원리를 알면 IT가 맛있다
IT COOKBOOK

IT CookBook

CentOS 리눅스

시스템 & 네트워크

[강의교안 이용 안내]

CentOS 리눅스

시스템 & 네트워크

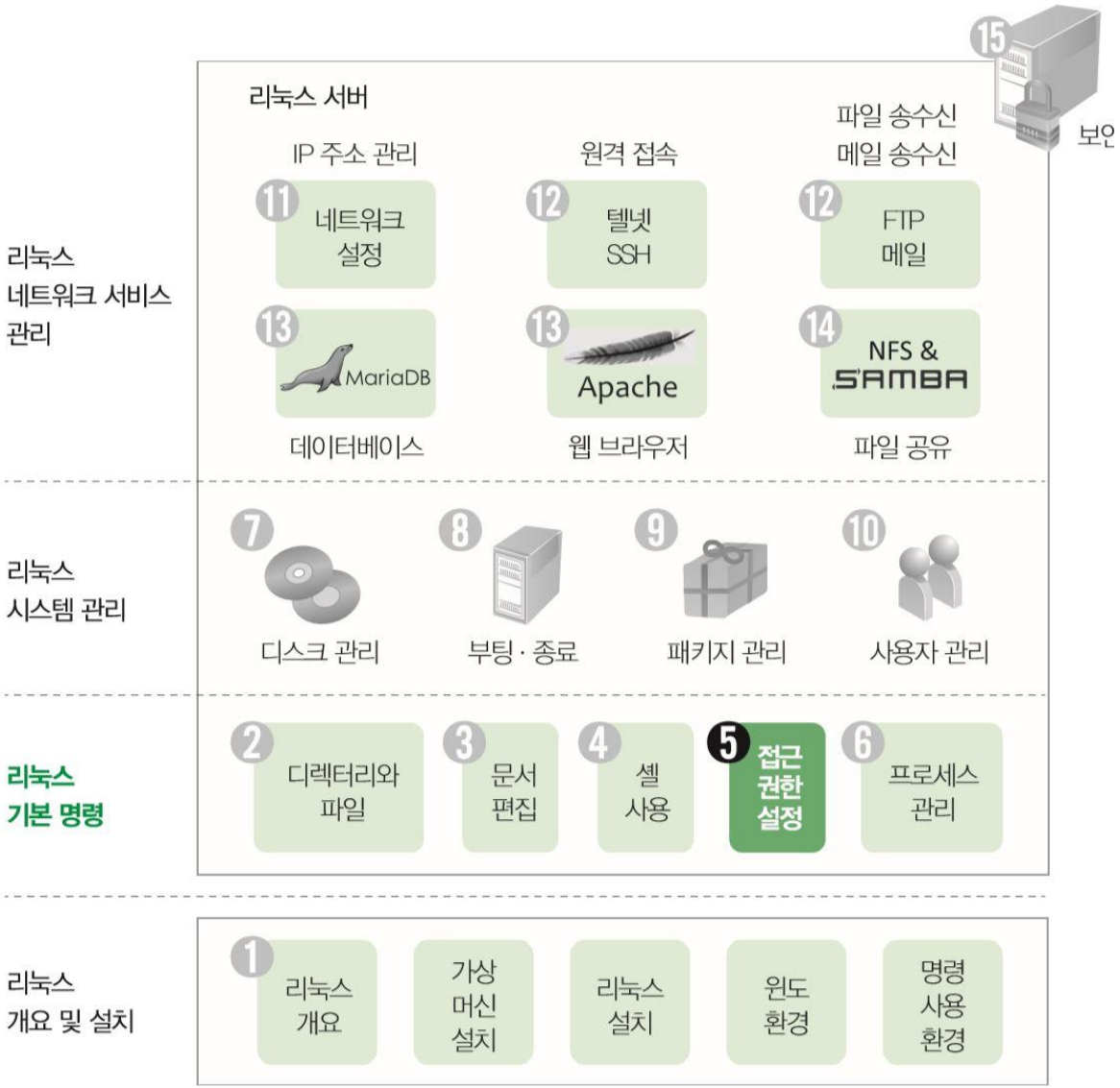
Chapter 05. 파일 접근 권한 관리

목차

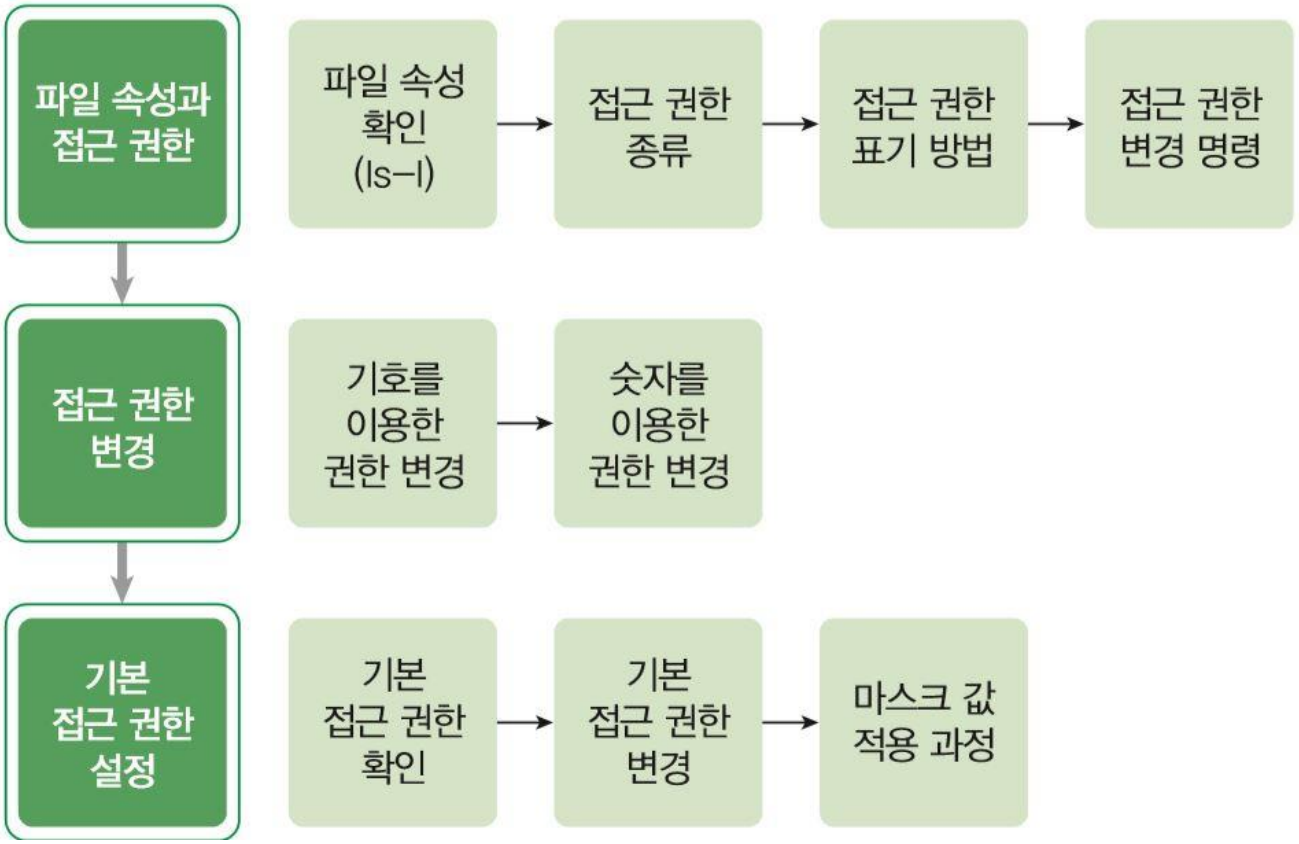
학습목표

- 파일의 속성과 접근 권한의 개념을 이해할 수 있다.
- 접근 권한의 종류와 표기 방법을 이해하고 설명할 수 있다.
- 기호와 숫자로 표기된 접근 권한을 바꿀 수 있다.
- 기본 접근 권한을 확인하고 원하는 값으로 바꿀 수 있다.

00.개요


15
보안
리눅스 서버
IP 주소 관리
원격 접속
파일 송수신
메일 송수신
11 네트워크 설정
12 텔넷 SSH
12 FTP 메일
13 MariaDB
13 Apache
14 NFS & SAMBA
데이터베이스
웹 브라우저
파일 공유
리눅스 네트워크 서비스 관리
7 디스크 관리
8 부팅 · 종료
9 패키지 관리
10 사용자 관리
리눅스 시스템 관리
2 디렉터리와 파일
3 문서 편집
4 셸 사용
5 접근 권한 설정
6 프로세스 관리
리눅스 기본 명령
1 리눅스 개요
가상 머신 설치
리눅스 설치
윈도 환경
