

파일 관리

Hadoop

Byeongjoon Noh

powernoh@sch.ac.kr



Contents

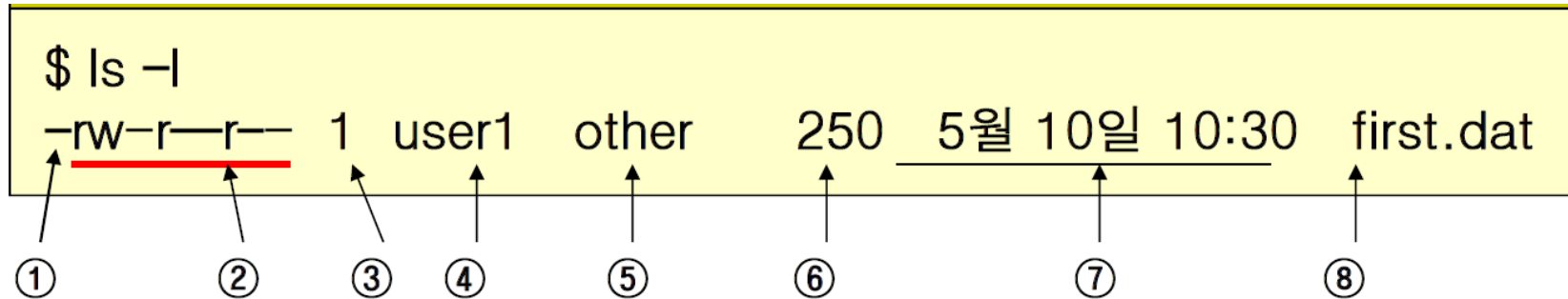
1. 파일 사용 권한 관리
2. 파일 읽기와 쓰기
3. 링크
4. 파일과 디렉토리 검색
5. 파일 백업과 압축

1. 파일 사용 권한 관리

파일의 속성

- ls -l 명령어로 파일과 디렉토리의 속성을 알 수 있음

```
$ ls -l
-rw-r--r-- 1 user1 other 250 5월 10일 10:30 first.dat
```



번호	값	의 미
①	-	파일 종류 (- : 일반파일, d: 디렉토리)
②	rw-r--r--	파일을 읽고,쓰고,실행할 수 있는 권한 표시
③	1	물리적 연결 개수
④	user1	파일 소유자의 사용자명
⑤	other	파일 소유자의 그룹명
⑥	250	파일 크기 (바이트 단위)
⑦	5월 10일 10:30	파일이 마지막으로 변경된 시간
⑧	first.dat	파일명

파일의 속성

- Linux 파일의 종류

문자	파일 유형
-	일반 (정규) 파일
d	디렉토리 파일
b	블럭 단위로 읽고 쓰는 블럭 장치 특수 파일
c	문자 단위로 읽고 쓰는 문자 장치 특수 파일
l	기호적 링크
p	파이프
s	소켓

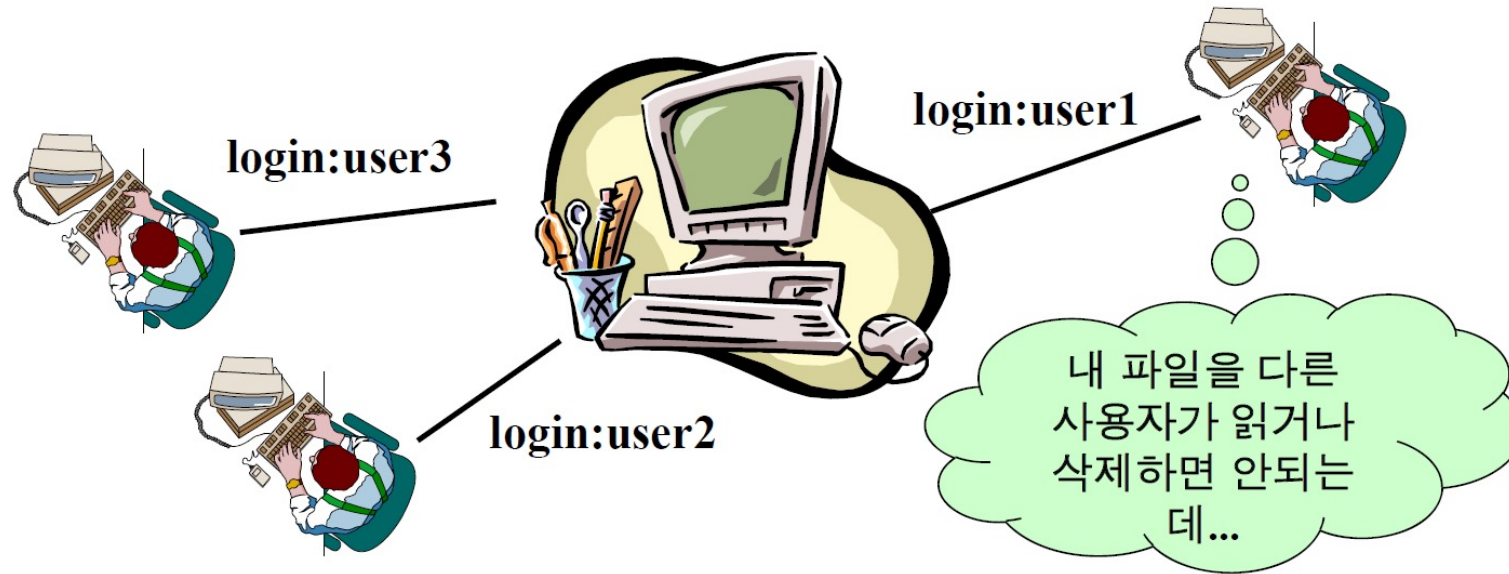
파일의 종류

- `file [파일명]`
 - 지정한 파일의 종류를 출력

- 실습
 - 홈 디렉토리에서 파일속성 확인
 - 루트 디렉토리에서 파일속성 확인
 - 특정 문자열을 포함하는 파일속성 확인 (* 활용)
 - `$ ls -l /dev/sd*`

파일의 사용 권한

- Unix 시스템에서 사용자가 자신의 파일 및 디렉토리를 다른 사용자로부터 보호하기 위해 접근(access)할 수 있는 권한을 변경하도록 함

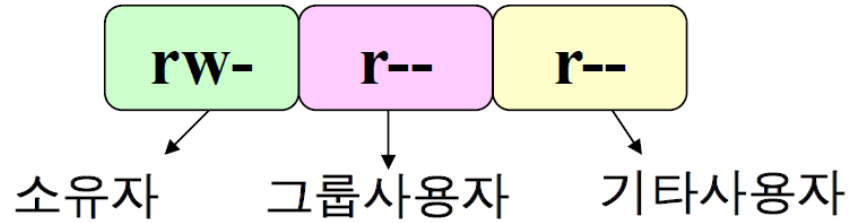


파일의 사용 권한

- 사용자 구분: 소유자, 그룹, 기타 사용자
 - Unix는 사용 권한 부여를 위해 사용자를 세 카테고리로 구분함
- 사용 권한의 종류
 - 사용 권한은 파일 유형 (일반 file or 디렉토리)에 따라 조금씩 다르게 해석

모드	일반 파일	디렉토리 파일
읽기 (r)	파일 내용을 읽을 수 있다	디렉토리가 포함하는 파일 목록을 읽을 수 있다
쓰기 (w)	파일을 수정/삭제 시킬 수 있다	디렉토리내에 파일을 생성,삭제할 수 있다
실행 (x)	파일을 실행 시킬 수 있다	cd 명령을 이용하여 디렉토리로 이동할 수 있다

파일의 사용 권한



- 문자의 의미

- r: 읽기 허가, w: 쓰기 허가, x: 실행 허가, -: 불허

- 다양한 사용 권한 조합

사용 권한	의미
<code>rxrx-rx-rx</code>	소유자는 읽기/쓰기/실행 권한을 모두 가지고 그룹과 기타사용자는 읽기와 실행권만 가짐
<code>r-rx-rx-rx</code>	소유자, 그룹, 기타사용자 모두 읽기와 실행권만 가짐
<code>rw-----</code>	소유자만 읽기/쓰기 권한을 갖고 그룹과 기타사용자는 아무 권한도 없음
<code>rw-rw-rw-</code>	소유자와 그룹, 기타사용자 모두 읽기와 쓰기 권한을 가지고 있음
<code>rxrxrx-rwx</code>	소유자, 그룹, 기타사용자 모두 읽기/쓰기/실행 권한을 가지고 있음
<code>rx-----</code>	소유자만 읽기/쓰기/실행권한을 가지고 있고 그룹과 기타사용자는 아무 권한도 없음

