목차

- 리눅스 시스템 관리

4. 파일시스템과 디스크 관리

1. 리눅스 파일시스템
2. 파일시스템 마운트
3. 디스크 관리

5. 리눅스의 부팅과 종료

1. 리눅스 시스템의 부팅
2. systemd 서비스

6. 소프트웨어 관리

1. 우분투 패키지
2. 우분투 패키지 설치
3. 소스 코드 설치

7. 사용자 관리

1. 사용자/그룹 관리
2. 사용자 정보 관리
3. 디스크 쿼터

- 리눅스 네트워크 관리

8. 네트워크 설정

1. 네트워크의 기초
2. 네트워크의 설정
3. 네트워크의 상태 확인

9. 원격 접속과 FTP

1. 텔넷과 SHH
2. VNC
3. 파일 송수신

10. Samba

1. Samba

1. 리눅스 개요 및 설치

1. 컴퓨터 기초

1. 컴퓨터의 유형(5가지)

- 메인프레임 컴퓨터, 슈퍼 컴퓨터

메인프레임 컴퓨터는 수천명의 사람들이 사용할 수 있는 컴퓨터,
슈퍼 컴퓨터는 복잡하고 빠른 계산을 수행할 수 있는 컴퓨터임.

- 서버(서버 컴퓨터)와 워크스테이션

서버는 다수의 사용자를 동시에 지원하는(serve) 컴퓨터이다. 지금은 서버가 메인프레임의 역할을 대체함.
워크스테이션은 고성능 데스크톱 컴퓨터이다. 전문인들이 주로 사용한다. 지금은 PC성능이 좋아짐에 따라 워크스테이션과 PC 사이의 구분이 불명확해짐

- 개인용 컴퓨터(PC. Personal Computer.)

- 모바일 컴퓨터

- 임베디드 컴퓨터, 사물인터넷, 유비쿼터스 컴퓨팅

소프트웨어로 프로그래밍 된 칩 자체가 사물에 삽입되고 일체화되는 것을 임베디드라고 함.

2. 컴퓨터의 구조

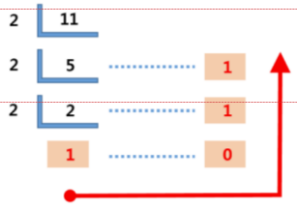
컴퓨터는 하드웨어와 소프트웨어로 구성됨

3. 진법의 번환

1) 10 진수 -> 2 진수

나름의 계산방법을 사용함. (나누기)

8 진수, 16 진수로 바꿀 때처럼 거듭제곱으로 표현해도 됨.



2) 10 진수 -> 8 진수, 16 진수

해당 10 진수 값을 8 또는 16 의 거듭제곱수로 표현함.

가장 큰 거듭제곱수로 나누고, 나머지를 그 다음으로 큰 거듭제곱수로 나눔.

각각의 몫이 8, 16 진수의 각 자리에 들어가는 수임.

$$
\begin{aligned}
1011 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\
&= 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 \\
&= 8 + 0 + 2 + 1 \\
&= 11
\end{aligned}
$$

3) 2 진수, 8 진수, 16 진수 -> 10 진수

각 자리 숫자와 자리 값들을 곱해서 더함.

자리 값은 자릿수를 n이라고 했을 때, 2진수는 $2^{(n-1)}$이고 8진수는 $8^{(n-1)}$이고 16진수는 $16^{(n-1)}$임.

4) 8 진수 <-> 2 진수

2 진수 세 자리와 8 진수 한 자리가 대응됨.

2 진수는 세 자리씩 끊어서 8 진수로 바꾸고, 8 진수는 한 자리씩 끊어서 2 진수로 바꿈.

(ex. 8 진수 075 는 2 진수로 111101 임.)

5) 16 진수 <-> 2 진수

2 진수 네 자리와 16 진수 한 자리가 대응됨.

2 진수는 네 자리씩 끊어서 16 진수로 바꾸고, 16 진수는 한 자리씩 끊어서 2 진수로 바꿈.

6) 16 진수 <-> 8 진수

그냥 10 진수로 먼저 바꾼 후에 전환하자.

메모 포함[이1]: 2, 8, 10, 16진수가 있는데
이들끼리 변환하는 경우의 수는 12가지임.
여기에 전부 정리했음.

메모 포함[이2]: 원리가 무엇일까?

메모 포함[이3]: 직관적으로 이해가 안 될 수도 있는데,
8진수를 10진수로 바꾼 후에 2진수로 바꿔봐도
이 방법과 결과가 같다.

4. OS (Operating System)

운영체제.

1) OS 의 핵심 기능

프로그램을 실행함.
문제에 더 쉽게 접근할 수 있게 함.
컴퓨터 시스템을 더 편리하게 사용할 수 있게 함.

2. 리눅스 기초

1. 리눅스의 역사

1) 핀란드 헬싱키대학의 리누스 베네딕스 토르발스가 리눅스 커널 개발, 1991년 8월 26일에 세상에 공개.
유닉스를 기반으로 만든 미닉스(MINIX)라는 교육용 운영체제를 이용하여 개발

2) GNU 프로젝트의 리눅스 커널로의 응용 프로그램 제공과 리눅스 재단을 통한 투자 등으로 리눅스는 발전해 나감. (커널만으로는 os를 사용할 수 없기 때문에 GNU의 응용프로그램 제공이 큰 기여를 했음)

3) 리눅스 커널에 GNU 응용 프로그램들을 적용하는 것은 쉽지 않은 작업이었기에 배포판을 뿌리기 시작.

4) 리눅스 배포판이 수많은 갈래로 퍼져나감.
크게 데비안 계열, 슬랙웨어 계열, 레드햇 계열로 나뉨.
리눅스 배포판은 "리눅스 커널+응용 프로그램" 으로 구성됨.

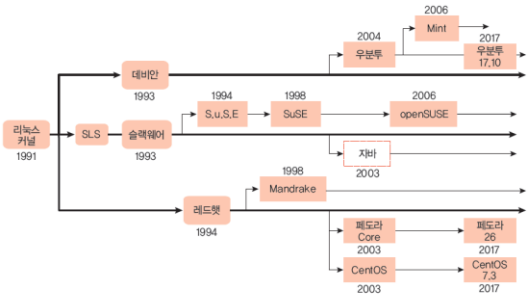


그림 1-5 주요 리눅스 배포판의 계통도

	Debian	Ubuntu	Cent OS	SUSE	Mint	Android	iOS	OSX	Win 7	Win 10
개발, 후원사	데비안 프로젝트	캐노니컬	레드햇	오픈수세 프로젝트, 노벨	클레망 르페브르 외	구글	애플	애플	MS	MS
컴퓨터와 랩톱에서 작동하는가?	o	o	o	o	o			o	o	o
휴대기기, 모바일기기에서 작동?		o				o	o			o
오픈소스?	o	o	o	o	o	o				
OS 계열	리눅스	리눅스 (영국)	리눅스 (미국)	리눅스 (독일)	리눅스 (우분투)	리눅스 기반	유닉스 기반 (BSD)	유닉스 기반	윈도	윈도

- 리눅스 입문용 대학강의와 리눅스 교재로는 우분투를 가장 많이 사용하고 있음, 라즈베리파이 등 단일기판 컴퓨터에서 사용 중
- Cent OS : Community Enterprise Operating System

메모 포함[이4]: IT산업계의 많은 분야에서 리눅스가 절반 이상의 점유율을 가지고 있다. (거의 대부분이 리눅스를 사용하는 분야도 많다.)

메모 포함[이5]: 리누스의 유닉스.

메모 포함[이6]: 매우 간단한 운영체제.

메모 포함[이7]: 리눅스는 유닉스를 기반으로 만들어졌기 때문에 유닉스와의 호환성이 좋다.

메모 포함[이8]: 그렇기의 리눅스의 정확한 이름은 GNU/리눅스라고 할 수 있다.

메모 포함[이9]: 보안은 유닉스가 리눅스와 윈도우보다 좋다.

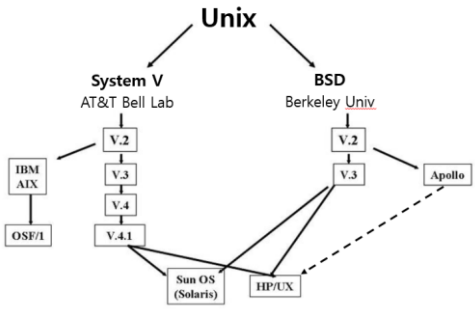
+ GNU 프로젝트 (GNU is Not Unix!) (그누)

리처드 스톨먼이 시작한 프로젝트로, 돈을 지불해야 하는 유닉스와는 다르게 자유 소프트웨어를 개발함.
유닉스 프로그램을 사용할 수 있게 함.
프로그램의 실행, 수정, 재배포에 대한 자유가 보장되고, 오픈소스 방식을 추구함.
GNU 프로젝트에서는 자체적으로 커널을 만들고 있었는데 실패하자 리눅스 커널과 협력하게 됨.
1989 년에 GPL (GNU General public License)이라는 자유 소프트웨어 라이센스가 만들어짐.

+ 유닉스

1969 년에 AT&T 의 벨연구소에서 어셈블리어로 개발 -> 1971 년에 c 언어로 재개발. (최초의 고급 언어)
여러 os 를 하나로 제어하기 위해 개발됨. (unify 로부터 유닉스라는 이름이 나왔다)
이식성 높음
AT&T 의 상용 유닉스와 BSD 의 오픈소스 버전 유닉스로 나뉨

메모 포함[이10]: 기계어와 일대일대응 되는 저급 언어.



+ 단일보드컴퓨터

하나의 보드에 완전한 컴퓨터를 구성해놓은 것.
단일보드컴퓨터의 하드웨어 위에 os를 올려서 사용해야 함. (물론 다른 컴퓨터들도 마찬가지임)

2. 리눅스의 특징

오픈소스, 공개 소프트웨어.
유닉스와의 완벽한 호환성. (애초에 유닉스에서 출발한 os 임)
다중 사용자 지원. 서버용 os 로 많이 사용됨.
대부분의 국가, 기업, 연구, 산업에서 사용됨.
편리한 GUI 환경 보유. (GNOME 등)

3. 리눅스의 구조

리눅스는 파일을 효율적인 관리를 위해 디렉토리들로 이루어진 계층 구조를 가짐. (역 tree 구조)

리눅스는 커널과 쉘, 응용 프로그램으로 구성되어 있음.

메모 포함[이11]: 윈도우는 윈도우로 이루어져 있다.

리눅스가 이런 구조로 이루어진 이유는
리눅스가 다중이용 os이기 때문이다.
os를 최적화해야 해서 자원 절약을 위해
GUI 대신 CLI구조를 가지게 되었다.

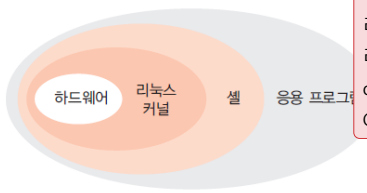


그림 1-7 리눅스의 구조

- 커널

OS의 핵심 영역
시스템의 모든 것을 통제함.
프로세스, 파일 시스템, 메모리, 장치 등 컴퓨터의 모든 자원을 관리함
커널 위에서 응용 프로그램이 수행됨.

- 셸

사용자와 커널 사이의 인터페이스. (명령어 해석기)
커널과 통신하고 응용프로그램을 제어.
여러 사람을 동시에 지원하기 위해 존재함.
리눅스는 배시 셸을 default로 사용.

- 응용 프로그램

여러 도구들.

+ 프롬프트

명령 입력을 기다리는 표시 (깜빡깜빡 하는 거)

프롬프트 다음에 명령을 입력하여 리눅스를 사용함.

4. 우분투 리눅스

데비안 계열의 리눅스로, 데비안 계열 중 가장 성공한 데스크톱 배포판.
unity를 사용하다가 GNOME(그놈)을 기본 데스크톱 환경 (GUI) 으로 사용하게 됨.
다양한 응용 프로그램 제공함. (배포판의 특성을 보면 당연한 것.)
장기 지원 버전(2년에 한 번)과 일반 버전(반 년에 한 번)을 구분하여 배포함.

- 셸 종료 명령어

exit 또는 ctrl + d : 터미널 종료

- 쉘의 명령행 편집법

backspace 또는 delete : 문자 지움
ctrl + w : 커서가 위치한 단어를 지우는데, 단어 맨 앞 칸부터 커서 앞 칸까지 지움. (띄여쓰기로 단어 구분됨)
ctrl + u : 명령행 처음부터 커서 앞 칸까지 다 지움.

메모 포함[이12]: $ 오른쪽에 작성하고 있는 명령을 편집하는 방법을 말함.

- 쉘 명령의 구조

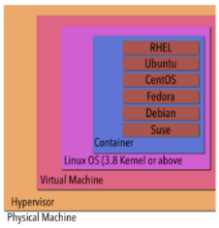
"명령 옵션 인자" 형태.
옵션 : 세부적인 내용 결정. - 또는 --로 시작함. 명령에 따라 옵션이 다르므로 해당 명령의 사용법 참조
인자 : 명령으로 전달되는 값. 주로 파일, 디렉토리 이름이 들어감. 명령에 따라 필요한 인자가 다르므로 해당 명령의 사용법 참조.

3. 가상머신(virtual machine)

1. 가상머신

os에 가상의 머신(시스템, PC)을 만들어 이곳에 다른 os를 설치할 수 있게 해주는 응용 프로그램.

여러가지 os는 가상머신화해서 Hypervisor라는 이름의 것 위에 올릴 수 있음.



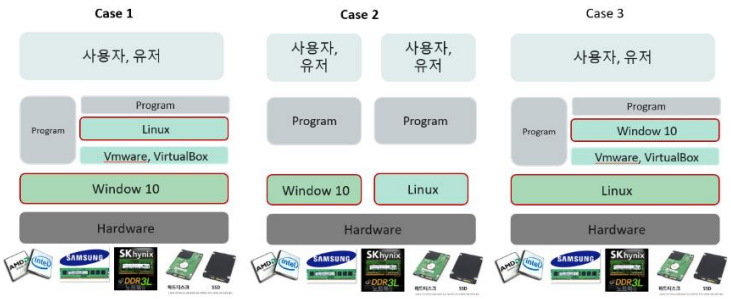
호스트 os: 실제 하드웨어에 설치된 os. (host: 주인)
게스트 os: 가상머신에 설치된 os.

2. 가상머신과 우분투 리눅스의 구축

여러 가상머신이 있지만 이 수업에서는 VMware를 사용함.
VMware는 유료 프로그램이지만, VMware Workstation Player라는 프로그램을 이용할 수 있음.
(VMware 홈페이지에서 다운받을 수 있음.)
가상머신이 설치되면, 우분투 홈페이지에서 이미지 파일을 다운로드.
설치된 우분투 파일을 가상머신에서 불러와서 실행시키면 구축 완료.

+ 하나의 pc 에서 리눅스와 윈도우를 같이 쓰는 방법들

(case 2 에서는 하드디스크를 2 개 사용하거나 하나의 하드디스크를 나누어 사용함. 부팅 시 os 선택.)



메모 포함[이13]: 가상의 PC로 이해하면 됨.

메모 포함[이14]: 보조기억장치.
정보를 저장할 수 있는 장치.

2. 리눅스 기본 명령

1. 명령어들

<파일시스템 관련 명령어들>

- pwd (present working directoy)

작업 디렉토리의 절대 경로 출력 명령어

```
pwd
```

- cd (change directory)

특정 위치로 작업 디렉토리를 이동하는 명령어

```
cd <path>
```

<path>에는 절대경로나 상대경로를 쓸 수 있음.

경로에 아무것도 안 적으면 사용자의 홈 디렉토리로 감

작업 디렉토리 내에 위치해 있는 디렉토리라면 경로 칸에 디렉토리명만 적어도 해당 디렉토리로 이동함.

+ ls -F로 확인할 수 있는 파일 종류

/로 끝나면 디렉토리.
@로 끝나면 심볼릭 링크 파일.
아무것도 없으면 일반 파일.
*로 끝나면 실행 파일. (실행 파일도 일반 파일이기는 함.)

메모 포함[이15]: + 파일명, 경로명 등은 항상 명령의 맨 마지막에 명시하도록 순서가 정해져 있다.

단, ln, find, mount 등은 제외.
ln는 <원본> <링크파일> 형식임.
find에서는 경로를 맨 앞에 써줘야 함.
mount는 마운트 포인트까지 작성해야 함.

메모 포함[이16]: cd 명령어로 작업 디렉토리를 이동했으면,
pwd 명령어로 확인하는 습관을 가지자.

- ls (소문자 엘) (list)

ls <option> <file>

작업 디렉토리의 파일목록 출력 명령어

파일을 적지 않으면 현재 디렉토리의 파일을 보여줌.

옵션들.

-a : 모든 파일목록 출력. (. 으로 시작하는 파일과 디렉토리도 보여줌.)

-A : . 와 .. 를 제외한 모든 파일목록 출력. (.과 ..로 시작하는 파일이 아니라 .과 .. 자체를 제외함)

-d : 디렉토리 자체를 지정. 디렉토리 자체의 정보를 출력할 때 사용.

-i (inode) : inode 번호 출력.

-R : 하위 디렉터리에 있는 파일까지 보기

-F : 파일과 함께 파일의 종류 출력. (/: 디렉토리, @: 심볼릭 링크, *: 실행파일)

-L : 심벌릭 링크 파일이 있는 경우 원본 파일의 정보 출력.

-l (소문자 엘) : 파일의 상세정보 출력. 파일 종류, 권한, 파일 소유자 등 확인 가능

메모 포함[이17]: dir, vdir등을 사용할 수도 있다.
비슷한 기능을 한다.

dir: ls
vdir: ls -l

메모 포함[이18]: 주로 설정 파일로 사용되는 파일임.

메모 포함[이19]: 어떤 디렉토리를 지정해서 ls명령어를 사용할 때, 해당 디렉토리 안에 있는 파일에 대한 상세 정보를 출력함. 그 디렉토리 자체의 정보를 보려면 d옵션을 넣어줘야 함.

메모 포함[이20]: 모든 파일은 inode 번호를 가지고 있음.
즉, 리눅스의 모든 것은 inode 번호를 가지고 있음.

+ ls 명령어 -l 옵션을 사용하여 출력한 상세정보의 의미

표 2-2 파일의 상세 정보

필드 번호	필드 값	의미
1	d	파일 종류
2	rwxr-xr-x	파일 접근 권한. 파일 소유자, 그룹, 기타 사용자가 파일을 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	4096	파일 크기(바이트 단위)
7	11월 8 23:24	파일이 마지막으로 수정된 시간
8	공개	파일명

표 2-3 파일의 종류

문자	파일 유형
-	일반(정규) 파일
d	디렉터리 파일
l	심벌릭 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)

(ex. -rwer--r-- 2 jhun jhun 4003 Mar 6 18:28 test1.c)

+ ls 명령어 -l 옵션 사용 시의 파일 접근 권한

사용자 유형별로 읽기(read), 쓰기(write), 실행(execute) 가능 여부를 나타냄.

사용자는 파일 소유자, 그룹 사용자, 기타 이용자로 구분.

다중 사용자 OS에서 파일 보호를 위해 사용.

ls -l 명령어의 두 번째 필드에서 확인할 수 있음
(2~10번째 글자들.)

문자 위치	의미
첫 번째	파일의 종류를 나타냄; d : 디렉터리, - : 일반 파일, l : 심볼릭 링크, p : 이름있는 파이프, 등
2~4번째	파일 소유자의 권한; 3개의 문자는 각각 읽기, 쓰기, 실행 권한을 나타내고, 각 자리에 r, w, x가 있으면 그 권한이 있음을, -가 있으면 권한이 없음을 뜻함
5~7번째	그룹 사용자의 권한; 3개의 문자는 각각 읽기, 쓰기, 실행 권한을 나타내고, 각 자리에 r, w, x가 있으면 그 권한이 있음을, -가 있으면 권한이 없음을 뜻함
8~10번째	기타 사용자의 권한; 3개의 문자는 각각 읽기, 쓰기, 실행 권한을 나타내고, 각 자리에 r, w, x가 있으면 그 권한이 있음을, -가 있으면 권한이 없음을 뜻함

<파일, 디렉토리 관련 명령어들>

- mkdir (make directory)

mkdir <option> <name>

디렉토리 생성 명령어

디렉토리명은 절대경로 또는 상대경로로 작성함.

경로가 아닌 디렉토리명만을 적으면 작업 디렉토리에 디렉토리를 생성함.

디렉토리를 한 번에 여러 개 만들려면 디렉토리 이름을 여려 개 적으면 됨.
공백 문자로 구분하면서 작성.
(ex. mkdir -p name1 name2 name3)

옵션들.
-m <mode> : 접근 권한을 설정하며 디렉토리 생성.
-p : 하위 디렉토리를 생성할 때, 중간 단계의 디렉토리가 없으면 자동으로 생성하면서 작업 수행.
(중간 디렉토리가 없는데 이 옵션을 사용하지 않는다면 디렉토리가 생성되지 않음.)

메모 포함[이21]: 숫자 모드로만 되는지 확인

- rmdir (remove directory)

rmdir <option> <directory>

비어있는 디렉토리 삭제 명령어

디렉토리를 한 번에 여러 개 삭제하려면 디렉토리 이름을 여러 개 적으면 됨.

옵션들.
-p : 해당 디렉토리를 삭제하며 그 디렉토리의 상위 디렉토리가 빈 디렉토리인 경우 상위 디렉토리도 삭제함.

메모 포함[이22]: 디렉토리가 비어있지 않은 경우에는 rm -r 을 사용해야 한다.

- touch

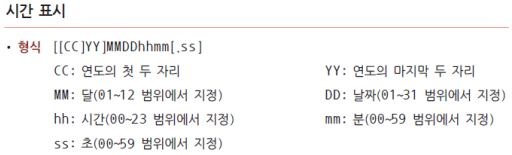
touch <option> <file>

빈 파일 생성, 파일 마지막 접근시간 또는 수정시간 수정 명령어

작업 디렉토리 내 파일의 수정 시간 기록 변경.
옵션을 작성하지 않으면 수정 시간을 현재 시간으로 변경됨.

해당 파일이 작업 디렉토리에 없을 경우 같은 이름의 크기가 0바이트인 파일 생성.

옵션들
-a : 접근 시간만 변경
-m : 수정 시간만 변경
-t <시간> : 명시한 시간으로 시간 수정.



- cp (copy) (~를 ~로)

```
cp <option> <source_file> <dest_file>
```

파일 복사 명령어 (이동 아님!)
첫 번째 인자에는 원본 파일을, 두 번째 인자에는 목적지 위치나 파일명 작성.

메모 포함[이23]: source. 출처.
destination. 목적지.

1) 첫번째 인자가 파일인 경우

두번째 인자가 파일인 경우 첫번째 인자의 파일의 이름을 두번째 인자로 바꾸어서 작업 디렉토리에 복사함.
두번째 인자가 디렉토리인 경우 첫번째 인자의 파일이 두번째 디렉토리 안에 복사함.
두번째 인자가 "디렉토리/파일명"인 경우 첫번째 인자의 파일의 이름이 두번째 인자의 파일명으로 바꿔서 복사함.

2) 첫번째 인자가 디렉토리인 경우

두번째 인자의 디렉토리 안에 첫번째 인자의 디렉토리가 복사됨.
두 번째 인자로 작성한 디렉토리가 없을 경우 해당 이름의 디렉토리를 생성.
디렉토리가 복사되면 원본 디렉토리(첫 번째 인자에 있던 거) 아래에 있던 모든 내용도 함께 복사.

인자가 3개 이상인 경우, 마지막 인자가 destination 임. 고로 마지막 인자는 반드시 디렉토리여야 함.
인자가 3개 이상인 경우에도 디렉토리와 파일을 혼용할 수 있음. -r 만 잘 써주면 됨.

option 종류

-r : 디렉토리를 복사할 때 사용.
-i : dest_file 이 이미 존재하는 파일이면 덮어쓸 것인지 물어봄. (기본적으로는 덮어씀)

메모 포함[이24]: 여기서의 -r은 디렉토리를 의미하는 듯.

메모 포함[이25]: i 옵션은 대체로 정말 실행할 것인지 확인 차 물어보는 기능으로 쓰인다.

- rm (remove)

```
rm <option> <file>
```

일반 파일, 디렉토리 삭제 명령어

메모 포함[이26]: 리눅스에서는 휴지통의 개념이 없다. 삭제 시 그냥 끝.

여러 개의 인자를 작성할 경우 그 인자 모두를 삭제함.

option 종류

-r : 디렉토리 삭제. 그 아래 파일들도 삭제됨.
-i : 파일을 정말 삭제할 것인지 물어봄.

메모 포함[이27]: -r을 작성해도 일반 파일을 삭제할 수 있음.
즉, 일반 파일과 디렉토리를 한꺼번에 삭제할 때에도 그냥 -r만 붙여주면 됨.

- mv (move) (~를 ~로)

```
mv <option> <source_file> <dest_file>
```

파일, 디렉토리 이동. 이름 변경 명령어

첫 번째 인자에는 원본 파일을, 두 번째 인자에는 목적지 위치나 파일명 작성.

메모 포함[이28]: 작업 디렉토리에 목적지 디렉토리가 있다면
그냥 이름만 적어도 된다.
없다면 경로 명시해 줘야 함.

1) 파일을 파일로 이동하기 (파일명 바꾸기)
두 번째 인자에 작성한 파일명으로 파일명이 바뀜.
작업 디렉토리 내에서 작업이 수행됨.
두 번째 인자에 작성한 파일명을 가진 파일이 이미 존재할 경우 덮어씀.

2) 파일 하나를 다른 디렉토리로 이동하기
두 번째 인자로 디렉토리를 쓸 경우 첫 번째 인자의 파일이 그 디렉토리로 이동함.
두 번째 인자를 디렉토리/파일명 꼴로 작성할 경우 파일이 해당 디렉토리로 이동하며 해당 파일명으로 바뀜.
해당 디렉토리의 쓰기 권한이 없을 경우 이동 불가.

3) 파일 여러 개를 다른 디렉토리로 이동하기
3개 이상의 인자를 사용할 수 있음.
앞 인자들에 작성된 파일들이 맨 마지막 인자에 작성된 디렉토리로 이동함.
마지막 인자에는 반드시 디렉토리가 들어가야 함.

4) 디렉토리를 디렉토리로 이동하기
인자에 디렉토리를 넣으면 디렉토리를 이동시킴.
두 번째 인자가 원래 있던 디렉토리인 경우에는 그 디렉토리 밑으로 첫 번째 인자에 있는 디렉토리를 이동.
두 번째 인자가 원래 있던 디렉토리가 아닌 경우에는 그 디렉토리명으로 첫 번째 인자에 있는 디렉토리의 이름을 바꿈.

옵션들
-r : 디렉토리 이동. 그 아래 파일들도 이동됨.
-i : dest_file 이 이미 존재하면 덮어쓸 것인지 물어봄.

메모 포함[이29]: -r을 작성해도 일반 파일을 삭제할 수 있음.
즉, 일반 파일과 디렉토리를 한꺼번에 삭제할 때에도 그냥 -r만 붙여주면 됨.

- file

```
file <file>
```

지정한 파일의 종류를 알려주는 명령어.

- sort

```
sort <option> <file>
```

텍스트 파일을 행 단위로 오름차순 정렬해서 출력해주는 명령어

-r : 내림차순으로 정렬

<파일 출력 명령어들>

- cat (catenate)

cat <option> <file>

파일 내용 출력 명령어.

cat 만 작성한다면 셸에 문자열을 입력하여 출력시킬 수 있음
(이 기능은 리다이렉션하지 않고는 어디에 쓰는 지 모르겠음)

옵션들.

-n : 행 번호를 붙여서 출력. 빈 행도 번호를 매김. (number)
-b : 행 번호를 붙여서 출력. 빈 행은 번호를 매기지 않음

cat <option> <file1> > <file2>

> 기호 : 출력 다시 지정.
file1의 내용을 file2에서 출력함. 즉, file2에 file1의 내용을 작성함.
file1 자리에는 여러 개의 인자(파일)가 들어가면 여러 개의 파일을 file2에서 출력함.

메모 포함[이30]: file2에 들어간 파일이 없을 경우 file1의 내용을 복사해서 출력한다.
file2에 들어간 파일이 있을 경우에는..?

file1을 작성하지 않고 > <file2>만 작성한다면 해당 이름을 가진 파일이 생성됨.
이때 해당 파일에 들어갈 내용을 작성하게 됨. <control>+d 키로 빠져나올 수 있음.

메모 포함[이31]: 원래 있는 파일이면?

- more

more <option> <file>

파일 내용을 한 화면 단위로 출력하는 명령어.
출력이 시작된 부분부터 위아래로 넘기며 읽을 수 있음. 특정 행에서 출력이 시작되었으면, 그 위로는 이동할 수 없음.
space bar -> 다음으로
b -> 전으로.

출력되지 않은 내용이 있으면 화면 하단에 "--More--(0%)" 형식으로 표시됨.

옵션들.

| pg : 명령어 뒤에 작성하여 페이지를 넘기는 형태로 파일 내용을 확인할 수도 있음. n 으로 넘김.
+<행 번호> : 출력을 시작할 행 번호를 지정함.

- less (리눅스+)

less <file>

파일 내용을 한 화면 단위로 출력하는 명령어.
처음부터 출력되고, 위아래로 넘기며 읽을 수 있음.

표 2-4 less 명령에서 사용하는 키와 동작

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

메모 포함[이32]: cat -> 단순 출력. 띄우기만 함.
more -> 아래로만 움직이며 확인 가능.
less -> 위아래 모두 움직이며 확인 가능.

- head

head <option><file>

파일 내용의 앞부분부터 행을 출력하는 명령어
default 값은 10개의 행 출력임.

옵션들.
-c <number> : 파일 앞부분부터 해당 숫자만큼의 byte 까지 출력.
-n <number> : 파일 앞부분부터 해당 숫자만큼의 행까지 출력. (number)

- tail

tail <option> <file>

파일 내용의 뒷부분부터 행을 출력하는 명령어.
default 값은 10개의 행 출력임.

옵션들.
+<행 번호> : 지정한 행부터 파일 내용 끝까지 출력.
-<숫자> : 화면에 출력할 행의 수를 지정함.
-f : 파일 출력을 종료하지 않고 반복적으로 출력. ctrl c 키로 출력을 종료함.
-c <number> : 파일 뒷부분부터 해당 숫자만큼의 byte 까지 출력.
-n <number> : 파일 뒷부분부터 해당 숫자만큼의 행까지 출력. (number)

메모 포함[이33]: 그리고 이거 옵션 섞어쓰는 건 어떻게 하는지 확인하기.

<파일 권한 관련 명령어들>

- chmod (change mode)

```
chmod <option> <mode> <file>
```

파일 접근권한 변경 명령어. (작성한 모드로 변경.)

mode 에는 기호 모드와 숫자 모드가 있음

-R : 하위 디렉토리까지 변경

1) 기호 모드

오른쪽 표의 기호를 사용해서 작성.

"사용자+설정+권한"의 형식으로, 필요한 만큼 작성.

사용자끼리, 권한끼리 적는 순서는 상관이 없음.

쉼표로 이어서 적을 때 띄어쓰기 하면 오류남.

