redhat CentOS

ENTERPRISE LINUX ADMIN GUIDE

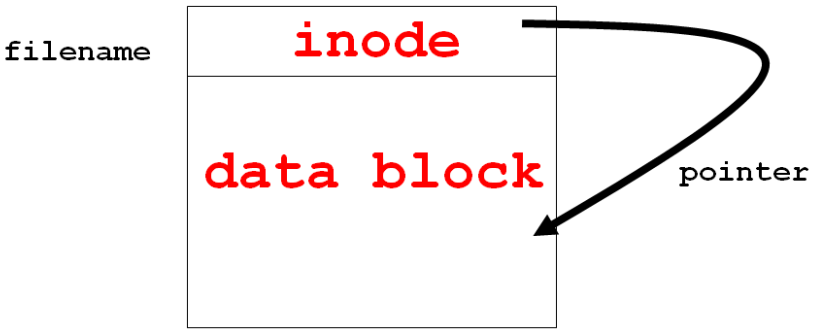
파일종류

단원목표

- 파일의 구조
- 일반 파일(Odinary File)
- 디렉토리 파일(Directory File)
- 링크파일(Link File)
  - 하드 링크
  - 소프트 링크
- 디바이스 파일(Device File)

파일의 구조

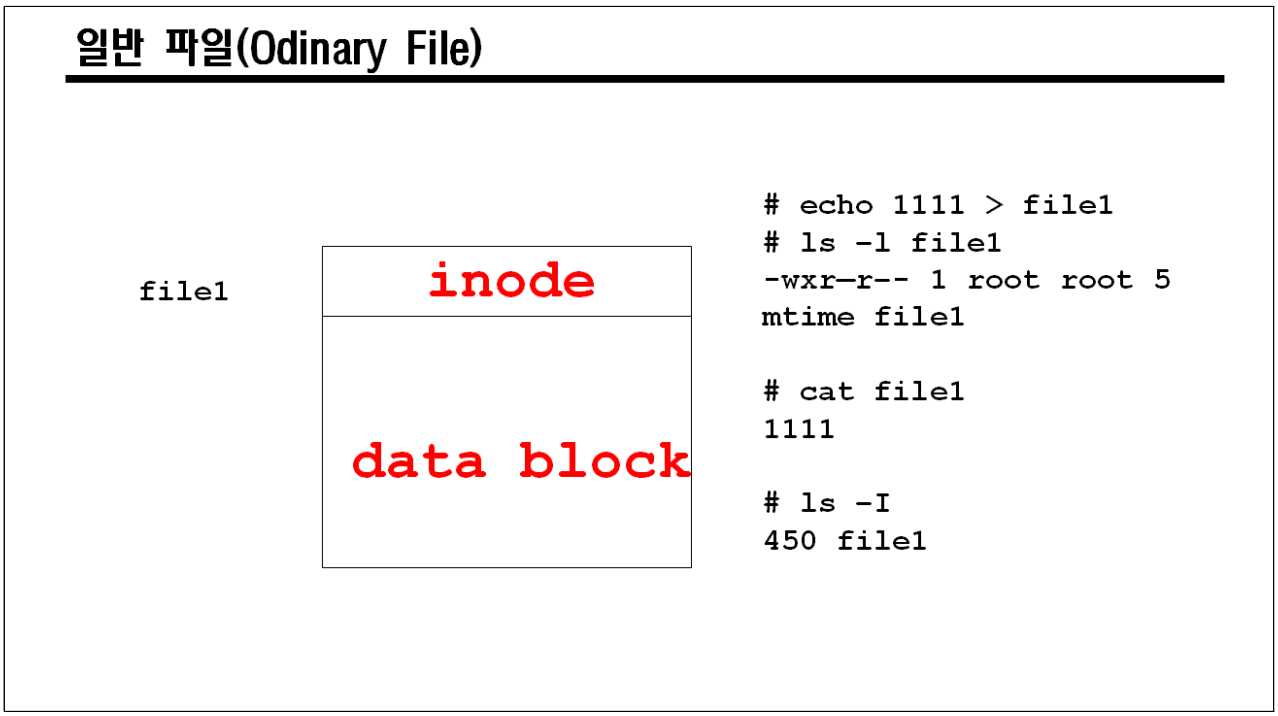
- 모든 파일은 다음과 같은 기본 구조를 가지고 있다.



