



Chapter 3. 파일 및 디렉토리 관련 명령어

3.1 명령어 구조

3.2 도움말 이용(man page)

3.3 파일 및 디렉토리 작업을 위한 명령어

3.4 디스플레이 명령어



명령어 구조

❖ 명령어(command)

- 유닉스 시스템이 무언가를 하도록 지시하는 프로그램
- 형식 : 명령어 [옵션][인자]
 - 인자(argument)란 명령어가 동작을 취할 대상(예: 파일)을 의미
 - 옵션(option)은 명령의 내용을 여러 가지 방식으로 다르게 적용시킬 수 있도록 함

❖ 유닉스 시스템에서 명령어는 대소문자를 구분하여 인식

- 예: `command`와 `Command`는 다르다.

❖ 일반적으로 옵션은 하이픈(-)이 앞에 오며, 대부분의 명령어에서 하나 이상의 옵션이 함께 사용 가능

- 명령어 `-[옵션][옵션][옵션]`
 - 예를 들어, `cecom%ls -alR` 은 현재 디렉터리는 물론 모든 서브 디렉터리까지 순환적으로 모든 파일에 대한 상세정보를 보여 줌



명령어 구조

- ❖ 사용자가 한 명령어를 입력하면 셸은 그 입력한 명령어가 내부 명령어인지 확인하고, 아니라면 PATH 환경변수에서 지정된 경로에서 실행파일을 찾게 됨
- ❖ 사용자 본인의 PATH 정보를 다음과 같이 확인할 수 있음
 - `cecom% echo $PATH`

3



도움말 이용(man page)

- ❖ 유닉스 매뉴얼(man 페이지라고 함)
 - 온라인으로 유닉스 시스템과 명령어 사용법을 설명
 - man 페이지를 사용하려면 프롬프트에서 man을 친 후 설명을 원하는 명령어를 타이핑

명령 구조:

`man [옵션] 명령어`

옵션 설명:

- k 키워드: 키워드와 매칭된 명령어 요약 줄을 열거한다
- M 패스 : man 페이지에 대한 패스를 보여준다
- a : 매칭되는 모든 man 페이지를 보여준다

4

도움말 이용 예

- ❖ 괄호 안의 번호는 man 페이지에서 이들이 찾아진 절을 의미

```

cecom% man chmod
페이지를 다시 포맷 중입니다. 기다려 주십시오... 완료
User Commands                               chmod(1)

NAME ← 명령어 이름과 목적을 설명
      chmod - change the permissions mode of a file

SYNOPSIS ← 명령어의 일반 형식을 설명
      chmod [ -fR ] <absolute-mode> file ...
      chmod [ -fR ] <symbolic-mode-list> file ...

--계속--(5%) ← 전체내용 중 5%만 출력되었다는 의미(종료하기 위해 ^c를 입력)
<중략 ...>

SEE ALSO
      ls(1), chmod(2), attributes(5), environ(5), largefile(5),
      getfacl(1), setfacl(1)
    
```

그림 3-3. chmod에 대한 man 사용 예

도움말 이용

표 3-1. man 페이지 구성 항목

항 목	의 미
NAME	해당 명령어에 대한 이름과 사용 목적을 간단히 설명
SYNOPSIS	해당 명령어에 대한 일반적 사용 형식을 나타냄
DESCRIPTION	해당 명령어에 대한 자세한 설명
FILES	해당 명령어가 사용하는 파일을 나타냄
SEE ALSO	해당 명령어에 대한 보다 더 많은 정보를 얻기 위해 참조해야 할 부분을 나타냄
DIAGNOSTICS	예상할 수 있는 에러에 대한 설명과 명령어가 실행했을 때 되돌려주는 에러코드 목록을 나타냄

파일 및 디렉토리 작업을 위한 명령어

❖ 디렉토리 내용 보기 : ls (list contents of directory)

- 유닉스 사용자가 가장 자주 사용하는 명령
 - `ls` 명령은 사용자의 디렉터리와 파일의 목록을 보여 줌. 옵션을 이용하면 파일의 형식, 크기, 접근권한, 파일생성 날짜, 수정날짜 등에 대한 정보를 볼 수 있음

