

터미널과 셸

1. 터미널과 셸의 개념

✓ 터미널(Terminal)

맥에서 터미널(Terminal) 앱을 실행하면 명령어를 입력할 수 있는 창이 열려. 여기서 우리는 "셸(Shell)"이라는 프로그램을 사용해 명령어를 실행해.

✓ 셸(Shell)

셸은 사용자가 입력한 명령어를 운영체제에 전달하고, 그 결과를 보여주는 프로그램이야. 맥에서는 기본적으로 `zsh` (Z shell)을 사용하지만, `bash` 도 많이 쓰여.

📌 명령어 입력 방식

명령어 [옵션] [인자]

- 명령어: 실행할 프로그램이나 기능
- 옵션(- 또는 --로 시작함): 기능을 추가로 조정하는 설정
- 인자: 명령어가 동작할 대상 (예: 파일, 폴더 등)

2. 기본적인 파일/디렉토리 관리

📌 현재 작업 디렉토리 확인 (pwd)

```
pwd
```

- 현재 작업 중인 디렉토리(폴더)의 경로를 출력함.

📌 디렉토리 내용 확인 (ls)

```
ls
```

- 현재 디렉토리의 파일과 폴더 목록을 출력함.
- 옵션

- `-l` : 상세 정보 표시
- `-a` : 숨김 파일 포함
- `-h` : 파일 크기를 사람이 읽기 쉽게 표시

```
ls -lah
```

👉 파일 목록을 자세하게(human-readable) 숨김 파일 포함해서 보여줌.

📌 디렉토리 이동 (cd)

`cd` 경로

- 특정 폴더로 이동할 때 사용
- 예제

```
cd ~/Documents # "문서" 폴더로 이동
cd ..          # 한 단계 위 폴더로 이동
cd /           # 최상위(root) 폴더로 이동
cd ~           # 홈 디렉토리로 이동
```

📌 디렉토리 만들기 (mkdir)

`mkdir` 새폴더이름

- 새로운 폴더를 생성함.
- 예제

```
mkdir my_project # 현재 디렉토리 밑에 my_project 폴더 생성
mkdir ~/Downloads/my_project # ~/ 는 홈 디렉토리 # /Users/mhb8436
```

👉 `my_project` 라는 폴더가 생성됨.

📌 파일 생성 (touch)

`touch` 파일이름

- 새로운 빈 파일을 생성함.
- 예제

```
cd ~/Downloads/my_project
```

```
touch newfile.txt
```

👉 newfile.txt 파일이 생성됨.

📌 파일/디렉토리 삭제 (rm, rmdir)

rm 파일이름

- 파일을 삭제함.
- 예제

```
cd ~/Downloads/my_project
touch oldfile.txt
ls -la
rm oldfile.txt
```

👉 oldfile.txt 파일이 삭제됨.

rmdir 폴더이름

- 빈 폴더를 삭제함.

rm -r 폴더이름

- 폴더 및 그 내부 파일을 강제로 삭제함.
- 예제

```
rm -r my_project
```

👉 my_project 폴더와 내부 파일들이 삭제됨.

3. 파일 읽기 및 수정

📌 파일 내용 보기 (cat, less)

cat 파일이름

- 파일 내용을 출력함. (작은 파일에 적합)

`less` 파일이름

- 큰 파일을 한 페이지씩 읽을 때 사용 (q 키로 종료)

파일 내용 검색 (grep)

`grep` 검색어 파일이름

- 파일 내에서 특정 단어를 검색할 때 사용
- 예제

```
grep "Hello" example.txt
```

👉 example.txt 에서 "Hello"가 포함된 줄을 출력

4. 프로세스와 시스템 정보 확인

현재 실행 중인 프로세스 확인 (ps, top)

`ps` aux

- 실행 중인 모든 프로세스를 리스팅

`top`

- 실시간으로 CPU, 메모리 사용량을 보여줌 (종료하려면 q 키)

프로세스 강제 종료 (kill)

`kill` 프로세스ID

- 특정 프로세스를 강제 종료
- 예제

```
kill 12345
```

👉 프로세스 ID가 12345 인 프로세스를 종료함.

