Missing Class Before College

Package Manger, Shell, Vscode and Markdown

Zhaojiacheng Zhou

September 6, 2025

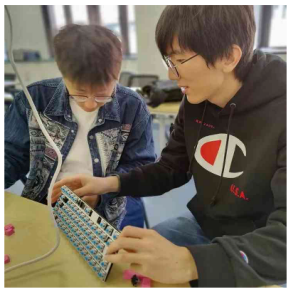
TechJI

Outline

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

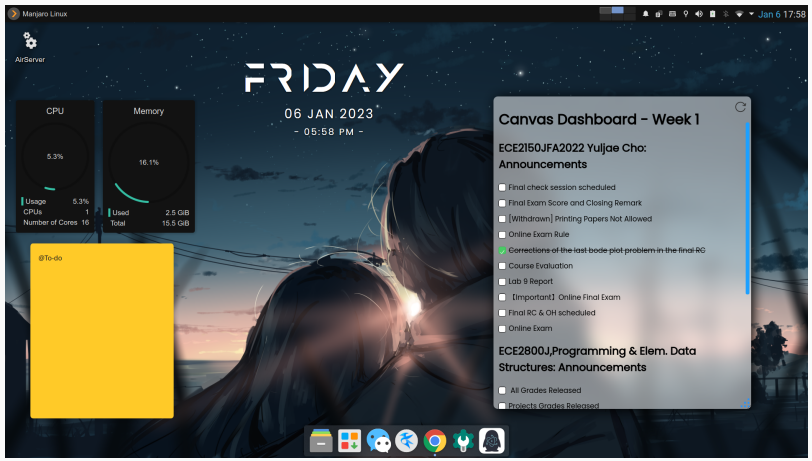
"Handle everything with speed and efficiency"

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.
- The core function of a package manager is to handle software dependencies and version conflicts to ensure that programs work correctly. It typically uses a local database to keep track of installed packages and their relationships. By working with software repositories or app stores, a package manager automates the entire software lifecycle, making it unnecessary to manually manage installations and updates.

Problem solved

- **Dependency Hell**: This is a common problem where 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**: They eliminate the need for users to manually download, compile, and install software, which can be a complex and time-consuming process. This is particularly useful for large operating systems with hundreds or thousands of distinct packages.
- **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**: These are the fundamental units of a package management system. 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, eliminating the need for users to search for and download software from various websites.
- **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, allowing the system to efficiently track and manage software and prevent conflicts.

Examples

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

Mirror

- 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 vscodium 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.t
 ↪  una.tsinghua.edu.cn/git/homebrew/brew.git"
export HOMEBREW_CORE_GIT_REMOTE="https://mirrors.t
 ↪  una.tsinghua.edu.cn/git/homebrew/homebrew-core
 ↪  .git"
export HOMEBREW_INSTALL_FROM_API=1
export HOMEBREW_API_DOMAIN="https://mirrors.tuna.t
 ↪  singhua.edu.cn/homebrew-bottles/api"
export HOMEBREW_BOTTLE_DOMAIN="https://mirrors.tun
 ↪  a.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 scilicon 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.t
 ↪  una.tsinghua.edu.cn/git/homebrew/homebrew-core
 ↪  .git"
for tap in core cask command-not-found; do
brew tap --custom-remote "homebrew/${tap}"
 ↪  "https://mirrors.tuna.tsinghua.edu.cn/git/home
 ↪  brew/homebrew-${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.
- Think of 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
- Improved file globbing and array handling
- Spelling correction and approximate completion
- Built-in support for Git and other version control systems
- macOS has made zsh the default shell since Catalina (10.15)

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

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

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
```

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

- **IntelliSense**: Smart code completion that understands your code context
- **Debugging**: Built-in debugging support for multiple languages
- **Git integration**: Built-in Git support for version control
- **Extensions**: Thousands of extensions to add new languages, themes, and tools
- **Customizable**: Highly customizable interface and keybindings
- **Integrated terminal**: Built-in terminal for running commands

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
- Popular extensions:
  - Prettier (code formatter)
  - ESLint (JavaScript linter)
  - GitLens (enhanced Git capabilities)
  - Markdown All in One
  - Bracket Pair Colorizer

Fun and useful extensions

- **Rainbow Brackets**: Colorize brackets for better readability
- **indent-rainbow**: Highlights indentation levels with colors
- **Code Spell Checker**: Catches common spelling errors in code
- **Polacode**: Take beautiful screenshots of your code
- **Thunder Client**: Lightweight REST API client (alternative to Postman)
- **Live Share**: Collaborate on code in real-time with others
- **Carbon Product Icons**: Beautiful icon theme for a fresh look
- **VSCode Peacock**: Subtly change the workspace color to help distinguish different projects

Customization tips

- **Themes**: Change the look of your editor with color themes
- **Fonts**: Use programming fonts like Fira Code with ligatures
- **Settings**: Customize everything through the Settings UI or settings.json
- **Keybindings**: Modify shortcuts to match your preferences
- **Snippets**: Create custom code snippets for frequently used patterns
- **Workspace settings**: Have different settings for different projects

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+Alt+Arrow keys (Shift+Option+Arrow on Mac)
- **Integrated terminal**: Ctrl+‘ (Ctrl+ on some keyboards) to open terminal
- **Zen Mode**: Ctrl+K Z to focus on your code without distractions
- **Split editor**: Ctrl+ (Cmd+ on Mac) to split your view

Tips for beginners

- Start with default settings and gradually customize as you learn
- Install extensions only when you need them to avoid clutter
- Use the integrated terminal instead of switching to external terminals
- Learn keyboard shortcuts to improve your coding speed
- Use the Explorer view to navigate your project files
- Take advantage of IntelliSense for code completion
- Use the integrated Git features for version control

Thank You

Questions?