Vi-vim 편집기 명령어 단축키

vi 시작 명령어

명령어	설명	예제
vi {파일명}	파일열기, 작성	vi test.txt

vi 커서 이동

커서	설명
h (←)	왼쪽으로 커서 이동
j (↓)	아래로 커서 이동
k (↑)	위로 커서 이동
l (→)	오른쪽으로 커서 이동

문자, 행 , 삽입 명령어

커서	설명
a	커서 오른쪽에 문자 삽입

ESC	종료

텍스트 삭제 명령어

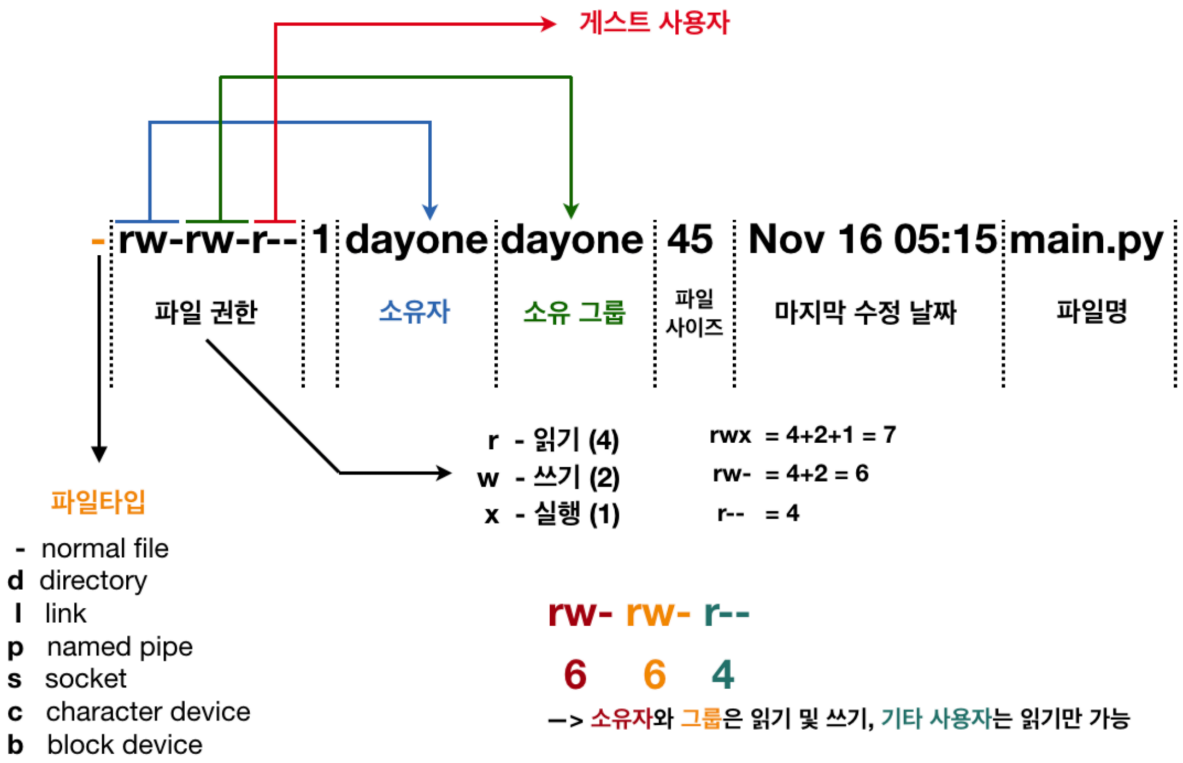
명령어	설명
x	커서가 있는 문자 삭제

보관 및 종료 명령어

명령어	설명
:w	변경사항 저장
:w {파일명}	변경사항 입력한 파일명으로 저장
:wq	변경사항 보관 후 vi 종료. ZZ 명령과 같음. :w(기록)과 :q(종료) 를 연속적으로 수행.
ZZ	변경사항 보관 후 vi 종료. 임시 버퍼의 내용을 vi로 호출할때 사용되었던 파일에 기록한 후 vi를 빠져나옴.
:q!	변경사항 보관하지 않고 종료
q	수정한 파일을 저장하지 않고 vi 종료
e!	수정한 것을 무시하고 다시 편집상태로