(ex. chmod g-r+xw,o+w test)

기호 분류	기호	의미
사용자	u	파일 소유자
	g	그룹 사용자
	o	기타 사용자
	a	모든 사용자
권한	r	읽기
	w	쓰기
	x	실행
설정	+	권한 추가
	-	권한 삭제
	=	권한 배정

2) 숫자 모드

각 사용자 별로 3가지 권한의 유무를 이진수로 나타내고 8진수로 바꾸어 작성. (결국 8진수 3글자로 표현됨.)

권한 순서는 r w x 이고, 권한이 있다면 1, 없다면 0으로 나타냄.

이진수 3자리는 3비트 정보이므로, 8진수 한 자리의 크기와 정확히 맞아떨어짐.

(ex. chmod 754 test)

- umask

파일, 디렉토리 생성 시에 적용되는 기본 접근권한 마스크 설정 명령어
파일, 디렉토리 생성 시에 부여하지 않을 권한을 지정하는 것. 권한 생성 X, 최대 권한에서 막을 권한을 지정.

일반파일 최대 권한: 666, 디렉토리 최대 권한: 777

umask <option> <마스크 값>

최대권한에서 마스크만큼 뺀 값이 기본 접근권한이 됨.
8 진수(숫자)로 나타낸 접근권한과 마스크 값을 서로 빼서 기본 접근권한을 구할 수도 있고,
접근권한과 마스크 값을 9 자리로 나타내서 마스크 값이 존재하는 곳의 접근권한을 지워서 구할 수도 있음.

마스크 값은 가릴 접근권한을 숫자로 표기한 것. 명령 입력 시에는 뒤의 3 글자만 작성해 줌.

umask 만 치면 현재 설정되어 있는 mask 가 뜸.

리눅스에 다시 로그인하면 default 값으로 초기화됨.

옵션들

-S : 마스크 값을 문자로 출력함. 여기서 출력되는 것들은 허용되는 기본 접근권한임. (string)

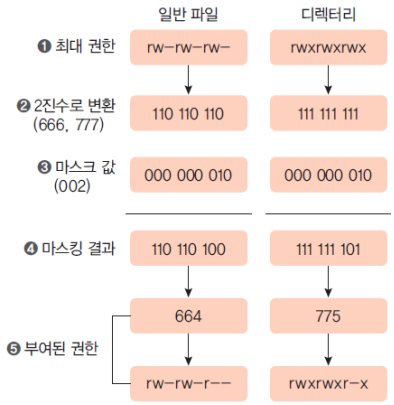


그림 5-8 마스크 값을 적용하는 과정

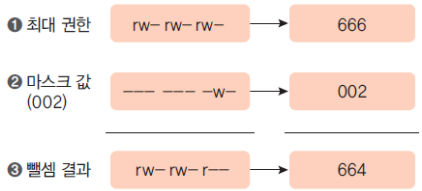


그림 5-9 마스크 값에 뺄셈을 적용하는 과정

- groups

사용자가 속한 그룹을 출력하는 명령어.

groups <사용자명>

groups 만 입력할 경우 명령을 사용하는 사용자가 속한 그룹을 출력함.

+ mask 값은 일반파일과 디렉토리에 한 번에 적용되는데, 둘의 최대 권한이 다르므로 결과에 유의해야 함.

메모 포함[이34]: 마스크로 가린다는 의미.
마스킹한다는 것.

메모 포함[이35]: 근데 이렇게 빼는 경우에는 의미적 오류가 나는 것 같은데?
마스크 값을 333이라 하면, 그냥 뺐을 때 일반파일의 기본 접근권한은 333으로 나옴.
그런데 9글자로 나타내서 확인하면 444임.
실제로 파일을 만들어 봐도 444임.

메모 포함[이36]: 4글자를 작성하면 작성한 것 중에서 뒤 3글자만 마스크 값의 뒤 3글자로 적용되고 맨 앞 글자는 반영되지 않는다.

메모 포함[이37]: umask만 입력하면 접근권한을 가릴 마스크 값이 출력되지만,
-S 옵션까지 입력하면 마스크 값이 반영된 기본 접근권한이 출력됨. 777에 반영된 것으로.

<파일 비교 명령어들>

- cmp (compare)

두 파일을 바이트 단위로 비교하는 명령어.
처음으로 다른 부분을 출력함.

```
cmp <option> <file1> <file2>
```

- diff (difference)

두 파일을 행 단위로 비교하는 명령어.
다른 행을 모두 찾아서 출력함.

```
diff <option> <file1> <file2>
```

<검색 관련 명령어들>

- grep

```
grep <option> <pattern> <file>
```

특정 일반 파일 내에서, 명시한 문자열 패턴을 찾는 명령어.

* (별표기호) 특수문자를 이용해서 여러 파일에서 찾을 수도 있음.

옵션들.
-i : 대문자, 소문자를 구분하지 않고 검색.
-l : 해당 패턴이 포함되어 있는 파일들의 파일명 출력.
-n : 행 번호와 함께 출력.
-c : pattern 과 일치하는 행의 개수 출력.
-v : 검색 조건을 만족하지 않는 행을 출력
-w : 패턴과 완전히 일치하는 단어만을 출력. (with 이면 with 만 잡고 without 은 안 잡음.)

- find

```
find <path> <option> <expression>
```

파일 및 디렉토리 검색 명령어

path 에 디렉토리명을 적으면 해당 디렉토리와 그 하위 디렉토리에서 검색함.

한 번에 여러가지 expression 을 쓸 수 있음.
expression 을 작성하지 않으면 -print 가 자동으로 수행됨.

옵션들.
-name <file> : 해당 파일명으로 검색. -> 문자열이 아니라 완전한 파일명을 작성해야 찾을 수 있음.
-type <file_type> : 파일 유형으로 검색.
-user <user_ID> : 해당 사용자가 소유한 파일 검색.
-perm <mode> : 해당 사용권한 유형을 가진 파일 검색. 숫자로 검색해야 함 (permission)

expression(동작)들 (띄어쓰기 반드시 지켜줘야 함)
-exec <command> {} ₩; : 검색된 파일에 해당 명령어 실행. (execute) (세미콜론 있는 거 주의)
-ok <command> {} ₩; : 사용자의 확인을 받아서 검색된 파일에 해당 명령어 실행. (ok 받으면 실행)
-print : 검색된 파일의 절대 경로명 출력. (기본 동작)x`
-ls : 검색된 결과를 긴 목록 형식으로 출력. inode 번호와 몇 가지 세부정보를 출력함.

+ 왜인지는 모르겠는데, -name 으로 검색할 때 특수문자를 사용하려면 큰따옴표 또는 작은따옴표로 묶어줘야 됨.

(ex. find . -name "[0-9]*" -exec ls -ld {} ₩;)

메모 포함[이38]: find 명령으로 찾은 파일의 절대 경로가
{}가 있는 위치에 대입되어 명령이 수행된다.

메모 포함[이39]: ₩ 이건 역슬래시.

메모 포함[이40]: 이런 공백 하나하나 잘 지켜줘야 한다..!

메모 포함[이41]: find로 찾은 것에 대한 작업은 웬만하면
-exec expression으로 처리하자.
| 이거 쓰는 건 확실하지 않음.

<셸 관련 명령어>

- echo

배시 셸의 문장 출력 명령어

```
echo <option> <text>
```

옵션들

-n : 인자를 출력한 후 개행 문자를 삽입하지 않음

- printf

배시 셸에서 c 언어이 표준 출력 함수인 printf 를 구현한 명령어.

단순히 인수만 작성한다면 그 인수를 출력함.
printf 함수의 형식에 따라 ' "문자열", 인수 2 인수 3 '과 이스케이프 시퀀스, 변환명세 등을 사용할 수 있음.
첫 번째 인수 바로 뒤에는 쉼표를 적어도 되지만, 두 번째 인수 뒤부터는 쉼표를 작성하면 경고 메시지 나옴.
(ex. printf "%d + %d = %d", 4 5 9 와 같이 입력할 수 있음.)
(연산까지 다 해주는 것 같지는 않음.)

```
printf <option> <인수>
printf <option> <printf 함수의 형식>
```

- export

셸 변수를 환경변수로 바꾸는 명령어

```
export <option> <셸 변수>
```

셸 변수를 정의하면서 바로 환경변수로 바꿀 수도 있음 (ex. export SOME=test)

옵션들.

-n : 환경 변수를 셸 변수로 바꿈. (셸 변수 자리에 환경변수를 넣으면 됨.)

- unset

지정한 환경설정 관련 변수를 해제하는 명령어.

```
unset <변수>
```

- chsh (change shell)

해당 사용자의 로그인 셸을 바꾸는 명령어.
해당 셸이 설치되어 있어야 함.

```
chsh <option> <user_name>
```

옵션들

-s <shell> : 지정한 셸(절대경로)으로 로그인 셸을 바꿈. (shell)

<프로세스 관련 명령어들>

- ps (process)

현재 실행 중인 프로세스들을 출력하는 명령어.

ps <option>

메모 포함[이42]: 유닉스, BSD 중 한 조합의 옵션을 주고
같은 기능을 하도록 다른 유형의 옵션을 사용해서 쓰라고 할 수도?

여기의 옵션들은 암기하는 것이 좋겠다.

옵션들.

-l (엘) : 프로세스 소유자, 부모 프로세스, 프로세스 그룹 등의 ID 출력

-a : 모든 사용자의 프로세스 출력

<유닉스>

-e : 시스템에서 실행 중인 모든 프로세스의 정보 출력

-f : 프로세스의 자세한 정보 출력

-u <UID> : 특정 사용자에 대한 모든 프로세스 정보 출력

-p <PID> : 해당 PID 를 가지는 프로세스의 정보를 출력

표 6-1 ps -f의 출력 정보

항목	의미	항목	의미
UID	프로세스를 실행한 사용자 ID	STIME	프로세스의 시작 날짜나 시간
PID	프로세스 번호	TTY	프로세스가 실행된 터미널의 종류와 번호
PPID	부모 프로세스 번호	TIME	프로세스 실행 시간
C	CPU 사용량(% 값)	CMD	실행되고 있는 프로그램 이름(명령)

<BSD>

a : 터미널에서 실행한 프로세스 정보 출력

u : 상세 프로세스 정보 출력

x : 시스템에서 실행 중인 모든 프로세스의 정보를 출력

표 6-2 STAT에 사용되는 문자의 의미

문자	의미	비고
R	실행 중(running)	
S	인터럽트가 가능한 대기(sleep) 상태	
T	작업 제어에 의해 정지됨(stopped) 상태	
Z	좀비 프로세스(defunct)	
STIME	프로세스의 시작 날짜나 시간	
s	세션 리더 프로세스	BSD 형식
+	포그라운드 프로세스 그룹	
l(소문자 L)	멀티스레드	

표 6-3 ps au의 출력 정보

항목	의미	항목	의미
USER	사용자 계정 이름	VSZ	사용 중인 가상 메모리의 크기(KB)
%CPU	퍼센트로 표시한 CPU 사용량	RSS	사용 중인 물리적 메모리의 크기(KB)
%MEM	퍼센트로 표시한 물리적 메모리 사용량	START	프로세스 시작 시간

<GNU>

--pid <PID> : 특정 PID 의 정보 출력

- pgrep (process grep)

```
pgrep <option> <pattern>
```

지정한 패턴과 일치하는 프로세스의 PID 를 출력해주는 명령어

-x : 패턴과 정확히 일치하는 프로세스의 정보를 출력
-n : 패턴을 포함하고 있는 가장 최근 프로세스의 정보를 출력
-u <사용자명> : 특정 사용자에 대한 모든 프로세스 정보를 출력
-l (엘) : 프로세스 이름을 PID 와 함께 출력
-t <term> : 특정 단말기와 관련된 프로세스의 정보를 출력

- kill

```
kill -<signal> <process>
```

프로세스에 signal 을 전송하는 명령어

시그널을 지정하지 않은 경우, 15번 시그널로 간주됨.

메모 포함[이43]: 관련된 파일을 정리하고 프로세스를 종료하는 시그널.

프로세스는 PID 또는 작업 번호를 작성해서 명시할 수 있음.
작업 번호는 %<작업 번호>를 작성하면 됨.

옵션들
-l (엘) : 사용 가능한 signal 출력. 'kill -l'을 쓰면 됨.

+ 시그널 (signal)

프로세스에 무언가 발생했음을 알리는 메시지.
프로세스는 각 시그널을 받았을 때 어떻게 처리할 것인지 동작이 지정되어 있음.
시그널은 미리 정의된 번호로 구분됨.
시그널을 받은 프로세스는 기본적으로 종료됨.

표 6-4 주요 시그널

시그널	번호	기본 처리	의미
SIGHUP	1	종료	터미널과의 연결이 끊어졌을 때 발생한다.
SIGINT	2	종료	인터럽트로 사용자가 Ctrl+c를 입력하면 발생한다.
SIGQUIT	3	종료, 코어덤프	종료 신호로 사용자가 Ctrl+\을 입력하면 발생한다.
SIGKILL	9	종료	이 시그널을 받은 프로세스는 무시할 수 없으며 강제로 종료된다.
SIGALRM	14	종료	알람에 의해 발생한다.
SIGTERM	15	종료	kill 명령이 보내는 기본 시그널이다.

다른 시그널들과는 달리, 9번 시그널은 반드시 대상을 죽임.

메모 포함[이44]: 프로세스 종료가 기본이지만 시그널을 무시하거나 다른 동작으로 하도록 지정되어 있다면
종료되지 않을 수 있다.

메모 포함[이45]: 단, 대상이 좀비 프로세스인 경우 종료되지 않을 수 있음.

- pkill (process kill)

pkill <name>

PID 가 아니라 프로세스의 이름(CMD)으로 프로세스를 찾아 종료하는 명령어.
같은 이름이 여러 개 검색될 경우 한 번에 모두 종료함.
사용자가 소유하고 있는 프로세스여야 함.

- killall (pkill 과 유사함)

killall <name>

PID 가 아니라 프로세스 이름(CMD)으로 프로세스를 찾아 종료하는 명령어.
같은 이름이 여러 개 검색될 경우 한 번에 모두 종료함.
사용자가 소유하고 있는 프로세스여야 함.

- top

top

현재 실행 중인 프로세스에 대한 실시간 정보를 출력하는 명령어
내부적으로 사용하는 명령도 존재함.

메모 포함[이46]: GNOME에 있는 '시스템 감지' 기능을
이용해서 확인할 수도 있다.

표 6-5 top 명령의 출력 정보

항목	의미	항목	의미
PID	프로세스 ID	SHR	프로세스가 사용하는 공유 메모리의 크기
USER	사용자 계정	%CPU	퍼센트로 표시한 CPU 사용량
PR	우선순위	%MEM	퍼센트로 표시한 메모리 사용량
NI	Nice 값	TIME+	CPU 누적 이용 시간
VIRT	프로세스가 사용하는 가상 메모리의 크기	COMMAND	명령 이름
RES	프로세스가 사용하는 메모리의 크기		

표 6-6 top 명령의 내부 명령

내부 명령	기능
Enter Space Bar	화면을 즉시 다시 출력한다.
h, ?	도움말 화면을 출력한다.
k	프로세스를 종료한다. 종료할 프로세스의 PID를 물어본다.
n	출력하는 프로세스의 개수를 바꾼다.
p	CPU 사용량에 따라 정렬하여 출력한다.
q	top 명령을 종료한다.
M	사용하는 메모리의 크기에 따라 정렬하여 출력한다.
u	사용자에 따라 정렬하여 출력한다.

- jobs

백그라운드 작업 목록을 출력하는 명령어

jobs %<작업 번호>

배시 셸의 내부 명령임.

작업 번호 작성 형식.

%<번호> : 해당 번호의 작업 정보 출력

%+, %% : 작업 순서가 +인 작업 정보 출력

%- : 작업 순서가 -인 작업 정보 출력

메모 포함[이47]: 실제로는 작업 정지된 포그라운드 프로세스도 작업으로 출력된다.

표 6-7 jobs 명령의 출력 정보

항목	출력 예	의미
작업 번호	[1]	작업 번호로서 백그라운드로 실행할 때마다 순차적으로 증가한다([1], [2], [3]…)
작업 순서	+	작업 순서를 표시한다. • +: 가장 최근에 접근한 작업 • -: + 작업보다 바로 전에 접근한 작업 • 공백: 그 외의 작업
상태	실행중	작업 상태를 표시한다. • 실행중: 현재 실행 중이다. • 완료: 작업이 정상적으로 종료되었다. • 종료됨: 작업이 비정상적으로 종료되었다. • 정지됨: 작업이 잠시 중단되었다.
명령	sleep 100 &	백그라운드로 실행 중인 명령이다.

- nohup

로그아웃한 후에도 백그라운드 작업을 계속 진행하게 하는 명령어

nohup <명령 or 프로그램> &

프로그램명을 작성한 경우 그 프로그램은 로그아웃 후에도 돌아감.

nohup 명령에 작성하는 명령은 반드시 백그라운드로 실행해야 함. (그래서 &를 적어둠)

명령의 출력을 별도로 디라이렉션하지 않으면 실행 결과와 오류 메시지가 현재 디렉토리의 nohup.out 파일로 자동 저장됨.

메모 포함[이48]: 홈 디렉토리가 아니라 현재 디렉토리에 생성된다.

- at

at <option> <시간>

예약한 명령을 정해진 시간에 한 번 수행하게 하는 명령어

at 으로 시간을 입력하면 프롬프트가 at>으로 바뀌고 설정할 명령을 작성하게 됨.
명령 작성이 끝났으면 ctrl + d 로 빠져나올 수 있음.

at 으로 설정한 작업의 결과는 주로 리다이렉션하는데, 파일로 출력을 전환하지 않은 경우 작업 결과가 메일로 전달됨. (이 경우 시스템에 메일이 설정되어 있어야 함)

at 명령으로 생성된 작업 파일은 /var/spool/at 에 저장됨.
예약된 명령이 시행되면 파일이 자동으로 삭제됨.

시간 지정 형식

at 4pm + 3 days : 지금부터 3일 후 오후 4시에 작업을 수행한다.
at 10am Jul 31 : 7월 31일 오전 10시에 작업을 수행한다.
at 1am tomorrow : 내일 오전 1시에 작업을 수행한다.
at 10:00am today : 오늘 오전 10시에 작업을 수행한다.

(HH:MM 또는 HHMM 등의 형식으로 작성 가능)

at 명령어 제한 -> /etc/at.allow(허가)와 /etc/at.deny(금지) 파일 이용
두 파일에는 모두 한 행에 하나의 사용자명을 적게 되어있음.

규칙 1. /etc/at.allow 파일이 존재하면 이 파일에 있는 사용자만 at 명령어를 사용할 수 있음. /etc/at.deny 파일은 무시됨.
규칙 2. /etc/at.allow 파일이 존재하지 않으면 /etc/at.deny 파일에 있는 사용자를 제외한 모든 사용자는 at 명령어를 사용할 수 있음. (/etc/at.deny 파일에 아무것도 적혀 있지 않다면 모든 사용자가 사용 가능.)
규칙 3. 두 파일 모두 존재하지 않는다면 root 만 at 명령어를 사용할 수 있음.

메모 포함[이49]: 초기 설정은 이 경우로 되어 있다. allow파일이 없고 deny파일이 비어 있는 상태.

옵션들.

-l (엘) : 현재 실행 대기 중이 명령의 전체 목록을 출력 (atq 명령과 동일)
-r <번호> : 현재 대기 중인 명령들 중에 해당 번호를 가진 명령을 삭제 (atrm 명령과 동일)
-d : at 작업을 삭제함. (atrm 명령과 동일)
-m : 작업이 완료되면 사용자에게 메일로 알려줌
-f <file> : 표준 입력 대신 실행할 명령을 파일로 저장함.

메모 포함[이50]: atq
현재 사용자의 at 명령으로 설정한 작업 목록을 보여주는 명령어. 슈퍼 유저일 경우 모든 사용자의 작업 목록을 보여줌.

메모 포함[이51]: atrm
지정된 작업 번호의 작업을 삭제하는 명령어.

- crontab

작업을 정해진 시간마다 반복해서 실행시켜주는 crontab 파일을 관리하는 명령어.

crontab 파일은 사용자별로 생성되고, 이 파일에 반복 실행할 작업이 저장됨.
이 파일에는 한 행의 하나의 작업을 설정함.

```
crontab -u <user ID> <option> <file>
```

crontab 파일의 형식
한 행이 다음과 같이 구성됨.
각 항목은 공백문자로 구분함.

분(0~59)	시(0~23)	일(1~31)	월(1~12)	요일(0~6)	작업 내용

(* 특수문자를 이용해서 여러 시각을 지정할 수 있음)
(요일은 일요일이 0, 토요일이 6)
(- 와 /를 이용할 수 있음. 1 시부터 23 시까지 2 시간 간격 -> 1-23/2 이렇게 작성하면 됨. */2 -> 2 분마다 실행.)

crontab 명령어 제한 -> /etc/cron.allow(허가)와 /etc/cron.deny(금지) 파일 이용
cron.deny 파일은 기본적으로 있지만 cron.allow 파일은 기본적으로 만들어져 있지 않아서 관리자가 만들어야 사용이 가능함.

규칙 1. /etc/cron.allow 파일이 존재하면 이 파일에 있는 사용자만 crontab 명령어를 사용할 수 있음.
/etc/cron.deny 파일은 무시됨.
규칙 2. /etc/cron.allow 파일이 존재하지 않으면 /etc/cron.deny 파일에 있는 사용자를 제외한 모든 사용자는 crontab 명령어를 사용할 수 있음. (/etc/at.deny 파일에 아무것도 적혀 있지 않다면 모든 사용자가 사용 가능.)
규칙 3. 두 파일 모두 존재하지 않는다면 시스템 관리자만 cron 명령어를 사용할 수 있거나 모든 사용자가 사용할 수 있음.

옵션들.
-e : 특정 사용자의 crontab 파일을 편집 또는 생성. (edit) (/var/spool/cron/crontabs 디렉토리에 저장됨.)
-l (엘) : crontab 파일 목록을 출력.
-r : crontab 파일을 삭제. (remove)

-e 옵션으로 편집 시에는 VISUAL 또는 EDITOR 환경변수에 지정된 편집기 사용.
지정이 되어있지 않으면 해당 명령 입력 시에 어떤 편집기를 사용할 것인지 물어봄.

<마운트 관련 명령어>

- mount

파일시스템을 마운트하는 명령어.

mount <option> <장치명> <마운트 포인트>

장치명은 경로로 작성해야 함.

mount 만 작성할 경우 현재 마운트되어 있는 정보를 출력함.

/etc/mtab 파일의 내용과 동일한 내용을 출력함. (/etc/fstab 파일과 유의)

옵션들.

-t <파일시스템 종류> : 파일시스템 종류를 지정.

-o <마운트 옵션> : 마운트 옵션 (/etc/fstab 파일 작성 시 사용하는 옵션을 씀) 지정.

-f : 마운트 할 수 있는지 점검만 함.

-r : 읽기만 가능하게 마운트. (= -o ro 와 동일)

- umount

파일시스템을 언마운트하는 명령어.

umount <option> <장치명 or 마운트 포인트>

옵션.

-t <파일시스템 종류> : 파일시스템 종류를 지정함.

메모 포함[이52]: ex.
/dev/sdb2 등으로.

메모 포함[이53]: 현재 시스템에 마운트된 파일시스템의 정보가 저장되어 있는 파일이다.

이 파일의 내용은 다음과 같은 형식으로 작성된다.
1. 장치명
2. 마운트 포인트
3. 파일시스템의 종류
4. 마운트 옵션
5. 사용하지 않는 항목 두 개(0 0)

마지막의 사용하지 않는 항목 두 개는 /etc/fstab 파일과의 호환성을 위해 추가된 항목이다.

<디스크 관련 명령어>

- fdisk

디스크의 파티션 관리 명령어

fdisk <option> <장치명>

root 계정의 권한이 있어야 사용할 수 있음. (sudo 사용)

작업 후 w 로 저장해줘야 함.

fdisk 명령어 입력 시 내부 명령을 통해 작업하게 됨.

메모 포함[이54]: usb, cd, 디스크, LVM 등에서 사용하는 핵심적인 내부 명령들은 암기해야 한다.

표 7-5 fdisk 명령의 내부 명령

명령	기능	명령	기능
a	부팅 파티션을 설정한다.	p	파티션 테이블을 출력한다.
b	BSD 디스크 라벨을 편집한다.	q	작업 내용을 저장하지 않고 종료한다.
c	도스 호환성을 설정한다.	s	새로운 빈 Sun 디스크 라벨을 생성한다.
d	파티션을 삭제한다.	t	파티션의 시스템 ID를 변경한다(파일 시스템 종류 변경).
l	사용 가능한 파티션의 종류를 출력한다.	u	항목 정보를 변경 · 출력한다.
m	도움말을 출력한다.	v	파티션 테이블을 검사한다.
n	새로운 파티션을 추가한다.	w	파티션 정보를 디스크에 저장하고 종료한다.
o	새로운 빈 DOS 파티션을 생성한다.	x	실린더 개수 변경 등 전문가를 위한 부가적 기능이다.

옵션들.

-b <크기> : 섹터 크기를 지정.

-l (소문자 엘) : 파티션 테이블 출력. -> 이때는 fdisk -l 만 입력.

- mkfs (MaKe File System)

리눅스 파일시스템을 생성하는 명령어.

mkfs <option> <장치명>

메모 포함[이55]: mkfs.ext2, mkfs.ext3, mkfs.ext4 이런 식으로도 사용할 수 있다. 각각 mkfs -t ext2, mkfs -t ext3, mkfs -t ext4와 동일하게 작동한다.

이때 -v 옵션으로 상세정보를 출력할 수 있다.

옵션.

-t <종류> : 파일시스템 종류 지정. (default 값은 ext2 임)

- mke2fs (MaKe Ext2 File System)

리눅스 확장 파일 시스템(ext2, ext3, ext4)을 생성하는 명령어.

mke2fs <option> <장치명>

별도의 설정 파일인 /etc/mke2fs.conf 파일에 파일시스템 종류마다 기본 설정 값이 들어 있음.

옵션들.

-t <종류> : 파일시스템 종류 지정 (default 값은 ext2 임)
-b <블록 크기> : 블록 크기를 바이트 수로 지정.
-c : 배드 블록 체크
-f <프래그먼트 크기> : 프래그먼트 크기를 바이트 수로 지정.
-i <inode 당 바이트 수> (소문자 아이) : inode 당 바이트 수 지정 (default 값은 4096 바이트)
-m <예약 블록 퍼센트> : 슈퍼유저에게 예약해둘 블록의 퍼센트를 지정 (default 값은 5%)

df <option> <file system or 마운트 포인트>

- df (disk free)

현재 시스템에서 사용 중인 파일시스템의 디스크 사용량 출력 명령어

df 만 입력하면 여러 파일 시스템들의 정보들을 출력함.

파일 시스템 란에 . 을 치면 현재 파일 시스템 사용 정도를 보여줌

옵션들.

-a : 모든 파일시스템들의 디스크 사용량 확인.
-k : 디스크 사용량을 KB 단위로 출력.
-m : 디스크 사용량을 MB 단위로 출력.
-h : 디스크 사용량을 적절한 단위를 붙여서 출력.
-t <파일시스템 종류> : 지정한 파일시스템 종류에 해당하는 디스크의 사용량을 출력.
-T : 파일시스템 종류도 출력.

- du (disk usage)

디렉토리나 사용자별 디스크 사용량 출력 명령어.

du <option> <directory>

du 만 입력하면 현재 디렉토리의 디스크 사용량을 출력함.
출력 단위는 KB 가 default 임.

특정 사용자의 디스크 사용량을 확인하려면 해당 사용자의 홈 디렉토리를 인자로 넣으면 됨.

옵션들.

-s : 특정 디렉토리의 전체 디스크 사용량만을 출력.
-h : 디스크 사용량을 적절한 단위를 붙여서 출력.

메모 포함[이56]: f를 filesystem으로 암기하자.

메모 포함[이57]: df 명령어에서는 대체로 옵션을 사용한다.
file system 란에 뭔가를 직접 작성하는 것은 보지 못했다.

- fsck (File System CheckK)

```
fsck <option> <장치명>
```

파일시스템을 검사하고 필요시 복구하는 명령어.

기본적으로는 /etc/fstab 에 저장된 파일시스템을 대상으로 함.

파일시스템 종류별로 fsck.ext2, fsck.ext3, fsck.ext4 명령도 제공함.

옵션들.

-f : 강제로 검사.
-b <백업 슈퍼 블록의 위치> : 슈퍼블록으로 지정한 백업 슈퍼블록을 사용.
-y : 모든 질문에 yes 로 대답.
-a : 파일시스템 검사에서 중 문제를 찾으면 자동으로 복구.

```
e2fsck <option> <장치명>
```

- e2fsck

리눅스의 확장 파일시스템(ext2, ext3, ext4)을 검사하고 필요시 복구하는 명령어.

e2fsck 명령어로 검사할 때는 해당 파일시스템의 마운트를 해제해야 함.
마운트를 해제하지 않을 경우 예상치 못한 오류가 발생할 수 있음.

옵션들.

-f : 강제로 검사.
-b <백업 슈퍼 블록의 위치> : 슈퍼블록으로 지정한 백업 슈퍼블록을 사용.
-y : 모든 질문에 yes 로 대답.
-J <ext3 or ext4> : ext3 나 ext4 파일시스템을 검사할 때 지정.

- badblocks

```
badblocks <option> <장치명>
```

장치의 배드 블록을 검사하는 명령어.

배드 블록은 fsck 나 e2fsck 명령으로도 검사가 가능하지만 badblocks 는 배드 블록을 위한 별도의 명령어임.

옵션들.

-v : 세부 검사 결과 출력.
-o <파일> : 검사 결과를 해당 파일에 출력(저장).

- dumpe2fs (dump ext2 file system)

파일시스템의 정보 출력 명령어.

dumpe2fs <장치명>

많은 정보가 출력될 경우 grep 명령어를 사용함.

(ex. dumpe2fs /dev/sdb1 | grep superblock)

메모 포함[이58]: dump : 파일시스템에 오류가 발생했을 때 관련 정보를 출력하기 위해 정보를 저장해 놓은 곳.

dd if=<file> of=<file> bs=<바이트 수> count=<블록 수>

- dd

지정한 블록 크기만큼 파일을 복사해 넣는 명령어.

이 수업에서는 슈퍼블록을 삭제하기 위해 사용했음. 암기할 필요는 없을 듯.

if=<파일> (Input File) : 지정한 파일에서 읽어 옴.

of=<파일> (Output File) : 지정한 파일에 복사함.

bs=<바이트 수> : 한 번에 읽어오고 복사할 바이트 수.

count=<블록 수> : <블록 수> 만큼 복사.

<시스템 관련 명령어>

- systemctl (systemd control)

systemctl <option> <명령> <유닛명>

systemd 를 제어하는 명령어.

유닛명 작성에 유닛 종류는 생략 가능.

systemctl 만 입력하면 동작 중인 유닛이 출력됨.

옵션들.

-a : 상태와 상관없이(inactive, active 상관없이) 유닛 전체를 출력.

-t <유닛 종류> : 지정한 종류의 유닛만 출력.

