File Sharing

FTP

- FTP
  - ✓ 개요
    - FTP(File Transfer Protocol)는 네트워크를 통해 컴퓨터 간에 파일을 전송하기 위한 가장 기본적인 표준 통신 규약
    - 내 컴퓨터(클라이언트)와 멀리 떨어진 컴퓨터(서버) 사이에 파일 전송용 고속도로를 놓는 것과 같음
    - 일반적인 통신과 달리 FTP는 두 개의 연결(포트)을 동시에 사용한다는 점이 독특함
  - ✓ FTP의 핵심 작동 원리: 두 개의 채널
    - 제어 채널(Control Channel / 포트 21): 사용자 아이디, 비밀번호 확인 및 파일을 올려라, 파일 리스트를 보여달라 같은 명령어를 전달
    - 데이터 채널(Data Channel / 포트 20): 실제 파일 데이터가 오가는 통로로 제어 채널에서 명령이 떨어지면 이 통로를 통해 실제 파일 전송이 이루어 짐
  - ✓ 접속 방식(전송 모드): FTP는 네트워크 환경(특히 방화벽)에 따라 두 가지 방식으로 작동
    - 능동 모드(Active): 서버가 클라이언트에게 접속을 시도하는데 클라이언트 PC의 방화벽이 외부 접속을 막으면 전송이 실패할 수 있음
    - 수동 모드(Passive): 클라이언트가 서버에 접속을 시도하는데 대부분의 웹 브라우저나 공유기 환경에서 권장되는 방식(더 안정적)

FTP

- FTP
  - ✓ FTP의 종류(보안성 강화)
    - 기본 FTP는 아이디와 비밀번호가 암호화되지 않은 채로 전송되어 해킹에 취약
    - SFTP (Secure FTP): SSH(Secure Shell) 기술을 사용하여 전송 과정 전체를 암호화하는 방식으로 현재 실무에서 가장 많이 쓰이는 방식
    - FTPS (FTP over SSL/TLS): 우리가 웹사이트 주소 앞에 붙는 HTTPS처럼 SSL 인증서를 사용하여 암호화하는 방식
  - ✓ 장단점 요약
    - 장점: 대용량 파일을 일괄 전송하기에 매우 효율적이며 전송 속도가 빠름
    - 단점: 보안 설정을 하지 않으면 정보 유출 위험이 크고 설정이 복잡할 수 있음

FTP

- vsftpd
  - ✓ 개요
    - vsftpd(Very Secure FTPD)는 우분투에서 기본적으로 제공되며 리눅스와 유닉스 환경에서 보안성과 성능이 우수한 FTP 서버로 인정받고 있음
    - vsftpd는 설치 및 운영이 쉬우므로 리눅스 환경에서 FTP 서버를 운영하는 데 많이 이용
    - 이와 관련된 내용이나 소스 파일의 다운로드 https://security.appspot.com/vsftpd.html 에서 할 수 있음
  - ✓ 설치
    - sudo apt -y install vsftpd

FTP

- vsftpd
  - ✓ 설정 파일: /etc/vsftpd.conf 파일
    - anonymous_enable: anonymous 사용자의 접속을 허가할지 설정
    - local_enable: 로컬 사용자의 접속 허가 여부 설정
    - wnte_enable: 로컬 사용자가 저장 삭제 디렉터리 생성 등의 명령을 실행하게 할지 설정(anonymous 사용자는 해당 없음)
    - anon_upload_enable: anonymous 사용자의 파일 업로드 허가 여부 설정
    - anon_mkdir_write_eanble: anonymous 사용자의 디렉터리 생성 허가 여부 설정
    - diriist_enable: 접속한 디렉터리의 파일 리스트를 보여줄지 설정
    - download_enable: 다운로드의 허가 여부 설정
    - listen_port: FTP 서비스의 포트 번호 설정(기본 21번)
    - deny_file: 업로드 금지 파일 지정(예: deny_file={*.mpg.*.mpeg,*.avi})
    - hide_file: 보여주지 않을 파일을 지정(예: hide_file={*.gif.*.jpg,*.png})
    - max_clients: FTP 서버의 동시 최대 접속자 수 지정
    - max_per_ip: 1개의 PC가 동시에 접속할 수 있는 접속자 수 지정