명령 구조:

`ls [옵션] [인자]`

옵션 설명:

- a 모든 파일을 열거. 히든 파일(.으로 시작하는 파일)도 함께 열거
- d 디렉터리명만 열거
- F 끝 부분에 항목의 유형을 표시
디렉터리 /
소켓 =
심볼릭 링크 @
실행가능 파일 *
- L 파일이 심볼릭 링크이면 그 링크가 참조하는 파일 혹은 디렉토리에 대한 정보를 보여 줌
- l 상세 정보를 보여 줌

7

파일 및 디렉토리 작업을 위한 명령어

❖ 인자(argument)를 사용하지 않는 경우

- 현재 디렉터리의 목록을 보여 줌
- 인자에는 파일 혹은 디렉터리 등이 올 수 있고, 메타문자(*, ?)를 이용한 매칭 이용 가능

```
cecom% ls
Test file1 file2 result
cecom% ls fil* ← 메타문자 적용
file1 file2
cecom% ls file?
file1 file2
```

그림 3-4. ls 명령어의 메타문자 적용한 예

8

파일 및 디렉토리 작업을 위한 명령어

- ❖ `ls` 명령의 `-l` 옵션을 이용하면 파일 및 디렉토리에 대한 모드 필드 10개(`-rw-r--r--`)의 문자를 확인할 수 있음
 - 첫 번째 문자는 다음 기호중의 하나가 된다
 - 문자 항목 유형

d	디렉터리	c	문자 유형의 특수 파일
-	일반 파일	l	심볼릭 링크
b	블록 유형의 특수 파일	s	소켓
- ❖ 나머지 9개의 문자는 3문자씩 3개의 세트로 구성
 - 파일 접근권한을 의미: 처음 3문자는 사용자 자신(user)의 접근권한을 나타내고, 두 번째 3문자는 그룹(group)에 대한 접근권한을, 마지막 3문자는 시스템 상의 기타 다른 사용자(other)들의 접근권한을 의미
- ❖ 각 문자에 대한 의미는 다음과 같다
 - r 읽기(read) 권한
 - w 쓰기(write) 권한
 - x 실행(execute) 권한
 - - 접근 불가

9

파일 및 디렉토리 작업을 위한 명령어

- ❖ 현재의 작업 디렉토리 확인 : `pwd`(return working directory name)
 - `pwd` 명령은 언제든지 파일시스템의 계층구조 상에서 자신이 어디에 있는지를 알려준다.

```
cecom% pwd
/home/redfox
```

- ❖ 디렉토리 이동(변경) : `cd` (change working directory)
 - `cd` 명령은 디렉토리를 변경. `cd` 명령은 절대경로명과 상대경로명 모두 사용할 수 있다.
 - 상대경로 : 현재 자신이 사용하고 있는 디렉토리부터 시작
 - 절대경로 : 루트(/) 디렉토리부터 시작
 - 명령 구조:
 - `cd [디렉터리]`

10

파일 및 디렉토리 작업을 위한 명령어

❖ *cd*의 사용 예

- *cd* 사용자의 홈 디렉터리로 변경
- *cd /* 시스템의 루트 디렉터리로 변경
- *cd ..* 한 단계 상위 디렉터리로 변경
- *cd ../..* 두 단계 상위 디렉터리로 변경
- *cd /full/path/name/from/root* 지정된 절대경로명으로 변경
 - (반드시 슬래시(/)가 먼저 나와야 한다)
- *cd path/from/current/location* 현재 위치에서 지정된 상대경로명으로 변경
 - (선행 슬래시(/)가 없다)
- *cd ~사용자명/디렉터리*
 - 지정된 사용자명의 해당 디렉터리로 변경
 - HOME 변수 지정