명령 사용 환경
리눅스 개요 및 설치

00.개요


파일 속성과 접근 권한
파일 속성 확인 (ls-l)
접근 권한 종류
접근 권한 표기 방법
접근 권한 변경 명령
접근 권한 변경
기호를 이용한 권한 변경
숫자를 이용한 권한 변경
기본 접근 권한 설정
기본 접근 권한 확인
기본 접근 권한 변경
마스크 값 적용 과정

01.파일 속성

파일 접근 권한 보호

- 리눅스는 파일에 무단으로 접근하는 것을 방지하고 보호하는 기능을 제공
- 사용자는 자신의 파일과 디렉터리 중에서 다른 사용자가 접근해도 되는 것과 그렇지 않은 것을 구분하여 접근 권한을 제한

파일 속성

```
[user1@localhost ~]$ ls -l /etc/hosts
-rw-r--r--. 1 root root 158  9월 10  2018 /etc/hosts
```

표 5-1 파일의 속성

번호	속성 값	의미
❶	–	파일의 종류(–: 일반 파일, d: 디렉터리)
❷	rw– r– –r– –	파일을 읽고 쓰고 실행할 수 있는 접근 권한 표시
❸	1	하드 링크의 개수
❹	root	파일 소유자의 로그인 ID
❺	root	파일이 속한 그룹 이름
❻	158	파일의 크기(바이트 단위)
❼	9월 10 2018	파일이 마지막으로 수정된 날짜
❽	/etc/hosts	파일명

