# AI 필기 요약 보고서

생성일: 2026-06-23 07:23 · 파일명: 20260623_001.pdf

## 챕터 구성

1. Chapter 1: Linux 시스템 기본 개념
2. Chapter 2: 터미널 명령어 및 쉘 스크립팅
3. Chapter 3: 소프트웨어 및 파이썬 환경 관리
4. Chapter 4: 시스템 보안 및 서비스 제어
5. Chapter 5: 대화형 개발 환경 Jupyter Notebook

## 분석 대상 파일

- **곽효건_2일차.txt** — 개인, hyogun, text

# Chapter 1: Linux 시스템 기본 개념

## 1.1 리눅스 시스템의 시작과 종료: init 프로세스

리눅스 시스템이 부팅될 때 가장 먼저 실행되는 프로세스는 **`init`**(PID 1)입니다. 이 프로세스는 시스템의 다른 모든 프로세스를 시작하고 관리하는 역할을 합니다. 시스템 종료 및 재부팅과 같은 핵심적인 시스템 동작도 `init` 프로세스를 통해 제어됩니다.

- **`init 0`**: 시스템 종료 (Shutdown)
- **`init 6`**: 시스템 재부팅 (Reboot)

*실무/시험 포인트:* 최신 리눅스 배포판(예: CentOS 7+, Ubuntu 15+)에서는 `systemd`가 `init` 프로세스의 역할을 대체하고 있으며, `systemctl poweroff`나 `systemctl reboot` 명령어를 주로 사용합니다. 하지만 `init`의 개념은 리눅스 시스템의 근간을 이해하는 데 중요하며, 일부 임베디드 시스템이나 구형 리눅스에서는 여전히 사용될 수 있습니다.

## 1.2 필수 명령어 및 파일 시스템 탐색

리눅스 터미널 환경에서 작업을 수행하기 위한 기본적인 명령어들을 숙지하는 것은 매우 중요합니다.

- **`pwd`**: **p**rint **w**orking **d**irectory의 약자로, 현재 작업 중인 디렉터리의 절대 경로를 출력합니다. 파일 시스템 내에서 자신의 위치를 파악하는 데 필수적인 명령어입니다.
- **`ls`**: **l**i**s**t의 약자로, 현재 디렉터리 또는 지정된 디렉터리 내의 파일 및 디렉터리 정보를 표시합니다. 다양한 옵션을 통해 출력 형식을 제어할 수 있습니다.
  - **`-a`**: (all) 숨김 파일(파일 이름이 `.`으로 시작하는 파일)을 포함하여 모든 파일을 표시합니다. 주로 환경 설정 파일들이 숨김 파일 형태로 존재합니다.
  - **`-l`**: (long) 파일의 상세 정보를 출력합니다. 권한, 링크 수, 소유자, 그룹, 파일 크기, 최종 수정일자, 파일 이름 순으로 표시됩니다.
  - **`ls -al`** 또는 **`ll`**: `-a`와 `-l` 옵션을 함께 사용하여 숨김 파일을 포함한 상세 정보를 출력하는 단축 표현입니다.

  *출력 예시 분석:* `ls -l`의 출력 결과 맨 앞 문자는 파일의 종류를 나타냅니다. `-`이면 일반 파일, `d`면 디렉터리, `l`이면 심볼릭 링크(바로가기) 등입니다. 이어서 나오는 9개의 문자는 권한 정보를 나타냅니다. 터미널 애플리케이션에 따라 파일 종류별로 색상이 다르게 표시되어 시각적으로 구분하기 용이합니다.
- **`cat`**: **cat**enate의 약자로, 주로 텍스트 파일의 내용을 화면에 출력하는 데 사용됩니다. 여러 파일을 연결하여 출력하거나, `>` 리다이렉션 기호와 함께 사용하여 새 파일을 생성하는 용도로도 활용될 수 있습니다. (예: `cat file1.txt file2.txt > combined.txt`)
- **`sudo`**: **s**uper**u**ser **do**의 약자로, 일반 사용자가 관리자(root) 권한으로 특정 명령어를 실행할 수 있게 해줍니다. 시스템 설정 변경, 패키지 설치, 중요한 파일 수정 등 관리자 권한이 필요한 작업에 사용됩니다. 보안을 위해 `sudoers` 파일을 통해 어떤 사용자가 어떤 명령어를 `sudo`로 실행할 수 있는지 제어할 수 있습니다.

## 1.3 파일 및 디렉터리 권한 관리: chmod

리눅스 파일 시스템에서 파일 및 디렉터리에 대한 접근 권한은 매우 중요합니다. 이는 시스템 보안과 안정성에 직결되며, 불필요한 접근을 제한하여 데이터를 보호합니다.

- **권한 분류**:
  - **사용자별**:
    - **소유자 (User)**: 파일이나 디렉터리를 생성한 사용자.
    - **그룹 (Group)**: 파일이나 디렉터리가 속한 그룹. 해당 그룹의 모든 구성원에게 적용됩니다.
    - **다른 사용자 (Others)**: 소유자도 아니고 해당 그룹의 구성원도 아닌 모든 사용자.
  - **권한 종류별**:
    - **읽기 (r, read)**: 파일의 내용을 읽거나 디렉터리의 파일 목록을 볼 수 있는 권한.
    - **쓰기 (w, write/수정)**: 파일의 내용을 변경하거나 디렉터리 내에 파일을 생성/삭제할 수 있는 권한.
    - **실행 (x, eXcute)**: 파일을 프로그램으로 실행하거나 디렉터리로 이동(`cd`)할 수 있는 권한.
- **`chmod`**: **ch**ange **mod**e의 약자로, 파일 또는 디렉터리의 권한을 수정하는 명령어입니다.
  - **기호 모드 (Symbolic Mode)**: 사용자와 권한을 기호로 지정하여 변경합니다.
    - **예시: `chmod u+x [파일명]`**: 해당 파일의 소유자(u)에게 실행(x) 권한을 부여합니다.
    - `u`: User (소유자), `g`: Group (그룹), `o`: Others (다른 사용자), `a`: All (모두)
    - `+`: 권한 추가, `-`: 권한 제거, `=`: 권한 설정
  - **숫자 모드 (Octal Mode)**: 각 권한에 숫자를 할당하여 권한을 설정하는 방법으로, 더 직관적이고 한 번에 여러 권한을 설정하기에 편리합니다.

    - r (읽기): 4
    - w (쓰기): 2
    - x (실행): 1

    각 사용자별 권한 숫자를 합산하여 세 자리 숫자로 표현합니다. 예를 들어, `chmod 755 [파일명]`은 소유자에게 읽기/쓰기/실행 (4+2+1=7) 권한을, 그룹 및 다른 사용자에게는 읽기/실행 (4+1=5) 권한을 부여합니다.

*자주 하는 오해:* 디렉터리에 실행 권한(x)이 없으면 해당 디렉터리 안으로 들어갈 수 없습니다 (`cd` 명령어 사용 불가). 파일의 실행 권한은 해당 파일을 프로그램으로 실행할 수 있게 하며, 쉘 스크립트 파일을 실행하기 위해서는 반드시 실행 권한이 필요합니다.

### 1.4 CLI 텍스트 편집기: Vi/Vim

리눅스 서버 환경에서는 그래픽 사용자 인터페이스(GUI)가 없는 경우가 많으므로, 터미널 기반의 텍스트 편집기 사용법을 익히는 것이 필수적입니다. `vi`와 `vim`은 이러한 환경에서 가장 널리 사용되는 편집기입니다.

- **`vi`**: **V**isual edi**t**or의 약자로, 모든 유닉스 및 리눅스 시스템에 기본적으로 탑재되어 있는 강력한 텍스트 편집기입니다. 학습 곡선이 가파르지만, 숙련되면 마우스를 사용하지 않고도 매우 빠르게 파일을 편집할 수 있습니다.
- **`vim`**: **vi im**proved의 약자로, `vi`의 기능을 확장하고 개선한 버전입니다. 문법 강조, 되돌리기(undo), 고급 검색/치환 등 다양한 편의 기능을 제공하여 `vi`보다 사용자 친화적입니다.
- **Vi/Vim의 모드**: `vi/vim`은 효율적인 편집을 위해 여러 가지 모드로 나뉘어 작동합니다.
    - **보기 모드 (Normal/Command Mode)**: `vi/vim`을 처음 실행했을 때의 기본 모드입니다. 파일 내용을 탐색하고, 복사(y), 붙여넣기(p), 삭제(d), 검색(/) 등 대부분의 편집 명령어를 실행하는 모드입니다.
    - **입력 모드 (Insert Mode)**: 보기 모드에서 `i` (insert, 현재 커서 위치에 삽입), `a` (append, 현재 커서 뒤에 삽입), `o` (open, 현재 줄 아래에 새 줄 삽입) 등의 키를 누르면 진입합니다. 이 모드에서는 일반적인 텍스트 편집기처럼 내용을 입력하거나 수정할 수 있습니다.
    - **명령 모드 (Last Line/Ex Mode)**: 보기 모드에서 `:` (콜론)을 누르면 화면 하단에 명령 프롬프트가 나타나며, 파일 저장(w), 종료(q), 파일 열기 등 고급 명령을 수행할 수 있습니다.
        - `:w`: 파일 저장 (**w**rite)
        - `:q`: 편집기 종료 (**q**uit)
        - `:wq` 또는 `:x`: 저장 후 종료
        - `:q!`: 변경 사항을 저장하지 않고 강제 종료 (**q**uit **!**, 에러 무시)
        - `:set nu`: 줄 번호 표시
- **모드 전환**:
    - 보기 모드에서 `i` 또는 `a` → 입력 모드
    - 입력 모드에서 `[Esc]` 키 → 보기 모드
    - 보기 모드에서 `:` → 명령 모드

*실무 팁:* `vi/vim` 사용은 처음에는 다소 복잡하게 느껴질 수 있지만, 리눅스 시스템 관리나 서버 개발 환경에서는 매우 효율적인 도구이므로 꾸준한 연습을 통해 익숙해지는 것이 좋습니다. 설정 파일 수정 시 특히 유용합니다.

