

리눅스

2주차 설치

3주차(16.03.23)

(2장) 디렉터리와 파일 사용하기

//로그인 암호 : 12510096

□ 리눅스란?

- 리눅스 + 유닉스

□ 장점

- 고급언어로 작성돼 다른 곳에 인식이 되는 장점
- 웹 서버(Pedora)를 사용하는 형식
- 명령, 형식, 옵션, 인자를 동시에 쓸 수 있다.

□ 단점

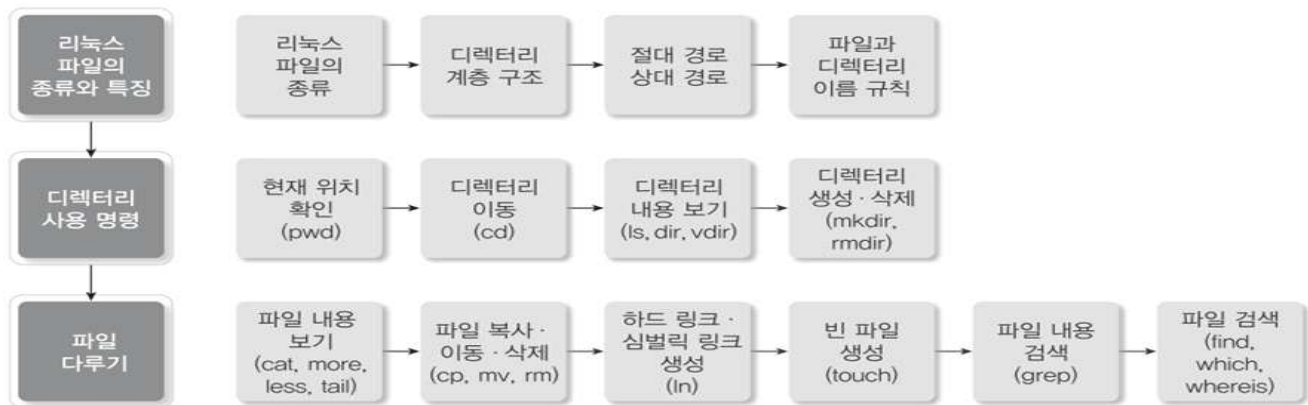
- 운영체제의 크기가 작다

□ 파일 종류

- 일반 파일
- 디렉터리 : `mkdir` 로 파일만들 때 띄워쓰면 파일 여러개 생성가능(`mkdir so1 so2 so3 so4`)
: `rm -rf 최상위폴더` 치면 최상위 폴더의 하위폴더까지 다 삭제
- 심볼릭 링크 : 원본 파일을 대신하여 다른 이름으로 파일명을 지정한 것(바로가기 파일과 비슷)
- 장치파일 : 리눅스에서는 하드 디스크나 키보드 등 각종 장치도 파일로 취급

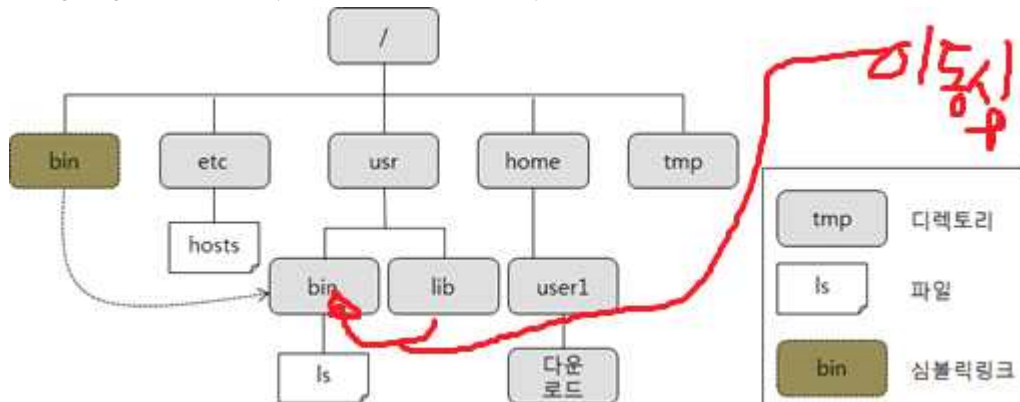
□ 명령의 구조

- 형식, 명령, 옵션, 인자



□ 경로명

- 각 경로를 구분하는 구분자로 슬래시(/)를 사용 ()
- 절대 경로명은 항상 (\)루트 디렉터리부터 시작
- 상대 경로명은 현재 디렉터리를 기준으로 시작 [user1에서 bin으로 갈 때 : ../../usr/bin]



□ 문제1. 현재 user1일 때 이동하는 것

디렉터리/파일명	절대경로	상대경로
/	(copy *.*) /	../..
home	(copy *.*) /home	../
tmp	(copy *.*) /tmp	../..tmp
lib	(copy *.*) /usr/lib	../..usr/lib
ls	(copy *.*) /usr/bin/ls	../..usr/bin/ls

□ 입력 실행키

입력	실행	입력	실행
date	현재 날짜와 시간 출력	ls / dir / vdir	디렉터리 내용 보기
clear	보이는 화면 지우기	mkdir / rmdir (띄워쓰면 여러개 생성)	디렉터리 생성 / 삭제
man	각종 명령 사용법 알려줌	cat / more / less / tail	파일내용보기
passwd	유저 패스워드 변경	cp / mv / rm	파일복사 / 이동 / 삭제
exit / Ctrl+d	접속해제	ln	하드 링크, 심벌릭 링크 생성
Ctrl + w / Ctrl + u	단어 지우기 / 문장 지우기	touch	빈 파일 생성
pwd	현재위치 확인	grep	파일 내용 검색
cd 폴더명 cd. / cd..	대상 폴더로 이동(디렉터리 이동) 이전파일 / 전전파일	find / which / whereis	파일 검색

□ ls

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

4주차(16.03.30)

□ cp : 파일(디렉터리) 복사하기

- * cp. ../*.mp3 >> cp(복사한다) .(현재폴더에 있는 것을) ..(한 폴더 아래에)/ *(이름은 같고).mp3(mp3로 바꾼다)
- * 디렉터리 복사시는 cp -r 입력 ex) cp file1 file2 / cp f1 f2 f3 dir1 / cp -r dir1 dir2

cp	
기능	파일이나 디렉터를 복사한다.
형식	cp [옵션] 파일명1/디렉터리명1 파일명2/디렉터리명2
옵션	-i : 대화식 복사 방법으로 파일명2가 이미 존재할 경우 덮어쓸 것인지 물어본다. -r : 디렉터를 복사할 때 지정한다.
사용 예	cp file1 file2 cp f1 f2 f3 dir1 cp -r dir1 dir2

- inode : 리눅스는 윈도우와 달리 사용자가 싱글이 아님

inode: 파일에 대한 정보를 가지고 있는 특별한 구조체로서 외부적으로는 번호로 표시되고, 내부적으로는 파일의 종류 및 크기, 소유자, 파일 변경 시간, 파일명 등 파일 상세 정보와 데이터 블록의 주소를 저장

□ ln : 파일 링크 만들기

- * 하드 링크 : 파일에 여러 개의 이름을 붙일 수 있는데, 이때 붙이는 파일명을 하드링크 / 동일한 파일을 있는 것처럼 만든 것 // 파일을 복사 붙여넣기 한 것.
- * 심벌릭 링크 : 원본 파일의 경로만 가지는 정보 // 바로가기와 같은 것
- * 링크와 복사의 차이 : 링크는 원본 그대로의 번호를 사용하고, 복사는 원본과 다른 파일을 새로 생성하여 번호가 원본과 다름

■ 심벌릭 링크와 하드 링크의 차이

- 심벌릭 링크는 하드 링크와 비교하여 다음과 같은 몇 가지 특징이 있다.
 - 파일의 종류가 l(소문자 l)로 표시된다.
 - 하드 링크의 개수가 하나이다. 즉, 원본 파일에 이름을 추가하는 것이 아니다.
 - 파일 이름 뒤에 원본 파일의 이름이 표시된다(->data1).
 - inode 번호가 원본 파일과 다르다. 즉, 원본 파일과 심벌릭 링크 파일은 별개의 파일이다.
 - 디렉터리에 심벌릭 링크 생성 가능
 - 파일시스템이 달라도 심벌릭 링크 생성 가능
- 심벌릭 링크 파일의 내용은 원본 파일의 경로
- 심벌릭 링크에서는 원본 파일이 삭제되면 심벌릭 링크로 연결할 수 없다는 점을 주의해야 한다

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

- * 만드는 법 : 1. ln [옵션] 원본파일명 링크파일명
2. ln data data1 // data1.l 하드링크파일 생성
3. ln -s data data2 // data2.sl 심벌릭링크파일 생성

□ find

- /usr 디렉터리에서 ls 파일의 위치를 검색 시 접근권한이 없는 디렉터리는 '허가 거부' 출력
// 특정 사용자 계정이 소유자인 파일을 찾고 싶으면
find /home -user user1(user1이란 사용자의 파일을 찾는 것)
: 네이버 블로그에 불법으로 올린 파일이 있는 것을 찾는 것.