01.파일 속성

파일 속성

파일의 종류

- 파일 속성의 첫 번째 항목은 파일의 종류를 표시
- -는 일반 파일을, d는 디렉터리를 의미
- 파일의 종류를 알려주는 명령

file

- **기능** 지정한 파일의 종류를 알려준다.
- **형식** file [파일명]
- **사용 예** file /etc/services

- ex) file 명령의 사용

```
[user1@localhost ~]$ file /etc/hosts /usr/bin
/etc/hosts: ASCII text
/usr/bin:   directory
```

- /etc/hosts는 일반 텍스트 파일 이고 /usr/bin은 디렉터리임

파일의 접근 권한 표시

- 파일의 소유자와 그룹이나 기타 사용자들이 파일에 대해 가지고 있는 접근 권한을 표시

하드 링크의 개수

- 하드 링크는 한 파일에 대해 여러 개의 파일명을 가질 수 있도록 하는 기능

01.파일 속성

파일 속성

- 파일 소유자의 로그인 ID
  - 리눅스에서 모든 파일은 소유자가 있음
- 파일 소유자의 그룹 이름
  - ls -l 명령에서 출력되는 그룹명은 파일이 속한 그룹
  - 사용자가 속한 기본 그룹은 시스템 관리자가 사용자를 등록할 때 결정
  - 사용자가 속한 그룹을 알려주는 명령은 groups

groups

- **기능** 사용자가 속한 그룹을 알려준다.
- **형식** groups [사용자명]

  - ex) groups 명령의 사용

```
[user1@localhost ~]$ groups
user1
[user1@localhost ~]$ groups root
root : root
```

- 파일의 크기
  - 파일의 크기를 바이트 단위로 알림
- 파일이 마지막으로 수정된 날짜
  - 파일이 마지막으로 수정된 날짜와 시간이 표시. 연도가 표시되지 않으면 올해를 의미

02.파일 접근 권한

파일 접근 권한

- 접근 권한 : 해당 파일을 읽고 쓰고 실행할 수 있는 권한, 사용자의 파일을 보호하는 가장 기본적인 보안 기능
- 리눅스는 사용자를 파일 소유자, 파일이 속한 그룹, 그 외 기타 사용자 세 카테고리로 구분하여 접근 권한을 적용
- 사용자 카테고리별로 접근 권한을 다르게 부여하여 파일을 보호할 수 있음

접근 권한 종류

- 읽기 권한, 쓰기 권한, 실행 권한 등 세 가지로 구성
- 대상이 파일인지 디렉터리인지에 따라 그 의미가 약간 다르게 해석

표 5-2 파일과 디렉터리의 접근 권한

권한	파일	디렉터리
읽기	파일을 읽거나 복사할 수 있다.	ls 명령으로 디렉터리 목록을 볼 수 있다(ls 명령의 옵션은 실행 권한이 있어야 사용할 수 있다).
쓰기	파일을 수정 · 이동 · 삭제할 수 있다(디렉터리에 쓰기 권한이 있어야 한다).	파일을 생성하거나 삭제할 수 있다.
실행	파일을 실행할 수 있다(셸 스크립트나 실행 파일의 경우).	cd 명령을 사용할 수 있다. 파일을 디렉터리로 이동하거나 복사할 수 있다.

02.파일 접근 권한

접근 권한 표기 방법

- 사용자 카테고리별로 누가 파일을 읽고 쓰고 실행할 수 있는지를 문자로 표현한 것
- 읽기 권한은 r, 쓰기 권한은 w, 실행 권한은 x로 나타내며, 해당 권한이 없는 경우에는 -로 표기
- 사용자 카테고리별로 세 가지 권한의 부여 여부를 rwx 세 문자를 묶어서 표기

```
[user1@localhost ~]$ ls -l /etc/hosts
-rw-r--r--. 1 root root 158  9월 10  2018 /etc/hosts
```

- 예시의 실제 접근 권한: rw-r--r–
  - (소유자가 읽기와 쓰기 권한을 가지고 있고 그룹과 기타 사용자는 읽기 권한만 가지고 있음을 나타냄)

그림 5-1 파일의 접근 권한 표기

표 5-3 다양한 접근 권한 조합의 예

접근 권한	의미
rwxr–xr–x	소유자는 읽기 · 쓰기 · 실행 권한을 모두 가지고 그룹과 기타 사용자는 읽기 · 실행 권한을 가지고 있다.
r–xr–xr–x	소유자, 그룹, 기타 사용자 모두 읽기 · 실행 권한을 가지고 있다.
rw–––––––	소유자만 읽기 · 쓰기 권한을 가지고 그룹과 기타 사용자는 아무 권한이 없다.
rw–rw–rw–	소유자, 그룹, 기타 사용자 모두 읽기 · 쓰기 권한을 가지고 있다.
rwxrwxrwx	소유자, 그룹, 기타 사용자 모두 읽기 · 쓰기 · 실행 권한을 가지고 있다.
rwx–––––––	소유자만 읽기 · 쓰기 · 실행 권한을 가지고 그룹과 기타 사용자는 아무 권한이 없다.
r––––––––	소유자만 읽기 권한을 가지고 있다.

02.파일 접근 권한

접근 권한 변경 명령

- 접근 권한을 바꾸려면 chmod 명령을 사용

chmod

- **기능** 파일이나 디렉터리의 접근 권한을 변경한다.
- **형식** chmod [옵션] 권한 파일(디렉터리)
- **옵션** -R: 하위 디렉터리까지 모두 변경할 수 있다.

