

# Linux 고급 명령어

---

# Contents

---

□ 사용자 및 그룹 관리

□ 정보검색 명령어

---

# 사용자 및 그룹 관리

---

## □ 사용자에 관하여

- 사용자 데이터베이스
- 사용자 관리도구
- 권한 및 그룹 설정
- SetUID와 SetGID

## □ 정보검색 명령어

---

# 사용자에 관하여

“리눅스에서 모든 파일과 프로그램은 어떤 사용자(user)에 의해 소유되어야 한다.”

## □ 파일이나 프로그램에 대한 접근 가능성

- **UID** (User ID)와 **GID** (Group ID)를 바탕으로 검사
- 수행중인 프로그램은 그것을 실행한 사용자의 권한과 허가권을 상속받는다.

## □ 홈 디렉토리 (Home Directory)

- 시스템에 로그인하는 모든 사용자가 자신의 환경 설정 파일을 저장하는 장소
  - 대부분 **/home** 에 홈 디렉토리를 둔다.
  - root 사용자의 홈 디렉토리
    - 유닉스 : /
    - 리눅스 배포판 : /root

```
[root@localhost home]# pwd
/home
[root@localhost home]# ls
jang kim lee
[root@localhost home]# _
```

jang kim lee

# 사용자에 관하여 (cont`d)

---

## □ 셸 (Shell)

- 커널과 사용자 프로그램의 중간에 위치하는 명령어 해석기
- 윈도우의 `command.com`이나 프로그램 관리자 또는 윈도우 탐색기와 유사한 프로그램이라고 생각하면 된다.
- 본셸(bourne shell), C셸(csh), 콘 셸(ksh), 배쉬(bash)셸이 존재한다.

## □ 시작 스크립트

- 환경 설정 파일은 셸 스크립트(shells script) 형태로 존재한다.
- **.bashrc**
  - BASH가 사용하는 시작 스크립트 파일의 이름

## □ **/etc/passwd** 파일

- 7가지의 필드로 구성되어 있고 필드의 구분은 콜론(:)에 의해서 구분
- UID는 0을 제외하고는 모든 사용자에게 유일한 숫자 부여

예) **root:x:0:0:root:/root:/bin/bash**

계정:패스워드:UID:GID:계정 COMMENT:사용자계정 디렉토리:사용자로그인 셸

```
[root@localhost ~]# cat /etc/passwd | more
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
[root@localhost ~]# cat /etc/passwd | more
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
```

# 사용자 데이터베이스 (cont`d)

## □ /etc/shadow 파일

- 암호화된 패스워드 필드 외에도 패스워드 유효기한 및 계정 사용여부에 관한 정보도 포함

예) root : \$1\$nnhTqCCE\$AGwXjd6/wcyzXoR3uKdUs/ : 14319 : 0 : 99999 : 7 : : :

- 로그인 이름
- 암호화된 패스워드
- 1970년 1월 1일부터 계산해서 최근에 패스워드를 변경한 날까지의 날 수
- 패스워드를 변경한 후에 다시 변경이 가능한 날까지 남은 날 수
- 패스워드를 변경해야만 하는 날까지 남은 날 수
- 패스워드가 만료되기 전에 사용자에서 미리 경고하는 시간
- 계정이 사용 불가능하게 되기 전에 패스워드를 변경해야만 하는 날까지 남은 날 수
- 1970년 1월 1일 이후로 계정 사용이 불가능하게 된 날까지의 날 수
- 예약 항목

```
[root@localhost ~]# cat /etc/shadow | more
root:$1$nnhTqCCE$AGwXjd6/wcyzXoR3uKdUs/ : 14319 : 0 : 99999 : 7 : : :
bin:!:14319:0:99999:7:::
daemon:!:14319:0:99999:7:::
adm:!:14319:0:99999:7:::
lp:!:14319:0:99999:7:::
sync:!:14319:0:99999:7:::
shutdown:!:14319:0:99999:7:::
halt:!:14319:0:99999:7:::
```

# 사용자 데이터베이스 (cont`d)

## □ **/etc/group** 파일

- 각 사용자는 적어도 하나의 그룹에 속해 있어야 하고, 필요에 따라 이 그룹, 저 그룹에 동시에 소속될 수도 있다.

예) root : x : 0 : root

그룹이름:그룹의 암호화된 패스워드:GID 숫자:쉼표로 분리된 소속 사용자들

```
[root@localhost ~]# cat /etc/group | more
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
tty:x:5:
disk:x:6:root
lp:x:7:daemon,lp
mem:x:8:
```



# 사용자 관리도구 (cont`d)

## □ users

- 현재 시스템에 로그인한 사용자 계정을 조회하는 명령어

## □ finger

- 사용자 정보 구하기
- Syntax

□ **finger** [-options] *username*

-l	정보를 자세히 출력한다. (default)
-s	간단하게 정보를 출력한다.