5주차(16.04.06)

(3장) 문서 편집하기

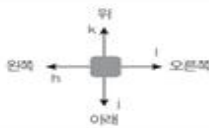
□ vi (모드형) [문서 편집같은 것임]

- * 모드형 : 입력모드와 명령 모드가 구분
 - 입력모드는 텍스트 입력, 명령모드는 텍스트를 수정 및 삭제, 복사 등 편집
 - vi는 모드형 편집기 (터미널에서 vi를 치면 vi(편집모드)가 됨)
- * 비모드형 (한글과 워드가 비모드형 편집기)
 - 입력 과 명령 모드가 구분 돼있지 않음, 편집 기능을 Ctrl이나 Alt같은 특수키와 함께 사용

명령모드 - vi 저장과 종료 명령

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

명령모드 - 커서 이동 명령 / 입력모드 전환

명령 키	기능															
k	커서를 한 행 위로 이동한다.															
j	커서를 한 행 아래로 이동한다.															
l	커서를 한 글자 오른쪽으로 이동한다.															
h	커서를 한 글자 왼쪽으로 이동한다.															
^ 또는 O	커서를 현재 행의 처음으로 이동한다.	[표 3-4] 입력 모드 전환 명령 키 <table border="1"> <thead> <tr> <th>명령 키</th> <th>기능</th> </tr> </thead> <tbody> <tr> <td>i</td> <td>커서 앞에 입력한다(현재 커서 자리에 입력한다).</td> </tr> <tr> <td>a</td> <td>커서 뒤에 입력한다(현재 커서 다음 자리에 입력한다).</td> </tr> <tr> <td>o</td> <td>커서가 위치한 행의 다음 행에 입력한다.</td> </tr> <tr> <td>I</td> <td>커서가 위치한 행의 첫 칼럼으로 이동하여 입력한다.</td> </tr> <tr> <td>A</td> <td>커서가 위치한 행의 마지막 칼럼으로 이동하여 입력한다.</td> </tr> <tr> <td>O</td> <td>커서가 위치한 행의 앞 행에 입력한다.</td> </tr> </tbody> </table>	명령 키	기능	i	커서 앞에 입력한다(현재 커서 자리에 입력한다).	a	커서 뒤에 입력한다(현재 커서 다음 자리에 입력한다).	o	커서가 위치한 행의 다음 행에 입력한다.	I	커서가 위치한 행의 첫 칼럼으로 이동하여 입력한다.	A	커서가 위치한 행의 마지막 칼럼으로 이동하여 입력한다.	O	커서가 위치한 행의 앞 행에 입력한다.
명령 키	기능															
i	커서 앞에 입력한다(현재 커서 자리에 입력한다).															
a	커서 뒤에 입력한다(현재 커서 다음 자리에 입력한다).															
o	커서가 위치한 행의 다음 행에 입력한다.															
I	커서가 위치한 행의 첫 칼럼으로 이동하여 입력한다.															
A	커서가 위치한 행의 마지막 칼럼으로 이동하여 입력한다.															
O	커서가 위치한 행의 앞 행에 입력한다.															
\$	커서를 현재 행의 마지막으로 이동한다.															
-	커서를 앞 행의 처음으로 이동한다.															
+ 또는 Enter	커서를 다음 행의 처음으로 이동한다.															
H	커서를 화면의 맨 윗행으로 이동한다.															
M	커서를 화면의 중간 행으로 이동한다.															
L	커서를 화면의 맨 아랫행으로 이동한다.															
w	커서를 다음 단어의 첫 글자 위치로 이동한다.															
b	커서를 앞 단어의 첫 글자 위치로 이동한다.															
e	커서를 다음 단어의 마지막 글자 위치로 이동한다.															

명령모드 - 내용 수정 명령

[표 3-9] 내용 수정 명령 키

명령 키	기능
r	커서가 위치한 글자를 다른 글자로 수정한다.
cw, #cw	커서 위치부터 현재 단어의 끝까지 수정한다. #에는 수정할 단어의 수를 지정한다. 예를 들어 3cw는 커서 위치부터 세 단어를 수정한다.
s, #s	커서 위치부터 [Esc] 키를 입력할 때까지 수정한다. #에는 수정할 글자의 수를 지정한다. 예를 들어 5s는 커서 위치부터 다섯 글자를 수정한다.
cc	커서가 위치한 행의 내용을 모두 수정한다.
C	커서 위치부터 행의 끝까지 수정한다.

[표 3-10] 내용 삭제 명령 키

명령 키	기능
x, #x	커서 위치의 글자를 삭제한다. #에는 삭제할 글자 수를 지정한다. 예를 들어 3x는 세 글자를 삭제한다.
dw, #dw	커서 위치의 단어를 삭제한다. #에는 삭제할 단어 수를 지정한다.
dd, #dd	커서 위치의 행을 삭제한다. #에는 삭제할 행의 수를 지정한다. 예를 들어 5dd는 커서 위치부터 다섯 행을 삭제한다.
D([Shift]+d)	커서 위치부터 행의 끝까지 삭제한다.

[표 3-12] 복사하기, 잘라내기, 붙이기 명령 키

명령 키	기능
yy, #yy	커서가 위치한 행을 복사한다. #에는 복사할 행의 수를 지정한다. 예를 들어 3yy는 세 행을 복사한다.
p	커서가 위치한 행의 아래쪽에 붙인다.
P	커서가 위치한 행의 위쪽에 붙인다.
dd, #dd	커서가 위치한 행을 잘라낸다. 삭제와 같은 기능이다. #에는 잘라낼 행의 수를 지정한다. 예를 들어 3dd는 세 행을 잘라낸다.

- 3번째 줄로 이동 : 3G or :3 엔터 / - 한 글자 수정 : r / 단어 수정 : cw, #s

[표 3-16] 바꾸기 명령 키

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

명령모드 - 이전 명령 취소		검색 명령 키	
명령 키	기능	명령 키	기능
u	명령을 취소한다.	/문자열	문자열을 아래 방향으로 검색한다.
U	해당 행에서 한 모든 명령을 취소한다.	?문자열	문자열을 위 방향으로 검색한다.
:e!	마지막으로 저장한 내용 이후의 것을 버리고 새로 작업한다.	n	원래 찾던 방향으로 다음 문자열을 찾는다.
		N	역방향으로 다음 문자열을 찾는다.

vi모드 파일 불러오기

[표 3-17] 파일 관련 명령 키

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

[표 3-19] 기타 명령 키

명령 키	기능
[Ctrl]+u (소문자 L)	현재 화면을 다시 출력한다.
[Ctrl]+g	현재 행 번호를 마지막 행에 출력한다.
[Shift]+j (대문자 J)	현재 행과 아랫행을 연결하여 한 행으로 만든다.
.	바로 직전에 했던 명령을 반복한다.