파일의 사용 권한

- 예시

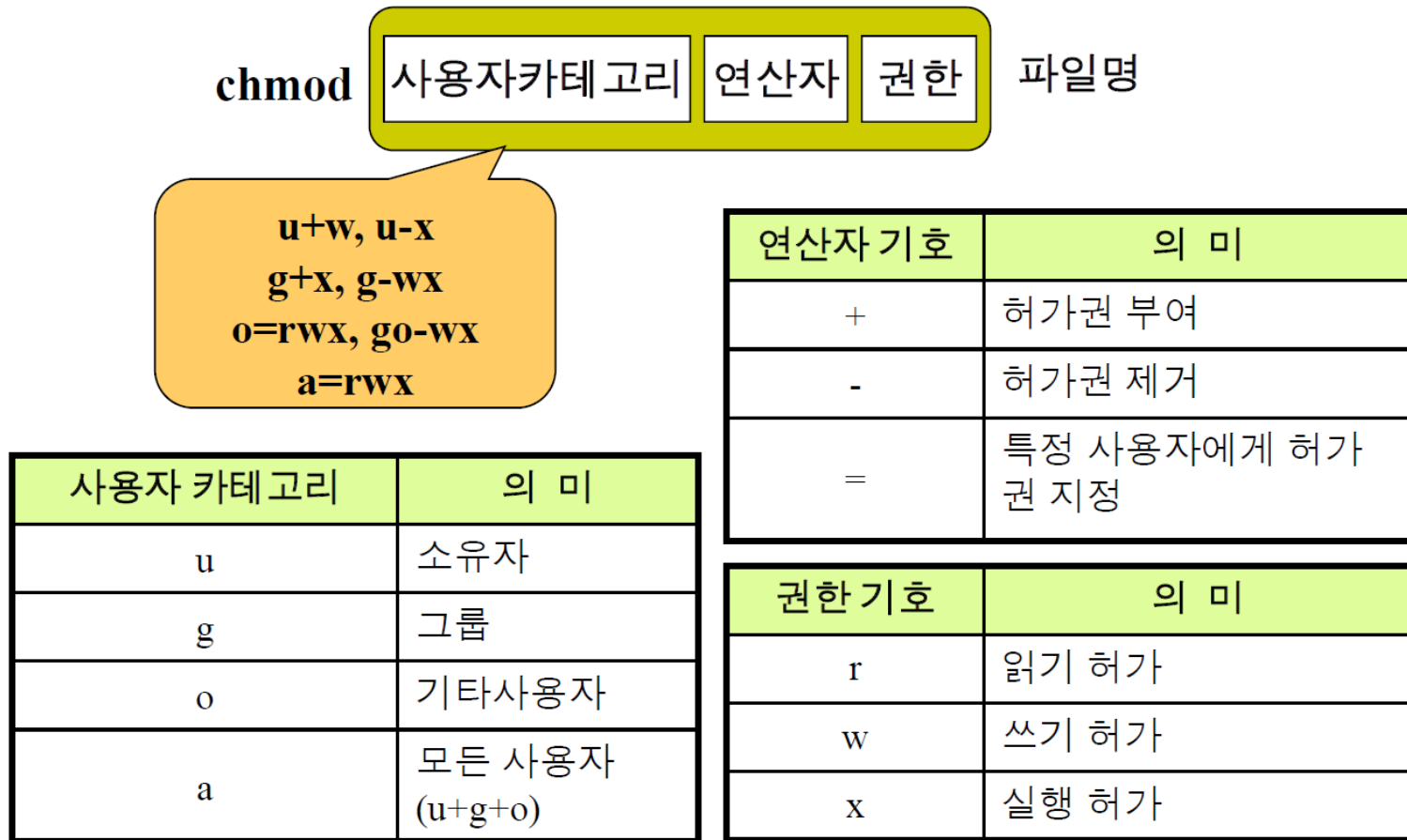
```
bj@bj:~$ ls -al
total 591120
drwxr-xr-x 19 bj  bj  4096 3월 1 15:59 .
drwxr-xr-x  3 root root 4096 3월 1 13:35 ..
-rw----- 1 bj  bj  1165 3월 1 15:04 .bash_history
-rw-r--r-- 1 bj  bj   220 3월 1 13:35 .bash_logout
-rwxrwxrwx 1 bj  bj  4108 3월 1 15:03 .bashrc
-rwxrwxrwx 1 bj  bj  3771 3월 1 13:45 .bashrc.save
drwxr-xr-x 13 bj  bj  4096 3월 1 13:46 .cache
drwx----- 10 bj  bj  4096 3월 1 13:39 .config
drwxr-xr-x  2 bj  bj  4096 3월 1 13:38 Desktop
drwxr-xr-x  2 bj  bj  4096 3월 1 13:38 Documents
drwxr-xr-x  2 bj  bj  4096 3월 1 13:53 Downloads
drwx-----  3 bj  bj  4096 3월 1 13:39 .gnupg
drwxr-xr-x 12 bj  bj  4096 3월 1 14:41 hadoop
-rw-rw-r-- 1 bj  bj 605187279 6월 15 2021 hadoop-3.3.1.tar.gz
drwxrwxr-x  4 bj  bj  4096 3월 1 14:50 hdata
drwxr-xr-x  3 bj  bj  4096 3월 1 13:38 .local
drwx-----  4 bj  bj  4096 3월 1 13:46 .mozilla
drwxr-xr-x  2 bj  bj  4096 3월 1 13:38 Music
drwxr-xr-x  2 bj  bj  4096 3월 1 13:38 Pictures
-rw-r--r-- 1 bj  bj   807 3월 1 13:35 .profile
drwxr-xr-x  2 bj  bj  4096 3월 1 13:38 Public
drwx-----  2 bj  bj  4096 3월 1 13:45 .ssh
-rw-r--r-- 1 bj  bj    0 3월 1 13:39 .sudo_as_admin_successful
drwxr-xr-x  2 bj  bj  4096 3월 1 13:38 Templates
-rw-rw-r-- 1 bj  bj    0 3월 1 15:59 test
-rw-rw-r-- 1 bj  bj    0 3월 1 15:59 test2
drwxr-xr-x  2 bj  bj  4096 3월 1 13:38 Videos
drwxr-xr-x  2 bj  bj  4096 3월 1 15:00 .vim
-rw-----  1 bj  bj  11144 3월 1 15:03 .viminfo
bj@bj:~$
```

파일의 사용 권한 변경

- `chmod [option] mode filename`
 - 자신이 소유한 파일의 사용 권한을 변경
 - option
 - -R: 하위 디렉토리 포함
 - mode
 - 변경할 사용권한 표시
 - 기호 모드, 8진수 모드

파일의 사용 권한 변경

- 기호모드
 - 기호를 이용하여 허가권 변경



파일의 사용 권한 변경

- 기호모드 사용법

- `$ chmod u-w first.dat`

- first.dat 파일의 소유자 쓰기 권한 제거

- `$ chmod g+wx first.dat`

- first.dat 파일의 그룹에 쓰기와 실행권한 부여

- `$ chmod go=rw first.dat`

- first.dat 파일의 그룹과 기타사용자에게 읽기와 쓰기 권한 부여

- 만약 first.dat 파일의 그룹과 기타사용자가 실행권한이 있었다면 제거됨

- `$ chmod u=rws first.dat`

- first.dat 파일의 소유자에게 rwx권한 부여

파일의 사용 권한 변경

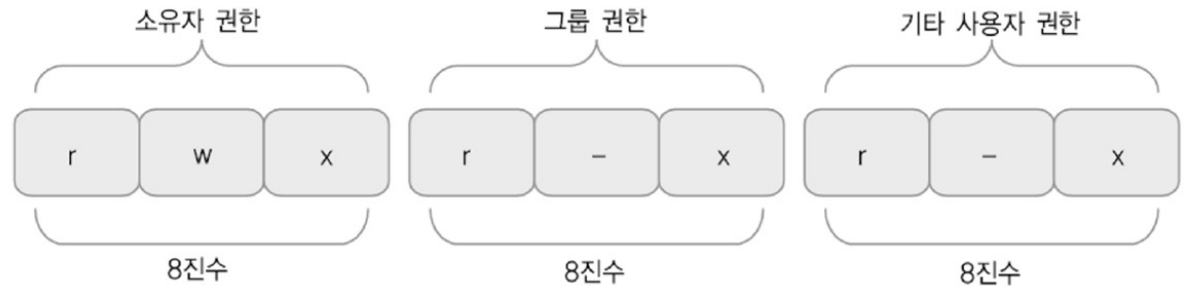
- 기호모드로 파일의 사용 권한 변경 실습 → **hosts** 파일의 최종 권한은?

```
$ cd /ubuntu/Desktop
$ mkdir Practice
$ cd Practice
$ cp /etc/hosts . (현재 hosts 권한: -rw-r--r--)
$ ls -l
$ chmod u+x hosts
$ chmod go+w hosts
$ chmod go-rw hosts
$ ls -l
```

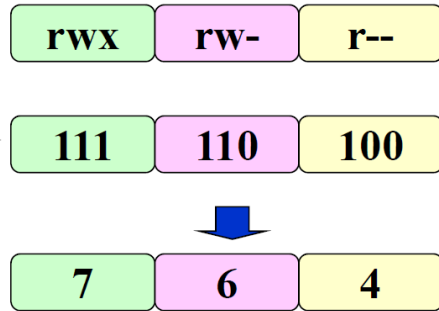
파일의 사용 권한 변경

- 숫자모드

- 숫자를 이용하여 허가권 변경
- 3자리 8진수를 이용하여 권한 표시



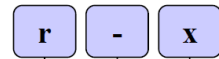
권한이 있으면 1, 없으면 0으로 표시



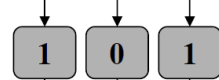
2진수

8진수

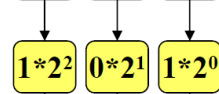
1) 사용 권한



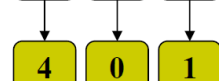
2) 2진수로 대체



3) 2진수 계산



4) 계산 결과 합산



5) 8진수 권한 값



8진수	2진수	권한	의미
0	000	--	아무 권한 없음
1	001	-x	실행 권한만 있음
2	010	-w-	쓰기 권한만 있음
3	011	-wx	쓰기, 실행 권한 있음
4	100	r-	읽기 권한만 있음
5	101	rx	쓰기, 실행 권한 있음
6	110	rw-	읽기, 쓰기 권한 있음
7	111	rwx	모든 권한 있음

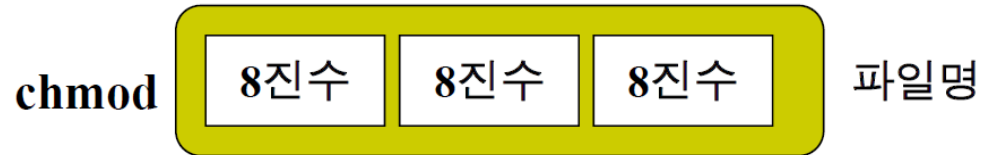
파일의 사용 권한 변경

- 숫자모드
 - 8진수로 표현한 사용 권한

사용 권한	8진수 모드값
rwxrwxrwx	777
rwxr-xr-x	755
rw-rw-rw-	666
r-xr-xr-x	555
rw-r--r--	644
rwx-----	700
rw-r-----	740
r-----	400
-----	000

파일의 사용 권한 변경

- 숫자모드 사용 예



(1) chmod 444 first.dat	(1) 444 = r--r--r--
(2) chmod 474 first.dat	(2) 474 = r—rwxr--
(3) chmod 475 first.dat	(3) 475 = r--rwxr-x
(4) chmod 464 first.dat	(4) 464 = r--rw-r--
(5) chmod 575 first.dat	(5) 575 = r-xrwxr-x
(6) chmod 755 first.dat	(6) 755 = rwxr-xr-x
(7) chmod 700 first.dat	(7) 700 = rwx-----

파일의 사용 권한 변경

- 숫자모드 실습 (숫자모드 활용하여 여러가지 권한 변경)

```
$ cd ~  
$ ls -l  
$ chmod 644 hosts  
$ ls -l  
$ chmod 666 hosts  
$ ls -l  
$ chmod 400 hosts
```

파일의 사용 권한 변경

- **umask [mask]**
 - 기본사용 권한을 변경하거나 출력
 - mask
 - mask 값을 지정하면 마스크를 이용하여 사용 권한 지정
 - mask 값을 지정하지 않으면 현재의 마스크 값을 출력
 - Linux 시스템에서는 기본 마스크값이 002로 설정되어 있음 (시스템 마다 조금씩 다름)

```
$ umask
22
$ umask 077
$ umask
77
```