FTP

- vsftpd
  - ✓ 설정 변경
    - /etc/vsftpd.conf 파일 수정
      - 익명 사용자 허가: anonymous_enable=YES
      - 로컬 사용자가 저장, 삭제, 디렉토리 생성: write_enable=YES
      - 익명 사용자의 업로드 허가 여부: anon_upload_enable=YES
      - 익명 사용자의 디렉토리 생성 허가 여부: anon_mkdir_write_enable=YES
    - 디렉토리 생성 및 접근 권한 변경
      - 익명 사용자는 사용자 이름이 anonymous 지만 리눅스 내부에서는 ftp
      - 익명 사용자의 홈 디렉토리로 이동: cd /srv/ftp
      - sudo mkdir pub
      - chmod 777 pub
      - cd pub
      - touch file1
      - ls –l
    - 서비스 재시작: sudo systemctl restart vsftpd

FTP

- ❖ vsftpd
  - ✓ 클라이언트에서 접속
    - ● ubuntu 터미널에서 작업
      - ◆ 접속: ftp 192.168.0.100

```
Connected to 192.168.0.100.
220 (vsFTPd 3.0.5)
Name (192.168.0.100:adam): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

FTP

- ❖ vsftpd
  - ✓ 클라이언트에서 접속
    - ● ubuntu 터미널에서 작업
      - ◆ 디렉토리 확인: ls

        229 Entering Extended Passive Mode (|||17382|)

        150 Here comes the directory listing.

        drwxrwxrwx 2 0 0 4096 May 05 22:45 pub

        226 Directory send OK.
      - ◆ 디렉토리 이동: cd pub

FTP

- vsftpd
  - ✓ 클라이언트에서 접속
    - ● ubuntu 터미널에서 작업
      - ◆ 파일 다운로드: get file1

        local: file1 remote: file1

        229 Entering Extended Passive Mode (|||50001|)

        150 Opening BINARY mode data connection for file1 (0 bytes).

        0 0.00 KiB/s

        226 Transfer complete.
      - ◆ 파일 업로드: put file2

        local: file2 remote: file2

        229 Entering Extended Passive Mode (|||50005|)

        150 Ok to send data.

        0 0.00 KiB/s

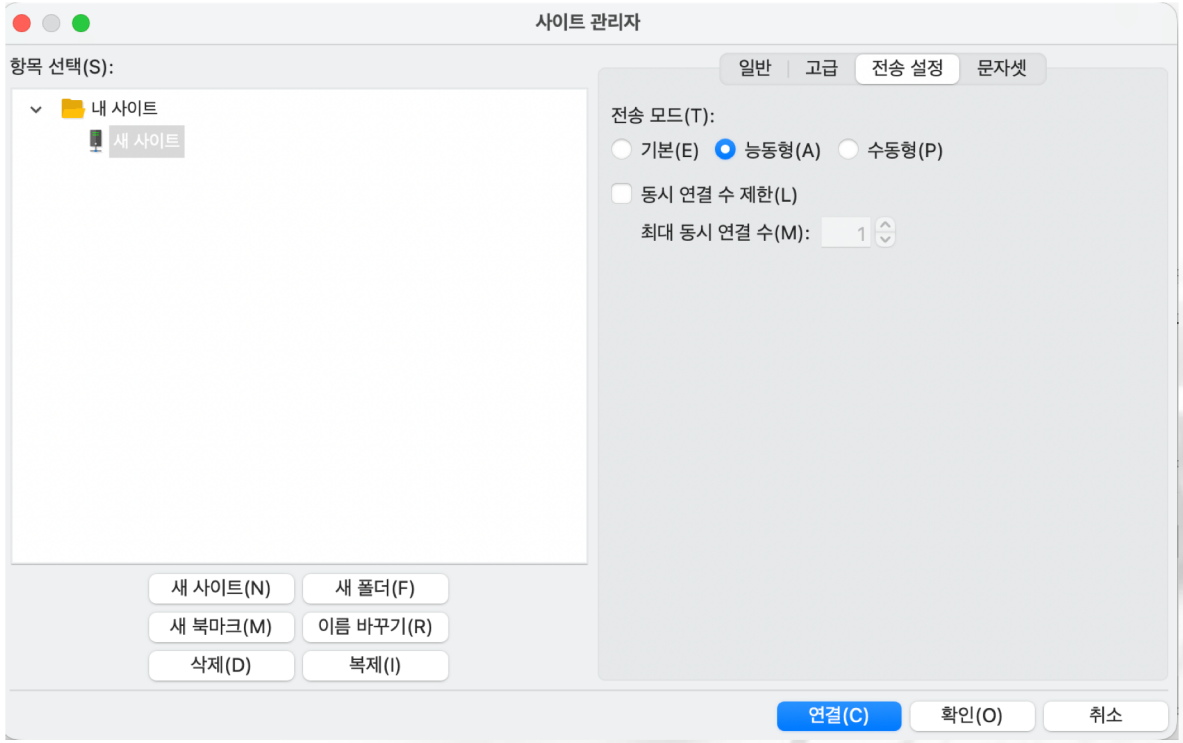
        226 Transfer complete.