### 1.5 쉘 스크립트와 환경 변수

쉘 스크립트는 여러 리눅스 명령어를 텍스트 파일에 작성하여 순차적으로 실행할 수 있게 해주는 강력한 도구입니다. 반복적인 작업을 자동화하거나 복잡한 작업을 단순화하는 데 주로 사용됩니다.

- **`.sh`**: 쉘 스크립트 파일의 일반적인 확장자입니다. 파일의 맨 첫 줄에는 어떤 쉘(예: `#!/bin/bash`)로 스크립트를 실행할지 지정하는 셔뱅(Shebang) 라인을 추가하는 것이 관례입니다.
- **쉘 스크립트 실행**:
    - 현재 디렉터리에 있는 쉘 스크립트 파일은 `./파일명` 형태로 실행할 수 있습니다. (예: `./myscript.sh`) 이때 파일에 실행 권한(`chmod +x myscript.sh`)이 있어야 합니다.
    - `bash 파일명` 또는 `sh 파일명` 형태로 직접 인터프리터를 지정하여 실행할 수도 있습니다. 이 경우 실행 권한이 없어도 됩니다.
- **`/etc/bashrc`**: Bash 쉘의 전역(시스템 전체) 기본 설정값들이 저장되어 있는 파일입니다. 이 파일은 Bash 쉘 스크립트 문법으로 작성되어 있으며, 모든 사용자의 Bash 쉘 세션에 적용됩니다. 사용자별 설정은 주로 홈 디렉터리의 `.bashrc`, `.profile` 파일에 저장됩니다.
- **환경 변수 PATH**: 쉘이 명령어를 입력받았을 때 실행 가능한 바이너리 파일(실행 파일)을 찾는 경로들의 목록을 담고 있는 환경 변수입니다. PATH에 등록된 경로는 어디서든 해당 디렉터리 내의 명령어를 직접 이름으로 실행할 수 있게 해줍니다.
    - **`export PATH=$PATH:/etc/miniconda3/bin/`**: 이 명령어는 현재 PATH 환경 변수에 새로운 경로(`/etc/miniconda3/bin/`)를 추가합니다. 기존 PATH 값은 `$PATH`로 참조하며, 여러 경로는 콜론(`:`)으로 구분됩니다. 이 설정을 통해 `/etc/miniconda3/bin` 디렉터리에 있는 실행 파일들을 어느 위치에서든 직접 실행할 수 있게 되어 편리합니다.

- **`source 파일명`**: 설정 파일(예: `.bashrc`, `/etc/bashrc`, `.profile` 등)을 수정한 후, 해당 변경 사항을 현재 쉘 세션에 즉시 적용시키는 명령어입니다. `.` (점) 명령어로도 사용됩니다 (예: `. ~/.bashrc`). 이 명령어를 사용하지 않으면, 변경 사항을 적용하기 위해 쉘 세션을 종료하고 다시 로그인해야 합니다.

### 1.6 시스템 서비스 관리: systemctl, SELinux, Firewalld

리눅스 시스템의 안정성과 보안을 유지하기 위해서는 시스템에서 실행되는 다양한 서비스들을 효과적으로 관리해야 합니다.

- **`systemctl`**: `systemd` 데몬에 의해 관리되는 서비스(백그라운드에서 계속 실행되는 프로그램)들을 제어하는 명령어입니다. 과거의 `service` 명령어를 대체하며, 서비스의 시작, 종료, 재시작, 상태 확인, 부팅 시 자동 실행 설정 등을 수행합니다.
  - **`systemctl stop [서비스명]`**: 서비스 중지 (시스템 재부팅 시 다시 시작될 수 있음)
  - **`systemctl start [서비스명]`**: 서비스 시작
  - **`systemctl restart [서비스명]`**: 서비스 재시작
  - **`systemctl status [서비스명]`**: 서비스의 현재 상태 출력 (활성화 여부, 실행 중 여부, 로그 등)
  - **`systemctl disable [서비스명]`**: 부팅 시 서비스 자동 실행 비활성화 (영구적으로 적용)
  - **`systemctl enable [서비스명]`**: 부팅 시 서비스 자동 실행 활성화 (영구적으로 적용)
- **SELinux (Security-Enhanced Linux)**: 리눅스 커널에 통합된 보안 모듈로, **MAC (Mandatory Access Control)**이라는 강제 접근 제어 방식을 통해 시스템 보안을 강화합니다. 이는 기존의 임의 접근 제어(DAC) 방식보다 더욱 세밀한 보안 정책을 강제할 수 있어, 특정 프로세스가 특정 파일에만 접근하도록 제한하는 등의 강력한 제어가 가능합니다.
  - **설정 파일**: `/etc/selinux/config`
  - **비활성화**: 설정 파일에서 `SELINUX=disabled`로 변경하여 비활성화할 수 있습니다.

  *주의 사항:* SELinux 설정 변경은 `source` 명령어로 적용되지 않으며, 시스템을 재부팅해야만 적용됩니다. SELinux는 강력한 보안 기능이지만, 잘못 설정하면 시스템 접근 문제를 일으키거나 특정 서비스가 제대로 작동하지 않을 수 있으므로 신중한 이해와 관리가 필요합니다.
- **Firewalld**: CentOS/RHEL 7 이상에서 사용되는 동적 방화벽 관리 도구입니다. 기존의 `iptables`와 달리 서비스 재시작 없이도 방화벽 규칙을 동적으로 적용할 수 있는 장점이 있습니다. 이를 통해 네트워크 보안 정책을 더욱 유연하고 효율적으로 관리할 수 있습니다. `firewalld` 역시 `systemctl`을 통해 시작, 중지, 상태 확인 등이 가능합니다.

*개념 연결:* 리눅스에서 이름 뒤에 `d`가 붙는 프로세스(예: `firewalld`, `systemd`)는 일반적으로 백그라운드에서 실행되는 "데몬(daemon)"을 의미합니다. 이러한 데몬들은 시스템의 핵심 기능을 제공하며, 대부분 `systemctl` 명령어를 통해 관리됩니다.

### 1.7 리눅스의 DNS 설정 파일: /etc/resolv.conf

리눅스 시스템에서 도메인 이름 시스템(DNS) 설정을 관리하는 파일은 **`/etc/resolv.conf`** 입니다. 이 파일에는 시스템이 도메인 이름을 IP 주소로 변환하기 위해 어떤 DNS 서버(`nameserver`)를 사용할지, 그리고 검색 도메인(`search`) 등을 정의합니다. 웹사이트 접속, 네트워크 리소스 접근, 패키지 관리자의 저장소 접근 등 네트워크를 사용하는 모든 활동에서 이 파일의 설정이 중요하게 작용합니다. DHCP 클라이언트 등에 의해 자동으로 갱신될 수 있습니다.

### 1.8 성능 최적화의 이해: 저장 장치와 캐시

컴퓨터 시스템의 성능은 CPU, 메모리, 저장 장치, 네트워크 등 여러 요소에 의해 결정됩니다. 이 중에서도 저장 장치의 속도는 전체 시스템의 병목 현상에 큰 영향을 미치며, 이를 완화하기 위한 핵심 개념이 '캐시'입니다.

- **저장 장치 특성**: 일반적으로 가격이 높을수록 속도가 빠르고 용량이 적은 경향이 있습니다 (예: CPU 레지스터 > CPU 캐시 > RAM > SSD > HDD). 이 계층 구조는 비용과 성능의 균형을 맞추기 위함입니다.
- **병목 현상 (Bottleneck)**: 시스템의 특정 구성 요소(주로 속도가 느린 저장 장치나 낮은 대역폭의 네트워크)가 전체 시스템의 처리 속도를 제한하는 현상입니다. 이는 아무리 다른 고성능 부품이 있더라도 가장 느린 부품에 의해 전체 시스템의 성능이 저하됨을 의미합니다.
- **캐시 (Cache)**: 병목 현상을 완화하고 전체적인 시스템 성능을 개선하기 위한 핵심 개념입니다.
  - **정의**: 서로 다른 속도를 가진 두 저장 장치 사이에 존재하며, '자주 사용되거나 앞으로 사용될 가능성이 높은 값을 미리 임시로 저장'해두는 고속 완충 기억기(buffer)입니다.
  - **작동 원리**: CPU가 데이터에 접근할 때, 먼저 접근 속도가 훨씬 빠른 캐시를 확인합니다. 데이터가 캐시에 있으면(캐시 히트) 빠르게 가져오고, 없으면(캐시 미스) 느린 주 저장 장치에서 데이터를 가져와 캐시에 저장하여 다음 접근 시 활용합니다. 이를 통해 느린 장치에 대한 접근 횟수를 줄여 전체적인 평균 접근 시간을 단축시킵니다.
  - **예시**: CPU 캐시(L1, L2, L3), 디스크 캐시, 웹 브라우저 캐시, DNS 캐시 등 컴퓨터 시스템의 다양한 계층에서 캐시 개념이 활용됩니다.

*개념 연결:* 캐시 메모리는 '시간 지역성(최근 접근한 데이터는 다시 접근될 가능성이 높다)'과 '공간 지역성(최근 접근한 데이터의 주변 데이터도 접근될 가능성이 높다)'이라는 데이터 접근의 지역성 원리를 활용하여 성능을 향상시킵니다.

### 1.9 패키지 관리 및 소프트웨어 설치

리눅스에서는 소프트웨어(패키지)를 효율적으로 설치, 업데이트, 제거하기 위한 패키지 관리 시스템을 사용합니다. 이는 수동 설치의 복잡성과 의존성 문제를 해결해줍니다.