■ 파일의 종류
- ■ 일반 파일(Egular File)
- ■ 디렉토리 파일(Directory File)
- ■ 링크 파일(Link File)
- ■ 장치 파일(Device File)
- ■ 소켓 파일(Socket File)
- ■ 도어 파일(Door File)
- ■ 파이프 파일(Pipe File)

파일과 디렉토리의 내용들을 다루기 위해서는 위와 같은 기본적인 명령어에 익숙해져야 한다. 리눅스 시스템에서 파일의 구조(File Structure) 파일이름, Inode (Index Node)와 데이터 블럭(Data block)로 구분할 수 있다. 파일의 이름은 유일해야 하고 대소문자는 다른 문자로 인식하게 된다. Inode는 파일의 속성정보와 데이터 블럭을 포인트하는 정보가 들어 있으며 ls 명령어에 -l 옵션을 사용하여 대부분의 정보를 확인 할 수 있다. 데이터 블록안에는 실제 파일에 저장되는 내용이 들어가게 된다. 일반파일, 디렉토리, 심볼릭링크 파일들은 모두 하나이상의 데이터 종류를 저장한다. 하지만 디바이스 파일은 데이터를 저장하지 않는다. 대신에 디바이스 파일은 디바이스 제어접근(Access)을 제공한다.

일반 파일(Odinary File)



리눅스에서는 모든 것을 파일로 다룬다. 일반 파일, 디렉토리 파일, 링크 파일등이 있다. 파일의 종류에 관한 정보는 ls 명령어의 -l 옵션을 사용하여 출력되는 화면에 첫 번째 문자에서 확인이 가능하다. 또한 file 명령어에 파일의 이름을 인자로 받아서 파일의 종류를 확인 할 수도 있다.

문자	파일 종류
-	일반 파일(Egular File)
d	디렉토리 파일(Directory File)
b	블록 디바이스 파일 (Block Device File, (예) /dev/sha, /dev/hda, /dev/fd0)
c	문자 디바이스 파일 (Character Device File, 입출력 장치)
l	심볼릭 링크(Symbolic Link File)

[참고] 자주 사용되는 기본 명령어
• 디렉토리에 관련된 기본 명령어: ls, pwd, mkdir, cd, mv 등 • 파일과 관련된 기본 명령어: cat, more, cp, rm, head, tail 등 • 파일과 디렉토리 검색에 관련한 기본 명령어: grep, find 등

(1) 파일의 이름

리눅스 시스템에서 파일의 이름은 파일을 접근하기 위해 사용한다. 파일은 Inode와 함께 파일의 구성요소로서 같은 디렉토리 안에서 파일의 이름은 유일한 것이어야 한다. 예를 들어 현재 디렉토리 밑에 존재하는 디렉토리와 같은 이름은 일반파일을 생성 할 수는 없다.

파일의 이름 생성 시 규칙
• 파일의 이름은 최대 255자까지 생성이 가능하다. • 파일의 이름은 문자와 숫자를 사용 할 수 있으며 대소문자를 구별한다. • 파일의 이름으로 .(Dot), _(Underbar), -(Dash) 사용가능하다. • 파일의 이름으로 특수문자(Metacharater)를 사용하면 안된다. 대표적인 특수문자는 *(Aterisk), /(Slash), W(Back Slash), ,(Comma), '(Single Quote), "(Double Quote) (Semicolon), &(Ampersand), |(Pipeline), <(Angle Blocket)등이다. • ".(Dot)"로 시작하는 파일의 이름은 시스템의 환경파일들이므로 일반파일의 이름으로 권장하지 않는다. • 파일의 이름에 공백이나 탭을 사용하는 것을 권장하지 않는다.

(2) Inode

Inode는 파일에 대한 정보를 담고 있는 부분이다. 일반적으로 Inode에는 크게 두가지 부분을 포함하고 있다. 첫 번째는 파일에 대한 속성정보(Ownership, Groupship, File Permission Mode등)와 데이터 블록을 가리키고 있는 포인터(Direct/Indirect Pointer)이다. Inode는 숫자로 되어져 있으며 각 파일 시스템은 자신의 Inode 테이블(Inode Tables)을 가지고 있다. 파일이 새로운 파일시스템이 만들어지는 경우 파일시스템 안에서는 새로운 Inode 번호을 할당 받게 된다.

[참고] Inode의 정보
• 파일의 종류(File Type) • 파일의 퍼미션모드(File Permission Mode) • 파일의 소유자, 그룹(File Ownership, Groupship) • 파일의 링크수(Hard Link Count) • 파일의 마지막 접근 시간(Access Time), 수정시간(Modification Time) • 파일의 크기(Bytes, 할당된 또는 사용중인 데이터블럭의 수) • 두가지 형태의 포인터(Directe Pointers and Indirect Pointers)

