

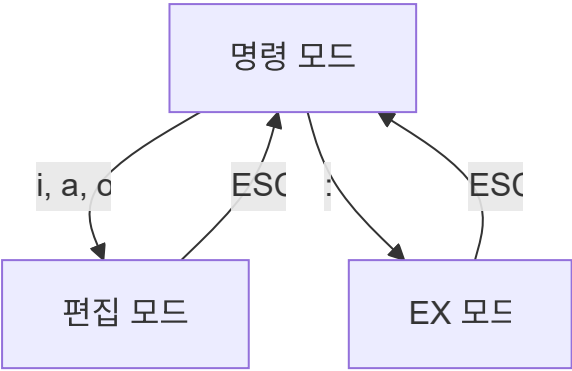
Linux 2.2

🔧 Vi 편집기와 리눅스 고급 명령어 완전 정리

📄 Vi 편집기

Vi 편집기는 리눅스에서 가장 널리 사용되는 텍스트 편집기로, 세 가지 모드로 구성된 모드형 편집기이다.

🔄 Vi 편집기의 3가지 모드



모드	설명	진입 방법
명령 모드	파일 편집, 이동, 복사 등의 명령을 실행하는 모드	Vi 시작 시 기본 모드
편집 모드	실제 텍스트를 입력하고 수정하는 모드	i, a, o 등
EX 모드	파일 저장, 종료, 설정 등을 수행하는 모드	:

🎯 명령 모드에서 편집 모드로 전환

📄 편집 모드 진입 명령어

명령어	설명
i	현재 커서 위치에서 입력 모드로 전환한다
a	현재 커서 다음 위치에서 입력 모드로 전환한다
o	현재 줄 아래에 새 줄을 만들고 입력 모드로 전환한다
I	현재 줄의 맨 앞에서 입력 모드로 전환한다
A	현재 줄의 맨 뒤에서 입력 모드로 전환한다
O	현재 줄 위에 새 줄을 만들고 입력 모드로 전환한다

명령어	설명
h	왼쪽으로 한 문자 이동한다
j	아래로 한 줄 이동한다
k	위로 한 줄 이동한다
l	오른쪽으로 한 문자 이동한다
0	현재 줄의 맨 앞으로 이동한다
\$	현재 줄의 맨 뒤로 이동한다
gg	파일의 첫 번째 줄로 이동한다
G	파일의 마지막 줄로 이동한다
nG	n번째 줄로 이동한다 (예: 10G)

명령 모드 - 복사와 붙여넣기

복사 및 붙여넣기 명령어

명령어	설명
yy	현재 줄을 복사한다
nyy	현재 줄부터 n줄을 복사한다 (예: 3yy)
yw	현재 커서부터 한 단어를 복사한다
p	커서 아래에 붙여넣는다
P	커서 위에 붙여넣는다

사용 예제

```
# 5줄 복사 후 다른 위치에 붙여넣기
5yy      # 5줄 복사
10G      # 10번째 줄로 이동
p        # 붙여넣기
```

명령 모드 - 문자열 치환

치환 명령어 구조

```
:%s/old_text/new_text/options
```

구성 요소	설명
%	전체 파일에 적용한다
s	치환(substitute) 명령이다
old_text	바뀔 원본 문자열이다
new_text	바꿀 새로운 문자열이다
g	한 줄에서 모든 일치하는 문자열을 바꾼다

💡 치환 예제

```
# 전체 파일에서 "Array"를 "Vector"로 바꾸기
:%s/Array/Vector/g

# 문자열 제거 (공백으로 치환)
:%s/unwanted_text//g

# 특정 줄 범위에서 치환 (1~10줄)
:1,10s/old/new/g

# 현재 줄에서만 치환
:s/old/new/g
```


편집 모드에서 명령 모드로 돌아가려면 `ESC` 키를 누른다.

⚙️ EX 모드 - 파일 저장 및 종료

명령 모드에서 `:` 을 입력하면 EX 모드로 전환된다.