- **repo (Repository)**: 패키지(파일, 프로그램)들이 모여 저장되어 있는 중앙 저장소입니다. 리눅스 패키지 매니저는 특정 패키지를 설치하려고 할 때 등록된 repo에서 해당 패키지를 찾아 다운로드합니다.
- **리눅스 패키지 매니저**:
  - **apt**: Debian, Ubuntu 등 Debian 계열 리눅스 배포판에서 주로 사용됩니다. (Advanced Package Tool)
  - **dnf**: Fedora, CentOS 8+, RHEL 8+ 등 Red Hat 계열 리눅스 배포판에서 사용되며, 기존의 yum을 대체하는 최신 패키지 매니저입니다. (Dandified YUM)

  *주의 사항:* 새로운 repo를 등록한 후에는 패키지 목록을 최신 상태로 갱신하기 위해 반드시 `apt update` (Debian/Ubuntu) 또는 `dnf makecache`/`dnf update` (CentOS/RHEL)와 같은 '업데이트' 명령어를 실행해야 합니다. 그렇지 않으면 새로 추가된 패키지나 최신 버전의 패키지를 찾을 수 없습니다.
- **wget**: 웹에서 파일을 내려받는 데 사용되는 명령줄 유틸리티입니다. HTTP, HTTPS, FTP 프로토콜을 지원하여 웹 서버에서 파일을 쉽게 다운로드할 수 있습니다. 스크립트 내에서 파일을 다운로드할 때 유용합니다. (예: `wget [다운로드_URL]`)
- **Anaconda (v3) & Miniconda**: 파이썬 기반 데이터 과학 환경 구축을 위한 도구입니다.
  - **Anaconda**: 파이썬 및 데이터 과학/머신러닝 분야에서 자주 사용되는 수백 개의 파이썬 패키지(Numpy, Pandas, Scikit-learn, TensorFlow 등)를 한 번에 설치해주는 통합 배포판입니다. 가상 환경 관리에 매우 유용합니다.
  - **Miniconda**: Anaconda의 경량화 버전으로, 핵심적인 conda 패키지 관리자와 파이썬 인터프리터만 포함하고 있습니다. 필요한 패키지만 선택적으로 설치하여 디스크 공간을 절약하고 환경을 더 세밀하게 제어할 수 있습니다. 서버 환경에서 주로 사용됩니다.

### 1.10 대화형 개발 환경: Jupyter Notebook

**Jupyter Notebook**은 웹 브라우저 환경에서 파이썬 코드를 작성하고, 실행하며, 결과를 텍스트, 이미지, 그래프 등 다양한 형태로 시각적으로 확인하고 공유할 수 있는 강력한 IDE(통합 개발 환경)입니다. 데이터 분석, 머신러닝 모델 개발, 교육 자료 작성 등에 널리 활용됩니다.

- **서버 실행**: 터미널에서 `jupyter notebook` 명령어로 서버를 시작합니다. 특정 IP 주소와 포트 번호를 지정하여 외부에서 접근 가능하도록 설정할 수 있습니다.
  - **예시**: `jupyter notebook --allow-root --ip=`아이피 `--port=`포트번호
    `--allow-root` 옵션은 root 권한으로 실행할 때 필요하며, 보안상 권장되지는 않습니다. 일반적으로는 일반 사용자 계정으로 실행하는 것이 좋습니다.
- **접속**: 서버 실행 후 출력되는 URL 또는 지정된 아이피`:`포트번호로 웹 브라우저에서 접속합니다.
- **인증 방식**:
  - **기본**: 토큰 인증 방식을 사용하며, 서버 시작 시 터미널에 생성되는 토큰 문자열을 복사하여 웹 브라우저에 입력해야 합니다.
  - **비밀번호 인증으로 변경**: 매번 토큰을 입력하는 불편함을 해소하기 위해 비밀번호 인증 방식으로 변경할 수 있습니다. `jupyter notebook password` 명령어를 실행하여 비밀번호를 설정합니다. 설정된 비밀번호는 이후 Jupyter Notebook 접속 시 사용됩니다.
- **셀(Cell)의 동작**: Jupyter Notebook은 코드를 '셀' 단위로 실행합니다. 각 셀은 독립적인 파이썬 코드 블록으로 볼 수 있지만, Jupyter 커널은 모든 셀에 걸쳐 상태를 공유합니다. 즉, 먼저 실행된 셀에서 정의된 변수나 함수 등의 값은 이후 실행되는 다른 셀에서 접근하고 사용할 수 있습니다. 이는 마치 하나의 파이썬 스크립트처럼 연속적으로 코드를 실행하는 것과 같지만, 각 단계를 개별적으로 실행하고 결과를 즉시 확인할 수 있다는 장점이 있습니다.

### 1.11 터미널 활용 팁 및 단축키

명령줄 인터페이스(CLI) 작업을 더욱 효율적으로 만들어주는 몇 가지 유용한 팁입니다.

- **&& (AND 연산자)**: 터미널에서 여러 명령어를 순차적으로 실행하고 싶을 때 사용합니다. 앞의 명령어가 **성공적으로 완료(종료 코드 0)**되어야 뒤의 명령어가 실행됩니다. (예: `mkdir mydir && cd mydir` - 디렉터리를 생성하고 성공하면 그 안으로 이동)
- **| (Pipe, 파이프)**: 앞 명령어의 **표준 출력(stdout)**을 뒷 명령어의 **표준 입력(stdin)**으로 전달합니다. 여러 명령어를 조합하여 복잡한 작업을 수행하거나 데이터 흐름을 조작할 때 매우 유용합니다. (예: `ls -l | grep .txt` - 현재 디렉터리 파일 목록을 상세하게 보여준 후, 그 결과에서 ".txt" 문자열을 포함하는 줄만 필터링하여 출력)
- **Control+C (^C)**: 현재 터미널에서 실행 중인 프로세스나 작업을 즉시 취소하고 중단하는 단축키입니다. 응답하지 않는 명령어, 무한 루프에 빠진 스크립트 등을 강제로 종료할 때 유용합니다.
- **쉘 히스토리**: 키보드의 **방향키 (위/아래)**를 사용하여 이전에 입력했던 명령어들을 쉽게 찾아 다시 실행하거나 수정할 수 있습니다. 이는 반복적인 작업이나 명령어 재사용 시 시간을 절약해주는 매우 기본적인 기능입니다. 히스토리 파일은 `~/.bash_history` 등에 저장됩니다.

# Chapter 2: 터미널 명령어 및 쉘 스크립팅

리눅스 환경에서 시스템을 효율적으로 제어하고 작업을 자동화하기 위해서는 터미널 명령어에 대한 이해와 쉘 스크립팅 능력이 필수적입니다. 이 챕터에서는 리눅스 시스템의 기본적인 개념부터 자주 사용되는 핵심 명령어, 그리고 쉘 스크립팅에 필요한 배경지식까지 폭넓게 다룹니다.

## 2.1 리눅스 시스템의 기본 개념

### 2.1.1 초기 프로세스와 시스템 상태

- **`init` 프로세스 (PID 1)**: 리눅스 시스템이 부팅된 후 가장 먼저 생성되는 프로세스입니다. 모든 다른 프로세스의 부모 역할을 합니다.
  - **0**: 시스템 종료 (Shutdown)
  - **6**: 시스템 재부팅 (Reboot)

  *실무 팁:* PID 1은 시스템의 안정성을 보장하며, 현대 리눅스 시스템에서는 대부분 `systemd`가 `init` 프로세스의 역할을 대체하고 있습니다.

### 2.1.2 DNS 설정

- **`/etc/resolv.conf`**: 리눅스 시스템의 DNS(Domain Name System) 설정 파일입니다. 도메인 이름을 IP 주소로 변환하는 데 사용되는 DNS 서버의 주소가 이곳에 명시됩니다.

  *시험 포인트:* 네트워크 문제 발생 시 이 파일을 가장 먼저 확인해야 할 설정 파일 중 하나입니다.

### 2.1.3 저장 장치와 캐시(Cache)

- **저장 장치의 특성**: 일반적으로 가격이 높을수록 속도가 빠르고 용량이 적은 경향이 있습니다. 예를 들어, CPU 캐시 > RAM > SSD > HDD 순으로 속도와 가격이 감소하고 용량이 증가합니다.
- **병목 현상 (Bottleneck)**: 서로 다른 속도를 가진 저장 장치나 시스템 구성 요소들 사이에서, 느린 부분이 전체 시스템 성능을 저하시키는 현상입니다.
- **캐시 (Cache)**: 병목 현상을 완화하고 전체적인 시스템 성능을 개선하기 위한 핵심 개념입니다.
  - **정의**: 속도가 빠른 저장 장치에 '자주 사용될 것으로 예상되는 값' 또는 '최근에 사용된 값'을 임시로 저장해두는 고속 완충 기억 장치입니다.
  - **목적**: 메인 저장 장치에 직접 접근하는 횟수를 줄여 데이터 접근 속도를 향상시킵니다. CPU 캐시, 디스크 캐시, 웹 캐시 등 다양한 계층에서 활용됩니다.

## 2.2 터미널 명령어의 활용

리눅스 터미널은 명령어를 입력하여 시스템과 상호작용하는 강력한 도구입니다. 다음은 자주 사용되는 핵심 명령어들입니다.

### 2.2.1 기본 파일 및 디렉터리 관리 명령어

- **`pwd` (Print Working Directory)**: 현재 작업 중인 디렉터리의 절대 경로를 화면에 출력합니다.

  ```
  pwd
  ```

- **`ls` (List)**: 현재 디렉터리 또는 지정된 디렉터리의 파일 및 하위 디렉터리 정보를 표시합니다.
  - **`-a` (all)**: 숨김 파일(이름이 .으로 시작하는 파일)을 포함하여 모든 파일을 표시합니다.
  - **`-l` (long)**: 파일의 상세 정보를 긴 형식으로 출력합니다.
  - **`-al` 또는 `ll`**: `-a`와 `-l` 옵션을 함께 사용하여 모든 파일의 상세 정보를 출력합니다.
  - **출력 내용 분석 (`ls -l` 기준)**:

    ```
    -rwxr-xr-x 1 user group 1024 Jan 1 10:00 filename.txt
    ```

    - 첫 번째 문자 (`-` 또는 `d`): `-`는 일반 파일, `d`는 디렉터리를 의미합니다.
    - 권한 (`rwxr-xr-x`): 파일 소유자, 그룹, 다른 사용자의 읽기(r), 쓰기(w), 실행(x) 권한을 나타냅니다.
    - 링크 수 (`1`): 해당 파일에 대한 하드링크 수입니다.
    - 소유자 (`user`): 파일의 소유자 계정입니다.
    - 그룹 (`group`): 파일이 속한 그룹입니다.
    - 파일 크기 (`1024`): 바이트 단위의 파일 크기입니다.
    - 수정 일자 (`Jan 1 10:00`): 파일이 마지막으로 수정된 날짜와 시간입니다.