[표 3-18] 셸 명령 실행 명령 키

명령 키	기능
:! 셸 명령	vi 작업을 잠시 중단하고 셸 명령을 실행한다(vi로 돌아오려면 [Enter] 키를 입력해야 한다).
:sh	vi를 잠시 빠져나가서 셸 명령을 실행한다(vi로 돌아오려면 exit 명령을 입력해야 한다).

vi모드 환경설정 - .exrc파일에 설정하기

[표 3-20] vi 환경 설정 명령 키

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

실습

현재 위치를 확인한다. [pwd]

홈 디렉터리가 아니면 홈 디렉터리로 이동한다. [cd home]

실습을 위한 기본 디렉터리를 만든다. [mkdir ch2]

ch2 디렉터리를 만들고 그 디렉터리로 이동하여 현재 위치를 확인한다.

[mkdir ch2 - cd ch2 - pwd]

one, two, three 디렉터리를 동시에 만든다. [mkdir one two three]

one 디렉터리 아래에 tmp/test 디렉터리를 만든다. 중간 경로인 tmp 디렉터리가 자동 생성되도록 한다. [cd one - mkdir -p tmp/mid/test]

two, three 디렉터리를 동시에 삭제한다. [rmdir two three]

실습을 마치고 홈 디렉터리로 이동한다. [cd home]

===== 중간고사 범위 끝 =====

(4장) 셸 : 명령어 해석기 기능, 프로그래밍 기능, 사용자 환경 설정 기능

- 명령어 해석기 기능 : 사용자와 커널 사이에서 명령을 해석
- 프로그래밍 기능 : 셸은 자체 내에 프로그래밍 기능이 있음 (반복적으로 수행하는 작업을 하나의 프로그램으로 작성 가능)
- 사용자 환경 설정 기능 : 사용자 환경을 설정할 수 있도록 초기화 파일 기능을 제공
 - 초기화 파일에는 명령을 찾아오는 경로를 설정
 - 파일과 디렉터리를 새로 생성할 때 기본 권한을 설정
 - 다양한 환경 변수 등을 설정

*** 셸의 종류** // 중요하진 않음

- * 본 셸 : 속도가 빠름 (명령 이름 sh)
- * 콘 셸 : 본 셸의 호환성 유지, 히스토리, 에일리어스 등의 C셸 특징 제공하면서 속도도 빠름 (명령 이름 ksh)
- * C 셸 : C언어와 유사, 에일리어스나 히스토리 같은 사용자 편의 기능 포함, 속도는 느림 (명령 이름 csh)
- * 배시 셸 : 본 셸과 호환성 유지, C셸, 콘 셸 기능 포함 (명령 이름 bash) // 리눅스의 기본 셸

*** 셸의 명령어**

- * echo : 화면에 한 줄의 문자열을 출력 한다.
 - echo text / 출력 : txt // echo "linux fedora" / 출력: linux fedora
- * printf : %지시자와 \문자를 이용하여 출력 형식을 지정 가능.
 - printf "linux fedora\n" // 출력 : linux fedora
 - printf "%d + %d = %d \n" 10 10 20 // 출력 : 10 + 10 =20

[표 4-1] 특수 문자 *

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 밑의 ch3 디렉터리로 복사한다.
ls -l h*d	파일명이 h로 시작하고 d로 끝나는 모든 파일의 상세 정보를 출력한다. hd, had, hard, h12345d 등 이 조건에 맞는 모든 파일의 정보를 볼 수 있다.

- * 특수 문자 ? 와 [] : ?는 하나의 문자를 나타내는 데 사용, []는 괄호 안의 문자 중 하나를 나타냄

[표 4-2] 특수 문자 ?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 제외된다.
ls -l tmp[135].txt	tmp 다음에 1, 3, 5 중 한 글자가 오고 파일의 확장자가 txt인 모든 파일의 이름을 출력한다. tmp1.txt, tmp3.txt, tmp5.txt 파일이 있으면 해당 파일의 상세 정보를 출력한다. tmp.txt는 제외된다.
ls -l tmp[1-3].txt	[1-3]은 1부터 3까지의 범위를 의미한다. 따라서 ls -l tmp[123].txt와 결과가 같다. tmp1.txt, tmp2.txt, tmp3.txt 파일이 있으면 해당 파일의 상세 정보를 출력한다.
ls [0-9]*	파일명이 숫자로 시작하는 모든 파일 목록을 출력한다.
ls [A-Za-z]*[0-9]	파일명이 영문자로 시작하고 숫자로 끝나는 모든 파일 목록을 출력한다.

- * 특수 문자 ~ 와 - : ~디렉터리를 나타내는 특수 문자

- ~ : 현재 작업 중인 사용자의 홈 디렉터리를 표시
// 다른 사용자의 로그인 ID와 함께 사용하면(~ 로그인 ID) 해당 사용자의 홈 디렉터리 표시
- : cd 명령으로 디렉터를 이전하기 직전의 작업 디렉터리 표시

[표 4-3] 특수 문자 ~와 -

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

- * 특수 문자 ; 과 | : 과 | 는 명령과 명령을 연결
 ‘;’ : 왼쪽부터 차례로 실행
 | : 왼쪽 명령의 실행 결과를 오른쪽 명령의 입력으로 전달)

[표 4-4] 특수 문자 ; 과 |

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

- * 특수 문자 ‘ ’ 와 “ ” : 문자를 감싸 문자열로 만들어주고, 문자열 안에 사용된 특수 문자의 기능을 없앴
 ex) echo ‘\$SELL’ 입력 시 \$SELL 출력
 ex) echo “\$SELL” 입력 시 셀의 종류가 출력 bash
- * 특수 문자 `` : ex) echo “Today is `data`”

□ 셀 특수문자

	문자	내용	예제
출력	echo	문자 출력	ex) echo Hello 입력 시 // Hello
	printf	문자 출력	ex) printf “%d + %d = %d” 10 10 20 // 10+10=20
문자개수	*	all을 의미	ex) cp */tmp 입력시 // 모든 /tmp파일을 복사
	?	문자 하나를 나타냄	ex) ls ?a* 입력 시 // a가 들어가는 이름을 가진 모든 파일을 출력
	[]	[] 안의 문자 중 하나 나타냄	ex) ls [a-z]* 입력 시 // [a-z]사이에 해당하는 이름을 가진 파일을 출력
디렉터리	~	사용자 디렉터리	ex) cp *.txt~/ch3 입력시 //확장자 txt인 모든 파일을 현재 작업중인 사용자의 tmp디렉터리로 복사
	-	이전 작업 디렉터리 (.. 과는 다름)	ex) cd - 입력 시 // 이전 작업 디렉터리로 이동
실행	;	여러 개의 명령을 동시에 실행 (왼쪽부터 차례로 실행)	ex) date; ls; pwd 입력시 // 날짜 출력, 현재 디렉터리 파일목록 출력, 현재 작업 디렉터리 절대경로
		왼쪽 명령의 실행 결과를 오른쪽 명령의 입력으로 전달	ex) ls -all more 입력시 // ls -al의 명령을 more로 전달 (모든 파일의 inode를 출력하고, 파일내용을 봄)
해석기능	‘ ’ (작은따옴표)	문자열 안에 사용된 특수 문자의 기능을 없앴	ex) echo ‘\$SELL’ 입력 시 // \$SELL
	“ ” (큰따옴표)	특수문자 사용 시 셀의 종류가 출력	ex) echo “\$SELL” 입력 시 셀의 종류 출력 // bash
	` ` (물결표시 위)	“ ” 안의 내용을 해석	ex) echo “Today is `data`” 입력 시 //Today is 2016. 04. 20. (수) 23:20:48 KST
상세정보	\	특수 문자 효과를 없애고 일반 문자처럼 처리	ex) ls -lt* 입력 시 (C에서 printf안에 특수문자쓸 때 처럼) // t*라는 이름을 가진 파일 상세정보 출력
입출력 방향을 바꾸는 특수 문자	>	기존 파일의 내용을 삭제하고 새로 결과를 저장(덮어쓰기)	ex) ls -l>res 입력 시 // ls -l명령의 실행 결과를 화면이 아닌 res 파일에 저장
		set +o noclobber로 덮어쓰기 방지할 수 있음 (-o로 해제)	ex) cat > data // 명령 실행 Hello Linux // 글자 입력 ^Z (Ctrl+z Ctrl+d) // 입력 종료
	<	원래 명령 쓰는 것 / 생각가능	ex) cat test 과 cat < test 와 같음
	>>	기존 파일의 내용 뒤에 결과를 추가	ex) ls -l>>res 입력 시 // ls -l명령의 실행 결과를 화면이 아닌 res 파일에 결과 추가 후 저장