📁 저장 및 종료 명령어

명령어	설명
<code>:w</code>	파일을 저장한다
<code>:w filename</code>	다른 이름으로 저장한다
<code>:wq</code>	저장하고 종료한다
<code>:wq!</code>	강제로 저장하고 종료한다
<code>:x</code>	저장하고 종료한다 (<code>:wq</code> 와 동일)
<code>:q</code>	종료한다 (변경사항이 없을 때)
<code>:q!</code>	강제 종료한다 (변경사항 무시)

🔧 설정 명령어

명령어	설명
<code>:set number</code> 또는 <code>:se nu</code>	행 번호를 표시한다
<code>:se nonu</code>	행 번호를 숨긴다
<code>:set autoindent</code>	자동 들여쓰기를 설정한다
<code>:syntax on</code>	문법 강조를 활성화한다

🏠 Vi 기본 설정 파일

홈 디렉토리에 `.vimrc` 파일을 생성하여 기본 설정을 저장할 수 있다.

```
vi ~/.vimrc
```

기본 설정 예제:

```
set number
set autoindent
set tabstop=4
syntax on
```

Vi 편집 과정 요약

```
# 1. 파일 열기
vi filename

# 2. 편집 모드로 전환
i

# 3. 텍스트 입력 및 편집
# (실제 내용 작성)

# 4. 명령 모드로 돌아가기
ESC

# 5. 저장하고 종료
:wq
```

리눅스 고급 명령어

ls 명령어 고급 옵션

ls 명령어는 파일과 디렉토리 목록을 출력하는 기본 명령어로, 다양한 옵션을 제공한다.

주요 옵션

옵션	설명
-l	파일의 상세 정보를 포함하여 출력한다
-a	숨김 파일과 디렉토리까지 모두 출력한다
-A	숨김 파일과 디렉토리를 출력하되 . 과 .. 은 제외한다
-m	파일 목록을 쉼표로 구분하여 한 줄에 출력한다
-F	파일 형식을 기호로 표시한다 (* , @ , / 등)
-R	하위 디렉토리까지 재귀적으로 출력한다
-n	UID와 GID를 숫자로 출력한다
-i	i-node 번호를 출력한다
-h	파일 크기를 사람이 읽기 쉬운 형태로 출력한다
-t	수정 시간 순으로 정렬하여 출력한다

파일 상세 정보 구조

ls -l 명령어로 출력되는 정보의 구조는 다음과 같다:

```
(파일 유형)(권한) (링크수) (소유자) (그룹) (크기) (날짜) (시간) (파일명)
-rw-r--r--      1      root      root    1024    Dec 25   10:30  example.txt
```

파일 유형 기호

기호	파일 유형
-	일반 파일이다
d	디렉토리이다
l	심볼릭 링크이다
b	블록 장치 파일이다
c	문자 장치 파일이다 (프린터, 스캐너 등)
p	파이프 파일이다
s	소켓 파일이다

사용자 및 그룹 ID

- 루트 계정: UID 0, GID 0
- 일반 사용자: UID 1000부터 시작, GID 1000부터 시작

cp 명령어 고급 옵션

파일과 디렉토리를 복사하는 cp 명령어의 고급 옵션들이다.

주요 옵션

옵션	설명
-i	덮어쓰기 전에 사용자 확인을 요청한다
-f	확인 없이 강제로 복사한다
-r 또는 -R	디렉토리 하위 내용을 재귀적으로 복사한다
-p	원본 파일의 권한과 소유자 정보를 유지한다
-b	덮어쓸 때 백업 파일을 생성한다
-S suffix	백업 파일의 확장자를 지정한다
-u	대상 파일보다 새로운 파일만 복사한다
-v	복사 과정을 상세히 출력한다

백업 파일 생성 예제


```
# .bak 확장자로 백업 파일 생성
cp -b -S .bak passwd passwd_new

# 결과: passwd~ (기본) 또는 passwd.bak (확장자 지정)
```

file 명령어

파일의 형식과 유형을 확인하는 명령어이다.

```
# 파일 유형 확인
file filename

# 여러 파일 동시 확인
file file1 file2 file3

# 심볼릭 링크 원본 파일 확인
file -L symbolic_link
```

출력 예제

```
$ file /etc/passwd
/etc/passwd: ASCII text

$ file /bin/ls
/bin/ls: ELF 64-bit LSB executable

$ file image.jpg
image.jpg: JPEG image data
```

wc 명령어 (Word Count)

파일의 행 수, 단어 수, 문자 수를 계산하는 명령어이다.