FTP

- vsftpd
  - ✓ 클라이언트에서 접속
    - Filezilla에서 접속
      - 기본적으로 Active 모드로 접속해야 하므로 Filezilla 설정 변경: [파일] > [사이트 관리자]

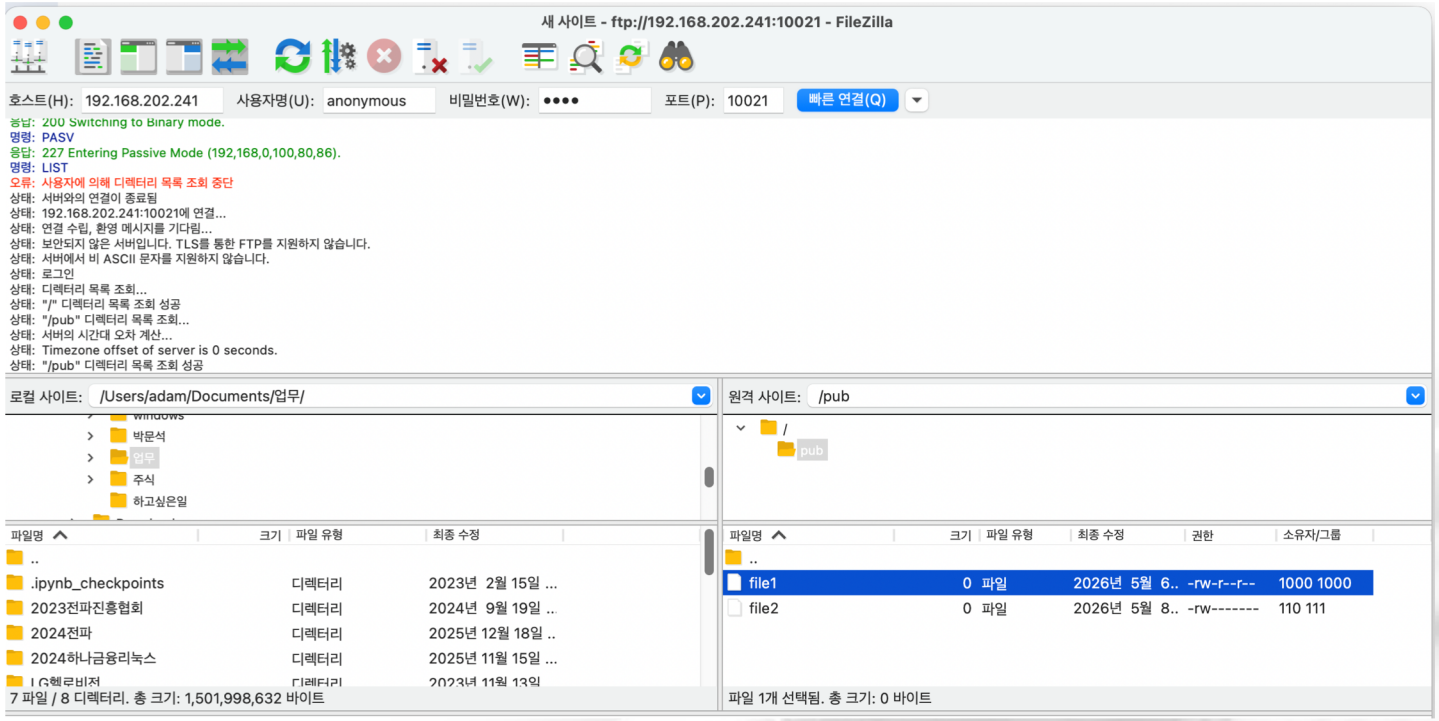
FTP

- vsftpd
  - ✓ 클라이언트에서 접속
    - Filezilla에서 접속
      - 기본적으로 Active 모드로 접속해야 하므로 Filezilla 설정 변경: [파일] > [사이트 관리자]



FTP

- ❖ vsftpd
  - ✓ 클라이언트에서 접속
    - ● Filezilla에서 접속



NFS

- NFS
  - ✓ 개요
    - NFS(Network File System)는 네트워크상에 있는 다른 컴퓨터의 파일 시스템을 마치 자신의 로컬 디스크에 있는 것처럼 사용할 수 있게 해주는 클라이언트/서버 프로토콜
    - 1984년 선 마이크로시스템즈(Sun Microsystems)에서 개발했으며 현재 유닉스 및 리눅스 환경에서 표준 파일 공유 프로토콜로 널리 사용
  - ✓ 주요 특징
    - 투명성(Transparency): 사용자는 파일이 원격 서버에 있는지 로컬 하드디스크에 있는지 구분하지 않고 동일한 명령어로 파일을 읽고 쓸 수 있음
    - 중앙 집중 관리: 데이터를 서버 한 곳에 모아 관리하므로 백업이나 보안 정책 적용이 효율적
    - 플랫폼 독립성: RPC(Remote Procedure Call) 메커니즘을 사용하여 서로 다른 운영체제 간에도 파일 공유가 가능

NFS

- NFS
  - ✓ 작동 원리
    - Export (서버): 서버는 자신의 특정 디렉토리를 네트워크에 공유(Export)하도록 설정
    - Mount (클라이언트): 클라이언트는 서버가 공유한 디렉토리를 자신의 로컬 디렉토리(마운트 포인트)에 연결
    - RPC 호출: 클라이언트에서 파일 접근 요청이 발생하면 NFS 프로토콜이 이를 RPC 메시지로 변환하여 서버에 전달하고 결과를 받아옴
  - ✓ 장점과 단점
    - 장점
      - 비용 절감: 각 워크스테이션마다 대용량 디스크를 설치할 필요가 없음
      - 편의성: 사용자 홈 디렉토리를 NFS로 설정하면 어떤 컴퓨터에서 로그인하든 동일한 작업 환경을 유지할 수 있음
    - 단점
      - 네트워크 의존성: 네트워크 장애 발생 시 파일 접근이 불가능해지며 네트워크 속도에 따라 성능 편차가 큼
      - 보안 취약성: 초기 버전(v2, v3)은 호스트 IP 기반의 인증 방식을 주로 사용하여 보안에 취약할 수 있으므로 주의가 필요

NFS

- NFS
  - ✓ 주요 활용 사례
    - 클러스터 서버: 여러 대의 웹 서버나 애플리케이션 서버가 동일한 데이터(이미지, 설정 파일 등)를 공유해야 할 때 사용
    - 데이터 백업: 여러 호스트의 데이터를 중앙 스토리지 서버로 모으는 용도로 활용
    - Kubernetes (K8s): 컨테이너 환경에서 영구 볼륨(Persistent Volume)으로 NFS를 연결하여 데이터의 휘발성을 방지