명령들.

```
start: 유닛을 시작한다.
stop: 유닛을 정지한다.
reload: 유닛의 설정 파일을 다시 읽어온다.
restart: 유닛을 재시작한다.
status: 유닛 상태를 출력한다.
enable: 부팅 시 유닛이 시작하도록 설정한다.
disable: 부팅 시 유닛이 시작하지 않도록 설정한다.
is-active: 유닛이 동작하고 있는지 확인한다.
is-enabled: 유닛이 시작되었는지 확인한다.
isolate: 지정한 유닛 및 이와 관련된 유닛만 시작하고 나머지는 정지한다.
kill: 유닛에 시그널을 전송한다.
```

- shutdown

shutdown <option> <time> <메시지>

시스템 종료 명령어.

time 에는 종료할 시간을 작성함. -> hh:mm, +m(m 분 후), now 와 같은 형식들 사용.

메시지에는 모든 사용자들에게 보낼 메시지를 작성함.
이 메시지는 명령을 사용한 즉시 전달됨. 작성한 시간이 되면 보내는 게 아님.

옵션들.

-k : 실제로 시스템을 종료하지는 않고, 작성한 메시지만 전달.

-r : 종료 후 재시작.

-h : 종료하며 halt 나 power-off 상태로 이동.

-c : 이전에 했던 shutdown 명령 취소.

h 옵션을 안 쓰면 종료가 안됨?

<소프트웨어 관리 관련 명령어>

- apt-cache

apt-cache <option> <서브 명령>

APT 캐시에서 정보를 검색하여 서브 명령을 수행하는 명령어.

옵션들.

-f : 검색 결과로 패키지에 대한 전체 기록을 출력함.

-h (help) : 도움말 출력.

서브 명령들.

status : APT 캐시(패키지 데이터베이스)에 대한 전반적인 통계 정보를 출력.

dump : 현재 설치되어 있는 패키지를 업그레이드.

search <키워드> : APT 캐시에서 키워드를 검색하여 해당하는 패키지 목록과 간단한 설명을 출력.

showpkg <패키지명> (show package) : 패키지에 대한 의존성, 역의존성 정보를 검색하여 출력.

show <패키지명> : 패키지에 대한 간단한 정보 출력.

pkgnames (package names) : 사용 가능한 모든 패키지의 이름을 출력.

메모 포함[이59]: cache -> 캐시.
캐시란 고속 기억 장치를 뜻한다.

메모 포함[이60]: APT 캐시 : 패키지 데이터베이스.

- apt-get

apt-get <option> <서브 명령>

패키지 관리 명령어

내부적으로 dpkg 명령을 수행함

옵션들.

-d : 패키지를 내려 받기만 함.

-f : 의존성이 깨진 패키지를 수정.

-h (help) : 도움말 출력.

서브 명령들.

update : /etc/apt/sources.list 에 명시한 패키지 저장소로부터 패키지 정보를 읽어 동기화함.

upgrade : 현재 설치되어 있는 패키지 중 새로운 버전이 있는 패키지를 모두 업그레이드.

install <패키지명> : 패키지 설치. 여러 개의 패키지명을 한 번에 명시하여 설치할 수 있음.

remove <패키지명> : 패키지 삭제. 패키지의 설정 파일은 삭제하지 않음.

purge <패키지명> : 패키지 삭제. 패키지의 설정 파일도 삭제함.

autoremove : 패키지 자동 정리 및 삭제.

download <패키지명> : 패키지를 설치하지 않고 현재 디렉토리로 내려 받기만 함.

autoclean : 불완전하거나 오래된 패키지 삭제.

clean : /var/cache/apt/archives 에 캐시되어 있는 모든 패키지 삭제. (디스크 공간 확보)

check : 의존성이 깨진 패키지 확인.

source : 패키지의 소스 코드를 내려 받아 압축을 품.

메모 포함[이61]: 패키지 설치 시 업그레이드를 하지 않으려면
명령 제일 뒤에 --no-upgrade 옵션을 작성한다.

새로운 패키지를 설치하지 않고 업그레이드만 하려면
명령 제일 뒤에 --only-upgrade 옵션을 작성한다.

메모 포함[이62]: remove와 패키지명 사이에 --purge 옵션을 작성하여 설정 파일까지 제거할 수도 있다.

메모 포함[이63]: --download-only 옵션을 source 앞에 입력하면 소스 코드를 내려 받기만 한다.

--compile 옵션을 source 앞에 입력하면 소스 코드를 내려 받아 압축을 풀고 컴파일까지 한다.

- dpkg (Debian package)

패키지 관리 명령어.

apt-get <option> <파일명 or 패키지명>

apt-get 보다 더 세부적인 기능을 제공함.

옵션들.

-l (소문자 엘) (list) : 설치된 패키지 목록 출력 (더 자세한 설명은 교재 p.481)
-l <패키지명> (소문자 엘) : 패키지의 설치 상태 출력
-s <패키지명> : 패키지의 상세 정보 출력
-S <경로명> : 경로명으로 명시한 파일을 포함하는 패키지 검색
-L <패키지명> : 패키지에서 설치된 파일 목록 출력
-c <.deb 파일> : .deb 파일의 내용 출력
-i <.deb 파일> (install) : 해당 파일 설치
-r <패키지명> (remove) : 해당 패키지 삭제. 설정 정보는 삭제하지 않음
-P <패키지명> (Purge) : 해당 패키지와 설정 정보 모두 삭제
-x <.deb 파일 디렉토리> : 해당 파일을 지정한 디렉토리에 풀어 놓음.

- aptitude

패키지 관리 명령어.

aptitude <서브 명령>

기능과 서브 명령들이 apt-get 과 유사하지만,
aptitude 명령어에서는 curses 프로그램을 사용할 수 있음.
curses 프로그램은 텍스트 그래픽 기능을 제공하는 패키지 관리 프로그램. -> 마우스로 메뉴 선택 가능.

서브 명령을 작성하지 않고 aptitude 만 입력하면 curses 프로그램이 실행됨.

서브 명령들.

show <패키지명> : 패키지의 설치 여부와 상관없이 상세 정보 출력.
search <키워드> : 키워드를 검색하여 일치하는 키워드가 포함된 패키지 목록 출력.
update : /etc/apt/sources.list 에 명시한 패키지 저장소로부터 패키지 정보를 읽어 동기화함.
upgrade : 현재 설치되어 있는 패키지 중 새로운 버전이 있는 패키지를 모두 업그레이드.
install <패키지명> : 패키지 설치. 여러 개의 패키지명을 한 번에 명시하여 설치할 수 있음.
remove <패키지명> : 패키지 삭제. 패키지의 설정 파일은 삭제하지 않음.
purge <패키지명> : 패키지 삭제. 패키지의 설정 파일도 삭제함.
download <패키지명> : 패키지를 설치하지 않고 내려 받기만 함.
clean : 패키지 캐시 디렉토리에서 모든 패키지 파일 삭제. (디스크 공간 확보)

메모 포함[이64]: 패키지 설치 시 업그레이드를 하지 않으려면
명령 제일 뒤에 --no-upgrade 옵션을 작성한다.

새로운 패키지를 설치하지 않고 업그레이드만 하려면
명령 제일 뒤에 --only-upgrade 옵션을 작성한다.

메모 포함[이65]: remove와 패키지명 사이에 --purge 옵션을 작성하여 설정 파일까지 제거할 수도 있다.

- snap

스냅 패키지 관리 명령어.

snap <option> <명령>

옵션.

-h : 도움말을 출력함.

명령들.

disable : 스냅 서비스와 실행 파일의 사용을 중지함.
download <스냅명> : 지정한 스냅 패키지를 내려받음.
enable : 스냅 서비스와 실행 파일의 사용을 시작함.
find <스냅명> : 지정한 스냅명으로 검색함.
info <스냅명> : 지정한 스냅의 상세 정보를 출력함.
install <스냅명> : 지정한 스냅 패키지를 설치함. (/snap 디렉토리에 설치됨)
list : 설치한 스냅 목록을 출력함.
remove <스냅명> : 지정한 스냅을 삭제함.

- tar (tape archive)

아카이브 생성, 추출 명령어

tar <기능><option> <아카이브 파일> <파일명>

생성될 아카이브 파일을 제일 앞에 하나 작성하고, 뒤에는 파일들을 작성함.

기능과 옵션을 붙여서 씀.
기능과 옵션은 기본적으로 <기능>vf 으로 작성함.

파일명에는 여러 개의 파일을 작성할 수 있음.

기능들.

c (create) : 새로운 tar 파일 생성
t : tar 파일의 내용 출력
x : tar 파일에서 원본 파일 추출. -> 아카이브 파일이 현재 위치한 곳에 원본 파일이 추출됨.
r : 새로운 파일 추가 -> 지정한 파일을 무조건 아카이브의 마지막에 추가함.
u (update) : 수정된 파일을 업데이트 -> 아카이브에 파일이 이미 있는지 확인하고 아카이브에 추가함.

옵션들

f : 아카이브 파일이나 테이프 장치를 지정함. f 오른쪽에 오는 이름을 아카이브명으로 간주함.
v : 처리하고 있는 파일의 정보 출력
h : 심볼릭 링크의 원본 파일을 포함함
p : 파일 복구 시 원래의 접근권한 유지
j : bzip2로 압축하거나 해제 -> 파일 압축까지 한 번에 수행할 수 있음.
z : gzip 으로 압축하거나 해제 -> 파일 압축까지 한 번에 수행할 수 있음.

메모 포함[이66]: 지정한 파일이 아카이브에 없거나 수정된 파일일 경우 파일을 추가한다.

지정한 파일이 아카이브에 있다면, 아무 것도 변경하지 않는다.

메모 포함[이67]: 아카이브의 맨 마지막에 추가된다. 파일 추출 시에는 앞부터 순서대로 추출되므로 이렇게 추가한 파일은 뒤쪽 순서에서 출력된다.

- gzip (g zip)

파일 압축 및 압축 해제 명령어

```
gzip <option> <파일명>
```

지정한 파일을 압축하여 <파일명>.gz 파일을 생성함.

옵션들.

-d : 파일 압축 해제

-l (소문자 엘) : 압축된 파일 목록 출력

-r : 명시한 하위 디렉토리 내의 모든 파일 압축. (<file>에 디렉토리를 적으면 됨.)

-t : 압축 파일 검사

-v : 압축 정보 출력

-9 : 최대한 압축함

메모 포함[이68]: gzip <-> bzip2 대립구도.

- zcat (zip cat)

gzip 으로 압축된 파일의 내용 출력 명령어.

```
zcat <파일명>
```

- gunzip (g unzip)

gzip 으로 압축된 파일의 압축 해제 명령어.

```
gunzip <파일명>
```

gunzip 명령어 대신 gzip 명령어의 -d 옵션을 사용할 수도 있음.

-bzip2 (b zip 2)

파일 압축 및 압축 해제 명령어

지정한 파일을 압축하여 <파일명>.bz2 파일을 생성함.

옵션들.

-d : 파일 압축 해제
-l (소문자 엘) : 압축된 파일 목록 출력
-t : 압축 파일 검사
-v : 압축 정보 출력
--best : 최대한 압축함

- bzcat (bzip2 cat)

gzip 으로 압축된 파일의 내용 출력 명령어.

```
bzcat <파일명>
```

- bunzip2 (b unzip 2)

gzip 으로 압축된 파일의 압축 해제 명령어.

```
bunzip2 <파일명>
```

bunzip2 명령어 대신 bzip2 명령어의 -d 옵션을 사용할 수도 있음.

- gcc

c 프로그램 컴파일을 수행하는 명령어.

```
gcc <option><파일명>
```

옵션.

-o <파일명> : a.out 대신 지정한 이름으로 실행 파일을 생성함.

- make

여러 개의 파일을 한 번에 컴파일하기 위한 명령어.

```
make
```

소스 코드와 함께 배포 받은 makefile 의 정보를 읽어서 여러 소스 파일을 컴파일하고 링크하여 실행 파일을 생성함.

<사용자 관리 관련 명령어들>

- useradd

useradd <option> <로그인 ID>

사용자 계정 생성 명령어.

옵션 없이 계정을 생성하면 홈 디렉토리와 셸이 지정되지 않음.

옵션들로 지정하는 것들은 /etc/passwd, /etc/shadow 파일의 각 항목에 해당함.

useradd 는 계정 생성 시에 암호를 설정하지 않으므로, passwd 명령어를 사용해서 암호를 설정해 줘야 함.

옵션들.

-u <UID> : UID 지정.
-o (overlap) : UID 중복 허용.
-g <GID> : 기본 그룹의 GID 지정.
-G <GID> : 2 차 그룹의 GID 지정.
-d <디렉토리명> : 홈 디렉토리 지정.
-s <셸> : 기본 셸 지정.
-c <설명> : 부가적인 설명 지정.
-D : 기본값을 설정하거나 출력.
-e <유효 기간> : EXPIRE 항목 지정. (YYYY-MM-DD 형식으로)
-f <비활성 일수> : INACTIVE 항목 지정.
-k <디렉토리명> : 계정 생성 시 초기 파일을 설정해 놓은 디렉토리를 지정. (/etc/skel 디렉토리)
-m : 홈 디렉토리 생성.

- adduser

useradd <option> <로그인 ID>

사용자 계정 생성 명령어.

옵션 없이 계정을 새성하면 기본 설정에 따라 계정을 생성함
기본 설정은 /etc/adduser.conf 파일에 저장되어 있음.

계정 생성 과정에서 암호를 지정함.
계정 생성 과정에서 설명에 입력할 내용을 추가로 확인함.

옵션들.

--uid <UID> : UID 지정.
--gid <GID> : 기본 그룹의 GID 지정.
--home <디렉토리명> : 홈 디렉토리 지정.
--shell <셸> : 기본 셸 지정.
--gecos <설명> : 설명 지정.

메모 포함[이69]: 홈 디렉토리와 셸 등이 자동으로 지정된다.

- usermod (user modify)

사용자 계정 정보 수정 명령어.

```
usermod <option> <로그인 ID>
```

계정과 관련된 모든 정보를 수정할 수 있음.
패스워드 에이징 정보를 수정할 수 있음.

옵션들.

-u <UID> : UID 지정.
-o (overlap) : UID 중복 허용.
-g <GID> : 기본 그룹의 GID 지정.
-G <GID> : 2 차 그룹의 GID 지정.
-d <디렉토리명> : 홈 디렉토리 지정.
-s <셸> : 기본 셸 지정.
-c <설명> : 부가적인 설명 지정.
-l <이름> : 명시한 이름으로 계정 이름을 바꿈.

-e <유효 기간> : EXPIRE 항목 지정. (YYYY-MM-DD 형식으로)
-f <비활성 일수> : INACTIVE 항목 지정.

- chage

패스워드 에이징 관리 명령어.

```
chage <option> <로그인 ID>
```

옵션들.

-m <MIN> : MIN 설정.
-M <MAX> : MAX 설정.
-W <WARNING> : WARNING 설정.
-I <INACTIVE> (대문자 아이) : INACTIVE 설정.
-E <EXPIRE> : EXPIRE 설정.

-l (소문자 엘) : 패스워드 에이징 설정 내용 확인.
-d : 암호를 변경한 마지막 날짜를 출력.

- userdel (user delete)

사용자 계정 삭제 명령어.

```
userdel <option> <로그인 ID>
```

-r 옵션을 지정하지 않으면 계정만 삭제되고 해당 계정 관련 파일들은 삭제되지 않음.

옵션들.

-r : 홈 디렉토리와 메일 디렉토리도 함께 삭제함. 해당 계정 소유의 파일은 삭제해 주지 않음.
-f : 사용자가 로그인 중 이어도 강제 삭제.

- groupadd

그룹 생성 명령어.

```
groupadd <option> <그룹명>
```

옵션을 지정하지 않으면 GID 를 가장 마지막 번호의 다음 번호로 자동 설정함.

옵션들.

-g <GID> : 그룹의 GID 를 지정함.

-o (overlap) : GID 의 중복을 허용함.

- addgroup

그룹 생성 명령어

```
addgroup <option> <그룹명>
```

옵션을 지정하지 않으면 /etc/adduser.conf 에 지정된 GID 를 기준으로 가장 마지막 번호의 다음 번호로 자동 설정됨.

옵션들.

--gid <GID> : 그룹의 GID 를 지정함.

- groupmod (group modify)

그룹 정보 수정 명령어.

```
groupmod <option> <그룹명>
```

옵션들.

-g <GID> : GID 지정.

-o (overlap) : GID 의 중복을 허용함.

-n <그룹명> (name) : 그룹명 지정. 파일이나 디렉토리의 소속 그룹명도 자동으로 수정됨.

- groupdel (group delete)

그룹 삭제 명령어

```
groupdel <그룹명>
```

- gpasswd (group password)

그룹 관리, 그룹 암호 관리 명령어.

gpasswd <option> <그룹명>

옵션을 사용하지 않고 gpasswd <그룹명>만 입력하면 해당 그룹명의 암호 설정 절차가 진행됨.

옵션들.

-a <로그인 ID> (add) : 해당 로그인 ID 의 사용자를 그룹에 추가함.

-d <로그인 ID> (delete) : 해당 로그인 ID 의 사용자를 그룹에서 삭제함.

-r (remove) : 그룹 암호 삭제.

- newgrp

소속 그룹 변경 명령어.

newgrp <그룹명>

소속 그룹을 2 차 그룹 중 하나로 바꿀 때는 암호가 필요 X.

소속 그룹을 2 차 그룹이 아닌 그룹으로 바꾸려고 하면 해당 그룹의 암호가 필요함.

메모 포함[이70]: ex. 소속 그룹을 testgroup 으로 바꾸려고 하면
testgroup 의 그룹 암호가 필요하다.

- who

현재 시스템을 사용하는 사용자의 정보를 출력하는 명령어.

who <option>

옵션을 작성하지 않고 who 만 입력하면 여러 가지 정보들을 출력함.

옵션들.

-q : 사용자명만 출력.

-H : 출력 항목의 제목을 포함하여 출력.

-b : 마지막으로 재시작한 날짜와 시간 출력

-m : 현재 사용자 계정의 정보 출력. -> UID 출력함.

-r : 현재 런레벨 출력.

- w

현재 시스템을 사용하는 사용자의 정보를 출력하는 명령어.

w <사용자명>

- last

시스템 로그인, 로그아웃 기록을 출력하는 명령어.

last

- groups

groups <로그인 ID>

소속 그룹 확인 명령어.

groups 만 입력하면 현재 사용자 계정이 속한 그룹을 출력함.

- passwd

passwd <사용자명>

사용자 계정의 비밀번호 변경 명령어.

passwd 만 입력하면 로그인한 계정의 비밀번호 변경 절차가 진행됨.
인자에 사용자 이름을 명시하면 그 이름을 가진 계정의 비밀번호 변경 절차가 진행됨.

옵션들.

-l <사용자 ID> (소문자 엘) (lock) : 지정한 계정의 암호를 잠금. -> /etc/shadow 파일의 암호가 !로 시작함.
-u <사용자 ID> (소문자 엘) (unlock) : 지정한 계정의 암호 잠금 해제
-d <사용자 ID> (소문자 엘) (delete) : 지정한 계정의 암호 삭제. -> /etc/shadow 파일의 암호가 비어 있음.

- chown (change owner)

chown <option> <사용자 ID> <파일명>

파일 소유자를 변경하는 명령어

이 명령어에서 명시한 사용자로 파일 소유자가 변경됨.

파일명 자리에는 디렉토리명도 들어갈 수 있음.

chown 은 root 권한으로만 사용이 가능함.

chown 으로 파일 소유자와 그룹을 동시에 바꾸려면 사용자 ID 인자에 <사용자 ID>:<그룹명> 을 작성함.

옵션.

-R : 서브 디렉토리의 소유자 또는 소유 그룹도 변경함.

- chgrp (change group)

chgrp <option> <그룹명> <파일명>

파일 소유 그룹을 변경하는 명령어

파일명 자리에는 디렉토리명도 들어갈 수 있음.

옵션.

-R : 서브 디렉토리의 소유 그룹도 변경함.

- quotacheck

쿼터 파일을 생성, 수정, 확인하기 위해 파일시스템을 스캔하는 명령어.

quotaoff 상태에서 사용해야 함.

```
quotacheck <option> <filesystem>
```

옵션들.

-a (all) : 전체 파일시스템을 스캔함. 이때 파일시스템을 인자로 작성하지 않아도 됨.
-u (user) : 사용자 쿼터 확인.
-g : 그룹 쿼터 확인.
-m : 파일시스템을 리마운트 하지 않음.
-v : 명령 진행 상황을 상세하게 출력.

- quotaon / quotaoff

파일시스템의 쿼터 기능을 활성화/비활성화 하는 명령어.

```
quotaon/quotaoff <option> <filesystem>
```

옵션들.

-a : 전체 파일시스템의 쿼터 기능 활성화/비활성화. 이때 파일시스템을 인자로 작성하지 않아도 됨.
-u : 사용자 쿼터 활성화.
-g : 그룹 쿼터 활성화.
-v : 명령 진행 상황을 상세하게 출력.

- edquota (edit quota)

쿼터 편집 명령어.

```
edquota <option> <계정 ID or 그룹명>
```

옵션들.

-u : 사용자 쿼터 설정.
-g : 그룹 쿼터 설정
-p <계정 ID> : 쿼터 설정 복사 -> 지정한 계정의 쿼터 설정을 뒤 인자에 명시한 계정에 복사함.

메모 포함[이71]: ex. edquota -p testuser jhun
-> testuser의 쿼터 설정을 jhun에 복사한다.

- quota

쿼터 정보 출력 명령아.

```
quota <option> <계정 ID or 그룹명>
```

옵션들.

-u : 사용자 쿼터 정보 출력.

-g : 그룹 쿼터 정보 출력.

- repquota (report quota)

파일시스템 내 전체 사용자들의 쿼터 정보를 요약하여 출력하는 명령어.

옵션들.

-a : 전체 파일시스템의 쿼터 정보를 출력.

-v : 사용량이 없는 쿼터의 정보도 출력.

-u : 사용자의 쿼터 정보를 출력.

-g : 그룹의 쿼터 정보를 출력.

```
repquota <option> <filesystem>
```

+ 인자에 파일시스템을 작성하는 방법

마운트 포인트를 명시함.

<네트워크 관련 명령어들>

- uname

시스템 정보를 출력하는 명령어.

uname <option>

옵션들.

-n : 호스트 이름 출력.

-a : 호스트 이름을 포함한 시스템 관련 정보 출력.

- hostname

호스트 이름을 출력하거나 설정하는 명령어.

hostname <호스트 이름>

hostname 만 입력하면 호스트 이름을 출력함.

이름을 지정하면 호스트 이름을 해당 이름으로 바꿈.

- ifconfig (interface configure)

네트워크 인터페이스를 설정하는 전통 명령어.

ifconfig <인터페이스명> <option> <IP 주소>

메모 포함[이72]: 인터페이스 환경 설정

ifconfig 만 입력하면 현재 설치된 네트워크 인터페이스의 정보들을 출력함.

수동으로 네트워크 인터페이스를 설정해 주려면 전부 직접 작성해 주면 됨.

-> ifconfig <인터페이스명> <IP 주소> netmask <주소> broadcast <주소>

옵션들.

-a : 시스템의 전체 인터페이스에 대한 정보를 출력함.

up/down : 인터페이스를 활성화/비활성화함.

netmask <주소> : 넷마스크 주소를 설정함.

broadcast <주소> : 브로드캐스트 주소를 설정함.

- route

라우팅 테이블을 편집하고 출력하는 명령어.

route <명령>

route 만 입력하면 현재 라우팅 테이블을 출력함.

기본 게이트웨이 추가	route add default gw 게이트웨이 주소 dev 인터페이스명 route add default gw 192.168.1.1 dev eth0
기본 게이트웨이 제거	route del default gw 게이트웨이 주소 route del default gw 192.168.1.1

명령들.

add : 라우팅 경로나 기본 게이트웨이를 추가.

del (delete) : 라우팅 경로나 기본 게이트웨이를 삭제.

- nslookup (name sever look up)

nslookup <도메인명>

DNS 서버와 대화식으로 질의하고 응답을 받는 명령어.

호스트 이름으로 IP 주소를 알아내는 명령어.

nslookup 명령어만 입력하면 프롬프트가 >로 바뀜.

이때 호스트 이름을 입력하면 IP 주소를 출력함.

메모 포함[이73]: uname - n이나 hostname으로 호스트 이름을 얻어내면 nslookup로 사용 중인 시스템의 IP주소를 알아낼 수 있다.

- ping

ping <option> <목적지 주소>

네트워크에서 통신이 가능한지 확인하는 명령어.

목적지 주소에는 IP 주소나 호스트 이름을 작성할 수 있음.

ping 명령을 입력하면 패킷을 보내기 시작함. 패킷 전송이 끝나면 통계 정보를 출력함.

패킷은 기본적으로 56 바이트임. (헤더 정보까지 포함하면 딱 64 바이트임)

옵션들.

-a : 통신이 가능하면 소리를 냄.

-q : 통계 정보만 출력.

-c : 보낼 패킷 수를 지정. 패킷 수를 지정하지 않으면 계속 패킷을 보내기 때문에 강제 종료해야 함.

- traceroute

목적지까지 패킷이 거치는 경로를 출력하는 명령어.

```
traceroute <목적지 주소>
```

출력 결과에서 *** 가 출력되는 부분은 네트워크가 연결되지 않는 부분임. -> 연결이 끊어지는 지점 확인 가능.
통신 장애가 있거나 traceroute 명령을 거부한다는 의미.

- whois

IP 주소가 어느 기관의 것인지 출력하는 명령어.

```
whois <IP 주소>
```

- netstat (network state)

네트워크의 상태 정보를 출력하는 명령어.

```
netstat <option>
```

옵션들.

-a : 모든 소켓 정보를 출력. -> 서비스 포트가 LISTEN 상태인 경우, 포트가 열러 있는 것임.
-r : 라우팅 테이블을 출력. -> route 를 입력한 것과 동일한 결과를 출력함.
-n : 호스트 이름 대신에 IP 주소를 출력.
-i : 모든 네트워크 인터페이스 정보를 출력.
-s : 프로토콜별로 네트워크 통계 정보를 출력.
-p : 열러 있는 포트를 사용하는 프로세스의 이름과 PID 를 출력.

- arp (address resolution protocol)

같은 네트워크에 연결된 시스템들의 MAC 주소와 IP 주소를 확인하는 명령어.

```
arp <IP 주소>
```

arp 만 입력하면 같은 네트워크에 연결된 시스템들의 MAC 주소와 IP 주소를 출력함.
(HWaddress 가 MAC 주소임)

IP 주소를 지정하면 해당 시스템의 MAC 주소를 출력함.

- tcpdump (TCP dump)

tcpdump <option>

패킷을 캡처하는 명령어.

tcpdump 만을 입력하면 현재 시스템에서 주고받는 모든 패킷을 캡처하여 패킷의 헤더 부분을 출력함.
이때 직접 종료하지 않으면 계속 출력함.
종료할 때는 캡처한 패킷의 개수를 출력함.

옵션들.

-c <숫자> : 지정한 패킷 수만큼만 캡처하고 종료함.
-i <인터페이스명> : 특정 인터페이스 지정.
-n : IP 주소로 출력.
-q : 간단한 형태로 출력.
-X : 패킷의 내용을 16진수와 ASCII 로 출력.
-w <파일명> : 캡처한 내용을 해당 파일에 저장. -> 이 옵션만을 사용하면 바이너리로 저장함.
-r <파일명> : 지정한 파일에서 캡처한 내용을 읽어 옴.
-> 바이너리 파일은 cat 이나 vi 로 읽을 수 없기 때문에 이 옵션을 사용해서 읽어야 함.
host <호스트명 또는 주소> : 해당 호스트가 받거나 보낸 패킷만 출력.
tcp <포트 번호> : 지정한 포트 번호 패킷만 캡처함.
ip : IP 패킷만 캡처함.

메모 포함[이74]: TCP/IP 프로토콜 할 때 그 TCP 인 것 같다.
TCP는 전송 계층인데, 패킷이 해당 계층과 관련이 있나 보다.

<Samba 관련 명령어들>

- smbclient (Samba client)

smbclient <option>

윈도우의 자원에 접근하기 위해 리눅스에서 사용하는 명령어.
서버의 자원에 접근하기 위해 사용하는 클라이언트.

옵션들.

-L <서버 IP 주소> : 서버에서 이용할 수 있는 서비스 목록 출력.
-U <사용자 계정> : 서버에 접속할 사용자 지정.

- smbpasswd (Samba password)

smbpasswd <option>

사용자의 삼바 접속 암호를 설정하는 명령어.

옵션들.

-a <계정> : 지정한 계정의 암호 설정.
-x <계정> : 지정한 계정의 암호 삭제.
-d <계정> : 지정한 계정을 사용 불가로 지정.
-e <계정> : 지정한 계정을 사용할 수 있게 함.

<기타 명령어들>

- whereis (리눅스+)

whereis <option> <command>

명령어의 절대 경로 출력 명령어.

옵션들.
-b : 바이너리 파일만 검색함.
-m : 메뉴얼 파일만 검색함.
-s : 소스 파일만 검색함.

- which

which <option> <command>

명령어의 절대 경로 출력 명령어
하나의 절대 경로를 찾으면 즉시 명령을 종료함. -> 하나의 경로만 출력됨.
에일리어스가 지정된 명령일 경우에는 해당 에일리어스를 먼저 검색해 출력함.

옵션.
-a : 해당 명령어로 접근할 수 있는 모든 경로 출력

- man

man <명령어>

명령어의 메뉴얼 출력 명령어

- clear

clear

터미널 화면을 초기화하는 명령어

- date

date <option> <format>

현재 날짜와 시각을 출력하거나 설정하는 명령어

옵션들
-s <string> : 해당 string 으로 시간 설정.

- ln (link)

link <option> <file> <link_file>

파일의 링크를 생성하는 명령어

디렉토리는 하드링크를 생성할 수 없음. 심볼릭 링크는 생성할 수 있음.

심볼릭 링크는 파일 시스템이 달라도 생성할 수 있음

옵션들.

-s : 심볼릭 링크 파일 생성. (symbolic)

+ 파일 링크

1) (하드) 링크

파일에 붙은 이름을 하드 링크라고 함.

하드 링크 파일은 원본 파일과 inode 번호가 같음. (단순히 기존 파일에 새로운 파일명을 추가하는 것임)

하드 링크파일을 vi 로 수정하면 원본파일에 반영됨.

원본파일을 제거해도 하드 링크 파일은 사용할 수 있음.

2) 심볼릭 링크

기존 파일을 가리키는, inode 가 다른 새로운 심볼릭 링크 파일 생성. (바로가기)

심볼릭 링크 파일의 내용은 원본 파일의 경로임. -> 원본 파일이 삭제되면 사용할 수 없는 파일이 됨.

원본파일을 제거하면 하드 링크 파일이 남아 있어도 심볼릭 링크 파일은 사용할 수 없음.

원본파일을 제거하고 새로운 파일을 원본 파일의 이름으로 만들면 새로 만든 파일에 심볼릭 링크가 걸림.

심볼릭 링크 파일의 파일 종류는 l(소문자 엘)로 시작함.

ls -l 명령어로 확인하면 원본 파일이 "링크파일 -> 원본파일" 형태로 명시됨.

심볼릭 링크 파일을 실행하면 원본파일의 경로가 아니라 원본파일의 내용이 출력됨.

심볼릭 링크 파일을 vi 로 열어 수정하면 원본파일과 하드 링크 파일에 반영됨.

+ ls -l 으로 확인할 수 있는 하드링크 개수의 기본값은 1임.

기본적으로 1개의 이름을 가지고 있다는 의미.

하나의 하드링크를 생성한 후 ls -l 로 보면 하드링크 개수가 2임.