- **username** 생략 : 시스템에 로그인한 모든 사용자 정보를 출력
- **username** 입력 : 특정 사용자의 정보를 출력

# 사용자 관리도구 (cont`d)

## □ users와 finger의 활용 예

```
[root@localhost ~]# users
kim root
[root@localhost ~]# finger
Login      Name      tty      Idle  Login Time  Office      Office Phone
kim        kim        tty7      39    Mar 16 19:47 (:0)
root       root       tty2      Mar 16 19:03
[root@localhost ~]# finger root
Login: root                                Name: root
Directory: /root                          Shell: /bin/bash
On since Mon Mar 16 19:03 (KST) on tty2
Mail last read Mon Mar 16 17:54 2009 (KST)
No Plan.
[root@localhost ~]# _
```

# 사용자 관리도구 (cont`d)

## □ useradd

■ 사용자 추가

■ Syntax

**useradd** [-options] *username*

<b>-c</b> <i>COMMENT</i>	사용자의 실제 이름을 세팅할 때 사용 (값에 공백이 있을 때는 돌레에 따옴표를 사용해야 한다.)
<b>-d</b> <i>HOME_DIR</i>	사용자의 홈 디렉토리를 직접 지정할 때 사용
<b>-u</b> <i>UID</i>	직접 <b>UID</b> 값을 지정하고 싶을 때 사용
<b>-g</b> <i>GROUP</i>	사용자의 기본 그룹을 지정(기본 그룹)
<b>-G</b> <i>GROUP[, ...]</i>	사용자의 기본 그룹 외에 추가적으로 속할 그룹을 지정(다중 그룹)
<b>-s</b> <i>SHELL</i>	사용자의 로그인 셸을 직접 지정하고 싶을 때 사용
<b>-m</b>	홈 디렉토리 자동 생성 옵션

# 사용자 관리도구 (cont`d)

## □ useradd

### ■ useradd -D

#### □ /etc/default/useradd

#### □ 계정 생성 기본 설정 파일 출력

```
[root@localhost ~]# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
[root@localhost ~]#
```

### Tip > ‘/etc/login.defs’ 와 ‘/etc/skel’

- useradd 명령을 입력하였을 때, 커널은 전체 사용자의 제약조건을 담고 있는 ‘/etc/login.defs’ 파일을 참조하여 사용자 메일 박스의 위치, 패스워드의 유지 방법, UID와 GID의 범위, 프린터 스푼의 위치 그리고 사용자 홈 디렉토리의 위치 등을 지정하게 된다.
- ‘/etc/skel’ 이라는 디렉토리 밑의 파일들을 새로운 사용자의 계정 디렉토리로 복사한다(사용자 기본설정 파일)

# 사용자 관리도구 (cont`d)

## □ passwd

- 계정 암호 지정 및 변경
- Syntax

**passwd** [-options] *username*

- 사용자 정보와 패스워드는 통상 '/etc/passwd' 파일에 저장되어 사용자가 로그인시 항상 인증 절차를 거치도록 설계된다
- 패스워드가 x인 경우 shadow 패스워드로 '/etc/shadow' 파일에 암호화되어 저장된다

<b>-d</b>	계정에 대해 패스워드를 지운다.
<b>-f</b>	강제로 실행한다.
<b>-l</b>	계정을 삭제하지 않고 시스템에 접속하지 못하도록 한다.
<b>-S</b>	계정이 어떤 암호 체계를 가지고 있는지 보여준다.
<b>-u</b>	-l 로 인한 lock을 해제한다.

# 사용자 관리도구 (cont`d)

## □ adduser

- useradd와 같이 새로운 사용자를 추가하는 명령어

- Syntax

**adduser** [-options] *username*

```
root@jyn-linux:~# adduser test1
Adding user `test1' ...
Adding new group `test1' (1001) ...
Adding new user `test1' (1001) with group `test1' ...
Creating home directory `/home/test1' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for test1
Enter the new value, or press ENTER for the default
  Full Name []: Test1
   Room Number []:
   Work Phone []:
   Home Phone []:
    Other []:
Is the information correct? [Y/n]
root@jyn-linux:~#
```

-c <i>COMMENT</i>	/etc/passwd 파일의 comment 필드에 넣을 정보 입력.
-d <i>HOME_DIR</i>	사용자의 홈 디렉토리를 직접 지정할 때 사용
-g <i>GROUP</i>	사용자의 기본 그룹을 지정(기본 그룹)
-e <i>EXPIRE_DATE</i>	계정 만료시기 설정 옵션. YYYY-MM-DD 형식으로 지정.

# 사용자 관리도구 (cont`d)

## □ su

- 임의의 사용자가 다른 사용자 계정으로 별도의 로그인 없이 전환하도록 하는 명령이다.