주요 옵션

옵션	설명
-l	행(line) 수를 출력한다
-w	단어(word) 수를 출력한다
-c	바이트(character) 수를 출력한다
-m	문자(multibyte character) 수를 출력한다
-L	가장 긴 줄의 길이를 출력한다

```
[root@192 ~]# wc -l java
10 java
[root@192 ~]# wc -w java
16 java
[root@192 ~]# wc -c java
97 java
```

💡 사용 예제

전체 정보 출력 (행, 단어, 바이트 순)

wc filename

행 수만 출력

wc -l filename

여러 파일의 통계 출력

wc *.txt

파이프를 통한 사용

ls -l | wc -l # 현재 디렉토리의 파일 개수

✂ split 명령어

큰 파일을 작은 단위로 분할하는 명령어이다.

```
[root@192 ~]# split -l 5 java
[root@192 ~]# ll
합계 16
-rw-----. 1 root root 991  1월  5 01:21 anaconda-ks.cfg
drwxr-xr-x. 2 root root   6  1월  5 19:40 dir1
-rw-r--r--. 1 root root  97  1월  5 20:06 java
-rw-r--r--. 1 root root  54  1월  5 20:09 xaa
-rw-r--r--. 1 root root  43  1월  5 20:09 xab
drwxr-xr-x. 2 root root   6  1월  5 01:22 공개
drwxr-xr-x. 2 root root   6  1월  5 01:22 다운로드
drwxr-xr-x. 2 root root   6  1월  5 01:22 문서
drwxr-xr-x. 2 root root  63  1월  5 01:45 바탕화면
drwxr-xr-x. 2 root root   6  1월  5 01:22 비디오
drwxr-xr-x. 2 root root   6  1월  5 01:22 사진
drwxr-xr-x. 2 root root   6  1월  5 01:22 서식
drwxr-xr-x. 2 root root   6  1월  5 01:22 음악
```

🔧 주요 옵션

옵션	설명
<code>-l n</code>	n줄 단위로 분할한다
<code>-b n</code>	n바이트 단위로 분할한다
<code>-C n</code>	줄을 깨뜨리지 않고 n바이트 단위로 분할한다
<code>-a n</code>	접미사 길이를 n자리로 설정한다
<code>-d</code>	숫자로 접미사를 생성한다 (기본은 알파벳)
<code>--additional-suffix=suffix</code>	추가 접미사를 지정한다

💡 사용 예제

```
# 기본 분할 (1000줄 단위)
split largefile.txt

# 5줄 단위로 분할
split -l 5 filename

# 1MB 단위로 분할
split -b 1M filename

# 고급 분할 (10줄, 4자리 숫자, .txt 확장자, file_ 접두어)
split -l 10 -a 4 -d --additional-suffix=.txt passwd file_
```

분할된 파일명 예제:

- 기본: xaa , xab , xac ...
- 숫자 접미사: x00 , x01 , x02 ...
- 접두어 + 숫자 + 확장자: file_0000.txt , file_0001.txt ...

cut 명령어

파일에서 특정 필드나 문자를 추출하는 명령어이다.

주요 옵션

옵션	설명
-c n	n번째 문자를 추출한다
-c n-m	n번째부터 m번째까지 문자를 추출한다
-f n	n번째 필드를 추출한다
-d delimiter	필드 구분자를 지정한다 (기본값: 탭)
--complement	지정된 부분을 제외하고 나머지를 출력한다

```
root@192:~  
[root@192 ~]# cat -n java  
 1 This is Java file.  
 2 Hello Java!  
 3 class  
 4 Method  
 5 interface  
 6 int, String, char, double  
 7 Vector  
 8 List  
 9 Map  
10  
[root@192 ~]#  
[root@192 ~]#  
[root@192 ~]# cut -c 3 java  
i  
l  
a  
t  
t  
t  
c  
s  
p
```

```
root@192:~  
[root@192 ~]# cut -c 5-10 java  
is Ja  
o Java  
s  
od  
rface  
Strin  
or  
  
[root@192 ~]#
```

💡 사용 예제

```
# 문자 단위 추출
cut -c 10 file1.txt          # 10번째 문자
cut -c 5-10 file1.txt        # 5~10번째 문자
cut -c 5,10,15 file1.txt     # 5, 10, 15번째 문자
cut -c 5- file1.txt          # 5번째부터 끝까지

# 필드 단위 추출 (기본 구분자: 탭)
cut -f 2 file1.txt           # 2번째 필드
cut -f 2-4 file1.txt         # 2~4번째 필드
cut -f 1,3 file1.txt         # 1, 3번째 필드

# 사용자 정의 구분자 사용
cut -d ':' -f 1 /etc/passwd   # 콜론 구분자로 첫 번째 필드 (사용자명)
cut -d ',' -f 2,4 data.csv    # 쉼표 구분자로 2, 4번째 필드
```

grep 명령어

파일에서 특정 패턴을 검색하는 강력한 명령어이다.