(3) Data Block

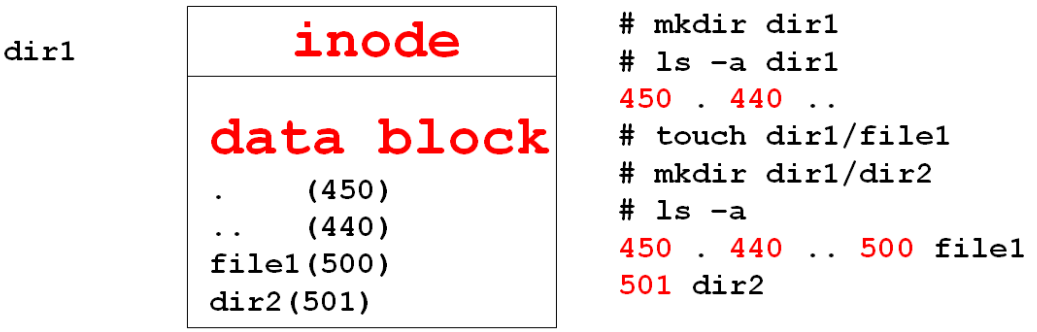
데이터블럭은 디스크 공간에 대한 단위(Units of Disk Space)로서 데이터를 저장하는 역할을 가진다. 일반파일, 디렉토리, 심볼릭 파일들은 데이터 블록을 사용하지만 일반파일과 다른 구조를 가지고 있는 디바이스 파일은 데이터 블럭에 데이터를 저장하지 않고 주 디바이스 숫자(Major Device Number)와 부 디바이스 숫자(Minor Device Number)를 담고 있다.

[참고] Data Block의 내용
• 일반파일의 경우 파일의 내용이 들어 있다. • 디렉토리인 경우 안에 포함된 파일과 디렉토리이름이 들어 있다.

일반파일은 리눅스에서 찾을 수 있는 거의 대부분의 파일종류이다. 데이터 블록에 들어가는 데이터는 많은 형태는 ASCII (American Standard Code for Information Interchange) 텍스트, 바이너리 데이터, 이미지 데이터, 데이터베이스 데이터, 애플리케이션 데이터 등이 있다. 일반파일을 만드는 방법 또한 많다.

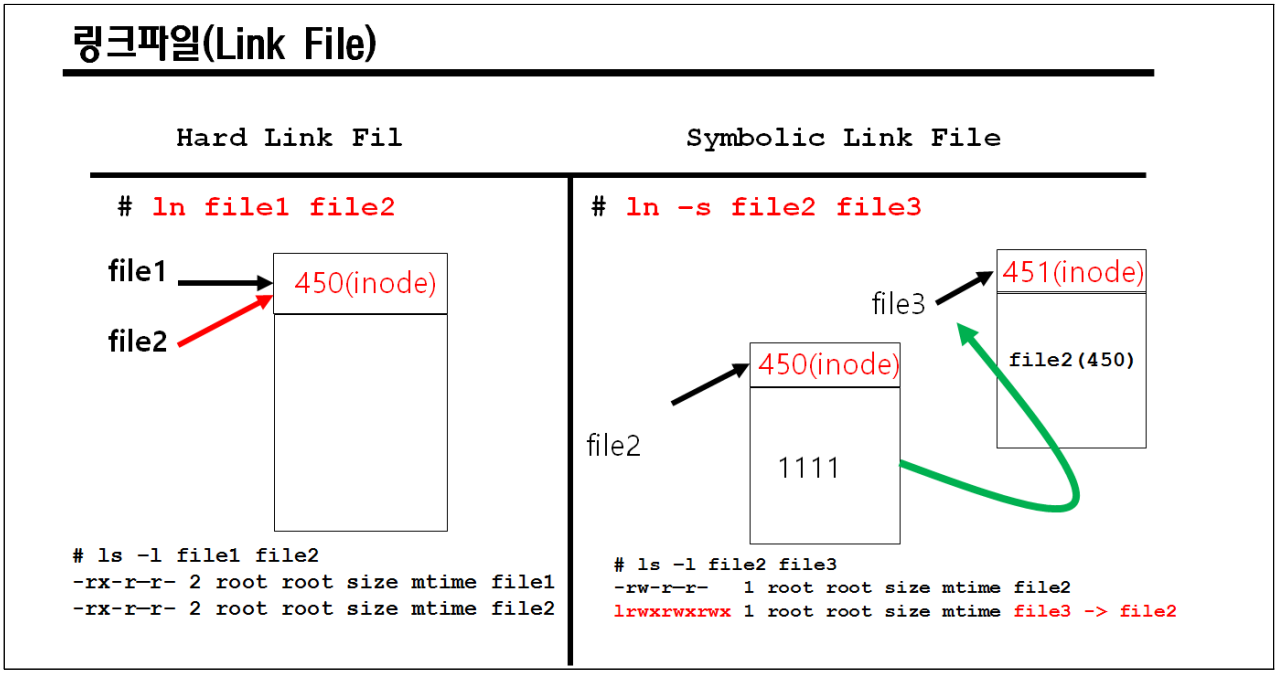
예를 들어 vi 편집기를 사용 할 수도 있고, 컴파일을 통해 바이너리를 생성 할 수도 있고, touch 명령어를 통해 빈 파일을 생성 할 수도 있다.

