

CentOS 리눅스

시스템 & 네트워크



Chapter 01. 리눅스 설치와 기본 사용법

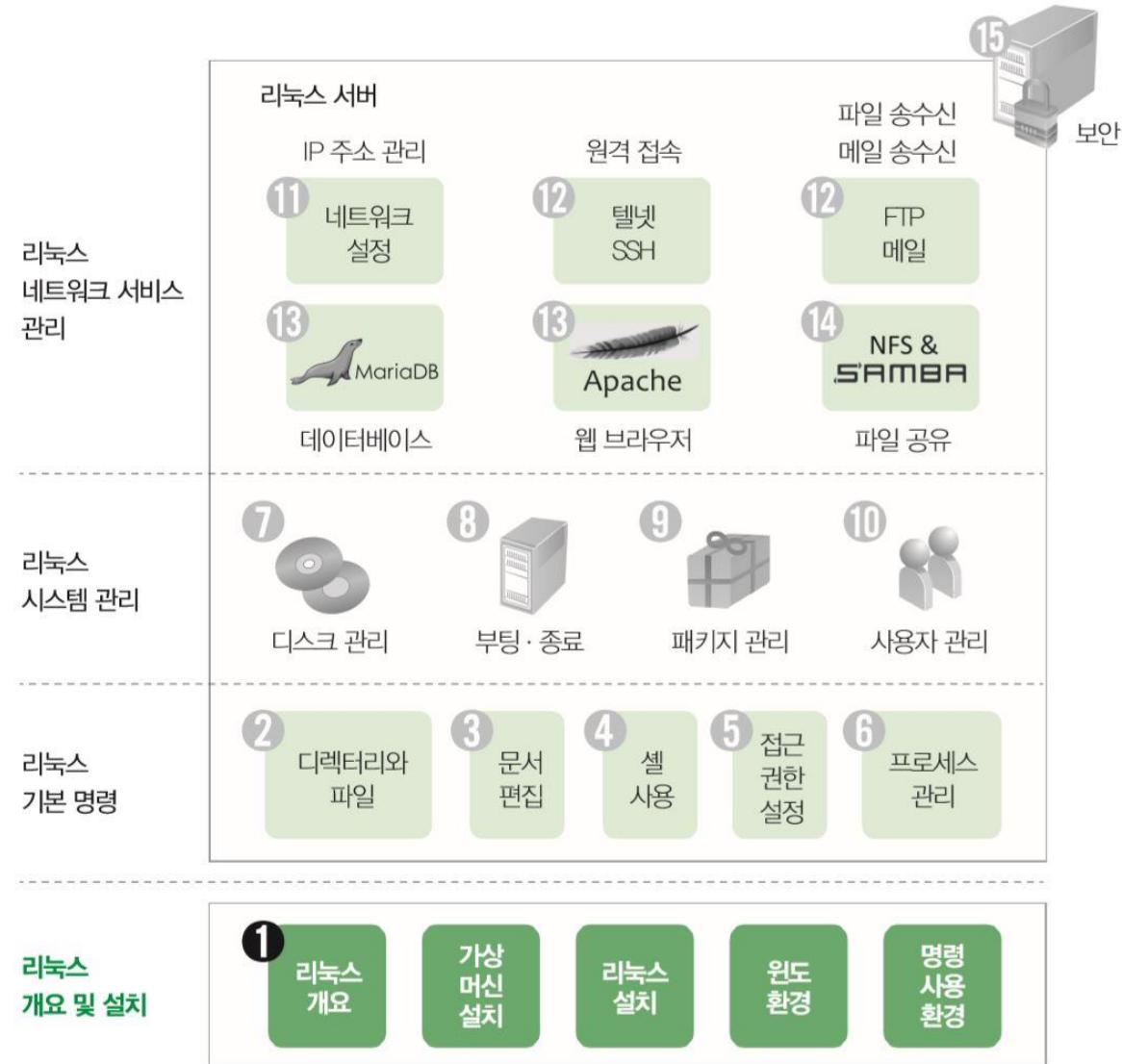
목차

- 00. 개요
- 01. 리눅스 기초
- 02. 리눅스 실습 환경 구축
- 03. 리눅스 원도 기본 사용법
- 04. 리눅스 명령 사용법

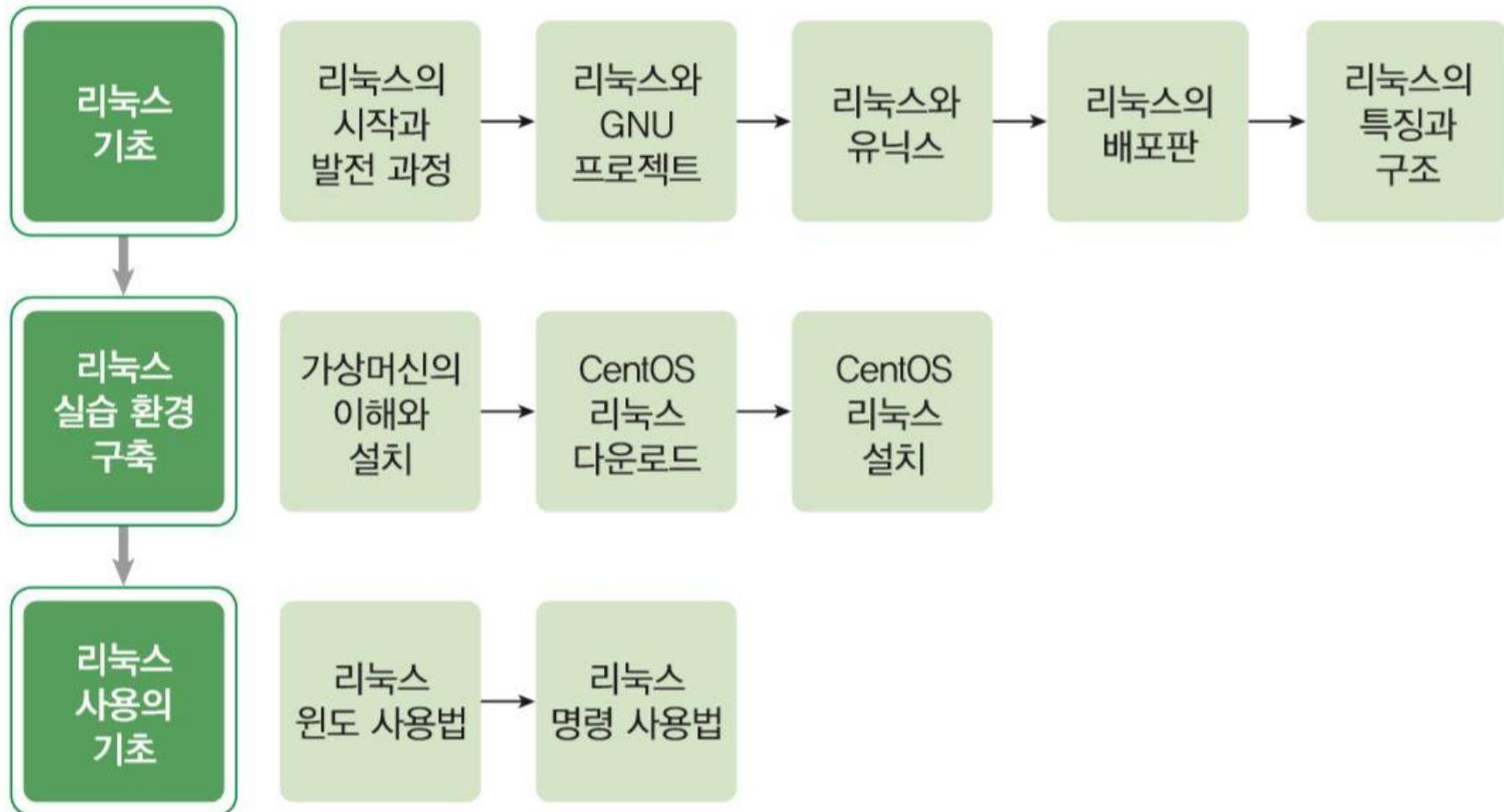
학습목표

- 리눅스의 탄생과 발전 과정을 설명할 수 있다.
- GNU 프로젝트, 리눅스 배포판, 가상머신이 무엇인지 설명할 수 있다.
- 가상머신을 설치하고 여기에 리눅스를 설치할 수 있다.
- 리눅스 원도 환경의 메뉴 구조를 이해하고 필요한 응용 프로그램을 실행할 수 있다.
- 리눅스에서 터미널 창을 열고 기본 명령을 사용할 수 있다.

00. 개요



00. 개요



01.리눅스 기초

■ 리눅스의 시작과 발전

- 핀란드 헬싱키대학교의 학생인 리눅스 베네딕트 토르발스가 처음 개발
- 미닉스(MINIX)라는 교육용 운영체제를 참조하여 개발
- 리눅스 개발 소식을 comp.os.minix 뉴스 그룹에 포스팅: 1991년 8월 26일 -> 리눅스 탄생일
 - 번역

미닉스를 사용하는 여러분, 안녕하세요.

저는 386(486) AT 클론에서 동작하는 (무료) 운영체제를 개발하고 있습니다(취미일 뿐이며 GNU처럼 거대하거나 전문적이지는 않습니다). 지난 4월부터 준비 중이며 이제 거의 다 되었습니다. 미닉스의 장단점에 대한 피드백을 보내주시면 제 OS에 반영하고 싶습니다[파일 시스템의 물리적 레이아웃은 (실용적 이유 때문에) 같습니다].

저는 현재 배시 셸(1.08)과 gcc(1.40)를 이식했는데 동작하는 것 같습니다. 이는 제가 몇 달 안에 실제적인 것을 작업할 수 있음을 의미하며, 대부분의 사람들이 원하는 기능이 무엇인지 알고 싶습니다. 어떤 제안이든 환영하지만 제안을 모두 구현한다고는 약속할 수 없습니다.

리눅스(torv...@kruuna.helsinki.fi)

추신: 물론 미닉스 코드를 전혀 사용하지 않았고 멀티스레드 파일 시스템을 가지고 있습니다. 이식 가능한 상태는 아니며(386 테스크 스위칭 등을 사용해서) 아마도 AT 하드디스크 외에는 지원할 수 없을 것입니다(제가 가진 게 그것뿐이라).

01.리눅스 기초

■ 리눅스의 시작과 발전

- 1991년 8월에 처음 알려진 이후 핵심 부분인 리눅스 커널이 공개 (0.01 버전)
- GNU 프로젝트 : 리눅스 커널에 응용 프로그램 제공 -> GNU/리눅스
- 2020년 1월 26일 기준, 안정 버전은 5.4.15이고 5.5 버전을 개발 중

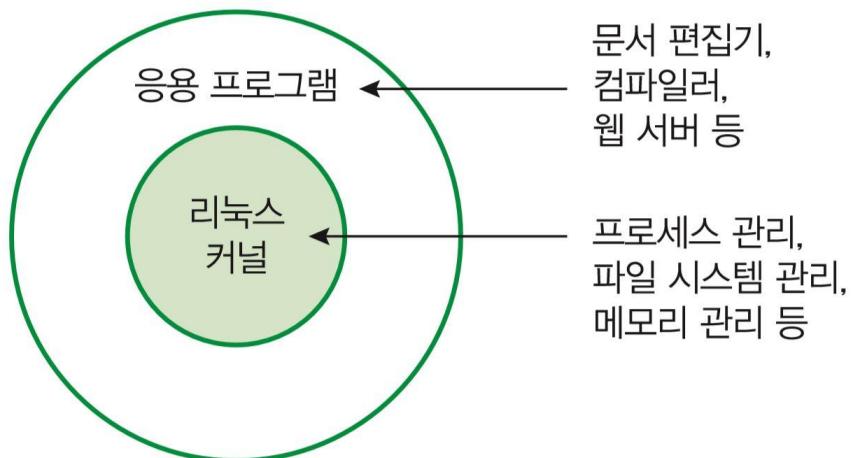


그림 1-1 커널과 응용 프로그램

01.리눅스 기초

■ 리눅스의 시작과 발전

- 리눅스 재단 : 2007년 설립된 비영리 단체
 - 리눅스 토르발스 지원
 - 마이크로소프트, 오라클, AT&T, 퀄컴, 시스코, 후지쯔, 화웨이, 인텔, 삼성전자, IBM 등
 - 리눅스 커널은 두세 달 간격으로 업데이트 버전이 배포
 - 리눅스 커널 아카이브(www.kernel.org)에서 누구나 소스와 패치를 내려 받을 수 있음

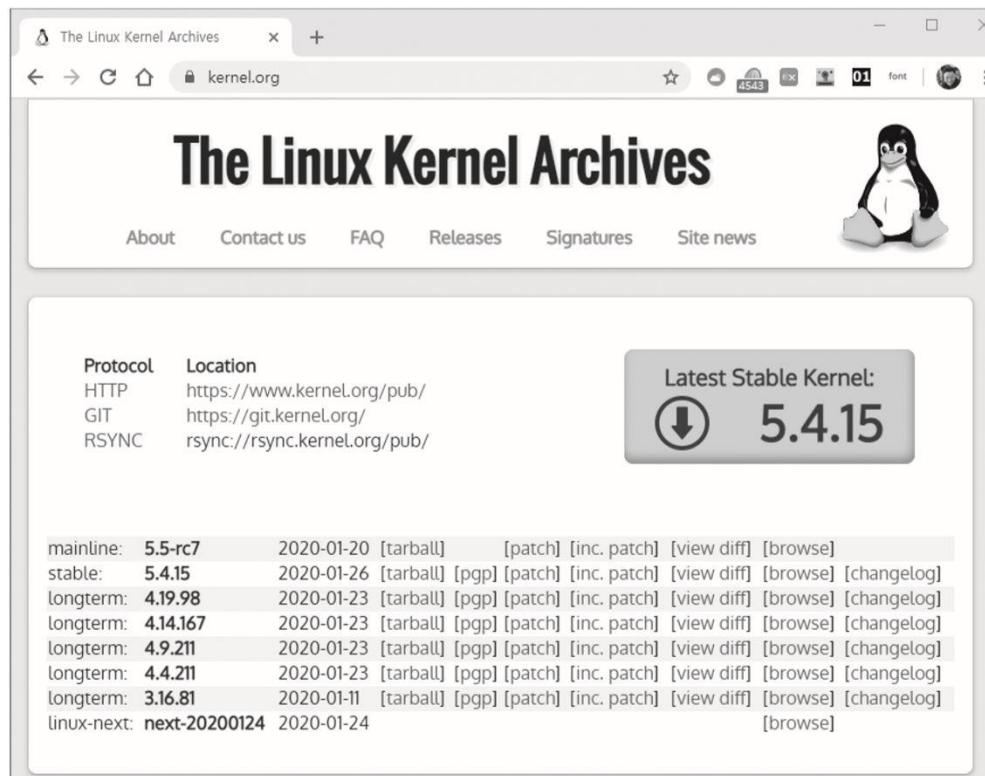


그림 1-2 리눅스 커널 아카이브 웹 사이트

01. 리눅스 기초

■ 리눅스와 GNU 프로젝트

- 리처드 스톤먼에 의해 리눅스보다 더 이른 시점에 시작
- 1985년 <GNU 선언문>을 발표, 비영리 기관인 '자유 소프트웨어재단 FSF'을 설립

■ GNU

- 유닉스와 호환되는 자유 소프트웨어를 개발하는 프로젝트
- 'GNU is Not Unix'의 약자
- 모든 소프트웨어를 자유롭게 사용하도록 하는데 목적
- 다음 4가지 자유를 보장
 - 프로그램을 어떠한 목적으로도 실행할 수 있는 자유
 - 프로그램이 어떻게 동작하는지 학습하고, 자신의 필요에 맞게 개작할 수 있는 자유
 - 이웃을 도울 수 있도록 복제물을 재배포할 수 있는 자유
 - 프로그램을 개선할 수 있는 자유와 개선된 이점을 공동체 전체가 누릴 수 있도록 발표할 자유
- GNU 프로젝트는 자체적으로 운영체제(커널)를 개발 중이었으나 완성하지 못함
- 리누스 토르발스가 개발한 리눅스 커널과 GNU의 각종 응용 프로그램이 결합, 완성된 형태의 운영체제 탄생

01.리눅스 기초

■ 리눅스와 GNU 프로젝트

▪ GPL

- 1989년에 제정된 GNU 프로젝트로 배포한 프로그램의 자유 소프트웨어 라이선스
- 다음 다섯 가지 의무를 저작권의 한 부분으로 강제하고 있음
 - 컴퓨터 프로그램을 어떠한 목적으로든 사용할 수 있다. 다만 법으로 제한하는 행위는 할 수 없다.
 - 컴퓨터 프로그램의 실행 복사본은 언제나 프로그램의 소스 코드와 함께 판매하거나 소스 코드를 무료로 배포해야 한다.
 - 컴퓨터 프로그램의 소스 코드를 용도에 따라 변경할 수 있다.
 - 변경된 컴퓨터 프로그램 역시 프로그램의 소스 코드를 반드시 공개 배포해야 한다.
 - 변경된 컴퓨터 프로그램 역시 반드시 똑같은 라이선스를 취해야 한다. 즉 GPL 라이선스를 적용해야 한다

▪ GPL 버전

- GPLv1
 - 1989년 1월에 발표, 이에 근거하여 프로그램을 배포할 때는 이해하기 쉬운 소스 코드를 같이 배포해야 한다는 조건
 - GPLv1 프로그램을 수정한 프로그램은 원래 프로그램과 마찬가지로 GPLv1을 따라야 함
- GPLv2
 - 1991년 6월에 발표, GPL 라이선스 프로그램을 배포하는 것을 막는 조건
- GPLv3
 - 2007년 6월에 발표, 소프트웨어 특허에 대한 대처, 다른 라이선스와의 호환성, 디지털 저작권 관리에 관한 내용이 포함

01. 리눅스 기초

■ 리눅스의 배포판

- 일반 사용자들이 리눅스를 쉽게 사용하도록 하는것이 목적
- 누구나 내려받아 설치하고, 수정하여 다시 자유롭게 배포가능
- 리눅스 커널 + 응용프로그램으로 구성

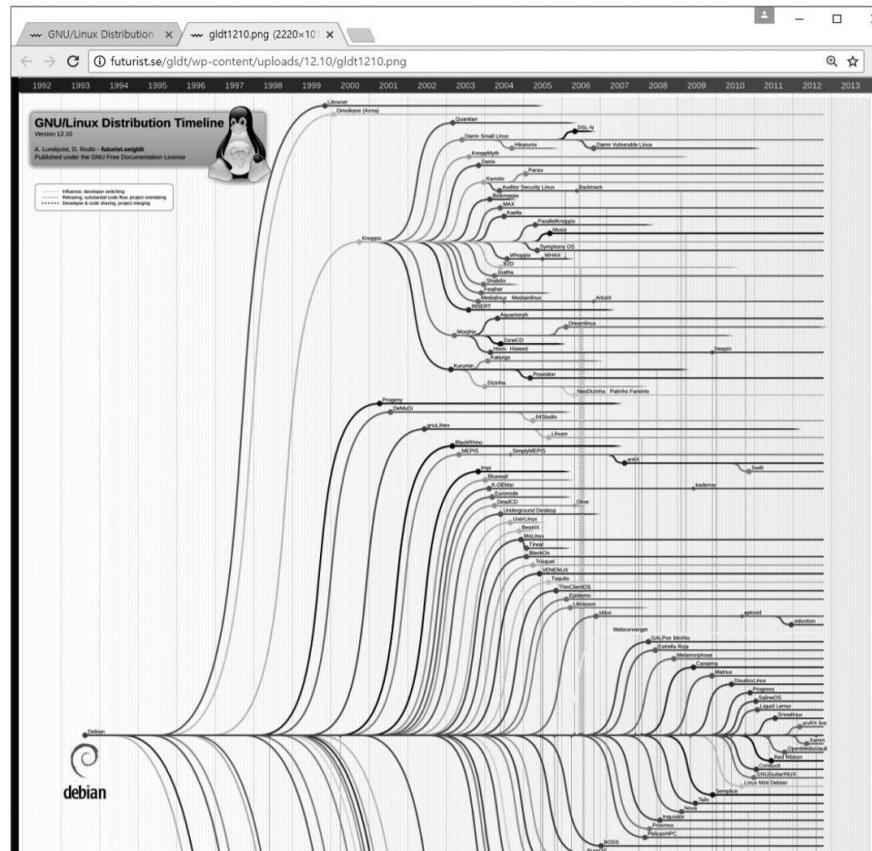


그림 1-3 리눅스 배포판의 계통도 일부(출처: <http://futurist.se/gldt>)

01. 리눅스 기초

■ 리눅스의 배포판

- 크게 레드햇 계열, 데비안 계열, 슬랙웨어 계열로 구분
- 기본 사용에는 큰 차이가 없지만 각 계열에 따라 패키지 관리 명령 등 일부 기능이 다름

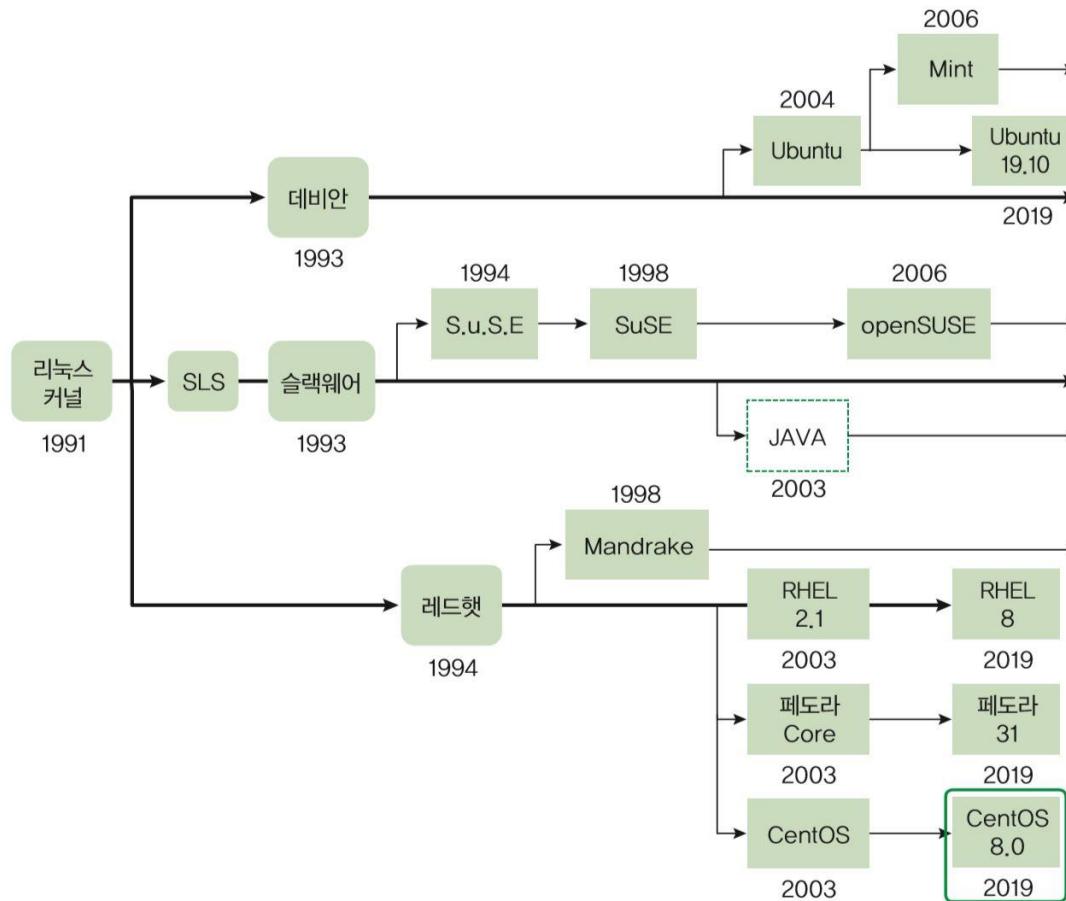


그림 1-4 주요 리눅스 배포판의 계통도

01. 리눅스 기초

■ 리눅스의 배포판

- 크게 레드햇 계열, 데비안 계열, 슬랙웨어 계열로 구분
- 기본 사용에는 큰 차이가 없지만 각 계열에 따라 패키지 관리 명령 등 일부 기능이 다름

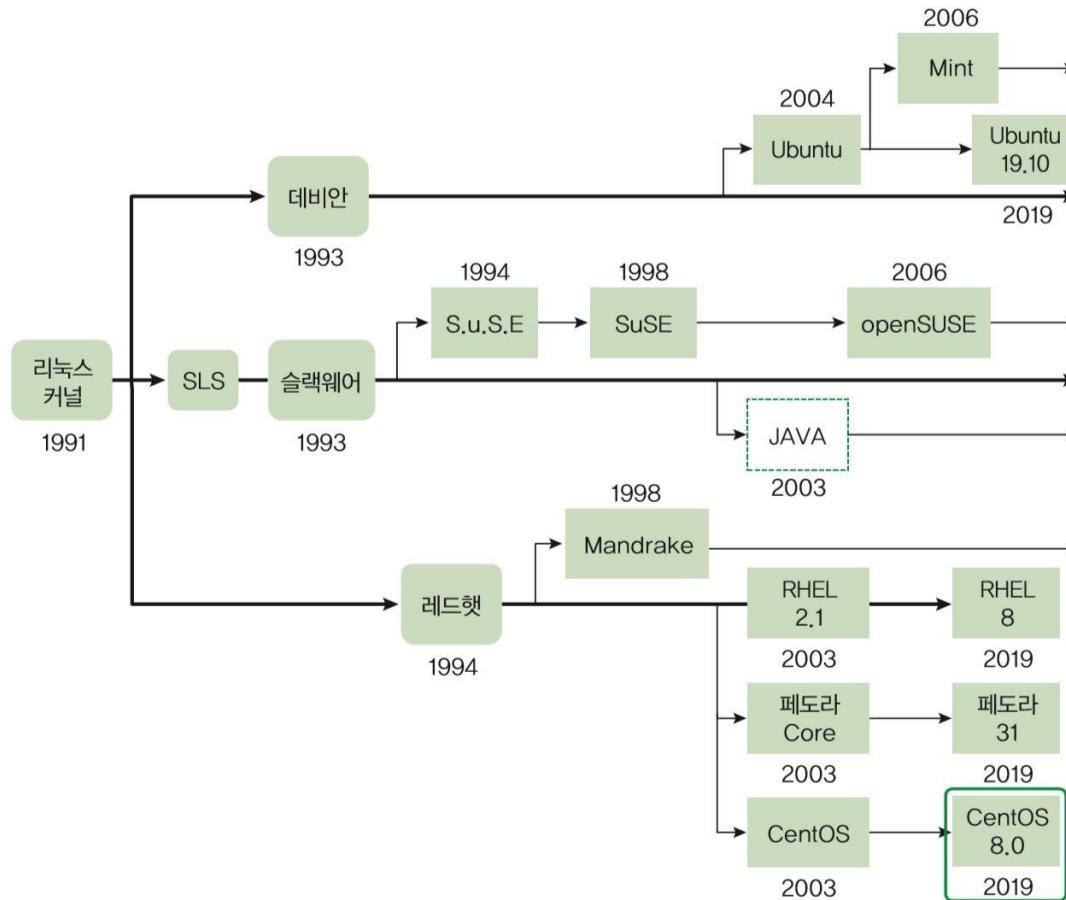


그림 1-4 주요 리눅스 배포판의 계통도

01. 리눅스 기초

■ 리눅스의 특징과 구조

■ 특징

- 리눅스는 공개 소프트웨어이며 무료로 사용할 수 있음
- 유닉스와의 완벽한 호환성을 유지
- 서버용 운영체제로 많이 사용
- 편리한 GUI 환경을 제공

■ 구조

• 커널

- 리눅스의 핵심
- 프로세스/메모리/파일시스템/장치 관리
- 컴퓨터의 모든 자원 초기화 및 제어 기능

• 셸

- 사용자 인터페이스
- 명령해석
- 프로그래밍기능
- 배시 셸을 기본으로 사용

• 응용 프로그램

- 각종 프로그래밍 개발도구
- 문서 편집 도구
- 네트워크 관련 도구 등

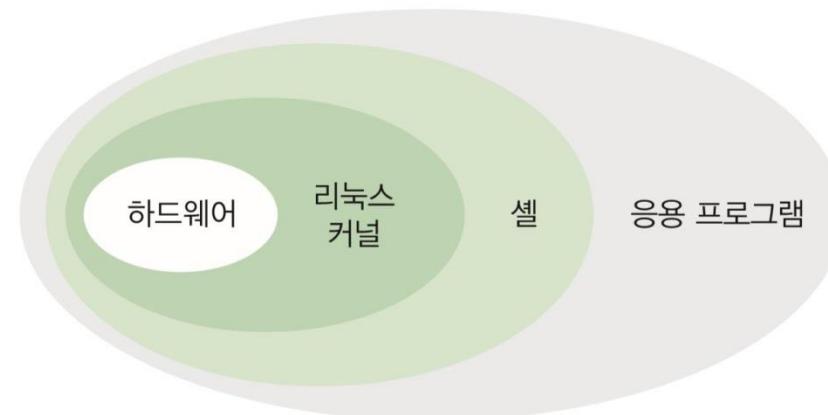


그림 1-5 리눅스의 구조

02. 리눅스 실습 환경 구축

■ 가상머신 개념과 종류

▪ 가상머신

- PC에 설치되어 있는 운영체제(호스트 OS)에 가상의 머신(시스템)을 생성한 후 여기에 다른 운영체제(게스트 OS)를 설치할 수 있도록 해주는 응용 프로그램

표 1-1 가상머신의 종류

가상머신	호스트 OS	게스트 OS
VMware	윈도 계열 운영체제, 대부분의 리눅스, 애플 맥	윈도 계열 운영체제, 대부분의 리눅스 배포판, 솔라리스, Mac OS
버추얼 PC	윈도 계열 운영체제	윈도 계열 운영체제, 일부 리눅스, 솔라리스
버추얼 박스	윈도 계열 운영체제, 대부분의 리눅스, 애플 맥, 솔라리스	윈도 계열 운영체제, 대부분의 리눅스 배포판, 솔라리스, Mac OS, OpenBSD

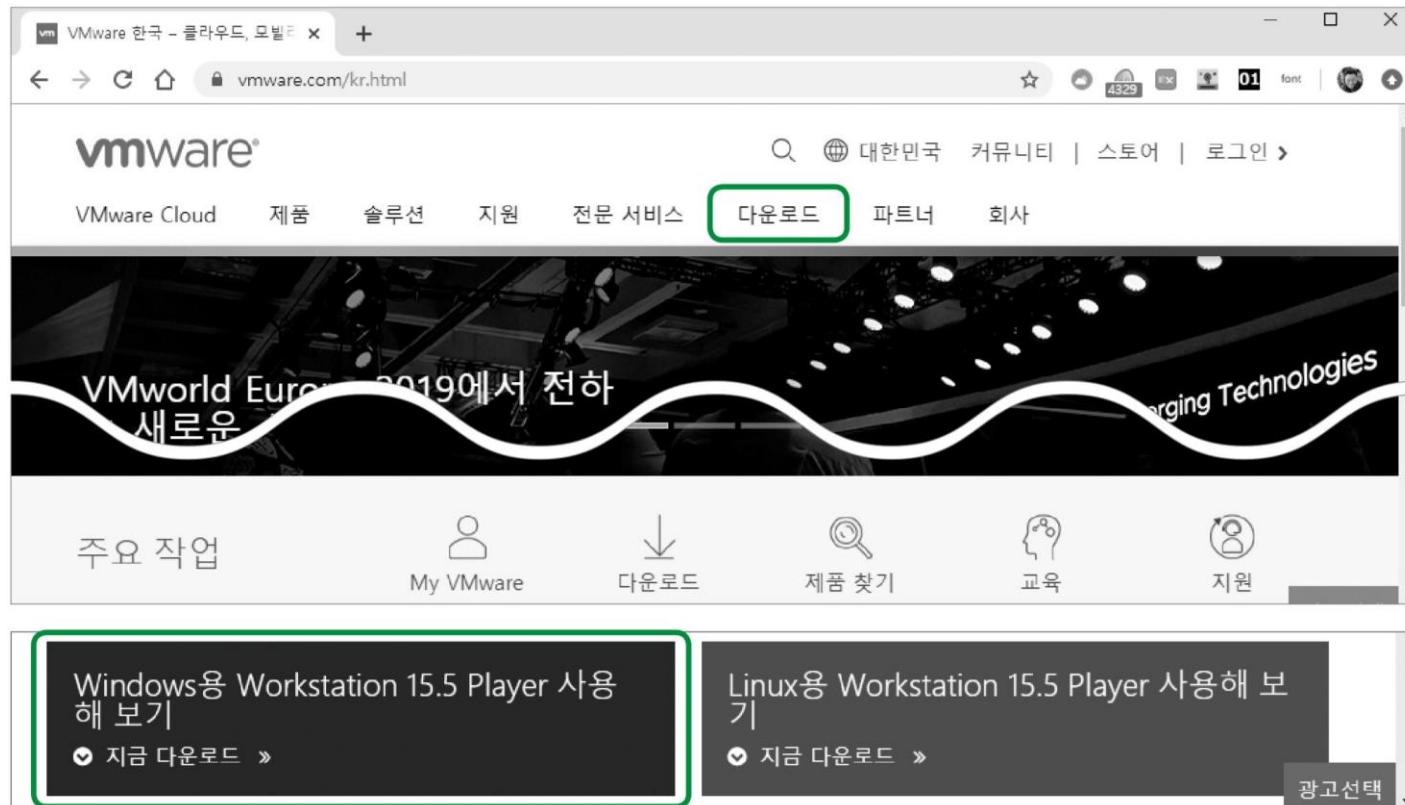
- VMware Workstation Pro
 - 유료 제품으로 30일간 평가판을 무료로 사용할 수 있음
- VMware Workstation Player
 - 비영리 및 개인의 경우 무료로 사용할 수 있는 제품
 - 시리얼 번호도 없이 바로 설치하여 사용할 수 있음
 - 상용 제품인 VMware Workstation Pro의 기능을 대부분 제공

02. 리눅스 실습 환경 구축

■ 가상머신의 설치

- [따라해보기] VMware 다운로드하고 설치하기

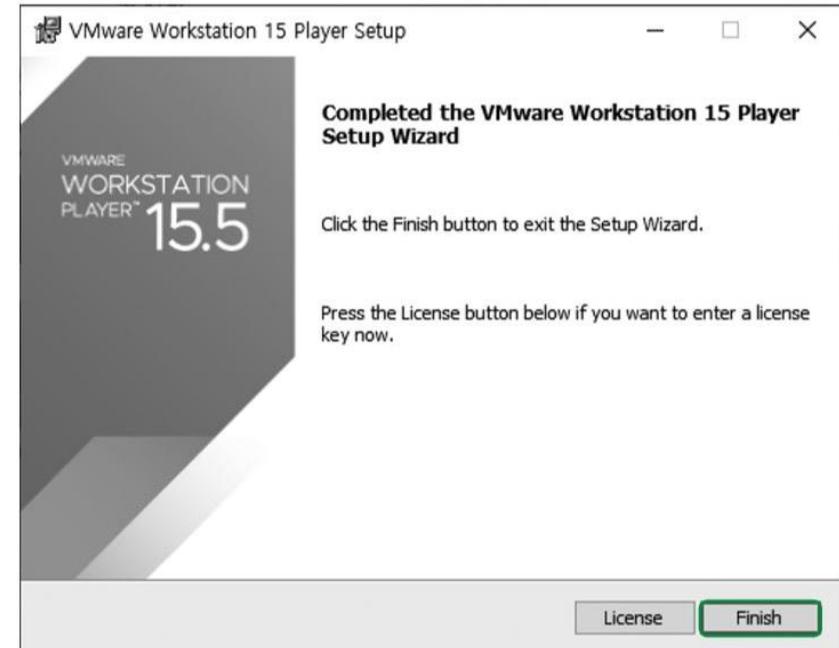
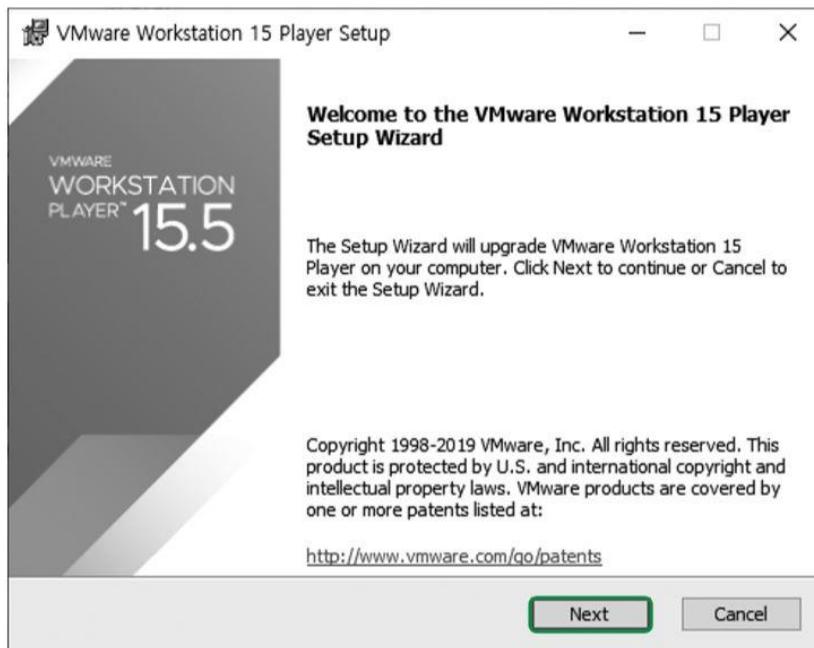
- ① VMware 다운로드하기



02. 리눅스 실습 환경 구축

■ 가상머신의 설치

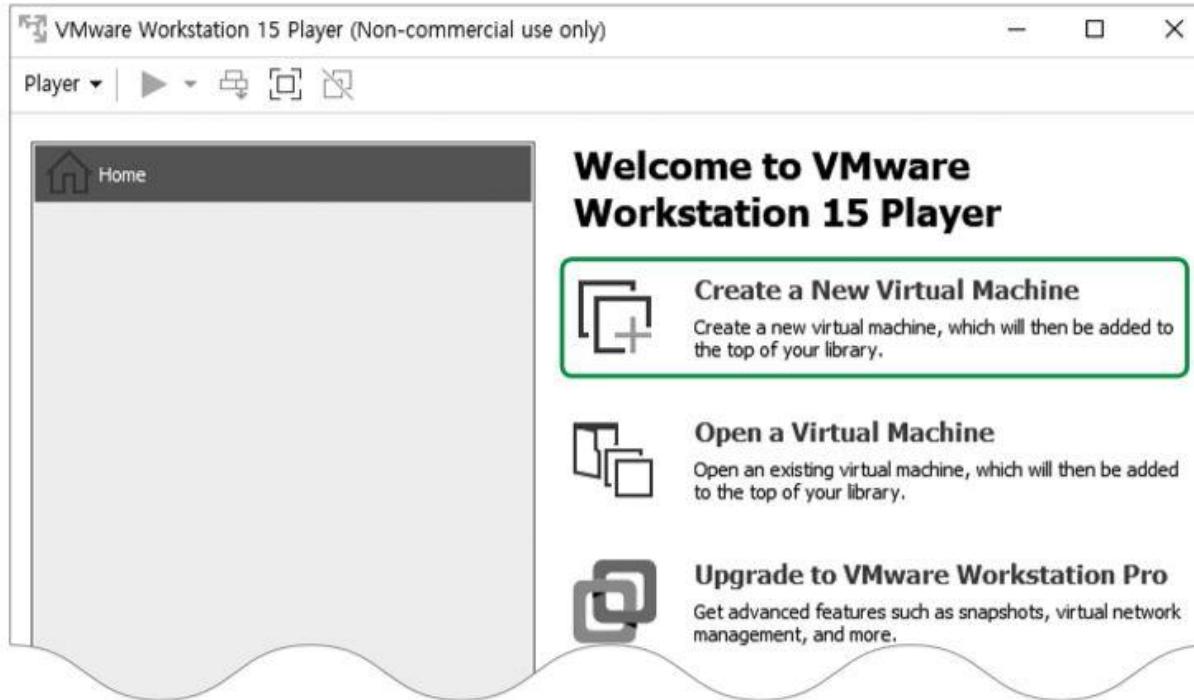
- [따라해보기] VMware 다운로드하고 설치하기
 - VMware 설치하기



02. 리눅스 실습 환경 구축

■ 가상머신의 설치

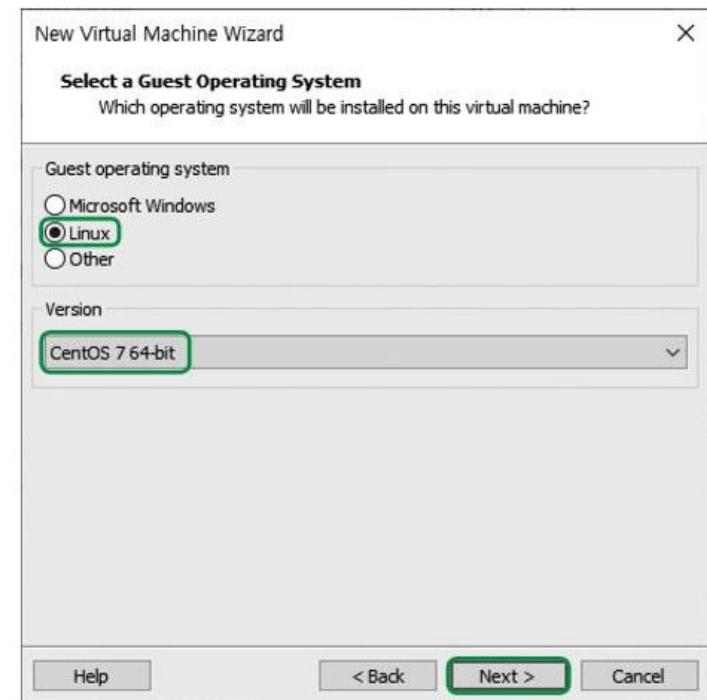
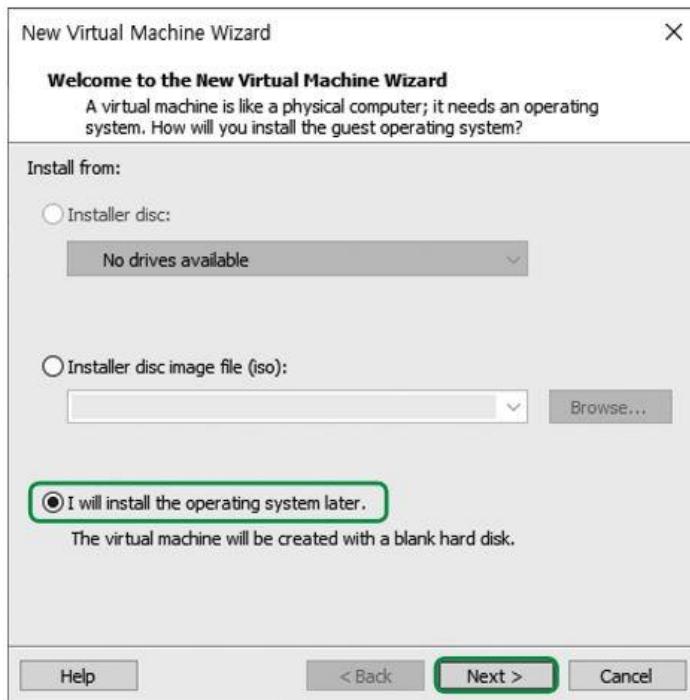
- [따라해보기] 가상머신 생성하기
 - ➊ VMware 실행하기



02. 리눅스 실습 환경 구축

■ 가상머신의 설치

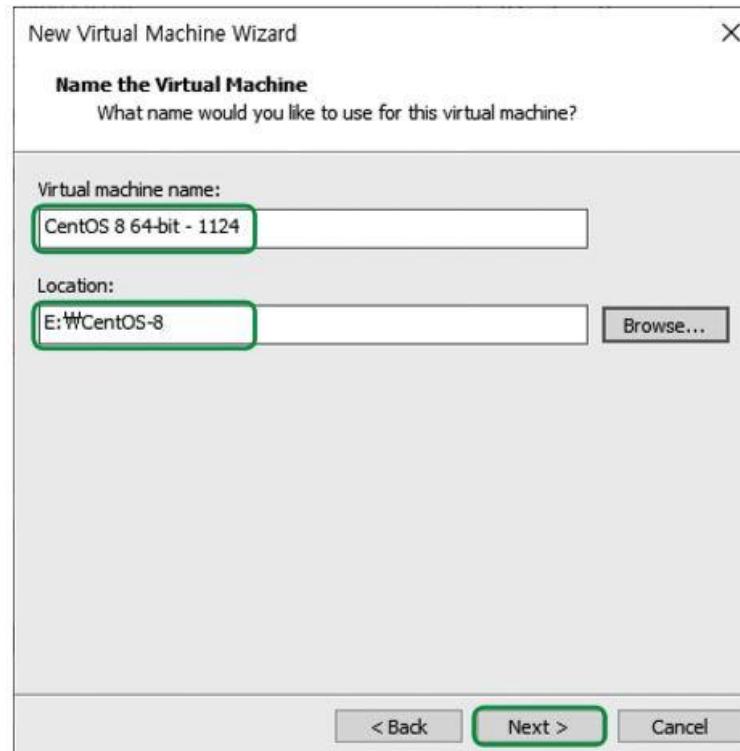
- [따라해보기] 가상머신 생성하기
 - ② 게스트 OS 설치 방법 선택하기



02. 리눅스 실습 환경 구축

■ 가상머신의 설치

- [따라해보기] 가상머신 생성하기
 - ③ 가상머신의 이름과 디스크 파일의 위치 지정하기



02. 리눅스 실습 환경 구축

■ 가상머신의 설치

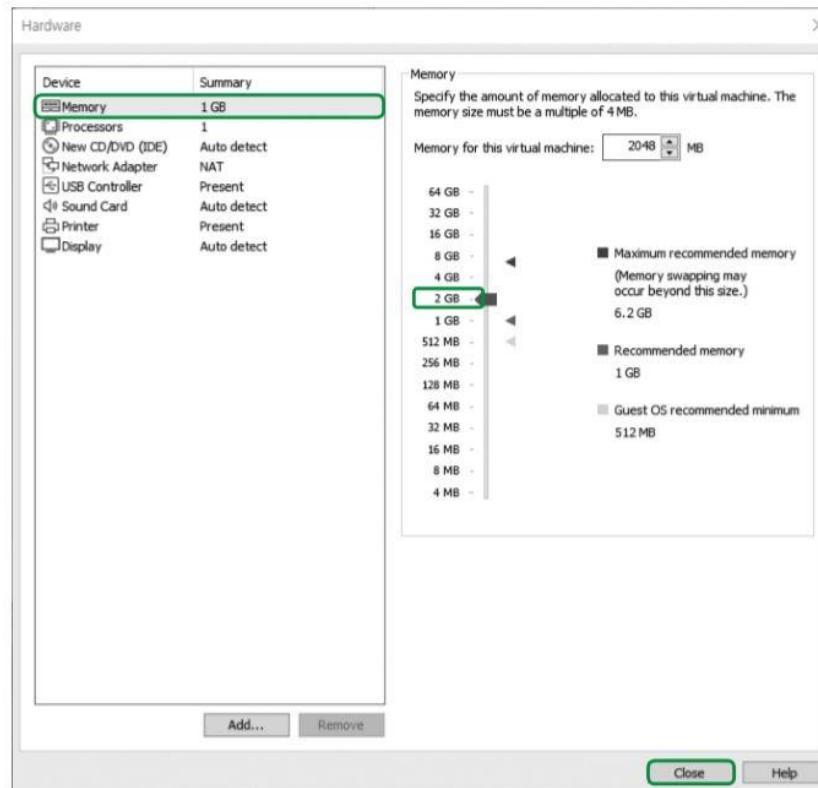
- [따라해보기] 가상머신 생성하기
 - ④ 디스크 파일의 크기와 저장 방식 지정하기