022를 의미
077을 의미

파일의 사용 권한 변경

- mask를 이용한 권한 생성
 - (기본사용권한) XOR (mask)
 - 기본사용권한: 일반 파일 rw-rw-rw- (666), 디렉토리 rwxrwxrwx (777) ← 시스템에서 지정되어 있음
 - default mask 002 적용

일반 파일

(기본사용권한) rw-rw-rw- (666)
(2진수 표현) 110 110 110
(마스크값 002) 000 000 010
(XOR) 110 110 100 (rw-rw-r--)

디렉토리 파일

(기본사용권한) rwxrwxrwx (777)
(2진수 표현) 111 111 111
(마스크값 002) 000 000 010
(XOR) 111 111 101 (rwxrwxr-x)

- → 아무 처리 하지 않은 상태에서 일반 파일이나 디렉토리 파일을 생성하면 부여되는 권한

파일의 사용 권한 변경

- mask를 이용한 권한 생성

- (기본사용권한) XOR (mask)

- 기본사용권한: 일반 파일 rw-rw-rw- (666), 디렉토리 rwxrwxrwx (777) ← 시스템에서 지정되어 있음
- mask를 022로 지정한 경우 (`$ umask 022`)

46. 다음 명령을 실행했을 경우에 'a.txt' 파일의 허가권 값으로 알맞은 것은?

```
$ umask 022
$ touch a.txt
```

- ① ----r--r--
- ② -rwxr-xr-x
- ③ -rw-r--r--
- ④ -rw-rw-r--

일반 파일

(기본사용권한) rw-rw-rw- (666)
 (2진수 표현) 110 110 110
 (마스크값) 000 010 010
 (XOR) 110 101 101 (rw-r--r--)

디렉토리 파일

(기본사용권한) rwxrwxrwx (777)
 (2진수 표현) 111 111 111
 (마스크값) 000 010 010
 (XOR) 111 101 101 (rwxr-xr-x)

파일의 사용 권한 변경

- mask를 이용한 권한 생성
 - 대표적인 mask 값
 - 002 (default)
 - 일반파일 664, 디렉토리 775로 지정됨
 - → 의미: 소유자와 그룹은 모두 할 수 있고, 기타 사용자는 쓰기 금지
 - 022
 - 일반파일 644, 디렉토리 755로 지정됨
 - → 의미: 소유자는 모두 할 수 있고, 그 이외의 사용자는 쓰기 금지
 - 077
 - 일반파일 600, 디렉토리 700으로 지정됨
 - → 소유자 이외에는 접근 금지

파일의 사용 권한 변경

- mask를 이용한 권한 변경 실습 → utmp2와 utest2의 권한은?

```
$ cd ~  
$ umask  
$ mkdir utmp  
$ touch utest  
$ ls -l  
$ umask 027  
$ umask  
$ mkdir utmp2  
$ touch utest2  
$ ls -l
```

2. 파일 읽기와 쓰기

Visual Interface

- vi 편집기는 초기 Unix 시스템에서 사용자들이 텍스트 파일을 편집하기 위해서 개발됨
 - 그 때 당시에는 GUI (Graphic User Interface)가 없었음, Only TUI (Text UI)
 - 모든 Unix 및 Unix 계열 시스템에서 기본적으로 제공

- vi의 주요 목적은 사용자가 텍스트 파일을 쉽고 빠르게 편집할 수 있도록 하는 것

vi 편집기 사용

- 행 단위 편집기 (라인 편집기)

- ed
- ex
- sed

- 화면 편집기

- vi
- vim
- nano
- emacs
- ...

16. 다음 중 emacs 편집기를 개발한 인물로 알맞은 것은?

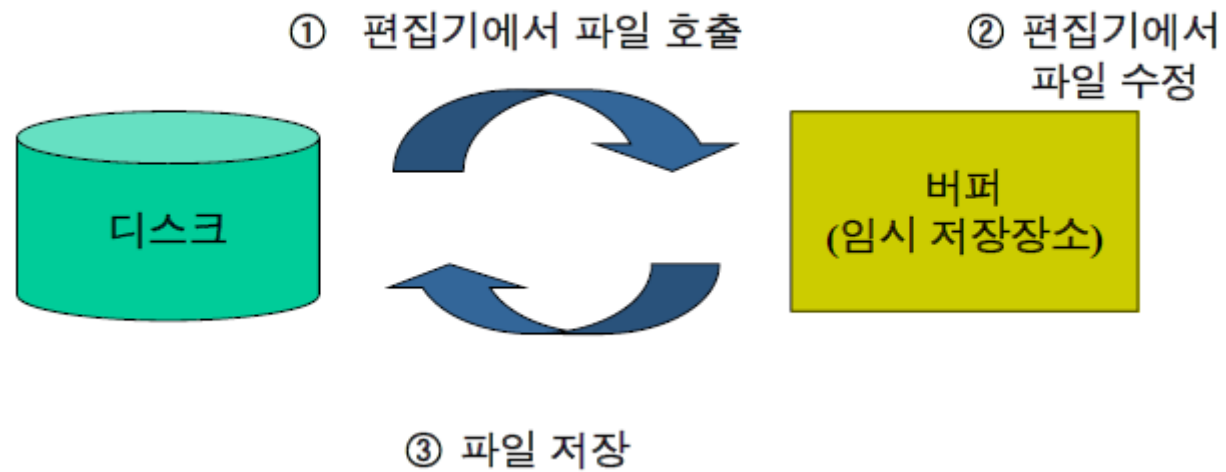
- ① 빌 조이 ② 리처드 스톨먼
- ③ 리누스 토발즈 ④ 브람 무레나르

워싱턴 대학에서 만든 유닉스용으로 만든 (㉠) 편집기는 리눅스 초기 배포판에 포함되었으나 최근에는 이 복제판인 (㉡) 편집기가 사용되고 있다.

- ① ㉠ vi, ㉡ vim ② ㉠ vi, ㉡ pico
- ③ ㉠ pico, ㉡ nano ④ ㉠ nano, ㉡ pico

vi 편집기 기본 원리

- 1) 파일을 연다 → 2) 수정한다 → 3) 닫는다



vi 편집기 열기

- **vi [파일이름]**
 - `$ vi`
 - 지정한 이름이 없으면 새로운 파일 생성
 - 파일 저장 시 이름을 지정
 - 지정한 이름이 있으면 기존 파일 열기
- vi 초기화면
 - 화면 크기에 따라 행/열의 수는 달라짐

```
ubuntu@server: ~/Desktop

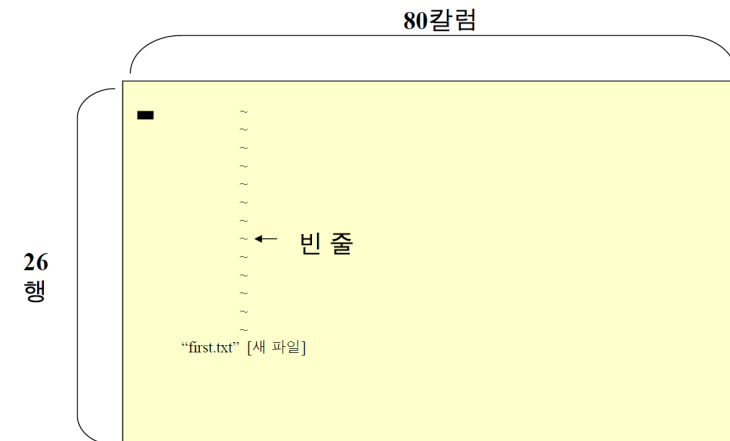
VIM - Vi IMproved

version 8.2.4919
by Bram Moolenaar et al.
Modified by team+vim@tracker.debian.org
Vim is open source and freely distributable

Help poor children in Uganda!
type :help iccf<Enter> for information

type :q<Enter> to exit
type :help<Enter> or <F1> for on-line help
type :help version8<Enter> for version info

Running in Vi compatible mode
type :set nocp<Enter> for Vim defaults
type :help cp-default<Enter> for info on this
```



vi 편집기 기본 원리

- 세 가지 모드

- 명령모드

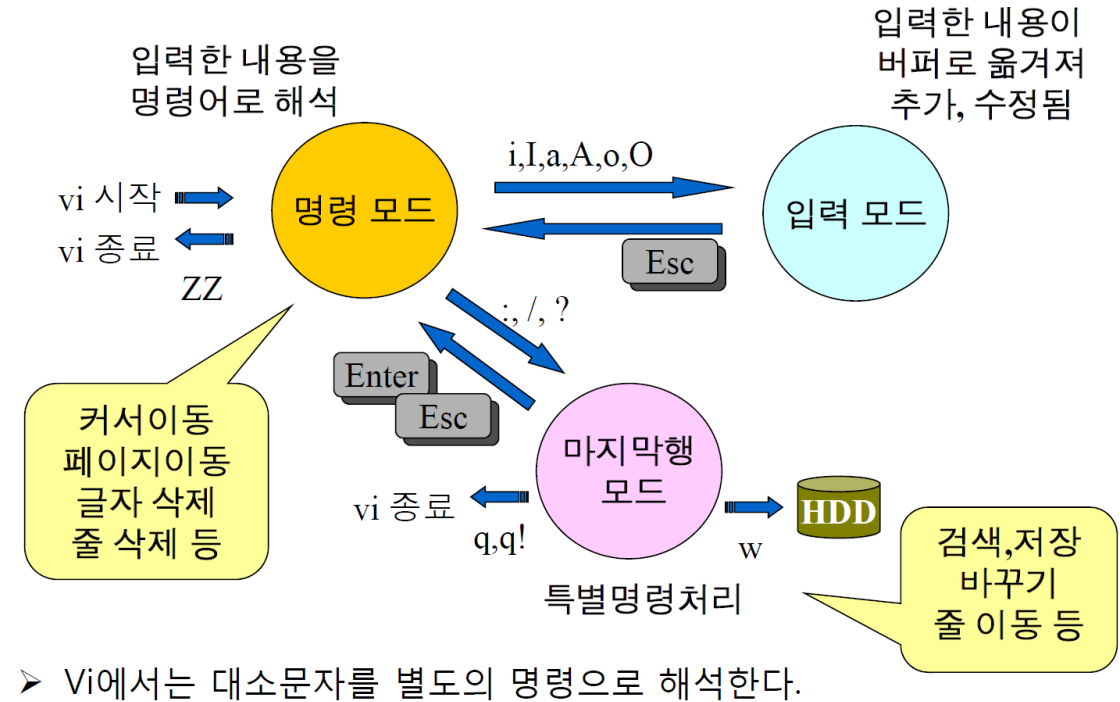
- 입력한 내용을 명령어로 해석
- 커서이동, 페이지이동, 글자 삭제, 줄 삭제 등