11

파일 및 디렉토리 작업을 위한 명령어

❖ 상대/절대 경로를 이용한 디렉토리 이동

```
cecom% pwd ← 현재 디렉토리 위치 확인
/home/redfox
cecom% cd /home/redfox/Test1 ← 절대경로 지정으로 Test1 디렉터리로 이동
cecom% pwd ← 이동된 디렉토리 위치 확인
/home/redfox/Test1
cecom%
```

그림 3-10. 절대경로를 이용한 디렉토리 이동

```
cecom% pwd
/home/redfox/Test1
cecom% cd ../Test2
cecom% pwd
/home/redfox/Test2
cecom% cd ← 인자 없이 단순히 cd만 치면 홈 디렉터리로 이동
```

그림 3-11. 상대경로를 이용한 디렉토리 이동

12

파일 및 디렉토리 작업을 위한 명령어

❖ 디렉토리 만들기 : mkdir (make directory)

- 사용자는 자신의 홈 디렉터리 아래에 서브 디렉터리를 생성하여 계층 디렉터리 구조 생성 가능
- 디렉터리의 절대경로명 혹은 상대경로명을 지정 가능

명령 구조:

mkdir [옵션] 디렉터리

옵션 설명:

- p 지정된 디렉터리에서 중간의 디렉터리도 같이 생성
- m mode 접근권한을 지정된 mode로 부여해서 디렉토리를 생성

```
cecom% ls
Test1 Test2 file1.c file2
cecom% mkdir Unix1
cecom% ls
Test1 Test2 Unix1 file1.c file2
cecom% mkdir /home/redfox/Unix2
cecom% ls
Test1 Test2 Unix1 Unix2 file1.c file2
```

그림 3-13. 상대, 절대경로명을 이용한 디렉토리 생성

파일 및 디렉토리 작업을 위한 명령어

```
cecom% mkdir -p Unix_a/Unix_aa
cecom% ls
Test1 Test2 Unix1 Unix2 Unix_a file1.c file2
cecom% ls -l
총 14
drwxr-xr-x 3 redfox student 512 10월 5일 16:56 Test1
drwxr-xr-x 2 redfox student 512 10월 19일 16:45 Test2
<중략...>
drwxr-xr-x 3 redfox student 512 10월 19일 17:29 Unix_a
-rwxr-xr-x 1 redfox student 26 10월 4일 19:28 file1.c
-rw-r--r-- 1 redfox student 26 10월 19일 01:29 file2
cecom% mkdir -m 700 Unix3
cecom% ls -l
총 16
drwxr-xr-x 3 redfox student 512 10월 5일 16:56 Test1
drwx----- 2 redfox student 512 10월 19일 17:41 Unix3
drwxr-xr-x 3 redfox student 512 10월 19일 17:29 Unix_a
-rwxr-xr-x 1 redfox student 26 10월 4일 19:28 file1.c
-rw-r--r-- 1 redfox student 26 10월 19일 01:29 file2
```

그림 3-14. mkdir의 "-p", "-m" 옵션 적용한 예

파일 및 디렉토리 작업을 위한 명령어

❖ 디렉토리 제거 : rmdir (remove directory)

- 사용자가 어떤 디렉토리를 제거하려면 먼저 그 디렉토리의 상위 디렉토리 내용부터 제거해야 하고, 또한 현재 위치가 제거하려는 디렉토리에 있다면 그 디렉토리를 제거 불가
- 명령 구조 : rmdir [옵션] 디렉토리
- 또 다른 방법으로 *rm* 명령(다음절에서 설명)을 사용하여 제거 가능
 - 이 명령을 사용하면 디렉토리가 비어있지 않더라도 디렉토리를 제거할 수 있다.
- 현재의 위치가 /home1/gildong에 있을 때 비어있는 디렉토리 /home1/gildong/data를 제거하고자 한다면
 - [/home1/gildong]% rmdir data 혹은
 - [/home1/gildong]% rmdir /home1/gildong/data

15

파일 및 디렉토리 작업을 위한 명령어

❖ 파일 삭제하기 : rm (remove a file)

- 파일을 삭제하는 명령어

명령 구조:

rm [옵션] 파일명

옵션 설명:

- i 파일 삭제 전에 사용자에게 확인을 요구함
- r 순환적으로 디렉토리를 제거. 먼저 파일을 제거하고 서브 디렉토리를 제거
- f 확인을 하지 않고 삭제

```
cecom% ls
Test1 Test2 Unix1 Unix2 Unix_a file1.c file2 file3
cecom% rm file1.c
cecom% ls
Test1 Test2 Unix1 Unix2 Unix_a file2 file3
cecom% rm file2 file3
cecom% ls
Test1 Test2 Unix1 Unix2 Unix_a
```

16

파일 및 디렉토리 작업을 위한 명령어

```
cecom% ls
Test1 Test2 Unix1 Unix2 Unix_a file1 file2
cecom% rm -i file1 file2
rm: file1(y/n)을(를) 제거합니까? y
rm: file2(y/n)을(를) 제거합니까? n
cecom% ls
Test1 Test2 Unix1 Unix2 Unix_a file2
```

그림 3-18. "-i" 옵션 적용한 예

```
cecom% ls -l file2
-r--r--r-- 1 redfox student 9 10월 20일 16:35 file2
cecom% rm file2
rm: file2: 덮어쓰기 방지 444 (y/n)? n
cecom% rm -f file2
cecom%
```

그림 3-19. "-f" 옵션 적용한 예

```
cecom% ls
Test1 Test2 Unix1 Unix2 Unix_a
cecom% rm -r Test1
cecom% ls
Test2 Unix1 Unix2 Unix_a
```

그림 3-20. "-r" 옵션 적용한 예

17

파일 및 디렉토리 작업을 위한 명령어

❖ 파일 복사하기 : cp (copy files)

- 특정 파일을 다른 이름의 파일로 복사하는 명령어
- 새로운 파일이 이미 존재하더라도 그냥 복사됨

명령 구조:

```
cp [옵션] 원본파일 새로운파일
cp [옵션] 원본파일 디렉토리
cp -r 디렉토리1 디렉토리2
```

옵션 설명:

```
-i 대화방식으로 진행한다
(확인을 기다려서 요청에 따라 진행한다)
-r 디렉터리를 순환적으로 복사한다
(서브 디렉터리도 모든 내용과 함께 복사한다)
```

18

파일 및 디렉토리 작업을 위한 명령어

```
cecom% ls
Test2 Unix1 Unix2 Unix_a file1 file2
cecom% ls Test2
file1
cecom% cp Test2 Test3
cp: Test2: 은(는) 디렉토리입니다
cecom% cp -r Test2 Test3
cecom% ls
Test2 Test3 Unix1 Unix2 Unix_a file1 file2
cecom% ls Test3
file1
cecom% ls
Test2 Test3 Unix1 Unix2 Unix_a file1 file2
cecom% cp -i file1 file2
cp: 겹쳐쓰기 file2 (y/n)?
```

그림 3-22. cp 명령어의 "-i", "-r" 옵션 사용 예

19

파일 및 디렉토리 작업을 위한 명령어

❖ cp 명령어와 inode와의 관계

- 각 파일에 대한 정보를 갖고 있는 inode는 파일당 하나씩 존재
- cp 로 인해 원본과 같은 파일이 새롭게 생성되므로 새로운 파일에 대한 inode가 생성됨을 확인할 수 있음

```
cecom% ls
Test2 Test3 Unix1 Unix2 Unix_a file1 file2
cecom% ls -i
 101385 Test2      190113 Unix1      221762 Unix_a    190125 file2
 253456 Test3     196441 Unix2      190124 file1
cecom% cp file1 file3
cecom% ls -i
 101385 Test2      190113 Unix1      221762 Unix_a    190125 file2
 253456 Test3     196441 Unix2      190124 file1    190128 file3
cecom% cp file1 ./Test2
cecom% ls -i ./Test2
101396 file1
```

그림 3-23. cp 명령어와 아이노드와의 관계

20

파일 및 디렉토리 작업을 위한 명령어

❖ 파일 이름 바꾸기 및 이동하기 : mv (move files)

- 파일 및 디렉토리의 이름을 변경(이전 파일은 존재하지 않음)
- 또한 지정된 파일 혹은 디렉터리가 현재의 경로가 아니라면 이것은 그 파일 혹은 디렉터리를 다른 장소로 옮겨주는 효과를 가짐.