02. 리눅스 실습 환경 구축

■ 가상머신의 설치

- [따라해보기] 가상머신 생성하기
⑤ 하드웨어 정보 수정하기

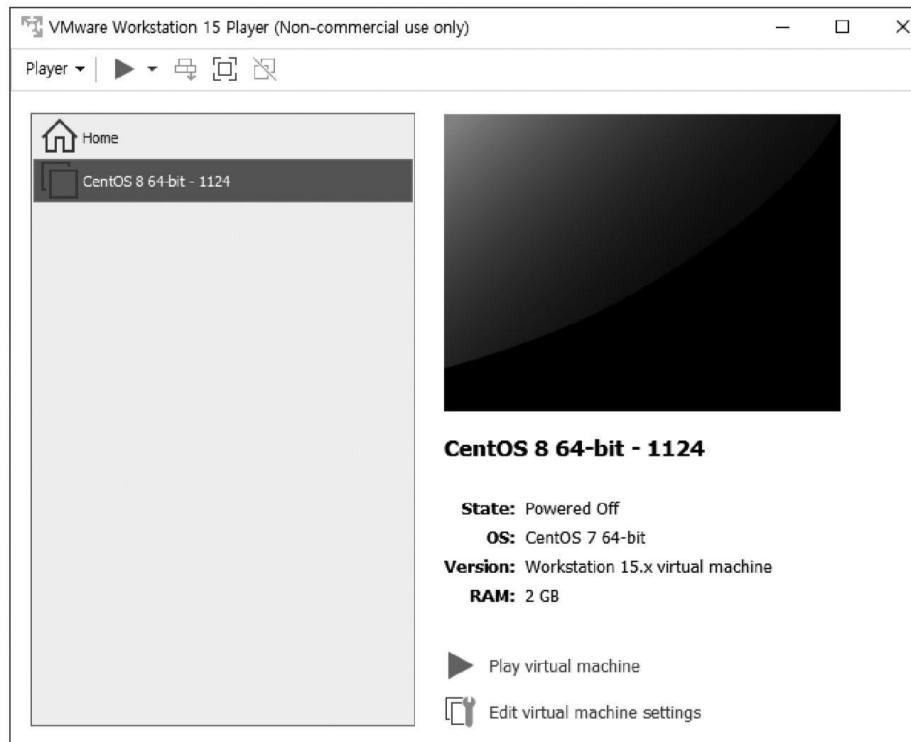


02. 리눅스 실습 환경 구축

■ 가상머신의 설치

- [따라해보기] 가상머신 생성하기

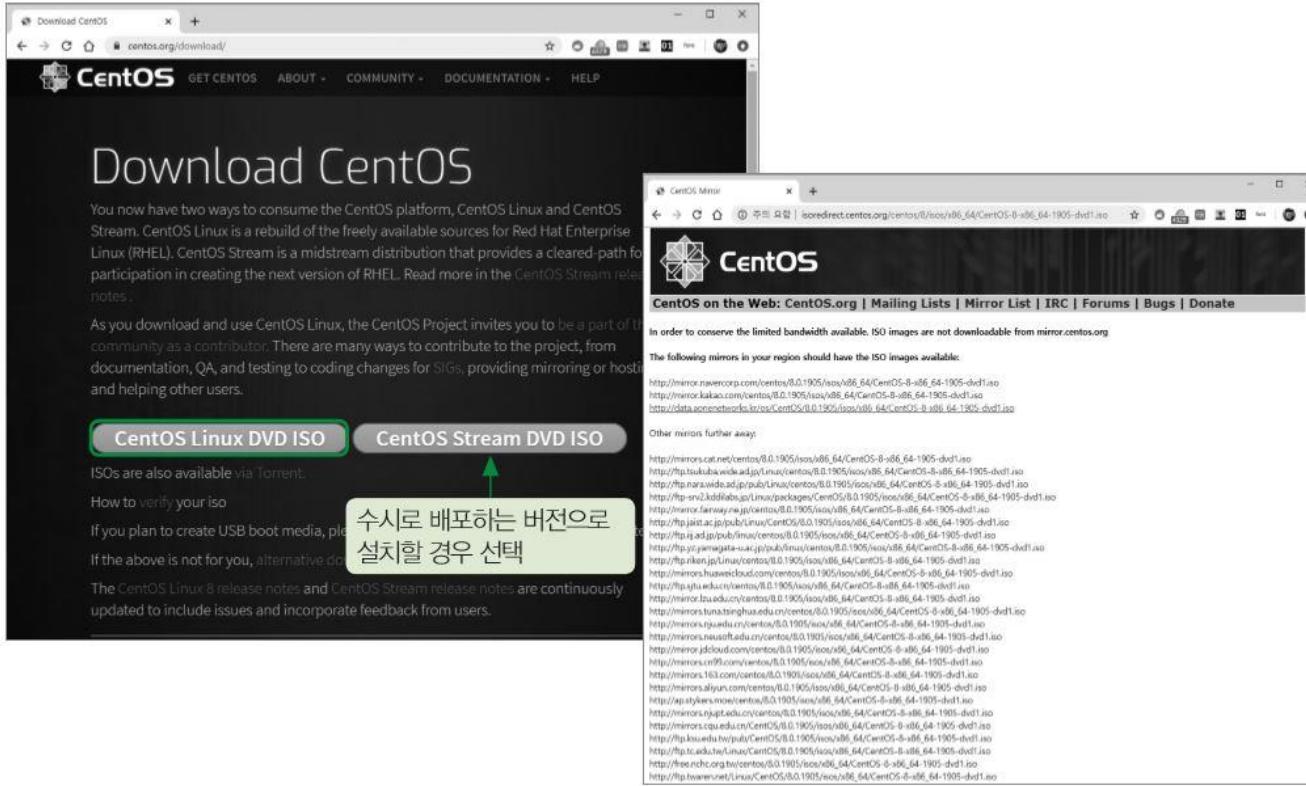
⑥ 가상머신 생성 완료하기



02. 리눅스 실습 환경 구축

■ 리눅스의 설치

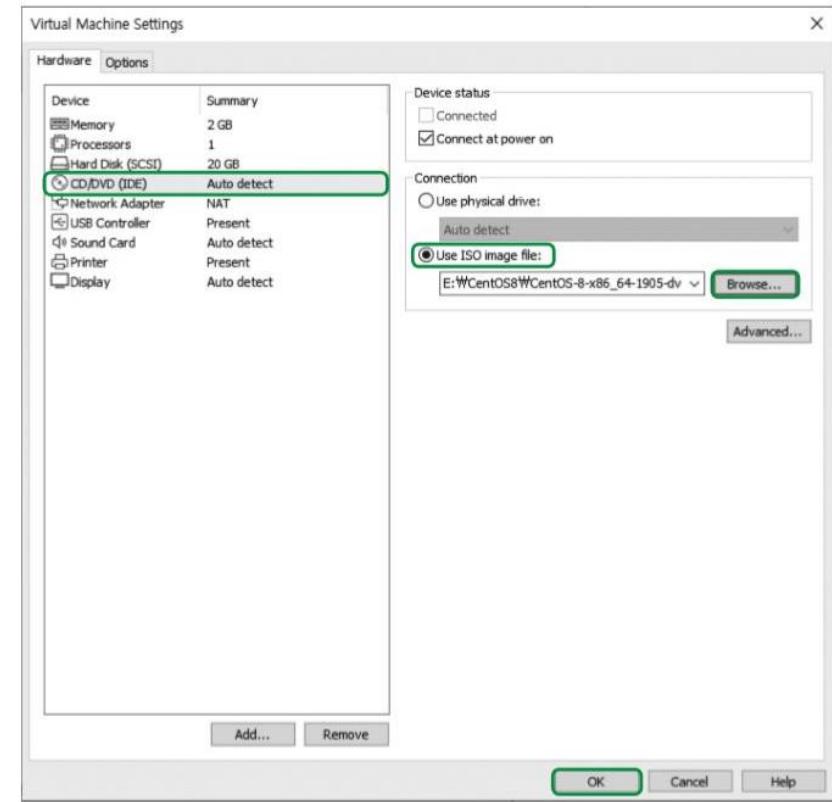
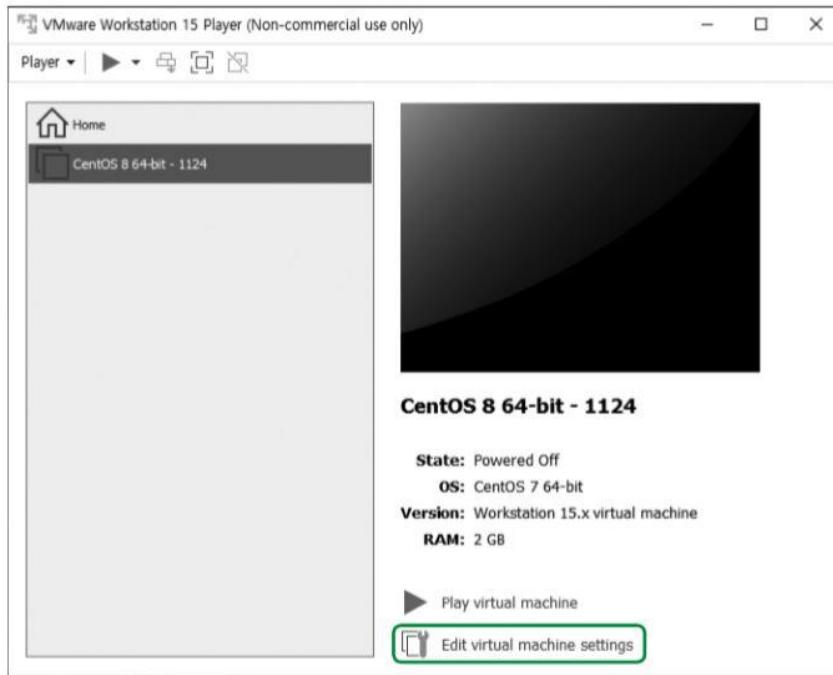
- [따라해보기] CentOS 다운로드하기
 - 이미지 파일(ISO 파일) 다운로드



02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- [따라해보기] CentOS 설치하기
 - ISO 파일 설정하고 VMware 시작하기



02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- [따라해보기] CentOS 설치하기

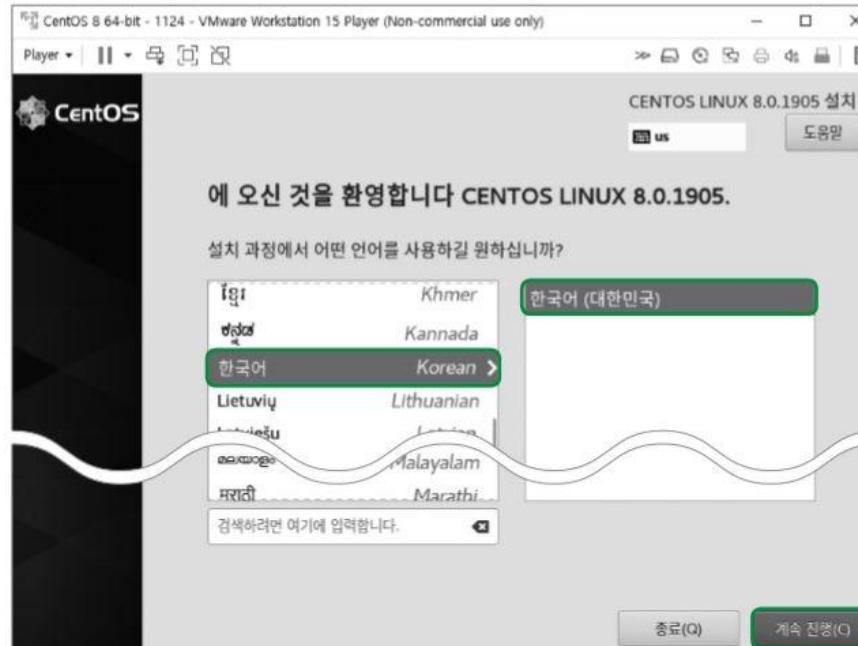
- ② 리눅스 설치를 위해 시스템 확인하기



02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- [따라해보기] CentOS 설치하기
 - ③ 설치 과정에서 사용할 언어 선택하기

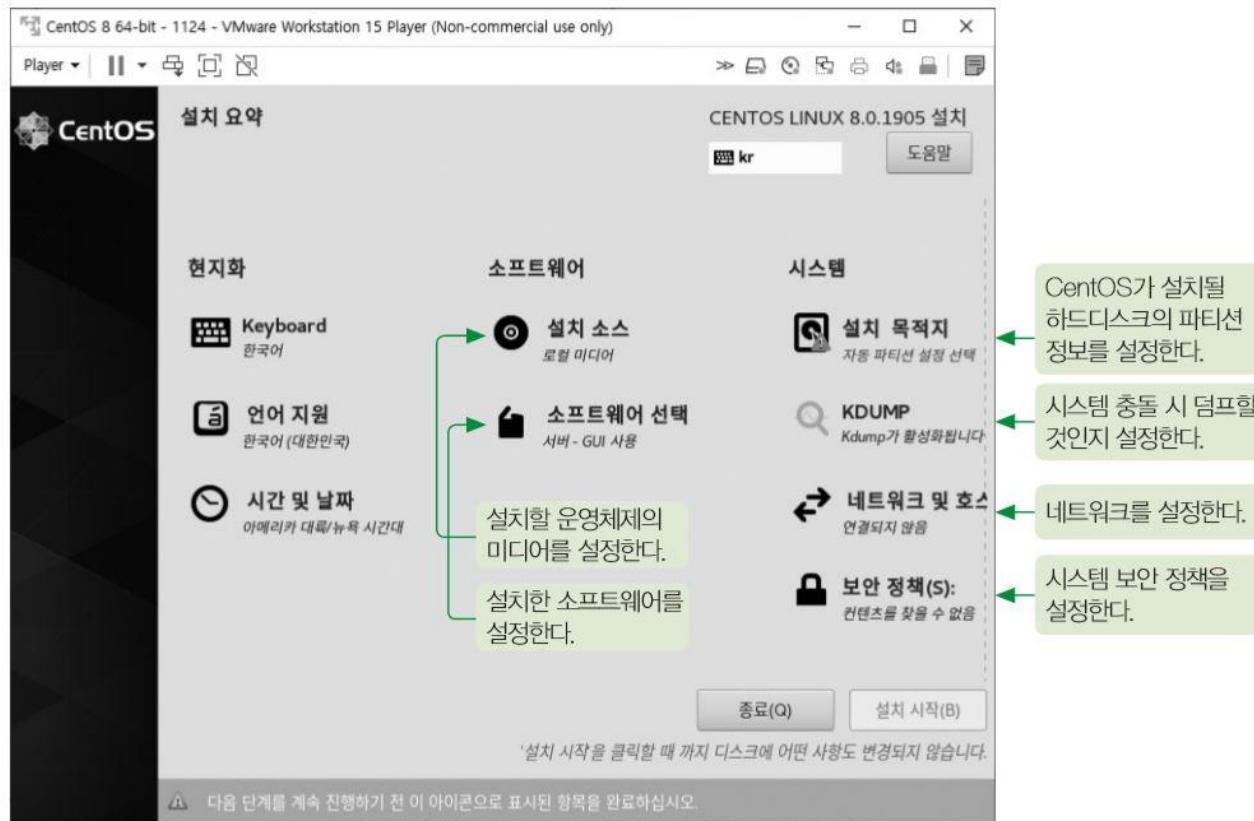


02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- [따라해보기] CentOS 설치하기

④ 설치 요약 확인하기



02. 리눅스 실습 환경 구축

■ 리눅스의 설치

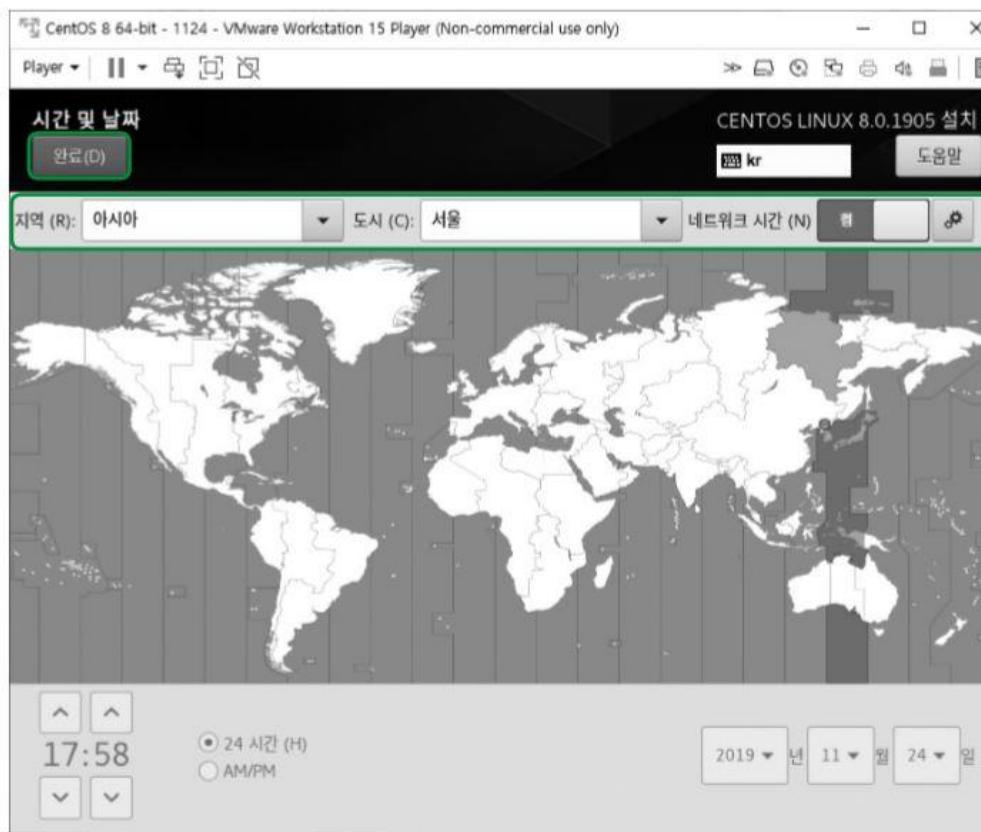
- [따라해보기] CentOS 설치하기
⑤ 네트워크 설정하기



02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- [따라해보기] CentOS 설치하기
 - ⑥ 시간과 날짜 설정하기

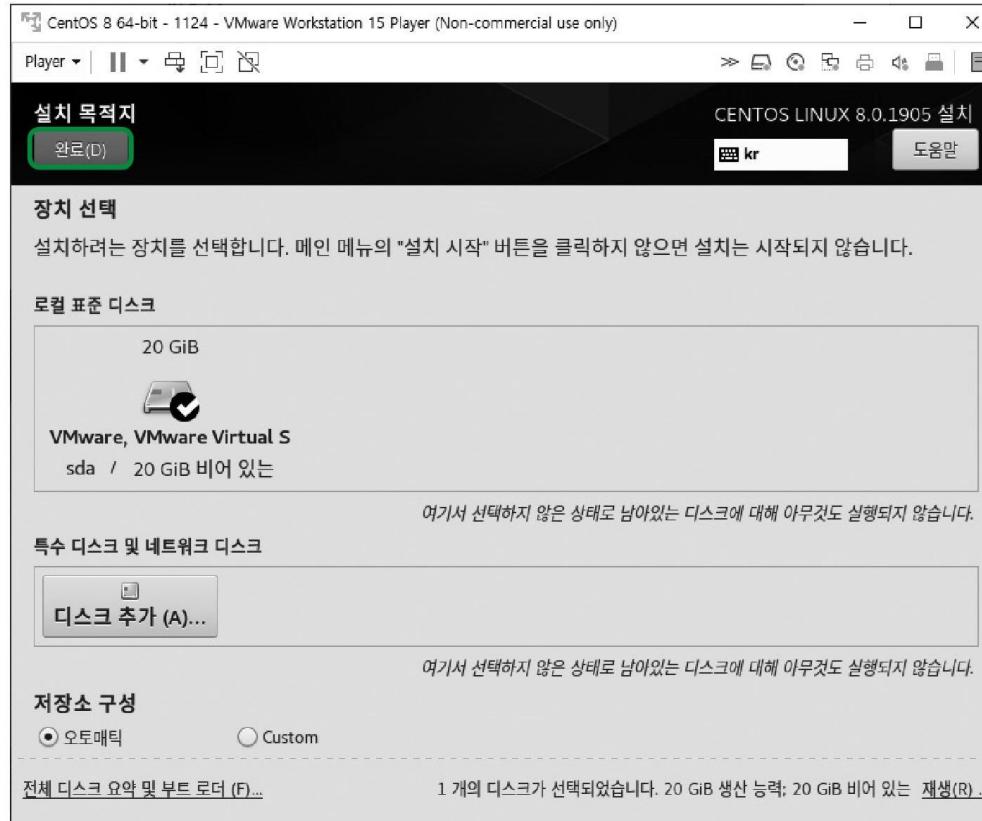


02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- [따라해보기] CentOS 설치하기

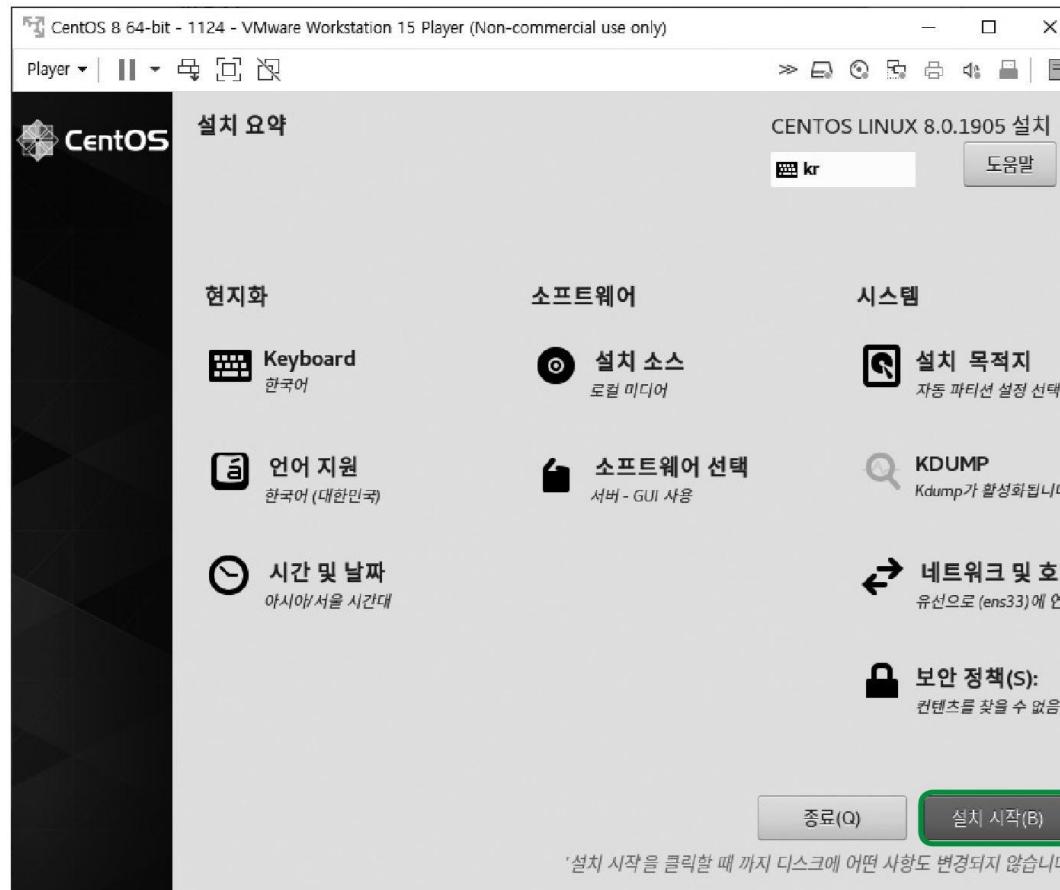
⑦ 설치 목적지 확인하고 디스크 파티션 설정하기



02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- [따라해보기] CentOS 설치하기
⑧ 설치 시작하기

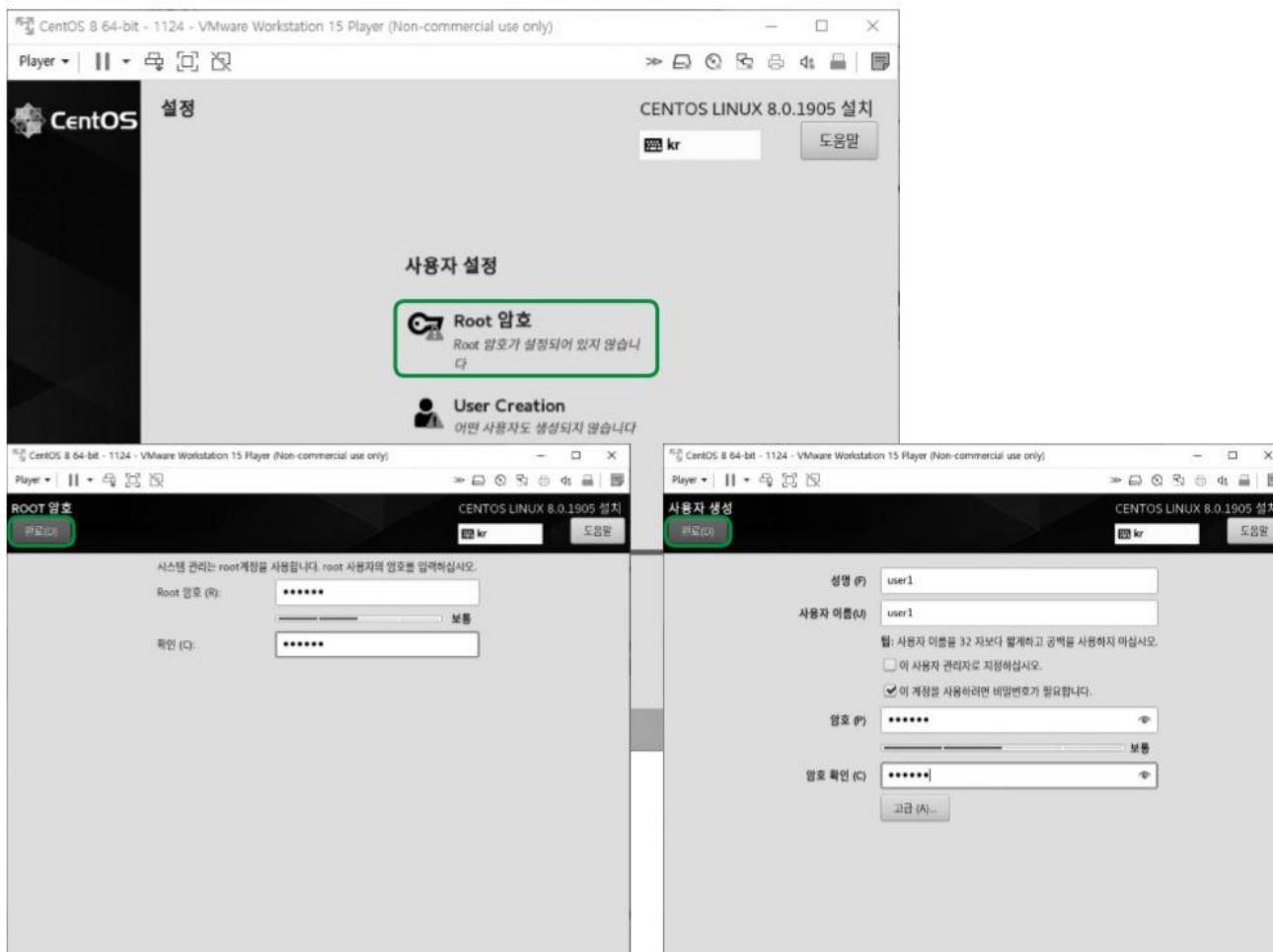


02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- [따라해보기] CentOS 설치하기

⑨ root 암호 설정하고 사용자 생성하기



02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- [따라해보기] CentOS 설치하기
⑩ 패키지 설치 완료하기

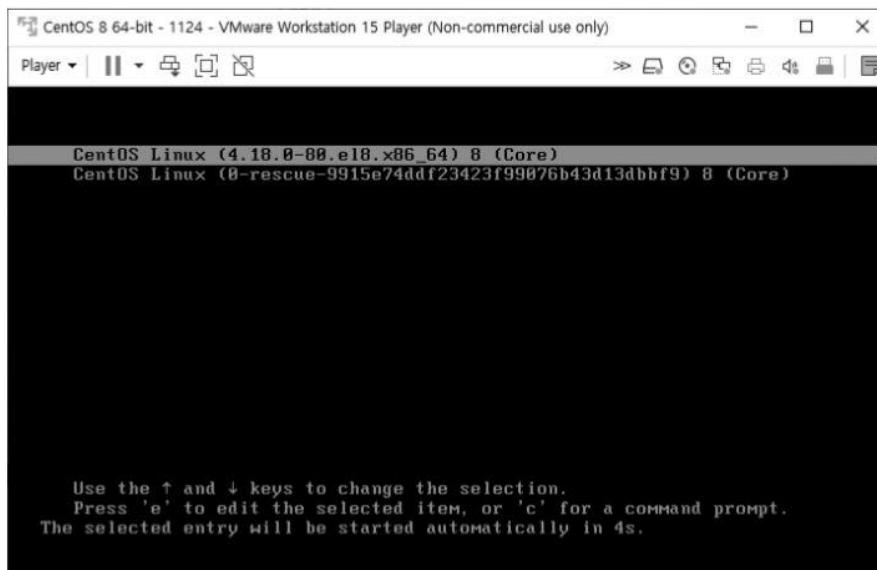
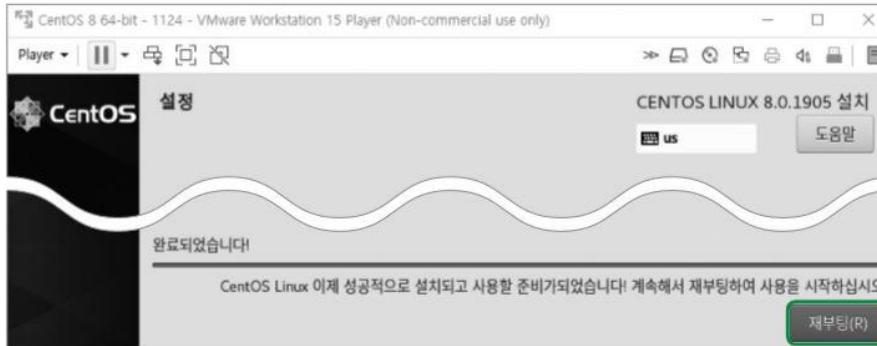


02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- [따라해보기] CentOS 설치하기

⑪ CentOS 부팅하기

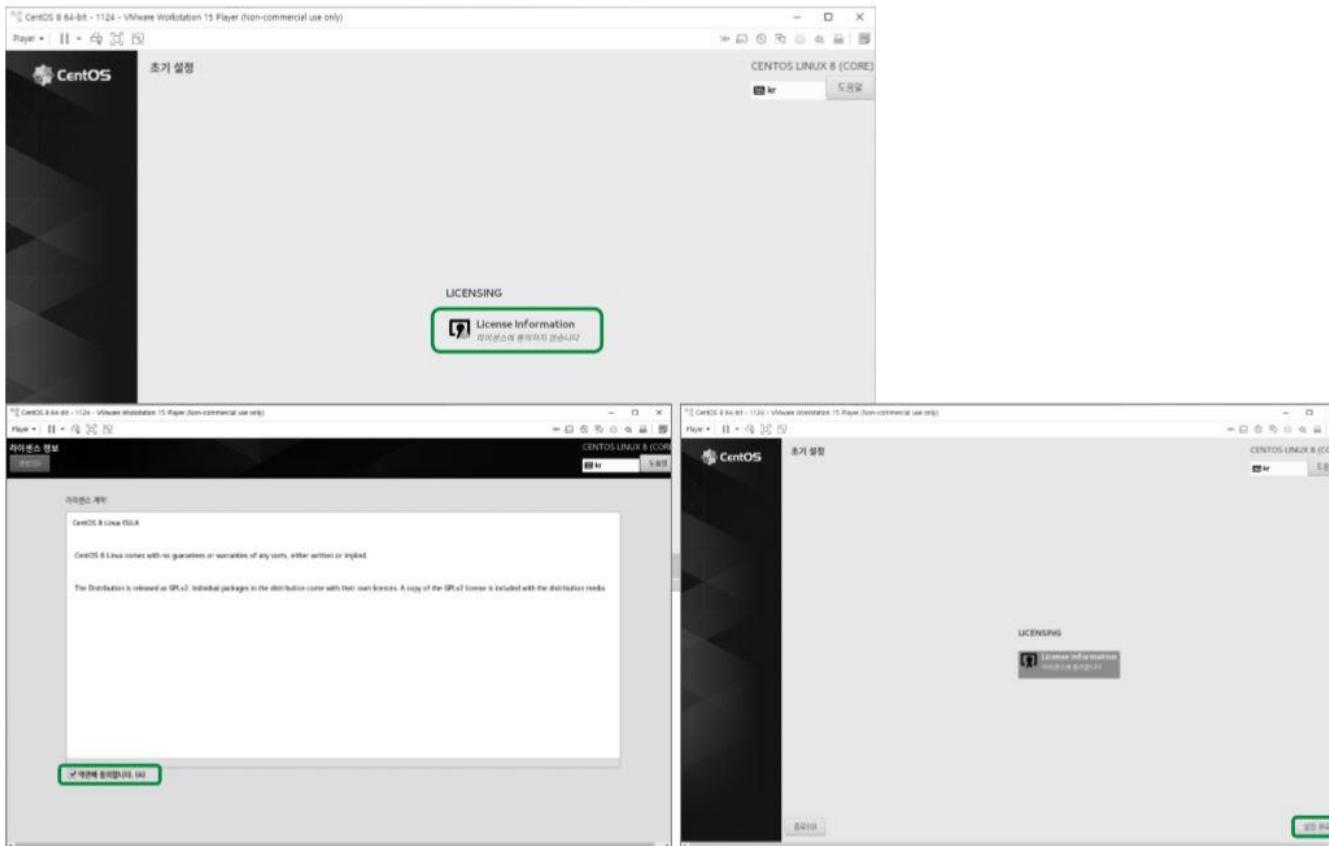


02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- [따라해보기] CentOS 설치하기

⑫ 라이선스 동의하기

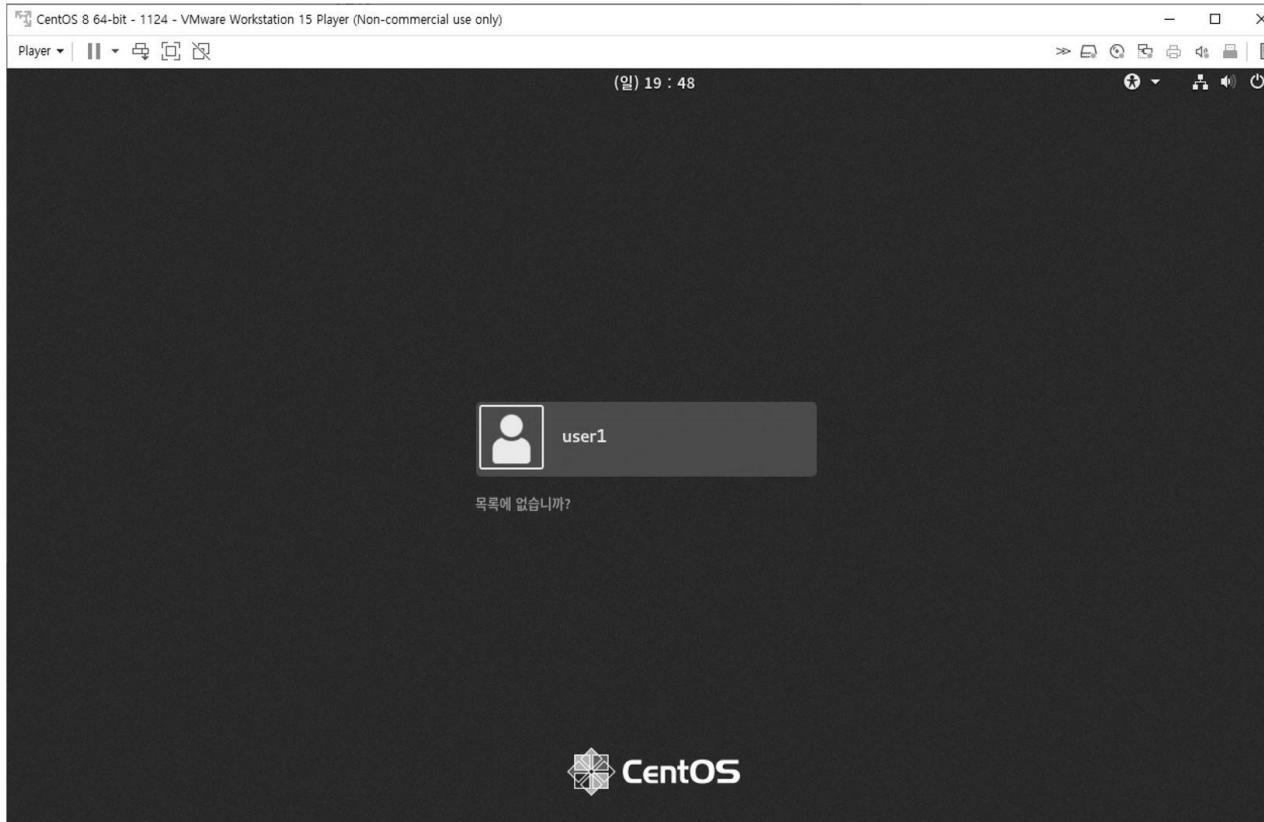


02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- [따라해보기] CentOS 설치하기

⑬ 로그인하기

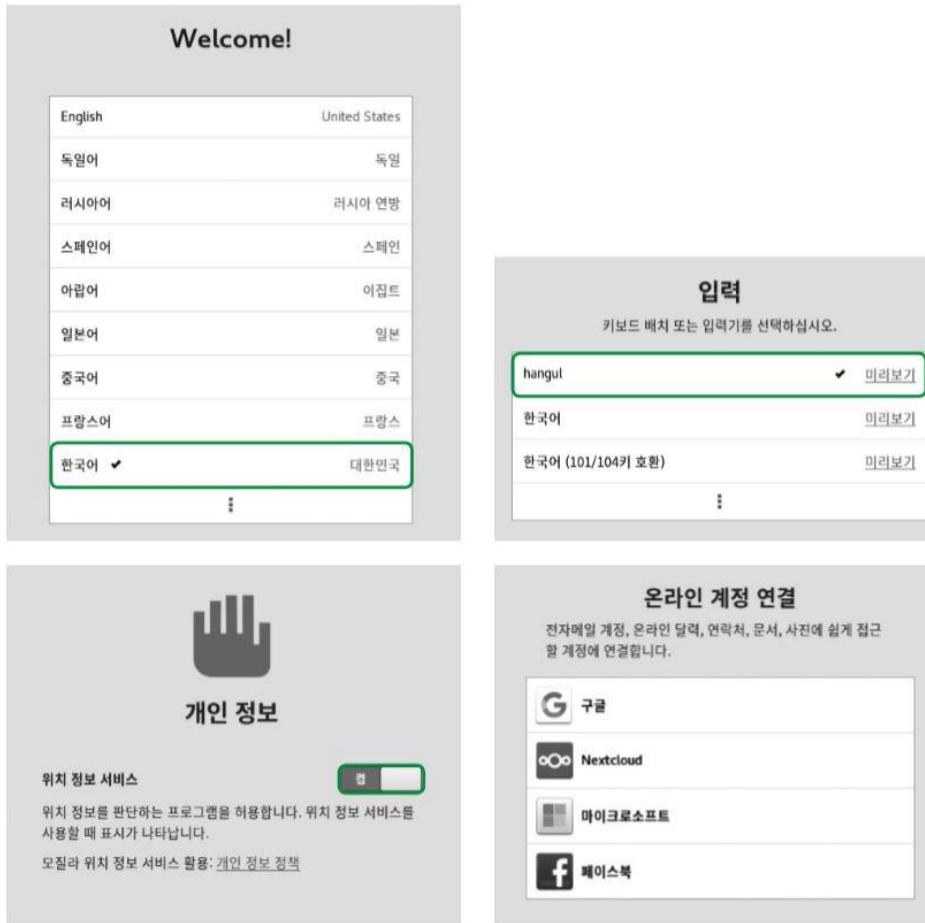


02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- [따라해보기] CentOS 설치하기

⑯ 그놈 초기 설정하기

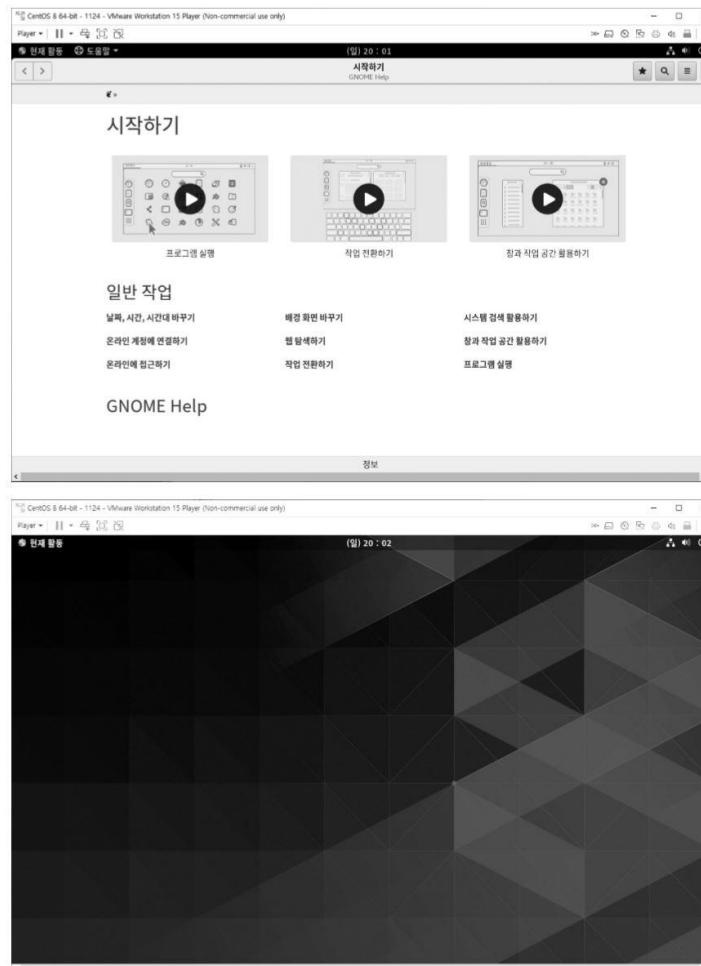


02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- [따라해보기] CentOS 설치하기

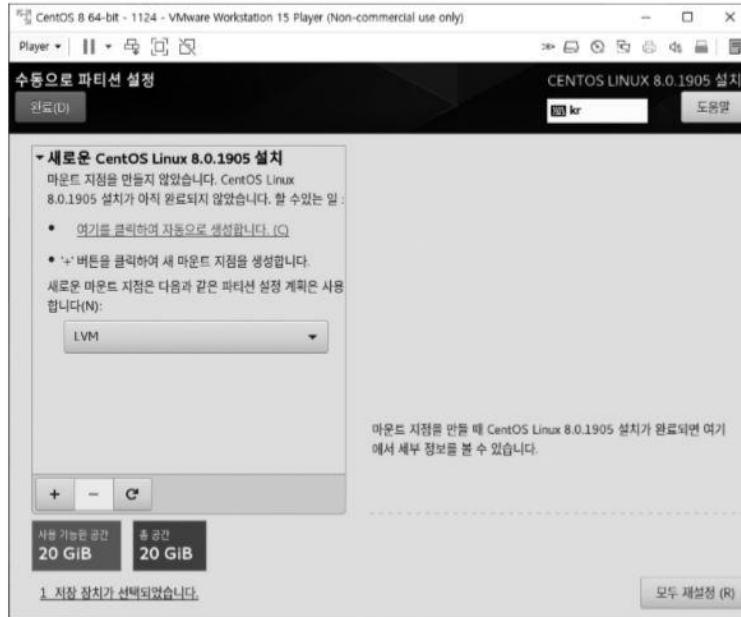
⑯ CentOS 8 초기 화면 확인하기



02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- 파티션을 수동으로 설정하기



02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- 파티션을 수동으로 설정하기

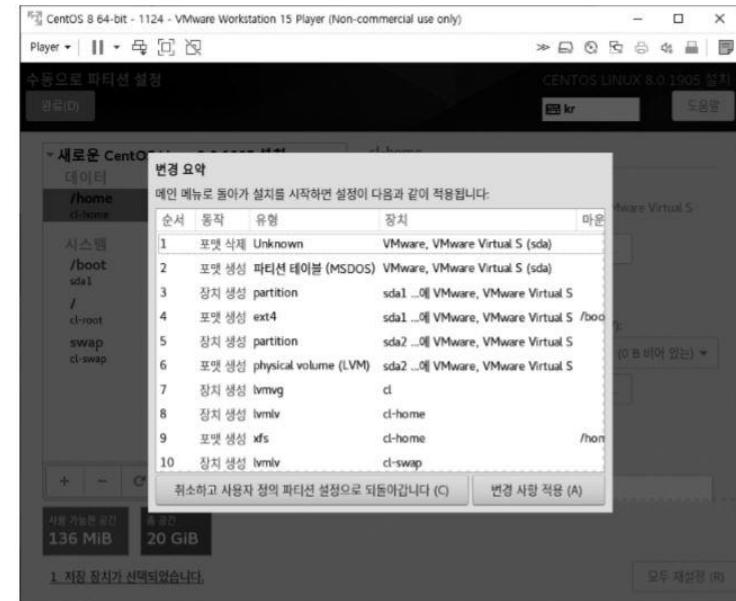
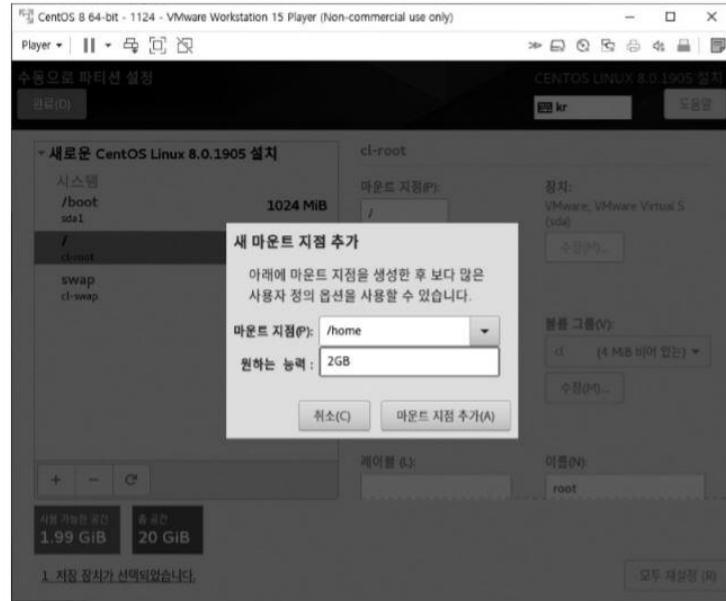
표 1-2 리눅스 파티션의 구분 예

마운트 포인트		내용
필수	/	root 파티션으로 시스템 설정 등 주요 내용이 저장된다.
	/boot	boot 파티션으로 리눅스 부팅 커널이 저장된다. 대략 500MB를 할당한다.
	swap	메모리(RAM)가 부족할 때 사용되는 영역으로 대개 RAM 크기의 2배 정도로 설정한다.
선택	/usr	명령어, 응용 프로그램 등이 주로 저장된다.
	/var	로그 파일 등이 주로 저장된다.
	/tmp	시스템 사용 중에 발생하는 임시 파일이 저장된다.
	/home	사용자 험 디렉터리가 위치한다.