□ 셸 변수 설정

- 변수이름 = 문자열 // `Some = test`

* `export` [변수이름] : 셸 환경 변수 설정

- `export [-n] [셸변수]`

* `env` : 전역변수만 출력

* `set` : 함수까지 다 출력

* `unset` : 변수 해제

* `alias`(에일리어스) : 변수에 별명을 붙임

- `alias` 이름='명령'

- `function cdpwd { cd $1;pwd }` 시 `cdpwd`가 `pwd`로 이동하는 함수별명이 생김 //배시셸에서는 `alias`로 인자전달x

* `unalias` : 별명 해제

□ ! : 셸 명령 재실행 (history시 이전에 했던 명령이 나옴 !줄번호로 재실행가능) // 위 방향키도 이전명령

- !!줄번호 : 바로 직전 명령을 재실행

* 명령 편집하기와 재실행

- `man history` : 메뉴의 히스토리를 보여줌

□ PS1 : 프롬프트 설정 // [user1@localhost ch4]\$ <<이건 프롬프트 값 // [사용자 구분자(@) 서버명]

- `PS1='Linux'`

- `PS1="\e[31;4mLinux $\e[0;0m"` // 사용시 `Linux $` 로 바뀜 [31 이 빨간색 숫자

- 프롬프트에서 사용할 수 있는 이스케이프 문자는 \ (역슬래쉬) 로 시작함(ppt참조)

□ 환경설정 파일 : 사용자 로그인 시 자동 실행 명령을 저장한 것

- 시스템 환경 설정 파일과 사용자 환경 설정 파일이 있음
- 셸마다 다른 이름의 파일을 사용

* 시스템 환경 설정 파일 : 시스템을 사용하는 전체 사용자의 공통 환경을 설정

- `/etc/profile` : 시스템 환경 설정(기타 등등)
- `/etc/bashrc` : 시스템 함수와 에일리어스 설정/ 기본 프롬프트 설정/ 서브 셸 // `vi`로 바꿈
- `/etc/.login` : 시스템 로그인시 실행되는 파일
- `/etc/profile.d/*.sh` : 언어나 명령별로 각각 필요한 환경을 설정 (필요시 설정 파일을 추가)

* 사용자 환경 설정 파일 : 각 사용자의 홈 디렉터리에 숨김 파일로 생성

- `~/.bash_profile` : 경로 추가 등 사용자가 정의하는 환경 설정
- `~/.bashrc` : 사용자가 정의하는 환경 수정
- 파일 적용 시 `. .bashrc` 또는 `source .bashrc` 로 적용
- `~/.bash_logout` : 로그아웃 시 실행할 필요가 있는 함수 등을 설정

(5장) 파일 접근 권한

□ 접근권한

- 읽기 권한(r), 쓰기 권한(w), 실행 권한(x), 권한이 없을 때 (-)
- rwx/r--/r-- 순서로하며, 소유자,그룹,기타사용자 순으로 읽음

* **file 파일명** : 지정한 파일의 종류를 알려줌

* **groups [사용자명]** : 사용자가 속한 그룹을 알려줌

* **chmod [옵션]** : 파일이나 디렉터리의 접근 권한을 변경 / -R : 하위 디렉터리까지 모두 변경할 수 있다.

- chmod + 사용자 카테고리 문자 + 연산자기호 + 접근권한문자 + 파일명

구분	문자/기호	의미
사용자 카테고리 문자	u	파일 소유자
	g	소유자가 속한 그룹
	o	소유자와 그룹 이외의 기타 사용자
	a	전체 사용자
연산자 기호	+	권한 부여
	-	권한 제거
	=	접근 권한 설정
접근 권한 문자	r	읽기 권한
	w	쓰기 권한
	x	실행 권한

권한 표기	의미
u+w	소유자(u)에게 쓰기(w) 권한 부여(+)
u-x	소유자(u)에게 실행(x) 권한 제거(-)
g+wx	그룹(g)에 쓰기(w)와 실행(x) 권한 부여(+)
+wx	모든 사용자에게 쓰기(w)와 실행(x) 권한 부여(+)
u=rwx	소유자(u)에게 읽기(r), 쓰기(w), 실행(x) 권한 부여(=)
go+w	그룹(g)과 기타 사용자(o)에게 쓰기(w) 권한 부여(+)
u+x,go+w	소유자(u)에게 실행(x) 권한을 부여하고(+) 그룹(g)과 기타사용자(o)에게 쓰기(w) 권한 부여(+)

* 권한을 숫자로 표기 : rwx를 2진수로 111, 7로 표기 가능 // chmod + 사용자 숫자 + 그룹 숫자 + 기타 사용자 숫자 + 파일명

- ex) chmod 740 test.txt // test.txt파일에 대하여 사용자는 rwx, 그룹은 r, 기타사용자는 -의 권한을 줌

* **umask [옵션] [마스크 값]** : 기본 접근 권한을 출력하거나 변경 / -S : 마스크 값을 문자로 출력

* 마스크 값 : 파일이나 디렉터리 생성 시 부여하지 않을 권한을 지정해 놓는 것

- 002일 경우 -----w- 이고, 기타 사용자에게 쓰기 권한은 부여하지 않겠다는 뜻

요청 권한	1	1	0	0	110 110 110
마스크	1	0	1	0	000 000 010
부여된 권한	0	1	0	0	110 110 100

전체	마스크값	일반 파일	디렉터리	의미
666(777)	022	644	755	그룹과 기타 사용자는 읽기만 가능
	027	640	750	그룹은 읽기와 실행만 가능, 기타 사용자의 접근권한은 모두 제거

* **SetUID(유저)** : 해당 파일이 실행되는 동안에는 파일을 실행한 사용자의 권한이 아니라 파일 소유자의 권한으로 실행

- 파일에 SetUID 설정 : 접근 권한에서 맨 앞자리에 4를 설정 // 해제할 때는 4를 빼줌
- ex) chmod 4755 set.exe // 실행시 -rwsr-xr-x 로 s가 설정

* **SetGID(그룹)** : SetGID가 설정된 파일을 실행하면 해당 파일이 실행하는 동안 파일소유 그룹의 권한으로 실행

- 파일에 SetGID 설정 : 접근 권한에서 맨 앞자리에 2를 설정 // 해제할 때는 2를 빼줌

* **스티키 비트(디렉터리)** : 디렉터리에 스티키 비트가 설정되어 있으면 누구나 디렉터리내의 파일 생성 가능

- 디렉터리에 스티키 비트 설정 : 접근 권한에서 맨 앞자리에 1을 설정 // 해제할 때는 1을 빼줌

* 프로세스는 부모-자식 관계를 가지고 있음 //ex) 부모(bash셸)-자식(ls, cd 등 명령어)

- * 데몬 프로세스 : 특정 서비스를 제공하기 위해 존재하며 리눅스 커널에 의해 실행 (부모에 해당)
- * 고아 프로세스 : 부모 프로세스가 종료 되었지만 자식 프로세스가 실행중인 것 (자식에 해당)
- * 좀비 프로세스 : 바이러스 같은 것, 자식 프로세스가 실행을 종료했는데도 테이블 목록에 남아있는 경우

* 유닉스(SVR4) 옵션 : 묶어서 사용할 수 있고, 붙임표로 시작한다 // ex) ps -ef

* BSD 옵션 : 묶어서 사용할 수 있고, 붙임표로 시작하지 않는다. // ex) ps aux

* **top** : 현재 실행 중인 프로세스에 대한 정보를 주기적으로 출력

* **kill [시그널] PID ...** : 지정한 시그널을 프로세스에 보낸다 /

시그널	의미
-2	인터럽트 시그널을 보낸다 (Ctrl + C)
-9	프로세스를 강제로 종료
-15	프로세스가 관련된 파일을 정리하고 프로세스를 종료. 종료되지 않는 프로세스가 있을 수 있다.

* **pkill** : 프로세스 번호 대신 명령어 이름으로 프로세스를 찾아 종료한다.