명령 구조:

```
mv [옵션] 파일1 파일2
mv [옵션] 파일 디렉토리
mv [옵션] 디렉토리1 디렉토리2
```

옵션 설명:

- i 새로운 파일 이름이 이미 존재하는 경우 확인을 시켜줌
- f 목적지에 같은 이름의 파일이 존재해도 묻지 않고 수행

- 이 명령은 파일의 내용은 있던 곳에 그대로 두고 디렉터리 테이블의 항목만 변경

21

파일 및 디렉토리 작업을 위한 명령어

```
[/home1/gildong]% ls
logfile1      public_html/  result2      unix2/
logfile1.new  result1       result3      unix3/
[/home1/gildong]% mv logfile1 logfile2
[/home1/gildong]% ls
logfile1.new  public_html/  result2      unix2/
logfile2      result1       result3      unix3/
[/home1/gildong]% mv unix3 unix
[/home1/gildong]% ls
logfile1.new  public_html/  result2      unix2/
logfile2      result1       result3      unix/
[/home1/gildong]% mv unix public html
[/home1/gildong]% ls public_html
data/         images/       index.html
[/home1/gildong]
```

디렉토리의 이름을 변경한다.

현재의 위치가 아니면
마치 디렉토리 전체를
옮긴 것과 같다!

22

파일 및 디렉토리 작업을 위한 명령어

❖ mv명령어와 inode와의 관계

- 다른 이름으로 복사한 후 기존 파일은 없어짐
- 실제 파일의 내용들을 이동하는 것이 아니라 계층적인 구조에 따라 이름만 이동시킴(inode 번호는 변하지 않음)

23

파일 및 디렉토리 작업을 위한 명령어

```
cecom% ls
TTT Test2 Test3 Unix1 Unix2 Unix_a fff file1 file2 file3
cecom% ls -i
 354840 TTT      190113 Unix1    190129 fff      190128 file3
 101385 Test2   196441 Unix2    190124 file1
 253456 Test3   221762 Unix_a   190125 file2
cecom% mv file1 file4
cecom% ls -i
 354840 TTT      190113 Unix1    190129 fff      190124 file4
 101385 Test2   196441 Unix2    190125 file2
 253456 Test3   221762 Unix_a   190128 file3
cecom% mv Unix1 Unix3
cecom% ls -i
 354840 TTT      196441 Unix2    190129 fff      190124 file4
 101385 Test2   190113 Unix3    190125 file2
 253456 Test3   221762 Unix_a   190128 file3
cecom% mv file2 Unix2
cecom% ls -i Unix2
190125 file2
```

그림 3-24. mv 명령어와 아이노드와의 관계

24

파일 및 디렉토리 작업을 위한 명령어

❖ 파일 찾기 : find (find files)

- 시스템 내에 존재하는 파일, 디렉토리, 특수파일들 중에서 원하는 파일을 찾을 때 사용하는 명령어
- 명령구조
 - find 경로명 옵션
 - 경로명은 루트 디렉토리(/)부터 시작하는 절대경로로 표시할 수 있고, './'으로 현재 디렉토리를 표시하는 상대경로로도 가능함

