命令列入門：開發者的超能力工具！ 

嘿！為什麼要學這玩意兒？

想像你在開發一個超酷的網站，需要把 100 個檔案全部重新命名。

使用圖形介面：一個一個點擊右鍵、重複上百次...

使用命令列：敲幾個字，咻的一聲！一秒鐘全部搞定！

命令列（CLI）讓重複瑣碎的工作變成一鍵完成！

命令列是什麼鬼東西？

簡單說，就是個文字版的電腦操作介面。

不用滑鼠點來點去，用打字聊天的方式跟電腦對話！

命令列對話示例

```
你敲：ls
電腦回：Desktop/ Documents/ Downloads/ ...

你敲：cd Desktop
電腦回：好，進去了！
```

Shell 扮演什麼角色？

Shell 就是你的「個人助理」

它負責：

1. **聽你指令**：把你打的字翻譯成電腦懂的語言
2. **執行任務**：實際去叫系統做事
3. **回報結果**：告訴你執行得怎麼樣

熱門 Shell 推薦

Shell 名稱	用在哪裡	厲害之處
PowerShell	Windows	最強大的 Windows Shell，功能超完整
Bash	Linux/macOS	全世界用得最多，寫腳本超強
Zsh	macOS/Linux	用起來超順手，主題超多
Fish	各平台	自動補全超聰明，新手友好

給新手的 Shell 建議

- **Windows 朋友**：從 PowerShell 開始
- **macOS 朋友**：用預設的 Zsh
- **Linux 朋友**：學 Bash，業界標準必備

晶晶體小字典

- **CLI**：命令列介面，用文字跟電腦對話
- **Shell**：殼層，負責翻譯你的指令給電腦聽
- **Terminal**：終端機，顯示你跟電腦對話的視窗
- **Prompt**：提示符號，等你輸入指令

開啟你的終端機大門 

Windows 環境大改造

Windows Terminal 安裝

Step 1：升級你的終端機體驗

Windows Terminal 就是終端機界的 iPhone！

1. 打開 **Microsoft Store**
2. 搜尋 **"Windows Terminal"**
3. 點擊 **"取得"** 下載安裝

Windows 啟動 Terminal

Step 2：啟動你的命令列宇宙

- 快捷鍵：按 `Win+X` → 點選 "Windows Terminal"
- 或者在開始功能表搜尋 "Terminal"

Step 3：確認你是 PowerShell 駕駛員

開啟後，確保選擇的是 "PowerShell"

PowerShell 的超能力

::: tip PowerShell 的超能力
PowerShell 不只是個 Shell！

它能操作 .NET 物件，就像給你的命令列裝了火箭推進器！🚀
:::

macOS 環境超簡單

macOS 早就準備好你的命令列戰場了！

開啟終端機三步驟：

1. 按 `Command+Space` 叫出 Spotlight
2. 輸入 "Terminal"
3. 按 Enter 進入命令列世界

macOS 檢查 Shell

檢查你的 Shell 身份：

開啟後，看到 `username@computer %` 嗎？

這就是 Zsh 的標記，你已經是時尚的 Zsh 用戶了！

macOS 切換到 Zsh

::: warning 如果看到的是 $ 符號
那是老派的 Bash。輸入這行指令切換到 Zsh：

```
chsh -s /bin/zsh
```

然後重新開啟 Terminal，就會看到 `%` 符號了！
:::

終端機介面大解密

打開終端機，你會看到這樣的畫面：

```
username@computer %
```

終端機介面解析

- **username@computer**：這是你的「身份證明」
  - 使用者名稱@電腦名稱
- **%**：Zsh 的「等你說話」符號
  - PowerShell 是 `PS>`
- ▉：游標在這裡閃啊閃，等你輸入指令

提示符號的密碼

- `%` 或 `$`：Shell 在說「我在聽，說吧！」
- `>`：指令還沒完，需要你繼續輸入
- 沒有符號：程式正在努力工作

基本導航指令：檔案系統的導航員 

你的導航工具箱

指令	功能	Windows 寫法	macOS 寫法
`pwd`	「我在哪裡？」	`Get-Location`	`pwd`
`ls`	「這裡有什麼？」	`Get-ChildItem`	`ls`
`ls -a`	「包含隱藏的？」	`Get-ChildItem -Hidden`	`ls -a`
`ls -l`	「詳細資料！」	`Get-ChildItem -Detailed`	`ls -l`
`cd`	「我要去那裡」	`Set-Location`	`cd`