  - ✓ 구현 순서
    - NFS 서버에 관련 패키지를 설치
    - NFS 서버의 /etc/exports에 공유할 디렉터리, 접근을 허가할 컴퓨터. 접근 권한을 지정
    - NFS 서비스를 실행
    - NFS 클라이언트에 관련 패키지를 설치
    - NFS 클라이언트에 showmount 명령을 실행해 NFS 서버에 공유된 디렉토리가 있는지 확인
    - NFS 클라이언트에서 mount 명령을 실행해 NFS 서버에 공유된 디렉토리를 마운트

NFS

- NFS 구현
  - ✓ 서버
    - 패키지 설치: sudo apt update && sudo apt install nfs-kernel-server
    - /share 디렉토리 생성: sudo mkdir -p /share && sudo chmod 777 /share
    - /share 디렉토리에 file 생성: sudo touch /share/file1
    - 설정 파일에 추가: sudo nano /etc/exports

      /share 192.168.0.0/24(rw,sync,no_subtree_check,no_root_squash,insecure)
    - 설정 반영: sudo exportfs -rav
    - nfs-server 부팅 시 바로 시작: sudo systemctl enable nfs-server
    - 방화벽 설정: sudo ufw allow from 192.168.0.0/24
    - 서비스 확인: sudo systemctl status nfs-kernel-server

NFS

- NFS 구현
  - ✓ 클라이언트
    - 패키지 설치: sudo apt -y install nfs-common
    - NFS 서버에 공유된 디렉토리 확인: showmount -e 192.168.0.100

      Export list for 192.168.0.100:

      /share 192.168.0.0/24
    - 마운트 할 디렉토리 생성: ~/myshare
    - 마운트: sudo mount -t nfs -o nfsvers=3 192.168.0.100:/share ~/myshare
    - 마운트 확인: ls ~/myshare

      file1

SSH를 이용한 파일 전송

- 개요
  - ✓ SCP와 SFTP 두 가지 방법을 주로 사용
  - ✓ 사용하는 이유
    - 강력한 데이터 암호화(Security)
      - 가장 결정적인 이유로 일반적인 FTP는 아이디, 비밀번호, 그리고 전송되는 파일 데이터를 평문(Plain Text) 상태로 주고받는데 만약 중간에 누군가 패킷을 가로챈다면 모든 정보를 그대로 읽을 수 있음
      - 암호화 전송: SSH는 전송되는 모든 데이터를 암호화하므로 중간에 데이터가 유출되더라도 내용을 해독할 수 없음
      - 공격 방지: 중간자 공격(Man-in-the-Middle)과 같은 네트워크 가로채기 공격으로부터 안전
    - 단일 포트 사용의 편의성(Simplicity)