02. 리눅스 실습 환경 구축

■ 리눅스의 설치

- 파티션을 수동으로 설정하기



03. 리눅스 윈도 기본 사용법

■ CentOS 윈도 환경의 구성

- CentOS 윈도 환경의 구성요소



그림 1-6 CentOS 윈도 상단의 구성 요소

03. 리눅스 원도 기본 사용법

■ CentOS 원도 환경의 구성

■ 현재 활동

- 자주 사용 하는 프로그램의 아이콘과 전체 프로그램을 확인할 수 있는 프로그램 표시 아이콘으로 구성
- 실행 중인 프로그램이 있을 경우 해당 프로그램의 아이콘이 추가로 표시

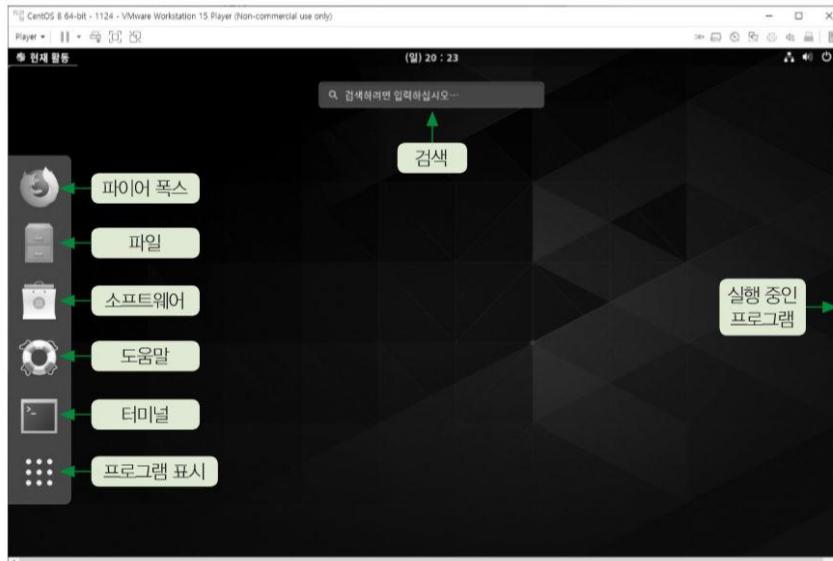


그림 1-7 현재 활동

03. 리눅스 원도 기본 사용법

■ CentOS 원도 환경의 구성

- 날짜와 시간 및 알림사항 관리
 - 날짜와 시간을 설정하고 알림 사항을 확인할 수 있음

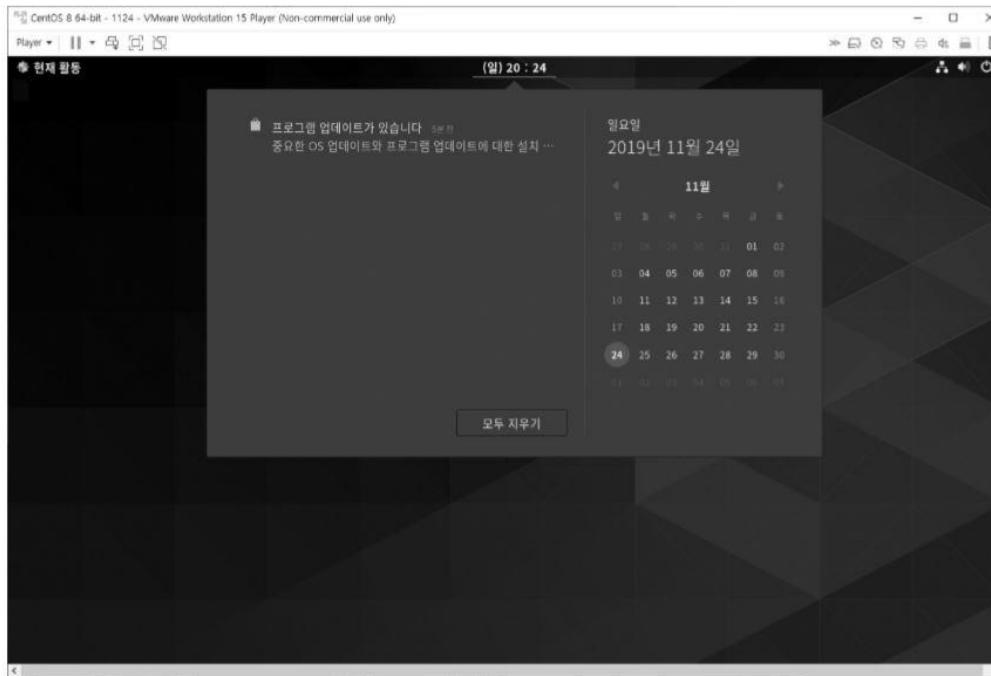


그림 1-8 날짜와 시간 설정 및 일정 확인

03. 리눅스 윈도 기본 사용법

■ CentOS 윈도 환경의 구성

- 기타 설정
 - 소리 설정, 네트워크 설정, 사용자 설정, 시스템 설정, 시스템 종료 등의 메뉴

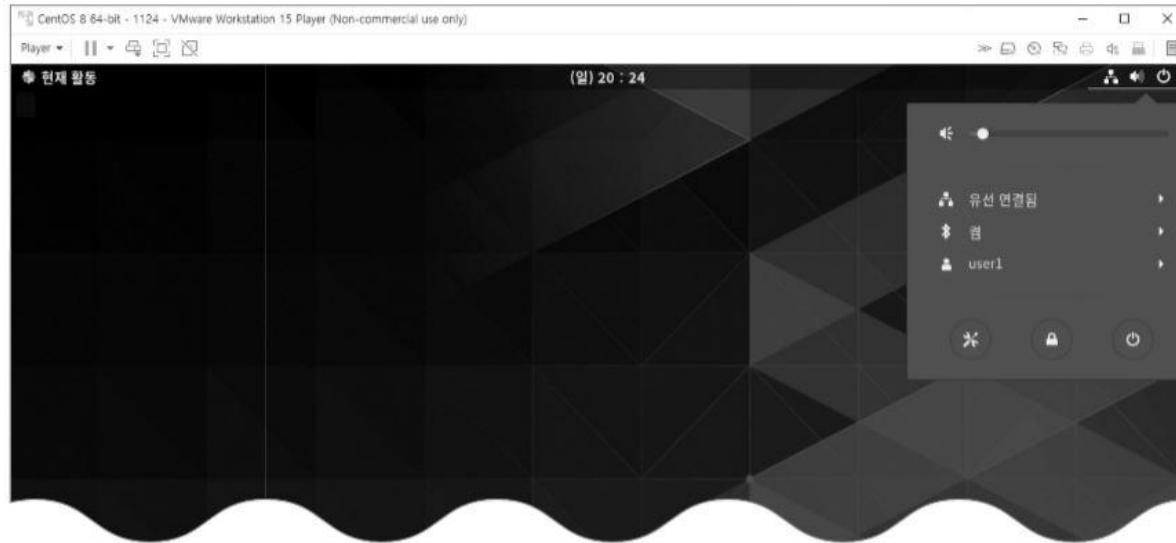


그림 1-9 소리, 블루투스, 네트워크, 사용자 설정 메뉴

03. 리눅스 원도 기본 사용법

■ [따라해보기] 한글 설정 입력하기

① 한글 입력기 패키지 설치하기

- 일반 사용자 계정이 아니라 root 계정에서 해야 함
 - 터미널에서 root 계정으로 전환하여 한글 입력기 패키지를 설치한 후 CentOS를 재부팅

```
[user1@localhost ~]$ su -                                → root 계정으로 전환  
암호:                                              → root 계정의 암호 입력  
[root@localhost ~]# yum -y install ibus-hangul      → 한글 입력기 설치  
(생략)  
설치됨:
```

완료되었습니다!

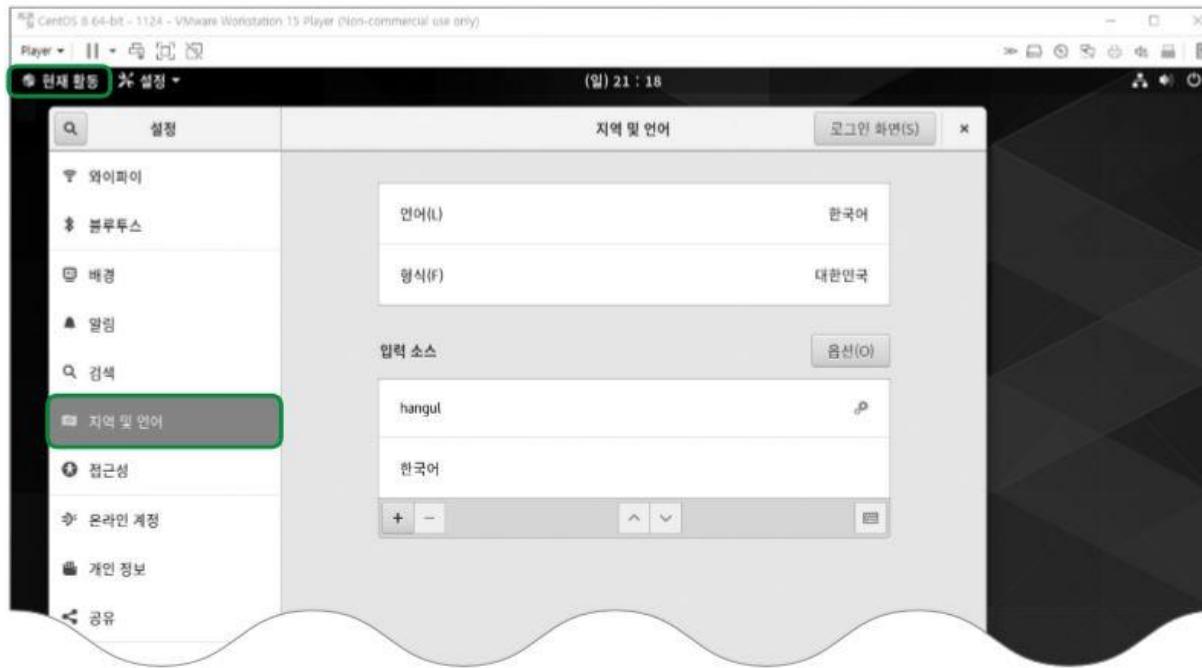
```
[root@localhost ~]# exit  
logout
```

→ root 계정에서 나가기

03. 리눅스 윈도 기본 사용법

■ [따라해보기] 한글 설정 입력하기

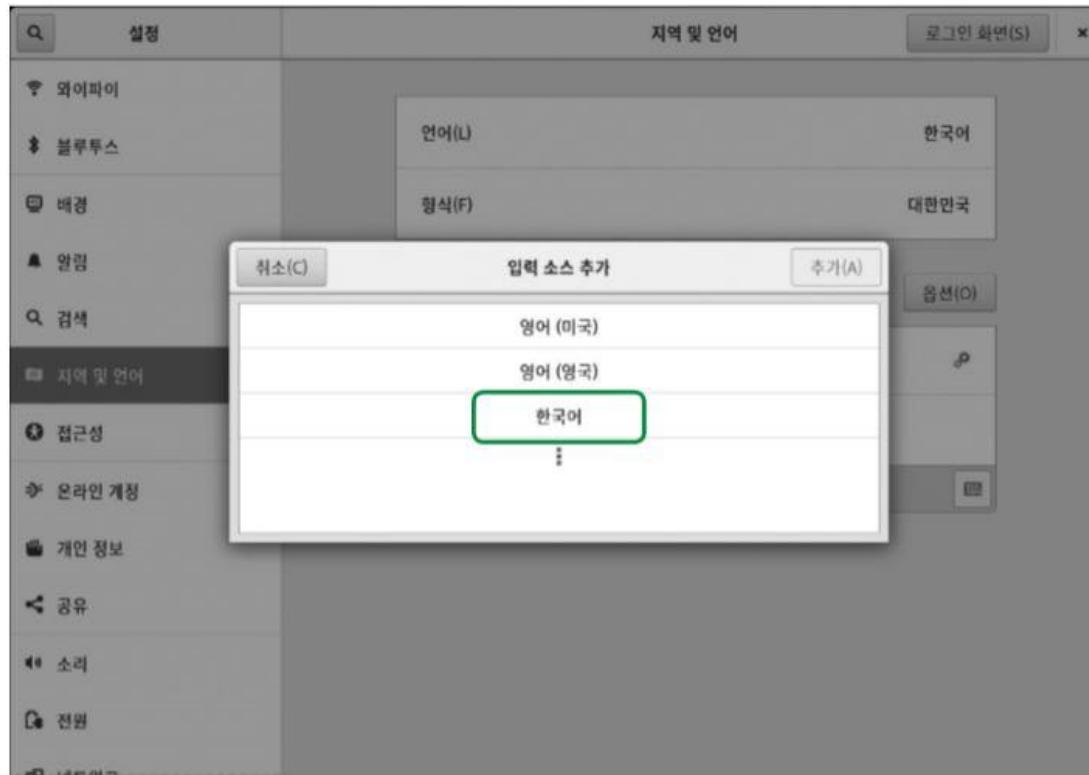
- ② 지역 및 언어 설정하기



03. 리눅스 윈도 기본 사용법

■ [따라해보기] 한글 설정 입력하기

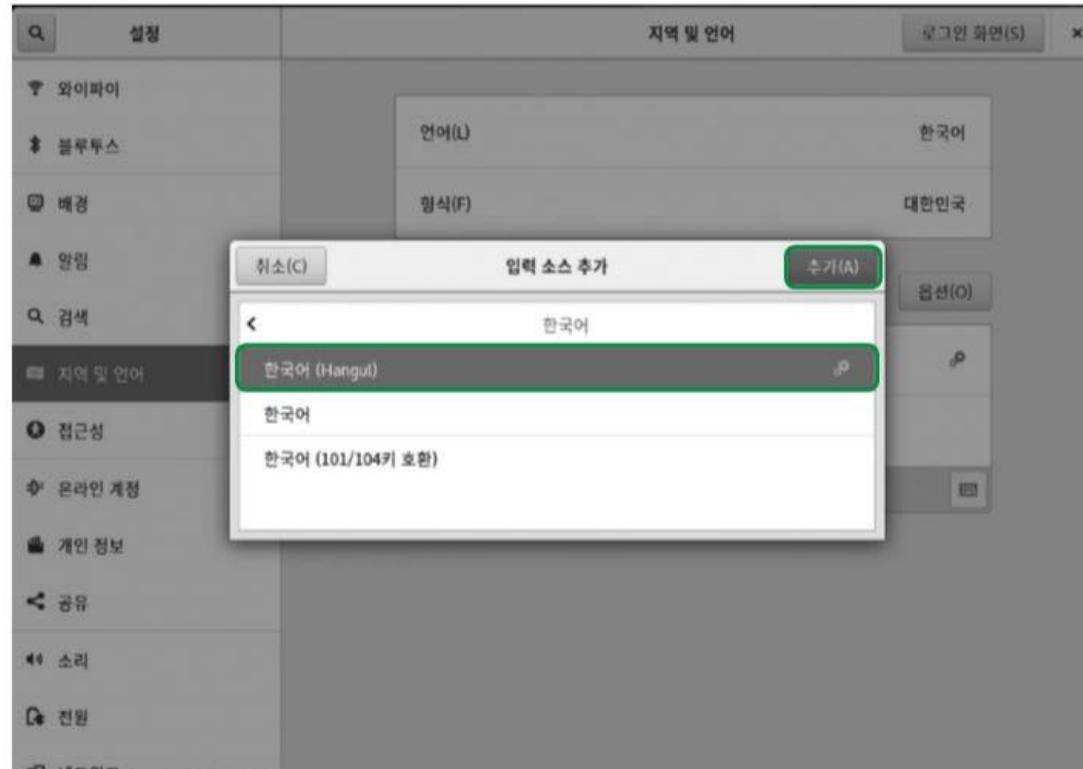
③ 입력 소스 추가하기



03. 리눅스 윈도 기본 사용법

■ [따라해보기] 한글 설정 입력하기

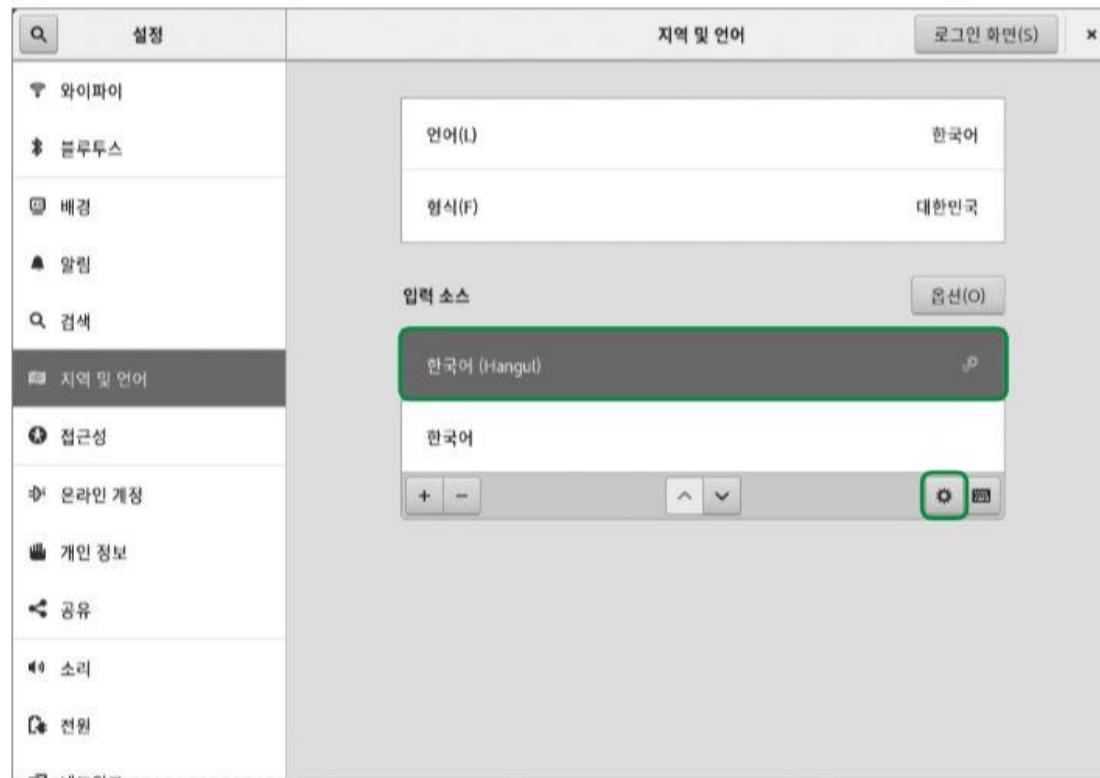
- ④ 한국어 선택하기



03. 리눅스 윈도 기본 사용법

■ [따라해보기] 한글 설정 입력하기

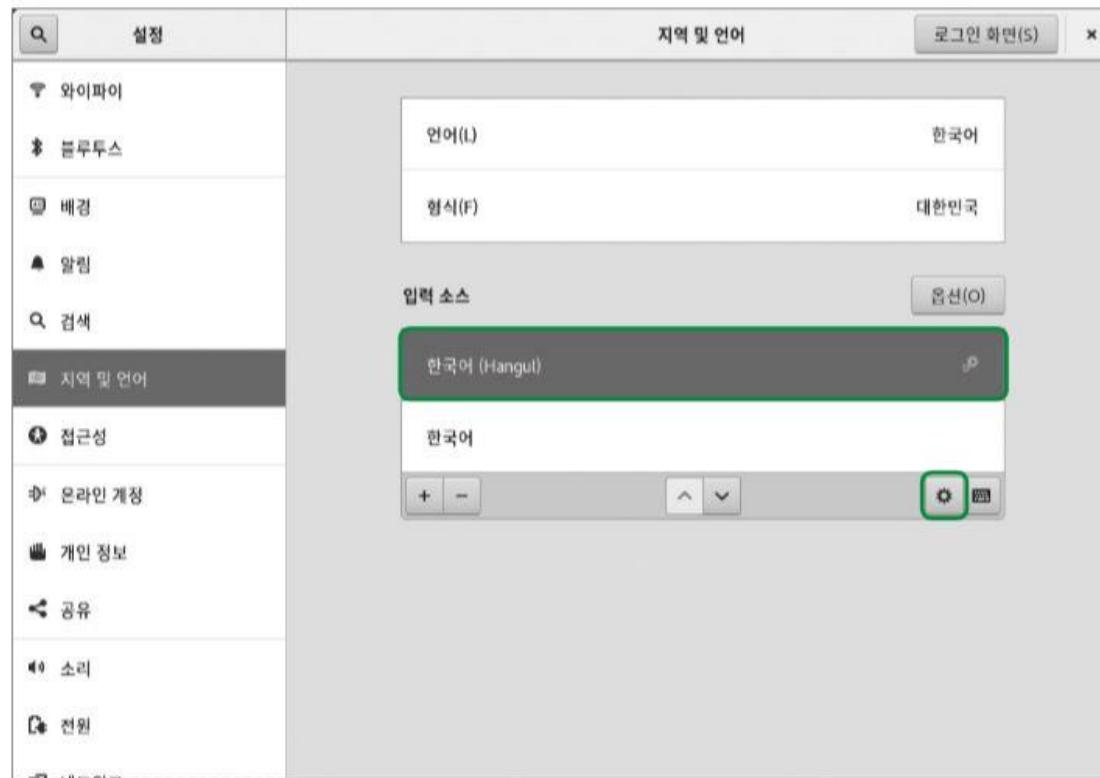
⑤ 한국어 설정 선택하기



03. 리눅스 윈도 기본 사용법

■ [따라해보기] 한글 설정 입력하기

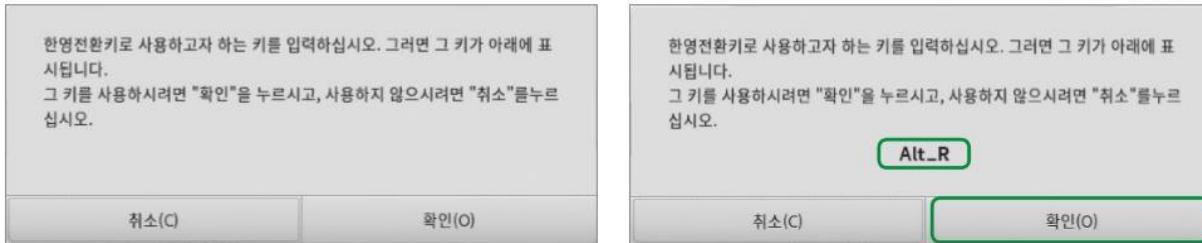
⑥ IBus 한글 설정하기



03. 리눅스 윈도 기본 사용법

■ [따라해보기] 한글 설정 입력하기

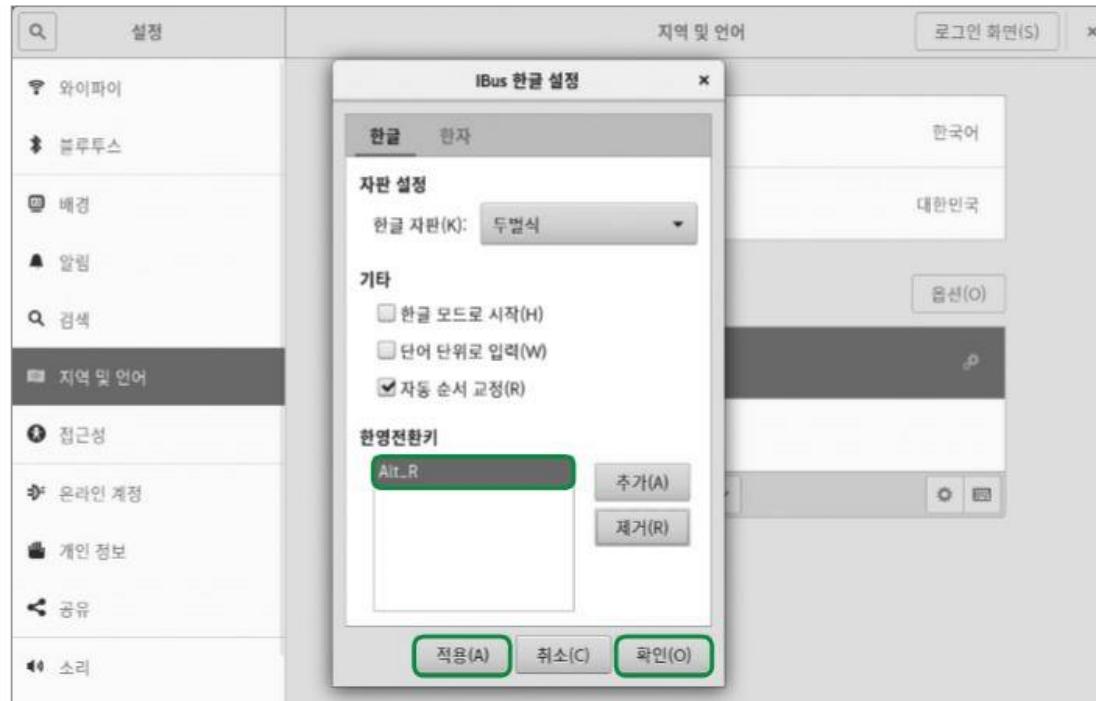
⑦ 한영 전환키 선택하기



03. 리눅스 윈도 기본 사용법

■ [따라해보기] 한글 설정 입력하기

⑧ 기존 한영 전환키 삭제하기



03. 리눅스 윈도 기본 사용법

■ [따라해보기] 한글 설정 입력하기

⑨ 기존 한국어 삭제하기



03. 리눅스 윈도 기본 사용법

■ [따라해보기] 한글 설정 입력하기

⑩ 한글 입력 확인하기

- 한영 전환키(Alt_R)로 한글과 영문을 전환하며 사용할 수 있음



03. 리눅스 원도 기본 사용법

■ [따라해보기] CentOS 원도의 사용

⑪ 파일

- 원도 탐색기처럼 파일을 관리하는 응용 프로그램



그림 1-10 파일 관리 창

03. 리눅스 윈도 기본 사용법

■ [따라해보기] 텍스트 편집기로 문서 열기

① 텍스트 편집기 실행하기

- [현재 활동]-[프로그램 표시]-[텍스트 편집기]를 선택

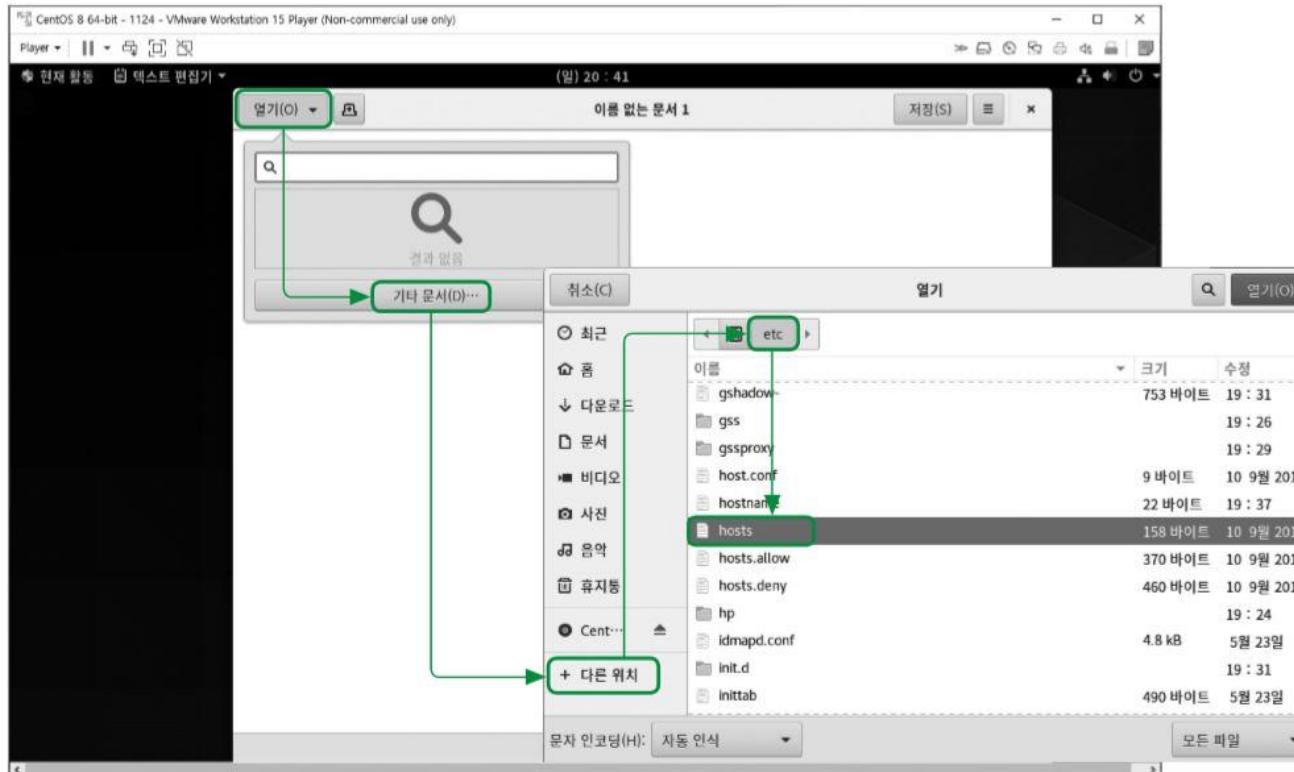


03. 리눅스 윈도 기본 사용법

■ [따라해보기] 텍스트 편집기로 문서 열기

② 텍스트 편집기에서 디렉터리 선택하기

- [열기]-[기타 문서]-[다른 위치]-[컴퓨터]-[etc]를 선택

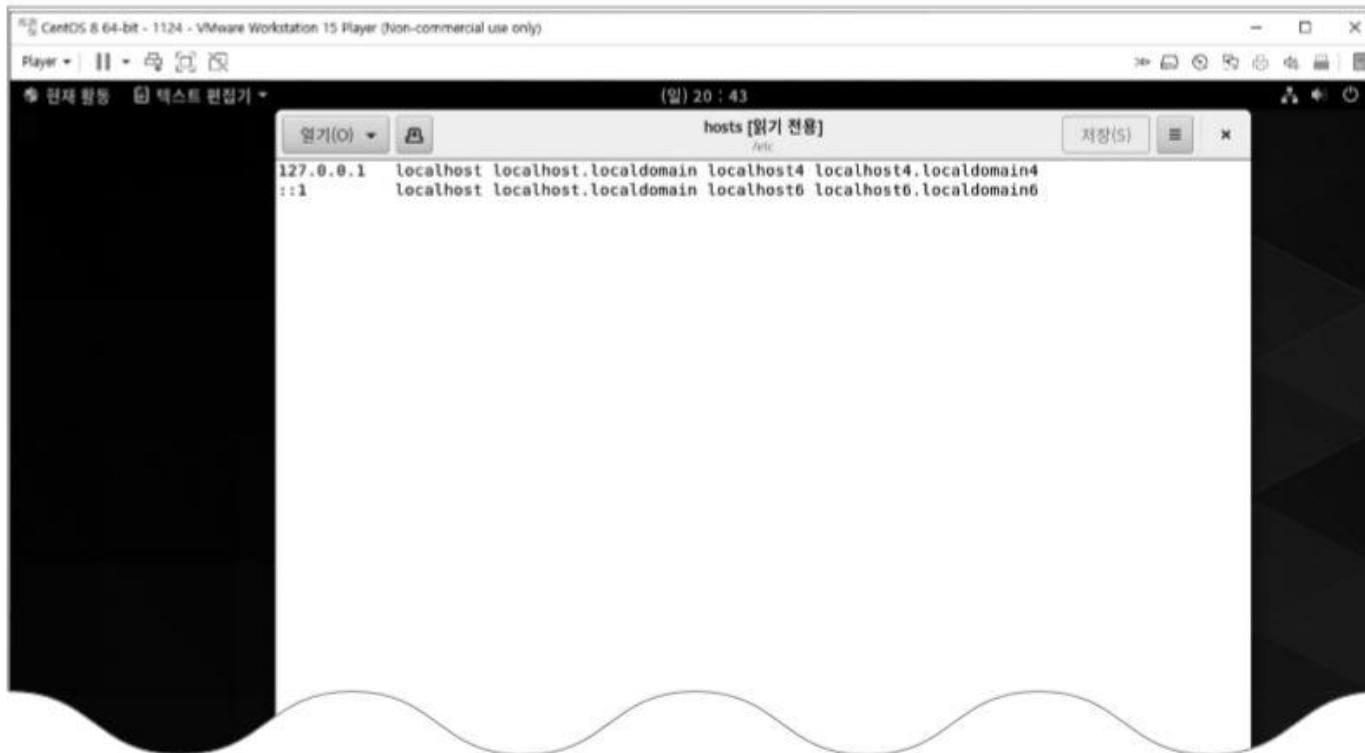


03. 리눅스 원도 기본 사용법

■ [따라해보기] 텍스트 편집기로 문서 열기

③ 텍스트 편집기에서 파일 열기

- etc 디렉터리에서 hosts 파일을 선택하고 [열기(O)]를 클릭하여 파일 실행



04. 리눅스 명령 사용법

■ 터미널의 시작과 종료

- [현재 활동]-[터미널]을 선택하여 동작

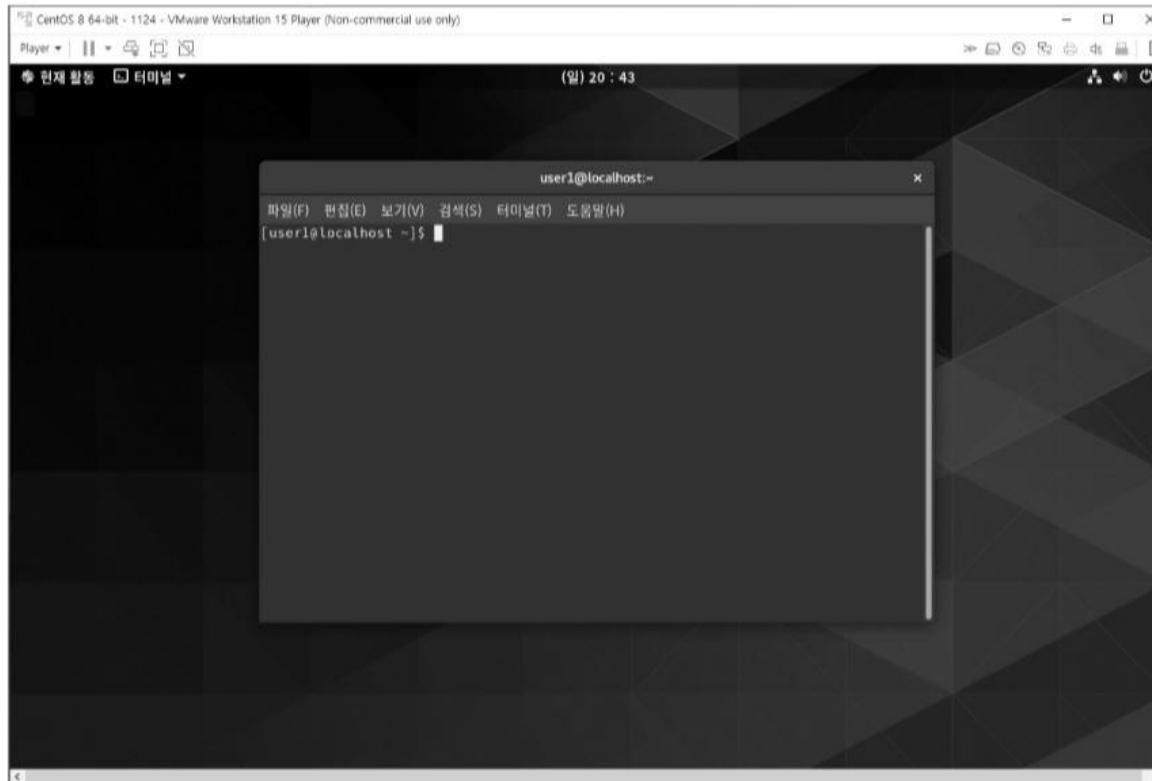


그림 1-12 터미널 창

04. 리눅스 명령 사용법

■ 터미널의 시작과 종료

- 종료 방법에는 두가지가 있음
- 마우스로 터미널 창의 오른쪽 위에 있는 × 를 클릭해도 됨
- exit 명령을 사용한 접속해제

```
[user1@localhost ~]$ exit
```

- [ctrl]+d를 사용한 접속해제

```
[user1@localhost ~]$ ^d → [Ctrl]+d
```

04. 리눅스 명령 사용법

■ 프롬프트 기호와 홈 디렉터리

■ 프롬프트

- 사용자의 명령 입력을 기다리는 표시
- 사용하는 셸에 따라 다르게 나타남
 - (배시셀: \$로 표시, 시스템 관리자의 경우에는 #로 표시)
- 프롬프트는 사용자가 다른 모양으로 바꿀 수 있음

■ 홈 디렉터리

- user1@localhost:~ : user1은 사용자 계정 이름, localhost는 호스트 이름, ~는 user1 사용자의 홈 디렉터리
- 사용자의 홈 디렉터리는 사용자 계정을 등록할 때 결정
- 홈 디렉터리 아래에 자신이 원하는 파일이나 디렉터리를 생성하며 작업 할 수 있음

04. 리눅스 명령 사용법

■ 명령행 편집

■ 문자 지우기

- 리눅스의 명령행에서 문자를 지울 때는 백스페이스[<-] 나 [delete] 를 사용
- 리눅스에 따라 둘 중 하나가 기본 값으로 설정, CentOS의 경우는[<-]

■ 단어 지우기

- 리눅스의 명령 행에서 단어를 지울 때는 [ctrl]+ w를 사용
- 이때 단어는 공백으로 구분

```
[user1@localhost ~]$ linux CentOS ➔ CentOS를 지우려면 Ctrl+w를 누른다.  
[user1@localhost ~]$ linux
```

■ 행 지우기

- 리눅스의 명령 행에서 행을 지울 때는 [ctrl]+ u를 사용
- 해당행의 내용을 모두 삭제 할 수 있음

```
[user1@localhost ~]$ linux CentOS ➔ 모두 지우려면 Ctrl+u를 누른다.  
[user1@localhost ~]$
```

04. 리눅스 명령 사용법

■ 명령의 구조와 사용

명령의 구조

- **형식** 명령 [옵션] [인자]
- **명령** 리눅스를 사용하기 위해 사용자가 입력하는 다양한 명령은 date, man, ls, cp, mv 등 수백 가지가 있다.
- **옵션** 옵션을 사용하여 명령의 세부 기능을 선택할 수 있다. 리눅스의 기능을 풍부하게 하는 중요한 특징으로 명령에 따라 있는 경우도 있고 없는 경우도 있다. 옵션은 - 나 -- 기호로 시작하며 영문 소문자나 대문자로 구성된다. 명령에 따라 어떤 옵션이 있고 그 기능이 무엇인지는 해당 명령의 사용법을 참조해야 한다.
- **인자** 인자는 명령으로 전달되는 값이며 주로 파일명이나 디렉터리명이 사용된다. 명령에 따라 없을 수도 있고 각 명령에 필요한 인자도 각기 다르므로 사용법을 참조해야 한다.

04. 리눅스 명령 사용법

■ 명령의 구조와 사용

- 명령만 사용한 경우 (ls 명령)

```
[user1@localhost ~]$ ls  
공개  다운로드  문서  바탕  화면  비디오  사진  서식  음악
```

- 명령과 옵션을 사용한 경우 (ls 명령+'-a')

```
[user1@localhost ~]$ ls -a  
.           .bash_history  .bashrc  .esd_auth  .pki      문서      사진  
..          .bash_logout   .cache   .local    공개      바탕  화면  서식  
.ICEauthority  .bash_profile  .config  .mozilla  다운로드  비디오  음악
```

- 명령과 인자를 사용한 경우 (ls 명령+ /tmp)

```
[user1@localhost ~]$ ls /tmp  
VMwareDnD  
anaconda.log  
dbus.log  
ifcfg.log  
(생략)
```

04. 리눅스 명령 사용법

■ 명령의 구조와 사용

- 명령, 옵션, 인자를 사용한 경우 (ls 명령 + '-a' + /tmp)

```
[user1@localhost ~]$ ls -a /tmp  
.  
..  
.ICE-unix  
.Test-unix  
.X0-lock  
.X1024-lock  
.X11-unix  
(중략)  
.viminfo  
VMwareDnD  
anaconda.log  
(생략)
```

04. 리눅스 명령 사용법

■ 기초 명령의 사용법

- date 명령
 - 현재 날짜와 시간을 출력

date

- **기능** 날짜와 시간을 출력한다.
- **형식** date

```
[user1@localhost ~]$ date  
2019. 11. 24. (일) 21:07:19 KST
```

- clear 명령
 - 현재 보이는 화면이 깨끗이 지워지고 커서가 화면의 왼쪽 상단으로 이동
 - 옵션이나 인자가 없음

clear

- **기능** 화면을 지운다.
- **형식** clear

```
[user1@localhost ~]$ clear
```

04. 리눅스 명령 사용법

■ 기초 명령의 사용법

- man 명령
 - 리눅스가 제공하는 각종 명령의 사용법을 자세히 보여줌
 - 명령의 옵션이나 인자 등 명령 사용법을 자세히 알고 싶을 때 매우 유용

man

- **기능** 명령 사용 방법을 화면에 출력한다.
- **형식** man 명령

- 예시) clear 명령의 자세한 사용법

```
[user1@localhost ~]$ man clear
```

04. 리눅스 명령 사용법

■ 기초 명령의 사용법

▪ man 명령

- 결과 출력

```
lear(1)                               General Commands Manual           clear(1)

NAME                                → 명령에 의한 간략한 설명
clear - clear the terminal screen

SYNOPSIS                            → 명령의 사용 방법 요약
clear [-Ttype] [-V] [-x]

DESCRIPTION                          → 명령에 대한 상세한 기술
clear clears your screen if this is possible, including its scrollback
buffer (if the extended "E3" capability is defined). clear looks in
the environment for the terminal type given by the environment variable
TERM, and then in the terminfo database to determine how to clear the
screen.

(생략)

PORTABILITY                         → 명령의 호환성과 관련된 사항
Neither IEEE Std 1003.1/The Open Group Base Specifications Issue 7
(POSIX.1-2008) nor X/Open Curses Issue 7 documents tset or reset.

The latter documents tput, which could be used to replace this utility
either via a shell script or by an alias (such as a symbolic link) to
run tput as clear.

SEE ALSO                             → 명령과 관련된 참고 사항
tput(1), terminfo(5)

This describes ncurses version 6.1 (patch 20180224).
                                         clear(1)
Manual page clear(1) line 78/100 (END) (press h for help or q to quit)
```

04. 리눅스 명령 사용법

■ 기초 명령의 사용법

- passwd 명령
 - 사용자 계정의 비밀번호 변경에 사용

passwd

- **기능** 사용자 계정의 비밀번호를 변경한다.
- **형식** passwd [인자]

```
[user1@localhost ~]$ passwd
user1 사용자의 비밀번호 변경 중
Current password:
새 암호:
새 암호 재입력:
passwd: 모든 인증 토큰이 성공적으로 업데이트 되었습니다.
```



CentOS 리눅스

시스템 & 네트워크

Chapter 02. 디렉터리와 파일 사용법

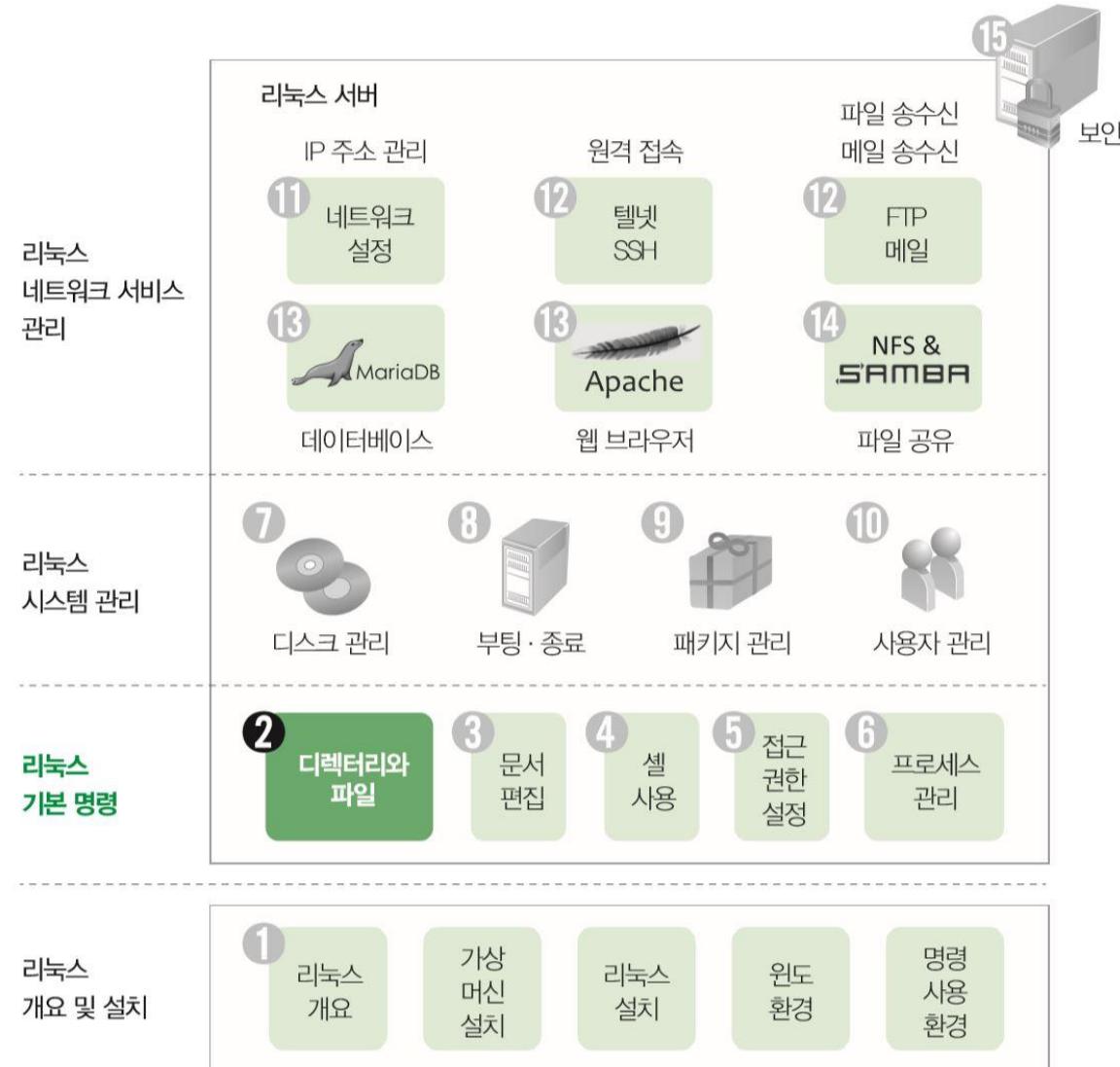
목차

- 00. 개요
- 01. 리눅스의 파일과 디렉터리
- 02. 디렉터리 관련 명령
- 03. 파일 관련 명령

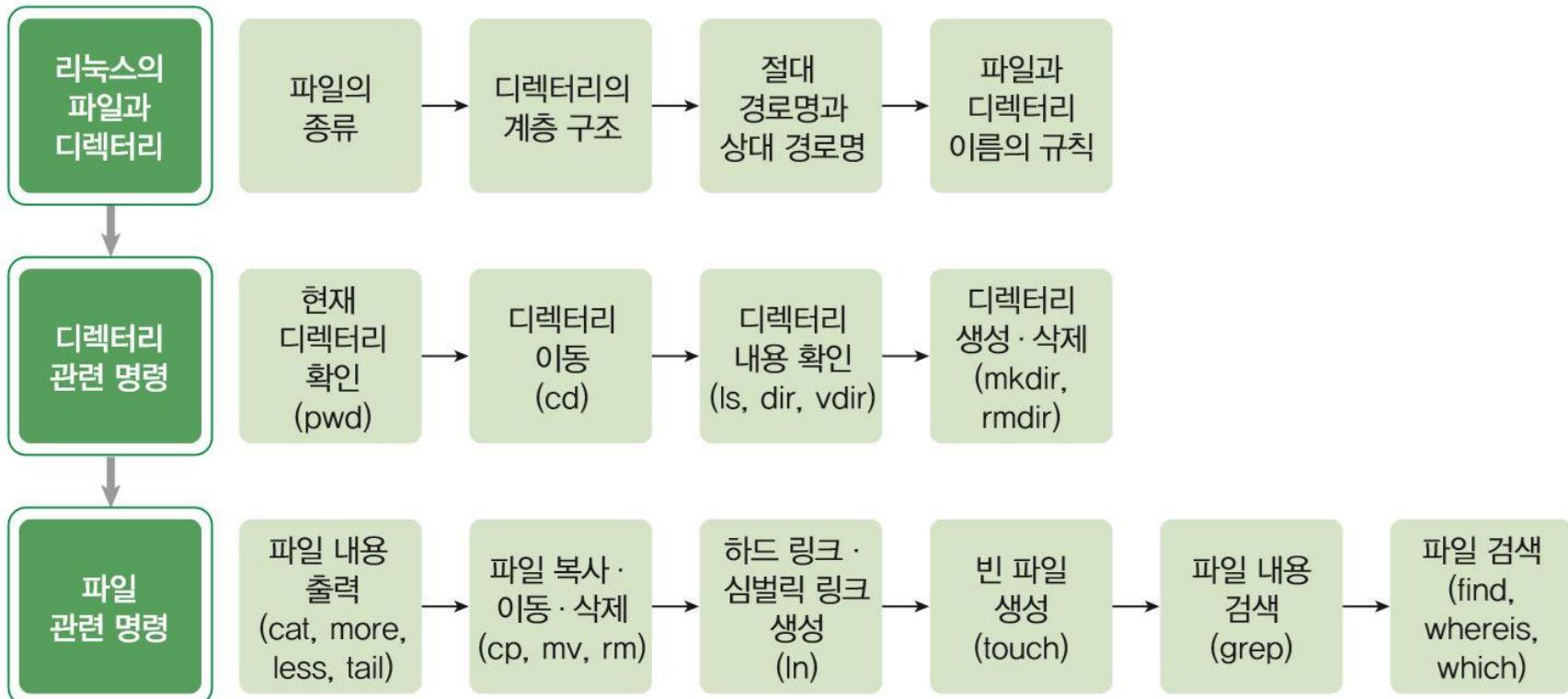
학습목표

- 리눅스 파일의 종류와 특징을 설명할 수 있다.
- 디렉터리 계층 구조를 보고 절대 경로명과 상대 경로명을 작성할 수 있다.
- 디렉터리를 이동하고, 디렉터리의 내용을 확인할 수 있다.
- 디렉터리를 만들고 삭제할 수 있다.
- 다양한 명령으로 파일 내용을 확인할 수 있다.
- 파일을 복사/이동/삭제할 수 있다.
- 파일 링크의 특징을 설명하고, 하드 링크와 심볼릭 링크를 만들 수 있다.
- 파일의 내용과 위치를 검색할 수 있다.