- chmod 명령으로 접근 권한을 변경할 때 기호 모드와 숫자 모드를 사용할 수 있음
  - 기호 모드: 접근 권한을 변경하기 위해 문자와 기호를 사용하여 권한을 표시
  - 숫자 모드: 접근 권한을 변경하기 위해 숫자를 사용

03.기호를 이용한 파일 접근 권한 변경

기호를 이용한 파일 접근 권한 변경

- 기호 모드 : 기호를 이용하여 파일 접근 권한을 변경하
- 사용자 카테고리 문자, 연산자 기호, 접근 권한 문자로 구성

그림 5-2 기호 모드의 구성 요소

- 사용자 카테고리: 소유자, 그룹, 기타 사용자를 나타내는 문자로 표기
- 연산자: 권한 부여나 제거를 나타 내는 기호로 표기
- 접근 권한 기호: 읽기, 쓰기, 실행을 나타내는 문자를 사용

표 5-4 기호 모드에서 사용하는 문자와 기호

구분	문자/기호	의미
사용자 카테고리 문자	u	파일 소유자
	g	파일 소유 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

표 5-5 기호 모드를 사용한 접근 권한 설정의 예

권한 표기	의미
u+w	소유자(u)에게 쓰기(w) 권한 부여(+)
u−x	소유자(u)의 실행(x) 권한 제거(−)
g+w	그룹(g)에 쓰기(w) 권한 부여(+)
o−r	기타 사용자(o)의 읽기(r) 권한 제거(−)
g+wx	그룹(g)에 쓰기(w)와 실행(x) 권한 부여(+)
+wx	모든 사용자에게 쓰기(w)와 실행(x) 권한 부여(+)
a+rwx	모든 사용자에게 읽기(r), 쓰기(w), 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 기타 사용자(o)에게 쓰기(w) 권한 부여(+)

03.기호를 이용한 파일 접근 권한 변경

기호를 이용한 파일 접근 권한 변경

- ch5 디렉터리와 파일을 준비

```
[user1@localhost ~]$ mkdir linux_ex/ch5
[user1@localhost ~]$ cd linux_ex/ch5
[user1@localhost ch5]$ cp /etc/hosts test.txt
```

- 현재 접근 권한을 확인 (현재 접근 권한은 rw-r--r--)

```
[user1@localhost ch5]$ ls -l
합계 4
-rw-r--r--. 1 user1 user1 158 11월  2 21:26 test.txt
```

- 소유자의 쓰기 권한을 제거 (u-w)

```
[user1@localhost ch5]$ chmod u-w test.txt
[user1@localhost ch5]$ ls -l
합계 4
-r--r--r--. 1 user1 user1 158 11월  2 21:26 test.txt
```

03.기호를 이용한 파일 접근 권한 변경

[따라해보기] 기호 모드로 접근 권한 변경하기

① 그룹에 쓰기와 실행 권한을 부여 (g+wx)

```
[user1@localhost ch5]$ chmod g+wx test.txt
[user1@localhost ch5]$ ls -l
합계 4
-r--rwxr--. 1 user1 user1 158 11월  2 21:26 test.txt*
```

② 기타 사용자에게 실행 권한을 부여 (o+x)

```
[user1@localhost ch5]$ chmod o+x test.txt
[user1@localhost ch5]$ ls -l
합계 4
-r--rwxr-x. 1 user1 user1 158 11월  2 21:26 test.txt*
```

③ 그룹과 기타 사용자의 실행 권한을 제거 (go-x)

```
[user1@localhost ch5]$ chmod go-x test.txt
[user1@localhost ch5]$ ls -l
합계 4
-r--rw-r--. 1 user1 user1 158 11월  2 21:26 test.txt
```

03.기호를 이용한 파일 접근 권한 변경

[따라해보기] 기호 모드로 접근 권한 변경하기

④ 모두에게 실행 권한을 부여 (a+x)

```
[user1@localhost ch5]$ chmod a+x test.txt
[user1@localhost ch5]$ ls -l
합계 4
-r-xrwxr-x. 1 user1 user1 158 11월  2 21:26 test.txt*
```

⑤ 소유자에게 쓰기 권한을 부여하고 그룹의 쓰기 권한은 제거 (u+w,g-w)

```
[user1@localhost ch5]$ chmod u+w,g-w test.txt
[user1@localhost ch5]$ ls -l
합계 4
-rwxr-xr-x. 1 user1 user1 158 11월  2 21:26 test.txt*
```

04.숫자를 이용한 파일 접근 권한 변경

숫자를 이용한 파일 접근 권한 변경

- 기호 모드는 전체적으로 권한을 조정할 때는 문자의 조합이 복잡해짐
- 이럴 때 숫자 모드로 chmod 명령을 사용하면 권한을 한 번에 원하는 대로 변경할 수 있어서 매우 편리

숫자로 환산하는 방법

- 숫자 모드에서는 각 권한이 있고 없고를 0과 1로 표기하고 이를 다시 환산하여 숫자로 표현
- 카테고리별로 권한의 조합에 따라 0부터 7로 나타내는 것