        - 파일 이름 (`filename.txt`): 파일 또는 디렉터리 이름입니다.
    - **색상 표시**: 터미널 애플리케이션에 따라 파일 종류(실행 파일, 디렉터리, 압축 파일 등)에 따라 다른 색상으로 표시하여 시각적인 구분과 편의성을 제공합니다.
- **`cat` (Concatenate)**: 하나 이상의 텍스트 파일을 화면에 출력하거나, 여러 파일을 연결하여 새로운 파일로 만듭니다. 주로 텍스트 파일의 내용을 빠르게 확인하는 데 사용됩니다.

```
cat file.txt
```

```
cat file1.txt file2.txt > combined.txt
```

#### 2.2.2 파일/디렉터리 권한 관리

- **권한 분류**:
    - **사용자별 분류**: 소유자 (User), 그룹 (Group), 다른 사용자 (Others)
    - **권한 종류별 분류**: 읽기 (r, read), 쓰기 (w, write), 실행 (x, execute)

    *핵심 개념:* 쓰기 권한은 파일의 내용을 수정할 수 있는 권한을 의미하고, 디렉터리에서는 파일 생성/삭제/이름 변경 권한을 의미합니다. 실행 권한은 파일을 프로그램처럼 실행할 수 있는 권한을 의미하고, 디렉터리에서는 해당 디렉터리 내부로 진입할 수 있는 권한을 의미합니다.
- **`chmod` (Change Mode)**: 파일 또는 디렉터리의 권한을 수정하는 명령어입니다.
    - **기호 모드 (Symbolic mode)**:
        - `u` (user), `g` (group), `o` (others), `a` (all)
        - `+` (권한 추가), `-` (권한 제거), `=` (권한 설정)
        - 예시: `chmod u+x [파일명]` (현재 유저에게 해당 파일의 실행 권한을 부여합니다.)
    - **숫자 모드 (Numeric mode)**:
        - `r=4`, `w=2`, `x=1`
        - 예시: `chmod 755 [파일명]` (소유자는 읽기/쓰기/실행, 그룹 및 다른 사용자는 읽기/실행 권한을 부여합니다. 7=4+2+1, 5=4+1)

#### 2.2.3 관리자 권한 및 웹 파일 다운로드

- **`sudo` (Superuser Do)**: 일반 사용자가 root(관리자) 권한으로 명령어를 실행할 수 있도록 합니다. 중요한 시스템 변경이나 민감한 파일 접근 시 사용됩니다.

```
sudo apt update
```

    *주의사항:* `sudo`는 강력한 권한을 부여하므로, 사용 시 항상 주의해야 합니다.
- **`wget`**: 웹 서버에서 파일을 내려받는 유틸리티입니다. URL을 지정하여 원격 파일을 현재 디렉터리로 다운로드합니다.

```
wget https://example.com/file.zip
```

#### 2.2.4 텍스트 에디터 (vi/vim)

- **`vi` (Visual Editor)**: 리눅스/유닉스 환경에서 가장 기본적인 CLI(Command Line Interface) 텍스트 에디터입니다.
- **`vim` (Vi Improved)**: `vi`를 확장하여 기능과 편의성을 개선한 에디터입니다.

    *핵심 개념:* `vi`/`vim`은 모드 기반 에디터로, 여러 모드를 전환하며 작업을 수행합니다.
    - **보기/일반 모드 (Normal/Command mode)**: 에디터 실행 시 기본 모드. 파일 내용을 보거나 이동, 삭제, 복사 등의 명령을 수행합니다.
    - **편집 모드 (Insert mode)**: `i` (insert), `a` (append) 등의 키를 눌러 진입합니다. 텍스트를 입력하거나 수정할 수 있습니다. `[Esc]` 키를 누르면 보기 모드로 돌아갑니다.
    - **명령 모드 (Command-line mode)**: 보기 모드에서 `:` (콜론)을 입력하여 진입합니다. 파일 저장, 종료, 검색, 바꾸기 등의 명령을 수행합니다.
        - `:w`: 파일 저장 (write)
        - `:q`: 에디터 종료 (quit)
        - `:wq` 또는 `:x`: 파일 저장 후 종료
        - `:q!`: 변경 사항을 저장하지 않고 강제 종료 (에러 무시)

#### 2.2.5 서비스 관리 (systemctl)

- **`systemctl` (System Control)**: `systemd` 데몬(내부적으로 계속 실행되는 프로그램)으로 관리되는 서비스의 시작/종료/재시작/상태 확인/부팅 시 자동 실행 등을 제어하는 명령어입니다. 과거의 `service` 명령어를 대체합니다.

  *데몬(Daemon)이란?* 백그라운드에서 특정 작업을 처리하기 위해 지속적으로 실행되는 프로그램을 의미합니다. 이름 뒤에 'd'가 붙는 경우가 많습니다 (예: `sshd`, `httpd`, `firewalld`).

  - `systemctl start [서비스명]`: 서비스 시작
  - `systemctl stop [서비스명]`: 서비스 정지 (시스템 재부팅 시 다시 시작될 수 있음)
  - `systemctl restart [서비스명]`: 서비스 재시작
  - `systemctl status [서비스명]`: 서비스 상태 확인
  - `systemctl enable [서비스명]`: 부팅 시 서비스 자동 실행 활성화
  - `systemctl disable [서비스명]`: 부팅 시 서비스 자동 실행 비활성화

### 2.3 쉘 스크립팅과 환경 변수

#### 2.3.1 쉘 스크립트 파일

- **`.sh` 확장자**: 쉘 스크립트 파일의 일반적인 확장자입니다.
- **실행 방법**: 현재 디렉터리에 있는 쉘 스크립트 파일을 실행하려면 `./파일명` 형식을 사용합니다.

  ```
  ./myscript.sh
  ```

  *참고:* 스크립트 파일에 실행 권한(`chmod u+x myscript.sh`)이 있어야 합니다.

#### 2.3.2 쉘 환경 설정 파일

- **`/etc/bashrc`**: bash 쉘의 전역 기본 설정값이 저장되어 있는 파일입니다. bash 쉘 스크립트 문법으로 작성되어 있습니다.
- **`source [파일명]`**: 설정 파일을 수정한 후 현재 쉘 세션에 수정 사항을 즉시 적용시키는 명령어입니다. 일종의 '새로고침' 역할을 합니다. 이 명령어를 사용하지 않으면 별도의 로그아웃 후 로그인 과정을 거쳐야 변경 사항이 적용됩니다.

  ```
  source ~/.bashrc
  ```

#### 2.3.3 환경 변수 (PATH)

- **`PATH` 환경 변수**: 터미널에 명령어를 입력했을 때, 쉘이 실행 가능한 바이너리 파일을 찾아보는 디렉터리 경로들의 목록입니다.
- **`export PATH=$PATH:/path/to/add/bin/`**: 환경 변수 PATH에 새로운 경로를 추가하는 방법입니다.
  - `$PATH`: 현재 PATH 환경 변수에 설정된 값들을 불러옵니다.
  - `:` (콜론): 여러 경로를 구분하는 데 사용됩니다.
  - `/path/to/add/bin/`: 새로 추가하려는 디렉터리 경로입니다.

  *예시:* `export PATH=$PATH:/etc/miniconda3/bin/` 명령어를 통해 miniconda의 실행 파일들을 PATH에 추가하여 어느 디렉터리에서든 miniconda 관련 명령어를 실행할 수 있게 합니다.

  *실무 팁:* 이 설정을 영구적으로 적용하려면 `~/.bashrc` 또는 `/etc/profile` 같은 쉘 설정 파일에 추가해야 합니다.

### 2.4 패키지 관리

- **`repo` (Repository)**: 소프트웨어 패키지(파일, 프로그램)들이 저장되어 있는 중앙 저장소입니다.
- **리눅스 패키지 매니저**: `apt` (Debian/Ubuntu 계열), `dnf` (Fedora/CentOS/RHEL 계열) 등은 특정 패키지를 설치하려고 할 때 등록된 `repo`에서 해당 패키지를 찾아 자동으로 다운로드 및 설치를 수행합니다.
  - 패키지 매니저는 `repo` 목록을 주기적으로 `update`하여 최신 패키지 정보를 동기화해야 합니다.
  - 예시: `sudo apt update`, `sudo dnf update`

### 2.5 보안 및 방화벽

#### 2.5.1 SELinux (Security-Enhanced Linux)

- **정의**: 리눅스 커널에 통합된 보안 모듈로, 강제 접근 제어 (MAC, Mandatory Access Control)를 구현합니다. 시스템의 잠재적인 보안 취약점을

줄이고, 악성 코드로부터 시스템을 보호하는 데 도움을 줍니다.

- **설정 파일**: `/etc/selinux/config`
- **비활성화**: `SELINUX=disabled`로 설정하여 비활성화할 수 있습니다.

  *주의사항:* `selinux` 설정은 `source` 명령어로 즉시 적용되지 않으며, 시스템을 재부팅해야 변경 사항이 적용됩니다.

  *실무 팁:* `selinux`는 강력한 보안 기능이지만, 때로는 특정 애플리케이션의 작동을 방해할 수 있습니다. 문제 해결을 위해 일시적으로 비활성화하는 경우가 있으나, 보안상 다시 활성화하거나 적절한 정책을 설정하는 것이 중요합니다.