00. 개요



00. 개요



01. 리눅스의 파일과 디렉터리

■ 파일의 종류

■ 파일

- 관련 있는 정보들의 집합
- 리눅스는 파일을 효과적으로 관리하기 위해 디렉터리를 사용
- 파일 시스템 : 디렉터리와 파일로 구성된 전체 집합
- 리눅스에서 파일은 사용 목적에 따라 일반 파일, 디렉터리, 심볼릭 링크, 장치 파일로 구분

■ 일반 파일

- 데이터를 저장하는 데 주로 사용
- 리눅스에서 사용하는 대부분의 파일은 일반 파일에 해당
- 바이너리 파일 : 실행 파일이나 이미지 파일의 경우 바이너리 형태로 데이터가 저장되어 사용

■ 디렉터리

- 리눅스에서는 디렉터리도 파일로 취급
- 디렉터리 파일에는 해당 디렉터리에 저장된 파일이나 하위 디렉터리에 대한 정보가 저장

■ 심볼릭 링크

- 원본 파일을 대신하여 다른 이름으로 파일명을 지정한 것(윈도의 바로 가기와 유사)

01. 리눅스의 파일과 디렉터리

■ 파일의 종류

- 장치 파일
 - 리눅스 시스템에 부착된 장치들을 관리하기 위한 특수 파일
 - ex) 하드디스크나 키보드 같은 각종 장치
- 파일의 종류를 확인하는 file 명령

```
[user1@localhost ~]$ file .bash_profile
.bash_profile: ASCII text
[user1@localhost ~]$ file 다운로드
다운로드: directory
```

```
[user1@localhost ~]$ file /usr/bin/pigz
/usr/bin/pigz: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
(생략)
```

01. 리눅스의 파일과 디렉터리

■ 디렉터리의 계층 구조

▪ 트리 구조

- 리눅스에서 파일을 효율적으로 관리하기 위해 디렉터리를 계층적으로 구성
- 모든 디렉터리의 출발점은 루트(root, 뿌리) 디렉터리이며, 빗금(/, 슬래시)으로 표시
- 그림으로 나타낼 때 루트 디렉터리가 맨 위에서 출발

▪ 루트 디렉터리와 서브 디렉터리

- 서브 디렉터리(하위 디렉터리) : 최상단 루트 디렉터리(/) 아래의 디렉터리 (etc, usr, home, tmp)
- 부모 디렉터리(상위 디렉터리) : 서브 디렉터리 위에 자신을 포함하고 있는 디렉터리 ('.'으로 표시)
- 루트 디렉터리를 제외하고 모든 디렉터리에는 부모 디렉터리가 있음

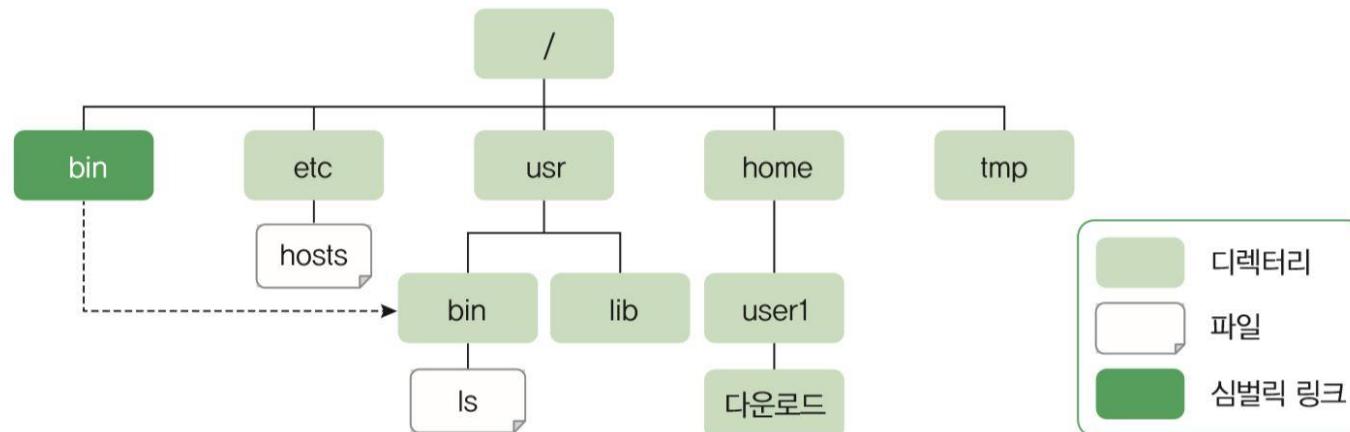


그림 2-1 디렉터리 계층 구조의 예

01. 리눅스의 파일과 디렉터리

■ 디렉터리의 계층 구조

- 루트 디렉터리와 서브 디렉터리
 - 디렉터리 이름의 끝에 붙은 /는 해당 파일이 디렉터리임을, @(앳)은 심볼릭 링크를 의미

```
[user1@localhost ~]$ ls -F /  
bin@ dev/ home/ lib64@ mnt/ proc/ run/ srv/ tmp/ var/  
boot/ etc/ lib@ media/ opt/ / sbin@ sys/ usr/
```

표 2-1 디렉터리의 주요 기능

디렉터리	기능
dev	장치 파일이 담긴 디렉터리다.
home	사용자 홈 디렉터리가 생성되는 디렉터리다.
media	DVD/CD나 USB 같은 외부 장치를 연결(마운트라고 함)하는 디렉터리다.
opt	추가 패키지가 설치되는 디렉터리다.
root	root 계정의 홈 디렉터리다. 루트(/) 디렉터리와 다른 것이므로 혼동하지 않도록 한다.
sys	리눅스 커널과 관련된 파일이 있는 디렉터리다.
usr	기본 실행 파일과 라이브러리 파일, 헤더 파일 등 많은 파일이 있다. 참고로 usr는 'Unix System Resource'의 약자이다.
boot	부팅에 필요한 커널 파일을 가지고 있다.
etc	리눅스 설정을 위한 각종 파일을 가지고 있다.
lost+found	파일 시스템에 문제가 발생하여 복구할 경우, 문제가 되는 파일이 저장되는 디렉터리로 보통은 비어 있다.

01. 리눅스의 파일과 디렉터리

■ 디렉터리의 계층 구조

- 루트 디렉터리와 서브 디렉터리
 - 디렉터리 이름의 끝에 붙은 /는 해당 파일이 디렉터리임을, @(앳)은 심볼릭 링크를 의미

디렉터리	기능
mnt	파일 시스템을 임시로 마운트하는 디렉터리다.
proc	프로세스 정보 등 커널 관련 정보가 저장되는 디렉터리다.
run	실행 중인 서비스와 관련된 파일이 저장된다.
srv	FTP나 Web 등 시스템에서 제공하는 서비스의 데이터가 저장된다.
tmp	시스템 사용 중에 발생하는 임시 데이터가 저장된다. 이 디렉터리에 있는 파일은 재시작하면 모두 삭제된다.
var	시스템 운영 중에 발생하는 데이터나 로그 등 내용이 자주 바뀌는 파일이 주로 저장된다.

01. 리눅스의 파일과 디렉터리

■ 디렉터리의 계층 구조

▪ 작업 디렉터리

- 현재 사용 중인 디렉터리
- 현재 디렉터리라고도 함
- .(마침표)로 표시하며, 작업 디렉터리의 위치는 pwd 명령으로 확인 할 수 있음

▪ 홈 디렉터리

- 각 사용자에게 할당된 디렉터리로 처음 사용자 계정을 만들 때 지정
- 사용자는 자신의 홈 디렉터리 아래에 파일이나 서브 디렉터리를 생성하며 작업할 수 있음
- ~(물결표)로 표시
 - ex) user1 계정의 홈 디렉터리를 나타내려면 '~user1'

01. 리눅스의 파일과 디렉터리

■ 절대 경로명과 상대 경로명

▪ 경로명

- 파일 시스템에서 디렉터리 계층 구조에 있는 특정 파일이나 디렉터리의 위치 표시 (구분자 : /)
- 가장 앞에 있는 / 는 루트 디렉터리를 뜻하지만 경로명 중간에 있는 / 는 구분자
 - ex) /usr/bin/ls에서 맨 앞의 /는 루트 디렉터리를 의미, 중간에 있는 / 두 개는 디렉터리명과 파일명을 구분하는 구분자
- 절대 경로명과 상대 경로명이 있음
 - 절대 경로명 : 항상 루트(/) 디렉터리부터 시작
 - 상대 경로명 : 현재 디렉터리를 기준으로 시작

표 2-2 상대 경로명과 절대 경로명의 특징

구분	특징
상대 경로명	<ul style="list-style-type: none">• 반드시 /로 시작한다.• 루트 디렉터리부터 시작하여 특정 파일이나 디렉터리의 위치에 이르기까지 중간에 있는 모든 디렉터리의 이름을 표시한다.• 특정 위치를 가리키는 절대 경로명은 항상 동일하다.
절대 경로명	<ul style="list-style-type: none">• / 이외의 문자로 시작한다.• 현재 디렉터리를 기준으로 서브 디렉터리로 내려가면 그냥 서브 디렉터리명으로 시작한다.• 현재 디렉터리를 기준으로 상위 디렉터리로 가려면 ..(마침표 두 개)로 시작한다.• 상대 경로명은 현재 디렉터리가 어디나에 따라 달라진다.

01. 리눅스의 파일과 디렉터리

■ 파일과 디렉터리 이름의 규칙

표 2-3 파일과 디렉터리 이름을 정하는 규칙과 예

규칙	<ul style="list-style-type: none">파일과 디렉터리의 이름에는 /을 사용할 수 없다. /는 경로명에서 구분자로 사용하기 때문이다.파일과 디렉터리의 이름에는 알파벳, 숫자, 하이픈(-), 언더스코어(_), 마침표(.)만 사용한다.파일과 디렉터리의 이름에는 공백문자, *, , “‘”, @, #, \$, %, ^, & 등을 사용하면 안 된다.파일과 디렉터리 이름의 영문은 대문자와 소문자를 구별하여 다른 글자로 취급한다.파일과 디렉터리의 이름이 .(마침표)로 시작하면 숨김 파일로 간주한다.
예	<ul style="list-style-type: none">좋은 이름 예: game.txt, hello.c, test, sample11나쁜 이름 예: &game, *dir, my home, game₩사용할 수 없는 이름 예: myhome/, /test, bad/name

02. 디렉터리 관련 명령

■ 현재 디렉터리 확인

- 현재 디렉토리의 위치를 확인할 때 사용

pwd

- **기능** 현재 디렉터리의 위치를 확인한다. 즉 현재 디렉터리의 절대 경로를 출력한다.
- **형식** `pwd`

- ex) 현재 디렉터리 확인

```
[user1@localhost ~]$ pwd  
/home/user1  
[user1@localhost ~]$
```

02. 디렉터리 관련 명령

■ 디렉터리 이동

- 현재 디렉터리에서 다른 디렉터리로 이동할 때 사용

cd

- 기능 지정한 디렉터리로 이동한다.
- 형식 cd [디렉터리]
- 사용 예 cd, cd /tmp, cd 다운로드

- 절대 경로명

```
[user1@localhost ~]$ cd /tmp  
[user1@localhost tmp]$ pwd  
/tmp  
[user1@localhost tmp]$
```

- 상대 경로명

```
[user1@localhost tmp]$ cd .. /usr/lib  
[user1@localhost lib]$ pwd  
/usr/lib  
[user1@localhost lib]$
```

02. 디렉터리 관련 명령

■ 디렉터리 이동

- 원래의 홈 디렉터리 이동 방법
 - cd /home/user1 : 절대 경로명을 사용하여 홈 디렉터리로 이동
 - cd ../../home/user1 : 현재 /usr/lib 디렉터리에 있으므로 이를 기준으로 상대 경로명을 사용하여 홈 디렉터리로 이동
 - Cd ~ : 홈 디렉터리를 나타내는 기호인 ~를 사용하여 홈 디렉터리로 이동
 - Cd : 목적지를 지정하지 않고 cd 명령만 사용하면 해당 계정의 홈 디렉터리로 이동
- cd방식 (4번째 방식)

```
[user1@localhost lib]$ cd  
[user1@localhost ~]$ pwd  
/home/user1  
[user1@localhost ~]$
```

02. 디렉터리 관련 명령

■ 디렉터리 내용 확인

- 디렉터리에 있는 파일이나 서브 디렉터리 등 디렉터리의 내용을 보는 명령
- 매우 많은 옵션이 있음

ls

- **기능** 디렉터리의 내용을 출력한다.
- **형식** ls [옵션] [디렉터리(파일)]
- **옵션**
 - a: 숨김 파일을 포함하여 모든 파일의 목록을 출력한다.
 - d: 디렉터리 자체의 정보를 출력한다.
 - i: 첫 번째 행에 inode 번호를 출력한다.
 - l: 파일의 상세 정보를 출력한다.
 - A: .(마침표)와 ..(마침표 두 개)를 제외한 모든 파일 목록을 출력한다.
 - F: 파일의 종류를 표시한다(*: 실행 파일, /: 디렉터리, @: 심볼릭 링크).
 - L: 심볼릭 링크 파일의 경우 원본 파일의 정보를 출력한다.
 - R: 하위 디렉터리의 목록까지 출력한다.
- **사용 예** ls ls -F ls -al /tmp

02. 디렉터리 관련 명령

■ 디렉터리 내용 확인

- 현재 디렉터리의 내용 확인하기 : ls

```
[user1@localhost ~]$ ls  
공개  다운로드  문서  바탕화면  비디오  사진  서식  음악  
[user1@localhost ~]$
```

- 숨김 파일 확인하기 : -a 옵션

- 리눅스에서는 파일명이나 디렉터리명이 .(마침표)로 시작하면 숨김 파일임
- 숨김 파일은 ls 명령을 사용으로는 보이지 않음. 숨김 파일은 -a(all) 옵션을 사용

```
[user1@localhost ~]$ ls -a  
.          .bash_logout    .cache    .esd_auth  공개        바탕화면    서식  
..         .bash_profile   .config   .local     다운로드    비디오      음악  
.ICEauthority .bashrc       .dbus     .mozilla  문서        사진  
[user1@localhost ~]$
```

02. 디렉터리 관련 명령

■ 디렉터리 내용 확인

- 파일의 종류 표시하기: -F 옵션
 - 파일의 종류를 구분하는 기호가 표시
 - / : 디렉터리, @ : 심볼릭 링크, * : 실행파일, 표시없음 : 일반파일

```
[user1@localhost ~]$ ls -F  
공개/ 다운로드/ 문서/ 바탕화면/ 비디오/ 사진/ 서식/ 음악/  
[user1@localhost ~]$
```

- a 옵션과 -F 옵션을 연결하여 사용 할 경우 숨긴 파일의 종류도 확인 가능

```
[user1@localhost ~]$ ls -aF  
./ .bash_logout .cache/.esd_auth 공개/ 바탕화면/ 서식/  
../ .bash_profile .config/.local/ 다운로드/ 비디오/ 음악/  
.ICEauthority .bashrc .dbus/.mozilla/ 문서/ 사진/  
[user1@localhost ~]$
```

02. 디렉터리 관련 명령

■ 디렉터리 내용 확인

- 지정한 디렉터리의 내용 출력하기
 - 해당 디렉터리로 이동하지 않고도 디렉터리의 내용을 확인할 수 있음

```
[user1@localhost ~]$ ls /tmp
ssh-MK1638YPdBHt
systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-bolt.service-E4WAnN
systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-chronyd.service-r9JFIX
systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-colord.service-kbviор
systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-cups.service-KjqyQ7
systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-fwupd.service-7zuZkS
systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-rtkit-daemon.service-vV5XcQ
tracker-extract-files.1000
vmware_6644-1003206777
[user1@localhost ~]$
```

- 옵션과 인자를 함께 사용할 수도 있음
 - ex) -F 옵션과 같이 사용한 경우

```
[user1@localhost ~]$ ls -F /tmp
ssh-MK1638YPdBHt/
systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-bolt.service-E4WAnN/
systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-chronyd.service-r9JFIX/
systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-colord.service-kbviор/
systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-cups.service-KjqyQ7/
systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-fwupd.service-7zuZkS/
systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-rtkit-daemon.service-vV5XcQ/
tracker-extract-files.1000/
vmware_6644-1003206777/
[user1@localhost ~]$
```

02. 디렉터리 관련 명령

■ 디렉터리 내용 확인

- 상세한 정보 출력하기: -l 옵션

```
[user1@localhost ~]$ ls -l  
합계 0  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 공개  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 다운로드  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 문서  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 바탕화면  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 비디오  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 사진  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 서식  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 음악  
[user1@localhost ~]$
```

02. 디렉터리 관련 명령

■ 디렉터리 내용 확인

- 상세한 정보 출력하기: -l 옵션

표 2-4 파일의 상세 정보

필드 번호	필드 값	의미
1	d	다음과 같은 파일 종류를 나타낸다. -: 일반(정규) 파일 d: 디렉터리 파일 l: 심볼릭 링크 파일 b: 블록 단위로 읽고 쓰는 블록 장치 파일 c: 섹터 단위로 읽고 쓰는 문자 장치 파일 p: 파이프 파일(프로세스 간 통신에 사용되는 특수 파일) s: 소켓(네트워크 통신에 사용되는 특수 파일)
2	rwxr-xr-x	파일 접근 권한, 파일의 소유자, 그룹, 기타 사용자가 읽고 수정하고 실행할 수 있는 권한이 어떻게 부여되어 있는지를 보여준다.
3	2	하드 링크의 개수
4	user1	파일 소유자
5	user1	파일이 속한 그룹
6	6	파일 크기(바이트 단위)
7	10월 27 21:11	파일이 마지막으로 수정된 시간
8	공개	파일명

02. 디렉터리 관련 명령

■ 디렉터리 내용 확인

- 디렉터리의 자체 정보 확인하기: -d 옵션

```
[user1@localhost ~]$ ls -l /
합계 24
lrwxrwxrwx.    1      7  5월 11 09:33 bin      → usr/bin
dr-xr-xr-x.    6   4096 10월 27 21:11 boot
drwxr-xr-x.   20  root 3300 11월  3 10:58 dev
drwxr-xr-x.  133 root root 8192 11월  3 10:58 etc
(생략)

[user1@localhost ~]$ ls -ld /
dr-xr-xr-x. 17 root root 224 10월 27 20:42 /
```

- 파일이 있는지 확인하기

```
[user1@localhost ~]$ ls .bashrc
.bashrc

[user1@localhost ~]$ ls game
ls: cannot access game: 그런 파일이나 디렉터리가 없습니다
```

02. 디렉터리 관련 명령

■ 디렉터리 내용 확인

- ls 명령과 비슷한 명령: dir, vdir

```
[user1@localhost ~]$ dir  
공개  다운로드  문서  바탕화면  비디오  사진  서식  음악  
[user1@localhost ~]$ vdir  
합계 0  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 공개  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 다운로드  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 문서  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 바탕화면  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 비디오  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 사진  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 서식  
drwxr-xr-x. 2 user1 user1 6 10월 27 21:11 음악
```

02. 디렉터리 관련 명령

■ 디렉터리 생성

mkdir

- **기능** 디렉터리를 생성한다.
- **형식** `mkdir [옵션] [디렉터리]`
- **옵션** `-p`: 하위 디렉터리를 계층적으로 생성할 때 중간 단계의 디렉터리가 없으면 자동으로 중간 단계 디렉터리를 생성하고 지정한 디렉터리를 생성한다.
- **사용 예** `mkdir temp`

■ 디렉터리 한 개 만들기

- 디렉터리를 한 개만 만들려면 `mkdir` 명령에 인자로 생성하려는 디렉터리명을 지정
- 디렉터리명은 상대 경로명이나 절대 경로명으로 지정

```
[user1@localhost ~]$ mkdir temp  
[user1@localhost ~]$ ls temp  
[user1@localhost ~]$ ls  
temp  공개  다운로드  문서  바탕화면  비디오  사진  서식  음악
```

02. 디렉터리 관련 명령

■ 디렉터리 생성

- 동시에 디렉터리 여러 개 만들기
 - 생성할 디렉터리를 여러 개 지정하면 동시에 만들 수 있음
 - 이때 디렉터리는 공백문자로 구분해야 함

```
[user1@localhost ~]$ mkdir tmp1 tmp2 tmp3  
[user1@localhost ~]$ ls  
temp  tmp2  공개      문서      비디오  서식  
tmp1  tmp3  다운로드  바탕화면 사진    음악
```

- 중간 디렉터리를 자동으로 만들기: -p 옵션

02. 디렉터리 관련 명령

■ 디렉터리 생성

- 중간 디렉터리를 자동으로 만들기: -p 옵션
 - 디렉터리명으로 지정한 경로 중 중간 단계의 디렉터리가 없을 경우 자동으로 중간 단계 디렉터리를 생성한 후 최종 디렉터리를 생성
 - ex) 경로에서 중간 단계 디렉터리가 없으므로 디렉터리를 생성 못함

```
[user1@localhost ~]$ mkdir temp/mid/han  
mkdir: `temp/mid/han' 디렉터리를 만들 수 없습니다: 그런 파일이나 디렉터리가 없습니다
```

- ex) -p 옵션 사용

```
[user1@localhost ~]$ mkdir -p temp/mid/han  
[user1@localhost ~]$ ls -R temp  
temp:  
mid  
  
temp/mid:  
han  
  
temp/mid/han:
```

02. 디렉터리 관련 명령

■ 디렉터리 삭제

▪ 디렉터리 삭제

- `rmdir` 명령으로 디렉터리를 삭제할 때는 해당 디렉터리가 비어 있어야 함

`rmdir`

- **기능** 디렉터리를 삭제한다.
- **형식** `rmdir [옵션] [디렉터리]`
- **옵션** `-p`: 지정한 디렉터리를 삭제하고, 그 디렉터리의 부모 디렉터리가 빈 디렉터리일 경우 부모 디렉터리도 자동으로 삭제한다.
- **사용 예** `rmdir temp`

- ex) mp3 디렉터리 삭제

```
[user1@localhost ~]$ rmdir tmp3  
[user1@localhost ~]$ ls  
temp tmp1 tmp2 공개 다운로드 문서 바탕화면 비디오 사진 서식 음악
```

- ex) 디렉터리가 비어있지 않으면 삭제 불가

```
[user1@localhost ~]$ rmdir temp  
rmdir: failed to remove `temp': 디렉터리가 비어있지 않음
```

02. 디렉터리 관련 명령

■ [따라해보기] 디렉터리 만들고 삭제하기

- ① 현재 위치를 확인하고, 홈 디렉터리가 아니면 홈 디렉터리로 이동

```
[user1@localhost ~]$ cd  
[user1@localhost ~]$ pwd  
/home/user1
```

- ② 실습을 위한 기본 디렉터리를 만듬

```
[user1@localhost ~]$ mkdir linux_ex  
[user1@localhost ~]$ cd linux_ex  
[user1@localhost linux_ex]$
```

- ③ 2장의 실습이므로 ch2 디렉터리를 만들고 그 디렉터리로 이동하여 현재 위치를 알아봄

```
[user1@localhost linux_ex]$ mkdir ch2  
[user1@localhost linux_ex]$ cd ch2  
[user1@localhost ch2]$ pwd  
/home/user1/linux_ex/ch2
```

- ④ one, two, three 디렉터리를 동시에 만듬

```
[user1@localhost ch2]$ mkdir one two three  
[user1@localhost ch2]$ ls  
one three two
```

02. 디렉터리 관련 명령

■ [따라해보기] 디렉터리 만들고 삭제하기

- ⑤ one 디렉터리 아래에 tmp/test 디렉터리를 만듬, 이때 중간 경로인 tmp 디렉터리가 자동 생성되도록 함

```
[user1@localhost ch2]$ mkdir -p one/tmp/test
```

```
[user1@localhost ch2]$ ls -R one
```

```
one:
```

```
tmp
```

```
one/tmp:
```

```
test
```

```
one/tmp/test:
```

- ⑥ one 디렉터리를 rmdir 명령으로 삭제함, 삭제가 되는가?

```
[user1@localhost ch2]$ rmdir one
```

```
rmdir: failed to remove `one': 디렉터리가 비어있지 않음
```

02. 디렉터리 관련 명령

■ [따라해보기] 디렉터리 만들고 삭제하기

- ⑦ two, three 디렉터리를 동시에 삭제

```
[user1@localhost ch2]$ rmdir two three  
[user1@localhost ch2]$ ls  
one
```

- ⑧ 실습을 마치고 홈 디렉터리로 이동

```
[user1@localhost ch2]$ cd  
[user1@localhost ~]$ pwd  
/home/user1
```

03. 파일 관련 명령

■ 파일 내용 출력

- 파일 내용을 연속으로 출력하기: cat

cat

- 기능 파일 내용을 출력한다.
- 형식 cat [옵션] [파일]
- 옵션 -n: 행 번호를 붙여서 출력한다.
- 사용 예 cat file1 cat -n file1

- ex) /etc/hosts 파일 내용 확인

```
[user1@localhost ~]$ cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1          localhost localhost.localdomain localhost6 localhost6.localdomain6
```

- ex) 행 번호 붙이기(-n 옵션)

```
[user1@localhost ~]$ cat -n /etc/hosts
1 127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
2 ::1          localhost localhost.localdomain localhost6 localhost6.localdomain6
```

03. 파일 관련 명령

■ 파일 내용 출력

- 화면 단위로 파일 내용 출력하기: more

more

- 기능 파일 내용을 화면 단위로 출력한다.
- 형식 more [옵션] [파일]
- 옵션 + 행 번호: 출력을 시작할 행 번호를 지정한다.
- 사용 예 more file1

- ex) :/etc/services 파일 내용 보기

```
[user1@localhost ~]$ more /etc/services
# /etc/services:
# $Id: services,v 1.49 2017/08/18 12:43:23 ovasik Exp $
#
# Network services, Internet style
# IANA services version: last updated 2016-07-08
(생략)
#
# service-name  port/protocol  [aliases ...]  [# comment]

tcpmux          1/tcp                      # TCP port service multiplexer
—More—(0%)
```

space bar : 다음 화면 출력

enter : 한 줄씩 스크롤

/문자열 : 해당 문자열 검색

q : 종료

03. 파일 관련 명령

■ 파일 내용 출력

- 파일 내용을 화면 단위로 출력하기: less
 - 파일 내용을 앞뒤로 스크롤하며 이동

less

- 기능 파일 내용을 화면 단위로 출력한다.
- 형식 less [파일]
- 사용 예 less file1

표 2-5 less 명령에서 사용하는 키와 동작

키	동작
j	한 행씩 다음 행으로 스크롤한다.
k	한 행씩 이전 행으로 스크롤한다.
[Space Bar], [Ctrl]+f	다음 화면으로 이동한다.
[Ctrl]+b	이전 화면으로 이동한다.

03. 파일 관련 명령

■ 파일 내용 출력

- 파일 내용을 화면 단위로 출력하기: less
 - ex) :/etc/services 파일을 less 명령으로 출력

```
[user1@localhost ~]$ less /etc/services
# /etc/services:
# $Id: services,v 1.49 2017/08/18 12:43:23 ovasik Exp $
(상략)
#
# service-name  port/protocol  [aliases ...]  [# comment]

tcpmux          1/tcp          # TCP port service multiplexer
/etc/services
```

03. 파일 관련 명령

■ 파일 내용 출력

- 파일 내용의 뒷부분 출력하기: tail

tail

- 기능 파일 뒷부분의 몇 행을 출력한다.
- 형식 tail [옵션] [파일]
- 옵션 +행 번호: 지정한 행부터 끝까지 출력한다.
 - 숫자: 화면에 출력할 행의 수를 지정한다(기본 값은 10).
 - f: 파일 출력을 종료하지 않고 주기적으로 계속 출력한다.

- ex) /etc/services 파일의 마지막 10행 출력

```
[user1@localhost ~]$ tail /etc/services
aigairserver    21221/tcp          # Services for Air Server
ka-kdp          31016/udp         # Kollective Agent Kollective Delivery
ka-sddp          31016/tcp          # Kollective Agent Secure Distributed Delivery
edi_service      34567/udp         # dhanalakshmi.org EDI Service
axio-disc        35100/tcp          # Axiomatic discovery protocol
axio-disc        35100/udp         # Axiomatic discovery protocol
pmwebapi         44323/tcp          # Performance Co-Pilot client HTTP API
cloudcheck-ping   45514/udp         # ASSIA CloudCheck WiFi Management keepalive
cloudcheck       45514/tcp          # ASSIA CloudCheck WiFi Management System
spremotetablet  46998/tcp          # Capture handwritten signatures
```

03. 파일 관련 명령

■ 파일 내용 출력

- 파일 내용을 지정한 숫자만큼 출력하기: -숫자 옵션

- ex) /etc/services 파일 뒷부분의 7행을 출력

```
[user1@localhost ~]$ tail -7 /etc/services
edi_service      34567/udp          # dhanalakshmi.org EDI Service
axio-disc        35100/tcp          # Axiomatic discovery protocol
axio-disc        35100/udp          # Axiomatic discovery protocol
pmwebapi         44323/tcp          # Performance Co-Pilot client HTTP API
cloudcheck-ping  45514/udp          # ASSIA CloudCheck WiFi Management keepalive
cloudcheck       45514/tcp          # ASSIA CloudCheck WiFi Management System
spremotetablet  46998/tcp          # Capture handwritten signatures
```

03. 파일 관련 명령

■ 파일 내용 출력

- 파일 내용을 주기적으로 반복 출력하기: -f 옵션
 - 파일 내용의 변화를 확인할 때 편리
 - tail 명령이 무한히 반복되기 때문에 [ctrl]+ C(^C)로 명령을 종료해야 함
 - ex) /etc/services 파일을 -f옵션으로 출력

```
[user1@localhost ~]$ tail -f /etc/services
aigairserver    21221/tcp          # Services for Air Server
ka-kdp          31016/udp         # Kollective Agent Kollective Delivery
ka-sddp          31016/tcp          # Kollective Agent Secure Distributed Delivery
edi_service      34567/udp         # dhanalakshmi.org EDI Service
axio-disc        35100/tcp          # Axiomatic discovery protocol
axio-disc        35100/udp         # Axiomatic discovery protocol
pmwebapi         44323/tcp          # Performance Co-Pilot client HTTP API
cloudcheck-ping  45514/udp         # ASSIA CloudCheck WiFi Management keepalive
cloudcheck       45514/tcp          # ASSIA CloudCheck WiFi Management System
spremotetablet  46998/tcp          # Capture handwritten signatures
^C
```

03. 파일 관련 명령

■ [따라해보기] 파일 내용 출력하기

- ① less 명령으로 /etc/services 파일의 내용을 출력

```
[user1@localhost ~]$ less /etc/services
# /etc/services:
# $Id: services,v 1.49 2017/08/18 12:43:23 ovasik Exp $
#
# Network services, Internet style
# IANA services version: last updated 2016-07-08
(생략)
## Each line describes one service, and is of the form:
#
# service-name  port/protocol  [aliases ...]  [# comment]

tcpmux          1/tcp                      # TCP port service multiplexer
/etc/services
```

03. 파일 관련 명령

■ [따라해보기] 파일 내용 출력하기

- ② [space bar] 를 눌러 다음 페이지를 출력

```
tcpmux      1/udp          # TCP port service multiplexer
rje         5/tcp          # Remote Job Entry
rje         5/udp          # Remote Job Entry
echo        7/tcp
(생략)
ftp-data    20/udp
# 21 is registered to ftp, but also used by fsp
ftp         21/tcp
ftp         21/udp      fsp fspd
ssh         22/tcp          # The Secure Shell (SSH) Protocol
ssh         22/udp          # The Secure Shell (SSH) Protocol
:
```

03. 파일 관련 명령

■ [따라해보기] 파일 내용 출력하기

- ③ k키를 네 번 눌러 위로 다시 이동

```
#  
# service-name  port/protocol  [aliases ...]      [# comment]  
  
tcpmux          1/tcp           # TCP port service multiplexer  
tcpmux          1/udp           # TCP port service multiplexer  
rje             5/tcp           # Remote Job Entry  
rje             5/udp           # Remote Job Entry  
  
(생략)  
chargen         19/tcp          ttyst source  
chargen         19/udp          ttyst source  
ftp-data        20/tcp            
ftp-data        20/udp            
# 21 is registered to ftp, but also used by fsp  
ftp              21/tcp            
:
```

03. 파일 관련 명령

■ [따라해보기] 파일 내용 출력하기

- ④ '/IEEE'를 입력하여 파일 내용 중에서 문자열 'IEEE'가 있는 곳을 찾음

```
#  
# service-name  port/protocol  [aliases ...]      [# comment]  
  
tcpmux      1/tcp          # TCP port service multiplexer  
tcpmux      1/udp          # TCP port service multiplexer  
rje         5/tcp          # Remote Job Entry  
rje         5/udp          # Remote Job Entry  
(생략)  
chargen     19/tcp         ttytst source  
chargen     19/udp         ttytst source  
ftp-data    20/tcp  
ftp-data    20/udp  
# 21 is registered to ftp, but also used by fsp  
ftp         21/tcp  
  
ieee-mms    651/tcp        # IEEE MMS  
ieee-mms    651/udp        # IEEE MMS  
hello-port  652/tcp        # HELLO_PORT  
hello-port  652/udp        # HELLO_PORT  
(생략)  
hap         661/tcp        # HAP  
hap         661/udp        # HAP  
pftp        662/tcp        # PFTP  
pftp        662/udp        # PFTP  
purenoise   663/tcp        # PureNoise  
:  
:
```

- ⑤ n 키를 눌러 문자열 'IEEE'를 계속 찾음
⑥ Q 키를 눌러 less 명령을 종료

03. 파일 관련 명령

■ 파일 복사

- 파일을 복사할 때는 cp 명령을 사용
- cp 명령으로 디렉터리도 복사할 수 있음

cp

- 기능** 파일이나 디렉터리를 복사한다.
- 형식** cp [옵션] [파일1(디렉터리1)] [파일2(디렉터리2)]
- 옵션**
 - i: 파일2가 이미 존재하면 덮어쓸 것인지 물어본다.
 - r: 디렉터리를 복사할 때 지정한다.
- 사용 예**

```
cp file1 file2
      cp f1 f2 f3 dir1
      cp -r dir1 dir2
```

■ ch2 디렉터리로 이동

```
[user1@localhost ~]$ cd linux_ex/ch2
[user1@localhost ch2]$ pwd
/home/user1/linux_ex/ch2
```

03. 파일 관련 명령

■ 파일 복사

- 두 인자가 모두 파일인 경우
 - ex) /etc/hosts 파일을 현재 디렉터리에 text1 파일로 복사

```
[user1@localhost ch2]$ ls  
one  
[user1@localhost ch2]$ cp /etc/hosts text1  
[user1@localhost ch2]$ ls  
one  text1
```

03. 파일 관련 명령

■ 파일 복사

- 두 번째 인자가 디렉터리인 경우
 - 파일을 해당 디렉터리 아래에 복사
 - ex) temp 디렉터리에 text1 파일 복사

```
[user1@localhost ch2]$ mkdir temp  
[user1@localhost ch2]$ cp text1 temp  
[user1@localhost ch2]$ ls temp  
text1
```

- ex) temp 디렉터리에 text1 파일을 text2파일로 복사

```
[user1@localhost ch2]$ cp text1 /etc  
cp: cannot create regular file '/etc/text1': 허가 거부
```

- 쓰기 권한이 없는 디렉터리에 파일을 복사할 경우 오류가 발생

```
[user1@localhost ch2]$ cp text1 temp/text2  
[user1@localhost ch2]$ ls temp  
text1 text2
```

03. 파일 관련 명령

■ 파일 복사

▪ 인자를 여러 개 지정할 경우

- cp명령에서 첫 번째 인자의 자리에 파일명 여러 개 지정 가능
- 마지막 인자는 반드시 디렉터리여야 함
- 마지막에 지정한 디렉터리로 앞서 지정한 파일들이 모두 복사됨
- ex) /etc/hosts와 /etc/services를 temp 디렉터리에 복사

```
[user1@localhost ch2]$ cp /etc/hosts /etc/services temp  
[user1@localhost ch2]$ ls temp  
hosts services text1 text2
```

▪ -i 옵션 사용하기

- -i 옵션을 사용하면 두 번째 인자로 지정한 파일이 이미 있는 파일인 경우 덮어서 복사할 것인지 선택
- y라고 답하면 덮어서 복사하고 n이라고 답하면 복사하지 않음
- ex) -i 옵션을 지정하고 /etc/hosts를 text1로 다시 복사

```
[user1@localhost ch2]$ cp -i /etc/hosts text1  
cp: overwrite `text1'? n
```

03. 파일 관련 명령

■ 파일 복사

▪ 디렉터리 복사하기

- 첫 번째 인자와 두 번째 인자에 모두 디렉터리를 지정하고 -r 옵션을 사용해야 함
- ex) temp 디렉터리를 temp2 디렉터리로 복사하는 데 -r 옵션을 지정하지 않은 경우

```
[user1@localhost ch2]$ cp temp temp2  
cp: omitting directory `temp'
```

- 두 번째 인자로 지정한 목적지 디렉터리가 존재하지 않는 경우 새로 생성
- 디렉터리가 복사되면 원본 디렉터리 아래에 있던 모든 내용도 함께 복사
- ex) temp 디렉터리를 temp2 디렉터리로 복사

```
[user1@localhost ch2]$ cp -r temp temp2  
[user1@localhost ch2]$ ls temp2  
hosts services text1 text2
```

- 두 번째 인자로 지정한 디렉터리가 이미 있는 디렉터리일 경우 원본 디렉터리가 목적지 디렉터리 아래에 원본 디렉터리와 같은 이름으로 복사
- ex) temp 디렉터리를 다시 temp2 디렉터리로 복사

```
[user1@localhost ch2]$ cp -r temp temp2  
[user1@localhost ch2]$ ls temp2  
hosts services temp text1 text2
```

03. 파일 관련 명령

■ 파일 이동과 파일명 변경

mv

- **기능** 파일 또는 디렉터리를 이동하거나 이름을 바꾼다.
 - **형식** mv [옵션] [파일1(디렉터리1)] [파일2(디렉터리2)]
 - **옵션** -i: 파일2(디렉터리2)가 존재하면 덮어쓸 것인지 물어본다.
 - **사용 예** mv file1 file2
- mv 명령의 첫 번째 인자로는 원본 파일이나 디렉터리를 지정, 두 번째 인자는 목적지 파일이나 디렉터리를 지정

■ 파일을 파일로 이동하기(파일명 바꾸기)

- 원본 파일의 파일명을 다른 파일명으로 바꿈
- 만약 두 번째 인자로 지정한 파일명이 이미 존재하는 파일이면 원본 파일의 내용으로 덮어쓰고 기존의 내용이 삭제. 두 번째 인자로 지정한 파일명이 존재하지 않는 파일이라면 새 파일이 생성
- ex) text1 파일을 data1 파일로 이동(파일명 변경)

```
[user1@localhost ch2]$ mv text1 data1  
[user1@localhost ch2]$ ls  
data1  one  temp  temp2
```

03. 파일 관련 명령

■ 파일 이동과 파일명 변경

- 파일을 다른 디렉터리로 이동하기
 - 두 번째 인자로 디렉터리를 지정할 경우 원본 파일을 지정한 디렉터리로 이동
 - ex) data1 파일을 temp 디렉터리로 이동

```
[user1@localhost ch2]$ mv data1 temp  
[user1@localhost ch2]$ ls  
one temp temp2  
[user1@localhost ch2]$ ls temp  
data1 hosts services text1 text2
```

- 두 번째 인자에 디렉터리와 파일명을 함께 지정할 경우 파일이 지정한 디렉터리로 이동하면 파일명도 바뀜

```
[user1@localhost ch2]$ cp temp/data1 text1  
[user1@localhost ch2]$ ls  
one temp temp2 text1  
[user1@localhost ch2]$ mv text1 temp/data2  
[user1@localhost ch2]$ ls temp  
data1 data2 hosts services text1 text2
```

- 쓰기 권한이 없는 디렉터리로 파일을 이동할 경우 오류발생

```
[user1@localhost ch2]$ mv temp/data2 /etc  
mv: cannot move `temp/data2' to `/etc/data2': 허가 거부
```

03. 파일 관련 명령

■ 파일 이동과 파일명 변경

- 여러 파일을 디렉터리로 이동하기
 - 첫 번째 인자에 파일을 여러 개 지정
 - 마지막 인자는 반드시 디렉터리여야 함
 - ex) temp 디렉터리에 있던 data1과 data2 파일을 현재 디렉터리로 이동

```
[user1@localhost ch2]$ ls temp
data1  data2  hosts  services  text1  text2
[user1@localhost ch2]$ mv temp/data1 temp/data2 .
[user1@localhost ch2]$ ls
data1  data2  one  temp  temp2
[user1@localhost ch2]$ ls temp
hosts  services  text1  text2
```

- -i 옵션 사용하기

- 두 번째 인자에 지정한 파일명이 기존에 있는 파일일 경우 덮어서 이동할 것인지를 물어봄

```
[user1@localhost ch2]$ mv -i data1 data2
mv: overwrite `data2'? n
[user1@localhost ch2]$ ls
data1  data2  me  temp  temp2
```

03. 파일 관련 명령

■ 파일 이동과 파일명 변경

- 디렉터리를 디렉터리로 이동하기(디렉터리명 바꾸기)
 - 인자를 모두 디렉터리로 지정하면 디렉터리가 이동
 - 두 번째 인자가 기존에 있던 디렉터리가 아닐 경우에는 디렉터리명이 변경
 - ex) temp2 디렉터리가 temp3 디렉터리로 이름 변경

```
[user1@localhost ch2]$ mv temp2 temp3  
[user1@localhost ch2]$ ls  
data1 data2 one temp temp3
```

- 두 번째 인자가 기존에 있던 디렉터리일 경우 원본 디렉터리가 두 번째 인자로 지정된 디렉터리 아래로 이동
- ex) temp3 디렉터리가 temp 디렉터리 아래로 이동

```
[user1@localhost ch2]$ ls  
data1 data2 me one temp temp3  
[user1@localhost ch2]$ mv temp3 temp  
[user1@localhost ch2]$ ls  
data1 data2 one temp  
[user1@localhost ch2]$ ls temp  
hosts services temp3 text1 text2
```

03. 파일 관련 명령

■ 파일 삭제

rm

- **기능** 파일을 삭제한다.
- **형식** rm [옵션] [파일 또는 디렉터리]
- **옵션** -i: 파일을 정말 삭제할 것인지 확인한다.
-r: 디렉터리를 삭제할 때 지정한다.
- **사용 예** rm file rm -r dir

- 삭제할 파일을 인자로 지정하면 해당 파일이 삭제
- 바로 삭제되어 복구할 수도 없으므로 파일을 삭제할 때는 신중해야 함
- ex) data2 파일 삭제

```
[user1@localhost ch2]$ ls  
data1  data2  me  one  temp  
[user1@localhost ch2]$ rm data2  
[user1@localhost ch2]$ ls  
data1  one  temp
```

03. 파일 관련 명령

■ 파일 삭제

- i 옵션 사용하기
 - i 옵션을 지정하고 rm 명령을 사용하면 정말 삭제할 것인지 물어봄
 - ex) data2 파일을 삭제

```
[user1@localhost ch2]$ rm -i data1  
rm: remove 일반 파일 'data1'? n  
[user1@localhost ch2]$ ls  
data1  me  one  temp
```

- 디렉터리 삭제하기

- rm 명령으로 디렉터리를 지울 때는 -r 옵션을 지정 (삭제된 디렉터리는 복구 불가능)
- ex) -r 옵션을 지정하지 않을 경우 오류 메시지 출력

```
[user1@localhost ch2]$ cd temp  
[user1@localhost temp]$ ls  
hosts  services  temp3  text1  text2
```

```
[user1@localhost temp]$ rm temp3  
rm: cannot remove `temp3': 디렉터리입니다
```

03. 파일 관련 명령

■ 파일 삭제

▪ 디렉터리 삭제하기

- rmdir 명령으로 temp3 디렉터리를 삭제하려고 하면 temp3 디렉터리가 비어 있지 않다는 오류 메시지가 출력

```
[user1@localhost temp]$ rmdir temp3
rmdir: failed to remove `temp3': 디렉터리가 비어있지 않음
[user1@localhost temp]$ ls temp3
hosts services temp text1 text2
```

- rm 명령에 -r 옵션을 지정하여 temp3 디렉터리를 삭제

```
[user1@localhost temp]$ ls
hosts services temp3 text1 text2
[user1@localhost temp]$ rm -r temp3
[user1@localhost temp]$ ls
hosts services text1 text2
```

03. 파일 관련 명령

■ 파일 삭제

▪ 디렉터리 삭제하기

- -i 옵션을 사용하여 디렉터리 삭제
- 삭제하려는 디렉터리 아래에 있는 파일이나 서브 디렉터리를 삭제할 것인지 계속 물어봄

```
[user1@localhost temp]$ cd ..
[user1@localhost ch2]$ ls
data1  one  temp
[user1@localhost ch2]$ rm -ri temp
rm: descend into directory `temp'? y
rm: remove 일반 파일 `temp/text1'? n
rm: remove 일반 파일 `temp/text2'? n
rm: remove 일반 파일 `temp/hosts'? n
rm: remove 일반 파일 `temp/services'? y
rm: remove 디렉터리 `temp'? n
[user1@localhost ch2]$ ls temp
hosts  text1  text2
```