디렉토리 파일(Directory File)




```
# mkdir dir1
# ls -a dir1
450 . 440 ..
# touch dir1/file1
# mkdir dir1/dir2
# ls -a
450 . 440 .. 500 file1
501 dir2
```

디렉토리의 데이터 블록 안에는 실제 디렉토리가 포함하고 있는 파일과 디렉토리에 대한 정보를 가지고 있다. 파일의 이름, 디렉토리의 이름 Inode 번호를 저장하고 있다. 디렉토리의 데이터 블록 안에는 어떤 파일 종류이든 파일의 종류에 맞는 정보가 들어 갈수 있다.



■ 링크 파일
- 하드링크 파일(Hard Link File)
- 심볼릭링크 파일(Symbolic Link File), 소프트 링크 파일(Soft Link File)

(1) 하드 링크 (Hard link)

원본파일의 경로를 저장하고 있는 파일을 말한다. 파일을 실제 경로가 아니라 사용하기 편리한 다른 경로로 접근할 수 있도록 지정하는 명령어이다. 링크파일의 type은 "ls -l" 했을 때 맨 앞의 글자가 "l"로 표시되어 있다.

하드 링크는 똑같은 파일크기로 원본 파일이 수정될 경우, 하드 링크된 파일도 원본과 동일하게 변경되며 항상 같은 내용을 유지할 수 있다. 원본이 삭제되어도 원본과 동일한 내용을 가지고 있으므로 자원을 공유하되 데이터를 안전하게 관리하고자 할 때 유용하게 사용할 수 있다. 하드링크는 디렉토리에는 만들 수 없다.

하드 링크는 파일명이 하나 더 만들어져 동일한 I-node 번호를 가르치게 된다. 하드 링크는 같은 I-node 번호를 저장하고 있으므로 원본 파일을 지워도 원본파일의 파일명만 지워질 뿐 inode가 지워지는 것이 아니기 때문에 링크로 inode값을 가르키는 링크 파일에 영향을 미치지 않는다.

원본 파일과 링크 파일의 권한은 항상 같다. 그 이유는 I-node에 권한이 저장이 되어 있기 때문이다. 즉 파일명에 대해서 권한을 부여하는 것이 아니고 inode번호에 대해 권한을 설정하기 때문이다. 이는 각기 다른 이름을 가진 하드링크의 파일명에 대한 권한을 변경해줘도 파일명에 권한 설정이 되는 것이 아니기 때문에 서로 다른 이름을 가진 하드링크 파일의 권한은 동일하게 유지가 되는 것이다.

(2) 심볼릭 링크, 소프트 링크 (Symbolic link, soft link)

일반적으로 링크라고 하면 심볼릭 링크를 말한다. 심볼릭 링크는 소프트 링크라고도 하며 불필요한 파일의 복사를 하지 않아도 된다. 보통 여러 디렉토리에서 동일한 라이브러리를 요구할 경우나, 하나의 파일을 여러 사람이 공통으로 사용할 경우도 많이 쓴다. 소프트 링크의 퍼미션은 모든 유저에게 모든 권한(rwxrwxrwx=777)을 준다.

포인트 하는 정보가 들어 있을 뿐 원본파일은 아니다. 심볼릭 size는 포인트 정보만 들어 있어 cat으로 보여지는 정보가 많던 적던간에 동일한 size를 갖는다. 따라서 윈도우에서 바탕화면에 바로가기가 되어있어도 많은 양의 리소스를 차지하지 않고 포인트 정보를 담은 size만이 disk의 공간을 차지하게 된다.

원본파일이 지워지게 되면 링크 파일에 영향을 미치게 된다. 소프트링크 파일의 권한은 항상 모든 권한이다.

[참고] 소프트 링크파일의 퍼미션은 항상 777인 이유는?

소프트링크를 만드는 이유는 모든 사용자가 자원을 공유하려 할 때 많이 사용하기 때문이다. 그런데 소프트링크파일은 그 안의 데이터 블록에 원본파일의 inode의 경로를 포함하는 것이므로 링크파일의 퍼미션이 모든 권한이라 하더라도 실제로는 원본 파일의 권한을 따르게 된다. 링크파일의 퍼미션을 변경하면 원본 파일의 퍼미션이 변경이 된다.