#### 2.5.2 Firewalld

- **정의**: CentOS/RHEL 7 이상에서 작동하는 동적 방화벽 관리 도구입니다. 기존의 `iptables`가 정적인 규칙을 사용한 것과 달리, `firewalld`는 서비스 재시작 없이도 방화벽 규칙을 동적으로 적용할 수 있는 장점이 있습니다.
- **특징**: 이름 뒤의 'd'는 데몬(daemon)을 의미하며, `systemctl`을 통해 서비스를 관리할 수 있습니다.
  - 예시: `systemctl start firewalld`, `systemctl status firewalld`

### 2.6 개발 환경 도구

#### 2.6.1 Anaconda & Miniconda

- **Anaconda (v3)**: 데이터 과학 및 머신러닝을 위한 파이썬 배포판입니다. 파이썬 인터프리터와 함께 데이터 분석, 과학 계산에 자주 사용되는 수많은 파이썬 패키지(NumPy, Pandas, Scikit-learn 등)를 미리 설치하여 제공합니다.
- **Miniconda**: Anaconda의 경량화 버전입니다. 파이썬 인터프리터와 `conda` 패키지 관리자만 포함되어 있으며, 필요한 패키지는 사용자가 직접 설치할 수 있습니다.

  *선택 가이드:* 전체 환경이 필요한 경우 Anaconda를, 최소한의 환경에서 시작하고 싶은 경우 Miniconda를 선택하는 것이 좋습니다.

#### 2.6.2 Jupyter Notebook

- **정의**: 웹 브라우저를 통해 파이썬 코드를 작성하고 실행하며, 그 결과를 실시간으로 확인하고 문서화할 수 있는 IDE(통합 개발 환경)입니다. 코드, 텍스트(마크다운), 이미지 등을 하나의 문서(Notebook)에 통합하여 대화식 코딩에 매우 유용합니다.
- **서버 실행 및 접속**:

  ```
  jupyter notebook --allow-root --ip=[아이피 주소] --port=[포트번호]
  ```

  - `--allow-root`: root 사용자로도 Jupyter Notebook을 실행할 수 있도록 허용합니다. (보안상 권장되지는 않지만, 특정 환경에서 필요할 수 있습니다.)
  - `--ip`: Jupyter 서버가 수신할 IP 주소를 지정합니다. `0.0.0.0`으로 설정하면 모든 네트워크 인터페이스에서 접근 가능합니다.
  - `--port`: Jupyter 서버가 사용할 포트 번호를 지정합니다.

  서버가 실행되면 웹 브라우저에서 `http://[아이피 주소]:[포트번호]`로 접속할 수 있습니다.
- **인증 방식 변경 (토큰 → 비밀번호)**:

  기본적으로는 토큰 인증 방식을 사용하지만, 매번 토큰을 복사하는 것이 번거로울 수 있습니다. 비밀번호 인증 방식으로 변경할 수 있습니다.

  ```
  jupyter notebook password
  ```

  이 명령어를 실행하면 새 비밀번호를 설정할 수 있습니다.
- **셀(Cell)의 동작**:
  - Jupyter Notebook의 각 셀은 독립적인 파이썬 코드 블록처럼 보이지만, 기본적으로 하나의 커널(Kernel)을 공유합니다.
  - 따라서 먼저 실행한 셀에서 정의된 변수나 함수 등의 값은 다음에 실행하는 셀에서 접근하여 사용할 수 있습니다. 이 특성을 활용하여 단계별로 코드를 개발하고 테스트할 수 있습니다.

### 2.7 기타 유용한 기능

#### 2.7.1 명령어 연산자

- **&& (AND)**: 여러 명령어를 순차적으로 실행할 때 사용합니다. 앞선 명령어가 성공적으로(종료 코드 0) 완료되어야 다음 명령어를 실행합니다.

  ```
  command1 && command2
  ```

*예시:* `make && make install` (컴파일이 성공해야 설치를 진행)

- **| (Pipe)**: 앞 명령어의 표준 출력(stdout)을 뒷 명령어의 표준 입력(stdin)으로 전달합니다. 명령어를 연결하여 복잡한 작업을 효율적으로 수행할 수 있습니다.

```
command1 | command2
```

*예시:* `ls -l | grep .txt` (현재 디렉터리 파일 목록을 상세하게 출력한 후, 그 결과에서 '.txt' 문자열을 포함하는 라인만 필터링하여 출력)

#### 2.7.2 터미널 단축키 및 히스토리

- **Control+C (^C)**: 현재 터미널에서 진행 중인 작업을 강제로 취소하거나 중단합니다. 무한 루프에 빠진 스크립트나 응답 없는 프로그램 종료에 유용합니다.
- **쉘 히스토리**: 터미널에서 방향키 위(↑)를 누르면 이전에 입력했던 명령어들을 순서대로 다시 불러올 수 있습니다. 이는 반복적인 명령어 입력 작업을 줄여 생산성을 높여줍니다.

# Chapter 3: 소프트웨어 및 파이썬 환경 관리

리눅스 환경에서 소프트웨어와 파이썬 개발 환경을 효율적으로 관리하는 것은 안정적이고 생산적인 작업 흐름을 위해 필수적입니다. 이 챕터에서는 리눅스 소프트웨어 설치 및 관리의 기본 개념부터 파이썬 특정 환경 설정에 이르기까지, 학습자가 시스템을 효과적으로 제어하고 활용할 수 있도록 돕는 내용을 다룹니다.

## 3.1. 리눅스 소프트웨어 패키지 관리

리눅스에서 소프트웨어를 설치하고 관리하는 주된 방법은 패키지 매니저를 활용하는 것입니다. 이는 의존성 문제를 해결하고, 소프트웨어 업데이트를 용이하게 하여 시스템 안정성을 높이는 데 기여합니다.

- **리포지토리 (Repository, Repo)**
  - **개념:** 소프트웨어 패키지(실행 파일, 라이브러리, 설정 파일 등)를 저장하고 있는 중앙 저장소입니다. 리눅스 배포판은 공식 리포지토리를 제공하며, 필요에 따라 서드파티 리포지토리를 추가할 수 있습니다.
  - **작동 방식:** 리눅스 패키지 매니저는 특정 패키지를 설치하려고 할 때, 시스템에 등록된 리포지토리에서 해당 패키지를 찾아 다운로드하고 설치합니다.
  - **실무 팁:** 새로운 리포지토리를 추가한 후에는 반드시 `update` 명령어를 실행하여 패키지 목록을 최신 상태로 갱신해야 합니다. (예: `sudo apt update` 또는 `sudo dnf check-update`)
- **패키지 매니저**
  - **종류:** 리눅스 배포판에 따라 `apt` (Debian/Ubuntu 계열), `dnf` 또는 `yum` (CentOS/RHEL 계열) 등 다양한 패키지 매니저가 사용됩니다. 이들은 패키지 검색, 설치, 제거, 업데이트 등의 기능을 제공합니다.
- **`wget`: 웹 파일 다운로드 유틸리티**
  - 웹 서버에서 파일을 다운로드할 때 사용하는 강력한 명령줄 유틸리티입니다. 패키지 매니저에 등록되지 않은 소프트웨어 설치 파일(예: `.sh` 스크립트)을 직접 다운로드할 때 유용합니다.
  - **예시:** `wget https://example.com/software-installer.sh`

## 3.2. 파이썬 환경 관리: Anaconda와 Miniconda

파이썬 개발 시 여러 프로젝트가 서로 다른 버전의 파이썬이나 라이브러리를 필요로 하는 경우가 많습니다. Anaconda나 Miniconda는 이러한 의존성 충돌 문제를 해결하고, 격리된 개발 환경을 구축하는 데 도움을 줍니다.

- Anaconda (v3)
  - **개념:** 파이썬 인터프리터와 함께 과학 계산, 데이터 분석, 머신러닝 등에 자주 사용되는 수백 개의 파이썬 패키지들을 미리 포함하고 있는 배포판입니다. 초보자에게 편리하지만, 용량이 크다는 단점이 있습니다.
- Miniconda
  - **개념:** Anaconda의 경량화 버전입니다. 파이썬 인터프리터와 `conda` 패키지 매니저만을 포함하고 있어, 필요한 패키지를 사용자가 직접 설치하여 환경을 구성할 수 있습니다. 더 가볍고 유연한 환경을 선호할 때 적합합니다.
  - **설치 과정 예시:**
    1. `wget`으로 Miniconda 설치 스크립트 다운로드 (예: `wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh`)
    2. 다운로드된 쉘 스크립트 실행 (`./Miniconda3-latest-Linux-x86_64.sh`)

## 3.3. 쉘 환경 설정 및 기본 명령어

소프트웨어를 설치하고 실행하기 위해서는 리눅스 쉘 환경을 이해하고 적절히 설정하는 것이 중요합니다.

- **`.sh` (쉘 스크립트 파일)**
  - 리눅스 터미널 명령어들을 텍스트 파일에 저장해 순차적으로 실행하는 스크립트 파일입니다. 소프트웨어 설치 스크립트나 자동화된 작업을 수행하는 데 사용됩니다.
  - **실행 방법:** 현재 디렉터리에 있는 쉘 스크립트는 `./파일명` 형태로 실행합니다. (예: `./install_script.sh`)
- **`sudo`: 슈퍼유저(root) 권한 실행**
  - 일반 사용자가 관리자(root) 권한이 필요한 명령어를 실행할 때 사용합니다. 시스템 전체에 영향을 주는 소프트웨어 설치나 설정 변경 시 필수적입니다.
  - **예시:** `sudo apt update`
- **환경 변수 `PATH`**

    - **개념:** 쉘이 명령어를 입력했을 때 실행 가능한 바이너리 파일을 찾아보는 디렉터리 경로들의 목록입니다.
    - **설정 및 활용:**
        - Miniconda와 같이 특정 경로에 설치된 실행 파일(예: `conda` 명령어)을 시스템의 어느 위치에서든 실행할 수 있도록 `PATH`에 추가해야 합니다.
        - `export PATH=$PATH:/etc/miniconda3/bin/`: 이 명령어는 기존 `PATH` 값에 `/etc/miniconda3/bin/` 경로를 추가합니다. 콜론(`:`)으로 경로들을 구분합니다.
        - **주의:** 이 설정은 현재 쉘 세션에만 유효합니다.
- **`/etc/bashrc` 및 `source` 명령어**
    - **`/etc/bashrc`:** Bash 쉘의 기본 설정값이 저장되어 있는 파일입니다. 시스템 전체 사용자에게 적용되는 환경 변수나 쉘 설정을 이곳에 추가할 수 있습니다. Bash 쉘 스크립트 문법으로 작성됩니다.
    - **`source 파일명`:** 설정 파일(예: `.bashrc`, `.profile` 등)을 수정한 후, 변경 사항을 현재 쉘 세션에 즉시 적용시키는 명령어입니다. 이 명령어를 사용하지 않으면 별도의 로그아웃 및 로그인(또는 터미널 재실행)이 필요합니다.
    - **예시:** `source ~/.bashrc` (사용자 홈 디렉터리의 `.bashrc` 파일을 새로고침)
    - **실무 팁:** 일반적으로 사용자별 설정은 `~/.bashrc` 또는 `~/.profile`에, 시스템 전체 설정은 `/etc/bashrc` 또는 `/etc/profile`에 추가합니다.