03. 파일 관련 명령

■ 파일 삭제

▪ 디렉터리 삭제하기

- -i 옵션을 사용하여 디렉터리 삭제
- 만약 디렉터리에 삭제되지 않은 파일이 있을 경우 디렉터리가 비어 있지 않다는 메시지를 출력하고 디렉터리를 삭제하지 않음

```
[user1@localhost ch2]$ rm -ri temp
rm: descend into directory 'temp'? y
rm: remove 일반 파일 'temp/text1'? n
rm: remove 일반 파일 'temp/hosts'? n
rm: remove 일반 파일 'temp/text2'? n
rm: remove 디렉터리 'temp'? y
rm: cannot remove `temp': 디렉터리가 비어있지 않음
[user1@localhost ch2]$ ls
data1  one  temp
```

03. 파일 관련 명령

■ 파일 삭제

▪ 디렉터리 삭제하기

- -i 옵션을 사용하여 디렉터리 삭제
- 만약 디렉터리에 삭제되지 않은 파일이 있을 경우 디렉터리가 비어 있지 않다는 메시지를 출력하고 디렉터리를 삭제하지 않음

```
[user1@localhost ch2]$ rm -ri temp
rm: descend into directory 'temp'? y
rm: remove 일반 파일 'temp/text1'? n
rm: remove 일반 파일 'temp/hosts'? n
rm: remove 일반 파일 'temp/text2'? n
rm: remove 디렉터리 'temp'? y
rm: cannot remove `temp': 디렉터리가 비어있지 않음
[user1@localhost ch2]$ ls
data1  one  temp
```

03. 파일 관련 명령

■ [따라해보기] 파일 복사·이동·삭제하기

- ① 실습을 위해 /etc/hosts 파일을 test.org로 복사

```
[user1@localhost ch2]$ cp /etc/hosts test.org  
[user1@localhost ch2]$ ls  
data1 one temp test.org
```

- ② test 디렉터리를 만듬

```
[user1@localhost ch2]$ mkdir test  
[user1@localhost ch2]$ ls  
data1 one temp test test.org
```

- ③ test.org 파일을 test 디렉터리로 복사

```
[user1@localhost ch2]$ cp test.org test  
[user1@localhost ch2]$ ls test  
test.org
```

- ④ test 디렉터리에 있는 test.org의 파일명을 test.bak로 변경

```
[user1@localhost ch2]$ mv test/test.org test/test.bak  
[user1@localhost ch2]$ ls test  
test.bak
```

03. 파일 관련 명령

■ [따라해보기] 파일 복사·이동·삭제하기

- ⑤ test.org 파일을 삭제

```
[user1@localhost ch2]$ rm test.org  
[user1@localhost ch2]$ ls  
data1 one temp test
```

- ⑥ test 디렉터리에 있는 test.bak를 현재 디렉터리에 test.org로 복사

```
[user1@localhost ch2]$ cp test/test.bak test.org  
[user1@localhost ch2]$ ls  
data1 one temp test test.org
```

- ⑦ test, one 디렉터리를 삭제

```
[user1@localhost ch2]$ rm -r test one  
[user1@localhost ch2]$ ls  
data1 temp test.org
```

03. 파일 관련 명령

■ 파일 링크

- 파일 링크 : 기존에 있는 파일에 새로운 파일명을 붙이는 것
- 하드링크 : 기존 파일에 새로운 파일명을 추가로 생성
- 심볼릭 링크 : 원본 파일을 가리키는 새로운 파일을 생성

■ 리눅스 파일의 구성

- 파일 = 파일명 + inode + 데이터 블록
 - 파일명 : 사용자가 파일에 접근할 때 사용하는 파일의 이름
 - Inode : 파일에 대한 정보를 가진 특별한 구조체. 외부적으로는 번호로 표시, 내부적으로는 파일의 종류 및 크기, 소유자, 파일 변경 시간, 파일명 등 파일 상세 정보와 데이터 블록의 주소를 저장
- 파일의 inode 번호는 ls -i 명령으로 확인 가능
 - 파일명 앞에 출력된 숫자가 inode 번호
 - 파일 이름은 다르지만 inode 번호가 같다면 같은 파일
- ex) ls -i 명령으로 파일의 inode 번호를 확인

```
[user1@localhost ch2]$ ls -i  
52343390 data1 18161958 temp 52343391 test.org
```

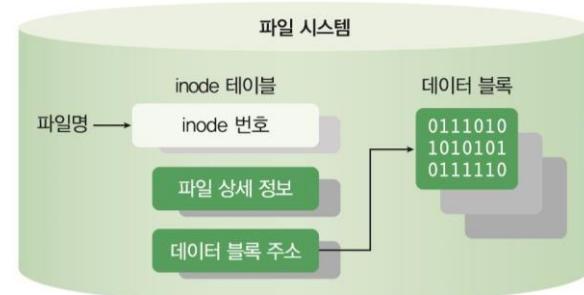


그림 2-2 리눅스 파일의 구성 요소

03. 파일 관련 명령

■ 파일 링크

- 하드 링크 : 한 파일에 여러 개의 이름을 붙일 수 있는데, 이때 붙이는 파일명

ln

- **기능** 파일의 링크를 생성한다.
- **형식** ln [옵션] [원본 파일] [링크 파일]
- **옵션** -s: 심볼릭 링크 파일을 생성한다.
- **사용 예** ln test hdtest ln -s test sbtest

- ex) data1에 대한 하드 링크로 data1.ln을 만듬

```
[user1@localhost ch2]$ ls -l  
합계 8  
-rw-r--r--. 1 user1 user1 158 11월 2 01:51 data1  
drwxrwxr-x. 2 user1 user1 45 11월 2 02:30 temp  
-rw-r--r--. 1 user1 user1 158 11월 2 02:33 test.org  
[user1@localhost ch2]$ ln data1 data1.ln  
[user1@localhost ch2]$ ls -l  
합계 12  
-rw-r--r--. 2 user1 user1 158 11월 2 01:51 data1  
-rw-r--r--. 2 user1 user1 158 11월 2 01:51 data1.ln  
drwxrwxr-x. 2 user1 user1 45 11월 2 02:30 temp  
-rw-r--r--. 1 user1 user1 158 11월 2 02:33 test.org
```

03. 파일 관련 명령

■ 파일 링크

- data1 파일에 하드 링크를 생성한 뒤 ls -i 명령으로 두 파일의 inode 값을 비교해보면 같음

```
[user1@localhost ch2]$ ls -i  
52343390 data1  52343390 data1.ln  18161958 temp  52343391 test.org
```

- 파일 내용을 확인해보면 data1이나 data1.ln은 같은 내용임 (같은 파일)

```
[user1@localhost ch2]$ cat data1  
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1      localhost localhost.localdomain localhost6 localhost6.localdomain6  
[user1@localhost ch2]$ cat data1.ln  
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1      localhost localhost.localdomain localhost6 localhost6.localdomain6
```

- 하드 링크는 같은 파일에 이름만 다르게 붙이는 것이지만, 복사는 완전히 독립적인 파일을 만듦

```
[user1@localhost ch2]$ cp data1 data1.cp  
[user1@localhost ch2]$ ls -i  
52343390 data1      52343390 data1.ln  52343391 test.org  
52343379 data1.cp   18161958 temp
```

03. 파일 관련 명령

■ 파일 링크

- 심볼릭 링크 만들기: -s 옵션
 - 심볼릭 링크 : 윈도의 바로가기처럼 원본 파일을 가리키는 파일
 - ex) data1 파일의 심볼릭 링크로 data1.sl을 생성

```
[user1@localhost ch2]$ ln -s data1 data1.sl  
[user1@localhost ch2]$ ls -i  
52343390 data1      52343390 data1.ln  18161958 temp  
52343379 data1.cp   52382459 data1.sl  52343391 test.org
```

- 심볼릭 링크의 inode 번호를 보면 원본 파일과 다른 번호
- ls -l 명령으로 확인해보면 파일의 종류가 'l(소문자 l)'로 표시되고 파일명도 '->'를 사용하여 원본 파일이 무엇인지를 알려줌

```
[user1@localhost ch2]$ ls -l data1.sl  
lrwxrwxrwx. 1 user1 user1 5 11월 2 02:44 data1.sl -> data1
```

03. 파일 관련 명령

■ 파일 링크

- 심볼릭 링크 만들기: -s 옵션
 - 심볼릭 링크와 하드 링크의 차이
 - 파일의 종류가 l(소문자 L)로 표시
 - 하드 링크의 개수가 하나, 즉 원본 파일에 이름을 추가하는 것이 아님
 - 파일명 뒤에 원본 파일의 이름이 표시(-> data1).
 - inode 번호가 원본 파일과 다른 파일 즉, 원본 파일과 심볼릭 링크 파일은 별개의 파일
 - 심볼릭 링크 파일의 내용은 원본 파일의 경로를 가짐
 - 심볼릭 링크의 내용을 출력하면 원본 파일의 경로가 출력되는 것이 아니라 원본 파일의 내용이 출력됨

```
[user1@localhost ch2]$ cat data1.sl  
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
```

- 심볼릭 링크의 경우 원본 파일이 삭제되면 심볼릭 링크로 연결할 수 없다는 점을 주의해야 함
- 원본 파일인 data1이 삭제되면 data1.sl의 내용을 출력했을 때 원본 파일이 없으므로 오류 메시지가 나옴

```
[user1@localhost ch2]$ rm data1  
[user1@localhost ch2]$ cat data1.sl  
cat: data1.sl: 그런 파일이나 디렉터리가 없습니다
```

03. 파일 관련 명령

■ [따라해보기] 하드 링크와 심볼릭 링크

- ① test.org 파일의 하드 링크로 test.ln을 생성

```
[user1@localhost ch2]$ ln test.org test.ln  
[user1@localhost ch2]$ ls  
data1.cp  data1.ln  data1.sl  temp  test.ln  test.org
```

- ② test.org 파일의 하드 링크로 test.ln2를 생성

```
[user1@localhost ch2]$ ln test.org test.ln2  
[user1@localhost ch2]$ ls  
data1.cp  data1.ln  data1.sl  temp  test.ln  test.ln2  test.org
```

- ③ test.org와 test.ln, test.ln2의 inode 번호가 같다는 것을 확인

```
[user1@localhost ch2]$ ls -i  
52343379 data1.cp  52382459 data1.sl  52343391 test.ln   52343391 test.org  
52343390 data1.ln  18161958 temp      52343391 test.ln2
```

03. 파일 관련 명령

■ 하드 링크와 심볼릭 링크

- ④ test.org와 test.ln, test.ln2의 하드 링크 개수를 확인

```
[user1@localhost ch2]$ ls -l  
합계 20  
-rw-r--r--. 1 user1 user1 158 11월 2 02:41 data1.cp  
-rw-r--r--. 1 user1 user1 158 11월 2 01:51 data1.ln  
lrwxrwxrwx. 1 user1 user1 5 11월 2 02:44 data1.sl -> data1  
drwxrwxr-x. 2 user1 user1 45 11월 2 02:30 temp  
-rw-r--r--. 3 user1 user1 158 11월 2 02:33 test.ln  
-rw-r--r--. 3 user1 user1 158 11월 2 02:33 test.ln2  
-rw-r--r--. 3 user1 user1 158 11월 2 02:33 test.org
```

- ⑤ temp 디렉터리에 대한 심볼릭 링크로 tmp를 만듬

```
[user1@localhost ch2]$ ls temp  
hosts text1 text2  
[user1@localhost ch2]$ ls tmp  
hosts text1 text2
```

03. 파일 관련 명령

■ 하드 링크와 심볼릭 링크

- ⑥ temp와 tmp 디렉터리의 내용이 같다는 것을 확인

```
[user1@localhost ch2]$ ln -s temp tmp  
[user1@localhost ch2]$ ls -l tmp  
lrwxrwxrwx. 1 user1 user1 4 11월 2 02:51 tmp -> temp
```

- ⑦ tmp 디렉터리와 test.ln, test.ln2, test.org 파일을 모두 삭제

03. 파일 관련 명령

■ 파일 관련 기타 유용한 명령

- 빈 파일 만들기, 접근/수정 시간 변경하기: touch

touch

- **기능** 빈 파일을 생성한다.
- **형식** touch [-acm] [-r ref_file | -t time] [파일]
- **옵션** -a: 접근 시간만 변경한다.
-m: 수정 시간만 변경한다.
-t [[CC]YY]MMDDhhmm[.ss]: 시간을 직접 입력한다.
- **사용 예** touch test

- ex) test 파일을 생성한 뒤 ls -l로 확인

```
[user1@localhost ch2]$ touch test  
[user1@localhost ch2]$ ls -l test  
-rw-rw-r--. 1 user1 user1 0 11월 2 02:55 test
```

03. 파일 관련 명령

■ 파일 관련 기타 유용한 명령

- 빈 파일 만들기, 접근/수정 시간 변경하기: touch
 - 이미 있는 파일에 대해 touch 명령을 옵션 없이 사용하면 파일의 수정 시간이 현재 시간으로 변경됨
 - ex) touch 명령을 사용하여 data1.cp의 수정 시간을 현재 시간으로 변경

```
[user1@localhost ch2]$ ls -l data1.cp
-rw-r--r--. 1 user1 user1 158 11월 2 02:41 data1.cp
[user1@localhost ch2]$ date
2019. 11. 02. (토) 02:56:17 KST
[user1@localhost ch2]$ touch data1.cp
[user1@localhost ch2]$ ls -l data1.cp
-rw-r--r--. 1 user1 user1 158 11월 2 02:56 data1.cp
```

03. 파일 관련 명령

■ 파일 관련 기타 유용한 명령

- 빈 파일 만들기, 접근/수정 시간 변경하기: touch
 - -t 옵션을 사용하면 변경할 시간을 지정할 수 있음

시간 표시

- **형식** [[CC]YY]MMDDhhmm[.ss]
- **설명** CC: 연도의 첫 두 자리
YY: 연도의 마지막 두 자리
MM: 달(01~12 범위 내 지정)
DD: 날짜(01~31 범위 내 지정)
hh: 시간(00~23 범위 내 지정)
mm: 분(00~59 범위 내 지정)
ss: 초(00~59 범위 내 지정)

- CC(연도)를 지정하지 않으면 [표 2-6]과 같이 YY 기본 값에 따라 CC를 자동으로 인식
- ex) 연도를 지정하지 않고 월, 일, 시간만 지정하여 test 파일의 수정 시간을 변경

표 2-6 연도 지정 방법

YY	69~99	00~68
CC	19	20

```
[user1@localhost ch2]$ ls -l test
-rw-rw-r--. 1 user1 user1 0 11월 2 02:55 test
[user1@localhost ch2]$ touch -t 12311200 test
[user1@localhost ch2]$ ls -l test
-rw-rw-r--. 1 user1 user1 0 12월 31 2019 test
```

03. 파일 관련 명령

■ 파일 관련 기타 유용한 명령

- 파일 내용 검색하기: grep

grep

- 기능 지정한 패턴이 포함된 행을 찾는다.
- 형식 grep [옵션] [패턴] [파일]
- 옵션 -i: 대문자 · 소문자를 모두 검색한다.
-l: 지정한 패턴이 포함된 파일명을 출력한다.
-n: 행 번호를 출력한다.
- 사용 예 grep root /etc/passwd
grep -n unix ~/*.txt
grep -l hello *.c

- ex) data 파일에서 'DHCP'가 포함된 행을 찾아 출력

```
[user1@localhost ch2]$ cp /etc/services data
[user1@localhost ch2]$ grep DHCP data
dhcp-failover    647/tcp          # DHCP Failover
dhcp-failover    647/udp          # DHCP Failover
```

- ex) -n 옵션을 사용해 검색된 행 번호도 함께 출력

```
[user1@localhost ch2]$ grep -n DHCP data
1413:dhcp-failover    647/tcp          # DHCP Failover
1414:dhcp-failover    647/udp          # DHCP Failover
```

03. 파일 관련 명령

■ 파일 관련 기타 유용한 명령

- 파일 찾기: find

find

- **기능** 지정한 위치에서 검색 조건에 맞는 파일을 찾는다.
- **형식** find [경로] [검색 조건] [동작]
- **검색조건**
 - name filename: 파일명으로 검색한다.
 - type 파일 종류: 파일 종류로 검색한다.
 - user loginID: 지정한 사용자가 소유한 모든 파일을 검색한다.
 - perm 접근 권한: 지정한 사용 권한과 일치하는 파일을 검색한다.
- **동작**
 - exec 명령 {}\\: 검색된 파일에 명령을 실행한다.
 - ok 명령 {}\\: 사용자의 확인을 받아서 명령을 실행한다.
 - print: 검색된 파일의 절대 경로명을 화면에 출력한다(기본 동작).
 - ls: 검색 결과를 긴 목록 형식으로 출력한다.
- **사용 예** find ~ -name hello.c
 - find /tmp -user user10 -exec rm {}\\;

- ex) 접근 권한이 없는 디렉터리는 검색할 수 없어서 '허가 거부' 메시지가 출력

```
[user1@localhost ch2]$ find /usr -name ls
/usr/bin/ls
find: '/usr/share/polkit-1/rules.d': 허가 거부
find: '/usr/libexec/initscripts/legacy-actions/auditd': 허가 거부
```

03. 파일 관련 명령

■ 파일 관련 기타 유용한 명령

▪ 파일 찾기: find

- -user 옵션 : 특정 사용자 계정이 소유자인 파일을 찾고 싶을 경우 사용

```
[user1@localhost ch2]$ find /home -user user1
[user1@localhost ch2]$ find /home -user user1
/home/user1
/home/user1/.mozilla
/home/user1/.mozilla/extensions
/home/user1/.mozilla/plugins
/home/user1/.bash_logout
/home/user1/.bash_profile
/home/user1/.bashrc
(생략)
```

03. 파일 관련 명령

■ 파일 관련 기타 유용한 명령

▪ 파일 찾기: find

- find 명령으로 검색한 모든 파일을 대상으로 동일한 작업을 수행하려면 -exec나 -ok 옵션 사용
- ex) /tmp 디렉터리 아래에 있는 user1 계정 소유의 파일을 전부 찾아서 삭제할 경우
 - find 명령으로 찾은 파일의 절대 경로가 exec 다음의 {}가 있는 위치에 삽입되어 명령이 처리
 - rm 명령과 {} 사이, {}와 \ 사이에 공백이 있어야 하며, \과 ;은 공백 없이 붙어야 함

```
[user1@localhost ch2]$ find /tmp -user user1 -exec rm {} \;
rm: cannot remove '/tmp/ssh-MK1638YPdBHt': 디렉터리입니다
rm: cannot remove '/tmp/.esd-1000': 디렉터리입니다
find: '/tmp/systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-fwupd.service-7zuZkS': 허가 거부
find: '/tmp/systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-chronyd.service-r9JFIX': 허가 거부
find: '/tmp/systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-rtkit-daemon.service-vV5XcQ': 허가
거부
find: '/tmp/vmware-root_6644-1003206777': 허가 거부
find: '/tmp/systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-cups.service-KjqyQ7': 허가 거부
find: '/tmp/systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-bolt.service-E4WAnN': 허가 거부
find: '/tmp/systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-colord.service-kbvier': 허가 거부
rm: cannot remove '/tmp/tracker-extract-files.1000': 디렉터리입니다
```

03. 파일 관련 명령

■ 파일 관련 기타 유용한 명령

▪ 파일 찾기: find

- find 명령으로 검색한 파일을 삭제하기 전에 하나씩 확인하고 싶으면 -exec 대신 -ok를 사용

```
[user1@localhost ch2]$ find /tmp -user user1 -ok rm {} \;;
< rm ... /tmp/ssh-MK1638YPdBHt > ? y
rm: cannot remove '/tmp/ssh-MK1638YPdBHt': 디렉터리입니다
< rm ... /tmp/.esd-1000 > ? n
find: '/tmp/systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-fwupd.service-7zuZkS':
허가 거부
find: '/tmp/systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-chronyd.service-
r9JF1X': 허가 거부
find: '/tmp/systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-rtkit-daemon.service-
vV5XcQ': 허가 거부
find: '/tmp/vmware-root_6644-1003206777': 허가 거부
find: '/tmp/systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-cups.service-KjqyQ7':
허가 거부
find: '/tmp/systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-bolt.service-E4WAnN':
허가 거부
find: '/tmp/systemd-private-400d63c85efc4ea2b3ab99cc84ba8663-colord.service-kbvier':
허가 거부
< rm ... /tmp/tracker-extract-files.1000 > ? n
[user1@localhost ch2]$
```

03. 파일 관련 명령

■ 파일 관련 기타 유용한 명령

- 명령의 위치 찾기: whereis, which
 - whereis 명령 : 지정한 명령을 고정된 특정 경로에서 검색

whereis

- **기능** 지정된 경로에서 명령의 바이너리 파일이나 매뉴얼 파일의 위치를 찾는다.
- **형식** whereis [옵션] [파일]
- **옵션**
 - b: 바이너리 파일만 검색한다.
 - m: 매뉴얼 파일만 검색한다.
 - s: 소스 파일만 검색한다.
- **사용 예** whereis ls

- 환경 변수 \$PATH와 \$MANPATH에 지정된 디렉터리를 검색하여 파일의 위치를 찾음
- ex) 환경 변수에 지정된 디렉터리들을 확인

```
[user1@localhost ch2]$ echo $PATH  
/home/user1/.local/bin:/home/user1/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin  
[user1@localhost ch2]$ echo $MANPATH
```

- ex) mv 명령의 위치를 whereis 명령으로 검색

```
[user1@localhost ch2]$ whereis mv  
mv: /usr/bin/mv /usr/share/man/man1/mv.1.gz /usr/share/man/man1p/mv.1p.gz
```

03. 파일 관련 명령

■ 파일 관련 기타 유용한 명령

- 명령의 위치 찾기: whereis, which
 - which 명령 : 앤리어스나 PATH 환경 변수로 지정된 경로에서 파일을 찾음

which

- 기능 명령 파일의 위치를 찾아서 그 경로나 앤리어스를 출력한다.
- 형식 which [명령]
- 사용 예 which ls

- which 명령은 파일을 찾으면 절대 경로를 출력하고 바로 종료
- 최대 하나의 경로만을 출력하며 이 경로는 우리가 명령을 입력할 때 실행되는 파일

```
[user1@localhost ch2]$ which mv  
/usr/bin/mv
```

03. 파일 관련 명령

■ [따라해보기] grep, find 명령 사용하기

- ① /etc/services 파일에서 문자열 'MacOS'가 있는 행을 찾아 행 번호와 함께 출력

```
[user1@localhost ch2]$ grep -n MacOS /etc/services
1437:mac-srvr-admin 660/tcp          # MacOS Server Admin
1438:mac-srvr-admin 660/udp          # MacOS Server Adminnn
```

- ② 홈 디렉터리에서 파일명이 data1.cp인 파일이 있는지 검색

```
[user1@localhost ch2]$ find ~ -name data1.cp
/home/user1/linux_ex/ch2/data1.cp
```

- ③ 홈 디렉터리에서 파일명이 data1.cp인 파일을 찾아 temp 디렉터리로 이동

```
[user1@localhost ch2]$ find ~ -name data1.cp -exec mv {} temp \;
[user1@localhost ch2]$ ls temp
data1.cp  hosts  text1  text2
```

CentOS 리눅스

시스템 & 네트워크



Chapter 03. 문서 편집

목차

00. 개요

01. 리눅스의 문서 편집기

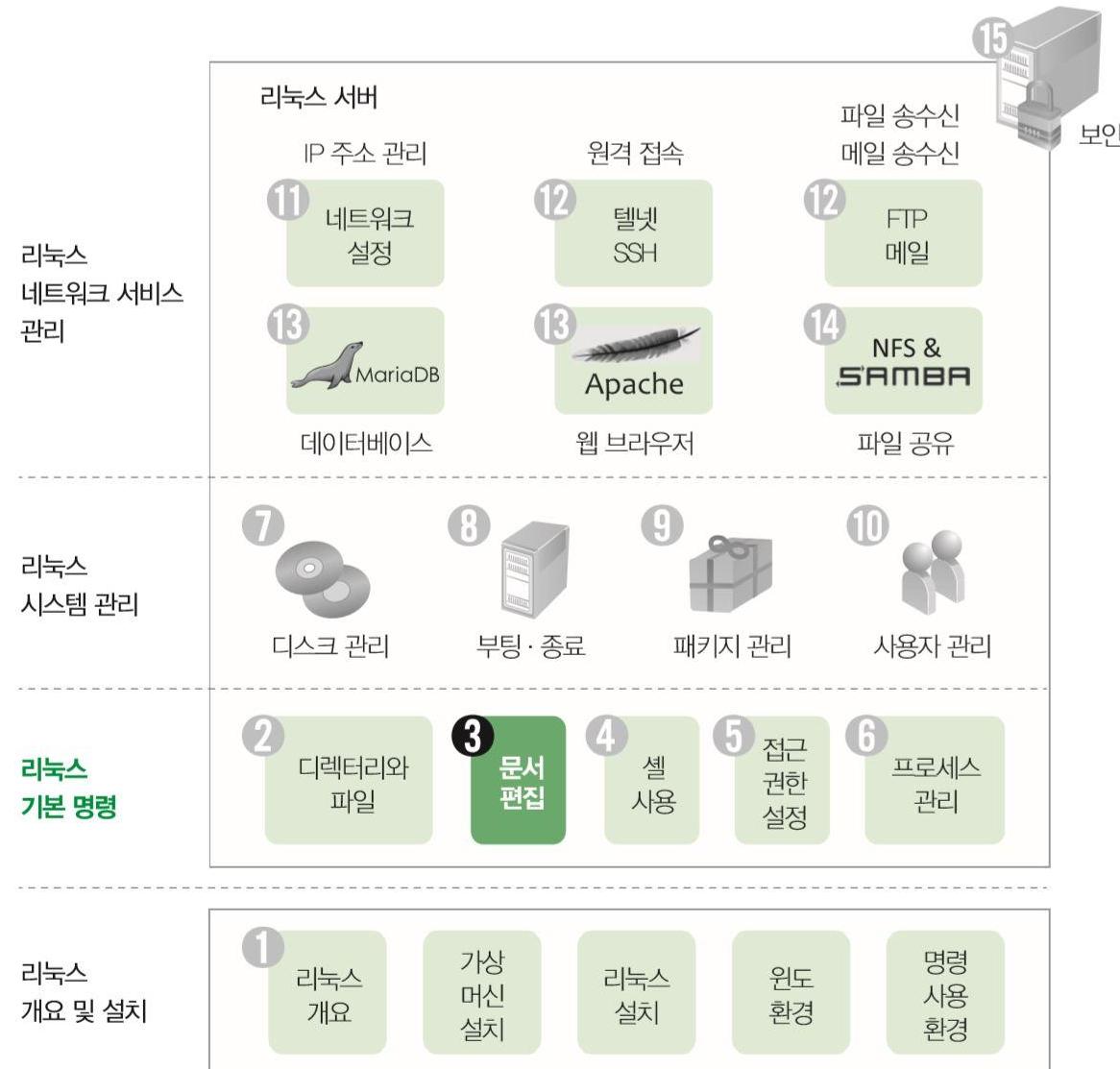
02. vi 사용법

03. vi 환경 설정

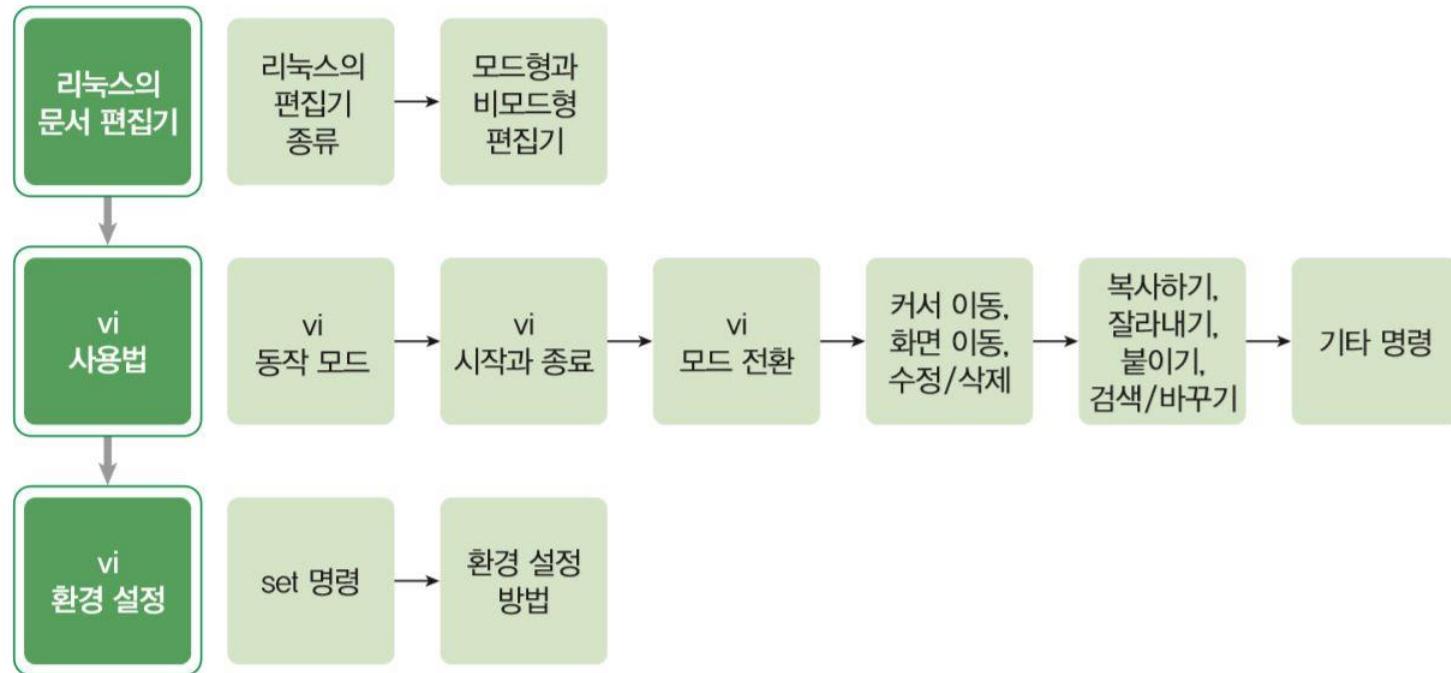
학습목표

- 리눅스에서 사용하는 편집기의 종류를 알아본다.
- 대표적인 화면 편집기인 vi를 사용할 수 있다.
- vi의 환경을 설정할 수 있다.

00. 개요



00. 개요



01. 리눅스의 문서 편집기

■ 리눅스의 편집기 종류

- GUI 환경인 그놈에서 제공하는 gedit
- 유닉스에서부터 사용했던 행 편집기(라인 편집기)와 화면 편집기

표 3-1 리눅스의 편집기 종류

구분	종류
행 단위 편집기	ed, ex, sed
화면 단위 편집기	vi, emacs(이맥스)
GUI 편집기	gedit

■ 행 단위 편집기

- ed : 유닉스 초기의 행 편집기로 사용이 불편하여 거의 사용 않음
- ex : 행 편집기이지만 단독으로 사용하기보다는 vi에 연결하여 vi를 더욱 강력하게 하는 다양한 기능을 제공
- sed : 스트림 편집기로, 일반 편집기와 달리 지시된 명령에 따라 파일의 내용을 일괄적으로 바꿔서 출력해줌

■ 화면 단위 편집기

- vi : 리눅스에서 일반적으로 사용할 수 있는 화면 편집기
- emacs(이맥스) : 제공하는 기능이 매우 다양하지만 사용법이 어렵고 복잡하여 전문적인 애호가 위주로 사용
(GNU Emacs는 무료로 배포되며, 별도로 설치해야 함)

01. 리눅스의 문서 편집기

■ 모드형과 비모드형 편집기

■ 모드형

- 입력 모드와 명령 모드가 구분
- 입력 모드 : 텍스트를 입력할 수 있는 모드
- 명령 모드 : 텍스트를 수정하고, 삭제하고, 복사와 붙이기 등 편집을 하는 모드
- 같은 글자라도 입력 모드에서는 텍스트로 처리하여 입력되고, 명령 모드에서는 텍스트로 입력되는 것이 아니라 편집 명령으로 사용
- vi는 모드형 편집기

■ 비모드형

- 입력 모드와 명령 모드가 구분되어 있지 않음
- 편집 기능을 Ctrl이나 Alt 같은 특수 키와 함께 사용
- 한글과 워드는 비모드형 편집기

표 3-2 모드형과 비모드형 편집기의 비교

구분	모드형(vi)		비모드형(메모장)
입력 모드	텍스트 입력		
명령 모드의 예	복사	yy	Ctrl+C
	붙이기	p	Ctrl+V
	저장	:wq, ZZ	Ctrl+S
모드 전환	i, a, o, Esc		해당 없음

02.vi 사용법

■ vi 동작 모드

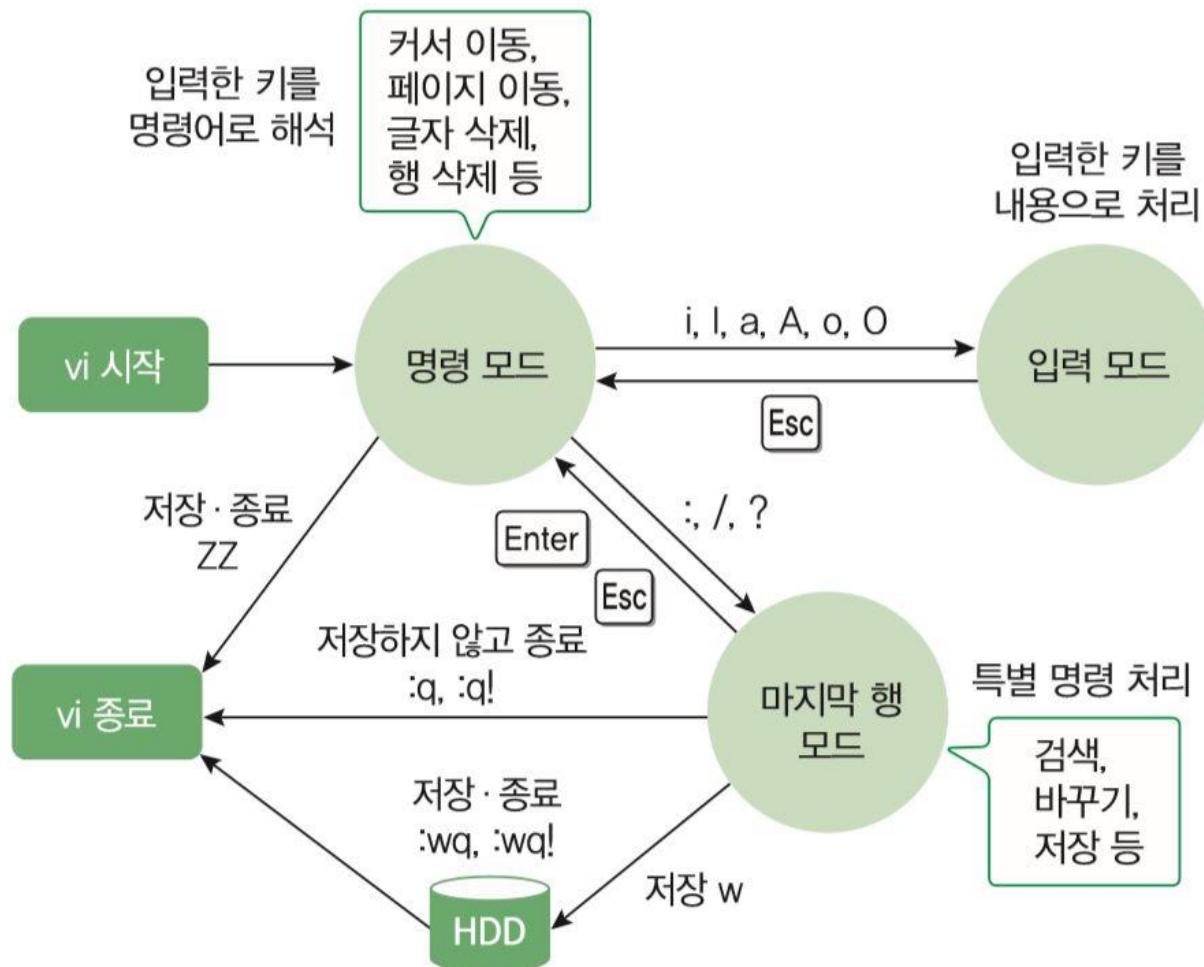


그림 3-1 vi 동작 모드

02.vi 사용법

■ vi 시작과 종료

vi

- **기능** 지정한 파일을 편집한다. 파일명을 지정하지 않으면 빈 파일이 열리고, 이 빈 파일의 파일명은 별도로 정할 수 있다.
- **형식** vi [파일]

■ vi 시작하기

- 파일을 지정할 경우: 해당 파일이 있으면 파일의 내용이 보이고, 없는 파일이면 빈 파일이 열린다.
- 파일을 지정하지 않을 경우: 그냥 빈 파일이 열린다(파일명은 저장할 때 지정 가능)

```
[user1@localhost ~]$ vi test.txt      → test.txt라는 파일이 열린다. 이 파일이 없으면  
                                         빈파일이 열린다.
```

```
[user1@localhost ~]$ vi      → 빈 파일이 열린다. 나중에 파일명을 정할 수 있다.
```

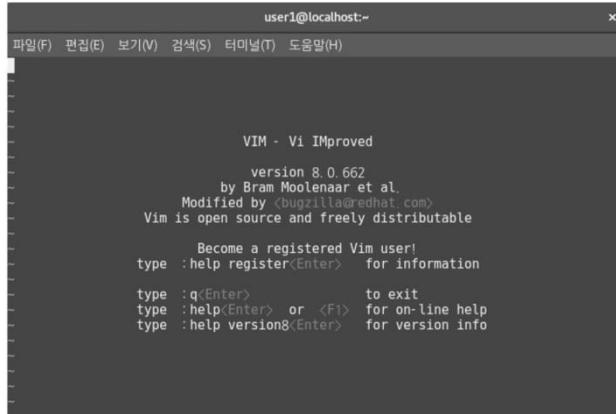


그림 3-2 vi의 초기 화면

02.vi 사용법

■ vi 시작과 종료

- vi 파일 저장하고 종료하기
 - 명령 모드나 마지막 행 모드에서 저장하고 종료 가능
 - 파일의 저장과 종료를 동시에 수행할 수 있는 명령키 [shift+zz]
 - 행 모드의 명령에서 !(느낌표)는 강제의 의미

표 3-3 vi의 저장과 종료 명령키

모드	명령키	기능
마지막 행 모드	:q	vi에서 작업한 것이 없을 때 그냥 종료한다.
	:q!	작업한 내용을 저장하지 않고 종료한다.
	:w 파일명	작업한 내용을 저장만 한다. 파일명을 지정하면 다른 파일로 저장한다.
	:wq, :wq!	작업한 내용을 저장하고 vi를 종료한다.
명령 모드	ZZ([shift]+zz)	작업한 내용을 저장하고 vi를 종료한다.

02.vi 사용법

■ vi 모드 전환

- vi는 처음에 명령 모드로 시작하므로 내용을 입력하려면 입력 모드로 전환해야 함

표 3-4 입력 모드 전환 명령키

명령키	기능
i	커서 앞에 입력한다(현재 커서 자리에 입력한다).
a	커서 뒤에 입력한다(현재 커서 다음 자리에 입력한다).
o	커서가 위치한 행의 다음 행에 입력한다.
I(대문자 i)	커서가 위치한 행의 첫 칼럼으로 이동하여 입력한다.
A	커서가 위치한 행의 마지막 칼럼으로 이동하여 입력한다.
O	커서가 위치한 행의 이전 행에 입력한다.

02.vi 사용법

■ vi 모드 전환

- i 명령키를 사용해 입력 모드로 전환하기
 - ch3 폴더를 만들고 그 아래의 test.txt 파일에서 작업

```
[user1@localhost ch3]$ vi test.txt
```

- vi를 실행한 뒤 명령 모드에서 i 명령 키를 입력하고 나서 다음 내용을 입력

```
CentOS linux study      → [Enter] 키를 누르면 다음 행으로 이동한다.  
I like linux[          → [Esc] 키를 누르면 명령 모드로 전환된다.  
~  
~  
~
```

- 입력 모드에서 다시 명령 모드로 전환하기 위해 Esc 키를 누르면 커서가 x 위로 이동

```
CentOS linux study  
I like linux[          → 명령 모드로 전환되고 커서가 x 위로 이동한다.  
~  
~  
~
```

02.vi 사용법

■ vi 모드 전환

- i와 a 명령키의 차이
 - 명령키 i는 커서 이전 위치에, a는 커서 다음 위치에 입력
 - 커서가 마지막 글자인 x 위에 있는 상태에서 i를 입력하고 [Space bar]+CentOS를 입력하면 커서의 위치인 x의 이전에 CentOS가 입력

```
CentOS linux study
I like linu CentoSx
~
```

→ 명령 모드로 전환되고 커서가 S 위로 이동한다.

- a 키로 입력 모드로 전환한 후 [Space bar]+linu를 입력하면 커서가 위치한 S 다음에 글자가 입력
- esc를 누르면 명령 모드로 전환되고 입력한 글자 중 마지막 글자인 u 위에 커서가 놓임

```
CentOS linux study
I like linu CentoS linux
~
```

→ 명령 모드로 전환되고 커서가 u 위로 이동한다.

02.vi 사용법

■ vi 모드 전환

- o 명령키를 사용하여 입력 모드로 전환
 - 명령 모드에서 현재 커서가 위치한 후 다음 행에 글자를 입력하려면 o를 누름
 - esc를 누르면 다시 명령 모드로 전환되고 커서는 아랫 행의 첫 칼럼 자리에 위치

```
CentOS linux study
I like linu CentOS linux      → 명령 모드에서 o를 입력하면 커서가 아랫 행으로 이동한다.
```

█

~

~

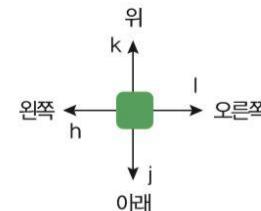
02.vi 사용법

■ 커서와 화면 이동

▪ 커서 이동하기

표 3-5 커서 이동 명령키

명령키	기능
k / j	커서를 한 행 위 / 아래로 이동한다.
l / h	커서를 한 글자 오른쪽 / 왼쪽으로 이동한다.
^ 또는 0 / \$	커서를 현재 행의 처음 / 마지막으로 이동한다.
- / + 또는 Enter ↵	커서를 앞 / 다음 행의 처음으로 이동한다.
H / M / L	커서를 화면의 맨 윗 / 중간 / 맨 아랫 행으로 이동한다.
w / e	커서를 다음 단어의 첫 / 마지막 글자로 이동한다.
b	커서를 앞 단어의 첫 글자로 이동한다.



02.vi 사용법

■ 커서와 화면 이동

▪ 커서 이동하기

- ① k 명령키를 입력하면 커서가 w 바로 위에 있는 t로 이동
- ② j 명령키를 입력하면 커서가 w 바로 아래에 있는 공백으로 이동
- ③ w 명령키를 입력하면 커서가 다음 단어 be의 첫 글자인 b로 이동
- ④ b 명령키를 입력하면 커서가 앞 단어 you의 첫 글자인 y로 이동
- ⑤ e 명령키를 입력하면 커서가 현재 단어 will의 마지막 글자인 l로 이동
- ⑥ 만약 다시 e 명령키를 입력하면 커서가 다음 단어 be의 마지막 글자인 e로 이동
- ⑦ ^ (캐럿) 명령키([shift]+6)를 입력하면 커서가 현재 행의 첫 글자인 F로 이동
- ⑧ - 명령키를 사용하면 커서가 윗행의 첫 글자인 Y로 이동
- ⑨ + 명령키를 사용하면 커서가 아랫행의 첫 글자인 H로 이동
- ⑩ H 명령키를 입력하면 커서가 화면의 첫 행으로 이동 M 명령키를 입력하면 커서가 화면의 중간 행으로 이동
- ⑪ L 명령키를 입력하면 커서가 화면의 마지막 행으로 이동, 내용이 화면 크기보다 작으면 내용의 마지막 행으로 커서가 이동
- ⑫ h나 l(소문자 L) 명령키를 입력하면 커서가 좌우로 한 칸씩 이동

1 Come, have breakfast.
10

2 Look at my hands and my feet.

3 He saw and believed.

4 Who do you say that I am?

5 I do know him and I keep his word.
11

6 You always have the poor with word.
8 1

7 Forgive and you will be forgiven.
6 4 5 3 7

8 He loved his own to the end.
9 2

9 The Lord is with you.
12

그림 3-3 커서 이동 명령키의 예

02.vi 사용법

■ 커서와 화면 이동

■ 커서 이동하기

- 초기의 유닉스 vi : 화살표 키로 커서를 이동할 수 없었음
- 리눅스 vi : 화살표 키로도 커서를 이동할 수 있음

표 3-6 기존 vi 명령키와 방향키, [Home] · [End]

명령키	방향키	명령키	방향키, [Home] · [End]
k	위 방향키(↑)	h	왼쪽 방향키(←)
j	아래 방향키(↓)	^ 또는 0	[Home]
l	오른쪽 방향키(→)	\$	[End]

02.vi 사용법

■ 화면 이동하기

- 파일 크기가 터미널의 화면 크기보다 클 경우 화면을 이동해야 함
- 아래 명령어 외에 [page up], [page down] 도 화면이동이 가능

표 3-7 화면 이동 명령기

기존 명령기	기능	추가 명령기
^u (Ctrl +u)	반 화면 위로 이동한다.	
^d (Ctrl +d)	반 화면 아래로 이동한다.	
^b (Ctrl +b)	한 화면 위로 이동한다.	Page Up
^f (Ctrl +f)	한 화면 아래로 이동한다.	Page Down
^y (Ctrl +y)	화면을 한 행만 위로 이동한다.	
^e (Ctrl +e)	화면을 한 행만 아래로 이동한다.	

```
#include <stdio.h>
main()
{
    printf("linux 01\n");
    printf("linux 02\n");
    printf("linux 03\n");
    printf("linux 04\n");
    printf("linux 05\n");
    printf("linux 06\n");
    printf("linux 07\n");
    printf("linux 08\n");
}
```

그림 3-4 화면 이동 명령기의 사용

02.vi 사용법

■ 화면 이동하기

▪ 특정 행으로 바로 이동하기

- 원하는 행으로 커서를 바로 이동
- 예시)
 - 50G → 50번째 행으로 이동
 - :30[Enter] → 30번째 행으로 이동
 - 행 번호 대신 \$를 입력하면 파일의 마지막 행으로 이동

표 3-8 특정 행으로 바로 이동하는 명령키

명령키	기능
G(<small>Shift</small> +g)	파일의 마지막 행으로 커서가 이동한다.
행 번호G(<small>Shift</small> +g)	지정한 행 번호로 커서가 이동한다.
:행 번호	지정한 행 번호로 커서가 이동한다(마지막 행 모드).
:\$	파일의 마지막 행으로 커서가 이동한다(마지막 행 모드).

02.vi 사용법

■ 내용 수정

- 수정을 마치면 [esc] 를 눌러서 명령 모드로 돌아가야 수정이 완료

표 3-9 내용 수정 명령키

명령키	기능
r	커서가 위치한 글자를 다른 글자로 수정한다.
cw, #cw	커서 위치부터 현재 단어의 끝까지 수정한다. #에는 수정할 단어의 수를 지정한다. 예를 들면 3cw는 커서 위치부터 세 단어를 수정한다.
s, #s	커서 위치부터 Esc를 입력할 때까지 수정한다. #에는 수정할 글자의 수를 지정한다. 예를 들면 5s는 커서 위치부터 다섯 글자를 수정한다.
cc	커서가 위치한 행의 내용을 모두 수정한다.
C	커서 위치부터 행의 끝까지 수정한다.