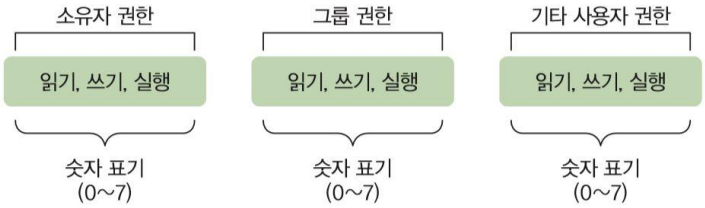


그림 5-3 숫자 모드의 구성 요소

04.숫자를 이용한 파일 접근 권한 변경

■ 숫자로 환산하는 방법

- 권한 묶음(rwx)을 숫자로 환산하는 방법
  - ① r-x라고 할 때 권한이 있는 것은 1로, 없는 것은 0으로 변환
  - ② 2진수 1, 0, 1로 변환
  - ③ 2진수를 각 자릿수별로 10진수로 환산하면 4, 0, 1이 됨
  - ④ 세 숫자를 더하면
  - ⑤ 최종 권한 값은 5가 됨

표 5-6 접근 권한과 숫자의 대응 관계

접근 권한	환산	숫자	의미
rwx	111 → 4+2+1	7	읽기, 쓰기, 실행
rw-	110 → 4+2+0	6	읽기, 쓰기
r-x	101 → 4+0+1	5	읽기, 실행
r--	100 → 4+0+0	4	읽기
-wx	011 → 0+2+1	3	쓰기, 실행
-w-	010 → 0+2+0	2	쓰기
--x	001 → 0+0+1	1	실행
---	000 → 0+0+0	0	권한이 없음

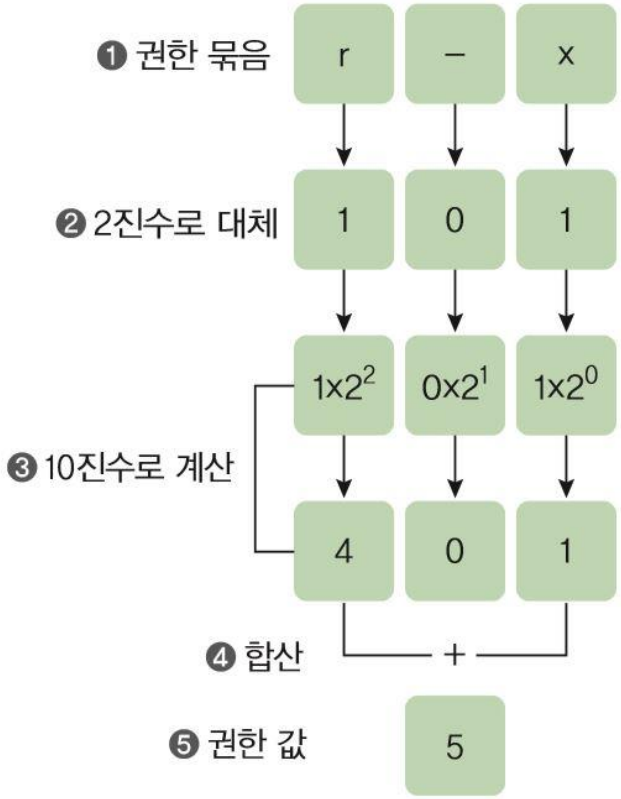


그림 5-4 권한을 숫자로 환산하는 과정

04.숫자를 이용한 파일 접근 권한 변경

숫자로 환산하는 방법

- 소유자 권한, 그룹 권한, 기타 사용자 권한을 각각 숫자로 환산하여 전체 접근 권한을 나타낼 수 있음

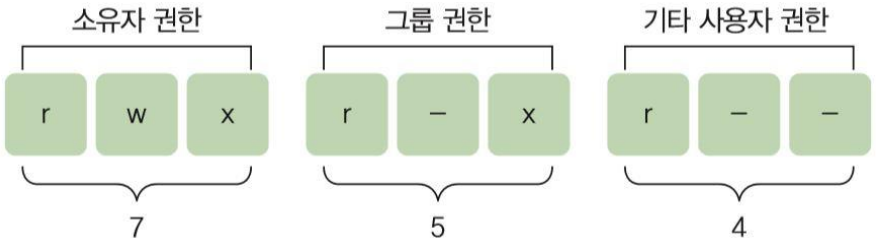


그림 5-5 전체 접근 권한을 숫자로 표기한 예

표 5-7 숫자로 표현한 접근 권한의 예

접근 권한	숫자 모드	접근 권한	숫자 모드
rwxrwxrwx	777	rw-r--r--	644
rwxr-xr-x	755	rwx------	700
rw-rw-rw	666	rw-r-----	640
r-xr-xr-x	555	r--------	400

04.숫자를 이용한 파일 접근 권한 변경

숫자 모드를 이용한 접근 권한 변경

그림 5-6 숫자 모드를 이용한 접근 권한 변경

- 숫자의 각 위치가 사용자 카테고리를 나타내기 때문에 사용자 카테고리를 따로 지정할 필요가 없음
- 항상 세 자리 수를 사용해야 하므로 변경하려는 사용자 카테고리의 권한 뿐만 아니라 그룹과 기타 사용자의 권한도 반드시 같이 명시