확인]

```
# cd /etc/
# ls -l grub.conf
```

```
lrwxrwxrwx 1 root root 22  1월 25 22:54 grub.conf -> ../boot/grub/grub.conf
                                          링크파일       원본파일
```

```
# ls -l /boot/grub/grub.conf
```

```
-rw------- 1 root root 600  1월 25 22:54 /boot/grub/grub.conf
```

```
# su - fedora
$ cat /etc/grub.conf
```

```
cat: /etc/grub.conf: 허가 거부됨
```

 # man ln

```
NAME
       ln - 파일 링크

SYNOPSIS
       ln [options] source [dest]
       ln [options] source... directory
       Options:
       [-bdfinsvF]  [-S backup-suffix] [-V {numbered,existing,simple}] [--ver-
       sion-control={numbered,existing,simple}]    [--backup]    [--directory]
       [--force]  [--interactive]  [--no-dereference] [--symbolic] [--verbose]
       [--suffix=backup-suffix] [--help] [--version]

DESCRIPTION
       이 문서는 더이상 최신 정보를 담고 있지않다.  그래서,  몇몇틀릴  경우 도
       있고,  부족한  경우도  있을  것이다.  완전한매뉴얼을 원하면, Texinfo 문
       서를 참조하기 바란다.

       이 매뉴얼 페이지는 ln 명령의 GNU  버전에  대한  것이다.  마지막 인자 가
       경 로 이름이면, 원본 파일을 그 경로안에 같은 이름으로 링크시킨다.  하나
       의 원본 파일 이름만 지정되면, 현재 경로 안에서 같은 이름으 링크시킨 다.
       각 각 원본과 대상 파일 이름이 있으면, 원본 파일을 대상 파일로 링크 시킨
       다.  초기값은 하드링크이며, 대상 파일이 이미 있으면, 링크하지 않는다.

   OPTIONS
       -s, --symbolic
              심 블릭 링크.  심블릭 링크를  지원하지  않는시스템에서  이옵션을
              사용할 경우에는 오류 메시지를보여준다.
```

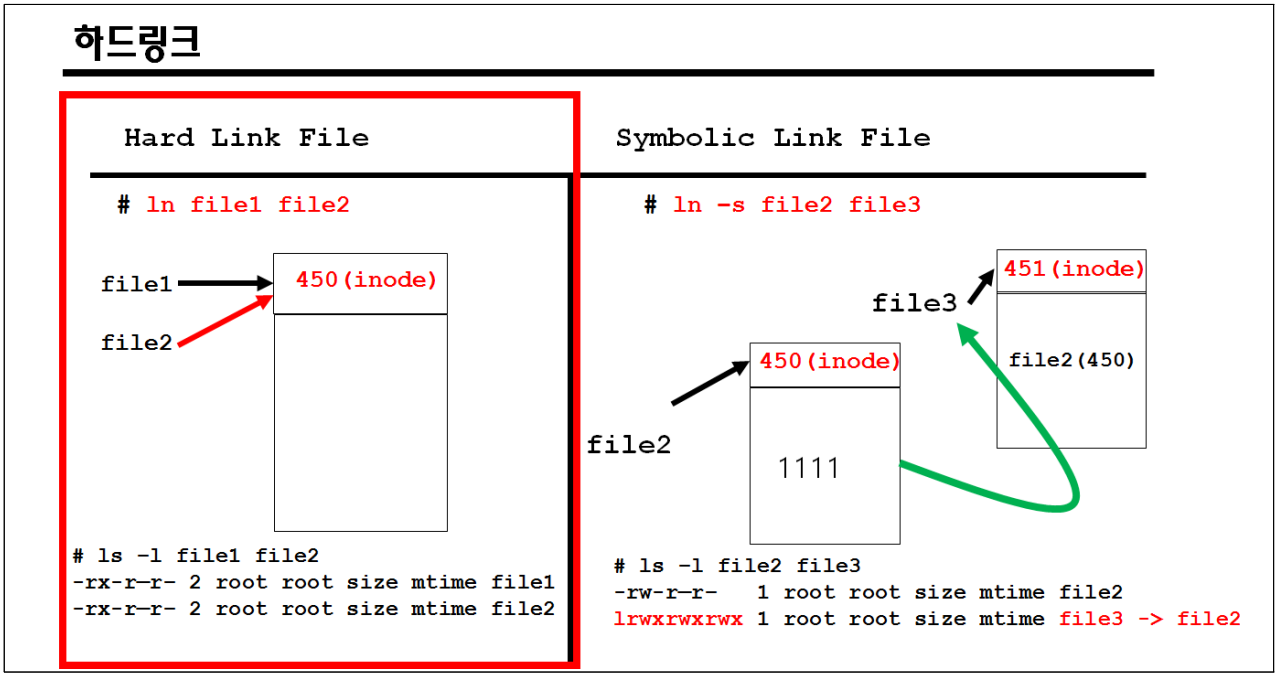
[명령어 형식]

```
# ln file1 file2
# ln -s file1 file2
```

(하드링크) **# ln file1 file2**
(심볼릭링크)**# ln -s file1 file2**

[명령어 옵션]