+ 옵션 작성법

옵션은 -ab 형태로도 작성할 수 있고, -a -b 형태로도 작성할 수 있음.

메모 포함[이75]: 닉네임 부여.

메모 포함[이76]: 파일 복사와의 차이점.
cp로 복사할 경우 둘은 별개의 파일임.
링크의 경우에는 둘의 파일명은 다르지만
inode 번호가 같기 때문에 동일한 파일을 가리킴.

메모 포함[이77]: vi로 심볼릭 링크 파일을 확인하면 원본 파일의 내용을 편집하게 됨.

메모 포함[이78]: 심볼릭 링크를 생성할 때 ln 명령어에 작성했던 원본파일의 이름을 가진 같은 경로의 파일을 만들면 해당 파일로 심볼릭 링크가 걸림.

2. 디렉토리와 파일

1. 리눅스 파일의 종류와 특징

리눅스는 모든 것을 파일이라고 함.

- 일반 파일 (regular file, data file)

데이터를 주로 저장하는 파일.

텍스트 파일, 이미지 파일, 실행 파일 등이 포함됨.
이미지 파일, 실행 파일은 바이너리 파일로 저장됨.
텍스트 파일은 문서 편집기로 열 수 있지만 이미지, 실행 파일은 응용 프로그램이 필요함.

- 디렉토리

파일을 분류하기 위해 사용하는 파일.

해당 디렉토리에 들어있는 파일이나 하위 디렉토리에 대한 정보가 저장됨. (ls 로 확인 가능)

- 심볼릭 링크

원본 파일을 다른 이름으로 만든 파일. (바로가기)

- 장치 파일

리눅스는 장치도 파일로 취급함.

2. 리눅스 파일의 구성

파일 -> 파일명 + inode + 데이터 블록

1) 파일명

파일 접근 시 사용하는 이름.

2) inode

외부적으로는 번호로 표시되는, 파일에 대한 상세 정보와 데이터 블록의 주소를 담고 있는 구조체.

파일명이 달라도 inode 번호가 같으면 같은 파일.

메모 포함[이79]: ls -l로 확인할 수 있는 상세 정보는 inode에 저장되어 있는 정보이다.

3. 디렉토리 (=폴더)

컴퓨팅에서 파일을 분류하기 위해 사용하는 공간

리눅스는 효율적인 관리를 위해 디렉토리들로 이루어진 계층 구조를 가짐. (역 Tree 구조.)

- 루트 디렉토리

파일시스템의 최상위 디렉토리. (모든 파일 및 디렉토리 포함.)

루트 디렉토리는 부모 디렉토리가 없음.

/

- 작업 디렉토리 (현재 디렉토리)

파일시스템 상에서 현재 위치를 의미.

. (점)

- 부모 디렉토리 (상위 디렉토리)

특정 디렉토리의 위에 있는 디렉토리.

.. (점점)

- 서브 디렉토리 (하위 디렉토리)

특정 디렉토리의 아래에 있는 디렉토리

- 홈 디렉토리

사용자에게 할당된 개인 디렉토리

리눅스 최초 실행 시 작업 디렉토리는 홈 디렉토리임.

사용자가 처음 사용자 계정을 만들 때 할당됨.

~

+ 특수한 디렉토리들.

dev (device) : 장치 파일이 담긴 디렉토리.
home : 사용자의 홈 디렉토리가 생성되는 디렉토리.
media : 외부 장치를 연결(마운트)하는 디렉토리.
opt : 추가 패키지가 설치되는 디렉토리. (option?)
root : root 계정의 홈 디렉토리. (루트 디렉토리와는 다른 것.)
sys : 리눅스 커널과 관련된 것들이 있는 디렉토리. (system?)
usr (Unix System Resource) : 실행 파일, 라이브러리 파일, 헤더 파일 등 여러 파일들이 들어있는 디렉토리.
boot : 부팅에 필요한 커널 파일이 들어있는 디렉토리.
etc : 리눅스 시스템 환경 설정을 위한 파일들이 들어있는 디렉토리. (etc: 기타)
lost+found : 파일 시스템에 문제가 생겨 복구할 경우, 문제가 되는 파일이 저장되어 있는 디렉토리.
(길을 잃은 파일 저장.) 보통은 비어 있음.
mnt : 파일 시스템을 임시로 마운트하는 디렉토리.
proc : 프로세스 정보 등 커널 관련 정보가 저장되는 디렉토리. (process)
run : 실행 중인 서비스와 관련된 파일들이 저장되는 디렉토리.
srv : 시스템에서 제공하는 서비스의 데이터가 저장되는 디렉토리. (service?)
tmp (temporary) : 시스템 사용 중 발생하는 임시 데이터가 저장되는 디렉토리.
var : 시스템 사용 중 발생하는 것들 중 내용이 자주 바뀌는 파일이 주로 저장되는 디렉토리. (variable?)

메모 포함[이80]: 이거 어느 정도는 암기해야 함.

+ 파일명 작성 규칙

파일의 이름에는 알파벳, 숫자, 붙임표(-), 밑줄(_), 마침표 만 사용할 수 있음.
공백문자 또한 사용할 수 없음.

마침표로 파일명이 시작되면 숨김 파일로 간주됨.

알파벳의 대문자와 소문자는 구분됨.

메모 포함[이81]: c언어에서 식별자를 정하는 규칙과는 달리,
리눅스 파일명에는 붙임표와 마침표까지 사용할 수 있고, 숫자로 시작해도 된다.

+ 파일 시스템

디렉토리와 파일로 구성된 전체 집합.

리눅스는 계층적 파일 시스템을 가지고 있음.

4. 경로 (path)

파일시스템의 특정 위치에서부터 지정된 위치까지 통과하게 되는 디렉토리 및 파일을 나열한 것.

특정 파일의 위치 표시.

각 파일을 구분하는 구분자로 /를 사용.

맨 앞에 /가 온다면 그건 구분자가 아니라 루트 디렉토리임.

- 절대경로

루트로부터 시작되는 경로.
/가 루트를 의미하므로, 반드시 /로부터 시작함.
특정 위치를 가리키는 절대경로는 항상 동일.

특정 위치까지 이동하면서 거치게 되는 모드 디렉토리명을 명시해야 함.

- 상대경로.

현재 디렉토리를 기준으로 시작되는 경로.
현재 위치가 어디냐에 따라 상대경로는 바뀔 수 있음.

현재 디렉토리를 기준으로 하위 디렉토리로 내려가려면 해당 디렉토리명을 명시.
현재 디렉토리를 기준으로 상위 디렉토리로 올라가려면 .. (점점)을 디렉토리명 대신 추가.

메모 포함[이82]: 올라가려면 반드시 ..을 써줘야 한다. 디렉토리명을 작성하면 작업 디렉토리 내에 해당 디렉토리로 가게 됨.

위로 올라가는 용어라고 생각하지 말고
상위 디렉토리를 가리키는 용어라고 생각해야
이해가 편함.

3. vi 편집기 (visual editor)

1. 리눅스의 문서 편집기 종류

행 단위 편집기: ed, ex, sed
화면 단위 편집기: vi, emacs
GUI 편집기: gedit

ed : 유닉스 초기에 사용되던 편집기로, 불편해서 잘 안씀.
ex : vi에 연결하여 다양한 기능들을 제공하는 데에 사용됨.
sed : 파일의 내용을 일관적으로 바꾸어 출력해 줌. 셸 스크립트 작성 시에 사용.
vi : 리눅스에서 일반적으로 사용함. ex 편집기를 기반으로 만들어짐.
emacs : 기능이 다양하지만 복잡하여 잘 애호가들이 사용.
gedit : GUI환경인 GNOME에서 제공하는 문서 편집기.

2. 모드형과 비모드형 편집기

1) 모드형
입력 모드와 명령 모드가 구분됨.
(ex. vi)

2) 비모드형
입력 모드와 명령 모드가 구분되어 있지 않음.
Ctil이나 Alt같은 특수 키를 이용해 편집 기능을 사용함.
(ex. 한글, 워드)

3. vi 편집기를 실행하는 법

"vi 파일명.확장자"
-> vi 편집기 실행.

vi 명령을 통해 vi를 실행함.

해당 파일이 없을 경우에는 그 이름으로 새 파일을 생성함.

vi만 칠 경우 빈 파일이 생성되고 이후에 이름을 설정해줘야 함.

4. 명령 모드와 입력 모드의 전환

입력 모드: 파일에 직접 입력하는 모드.

명령 모드: vi 명령어를 사용하는 모드.

마지막 행 모드: 커서가 화면의 맨 하단으로 이동해서 특별한 명령들을 사용할 수 있는 모드.
:, /, ?를 눌러 시작함
enter키를 누르면 작성한 명령 실행, esc키를 누르면 명령 모드로 돌아감. (작성하던 것 취소)

메모 포함[이83]: 여기서는 명령 모드에 포함시켜 설명함

- 명령모드->입력모드 : i, a, I, A, o, O 중 하나를 입력

i : 커서 위치에 문자 입력 시작
a : 커서 위치 다음 칸에 문자 입력 시작
I (대문자 아이) : 커서가 위치한 행의 맨 앞에 문자 입력 시작 (행의 첫 칼럼으로 이동하여 입력)
A : 커서가 위치한 행의 맨 끝에 문자 입력 시작 (행의 마지막 칼럼으로 이동하여 입력)
o : 커서가 위치한 행 다음 행에 문자 입력 시작
O : 커서가 위치한 행의 이전 행에 문자 입력 시작

- 입력모드->명령모드: Esc 키 입력

+ 처음 vi를 실행시켰을 때는 명령 모드로 시작한다.

+ 명령 모드가 기본이고 입력 모드라는 기능이 있는 것이라고 생각하면 편함.
즉, 입력을 위해 특정한 전환 키를 사용하여 해당하는 기능을 수행하게 하는 것

5. 명령

- 저장 및 종료 명령

:w : 현재 파일을 문서로 저장

:w <filename> : 해당 파일명으로 새로운 문서 저장

:wq, :wq!, ZZ : 문서 저장 후 vi 종료

:q! : 문서를 저장하지 않고 vi 종료

:q : 작업한 것이 없을 때 단순 종료

- 커서 이동 명령

h : 커서를 왼쪽으로 이동 (h j k l 암기.)

j : 커서를 아래로 이동

k : 커서를 위로 이동

l (소문자 엘) : 커서를 오른쪽으로 이동

w : 커서를 오른쪽 단어 첫번째 글자로 이동

e : 커서를 오른쪽 마지막 글자로 이동 (현재 단어의 마지막부터 시작함)

b : 커서를 왼쪽 단어 맨 앞 글자로 이동 (back)

^, 0, <home 키>: 커서를 현재 행의 맨 왼쪽으로 이동

$, <end 키> : 커서를 현재 행의 맨 오른쪽으로 이동

+, <enter 키> : 커서를 다음 행의 맨 앞으로 이동

-: 커서를 이전 행의 맨 앞으로 이동

H : 커서를 화면의 맨 위 행 맨 앞으로 이동 (맨 위는 아닐 수 있음. 커서가 갈 수 있는 곳까지 감) (High)

M : 커서를 화면의 중간 행 맨 앞으로 이동. 행의 개수가 짝수라면 두 행 중 위의 행으로 이동. (Middle)

L : 커서를 화면의 맨 아래 행 맨 앞으로 이동 (맨 아래는 아닐 수 있음. 커서가 갈 수 있는 곳까지 감) (Low)

G : 커서를 문서의 마지막 행 맨 앞 글자로 이동 (Go)

<인수>G, :<인수> : 커서를 인수에 해당하는 행으로 이동

:$: 커서를 파일의 마지막 행으로 이동

메모 포함[이84]: 줄임말들.
w : write
q : quit
y : yank
p : put
d : delete
c : change

메모 포함[이85]: G: 문서의 끝 의미
0: 행의 맨 처음부터 커서 앞 칸까지 의미
$, 대문자 하나: 커서부터 행의 맨 마지막까지 의미
w: 단어 의미. 현재 단어에서는 커서 위치부터 문자 끝까지를 의미. -> 앞에 숫자 명시할 수 있음.
동일한 문자 2개 반복: 하나의 행 전체를 의미 -> 앞에 숫자 명시할 수 있음.
문자 하나: 해당 문자가 가지는 의미 -> 앞에 숫자 명시할 수 있음.
<block> 문자: 해당 블록에 문자의 의미 적용
! : 중단하는 것을 의미
:<인수> : 인수에 해당하는 행 의미

메모 포함[이86]: 아래가 기본 방향이기 때문에 왼쪽에 있음.

메모 포함[이87]: -를 사용할 때는 +와 다르게 shift 안 눌러도 되는 거 유의하자!

- 화면 이동 명령

표 3-7 화면 이동 명령 키

기존 명령 키	기능	추가 명령 키
^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.	

(up, down, back, front, y, e로 암기.)

- 삭제 명령 (x, d)

x, #x : 커서 위치 문자 하나 삭제, #에는 삭제할 글자 수를 작성할 수 있음

X : 커서 위치 이전 칸 문자 삭제

dw, #dw : 커서가 위치한 단어를 삭제. 현재 위치한 단어는 커서 뒤부터만 삭제함. #에는 삭제할 단어 수를 작성할 수 있음

dd, #dd : 커서 위치 행 삭제(잘라두기), #에는 삭제(잘라두기)할 행 수를 작성할 수 있음

d$, D : 커서 위치부터 행의 끝까지 삭제

d0 : 현재 행의 처음부터 커서위치 앞 문자까지 삭제

dG : 현재 행 위치부터 문서의 끝까지 삭제

:<block>d : 지정된 <block> 삭제

:#d : #에 지정한 행을 삭제. (잘라두기)

- 수정 명령 (r, c, s)

r : 커서가 위치한 문자를 삭제하고 하나의 문자를 입력 받은 후 다시 명령모드로 전환.

R : 수정모드인 입력모드로 전환

cw, #cw : 커서 위치부터 단어 끝까지 삭제한 후 입력모드로 전환, #에 수정할 단어의 개수를 작성할 수 있음.

cc : 커서가 위치한 행 전체 삭제한 후 입력모드로 전환

c0 : 현재 행의 처음부터 커서 위치 앞 문자까지 제거 후 입력모드로 전환

c$, C : 커서 위치부터 행 끝까지 제거 후 입력모드로 전환

s, #s : 커서 위치의 문자를 하나 지우고 입력모드로 전환. #에는 삭제할 글자의 수를 작성할 수 있음.

+ 수정 명령에서 r 빼고는 모두 사용 후 입력모드로 바뀜

+ 한 문장이 길어져 아래 행까지 내려갈 경우 아래 행이 아니라 한 문장 취급됨

메모 포함[이88]: 해당 내용을 삭제하고 입력모드로 바꾸는 것이
수정하는 것이다.

메모 포함[이89]: 삭제 명령과 형식이 유사함.
d대신 c를 쓰면 입력모드가 되는 것!

메모 포함[이90]: 여기에서 수정모드를 사용할 수 있게 하는 것은 R 명령어 하나뿐이다.

수정모드 : 글자를 타이핑할 경우 기존의 문자가 밀리는 것이 아니라 삭제되는 모드이다.
삽입모드 : 글자를 타이핑할 경우 기존의 문자가 밀리는 모드이다.

워드 프로그램에서는 보통 insert키로 이 모드를 전환하게 되어있다.

- 복사, 붙여넣기 명령 (y, p)

yy, #yy : 커서 위치 행 복사, #에 복사할 행의 수를 작성할 수 있음.
y0 : 현재 행의 처음부터 커서 위치 앞 칸까지 복사
y$: 커서 위치부터 행의 끝까지 복사
yw : 커서 위치부터 단어 끝까지 복사
yG : 현재 행 위치부터 문서의 끝까지 복사
:<block>y : <block> 복사
:#y : #로 지정한 행 복사
p (소문자) : 복사한 내용을 커서 위치 다음 행에 붙여넣기
P (대문자) : 복사한 내용을 커서 위치 위 행에 붙여넣기
:#pu : #로 지정한 행 다음 행에 붙여넣기.
:pu : 커서 위치 다음 행에 붙여넣기
:<block> co <인수> : <block>을 복사하여 <인수> 행 다음에 붙여넣기 (copy)

메모 포함[이91]: 1. 복사 명령어에 따라 붙여 넣어지는 위치가 다르다.
다음 행에 삽입: yy, yG, <block> y
커서 오른쪽 칸에 삽입: y0, y$, yw

2. 삭제 명령을 수행하고 p를 사용하면 삭제된 내용이 붙여 넣어진다. p는 붙여넣기 보다는 잘라넣기가 더 정확한 설명이다.

- 문자열 검색, 대체 명령

/<인수> : 커서 위치부터 아래 방향으로 인수(문자열) 검색
?<인수> : 커서 위치 위 방향으로 인수(문자열) 검색
n : 원래 찾던 방향으로 이전 인수 계속 검색 (next)
N : 원래 찾던 반대 방향으로 이전 인수 계속 검색 (Next)

:s/인수1/인수2/ : 커서가 위치한 행의 첫번째 인수1을 인수2로 변환
:%s/인수1/인수2/g : 문서 내의 모든 인수1을 인수2로 변환 (global) (%는 문서 전체를 의미함)
:<block>s/인수1/인수2/ : <block>의 모든 행에서 제일 앞 인수1 하나를 인수2로 변환
:<block>s/인수1/인수2/g : <block>의 모든 행에서 인수1을 인수2로 변환 (global)
:<block>s/인수1/인수2/gc : <block>의 모든 행에서 인수1을 인수2로 변환하는데 매번 수정 여부를 물어봄. (global check)

:g/인수1/s//인수2/gc : 문서 전체에서 인수1을 탐색하고, 선택에 따라 인수2로 변환할 수 있는 기능 실행

메모 포함[이92]: :s/인수1/인수2/ 이거는 고정.
맨 앞과 뒤에 문자들 작성.
앞에는 적용범위, 뒤에는 세부사항.(g, c)
%: 문서 전체
g: 모든 것 변환. (global) 이거 안 쓰면 하나만.
c: 확인받음 (check)
블록 사용가능.

메모 포함[이93]: 아래 방향이 기본이기 때문에 위 방향은 shift를 눌러야 함.

+ 인수로 입력한 문자열이 한 단어로 존재하지 않아도 그 문자의 배열이라면 검색이 됨.
+ 검색 시 대문자 소문자의 구분은 기본적인 것.

메모 포함[이94]: hello를 검색하면 "hello"만이 아니라 "qerqwrhello"같은 곳에서도 찾을 수 있다는 의미.

- 다중 파일 편집 명령 (n, e, r)

:r <file> : 지정한 파일을 읽어서 현재 커서 위치에 삽입 (read)
:e <file> : 지정한 파일을 편집 시작 (현재 파일을 저장하지 않고 이 명령을 사용하면 오류 메시지가 출력됨.)
:n : vi 시작 시 여러 파일을 지정했을 경우, 다음으로 해당되는 파일을 편집 시작 (next)

- 범위 지정 명령 (블록)

<인수1>,<인수2> : 인수1, 인수2에는 행 숫자를 넣음. 인수 1번째 행~인수 2번째 행 블록이 만들어짐.

특수한 의미를 가지는 용어들도 사용할 수 있음

$: 파일에서 마지막 행

. : 현재 행 (커서가 있는 행)

% : 전체 행

+인수 : 현재 행으로부터 아래로 인수만큼의 행을 추가로 포함 (현재 행 또한 포함됨)

-인수 : 현재 행으로부터 위로 인수만큼의 행을 추가로 포함 (현재 행 또한 포함됨)

메모 포함[이95]: 항상 오른쪽, 아래가 기준이다.
글 읽는 방향.

(ex.

표 3-13 범위 지정 명령 키

명령 키	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.

)

+ 인수1이 인수2보다 클 경우, 시스템에서 사용자 의도대로 입력한 것인지 확인 절차를 거친 후 수행하게 됨

+ 결과적으로 선택된 행이 하나일 경우, 그 행에서만 명령을 수행함

+ 블록 지정 시에는 인수에 연산을 쓸 수도 있음.

- 셸 명령어 사용 명령

:! <셸 명령> : vi 작업을 잠시 중단하고 셸 명령을 실행함. (<enter>키를 누르면 다시 vi로 돌아옴.)
:sh : vi를 잠시 중단하고 셸로 감. ('exit'을 작성하면 vi로 돌아옴.) (shell)

- 명령 취소 명령

:e! : 수정된 사항을 취소. (마지막 저장 명령 직후의 상태로 복귀. vi가 닫히지는 않음.)
u : 이전 명령 취소 (undo)
U : 커서가 위치한 행에서 한 모든 명령 취소. (Undo)

- 그 외 명령

~ : 대문자를 소문자로, 소문자를 대문자로 변환
. : 마지막 수행 명령 반복
<ctrl> G, <ctrl> g : 문서 이름과 현재 커서의 위치 정보 표시
<ctrl> l (소문자 엘) : 현재 화면을 다시 출력 (refresh해줌.)
J : 현재 행과 아래 행을 연결해서 하나의 행으로 만듦 (join)

6. vi 환경설정

1) EXINIT 환경 설정 변수에 지정하여 환경설정. (환경변수에 저장하기만 하면 로그아웃 시 초기화됨)

2) 홈 디렉터리에 .exrc 또는 .vimrc 파일을 만든 후 환경 설정 명령을 작성해서 환경설정
(vi를 실행할 때마다 해당 파일을 확인하여 초기화하므로 계속 사용할 수 있음)

3) vi의 마지막 행 모드에서 명령으로 환경설정. (vi 껐다 키면 초기화됨)

set cindent : c언어 프로그래밍에 필요한 자동 들여쓰기 기능
tabstop=<숫자> : 탭 키의 간격 조절 기능
set shiftwidth=<숫자> : 자동 들여쓰기의 간격 조절 기능
set nu : 행에 번호를 표시함
set nonu : 행 번호 감춤
set ruler : 현재 커서의 위치 출력 기능
set title : 타이틀 바에 현재 작업 중인 문서 이름 출력 기능
syntax on : 문법 구분 기능 (색깔)
set hlsearch : 패턴 검색 시 해당 부분의 하이라이트 표시 기능 (highlight search)
set list : 가시화되지 않은 특수문자들을 가시화함.
set nolist : 가시화된 특수문자들을 비가시화함.
set showmode : 현재 모드를 표시함
set noshowmode : 현재 모드를 숨김
set : set으로 설정한 vi 환경 설정 값들을 출력함
set all : 모든 vi 환경 변수와 현재 값을 출력함.

메모 포함[이96]: EXINIT='<명령>' ; export EXINIT
을 적으면 설정 가능.

메모 포함[이97]: vi를 실행할 때마다 이 파일을 확인하게 됨.

7. 버퍼

복사하거나 잘라낸 내용을 임시로 저장해두는 공간.
버퍼에 저장한 것은 로그아웃 해도 남아있음. (네임드 언네임드 모두 남아있음.)

메모 포함[이98]: 하나의 파일에서 복사한 것을 저장해 두면
다른 파일에서도 그것을 사용할 수 있음.

1) 네임드 버퍼

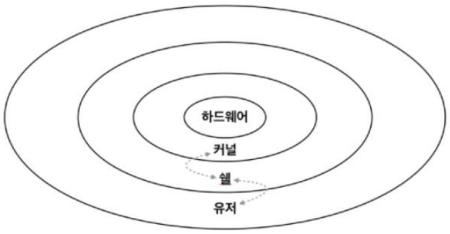
"<버퍼 이름><명령어>

이름을 붙여서 사용할 수 있는 버퍼.
이름을 붙이면 각각 독립적으로 내용을 저장하고 사용할 수 있음.
네임드 버퍼에 내용을 저장하거나 붙여 넣으려면 명령어 앞에 ' "버퍼 이름'을 적어 주기만 하면 됨.
작성되고 있는 내용이 시각적으로 드러나지 않음.
네임드 버퍼의 이름은 알파벳 또는 숫자로 함. (숫자를 사용할 경우 숫자 버퍼라고도 함)

2) 언네임드 버퍼

이름을 붙이지 않은 버퍼.
yy 키로 복사하거나 dd 키로 잘라낸 경우 그 내용이 저장되는 곳.
하나의 내용만 저장함.

4. Shell



1. 셸

커널과 유저 사이에서 소통을 돕는 부분.

2. 셸의 기능

명령어 해석기, 프로그래밍, 환경설정 등의 기능을 수행함.

1) 명령어 해석기

사용자와 커널 사이에서 명령을 번역하고 해석함.

2) 프로그래밍 기능

셸은 자체에 간단한 프로그래밍 기능이 있음.
반복적으로 수행하는 등의 작업을 하나의 프로그램으로 만들 수 있음.
셸에서 만든 프로그램을 '셸 스크립트'라고 함

메모 포함[이99]: echo, printf 등, 배시 셸 명령어 등이 그 예시이다.

3) 사용자 환경 설정 기능

셸은 사용자 환경 설정 기능을 위해 초기화 파일을 제공함. 이 파일에 사용자별 환경이 작성됨.

3. 셸의 종류

본 셸, 콘 셸, C 셸, 배시 셸, 대시 셸.

1) 본 셸

명령 이름은 sh (최초의 셸이므로 shell 의 sh.)
최초의 셸
개발자인 스티븐 본의 이름을 따서 본 셸이라고 함
처리 속도가 빠르지만 편의를 위한 기능을 제공하지 못함. (히스토리, 에일리어스, 작업 제어 등)

지금도 많은 셸 스크립트가 본 셸을 기반으로 함.
현재 사용되는 대부분의 본 셸은 오리지널이 아니라 콘 셸이나 배시 셸로 심볼릭 링크가 되어있음.

2) C 셸

명령 이름은 csh
빌 조이가 개발.

본 셸의 기능을 확장한 것. 본 셸에는 없던 사용자 편의 기능들을 제공함.
여러 기능들을 제공하기 때문에 편리하지만, 크기가 크고 느림.
본 셸과의 호환성이 없음. C 셸은 본 셸을 기반으로 만들어진 것이 아님.

C 셸은 c 언어와 관련이 있음.
셸 스크립트 작성 구문 형식이 C 언어와 같음. (그래서 이름이 C 셸)
커널(핵심)은 c 언어로 작성되어 있기 때문에 C 셸은 커널과 직접 소통할 수 있음.

3) 콘 셸

명령 이름은 ksh
데이비드 콘이 개발.
본 셸과의 호환성이 있음
C 쉘의 여러 편의 기능들을 모두 제공하면서도 빠름.

4) 배시 셸

명령 이름은 bash
브레인 폭스가 개발.

C 셸과 콘 셸의 기능을 포함하면서도 본 셸을 기반으로 만들어짐.
본 셸과의 호환성 있음.

GPL 라이선스에 의거하여 자유롭게 사용 가능
리눅스의 기본 셸. 리눅스 셸이라고도 함.

메모 포함[이100]: 이 수업에서 사용하는 우분투의 로그인 셸은
배시 셸이다.

메모 포함[이101]: GNU에서 만든 자유 소프트웨어 라이센스.
대표적으로 리눅스 커널이 사용함.

5) 대시 셸

포직스 표준을 준수하며 개발.
본 셸을 기반으로 만들어짐.
우분투에서는 6.10 버전 이후부터 대시 셸을 사용함.
본 셸에 비해 부팅 시 셸 스크립트 실행이 빠르고, 파일 크기가 작고, 신뢰성이 높음.

'ls -l /bin/sh' 명령을 입력하면 대시 셸이 심볼릭 링크되어 있음. (-l 옵션 안 넣으면 안 보임)

+ 사용 중인 셸을 확인하는 방법

1) ps 명령어로 프로세스 정보를 확인하면 볼 수 있음.
2) grep 명령어로 사용자 정보가 들어 있는 파일(/etc/passwd)에서 사용자명으로 검색하면 찾을 수 있음.
3) 환경변수 SHELL 을 echo 명령어로 확인하여 알 수 있음.

+ 기본 프롬프트 모양

$: 본 셸, 콘 셸, 배시 셸의 기본 프롬프트
%: C 셸의 기본 프롬프트

4. 로그인 셸과 서브 셸

1) 로그인 셸

사용자가 로그인하면 자동으로 실행돼서 명령을 기다리는 셸.

chsh 명령어로 해당 사용자의 로그인 셸을 바꿀 수 있음.

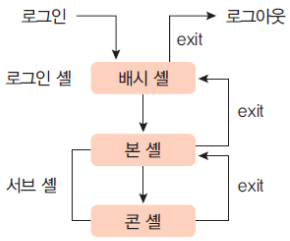


그림 4-2 로그인 셸과 서브 셸

2) 서브 셸

프롬프트에서 생성한 또 하나의 셸

서브 셸은 또 다른 서브 셸을 생성할 수 있음.

ctrl + d, exit 등을 입력하여 이전 셸로 돌아갈 수 있음.

5. 셸 내장 명령

셸은 셸 안에 내장된 명령들을 가지고 있음.
일반적인 리눅스 명령들은 실행파일로 저장되지만, 셸에 내장된 명령은 셸 안에 포함되어 있음.

셸의 내장 명령으로는, cd, echo, printf 등이 있음.

셸은 대기 상태로 있다가 명령 작성 시 내장 명령언지 우선 판단함.
내장 명령이 아니라면 자식 프로세스 하나를 생성해 해당 명령어의 실행 파일을 실행하게 함.
자식 프로세스가 실행을 종료하면 원래의 셸로 돌아가 프롬프트를 출력함.

6. 셸의 특수문자들

셸에서는 편의를 위한 특수문자들을 사용할 수 있음.
셸은 명령을 받으면 우선 특수문자가 있는지부터 확인함.

1) * (임의의 문자열을 나타내는 와일드 카드)

메모 포함[이102]: 어떤 용도로도 쓰일 수 있는 카드.

임의의 문자열을 나타냄.
*만 사용하거나 다른 문자열과 붙여서 사용할 수 있음.

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 해당한다는 데 주의한다.
cp *.txt ../ch3	확장자가 txt인 모든 파일을 상위 디렉터리 아래의 ch3 디렉터리로 복사한다.
ls -l h*d	파일명이 h로 시작하고 d로 끝나는 모든 파일의 상세 정보를 출력한다. hd, had, hard, h12345d 등이 조건에 맞는 모든 파일의 정보를 볼 수 있다.

2) ?, [ ] (하나의 문자를 나타내는 와일드 카드)

하나의 문자를 나타냄.
?는 임의의 한 문자를, [ ]는 괄호 안에 있는 문자 중 하나의 문자를 나타냄.
[ ]를 사용할 때는 -를 이용해서 숫자나 알파벳을 범위로 나타낼 수 있음. (ex. a-z, 0-4)

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 -l 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 -l tmp[1-3].txt	[1-3]은 1부터 3까지의 범위를 의미한다. 따라서 ls -l 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.

3) ~, - (디렉토리를 나타내는 특수문자)

특정 디렉토리를 나타냄.
~는 현재 작업자의 홈 디렉토리를 나타내고, '~<로그인 ID>'로 사용하면 해당 사용자의 홈 디렉토리를 나타냄.
-는 cd 명령으로 이동하기 이전의 작업 디렉토리를 나타냄.
(ex. cd -를 입력하면 cd 명령어로 이동하기 직전의 작업 디렉토리로 돌아감)

4) ;와 | (<shift>역슬래시) (명령과 명령을 이어주는 특수문자, 파이프)

명령과 명령을 연결함.
;는 명령들을 구분해 주고, 왼쪽부터 차례대로 수행하게 함.
|는 명령들을 구분해 주고, 왼쪽 명령의 수행 결과를 오른쪽 명령의 입력으로 전달함.