02.vi 사용법

■ 내용 수정

▪ 한 글자 수정하기: r 명령 키

- 수정하려는 글자 위에 커서를 놓은 후 r 명령 키를 먼저 입력하고 바꾸려는 새 글자를 입력
- r 명령 키는 계속 명령 모드를 유지하므로 수정한 후 Esc키를 누를 필요가 없음

```
CentOS [winux study]      → r 명령키로 글자를 수정한다(l→w).
```

```
I like linu CentOS linux
```

```
~
```

▪ 단어 수정하기: cw, #s 명령 키

- 한 단어를 바꿀 때는 cw 명령 키나 #s 명령 키를 사용
- vi에서는 단어를 공백문자나 특수문자로 구분
 - ex) 'winux'를 다른 단어로 바꾸려면 cw 명령 키나, 글자 수가 다섯 개이므로 5s로 수정

```
CentOS [study]      → cw 명령키 입력 시 winux가 사라진다.
```

```
I like linu CentOS linux
```

```
~
```

```
(생략)
```

```
~
```

```
— 끼워넣기 —
```

```
CentOS editor [study]      → 수정 완료 후 Esc를 눌러야 명령 모드로 전환된다.
```

```
I like linu CentOS linux
```

```
~
```

02.vi 사용법

■ 내용 수정

- 행 단위 수정하기: C, cc 명령키
 - C 명령키
 - 현재 커서가 위치한 r부터 행의 끝까지 수정하려면 C 명령키를 사용
 - r부터 행의 마지막까지 모두 지워지고 입력 모드로 전환되어 입력을 기다림

```
CentOS editor [r] → r부터 모두 지워지고 입력 모드로 전환된다.  
I like linu CentOS linux
```

~
(생략)
~
-- 끼워넣기 --

- 수정할 대상의 글자 수와 상관없이 원하는 대로 입력

```
CentOS editor vi  
I like linu CentOS linux
```

~

02.vi 사용법

■ 내용 수정

▪ 행 단위 수정하기: C, cc 명령키

- cc 명령키
 - cc 명령키를 입력하면 현재 행의 모든 내용이 삭제되고, 커서가 행의 처음으로 이동하여 새로운 입력을 기다림

→ 모두 지워지고 행의 처음으로 이동한다. 입력 모드로 전환된다.

```
I like linu CentOS linux
```

~

(생략)

~

-- 끼워넣기 --

- 원하는 내용을 입력하고 [esc] 를 눌러 명령 모드로 전환하면 수정이 완료

```
CentOS editor vi study  
I like linu CentOS linux
```

~

02.vi 사용법

■ 내용 삭제

- 입력 모드에서 내용을 입력하는 도중에 틀린 글자를 삭제할 때는 [←] 키나 [Delete]를 사용
- 명령 모드에서 글자나 행을 삭제하려면 [표 3-10]과 같은 명령키를 사용
- 한 번에 여러 글자나 행을 삭제하려면 명령키 앞에 글자나 행의 수를 숫자로 지정
 - ex) 한 글자를 삭제하는 것은 x 명령키, 세 글자를 삭제하려면 3x

표 3-10 내용 삭제 명령키

명령키	기능
x, #x	커서 위치의 글자를 삭제한다. #에는 삭제할 글자 수를 지정한다.
dw, #dw	커서 위치의 단어를 삭제한다. #에는 삭제할 단어 수를 지정한다.
dd, #dd	커서 위치의 행을 삭제한다. #에는 삭제할 행의 수를 지정한다.
D([Shift]+d)	커서 위치부터 행의 끝까지 삭제한다.

- x 명령 키로 현재 커서가 놓인 'l' 한 글자만 삭제

```
CentOS editor vi study  
I like linu CentOS linux
```

~

```
CentOS editor vi study  
I like inu CentOS linux
```

→ x 명령키로 커서 위치의 l을 삭제한다.

~

02.vi 사용법

■ 내용 삭제

- 나머지 글자를 모두 지우려면 글자 수를 지정하여 3x를 입력하거나 단어를 지우는 dw 명령키를 사용
- 3x로 지우는 경우와 dw로 지우는 경우, 지워지고 난 후 커서의 위치에 차이가 있음

```
CentOS editor vi study  
I like CentOS linux      → dw로 삭제하면 커서가 다음 단어의 첫 글자로 이동한다.
```

~

```
CentOS editor vi study  
I like [C]entOS linux      → 3x로 삭제하면 커서가 공백문자에 위치한다.
```

~

- 현재 커서 위치부터 행의 끝까지 삭제하려면 D([Shift]+d) 명령키를 입력

```
CentOS editor vi study  
I like [D]
```

~

- 커서가 위치한 현재 행을 지우려면 dd 명령키를 입력

```
CentOS editor vi study  
[D]
```

~

02.vi 사용법

■ 명령 취소

표 3-11 이전 명령 취소 명령키

명령키	기능
u	명령을 취소한다.
U	해당 행에서 한 모든 명령을 취소한다.
:e!	마지막으로 저장한 내용 이후의 것을 버리고 새로 작업한다.

- u 명령키를 입력하면 앞의 예에서 삭제되었던 행이 복구

```
CentOS editor vi study  
I like CentOS linux  
~
```

- 모든 작업을 취소하고 원래대로 복구하려면 U 명령키를 이용

02.vi 사용법

■ [따라해보기] vi로 입력·수정·삭제·복구하기

- ① vi로 새로운 파일인 exec.txt 파일을 실행

```
[user1@localhost ch3]$ vi exec.txt
```

- ② i를 입력하여 입력 모드로 전환하고 다음 내용을 입력

```
Good morning everyone.  
Nice to meet you.  
I am a linux beginner.  
Now introduce yourself.  
~  
~
```

- ③ [esc]를 입력하여 명령 모드로 전환하고 :w 명령으로 파일 내용을 저장

```
Good morning everyone.  
Nice to meet you.  
I am a linux beginner.  
Now introduce yourself.  
~  
:w
```

02.vi 사용법

■ [따라해보기] vi로 입력·수정·삭제·복구하기

- ④ 커서를 3행의 beginner로 이동
3G와 l 명령키(또는 오른쪽 화살표나 w 명령키와 h 명령키)를 사용하여 이동할 수 있음

```
Good morning everyone.  
Nice to meet you.  
I am a linux beginner.  
Now introduce yourself.  
~  
~
```

- ⑤ beginner를 expert로 수정cw나 8s 명령키를 사용한다. 수정 후에는[esc]를 눌러 명령 모드로 전환

```
Good morning everyone.  
Nice to meet you.  
I am a linux expert.  
Now introduce yourself.  
~  
~
```

- ⑥ - 명령키로 2행의 첫 글자인 N으로 이동

```
Good morning everyone.  
Nice to meet you.  
I am a linux expert.  
Now introduce yourself.  
~  
~
```

02.vi 사용법

■ [따라해보기] vi로 입력·수정·삭제·복구하기

- ⑦ w 명령키로 커서를 meet로 이동하고, 단어 meet를 dw 명령키로 삭제

```
Good morning everyone.  
Nice to you.  
I am a linux expert.  
Now introduce yourself.  
~  
~
```

- ⑧ 단어 y부터 행의 끝까지 D([Shift]+d) 명령키로 삭제

```
Good morning everyone.  
Nice to  
I am a linux expert.  
Now introduce yourself.  
~  
~
```

02.vi 사용법

■ [따라해보기] vi로 입력·수정·삭제·복구하기

- ⑨ U 명령키로 2행에서 한 모든 삭제를 취소

```
Good morning everyone.  
Nice to meet you.  
I am a linux expert.  
Now introduce yourself.  
~  
~
```

- ⑩ :wq 명령으로 파일 내용을 저장하고 종료

```
Good morning everyone.  
Nice to meet you.  
I am a linux expert.  
Now introduce yourself.  
~  
:wq
```

02.vi 사용법

■ 복사 및 잘라 붙이기

- 복사하기, 잘라내기, 붙이기

표 3-12 복사하기, 잘라내기, 붙이기 명령키

명령키	기능
yy, #yy	커서가 위치한 행을 복사한다. #에는 복사할 행의 수를 지정한다.
p	커서가 위치한 행의 아래쪽에 붙인다.
P	커서가 위치한 행의 위쪽에 붙인다.
dd, #dd	커서가 위치한 행을 잘라둔다. 삭제와 같은 기능이다. #에는 잘라둘 행의 수를 지정한다.

- yy 명령키로 해당 행을 복사하고고, 원하는 행으로 커서를 이동하여 p 명령 키를 입력하면 '복사하여 붙이기'
 - dd 명령키로 행을 삭제한 후 p 명령키를 입력하면 '잘라서 붙이기'
-
- 복사하거나 잘라내기를 하면 내용이 임시 버퍼에 저장되기 때문에 복사하거나 잘라내기를 한 뒤에는 다른 명령을 사용하지 말고 즉시 원하는 위치로 이동하여 붙이기를 하는 것이 좋음

02.vi 사용법

■ 복사 및 잘라 붙이기

▪ 복사하기, 잘라내기, 붙이기

- 앞서 저장해둔 test.txt 파일을 vi로 열면 커서는 f 위에 위치

```
CentOS editor vi study
I like CentOS linux
~
```

- 현재 커서가 있는 행만 복사하려면 그냥 yy 명령 키만 입력
- 1행과 2행을 함께 복사하려면 2yy를 입력
- ex) 2yy 명령 키로 두 행을 모두 복사한 다음 아랫행으로 이동하여 p 명령 키로 붙이기

```
CentOS editor vi study
I like CentOS linux
CentOS editor vi study
I like CentOS linux
~
```

- dd 명령 키는 삭제뿐만 아니라 잘라내기를 할 때도 사용
- ex) 현재 위치인 3행을 잘라 2행 위에 붙이기 → dd 키를 입력한 다음 커서를 2행으로 이동하여 P 키를 입력

```
CentOS editor vi study
CentOS editor vi study
I like CentOS linux
I like CentOS linux
~
```

02.vi 사용법

■ 복사 및 잘라 붙이기

▪ 네임드 버퍼 사용하기

- 언네임드 버퍼
 - 이름을 붙이지 않은 버퍼
 - yy 명령 키로 복사하거나 dd 명령 키로 잘라낼 경우, 복사하거나 잘라낸 내용이 저장
 - 언네임드 버퍼는 내용을 하나만 저장
- 네임드 버퍼
 - 이름을 붙여서 사용할 수 있는 버퍼
 - 네임드 버퍼를 사용하면 각각 다른 이름을 붙인 버퍼에 독립적으로 내용을 저장하고 사용
 - 네임드 버퍼에 이름을 붙일 때는 "+문자 또는 "+숫자(숫자 버퍼)의 형태로 사용
 - 네임드 버퍼 : "a, "b, "c, "d, ..., "z
 - 숫자 버퍼 : "1, "2, ..., "9
 - 네임드 버퍼에 커서가 위치한 행을 저장하려면 '버퍼 이름+yy'를 입력
 - "a 버퍼에 복사한 내용을 저장하려면 "ayy를 입력
 - 네임드 버퍼의 내용을 붙이려면 "ap와 같이 '버퍼 이름+p'를 입력

02.vi 사용법

■ 복사 및 잘라 붙이기

▪ 네임드 버퍼 사용하기

- 현재 파일 내용

```
CentOS editor vi study
CentOS editor vi study
I like CentOS linux
I like CentOS linux
~
```

- 2행을 "a 버퍼에 잘라서 저장하고, "a를 먼저 입력 후 dd 입력

```
CentOS editor vi study
I like CentOS linux
I like CentOS linux
~
```

- 다시 2행을 잘라 "b 버퍼에 저장하고

- 커서를 1행으로 이동
- "b 버퍼의 내용을 1행 위 행에 붙여 넣으려면 "bP를 입력
- "a 버퍼의 내용을 1행 아래 행에 붙여 넣으려면 "ap를 입력

```
I like CentOS linux
CentOS editor vi study
CentOS editor vi study
I like CentOS linux
~
```

02.vi 사용법

■ 복사 및 잘라 붙이기

▪ 네임드 버퍼 사용하기

- 현재 파일 내용

```
CentOS editor vi study
CentOS editor vi study
I like CentOS linux
I like CentOS linux
~
```

- 2행을 "a 버퍼에 잘라서 저장하고, "a를 먼저 입력 후 dd 입력

```
CentOS editor vi study
I like CentOS linux
I like CentOS linux
~
```

- 다시 2행을 잘라 "b 버퍼에 저장하고

- 커서를 1행으로 이동
- "b 버퍼의 내용을 1행 위 행에 붙여 넣으려면 "bP를 입력
- "a 버퍼의 내용을 1행 아래 행에 붙여 넣으려면 "ap를 입력

```
I like CentOS linux
CentOS editor vi study
CentOS editor vi study
I like CentOS linux
~
```

02.vi 사용법

■ 복사 및 잘라 붙이기

- 마지막 행 모드에서 복사하기, 잘라내기, 붙이기
 - 범위 지정하기
 - .(마침표)는 커서가 위치한 현재 행을, \$는 마지막 행을, %는 전체 행을 의미

표 3-13 범위 지정 명령기

명령기	기능
1, \$ 또는 %	1행부터 마지막 행까지 지정한다.
1..	1행부터 커서가 있는 행까지 지정한다.
.,\$	커서가 있는 행부터 마지막 행까지 지정한다.
-3	현재 행과 이전 세 행까지(총 네 행) 지정한다.
10, 20	10행부터 20행까지 지정한다.

- 복사하기, 잘라내기, 붙이기

표 3-14 마지막 행 모드에서의 복사하기, 잘라내기, 붙이기 명령기

명령기	기능
:#y	#로 지정한 행을 복사한다. 예를 들면 3y는 세 행을 복사한다.
:<범위>y	범위로 지정한 행을 복사한다. 예를 들면 2,4y는 2~4행을 복사한다.
:#d	#로 지정한 행을 잘라낸다(삭제). 예를 들면 3d는 세 행을 잘라낸다.
:<범위>d	범위로 지정한 행을 잘라낸다(삭제). 예를 들면 1,4d는 1~4행을 잘라낸다.
:pu	현재 행 다음에 버퍼의 내용을 붙인다.
:#pu	#로 지정한 행 다음에 버퍼의 내용을 붙인다. 예를 들면 4pu와 같이 지정한다.

02.vi 사용법

■ 복사 및 잘라 붙이기

- 마지막 행 모드에서 복사하기, 잘라내기, 붙이기

- 복사하기, 잘라내기, 붙이기
 - 커서는 2행에 위치

```
I like CentOS linux  
CentOS editor vi study  
CentOS editor vi study  
I like CentOS linux  
~
```

- :2,3d 명령키로 2행과 3행을 잘라내고 :2pu 명령키로 2행 다음에 잘라낸 것을 붙임

```
I like CentOS linux  
I like CentOS linux  
CentOS editor vi study  
CentOS editor vi study  
~
```

- :1,2y 명령키로 1행부터 2행까지 복사하고 :4pu 명령키로 붙여넣음

```
I like CentOS linux  
I like CentOS linux  
CentOS editor vi study  
CentOS editor vi study  
I like CentOS linux  
I like CentOS linux  
~
```

02.vi 사용법

■ 검색 및 바꾸기

■ 검색하기

- 검색하기 위해 마지막 행으로 이동할 때는 :이 아니라 /이나 ?를 입력
- /이나 ? 다음에 찾고자 하는 문자열을 입력하고
- [Enter]를 누르면 검색이 진행되고, 해당 문자열을 찾으면 그 문자열의 시작 위치로 커서가 이동

표 3-15 검색 명령어

명령어	기능
/문자열	문자열을 아래 방향으로 검색한다.
?문자열	문자열을 위 방향으로 검색한다.
n	원래 찾던 방향으로 다음 문자열을 검색한다.
N	역방향으로 다음 문자열을 검색한다.

02.vi 사용법

■ 검색 및 바꾸기

▪ 검색하기

- 커서가 6행에 있는 상태에서 문자열을 검색하기 위해 /을 입력하면 커서가 마지막 행으로 이동
- 검색할 문자열인 'CentOS'를 입력하고 [Enter]를 누르면 커서 위치보다 뒤쪽에 위치한 같은 행의 CentOS로 커서가 이동

```
I like CentOS linux
I like CentOS linux
CentOS editor vi study
CentOS editor vi study
I like CentOS linux
I like CentOS linux
~
(생략)
/CentOS
```

- 계속 CentOS를 검색하려고 n을 입력하면 6행이 파일의 마지막 행이므로 '끝까지 찾았음, 처음부터 계속'이라는 메시지를 출력하고 1행의 CentOS로 커서가 이동

```
I like CentOS linux
I like CentOS linux
CentOS editor vi study
CentOS editor vi study
I like CentOS linux
I like CentOS linux
~
(생략)
끝까지 찾았음, 처음부터 계속
```

02.vi 사용법

■ 검색 및 바꾸기

▪ 바꾸기

- 기존의 문자열을 다른 문자열로 바꾸려면 먼저 :을 입력하여 마지막 행 모드로 전환
- 커서 위치의 문자열만 바꿀 수도 있고, 파일 전체나 특정 범위 내에서 해당하는 문자열을 모두 바꿀 수도 있음

표 3-16 바꾸기 명령기

명령기	기능
:s/문자열1/문자열2/	커서가 위치한 행에서 첫 번째로 나오는 문자열1을 문자열2로 바꾼다.
:%s/문자열1/문자열2/g	파일 전체에서 모든 문자열1을 문자열2로 바꾼다.
:<범위>s/문자열1/문자열2/	범위 내 모든 각 행에서 첫 번째로 나오는 문자열1을 찾아 문자열2로 바꾼다.
:<범위>s/문자열1/문자열2/g	범위 내 모든 행에서 문자열1을 문자열2로 바꾼다.
:<범위>s/문자열1/문자열2/gc	범위 내 모든 행에서 문자열1을 문자열2로 바꿀 때 수정할지 여부를 묻는다.

02.vi 사용법

■ 검색 및 바꾸기

▪ 바꾸기

- ex) 현재 커서의 CentOS를 CENTOS로 변경할 경우 명령은 :s/CentOS/CENTOS/

```
I like CENTOS linux
I like CentOS linux
CentOS editor vi study
CentOS editor vi study
I like CentOS linux
I like CentOS linux
~
(생략)
:s/CentOS/CENTOS/
```

- ex) 3~4행에 있는 editor를 CentOS로 바꿀 경우 의 명령은 :3,4s/editor/CentOS/

```
I like CENTOS linux
I like CentOS linux
CentOS CentOS vi study
CentOS CentOS vi study
I like CentOS linux
I like CentOS linux
~
(생략)
:3,4s/editor/CentOS/
```

02.vi 사용법

■ 검색 및 바꾸기

▪ 바꾸기

- 명령에 g가 있을 경우 해당 행이나 범위 내에서 바꾸려는 문자열을 모두 찾아 다른 문자열로 바꿈
- ex) 행에서 :s/fedora/FEDORA/을 수행하고 4행에서 :s/fedora/FEDORA/g를 수행하여 결과의 차이를 비교

```
I like CENTOS linux
I like CentOS linux
CENTOS CentOS vi study
CENTOS CENTOS vi study
I like CentOS linux
I like CentOS linux
~
(생략)
:s/CentOS/CENTOS/g
```

- ex) 파일 전체의 문자열을 바꿀 경우 :%s/CentOS/CENTOS/g 또는 :1,\$s/CentOS/CENTOS/g를 사용

```
I like CENTOS linux
I like CENTOS linux
CENTOS CENTOS vi study
CENTOS CENTOS vi study
I like CENTOS linux
I like CENTOS linux
~
(생략)
4개 바꿨음 4 행에서
```

02.vi 사용법

■ 검색 및 바꾸기

▪ 바꾸기

- 명령에 g가 있을 경우 해당 행이나 범위 내에서 바꾸려는 문자열을 모두 찾아 다른 문자열로 바꿈
- ex) 행에서 :s/fedora/FEDORA/을 수행하고 4행에서 :s/fedora/FEDORA/g를 수행하여 결과의 차이를 비교

```
I like CENTOS linux
I like CentOS linux
CENTOS CentOS vi study
CENTOS CENTOS vi study
I like CentOS linux
I like CentOS linux
~
(생략)
:s/CentOS/CENTOS/g
```

- ex) 파일 전체의 문자열을 바꿀 경우 :%s/CentOS/CENTOS/g 또는 :1,\$s/CentOS/CENTOS/g를 사용

```
I like CENTOS linux
I like CENTOS linux
CENTOS CENTOS vi study
CENTOS CENTOS vi study
I like CENTOS linux
I like CENTOS linux
~
(생략)
4개 바꿨음 4 행에서
```

02.vi 사용법

■ [따라해보기] vi 편집 방법 익히기

- ① vi로 새로운 파일인 exec2.txt 파일을 실행

```
[user1@localhost ch3]$ vi exec2.txt
```

- ② i를 입력하여 입력 모드로 전환하고 다음 내용을 입력

```
Good morning everyone.  
My name is Gildong Hong.  
This is a living room.  
~
```

- ③ [esc]를 입력하여 명령 모드로 전환하고 :w 명령으로 파일 내용을 저장

```
Good morning everyone.  
My name is Gildong Hong.  
This is a living room.  
~  
:w
```

- ④ 1행을 복사하여 3행 다음에 붙임

```
Good morning everyone.  
My name is Gildong Hong.  
This is a living room.  
Good morning everyone. → ① 1G 입력 → ② yy 입력 → ③ jj 입력 → ④ p 입력  
~  
~
```

02.vi 사용법

■ [따라해보기] vi 편집 방법 익히기

- ⑤ 파일에서 morning이란 단어를 검색

```
Good morning everyone.  
My name is Gildong Hong.  
This is a living room.  
Good morning everyone. → ① /morning  입력 → ② n 입력  
~  
/morning
```

- ⑥ 파일에서 morning이란 단어를 모두 찾아 afternoon으로 바꿈

```
Good afternoon everyone.  
My name is Gildong Hong.  
This is a living room.  
Good afternoon everyone. → :%s/morning/afternoon/g  입력  
~  
:%s/morning/afternoon/g
```

- ⑦ 저장한 후 종료

```
Good afternoon everyone.  
My name is Gildong Hong.  
This is a living room.  
Good afternoon everyone.  
~  
:wq
```

02.vi 사용법

■ 기타 유용한 명령키

- 파일 읽어오기, 여러 파일 편집하기

표 3-17 파일 관련 명령키

명령키	기능
:r 파일	지정한 파일을 읽어들여 현재 커서 위치에 삽입한다.
:e 파일	지정한 파일로 전환한다(기존 파일을 :w로 저장한 뒤에 실행해야 한다).
:n	vi 시작 시 여러 파일을 지정했을 경우 다음 파일로 작업을 이동한다.

02.vi 사용법

■ 기타 유용한 명령어

- 파일 읽어오기, 여러 파일 편집하기
 - 다른 파일 읽어오기
 - test.txt 파일을 열고 커서를 4행으로 이동

```
I like CENTOS linux
I like CENTOS linux
CENTOS CENTOS vi study
CENTOS CENTOS vi study
I like CENTOS linux
I like CENTOS linux
~
```

- :r exec2.txt를 실행하면 exec2.txt 파일의 내용이 test.txt 파일의 4행 다음에 삽입

```
I like CENTOS linux
I like CENTOS linux
CENTOS CENTOS vi study
CENTOS CENTOS vi study
Good afternoon everyone.
```

```
This is a living room.
My name is Gildong Park.
This is a living room.
My name is Gildong Hong.
Good afternoon everyone.
I like CENTOS linux
I like CENTOS linux
~
```

02.vi 사용법

■ 기타 유용한 명령키

- 파일 읽어오기, 여러 파일 편집하기
 - 파일 편집을 마치고 다른 파일 편집하기
 - :e 명령 키는 현재 작업 중인 파일의 작업을 마치고 다른 파일을 편집하려고 할 때 사용
 - ex) test.txt 파일 편집을 완료하고 exec.txt 파일 편집으로 바꾸려면 :e exec.txt 사용
 - 이때 작업 중이던 파일을 먼저 저장하고 :e 명령을 실행해야 함
 - 파일을 저장하지 않고 :e exec.txt를 실행하면 다음과 같은 오류 메시지가 출력

```
I like CENTOS linux
I like CENTOS linux
CENTOS CENTOS vi study
CENTOS CENTOS vi study
Good afternoon everyone.
This is a living room.
(생략)
~
E37: 마지막으로 고친 뒤 저장되지 않았습니다 (무시하려면 ! 더하기)
계속하려면 엔터 혹은 명령을 입력하십시오
```

- 여러 파일 편집하기

```
[user1@localhost ch3]$ vi test.txt exec.txt exec2.txt
```

- vi를 시작할 때 파일명을 여러 개 지정
- 파일 작업을 마치고 다음 파일로 이동하려면 :n을 입력

02.vi 사용법

■ 기타 유용한 명령키

- vi에서 셸 명령 사용하기

표 3-18 셸 명령 실행 명령키

명령키	기능
:! 셸 명령	vi 작업을 잠시 중단하고 셸 명령을 실행한다(vi로 돌아오려면 Enter键를 누른다).
:sh	vi를 잠시 빠져나가서 셸 명령을 실행한다(vi로 돌아오려면 exit 명령을 입력한다).

- ::! 명령키 이용하기
 - vi를 빠져나가거나 하는 번거로움 없이 바로 이용할 수 있다는 장점
 - 다시 vi 작업으로 돌아가려면 Enter키 입력
 - ex) test.txt 파일을 편집하는 중에 ::! ls 명령 실행

(생략)

~

~

::! ls

```
[user1@localhost ch3]$ vi test.txt
```

```
exec.txt    exec2.txt    test.txt
```

계속하려면 엔터 혹은 명령을 입력하십시오

02.vi 사용법

■ 기타 유용한 명령키

▪ vi에서 셸 명령 사용하기

- :sh 명령키 이용하기
 - ':! 셸 명령'은 한 번에 하나의 셸 명령만 실행
 - 실행할 셸 명령이 여러 개라면 :sh 명령 키로 vi를 잠시 빠져나가서 셸 작업을 수행하고 다시 돌아오는 것이 편리

(생략)

~

~

:sh

- ex) 필요한 셸 작업을 마치고 다시 vi로 돌아오려면 exit를 입력

```
[user1@localhost ch3]$ ls
exec.txt    exec2.txt    test.txt
[user1@localhost ch3]$ exit
```

02.vi 사용법

■ 기타 유용한 명령키

■ 기타 명령키

표 3-19 기타 명령키

명령키	기능
[Ctrl]+l(소문자 L)	현재 화면을 다시 출력한다.
[Ctrl]+g	현재 커서 위치의 행 번호를 마지막 행에 출력한다.
[Shift]+j(대문자 J)	현재 행과 아래 행을 연결하여 한 행으로 만든다.
.(마침표)	바로 직전에 했던 명령을 반복한다.
~(틸드)	커서 위치의 글자를 대문자나 소문자로 바꾼다.

• 화면 다시 출력하기

- vi 작업 도중에 시스템 메시지나 다른 사용자가 보낸 메시지가 출력되어 화면이 이상하게 보일 때 [Ctrl]+l(소문자 L) 명령키를 입력하면 메시지가 사라지고 원래 작업 중이던 내용이 다시 출력

02.vi 사용법

■ 기타 유용한 명령키

■ 기타 명령키

- 행 연결하기
 - 행을 연결할 때는 J 명령키를 사용
 - 커서가 위치한 행과 다음 행을 하나의 행으로 만들어줌

```
I like CENTOS linux  
I like CENTOS linux  
CENTOS CENTOS vi study  
(생략)  
~
```

- J 명령키를 입력하면 두 번째 행과 합쳐짐
- 두 행이 합쳐질 때 사이에 공백문자가 삽입되며, 커서가 그 공백문자로 이동

```
I like CENTOS linux I like CENTOS linux  
CENTOS CENTOS vi study  
(생략)  
~
```

02.vi 사용법

■ 기타 유용한 명령키

■ 기타 명령키

- 이전 명령 반복하기
 - .(마침표) 명령기는 바로 앞에 했던 명령을 반복적으로 수행
 - 커서 이동 키는 . 명령기를 입력해도 반복 수행되지 않음

```
I like CENTOS linux I like CENTOS linuxCENTOS CENTOS vi study  
(생략)
```

~

- 대·소문자 전환하기

- ~ 명령기는 커서가 놓인 곳의 글자가 소문자이면 대문자로, 대문자이면 소문자로 바꿔줌
- 앞의 예에서 커서를 x로 옮기고 ~ 명령키를 입력하면 X로 바뀌고 커서가 오른쪽으로 한 칸 이동

```
I like CENTOS linux I like CENTOS linuxXCENTOS CENTOS vi study  
(생략)
```

~

02.vi 사용법

■ [따라해보기] 기타 유용한 명령 익히기

- ① vi로 새로운 파일인 exec3.txt 파일을 염

```
[user1@localhost ch3]$ vi exec3.txt
```

- ② i를 입력하여 입력 모드로 전환하고 다음 내용을 입력

```
Good  
morning  
everyone.  
~  
:w
```

- ③ [esc]를 입력하여 명령 모드로 전환하고 :w 명령으로 파일 내용을 저장

```
Good  
morning  
everyone.  
~
```

02.vi 사용법

■ [따라해보기] 기타 유용한 명령 익히기

- ④ 3행 다음에 exec2.txt 파일을 읽어들임

```
Good
morning
everyone.

Good afternoon everyone.      → :r exec2.txt Enter를 입력한다.

This is a living room.
My name is Gildong Park.
This is a living room.
My name is Gildong Hong.
Good afternoon everyone.

~
"exec2.txt" 6L, 138C
```

- ⑤ 1행과 2행을 연결하여 한 행으로 만듬

```
Good morning      → ① 1G로 이동 → ② J 입력
everyone.

Good afternoon everyone.

(생략)

~
"exec2.txt" 6L, 138C
```

02.vi 사용법

■ [따라해보기] 기타 유용한 명령 익히기

- ⑦ 반복 명령키로 1행과 2행을 연결하여 한 행으로 만듬

Good morning everyone. → .을 입력한다.

Good afternoon everyone.

(생략)

~

"exec2.txt" 6L, 138C

- ⑧ ~명령키로 1행의 everyone을 EVERYONE으로 변경

Good morning EVERYONE. → ~~~~~ 입력

Good afternoon everyone.

(생략)

~

"exec2.txt" 6L, 138C

02.vi 사용법

■ [따라해보기] 기타 유용한 명령 익히기

- ⑧ exec3.txt 파일을 저장하고, 셀 명령을 실행하여 현재 디렉터리의 경로를 확인

```
Good morning EVERYONE.  
Good afternoon everyone.  
(생략)  
~  
:! pwd      → ① :w [Enter]로 저장 → ② :! pwd [Enter]
```

```
[user1@localhost ch3]$ vi exec3.txt  
  
/home/user1/linux_ex/ch3          → ① 경로 정보를 출력한다.  
  
계속하려면 엔터 혹은 명령을 입력하십시오      → ② [Enter]를 입력하여 vi로 복귀한다.
```

- ⑨ vi를 종료

03.vi 환경 설정

■ vi 환경 설정

- vi의 환경을 설정하는 세 가지 방법
 - 사용자 홈 디렉터리에 .exrc 파일로 저장
 - 환경 변수 EXTINIT에 지정
 - vi의 마지막 행 모드에서 명령으로 설정

표 3-20 vi 환경 설정 명령

set 명령과 옵션	기능
set nu	파일 내용의 각 행에 행 번호를 표시한다(보이기만 할 뿐 저장되지는 않는다).
set nonu	행 번호를 감춘다.
set list	눈에 보이지 않는 특수문자를 표시한다(tab :^I, eol :\$ 등).
set nolist	특수문자를 감춘다.
set showmode	현재 모드를 표시한다.
set noshowmode	현재 모드를 감춘다.
set	set으로 설정한 모든 vi 환경 설정 값을 출력한다.
set all	모든 vi 환경 변수와 현재 값을 출력한다.

03.vi 환경 설정

■ 행 번호 표시

- :set nu를 입력하면 다음과 같이 행 번호가 표시
- 행 번호는 사용자의 편의를 위해 보이는 것으로 파일에 저장되지는 않음
- :set nonu를 입력하면 행 번호가 없어짐

파일: exec2.txt	set nu 실행 후
Good afternoon everyone.	1 Good afternoon everyone.
This is a living room.	2 This is a living room.
My name is Gildong Park.	3 My name is Gildong Park.
This is a living room.	4 This is a living room.
My name is Gildong Hong.	5 My name is Gildong Hong.
Good afternoon everyone.	6 Good afternoon everyone.
~	~
:set nu	:set nu

■ 특수문자 표시

- 특수문자를 보려면 :set list 명령을 입력
- exec2.ext 파일에 탭을 하나 추가하고 :set list 명령을 실행

파일: exec2.txt	set list 실행 후
Good afternoon everyone.	Good afternoon everyone.\$
This is a living room.	This is a living room.\$
My name is Gildong Park.	My name is ^Gildong Park.\$
This is a living room.	This is a living room.\$
My name is Gildong Hong.	My name is Gildong Hong.\$
Good afternoon everyone.	Good afternoon everyone.\$
~	~
:set list	:set list

03.vi 환경 설정

■ 환경 설정 값 표시

- :set 명령은 현재 사용 중인 vi의 환경 설정 상태를 보여줌

```
~  
:set  
— 옵션 —  
commentstring=      incsearch          scrolloff=5          ttymouse=xterm2  
display=truncate   langnoremap        showcmd             viminfo='20,"50  
filetype=text       nolangremap       syntax=text         wildmenu  
helplang=ko         nrformats=bin,hex  ttimeout            t_Sb=^[[4%dm  
history=200          ruler              ttimeoutlen=100    t_Sf=^[[3%dm  
hlsearch           scroll=11          ttyfast  
backspace=indent,eol,start  
comments=fb:-,fb:  
fileencoding=utf-8  
fileencodings=ucs-bom,utf-8,latin1  
guicursor=n-v-c:block,o:hor50,i-ci:hor15,r-cr:hor30,sm:block,a:blinkon0  
계속하려면 엔터 혹은 명령을 입력하십시오
```

03.vi 환경 설정

■ 모든 환경 변수 표시

- vi의 환경 설정에 사용되는 모든 환경 변수와 현재 설정 값을 보여주는 명령은 :set all

```
:set all
— 옵션 —
aleph=224          fileformat=unix      mouse=           swapfile
noarabic           nofileignorecase    mousemodel=extend  swapsync=fsync
arabicshape        filetype=text       mousetime=500     switchbuf=
noallowrevins     fixendofline      nrformats=bin,hex   synmaxcol=3000
noaltkeymap        nofkmap          nonumber         syntax=text
ambiwidth=single   foldclose=        numberwidth=4     tabline=
noautochdir        foldcolumn=0      omnifunc=        tabpagemax=10
noautoindent       foldenable        operatorfunc=   tabstop=8
noautoread         foldexpr=0       nopaste          tagbsearch
noautowrite        foldignore=#     pastetoggle=    tagcase=followic
noautowriteall     foldlevel=0      patchexpr=     taglength=0
background=light    foldlevelstart=-1  patchmode=     tagrelative
                          

nobackup           foldmethod=manual  nopreserveindent  tagstack
backupcopy=auto    foldminlines=1    previewheight=12   term=xterm
backupext=~        foldnestmax=20    nopreviewwindow   notermrbidi
backupskip=/tmp/*
balloondelay=600   formateexpr=      printdevice=     termencoding=
noballoonevalterm formatoptions=tcq  printencoding=   notermguicolors
formatprg=         formatprg=       printfont=courier termkey=
balloonexpr=       fsync           printmbcharset= termsize=
belloff=           nogdefault      printmbfont=    termwinkey=
nobinary           helpheight=20    printoptions=   termwinsize=
— 더 —
```

03.vi 환경 설정

■ 환경 설정 파일과 변수 이용

▪ .exrc 파일에 환경 설정

- 사용자 홈 디렉터리에 .exrc 파일로 저장
- 기본적으로 없는 파일이므로 사용자가 만들어야 함
- 파일에는 set 명령과 옵션만 지정
- 이 파일이 있을 경우 vi를 시작할 때마다 확인하므로 모든 파일에 동일하게 적용 가능

```
set nu  
set list  
set showmode
```

▪ EXINIT 환경 변수에 설정하기

- vi 환경 설정은 다음과 같이 셸의 환경 변수인 EXINIT에도 가능

```
[user1@localhost ch3]$ EXINIT='set nu list'  
[user1@localhost ch3]$ export EXINIT  
[user1@localhost ch3]$
```

CentOS 리눅스

시스템 & 네트워크



Chapter 04. 셸 사용법

목차

00. 개요

04. 배시셀 환경 설정

01. 셀의 기능과 종류

05. 앤리어스와 히스토리

02. 셀 기본 사용법

06. 프롬프트 설정

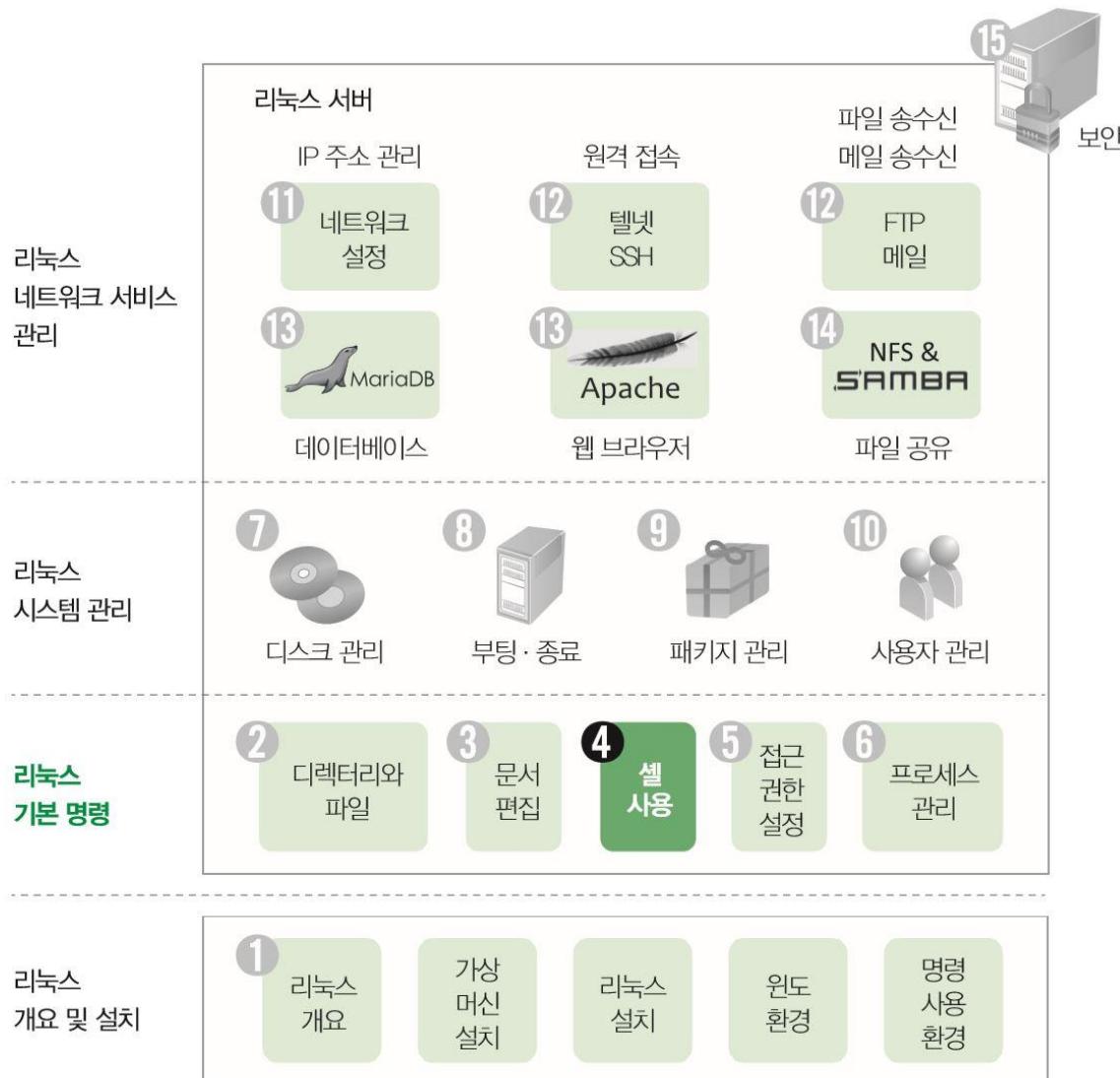
03. 입출력 방향 변경

07. 환경 설정 파일

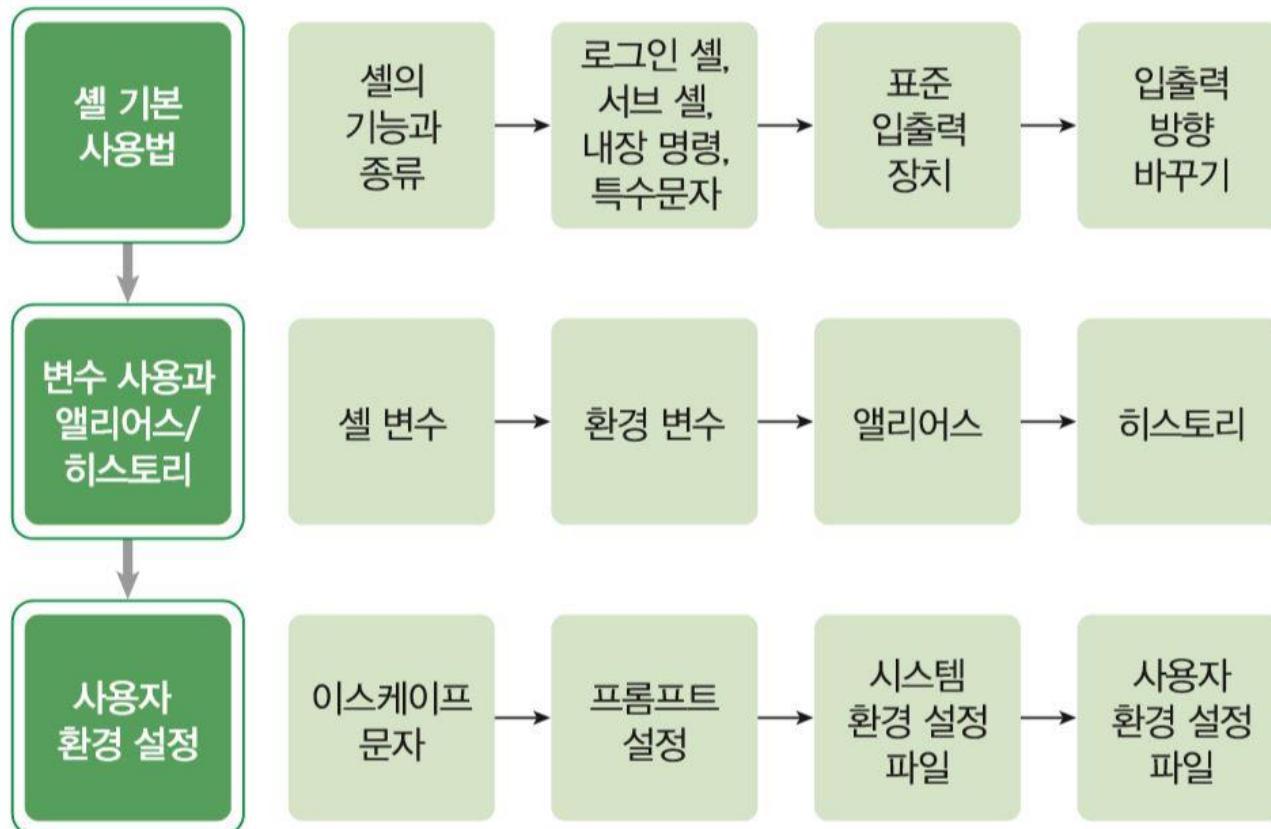
학습목표

- 셀의 기능과 종류를 설명할 수 있다.
- 셀 특수문자의 종류를 이해하고 필요에 따라 적절하게 사용할 수 있다.
- 표준 입출력 장치를 이해하고 입출력 방향을 설정할 수 있다.
- 새로운 앤리어스를 만들거나 필요 없는 앤리어스를 삭제할 수 있다.
- 이스케이프 문자를 이해하고 프롬프트를 원하는 형태로 바꿀 수 있다.
- 시스템과 사용자 환경 설정 파일을 구분하고 사용자 환경을 직접 설정할 수 있다.

00. 개요



00. 개요



01. 셸의 기능과 종류

■ 셸의 기능

- 명령어 해석기 기능
 - 사용자와 커널 사이에서 명령을 해석하여 전달하는 해석기와 번역기 기능
 - 로그인 셸 : 사용자가 로그인하면 셸이 자동으로 실행되어 사용자가 명령을 입력하기를 기다림
`/etc/passwd` 파일에 사용자별로 지정
 - 프롬프트 : 셸이 사용자의 명령을 기다리고 있음을 나타내는 표시
 - 셸은 대기상태에서 입력된 명령이 셸 자체의 내장명령인지 먼저 판단
 - 내장 명령은 특별한 추가 작업 없이 바로 실행되나, 내장 명령이 아니라면 셸은 자식 프로세스를 만들고 자식 프로세스가 명령 파일을 실행하게 함
 - 자식 프로세스가 명령 파일을 실행하는 동안 부모인 셸은 대기 상태가 되며, 자식 프로세스가 실행을 종료하면 셸의 대기 상태가 해제되어 복귀하면서 다시 프롬프트를 출력
- 프로그래밍 기능
 - 셸은 자체 내에 프로그래밍 기능이 있어 반복적으로 수행하는 작업을 하나의 프로그램으로 작성 가능
 - 셸 스크립트 : 셸의 프로그래밍 기능을 이용하여 여러 명령을 사용하여 반복적으로 수행하는 작업을 하나의 프로그램으로 작성한 셸 프로그램
- 사용자 환경 설정 기능
 - 사용자 환경을 설정할 수 있도록 초기화 파일 기능을 제공
 - 초기화 파일에는 명령을 찾아오는 경로를 설정하거나, 파일과 디렉터리를 새로 생성할 때 기본 권한을 설정하거나, 다양한 환경 변수 등을 설정

01. 셸의 기능과 종류

■ 셸의 종류

■ 본셸

- 유닉스 V7에 처음 등장한 최초의 셸
- 개발자의 이름인 스티븐 본의 이름을 따서 본 셸이라고 함
- 초기에 단순하고 처리 속도가 빨라서 많이 사용
- 지금도 시스템 관리 작업을 수행하는 많은 셸 스크립트는 본 셸을 기반으로 함
- 히스토리, 에일리어스, 작업 제어 등 사용자의 편의를 위한 기능을 제공하지 못해 이후에 다른 셸들이 등장
- 현재 본셸은 기존의 오리지널 본셸이 아니라 본셸의 업그레이드 버전이라 할 수 있는 콘셸이나 배시셸로 심볼릭 링크가 됨
- CentOS 8에서 본셸의 경로를 확인해보면 배시셸과 심볼릭 링크로 연결되어 있음을 알 수 있음

```
[user1@localhost ~]$ ls -l /bin/sh  
lrwxrwxrwx. 1 root root 4 5월 10 20:27 /bin/sh → bash
```

01. 셸의 기능과 종류

■ 셸의 종류

■ C셀

- 캘리포니아대학교의 빌 조이가 개발
- 2BSD 유닉스에 포함되어 발표
- 본 셸에는 없던 에일리어스나 히스토리 같은 사용자 편의 기능을 포함
- 셸 스크립트 작성을 위한 구문 형식이 C 언어와 같아 C 셸이라는 이름을 가지게 되었음
- C 셸의 명령 이름은 csh

■ 콘셀

- 1980년대 중반 AT&T 벨연구소의 데이비드 콘이 콘 셸을 개발
- 유닉스 SVR 4에 포함되어 발표
- C 셸과 달리 본 셸과의 호환성을 유지하고 히스토리, 에일리어스 기능 등 C 셸의 특징도 모두 제공하면서 처리 속도도 빠름
- 콘 셸의 명령 이름은 ksh

■ 배시셀

- 본 셸을 기반으로 개발된 셸로서 1988년 브레인 폭스가 개발
- 본 셸과 호환성을 유지하면서 C 셸, 콘 셸의 편리한 기능도 포함
- 배시 셸의 명령 이름은 bash
- 배시 셸의 모든 버전은 GPL 라이선스에 의거하여 자유롭게 사용 가능
- 리눅스의 기본 셸로 제공되고 있어 리눅스 셸로도 많이 알려짐

02. 셸 기본 사용법

■ 셸 지정 및 변경

- 프롬프트 모양 참조
 - 본 셸, 배시 셸, 콘 셸의 기본 프롬프트: \$
 - C 셸의 기본 프롬프트: %
- 사용자 정보 확인: /etc/passwd 파일
 - 가장 앞에 나온 정보가 로그인 ID
 - 사용자 정보의 가장 마지막에 나온 /bin/bash가 기본 셸

```
[user1@localhost ~]$ grep user1 /etc/passwd
user1:x:1000:1000:user1:/home/user1:/bin/bash
```

02. 셸 기본 사용법

■ 셸 지정 및 변경

- 기본 셸 변경하기: chsh

chsh

- **기능** 사용자 로그인 셸을 바꾼다.
- **형식** chsh [옵션] [사용자명]
- **옵션** -s shell: 지정하는 셸(절대 경로)로 로그인 셸을 바꾼다.
-l: /etc/shells 파일에 지정된 셸을 출력한다.
- **사용 예** chsh -l
chsh -s /bin/sh user1
chsh

- 바꿀 수 있는 셸의 종류: /etc/shells 파일에 지정
- /etc/shells 파일의 내용을 보면 본셀(/bin/sh)과 배시셸(/bin/bash)이 사용 가능

```
[user1@localhost ~]$ cat /etc/shells
/bin/sh
/bin/bash
/usr/bin/sh
/usr/bin/bash
```

- chsh -l 명령으로도 확인

```
[user1@localhost ~]$ chsh -l
/bin/sh
/bin/bash
/usr/bin/sh
/usr/bin/bash
```

02. 셸 기본 사용법

■ 셸 지정 및 변경

▪ 기본 셸 변경하기: chsh

- 로그인 셸을 변경하기
- 그냥 셸의 이름만 지정하면 셸의 절대 경로를 입력해야 한다는 메시지가 출력

```
[user1@localhost ~]$ chsh -s sh user1  
Changing shell for user1.  
chsh: shell must be a full path name → 절대 경로로 입력하라는 메시지가 출력된다.
```

- 바꾸려는 셸의 절대 경로를 지정하면 셸이 변경

```
[user1@localhost ~]$ chsh -s /bin/sh user1  
Changing shell for user1.
```

암호:

Shell changed. → 로그인 셸이 변경되었다.