* **skill** : 유저 이름을 넣고 종료

□ 포그라운드, 백그라운드 프로세스

* **포그라운드 작업** : 사용자가 입력한 명령이 실행되어 결과가 출력될 때까지 기다려야 하는 방식

* **백그라운드 작업** : 명령의 처리가 끝나는 것과 관계없이 곧바로 프롬프트가 출력되어 다른 작업 가능.
(뒤에 &붙이면 백그라운드작업)

□ 작업제어

* **시스템정보 (GNOME)** : 리눅스의 작업관리자

* **jobs [%작업번호]** : 백그라운드 작업을 모두 보여준다. 특정 작업 번호를 지정하면 해당 작업의 정보만 보여준다.

옵션	의미
%번호	해당 번호의 작업 정보를 출력
%+ 또는 %%	작업 순서가 +인 작업 정보를 출력
%-	작업 순서가 -인 작업 정보를 출력

* 작업 전환하기

명령	의미
Ctrl+z 또는 stop [% 작업번호]	포그라운드 작업을 중지 (종료가 아닌 잠시 중단)
bg [%작업 번호]	작업 번호가 지시하는 작업을 백그라운드 작업으로 전환
fg [%작업 번호]	작업 번호가 지시하는 작업을 포그라운드 작업으로 전환

* **ctrl + z** : 실행중인 프로그램을 백그라운드로 전환

* **ctrl + c** : 포그라운드 작업 취소

* **nohup 명령&** : 로그아웃 후에도 백그라운드 작업 계속 실행 // ex) nohup find / -name passwd &
// 명령어 뒤에 &를 붙이면 백그라운드 작업

* **at [옵션] 시간** : 작업 예약, 특정한 시간에도 작업을 수행할 수 있도록 예약가능 //ex) at -l

옵션	의미
-l	현재 실행 대기 중인 명령의 전체 목록을 출력(atq 명령과 동일)
-r 작업 번호	현재 실행 대기 중인 명령에서 해당 작업 번호를 삭제(atrm과 동일)
-m	출력 결과가 없더라도 작업이 완료되면 사용자에게 메일로 알려준다.
-f 파일	표준 입력 대신 실행할 명령을 파일로 지정

- Ctrl + d 시 명령 종료 - atq : 작업 확인 - at -d, atrm = 작업예약 삭제하기.

- /etc/at.deny : deny파일에 등록된 사용자는 at사용이 불가능(deny파일이 있을 경우)

- /etc/at.allow : allow파일에 등록된 사용자는 at 사용이 가능(allow 파일이 있을 경우)

- deny와 allow 둘다 없는 경우 : root만 at 사용이 가능

// 이 규칙들은 crontab도 동일

* **crontab [-u 사용자ID] [옵션] [파일 이름]** : 정해진 시간에 반복 실행

옵션	의미
-e	사용자의 crontab 파일을 편집
-l	crontab 파일의 목록을 출력
-r	crontab 파일을 삭제

12주차 (16.05.25)

- `cd /mnt` : 이동
- `cp /etc/host` : 복사
- `ls` : 확인

□ 리눅스 파일 시스템 구조

□ ext4 블록그룹 유형

- ext4 블록 그룹 유형 : 동일한 정보를 그룹0에 다 가지고 있고, 상위을 제외한 정보를 a그룹에, b그룹에는 정보만 가지고 있음.
 - * 블록 그룹 0 : 파일 시스템의 첫 번째 블록 그룹으로 특별하게 그룹 0 패딩과 슈퍼블록, 그룹 디스크립터를 가지고 있다.
 - * 블록 그룹 a : 파일 시스템에서 첫 번째 블록 그룹이 아닌 블록 그룹으로 그룹 0 패딩이 없으나 슈퍼블록과 그룹 디스크립터에 대한 복사본을 가지고 있다.
 - * 블록 그룹 b : 파일 시스템에서 첫 번째 블록 그룹이 아닌 블록 그룹으로 그룹 0 패딩, 슈퍼블록, 그룹 디스크립터가 없고 바로 데이터 블록 비트맵으로 시작한다.

○ ext 파일 시스템의 구조 : 데이터의 영역들이 node 개념으로 관리됨(inode)



[그림 7-3] ext4 파일 시스템의 구조

* 그룹 0패딩 : 블록 그룹 0의 첫 1,024바이트는 특별한 용도로 사용되는데, x86 부트 섹터와 부가정보를 저장

* 슈퍼블록 : 파일 시스템과 관련된 다양한 정보가 저장

- 전체 inode의 개수
- 할당되지 않은 블록(free block)의 개수
- 첫 번째 데이터 블록의 주소
- 그룹당 블록의 개수
- 파일 시스템의 상태
- 전체 블록의 개수
- 할당되지 않은 inode(free inode)의 개수
- 블록의 크기
- 마운트 시간
- 그룹 디스크립터의 크기
- * 슈퍼블록에 문제가 생길 경우 전체 파일 시스템을 사용할 수 없게됨
- * 슈퍼블록을 다른 블록 그룹에 복사하고, 블록 그룹 0의 슈퍼블록을 읽을 수 없을 경우 복사본을 사용하여 복구

* 그룹 디스크립터와 GDT 예약 블록 : 그룹 디스크립터도 블록 그룹 0에 있는 것으로 슈퍼 블록의 다음에 위치

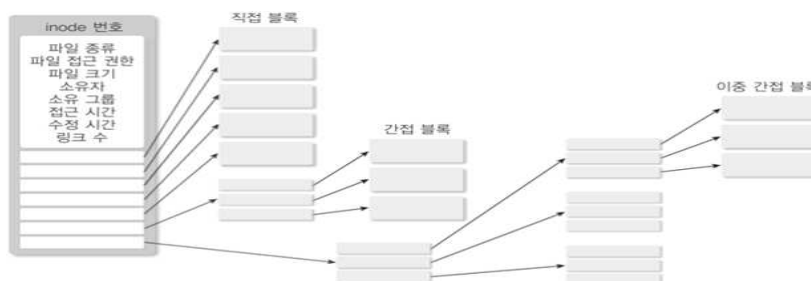
- 그룹 디스크립터에 저장되는 정보
 - 블록 비트맵의 주소
 - inode 테이블의 주소
 - 할당되지 않은 inode의 개수
 - 블록 비트맵, inode 비트맵 체크섬
 - inode 비트맵의 주소
 - 할당되지 않은 블록의 개수
 - 디렉터리의 개수

* 데이터 블록 비트맵 : 블록 그룹에 포함된 데이터 블록의 사용 여부를 확인하는 데 사용

* inode 비트맵 : inode 테이블의 항목(inode)이 사용 중인지를 표시

* 데이터 블록 : 실제 데이터를 저장

* inode : 파일 정보를 저장



[그림 7-4] inode의 구조

□ 디스크 추가 설치

○ 디스크 마운트 - 마운트 포인트 준비하기 - 파일 시스템 마운트하기

○ fdisk [옵션] 장치명 : ex) fdisk /dev/sdb , fdisk -l

옵션	의미
-b <크기>	섹터 크기를 지정한다(512, 1024, 2048, 4096)
-l	파티션 테이블을 출력

*fdisk 내부 명령

내부명령	기능	내부명령	기능
a	부팅 파티션을 설정	p	파티션 테이블을 출력
b	BSD 디스크 라벨을 편집	q	작업 내용을 저장하지 않고 종료
c	도스 호환성을 설정	s	새로운 빈 Sun 디스크 라벨을 생성
d	파티션을 삭제	t	파티션의 시스템 ID를 변경한다 (파일 시스템 종류)
l	사용 가능한 파티션 종류를 출력	u	항목 정보를 변경, 출력한다.
m	도움말 출력	v	파티션 테이블을 검사
n	새로운 파티션을 추가	w	파티션 정보를 디스크에 저장하고 종료
o	새로운 빈 DOS 파티션을 생성	x	실린더 개수 변경 등 전문가를 위한 부가적 기능

○ 마운트란?

mnt : 장치를 이야기함 (c드라이브와 같은 개념) / 리눅스에서는 모든 장치가 디렉터리 개념
ex) mkdir /mnt/hdd1 // 앞에 mnt가 붙으면 다른 장치라고 보면 됨.
/ : 마운트 포인트임. (설치할 때 마운트 포인트를 지정했음)