- 입력모드

- 입력한 내용이 버퍼로 옮겨져 파일의 내용이 수정됨

- 마지막행 모드

- 특별명령을 처리함
- 검색, 저장, 줄 바꾸기, 줄 이동 등



입력 명령

- 입력 명령
 - 명령모드 → 입력모드

명령키	수행 작업
i	커서 앞에 삽입
a	커서 뒤에 삽입
o	현재 줄 다음에 삽입
I	현재 줄 첫 칸 앞에 텍스트 입력
A	현재 줄 끝에 텍스트 입력
O	현재 줄 앞에 삽입

- 주의! 입력이 끝나면 반드시 “입력모드”에서 “명령모드”로 돌아와야 함 (ESC)

vi 편집기 실습

- 입력 및 저장 실습

- 1) 실습 디렉토리 구성

```
$ cd  
$ mkdir linux_practice  
$ cd linux_practice  
$ vi test.txt
```

- 2) text.txt 파일 생성

- 3) vi 편집기로 text.txt 오픈 후 아래 내용 입력 ('i'로 입력모드로 모드변경)

```
mane is Gil-dong Hong.  
I wake up in the morning.  
This is a living loom.  
A that's a bedroom.
```

- 4) ESC키 → :w 입력으로 저장

vi 편집기 실습

- 입력 및 저장 실습
 - 다시 vi 편집기로 text.txt 파일 열어서 아래 내용으로 수정

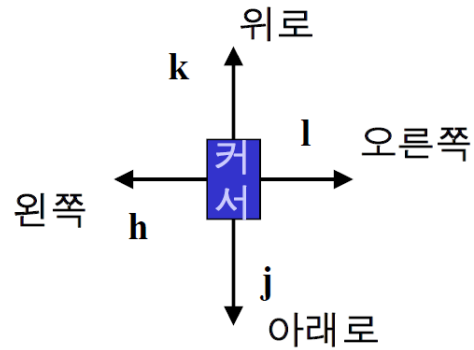
```
My name is Gil-dong Hong.  
I wake up in the morning.  
This is a living room.  
And that's a bedroom.
```

- 수정 완료 후 저장

- 참고
 - :set nu ← 줄 번호 출력
 - :set nonu ← 줄 번호 해제

vi 편집기 커서이동명령

- 화살표 키 이용
- h, j, k, l 키 이용



이동	명령어
한 행 위	k
한 행 아래	j
한 문자 오른쪽	l
한 문자 왼쪽	h
행의 시작	^ 또는 0
행의 마지막	\$
이전 행의 처음	-
다음 행의 처음	+ 또는 ↵

vi 편집기 커서이동명령

- 현재 화면에서 커서 이동

이동	명령키
키 화면 맨 위로	H
키 화면 중간으로	M
키 화면 맨 아래로	L
다음 단어의 첫문자로	w
이전 단어의 첫문자로	b
다음 단어의 끝 글자로	e

- 지정한 곳으로 이동

이동	명령키
줄 번호 n 위치로	:n 또는 nG
파일의 끝 줄로 이동	:\$ 또는 G
n줄 만큼 앞으로 이동	n+
n줄 만큼 뒤로	n-
현재 문장의 처음으로	(
다음 문장의 처음으로)
현재 문단의 처음으로	{
다음 문단의 처음으로	}

vi 편집기 커서이동명령

- 커서이동 예제

- 현재 커서는 8번째 줄 'u'에 있음

The image shows a screenshot of a vi editor window with a yellow background. The code is as follows:

```
1 #include <stdio.h>
2
3 main() {
4     char c;
5
6     printf("Hello, C World\n");
7     printf("=====\n");
8     printf("select menu item\n");
9     printf("1. unix\n");
10    printf("2. linux\n");
11    printf("=====\n");
12 }
```

Annotations and cursor positions:

- H**: Home key, points to the start of line 1.
- M**: Move to start of line, points to the start of line 6.
- ^**: Move to start of line, points to the start of line 8.
- L**: Move to end of line, points to the end of line 12.
- k**: Move up one line, points to line 5.
- w**: Move forward to next word, points to the end of "World" on line 6.
- \$**: Move to end of line, points to the end of line 8.
- e**: Move forward to end of word, points to the end of "item" on line 8.
- j**: Move down one line, points to line 9.
- b**: Move backward to start of word, points to the start of "select" on line 8.

Red circles highlight the characters being moved to: the '#' on line 1, the 'p' on line 6, the 'u' on line 8, the 'i' on line 8, the 'x' on line 9, and the '}' on line 12.

vi 편집기 화면이동

- 화면에 나타나지 않은 부분으로 화면 이동
 - ^ = Ctrl 키

이동	명령키
반 화면 위로	^u
반 화면 아래로	^d
한 화면 위로	^b
한 화면 아래로	^f
한 줄만 위로	^y
한 줄만 아래로	^e

vi 편집기 커서이동/화면이동 실습

- /etc/profile 파일을 복사하여 text2.txt로 이름 변경 후 vi 편집기 내에서 조작

내용 삭제 및 취소

- 명령 모드에서 동작

명령어	삭제 대상	수행 작업
x, #x	문자	커서 위치의 문자 삭제(예:3x)
dw, #dw	단어	커서 위치의 단어 삭제
dd, #dd	줄	커서 위치의 줄 삭제
D(shift-d)	줄의 일부	커서 위치부터 줄 끝까지 삭제
u		방금 수행한 명령 취소
U		해당 줄의 모든 편집 취소

내용 삭제 및 취소 실습

- 아까 만든 text.txt 파일에서 다음 명령어들만 입력하였을 때 결과는?

My name is Gil-dong Hong.
I wake up in the morning.
This is a living room.
And that's a bedroom.

- 1) 1G
- 2) 11x
- 3) jj
- 4) IIII
- 5) D
- 6) j
- 7) dd
- 8) u
- 9) dd
- 10) :w

내용 삭제 및 취소

- (참고) 명령 모드에서 동작

키	수정 대상	수행 작업
r	문자	현재 커서위치의 한 문자 변경
R	문자열	현재 커서부터 ESC 입력까지 변경
cw	단어	커서 위치부터 현재 단어의 끝까지 내용 변경
cc	줄	커서가 위치한 줄의 내용 변경
s, ns	문자열	현재 커서부터 내용 변경(예:5s)
C	줄 일부	커서 위치에서 줄 끝까지 내용 변경

편집기능 – 복사, 잘라내기, 붙이기

- 명령 모드에서 동작

명령어	수행 작업
yy, #yy	현재 행을 버퍼로 복사 (예:4yy)
p	현재 행 다음에 버퍼 내용 삽입
P(shift+ p)	현재 행 위쪽에 버퍼 내용을 삽입
dd, #dd	현재 행을 잘라내기

- 행 삭제와 잘라내기는 동일한 동작임

버퍼의 사용

- 버퍼
 - vi는 작업 내용을 버퍼에 저장 → 실행 취소 가능
 - 복사하기, 잘라내기에 활용
- 버퍼 종류
 - Unnamed buffer (이름 없는 버퍼)
 - Named buffers (이름이 있는 버퍼): “a, “b, ... “z
 - Numbered buffers (번호가 있는 버퍼): “1, “2, ...”9
- 예시
 - “a3yy → 현재 행부터 아래로 3줄을 a 버퍼에 저장
 - “ap → a 버퍼의 내용을 붙이기

범위지정 방법

- 편집하는 범위를 지정하는 방법
- 마지막행 모드(:) 에서 사용
- “: 범위 편집 명령” 형태로 사용

범위	의 미
1,\$	첫 줄에서 마지막 줄까지(파일내의 모든 줄)
%	첫 줄에서 마지막 줄까지(파일내의 모든 줄)
1,.	첫 줄에서 현재 줄까지
.,\$	현재 줄에서 마지막 줄까지
.-2	현재 줄에서 앞쪽으로 2번째 줄
10,20	10번째 줄에서 20번째 줄까지

마지막행 모드에서 복사와 잘라내기

- 마지막행 모드(:) 에서 사용

명령어	수행 작업
:#y	#으로 지정한 행을 복사(:10y -> 10행을 복사)
:<범위>y	범위로 지정한 행을 복사(예, :10,20y -> 10행~20행까지 복사)
:#d	#으로 지정한 행을 삭제(:10d -> 10행을 삭제)
:<범위>d	범위로 지정한 행을 삭제(예, :10,20d -> 10행~20행을 삭제)
:pu	현재 행 다음에 버퍼내용 붙이기
:#pu	#으로 지정한 행 다음에 버퍼내용 붙이기(예, :5pu)

검색 기능

- 마지막행 모드(:) 에서 사용

명령어	수행 작업
/문자열	현재 위치부터 파일 앞쪽으로 문자열 탐색
?문자열	현재 위치부터 파일 뒤쪽으로 문자열 탐색
n	다음 문자열 탐색
N	역방향으로 문자열 탐색

검색 기능 실습

- text.txt 파일을 계속 활용하여 마지막줄 모드에서 다음 명령어 입력

1) `:/room`

2) `n`

3) `:?name`

4) `n`

5) `N`

바꾸기 기능

- 마지막행 모드(:) 에서 사용

명령어	수행 작업
:s/문자열1/문자열2/	커서가 위치한 줄에서만 문자열1을 문자열2로 바꿈
:<범위>s/문자열1/문자열2/	<범위>안의 모든 줄에 대해서 각 줄의 첫번째 문자열1을 찾아 문자열2로 바꿈
:<범위>s/문자열1/문자열2/g	<범위>안의 모든 줄에 대해서 모든 문자열1을 문자열2로 바꿈
:<범위>s/문자열1/문자열2/gc	<범위>안의 모든 줄에 대해서 각 문자열1을 문자열2로 치환할 때 수정할지 안 할지를 묻는다

바꾸기 기능 실습

- text.txt 파일을 계속 활용하여 마지막줄 모드에서 다음 명령어 입력

1) :s/dong/sun/

2) :%s/is/was/g

3) :2,5 s/And/But/g

4) :w

- 결과

```
My name was Gil-sun Hong.  
But that's a bedroom.  
I wake up in the morning.  
This was a living room.  
But that's a bedroom.
```

기타 기능

- 파일 읽어오기 / 여러 파일 편집

명령어	수행
:r 파일명	지정한 파일을 현재 커서 위치에 삽입
:e 파일명	현재 파일 대신 지정한 파일을 읽음
:n	vi시작시 여러 파일을 지정하였을 경우 다음 파일로 이동

- vi에서 셸 명령 실행

명령어	수행 작업
:!명령	vi를 중단하고 지정한 명령 수행 (vi로 돌아올 때 ↵)
:sh	vi를 잠시 빠져나가서 셸을 수행 (vi로 돌아올때 exit)

셸 명령 실행 실습

- text.txt 파일로 계속 실습 진행

1) `:!ls -l`

2) Esc 키

3) `:sh`

4) `ls -l`

5) `exit`

17. 다음 중 vi 편집기에서 줄의 시작이 linux 일 때 Linux로 치환하는 명령으로 알맞은 것은?