파일 및 디렉토리 작업을 위한 명령어

옵 션	의 미
-name 파일명	지정한 파일명이 찾은 파일명과 일치하면 출력된다. 이때 파일명으로 메타문자([, ?, *)와 함께 사용할 수 있다.
-type 파일형	지정한 파일형이 찾은 파일명과 일치하면 출력된다. 이때 사용되는 파일형은 다음과 같다. b : 블록 특수파일 c : 문자 특수파일 d : 디렉토리 p : 파이프인 fifo 파일 f : 일반파일 e : 연결파일
-user 로그인명	지정한 사용자 ID가 찾은 사용자 ID와 일치하면 출력한다.
-size 수	파일의 크기를 이용해서 찾는다. 파일 크기는 블록단위(1블록 = 512바이트)이다. 지정한 파일 크기보다 작은 파일을 찾으려면 파일 크기 앞에 - 기호를 붙이고 큰 파일을 찾으려면 + 기호를 붙인다.
-print	찾은 파일명을 화면에 출력한다.
-atime 수	24시간 중 지정한 시간에 접근(access)된 적이 있는 파일을 찾는다.
-mtime 수	24시간 중 지정한 시간에 변경된 파일을 찾는다. (0을 지정하면 지난 24시간 동안을 의미한다.)
-exec 명령어	명령어를 실행한다. 이 명령어는 ₩; 으로 끝을 맺으며 명령어 인수 {}는 현재의 경로명으로 대체한다.

파일 및 디렉토리 작업을 위한 명령어

```
cecom% find /etc -name passwd -print ← ①
/etc/default/passwd
/etc/passwd
cecom% find . -name file3 -print ← ②
./file3
cecom% pwd
/home/redfox
cecom% find /home/redfox -name file3 -print ← ③
/home/redfox/file3
cecom% find /etc -size +10 -print ← ④
/etc/default/lu
/etc/dhcp/inittab
/etc/fs/hsfs/mount
<이하생략...>
cecom% find . -size -10 -print ← ⑤
./file3
./fff
./TTT
./file4
<이하생략...>
```

그림 3-26. find 명령어 사용 예

27

파일 및 디렉토리 작업을 위한 명령어

```
cecom% find . -name 'file*' -exec ls -l {} \; ← ⑥
-rw-r--r-- 1 redfox student 9 10월 20일 18:39 ./Test3/file1
-rw-r--r-- 1 redfox student 9 10월 20일 18:51 ./Test2/file1
-rw-r--r-- 1 redfox student 9 10월 20일 18:28 ./Unix2/file2
-rw-r--r-- 1 redfox student 9 10월 20일 18:51 ./file3
-rw-r--r-- 1 redfox student 9 10월 20일 18:28 ./file4
cecom% find . -type d -print ← ⑦
./Unix3
./Test3
./Test2
<이하생략>
cecom% find . -mtime 0 -print ← ⑧
.
./Unix2
```

그림 3-26. find 명령어 사용 예

28



디스플레이 명령어

- ❖ 파일의 내용을 보는데 사용할 수 있는 명령은 많다.
- ❖ 물론 부록에 설명되어 있는 vi 편집기를 사용하여 파일의 내용을 보거나 편집할 수 있다.
- ❖ 여기서는 일반적으로 파일의 내용을 볼 수 있는 명령들에 대해 알아본다.

29



디스플레이 명령어

- ❖ **echo**
 - *echo* 명령은 사용자가 입력한 내용을 그대로 표준출력 장치에 다시 보여주는 명령. 일반적으로 라인피드(line-feed)에 의해 종료하지만, 이것을 변경할 수 있는 옵션은 많다.

명령 구조:

```
echo [텍스트_문자열]
```

옵션 설명:

-n	<new-line>을 프린트하지 않는다(BSD)
\c	<new-line>을 프린트하지 않는다(SVR4)
\On	n: 8-비트 ASCII 문자 코드(SVR4)
\t	탭(SVR4)
\f	폼피드(SVR4)
\n	새로운 줄(SVR4)
\v	수직 탭(SVR4)

30

디스플레이 명령어

❖ cat

- 파일의 내용을 보여주거나 파일들을 연결할 때 사용

명령 구조:

cat [옵션] 파일

옵션 설명:

- n 각 줄에 줄 번호를 붙여준다
- v 탭, 새로운 줄, 폼 피드를 제외하고 프린트할 수 없는 문자를 보여줌
- e 각 줄의 끝에 \$ 기호를 보여준다(-v와 함께 사용할 때)

