command Palette:** Ctrl+Shift+P (Cmd+Shift+P on Mac) for all commands
- **Multi-cursor:** Alt+Click (Option+Click on Mac) for multiple cursors
- **Column selection:** Shift+Arrow keys (Shift+Arrow on Mac)
- **Integrated terminal:** Ctrl+' (Ctrl+õn some keyboards) to open terminal
- **Split editor:** Ctrl+ (Cmd+ on Mac) to split your view

You can search in command palette with key “Preferences: Open Keyboard Shortcuts” to customize your keybindings

Recommended extensions

- **Theme:** Personally I use github theme
- **Remote-ssh:** Useful for windows users to connect their wsl
- **Remote - SSH: Editing Configuration Files:** Edit SSH configuration files conveniently
- **Gitlens:** Beautify git visualizer
- **Vim:** Vim keybindings and cmd mode for vscode

- Install a theme for your vscode
- Install Markdown all in one, markdownlint, Markdown Preview Github Styling and Markdown PDF

Tips for beginners

- Start with default settings and gradually customize as you learn
- Install extensions only when you need them to avoid clutter
- Learn keyboard shortcuts to improve your coding speed
- Use the Explorer view to navigate your project files
- Use the integrated Git features for version control

Explore more features by
attending vscode workshop!

Markdown

Check markdown dir

Recommendation

Recommended websites

- Sub websites from stack exchange like Stackoverflow and Math Stackexchange
- Github
- Arch wiki

The Correct Way to Use Search Engines

- Use reliable search engines for technical queries, e.g., **Google**, **DuckDuckGo**, **Bing**.
- Avoid region-limited engines for technical searches (e.g., **Baidu**) due to irrelevant advertisements or outdated documentation.
- Use advanced search operators for more precise results:
 - `site:stackoverflow.com <query>` → search only on StackOverflow
 - `filetype:pdf <query>` → search for PDF files
 - `"exact phrase"` → search exact matches
- Use programming-specific search tools:
 - GitHub search: find code examples and projects
 - StackOverflow: find solutions to common errors
 - Official documentation: most reliable, do always verify with source

The Correct Way to Use AI

- **Define the problem clearly:**
 - Be specific about your goal, the role of AI in this process and expected output.
 - Break complex tasks into smaller, manageable sub-tasks.
- **Validate AI output:**
 - Cross-check AI suggestions with official documentation, scientific sources, or authoritative references.
 - Treat AI output as a recommendation, not absolute truth.
- **Leverage AI effectively:**
 - Avoid blindly copying output; understand and adapt it, else, you learn nothing from this chat but waste your time.
 - Make sure you understand what AI is doing. AI may create dangerous cmd and ruin up your env.
 - Don't abuse AI in hard homework. Most homework is deliberately designed to help you understand the course content.
- **Maintain privacy and security:**
 - Do not share sensitive data unless you trust the platform.
 - Prefer anonymized or synthetic examples when possible.

The Correct Way to Use AI

- Google Gemini can read pdf and images without limitation and you can use Pro for free for 12 months after college verification.
- Chatgpt deepresearch is especially helpful when you want to learn a field you don't know. It is a very good tools when you are doing survey in research.
- Qwen-code, gemini-cli is free agent coding tools you can use.

- Maintain your passion
- Explore more possibilities
- Don't push yourself too hard

Questions?