옵션	설명
-b, --backup	대상 파일이 있다면 백업파일을 생성한다.
-f, --force	링크를 생성할 대상 파일이 있더라도 강제적으로 새로운 링크를 생성한다.
-i, --interactive	링크를 생성할 대상 파일이 있을 경우, 삭제 유무를 사용자에게 물어 본다.
-n, --no-dereference	링크할 원본이 심볼릭 파일이면, 그 심볼릭 파일의 대상 파일을 추적하여 링크한다.
-s, --symbolic	링크할 원본이 심볼릭 파일이면, 심볼릭 파일을 링크한다.
-S, --suffix backup-suffix	링크를 생성할 대상 파일이 이미 있을 경우, 이전의 대상파일을 백업할 파일의 확장자를 지정한다.
-V, --version-control {numbered, existing, simple}	백업하는 방법을 지정한다. t,numbered : 항상 번호로 된 백업파일을 만든다. nil, existing : 대상파일이 있을 경우에만 백업파일을 만든다. never, simple : 간단한 백업을 만든다.



(실습 준비)
cd /test ; rm -rf /test/*

echo 1111 > file1
ls -l file1

```
-rw-r--r-- 1 root root 5 Feb 13 13:06 file1
```

cat file1

```
1111
```

(하드 링크 실습)
ln file1 file2
#

ls -li

```
5489017 -rw-r--r-- 2 root root 5 Feb 13 13:06 file1
5489017 -rw-r--r-- 2 root root 5 Feb 13 13:06 file2
```

echo 2222 >> file2
cat file2

```
1111
2222
```

cat file1

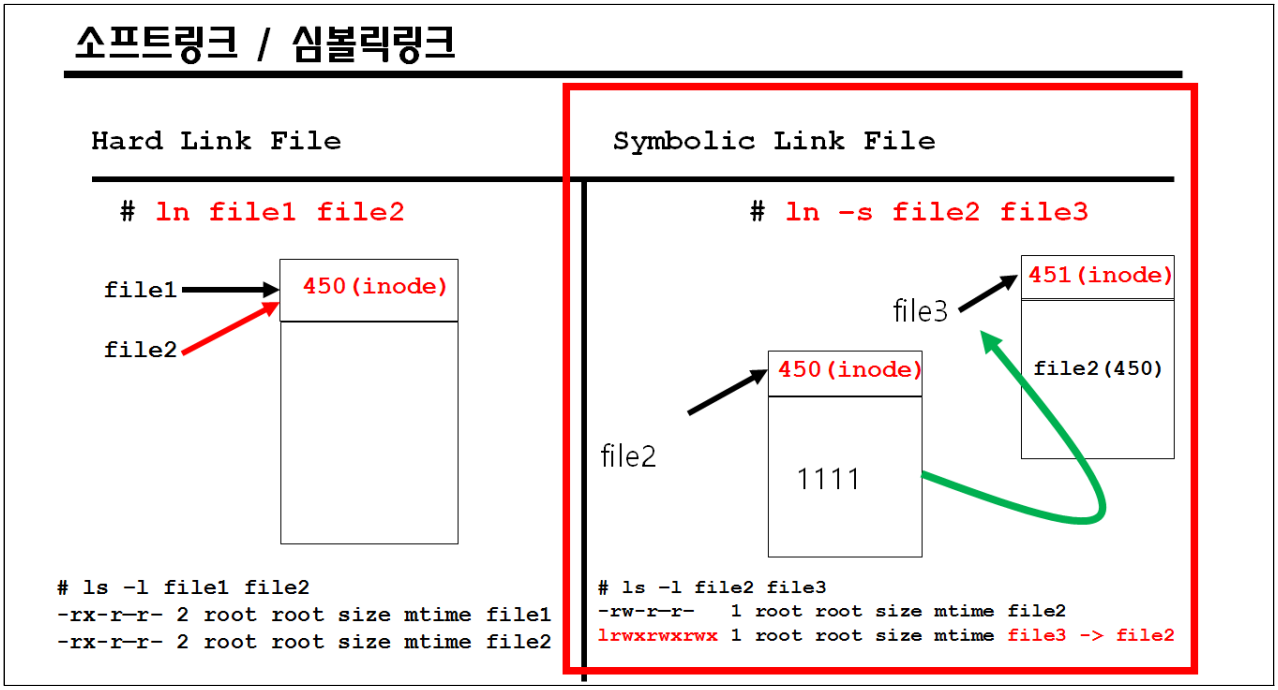
```
1111
2222
```

rm file1
cat file2

```
1111
2222
```

ls -l

```
-rw-r--r-- 1 root root 10 Feb 13 13:14 file2
```




```
# ln -s file2 file3
#
```

```
# ls -li
```

```
5489017 -rw-r--r-- 1 root root 10 Feb 13 13:14 file2
5488913 lrwxrwxrwx 1 root root  5 Feb 13 13:18 file3 -> file2
```

file3은 file2번을 링크하며 file2번의 INODE가 저장되어있다.

```
# echo 3333 >> file3
# cat file3
```

```
1111
2222
3333
```

```
# cat file2
```

```
1111
2222
3333
```

```
# rm file2
#
```

```
# cat file3
```

```
cat: file3: No such file or directory
```

[EX3] 하드 링크 & 심볼릭 링크 설명

하드링크(**# ln file1 file2**)
- 두개의 파일의 inode 번호가 동일한가?
- ls -li 특이한 변화는?
- 두개의 파일 사이즈는 동일한가?
- file2의 편집하면?