#### 3.3.1. 자주 사용하는 리눅스 기본 명령어

소프트웨어 관리 및 환경 설정 과정에서 자주 사용되는 핵심 명령어들입니다.

- **`cat` (concatenate)**
    - 주로 텍스트 파일을 화면에 출력하는 데 사용됩니다. 여러 파일을 하나로 연결하여 출력하거나, 간단하게 파일 내용을 확인할 때 유용합니다.
    - **예시:** `cat /etc/resolv.conf` (DNS 설정 파일 내용 확인)
- **`pwd` (print working directory)**
    - 현재 작업 중인 폴더의 절대 경로를 출력합니다. 현재 위치를 파악하는 데 필수적입니다.
- **`ls` (list)**
    - 현재 디렉터리의 파일 및 디렉터리 정보를 표시합니다. 다양한 옵션을 통해 상세 정보를 확인할 수 있습니다.
        - `-a`: 숨김 파일(`.`으로 시작하는 파일) 포함 모든 파일 표시.
        - `-l`: 파일의 상세 정보 (권한, 소유자, 그룹, 크기, 수정일자 등) 출력.
        - `-al` 또는 `ll`: `-a`와 `-l` 옵션을 함께 사용 (숨김 파일 포함 상세 목록).
    - **출력 정보 해석:**
        - 맨 앞 문자: `-` (파일), `d` (디렉토리).
        - 이후 9개 문자: 파일/디렉토리의 권한 (소유자/그룹/다른 사용자 각각 읽기(r), 쓰기(w), 실행(x)).
    - **팁:** 터미널 애플리케이션에 따라 파일 종류마다 색상이 다르게 표시되어 시각적으로 구분하기 쉽습니다.
- **`chmod` (change mode)**
    - 파일 또는 디렉터리의 권한을 수정하는 명령어입니다. 특히 실행 권한(x)은 스크립트나 바이너리 파일을 실행하기 위해 중요합니다.
    - **권한 분류:**
        - **사용자별:** 소유자 (u), 그룹 (g), 다른 사용자 (o), 모든 사용자 (a).
        - **권한 종류별:** 읽기 (r, read), 쓰기 (w, write/수정), 실행 (x, execute).
    - **예시:** `chmod u+x [파일명]` (현재 유저(u)에게 해당 파일의 실행(x) 권한을 부여).
    - **참고:** 숫자로 권한을 설정할 수도 있습니다 (예: `chmod 755 filename`). r=4, w=2, x=1.
- **`vi` / `vim` (Visual Editor / VI Improved)**
    - CLI(Command Line Interface) 환경에서 사용하는 강력한 텍스트 에디터입니다. 설정 파일 등을 수정할 때 필수적으로 사용됩니다.
    - **주요 모드:**
        1. **보기 모드 (Normal/Command Mode):** 파일 내용을 보고, 커서를 이동하며, 삭제/복사 등의 명령을 내리는 기본 모드입니다.
        2. **편집 모드 (Insert Mode):** `i` (현재 커서 위치에 삽입), `a` (현재 커서 다음 위치에 삽입) 등의 명령어를 통해 진입하며, 텍스트를 입력할 수 있습니다. `[Esc]` 키를 눌러 보기 모드로 돌아갑니다.
        3. **명령 모드 (Ex Mode):** 보기 모드에서 `:` (콜론)을 입력하여 진입합니다. 파일 저장, 종료, 검색/교체 등 고급 명령을 수행합니다.

    - `:w`: 파일 저장 (write).
    - `:q`: 파일 종료 (quit).
    - `:wq`: 저장 후 종료.
    - `:q!`: 저장하지 않고 강제 종료 (변경 사항 무시).

### 3.4. Jupyter Notebook: 파이썬 통합 개발 환경

Jupyter Notebook은 웹 브라우저 기반의 파이썬 통합 개발 환경(IDE)으로, 코드를 작성하고 실행하며 결과를 시각적으로 확인하는 데 매우 유용합니다.

- **주요 기능 및 특징**
    - **코드 셀:** 파이썬 코드를 작성하고 실행합니다.
    - **마크다운 셀:** 문서화, 주석, 설명 등을 마크다운 문법으로 작성합니다.
    - **실행 환경:** 각 셀은 기본적으로 하나의 파이썬 코드 파일과 유사하게 작동하지만, 먼저 실행한 셀에서 정의된 변수나 함수는 이후 실행되는 셀에서 접근 가능합니다. 이는 대화형 데이터 분석에 유리합니다.
- **Jupyter Notebook 서버 실행**
    - Jupyter Notebook은 웹 기반이므로, 서버를 먼저 실행해야 웹 브라우저를 통해 접속할 수 있습니다.
    - **명령어 예시:** `jupyter notebook --allow-root --ip=`아이피주소 `--port=`포트번호
        - `--allow-root`: root 권한으로 실행을 허용합니다 (일반적으로 보안상 권장되지 않으나, 특정 환경에서는 필요할 수 있음).
        - `--ip=`아이피주소: 서버가 바인딩될 IP 주소를 지정합니다 (예: `0.0.0.0`은 모든 인터페이스에서 접근 허용).
        - `--port=`포트번호: 서버가 사용할 포트 번호를 지정합니다 (기본값은 8888).
- **인증 방식 변경: 토큰 → 비밀번호**
    - Jupyter Notebook은 기본적으로 토큰 인증 방식을 사용하지만, 매번 토큰을 복사하는 것이 번거로울 수 있습니다.
    - **비밀번호 설정:** `jupyter notebook password` 명령어를 통해 비밀번호를 설정하여 더 편리하게 접속할 수 있습니다.

### 3.5. 시스템 서비스 관리: `systemctl`

리눅스에서 백그라운드로 실행되는 프로그램(서비스 또는 데몬)들은 `systemd`라는 시스템 및 서비스 관리자에 의해 제어됩니다. `systemctl` 명령어는 이러한 서비스들을 관리하는 데 사용됩니다.

- **`systemctl` 명령어의 주요 기능**
    - `stop [서비스명]`: 서비스 중지 (시스템 재부팅 시 다시 시작될 수 있음).
    - `disable [서비스명]`: 서비스의 부팅 시 자동 실행 비활성화 (재부팅해도 유지).
    - `start [서비스명]`: 서비스 시작.
    - `status [서비스명]`: 서비스의 현재 상태(실행 중인지, 오류가 있는지 등) 출력.
    - `enable [서비스명]`: 서비스의 부팅 시 자동 실행 활성화.
    - `restart [서비스명]`: 서비스 재시작.
- **예시: `firewalld` (동적 방화벽 데몬)**
    - CentOS/RHEL 7 이상에서 사용되는 방화벽 관리 도구입니다. 기존 `iptables`와 달리 서비스 재시작 없이도 동적으로 방화벽 규칙을 적용할 수 있습니다.
    - `systemctl status firewalld`: 방화벽 서비스 상태 확인.
    - `systemctl stop firewalld`: 방화벽 서비스 중지.
    - `systemctl disable firewalld`: 방화벽 서비스의 자동 시작 비활성화.
    - **참고:** `selinux` (Security-Enhanced Linux)는 리눅스 커널에 통합된 보안 모듈이지만, 그 비활성화(`/etc/selinux/config` 파일에서 `SELINUX=disabled`)는 `systemctl`로 직접 제어되지 않으며, 시스템 재부팅 후에 적용됩니다.

### 3.6. 유용한 리눅스 단축키

- **`Control+C` (`^C`)**: 현재 터미널에서 진행 중인 작업을 강제로 취소하거나 중단할 때 사용합니다. 무한 루프에 빠진 스크립트나 응답 없는 명령어를 종료하는 데 매우 유용합니다.
- **방향키 (위/아래)**: 이전에 입력했던 명령어 히스토리를 탐색할 수 있습니다. 반복적으로 동일한 명령어를 입력할 때 시간을 절약할 수 있습니다.

# Chapter 4: 시스템 보안 및 서비스 제어

## 4.1 시스템 관리자 권한 이해: `sudo`

리눅스 시스템에서 일반 사용자가 관리자(root) 권한이 필요한 작업을 수행해야 할 때 사용하는 가장 일반적인 명령어가 바로 `sudo`입니다. `sudo`는 "superuser do"의 약자로, 현재 사용자가 일시적으로 root 권한을 빌려 특정 명령어를 실행할 수 있도록 해줍니다.

- **기능**: `sudo [명령어]` 형태로 사용하며, 뒤에 오는 명령어를 root 권한으로 실행합니다.
- **장점**:
  - **보안성 강화**: 항상 root 계정으로 로그인하여 작업하는 것은 매우 위험합니다. `sudo`를 사용하면 필요한 경우에만 관리자 권한을 사용하여 불필요한 위험을 줄일 수 있습니다.
  - **책임성**: `sudo`를 통해 실행된 명령어는 누가 어떤 명령어를 실행했는지 로그를 남기므로, 시스템 관리의 투명성을 높입니다.
  - **편의성**: root 비밀번호를 공유하지 않고도 특정 사용자에게 관리 권한을 부여할 수 있습니다.
- **실무 포인트**: `/etc/sudoers` 파일을 통해 어떤 사용자가 어떤 명령어를 `sudo`로 실행할 수 있는지 상세하게 설정할 수 있습니다. 이 파일은 반드시 `visudo` 명령어를 통해 편집해야 문법 오류로 인한 시스템 접근 불가 사태를 방지할 수 있습니다.