5) ' ' (작은 따옴표), " " (큰 따옴표) (의미를 제거하는 특수문자)

글자를 문자열로 취급하고 특수문자 기능을 없앰.
셸이 해당 문자를 특수문자로 인식하게 하고 싶지 않다면 이 기능을 이용함.
' '는 모든 특수문자를 일반문자로 만듦.
" "는 $, ₩, ` `(백틱)을 제외한 모든 특수문자를 일반문자로 만듦.

메모 포함[이103]: c언어와는 다르게 작은 따옴표 또한 문자열을 다룰 때 사용한다.

표 4-5 특수문자 ' '와 " "

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

6) $ (변수를 위한 특수문자.)

$<변수이름>

변수가 가지고 있는 값을 사용함. (It indicates the value the variable holds.)

(ex. JHUN=32; echo $JHUN 이면 32 를 출력함)

7) ``(백틱) (의미를 사용하는 특수문자)

`` 안에 있는 문자열을 명령으로 해석하여 해당 명령을 수행한 후 그 결과를 `` 위치에 사용함.
그냥 따옴표와는 반대되는 느낌의 성질. 따옴표는 의미를 제거하고 백틱은 의미를 사용함.

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 2018. 01. 20. (토) 18:32:45 KST
ls /usr/bin/`uname -m`	uname -m 명령의 실행 결과를 문자열로 바꾸어 파일명으로 사용한다.

8) ₩ (역슬래시)

바로 뒤에 오는 특수문자의 의미를 제거하고 일반문자로 취급하거나 이스케이프 문자에 사용함.

의미를 제거할 때는 특수문자 바로 앞에 사용함.
(ex. rm -r test.₩*이면 test.으로 시작하는 파일들을 지우는 것이 아니라 test.*라는 파일을 지움)

+ 이스케이프 문자

₩(역슬래시)로 시작하는 특수한 문자.

이스케이프 문자는 ₩와 알파벳 하나로 이루어져 있지만 한 글자로 취급됨.

표 4-12 이스케이프 문자

이스케이프 문자	기능
\a	ASCII 종소리 문자(07)
\d	'요일 월 일' 형식으로 날짜를 표시한다(예: Wed May 1).
\e	ASCII의 이스케이프 문자로 터미널에 고급 옵션을 전달한다.
\h	첫 번째 .(마침표)까지의 호스트 이름(예: server.co.kr에서 server)
\H	전체 호스트 이름
\n	줄 바꾸기
\s	셸 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	배시 셸의 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 끝부분을 표시한다.

메모 포함[이104]: +/! 이거는 암기. 교재 문제에 나와있음.

+ h 는 host
s 는 shell
t 는 time
u 는 user
v 는 version
w 는 working directory

₩[₩]는 주석?

9) > (출력 리다이렉션 방법 1)

<command> <number> > <file>

출력되는 정보의 방향을 해당 파일로 리다이렉션함.

정보를 저장할 파일이 이미 존재하면 덮어쓰기함. (원래의 정보를 지우고 해당 정보를 저장함)

지정한 이름의 파일이 없으면 해당 파일을 생성해서 수행함.

<number>에는 표준 입출력 장치의 파일 디스크립터를 명시함.
특정 번호를 명시하면 일시적으로 오른쪽에 오는 파일을 해당 번호로 취급하겠다는 의미.
아무것도 명시하지 않은 경우, 1 로 취급됨.
출력된 실행 결과를 리다이렉션하려면 1 번을, 출력된 오류 메시지를 리다이렉션하려면 2 를 작성함.

연속해서 리다이렉션할 수 있음.

메모 포함[이105]: 표준 입출력 장치를 일시적으로 파일로 바꾸는 것.

10) >> (출력 리다이렉션 방법 2)

<command> <number> >> <file>

출력되는 정보의 방향을 해당 파일로 리다이렉션함.
출력되는 문자열을 해당 파일에 추가함.
>와는 달리 삭제하고 넣는 게 아니라 파일 내용의 아래쪽에 붙여 넣음.

지정한 이름의 파일이 없으면 해당 파일을 생성해서 수행함.

11) < (입력 리다이렉션 특수기호)

<command> <number> < <file>

입력되는 정보를 해당 파일에서 가져오도록 리다이렉션함.
해당 file 의 데이터를 명령에 입력함.

<는 0<를 생략하여 표현한 형태임.

(cat 명령어는 파일을 읽어서 출력하는데, 여기서 표준 입력이 자동으로 리다이렉션된 것)

+ 실행 결과와 오류 메시지를 한 번에 리타이렉션하기

<명령> > <파일 1> 2> <파일 2> 이면,
명령을 수행 후 출력되는 문자열을 파일 1 에 저장하고 오류가 발생하면 해당 오류 메시지를 파일 2 에 저장함.
<명령> > <파일 1>의 오류 메시지인지, <명령>의 오류 메시지인지 명확히는 모르겠음

+ 표준 입출력 장치 (기본값)

표준 입력 장치 (0): 셸이 작업할 때 필요한 정보를 받아들이는 장치 -> 키보드

표준 출력 장치 (1): 실행 결과를 내보내는 장치 -> 모니터

표준 오류 장치 (2): 오류 메시지를 내보내는 장치 -> 모니터

표준 입출력 장치를 일시적으로 파일로 바꾸는 것을 '리다이렉션'이라고 함. (redirection)

메모 포함[이106]: 셸이 출력하는 것은 두 가지가 있다.
하나는 실행 결과이고
다른 하나는 오류 메시지이다.

+ 파일 디스크립터 (descript)

파일 관리를 위해 붙이는 번호.

리눅스에서는 장치도 파일로 관리하기 때문에 표준 입출력 장치에 번호가 붙어있음.

표 4-9 표준 입출력 장치의 파일 디스크립터

파일 디스크립터	파일 디스크립터 대신 사용하는 이름	정의
0	stdin	명령의 표준 입력
1	stdout	명령의 표준 출력
2	stderr	명령의 표준 오류

(standard input, output, error)

+ 덮어쓰기 방지 (> 등의 결과를 막아 줌)

해당 명령을 환경 설정하면 항상 적용됨.

set -o noclobber : 덮어쓰기를 할 수 없게 설정함. (clobber: 손실을 가하다.)

set +o noclobber : 덮어쓰기를 할 수 있게 설정함.

+ cat 명령어로 파일 편집하기

cat 만 작성하면 셸에 입력을 진행할 수 있음. 이때 표준 출력을 특정 파일로 리다이렉션하면 그 파일에 내용을 작성할 수 있음.

+ 출력 결과 디라이렉션해서 버리기

/dev/null 로 오류 메시지 등을 리다이렉션하면 그 메시지는 삭제됨.

null 로 정보를 보내면 그 정보는 삭제됨. (휴지통 이런 게 아니라 바로 삭제됨)

+ 표준 출력과 표준 오류를 한 파일로 리다이렉션하기

<명령> > <file> 2>&1 이라고 하면 오류 메시지 또한 다른 내용을 덮어쓰지 않고 file 에 저장됨.

&1 이 표준 출력 파일을 가리킨다고 함.

>와 &1 사이에 띄어쓰기가 있으면 안 됨.

메모 포함[이107]: 왜 그런 지는 제대로 안 알려줌.

5. 셸의 유용한 기능들

메모 포함[이108]: 배시 셸에 관한 내용을 설명하는 듯

1. 셸 환경설정 관련 변수

셸의 환경을 설정하기 위한 값을 저장할 수 있도록 셸 변수와 환경변수가 존재함.

1) 셸 변수 (지역변수)

현재 셸에서만 적용되는 환경 설정 값을 저장함.
부모 셸에서 자식 셸로 상속되지 않음.

2) 환경변수 (전역변수)

모든 셸에서 적용되는 환경 설정 값을 저장함.
부모 셸에서 자식 셸로 상속됨. (복사됨.)

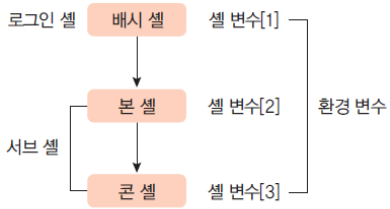


그림 4-4 셸 변수와 환경 변수

+ 주요 환경 변수

환경 변수는 관습적으로 대문자로 씀.

표 4-10 주요 셸 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 셸
LOGNAME	사용자 계정 이름		

2. 셸 환경설정 관련 변수 관리

1) 전체 변수의 내용 출력 (셸의 변수 출력 명령어)

set : 셸 변수와 환경변수 모두 출력, 함수로 정의된 것들도 출력. (settings)

env : 환경변수만 출력 (environment)

2) 특정 변수의 내용 출력

echo 명령어와 특수문자 $를 이용해서 출력할 수 있음.

set이나 env의 값을 |로 받아서 grep등을 이용해 내용을 출력할 수도 있음.

```
echo $<변수 이름>
```

메모 포함[이109]: 여기서 $ 특수문자의 의미가 무엇인지 조사하기

3) 셸 변수 설정

변수명과 문자열을 명시해 셸 변수를 정의할 수 있음.

= 좌우에 공백이 있으면 안됨. 공백이 있어야 대입으로 취급됨.

대입하는 문자열에 띄어쓰기가 있다면 작은 따옴표로 묶어줘야 함.

(ex. SOME=test)

```
<변수명>='<문자열>'
```

메모 포함[이110]: 큰따옴표도 되는 것 같지만 일단은 에일리어스와 같은 작은 따옴표로 통일한다.

4) 환경 변수 설정하기

환경 변수를 만들기 위해서는 우선 셸 변수를 정의하고, export 명령으로 이를 환경 변수로 변경해야 함.

```
export <option> <셸 변수>
```

메모 포함[이111]: export: 수출하다.
즉, 셸 변수를 다른 셸들로 수출하는 것.
모든 셸에서 사용하는 환경변수로 만든다는 것.

5) 변수 해제하기

unset 명령으로 변수를 해제할 수 있음. (셸 변수, 환경변수 모두에 사용 가능)

```
unset <변수>
```

3. 에일리어스 (가명) (도구 만들기)

alias <name>='<command>'

특정 명령 또는 명령들에 가명을 부여하는 기능.
; 또는 | 특수기호를 사용해서 여러 명령을 넣을 수 있음.

에일리어스 설정 시에 =양옆에는 띄어쓰기를 하면 안됨.
명령 중간에서 띄어쓰기를 하려면 전체를 작은따옴표로 묶어줘야 함.

원래 있는 alias 를 다시 설정할 경우 설정한 것으로 갱신됨.

이미 존재하는 명령의 이름으로 에일리어스를 설정할 수도 있음.
셸은 명령을 실행할 때 에일리어스가 있는지를 먼저 확인하기 때문에 명령 대신 에일리어스가 실행됨.

alias 를 입력하면 이미 만들어진 에일리어스 목록을 보여줌.
시스템에 따라 이미 만들어진 에일리어스가 다름.

unalias 로 에일리어스를 삭제할 수 있음.

unalias <alias>

메모 포함[이112]: + alias는 '가명'이란 뜻.

+ 에일리어스와 히스토리는 본 셸에는 없던 기능으로, C 셸에 처음 등장하였다.

메모 포함[이113]: 큰따옴표도 되는 것 같음.

4. 이전 명령 사용하기 (히스토리)

1) history: 명령 입력 기록을 출력하는 기능.

history 를 입력하면 명령 입력 기록을 출력할 수 있음.

history

2) ! 문자를 이용해서 명령을 재실행 할 수 있음.

표 4-11 !를 사용한 명령 재실행 방법

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

3) 화살표 키를 이용해서 이전 명령들을 사용할 수 있음.

4) 히스토리의 저장

배시 셸은 로그아웃 시에 history 목록을 사용자 홈 디렉토리 아래의 숨김 파일 .bash_history 에 저장함.

메모 포함[이114]: 변수나 에일리어스와는 달리, 로그아웃해도 사용할 수 있음.

5. 프롬프트 설정

1) 프롬프트 설정 환경변수

환경변수 PS1에 문자열을 저장하면 프롬프트를 바꿀 수 있음.

메모 포함[이115]: Prompt System을 줄인 것으로 추정.

PS1 에 '<문자열>'을 지정하면 해당 문자열이 프롬프트가 됨.
PS1 에 '$<환경변수>'를 지정하면 해당 환경변수가 수행되고 그게 프롬프트가 됨.
PS1 에 "`<명령>`"을 지정하면 해당 명령이 수행되고 그게 프롬프트가 됨. (백틱 사용.)
PS1 에 '<이스케이프 문자>'를 지정하면 해당 이스케이프 문자가 수행되고 그게 프롬프트가 됨.

위에 나온 것들을 혼용할 수 있음.

(ex.

```
user1@myubuntu:~/linux_ex/ch4$ PS1='LINUX ] '
LINUX ]
```

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

```
[/home/user1/linux_ex]  PS1='`uname -n` $ '
myubuntu $
```

```
myubuntu $ PS1='[\u \T] \!$ '
[user1 10:54:50] 103$
```

)

2) 프롬프트 색상 설정하기

시간 나면 해보기.

컬러 프롬프트

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

표 4-13 프롬프트 컬러 번호

컬러	글자색 번호	배경색 번호
검은색	30	40
빨간색	31	41
초록색	32	42
갈색	33	43
파란색	34	44
보라색	35	45
청록색	36	46
하얀색	37	47

표 4-14 프롬프트 특수 기능 번호

번호	기능
0	기본 색상
1	볼드
4	밑줄
5	반짝임
7	역상
10	기본 폰트
38	밑줄 사용 가능
39	밑줄 사용 불가능

컬러 프롬프트 설정 예

PS1="[\u@\h\w]\\$"
PS1="[\u@\h:$(pwd)]\\$"

① 파란색으로 설정하기

```
[user1 10:54:50] 103$ PS1='\e[34mLinux $\e[0;0m'
Linux $      → 파란색
```

② 파란색의 볼드로 설정하기

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

③ 밑줄 친 빨간색으로 설정하기

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

④ 배경은 갈색, 글자

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

6. 환경설정 파일

1. 환경설정 파일

사용자가 로그인할 때마다 자동으로 실행되는 명령을 저장한 파일.
셸마다 다른 이름의 파일을 사용함. (이 수업에서는 배시 셸의 것을 설명함)

파일로 저장하지 않은 셸 변수, 환경변수, 에일리어스 로그아웃하면 사라진다. 계속 사용하려면 파일로 저장해야 함.

시스템 환경설정 파일과 사용자 환경설정 파일이 있음.
로그인 시에, 시스템 환경설정 파일이 먼저 실행되어 시스템 공통 환경을 만들고,
그 후 사용자 환경설정 파일을 순서대로 실행하여 사용자별 환경을 설정함.

메모 포함[이116]: 파일로 저장하지 않은 셸 변수, 환경변수, 에일리어스 로그아웃하면 사라진다.

메모 포함[이117]: 로그인할 때마다 이곳에 적힌 내용으로 환경설정을 초기화하기 때문에 '초기화 파일'이라고도 부른다.

2. 시스템 환경설정 파일

시스템을 사용하는 전체 사용자의 공통 환경을 설정하는 파일.
어떤 사용자가 로그인하던 반드시 실행됨.
관리자가 관리하는 파일임. 일반 사용자는 수정할 수 없음.

배시 셸에는 /etc 디렉토리 아래에 있음.

표 4-15 배시 셸의 시스템 환경 설정 파일

파일	기능
/etc/profile	• 본 셸이나 본 셸과 호환되는 모든 셸에 공통으로 적용되는 .profile 파일이다. • 배시 셸의 경우 /etc/bash.bashrc 파일을 실행한다. • 배시 셸이 아닌 경우 프롬프트를 #(root 사용자)나 $(일반 사용자)로 설정한다. • /etc/profile.d/*.sh 파일을 실행한다.
/etc/bash.bashrc	• 시스템 공통으로 적용되는 .bashrc 파일이다. • 기본 프롬프트를 설정한다. • sudo 명령과 관련된 힌트를 설정한다.
/etc/profile.d/*.sh	• 언어나 명령별로 각각 필요한 환경을 설정한다. • 필요시 설정 파일을 추가한다.

(위의 것 두 개는 업데이트 시 일관성 있는 관리를 위해(?) 수정하지 않는 것이 좋다고 함.)

3. 사용자 환경설정 파일

각 사용자의 홈 디렉토리에 숨김 파일로 존재함.
각 사용자가 수정 가능.

배시 셸에서 기본적으로 설정되는 사용자 환경설정 파일들.

표 4-16 배시 셸의 사용자 환경 설정 파일

파일	기능
~/.profile	• 경로 추가 등 사용자가 정의하는 환경을 설정한다. • .bashrc 파일이 있으면 실행한다.
~/.bashrc	• 히스토리의 크기를 설정한다. • 기본 에일리어스나 함수 등을 설정한다.
~/.bash_aliases	• 사용자가 정의한 에일리어스를 별도 파일로 저장한다.
~/.bash_logout	• 로그아웃 시 실행할 필요가 있는 함수 등을 설정한다.

각 파일에는 로그인 시 수행될 명령을 작성하면 됨. (ex. alias rm='rm -i' 등 완전한 명령으로.)

4. 환경설정 파일 적용법

사용자 환경설정 파일을 수정했으면 적용을 해야 함.

방법 1) source 명령어를 이용해서 파일명을 명시하기.
방법 2) source 자리에 . 을 작성해도 됨.
방법 3) 로그아웃 했다가 다시 로그인하기.

source <파일명>
. <파일명>

+ 다른 셸의 환경설정 파일

(중요한 건 아닌 것 같음. 설명도 대충 넘기심.)

표 4-17 다른 셸의 환경 설정 파일

셸	시스템 초기화 파일	사용자 초기화 파일	실행 조건	실행 시기		
				로그인	서브 셸	로그아웃
본 셸	/etc/profile	$HOME/.profile		○		
콘 셸	/etc/profile	$HOME/.profile		○		
		$HOME/.kshrc	ENV 변수 설정	○	○	
C 셸	/etc/.login	$HOME/.login		○		
		$HOME/.cshrc		○	○	
		$HOME/.logout				○

7. 리눅스 접근권한 관리

리눅스는 다중이용자를 위한 시스템이기 때문에 이런 것들이 필요함.

1. 파일의 속성

ls -l 명령으로 확인 가능.

시스템 관련 파일들은 root 가 소유하고 있고, 일반 파일은 파일을 생성한 사람이 소유자임.

ls -l 명령어로 출력되는 그룹명은 파일이 속한 그룹임. 사용자가 속한 그룹은 groups 명령어로 확인할 수 있음
사용자의 그룹은 시스템 관리자만이 변경할 수 있음.
해당 그룹에 있는 사용자들은 권한을 받아 파일을 공유할 수 있음.

파일의 크기는 바이트 단위로 명시됨.

메모 포함[이118]: 시스템 관리자가 사용자를 등록할 때 결정함.

표 5-1 파일의 속성

번호	속성 값	의미
❶	–	파일의 종류(–: 일반 파일, d: 디렉터리)
❷	rw–r––r––	파일을 읽고 쓰고 실행할 수 있는 접근 권한 표시
❸	1	하드 링크의 개수
❹	root	파일 소유자의 로그인 ID
❺	root	파일 소유자의 그룹 이름
❻	223	파일의 크기(바이트 단위)
❼	11월 8 23:13	파일이 마지막으로 수정된 날짜와 시간
❽	/etc/hosts	파일명

2. 접근권한

1) 접근권한의 종류

일반 파일과 디렉토리 각각에서 r, w, x 권한의 의미.

메모 포함[이119]: read write execution.

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.

(ls 명령으로 디렉토리 목록을 볼 수 있다는 것은, ls -l 로 보이는 세부정보를 본다는 게 아니라,
디렉토리 내에 어떤 파일이 있는지 본다는 것. ls test 이렇게 사용해서.)

2) 접근권한 표기법

기호로 표기

접근권한을 9 자리 문자로 나타냄.

사용자 카테고리별로 rwx 씩 묶어서 표기.
사용자에는 순서대로 파일 사용자(user), 그룹 사용자(group), 기타 사용자(others)가 있음.
해당 권한이 있는 경우 알파벳으로, 없는 경우 -로 표기.

숫자로 표기

접근권한을 세 자리 8진수로 나타냄.
원래는 4자리지만 맨 앞자리는 특수 접근권한을 나타내므로 일반적으로는 취급하지 않음.

접근권한을 숫자로 작성하는 방법:
첫째, 기호로 접근권한을 표기함.
둘째, 각 접근권한이 존재하면 1로, 존재하지 않으면 0으로 나타내서 이진수로 변환함.
셋째, 해당 이진수를 8진수로 바꾸어 작성. (결국 3글자로 표현됨.)

메모 포함[이120]: 진법의 변환에 정리되어 있는 방식대로,
8진수->2진수
8진수를 1자리씩 끊어서 2진수로 변환함.
2진수->8진수
2진수를 3자리씩 끊어서 8진수로 변환함.

메모 포함[이121]: 이진수 3자리는 3비트 정보이므로, 8진수 한 자리의 크기와 정확히 맞아떨어짐.

(ex. rwxrw-r--은 111 110 100 이므로, 764 로 표현됨)

3. 접근권한 변경

chmod <option> <mode> <file>

chmod 명령어를 사용함.

일반 사용자는 다른 사용자가 소유한 파일의 접근권한을 수정할 수 없음.

mode 에는 기호 또는 숫자로 모드를 작성함.

1) 기호 모드

오른쪽 표의 기호를 사용해서 작성.
"사용자+설정+권한"의 형식으로, 필요한 만큼 작성.
여러 종류의 사용자들의 권한 수정 시에는 쉼표 사용
사용자끼리, 권한끼리 적는 순서는 상관이 없음.
설정+권한을 여러 개 한 번에 적을 수 있음
쉼표로 이어서 적을 때 띄어쓰기 하면 오류남.
(ex. chmod g-r+xw,o+w test)

기호 분류	기호	의미
사용자	u	파일 소유자
	g	그룹 사용자
	o	기타 사용자
	a	모든 사용자
권한	r	읽기
	w	쓰기
	x	실행
설정	+	권한 추가
	-	권한 삭제
	=	권한 배정

메모 포함[이122]: 권한 배정은 작성한 권한을 그대로 부여하는 것

2) 숫자 모드

숫자로 표기한 접근권한을 모드에 명시할 수 있음.

접근권한의 일부만 수정할 때에도 세 자리 모두를 명시해야 함.

메모 포함[이123]: 한 자리만 명시한 경우 의도되지 않은 결과가 발생할 수 있음.
(ex. 4만을 입력하면 400이 아니라 004로 인식됨)

해당 접근권한으로 접근권한이 지정됨.

(ex. chmod 754 test)

4. 기본 접근권한

umask <option> <마스크 값>

파일, 디렉토리 생성 시에 자동적으로 설정되는 접근권한.

일반파일의 경우, default 값은 664 임. (0002 mask 가 씌워져 있는 값)
디렉토리의 경우, default 값은 775 임. (0002 mask 가 씌워져 있는 값)

메모 포함[이124]: 이 default 값은 시스템마다 다른 듯?
책과 내 우분투의 default값이 다름.
여기에는 책에 적힌 것을 작성해 뒀음.

기본 접근권한을 변경하려면 umask 명령어를 사용.

마스크 값은 기본적으로 4 글자인데, 여기서 맨 앞 글자는 특수 접근권한을 의미하고 뒤에 3 글자가 실질적인 마스크 값임.

5. 특수 접근권한

숫자로 표기한 접근권한은 원래 4 자리인데, 보통 생략되는 맨 앞자리는 특수 접근권한을 의미함.
맨 앞자리가 0 이 아닌 경우, 특수 접근권한이 설정되어 있는 것.
특수 접근권한을 설정하면 지정된 실행 권한 기호가 변환됨.

맨 앞자리가 0 이면, 일반적인 접근 권한. 이 경우, 0 을 생략하여 작성하기도 함.
맨 앞자리가 1 이면, 스티키 비트 (sticky bit)
맨 앞자리가 2 이면, SetGID (Group ID)
맨 앞자리가 4 이면, SetUID (User ID)

1) SetUID (User ID) (파일)

해당 파일이 실행되는 동안에는 파일을 실행한 사용자가 일시적으로 파일 소유자의 권한을 가지게 하는 특수 접근권한.

SetUID 가 설정되면 해당 파일의 접근권한 중, 소유자의 실행 권한에 x 대신 s 가 생김. (Set 의 s, user 에.)

(ex. passwd 명령의 실행 파일은 SetUID 가 설정되서 있어, passwd 명령을 사용하면 이 파일의 소유자인 root 의 권한을 사용하여 암호를 수정할 수 있음)

2) SetGID (Group ID) (파일)

해당 파일이 실행되는 동안에는 파일을 실행한 사용자가 일시적으로 파일 소유자의 그룹 권한을 가지게 하는 특수 접근권한.

SetGID 가 설정되면 해당 파일의 접근권한 중, 그룹의 실행 권한에 x 대신 s 가 생김. (Set 의 s, group 에.)

3) 스티키 비트 (디렉토리)

디렉토리에 설정하는 특수 접근권한.
해당 디렉토리에 누구나 파일을 생성할 수 있게 함.
파일을 만든 사람이 소유자가 되고, 소유자를 제외한 사용자는 파일은 삭제할 수 없음. 본인이 생성한 파일만 삭제할 수 있음.

스티키 비트가 설정되면 해당 디렉토리의 접근권한 중, 기타 사용자의 실행 권한에 t 가 붙음. (sticky 의 t 인듯)

(ex. /tmp 디렉토리.)

메모 포함[이125]: umask에서는 다룰 수 없는 개념임?

메모 포함[이126]: 특수 접근권한을 설정할 일반 파일이나 디렉토리는 s또는 t가 들어가는 자리에 실행 권한을 가져야 한다.
(해당 실행권한이 변하는 것으로 이해하자. 변하려면 우선 존재해야 한다.)

실행권한이 없는 것에 특수 접근권한을 설정하면 s 대신 S, 또는 t 대신 T가 표시된다. 오류가 발생했다는 표시이다.

실행 권한을 미리 설정하지 않아도 해당 자리에 실행 권한이 존재하기만 하면 되므로,
chmod 4777 file처럼 실행권한을 주면서 특수 접근권한을 설정할 수 있다.

메모 포함[이127]: chmod를 이용해 설정할 수 있음

메모 포함[이128]: 445처럼 3자리로 작성한 경우에는 맨 앞자리가 0인 것.

메모 포함[이129]: 이를 악용한 해킹도 존재한다고 함. root의 권한을 가져오는 등.

메모 포함[이130]: 수정도 안 되는 거 같기는 함.
읽기, 실행은 되는 거?
(될 거 같기는 함.)

메모 포함[이131]: 그럼에도 root는 다 할 수 있음.

3. 리눅스 프로세스 관리

1. 리눅스 프로세스 관리

1. 프로세스

현재 시스템에서 실행 중인 프로그램
리눅스는 기본적으로 다중 프로세스 시스템임.
사용자가 입력한 명령은 프로세스가 되어 실행됨.

2. 부모-자식 관계

리눅스의 모든 프로세스는 부모-자식 관계를 가지고 있음.
부모 프로세스가 자식 프로세스를 생성하면 자식 프로세스는 명령을 수행하고 결과를 리턴한 후 사라짐.
자식 프로세스는 또 다른 자식 프로세스를 생성할 수 있음.

1 번 프로세스(systemd)는 모든 시스템 프로세스의 부모 프로세스임.
2 번 프로세스(kthreadd)는 모든 스레드의 부모 프로세스임.

메모 포함[이132]: 부팅할 때 스케줄러가 실행한 1번(systemd)과 2번(kthreadd) 프로세스를 제외하면 모든 프로세스는 부모 프로세스를 가지고 있음.

부모-자식 관계는 모든 프로세스가 가지고 있음.

메모 포함[이133]: c언어의 함수 느낌이다.

3. 프로세스의 번호

각 프로세스는 고유한 번호인 PID 를 가짐.

PID 는 1 번부터 시작하고 프로세스가 생성되면 숫자가 하나씩 증가되어 부여됨.

1 번 프로세스는 systemd 이고, 2 번 프로세스는 kthreadd 임.

+ 부모 프로세스의 ID 는 PPID 라 함. (Parent Process ID)

+ 셸도 하나의 프로세스임.

4. 특수한 프로세스들

잠깐 실행되었다가 종료되는 일반적인 프로세스와는 달리, 특수한 프로세스들이 존재함.

1) 데몬 프로세스

커널이 실행하는 프로세스

항상 실행되며 대기 상태로 있다가 요청이 들어오면 특정 서비스를 지원해줌. (그림자 같은 존재)

2) 고아 프로세스

자식 프로세스의 부모 프로세스가 없어지면 그 자식 프로세스는 고아 프로세스가 됨.

1 번 프로세스가 고아 프로세스의 새로운 부모 프로세스가 되어 고아 프로세스의 작업 종료를 지원함.

길을 잃은 파일들은 lost+found 디렉토리로 가기도 함.

3) 좀비 프로세스 (defunct 프로세스)

자식 프로세스가 종료 후에도 프로세스 테이블 목록에 남아 있는 자식 프로세스.

부모 프로세스가 자식 프로세스의 종료 정보를 제대로 처리하지 않아서 만들어짐.

일반적인 경우, 자식 프로세스는 종료될 때 종료 정보를 보냄.

종료 정보를 받으면 부모 프로세스는 자식 프로세스를 프로세스 작업 테이블 목록에서 삭제함.

좀비 프로세스는 실행되지는 않지만 처리하지 않으면 프로세스 작업 테이블을 차지하면서 정상적인 프로세스의 실행을 방해할 수 있음.

kill 명령으로 제거 불가능.

SIGCHLD 시그널을 부모 프로세스에 보내서 자식 프로세스를 정리하게 해서 해결할 수 있음.

부모 프로세스 자체를 종료하여 해결할 수 있음.

메모 포함[이134]: 1번 프로세스에서 고아 프로세스를 관리하지 않으면 여러 자원을 차지하면서 시스템의 성능을 저하시킬 수 있음.

메모 포함[이135]: 1번 프로세스는 주기적으로 자식 프로세승의 종료 정보를 확인하여 정리한다.

메모 포함[이136]: defunct : 현존하지 않는.

메모 포함[이137]: 부모 프로세스가 종료되면 좀비 프로세스는
고아 프로세스가 되고 1번 프로세스가 부모가 된다.

5. 프로세스 관리 명령

1) 프로세스 목록 확인

ps 명령어를 이용해서 확인 가능.

2) 특정 프로세스 확인

ps | grep 을 이용해서 확인 가능.
ps 명령어에 옵션을 붙여서 확인 가능.
pgrep 명령어를 이용해서 확인 가능.

3) 프로세스 종료

kill, pkill 명령어를 이용해서 시그널 전송 가능.

4) 프로세스 관리 도구

프로세스 정보를 가공하여 보여줌.

1. top 명령어

2. 시스템 감시

GNOME 에서 제공하는 도구.
프로세스 관련 정보들을 보여줌.

6. 포그라운드, 백그라운드

1) 포그라운드 프로세스 (전위)

처리가 끝날 때까지 기다려야 하는 포그라운드 방식으로 처리되는 프로세스
포그라운드 프로세스가 끝날 때까지 프롬프트가 출력되지 않음.