심볼릭 링크(**# ln -s file1 file2**)
- 두개의 파일의 inode 번호가 동일한가?
- ls -li 특이한 변화는?
- 두개의 파일 사이즈는 동일한가?
- file2의 편집하면?

[EX4] 하드 링크 & 심볼릭 링크의 차이점
- 파일시스템을 넘어서 링크를 걸수 있는가?
- 디렉토리에 링크를 걸수 있는가?

[참고] 심볼릭 링크 -> 윈도우의 바탕화면 아이콘
"아이콘".lnk

(주의) 심볼릭 링크는 일반적으로 상대경로 사용하지 않는다.
(X) # ln -s dir1 dir2
(O) # ln -s /test/dir1 /test/dir2

(실무 예) 웹서버(WAS 서버)에서 웹소스 디렉토리 마이그레이션 하는 작업

```
----- DAUM Web Server -----                  ----- Web Client -----
        httpd(80)        <----------------   http://www.daum.net:80
        /was/index.html
```

작업: /was ---> /zeus
```
# ln -s /was /zeus
```

(실무 예) 관리 디렉토리 통합
FTP : /etc/vsftpd
DNS : /var/named
WEB : /var/www/html

```
# cd /test ; rm -rf /test/*
# ln -s /etc/vsftpd FTP       /* FTP 서비스 프로그램 디렉토리 */
# ln -s /var/named DNS        /* DNS 서비스 프로그램 디렉토리 */
# ln -s /var/www/html WEB     /* WEB 서비스 프로그램 디렉토리 */
# cd /test ; ls
# cd FTP
# cd ../DNS
# cd ../WEB
```

(실무 예) 버전 관리
/usr/local/tomcat-4.X ----> /usr/local/tomcat
/usr/local/tomcat-5.X ----> /usr/local/tomcat
/usr/local/tomcat-6.X ----> /usr/local/tomcat

(초기 프로그램을 설치하는 경우)
```
# cd /usr/local
# ls
tomcat-4.X
```

```
# ln -s tomcat-4.x tomcat
# cd tomcat
```
설정 작업(EX: /usr/local/tomcat)

(버전 업그레이드 하는 경우)
```
# cd /usr/local
# ls
tomcat-4.X  tomcat-5.X  tomcat
```

```
# rm tomcat
# ln -s tomcat-5.x tomcat
```

```
# cd tomcat
```
이전설정(/usr/local/tomcat) 복사

디바이스 파일(Device File)

■Block Device File
■Character Device File(=Raw Device File)

```
crw------- 1 root root     5,    1  5월 12 10:52 console
lrwxrwxrwx 1 root root          11  5월 12 10:52 core -> /proc/kcore
drwxr-xr-x 3 root root          60  5월 12 10:52 cpu
crw------- 1 root root    10,   63  5월 12 10:52 cpu_dma_latency
drwxr-xr-x 6 root root         120  5월 12 10:52 disk
brw-r----- 1 root disk     8,    0  5월 12 10:52 sda
brw-r----- 1 root disk     8,    1  5월 12 10:52 sda1
brw-r----- 1 root disk     8,    2  5월 12 10:52 sda2

# ls -l /dev | grep '^b'
# ls -l /dev | grep '^c'
```

■ 디바이스 파일(장치 파일, Device File)
- 블럭 디바이스 파일(Block Device File)
- 캐릭터 디바이스 파일(Character Device File) = Raw Device File

■ 블럭 디바이스 파일
- 블럭 단위로 I/O 발생
- 디스크 디바이스(Disk Device)이면 I/O 단위는 4K(4096 Bytes)

■ 캐릭터 디바이스 파일
- 바이트 단위로 I/O 발생
- 디스크 디바이스(Disk Device)이면 I/O 단위는 512 Bytes(1 Sector = 512 bytes)

■ Major Device 번호
- 장치의 종류
- 장치의 종류가 틀리면 Major Device 번호가 틀리다.

■ Minor Device 번호
- 개별적인 장치의 종류 또는 동작 방법의 차이
- 개별적인 장치의 종류가 틀리면 Minor Device 번호가 틀리다.
- 또는 같은 장치라도 동작 방법이 틀리면 Minor Device 번호가 틀리다.