5. 기타 유용한 명령어

명령어 실행 기록 보기 (history)

```
history
```

- 사용자가 실행한 명령어 목록을 출력

명령어 도움말 보기 (man)

```
man 명령어이름
```

- 특정 명령어의 매뉴얼을 확인할 수 있음
- 예제

```
man ls
```

👉 `ls` 명령어의 설명을 볼 수 있음 (`q` 키로 종료)

명령어 자동완성

- Tab 키를 누르면 파일/디렉토리 이름 자동완성 가능
- ↑ (위쪽 화살표): 이전에 실행한 명령어 불러오기

실습 과제

아래 실습을 따라 하면서 익숙해져 보자!

1 터미널 열기

- Command + Space 누르고 Terminal 검색 후 실행

2 현재 디렉토리 확인

```
pwd
```

3 연습용 폴더 생성

```
mkdir terminal_practice  
cd terminal_practice
```

4 파일 생성하고 내용 확인

```
touch test.txt
ls -l
```

5 파일 삭제하고 폴더 삭제

```
rm test.txt
cd ..
rmdir terminal_practice
```



Nano 에디터 사용법 (맥 터미널)

Nano는 간단하면서도 강력한 터미널 기반 텍스트 편집기야. **Vim**이나 **Emacs**보다 사용하기 쉬워서 초보자들에게 추천해.

1. Nano 실행하기

터미널에서 다음 명령어를 입력하면 nano를 실행할 수 있어.

```
nano 파일이름
```

예제:

```
nano myfile.txt
```

👉 myfile.txt 파일이 열리며, 없으면 새로 생성됨.

2. Nano 인터페이스 이해하기

Nano를 실행하면 화면 하단에 단축키 목록이 나와.

- **^** 기호는 **Ctrl** 키를 의미해.
 - 예: **^X** → **Ctrl + X**
- **M-** 기호는 **Option(⌘)** 키를 의미해.
 - 예: **M-U** → **Option + U**

3. 기본적인 조작법

텍스트 입력


- 그냥 타이핑하면 파일에 바로 입력됨.

파일 탐색 및 이동

단축키	기능
Ctrl + A	줄의 맨 앞으로 이동
Ctrl + E	줄의 맨 끝으로 이동
Ctrl + Y	이전 페이지로 이동 (스크롤 업)
Ctrl + V	다음 페이지로 이동 (스크롤 다운)
Ctrl + _ (Ctrl + Shift + -)	특정 줄 번호로 이동

4. 텍스트 수정하기

단축키	기능
Ctrl + K	현재 줄 삭제
Ctrl + U	마지막으로 삭제한 줄 붙여넣기
Ctrl + W	텍스트 검색
Ctrl + \	특정 단어 찾아서 변경 (찾기 및 바꾸기)

 예제: "hello"를 "hi"로 변경

1. Ctrl + \ 누르기
2. "hello" 입력 후 Enter
3. "hi" 입력 후 Enter
4. A (모두 바꾸기) 또는 Y (한 개씩 바꾸기) 선택

5. 파일 저장 및 종료

단축키	기능
Ctrl + O	저장 (Enter 키를 한 번 더 눌러야 저장됨)

단축키	기능
Ctrl + X	Nano 종료
Ctrl + X 후 Y	변경 내용 저장 후 종료
Ctrl + X 후 N	변경 내용 저장 없이 종료

6. 실습 과제

1 파일 열기

```
nano practice.txt
```

2 텍스트 입력하기

"Hello, Nano!"를 입력

3 파일 저장 (Ctrl + O → Enter)

4 Nano 종료 (Ctrl + X)

5 파일 확인

```
cat practice.txt
```

이제 Nano를 쉽게 사용할 수 있을 거야! 🚀

추가 질문 있으면 편하게 물어봐~ 😊

vi 에디터

다음은 macOS에서 vi 편집기를 처음 배우는 초보 개발자를 위한 연습 가이드입니다.