## 4.2 리눅스 서비스 제어: `systemctl`

현대의 대부분의 리눅스 배포판(CentOS/RHEL 7+, Ubuntu 15.04+ 등)에서는 `systemd`라는 시스템 및 서비스 관리자를 사용하며, `systemctl`은 이 `systemd`를 제어하는 핵심 유틸리티입니다. `systemctl`은 서비스(데몬), 소켓, 마운트 지점 등 다양한 시스템 유닛을 관리할 수 있습니다.

- **`systemd`란?**: 리눅스 커널 부팅 후 가장 먼저 실행되는 프로세스(PID 1)인 `init` 시스템을 대체하는 것으로, 시스템 부팅 프로세스 및 실행 중인 서비스(데몬)를 효율적으로 관리합니다.
- **`d`의 의미**: 많은 리눅스 서비스 이름 끝에 붙는 `d`(예: `firewalld`)는 'daemon'의 약자입니다. 데몬은 백그라운드에서 계속 실행되면서 특정 작업을 수행하는 프로그램입니다.
- **주요 `systemctl` 명령어**:
  - `systemctl start [서비스명]`: 서비스 시작. 현재 세션에서만 적용되며, 재부팅 시 다시 꺼질 수 있습니다.
  - `systemctl stop [서비스명]`: 서비스 중지.
  - `systemctl restart [서비스명]`: 서비스 재시작.
  - `systemctl status [서비스명]`: 서비스의 현재 상태(활성화 여부, 실행 중인지, 오류 발생 여부 등)를 확인합니다.
  - `systemctl enable [서비스명]`: 서비스 자동 시작 활성화. 시스템 부팅 시 해당 서비스가 자동으로 시작되도록 설정합니다. 이 설정은 재부팅 후에도 유지됩니다.
  - `systemctl disable [서비스명]`: 서비스 자동 시작 비활성화. 시스템 부팅 시 서비스가 자동으로 시작되지 않도록 설정합니다.
- **`stop` vs `disable`**:
  - `stop`은 현재 실행 중인 서비스를 멈추는 것이고, `disable`은 서비스가 부팅 시 자동으로 시작되지 않도록 설정하는 것입니다. 예를 들어, `systemctl stop httpd`는 웹 서버를 즉시 멈추지만, 시스템을 재부팅하면 `httpd`가 `enable`되어 있다면 다시 시작됩니다. `systemctl disable httpd`를 사용하면 다음 부팅 시 웹 서버가 시작되지 않습니다.
- **실무 포인트**: 서비스 문제 발생 시 `systemctl status [서비스명]`을 통해 상태를 확인하는 것이 문제 해결의 첫걸음입니다. 로그 경로도 함께 표시되므로 유용합니다.

## 4.3 리눅스 보안 모듈: SELinux

SELinux(Security-Enhanced Linux)는 미국 국가안보국(NSA)이 개발한 리눅스 커널의 보안 모듈로, 기존의 DAC(Discretionary Access Control, 소유자/그룹/기타 사용자 권한) 방식으로는 불가능했던 강력한 강제적 접근 제어(MAC, Mandatory Access Control) 기능을 제공합니다. 이는 시스템 보안을 한층 강화하여, 프로세스가 접근할 수 있는 자원을 세밀하게 통제합니다.

- **기능**: 모든 파일, 프로세스, 포트 등에 보안 컨텍스트(Security Context)를 부여하고, SELinux 정책에 따라 접근을 허용하거나 차단합니다. 이는 해커가 시스템의 취약점을 통해 특정 프로세스를 장악하더라도, 해당 프로세스의 활동을 제한하여 시스템 전체에 미치는 피해를 최소화하는 데 도움을 줍니다.
- **설정 파일**: `/etc/selinux/config` 파일에서 SELinux의 작동 방식을 설정합니다.
  - `SELINUX=enforcing`: 정책에 따라 접근을 강제 적용하고 차단합니다.
  - `SELINUX=permissive`: 정책 위반 시 경고만 기록하고 실제 차단은 하지 않습니다. (디버깅 모드로 활용)
  - `SELINUX=disabled`: SELinux 기능을 완전히 비활성화합니다.
- **비활성화 및 주의사항**:
  - SELinux는 강력한 보안 기능을 제공하지만, 잘못 설정될 경우 서비스 실행에 방해가 될 수 있습니다. 특히 특정 서비스를 처음 설정할 때 예기

치 않은 문제가 발생하면, 임시로 `SELINUX=permissive` 또는 `SELINUX=disabled`로 설정하여 SELinux가 문제의 원인인지 확인하기도 합니다.
    - `/etc/selinux/config` 파일을 수정한 후에는 **시스템을 재부팅해야만 변경 사항이 적용됩니다.** `source` 명령으로는 적용되지 않습니다. 이는 SELinux가 커널 레벨에서 작동하는 모듈이기 때문입니다.
    - **시험/실무 포인트**: 시스템 보안을 위해서는 SELinux를 활성화(`enforcing`) 상태로 유지하는 것이 권장됩니다. 단순히 비활성화하는 것보다, 문제가 되는 서비스를 SELinux 정책에 맞게 구성하거나 필요한 예외 규칙을 추가하는 방법을 학습하는 것이 중요합니다.

### 4.4 동적 방화벽 관리: `firewalld`

`firewalld`는 CentOS/RHEL 7 이상 버전에서 기본으로 제공되는 동적 방화벽 관리 도구입니다. 기존의 `iptables`와는 달리, 서비스 재시작 없이도 방화벽 규칙을 동적으로 적용할 수 있어 관리의 편의성과 유연성을 크게 향상시켰습니다.

- **동적 방화벽**:
    - `firewalld`의 가장 큰 특징은 '동적'이라는 점입니다. `iptables`는 규칙을 변경하면 전체 방화벽을 재시작해야 했기 때문에 잠시 동안 네트워크 연결이 끊길 수 있었습니다. 반면 `firewalld`는 개별 규칙만 변경하고 즉시 적용할 수 있어 서비스 중단 없이 방화벽을 관리할 수 있습니다.
- **존(Zone) 기반 관리**:
    - `firewalld`는 네트워크 인터페이스를 '존(Zone)'이라는 개념으로 분류하여 관리합니다. 각 존에는 사전 정의된 보안 수준이 있으며, 관리자는 특정 네트워크 인터페이스를 적절한 존에 할당함으로써 보안 정책을 쉽게 적용할 수 있습니다. (예: public, trusted, home 등)
- **`systemctl`과의 연동**:
    - `firewalld` 역시 `d`가 붙은 이름에서 알 수 있듯이 데몬(daemon)입니다. 따라서 `systemctl` 명령어를 사용하여 `firewalld` 서비스를 시작, 중지, 재시작, 상태 확인 및 부팅 시 자동 실행 여부를 제어합니다.
    - 예시: `systemctl status firewalld`, `systemctl enable firewalld`
- **실무 포인트**: `firewall-cmd` 명령어를 사용하여 포트 개방, 서비스 허용, 존 변경 등 `firewalld`의 상세한 규칙을 설정합니다. 설정 변경 후에는 `firewall-cmd --runtime-to-permanent` 또는 `firewall-cmd --reload` 명령어를 통해 변경 사항을 영구적으로 저장하거나 즉시 적용해야 합니다.

# Chapter 5: 대화형 개발 환경 Jupyter Notebook

## Jupyter Notebook 소개

Jupyter Notebook은 웹 브라우저를 통해 파이썬 코드를 작성하고, 실행하며, 그 결과를 실시간으로 확인하고 문서화할 수 있는 강력한 대화형 개발 환경(IDE)입니다. 코드 셀, 마크다운 셀 등을 조합하여 코드와 설명을 한 문서에 담을 수 있어 데이터 분석, 머신러닝 모델 개발, 교육 자료 작성 등 다양한 분야에서 널리 활용됩니다.

Jupyter Notebook의 가장 큰 특징은 '대화형'이라는 점입니다. 각 코드 셀을 독립적으로 실행할 수 있으며, 먼저 실행된 셀에서 정의된 변수나 함수는 이후에 실행되는 셀에서도 접근할 수 있습니다. 이는 마치 파이썬 인터프리터를 사용하는 것처럼 실시간으로 코드를 실험하고 결과를 확인할 수 있게 해줍니다.

## Jupyter Notebook 환경 구성 (Miniconda 활용)

Jupyter Notebook은 일반적으로 파이썬 패키지 매니저인 `pip`을 통해 설치할 수 있지만, 파이썬과 함께 자주 사용되는 핵심 패키지들을 미리 포함하고 있는 Anaconda나 그 경량화 버전인 Miniconda를 통해 설치하는 것이 편리하고 권장됩니다. Miniconda는 Anaconda의 핵심 기능만 포함하고 있어 필요한 패키지만 설치하여 시스템 자원을 효율적으로 사용할 수 있습니다.

### 1. Miniconda 설치 스크립트 다운로드 및 실행

- 웹에서 Miniconda 설치 스크립트를 내려받기 위해 `wget` 유틸리티를 사용합니다.
- `wget [Miniconda 설치 스크립트 URL]`
- 다운로드된 `.sh` 확장자를 가진 쉘 스크립트 파일을 현재 디렉터리에서 실행합니다.
- `./Miniconda3-latest-Linux-x86_64.sh` (파일명은 버전에 따라 다를 수 있습니다.)

### 2. 환경 변수 PATH 설정

Miniconda를 설치하면 파이썬 실행 파일 및 `conda`, `jupyter` 등의 명령어가 설치된 경로가 생깁니다. 이 경로를 시스템의 환경 변수 `PATH`에 추가해야 터미널 어디에서든 해당 명령어들을 바로 사용할 수 있습니다.