2) 백그라운드 프로세스 (후위)

처리가 끝나지 않아도 다른 작업을 할 수 있는 백그라운드 방식으로 처리되는 프로세스

&는 백그라운드 작업을 의미함.
명령을 백그라운드로 실행하고 싶으면 명령의 마지막에 &기호를 추가하면 됨.
(ex. sleep 100 &)

백그라운드 프로세스가 끝나면 예기치 않게 결과가 화면에 출력될 수 있으므로 백그라운드 프로세스의 결과는 주로 출력의 방향을 파일로 리다이렉션함.

메모 포함[이138]: 백그라운드 작업을 제어하기 위해서는 kill 명령어를 이용해서 signal 을 보내야만 하는가?

메모 포함[이139]: 백그라운드 프로세스 실행 방법
1. 명령 작성 시에 & 기호를 붙인다.
2. 포그라운드 프로세스를 작업 전환한다.

7. 작업 제어

작업 전환, 작업 일시 중지, 작업 종료를 통틀어 작업 제어라고 함.

작업 제어 도구가 관리하는 프로세스를 작업이라고 함.

작업 또한 프로세스이기 때문에 PID 를 가짐.

하나의 터미널에서 여러 개의 프로세스를 운용하기 위한 기능임.

작업은 작업 번호를 가짐. 작업 번호는 항상 %<번호>로 명시함.

1) 작업 목록

jobs 명령어로 볼 수 있음.

2) 작업 전환, 작업 일시 중지

작업 전환: 포그라운드 작업, 백그라운드 작업을 서로 전환하는 것.
작업 일시 중지: 일시적으로 작업을 멈추는 것.

우선 포그라운드 작업을 정지해야 백그라운드로 전환할 수 있음.

작업 전환, 작업 일시 중지 명령어들

명령	기능
Ctrl+z 또는 stop [%작업 번호]	포그라운드 작업을 정지한다(종료하는 것이 아니라 잠시 중단하는 것이다).
bg [%작업 번호]	작업 번호가 지시하는 작업을 백그라운드 작업으로 전환한다.
fg [%작업 번호]	작업 번호가 지시하는 작업을 포그라운드 작업으로 전환한다.

작업 번호를 명시하지 않고 bg, fg 만 입력할 경우 작업 순서가 +인 작업에 적용됨.

4) 작업 종료

가. 포그라운드 작업

ctrl + c 로 대부분 종료 가능.

다른 터미널에서 해당 프로세스의 PID 를 찾아 강제로 종료 가능.

나. 백그라운드 작업

kill 또는 pkill 명령으로 종료할 수 있음.

로그아웃하면 다 종료됨.

로그아웃 후에도 백그라운드 작업을 수행하게 하려면 nohup 명령어를 사용해야 함.

5) 작업 예약

at, crontab 등의 명령어로 작업을 예약할 수 있음.

메모 포함[이140]: ctrl + z 와 ctrl + c 구분 유의하기. ctrl + z 는 종료가 아니라 정지이다.

메모 포함[이141]: ctrl + c 는 인터럽트 시그널(2 번 시그널)을 포그라운드 프로세스에 보냄.

2. 데몬 프로세스

리눅스의 백그라운드에서 동작하면서 특정 서비스를 제공하는 프로세스.

1. 동작 방식

독자형과 슈퍼 데몬에 의한 동작 방식이 있음

1) 독자형

데몬 혼자 스스로 동작하는 방식.

시스템의 백그라운드에서 항상 동작함.

해당 데몬이 자주 쓰이지 않는 경우에는 시스템 자원이 낭비됨.

2) 슈퍼 데몬에 의해 동작하는 방식

데몬을 관리하는 슈퍼 데몬에 의해 동작하는 방식.

평소에는 슈퍼 데몬만 동작하다가 필요시 해당 슈퍼 데몬이 해당 데몬을 불러 사용함.

서비스 응답 시간이 길어질 수 있지만 자원을 효율적으로 사용함.

메모 포함[이142]: 항상 동작하는 것이 아니기 때문.

2. 슈퍼 데몬

데몬을 관리하는 데몬.

네트워크 서비스를 제공하는 데몬들을 관리함. -> 텔넷 서버 등
우분투의 슈퍼 데몬의 이름은 xinetd 임.

메모 포함[이143]: 유닉스에서는 inetd이다.
internet service daemon
inetd에 보안 기능이 포함된 것이 xinetd이다.

3. 조상 데몬

대부분의 데몬을 동작시키는 데몬

systemd 데몬과 커널 스레드 데몬이 있음.

1) systemd (system daemon)

1 번 프로세스.

프로세스 대부분의 조상 프로세스임.

2) 커널 스레드 데몬 (kthreadd, kernel thread daemon)

커널 데몬을 동작시키는 조상 데몬.

2 번 프로세스.

4. 주요 데몬들

표 8-4 리눅스의 주요 데몬

데몬	기능	데몬	기능
atd	특정 시간에 실행하도록 예약한 명령을 실행한다 (at 명령으로 예약).	smtpd	메일 전송 데몬이다.
		popd	기본 편지함 서비스를 제공한다.
crond	주기적으로 실행하도록 예약한 명령을 실행한다.	routed	자동 IP 라우터 테이블 서비스를 제공한다.
dhcpd	동적으로 IP 주소를 부여하는 서비스를 제공한다.	smb	삼바 서비스를 제공한다.
httpd	웹 서비스를 제공한다.	syslogd	로그 기록 서비스를 제공한다.
lpd	프린트 서비스를 제공한다.	sshd	원격 보안 접속 서비스를 제공한다.
nfs	네트워크 파일 시스템 서비스를 제공한다.	in.telnetd	원격 접속 서비스를 제공한다.
named	DNS 서비스를 제공한다.	ftpd	파일 송수신 서비스를 제공한다.
sendmail	이메일 서비스를 제공한다.	ntpd	시간 동기화 서비스를 제공한다.

+ 커널 데몬

커널의 일부분을 프로세스처럼 관리하는 데몬.

ps 명령어로 확인하면 대괄호([])로 묶여 있음.

4. 파일시스템과 디스크관리

1. 리눅스 파일시스템

1. 파일시스템

파일들을 구조적으로 관리하는 체계.
파일과 디렉토리를 관리하려면 파일시스템이 필요함.
구조에 따라 분류된 다양한 파일 시스템이 있음.

/proc/filesystmes 파일을 vi 로 확인하여 현재 시스템이 지원하는 파일시스템을 확인할 수 있음.

리눅스의 모든 파일시스템들은 기본적으로 유닉스 운영체제에서 유래된 공통개념을 바탕으로 함.
1) 파일은 inode 로 관리된다. -> inode 는 파일시스템의 핵심 요소임.
2) 디렉토리는 파일의 목록을 가지고 있는 파일일 뿐이다.
3) 특수 파일을 통해 장치에 접근할 수 있다.
4) 파일시스템은 파일 정보를 관리하는 자료구조를 가진다.

2. 파일시스템 종류

1) ext (ext1) (Extended File System, extfs)

MFS 의 기능을 확장한 파일시스템.
파일시스템의 최대 크기는 2GB 임.
파일 이름의 길이는 255 바이트까지 지원함.

메모 포함[이144]: 그래서 Extended File System이라고 함.

문제점.
inode 지원 안됨.
데이터 수정 시간 지원 안됨.
파일시스템이 복잡하고 파편화됨.

이런 문제점들 때문에 현재 리눅스에서는 ext 를 사용하지 않음.

2) ext2 (Second Extended File System. Ext2 fs)

ext 를 업그레이드하여 만들어진 파일시스템.
파일시스템의 최대 크기는 이론적으로 32TB 임. (초기에는 2TB 로 제한되었음)
ext3 이 나오기 전까지 리눅스의 표준 파일시스템으로 사용되었음.
매우 안정적이어서 현재에도 특정 부분들에 사용됨.

메모 포함[이145]: 초기에는 버그가 있었지만 사용이 증가하면서 현재는 매우 안정적인 파일시스템이 되었다.

3) ext3 (Third Extended File System)

ext2 를 업그레이드하여 만들어진 파일시스템.
파일시스템의 최대 크기는 블록 크기에 따라 2~32TB 임.
서브 디렉토리 개수 최대 320000 개.

메모 포함[이146]: ext2는 ext3와 호환된다.
ext2의 파일을 변경 없이 ext3에 바로 이식할 수 있다.

장점.

저널링 기능 도입

문제점.

최신 파일시스템 기능을 지원하지 않음. (inode 동적할당, 다양한 블록 크기 등의 기능들)
온라인 조각 모음 기능이 없음.

4) ext4 (Forth Extended File System)

ext3 을 업그레이드하여 만들어진 파일시스템.
파일시스템 최대 크기는 1EB 이상임.
16TB 이상의 파일 지원.

메모 포함[이147]: ext2, ext3와 호환성이 있다.

5) XFS (eXtended File System)

고성능 저널링 파일시스템.
실리콘 그래픽스가 개발하여 GNU GPL 로 공개됨.
대부분의 리눅스 배포판에서 지원함.
64 비트 파일시스템으로, 16EB 까지 지원함.

6) 기타

다른 운영체제의 시스템이나 외부 저장 장치를 위한 파일시스템들.

표 7-1 리눅스에서 지원하는 기타 파일 시스템

파일 시스템	기능
msdos	MS-DOS 파티션을 사용하기 위한 파일 시스템이다.
iso9660	CD-ROM, DVD의 표준 파일 시스템으로 읽기 전용으로 사용된다.
nfs	network file system으로 원격 서버의 디스크를 연결할 때 사용된다.
ufs	Unix file system으로 유닉스의 표준 파일 시스템이다.
vfat	윈도 95, 98, NT를 지원하기 위한 파일 시스템이다.
hpfs	HPFS를 지원하기 위한 파일 시스템이다.
ntfs	윈도의 NTFS를 지원하기 위한 파일 시스템이다.
sysv	유닉스 시스템V를 지원하기 위한 파일 시스템이다.
hfs	맥 컴퓨터의 hfs 파일 시스템을 지원하기 위한 파일 시스템이다.

7) 특수 용도의 가상 파일 시스템들

가상 파일 시스템: 디스크가 아니라 메모리에서 생성되어 사용되는 파일 시스템.

표 7-2 리눅스의 가상 파일 시스템

파일 시스템	기능
swap	• 스왑 영역을 관리하기 위한 스왑 파일 시스템이다. • 우분투 17.04부터는 스왑 파일 시스템 대신 스왑 파일을 사용한다.
tmpfs	• temporary file system으로 메모리에 임시 파일을 저장하기 위한 파일 시스템이며, 시스템이 재시작할 때마다 기존 내용이 없어진다. • /run 디렉터리를 예로 들 수 있다.
proc	• proc 파일 시스템으로 /proc 디렉터리이다. • 커널의 현재 상태를 나타내는 파일을 가지고 있다.
ramfs	• 램디스크를 지원하는 파일 시스템이다.
rootfs	• root file system으로 / 디렉터리이다. • 시스템 초기화 및 관리에 필요한 내용을 관리한다.

+ 저널링(journaling) 기능

강화된 데이터 복구 기능.
저널은 로그 기록을 의미함.

저널링에서는 데이터를 디스크에 기록하기 전에, 먼저 수정 사항을 저널에 기록함.
문제가 발생해서 디스크에 데이터를 기록하지 못했더라도 저널의 내용으로 빠른 복구가 가능함.

+ ext 는 리눅스 고유의 디스크 기반 파일시스템임.

+ MFS (Minix File System)

미닉스의 파일 시스템.

리눅스 초기에는 MFS 를 사용했음.
리눅스 고유의 파일시스템인 ext 를 개발하면서 MFS 를 사용하지 않게 됨.

3. ext4 파일시스템의 구조

ext4 파일시스템은 디스크를 논리적인 블록의 집합(블록 그룹)으로 구분함.
블록 하나의 크기는 4KB 가 default 임.

1) 블록 그룹 유형

블록 그룹 0: 파일시스템의 첫 번째 블록 그룹. 그룹 0 패딩, 슈퍼블록, 그룹 디스크립터 소유.
블록 그룹 a: 파일시스템의 첫 번째 블록 그룹이 아닌 블록 그룹. 슈퍼블록과 그룹 디스크립터의 복사본 소유.
블록 그룹 b: 파일시스템의 첫 번째 블록 그룹이 아닌 블록 그룹.

그림 7-3 ext4 파일 시스템의 구조

2) 블록들

그룹 0 패딩: x86 부트 섹터와 부가 정보를 저장하는 특별한 용도.

슈퍼블록: 파일시스템 관련 정보들 저장. 디스크에 대한 모든 정보들을 가지고 있음.

그룹 디스크립터: 여러 정보들 저장.

GDT 예약 블록: 그룹 디스크립터의 확장을 위한 예비 공간.

데이터 블록 비트맵: 블록 그룹에 포함된 데이터 블록의 사용 여부 확인, 비트맵에서 각 데이터 블록은 1 비트로 표현됨.

inode 비트맵: inode 테이블의 항목이 사용 중인지 표시. 비트맵에서 각 inode 테이블 항목은 1 비트로 표현됨.

inode 테이블: inode 번호 목록을 inode 테이블이라 하는 것? (그런 것 같기는 함)

데이터 블록: 실제 데이터 저장. 데이터 블록의 크기는 시스템 설정에 따라 1~8KB 까지 지정 가능.

메모 포함[이148]: a와 b는 백업된 것.
문제가 발생하면 복사본을 사용해서 복구한다.

메모 포함[이149]: 블록 그룹 0의 슈퍼블록과 그룹 디스크립터를
백업해 둔 것.

메모 포함[이150]: 뭔진 모르겠고 이런 것들이 있다 정도로만 암기하자.

메모 포함[이151]: 슈퍼블록에 문제가 생기면 전체 파일시스템을 사용할 수 없게 된다.
-> 다른 블록 그룹에 슈퍼블록을 복사해 두고, 문제가 발생하면 복사본을 사용해 복구한다.

메모 포함[이152]: 슈퍼블록과 그룹 디스크립터에 저장된 상세한
정보 목록은 교재 p.342, p.343에 나와 있다.

메모 포함[이153]: 일반 파일은 데이터 블록에 파일 내용을 저장한다.

디렉토리는 데이터 블록에 디렉토리 내의 파일, 서브 디렉토리의 정보(이름, inode)를 저장한다.

4. inode의 구조

inode는 크게 파일 정보를 저장하는 부분과, 실제 파일 내용이 저장되어 있는 데이터 블록의 주소를 저장하는 부분으로 나누어짐.

1) 파일 정보를 저장하는 부분

ls -l 명령으로 확인할 수 있는 정보들.

2) 데이터 블록의 주소를 저장하는 부분

직접 블록, 간접 블록, 이중 간접 블록으로 이루어져 있음.

직접 블록: 데이터 블록에 대한 주소를 직접 가지고 있음. (자주 쓰이는 것들)
간접 블록, 이중 간접 블록: 데이터 블록에 대한 주소를 직접 가지고 있는 블록의 주소를 가지고 있음. (자주 쓰이지 않는 것들)

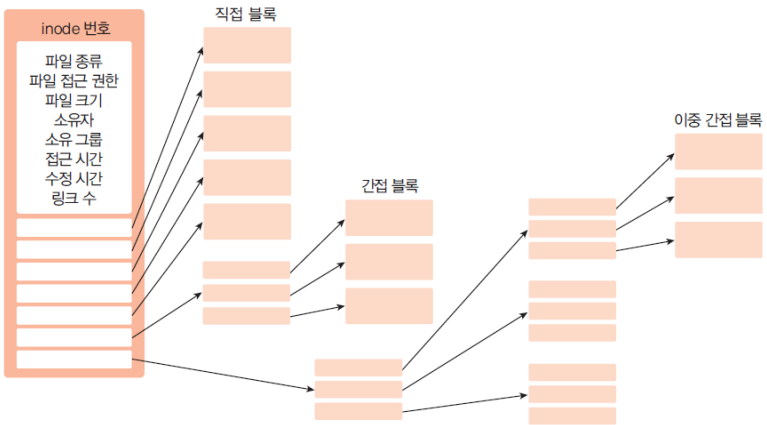


그림 7-4 inode의 구조

메모 포함[이154]: 모든 파일과 디렉토리는 하나의 inode를 가진다.

메모 포함[이155]: 즉, ls -l은 inode에 저장되어 있는 파일 정보를 출력하는 명령이다.

5. 파일 시스템과 계층 구조

실제 파일이 저장되어 있는 파일 시스템은 디렉토리 계층 구조에 연결(마운트)되어야 사용할 수 있음.

메모 포함[이156]: 파일들을 구조적으로 관리하는 체계를 파일시스템이라고 하는데, 왜 실제 파일이 저장되어 있지?
계층 구조를 가지는 디렉토리들을 포함하고 있는 파일시스템을 의미하는 것인가?

메모 포함[이157]: 디렉토리 계층 구조는 파일과 디렉토리를 정리하고 관리하는 구조이다.

1) 한 파일시스템

디렉토리 계층 구조의 파일들을 하나의 파일시스템에 두는 것.
/ 디렉토리에 해당 파일시스템을 연결(마운트)함.

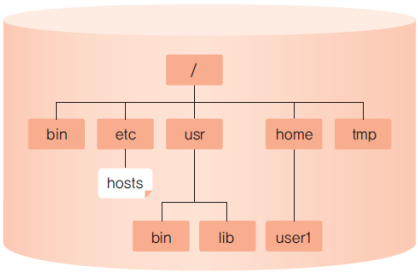


그림 7-5 한 파일 시스템으로 구성하기

2) 여러 파일시스템

디렉토리 계층 구조의 파일들을 여러 개의 파일시스템에 두는 것.
이 경우 일부 파일시스템에 문제가 생겨도 다른 파일시스템의 파일들은 안전함.

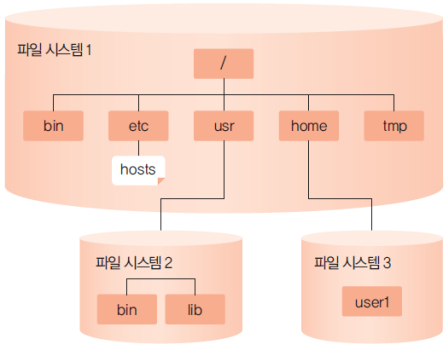


그림 7-6 여러 파일 시스템으로 구성하기

(파일시스템 2 는 /usr 에, 파일시스템 3 은 /home 에 연결되어 있음)

+ / 디렉토리에 연결된 파일 시스템을 루트 파일시스템이라고 함.

6. 파일시스템 생성

파티션에서 파일과 디렉토리를 관리하기 위한 구조를 만드는 과정.

mkfs 또는 mke2fs 명령어를 사용함.

2. 파일시스템 마운트

1. 마운트

파일시스템을 계층 구조의 특정 디렉토리와 연결하는 것.

마운트하지 않으면 사용자가 해당 계층 구조에 접근할 수 없음.

2. 마운트 포인트

디렉토리 계층 구조에서 파일시스템이 연결되는 디렉토리.

3. 파일시스템 마운트 설정 파일

마운트 설정 파일을 사용하여 시스템 부팅 시 파일시스템이 마운트되게 할 수 있음.

/etc/fstab 파일에 설정함.

1) /etc/fstab 파일 (filesystem table, 파일시스템 표)

시스템의 마운트 설정을 가지고 있는 파일.
이 파일에 오류가 있으면 부팅이 중지될 수도 있음.

이 파일을 수정하고 나면 다시 마운트 해줘야 적용됨.
-> sudo mount -o remount / 명령어로 이를 수행할 수 있음.

2) /etc/fstab 파일의 구조 (순서대로)

그림 7-7 /etc/fstab 파일의 구조

장치명: 파일시스템 장치 이름. 파일시스템이 구축된 물리적인 디스크를 지정하는 것.

파일시스템의 종류: ex2, ex3, ex4 등의 종류를 명시.

옵션: 아래 표에 나와 있듯이 여러가지 옵션이 있음.

덤프 관련 설정: 0 은 dump 명령 사용 불가, 1 은 dump 명령 사용 가능.

파일 점검 옵션: 0 은 파일시스템을, 1 은 루트 파일시스템을, 2 는 루트 파일시스템 이외의 파일시스템을 부팅 시에 fsck 명령으로 점검하지 않도록 함.

표 7-3 파일 시스템 속성 설정 옵션

속성	의미
defaults	일반적인 파일 시스템에 지정하는 속성이다. rw, nouser, auto, exec, suid 속성을 모두 포함한다.
auto	부팅 시 자동으로 마운트한다.
exec	실행 파일이 실행되는 것을 허용한다.
suid	setuid, setgid의 사용을 허용한다.
ro	읽기 전용 파일 시스템이다.
rw	읽기, 쓰기가 가능한 파일 시스템이다.
user	일반 사용자도 마운트가 가능하다.
nouser	일반 사용자의 마운트가 불가능하다. root만 마운트할 수 있다.
noauto	부팅 시 자동으로 마운트하지 않는다.
noexec	실행 파일이 실행되는 것을 허용하지 않는다.
nosuid	setuid, setgid의 사용을 금지한다.
usrquota	사용자별로 디스크 쿼터 설정이 가능하다.
grpquota	그룹별로 디스크 쿼터 설정이 가능하다.

메모 포함[이158]: 디스크 쿼터는 사용자별로 디스크를 할당하는 것.

+ USB 저장공간 사용

USB 는 데이터가 손상되는 것을 막기 위해 가능한 모든 저장공간을 제공하지 않음.
(ex. 8 기가짜리 USB 는 7.5 기가만 사용할 수 있음)

4. 마운트 관련 명령

mount <option> <장치명> <마운트 포인트>>

1) mount 명령어

장치와 디렉토리를 연결할 때에는 mount 명령어를 사용함.

umount <option> <장치명 or 마운트 포인트>

2) umount 명령어

장치와 디렉토리의 연결을 해제할 때에는 umount 명령어를 사용함.

장치와 디렉토리의 연결을 해제하는 것을 언마운트라고 함.

5. mount 명령을 이용한 장치 연결법

1) 리눅스용 USB 마운트

1. USB 를 꽂고 리눅스 시스템에 인식시킨다.
2. fdisk -l (소문자 엘) 명령어로 USB 메모리의 장치명을 확인한다.
3. fdisk 명령을 사용해서 USB 메모리에 리눅스 파티션을 생성한다.
4. 생성한 파티션을 포멧하여 파일 시스템을 생성한다. 파일시스템을 생성할 때는 mke2fs 명령을 사용한다.
5. USB 장치를 마운트한다.

사용이 끝났으면 umount 명령어로 장치 연결을 제거한 후 USB 를 슬롯에서 뽑는다.

2) 윈도우용 USB 마운트 -> 바로 마운트

1. USB 를 꽂고 리눅스 시스템에 인식시킨다.
2. fdisk -l (소문자 엘) 명령어로 USB 메모리의 장치명을 확인한다.
3. 마운트한다. -> fdisk -l 를 사용했을 때 확인할 수 있는 포맷 종류를 명시해 줘야 한다.

사용이 끝났으면 umount 명령어로 장치 연결을 제거한 후 USB 를 슬롯에서 뽑는다.

3) 외장 CD-ROM 마운트 -> 바로 마운트

1. CD-ROM 장치를 연결한 후 리눅스 시스템에 인식시킨다.
2. fdisk -l (소문자 엘) 명령어로 CD-ROM 의 장치명을 확인한다. (/dev/cdrom 이 기본 장치명이다.)
3. 마운트한다.

사용이 끝났으면 umount 명령어로 장치 연결을 제거한다.

메모 포함[이159]: 여기에서는 하드디스크 마운트, 원격 디스크 연결, CD-ROM 마운트, USB 마운트 중에서 CD-ROM과 USB의 마운트를 정리한다.

하드디스크 마운트와 원격 디스크 연결은 뒤에 정리한다.

하드디스크, CD-ROM, USB 기본적인 연결 과정
1. 장치명 확인
2. 파티션 생성
3. 파일시스템 생성
4. 마운트

CD-ROM과 윈도우용 USB 연결 시에는 2, 3번 과정을 건너뛴다.

메모 포함[이160]: 윈도우용 USB는 파일시스템 종류가 FAT 또는 NTFS이다. fdisk -l에서 확인한 후 mount 명령어 사용 시에 -t 옵션으로 해당 파일시스템 종류를 작성해줘야 한다.

메모 포함[이161]: PC 본체에 장착되어 있는 내장 CD-ROM은 마운트가 필요하지 않다.

3. 디스크 관리

1. 가상머신의 하드디스크 구조

가상머신은 IDE, SATA, SCSI 로 구성되어 있음.

IDE 디스크 컨트롤러 2 개
SATA 컨트롤러 4 개
SCSI 디스크 컨트롤러 4 개.

IDE, SATA, SCSI 는 장치를 연결하는 인터페이스임.

오른쪽 그림에서는 SATA 컨트롤러에 CD 하나가, SCSI 에 여러 디스크들이 연결되어 있음.

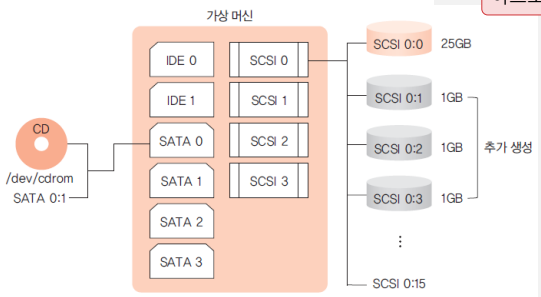


그림 7-13 가상 머신 디스크 추가 구성의 개념

메모 포함[이162]: 가상머신은 실제 머신을 구현한 것이므로 구조에 있어서 실제 머신과 큰 차이는 없다.

2. 디스크 장치의 이름

디스크와 관련된 작업을 진행하려면 디스크 장치의 이름을 알아야 함.

1) 이름 규칙

/dev/sd 로 시작하는 이름을 사용함.
컨트롤러에 연결되는 순서에 따라 이름에 알파벳을 붙임.
파티션은 이름에 숫자를 추가로 붙임.

메모 포함[이163]: ex.
/dev/sda -> 첫 번째 디스크
/dev/sdb -> 두 번째 디스크
/dev/sdc -> 세 번째 디스크

메모 포함[이164]: ex.
/dev/sda -> 첫 번째 디스크 전체를 의미
/dev/sda1 -> 첫 번째 파티션
/dev/sda2 -> 두 번째 파티션
/dev/sda3 -> 세 번째 파티션

2) 이름 확인

fdisk -l 명령어로 장치의 이름 확인 가능.

장치 이름은 여러 명령에 사용되므로 이를 확인하는 것이 중요함.

3. 추가 디스크 설치

1) 추가 디스크 설치 방법

1. 새 디스크를 장착한다. (가상머신에서는 특정 설정 사용)
2. 디스크의 파티션을 생성한다. (fdisk 명령어 사용, n 으로 만들고 w 로 저장, 시작 섹터는 기본값으로 함)
3. 파티션을 포맷하여 파일시스템을 생성한다. (mkfs 또는 mke2fs 명령어로 파일시스템 생성)
4. 마운트한다. (mount 명령어 사용. /mnt 에 hdd 등의 이름으로 마운트)

메모 포함[이165]: 가상머신에서는 설정을 조작해서 장착할 수 있다. 실제 머신에서는 하드웨어를 장착한다.

메모 포함[이166]: 파일시스템 생성 == 포맷인 것 같다.

4. 파티션

fdisk <option> <장치명>

하나의 디스크를 독립된 영역으로 구분하는 것.

하나의 디스크 전체를 하나의 파티션으로 취급할 때도 파티션 작업은 반드시 수행해야 함.

파티션 작업은 fdisk 명령을 이용함.

파일시스템 마운트 시에도 사용함.

메모 포함[이167]: 윈도우에서 하나의 디스크를 C 드라이브, D 드라이브 등으로 나누어 사용하는 것과 같은 개념이다.

5. 디스크 마운트

마운트 포인트를 준비하고, mount 명령어로 마운트함.

이때 마운트하려는 파일시스템의 종류에 맞춰서 -t 옵션으로 파일시스템을 지정해줌.
(반드시 지정해야 하는 것은 아님)

6. LVM

독립된 디스크 파티션들을 하나로 연결해서 사용할 수 있게 해 주는 관리 도구.

1) 관련 용어

PV (Physical Volume) : 실제 하드디스크의 파티션. (ex. /dev/sdb1, /dev/sdb2)
VG (Volume Group) : 여러 개의 PV 를 묶은 것.
LV (Logical Volume) : VG 를 다시 적절한 크기의 파티션들로 나눈 각 파티션.
PE (Physical Extent) : PV 가 가진 일정한 블록.
LE (Logical Extent) : LV 가 가진 일정한 블록.

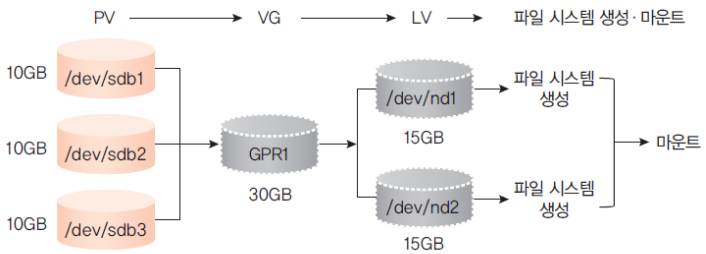


그림 7-21 LVM의 기본 개념

2) 관련 명령들

표 7-6 LVM 관련 명령

구분	기능	명령
PV	PV 생성	pvcreate 파티션 이름
	PV 상태 확인	pvscan
VG	VG 생성	vgcreate VG명 파티션(PV)명1 파티션(PV)명2…
	VG 활성화	vgchange –a y VG명
	VG 비활성화	vgchange –a n VG명
	VG 삭제	vgremove VG명
	VG 정보 확인	vgdisplay –v VG명
	VG에 PV 추가	vgextend VG명 PV명
	VG에서 PV 삭제	vgreduce VG명 PV명
	VG명 변경	vgrename 기존 VG명 새 VG명
LV	LV 생성	lvcreate –l PE 수 VG명 –n LV명
	LV 삭제	lvremove LV명
	LV 상태 확인	lvscan
	LV 용량 확대	lvextent –l +PE 수 LV명
	LV 용량 축소	lvextent –l –PE 수 LV명

메모 포함[이168]: pvcreate
pvscan

vgcreate
vgchange
vgdisplay -> 해당 Volume Group 의 PV, VG, LV 에 대한 정보를 확인할 수 있음.

lvcreate

LVM 생성 시에 필수적인 것들 정도는 암기.

3) LVM 생성 과정

1. 새 디스크를 장착한다. (가상머신에서는 설정 조작)
2. 디스크의 파티션을 생성한다. (fdisk 명령어 사용, n 으로 만들고 w 로 저장, 시작 섹터는 기본값으로 함)
3. 우분투에 lvm2 패키지를 설치한다.
4. 각 파티션의 파일시스템 종류를 8(Linux)3 에서 8e(Linux LVM)로 변경한다.
(fdisk 명령어 사용. 명령어의 대상은 파티션이 속해 있는 디스크임. 내부 명령 t 사용. 변경 후 w 로 저장)
5. 각 파티션의 PV 를 생성한다. (pvcreate 명령어 사용) (pvscan 명령어로 목록 확인 가능)
6. PV 들을 통합하여 VG 를 생성한다.
7. 생성된 VG 를 활성화한다. (vgdisplay 명령어로 상태 확인 가능)
8. LV 를 생성한다. (lvcreate 명령어 사용.)
9. LV 에 파일시스템을 생성한다. (mkfs 명령어 또는 mke2fs 명령어 사용)
10. LV 를 마운트한다. (mount 명령어 사용)

메모 포함[이169]: 가상머신에서는 설정을 조작해서 장착할 수 있다. 실제 머신에서는 하드웨어를 장착한다.