① `:% s/^linux/Linux/`

② `:% s/₩<linux/Linux/`

③ `:% s/₩<linux₩>/Linux/`

④ `:% s/$linux/Linux/`

18. 다음 중 vi 편집기에서 현재 커서가 위치한 줄부터 아래 방향으로 3줄 복사하는 명령으로 알맞은 것은?

① `3j`

② `3p`

③ `3dd`

④ `3yy`

vi 환경 설정

- vi 환경을 설정하는 특수명령과 변수들

명령어	수행 작업
<code>:set nu</code>	파일 내용의 각 줄에 줄 번호 표시 (보이기만 할 뿐 저장은 되지 않는다.)
<code>:set nonu</code>	줄 번호 취소
<code>:set list</code>	눈에 보이지 않는 특수문자표시(<code>tab:^I, eol:\$</code> 등)
<code>:set nolist</code>	특수문자보기 기능 취소
<code>:set showmode</code>	현재 모드 표시
<code>:set noshowmode</code>	현재 모드 표시기능 취소
<code>:set</code>	set으로 설정한 모든 vi변수 출력
<code>:set all</code>	모든 vi 변수와 현재 값 출력

20. 다음은 vi 편집기 실행 시에 자동으로 행 번호가 나타나도록 설정하는 과정이다. (괄호) 안에 들어갈 파일명과 설정 내용의 조합으로 알맞은 것은?

```
[ihduser@kait ~]$ cat > ( ㉠ )  
( ㉡ )
```

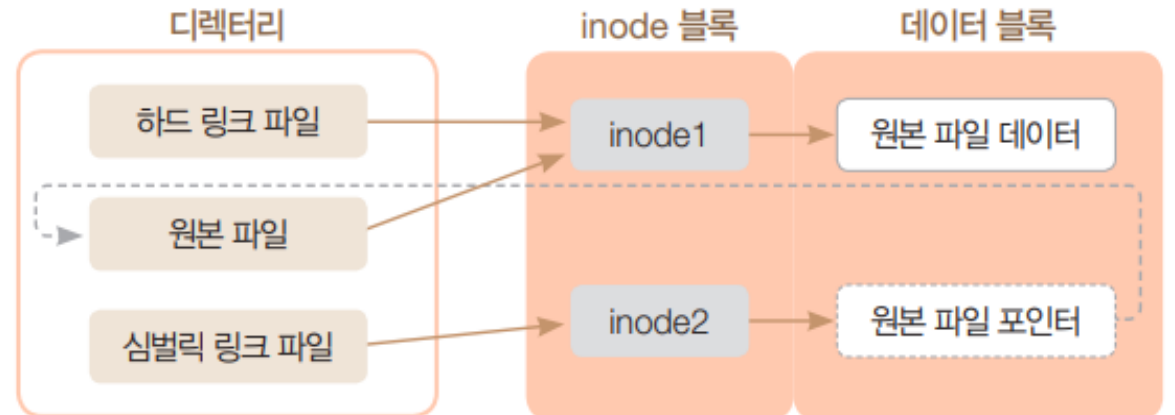
- ① ㉠ .virc, ㉡ set no ② ㉠ .virc, ㉡ set nu
③ ㉠ .exerc, ㉡ set no ④ ㉠ .exerc, ㉡ set nu

3. 링크

하드 링크와 심벌릭 링크

- inode
 - 파일 시스템 내에서 파일이나 디렉토리의 메타데이터를 저장하는 구조

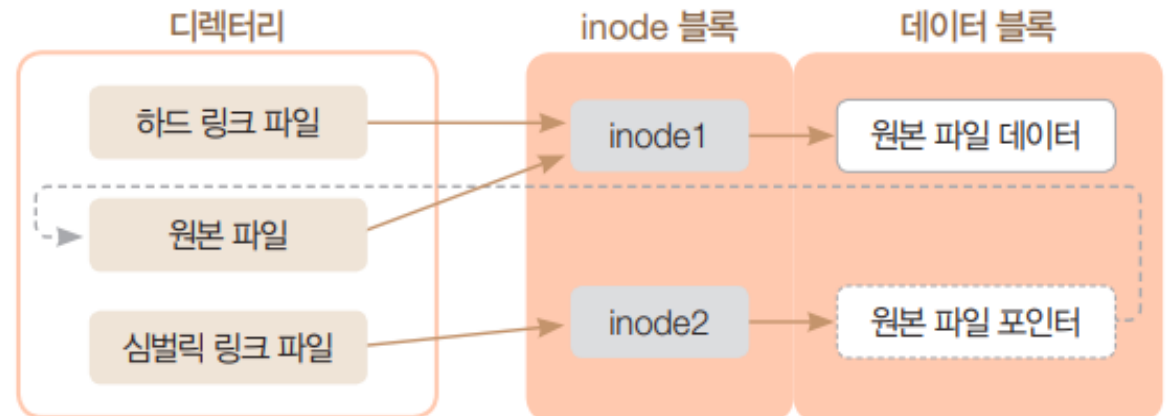
- 파일의 링크는 하드 링크 (hard link)와 심벌릭 링크(symbolic link or soft link)로 구분됨
- 하드 링크 (Hard link)
 - 파일의 실제 데이터에 대한 직접적인 포인터
 - 하드 링크를 생성하면 원본 파일과 동일한 물리적 위치를 가리키는 새로운 파일명이 생성
 - 파일 시스템 내에서 같은 inode 공유
 - → 하나를 수정하면 다른 하나도 동일하게 수정
 - 원본 파일을 삭제해도 하드 링크를 통해 데이터에 계속 접근 가능
 - 같은 파일 시스템 내에서만 생성
 - 디렉토리에 대해서는 만들 수 없음
 - In 원본 파일 링크파일명



하드 링크와 심벌릭 링크

- inode
 - 파일 시스템 내에서 파일이나 디렉토리의 메타데이터를 저장하는 구조

- 심벌릭 링크 (Symbolic link)
 - 파일이나 디렉토리를 가리키는 파일의 경로를 포함하는 특별한 파일
 - Windows의 바로가기
 - 원본 파일에 대한 간접적인 포인터 (하나를 수정하면 다른 하나도 동일하게 수정 ← 하드 링크와 같음)
 - 원본 파일의 위치나 이름이 변경되면 더 이상 유효하지 않음
 - 원본 파일 삭제 시 “죽은” 링크라고 함
 - 다른 파일 시스템에 대해서 생성 가능
 - 디렉토리에 대해서도 생성 가능
 - `ln -s 원본파일 링크파일명`



하드 링크와 심벌릭 링크 실습

- 1. 파일 생성하고 확인
 - home 디렉토리에 linkdir 디렉토리 생성
 - linkdir 디렉토리 내 originalfile 생성
 - originalfile 내 내용 입력
 - 파일 내용 확인

하드 링크와 심벌릭 링크 실습

- 2. 하드 링크와 심벌릭 링크 생성

```
ln originalfile hardlink    -- 하드 링크 생성
ln -s originalfile softlink -- 심벌릭 링크(소프트 링크) 생성
ls -il                     -- -il 옵션은 inode 번호를 맨 앞에 출력
cat hardlink               -- 하드 링크의 내용 확인
cat softlink               -- 심벌릭 링크의 내용 확인
```

- 3. 원본 파일을 다른 곳으로 이동하고 하드 링크와 파일 확인

```
mv originalfile ../ -- 원본 파일을 상위 디렉터리(..)로 이동
ls -il
cat hardlink
cat softlink
```

- 4. 원본 파일을 현재 디렉터리에 다시 가져오면 심벌릭 링크가 원상태로 복구됨

4. 파일과 디렉토리 검색

파일 내용 검색

- `grep [option] pattern filenames`
 - 지정한 파일의 내용을 검색
 - 정규표현식으로 표현 가능
 - option

옵션	기능
<code>-i</code>	대소문자를 무시하고 검색
<code>-l</code>	해당 패턴이 들어있는 파일 이름을 출력
<code>-n</code>	각 라인의 번호도 함께 출력
<code>-v</code>	명시된 패턴과 일치하지 않는 줄을 출력
<code>-c</code>	패턴과 일치하는 라인수 출력
<code>-w</code>	패턴이 하나의 단어로 된 것만 검색

파일 내용 검색

- grep 사용 예제
 - grep.dat 파일을 생성하고 다음 내용 입력
 - `$ touch grep.dat`
 - `$ vi grep.dat`
 - ...

grep.dat

```
UNIX 12345
unix+ 123
system admin
Network 5
root other sh
sjyoun prof ksh
jongwon prof KSH
ROOT other csh
ck07555 student ksh
CK08777 student bash
```

파일 내용 검색

- grep 사용 예제 (cont'd)

- 1) 기본 검색

- `$ grep unix grep.dat`

- → “unix” 라는 단어가 포함된 모든 줄을 출력

- 2) 대소문자 무시 (option -i)

- `$ grep -i unix grep.dat`

- → 대소문자를 무시하고 unix라는 단어가 포함된 모든 줄을 출력

grep.dat

```
UNIX 12345
unix+ 123
system admin
Network 5
root other sh
sjyoun prof ksh
jongwon prof KSH
ROOT other csh
ck07555 student ksh
CK08777 student bash
```

파일 내용 검색

- grep 사용 예제 (cont'd)

- 3) 줄번호 출력 (option -n)

- `$ grep -n unix grep.dat`

- → “unix” 라는 단어가 포함된 모든 줄의 줄 번호를 내용과 함께 출력

- 4) 파일 이름 출력 (option -l)

- `$ grep -l unix *`

- → 현재 디렉토리 내 모든 파일 (*)에서 “unix”라는 문자가 포함된 파일의 이름을 출력

- *: 정규표현식 중 하나

grep.dat

```
UNIX 12345
unix+ 123
system admin
Network 5
root other sh
sjyoun prof ksh
jongwon prof KSH
ROOT other csh
ck07555 student ksh
CK08777 student bash
```