- 로그인 셸이 변경되었는지 /etc/passwd 파일을 확인해보면 셸이 /bin/sh로 바뀜

```
[user1@localhost ~]$ grep user1 /etc/passwd  
user1:x:1000:1000:user1:/home/user1:/bin/sh
```

02. 셸 기본 사용법

■ 셸 지정 및 변경

▪ 로그인 셸과 서브 셸

- 프롬프트에서 다른 셸을 실행할 수 있는데 이를 서브 셸이라 함
- 서브 셸은 또 다른 서브 셸 생성 가능, 또한 여러개의 셸이 사슬처럼 연결될 수 있음
- 서브 셸을 종료하는 명령: ^d([ctrl]+d), exit 등
- 서브 셸이 종료되면 서브 셸을 실행했던 이전 셸 환경으로 복귀
- 로그인 셸에서 로그아웃하면 접속 해제

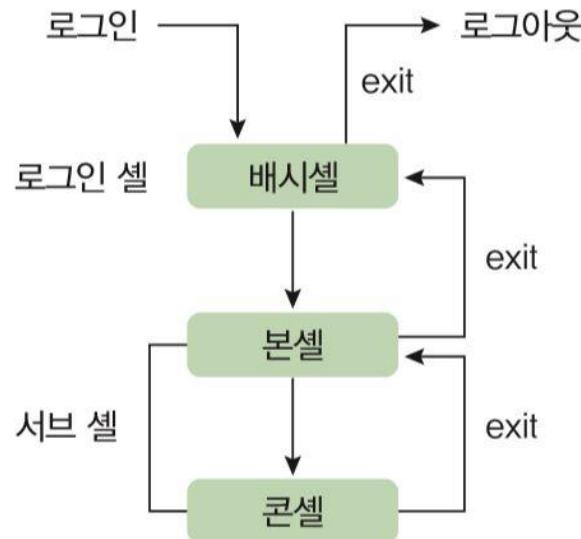


그림 4-1 로그인 셸과 서브 셸

02. 셸 기본 사용법

■ 셸 내장 명령

- 셸은 자체적으로 내장 명령을 가지고 있음
- 셸 내장 명령은 별도의 실행 파일이 없고 셸 안에 포함
- 대표적인 예시로는 cd인데 /usr/bin 디렉터리를 보면 cd 명령이 파일로 존재
- file 명령으로 /usr/bin/cd 파일의 종류를 확인

```
[user1@localhost ~]$ file /usr/bin/cd  
/usr/bin/cd: POSIX shell script, ASCII text executable
```

- /usr/bin/cd는 셸 스크립트이며 텍스트 파일이므로 파일의 내용을 볼 수 있음

```
[user1@localhost ~]$ cat /usr/bin/cd  
#!/bin/sh  
builtin cd "$@"
```

- 그러나 일반적인 실행 파일은 바이너리 파일이므로 cat 명령으로 파일의 내용을 확인할 수 없음

```
[user1@localhost ~]$ file /usr/bin/pwd  
/usr/bin/pwd: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically  
linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=c7  
7330cf70af88d0af68f05b1bde8dc6dd7cf3b5, stripped, too many notes (256)  
[user1@localhost ~]$ file /usr/bin/gcc  
/usr/bin/tmon: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically  
linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=77  
ca9e3bb57f9094c2f6a1047851437e4233e159, stripped
```

02. 셸 기본 사용법

■ 출력 명령

- 출력 명령: 출력 명령은 문자열 출력이나 셸 변수 값 출력, 명령 결과 출력 등에 사용 echo와 printf가 있음
- echo 명령: 모든 셸에서 공통적으로 제공하는 출력 명령

echo

- 기능 화면에 한 줄의 문자열을 출력한다.
- 형식 echo [-n] [문자열]
- 옵션 -n: 마지막에 줄 바꿈을 하지 않는다.
- 사용 예 echo
 - echo text
 - echo -n text

- ex) echo 명령을 사용하여 문자열을 출력

```
[user1@localhost ~]$ echo linux
linux
[user1@localhost ~]$ echo "CentOS Linux"
CentOS Linux
```

02. 셸 기본 사용법

■ 출력 명령

- printf: C 언어의 출력 함수인 printf 처럼 % 지시자와 / 문자를 이용하여 출력 형식을 지정

echo

- 기능 화면에 한 줄의 문자열을 출력한다.
- 형식 echo [-n] [문자열]
- 옵션 -n: 마지막에 줄 바꿈을 하지 않는다.
- 사용 예 echo
 - echo text
 - echo -n text

- /n 없이 출력할 경우 줄 바꿈이 되지 않음, %d와 같은 형식 지정자도 사용할 수 있음

```
[user1@localhost ~]$ echo linux
linux
[user1@localhost ~]$ echo "CentOS Linux"
CentOS Linux
```

02. 셸 기본 사용법

■ 특수문자 사용 방법

- 셸은 사용자가 더욱 편리하게 명령을 입력하고 실행할 수 있도록 다양한 특수문자를 제공
- 주요 특수문자는 *, ?, |, :, [], ~, ' ', " ", ` `` 등
- 명령을 입력하면 셸은 먼저 특수문자가 있는지 확인하고 이를 적절한 형태로 변경한 후 명령을 실행

■ 특수문자 *

- *(별표)는 임의의 문자열을 나타내는 특수문자
- 흔히 사용하는 특수문자 중 하나
- 셸에서 *는 0개 이상의 문자로 대체
- 주로 여러 파일의 이름을 나열할 때 파일 이름을 간단히 표시하는 데 사용
- 명령을 실행할 때 파일명을 적어야 하는 자리에 *를 사용하면 모든 파일을 나타냄

표 4-1 특수문자 *

사용 예	의미
ls *	현재 디렉터리의 모든 파일과 서브 디렉터리를 나열한다. 서브 디렉터리의 내용도 출력한다.
cp * /tmp	현재 디렉터리의 모든 파일을 /tmp 디렉터리 아래로 복사한다.
ls -F t*	t, tmp, temp와 같이 파일명이 t로 시작하는 모든 파일의 이름과 파일 종류를 출력한다. t도 해당 한다는 데 주의한다.
cp *.txt .. /ch3	확장자가 txt인 모든 파일을 상위 디렉터리 아래의 ch3 디렉터리로 복사한다.
ls -h*d	파일명이 h로 시작하고 d로 끝나는 모든 파일의 상세 정보를 출력한다. hd, had, hard, h12345d 등 이 조건에 맞는 모든 파일의 정보를 볼 수 있다.

02. 셸 기본 사용법

■ 특수문자 사용 방법

■ 특수문자 ?와 []

- ?는 길이가 1인 임의의 한 문자를, []는 괄호 안에 포함된 문자 중 하나를 나타냄
- []는 사용할 여러 문자를 나열하거나 범위를 지정할 수 있으며, 다른 특수문자와 혼합하여 사용할 수도 있음

표 4-2 특수문자 ?와 []

사용 예	의미
ls t?.txt	t 다음에 임의의 한 문자가 오고 파일의 확장자가 txt인 모든 파일의 이름을 출력한다. t1.txt, t2.txt, ta.txt 등이 해당된다. 단, t.txt는 제외한다.
ls -t tmp[135].txt	tmp 다음에 1, 3, 5 중 하나가 오고 파일의 확장자가 txt인 모든 파일의 이름을 출력한다. tmp1.txt, tmp3.txt, tmp5.txt 파일이 있으면 해당 파일의 상세 정보를 출력한다. 단, tmp.txt는 제외한다.
ls -t tmp[1-3].txt	[1-3]은 1부터 3까지의 범위를 의미한다. 따라서 ls -t tmp[123].txt와 결과가 같다. 즉 tmp1.txt, tmp2.txt, tmp3.txt 파일이 있으면 해당 파일의 상세 정보를 출력한다.
ls [0-9]*	파일명이 숫자로 시작하는 모든 파일의 목록을 출력한다.
ls [A-Za-z]*[0-9]	파일명이 영문자로 시작하고 숫자로 끝나는 모든 파일의 목록을 출력한다.

02. 셸 기본 사용법

■ 특수문자 사용 방법

■ 특수문자 ~ 와 -

- ~(틸드)와 -(하이픈)은 디렉터리를 나타내는 특수문자
- ~만 사용하면 현재 작업 중인 사용자의 홈 디렉터리를 나타내고, 다른 사용자의 로그인 ID와 함께 사용하면 (~로그인 ID) 해당 사용자의 홈 디렉터리를 나타냄
- 는 cd 명령으로 현재 디렉터리로 이동하기 직전의 디렉터리를 나타냄

표 4-3 특수문자 ~와 -

사용 예	의미
cp *.txt ~/ch3	확장자가 txt인 모든 파일을 현재 작업 중인 사용자의 홈 디렉터리 아래 tmp 디렉터리로 복사한다.
cp ~user2/linux.txt .	user2라는 사용자의 홈 디렉터리 아래에서 linux.txt 파일을 찾아 현재 디렉터리로 복사한다.
cd -	이전 작업 디렉터리로 이동한다.

■ 특수문자 ;과 |

- ;(세미콜론)과 |(파이프)는 명령과 명령을 연결
- ;은 연결된 명령을 왼쪽부터 차례로 실행하고, |는 왼쪽 명령의 실행 결과를 오른쪽 명령의 입력으로 전달

표 4-4 특수문자 ;과 |

사용 예	의미
date; ls; pwd	왼쪽부터 차례대로 명령을 실행한다. 즉 날짜를 출력한 후 현재 디렉터리의 파일 목록을 출력하고, 마지막으로 현재 작업 디렉터리의 절대 경로를 보여준다.
ls -al more	루트 디렉터리에 있는 모든 파일의 상세 정보를 한 화면씩 출력한다. ls -al / 명령의 결과가 more 명령의 입력으로 전달되어 페이지 단위로 출력되는 것이다.

02. 셸 기본 사용법

■ 특수문자 사용 방법

■ 특수문자 ‘ ’와 “ ”

- ‘ ’(작은따옴표)와 “ ”(큰따옴표)는 문자를 감싸서 문자열로 만들어주고, 문자열 안에 사용된 특수 문자의 기능을 없앰
- ‘ ’는 모든 특수 문자를, “ ”는 \$, ` , / 을 제외한 모든 특수 문자를 일반 문자로 간주하여 처리

표 4-5 특수문자 ‘ ’와 “ ”

사용 예	의미
echo '\$SHELL'	\$SHELL 문자열이 화면에 출력된다.
echo "\$SHELL"	셀 환경 변수인 SHELL에 저장된 값인 현재 셸의 종류가 화면에 출력된다. /bin/bash를 예로 들 수 있다.

■ 특수문자 ` `

- ` ` (백쿼터)는 키보드 상단의 숫자 1 키 앞에 있는 문자를 사용
- 모양이 작은따옴표(' ')와 유사하므로 주의

표 4-6 특수문자 ``

사용 예	의미
echo "Today is `date`"	'date'가 명령으로 해석되어 date 명령의 실행 결과로 바뀐다. 결과적으로 다음과 같이 출력된다. Today is 2019. 11. 01. (금) 22:10:35 EDT
ls /usr/bin/uname -m`	uname -m 명령의 실행 결과를 문자열로 바꾸어 파일 이름으로 사용한다.

02. 셸 기본 사용법

■ 특수문자 사용 방법

■ 특수문자 \

- \ (역슬래시)는 특수문자 바로 앞에 사용하여, 해당 특수문자의 효과를 없애고 일반 문자처럼 처리

표 4-7 특수문자 \

사용 예	의미
ls -t *	t*라는 이름을 가진 파일의 상세 정보를 출력한다. \ 없이 t*를 사용하면 t로 시작하는 모든 파일의 상세 정보를 출력한다.
echo \\$SHELL	\\$SHELL을 화면에 출력한다. echo '\$SHELL'과 결과가 같다.

■ 특수문자 >, <, >>

- 입출력의 방향을 바꾸는 특수문자

표 4-8 특수문자 >, <, >>

사용 예	의미
ls -> res	ls - 명령의 실행 결과를 화면이 아닌 res 파일에 저장한다.

03. 입출력 방향 변경

■ 표준 입출력 장치

- 표준 입력 장치: 셀이 작업을 수행하는 데 필요한 정보를 받아들이는 장치 -> 키보드
- 표준 출력 장치: 실행 결과를 내보내는 장치 -> 모니터
- 표준 오류 장치: 오류 메시지를 내보내는 장치 -> 모니터

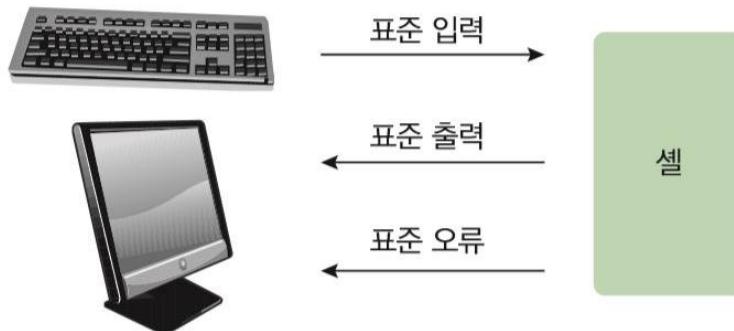


그림 4-2 표준 입출력 장치

- 파일 디스크립터: 셀은 작업 중 필요한 파일에 일련번호를 붙여서 관리
표준 입출력 장치 또한 파일로 관리되기 때문에 파일 디스크립터가 부여되어 있음

표 4-8 특수문자 <, >, <>

사용 예	의미
ls -> res	ls 명령의 실행 결과를 화면이 아닌 res 파일에 저장한다.

- 리다이렉션: 표준 입출력 장치를 특수문자를 사용하여 파일로 바꾸는 것

03. 입출력 방향 변경

■ 출력 리다이렉션

- 출력 리다이렉션: 출력의 방향을 바꾸기
- > : 기존 파일의 내용을 삭제하고 새로 결과를 저장
- >> : 기존 파일의 내용 뒤에 결과를 추가

■ 파일 덮어쓰기: >

- >: 표준 출력 파일을 바꾸는 특수문자

>

• **기능** 파일 리다이렉션(덮어쓰기)을 한다.

• **형식** 명령 1> 파일명
명령 > 파일명

- 1: 파일 디스크립터 1번(표준 출력, 화면)
- 셀은 >를 사용한 리다이렉션에서 지정한 이름의 파일이 없으면 파일을 생성해서 명령의 수행 결과를 저장
- 파일이 있으면 이전의 내용이 없어지고 명령의 수행 결과로 대체

03. 입출력 방향 변경

■ 출력 리다이렉션

▪ 파일 덮어쓰기: >

- linux_ex 디렉터리 아래에 ch4 디렉터리를 생성하고 이 디렉터리에서 작업

```
[user1@localhost ~]$ mkdir linux_ex/ch4
[user1@localhost ~]$ cd linux_ex/ch4
[user1@localhost ch4]$ ls out1          → out1이 있는지 확인한다.
ls: cannot access 'out1': 그런 파일이나 디렉터리가 없습니다
[user1@localhost ch4]$ ls -al          → 명령의 결과가 화면(표준 출력)으로 출력된다.
합계 0
drwxrwxr-x. 2 user1 user1 6 11월 2 01:14 .
drwxrwxr-x. 4 user1 user1 28 11월 2 01:14 ..
[user1@localhost ch4]$ ls -al > out1      → 명령의 결과를 out1 파일에 저장한다.
[user1@localhost ch4]$ cat out1          → 파일 내용을 확인한다.
합계 0
drwxrwxr-x. 2 user1 user1 18 11월 2 01:19 .
drwxrwxr-x. 4 user1 user1 28 11월 2 01:14 ..
-rw-rw-r--. 1 user1 user1 0 11월 2 01:19 out1
[user1@localhost ch4]$ date > out1      → 명령의 결과를 out1 파일에 저장한다.
[user1@localhost ch4]$ cat out1          → ls 명령의 실행 결과가 없어졌다.
2019. 11. 02. (토) 14:21:36 KST
```

03. 입출력 방향 변경

■ 출력 리다이렉션

▪ 파일 덮어쓰기: >

- set 명령을 사용하면 예상치 않게 파일 내용을 덮어쓰는 일을 예방

```
[user1@localhost ch4]$ set -o noclobber  
[user1@localhost ch4]$ ls > out1  
-bash: out1: cannot overwrite existing file
```

- 설정을 해제하려면 다음과 같이 + 옵션을 사용

```
[user1@localhost ch4]$ set +o noclobber  
[user1@localhost ch4]$ ls > out1
```

- 일반적으로 cat 명령은 파일의 내용을 화면에 출력할 때 사용하지만 표준 입력으로부터 입력을 받아 표준 출력으로 보낼 수도 있음
- cat 명령의 결과를 파일로 리다이렉션 하면 키보드의 입력을 새로운 파일을 생성할 수 있음

```
[user1@localhost ch4]$ cat > out1  
CentOS Linux  
I love Linux.  
^D  
[user1@localhost ch4]$ cat out1  
CentOS Linux  
I love Linux.
```

→ 표준 입력을 받아 out1에 저장한다.
→ 내용을 입력한다.
→ 입력을 종료한다.
→ 파일 내용을 확인한다.

03. 입출력 방향 변경

■ 출력 리다이렉션

- 파일에 내용 추가하기: >>

»

- 기능 파일에 내용을 추가한다.
- 형식 명령 >> 파일명

- 지정한 파일명의 파일이 없으면 파일을 생성하고, 해당 파일이 있으면 그 파일의 끝에 명령의 실행 결과를 추가

```
[user1@localhost ch4]$ cat out1
```

→ 기존 파일 내용을 확인한다.

```
CentOS Linux
```

```
I love Linux.
```

```
[user1@localhost ch4]$ date >> out1
```

→ 리다이렉션한다(내용 추가).

```
[user1@localhost ch4]$ cat out1
```

→ 파일 내용을 확인한다.

```
CentOS Linux
```

```
I love Linux.
```

```
2019. 11. 02. (토) 14:31:14 KST
```

→ 추가된 내용이다.

03. 입출력 방향 변경

■ 오류 리다이렉션

- 표준 오류도 기본적으로 화면으로 출력되며 표준 출력처럼 리다이렉션 가능
- ex) 표준 출력과 표준 오류

```
[user1@localhost ch4]$ ls  
out1  
[user1@localhost ch4]$ ls /abc  
ls: cannot access '/abc': 그런 파일이나 디렉터리가 없습니다
```

→ 정상 실행(표준 출력)
→ 오류 메시지(표준 오류)

- 두 ls명령이 다르다는 것을 확인하기 위해 위의 두 명령을 실행하면서 표준 출력을 리다이렉션

```
[user1@localhost ch4]$ ls > ls.out  
[user1@localhost ch4]$ ls /abc > ls.err  
ls: cannot access '/abc': 그런 파일이나 디렉터리가 없습니다  
[user1@localhost ch4]$ cat ls.err  
[user1@localhost ch4]$ cat ls.out  
  
ls.out  
out1
```

→ 표준 출력 리다이렉션
→ 표준 출력 리다이렉션
→ 오류 메시지가 화면에 출력된다.
→ 오류 메시지가 저장되지 않았다.
→ 표준 출력 내용이 출력된다.

- 정상적으로 리다이렉션되어 결과가 ls.out 파일에 저장됨
- 오류 메시지는 ls.err 파일에 저장되지 않고 화면에 그대로 출력
- 오류 메시지는 표준출력이 아니기 때문에 리다이렉션이 안됨

03. 입출력 방향 변경

■ 오류 리다이렉션

2>

- **기능** 표준 오류 메시지를 파일에 저장한다.
- **형식** 명령 2> 파일명

- 2>: stderr 파일로 나갈 내용을 지정한 파일에 저장하라고 셸에 알림
- 오류 리다이렉션에서는 파일 디스크립터 번호를 생략할 수 없음

```
[user1@localhost ch4]$ ls /abc 2> ls.err          → 표준 오류를 리다이렉션한다.  
[user1@localhost ch4]$ cat ls.err  
ls: cannot access '/abc': 그런 파일이나 디렉터리가 없습니다 → 파일에 저장된 메시지이다.
```

- 표준 출력과 표준 오류를 한 번에 리다이렉션하기
 - 명령의 실행 결과와 오류 메시지를 한 번에 리다이렉션 하여 각각 파일에 저장하려면 >와 2>를 함께 사용

```
[user1@localhost ch4]$ ls . /abc > ls.out 2> ls.err
```

- 오류 메시지 버리기
 - 오류 메시지를 무시하고 싶을 때는 리눅스의 특수 파일인 /dev/null 을 사용

```
[user1@localhost ch4]$ ls /abc 2> /dev/null
```

03. 입출력 방향 변경

■ 오류 리다이렉션

- 표준 출력과 표준 오류를 한 파일로 리다이렉션하기
 - 명령의 정상 실행 결과를 파일로 리다이렉션 (>)
 - 그 명령 전체의 오류 메시지를 1번 파일(표준 출력 파일, &1이라고 표현함)로 리다이렉션 (2>)

```
[user1@localhost ch4]$ ls . /abc > ls.out 2>&1
[user1@localhost ch4]$ cat ls.out
ls: cannot access '/abc': 그런 파일이나 디렉터리가 없습니다
.:
ls.err
ls.out
out1
```

- 오류 메시지를 저장한다
- 현재 디렉터리 내용이다

03. 입출력 방향 변경

■ 입력 리다이렉션

- <: 표준 입력 장치 파일을 리다이렉션하는 기능을 제공하는 특수문자

<

- **기능** 표준 입력을 바꾼다.
- **형식** 명령 0< 파일명
명령 < 파일명

- cat 명령: 파일의 내용이나 키보드로 입력을 받아 화면에 출력하는 명령
- cat 명령이 파일을 읽어서 출력하는 기능은 < 를 생략한 것

```
[user1@localhost ch4]$ cat out1  
CentOS Linux  
I love Linux.
```

→ 파일 내용을 출력한다(< 생략).

```
2019. 11. 02. (토) 14:31:14 KST  
[user1@localhost ch4]$ cat < out1  
CentOS Linux  
I love Linux.
```

→ 표준 입력을 리다이렉션한다(< 사용).

```
2019. 11. 02. (토) 14:31:14 KST  
[user1@localhost ch4]$ cat 0< out1  
CentOS Linux  
I love Linux.  
2019. 11. 02. (토) 14:31:14 KST
```

→ 표준 입력을 리다이렉션한다(0< 사용).

03. 입출력 방향 변경

■ [따라해보기] 특수문자 사용하고 입출력 방향 바꾸기

- ① 현재 디렉터리에서 l(소문자 L)로 시작하는 모든 파일의 상세 정보를 확인

```
[user1@localhost ch4]$ ls -l l*
```

- ② 임시로 사용할 temp 디렉터리를 제작

```
[user1@localhost ch4]$ mkdir temp
```

- ③ l로 시작하는 모든 파일을 temp 디렉터리로 이동

```
[user1@localhost ch4]$ mv l* temp
```

- ④ uname 명령은 운영체제의 이름을 출력함, 이 명령을 사용하여 'This is Linux System.'이라는 문장을 출력

- 백워터 기호를 작은따옴표와 헷갈리지 않도록 유의

```
[user1@localhost ch4]$ echo "This is `uname` System."
```

```
This is Linux System.
```

03. 입출력 방향 변경

■ [따라해보기] 특수문자 사용하고 입출력 방향 바꾸기

- ⑤ 출력 방향 바꾸기를 통해 ④번 명령의 실행 결과를 u.out 파일에 저장

```
[user1@localhost ch4]$ echo "This is `uname` System."  
This is Linux System.
```

- ⑥ date 명령의 실행 결과를 u.out 파일에 저장. u.out 파일에는 어떤 내용이 남아 있을지 생각해봄

```
[user1@localhost ch4]$ echo "This is `uname` System." > u.out
```

04. 배시셀 환경 설정

■ 셀 변수와 환경 변수

- 셀의 환경을 설정하기 위한 값을 저장할 수 있도록 셀 변수와 환경 변수를 제공
- 셀 변수: 현재 셀에서만 사용이 가능하고 서브 셀로는 전달되지 않음 (지역변수)
- 환경 변수: 로그인 셀뿐만 아니라 서브 셀로도 전달 (전역변수)
- 환경 변수의 이름에는 대문자를 사용

표 4-10 주요 셀 환경 변수

환경 변수	의미	환경 변수	의미
HISTSIZE	히스토리 저장 크기	PATH	명령을 탐색할 경로
HOME	사용자 홈 디렉터리의 절대 경로	PWD	작업 디렉터리의 절대 경로
LANG	사용하는 언어	SHELL	로그인 셀
LOGNAME	사용자 계정 이름		

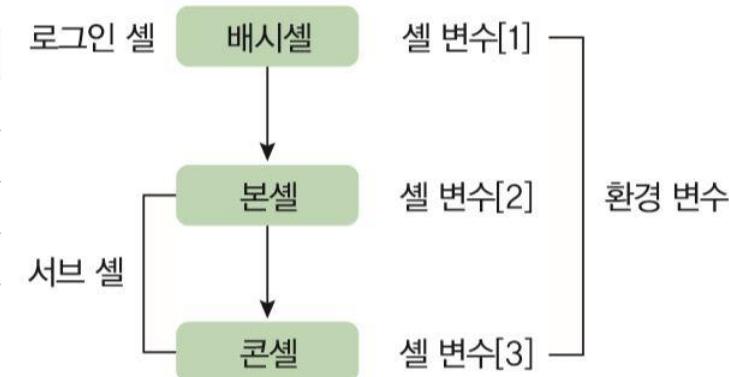


그림 4-3 셀 변수와 환경 변수

04. 배시셀 환경 설정

■ 셸 변수와 환경 변수 출력

- 전체 변수 출력하기: set, env
 - set: 셸 변수와 환경 변수를 모두 출력
 - env: 환경 변수만 출력
- set 명령의 실행 결과

```
[user1@localhost ch4]$ set
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extquote
:force_fignore:histappend:interactive_comments:login_shell:progcomp:promptvars:s
ourcepath
BASHRCSCORED=Y
BASH_ALIASES=()
BASH_ARGC=()
(중략)
quote_readline ()
{
    local quoted;
    _quote_readline_by_ref "$1" ret;
    printf %s "$ret"
}
```

04. 배시셸 환경 설정

■ 셸 변수와 환경 변수 출력

- 전체 변수 출력하기: set, env
 - env 명령의 실행 결과

```
[user1@localhost ch4]$ env  
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd  
=40;33;01:or=40;31;01:mi=01;05;37;41:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;4  
(생략)  
PATH=/home/user1/.local/bin:/home/user1/bin:/usr/local/bin:/usr/bin:/usr/local/s  
bin:/usr/sbin  
HISTSIZE=1000  
LESSOPEN=||/usr/bin/lesspipe.sh %s  
_=usr/bin/env  
OLDPWD=/home/user1
```

- 특정 변수 출력하기: echo

- 전체 변수를 한 번에 출력하지 않고 개별 변수의 값을 출력하기 위해서 'echo' 명령을 사용

```
[user1@localhost ch4]$ echo $SHELL  
/bin/bash
```

04. 배시셸 환경 설정

■ 셸 변수와 환경 변수 설정

셸 변수 정의

- 형식 변수명=문자열
- 사용 예 SOME=test

- 변수명과 문자열 사이에 공백이 있으면 안 됨
- ex) 셸 변수를 설정

```
[user1@localhost ch4]$ SOME=test  
[user1@localhost ch4]$ echo $SOME  
test
```

- set 명령과 env 명령으로 각각 확인해보면, env 명령으로는 SOME 변수를 확인할 수 없음
 - SOME이 아직 환경 변수로 설정되지 않았기 때문

```
[user1@localhost ch4]$ set | grep SOME  
SOME=test  
[user1@localhost ch4]$ env | grep SOME
```

04. 배시셀 환경 설정

■ 셸 변수와 환경 변수 설정

▪ 환경 변수 설정하기: export

- 환경 변수를 설정하려면 먼저 셸 변수를 정의하고, export 명령을 사용하여 이를 환경 변수로 변경함
- export: 셸 내장 명령

환경 변수 정의

- 기능** 지정한 셸 변수를 환경 변수로 바꾼다.
- 형식** `export [옵션] [셸 변수]`
- 옵션** `-n`: 환경 변수를 셸 변수로 변경한다.
- 사용 예**
`export SOME`
`export SOME=test`

- 셸 변수인 `SOME`을 다음과 같이 환경 변수로 변경할 수 있음

```
[user1@localhost ch4]$ export SOME  
[user1@localhost ch4]$ env | grep SOME  
SOME=test
```

- export 명령과 함께 변수를 정의하면 한 번에 환경 변수로 설정할 수 있음

```
[user1@localhost ch4]$ export SOME1=test1  
[user1@localhost ch4]$ echo $SOME1  
test1  
[user1@localhost ch4]$ env | grep SOME  
SOME=test  
SOME1=test1
```

04. 배시셸 환경 설정

■ 셸 변수와 환경 변수 설정

- 환경 변수를 다시 셸 변수로 변경하기: `export -n`
 - `export -n` 명령: `export` 명령으로 환경 변수로 전환한 셸 변수를 다시 셸 변수로 변경할 때 사용

```
[user1@localhost ch4]$ export -n SOME1
[user1@localhost ch4]$ env | grep SOME
SOME=test
```

04. 배시셸 환경 설정

■ 변수 해제

- `unset` 명령: 지정한 변수를 해제할 때 사용

unset

- **기능** 지정한 변수를 해제한다.
- **형식** `unset [변수]`
- **사용 예** `unset SOME`

- ex) `SOME`, `SOME1` 변수를 해제

```
[user1@localhost ch4]$ unset SOME
[user1@localhost ch4]$ unset SOME1
[user1@localhost ch4]$ echo $SOME

[user1@localhost ch4]$ echo $SOME1

[user1@localhost ch4]$
```

- 변수 설정이 해제되었으므로 아무값도 출력되지 않음

05. 앤리어스와 히스토리

■ 앤리어스

- 우리말로 '별명'을 의미
- 기존의 명령을 대신하여 다른 이름(별명)을 붙일 수 있도록 하는 기능
- 긴 명령 대신 짧은 명령을 만들어 사용 가능
- 여러 명령을 연결하여 하나의 명령으로 만들 수도 있음
- 자주 사용하는 옵션을 포함하여 새로운 이름을 붙여서 사용 가능
- 앤리어스를 만들 때는 alias 명령을 사용하여 만듬

alias

- **기능** 앤리어스를 생성한다.
- **형식** alias 이름='명령'
- **사용 예** alias: 현재 설정된 별칭 목록을 출력한다.
alias 이름='명령': 명령을 수정해 사용하는 경우다.
alias 이름='명령;명령2;...': 여러 명령을 하나의 이름으로 사용하는 경우다.

05. 앤리어스와 히스토리

■ 앤리어스

- 아무것도 지정하지 않고 alias 명령을 실행시키면 현재 설정되어 있는 앤리어스가 출력

```
[user1@localhost ch4]$ alias  
alias egrep='egrep --color=auto'  
alias fgrep='fgrep --color=auto'  
alias grep='grep --color=auto'  
alias l.='ls -d .* --color=auto'  
alias ll='ls -l --color=auto'  
  
alias ls='ls --color=auto'  
alias vi='vim'  
alias which='(alias; declare -f) | /usr/bin/which --tty-only --read-alias --read-  
functions --show-tilde --show-dot'  
alias xzgrep='xzgrep --color=auto'  
alias xzfgrep='xzfgrep --color=auto'  
alias xzgrep='xzgrep --color=auto'  
alias zgrep='zgrep --color=auto'  
alias zfgrep='zfgrep --color=auto'  
alias zgrep='zgrep --color=auto'
```

05. 앤리어스와 히스토리

■ 앤리어스

- 앤리어스 중에서 I.과 II이 ls 명령에 옵션을 지정하여 설정한 뒤 실행

```
[user1@localhost ch4]$ ls
ls.err  ls.out  out1  temp  u.dat  u.out  un.out
[user1@localhost ch4]$ l.
.
.
[user1@localhost ch4]$ ll
합계 24
-rw-rw-r--. 1 user1 user1 75 11월  2 15:26 ls.err
-rw-rw-r--. 1 user1 user1 97 11월  2 15:26 ls.out
-rw-rw-r--. 1 user1 user1 60 11월  2 14:31 out1
drwxrwxr-x. 2 user1 user1 34 11월  2 14:46 temp/
drwxrwxr-x. 2 user1 user1  6 11월  2 15:30 tmp/
-rw-rw-r--. 1 user1 user1 124 11월  2 15:33 u.dat
-rw-rw-r--. 1 user1 user1 33 11월  2 14:47 u.out
```

- I.의 경우 ls -d .*를 앤리어스 한 것이며 .(마침표)로 시작하는 경로를 보여줌
- II의 경우 ls -l 명령을 실행하도록 앤리어스 한 것
- 자주 사용하는 명령과 옵션을 묶어서 앤리어스를 설정하면 편리

05. 앤리어스와 히스토리

■ 앤리어스

▪ 앤리어스 설정 및 삭제하기

- 앤리어스는 alias 명령으로 설정하는데, 셸 변수를 지정하듯이 '앤리어스 이름=명령' 형식을 사용
 - =를 중심으로 좌우에 공백이 있으면 안됨, 공백이 있을 경우에는 작은따옴표로 묶어야 함
- ex) ls 명령에 -F 옵션을 지정하여 다시 ls 명령으로 지정

```
[user1@localhost ch4]$ mkdir tmp  
[user1@localhost ch4]$ ls  
ls.err  ls.out  out1  temp  tmp  u.dat  u.out  un.out  
[user1@localhost ch4]$ alias ls='ls -F'          → 공백이 있으면 작은따옴표를 사용한다.  
[user1@localhost ch4]$ ls                         → 앤리어스의 ls를 실행한다.  
ls.err  ls.out  out1  temp/  tmp/  u.dat  u.out  un.out → ls -F의 결과를 출력한다.
```

- ex) 앤리어스를 사용하여 rm 명령에 안전장치를 사용

```
[user1@localhost ch4]$ alias rm='rm -i'  
[user1@localhost ch4]$ rm out1  
rm: remove 일반 파일 'out1'? n
```

- 앞서 설정한 앤리어스는 unalias 명령으로 삭제

unalias

- **기능** 앤리어스를 삭제한다.
- **형식** unalias 앤리어스

```
[user1@localhost tmp]$ unalias ls  
[user1@localhost tmp]$ unalias rm
```

05. 앤리어스와 히스토리

■ 앤리어스

▪ 앤리어스에 인자 전달하기

- 디렉터리를 이동한 후에 올바른 위치인지 확인하기 위해 cd 명령과 pwd 명령을 묶어서 앤리어스를 제작

```
[user1@localhost ch4]$ alias cd='cd;pwd' → ;으로 명령을 연결하여 앤리어스를 만든다.  
[user1@localhost ch4]$ cd  
/home/user1  
[user1@localhost ~]$ cd linux_ex/ch4  
/home/user1
```

- 배시셀에서는 앤리어스로 인자를 전달할 수 없기 때문에 원하는 디렉터리로 이동할 수 없음
- 배시셀에서 인자를 전달하려면 프로그래밍 기능에서 함수를 사용해야함
- 먼저 unalias 명령으로 기존 앤리어스를 삭제 후 함수를 입력

```
[user1@localhost ch4]$ unalias cd → 앤리어스를 해제한다.  
[user1@localhost ch4]$ function cdpwd { → 함수 입력을 시작한다.  
  > cd $1;pwd → 프롬프트가 >로 바뀐다. 내용을 입력한다.  
  > } → 함수 입력을 종료한다.  
[user1@localhost ch4]$ cdpwd /tmp  
/tmp
```

05. 앤리어스와 히스토리

■ 히스토리

- 이전 명령 보기: history
 - 배시셀 내장 명령인 history 명령으로 사용자가 이전에 입력한 명령을 다시 불러 사용
 - ex) history 실행

```
[user1@localhost tmp]$ history  
(생략)  
114 unalias cd  
115 function cdpwd { cd $1;pwd; }  
116 cdpwd /tmp  
117 unalias ls  
118 unalias rm  
119 history
```

05. 앤리어스와 히스토리

■ 히스토리

▪ 명령 재실행하기: !

- 특수문자 !를 사용하여 실행했던 명령을 지정

표 4-11 !를 사용한 명령 재실행 방법

사용법	기능
!!	바로 직전에 실행한 명령을 재실행한다.
! <u>번호</u>	히스토리에서 해당 번호의 명령을 재실행한다.
! <u>문자열</u>	히스토리에서 해당 문자열로 시작하는 마지막 명령을 재실행한다.

- ex) !!를 사용하여 바로 직전에 실행한 명령을 다시 실행

```
[user1@localhost tmp]$ cd ~/linux_ex/ch4
[user1@localhost ch4]$ ls
ls.err  ls.out  out1  temp  tmp  u.dat  u.out  un.out
[user1@localhost ch4]$ !!
→ 바로 직전 명령을 재실행한다.
ls
ls.err  ls.out  out1  temp  tmp  u.dat  u.out  un.out
```

05. 앤리어스와 히스토리

■ 히스토리

- 명령 재실행하기: !
 - history 명령으로 확인하고 재실행할 명령의 히스토리 번호나 명령의 앞 글자를 이용

```
[user1@localhost ch4]$ history  
(생략)  
121 cd ~/linux_ex/ch4  
122 ls  
123 history  
[user1@localhost ch4]$ !122          ➔ 히스토리 번호로 재실행한다.  
ls  
ls.err ls.out out1 temp tmp u.dat u.out un.out  
[user1@localhost ch4]$ !1          ➔ 명령의 앞 글자로 재실행한다.  
ls  
ls.err ls.out out1 temp tmp u.dat u.out un.out
```

05. 앤리어스와 히스토리

■ 히스토리

- 명령 편집 및 재실행하기
 - 화살표 키를 사용하여 오류가 난 명령을 다시 프롬프트로 불러내서 수정한 뒤 재실행 가능

- ➊ 편집과 재실행 예1 : 명령에 오타를 입력

```
[user1@localhost ch4]$ man hisdory  
No manual entry for hisdory
```

- ➋ 프롬프트에서 ↑ 키를 누르면 방금 실행한 명령이 다시 나타남

```
[user1@localhost ch4]$ man hisdory
```

- ➌ 좌우 화살표로 커서를 이동하여 [back space]로 삭제한 후 다시 글자를 입력하고 [enter]키를 눌러서 실행

```
[user1@localhost ch4]$ man history
```

05. 앤리어스와 히스토리

■ 히스토리

- 히스토리 저장하기
 - 로그아웃 할 때 홈 디렉터리 아래의 숨김 파일인 .bash_history에 히스토리 저장

```
[user1@localhost ch4]$ more ~/.bash_history  
pwd  
ls  
ls -a  
ls /tmp  
(생략)
```

05. 앤리어스와 히스토리

■ [따라해보기] 환경 변수, 앤리어스, 히스토리

- ① 셸 변수 TESTA를 설정하고 출력

```
[user1@localhost ch4]$ TESTA=testa  
[user1@localhost ch4]$ echo $TESTA  
testa
```

- ② 본셀(sh)을 실행하여 서브 셸로 이동

```
[user1@localhost ch4]$ sh  
sh-4.4$
```

- ③ 셸 변수 TESTA가 출력되는지 확인, 출력이 되지 않는 이유는 무엇일까?

```
sh-4.4$ echo $TESTA  
  
sh-4.4$
```

05. 앤리어스와 히스토리

■ [따라해보기] 환경 변수, 앤리어스, 히스토리

- ④ 서브 셸에서 로그인 셸로 복귀

```
sh-4.4$ exit  
exit  
[user1@localhost ch4]$
```

- ⑤ pwd 명령과 ls 명령을 묶어서 앤리어스 pls를 만듬

```
[user1@localhost ch4]$ alias pls='pwd;ls'  
[user1@localhost ch4]$ pls  
/home/user1/linux_ex/ch4  
ls.err ls.out out1 temp tmp u.dat u.out un.out
```

- ⑥ ↑로 이전 명령을 불러서 clear 명령에 대한 앤리어스 c를 만듬

[user1@localhost ch4]\$ ↑	→ ↑를 누른다.
[user1@localhost ch4]\$ pls	→ 바로 이전 명령인 pls를 출력한다.
[user1@localhost ch4]\$ ↑	→ 다시 ↑를 누른다.
[user1@localhost ch4]\$ alias pls='pwd;ls'	→ alias 명령을 출력한다.
[user1@localhost ch4]\$ alias c=clear	→ 명령을 수정한다.

06. 프롬프트 설정

■ 이스케이프 문자와 프롬프트 설정

```
[user1@localhost ch4]$ echo $PS1  
[\u@\h \W]\$
```

→ PS1의 현재 설정값

- PS1의 설정값을 보면 \u, \h, \W 등 의미를 알 수 없는 문자로 구성
- 이 문자들을 이스케이프 문자라고 하며 두 글자가 아닌 한 글자로 처리
- [\u@\h \W]\\$를 해석해보면 \u는 사용자 이름, \h는 호스트 이름, \W는 현재 작업 디렉터리명

표 4-12 이스케이프 문자

이스케이프 문자	기능
\a	ASCII 종소리 문자(07)
\d	'요일 월 일' 형식으로 날짜를 표시한다(Wed May 1).
\e	ASCII의 이스케이프 문자로 터미널에 고급 옵션을 전달한다.
\h	첫 번째 .(마침표)까지의 호스트 이름(server.co.kr에서 server)
\H	전체 호스트 이름
\n	줄 바꾸기
\s	셀 이름
\t	24시간 형식으로 현재 시간을 표시한다(HH:MM:SS 형식).
\T	12시간 형식으로 현재 시간을 표시한다(HH:MM:SS 형식).
\@	12시간 형식으로 현재 시간을 표시한다(오전/오후 형식).
\u	사용자 이름
\v	배시셸의 버전
\w	현재 작업 디렉터리(절대 경로)
\W	현재 작업 디렉터리의 절대 경로에서 마지막 디렉터리명
\!	현재 명령의 히스토리 번호
\[출력하지 않을 문자열의 시작 부분을 표시한다.
\]	출력하지 않을 문자열의 끝 부분을 표시한다.

06. 프롬프트 설정

■ 이스케이프 문자와 프롬프트 설정

- PS1의 값을 바꾸기 전에 먼저 현재 PS1의 값을 임시 변수에 저장

```
[user1@localhost ch4]$ PROMPT=$PS1
```

- ① 먼저 간단한 문자열로 변경

```
[user1@localhost ch4]$ PS1='LINUX ] '  
LINUX ]
```

- ② 환경 변수를 사용

```
LINUX ] PS1='[$PWD] '  
[/home/user1/linux_ex/ch4] cd ..  
[/home/user1/linux_ex]
```

- ③ 특수문자 ``를 이용하여 명령의 실행 결과를 사용

```
[/home/user1/linux_ex] PS1='`uname -n` $ '  
localhost.localdomain $
```

- ④ 이스케이프 문자 \\$, \\$T, \\$!를 사용

```
localhost.localdomain $ PS1='[\u \T] \$!$ '  
[user1 04:16:59] 138$
```

06. 프롬프트 설정

■ 컬러 프롬프트 설정

▪ 컬러 프롬프트 정의하기

컬러 프롬프트

• 형식 PS1= '\[\e[x;y;n\] 프롬프트 \[\e[x;y;0\]'

- 첫 번째 문자 \ : 화면에 출력하지 않을 문자의 시작을 알림
- 두 번째 문자 \ e: 터미널 제어를 알림
- x;y: 컬러 번호 (x: 글자색, y: 배경색)
- n: 밑줄이나 역상 표시 등 특수 기능

표 4-13 프롬프트 컬러 번호

컬러	글자색 번호	배경색 번호
검정색	30	40
빨강색	31	41
녹색	32	42
갈색	33	43
파란색	34	44
보라색	35	45
청록색	36	46
흰색	37	47

표 4-14 프롬프트 특수 기능 번호

번호	기능
0	기본 색
1	굵게
4	흑백에서 밑줄
5	반짝임
7	역상
10	기본 폰트
38	밑줄 사용 가능
39	밑줄 사용 불가능

06. 프롬프트 설정

■ 컬러 프롬프트 설정

▪ 컬러 프롬프트 설정하기

- 파란색으로 설정하기

```
[user1 04:16:59] 138$ PS1='`e[34mLinux $`e[0;0m '  
Linux $ → 파란색
```

- 파란색 볼드로 설정하기

```
Linux $ PS1='`e[34;1mLinux $`e[0;0m '  
Linux $ → 파란색, 볼드
```

- 밑줄 친 빨간색으로 설정하기

```
Linux $ PS1='`e[31;4mLinux $`e[0;0m '  
Linux $ → 빨간색, 밑줄
```

- 배경은 갈색, 글자는 보라색, 프롬프트는 '사용자명@호스트명 '\$'로 설정하기

```
Linux $ PS1='`e[35;43m\u@\h $`e[0;0m "\'  
user1@localhost $ → 갈색 배경, 보라색 글자
```

06. 프롬프트 설정

■ 컬러 프롬프트 설정

- 출력하지 않을 문자열 설정 추가하기

- 컬러로 바꾼 프롬프트 상태에서 문자를 계속 입력

```
aaar1@localhost $ aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa → 중간에 이동
```

- 터미널의 화면 끝에 가기 전에 커서가 앞으로 돌아오고 같은 행을 겹쳐서 기록
 - \[와 \]를 사용하여 출력되지 않을 문자를 표시

```
user1@localhost $ PS1='[\e[35;43m]\u001b[0;0m]'  
user1@localhost $
```

- 정상적으로 줄 바꿈이 되는지 다시 문자를 입력해서 확인

```
user1@localhost $ ffffffffffffffffffffff fff  
ffffffffff ffffff ffffff ffffff ffffff^C  
user1@localhost $
```

- 프롬프트를 처음에 저장했던 값으로 되돌림

```
user1@localhost $ PS1=$PROMPT  
[user1@localhost ch4]$
```

07.환경 설정 파일

■ 환경 설정 파일

- 환경 설정 파일: 사용자가 로그인할 때마다 자동으로 실행되는 명령을 저장한 것
- 시스템 환경 설정 파일과 사용자 환경 설정 파일이 있음
- 셀마다 다른 이름의 파일을 사용

■ 시스템 환경 설정 파일

- 시스템을 사용하는 전체 사용자의 공통 환경을 설정하는 파일
- 일반 사용자가 수정할 수 없으며 시스템 관리자가 관리

표 4-15 배시셸의 시스템 환경 설정 파일

파일	기능
/etc/profile	<ul style="list-style-type: none">• 시스템 공통으로 적용되는 환경 변수를 설정한다.<ul style="list-style-type: none">– PATH: 기본 명령 경로를 설정한다.– USER, LOGNAME: 사용자 UID와 이름을 설정한다.– HOSTNAME: 호스트명을 설정한다.– HISTSIZE: 히스토리 크기를 설정한다.• 기본 접근 권한을 설정한다(5장 참조).• /etc/profile.d/*.sh를 실행한다.
/etc/bashrc	<ul style="list-style-type: none">• 시스템 공통으로 적용되는 함수와 앤리어스를 설정한다.• 기본 프롬프트를 설정한다.• 서브 셸을 위한 명령 경로를 설정한다.• 서브 셸을 위한 기본 접근 권한을 설정한다.
/etc/profile.d/*sh	<ul style="list-style-type: none">• 언어나 명령별로 각각 필요한 환경을 설정한다.• 필요시 설정 파일을 추가한다.