04.숫자를 이용한 파일 접근 권한 변경

숫자 모드를 이용한 접근 권한 변경

- ex) test.txt 파일의 현재 접근 권한은 다음과 같이 644(rw-r--r--)

```
[user1@localhost ch5]$ ls -l
합계 4
-rw-r--r--. 1 user1 user1 158 11월  2 21:26 test.txt
```

- 소유자의 쓰기 권한을 제거: 기호로는 u-w이지만 이를 숫자로 표기하면 r--r--r--이므로 444

```
[user1@localhost ch5]$ chmod 444 test.txt
[user1@localhost ch5]$ ls -l
합계 4
-r--r--r--. 1 user1 user1 158 11월  2 21:26 test.txt
```

- 그룹에 쓰기와 실행 권한을 부여: r--rwxr--이므로 이를 숫자로 표기하면 474

```
[user1@localhost ch5]$ chmod 474 test.txt
[user1@localhost ch5]$ ls -l
합계 4
-r--rwxr--. 1 user1 user1 158 11월  2 21:26 test.txt*
```

04.숫자를 이용한 파일 접근 권한 변경

[따라해보기] 숫자 모드로 접근 권한 변경하기

① 기타 사용자에게 실행 권한을 부여 (o+x) r—rwxr-x → 475

```
[user1@localhost ch5]$ chmod 475 test.txt
[user1@localhost ch5]$ ls -l
합계 4
-r--rwxr-x. 1 user1 user1 158 11월  2 21:26 test.txt*
```

② 그룹과 기타 사용자의 실행 권한을 제거 (go-x) r—rw-r-- → 464

```
[user1@localhost ch5]$ chmod 464 test.txt
[user1@localhost ch5]$ ls -l
합계 4
-r--rw-r--. 1 user1 user1 158 11월  2 21:26 test.txt
```

③ 모두에게 실행 권한을 부여 (a+x) r-xrwxr-x → 575

```
[user1@localhost ch5]$ chmod 575 test.txt
[user1@localhost ch5]$ ls -l
합계 4
-r-xrwxr-x. 1 user1 user1 158 11월  2 21:26 test.txt*
```

04.숫자를 이용한 파일 접근 권한 변경

- **[따라해보기] 숫자 모드로 접근 권한 변경하기**

④ 소유자에게 쓰기 권한을 부여하고 그룹의 쓰기 권한은 제거 (u+w,g-w) rwxr-xr-x → 755

```
[user1@localhost ch5]$ chmod 755 test.txt
[user1@localhost ch5]$ ls -l
합계 4
-rwxr-xr-x. 1 user1 user1 158 11월  2 21:26 test.txt*
```

⑤ 소유자의 권한만 남기고 나머지 사용자의 권한은 모두 제거 rwx------ → 700

```
[user1@localhost ch5]$ chmod 700 test.txt
[user1@localhost ch5]$ ls -l
합계 4
-rwx------. 1 user1 user1 158 11월  2 21:26 test.txt*
```

05.기본 접근 권한 설정

기본 접근 권한

- 리눅스에서는 파일이나 디렉터리를 생성할 때 기본 접근 권한이 자동적으로 설정
- 일반 파일의 경우 소유자와 그룹은 읽기와 쓰기 권한이 설정되고 기타 사용자는 읽기 권한만 설정
- 디렉터리의 경우 소유자와 그룹은 읽기, 쓰기, 실행 권한이 설정되고 기타 사용자는 읽기, 실행 권한만 설정

- 다음 예에서 새로 만든 centos.txt 파일과 temp 디렉터리의 접근 권한을 보면 기본 접근 권한으로 설정

```
[user1@localhost ch5]$ touch centos.txt
[user1@localhost ch5]$ mkdir temp
[user1@localhost ch5]$ ls -l
합계 4
-rw-rw-r--. 1 user1 user1   0 11월  2 21:49 centos.txt
drwxrwxr-x. 2 user1 user1   6 11월  2 21:49 temp/
----------.  1 user1 user1 158 11월  2 21:26 test.txt
```

05.기본 접근 권한 설정

기본 접근 권한 확인 및 변경

- umask 명령 : 기본 접근 권한을 확인하고 설정하는 데 사용

umask

- **기능** 기본 접근 권한을 출력하거나 변경한다.
- **형식** umask [옵션] [마스크 값]
- **옵션** -S: 마스크 값을 문자로 출력한다.
- **사용 예** umask 022
  umask

- 예시에서 0002에서 맨 앞의 0은 다른 용도가 있으며, 기본 접근 권한은 002로 적용

```
[user1@localhost ch5]$ umask
0002
```

마스크 값의 의미

- 마스크 값 : 파일이나 디렉터리 생성 시 부여하지 않을 권한을 지정해놓는 것
- 마스크 값이 002일 경우 이는 -------w-이고, 기타 사용자에게 쓰기 권한은 부여하지 않겠다는 뜻
- ex) : umask 명령에 -S 옵션을 적용하여 마스크 값을 문자로 출력

```
[user1@localhost ch5]$ umask -S
u=rwx, g=rwx, o=rx
```

  - 'u=rwx, g=rwx, o=rx': 사용자와 그룹에는 읽기, 쓰기, 실행 권한을 부여,기타 사용자의 경우에는 읽기와 실행 권한만 부여

05.기본 접근 권한 설정

기본 접근 권한 확인 및 변경

- umask 명령 : 기본 접근 권한을 확인하고 설정하는 데 사용

umask

- **기능** 기본 접근 권한을 출력하거나 변경한다.
- **형식** umask [옵션] [마스크 값]
- **옵션** -S: 마스크 값을 문자로 출력한다.
- **사용 예** umask 022
  umask