파일 내용 검색

- grep 사용 예제 (cont'd)
 - 5) 단어 검색 (option -w)
 - `$ grep -w unix grep.dat`
 - → “unix” 라는 독립적인 단어를 포함하는 줄의 내용을 출력
 - 6) 단어 수 세기 (option -c)
 - `$ grep -c unix grep.dat`
 - → unix의 출현 빈도 출력
 - 7) 패턴 불일치 (option -v)
 - `$ grep -v unix grep.dat`
 - → unix라는 단어가 없는 줄의 내용을 출력

grep.dat

```
UNIX 12345
unix+ 123
system admin
Network 5
root other sh
sjyoun prof ksh
jongwon prof KSH
ROOT other csh
ck07555 student ksh
CK08777 student bash
```

파일 내용 검색

- grep 사용 예제 (cont'd)
 - 다중 옵션 사용 가능
 - `$ grep -ni unix grep.dat`
 - `$ grep -il unix *`
 - `$ grep -ci unix grep.dat`

grep.dat

```
UNIX 12345
unix+ 123
system admin
Network 5
root other sh
sjyoun prof ksh
jongwon prof KSH
ROOT other csh
ck07555 student ksh
CK08777 student bash
```


파일 내용 검색

- grep 실습
 - /etc/passwd 파일 내에서 “root”를 정확하게 포함하고 있는 줄의 내용과 줄 번호를 함께 출력
 - 아래 내용을 포함하는 h.txt 파일 생성 후 해당 파일 내 “sh”라는 단어를 정확하게 몇 번 나오는지 출력
 - h.txt

```
root other sh
ROOT csh user1
USER2 12root ksh
user1 KSH csh
```

- 1이 들어있지 않는 줄을 검색하기 위한 명령은?
- User를 대소문자 구분없이 검색하기 위한 명령은?
- sh가 들어간 줄의 수를 구하는 명령은?
- 정확히 sh만 들어간 줄의 수를 구하는 명령은?

Regular expression

- 정규 표현식
 - 데이터 검색, 복잡한 패턴 매칭을 보조하는 특별한 문자 집합 및 표현
 - regexp, regex 등으로 표현됨
 - SQL, R, Python 등에서도 다양하게 활용됨
 - find, grep 등의 명령어와 조합
 - 정규표현식으로 구성된 식은 따옴표(")로 감싸주어야 함

Regular expression 예제

test.txt

The quick brown fox jumps over the lazy dog dg doog.
 Pack my box with five dozen liquor jugs.
 Jackdaws love my big sphinx of quartz pack.
 Jinxed wizards pluck ivy from the big quilt.
 The five boxing wizards jump juump jujump quickly.

- 기본 메타문자 (확장된 정규표현식 사용시 -E option 필요)

메타문자	의미	예제 패턴	예제의 결과
.	임의의 한 문자	f.x	The quick brown fox jumps over the lazy dog dg doog.
[]	문자 집합 중 하나	[Pp]ack	Pack my box with five dozen liquor jugs. Jackdaws love my big sphinx of quartz pack .
[^]	부정 문자 집합	[^a-zA-Z]	The quick brown fox jumps over the lazy dog dg doog. Pack my box with five dozen liquor jugs. Jackdaws love my big sphinx of quartz pack. Jinxed wizards pluck ivy from the big quilt. The five boxing wizards jump juump jujump quickly.
^	줄의 시작	^The	The quick brown fox jumps over the lazy dog dg doog. The five boxing wizards jump juump jujump quickly.
\$	줄의 끝	doog.\$	The quick brown fox jumps over the lazy dog dg doog .

Regular expression 예제

test.txt

The quick brown fox jumps over the lazy dog dg doog.
 Pack my box with five dozen liquor jugs.
 Jackdaws love my big sphinx of quartz pack.
 Jinxed wizards pluck ivy from the big quilt.
 The five boxing wizards jump juump jujump quickly.

- 수량자 (확장된 정규표현식 사용시 -E option 필요)

메타문자	의미	예제 패턴	예제의 결과
*	0회 이상 반복	do*g	The quick brown fox jumps over the lazy dog dg doog .
+	1회 이상 반복	ju+mp	The quick brown fox jumps over the lazy dog dg doog. The five boxing wizards jump juump jujump quickly.
?	0회 또는 1회	jumps?	The quick brown fox jumps over the lazy dog dg doog. The five boxing wizards jump juump jujump quickly.
{n}	정확히 n회 반복	o{2}	The quick brown fox jumps over the lazy dog dg doog .
{n,}	최소 n회 반복	u{2,}	The five boxing wizards jump juump jujump quickly.
{n,m}	n회에서 m회 반복	i{1,3}	The quick brown fox jumps over the lazy dog dg doog. Pack my box with five dozen liquor jugs. Jackdaws love my big sphinx of quartz pack. J inxed w izards pluck i vy from the big quilt. The five box i ng w izards jump juump jujump quickly.

Regular expression 예제

test.txt

The quick brown fox jumps over the lazy dog dg doog.
 Pack my box with five dozen liquor jugs.
 Jackdaws love my big sphinx of quartz pack.
 Jinxed wizards pluck ivy from the big quilt.
 The five boxing wizards jump juump jujump quickly.

- 위치 지정 (확장된 정규표현식 사용시 -E option 필요)

메타문자	의미	예제 패턴	예제의 결과
\wb	단어 경계	'\bjump'	The quick brown fox jumps over the lazy dog. The five boxing wizards jump quickly.
		'\bjump\wb'	The five boxing wizards jump quickly.
\wB	단어 경계가 아님	'iv\wB'	Pack my box with five dozen liquor jugs. Jinxed wizards pluck ivy from the big quilt. The five boxing wizards jump quickly
		'\wBiv\wB'	Pack my box with five dozen liquor jugs. The five boxing wizards jump quickly.

Regular expression 예제

test.txt

The quick brown fox jumps over the lazy dog dg doog.
Pack my box with five dozen liquor jugs.
Jackdaws love my big sphinx of quartz pack.
Jinxed wizards pluck ivy from the big quilt.
The five boxing wizards jump juump jujump quickly.

- 특수 문자 (확장된 정규표현식 사용시 -E option 필요)

메타문자	의미	예제 패턴	예제의 결과
<code>\ww</code>	단어를 구성하는 문자	<code>'\ww+'</code>	The quick brown fox jumps over the lazy dog dg doog. Pack my box with five dozen liquor jugs. Jackdaws love my big sphinx of quartz pack. Jinxed wizards pluck ivy from the big quilt. The five boxing wizards jump juump jujump quickly.
<code>\WW</code>	단어를 구성하지 않는 문자	<code>'\WW+'</code>	The quick brown fox jumps over the lazy dog dg doog. Pack my box with five dozen liquor jugs. Jackdaws love my big sphinx of quartz pack. Jinxed wizards pluck ivy from the big quilt. The five boxing wizards jump juump jujump quickly.

Regular expression 예제

- `'\bbw{3,}\bb'`
 - 알파벳 문자가 3회 이상 반복되는 단어 찾기

- `'\bw\ww*h\bb'`
 - w로 시작해서 h로 끝나는 단어 찾기 (e.g., with)

- `'[aeiou]\w[aeiou]'`
 - 두 개의 모음 사이에 있는 한 글자 찾기

- `'\b[^e\ws]+\bb'`
 - e와 공백을 포함하지 않는 단어 찾기

Regular expression 예제

grep.dat

```
UNIX 12345
unix+ 123
system admin
Network 5
root other sh
sjoun prof ksh
jongwon prof KSH
ROOT other csh
ck07555 student ksh
CK08777 student bash
```

- \$ grep -E '[78]+' grep.dat
 - ck07555 student ksh
 - CK08777 student bash
- \$ grep -E 'csh|bash' grep.dat
 - ROOT other csh
 - ck07555 student ksh

Regular expression 예제

grep.dat

```
UNIX 12345
unix+ 123
system admin
Network 5
root other sh
sjyoun prof ksh
jongwon prof KSH
ROOT other csh
ck07555 student ksh
CK08777 student bash
```

- `$ grep -E '^root' grep.dat`
- `$ grep -E 'sh$' grep.dat`
- `$ grep -E 'r..t' grep.dat`
- `$ grep -E 'oo*' grep.dat`
- `$ grep -E '[0-9].*' grep.dat`
- `$ grep -E '[^c]sh' grep.dat`

Regular expression 예제

test2.txt

In 2024, the global tech conference will be held in San Francisco on July 24th.
This event aims to bring together innovators from around the world.
For more details, check our website at <http://tech-conference.com>.
Questions?
Email us at info@tech-conference.com or reach out via our hotline at +1-800-555-0199

- 이메일 주소 찾기
 - `$ grep -E '[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\w.[a-zA-Z]{2,}' test2.txt`
 - > Email us at info@tech-conference.com or reach out via our hotline at +1-800-555-0199

Regular expression 예제

test2.txt

In 2024, the global tech conference will be held in San Francisco on July 24th.
This event aims to bring together innovators from around the world.
For more details, check our website at <http://tech-conference.com>.
Questions?
Email us at info@tech-conference.com or reach out via our hotline at +1-800-555-0199

- 이메일 주소 찾기
 - `$ grep -E '[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\w.[a-zA-Z]{2,}' test2.txt`
 - 대괄호 [] 내의 문자 집합은 해당 문자 중 하나와 일치한다는 것을 의미
 - a-z A-Z 0-9 . _ % + - 중 하나와 일치하는 것을 찾음

Regular expression 예제

test2.txt

In 2024, the global tech conference will be held in San Francisco on July 24th.
This event aims to bring together innovators from around the world.
For more details, check our website at <http://tech-conference.com>.
Questions?
Email us at info@tech-conference.com or reach out via our hotline at +1-800-555-0199

- 이메일 주소 찾기
 - `$ grep -E '[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\w.[a-zA-Z]{2,}' test2.txt`
 - [] 내의 패턴이 한 번 이상 반복됨을 의미함

Regular expression 예제

test2.txt

In 2024, the global tech conference will be held in San Francisco on July 24th.
This event aims to bring together innovators from around the world.
For more details, check our website at <http://tech-conference.com>.
Questions?
Email us at info@tech-conference.com or reach out via our hotline at +1-800-555-0199

- 이메일 주소 찾기
 - `$ grep -E '[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\w.[a-zA-Z]{2,}' test2.txt`
 - 이메일 형식에서 도메인과 주소를 구분하는 @

Regular expression 예제

test2.txt

In 2024, the global tech conference will be held in San Francisco on July 24th.
This event aims to bring together innovators from around the world.
For more details, check our website at <http://tech-conference.com>.
Questions?
Email us at info@tech-conference.com or reach out via our hotline at +1-800-555-0199

- 이메일 주소 찾기
 - `$ grep -E '[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\w.[a-zA-Z]{2,}' test2.txt`
 - 도메인 이름에서 사용될 수 있는 문자 집합이 하나 이상 반복됨