메모 포함[이170]: ex.
sudo fdisk /dev/sdb1 이 아니라
sudo fdisk /dev/sdb 로 한 후 파티션 1로 들어가서 설정해야 한다.

메모 포함[이171]: 여기서는 하나의 LV를 생성하는 방법만 알려주는 것 같다.

그림 7-22 LVM 생성 단계

+ vgdisplay 명령어

해당 Volume Group 의 PV, VG, LV 에 대한 정보를 확인할 수 있음.

8. 디스크 관리

1) 디스크 사용량 확인

파일시스템의 정보를 확인할 때는 df 를 사용.
특정 디렉토리의 정보를 확인할 때는 du 를 사용.

2) 파일시스템 검사, 복구

가. fsck 명령어 또는 e2fsck 명령어로 파일시스템을 검사하고 복구할 수 있음.

나. badblocks 명령어로 장치의 배드 블록을 검사할 수 있음.

다. e2fsck 명령어로 백업 슈퍼블록을 이용해 파일시스템을 복구할 수 있음.
1. dumpe2fs 명령어로 백업 슈퍼블록의 위치 파악. (dumpe2fs <장치명> | grep superblock)
2. e2fsck 명령어로 복구. (sudo e2fsck -b <백업 슈퍼블록 위치> -y <장치명> 입력)

메모 포함[이172]: 이 부분 다시 읽어 보기

+ 배드 블록

디스크에 발생하는 배드 섹터를 검사하기 위한 기능. (정확히 뭔 지는 잘 모르겠음)

배드 블록으로 인한 데이터 유실은 디스크에 발생하는 심각한 문제임.

+ 슈퍼블록 삭제하기

sudo dd if=/dev/zero of=/dev/sdd1 bs=4096 count=20 입력.

파일시스템의 앞부분 20 블록의 값을 0 으로 채우는 것.

+ 백업 슈퍼블록

기본 슈퍼블록에 문제가 있을 경우 파일시스템을 사용할 수 없음.
이때 백업 슈퍼블록을 사용하여 파일시스템을 복구할 수 있음..

5. 리눅스의 부팅과 종료

1. 리눅스 시스템의 부팅

1. 부팅

PC를 켜는 순간부터 로그인 프롬프트가 출력될 때까지의 과정.

크게 PC 부팅(하드웨어)과 리눅스 부팅(OS)으로 나뉨.

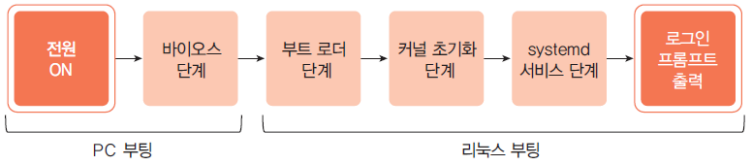


그림 8-2 리눅스의 부팅 과정

2. 바이오스 단계

바이오스(BIOS, Basic Input/Output System)가 작업을 수행하는 단계.

바이오스가 하는 일. -> 상태확인, 부트로더 로드

1. PC 하드웨어의 상태를 확인한다.
2. 부팅 장치를 선택한다.
3. 부팅 디스크의 첫 섹터에서 512 바이트(MBR)를 메모리에 로딩한다.
4. 부트 로더를 찾아 메모리에 로딩한다.

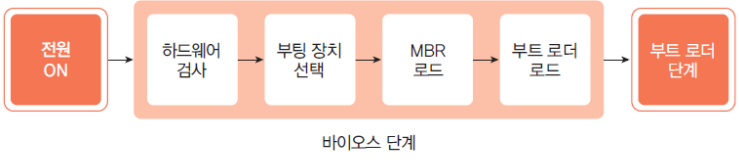


그림 8-3 바이오스 단계의 세부 동작

메모 포함[이173]: 주로 ROM에 저장되기 때문에 ROM-BIOS라고도 부른다.

메모 포함[이174]: 이 512B를 '마스터 부트 레코드(MBR, Master Boot Record)'라고 한다.

MBR에는 디스크의 어느 파티션에 부트 로더(2차 부팅 프로그램)가 있는지 저장되어 있다.

2. 부트 로더 단계

부트 로더가 작업을 수행하는 단계.

리눅스 커널을 메모리에 로딩함.

3. 커널 초기화 단계

커널이 작업을 수행하는 단계.

1. 커널이 기본적인 초기화 작업을 수행한다.
(ex. 시스템에 연결된 장치들을 검사한다.)

2. 커널 프로세스들을 생성한다. -> 커널의 여러 작업을 수행한다.

4. systemd 서비스 단계

systemd 서비스가 작업을 수행하는 단계.

1. systemd 서비스가 다양한 서비스를 동작 시킨다.
2. 각 서비스가 시작하는 과정이 화면에 출력된다. 우분투에서는 이를 부트 스플래시로 가린다.
3. 서비스가 정상적으로 시작하는지에 대한 메시지들이 출력된다.

5. 로그인 프롬프트 출력

systemd 서비스 단계에서 데몬을 모두 실행하면 GDM 을 동작 시킴.

+ 커널 프로세스

fork를 사용하지 않고 만든 프로세스와 스레드

1. 일반적인 프로세스들과 구분하여 대괄호([])로 표시한다.
2. 주로 PID가 낮다.
3. 개수와 종류가 리눅스의 버전과 종류에 따라 다르다.

메모 포함[이175]: 그런데 systemd보다 이게 순서가 먼저이면
kthreadd가 커널 데몬들을 생성하니까
kthreadd가 1번 프로세스가 되는 것 아닌가?₩

커널 데몬과 커널 프로세스가 다른가?

둘 다 []로 묶기는 하는데..

메모 포함[이176]: 리눅스가 본격적으로 동작하기 시작하는 단계이다.

메모 포함[이177]: 이 메시지를 확인하는 방법

알 필요가 있을까?

1. 부팅할 때 출력되게 하는 방법
/etc/default/grub 파일에서
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
를 찾아 quiet를 지운다.
sudo update-grub 명령을 입력한다.

2. dmesg 명령 입력
-> 데몬의 시작 뿐만 아니라 하드웨어 검사와 관련된 내용도 출력된다.

3.. more /var/log/boot.log

+ fork

일반적으로 프로세스를 만드는 방식.

+ 부트 스플래시

우분투 시작 시에 나오는 화면

그림 8-5 우분투의 부트 스플래시 화면

+ GDM (GNOME Display Manager)

그래픽 로그인 시스템.

2. systemd 서비스

메모 포함[이178]: system daemon

1. init 프로세스

스크립트를 순차적으로 실행하여 다른 프로세스를 동작시키는 프로세스.

메모 포함[이179]: 쉘 스크립트를 말하는 것 같다.

init 과 관련된 스크립트 파일들은 /etc/init.d 디렉토리에 들어 있음.

2. 런레벨

시스템의 상태를 나타내는 한 자리 숫자. (문자 S, s 포함)

init 프로세스는 시스템을 7 개의 단계로 구분하고 각 단계별로 특정 셸 스크립트를 실행함.
이 단계가 런레벨임.

systemd 에서도 사용됨.

런레벨 별로 실행하는 스크립트 파일은 /etc/init.d 디렉토리에 있는 파일에 대한 심볼릭 링크임.

각 런레벨 별 의미

표 8–1 우분투의 런레벨

런레벨	의미	관련 스크립트의 위치
0	시스템 종료	/etc/rc0.d
1, S	응급 복구 모드(단일 사용자 모드)	/etc/rc1.d, /etc/rcS.d
2	다중 사용자 모드	/etc/rc2.d
3		/etc/rc3.d
4		/etc/rc4.d
5	그래피컬 다중 사용자 모드	/etc/rc5.d
6	재시작	/etc/rc6.d

('관련 스크립트의 위치'에 있는 파일들은 심볼릭 링크임)

+ upstart

init 이전에 사용하던 것.

리눅스에서 자체적으로 개발함.

현재는 기본적으로 설치 되어있지 않음.

upstart 와 관련된 몇몇 스크립트 파일들은 /etc/init 디렉토리에 *.conf 형식으로 들어 있음.

---

3. systemd

리눅스의 시스템과 서비스 관리자.

기존의 init 스크립트를 대체한 것.
init 프로세스가 하던 작업들을 수행함.

1 번 프로세스임. (대부분의 프로세스의 조상 프로세스)

메모 포함[이180]: ps -ef 명령어로 확인했을 때 1 번 프로세스가 여전히 init 이라고 뜰 수 있는데, 이는 systemd 의 심볼릭 링크이다.

1) init 과 비교했을 때 systemd 의 장점

1. 소켓 기반으로 동작하여 inetd 와 호환성을 유지한다.
2. 셸과 독립적으로 부팅이 가능하다.
3. 마운트 제어가 가능하다.
4. fsck 제어가 가능하다.
5. 시스템 상태에 대한 스냅숏을 유지한다.
6. SELinux 와 통합이 가능하다.
7. 서비스에 시그널을 전달할 수 있다.
8. shutdown 전에 사용자 세션의 안전한 종료가 가능하다

2) 유닛

systemd 가 시스템을 시작하고 관리할 때 사용하는 구성 요소.

systemd 는 관리 대상의 이름을 "<서비스명>.<유닛 종류>" 형태로 함.

각 유닛은 자신과 동일한 이름의 설정 파일을 가짐.

각 유닛에 대한 세부 정보는 man 명령어를 사용해 확인할 수 있음.

표 8-2 systemd 유닛의 종류

유닛	기능	예
service	가장 명백한 유닛으로 데몬을 시작·종료·재시작·로딩한다.	atd.service
socket	소켓을 관리하는 유닛으로 AF_INET, AF_INET6, AF_UNIX 소켓 스트림과 데이터그램, FIFO를 지원한다.	dbus.socket
device	리눅스 장치 트리에 있는 장치를 관리한다.	sys-module-fuse.device
mount	디렉터리 계층 구조의 마운트 포인트를 관리한다.	boot.mount
automount	디렉터리 계층 구조에서 자동 마운트 포인트를 관리한다.	proc-sys-fs-binfmt_misc.automount
target	유닛을 그루핑한다(예: multi-user.target→런레벨 5에 해당하는 유닛).	basic.target multi-user.target
swap	스왑 장치를 관리한다.	dev-mapper-fedora\x2dswap.swap
path	경로를 관리한다.	cups.path
timer	타이머와 관련된 기능을 관리한다.	dnf-makecache.timer
slice	프로세스 그룹의 자원을 계층적으로 관리한다.	system-getty.slice
scope	외부에서 생성된 프로세스를 관리한다.	init.scope

메모 포함[이181]: systemd 는 유닛들을 구분하여 관리한다.

3) systemd 와 런레벨

systemd 는 런레벨에 따른 target 유닛을 가짐.

심볼릭 링크는 편의성을 위해 존재함.

이 파일들은 /lib/systemd/system 디렉토리에 들어 있음.

표 8-3 런레벨과 target 유닛의 관계

런레벨	target 파일(심벌릭 링크)	target 원본 파일
0	runlevel0.target	poweroff.target
1	runlevel1.target	rescue.target
2	runlevel2.target	multi-user.target
3	runlevel3.target	
4	runlevel4.target	
5	runlevel5.target	graphical.target
6	runlevel6.target	reboot.target

관련 명령들.

현재 target 확인 -> systemd get-default

현재 런레벨 확인 -> runlevel

기본 target 지정 -> systemctl set-default <타겟 이름>.target

런레벨 변경 -> systemctl isolate <타겟 이름 또는 원본 파일 이름>, init <런레벨>, telinit <런레벨>

단일 사용자 모드로 전환 (런레벨을 1 로 변경) -> 런레벨 변경 명령 사용.

다중 사용자 모드로 전환 -> reboot, systemctl default

+ target 이란? 기본 타겟 지정?

4) systemctl 명령어 (systemd control)

이 명령어로 systemd 를 제어함.

```
systemctl <option> <명령> <유닛명>
```

메모 포함[이182]: 유닛 종류 중 하나.
유닛을 그루핑하기 위한 것.

메모 포함[이183]: 이 명령은 심볼릭 링크인 default.target이
가리키는 타겟 파일을 뒤에 명시한 파일로
변경한다.

메모 포함[이184]: 말 그대로 이름만 작성하면 된다.
유닛 종류는
명시할 필요 없다.

메모 포함[이185]: 유닛을 그루핑한다고 하는데, 이것의 의미가
여러 유닛들을 몇 가지 그룹으로 나눈 후
특정 런레벨에는 특정 그룹에 있는 유닛들만
지정할 수 있게 하는 것인가?
-> 책 다시 읽어보기.

target은 해당 런레벨에 실행하는 스크립트를 의미하는 것인가?

3. 리눅스 시스템의 종료

리눅스에서 시스템을 종료하는 방법들.

1. shutdown 명령 사용

가장 정상적으로 리눅스 시스템을 종료하는 방법임.

```
shutdown <option> <time> <메시지>
```

2. 런레벨 변경

런레벨을 0 으로 변경하면 시스템이 종료됨.
런레벨을 6 으로 변경하면 시스템이 재시작됨.

3. reboot, halt, poweroff 명령 사용

/var/log/wtmp 파일에 시스템 종료 기록을 남기고 시스템을 종료 또는 재시작하는 명령어들.

이 명령들은 런레벨이 1~5 일 때 내부적으로 shutdown 명령을 호출함.

옵션들.
-n : 종료 또는 재시작 전에 sync 를 호출하지 않음. (-d 옵션을 포함함)
-w : 실제로 종료 또는 재시작하지 않지만 wtmp 파일에 기록을 남김.
-d : wtmp 파일에 기록을 남기지 않음.
-f : 강제로 명령 실행, shutdown 을 호출하지 않음.
-p : 시스템의 전원을 끔.

이 명령들은 systemctl 의 심볼릭 링크임.
즉, 모두 systmectl 명령을 사용하고 있는 것임.

4. 부트 로더

1. 부트 로더

커널에 메모리를 로딩하는 역할을 수행.

리눅스에는 LILO 와 GRUB 두 가지 부트 로더가 있음. 우분투에서는 GRUB 이 기본임.

메모 포함[이186]: 정통 부트로더.

일반적으로 부트 로더를 실행시키면 부팅할 운영체제를 선택할 수 있는 메뉴를 제공함.
멀티 부팅이 아닐 경우 메뉴를 출력하지 않고 부팅 작업을 진행함.

부팅 시 GRUB 메뉴를 출력하는 방법

1. vi 로 /etc/default/grub 파일을 연다.
2. GRUB_HIDDEN_TIMEOUT=0 에서 맨 앞 G 앞에 #을 추가한다.
3. sudo update-grub 명령으로 변경 사항을 적용한다.

2. GRUB (grand unified bootloader)

정통 부트로더인 LILO 의 단점을 보완하여 GNU 프로젝트에서 개발한 부트 로더.

보완점들.

1. 윈도에서도 사용 가능
2. 설정, 사용 편리
3. 부팅 시 명령을 사용하여 수정 가능
4. 멀티 부팅 기능 지원

최신 버전은 GRUB2 이고, 우분투에서는 GRUB2 를 기본으로 사용함.

+ 커널 생성 위치

리눅스 커널은 /boot 디렉토리 밑에 "vmlinuz-<버전명>" 형태로 제공됨.

3. GRUB2 관련 디렉토리, 파일들

1) /boot/grub/grub.cfg 파일

GRUB 기본 설정 파일.

기존의 menu.lst 파일을 대체하는 파일.

/etc/default/grub 파일과, /etc/grub.d 디렉토리 안의 스크립트를 읽어서 생성됨.
-> /boot/grub/grub.cfg 파일은 수정 불가능. 수정이 필요하면 읽어오는 파일이나 스크립트를 수정해야 함.

2) /etc/default/grub 파일

GRUB 설정 내용이 저장되어 있는 파일.

GRUB 스크립트가 이 파일을 읽어 grub.cfg 파일에 기록함. -> menu.lst 파일과 유사한 역할을 수행함.

이 파일을 수정한 경우 update-grub 명령으로 변경 내용을 적용해 줘야 함.

3) /etc/grub.d 디렉토리

GRUB 스크립트를 가지고 있는 디렉토리.

GRUB 의 명령이 실행될 때 순서대로 읽혀 grub.cfg 파일을 생성함.

4. GRUB2 부트 로더를 이용해서 계정의 암호 복구하기

작업 순서.

1. 시스템을 재시작 한다.
2. 부팅 중에 GRUB 편집 모드로 전환한다.
3. 단일 사용자 모드로 부팅한다.
4. 시스템을 재시작 한다. -> 단일 사용자 모드이기 때문에 root 가 되어 암호를 복구할 수 있음.

5. 복구 모드로 부팅하기

우분투가 부팅되지 않는 경우 복구 모드에서 복구에 필요한 작업을 수행할 수 있음.

메모 포함[이187]: 가장 기본적인 서비스만 제공하며 명령 모드로 작업할 수 있는 모드이다.

작업 순서.

1. 시스템 재시작 중에 GRUB 메뉴 초기 화면에서 복구 모드를 선택한다.
2. root 로 로그인한다.
3. 루트 파일시스템을 읽기/쓰기가 모두 가능하도록 다시 마운트한다.
4. 시스템을 재시작 한다.

6. 소프트웨어 관리

+ 리눅스의 소프트웨어 배포

리눅스는 소프트웨어를 "소스코드 형식" 또는 바로 설치하여 사용할 수 있는 "패키지 형식"으로 배포함.

1. 우분투 패키지

1. 우분투 패키지의 구성

우분투에서는 데비안 계열의 표준 패키지인 deb 패키지를 사용함.

1) 우분투 패키지의 특징

바이너리 파일임. -> 컴파일 필요 X

설치, 업데이트, 제거가 굉장히 쉬움.

패키지의 설치 상태를 검증할 수 있음.

패키지에 대한 정보를 제공함.

해당 패키지와 의존성을 가지고 있는 패키지가 무엇인지 알려줌.

2) 우분투 패키지의 카테고리

자유 소프트웨어 지침에 따라 카테고리가 4 가지로 분류됨

main : 우분투에 의해 공식적으로 지원됨. 자유 배포 가능.

restricted : 우분투에 의해 공식적으로 지원됨. 자유 배포는 제한됨.

universe : 자유 배포가 가능할 수도 있고, 아닐 수도 있음. 기술적 지원을 보장하지 않음. (모르는 것)

multiverse : 자유 소프트웨어가 아닌 소프트웨어가 포함되어 있음. (분명히 섞여 있는 것)

메모 포함[이188]: 바이너리 패키지에는 크게 두 가지 종류가 있다
RPM과 deb이다.
RPM은 래드햇 계열 리눅스에서 주로 사용한다.

우분투는 데비안 계열의 리눅스이므로 deb 패키지를 사용한다.

메모 포함[이189]: 설치하면 관련 디렉토리에 자동으로 설치된다.
삭제하면 관련 파일들을 일괄적으로 삭제한다.
기존 패키지를 삭제하지 않고 업데이트할 수 있다.

메모 포함[이190]: 리눅스에서 사용할 수 있는 대부분의 소프트웨어는 universe 카테고리에 속한다.

3) 우분투 패키지의 이름 형식

<파일명>_<버전>-<리비전>_<아키텍처>.deb

버전 : 패키지의 버전.

리비전 : 소스의 버전이 바뀌지는 않았지만 보안, 의존성, 스크립트 등에 변화가 있음을 나타냄.

아키텍처 : 사용하는 시스템 아키텍처.

확장자 : 우분투에서는 deb 형식을 사용하므로 확장자는 .deb 임.

메모 포함[이191]: i386은 인텔을 의미한다.
all은 시스템과 상관없는 문서나 스크립트를 의미한다.

2. 우분투 패키지 저장소

패키지와 패키지에 대한 정보가 저장되어 있는 서버.

지속적인 업그레이드를 관리함.
패키지 저장소에서 최신 패키지를 내려 받아 사용할 수 있음.

1) sources.list 파일

패키지 저장소에 대한 정보는 /etc/apt/sources.list 파일에 저장되어 있음.

이 파일을 수정하여 저장소를 추가, 삭제할 수 있음.
수정한 후에 apt-get update 명령을 사용해서 적용해야 함.

2) sources.list 파일의 형식

패키지 유형, 저장소 주소, 버전 정보, 카테고리

한 줄에 하나의 패키지 저장소에 대한 정보가 저장됨.

sources.list 파일의 형식은 <패키지 유형, 저장소 주소, 버전 정보, 카테고리>임.

패키지 유형 : deb 는 바이너리 패키지의 저장소를, deb-src 는 패키지의 소스 저장소를 의미함.
저장소 주소 : http 프로토콜을 사용하는 URL 주소를 사용함.
버전 정보 : 패키지에 해당하는 버전의 이름을 표시함.
카테고리 : 해당 소프트웨어의 카테고리를 표시함.

메모 포함[이192]: 버전의 번호가 아니라 이름을 표시한다.

2. 우분투 패키지 설치

우분투에 패키지를 설치하는 방법에는 APT 명령 사용, dpkg 명령 사용, aptitude 명령 사용, 소프트웨어 센터 사용 등이 있음.

1. APT 명령

1) apt-cache 명령 (apt-캐시)

APT 캐시(패키지 데이터베이스)에서 정보를 검색하여 서브 명령을 수행하는 명령어.

2) apt-get 명령

패키지를 관리하는 명령어.
패키지의 설치, 삭제, 업데이트 등을 수행함.

내부적으로는 dpkg 명령을 수행하는 것.

2. dpkg 명령

패키지를 관리하는 명령어.

apt-get 보다 더 세부적인 기능을 제공함.

3. aptitude 명령

패키지를 관리하는 명령어.

기능과 서브 명령들이 apt-get 과 유사하지만, aptitude 명령어에서는 curses 프로그램을 사용할 수 있음.
curses 프로그램은 텍스트 그래픽 기능을 제공하는 패키지 관리 프로그램임. -> 마우스로 메뉴 선택 가능.

4. 스냅 패키지 설치

1) 스냅

우분투가 비교적 최근에 도입한 샌드박스 형태의 패키지 형식.

프로그램이 사용하는 모든 라이브러리를 패키지 안에 포함함.

기존의 deb 패키지와 호환성을 유지함.

2) 스냅의 장단점

장점

1. 다른 패키지나 라이브러리와의 의존성을 고려하지 않아도 됨.
2. 보안이 강화됨.

단점

1. 패키지 용량이 커짐.

3) snap 명령어

스냅 패키지 관리 명령어.

스냅 패키지를 설치, 설정, 삭제함.

5. 우분투 소프트웨어 센터 (GNOME 소프트웨어 센터)

소프트웨어를 내려 받을 수 있는 소프트웨어 마켓.

메모 포함[이193]: 구글플레이나 앱스토어와 동일한 개념이다.

+ 샌드박스

외부에서 유입된 파일을 보호된 영역에서 실행하는 방식.

외부의 파일이 내부 시스템에 악영향을 주는 것을 방지함.

주로 안드로이드나 애플에서 사용하는 패키지 개념임.

3. 소스 코드 설치

1. 아카이브

파일이나 디렉토리를 묶어서 하나로 만든 것.

백업이나 다른 시스템과 파일을 주고받는 일을 위해서 아카이브 파일을 생성함.

메모 포함[이194]: 원래는 보관소, 저장소 등의 의미로 사용되지만, 여기에서는 조금 다른 의미로 사용되었다.

1) tar 명령어

파일이나 디렉토리를 묶어서 하나의 아카이브 파일을 생성하거나, 아카이브 파일을 추출하는 명령어.

아카이브 파일을 만들어도 원본 파일은 제거되지 않음.

2. 파일 압축

1) gzip 명령어

파일을 압축하는 명령어.

zcat 명령어로 해당 압축 파일의 내용을 볼 수 있음.

gunzip 명령어로 해당 압축 파일의 압축을 풀 수 있음.

2) bzip2 명령어

파일을 압축하는 명령어.

gzip 에 비해 압축률을 좋지만 속도가 느림.

bzcat 명령어로 해당 압축 파일의 내용을 볼 수 있음.

bunzip2 명령어로 해당 압축 파일의 압축을 풀 수 있음.

3. 소프트웨어 컴파일

소스 코드로 이루어진 프로그램을 사용하기 위한 것.

1) 컴파일러

고급 언어를 기계어로 변환하는 소프트웨어.

우분투에서 c 언어 컴파일러 이름은 gcc 임.

2) 컴파일러 설치

gcc 패키지를 설치하면 컴파일러를 사용할 수 있음.

3) 컴파일 과정

1. 프로그램 작성.
2. gcc 명령을 사용하여 컴파일.
3. a.out 이라는 이름의 실행파일 생성됨.

4) make 명령

여러 개의 파일을 한 번에 컴파일하기 위한 명령어.

소스 코드와 함께 배포 받은 makefile 의 정보를 읽어서 여러 소스 파일을 컴파일하고 링크하여 실행 파일을 생성함.

메모 포함[이195]: 정보가 저장되어 있는 파일이다.

7. 사용자 관리

1. 사용자/그룹 관리

사용자/그룹의 생성, 수정, 삭제

1. 사용자 계정 관련 파일 (5 가지)

1) /etc/passwd

사용자 계정 정보가 저장된 기본 파일.

한 줄에 사용자 한 명에 대한 정보가 저장됨.

형식은 <로그인ID>:x:<UID>:<GID>:<설명>:<홈 디렉토리>:<로그인 쉘> 임.

누구나 읽을 수 있음.

로그인 ID : 사용자 계정의 이름
x : 초기 유닉스 시스템에서 사용자 암호를 지정하던 항목. 지금은 사용하지 않아 x 만 명시함.
UID : 사용자 ID
GID : 그룹 ID
설명 : 관련 설명들.
홈 디렉토리 : 사용자 계정에 할당된 홈 디렉토리의 절대 경로 지정.
로그인 쉘 : 사용자의 로그인 쉘 지정.

메모 포함[이196]: 보안상의 이유로 /etc/shadow 파일에 별도로 보관한다.

메모 포함[이197]: 로그인 ID가 달라도 UID가 같으면 같은 사용자이다. 그러므로 UID가 겹치지 않도록 유의해야 한다.

+ UID, 계정 이름, 로그인 ID

사용자 계정 이름과 로그인 ID 는 같은 것.
UID 는 사용자의 고유번호. 로그인 ID 가 달라도 UID 가 같으면 같은 사용자이다.

2) /etc/shadow

사용자 암호에 관한 정보를 관리하는 파일.

아무나 읽거나 쓸 수 없게 되어 있음.

한 줄에 하나의 사용자에 대한 정보가 저장됨.
형식은 <로그인 ID>:<암호>:<최종 변경일>:<MIN>:<MAX>:<WARNING>:<INACTIVE>:<EXPIRE>:<Flag> 임.

암호 : 암호화된 비밀번호. 암호를 설정하지 않았다면 비어 있음. 시스템 관리자는 !!가 들어가 있음. 잠긴 암호일 경우 !로 시작함.
최종 변경일 : 암호가 마지막으로 변경될 날짜. 1970 년 1 월 1 일을 기준으로 날수를 기록함.
MIN : 암호 변경 후 사용해야 하는 최소 기간.
MAX : 암호를 사용할 수 있는 최대 기간.
WARNING : 암호가 만료되기 전 경고를 시작하는 날수.
INACTIVE : 암호 만료 후에도 로그인이 가능한 날수.
EXPIRE : 사용자 계정이 만료되는 날. 1970 년 1 월 1 일을 기준으로 한 날수로 표시.
Flag : 향후 사용할 목적으로 비워 둔 항목.

메모 포함[이198]: <MIN>:<MAX>:<WARNING>:<INACTIVE>:<EXPIRE> 까지의 항목을 패스워드 에이징이라고 한다. 나이를 지정하는 것이다.

메모 포함[이199]: 이렇게 암호화한 것은 복호화 할 수 없다.
사용자가 암호를 입력하면 입력한 것을 암호화한 후 이전에 암호화해 둔 것과 비교한다.

복호화 또는 디코딩은 부호화 된 정보를 부호화 되기 전으로 되돌리는 처리 혹은 그 처리 방식을 말한다.

메모 포함[이200]: ex. WARNING이 7이면 만료 7일 전부터 경고 메시지가 나타난다.

3) /etc/login.defs (login defaults)

사용자 계정의 설정과 관련된 기본값을 지정한 파일.

한 줄에 하나의 항목에 대한 정보가 저장됨.
형식은 <항목> <기본값> 임.

표 10-1 /etc/login.defs 파일의 내용

항목	기본값	의미
MAIL_DIR	/var/mail	기본 메일 디렉터리
PASS_MAX_DAYS	99999	패스워드 에이징
PASS_MIN_DAYS	0	
PASS_WARN_AGE	7	
UID_MIN, UID_MAX	1000~60000	사용자 계정의 UID 범위
SYS_UID_MIN, SYS_UID_MAX	100~999	시스템 계정의 UID 범위
GID_MIN, GID_MAX	1000~60000	사용자 계정의 GID 범위
SYS_GID_MIN, SYS_GID_MAX	100~999	시스템 계정의 GID 범위
UMASK	022	umask 값 설정
USERGROUPS_ENAB	yes	사용자 계정 삭제 시 그룹 삭제 여부
ENCRYPT_METHOD	SHA512	암호화 기법

4) /etc/group

그룹의 정보가 저장된 파일.

사용자의 그룹 중 기본 그룹은 /etc/passwd 파일에 지정되고, 2 차 그룹은 /etc/group 파일에 지정됨.
특정 그룹에 대한 정보에는 해당 그룹이 기본 그룹인 사용자는 명시되지 않음.

한 줄에 하나의 그룹에 대한 정보가 저장됨.
형식은 <그룹명>:<x>:<GID>:<그룹 멤버> 임,

x : 그룹 암호를 저장하는 곳.
GID : 그룹 ID. (그룹을 식별하는 번호)
그룹 멤버 : 그룹에 속한 사용자들의 계정 이름. 쉼표로 구분함.

메모 포함[이201]: gpasswd 아님!! 조심하자.

메모 포함[이202]: 그룹 암호는 이곳이나 /etc/gshadow 파일에 저장된다.

메모 포함[이203]: 여기에 명시된 그룹 멤버들에게 이 그룹은 2차 그룹이다.

5) /etc/gshadow

그룹 암호가 저장된 파일.

리눅스에서 독자적으로 만든 파일로, 유닉스에는 없음.

한 줄에 하나의 그룹 암호에 대한 정보가 저장됨.
형식은 <그룹명>:<그룹 암호>:<관리자>:<그룹 멤버> 임.

그룹 암호 : 암호화된 그룹 암호.
관리자 ; 그룹의 암호나 멤버를 수정할 수 있는 사용자 계정 이름. 쉼표로 구분.
그룹 멤버 : 그룹에 속한 사용자들의 계정 이름. 쉼표로 구분함.

메모 포함[이204]: 유닉스에서는 /etc/group 파일에만 명시할 수 있다.

메모 포함[이205]: 여기에 명시된 그룹 멤버들에게 이 그룹은 2차 그룹이다.

+ 그룹

리눅스에서는 사용자는 무조건 한 개 이상에 그룹에 포함됨.

사용자의 기본 그룹은 사용자 등록 시에 결정됨.

소속 그룹을 지정하지 않으면 사용자의 로그인 ID 가 그룹으로 등록됨.