07.환경 설정 파일

■ 시스템 환경 설정 파일

```
[user1@localhost ch4]$ cd  
[user1@localhost ~]$ more /etc/profile  
# /etc/profile  
  
# System wide environment and startup programs, for login setup  
# Functions and aliases go in /etc/bashrc  
  
# It's NOT a good idea to change this file unless you know what you  
# are doing. It's much better to create a custom.sh shell script in  
# /etc/profile.d/ to make custom changes to your environment, as this  
# will prevent the need for merging in future updates.  
  
pathmunge () {  
    case ":${PATH}:" in  
(생략)
```

07.환경 설정 파일

■ 사용자 환경 설정 파일

- 각 사용자의 홈 디렉터리에 숨김 파일로 있으며 사용자가 내용을 수정하고 관리할 수 있음
- 사용자가 로그인하면 제일 먼저 시스템 환경 설정 파일이 실행되어 시스템 공통 환경을 만듬
- 이후 사용자 환경 설정 파일을 순서대로 실행하여 사용자별 환경을 설정

표 4-16 배시셸의 사용자 환경 설정 파일

파일	기능
~/.bash_profile	<ul style="list-style-type: none">• .bashrc 파일이 있으면 실행한다.• 경로 추가 등 사용자가 정의하는 환경 설정 파일이다.
~/.bashrc	<ul style="list-style-type: none">• /etc/bashrc 파일이 있으면 실행한다.• 사용자가 정의하는 앤리어스나 함수 등을 설정한다.
~/.bash_logout	<ul style="list-style-type: none">• 로그아웃 시 실행할 필요가 있는 함수 등을 설정한다.

07.환경 설정 파일

■ 사용자 환경 설정 파일

- .bash_logout 파일에 사용자가 필요한 내용을 각 파일에 추가로 설정

```
[user1@localhost ~]$ cat .bash_profile      → .bash_profile 출력
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs
[user1@localhost ~]$ cat .bashrc           → .bashrc 출력
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific environment
PATH="$HOME/.local/bin:$HOME/bin:$PATH"

export PATH

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
[user1@localhost ~]$ cat .bash_logout        → .bash_logout 출력
# ~/.bash_logout
```

07.환경 설정 파일

■ 사용자 환경 설정 파일

- 사용자 환경 설정 파일 만들기
 - vi로 .bashrc 파일을 열고 앤리어스를 입력

07.환경 설정 파일

■ 사용자 환경 설정 파일

- 사용자 환경 설정 파일 적용하기
 - 사용자 환경 설정 파일을 수정을 적용하는 방법
 - 로그아웃 했다가 다시 로그인
 - .(마침표) 명령이나 source 명령을 사용하여 환경 설정 파일을 실행
- 다른 셀의 환경 설정 파일

표 4-17 다른 셀의 환경 설정 파일

셀	시스템 초기화 파일	사용자 초기화 파일	실행 조건	실행 시기		
				로그인	서브 셀	로그아웃
본셀	/etc/profile	\$HOME/.profile	-	○		
콘셀	/etc/profile	\$HOME/.profile	-	○		
		\$HOME/.kshrc	ENV 변수 설정	○	○	
C셀	/etc/.login	\$HOME/.login	-	○		
		\$HOME/.cshrc	-	○	○	
		\$HOME/.logout	-			○

07.환경 설정 파일

■ [따라해보기] 사용자 환경 설정 파일 수정하기

- ① .bash_profile 파일을 vi로 실행

```
[user1@localhost ~]$ vi .bash_profile
```

- ② .bash_profile 파일의 내용에 행 번호가 보이도록 설정

```
1 # .bash_profile
2
3 # Get the aliases and functions
4 if [ -f ~/.bashrc ]; then
5     . ~/.bashrc
6 fi
7
8 # User specific environment and startup programs
~
:set nu
```

- ③ 8행 다음에 환경변수 HISTSIZE를 추가하고 크기를 1500으로 설정한 뒤 파일을 저장한 후 종료

(생략)

```
8 # User specific environment and startup programs
9 HISTSIZE=1500
~
:wq
[user1@localhost ~]$
```

07.환경 설정 파일

■ [따라해보기] 사용자 환경 설정 파일 수정하기

- ④ .bashrc 파일을 vi로 열고 행 번호가 보이도록 설정

```
[user1@localhost ~]$ vi .bashrc
 1 # .bashrc
 2
 3 # Source global definitions
 4 if [ -f /etc/bashrc ]; then
 5     . /etc/bashrc
 6 fi
 7
 8 # User specific environment
 9 PATH="$HOME/.local/bin:$HOME/bin:$PATH"
10 export PATH
11
12 # Uncomment the following line if you don't like systemctl's auto-paging feature:
13 # export SYSTEMD_PAGER=
14
15 # User specific aliases and functions
16 alias rm='rm -i'
17 alias ls='ls -F'
~
:set nu
```

07.환경 설정 파일

■ [따라해보기] 사용자 환경 설정 파일 수정하기

- ⑤ 9행의 경로에 /etc 경로를 추가하고 파일을 저장한 후 종료 경로를 추가할 때 구분자로 :을 입력

(생략)

```
9 PATH="$HOME/.local/bin:$HOME/bin:$PATH:/etc"
```

~

:wq

```
[user1@localhost ~]$
```

- ⑥ .bash_profile 파일을 실행 .bashrc는 .bash_profile이 실행시키므로 자동으로 실행됨

- 경로에 /etc 디렉터리가 추가되었는지 확인

```
[user1@localhost ~]$ echo $PATH  
/home/user1/.local/bin:/home/user1/bin:/home/user1/.local/bin:/home/user1/bin:/home/  
user1/.local/bin:/home/user1/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/etc
```

CentOS 리눅스

시스템 & 네트워크



Chapter 05. 파일 접근 권한 관리

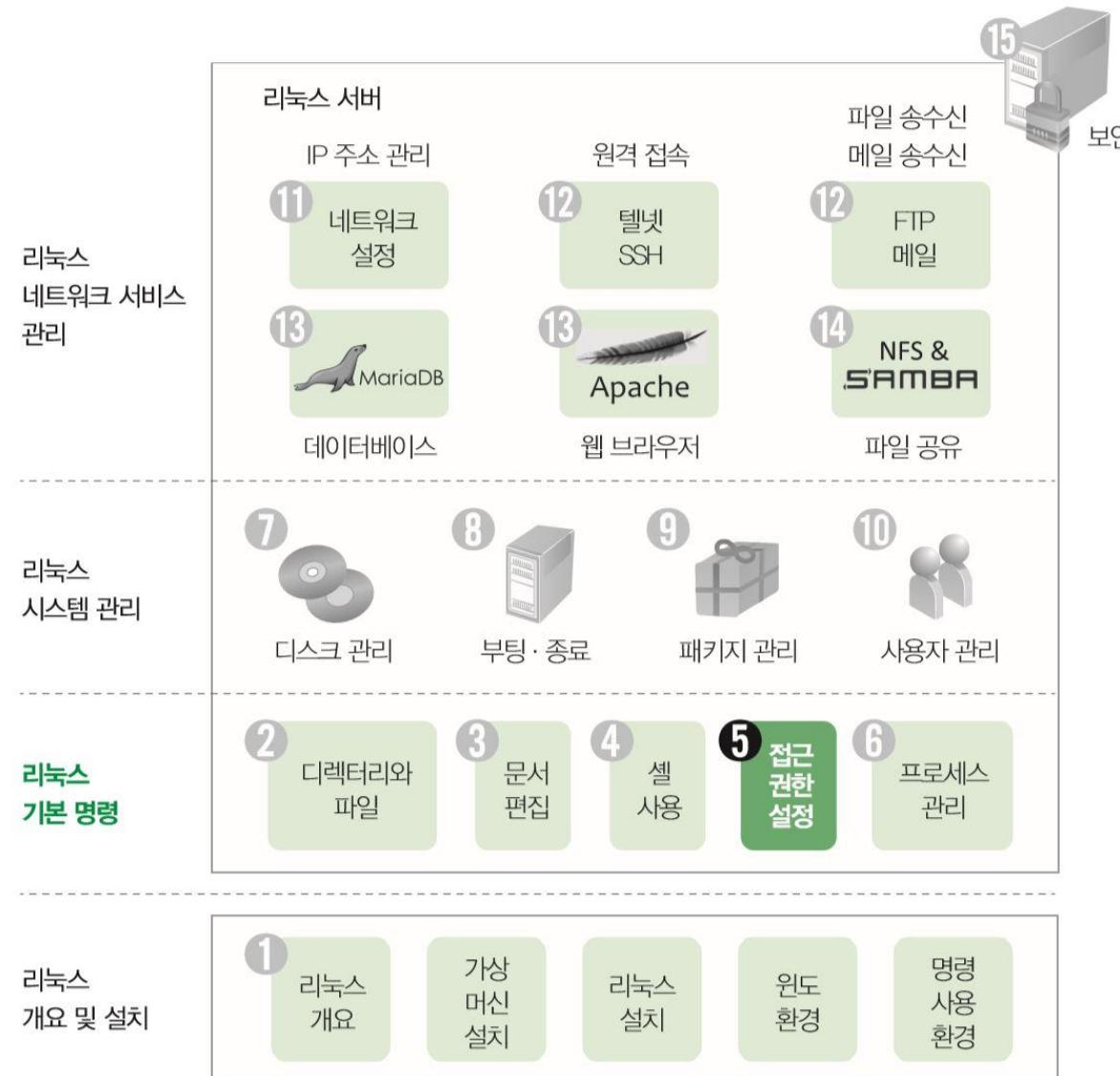
목차

00. 개요
01. 파일 속성
02. 파일 접근 권한
03. 기호를 이용한 파일 접근 권한 변경
04. 숫자를 이용한 파일 접근 권한 변경
05. 기본 접근 권한 설정
06. 특수 접근 권한 설정

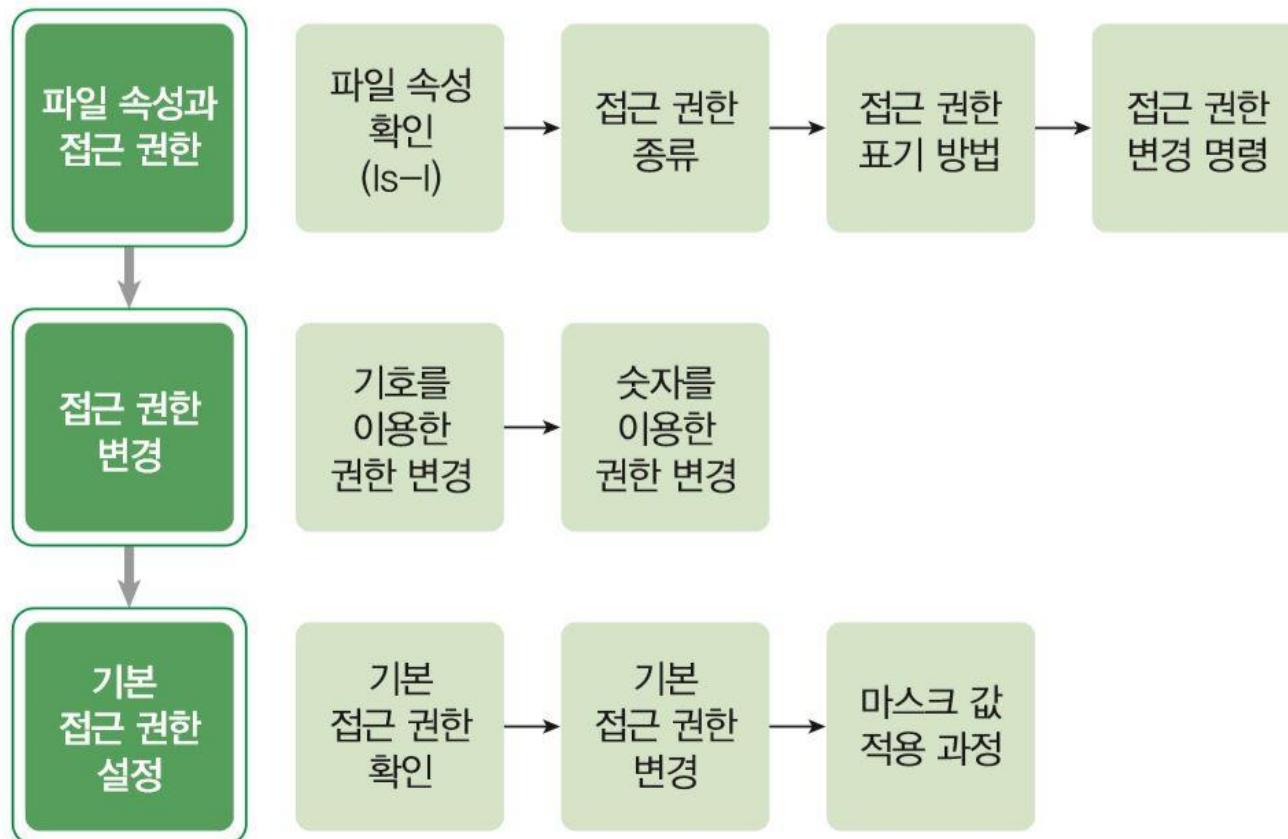
학습목표

- 파일의 속성과 접근 권한의 개념을 이해할 수 있다.
- 접근 권한의 종류와 표기 방법을 이해하고 설명할 수 있다.
- 기호와 숫자로 표기된 접근 권한을 바꿀 수 있다.
- 기본 접근 권한을 확인하고 원하는 값으로 바꿀 수 있다.

00. 개요



00. 개요



01. 파일 속성

■ 파일 접근 권한 보호

- 리눅스는 파일에 무단으로 접근하는 것을 방지하고 보호하는 기능을 제공
- 사용자는 자신의 파일과 디렉터리 중에서 다른 사용자가 접근해도 되는 것과 그렇지 않은 것을 구분하여 접근 권한을 제한

■ 파일 속성

```
[user1@localhost ~]$ ls -l /etc/hosts  
-rw-r--r--. 1 root root 158 9월 10 2018 /etc/hosts
```

표 5-1 파일의 속성

번호	속성 값	의미
①	-	파일의 종류(-: 일반 파일, d: 디렉터리)
②	rw- r--r--	파일을 읽고 쓰고 실행할 수 있는 접근 권한 표시
③	1	하드 링크의 개수
④	root	파일 소유자의 로그인 ID
⑤	root	파일이 속한 그룹 이름
⑥	158	파일의 크기(바이트 단위)
⑦	9월 10 2018	파일이 마지막으로 수정된 날짜
⑧	/etc/hosts	파일명

01. 파일 속성

■ 파일 속성

■ 파일의 종류

- 파일 속성의 첫 번째 항목은 파일의 종류를 표시
- -는 일반 파일을, d는 디렉터리를 의미
- 파일의 종류를 알려주는 명령

file

- **기능** 지정한 파일의 종류를 알려준다.
- **형식** file [파일명]
- **사용 예** file /etc/services

- ex) file 명령의 사용

```
[user1@localhost ~]$ file /etc/hosts /usr/bin  
/etc/hosts: ASCII text  
/usr/bin: directory
```

- /etc/hosts는 일반 텍스트 파일이고 /usr/bin은 디렉터리임

■ 파일의 접근 권한 표시

- 파일의 소유자와 그룹이나 기타 사용자들이 파일에 대해 가지고 있는 접근 권한을 표시

■ 하드 링크의 개수

- 하드 링크는 한 파일에 대해 여러 개의 파일명을 가질 수 있도록 하는 기능

01. 파일 속성

■ 파일 속성

- 파일 소유자의 로그인 ID
 - 리눅스에서 모든 파일은 소유자가 있음
- 파일 소유자의 그룹 이름
 - ls -l 명령에서 출력되는 그룹명은 파일이 속한 그룹
 - 사용자가 속한 기본 그룹은 시스템 관리자가 사용자를 등록할 때 결정
 - 사용자가 속한 그룹을 알려주는 명령은 groups

groups

- **기능** 사용자가 속한 그룹을 알려준다.
- **형식** groups [사용자명]

- ex) groups 명령의 사용

```
[user1@localhost ~]$ groups  
user1  
[user1@localhost ~]$ groups root  
root : root
```

■ 파일의 크기

- 파일의 크기를 바이트 단위로 알림

■ 파일이 마지막으로 수정된 날짜

- 파일이 마지막으로 수정된 날짜와 시간이 표시. 연도가 표시되지 않으면 올해를 의미

02. 파일 접근 권한

■ 파일 접근 권한

- 접근 권한 : 해당 파일을 읽고 쓰고 실행할 수 있는 권한, 사용자의 파일을 보호하는 가장 기본적인 보안 기능
- 리눅스는 사용자를 파일 소유자, 파일이 속한 그룹, 그 외 기타 사용자 세 카테고리로 구분하여 접근 권한을 적용
- 사용자 카테고리별로 접근 권한을 다르게 부여하여 파일을 보호할 수 있음

■ 접근 권한 종류

- 읽기 권한, 쓰기 권한, 실행 권한 등 세 가지로 구성
- 대상이 파일인지 디렉터리인지에 따라 그 의미가 약간 다르게 해석

표 5-2 파일과 디렉터리의 접근 권한

권한	파일	디렉터리
읽기	파일을 읽거나 복사할 수 있다.	ls 명령으로 디렉터리 목록을 볼 수 있다(ls 명령의 옵션은 실행 권한이 있어야 사용할 수 있다).
쓰기	파일을 수정 · 이동 · 삭제할 수 있다(디렉터리에 쓰기 권한이 있어야 한다).	파일을 생성하거나 삭제할 수 있다.
실행	파일을 실행할 수 있다(셸 스크립트나 실행 파일의 경우).	cd 명령을 사용할 수 있다. 파일을 디렉터리로 이동하거나 복사할 수 있다.

02. 파일 접근 권한

■ 접근 권한 표기 방법

- 사용자 카테고리별로 누가 파일을 읽고 쓰고 실행할 수 있는지를 문자로 표현한 것
- 읽기 권한은 r, 쓰기 권한은 w, 실행 권한은 x로 나타내며, 해당 권한이 없는 경우에는 -로 표기
- 사용자 카테고리별로 세 가지 권한의 부여 여부를 rwx 세 문자를 둘이어서 표기

```
[user1@localhost ~]$ ls -l /etc/hosts  
-rw-r--r--. 1 root root 158 9월 10 2018 /etc/hosts
```

- 예시의 실제 접근 권한: rw-r--r--
 - (소유자가 읽기와 쓰기 권한을 가지고 있고 그룹과 기타 사용자는 읽기 권한만 가지고 있음을 나타냄)



그림 5-1 파일의 접근 권한 표기

표 5-3 다양한 접근 권한 조합의 예

접근 권한	의미
rwxr-xr-x	소유자는 읽기 · 쓰기 · 실행 권한을 모두 가지고 그룹과 기타 사용자는 읽기 · 실행 권한을 가지고 있다.
r-xr-xr-x	소유자, 그룹, 기타 사용자 모두 읽기 · 실행 권한을 가지고 있다.
rw-----	소유자만 읽기 · 쓰기 권한을 가지고 그룹과 기타 사용자는 아무 권한이 없다.
rw-rw-rw-	소유자, 그룹, 기타 사용자 모두 읽기 · 쓰기 권한을 가지고 있다.
rwxrwxrwx	소유자, 그룹, 기타 사용자 모두 읽기 · 쓰기 · 실행 권한을 가지고 있다.
rwx-----	소유자만 읽기 · 쓰기 · 실행 권한을 가지고 그룹과 기타 사용자는 아무 권한이 없다.
r-----	소유자만 읽기 권한을 가지고 있다.

02. 파일 접근 권한

■ 접근 권한 변경 명령

- 접근 권한을 바꾸려면 chmod 명령을 사용

chmod

- 기능** 파일이나 디렉터리의 접근 권한을 변경한다.
- 형식** chmod [옵션] 권한 파일(디렉터리)
- 옵션** -R: 하위 디렉터리까지 모두 변경할 수 있다.

- chmod 명령으로 접근 권한을 변경할 때 기호 모드와 숫자 모드를 사용할 수 있음
 - 기호 모드: 접근 권한을 변경하기 위해 문자와 기호를 사용하여 권한을 표시
 - 숫자 모드: 접근 권한을 변경하기 위해 숫자를 사용

03. 기호를 이용한 파일 접근 권한 변경

■ 기호를 이용한 파일 접근 권한 변경

- 기호 모드 : 기호를 이용하여 파일 접근 권한을 변경하
- 사용자 카테고리 문자, 연산자 기호, 접근 권한 문자로 구성



그림 5-2 기호 모드의 구성 요소

- 사용자 카테고리: 소유자, 그룹, 기타 사용자를 나타내는 문자로 표기
- 연산자: 권한 부여나 제거를 나타내는 기호로 표기
- 접근 권한 기호: 읽기, 쓰기, 실행을 나타내는 문자를 사용

표 5-4 기호 모드에서 사용하는 문자와 기호

표 5-5 기호 모드를 사용한 접근 권한 설정의 예

구분	문자/기호	의미	권한 표기	의미
사용자 카테고리 문자	u	파일 소유자	u+w	소유자(u)에게 쓰기(w) 권한 부여(+)
	g	파일 소유 그룹	u-x	소유자(u)의 실행(x) 권한 제거(-)
	o	소유자와 그룹 이외의 기타 사용자	g+w	그룹(g)에 쓰기(w) 권한 부여(+)
	a	전체 사용자	o-r	기타 사용자(o)의 읽기(r) 권한 제거(-)
연산자 기호	+	권한 부여	g+wx	그룹(g)에 쓰기(w)와 실행(x) 권한 부여(+)
	-	권한 제거	+wx	모든 사용자에게 쓰기(w)와 실행(x) 권한 부여(+)
	=	접근 권한 설정	a+rwx	모든 사용자에게 읽기(r), 쓰기(w), 실행(x) 권한 부여(=)
접근 권한 문자	r	읽기 권한	u=rwx	소유자(u)에게 읽기(r), 쓰기(w), 실행(x) 권한 부여(=)
	w	쓰기 권한	go+w	그룹(g)과 기타 사용자(o)에게 쓰기(w) 권한 부여(+)
	x	실행 권한	u+x, go+w	소유자(u)에게 실행(x) 권한을 부여하고(+) 그룹(g)과 기타 사용자(o)에게 쓰기(w) 권한 부여(+)

03. 기호를 이용한 파일 접근 권한 변경

■ 기호를 이용한 파일 접근 권한 변경

- ch5 디렉터리와 파일을 준비

```
[user1@localhost ~]$ mkdir linux_ex/ch5  
[user1@localhost ~]$ cd linux_ex/ch5  
[user1@localhost ch5]$ cp /etc/hosts test.txt
```

- 현재 접근 권한을 확인 (현재 접근 권한은 rw-r--r--)

```
[user1@localhost ch5]$ ls -l  
합계 4  
-rw-r--r--. 1 user1 user1 158 11월  2 21:26 test.txt
```

- 소유자의 쓰기 권한을 제거 (u-w)

```
[user1@localhost ch5]$ chmod u-w test.txt  
[user1@localhost ch5]$ ls -l  
합계 4  
-r--r--r--. 1 user1 user1 158 11월  2 21:26 test.txt
```

03. 기호를 이용한 파일 접근 권한 변경

■ [따라해보기] 기호 모드로 접근 권한 변경하기

- ① 그룹에 쓰기와 실행 권한을 부여 (g+wx)

```
[user1@localhost ch5]$ chmod g+wx test.txt  
[user1@localhost ch5]$ ls -l  
합계 4  
-r--rwxr--. 1 user1 user1 158 11월  2 21:26 test.txt*
```

- ② 기타 사용자에게 실행 권한을 부여 (o+x)

```
[user1@localhost ch5]$ chmod o+x test.txt  
[user1@localhost ch5]$ ls -l  
합계 4  
-r--rwxr-x. 1 user1 user1 158 11월  2 21:26 test.txt*
```

- ③ 그룹과 기타 사용자의 실행 권한을 제거 (go-x)

```
[user1@localhost ch5]$ chmod go-x test.txt  
[user1@localhost ch5]$ ls -l  
합계 4  
-r--rw-r--. 1 user1 user1 158 11월  2 21:26 test.txt
```

03. 기호를 이용한 파일 접근 권한 변경

■ [따라해보기] 기호 모드로 접근 권한 변경하기

- ④ 모두에게 실행 권한을 부여 (a+x)

```
[user1@localhost ch5]$ chmod a+x test.txt  
[user1@localhost ch5]$ ls -l  
합계 4  
-r-xrwxr-x. 1 user1 user1 158 11월  2 21:26 test.txt*
```

- ⑤ 소유자에게 쓰기 권한을 부여하고 그룹의 쓰기 권한은 제거 (u+w,g-w)

```
[user1@localhost ch5]$ chmod u+w,g-w test.txt  
[user1@localhost ch5]$ ls -l  
합계 4  
-rwxr-xr-x. 1 user1 user1 158 11월  2 21:26 test.txt*
```

04. 숫자를 이용한 파일 접근 권한 변경

■ 숫자를 이용한 파일 접근 권한 변경

- 기호 모드는 전체적으로 권한을 조정할 때는 문자의 조합이 복잡해짐
- 이럴 때 숫자 모드로 chmod 명령을 사용하면 권한을 한 번에 원하는 대로 변경할 수 있어서 매우 편리

■ 숫자로 환산하는 방법

- 숫자 모드에서는 각 권한이 있고 없고를 0과 1로 표기하고 이를 다시 환산하여 숫자로 표현
- 카테고리별로 권한의 조합에 따라 0부터 7로 나타내는 것

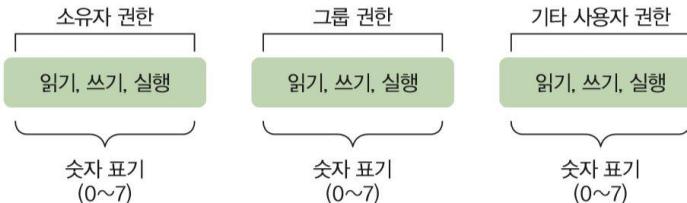


그림 5-3 숫자 모드의 구성 요소

04. 숫자를 이용한 파일 접근 권한 변경

■ 숫자로 환산하는 방법

▪ 권한 묶음(rwx)을 숫자로 환산하는 방법

- ① r-x라고 할 때 권한이 있는 것은 1로, 없는 것은 0으로 변환
- ② 2진수 1, 0, 1로 변환
- ③ 2진수를 각 자릿수별로 10진수로 환산하면 4, 0, 1이 됨
- ④ 세 숫자를 더하면
- ⑤ 최종 권한 값은 5가 됨

표 5-6 접근 권한과 숫자의 대응 관계

접근 권한	환산	숫자	의미
rwx	111 → 4+2+1	7	읽기, 쓰기, 실행
rw-	110 → 4+2+0	6	읽기, 쓰기
r-x	101 → 4+0+1	5	읽기, 실행
r--	100 → 4+0+0	4	읽기
-wx	011 → 0+2+1	3	쓰기, 실행
-w-	010 → 0+2+0	2	쓰기
--x	001 → 0+0+1	1	실행
---	000 → 0+0+0	0	권한이 없음

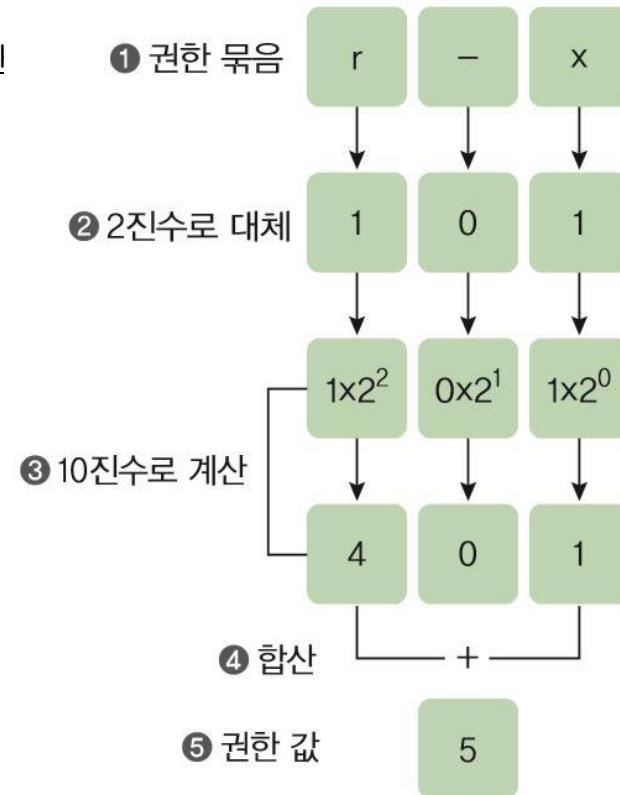


그림 5-4 권한을 숫자로 환산하는 과정

04. 숫자를 이용한 파일 접근 권한 변경

■ 숫자로 환산하는 방법

- 소유자 권한, 그룹 권한, 기타 사용자 권한을 각각 숫자로 환산하여 전체 접근 권한을 나타낼 수 있음

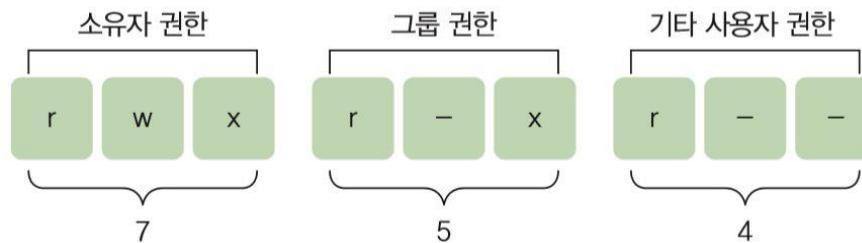


그림 5-5 전체 접근 권한을 숫자로 표기한 예

표 5-7 숫자로 표현한 접근 권한의 예

접근 권한	숫자 모드	접근 권한	숫자 모드
rwxrwxrwx	777	rw-r--r--	644
rwxr-xr-x	755	rwX-----	700
rw-rw-rw	666	rw-r-----	640
r-xr-xr-x	555	r-----	400

04. 숫자를 이용한 파일 접근 권한 변경

■ 숫자 모드를 이용한 접근 권한 변경

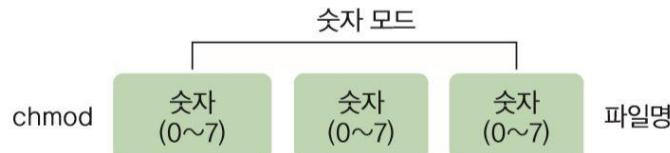


그림 5-6 숫자 모드를 이용한 접근 권한 변경

- 숫자의 각 위치가 사용자 카테고리를 나타내기 때문에 사용자 카테고리를 따로 지정할 필요가 없음
- 항상 세 자리 수를 사용해야 하므로 변경하려는 사용자 카테고리의 권한 뿐만 아니라 그룹과 기타 사용자의 권한도 반드시 같이 명시

04. 숫자를 이용한 파일 접근 권한 변경

■ 숫자 모드를 이용한 접근 권한 변경

- ex) test.txt 파일의 현재 접근 권한은 다음과 같이 644(rw-r--r--)

```
[user1@localhost ch5]$ ls -l  
합계 4  
-rw-r--r--. 1 user1 user1 158 11월  2 21:26 test.txt
```

- 소유자의 쓰기 권한을 제거: 기호로는 u-w이지만 이를 숫자로 표기하면 r--r--r--이므로 444

```
[user1@localhost ch5]$ chmod 444 test.txt  
[user1@localhost ch5]$ ls -l  
합계 4  
-r--r--r--. 1 user1 user1 158 11월  2 21:26 test.txt
```

- 그룹에 쓰기와 실행 권한을 부여: r--rwxr--이므로 이를 숫자로 표기하면 474

```
[user1@localhost ch5]$ chmod 474 test.txt  
[user1@localhost ch5]$ ls -l  
합계 4  
-r--rwxr--. 1 user1 user1 158 11월  2 21:26 test.txt*
```

04. 숫자를 이용한 파일 접근 권한 변경

■ [따라해보기] 숫자 모드로 접근 권한 변경하기

- ① 기타 사용자에게 실행 권한을 부여 (o+x) r—rwxr-x → 475

```
[user1@localhost ch5]$ chmod 475 test.txt  
[user1@localhost ch5]$ ls -l  
합계 4  
-r--rwxr-x. 1 user1 user1 158 11월  2 21:26 test.txt*
```

- ② 그룹과 기타 사용자의 실행 권한을 제거 (go-x) r—rw-r-- → 464

```
[user1@localhost ch5]$ chmod 464 test.txt  
[user1@localhost ch5]$ ls -l  
합계 4  
-r--rw-r--. 1 user1 user1 158 11월  2 21:26 test.txt
```

- ③ 모두에게 실행 권한을 부여 (a+x) r-xrwxr-x → 575

```
[user1@localhost ch5]$ chmod 575 test.txt  
[user1@localhost ch5]$ ls -l  
합계 4  
-r-xrwxr-x. 1 user1 user1 158 11월  2 21:26 test.txt*
```

04. 숫자를 이용한 파일 접근 권한 변경

■ [따라해보기] 숫자 모드로 접근 권한 변경하기

- ④ 소유자에게 쓰기 권한을 부여하고 그룹의 쓰기 권한은 제거 (u+w,g-w) rwxr-xr-x → 755

```
[user1@localhost ch5]$ chmod 755 test.txt  
[user1@localhost ch5]$ ls -l  
합계 4  
-rwxr-xr-x. 1 user1 user1 158 11월  2 21:26 test.txt*
```

- ⑤ 소유자의 권한만 남기고 나머지 사용자의 권한은 모두 제거 rwx----- → 700

```
[user1@localhost ch5]$ chmod 700 test.txt  
[user1@localhost ch5]$ ls -l  
합계 4  
-rwx-----. 1 user1 user1 158 11월  2 21:26 test.txt*
```

05. 기본 접근 권한 설정

■ 기본 접근 권한

- 리눅스에서는 파일이나 디렉터리를 생성할 때 기본 접근 권한이 자동적으로 설정
- 일반 파일의 경우 소유자와 그룹은 읽기와 쓰기 권한이 설정되고 기타 사용자는 읽기 권한만 설정
- 디렉터리의 경우 소유자와 그룹은 읽기, 쓰기, 실행 권한이 설정되고 기타 사용자는 읽기, 실행 권한만 설정
- 다음 예에서 새로 만든 centos.txt 파일과 temp 디렉터리의 접근 권한을 보면 기본 접근 권한으로 설정

```
[user1@localhost ch5]$ touch centos.txt  
[user1@localhost ch5]$ mkdir temp  
[user1@localhost ch5]$ ls -l  
합계 4  
-rw-rw-r--. 1 user1 user1 0 11월 2 21:49 centos.txt  
drwxrwxr-x. 2 user1 user1 6 11월 2 21:49 temp/  
-----. 1 user1 user1 158 11월 2 21:26 test.txt
```

05. 기본 접근 권한 설정

■ 기본 접근 권한 확인 및 변경

- umask 명령 : 기본 접근 권한을 확인하고 설정하는 데 사용

umask

- 기능 기본 접근 권한을 출력하거나 변경한다.
- 형식 umask [옵션] [마스크 값]
- 옵션 -S: 마스크 값을 문자로 출력한다.
- 사용 예 umask 022
umask

- 예시에서 0002에서 맨 앞의 0은 다른 용도가 있으며, 기본 접근 권한은 002로 적용

```
[user1@localhost ch5]$ umask  
0002
```

■ 마스크 값의 의미

- 마스크 값 : 파일이나 디렉터리 생성 시 부여하지 않을 권한을 지정해놓는 것
- 마스크 값이 002일 경우 이는 -----w-이고, 기타 사용자에게 쓰기 권한은 부여하지 않겠다는 뜻
- ex) : umask 명령에 -S 옵션을 적용하여 마스크 값을 문자로 출력

```
[user1@localhost ch5]$ umask -S  
u=rwx, g=rwx, o=rx
```

- 'u=rwx, g=rwx, o=rx': 사용자와 그룹에는 읽기, 쓰기, 실행 권한을 부여, 기타 사용자의 경우에는 읽기와 실행 권한만 부여

05. 기본 접근 권한 설정

■ 기본 접근 권한 확인 및 변경

- umask 명령 : 기본 접근 권한을 확인하고 설정하는 데 사용

umask

- 기능 기본 접근 권한을 출력하거나 변경한다.
- 형식 umask [옵션] [마스크 값]
- 옵션 -S: 마스크 값을 문자로 출력한다.
- 사용 예 umask 022
umask

- 예시에서 0002에서 맨 앞의 0은 다른 용도가 있으며, 기본 접근 권한은 002로 적용

```
[user1@localhost ch5]$ umask  
0002
```

■ 마스크 값의 의미

- 마스크 값 : 파일이나 디렉터리 생성 시 부여하지 않을 권한을 지정해놓는 것
- 마스크 값이 002일 경우 이는 -----w-이고, 기타 사용자에게 쓰기 권한은 부여하지 않겠다는 뜻
- ex) : umask 명령에 -S 옵션을 적용하여 마스크 값을 문자로 출력

```
[user1@localhost ch5]$ umask -S  
u=rwx, g=rwx, o=rx
```

- 'u=rwx, g=rwx, o=rx': 사용자와 그룹에는 읽기, 쓰기, 실행 권한을 부여, 기타 사용자의 경우에는 읽기와 실행 권한만 부여

05.기본 접근 권한 설정

■ 기본 접근 권한 확인 및 변경

▪ 마스크 값 변경하기

- umask 명령으로 마스크 값을 변경

```
[user1@localhost ch5]$ umask 077  
[user1@localhost ch5]$ umask  
0077
```

- umask로 마스크 값을 바꾸면 파일이나 디렉터리를 생성할 때 적용되는 기본 접근 권한도 바뀜

```
[user1@localhost ch5]$ touch linux.txt  
[user1@localhost ch5]$ ls -l  
합계 4  
  
-rw-rw-r--. 1 user1 user1 0 11월 2 21:49 centos.txt  
-rw-----. 1 user1 user1 0 11월 2 21:56 linux.txt  
drwxrwxr-x. 2 user1 user1 6 11월 2 21:49 temp/  
-----. 1 user1 user1 158 11월 2 21:26 test.txt
```

05. 기본 접근 권한 설정

■ 마스크 값의 적용 과정

- 마스크 값을 비트로 표시했을 때 그 값이 1인 경우 대응하는 권한은 제외

표 5-8 umask 진리표

요청 권한	1	1	0	0
마스크	1	0	1	0
부여된 권한	0	1	0	0

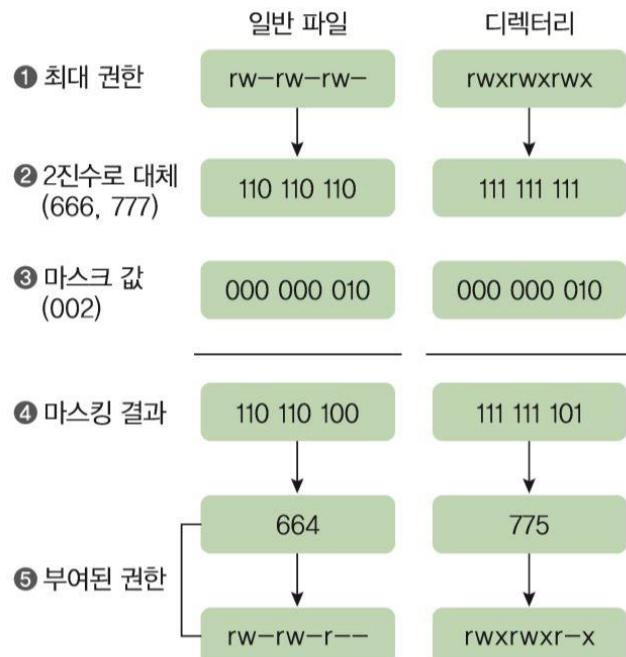
- 마스크 값이 1이면 요청 권한이 1인지 0인지에 상관없이 부여되는 권한은 0이 됨
- 마스크 값이 0일 경우 해당 요청 권한이 그대로 부여됨
- 리눅스 시스템은 파일이나 디렉터리를 생성할 때 부여하려는 접근 권한과 마스크 값을 진리표를 기준으로 마스킹하여 기본 접근 권한을 결정

05. 기본 접근 권한 설정

■ 마스크 값의 적용 과정

▪ 마스크 값 계산 방법

- 일반 파일이 가질 수 있는 최대 접근 권한은 666 (rw-rw-rw)
- 현재 마스크 값이 002일 때 666과 002를 [표 5-8]에 따라 마스킹하면 결과가 664임을 알 수 있음
- 따라서 마스크 값이 002일 때 일반 파일을 생성하면 기본 접근 권한이 664

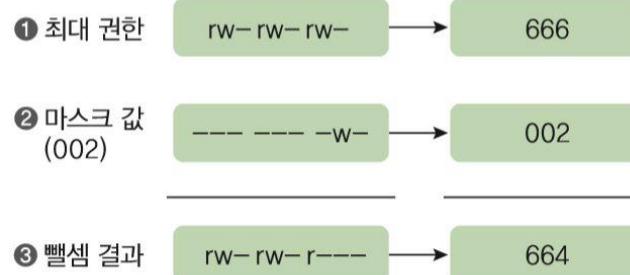


(a) 마스크 값을 적용하는 과정

그림 5-7 마스크 값 계산 방법

표 5-8 umask 진리표

요청 권한	1	1	0	0
마스크	1	0	1	0
부여된 권한	0	1	0	0



(b) 마스크 값에 빼셈을 적용하는 과정

05. 기본 접근 권한 설정

■ 마스크 값의 적용 과정

■ 여러가지 마스크 값

- 리눅스의 기본 마스크 값은 002인데 이를 필요에 따라 적절히 변경하여 자신의 파일과 디렉터리를 보호할 수 있음

표 5-9 마스크 값의 의미

마스크 값	일반 파일	디렉터리	의미
022	644	755	그룹과 기타 사용자는 읽기만 가능하다.
077	600	700	그룹과 기타 사용자의 접근 권한을 모두 제거한다.
027	640	750	그룹은 읽기와 실행만 가능하고 기타 사용자의 접근 권한을 모두 제거한다.

- 마스크 값이 파일에는 적합하지만 디렉터리에는 적합하지 않을 수도 있기 때문에 umask로 마스크 값을 바꿀 때 파일과 디렉터리에 모두 적용해봐야 함
- ex) : 마스크 값이 026일 경우, 파일은 640으로 적합하지만 디렉터리는 751이 되어 기타 사용자의 접근 권한이 이상

05. 기본 접근 권한 설정

■ [따라해보기] 기본 접근 권한 변경하기

- ① 현재의 기본 접근 권한을 확인

```
[user1@localhost ch5]$ umask  
0077
```

- ② 그룹과 기타 사용자가 읽기와 실행을 할 수 없도록 기본 접근 권한을 변경

```
[user1@localhost ch5]$ umask 022  
[user1@localhost ch5]$ umask  
0022
```

- ③ 파일을 생성하여 기본 접근 권한이 변경되었는지 확인

```
[user1@localhost ch5]$ touch mask.txt  
[user1@localhost ch5]$ ls -l mask.txt  
-rw-r--r--. 1 user1 user1 0 11월 2 22:02 mask.txt
```

06. 특수 접근 권한 설정

■ 특수 접근 권한

- 접근 권한은 원래 4자리 숫자

```
[user1@localhost ch5]$ umask  
0002
```

- 생략된 맨 앞자리는 특수 접근 권한 의미
- 맨 앞자리 숫자가 0이면 일반적인 접근 권한이지만 이 숫자가 1, 2, 4이면 특수 접근 권한이 설정
- SetUID : 맨 앞자리가 4
- SetGID : 맨 앞자리가 2
- 스티키 비트(sticky bit) : 맨 앞자리가 1

06. 특수 접근 권한 설정

■ SetUID

- SetUID가 설정된 파일을 실행하면 해당 파일이 실행되는 동안에는 파일을 실행한 사용자의 권한이 아니라 파일 소유자의 권한이 적용
- ex) set.exe 파일을 만들고 실행 권한을 부여

```
[user1@localhost ch5]$ touch set.exe  
[user1@localhost ch5]$ chmod 755 set.exe      ➔ 실행 권한을 부여한다.  
[user1@localhost ch5]$ ls -l set.exe  
-rwxr-xr-x. 1 user1 user1 0 11월  2 22:09 set.exe*
```

- ex) set.exe 파일 앞에 4755로 수정하면 다음과 같이 SetUID가 설정

```
[user1@localhost ch5]$ chmod 4755 set.exe  
[user1@localhost ch5]$ ls -l set.exe  
-rwsr-xr-x. 1 user1 user1 0 11월  2 22:09 set.exe*
```

- SetUID가 설정되면 소유자의 실행 권한에 's'가 표시
 - set.exe 파일을 실행하면 항상 user1의 권한을 가진다는 의미
- ex) passwd 명령

```
[user1@localhost ch5]$ ls -l /usr/bin/passwd  
-rwsr-xr-x. 1 root root 34928  5월 12 00:14 /usr/bin/passwd*
```

- passwd명령은 사용자 계정의 암호를 바꾸는 명령이나 root계정으로만 수정이 가능
- 그러나 passwd 명령에는 SetUID가 설정되어 있기 때문에 소유자인 root 권한으로 실행되어 변경 가능

06. 특수 접근 권한 설정

■ SetGID

- SetGID는 SetUID와 거의 동일
- SetGID가 설정된 파일을 실행하면 해당 파일이 실행 되는 동안에는 파일 소유 그룹의 권한으로 실행
- SetGID는 접근 권한의 맨 앞자리에 2를 설정해야 함
- set.exe 파일에 SetGID를 설정하면 다음과 같이 그룹의 실행 권한에 's'가 표시

```
[user1@localhost ch5]$ chmod 2755 set.exe  
[user1@localhost ch5]$ ls -l set.exe  
-rwxr-sr-x. 1 user1 user1 0 11월 2 22:09 set.exe*
```

■ 스티키 비트

- 스티키 비트는 디렉터리에 설정
- 디렉터리에 스티키 비트가 설정되어 있으면 이 디렉터리에는 누구나 파일을 생성할 수 있음
- 파일은 파일을 생성한 계정으로 소유자가 설정되며, 다른 사용자가 생성한 파일은 삭제 불가
- /tmp 디렉터리가 대표적
- 스티키 비트는 접근 권한에서 맨 앞자리에 1을 설정

```
[user1@localhost ch5]$ chmod 1755 temp  
[user1@localhost ch5]$ ls -ld temp  
drwxr-xr-t. 2 user1 user1 6 11월 2 21:49 temp/
```