31

디스플레이 명령어

```
cecom% ls
TTT  Test2  Test3  Unix2  Unix3  Unix_a  fff  file3  file4
cecom% cat file3
자료구조      데이터통신
자바          그리드컴퓨팅
병렬처리     시스템프로그래밍
임베디드     영상처리
cecom% cat -n file3
 1 자료구조      데이터통신
 2 자바          그리드컴퓨팅
 3 병렬처리     시스템프로그래밍
 4 임베디드     영상처리
cecom% cat -v file3
자료구조      데이터통신
자바          그리드컴퓨팅
병렬처리     시스템프로그래밍
임베디드     영상처리
cecom% cat -ve file3
자료구조      데이터통신$
자바          그리드컴퓨팅$
병렬처리     시스템프로그래밍$
임베디드     영상처리$
```

그림 3-28. cat 명령어 사용 예

32

디스플레이 명령어

❖ more, less, pg – page through a file

- *more, less, pg* 명령은 한번에 한 페이지씩 파일의 내용을 보여준다. (유닉스 시스템에 따라 지원하지 않는 명령이 있을 수 있다.)

명령 구조:

```
more [옵션][+/pattern][파일명]
less [옵션][+/pattern][파일명]
pg [옵션][+/pattern][파일명]
```

➢ 이들 명령은 한 화면을 보여주고 나서 화면 하단에 프롬프트를 보인다

- *more*의 경우 --계속--(6%)
- *pg*의 경우 :

more	page	pg	의 미
-c	default	-c	파일 내용을 보여주기 전에 화면을 깨끗이 지운다.
-w	-w	default	입력을 마쳤을 때 빠져나가지 않고 기다린다.
-lines	-lines	-lines	한 화면에 보여줄 줄 수를 지정한다.(첫번째 줄부터 지정된 lines 값 만큼 보여짐)

표 3-8 more/page, pg 에서 사용되는 옵션 33

```
[ce:/home1/gildong/unix2]% pg curriculum
Data Structures[2/1] Artificial Intelligence[3/1]
Internet programming[3/1] Operating System[3/2]
Software Engineering[3/2] Object-Oriented Programming[2/2]
Assembly Language[2/1] Digital Engineering[2/2]
Computer Network[4/1] Visual Programming[3/1]
(파일끝):h
```

```
-----
h          도움말
q 또는 Q   끝
<blank> 또는 <newline> 다음 페이지
l          다음 행
d 또는 <^D> 페이지의 반 출력
. 또는 <^L> 현재 페이지 다시 출력
f          다음 페이지로 스킵
n          다음 파일
p          이전 파일
$          지난 페이지
w 또는 z   윈도우 크기 지정, 다음 페이지 출력
s 보관파일 보관 파일에 현재 파일 보관
/패턴/    패턴을 앞에서 찾기
?패턴? 또는 ^패턴^ 패턴을 뒤에서 찾기
!명령어   명령어 수행
```

모든 명령어는 숫자로 시작, 즉:

```
+1<newline> (next page); -1<newline> (previous page); 1<newline> (page 1).
```

자세한 사항은 매뉴얼 참고.

```
-----
:q
```

디스플레이 명령어

- ❖ **head** – 파일의 시작부분을 보여준다
 - *head* 명령은 파일의 시작부분을 보여준다.

명령 구조:

head [옵션] 파일

옵션 설명:

-number 파일의 처음 n줄을 표시함

```
cecom% head file3
자료구조        데이터통신
자바            그리드컴퓨팅
병렬처리        시스템프로그래밍
임베디드        영상처리
cecom% head -2 file3
자료구조        데이터통신
자바            그리드컴퓨팅
```

그림 3-32. head 명령어 사용 예

디스플레이 명령어

- ❖ **tail**
 - 파일의 끝부분으로부터 n개의 줄을 화면에 보여 줌

명령 구조:

tail [옵션] 파일

옵션 설명:

-number 파일의 끝에서부터 n개의 줄을 화면에 표시함

```
cecom% tail file3
자료구조        데이터통신
자바            그리드컴퓨팅
병렬처리        시스템프로그래밍
임베디드        영상처리
cecom% tail -2 file3
병렬처리        시스템프로그래밍
임베디드        영상처리
```

그림 3-33. tail 명령어 사용 예