리눅스 파일, 디렉토리 정보

ls -l 하여 현재 위치의 구체적인 정보 얻을 수 있음
파일타입/권한/파일개수/소유자/소유그룹/파일사이즈/마지막수정시각/파일명



ls 명령어의 모든 옵션들

- -A : .와 ..을 제외하고 목록을 출력합니다.
- -b : 알파벳 순으로 목록을 출력합니다.
- -B : ~로 끝나는 백업파일을 제외하고 목록을 출력합니다.
- -c : 마지막으로 변경된 시간을 목록을 출력합니다.
- -C : 파일이나 디렉토리를 열로 목록을 출력합니다.
- -d : 지정 경로에 있는 최상위 디렉토리의 목록만 출력합니다.
- -D : emacs를 위한 출력행태를 생성합니다.
- -f : 정렬하지 않고 출력합니다. 컬러을 해제합니다.
- -F : 실행파일은 *, 경로 /, 소켓=, 링크 @ 등의 지시자로 출력합니다.
- -g : 사용자 권한을 출력하지 않는다.
- -G : -l 과 같이 사용시 그룹권한을 출력하지 않는다.
- -h : K, M, G 단위를 사용하여 파일 크기를 사람이 보기 좋게 표시합니다.
- -H : 심볼릭 링크의 실제 참조하는 목록을 출력합니다.
- -i : 파일의 인덱스 값을 출력합니다.
- -I (대문자 i) : 지정 파티션을 제외하고 출력합니다.
- -k : 용량을 킬로바이트로 출력합니다.
- -l : 자세한 내용을 출력합니다. 내용 > 권한, 파일 수, 소유자, 그룹, 파일크기, 수정일자, 파일이름
- -lu : mtime (수정 시간)을 atime(접근 시간)을 출력합니다. (default는 수정 시간)

- -lc : mtime (수정 시간)을 ctime(변경 시간)을 출력합니다. (default는 수정 시간)
- -L : 심볼릭 링크의 정보를 출력할때 원본 파일의 정보를 출력합니다.
- -m : 콤마로 구분하여 출력합니다.
- -n : 사용자와 그룹권한을 숫자로 표시합니다.
- -i : 그룹권한을 출력하지 않는다.
- -p : 디렉토리에 /를 추가합니다.
- -q : 그래픽이 아닌 문자 대신에 ?를 출력합니다.
- -Q : 파일, 디렉토리를 쌍따옴표 안에 출력합니다.
- -r : 반대로 출력합니다. (default는 알파벳 순서)
- -R : 하위 디렉토리까지 출력합니다.
- -s : 블록에 할당된 크기를 출력합니다.
- -S : 파일 크기 순으로 정렬하여 출력합니다.
- -t : 파일이 수정된 시간 기준으로 정렬하여 출력합니다.
- -T : 8대신 COLS을 지정하여 출력합니다.
- -u : -lt와 같이 사용시 생성 시간 기준으로 출력하고 -l과 사용시 생성시간 출력이름순으로 출력합니다.
- -U : 컬러을 유지하면서 정렬하지 않고 출력합니다.
- -w : width 길이를 설정하여 출력합니다.
- -x : 상세출력되는 리스트를 파일이름으로 하나의 라인에 출력합니다.
- -X : 확장자의 알파벳순으로 정렬하여 출력합니다.
- -Z : SELinux 보안 모듈을 출력합니다.

링크

1. 단순 복사
   inode 값이 다름 -> 독립 파일 -> 수정, 삭제시 영향 없음

2. 심볼릭 링크(symbolic link, soft link)
   ls -s [원본파일] [링크파일]
   윈도우의 바로가기 기능과 유사
   call by reference. 원본의 위치 정보(포인터) 포함
   심볼릭 링크 크기 < 원본 파일
   원본이 삭제되면 심볼릭 링크는 에러를 나타냄
   심볼릭과 원본이 수정되면 서로 반영됨

3. 하드링크(hard link)
   ln [원본파일] [링크파일]
   원본 파일의 inumber가 복사되어서 동일한 inode를 사용
   하드와 원본이 수정되면 서로 반영됨
   원본이 삭제되어도 하드링크는 유지됨

쉘스크립트

쉘이 실행할 행동을 명령어로 적어 둔 것이 쉘스크립트. '.sh' 파일 확장자로 작성한다

만드는 방법(쉘 선언)
vi XX.sh
쉘 스크립트 1번 줄에는 #!/bin/bash를 적어야 함.

리눅스 sh 실행 권한 부여
Chmod 755 XX.sh

실행하는 방법
./XX.sh

git commit

gitignore

하나씩 gitignore에 넣는 것은 귀찮으니까 git add.을 이용해 전체 파일을 추가하고 커밋한다.
이때 보안, 무관련, 용량이 커 제외해야 하는 등의 경우는 다음과 같은 방식으로 무시한다.

1. git rm 로 각각 제거
2. .gitignore 디렉토리에 무시할 파일을 넣어주기

gitignore: 깃에서 제외시키는 파일. 커밋할 때 필요하지 않은 파일은 업데이트시키지 않음

gitignore 사용법

1. Git init을 한 폴더에 .gitignore라는 이름 의 파일을 만든다
   git init: 프로젝트 폴더 초기화
2. .gitignore 에 한 줄 씩 제외할 파일 혹은 폴더를 쓴다

 # 특정 파일 fileName 제외하기

```
fileName.js
```

 # 현재 경로에 있는 fileName_1 만 제외하기. (다른경로 fileName_1 말고)

```
/fileName.js
```

 # 특정 폴더 node_module 안의 파일 다 제외하기

```
node_module/
```

 # 특정 경로의 특정 파일 제외하기

```
folder/my.txt
```

 # 특정 경로 아래의 모든 fileName_2 제외하기

```
folder/**/fileName_2.txt
```

 # 특정 확장자 파일 다 제외하기

```
*.txt
```

 # 예외 만들기

```
!fileName.txt
```