2. 사용자 계정 관리 명령

사용자 계정을 생성, 수정, 삭제, 에이징하는 명령들.

1) 사용자 계정 생성

useradd 명령어, adduser 명령어를 사용함.

우분투에서는 adduser 명령어를 사용할 것을 권고함.

useradd 명령어와 adduser 명령어는 유사하지만,
useradd 명령어에서는 패스워드 에이징 관련 설정을 할 수 있지만, 암호를 지정할 수 없고, 옵션을 입력하지 않으면 아무것도 지정되지 않음.
adduser 명령어에서는 패스워드 에이징 관련 설정을 할 수 없지만, 계정 생성 과정에서 암호를 지정할 수 있고, 옵션을 입력하지 않아도 기본 설정이 적용됨.

2) 사용자 계정 정보 수정

usermod 명령어를 사용함.

3) 사용자 계정 삭제

userdel 명령어를 사용함.

4) 패스워드 에이징

패스워드 에이징은 useradd 명령아, passwd 명령어, chage 명령어 등으로 관리할 수 있음.

chage 명령어는 모든 항목을 수정할 수 있음.

메모 포함[이206]: 에이징 항목에는 MAX, MIN, WARNING, INACTIVE, EXPIRE가 있다. (5가지)

표 10-2 패스워드 에이징 관련 명령

항목	useradd, usermod, passwd 명령	chage 명령
MIN	passwd -n 날수	chage -m
MAX	passwd -x 날수	chage -M
WARNING	passwd -w 날수	chage -W
INACTIVE	useradd -f 날수 usermod -f 날수	chage -I(대문자 i)
EXPIRE	useradd -e 날짜(YYYY-MM-DD) usermod -e 날짜(YYYY-MM-DD)	chage -E

3. 그룹 관리 명령

그룹을 생성, 수정, 삭제하는 명령들.

1) 그룹 생성

groupadd 명령어, addgroup 명령어를 사용함.

groupadd 는 useradd 와, addgroup 은 adduser 와 유사함.

생성된 그룹은 /etc/group 파일에 저장됨.

2) 그룹 정보 수정

groupmod 명령어를 사용함.

3) 그룹 삭제

groupdel 명령어 사용함.

4) 그룹 암호 관리

gpasswd 명령어를 사용함.

5) 그룹 변경

newgrp 명령어를 사용함.

이때 그룹 암호가 필요할 수 있음.

2. 사용자 정보 관리

사용자 정보 관리와 관련된 자세한 명령들.

1. UID 와 EUID

1) UID 와 EUID 의 차이

UID (RUID, real user ID) : 실제 사용자 ID. 사용자가 로그인할 때 사용한 계정의 ID
EUID (effective user ID) : 유효 사용자 ID. 현재 명령을 수행하는 주체의 ID.

대부분의 경우에는 UID 와 EUID 가 동일함.

2) UID 와 EUID 가 달라지는 경우

1. 실행 파일에 SetUID 가 설정되어 있는 경우.
-> 실행 파일을 실행한 프로세스의 UID 는, 사용자 계정의 UID 가 아니라 실행 파일 소유자의 UID 가 됨. 이때 실행 파일 소유자의 UID 가 EUID 임.

2. su 명령을 사용하여 다른 계정으로 전환한 경우.
-> 최초 로그인 시에는 UID 와 EUID 가 동일하지만, su 명령을 사용하면 달라짐.

2. 사용자 확인 명령

1) 현재 로그인한 사용자 확인 명령

who 명령어, w 명령어, last 명령어를 사용.

2) UID 와 EUID 확인 명령

whoami 명령어, who am i 명령어(who -m 와 동일), id 명령어를 사용.

UID 출력 -> who am i, who -m
EUID 출력 -> whoami, id

whoami
who am i
id

3) 소속 그룹 확인 명령

groups 명령어를 사용함.

3. root 권한 사용하기

1) su 명령어를 사용하여 root 계정으로 전환하기

모든 권한을 넘기는 것이므로 보안상 굉장히 취약함.

메모 포함[이207]: su <로그인 ID> 로 작성하는 명령인 것 같은데, 교재에 제대로 나와있지 않다.

2) sudo 권한 설정하기.

sudo <명령>

특정 명령어에 대한 제한적인 권한 부여.

/etc/sudoers 파일에 저장된 내용에 따라 sudo 를 사용한 명령어에 대한 제한적인 권한이 설정됨.

/etc/sudoers 파일의 형식은 <사용자 ID> <호스트>=<명령어의 절대 경로> 임.

메모 포함[이208]: ex. jhun ALL=/usr/sbin/useradd

여러 명령어를 작성할 경우 쉼표로 구분함.

메모 포함[이209]: 어떤 것인지는 모르겠는데 예시를 보면
전부 ALL로 명시하기는 했다.

4. passwd 명령 사용하기

passwd 명령어는 암호 설정 외에도 여러 암호 관리 기능을 수행함.

5. 파일 및 디렉토리의 소유자와 소유 그룹 변경하기

파일이나 디렉토리는 그것을 생성한 사용자의 계정과 그룹이 소유자와 소유 그룹으로 설정됨.

소유자 변경에는 chown 명령어를 사용함.
소유 그룹 변경에는 chgrp 명령어를 사용함.

3. 디스크 쿼터 (quota)

1. 쿼터 (quota, 한도)

1) 쿼터

디스크 사용량을 제한하는 기능.

파일의 전체 용량을 제한하는 방법과, 사용 가능한 총 파일 개수를 제한하는 방법이 있음.

하드 리미트와 소프트 리미트를 설정해서 제한함.

2) 쿼터 값 설정

하드 리미트와 소프트 리미트를 설정함.

하드 리미트 : 사용자가 넘을 수 없는 최대치.
소프트 리미트 : 일정 기간 내에는 넘을 수 있는 최대치.
그레이스 (grace, 유예기간) : 소프트 리미트를 초과해도 유예하는 일정 기간.

3) 쿼터 사용을 위한 사전 설정

1. quota 패키지 설치

2. 파일시스템의 마운트 옵션 란에 쿼터 속성 설정 -> /etc/fstab 파일에 usrquota 또는 grpquota 속성 설정.
usrquota : 개별 사용자의 쿼터를 제한할 수 있는 속성.
grpquota : 개별 그룹의 쿼터를 제한할 수 있는 속성.

3. 쿼터 속성 적용 -> 파일시스템을 다시 마운트 해야 함.

4. 쿼터 정보를 저장하는 데이터베이스 파일 생성 - quotacheck -augvm 명령어 사용.

5. 쿼터 사용 활성화 -> quotaon 명령어 사용.

메모 포함[이210]: 마운트 포인트가 / 인 경우 언마운트 할 수 없다. 이때는 sudo mount -o remount / 명령어를 사용한다.

메모 포함[이211]: 디스크 쿼터를 관리하는 데이터베이스는 두 개의 파일로 구성되어 있다.
aquota.user와 aquota.group 이다.

데이터베이스 파일은 해당 파일시스템의 최상위 디렉토리에 생성된다.

2. 쿼터 설정

edquota 명령어로 쿼터를 설정함.

3. 쿼터 확인

quota 명령어로 쿼터 정보를 확인함.

repquota 명령어로 파일시스템 전체 사용자의 쿼터 설정을 요약하여 출력함.

8. 네트워크 설정

1. 네트워크의 기초

1. TCP/IP 프로토콜

1) 프로토콜

컴퓨터와 컴퓨터 사이에 데이터를 어떻게 주고받을 지 정의한 통신 규약.

같은 프로토콜을 가지고 있다면 통신이 가능함.

인터넷 네트워크는 TCP/IP 프로토콜을 따름.

2) TCP/IP 프로토콜의 계층

TCP/IP 프로토콜은 다양한 프로토콜의 집합임.

각 계층별로 프로토콜이 존재하며 수행하는 역할이 다름.

이 중 TCP 와 IP 프로토콜을 대표로 하여 TCP/IP 프로토콜이라고 부르는 것.

[표 11-1] TCP/IP 프로토콜 모델의 계층별 역할과 대표 프로토콜

계층	기능	프로토콜	전송 단위
응용 계층	서비스 제공 응용 프로그램	DNS, FTP, SSH, HTTP, Telnet	메시지
전송 계층	응용 프로그램으로 데이터를 전달, 데이터 흐름 제어 및 전송 신뢰성 담당	TCP, UDP	세그먼트
네트워크 계층	주소 관리 및 경로 탐색	IP, ICMP	패킷
링크 계층	네트워크 장치 드라이버	ARP	프레임
물리 계층	케이블 등 전송 매체	구리선, 광케이블, 무선	비트

(아래부터 첫 번째 계층, 맨 위가 다섯 번째 계층임)

+ 이더넷 (ethernet)

현재 대부분의 네트워크에서는 이더넷 방식의 인터페이스를 사용함.

이 곳에 정리하는 내용은 이더넷 방식의 네트워크 인터페이스 기준임.

2. 주소

컴퓨터와 컴퓨터를 구분하는 역할을 하는 것.

컴퓨터 주소에는 MAC 주소, IP 주소, 호스트 이름 등이 있음.

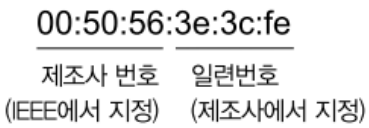
3. MAC 주소 (media access control)

하드웨어를 위한 주소.

TCP/IP 프로토콜의 계층 중 링크 계층에서 사용함.

네트워크 인터페이스 카드 (랜 카드) 에 저장된 주소.
네트워크 인터페이스 카드가 생성될 때 부여됨.
일반적으로는 수정 불가능.

이더넷 주소, 하드웨어 주소, 물리 주소라고도 함.



1) 형식

콜론(:) 이나 붙임표(-)로 구분되는 여섯 개(각각 두 자리)의 16 진수로 나타냄.
총 48 비트. (16 진수 한 자리는 4 비트)
앞 세자리는 제조사 번호, 뒤 세자리는 일련번호임.

+ 인터페이스

서로 다른 두 개의 시스템이나 장치 사이에서 정보나 신호를 주고받는 경우의 접점이나 경계면.

메모 포함[이212]: 각 하드웨어를 구분하는 역할을 수행한다.

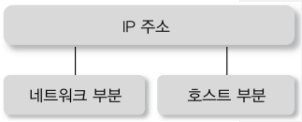
메모 포함[이213]: 6 개의 두 자리 16 진수이므로. 16 진수 한 자리가 4 비트이다.

4. IP 주소

인터넷으로 연결된 네트워크에서 각 컴퓨터를 구분하기 위해 사용하는 주소.

컴퓨터가 인터넷에 연결되려면 IP 주소가 있어야 함.

TCP/IP 프로토콜 중 3~5 계층에서 사용함.



1) 형식 (IPv4)

1 바이트 크기의 네 개의 숫자를 마침표(.)로 구분하여 나타냄.

총 32 비트(4 바이트).

네트워크를 구분하는 네트워크 주소 부분과 네트워크 안에서 특정 컴퓨터를 식별하는 호스트 주소 부분으로 구성됨.

메모 포함[이214]: IP 버전 4.
현재 이 주소는 고갈되어서 IPv6를 쓴다.
IPv6는 16진수를 쓰고 크기가 훨씬 큼.

메모 포함[이215]: ex. 192.31.123.5

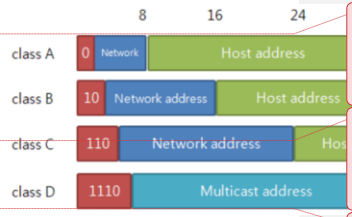
2) 클래스

32 비트 중 몇 비트를 네트워크 부분으로 사용하고 몇 비트를 호스트 부분으로 사용할 지에 따라 클래스가 나뉨.

A 클래스, B 클래스, C 클래스가 있음.

C 클래스에서는 앞의 3 바이트가 네트워크 부분이고 뒤 1 바이트가 호스트 부분임.

1 바이트(0~255) 중 호스트 주소로 할당할 수 있는 부분은 1~254 임.



메모 포함[이216]: 0은 네트워크 주소를 나타내는 데 쓴다.
255는 브로드캐스트 주소로 쓴다.

3) 넷마스크 (network mask)

IP 주소에서 네트워크 부분을 알려주는 역할을 하는 것.

IP 주소와 AND 연산을 해서 네트워크 부분만 남김. (mask 를 씌움)

하나의 네트워크를 여러 작은 네트워크들(서브넷)로 분리할 때도 사용함.

서브넷 마스크라고도 함.

C 클래스의 IP 주소의 경우 기본 넷마스크가 255.255.255.0 임.

메모 포함[이217]: A, B, C 클래스 중 무엇인지 알려주는 것이 아니라, 특정 클래스인 것이 정해졌을 때의 네트워크 부분을 알려주는 것이다.

메모 포함[이218]: 과정.
1. IP 주소와 넷마스크를 각각 2진수로 바꾼다.
2. AND 연산을 진행한다.

AND 연산이란?
같은 자리의 두 수를 비교해서 둘 다 1이면 1, 둘 중에 하나라도 0이면 0이 되는 것.

ex. 192.168.100.5에 C 클래스 기본 넷마스크인 255.255.255.0을 적용하면 192.168.100.0이 됨.

4) 브로드캐스트 주소 (broadcast, 널리 알리다.)

같은 네트워크 내의 모든 컴퓨터에 메시지를 보낼 때 사용하는 것.

호스트 부분의 비트를 전부 1 로 함. -> C 클래스에서 4 번째 바이트의 255 는 브로드캐스트 주소임.

5. 호스트 이름

IP 주소를 대신하는 이름.

IP 주소를 사람이 그대로 사용하기에는 불편해서 호스트 이름을 사용하게 됨.

네트워크 서비스를 제공하는 서버 컴퓨터는 용도에 따라 호스트 이름을 붙여 사용함.
개인용 PC 등에는 호스트 이름을 사용하지 않음.

메모 포함[이219]: ex. 202.179.177.21 대신 www.naver.com 을 사용한다.

1) 형식

IP 주소처럼 네트워크 부분과 호스트 부분으로 구성됨.

메모 포함[이220]: ex. www.naver.com에서 www가 호스트 부분, naver.com이 네트워크 부분이다.

+ 도메인 이름과 호스트 이름의 차이?

메모 포함[이221]: 여기서 구체적으로 설명해주지 않으니, 일단은 비슷한 것으로 취급하자.

6. 포트 번호

각 서비스를 구분하는 번호.

사용자가 요청한 서비스를 제공하려면 서비스들을 구분할 수 있어야 함.

TCP/IP 프로토콜의 전송 계층에서 포트 번호를 사용함.

/etc/services 파일에 서비스 별 포트 번호를 정의함.

7. 게이트웨이(라우터) (route, 길)

네트워크와 네트워크를 연결할 때 연결점이 되는 장치.
패킷을 확인하여 내부의 네트워크로 보낼 지, 외부의 네트워크로 보낼 지 결정함.

TCP/IP 프로토콜의 네트워크 계층에서 다룸.

게이트웨이 주소가 설정되어 있지 않으면 내부 네트워크의 컴퓨터에만 접속할 수 있음.
같은 네트워크의 컴퓨터와는 통신이 되는데 외부의 것과는 안 된다면, 게이트웨이 주소가 올바른 지 확인해야 함.

라우팅 테이블에는 게이트웨이에 대한 정보들이 들어 있음.

메모 포함[이222]: 전 세계에는 여러 개의 네트워크가 존재하고, 이런 네트워크들을 연결한 것이 인터넷이다.

메모 포함[이223]: 게이트웨이 또한 하나의 컴퓨터이다.

8. DNS (domain name system)

호스트명을 IP 주소로 변환하는 서비스.
이 서비스가 설정되어 있지 않으면 호스트명으로 서버에 접속할 수 없음.

DNS 서버의 주소는 /etc/resolv.conf 파일에 저장되어 있음.

+ DHCP

클라이언트가 요청하면 IP 주소를 할당해주는 프로토콜.

2. 네트워크 설정

네트워크 사용을 위해 설정해야 하는 것들 (5 가지) :
IP 주소, 넷마스크, 브로드캐스트 주소, 게이트웨이(라우터) 주소, DNS 주소

1. ifconfig 명령어

네트워크 인터페이스를 설정하는 전통적인 명령어.

IP 주소, 넷마스크, 브로드캐스트 주소 설정 가능.

1) 파일에 저장

ifconfig 명령어로 지정한 네트워크 인터페이스 설정이 항상 적용되게 하려면 파일에 저장해야 함.
-> /etc/network/interfaces 파일에 저장.

2. 게이트웨이(라우터) 설정

route 명령어 사용.

3. DNS 설정

nslookup 명령어를 사용함.

4. 호스트 이름 설정

IP 주소 대신 사용할 호스트 이름을 설정할 수 있음.

1) 호스트 이름 확인

hostname, uname 명령어를 사용함.

2) 호스트 이름 설정

hostname 명령어를 사용함.

3) 호스트 이름 설정 파일

시스템을 다시 시작했을 때도 변경한 호스트 이름을 유지하려면 파일에 설정해야 함.

/etc/hostname 파일을 수정해서 설정함.

3. 네트워크 상태 확인

1. 통신 확인

ping 명령어를 사용함.

보안상 문제 때문에 ping 패킷에 응답하지 않는 시스템도 있기 때문에 ping으로 확인이 안 된다고 해당 시스템이 동작하지 않는 건 아닐 수 있음.

2. 통신 경로 확인

traceroute 명령어를 사용함.

ping 명령어는 연결이 되는지 안되는 지만 알려주기 때문에 구체적으로 문제가 있는 지점을 찾으려면 traceroute 명령어를 사용해야 함.

3. 네트워크 상태 정보 확인

netstat 명령어를 사용함.

4. MAC 주소와 IP 주소 확인

arp 명령어를 사용함.

5. 패킷 캡처

패킷을 캡처하고 분석하여 네트워크 상태를 확인할 수 있음.

tcpdump 명령어를 사용함.

9. 원격 접속과 FTP

리눅스 관련 실무에서는 리눅스 시스템에서 직접 작업하기보다는 원격으로 작업하는 경우가 일반적임.
리눅스는 서버로 사용되는 경우가 많은데 서버 컴퓨터는 사용자와 다른 위치에 있는 경우가 많음.

메모 포함[이224]: 원격 접속, FTP, Samba 등 두 가지 객체 사이에 뭔가 주고받는 형태의 기능들은 서버와 클라이언트가 존재한다.

특히 이 장에 나오는 것들은 하나같이 똑같은 방식이다. 서버와 클라이언트를 각각 설치하여 접속하는 것이다.

1. 텔넷과 SSH

텔넷과 SSH 는 TCP/IP 프로토콜 중 5 단계(응용 프로그램)에 있는 프로토콜임.

1. 텔넷

리눅스에 원격 접속하는 프로그램.
텔넷은 원래 TCP/IP 프로토콜 중 5 단계에 있는 프로토콜의 이름임.

패킷이 암호화되어 있지 않아 해킹에 취약함.

1) 텔넷 서버와 텔넷 클라이언트

텔넷 서비스를 제공하려면 텔넷 서버가 필요함.
텔넷 서비스를 사용하려면 텔넷 클라이언트가 있어야 함.

2) 텔넷 서버 설치

1. xinetd 슈퍼데몬 패키지를 설치한다. -> 텔넷 서버 데몬은 xinetd 슈퍼데몬에 의해 동작함.
2. 텔넷 서버 패키지를 설치한다.
3. xinetd 와 텔넷 서버를 활성화한다.

2. 텔넷 사용법

1) 텔넷 모드 사용

1. telnet 을 입력한다.
2. open <IP 주소>를 입력해 리눅스에 접속하거나 quit 을 입력해 텔넷을 종료한다.

메모 포함[이225]: 클라이언트의 IP 주소.

2) 직접 서버에 접속

1. telnet <주소 or 호스트 이름>을 입력해 바로 해당 시스템에 접속.

3. 윈도우에서 텔넷 사용하기

윈도우에서 제공하는 텔넷 클라이언트나 HPuTTY 등의 프로그램을 사용해야 함.

1) 윈도우 텔넷 사용

1. 윈도우 설정에서 텔넷 클라이언트 활성화
2. 텔넷을 실행한다. -> 텔넷 모드가 실행됨.

한글이 깨져 보일 수 있음.

2) HPuTTY 사용

1. HPuTTY 프로그램을 설치 또는 비설치 방식으로 실행한다.
2. 주소 등을 입력하고, 접속 형식으로 telnet 을 지정한다.

한글이 깨져 보이지 않음.

메모 포함[이226]: 원격 접속 프로그램 PuTTY의 한글화(H) 버전이다.

4. SSH

리눅스에 원격 접속하는 프로그램.
텔넷은 TCP/IP 프로토콜 중 5 단계(응용)에 있는 프로토콜임.

텔넷과 기본적인 기능이 동일하지만 패킷이 암호화되어 있음.

1) 윈도에서 사용

1. HPuTTY 프로그램을 설치 또는 비설치 방식으로 실행한다.
2. 주소 등을 입력하고, 접속 형식으로 shh 를 지정한다.
3. ssh 인증 키를 생성한다.

2. VNC

원격에서 그래픽 환경으로 접속할 수 있게 하는 기능.

1. VNC (virtual network computing)

기본적인 개념은 텔넷, SHH 와 동일함.
텔넷이나 SHH 는 CLI 방식으로 사용하게 되지만, VNC 서버를 사용하면 GUI 환경을 사용할 수 있음.

GUI 방식이므로 속도가 느림.

텔넷처럼 VNC 도 VNC 서버와 VNC 클라이언트가 있어야 함.

2. VNC 서버

1) VNC 서버 설치

VNC 서버 패키지와 xfce4 패키지를 설치.

2) VNC 서버 설정 (일반 사용자가 설정하는 방식)

1. vncserver 를 실행한다. -> 암호를 입력 받고, 여러 파일들이 생성됨.
2. xstartup 파일 수정. -> 그래픽 환경인 X 윈도를 위한 파일.
3. vncsever 를 종료한 후 다시 실행.
4. 해상도 설정
5. 방화벽 제거

메모 포함[이227]: VNC 설정은 사용자별로 진행해야 한다.

3. VNC 클라이언트

VNC 클라이언트는 리눅스와 윈도우 모두에서 사용 가능함.

1) 리눅스용 VNC 클라이언트 설치

1. vinagre 패키지 설치 -> GNOME 용 VNC 클라이언트
2. vinagre <IP 주소>:<디스플레이 번호> 를 입력하여 VNC 를 실행한다.
3. 암호를 입력하여 접속한다.

2) 윈도우용 VNC 클라이언트 설치

1. UltraVNC 프로그램을 설치한다.
2. UltraVNC Viewer 를 실행한다.
3. 서버 주소를 입력하여 접속을 시도한다.
4. 암호를 입력하여 접속한다.

메모 포함[이228]: 여러 가지 윈도우용 VNC 클라이언트 중 하나.

3. 파일 송수신

원격으로 파일을 주고받는 기능.

1. FTP (file transfer protocol)

파일 송수신에 사용하는 프로토콜.

TCP/IP 프로토콜의 5 계층에 있음.

리눅스 뿐만 아니라 이 프로토콜을 따르는 다른 운영체제와 파일을 주고받을 수도 있음.

텔넷, VNC 처럼 FRP 또한 사용하려면 FTP 서버와 FTP 클라이언트가 설치되어 있어야 함.

2. FTP 서버

1) FTP 서버 설치

FTP 서버 중 하나인 vsFTPD 패키지를 설치. -> FRP 는 설치하자마자 동작을 시작함.

2) FTP 서버 설정

FTP 가 정상적으로 동작하는지 확인.

메모 포함[이229]: FTP는 텔넷처럼 xinetd 슈퍼데몬에 의해 동작하게 할 수도 있지만 여기에서는 독자형 방식으로 설정한다.

3. FTP 클라이언트

리눅스에는 기본적으로 FTP 클라이언트가 설치되어 있음.

1) FTP 서버 접속

1. FTP 서버의 호스트명/IP 주소, 로그인명, 암호를 준비한다.
2. 윈도우의 cmd 창에서 ftp 를 입력하여 리눅스 시스템에 접속한다.
3. 로그인명과 암호를 입력하여 접속한다.

2) 파일 송수신하기

1. FTP 서버에 접속한다.
2. pwd, dir, ls 명령 등으로 파일명을 확인한다.
3. get, mget 으로 파일을 가져온다. / put, mput 으로 파일을 보낸다.

get, put 은 한 번에 하나의 파일만을 송수신할 수 있음.
mget, mput 은 한 번의 여러 개의 파일을 송수신할 수 있음.

prompt 를 입력해 대화형 모드를 끌 수 있음.

get <file>
mget <file> <file> . . .

3) FTP 내부 명령

메모 포함[이230]: get, mget, prompt 정도만 암기한다.

표 12-1 ftp의 내부 명령

내부 명령	기능
cd 원격 디렉터리	원격 호스트의 디렉터리를 이동한다.
lcd 지역 디렉터리	지역 호스트의 디렉터리를 이동한다.
pwd	원격 호스트의 디렉터리를 출력한다.
!pwd	지역 호스트의 디렉터리를 출력한다.
ls 또는 dir	원격 호스트의 파일 목록을 출력한다. dir 명령은 상세한 파일 정보를 출력한다.
!ls	지역 호스트의 파일 목록을 출력한다.
mkdir 원격 디렉터리	원격 호스트에 디렉터리를 생성한다.
rmdir 원격 디렉터리	원격 호스트의 디렉터리를 삭제한다.
get 원격 파일명 [지역 파일명]	원격 파일 하나를 지역 호스트로 가져온다. 지역 파일명을 지정하면 지정한 파일명으로 저장하고, 지정하지 않으면 원격 파일명과 동일한 파일명으로 저장한다.
mget 원격 파일명	원격 호스트에서 여러 개의 파일을 가져온다.
put 지역 파일명 [원격 파일명]	지역 파일 하나를 원격 호스트로 보낸다. 원격 파일명을 지정하면 지정한 파일명으로 저장하고, 지정하지 않으면 지역 파일명과 동일한 파일명으로 저장한다.
mput 지역 파일명	여러 개의 지역 파일을 보낸다.
prompt	mget이나 mput 명령 사용 시 파일 전송 여부를 물어볼 것인지를 결정한다.
hash	파일이 전송되는 동안 #를 출력하여 진행 상황을 알려준다.
bin	바이너리 파일을 송수신할 것임을 지정한다.
bye	ftp를 종료한다.
open	ftp로 접속할 호스트를 입력하도록 한다.
user	사용자명을 다시 입력할 수 있도록 한다.
? 또는 help [명령]	명령에 대한 도움말을 출력한다.

4. 익명 FTP 사용하기

FTP 서비스를 제공하는 호스트에 등록된 로그인명을 몰라도 익명으로 FTP 를 사용할 수 있게 하는 기능.

anonymous 라는 로그인명을 사용함.
암호로는 자신의 이메일 주소를 입력함.

우분투 vsFTPD 는 기본적으로 익명 FTP 를 허용하지 않음. -> 설정 파일을 수정하여 허용 가능.

10. Samba

1. Samba

1. Samba (이기종 연결방법)

분산파일을 공유하는 프로토콜.
윈도우와 리눅스 사이에서 디렉토리/폴더를 공유할 수 있게 하는 프로토콜.

1) 클라이언트와 서버

리눅스의 디렉토리를 윈도우로 가져옴 -> 윈도우가 삼바 클라이언트이고 리눅스가 삼바 서버임.
윈도우의 파일을 리눅스로 가져옴 -> 리눅스가 삼바 클라이언트이고 윈도우가 삼바 서버임.

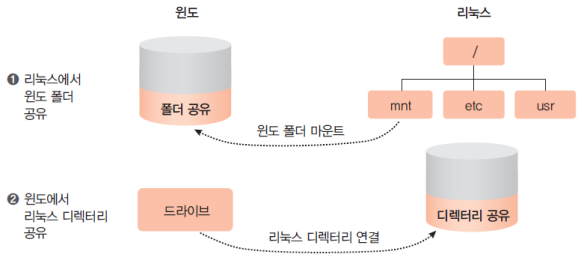


그림 14-3 윈도와 리눅스의 폴더·디렉터리 공유 방법

메모 포함[이231]: 서로 다른 종류의 os를 연결하는 것.
여기서는 윈도우와 리눅스를 연결한다.

2. 리눅스에서 윈도우의 폴더 가져오기

1) 윈도우

리눅스 사용자를 추가하고 폴더를 공유해 두어야 함.

윈도우에서는 삼바 서버를 별도로 설치할 필요가 없음 -> 폴더 공유 기능을 사용함.

공유 폴더 생성

1. 폴더를 생성하고 속성에서 공유 설정을 사용한다.
2. 설정 중 파일 공유 창에 'Everyone'으로 사용자를 추가한다.
3. 설정을 완료한다.

공유 사용자 추가 -> 리눅스에서 공유 폴더에 접근할 때 사용할 사용자.

1. 사용자 설정에서 계정 추가를 시작한다.
2. 사용자 이름을 root 로 하고 암호를 설정하여 계정을 생성한다.

표 14-3 리눅스에서 윈도 폴더를 공유하기 위해 해야 할 작업

리눅스	윈도
삼바 클라이언트(samba-client) 설치 삼바 마운트(smbmount)	리눅스 사용자 추가 폴더 공유

2) 리눅스

1. smbclient 패키지를 설치한다. (Samba client) -> 서버의 자원에 접근하기 위해 사용하는 클라이언트.
2. smbclient -L <서버 IP 주소> 를 입력하여 윈도우의 자원을 확인한다. -> 이때 설정해둔 암호를 입력해야 함.
3. smbclient -U <계정 이름> 을 입력하여 폴더에 접속한다.

4. cifs-utils 패키지를 설치한다. -> cifs 파일 시스템을 사용하기 위해.
5. 윈도우의 공유 폴더를 리눅스 시스템에 마운트한다. -> 파일시스템을 cifs 로 지정해 줘야 함.

메모 포함[이232]: IP 주소는 윈도우의 cmd 창에서 ipconfig 명령으로 확인할 수 있다.

메모 포함[이233]: 아래와 같은 형식으로 작성해야 한다.

mount -t cifs //<윈도우 IP 주소>/<공유 폴더명> <마운트 포인트>

3. 윈도우에서 리눅스의 디렉토리 가져오기

1) 리눅스

1. Samba 서버 패키지를 설치한다.
2. Samba 설정 파일을 수정한다.
3. 삼바 접속을 허용할 사용자 계정의 암호를 smbpasswd 명령으로 지정한다.

2) 윈도우 -> 딱히 할 작업 없음. 연결만 해주면 됨.

1. 윈도우의 네트워크 드라이브 연결 창을 띄운다.
2. 폴더 란에 //<서버 IP 주소>/<사용자 계정> 을 입력하고 암호를 입력한다. -> 리눅스 사용자의 계정을 입력.
3. 해당 계정의 홈 디렉토리가 윈도우의 드라이브에 연결된다,

메모 포함[이234]: 드라이브 문자는 Z부터 시작해서 거꾸로 순차적으로 할당된다.

표 14-4 리눅스에서 삼바 서버 역할을 위해 해야 할 작업

리눅스	윈도
삼바 서버(samba) 설치 삼바 서버 설정 공유할 디렉터리 생성	리눅스 디렉터리 공유

3) 리눅스 클라이언트에서 접속하기

리눅스에서도 smbclient -U <사용자명>으로 다른 리눅스 시스템에서 생성한 Samba 서버에 접속할 수 있음.