「我在哪裡？」- 顯示目前位置

為什麼需要這個？

因為檔案系統就像個大迷宮，你需要知道自己站在哪裡！

pwd 指令範例

Windows PowerShell 玩家：

```
Get-Location   # 完整版
pwd            # 簡寫版
```

macOS Zsh 玩家：

```
pwd
```

pwd 實際輸出

實際看到的樣子：

```
Windows: C:\Users\yourname\Desktop
macOS: /Users/yourname/Desktop
```

「這裡有什麼？」- 列出檔案和資料夾

為什麼需要這個？

就像進到新房間，你想知道裡面有什麼家具！

ls 基本掃描

Windows PowerShell 玩家：

```
Get-ChildItem   # 完整版
ls              # 簡寫版
```

macOS Zsh 玩家：

```
ls
```

ls 進階掃描模式 - Windows

```
# 看見隱形檔案（那些 . 開頭的傢伙）
Get-ChildItem -Hidden
# 或
Get-ChildItem -Force

# 詳細掃描（權限、大小、修改時間）
Get-ChildItem -Detailed

# 完整掃描（隱形 + 詳細）
Get-ChildItem -Hidden -Detailed
```

ls 進階掃描模式 - macOS

```
# 看見隱形檔案
ls -a

# 詳細掃描（權限、大小、修改時間）
ls -l

# 完整掃描（隱形 + 詳細）
ls -al
```

實際動手玩玩看！🎮

練習目標：

熟悉導航指令，探索你的檔案系統宇宙

第一關：基本探索

1. **開啟終端機**（照前面教的開啟）
2. **確認你在哪裡：**

```
pwd   # Windows 玩家用 Get-Location
```

繼續探索

3. 看看家裡有什麼：

```
ls ~  # Windows 玩家用 Get-ChildItem ~
```

4. 發現隱藏的寶藏：

```
ls -a ~  # Windows 玩家用 Get-ChildItem ~ -Hidden
```

練習小密技

- 注意 Windows 和 macOS 的路徑寫法差異
- 發現那些 `.` 開頭的神秘檔案
- 比較 `ls -l` 的詳細資訊有多豐富

隱藏檔案的隱藏任務 🕵️

什麼是隱藏檔案？

檔案名稱以點 (`.`) 開頭的傢伙們，就像間諜一樣躲在檔案系統的陰影裡。

一般檔案瀏覽器不會顯示它們！

隱藏檔案的用途

它們其實是：

- **系統設定檔**：應用程式的「個人檔案」
- **版本控制檔案**：像 `.git` 這種程式碼時間機器的零件
- **使用者偏好設定**：你的個人化設定

常見的隱藏英雄們

檔案/資料夾	超能力	活躍地區
`.git/`	版本控制超能力	全宇宙
`.bashrc`	Bash Shell 的設定書	Linux/macOS
`.zshrc`	Zsh Shell 的設定書	macOS/Linux
`.DS_Store`	macOS 資料夾的記憶	macOS
`.ssh/`	SSH 金鑰的藏寶庫	全宇宙

隱藏檔案的警告標誌

::: warning 隱藏檔案的警告標誌
隱藏檔案通常是系統的「心臟」部位。

亂刪或亂改可能會讓應用程式生病或設定跑掉！
:::

資料夾間移動

目的：

在檔案系統中切換位置，就像在檔案總管中雙擊進入資料夾

cd 基本移動指令 - Windows

Windows PowerShell：

```
# 進入指定資料夾
Set-Location Desktop            # 進入 Desktop 資料夾
Set-Location Documents          # 進入 Documents 資料夾
Set-Location "My Projects"      # 處理有空格的資料夾名稱

# 回到上一層
Set-Location ..

# 回到家目錄
Set-Location ~
```

cd 基本移動指令 - macOS

macOS zsh：

```
# 進入指定資料夾
cd Desktop              # 進入 Desktop 資料夾
cd Documents            # 進入 Documents 資料夾
cd "My Projects"        # 處理有空格的資料夾名稱

# 回到上一層
cd ..

# 回到家目錄
cd ~
```