Regular expression 예제

test2.txt

In 2024, the global tech conference will be held in San Francisco on July 24th.
This event aims to bring together innovators from around the world.
For more details, check our website at <http://tech-conference.com>.
Questions?
Email us at info@tech-conference.com or reach out via our hotline at +1-800-555-0199

- 이메일 주소 찾기
 - `$ grep -E '[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\w.[a-zA-Z]{2,}' test2.txt`
 - 하나의 문자를 나타내는 정규표현식 . 이 아닌 문자 그대로 (literal) 의 마침표를 의미

Regular expression 예제

test2.txt

In 2024, the global tech conference will be held in San Francisco on July 24th.
This event aims to bring together innovators from around the world.
For more details, check our website at <http://tech-conference.com>.
Questions?
Email us at info@tech-conference.com or reach out via our hotline at +1-800-555-0199

- 이메일 주소 찾기
 - `$ grep -E '[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\w.[a-zA-Z]{2,}' test2.txt`
 - 최상위도메인(TLD)에서 사용될 수 있는 문자 조합이 두 번 이상 반복되는 것을 찾음
 - .com, .edu 정도를 찾을 수 있음
 - ac.kr, go.jp 같은 조합은 못 찾음
 - 보완방법: `[a-zA-Z]{2,}\w.[a-zA-Z]{2,}`

Regular expression 예제

- Quiz – 다음 정규표현식을 보고 찾을 수 있는 것을 모두 고르시오
 - *.png
 - image.png
 - photo_of_the_sky.png
 - image.jpg
 - document.pdf
 - $\mathbb{W}^*[A-Z].*[.,]$\mathbb{W}$
 - Hello,
 - This is a sentence
 - hello.
 - This is a correct

Regular expression 예제

- Quiz – 다음 정규표현식을 보고 찾을 수 있는 것을 모두 고르시오
 - `bW?right`
 - bright
 - brright
 - b?right
 - `s+right`
 - right
 - sright
 - sight
 - ssright
 - ssssrigh

파일 검색

- find 명령어
 - 사용자가 시스템 내 존재하는 특정 파일을 찾을 때 사용
 - 검색 범위를 디렉토리 단위로 지정
 - 특정 파일의 이름, 여러 개의 파일을 지정하는 패턴, 파일의 속성을 조합하여 검색 가능
 - (정규)표현식과 일치하는 파일에 대해 파일의 절대 경로를 출력하거나 특정 명령 실행 가능

파일 검색

- **find 경로 검색조건 [동작]**
 - 경로
 - 파일을 찾을 디렉토리의 절대경로 또는 상대경로
 - 검색조건
 - 파일을 찾기 위한 검색 기준
 - and, or을 이용하여 조건 결합 가능
 - 동작
 - 파일의 위치를 찾은 후 수행할 동작 지정
 - 기본 동작은 파일의 정대 경로를 화면에 출력

파일 검색

- 경로 설정 예

경로 표현	찾기 시작 위치
~	홈 디렉토리에서 찾기 시작
.	현재 디렉토리에서 찾기 시작
/etc	/etc 디렉토리에서 찾기 시작 (절대 경로)
/	/(root) 디렉토리에서 찾기 시작 (전체 파일 시스템 검색)
unix	unix 디렉토리에서 찾기 시작 (상대 경로)

파일 검색

- 검색조건 종류

검색조건표현	의미	기능
-name filename	파일 이름	특정 파일명에 일치하는 파일 검색 메타 문자(*,?)사용도 가능하나 “ “안에 있어야 함
-type	파일 종류	특정 파일 종류에 일치하는 파일 검색(f,d)
-mtime [+]-n -atime [+]-n	수정(접근) 시간	수정(접근)시간이 +n일보다 오래되거나, -n일보다 짧거나 정확히 n일에 일치하는 파일 검색
-user loginID	사용자 ID	loginID가 소유한 파일 모든 파일 검색
-size [+]-n	파일 크기	+n보다 크거나, -n보다 작거나, 정확히 크기가 n인 파일 검색(n=512bytes)
-newer	기준 시간	기준 시간보다 이후에 생성된 파일 검색
-perm	사용 권한	사용 권한과 일치하는 파일 검색(8진수)

파일 검색

- 동작 종류

동작	정의
-exec 명령 {} \;	exec 옵션은 \;으로 끝남 검색된 파일은 {} 위치에 적용됨
-ok 명령 {} \;	exec의 확인모드 형태 사용자의 확인을 받아야 명령을 적용(rm -i)
-print	화면에 경로명을 출력 (기본 동작)
-ls	긴 목록 형식으로 검색 결과를 출력

- 검색 조건의 결합 기호

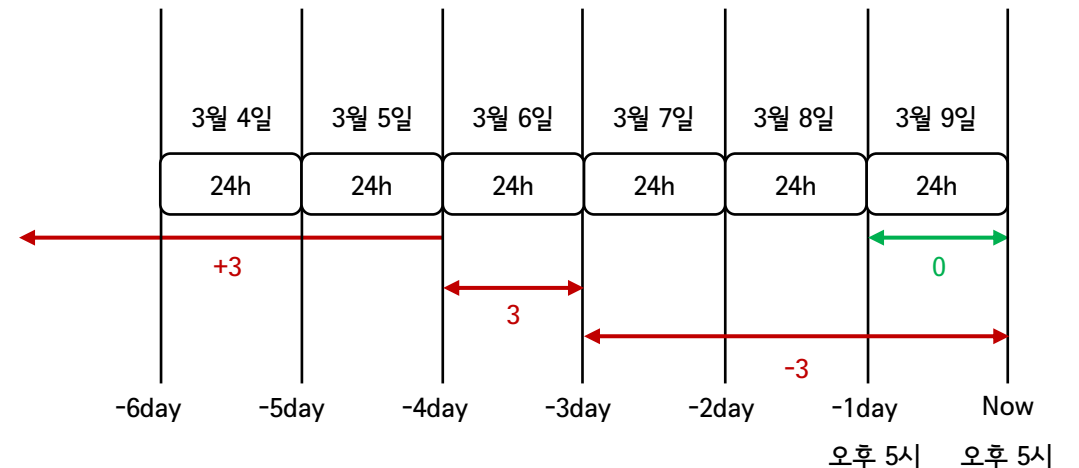
- -a: and
- -o: or
- !: not

find 사용 예제

- -name: 이름으로 찾기
 - `$ find ~ -name grep.dat`
- -type: 파일 타입으로 찾기
 - f: 파일, d: 디렉토리, l: **심볼릭 링크**
 - `$ find ~ -type d`
- -newer: 옵션 뒤에 적힌 파일보다 최근에 변경된 파일 검색
 - `$ find ~ -newer g.dat`

find 사용 예제

- -mtime: 파일 수정 (modify) 시점으로 찾기
- -atime: 파일 접근 (access) 시점으로 찾기
- -ctime: 파일 속성값(권한, 소유주 등)이 바뀐 (change) 시점으로 찾기
 - +/- 일 단위
 - +n: n+1일 이상 지난 날짜에 수정된 파일 검색
 - +3 → 3일 보다 오래전의 파일들 (3일 포함X)
 - -n: n일 이내에 수정된 파일 검색
 - -3 → 3일 이내의 파일들 (3일, 현재 모두 포함)
 - n: n일전에 수정된 파일 검색
 - 3 → 3일 전에 수정된 파일들
 - 0: 정확히 24시간 전 이내로 수정된 파일들

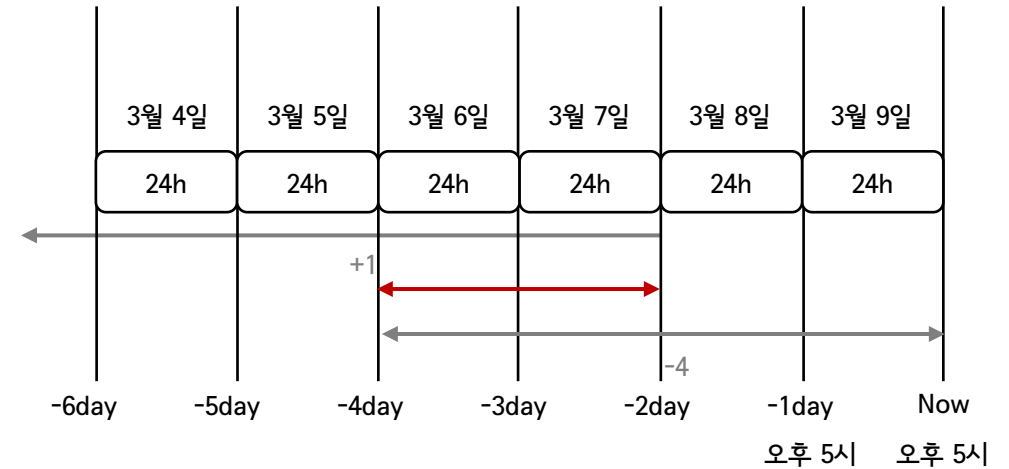
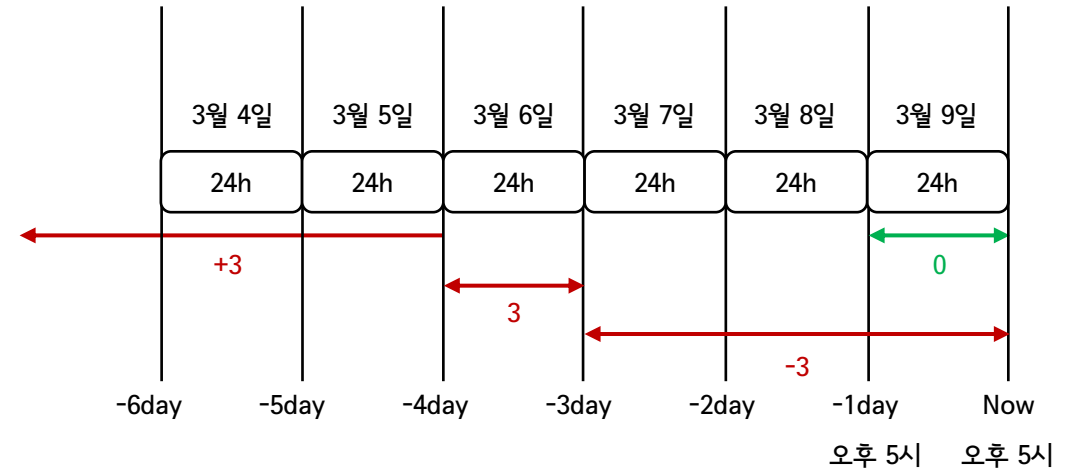