- 예시에서 0002에서 맨 앞의 0은 다른 용도가 있으며, 기본 접근 권한은 002로 적용

```
[user1@localhost ch5]$ umask
0002
```

마스크 값의 의미

- 마스크 값 : 파일이나 디렉터리 생성 시 부여하지 않을 권한을 지정해놓는 것
- 마스크 값이 002일 경우 이는 -------w-이고, 기타 사용자에게 쓰기 권한은 부여하지 않겠다는 뜻
- ex) : umask 명령에 -S 옵션을 적용하여 마스크 값을 문자로 출력

```
[user1@localhost ch5]$ umask -S
u=rwx, g=rwx, o=rx
```

  - 'u=rwx, g=rwx, o=rx': 사용자와 그룹에는 읽기, 쓰기, 실행 권한을 부여,기타 사용자의 경우에는 읽기와 실행 권한만 부여

05.기본 접근 권한 설정

기본 접근 권한 확인 및 변경

마스크 값 변경하기

- umask 명령으로 마스크 값을 변경

```
[user1@localhost ch5]$ umask 077
[user1@localhost ch5]$ umask
0077
```

- umask로 마스크 값을 바꾸면 파일이나 디렉터리를 생성할 때 적용되는 기본 접근 권한도 바뀜

```
[user1@localhost ch5]$ touch linux.txt
[user1@localhost ch5]$ ls -l
합계 4
-rw-rw-r--.  1 user1 user1   0 11월  2 21:49 centos.txt
-rw-------.  1 user1 user1   0 11월  2 21:56 linux.txt
drwxrwxr-x. 2 user1 user1   6 11월  2 21:49 temp/
----------.  1 user1 user1 158 11월  2 21:26 test.txt
```

05.기본 접근 권한 설정

마스크 값의 적용 과정

- 마스크 값을 비트로 표시했을 때 그 값이 1인 경우 대응하는 권한은 제외

표 5-8 umask 진리표

요청 권한	1	1	0	0
마스크	1	0	1	0
부여된 권한	0	1	0	0

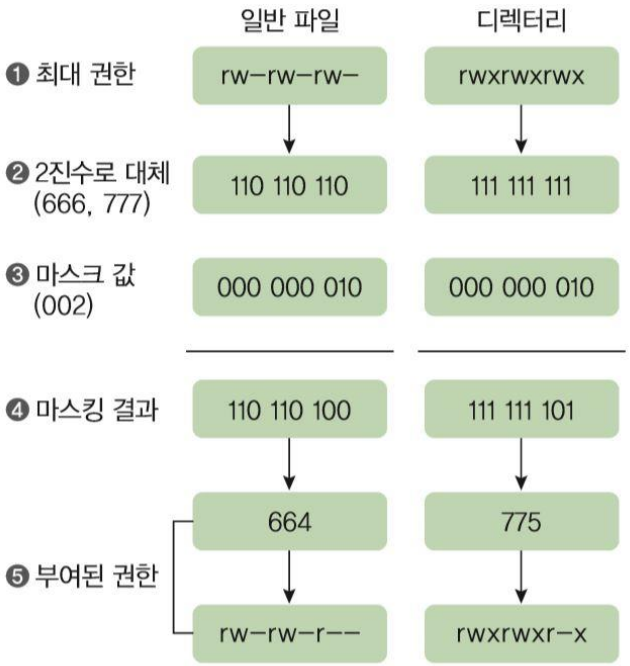
- 마스크 값이 1이면 요청 권한이 1인지 0인지에 상관없이 부여되는 권한은 0이 됨
- 마스크 값이 0일 경우 해당 요청 권한이 그대로 부여됨
- 리눅스 시스템은 파일이나 디렉터리를 생성할 때 부여하려는 접근 권한과 마스크 값을 진리표를 기준으로 마스킹하여 기본 접근 권한을 결정

05.기본 접근 권한 설정

■ 마스크 값의 적용 과정

▪ 마스크 값 계산 방법

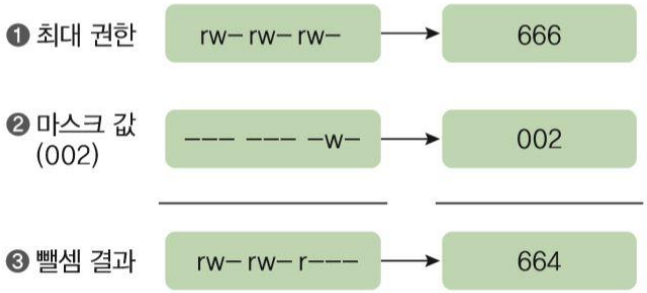
- 일반 파일이 가질 수 있는 최대 접근 권한은 666 (rw-rw-rw)
- 현재 마스크 값이 002일 때 666과 002를 [표 5-8]에 따라 마스킹하면 결과가 664임을 알 수 있음
- 따라서 마스크 값이 002일 때 일반 파일을 생성하면 기본 접근 권한이 664