터미널 환경에서 텍스트 편집을 익히는 데 필수적인 단계를 예제와 함께 설명드리겠습니다.

1. vi 편집기 기본 개념

- vi는 CLI(터미널) 환경에서 작동하는 강력한 텍스트 편집기입니다.
- 두 가지 모드를 전환하며 사용합니다:
 - 명령 모드(Command Mode): 파일 탐색, 복사/붙여넣기, 저장 등 명령 실행.
 - 입력 모드(Insert Mode): 텍스트 입력 및 수정.

2. 기본 명령어 정리 (Cheat Sheet)

모드 전환

명령어	설명
i	현재 커서 위치에서 입력 모드 시작
a	커서 다음 위치에서 입력 모드 시작
ESC	명령 모드로 돌아가기

파일 관리

명령어	설명
:w	파일 저장
:q	vi 종료 (변경 사항 없을 때)
:q!	강제 종료 (변경 사항 저장 안 함)
:wq	저장 후 종료

커서 이동

명령어	설명
h	왼쪽 이동
j	아래 이동
k	위 이동
l	오른쪽 이동
gg	파일 첫 줄로 이동
G	파일 마지막 줄로 이동
: [숫자]	특정 줄로 이동 (예: :10 → 10번 줄)

텍스트 편집

명령어	설명
x	커서 위치 문자 삭제
dd	현재 줄 삭제
yy	현재 줄 복사
p	복사한 내용 붙여넣기

명령어	설명
u	실행 취소 (Undo)

3. 연습 예제: 간단한 파일 생성 및 편집

예제 1: 첫 번째 파일 만들기

- 터미널을 열고 다음 명령어 실행:

```
vi hello.txt
```

- 입력 모드 진입: `i` 키를 눌러 텍스트 입력 시작.
- 다음 내용 입력:

```
Hello, World!  
This is my first vi file.  
I'm learning vi on macOS.
```

- 명령 모드로 전환: `ESC` 키 누르기.
- 저장 후 종료: `:wq` 입력 후 엔터.

예제 2: 기존 파일 수정하기

- 터미널에서 파일 다시 열기:

```
vi hello.txt
```

- 두 번째 줄 수정:
 - 커서를 `j` 키로 두 번째 줄로 이동.
 - `i` 키로 입력 모드 진입 후 문장 끝에 `(edited)` 추가.
- 세 번째 줄 삭제:
 - 명령 모드에서 `dd` 입력하여 줄 삭제.
- 저장 없이 종료: `:q!` 입력 (변경 사항 무시).

4. 실전 연습 문제

문제 1: 코드 주석 추가

1. `vi code.py` 로 새 파일 생성.
2. 다음 코드 입력:

```
print("Hello")
print("Welcome to vi")
```

3. 각 줄 앞에 `#` 를 추가하여 주석 처리 (`i` 로 입력 모드 진입).
4. `:wq` 로 저장 후 종료.

문제 2: 텍스트 찾기 및 교체

1. `vi story.txt` 로 파일 열기.
2. 다음 텍스트 입력:

```
The cat sat on the mat.
The dog barked at the cat.
```

3. 모든 **"cat"**을 **"bird"**로 교체:
 - 명령 모드에서 `:%s/cat/bird/g` 입력.
4. 결과 확인 후 저장.

5. 문제 해결 팁

- 입력 모드에서 벗어날 수 없을 때: ESC 를 여러 번 누르세요.
- 파일을 실수로 덮어썼을 때: `:q!` 로 강제 종료 후 다시 시도.
- 줄 번호 표시: 명령 모드에서 `:set number` 입력.
- 화면 갱신: `Ctrl + L` (터미널 깨짐 현상 방지).