- **`vi` 또는 `vim`을 이용한 설정 파일 편집:**

  환경 변수 `PATH`를 영구적으로 설정하려면 쉘의 설정 파일(예: `/etc/bashrc` 또는 사용자 홈 디렉토리의 `~/.bashrc`)을 수정해야 합니다. `vi`나 `vim`은 강력한 CLI(Command Line Interface) 텍스트 에디터로, 리눅스 서버 환경에서 설정 파일을 편집할 때 필수적으로 사용됩니다.

  ```
  sudo vi /etc/bashrc
  ```

  또는

  ```
  vi ~/.bashrc
  ```

  `vi` 편집기 사용법:

  - `i` 또는 `a`를 눌러 **편집(Insert) 모드**로 전환합니다.
  - 파일의 끝에 다음 줄을 추가합니다 (경로는 Miniconda 설치 경로에 따라 다를 수 있습니다).

    ```
    export PATH=$PATH:/etc/miniconda3/bin/
    ```

  - `[ESC]`를 눌러 **명령(Command) 모드**로 돌아갑니다.
  - `:wq`를 입력하여 파일을 저장하고 종료합니다. (`w`: write, `q`: quit)
  - `:q!`는 변경 사항을 저장하지 않고 종료할 때 사용합니다.

- **변경 사항 적용:**

  설정 파일을 수정한 후에는 해당 쉘 세션에 변경 사항을 적용해야 합니다. `source` 명령어를 사용하면 됩니다. 그렇지 않으면 로그아웃 후 다시 로그인해야 변경 사항이 적용됩니다.

  ```
  source /etc/bashrc
  ```

  또는

  ```
  source ~/.bashrc
  ```

### Jupyter Notebook 실행 및 접속

환경 변수 설정이 완료되면 `jupyter notebook` 명령어로 Jupyter 서버를 실행할 수 있습니다. 특히 원격 서버에서 실행하여 웹 브라우저를 통해 접속할 경우 몇 가지 옵션을 지정하는 것이 일반적입니다.

- **Jupyter 서버 실행 명령어:**

```
jupyter notebook --allow-root --ip=0.0.0.0 --port=8888
```

  - `--allow-root`: `root`(관리자) 권한으로 Jupyter를 실행할 때 필요한 옵션입니다. 보안상 일반 사용자 계정으로 실행하는 것이 더 안전하지만, 특정 서버 환경에서는 `root`로 실행해야 할 수도 있습니다.
  - `--ip=[아이피 주소]`: Jupyter 서버가 수신할 IP 주소를 지정합니다. `0.0.0.0`으로 설정하면 모든 네트워크 인터페이스(즉, 외부 IP 주소를 포함하여)에서 접속을 허용합니다. 특정 IP만 허용하려면 해당 IP 주소를 입력합니다.
  - `--port=[포트 번호]`: Jupyter 서버가 사용할 포트 번호를 지정합니다. 기본값은 8888이지만, 다른 서비스와 충돌하거나 특정 포트를 사용해야 할 경우 변경할 수 있습니다.
  - *참고:* 원격 서버에서 실행할 경우, 웹 브라우저를 자동으로 열지 않도록 `--no-browser` 옵션을 추가할 수 있습니다.

- **웹 브라우저로 접속:**

  Jupyter 서버가 실행되면, 웹 브라우저를 열고 다음 URL로 접속합니다.

```
http://[서버 IP 주소]:[지정한 포트 번호]
```

  예시: `http://192.168.1.100:8888`

### 인증 방식 설정

Jupyter Notebook은 기본적으로 토큰 인증 방식을 사용합니다. 서버를 처음 실행하면 터미널에 출력되는 URL에 포함된 토큰을 복사하여 웹 브라우저의 로그인 페이지에 붙여넣어야 합니다. 반복적으로 접속할 때마다 이 토큰을 찾는 것이 번거로울 수 있으므로, 비밀번호 인증 방식으로 변경하는 것이 편리합니다.

- **비밀번호 설정 명령어:**

```
jupyter notebook password
```

  이 명령어를 실행하면 새 비밀번호를 두 번 입력하게 됩니다. 비밀번호 설정 후에는 Jupyter 서버를 다시 시작해야 변경 사항이 적용될 수 있습니다.

### Jupyter Notebook의 핵심 개념: 셀(Cell)

Jupyter Notebook은 '셀(Cell)' 단위로 구성됩니다. 각 셀은 독립적인 실행 단위이며, 크게 코드 셀과 마크다운 셀로 나뉩니다.

- **코드 셀:** 파이썬 코드를 작성하고 실행하는 공간입니다. 셀을 실행하면 그 결과(출력, 에러 메시지 등)가 셀 바로 아래에 표시됩니다.
- **마크다운 셀:** 텍스트, 제목, 목록, 이미지 등을 마크다운 문법으로 작성하여 코드에 대한 설명이나 분석 결과를 문서화하는 데 사용됩니다.

가장 중요한 특징은 **"먼저 실행한 셀의 변수 등의 값은 다음에 실행하는 셀에서 접근 가능하다"**는 것입니다. 이는 Jupyter Notebook이 내부적으로 파이썬 커널을 유지하며 세션의 상태(변수 값, 함수 정의 등)를 공유하기 때문입니다. 이러한 특성 덕분에 대화형으로 코드를 탐색하고 점진적으로 개발해 나갈 수 있습니다.

*실무 팁:* 셀의 실행 순서가 중요합니다. 특정 셀이 의존하는 변수가 정의된 셀이 먼저 실행되지 않으면 오류가 발생할 수 있습니다. `런타임` 메뉴에서 `모든 셀 실행` 또는 `선택한 셀까지 실행` 등의 기능을 활용하여 순서를 관리하는 것이 좋습니다.

### Jupyter Notebook 원격 접속 시 고려사항 (Linux 서버 환경)

Jupyter Notebook을 원격 Linux 서버에서 실행하고 로컬 웹 브라우저로 접속하려면 서버의 보안 설정(방화벽, SELinux)을 확인하고 필요에 따라 조정해야 합니다.

#### 1. 방화벽 설정 (firewalld)

`firewalld`는 CentOS/RHEL 7 이상에서 사용되는 동적 방화벽 관리 도구입니다. Jupyter Notebook이 사용하는 포트(기본 8888)가 방화벽에 의해 차단되어 있지 않은지 확인해야 합니다.

- **방화벽 상태 확인:**

```
systemctl status firewalld
```

- **방화벽 서비스 시작/중지/재시작:**

```
systemctl start firewalld
systemctl stop firewalld   # 임시 중지 (재부팅 시 다시 켜짐)
systemctl restart firewalld
```

- **부팅 시 자동 실행 설정/비활성화:**

```
systemctl enable firewalld   # 부팅 시 자동 실행 활성화
systemctl disable firewalld  # 부팅 시 자동 실행 비활성화
```

- **특정 포트 영구적으로 열기 (권장):**

  방화벽을 완전히 끄는 대신, Jupyter가 사용하는 포트만 열어주는 것이 보안상 더 좋습니다.

```
sudo firewall-cmd --permanent --add-port=[포트번호]/tcp
sudo firewall-cmd --reload
```

  [포트번호] 부분에 Jupyter가 사용하는 포트(예: 8888)를 입력합니다.

#### 2. SELinux 설정 (Security-Enhanced Linux)

SELinux는 리눅스 커널에 통합된 보안 모듈로, 강제 접근 제어(MAC) 기능을 제공합니다. SELinux가 활성화되어 있으면 Jupyter Notebook의 특정 동작이나 포트 사용이 제한될 수 있습니다.

- **SELinux 설정 파일 확인:**

```
cat /etc/selinux/config
```

- **SELinux 비활성화:**

  `/etc/selinux/config` 파일에서 `SELINUX=enforcing` 부분을 찾아 `SELINUX=disabled`로 변경합니다.

```
sudo vi /etc/selinux/config
```

  파일 수정 후에는 `source` 명령어로 적용되지 않으며, **시스템을 재부팅해야 변경 사항이 적용됩니다.**

  *주의:* SELinux는 시스템 보안을 강화하는 중요한 기능이므로, 특별한 이유 없이는 비활성화하지 않는 것이 좋습니다. 대신, 특정 서비스에 대한 정책을 추가하는 것이 보안상 더 바람직합니다.

#### 3. 권한 문제 (sudo, chmod)

리눅스 환경에서 파일을 다운로드하거나 특정 디렉토리에 접근할 때 권한 문제가 발생할 수 있습니다.

- `sudo`: `superuser do`의 약자로, 관리자(root) 권한으로 뒤따르는 명령어를 실행합니다. 중요한 시스템 파일 수정이나 패키지 설치 시 주로 사용됩니다.
- `chmod`: 파일 또는 디렉토리의 권한을 수정하는 명령어입니다. 예를 들어, 쉘 스크립트 파일을 실행 가능하게 만들려면 다음과 같이 권한을 부여할 수 있습니다.

```
chmod u+x [파일명]
```

  (현재 사용자(u)에게 실행(x) 권한을 추가(+x)합니다.)

### 기타 유용한 리눅스 명령어 (Jupyter 환경 관리 시)

Jupyter Notebook 환경을 설정하고 관리할 때 유용하게 사용될 수 있는 몇 가지 리눅스 기본 명령어가 있습니다.

- `cat [파일명]`: 텍스트 파일의 내용을 화면에 출력합니다. 설정 파일이나 스크립트 내용을 빠르게 확인할 때 유용합니다.
- `pwd`: `print working directory`의 약자로, 현재 작업 중인 폴더의 절대 경로를 출력합니다.
- `ls`: 현재 디렉토리의 파일 및 디렉토리 목록을 표시합니다.

    - `ls -a`: 숨김 파일을 포함하여 모든 파일 및 디렉토리를 표시합니다.
    - `ls -l`: 파일 및 디렉토리의 상세 정보(권한, 소유자, 크기, 수정일 등)를 출력합니다.
    - `ls -al` (또는 `ll`): 숨김 파일을 포함한 상세 목록을 출력합니다.

  `ls -al`의 출력 형식: `(권한/파일 수/소유자/그룹/파일 크기/수정일자/파일이름)`

- `Control+C (^C)`: 터미널에서 현재 실행 중인 프로세스(예: Jupyter 서버)를 강제로 종료합니다.
- 쉘 히스토리: 터미널에서 위/아래 방향키를 누르면 이전에 입력했던 명령어들을 순서대로 다시 불러올 수 있습니다.