(a) 마스크 값을 적용하는 과정

표 5-8 umask 진리표

요청 권한	1	1	0	0
마스크	1	0	1	0
부여된 권한	0	1	0	0



(b) 마스크 값에 뺄셈을 적용하는 과정

그림 5-7 마스크 값 계산 방법

05.기본 접근 권한 설정

■ 마스크 값의 적용 과정

▪ 여러가지 마스크 값

- 리눅스의 기본 마스크 값은 002인데 이를 필요에 따라 적절히 변경하여 자신의 파일과 디렉터리를 보호할 수 있음

표 5-9 마스크 값의 의미

마스크 값	일반 파일	디렉터리	의미
022	644	755	그룹과 기타 사용자는 읽기만 가능하다.
077	600	700	그룹과 기타 사용자의 접근 권한을 모두 제거한다.
027	640	750	그룹은 읽기와 실행만 가능하고 기타 사용자의 접근 권한을 모두 제거한다.

- 마스크 값이 파일 에는 적합하지만 디렉터리에는 적합하지 않을 수도 있기 때문에 umask로 마스크 값을 바꿀 때 파일과 디렉터리에 모두 적용해봐야 함

- ex) : 마스크 값이 026일 경우, 파일은 640으로 적합하지만 디렉터리는 751이 되어 기타 사용자의 접근 권한이 이상

05.기본 접근 권한 설정

[따라해보기] 기본 접근 권한 변경하기

① 현재의 기본 접근 권한을 확인

```
[user1@localhost ch5]$ umask
0077
```

② 그룹과 기타 사용자가 읽기와 실행을 할 수 없도록 기본 접근 권한을 변경

```
[user1@localhost ch5]$ umask 022
[user1@localhost ch5]$ umask
0022
```

③ 파일을 생성하여 기본 접근 권한이 변경되었는지 확인

```
[user1@localhost ch5]$ touch mask.txt
[user1@localhost ch5]$ ls -l mask.txt
-rw-r--r--. 1 user1 user1 0 11월  2 22:02 mask.txt
```

06.특수 접근 권한 설정

특수 접근 권한

- 접근 권한은 원래 4자리 숫자

```
[user1@localhost ch5]$ umask
0002
```

- 생략된 맨 앞자리는 특수 접근 권한 의미
- 맨 앞자리 숫자가 0이면 일반적인 접근 권한이지만 이 숫자가 1, 2, 4이면 특수 접근 권한이 설정

- SetUID : 맨 앞자리가 4
- SetGID : 맨 앞자리가 2
- 스티키 비트(sticky bit) : 맨 앞자리가 1

06.특수 접근 권한 설정

SetUID

- SetUID가 설정된 파일을 실행하면 해당 파일이 실행되는 동안에는 파일을 실행한 사용자의 권한이 아니라 파일 소유자의 권한이 적용
- ex) set.ext 파일을 만들고 실행 권한을 부여

```
[user1@localhost ch5]$ touch set.exe
[user1@localhost ch5]$ chmod 755 set.exe        ➜ 실행 권한을 부여한다.
[user1@localhost ch5]$ ls -l set.exe
-rwxr-xr-x. 1 user1 user1 0 11월  2 22:09 set.exe*
```

- ex) set.exe 파일 앞에 4755로 수정하면 다음과 같이 SetUID가 설정

```
[user1@localhost ch5]$ chmod 4755 set.exe
[user1@localhost ch5]$ ls -l set.exe
-rwsr-xr-x. 1 user1 user1 0 11월  2 22:09 set.exe*
```

- SetUID가 설정되면 소유자의 실행 권한에 's'가 표시
  - set.exe 파일을 실행하면 항상 user1의 권한을 가진다는 의미

- ex) passwd 명령

```
[user1@localhost ch5]$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 34928  5월 12 00:14 /usr/bin/passwd*
```

- passwd명령은 사용자 계정의 암호를 바꾸는 명령이나 root계정으로만 수정이 가능
- 그러나 passwd 명령에는 SetUID가 설정되어 있기 때문에 소유자인 root 권한으로 실행되어 변경 가능

06.특수 접근 권한 설정

SetGID

- SetGID는 SetUID와 거의 동일
- SetGID가 설정된 파일을 실행하면 해당 파일이 실행 되는 동안에는 파일 소유 그룹의 권한으로 실행
- SetGID는 접근 권한의 맨 앞자리에 2를 설정해야 함

- set.exe 파일에 SetGID를 설정하면 다음과 같이 그룹의 실행 권한에 ‘s’가 표시

```
[user1@localhost ch5]$ chmod 2755 set.exe
[user1@localhost ch5]$ ls -l set.exe
-rwxr-sr-x. 1 user1 user1   0 11월  2 22:09 set.exe*
```

스티키 비트

- 스티키 비트는 디렉터리에 설정
- 디렉터리에 스티키 비트가 설정되어 있으면 이 디렉터리에는 누구나 파일을 생성할 수 있음
- 파일은 파일을 생성한 계정으로 소유자가 설정되며, 다른 사용자가 생성한 파일은 삭제 불가
- /tmp 디렉터리가 대표적
- 스티키 비트는 접근 권한에서 맨 앞자리에 1을 설정

```
[user1@localhost ch5]$ chmod 1755 temp
[user1@localhost ch5]$ ls -ld temp
drwxr-xr-t. 2 user1 user1 6 11월  2 21:49 temp/
```

Thank you