6. 마무리

- **vi**는 처음에 익히기 어렵지만, 반복 연습하면 터미널 작업 효율성이 크게 향상됩니다.
- 매일 10분씩 기본 명령어를 연습해 보세요!

```
# 추가 연습을 위해 매일 할 일:
vi diary.txt # 일기 작성 후 :wq로 저장
```

HomeBrew & Postgresql 설치

1. Homebrew가 설치되어 있는지 확인

터미널을 열고 다음 명령어를 실행합니다.

```
brew --version  
# https://brew.sh/ 접속 후 명령어 카피
```

Homebrew가 설치되지 않았다면 먼저 설치하세요.

```
/bin/bash -c "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

2. PostgreSQL 설치

다음 명령어를 실행하여 PostgreSQL을 설치합니다.

```
brew install postgresql
```

3. PostgreSQL 서비스 시작

설치 후 PostgreSQL을 실행하려면 다음 명령어를 입력하세요.

```
brew services start postgresql
```

서비스를 중지하려면:

```
brew services stop postgresql
```

4. PostgreSQL 버전 확인

```
postgres --version
```

5. PostgreSQL 기본 설정

PostgreSQL이 실행되었는지 확인하고, psql 명령어를 사용하여 PostgreSQL 셸에 접속할 수 있습니다.

```
psql postgres
```

만약 psql: error: connection to server on socket ... failed 오류가 발생하면 PostgreSQL이 실행 중인지 확인하세요.

```
brew services list
```

실행되지 않았다면 다시 시작하세요.

```
brew services start postgresql
```

6. PostgreSQL 기본 설정

- 현재 로그인한 macOS 사용자와 동일한 이름의 PostgreSQL 사용자 계정이 자동으로 생성됩니다.
- 새 사용자 계정을 생성하려면 다음 명령을 실행하세요.

```
# 데이터베이스 및 사용자 생성
psql postgres
```

```
``sql
CREATE DATABASE mydb; # 데이터베이스 생성
CREATE USER myuser WITH PASSWORD 'mypassword'; # 유저 생성
GRANT ALL PRIVILEGES ON DATABASE mydb TO myuser; # 데이터베이스 권한 유저 매핑
핑
````
```

## \*\*7. DBeaver 설치\*\*

**DBeaver**는 다양한 데이터베이스를 관리할 수 있는 오픈소스 SQL 클라이언트 및 데이터베이스 관리 도구입니다.

- \* MySQL, PostgreSQL, SQLite, Oracle, SQL Server 등 다양한 DBMS 지원
- \* GUI 환경에서 SQL 실행, 데이터 조회, 테이블 관리, ERD 다이어그램 지원
- \* 플러그인 확장 가능

---

### \*\*Homebrew로 DBeaver 설치 방법 (macOS)\*\*

**Homebrew**를 사용하면 터미널에서 간단히 설치할 수 있습니다.

### \*\*1. 설치\*\*

```
```bash
brew update
brew install --cask dbeaver-community
```

- `--cask` 옵션을 붙이면 GUI 앱을 설치할 수 있습니다.
- `dbeaver-community` 는 오픈소스 무료 버전입니다. (유료 버전은 `dbeaver-enterprise`)

3. 설치 확인

```
brew list --cask | grep dbeaver
```

출력 예시:

```
dbeaver-community
```

4. DBeaver 실행

```
open -a DBeaver
```

또는

Launchpad에서 "DBeaver" 검색 후 실행

◆ DBeaver 사용 방법 (기본)

1 새 데이터베이스 연결

- Database → New Connection
- 원하는 데이터베이스 선택 (MySQL, PostgreSQL 등)
- Host, Port, Username, Password 입력
- Test Connection → Finish

2 SQL 실행

- 연결한 데이터베이스에서 SQL Editor 열기
- SQL 문 작성 후 Run (Ctrl+Enter or Cmd+Enter)

3 테이블 및 데이터 관리

- 좌측 Database Navigator 에서 테이블 클릭

- Data 탭에서 데이터 조회 및 수정 가능