주요 옵션

옵션	설명
-n	일치하는 줄의 번호를 함께 출력한다
-i	대소문자를 구분하지 않는다
-v	패턴과 일치하지 않는 줄을 출력한다
-c	일치하는 줄의 개수만 출력한다
-l	패턴이 포함된 파일명만 출력한다
-r	디렉토리를 재귀적으로 검색한다
-A n	일치하는 줄 뒤의 n줄도 함께 출력한다
-B n	일치하는 줄 앞의 n줄도 함께 출력한다
-C n	일치하는 줄 앞뒤 n줄을 함께 출력한다

```
[root@192 ~]# cat -n java
 1 This is Java file.
 2 Hello Java!
 3 class
 4 Method
 5 interface
 6 int, String, char, double
 7 Vector
 8 List
 9 Map
10
[root@192 ~]# grep 'Hello' java
Hello Java!
[root@192 ~]# grep -n 'This' java
1:This is Java file.
[root@192 ~]# grep '^int' java
interface
int, String, char, double
[root@192 ~]#
```

정규표현식 패턴

패턴	설명
<code>^pattern</code>	줄의 시작 부분에서 패턴을 찾는다
<code>pattern\$</code>	줄의 끝 부분에서 패턴을 찾는다
<code>.</code>	임의의 한 문자를 의미한다
<code>*</code>	앞 문자가 0개 이상 반복된다
<code>[abc]</code>	a, b, c 중 하나의 문자를 의미한다
<code>[a-z]</code>	a부터 z까지의 소문자를 의미한다

사용 예제

```
# 기본 검색
grep 'linux' file.txt          # linux가 포함된 줄 출력
grep -n 'linux' file.txt      # 줄 번호와 함께 출력
grep -i 'LINUX' file.txt      # 대소문자 구분 없이 검색

# 정규표현식 사용
grep '^linux' file.txt        # linux로 시작하는 줄
grep 'linux$' file.txt        # linux로 끝나는 줄
grep 'l.nux' file.txt         # l과 nux 사이에 한 문자가 있는 경우

# 고급 검색
grep -v 'error' log.txt       # error가 없는 줄만 출력
grep -c 'warning' log.txt     # warning이 포함된 줄 개수
grep -r 'function' /home/user/ # 디렉토리 전체에서 재귀 검색

# 컨텍스트 포함 검색
```

<code>grep -A 3 'error' log.txt</code>	# error 다음 3줄까지 출력
<code>grep -B 2 'error' log.txt</code>	# error 이전 2줄까지 출력
<code>grep -C 2 'error' log.txt</code>	# error 앞뒤 2줄씩 출력

마무리

Vi 편집기와 고급 리눅스 명령어들은 시스템 관리와 파일 처리에서 매우 중요한 도구들이다. Vi 편집기의 모드 개념을 이해하고, 각 명령어의 다양한 옵션을 활용하면 더욱 효율적인 작업이 가능하다.

학습 팁


1. **Vi 편집기**: 처음에는 어려울 수 있지만, 반복 연습을 통해 익숙해지면 매우 강력한 도구가 된다.
2. **명령어 옵션**: `man` 명령어를 사용하여 각 명령어의 전체 옵션을 확인할 수 있다.
3. **파이프 활용**: 여러 명령어를 파이프(`|`)로 연결하여 복합적인 작업을 수행할 수 있다.

예: 특정 사용자의 프로세스 개수 확인

```
ps aux | grep username | wc -l
```

예: 로그 파일에서 에러 라인만 추출하여 파일로 저장

```
grep -i error /var/log/syslog | cut -d ' ' -f 1-3 > error_summary.txt
```

 **참고**: 실제 운영 환경에서는 중요한 파일을 수정하기 전에 항상 백업을 생성하는 것이 좋다.