路徑表示法

絕對路徑 vs 相對路徑：

類型	說明	範例
絕對路徑	從根目錄開始的完整路徑	`/Users/username/Desktop` `C:\Users\username\Desktop`
相對路徑	相對於目前位置的路徑	`Desktop/file.txt` `../Documents/file.txt`

路徑特殊符號

符號	意義	Windows 範例	macOS 範例
`~`	家目錄	`~\Documents`	`~/Documents`
`.`	目前資料夾	`.\file.txt`	`./file.txt`
`..`	上一層資料夾	`..\file.txt`	`../file.txt`
`/`	路徑分隔符	`C:\folder\file`	`/folder/file`

進階移動技巧

多層次移動：

```
# 從家目錄直接進入深層資料夾
cd ~/Projects/web-dev/my-app/src

# 從目前位置向上移動多層
cd ../../../

# 組合使用
cd ../other-project
```

歷史位置切換

歷史位置切換：

```
# PowerShell
Set-Location -   # 回到上一個位置

# Zsh/Bash
cd -   # 回到上一個位置
```

實際操作練習

練習目標：

熟悉資料夾導航，理解路徑概念

練習步驟 1-2

1. 從家目錄開始：

```
pwd   # 確認目前位置
ls    # 查看家目錄內容
```

2. 進入 Desktop 資料夾：

```
cd Desktop
pwd   # 確認新位置
ls    # 查看 Desktop 內容
```

練習步驟 3-4

3. 測試相對路徑：

```
cd ..  # 回到上一層（家目錄）
cd Desktop  # 再次進入 Desktop
```

4. 使用絕對路徑：

```
cd ~/Desktop  # 直接從家目錄進入 Desktop
```

5. 快速回到家目錄：

```
cd ~
pwd
```

練習重點

- 觀察路徑如何變化
- 比較絕對路徑和相對路徑的差異
- 練習使用 `..` 返回上一層
- 熟悉 `~` 符號的便利性

路徑表示法差異

Windows 使用反斜槓 `\` 作為分隔符

Unix-like 系統 (macOS/Linux) 使用正斜槓 `/`

PowerShell 支援兩種表示法，但建議使用正斜槓以保持一致性

SSH 任意門：安全連線的魔法通道 🔐

SSH 是什麼？

SSH（Secure Shell）就像哆啦A夢的任意門！

讓你能夠安全地從一個地方「傳送」到另一個遠端的電腦世界！

SSH 的魔法

想像一下：

- 你在家裡打開任意門，一腳踏進去...
- 瞬間就出現在學校的電腦實驗室裡！
- 而且途中還穿過了重重加密保護，完全不會被壞人偷窺！

SSH 超能力 1：安全認證

🔑 安全認證

就像任意門需要正確的鑰匙才能開門

```
# 生成你的專屬魔法鑰匙
ssh-keygen -t rsa -b 4096

# 複製鑰匙到遠端門戶
ssh-copy-id user@remote-server.com
```

SSH 超能力 2：加密通道

🌐 加密通道

所有資料傳輸都像通過魔法結界保護

```
# 開啟安全連線通道
ssh user@remote-server.com

# 還可以帶著檔案一起傳送
scp myfile.txt user@remote-server.com:~/destination/
```

SSH 超能力 3：遠端執行

⚡ 遠端執行

在遠端世界執行魔法指令

```
# 遠端執行指令（不用真正傳送過去！）
ssh user@remote-server.com 'ls -la /var/www'

# 甚至開啟遠端的圖形介面程式
ssh -X user@remote-server.com firefox
```

SSH 的實際用途

- **伺服器管理**：遠端管理你的網站伺服器
- **團隊協作**：安全地連接到同事的開發環境
- **部署程式**：將程式碼安全地傳送到生產環境

SSH 安全提醒

- 永遠不要用密碼認證，改用金鑰對！
- 定期更換你的 SSH 鑰匙
- 只連接到信任的伺服器

檔案操作指令

檔案操作指令概覽

操作	Windows	macOS	目的
複製	`Copy-Item`	`cp`	建立檔案備份
移動	`Move-Item`	`mv`	搬移或重新命名
刪除	`Remove-Item`	`rm`	移除檔案/資料夾
查看	`Get-Content`	`cat`	顯示檔案內容