find 사용 예제

- mtime 예제

- `$ find ./ -mtime -1`
- `$ find ~ -mtime 0`
- `$ find ~ -mtime +1`

- `$ find ./ -mtime +1 -mtime -4`
 - 48시간 이전 96시간 이내에 작성된 파일 조회



find 사용 예제

- 그 외 간간히 쓰이는 옵션들
 - -size: 크기에 따른 검색 (+/-: 1 = 512byte (block) 기준)
 - `$ find . -size 1`
 - `$ find . +size 10`
 - -perm: 권한에 따른 검색
 - `$ find ./ -perm 0755`
 - `$ find ./ -perm 0664`

find 사용 예제

- 검색조건 조합

- `$ find ~ -type d -name Unix`

- 별도의 조건 없이 이어서 옵션을 달면 자동으로 and 처리

- `$ find ./ -type d -o -perm 0755`

- 파일 유형이 디렉토리이거나 권한이 755인 파일

- `$ find ../ ! -newer h.dat`

- h.dat 파일의 수정일자보다 이후에 수정된 파일이 아닌 파일들

find 사용 예제

- 동작 조건 – 검색된 파일 처리

- 삭제

- `$ find ~ -name test.dat -exec rm {} \;`

- 파일명 변경

- `$ find ~ -name test.dat -exec mv {} test2.dat \;`

grep과 파이프를 이용한 파일 찾기

- grep 명령은 파이프와 함께 자주 사용됨
 - 파이프 (|)
 - 한 명령의 실행 결과를 다음 명령의 입력으로 전달
 - 파이프 양쪽에 명령이 와야 함 → \$ 명령 1 | 명령 2 | 명령 3
 - ex) \$ ls /etc | more
- grep과 파이프의 조합
 - \$ ls -l | grep

find 사용 실습

- 각 명령의 의미 파악 연습

```
1) $ find /etc -type l
2) $ find . -mtime -1 -name report.txt
3) $ find ~ -size +2
4) $ find . -ls
5) $ find /export/home -user user1
6) $ find . -name *.dat -exec mv {} Practice \;
7) $ find . -type f
```

- 1) /etc 디렉토리에 있는 심볼릭링크 파일 찾기
- 2) 어제 작업한 report.txt 파일 찾기
- 3) 크기가 2블록(1KB)보다 큰 파일 찾기
- 4) 전체 파일
- 5) User1 사용자의 파일
- 6) *.dat 파일 찾아 Practice 디렉토리로 이동
- 7) 현재 디렉토리에서 파일 찾기

명령어 찾기

- **whereis 명령**

- 해당 명령의 지정된 경로 출력
- `$ whereis ls`
- `$ whereis find`
- `$ whereis ifconfig`

- **which 명령**

- PATH 환경변수에 지정된 경로에서 명령을 찾음
- `$ which ls`
- `$ which find`
- `$ which ifconfig`

Assignment 01

- LMS 공지 참조

5. 파일 백업과 압축

Getting started

- 파일을 하나로 묶고 압축하는 이유는 무엇인가?
 - 파일의 용량을 줄여 디스크 사용량을 효율적으로 사용
 - 파일 전송(업로드/다운로드) 시간 단축
 - 배포의 편리성
 - 파일보관의 편리성

파일 아카이브

- 아카이브 (Archive)
 - 파일과 디렉토리를 묶어 하나로 만든 것
 - tar 명령어를 사용하여 아카이브화 할 수 있음

- tar 옵션 [아카이브파일이름] 파일이름

옵션	기능
c	tar 파일 생성
t	tar 파일 목록 보기
x	tar 파일 풀기
f	아카이브 파일이나 tape 장치 등 지정
v	표준 출력으로 실행 내용 출력
h	심볼릭 링크의 원본 파일 포함

파일 아카이브

- 아카이브 생성: cvf

- `$ tar cvf c.tar *.txt`

- 확장자가 txt인 파일을 묶어서 c.tar 아카이브를 생성

- 아카이브 생성 확인 (목록보기): tvf

- `$ tar tvf c.tar`

- c.tar 아카이브 내 파일들의 목록 출력

- 아카이브 풀기: xvf

- `$ tar xvf c.tar`

- tar는 '-' 없이 옵션을 바로 활용하기도 함

- cvf

- c: create

- v: verbose (상세 정보표시)

- f: file (사용할 파일명 지정)

파일 아카이브

- 아카이브 업데이트: uvf
 - 수정 파일에 한해 tar 파일 마지막에 생성
 - 아카이브 풀기 시 최신 파일로 교체
 - `$ tar uvh c.tar *.txt`
- 파일 추가: rvf
 - 수정여부와 상관없이 지정된 파일 추가
 - `$ tar rvf c.tar test`

파일 아카이브 실습

1) \$ cd

2) \$ mkdir linux

3) \$ cd linux

4) \$ cp /etc/hosts ./

5) \$ cp /etc/passwd ./

6) \$ cp /etc/networks ./

7) \$ cd ..

8) \$ tar cvf linux.tar linux

9) \$ tar tvf linux.tar

10) \$ mkdir linux/test

11) \$ mv linux.tar linux/test

12) \$ cd linux/test

13) \$ tar xvf linux.tar

14) \$ ls

파일 압축 – gzip / gunzip

- **gzip [옵션] 파일이름**
 - 확장자가 .gz인 압축 파일 생성
 - 옵션
 - -d: 압축해제 (unzip과 동일)
 - -l: 압축된 파일의 내용 출력
 - -r: 현재 디렉토리부터 하위 디렉토리까지 모두 압축
 - -v: 압축 정보 화면에 출력

```
[root@localhost linux]# ll
합계 12
-rw-r--r--. 1 root root 158 2012-03-28 09:17 hosts
-rw-r--r--. 1 root root 58 2012-03-28 09:18 networks
-rw-r--r--. 1 root root 2060 2012-03-28 09:17 passwd
[root@localhost linux]# gzip passwd
[root@localhost linux]# ll
합계 12
-rw-r--r--. 1 root root 158 2012-03-28 09:17 hosts
-rw-r--r--. 1 root root 58 2012-03-28 09:18 networks
-rw-r--r--. 1 root root 811 2012-03-28 09:17 passwd.gz
[root@localhost linux]# gzip -d passwd.gz
[root@localhost linux]# ll
합계 12
-rw-r--r--. 1 root root 158 2012-03-28 09:17 hosts
-rw-r--r--. 1 root root 58 2012-03-28 09:18 networks
-rw-r--r--. 1 root root 2060 2012-03-28 09:17 passwd
```


파일 압축 – gzip / gunzip

- gunzip 파일이름
 - 확장자가 .gz인 압축 파일 해제

```
[root@localhost linux]# ll
합계 12
-rw-r--r--. 1 root root 158 2012-03-28 09:17 hosts
-rw-r--r--. 1 root root 58 2012-03-28 09:18 networks
-rw-r--r--. 1 root root 2060 2012-03-28 09:17 passwd
[root@localhost linux]# gzip ./
[root@localhost linux]# ll
합계 12
-rw-r--r--. 1 root root 80 2012-03-28 09:17 hosts.gz
-rw-r--r--. 1 root root 74 2012-03-28 09:18 networks.gz
-rw-r--r--. 1 root root 811 2012-03-28 09:17 passwd.gz
[root@localhost linux]# gunzip ./
[root@localhost linux]# ll
합계 12
-rw-r--r--. 1 root root 158 2012-03-28 09:17 hosts
-rw-r--r--. 1 root root 58 2012-03-28 09:18 networks
-rw-r--r--. 1 root root 2060 2012-03-28 09:17 passwd
```

파일 압축 – zip / unzip

- **zip [옵션] 압축파일이름 압축대상파일**

- 확장자가 .zip인 압축 파일 생성
- 옵션
 - -v: 압축 정보 화면에 출력
 - -r: 디렉토리 압축
 - -u: 수정되거나 추가된 파일만 출력

- gzip과 zip의 차이

- gzip: 단일 파일의 압축 용도 → tar로 묶은 후 압축
 - 압축할 때 원본 파일을 압축 파일로 대체
- zip: 여러 개의 파일/디렉토리를 바로 압축
 - Cross-platform 호환성 (Windows, Mac, Linux, etc.)

```
[root@localhost linux]# ll
합계 12
-rw-r--r--. 1 root root 158 2012-03-28 09:17 hosts
-rw-r--r--. 1 root root 58 2012-03-28 09:18 networks
-rw-r--r--. 1 root root 2060 2012-03-28 09:17 passwd
[root@localhost linux]# zip passwd.zip passwd
adding: passwd (deflated 62%)
[root@localhost linux]# ll
합계 16
-rw-r--r--. 1 root root 158 2012-03-28 09:17 hosts
-rw-r--r--. 1 root root 58 2012-03-28 09:18 networks
-rw-r--r--. 1 root root 2060 2012-03-28 09:17 passwd
-rw-r--r--. 1 root root 930 2012-03-29 12:53 passwd.zip
```

```
[root@localhost ~]# zip -r linux.zip linux
adding: linux/ (stored 0%)
adding: linux/networks (deflated 19%)
adding: linux/passwd (deflated 62%)
adding: linux/hosts (deflated 65%)
[root@localhost ~]# ll linux.zip
-rw-r--r--. 1 root root 1437 2012-03-29 12:54 linux.zip
[root@localhost ~]#
```

파일 압축 – zip / unzip

- unzip (압축)파일이름
 - 확장자가 .zip인 압축 파일 해제

```
[root@localhost ~]# ll linux.zip
-rw-r--r--. 1 root root 1437 2012-03-29 12:54 linux.zip
[root@localhost ~]# unzip linux.zip
Archive:  linux.zip
  creating: linux/
  inflating: linux/networks
  inflating: linux/passwd
  inflating: linux/hosts
```

파일 압축 실습

1) \$ cd

2) \$ mkdir linux

3) \$ cd linux

4) \$ cp /etc/hosts ./

5) \$ cp /etc/passwd ./

6) \$ cp /etc/networks ./

7) \$ cd ..

8) \$ mkdir Backup

9) \$ cd linux

10) \$ tar cvf linux.tar *

11) \$ mv linux.tar ~/Backup

12) \$ cd ~/Backup

13) \$ ls

14) \$ Gzip linux.tar

기타 파일 압축 확장자

- tar.bz2

- tar -jcvf .bz2로 압축

- tar -jxvf .bz2 파일 압축 풀기

11. 다음 중 아파치 웹 서버 소스 파일을 내려받은 후 압축을 해제하는 과정이다. (괄호) 안에 들어갈 내용으로 알맞은 것은?

```
# tar ( 괄호 ) httpd-2.4.53.tar.bz2
```

① jxvf

② Jxvf

③ zxvf

④ Zxvf

End of slide