[EX1] 블럭 디바이스/캐릭터 디바이스 파일 확인

```
# ls -l /dev | grep '^b'
brw-rw---- 1 root floppy   2,   0 6월 11 13:02 fd0
brw-rw---- 1 root floppy   2,  84 6월 11 13:02 fd0u1040
brw-rw---- 1 root floppy   2,  88 6월 11 13:02 fd0u1120
brw-rw---- 1 root floppy   2,  28 6월 11 13:02 fd0u1440
brw-rw---- 1 root floppy   2,  44 6월 11 13:02 fd0u1680
..... (중략) .....
brw-rw---- 1 root floppy   2,  68 6월 11 13:02 fd0u830
brw-rw---- 1 root disk    22,   0 6월 11 13:02 hdc
brw-r----- 1 root disk     7,   0 6월 11 13:02 loop0
brw-r----- 1 root disk     7,   1 6월 11 13:02 loop1
..... (중략) .....
brw-r----- 1 root disk     9,   0 6월 11 13:02 md0
brw-r----- 1 root disk     1,   0 6월 11  2010 ram0
brw-r----- 1 root disk     1,   1 6월 11  2010 ram1
brw-r----- 1 root disk     1,  10 6월 11  2010 ram10
brw-r----- 1 root disk     1,  11 6월 11  2010 ram11
..... (중략) .....
brw------- 1 root root     8,   1 6월 11  2010 root
brw-r----- 1 root disk     8,   0 6월 11  2010 sda
brw-r----- 1 root disk     8,   1 6월 11 13:02 sda1
brw-r----- 1 root disk     8,   2 6월 11  2010 sda2
..... (중략) .....
```

```
# ls -l /dev | grep '^c'
crw-rw---- 1 root audio   14,  12 6월 11 13:02 adsp
crw------- 1 root root    10, 175 6월 11 13:02 agpgart
crw-rw---- 1 root audio   14,   4 6월 11 13:02 audio
crw------- 1 root root    10,  61 6월 11 13:02 autofs
crw------- 1 root root     5,   1 6월 11 13:02 console
crw-rw---- 1 root root    14,   9 6월 11 13:02 dmmidi
crw-rw---- 1 root audio   14,   3 6월 11 13:02 dsp
crw-rw-rw- 1 root root     1,   7 6월 11 13:02 full
crw------- 1 root root    10, 228 6월 11 13:02 hpet
crw------- 1 root root     1,  11 6월 11 13:02 kmsg
crw-rw---- 1 root lp       6,   0 6월 11 13:02 lp0
crw-r----- 1 root kmem     1,   1 6월 11 13:02 mem
crw-rw---- 1 root audio   14,   2 6월 11 13:02 midi
crw-rw---- 1 root audio   14,   0 6월 11 13:02 mixer
crw-rw-rw- 1 root root     1,   3 6월 11 13:02 null
crw-rw---- 1 root root    10, 144 6월 11 13:02 nvram
crw------- 1 root root     1,  12 6월 11 13:02 oldmem
crw-rw---- 1 root lp      99,   0 6월 11 13:02 parport0
crw-rw---- 1 root lp      99,   1 6월 11 13:02 parport1
crw-rw---- 1 root lp      99,   2 6월 11 13:02 parport2
crw-rw---- 1 root lp      99,   3 6월 11 13:02 parport3
crw-r----- 1 root kmem     1,   4 6월 11 13:02 port
crw------- 1 root root   108,   0 6월 11 13:02 ppp
crw-rw-rw- 1 root tty      5,   2 6월 11 13:59 ptmx
crw-rw-rw- 1 root root     1,   8 6월 11 13:02 random
crw------- 1 root root   162,   0 6월 11 13:02 rawctl
crw-r--r-- 1 root root    10, 135 6월 11  2010 rtc
crw-rw---- 1 root audio   14,   1 6월 11 13:02 sequencer
crw-rw---- 1 root audio   14,   8 6월 11 13:02 sequencer2
crw------- 1 root root    21,   0 6월 11 13:02 sg0
crw------- 1 root root    10, 231 6월 11 13:02 snapshot
crw------- 1 root root     4,   0 6월 11  2010 systty
crw-rw-rw- 1 root tty      5,   0 6월 11 13:02 tty
crw-rw---- 1 root root     4,   0 6월 11  2010 tty0
crw------- 1 root root     4,   1 6월 11 13:02 tty1
crw-rw---- 1 root tty      4,  10 6월 11  2010 tty10
crw-rw---- 1 root tty      4,  11 6월 11  2010 tty11
..... (중략) .....
cr--r--r-- 1 root root     1,   9 6월 11 13:02 urandom
crw------- 1 root root   442,   0 6월 11 13:02 usbdev1.1_ep00
crw------- 1 root root   442,   0 6월 11 13:02 usbdev1.1_ep81
crw------- 1 root root   442, 2048 6월 11 13:02 usbdev2.1_ep00
crw------- 1 root root   442, 2048 6월 11 13:02 usbdev2.1_ep81
```