複製檔案 - Windows

Windows PowerShell：

```
# 複製檔案並改名
Copy-Item file.txt file_backup.txt

# 複製到目前資料夾
Copy-Item ~/Documents/report.pdf .

# 複製多個檔案
Copy-Item *.txt backup/
```

複製檔案 - macOS

macOS zsh：

```
# 複製檔案並改名
cp file.txt file_backup.txt

# 複製到目前資料夾
cp ~/Documents/report.pdf .

# 複製多個檔案
cp *.txt backup/
```

複製資料夾 - Windows

```
# 複製整個資料夾
Copy-Item -Recurse my-folder my-folder-backup

# 複製資料夾內容（不含資料夾本身）
Copy-Item -Recurse source-folder/* destination-folder/
```

複製資料夾 - macOS

```
# 複製整個資料夾
cp -R my-folder my-folder-backup

# 複製資料夾內容（不含資料夾本身）
cp -R source-folder/* destination-folder/
```

移動和重新命名

目的：

搬移檔案位置或更改檔案名稱

移動檔案 - Windows

Windows PowerShell：

```
# 移動檔案到其他資料夾
Move-Item file.txt ~/Documents/

# 重新命名檔案
Move-Item old_name.txt new_name.txt

# 移動並重新命名
Move-Item file.txt ~/Documents/new_name.txt
```

移動檔案 - macOS

macOS zsh：

```
# 移動檔案到其他資料夾
mv file.txt ~/Documents/

# 重新命名檔案
mv old_name.txt new_name.txt

# 移動並重新命名
mv file.txt ~/Documents/new_name.txt
```

移動 vs 複製

- **複製 (cp/Copy-Item)**：保留原檔案，建立新副本
- **移動 (mv/Move-Item)**：將檔案搬到新位置，原位置不再有該檔案

刪除檔案和資料夾

目的：

永久移除檔案或資料夾

⚠️ 重要警告

命令列的刪除操作無法復原！

請務必確認要刪除的檔案。

安全刪除 - Windows

Windows PowerShell：

```
# 刪除前確認
Remove-Item -Confirm unwanted_file.txt

# 刪除資料夾前確認
Remove-Item -Confirm -Recurse old_folder
```

安全刪除 - macOS

macOS zsh：

```
# 刪除前確認
rm -i unwanted_file.txt

# 刪除資料夾前確認
rm -ri old_folder
```

直接刪除 - Windows

Windows PowerShell：

```
# 刪除檔案
Remove-Item unwanted_file.txt

# 刪除資料夾
Remove-Item -Recurse old_folder
```

直接刪除 - macOS

macOS zsh：

```
# 刪除檔案
rm unwanted_file.txt

# 刪除資料夾
rm -r old_folder
```

查看檔案內容

目的：

顯示檔案的文字內容

查看檔案 - Windows

Windows PowerShell：

```
# 查看檔案內容
Get-Content myfile.txt

# 查看前幾行
Get-Content myfile.txt -Head 10

# 查看後幾行
Get-Content myfile.txt -Tail 10

# 合併顯示多個檔案
Get-Content file1.txt, file2.txt
```

查看檔案 - macOS

macOS zsh：

```
# 查看檔案內容
cat myfile.txt

# 查看前幾行
head -10 myfile.txt

# 查看後幾行
tail -10 myfile.txt

# 合併顯示多個檔案
cat file1.txt file2.txt
```

開啟編輯器

目的：

使用 VSCode 編輯檔案

開啟 VSCode - Windows

Windows PowerShell：

```
# 開啟 VSCode
code

# 在目前資料夾開啟專案
code .

# 編輯特定檔案
code myfile.txt
```

開啟 VSCode - macOS

macOS zsh：

```
# 開啟 VSCode
code

# 在目前資料夾開啟專案
code .

# 編輯特定檔案
code myfile.txt
```

實際操作練習

練習目標：

熟悉檔案操作，練習安全使用指令

練習步驟 1-2

1. 建立測試檔案：

```
echo "Hello World" > test.txt
# PowerShell: New-Item test.txt -Value "Hello World"
ls  # 確認檔案建立
```

2. 複製檔案：

```
cp test.txt backup.txt
# PowerShell: Copy-Item test.txt backup.txt
ls  # 確認複製成功
```