      - 전통적인 FTP는 제어용 포트(21번)와 데이터 전송용 포트(20번 등)를 따로 사용하며 패시브 모드 설정 시 수많은 포트를 개방해야 하는 번거로움이 있음
      - 22번 포트 하나로 해결: SSH는 기본적으로 22번 포트 하나만 방화벽에서 허용하면 원격 접속(CLI)과 파일 전송을 모두 처리할 수 있으므로 관리 효율성이 매우 높음

SSH를 이용한 파일 전송

- ❖ 개요
  - ✓ 사용하는 이유
    - 별도의 서버 설치 불필요(No Extra Setup)
      - ◆ 대부분의 리눅스/유닉스 계열 서버(Ubuntu, CentOS 등)와 macOS에는 SSH 서비스가 기본적으로 설치되어 실행
      - ◆ 파일 전송을 위해 별도의 FTP 서버(vsftpd, ProFTPD 등)를 설치하고 설정할 필요가 없는데 서버 계정만 있다면 즉시 파일을 주고받을 수 있음
    - 데이터 무결성 검증(Integrity)
      - ◆ 전송 중에 데이터가 손실되거나 변조되지 않았는지 확인하는 기능이 포함되어 있음
      - ◆ 파일 전송이 완료되면 SSH 프로토콜 내부적으로 데이터가 원본과 일치하는지 검증하므로 파일이 깨진 상태로 전송되는 것을 방지
    - 인증 방식의 다양성(Authentication)
      - ◆ 비밀번호 방식 외에도 훨씬 안전한 SSH Key(공개키/개인키) 방식을 사용할 수 있음
      - ◆ 키 기반 인증을 사용하면 비밀번호 무차별 대입 공격(Brute-force)을 원천 차단할 수 있으며 자동화된 스크립트나 CI/CD 파이프라인에서 안전하게 파일을 배포할 수 있음

SSH를 이용한 파일 전송

- 방법
  - ✓ SCP(Secure Copy): 가장 빠르고 간단하게 파일을 복사할 때 사용하는데 명령어 한 줄로 전송이 끝나는 방식
    - 로컬에서 원격지로 파일 보내기
      - scp [파일명] [계정]@[IP주소]:[저장경로]
      - 예: scp test.txt adam@192.168.0.100:/home/adam/test
    - 원격지에서 로컬로 파일 보내기
      - scp [계정]@[IP주소]:[파일경로] [로컬경로]
      - 예: scp adam@192.168.0.100:/home/adam/test/sample.txt ./
      - 디렉토리 전체를 전송할 때는 –r 옵

SSH를 이용한 파일 전송

- 방법
  - ✓ SFTP(Secure File Transfer Protocol): SSH 위에서 동작하는 FTP 방식으로, 접속을 유지한 상태에서 인터랙티브하게 파일을 관리할 수 있는데 파일 목록 확인이나 삭제 등이 가능
    - 접속: sftp [계정]@[IP주소]
    - 명령어
      - ls: 원격 서버 파일 목록 보기
      - put [파일명]: 파일을 서버로 업로드
      - get [파일명]: 서버에서 파일을 다운로드
      - exit: 접속 종료

SSH를 이용한 파일 전송

- pem 파일을 이용한 인증
  - ✓ pem 파일 생성: ssh-keygen -t rsa -b 4096 -m PEM -f 파일경로
  - ✓ 파일 소유자에게만 권한 부여: chmod 400 파일경로