○ df [옵션] [파일 시스템] : 디스크의 남은 공간에 대한 정보를 출력 // 파일 시스템 별

옵션	기능
-a	모든 파일 시스템을 대상으로 디스크 사용량을 확인
-k	디스크 사용량을 KB 단위로 출력
-m	디스크 사용량을 MB 단위로 출력
-h	디스크 사용량을 알기 쉬운 단위(GB, MB, KB 등)로 출력
-t 파일 시스템 종류	지정한 파일 시스템 종류에 해당하는 디스크 사용량을 출력
-T	파일 시스템의 종류도 출력

○ du [옵션] [디렉터리] : 디스크의 사용 공간에 대한 정보를 출력 // 디렉터리나 사용자 별

옵션	기능
-s	특정 디렉터리의 전체 사용량을 출력
-h	디스크 사용량을 알기 쉬운 단위(GB, MB, KB 등)로 출력

○ fsck [옵션] 장치명 : 리눅스의 파일 시스템 검사 (복구 작업도 수행)

옵션	기능
-f	강제로 검사
-b 슈퍼블록	슈퍼블록으로 지정한 백업 슈퍼블록을 사용
-y	모든 질문에 yes로 대답하도록 한다.
-a	파일 시스템 검사에서 문제가 발생했을 때 자동으로 복구한다.

○ e2fsck [옵션] 장치명 : 리눅스의 확장 파일 시스템 검사 (복구 작업도 수행)

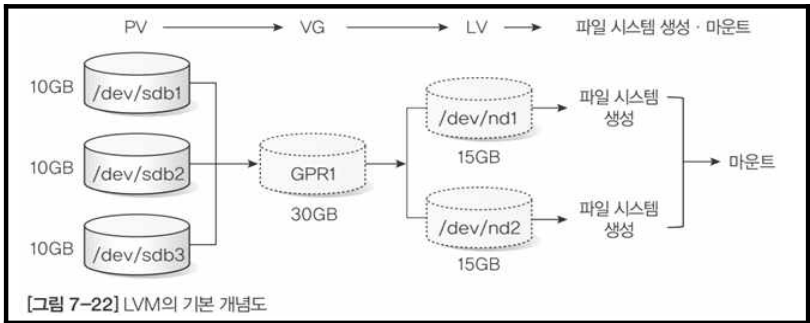
옵션	기능
-f	강제로 검사
-b 슈퍼블록	슈퍼블록으로 지정한 백업 슈퍼블록을 사용
-y	모든 질문에 yes로 대답하도록 한다.
-j ext3/ext4	ext3나 ext4 파일 시스템을 검사할 때 지정

○ badblocks [옵션] 장치명 : 장치의 배드 블록을 검사

옵션	기능
-v	검색 결과를 자세하게 출력
-o 출력 파일	검색한 배드 블록 목록을 지정한 출력 파일에 저장

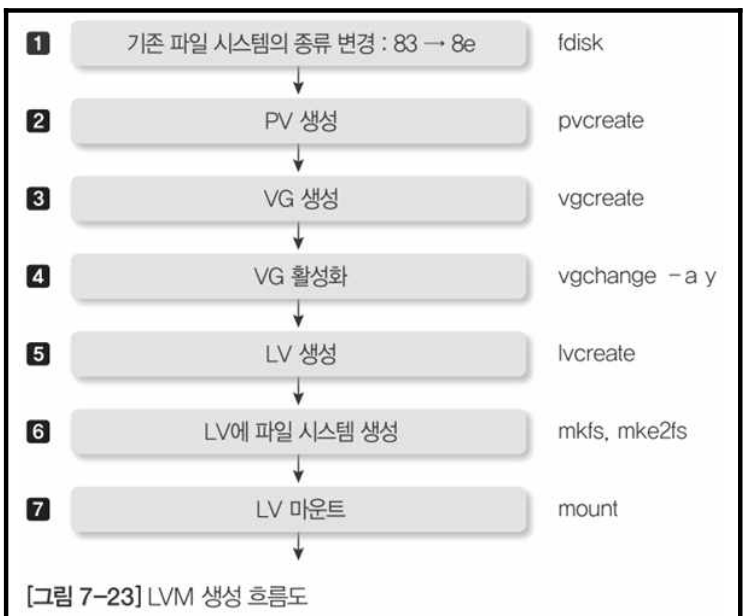
□ LVM / 여러 디스크를 하나처럼 사용하기 (시험에 나옴!)

- * PV(물리 볼륨) : /dev/sdb1, /dev/sdb2 같은 실제 하드디스크의 파티션을 의미
- * VG(볼륨 그룹) : 여러개의 PV를 그룹으로 묶은 것 // ex) /dev/sdb1, /dev/sdb2를 묶은 그룹 GRP1
- * LV(논리 볼륨) : VG를 다시 적절한 크기의 파티션으로 나눌 때 각 파티션을 LV라고 함
- * PE : PV가 가진 일정한 블록을 의미
- * LE : LV가 가진 일정한 블록을 의미



<< 시험에 나옴

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



(8장) 리눅스 시스템 : ■ : PC부팅 ■ : 리눅스 부팅

- [전원on - 바이오스 단계] - [부트 로더 단계 - 커널 초기화 단계 - **systemd 서비스 단계** - 로그인 프롬프트 출력]

□ 리눅스 부팅 단계

□ **systemd 서비스 단계** : 리눅스 본격적으로 동작 단계, 리눅스 안의 모든 데몬을 관리

- init 스크립트를 대체한 것, * **init 프로세스** : 메인 프로세스(조상 프로세스), 리눅스 안의 모든 데몬을 관리
- init은 1번 프로세스의 이름
- 리눅스 설정에는 부트 화면이 출력되지 않고 그림(스플래시) 화면이 출력 됨 // 부트화면 출력시 Ctrl + D
// **init 런레벨** : 2) NFS를 사용하지않음(네트워크를 사용하지 않음)

○ 장점

- 소켓 기반으로 동작하여 inetd와 호환성을 유지한다. // **inetd** : 리눅스에서 관리하는 서버 데몬 (인터넷 개념)
- 셸과 독립적으로 부팅이 가능하다.
- 마운트 제어가 가능하다.
- fsck 제어가 가능하다.
- 시스템 상태에 대한 스냅샷을 유지한다.
- SELinux와 통합이 가능하다.
- 서비스에 시그널을 전달할 수 있다.
- 쉼다운 전에 사용자 세션의 안전한 종료가 가능하다.

□ **systemd 유닛의 종류**

유닛의 종류	기능	예
service	가장 명백한 유닛으로 데몬을 시작, 종료, 재시작, 로딩한다.	atd.service
socket	소켓을 관리하는 유닛으로 AF_INET.AF_INET6, AF_UNIX소켓 스트림과 데이터그램, FIFO를 지원한다.	dbus.socket
device	리눅스 장치 트리에 있는 장치를 관리한다.	dec-sda.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	유닛들을 그룹핑한다(예 : multi-user.target->런레벨 5에 해당하는 유닛)	default.target runlevel0.target
snapshot	다른 유닛을 참조하기 위한 유닛	foo.snapshot
swap	스왑 장치를 관리한다.	foo.swqp
path	경로를 관리한다.	cups.path
timer	타이머와 관련된 기능을 관리한다.	systemd-readahead-done.timer

□ **systemctl [옵션] [명령] [유닛이름] : systemd를 제어 (윈도우 작업관리자-서비스 기능)**

[옵션]	기능
-a	상태와 관계없이 유닛 전체를 출력한다.
-t 유닛 종류	지정한 종류의 유닛만 출력한다.

[명령]	기능
start	유닛을 시작한다.
stop	유닛을 종료한다.
reload	유닛의 설정 파일을 다시 읽어온다.
restart	유닛을 재시작한다.
status	유닛의 상태를 출력한다.
enable	부팅 시 유닛이 시작되도록 설정한다.
disable	부팅 시 유닛이 시작되지 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.