練習步驟 3-4

3. 查看檔案內容：

```
cat test.txt
# PowerShell: Get-Content test.txt
```

4. 移動檔案：

```
mkdir test-folder
# PowerShell: New-Item test-folder -ItemType Directory
mv backup.txt test-folder/
# PowerShell: Move-Item backup.txt test-folder/
ls test-folder/  # 確認移動成功
```

練習步驟 5 與重點

5. **清理測試檔案：**

```
rm -i test.txt
# PowerShell: Remove-Item -Confirm test.txt
rm -ri test-folder
# PowerShell: Remove-Item -Confirm -Recurse test-folder
```

練習重點：

- 養成使用 `-i` 或 `-Confirm` 的習慣
- 刪除前先確認檔案位置和名稱
- 練習不同的檔案操作組合

CLI 心智模型（初學者版）

CLI 核心概念

- **Shell**：像是 `bash` / `zsh` / `fish` ，負責解讀你輸入的指令
- **Terminal**：顯示輸入/輸出的視窗
- **Command**： `程式` + `參數` + `選項`
  - 例如： `ls -al /usr`

路徑與快捷鍵

- **目前路徑**：`pwd` 顯示
  - 相對路徑：`./`, `..`
  - 絕對路徑：`/`, `~/`
- **Tab 補全**：輸入部分名稱後按 `Tab` 補齊

安全操作與常見選項

- ⚠️ 刪除：優先用 `rm -i`（互動確認），避免誤刪
- `rm -r` 會遞迴刪資料夾
- 常見選項：
  - `ls -a` 顯示隱藏檔
  - `ls -l` 詳細列表
  - `cp -R` 複製資料夾
- 路徑小幫手：`~` 家目錄、`.` 目前、`..` 上一層

更多常用指令（精選）

檔案與資料夾指令

```
# 建立與刪除
mkdir demo && cd demo          # 建立資料夾並進入
touch notes.txt                # 建立空檔案
rmdir empty-dir                # 刪除空資料夾
```

檢視內容指令

```
# 檢視檔案
less notes.txt                  # 可翻頁檢視，q 離開
head -n 5 notes.txt             # 前 5 行
tail -n 5 notes.txt             # 後 5 行
tail -f                         # 即時追蹤
```

搜尋指令

```
# 搜尋文字與檔案
grep -R "TODO" .              # 遞迴搜尋文字
find . -name "*.md"           # 依檔名尋找
```

macOS 小技巧

```
# macOS 專屬
open .                          # Finder 開啟目前資料夾
pbcopy < notes.txt              # 複製到剪貼簿
pbpaste                         # 從剪貼簿貼上
```

資訊與協助

```
# 查詢指令
which node                    # 程式所在路徑
whoami; date                  # 使用者與現在時間
man ls                        # 手冊
cmd --help                    # 指令說明
```

管線與重導向詳解

管線 (|)：讓指令接力執行

管線範例

```
# 統計檔案行數
cat notes.txt | wc -l

# 找特定文字並計算數量
grep "TODO" notes.txt | wc -l

# 顯示前 3 行然後排序
head -3 notes.txt | sort
```

重導向詳解

重導向 (>)：把輸出存到檔案

重導向範例

```
# 把指令結果存到檔案
ls > file_list.txt

# 覆蓋檔案內容
echo "New content" > myfile.txt

# 追加到檔案後面 (>>)
echo "More content" >> myfile.txt
```

為什麼這麼有用？

管線就像工廠的生產線，一個環節接一個環節！

重導向就像把結果「導」到檔案裡保存起來，
這樣就可以重複使用結果，不用每次都重新執行！

迷你練習（可直接複製貼上）

```
mkdir play && cd play
echo "line1" > a.txt && echo "line2" >> a.txt
cp a.txt b.txt && mv b.txt notes.txt
grep -n "line" a.txt
tail -n 1 a.txt
cd .. && rm -r play # 若不確定，改用 rm -ri play 逐一確認
```

CLI Cheat Sheet（10 行）

```
pwd; ls -al             # 目前位置與清單
cd ..; cd -             # 上一層；返回前一路徑
mkdir x; touch f        # 建資料夾/檔案
cp -R src dst           # 複製資料夾
mv a b                  # 重新命名/搬移
rm -ri folder           # 安全刪除資料夾
cat/less/head/tail      # 檢視檔案
grep -R "text" .        # 搜尋文字
open .                  # Finder 開啟
man cmd / cmd --help # 查看說明
```