  - ✓ SCP(Secure Copy)
    - 로컬 → 원격지로 파일 보내기
      - scp -i your-key.pem [파일명] [계정]@[IP주소]:[저장경로]
    - 원격지에서 로컬로 파일 보내기
      - scp -i your-key.pem [계정]@[IP주소]:[파일경로] [로컬경로]
  - ✓ SFTP를 이용한 전송
      - 대화형 모드로 파일을 관리하고 싶을 때 사용
      - sftp -i your-key.pem [계정]@[IP주소]

Samba Server

- Samba
  - ✓ 개요
    - Samba(삼바)는 리눅스나 유닉스 계열 운영체제와 윈도우 운영체제 간의 파일 및 프린터 공유를 가능하게 해주는 오픈 소스 소프트웨어
    - 마이크로소프트의 네트워크 프로토콜인 SMB(Server Message Block) 및 CIFS(Common Internet File System)를 구현하여 서로 다른 OS가 마치 같은 네트워크에 있는 것처럼 자원을 공유할 수 있게 함
    - Samba는 특히 내부 네트워크(LAN) 환경에서 여러 사용자가 동시에 같은 파일을 조회하거나 수정해야 하는 협업 환경에서 매우 강력한 도구

Samba Server

- Samba
  - ✓ 주요 특징
    - 이 기종 간의 호환성(Cross-platform): 윈도우의 네트워크 공유 폴더를 리눅스에서 마운트하거나 리눅스의 특정 디렉토리를 윈도우 PC에서 네트워크 드라이브(Z:, Y: 등)로 연결하여 사용할 수 있음
    - 파일 및 프린터 공유: 단순한 파일 전송을 넘어 서버에 연결된 프린터를 네트워크 상의 다른 컴퓨터들이 공유해서 사용할 수 있도록 지원
    - 액티브 디렉토리(Active Directory) 통합: 윈도우 도메인 컨트롤러와 연동하여 사용자 인증 및 권한 관리를 통합적으로 수행할 수 있음
    - 강력한 보안 및 권한 설정: 사용자별, IP별로 접근 권한을 세밀하게 제어할 수 있으며 암호화된 인증 방식을 지원
    - 성능과 안정성: 가볍고 효율적인 설계로 저사양 하드웨어에서도 안정적인 파일 서버 역할을 수행

Samba Server

- Samba
  - ✓ 주요 구성 요소 (데몬)
    - smbd: 파일 공유 및 프린터 공유 서비스를 제공하며 사용자 인증(Login)과 리소스에 대한 접근 권한(Locking)을 담당
    - nmbd: 윈도우 네트워크 상에서 컴퓨터 이름을 IP 주소로 변환해주는 이름 해석(Name Resolution) 서비스와 브라우징 기능을 담당 (윈도우의 네트워크 환경에서 컴퓨터 이름이 보이게 함)
  - ✓ SSH 전송(SCP/SFTP)과의 차이점

구분	SSH (SCP/SFTP)	Samba (SMB)
주 사용 용도	일회성 파일 전송, 원격 관리, 배포	지속적인 네트워크 드라이브 연결
연결 방식	필요할 때만 접속 (Session)	항상 연결된 것처럼 사용 (Mount)
체감 성능	전송 시 암호화 부하가 다소 있음	로컬 드라이브처럼 빠른 탐색 가능
편의성	명령어 위주 (CLI 중심)	윈도우 탐색기 위주 (GUI 중심)