□ **systemd와 런레벨** // 런레벨 : 현재 시스템의 상태를 나타내는 한 자리 숫자

- 런레벨에 대응하는 것이 systemd의 target유닛이다.
- isolate : systemd의 런레벨 변경
- init, telinit : 둘다 런레벨 변경 기능 (구버전 사용자들을 위한 배려)

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

- 런레벨을 0이나 6으로 지정하면 시스템을 종료

○ runlevel : 현재 런레벨 확인

□ **리눅스 시스템 종료**

○ shutdown [옵션] [시간] [메시지] //ex) shutdown -h now

옵션	기능
-k	실제로 시스템을 종료하는 것이 아니라 사용자들에게 메시지만 전달
-r	종료 후 재시작한다.
-h	종료하여 halt 상태로 이동한다.
-f	빠른 재시작으로 이 과정에서 fsck를 생략할 수도 있다.
-c	이전에 내렸던 shutdown 명령을 취소한다.
시간	종료할 시간(hh:mm, +m, now)
메시지	모든 사용자에게 보낼 메시지

- telinit 0 또는 telinit 6 사용
- halt 명령 사용
- poweroff 명령 사용
- reboot 명령 사용

옵션	기능
-n	재시작이나 종료 전에 sync를 호출하지 않는다.
-w	실질적으로 재시작하거나 종료하지는 않지만 wtmp 파일에 기록을 남긴다.
-d	wtmp 파일에 기록을 남기지 않는다. -n 옵션은 -d옵션을 포함한다.
-f	강제로 명령을 실행하며 shutdown을 호출하지 않는다.
-p	시스템의 전원을 끈다.

□ **데몬 프로세스** : 독자적으로 돌아가는 프로그램

○ 독자형

- 시스템의 백그라운드에서 서비스별로 항상 동작
- 자주 호출되는 데몬이 아니라면 시스템의 자원을 낭비할 우려

○ 슈퍼데몬에 의한 동작 방식

- 평소에는 슈퍼 데몬만 동작하다가 서비스 요청이 오면 슈퍼 데몬이 해당 데몬을 동작 시킴
- 독자형보다는 서비스에 응답하는 데 시간이 약간 더 걸릴 수 있지만 자원을 효율적으로 사용한다는 장점

○ 슈퍼 데몬

- inetd : 유닉스의 슈퍼 데몬 // 리눅스의 인터넷 데몬
- xinetd : 페도라에서 보안 기능이 포함된 슈퍼 데몬

○ 리눅스의 주요 데몬

데몬	기능	데몬	기능
atd	특정 시간에 실행하도록 예약한 명령을 실행한다 (at 명령으로 예약)	smtpd	메일 전송 데몬
crond	주기적으로 실행하도록 예약한 명령을 실행한다	popd	기본 편지함 서비스를 제공
dhcpcd	동적으로 IP주소를 부여할 수 있도록 하는 서비스를 제공한다.	routed	자동 IP라우터 테이블 서비스를 제공
httpd	웹 서비스를 제공한다.	smd	삼바 서비스를 제공한다. (윈도우 시스템과 메시지 주고받는 역할)
lpd	프린트 서비스를 제공한다.	syslogd	로그 기록 서비스를 제공
nfs	네트워크 파일 시스템 서비스를 제공한다.	sshd	원격 보안 접속 서비스를 제공
named	DNS 서비스를 제공한다.	inetd	원격 접속 서비스를 제공
sendmail	이메일 서비스를 제공한다.	ftpd	파일 송수신 서비스를 제공
		ntpd	시간 동기화 서비스를 제공

□ 부트 로더 : 리눅스에서는 F10 (PC시작시 F9이나 Delete 누르는 것과 같음)

○ GRUB : 리눅스 에서의 부트 로더

(9장) 소프트웨어 관리하기

○ RPM : 레드햇에서 만든 패키지 관리 도구

- 바이너리 파일로 구성되어 있어 컴파일 필요 없다.
- 단점 : 패키지 의존성에 따라 관련 패키지가 먼저 설치되어 있어야 한다.

○ RPM 패키지 이름 구성 : audit-2.3.2-1.fc19.x86_64.rpm

* 패키지 버전 : 1.0(기능추가)0 / 1.00 이 되면 완성형 / 2.00 은 새로 만든 것

○ .deb : 데비안, 우분투 계열에서 사용하는 패키지

□ RPM 패키지 설치

○ rpm [옵션] : RPM 패키지를 관리

옵션	기능
-vv	매우 자세하게 디버깅 정보를 출력
--quiet	최대한 출력을 자제하고 오류 메시지만 출력
--help	도움말을 출력
--version	사용 중인 rpm의 버전을 출력

○ rpm -i [설치 옵션] 패키지명 : RPM 패키지를 설치

옵션	기능
-h	해시 기호(#)를 출력한다.
-v	설치 과정에 대한 메시지를 출력한다.
--replavfiles	이미 설치된 다른 패키지의 파일을 덮어쓰면서라도 패키지를 강제로 설치한다.
--replacepkgs	패키지가 이미 설치되어 있어도 다시 설치한다.
--test	설치하지는 않고 충돌 사항이 있는지만 점검하고 보고한다.

○ rpm [설치 옵션] 패키지명 : RPM 패키지를 업그레이드

옵션	기능
-h	해시 기호(#)를 출력한다.
-v	설치 과정에 대한 메시지를 출력한다.
-i	패키지를 설치한다.
-u	패키지를 업그레이드하여 설치한다.

○ rpm -e 패키지명 : RPM 패키지를 삭제

□ yum 패키지 설치

- yum [옵션] [명령] [패키지명] : 계— 기반의 패키지를 자동으로 설치한다.

옵션	기능
-h	도움말을 출력한다.
-y	설치 과정의 모든 질문에 yes로 대답한다.
-v	자세한 메시지를 출력한다.

명령	기능
install	패키지 설치
update	패키지 업데이트
check-update	패키지 확인
remove	패키지 삭제
list	패키지 목록 확인
info	패키지 정보 확인

□ 파일 아카이브와 압축 : 여러 파일이나 디렉토리를 묶어 마그네틱테이프와 같은 이동식 저장 장치에 보관하기 위해 사용하는 명령

- tar 기능[옵션] [아카이브 파일] 파일 이름 : 파일과 디렉토리를 묶어 하나의 아카이브 파일을 생성

기능	기능
c	새로운 tar 파일을 생성
t	tar 파일의 내용을 출력
x	tar 파일에서 원본 파일을 추출
r	새로운 파일을 추가
u	수정된 파일을 업데이트

옵션	기능
f	아카이브 파일이나 테이프 장치를 지정한다. 파일 이름을 “-”로 지정하면 tar 파일 대신 표준 입력에서 읽어들이다.
v	처리하고 있는 파일의 정보를 출력한다.
h	심벌릭 링크의 원본 파일을 포함한다.
p	파일 복구 시 원래의 접근 권한을 유지한다.
j	bzip2로 압축하거나 해제한다.
z	gzip로 압축하거나 해제한다.

- tvf : 아카이브 내용 확인
- xvf : 아카이브 풀기 // 아카이브를 풀어야 사용가능하다.
- uvf : 아카이브 업데이트

□ 소프트웨어 컴파일

- gcc : 리눅스에서 사용하는 C 컴파일러 패키지 이름
 - 설치 확인 : rpm -qa | grep gcc
- gcc 설치 : yum install gcc // yum을 사용하여 gcc를 인스톨
- 리눅스 C프로그램 사용방법
 - 소스파일 준비하기 : vi 이름.c
 - C프로그램 컴파일 : gcc 이름.c
 - C프로그램 실행하기 : ./a.out // 실행파일 a.out은 자동으로 설정됨 / 설정을 바꾸면 ./hello 등
 - * 실행파일명 변경 : -o 옵션 사용 // ex) gcc -o hello hello.c : 실행파일(hello) / 소스파일(hello.c)
 - // extern int two(); : extern은 외부의 파일(다른 파일의 메소드)
 - make 명령을 사용하면 c파일이 나온다. -> ./실행파일명 으로 실행
- makefile 작성하기 : 패키지 형태

```
vi makfile 명령실행
TARGET=one
OBJECTS=one.o two.o
${TARGET} : ${OBJECTS}
    gcc -o ${TARGET} ${OBJECTS}

one.o : one.c
    gcc -c one.c
two.o : two.c
    gcc -c two.c
:wq
```

(10장) 사용자 관리

- /etc/passwd 파일의 구조 => 로그인 ID : x : UID : GID : 설명 : 홈 디렉터리 : 로그인 셸
 - x : 초기 유닉스 시스템에서 사용자 암호를 저장하던 항목
 - UID : 사용자 ID번호로 시스템이 사용자를 구별하기 위해 사용하는 번호
 - GID : 그룹 ID

□ 사용자계정 관리 명령

- useradd [옵션] 로그인 ID : 사용자 계정 생성하기

옵션	기능
-m	home을 만들어줌
-u uid	UID를 지정한다
-o	UID의 중복을 허용
-g gid	기본 그룹의 GID를 지정
-G gid	2차 그룹의 GID를 지정
-d 디렉터리명	홈 디렉터리를 지정
-s 셸	기본 셸을 지정
-c 설명	사용자의 이름 등 부가적인 설명을 지정
-D	기본 설정 값을 설정하거나 출력
-e	EXPIRE 항목을 설정(YYYY -MM -DD)
-f	INACTIVE 항목을 설정
-k	계정 생성 시 복사할 초기 v kdlfdlsk 디렉터리를 설정해놓은 디렉터리를 지정

- userdel [옵션] 로그인 ID : 사용자 계정 삭제하기

옵션	기능
-r	홈 디렉터리를 삭제
-f	사용자가 로그인 중이어도 강제로 삭제

□ 그룹관리 명령

- groupadd [옵션] 그룹명 : 그룹 추가하기

옵션	기능
-g gid	그룹의 GID를 지정
-o	GID의 중복을 허용

- groupmod [옵션] 그룹명 : 그룹 수정하기

옵션	기능
-g gid	그룹의 GID를 수정
-o	GID의 중복을 허용
-n 그룹명	그룹명을 다른 이름으로 바꾼다.

- groupdel 그룹명 : 그룹 삭제하기

- gpasswd [옵션] 그룹명 : 그룹 삭제하기

옵션	기능
-a 사용자 계정	사용자 계정을 그룹에 추가
-d 사용자 계정	사용자 계정을 그룹에서 삭제
-r	그룹 암호를 삭제

- newgrp 그룹명 : 소속 그룹을 다른 그룹으로 바꾼다

□ 사용자 정보관리 명령

- who [옵션] : 현재 시스템을 사용하는 사용자의 정보를 출력

옵션	기능
-q	사용자의 이름만 출력
-H	출력 항목의 제목도 함께 출력
-b	마지막으로 재시작한 날짜와 시간을 출력
-m	현재 사용자 계정의 정보를 출력
-r	현재 런레벨을 출력

- w [사용자 이름] : 현재 시스템을 사용하는 사용자의 정보와 작업 정보를 출력
- last : 시스템에 로그인하고 로그아웃한 정보를 출력
- UID와 EUID 확인하기 : whoami, who am I, id
 - UID 출력 : who am I, who -m
 - EUID 출력 : whoami, id

- groups [계정명] : 현사용자 계정이 속한 그룹을 출력
- passwd [옵션] [사용자 계정] : 사용자 계정의 암호를 수정

옵션	기능
-l 사용자 계정	지정한 계정의 암호를 잠근다
-u 사용자 계정	암호 잠금을 해제
-d 사용자 계정	지정한 계정 암호를 삭제

- chown [옵션] 사용자 계정 파일명/디렉터리명 : 파일 디렉터리의 소유자와 소유 그룹을 변경

옵션	기능
-R	서브 디렉터리의 소유자와 소유 그룹도 변경한다.

- chgrp [옵션] 사용자 계정 파일명/디렉터리명 : 파일 디렉터리의 소유 그룹을 변경

옵션	기능
-R	서브 디렉터리의 소유자와 소유 그룹도 변경한다.

(11장) 네트워크 설정

□ 네트워크 기초

- TCP/IP 프로토콜 모델의 계층별 역할과 대표 프로토콜

계층	기능	프로토콜	전송단위
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

- TCP/IP 프로토콜 모델
 - 응용계층 - 전송계층 - 네트워크 계층 - 링크 계층 - 물리 계층

- * MAC(media access control) : 하드웨어를 위한 주소, 네트워크 인터페이스 카드(랜 카드)에 저장된 주소
- * IP(internet protocol) 주소 : 인터넷으로 연결된 네트워크에서 각 컴퓨터를 구분하기 위해 사용

- uname [옵션] : 시스템 정보를 출력

옵션	기능
-m	하드웨어 종류를 출력
-n	호스트 이름을 출력
-r	운영체제의 릴리즈 정보를 출력
-s	운영체제의 이름을 출력
-v	운영체제의 버전을 출력
-a	위의 모든 정보를 출력

□ 네트워크 설정

- ifconfig [인터페이스명] [옵션] [값] : 네트워크 인터페이스의 IP 주소를 설정

옵션	기능
-a	시스템의 전체 인터페이스에 대한 정보를 출력
up/down	인터페이스를 활성화, 비활성화한다.
netmask 주소	넷마스크 주소 설정
broadcast 주소	브로드캐스트 주소를 설정

- 네트워크 인터페이스 수동으로 설정
 - ifconfig 인터페이스명 IP주소 netmask 넷마스크 주소 broadcast 브로드캐스트 주소
 - ex) ifconfig eth0 192.168.0.14 netmask 255.255.255.0 broadcast 192.168.0.255

- route [명령] : 라우팅 테이블을 편집하고 출력한다.
 - add : 라우팅 경로나 기본 게이트웨이를 추가한다.
 - del : 라우팅 경로나 기본 게이트웨이를 삭제한다.

- nslookup [도메인명] : DNS 서버와 대화식으로 질의하고 응답을 받는다

□ 네트워크 상태 확인

- ping [옵션] 목적지주소 : 네트워크 장비에 신호(ECHO_REQUEST)를 보낸다.
- traceroute 목적지주소 : 목적지까지 패킷이 거치는 경로를 출력
- netstat [옵션] : 네트워크의 상태 정보를 출력
- arp [IP주소] : ARP 캐시 정보를 관리
- tcpdump [옵션] : 네트워크상의 트래픽을 덤프

(12장) 원격접속과 FTP

□ 원격접속

- 텔넷 : 텔넷 클라이언트와 리눅스 사이에 데이터를 원격으로 주고받을 수 있는 서버
- SSH : 텔넷 서버에서 텔넷 클라이언트와 리눅스 사이에 주고받은 데이터의 암호화 기능이 추가된 기능
- DNC 서버 : 원격접속 시 화면 나오는 것
- VNC 서버 : 원격에서 그래픽 환경으로 접속할 수 있는 서버
- FTP(file transfer protocol) 서버 : 파일 넘길 수 있는 서버

옵션	기능
cd 원격 디렉터리	원격 호스트의 디렉터리를 이동
lcd 지역 디렉터리	지역 호스트의 디렉터리를 이동
pwd	원격 호스트의 디렉터리를 출력
!pwd	지역 호스트의 디렉터리를 출력
ls 또는 dir	원격 호스트의 파일 목록을 출력. dir 명령은 상세한 파일 정보를 출력
!ls	지역 호스트의 파일 목록을 출력
mkdir 원격 디렉터리	원격 호스트에 디렉터리를 생성
rmdir 원격 디렉터리	원격 호스트에 디렉터리를 삭제
get 원격 파일명 [지역 파일명]	원격 파일 하나를 지역 호스트로 가져온다. 지역 파일명을 지정하면 지정한 파일명으로 저장하고, 지정하지 않으면 원격 파일명과 동일한 파일명으로 지정한다.
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.
grompt	mget이나 mput 명령 사용 시 파일 전송 여부를 물어볼 것인지를 결정
hash	파일 전송되는 동안 #를 출력하여 진행 상황을 알려준다.
bye	ftp를 종료한다.
open	ftp로 접속할 호스트를 입력하도록 한다.
user	사용자명을 다시 입력할 수 있도록 한다.
? 또는 help [명령]	명령에 대한 도움말을 출력