Tip > 'su' 와 'su -'의 차이점

- 'su'는 현재 사용자의 환경을 그대로 물려받은 채 root의 권한을 가진다.
- 'su -'는 직접 root 사용자로 로그인 했을 때와 동일한 환경으로 초기화된다.

```
[root@localhost test]# useradd park
[root@localhost test]# su park
[park@localhost test]$ exit
exit
[root@localhost test]# su - park
[park@localhost ~]$
```

# 사용자의 관리도구 (cont`d)

## □ usermod

### ■ 계정 사용자 정보 변경

- 사용자의 로그인 이름이나 **UID**를 변경하기 전에 사용자가 현재 시스템에 로그인하고 있거나 프로세스를 수행시키고 있는지를 확인해야 한다.

### ■ Syntax

- **usermod** [-options] *username*

<b>-c</b> <i>COMMENT</i>	사용자 설명 필드의 내용을 변경
<b>-d</b> <i>HOME_DIR</i>	사용자의 홈 디렉토리를 변경
<b>-u</b> <i>UID</i>	사용자의 <b>UID</b> 값을 변경
<b>-g</b> <i>GROUP</i>	사용자의 그룹을 변경
<b>-s</b> <i>SHELL</i>	사용자의 로그인 셸을 변경
<b>-l</b> <i>USERNAME</i>	사용자의 계정명을 변경



# 사용자 관리도구 (cont`d)

---

## □ userdel

### ■ 사용자 계정 삭제

- /etc/passwd 및 /etc/shadow 파일에 있는 엔트리와 /etc/group 파일에 있는 관련 사항을 자동으로 제거한다.

### □ Syntax

**userdel** [-options] *username*

-f	파일 소유자와 상관 없이 강제로 삭제
-r	홈 디렉토리와 메일 저장소를 모두 삭제

# 사용자 관리도구 (cont`d)

## □ Step by Step : 사용자 추가 및 편집하기

- Ctrl + Alt + F1 을 눌러 터미널을 연다.
- Root 계정으로 로그인 한다.

```
Ubuntu 10.10 jyn-linux tty5
jyn-linux login: root
Password:
Last login: Mon Mar  7 16:49:49 KST 2011 on tty2
Linux jyn-linux 2.6.35-27-generic #48-Ubuntu SMP Tue Feb 22 20:25:29 UTC 2011 i6
86 GNU/Linux
Ubuntu 10.10

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/

root@jyn-linux:~# _
```

# 사용자 관리도구 (cont`d)

## □ Step by Step : 사용자 추가 및 편집하기

- adduser 명령어를 사용하여 자신의 계정을 추가한다
- finger 명령을 사용하여 사용자 정보를 확인한다.
  - finger 명령이 동작을 하지 않을 때
    - sudo apt-get install finger

```
root@jyn-linux:~# adduser myid
Adding user `myid' ...
Adding new group `myid' (1001) ...
Adding new user `myid' (1001) with group `myid' ...
Creating home directory `/home/myid' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for myid
Enter the new value, or press ENTER for the default
  Full Name []: Joun youngnam
  Room Number []:
  Work Phone []:
  Home Phone []:
  Other []:
Is the information correct? [Y/n] y
root@jyn-linux:~# _
```

```
root@jyn-linux:~# finger myid
Login: myid                                Name: Joun youngnam
Directory: /home/myid                     Shell: /bin/bash
On since Mon Mar  7 16:50 (KST) on tty4    1 hour 42 minutes idle
(messages off)
On since Mon Mar  7 17:56 (KST) on tty2    1 hour 37 minutes idle
(messages off)
On since Mon Mar  7 17:19 (KST) on tty3    1 hour 42 minutes idle
(messages off)
No mail.
No Plan.
root@jyn-linux:~# _
```

# 사용자 관리도구 (cont`d)

## □ Step by Step : 사용자 추가 및 편집하기

- Ctrl+Alt+F2 으로 새로 콘솔을 열면 새로운 계정으로 로그인 가능하다.

```
Ubuntu 10.10 jyn-linux tty2
jyn-linux login: myid
Password:
Linux jyn-linux 2.6.35-27-generic #48-Ubuntu SMP Tue Feb 22 20:25:29 UTC 2011 i686 GNU/Linux
Ubuntu 10.10

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

myid@jyn-linux:~$
```

# 사용자 관리도구 (cont`d)

## □ Step by Step : 사용자 추가 및 편집하기 (cont`d)

- logout 한 후 Ctrl+Alt+F1 으로 이전 콘솔로 돌아간다.
- usermod 명령을 사용하여 새로운 사용자 정보를 변경한다.
- finger 명령을 통해 바뀐 사용자 정보를 확인한다.
- userdel 명령을 사용하여 새로운 사용자를 삭제한다.

```
root@jyn-linux:~# finger myid
Login: myid                      Name: Joun youngnam
Directory: /home/myid           Shell: /bin/bash
On since Mon Mar  7 16:50 (KST) on tty4  1 hour 46 minutes idle
(messages off)
On since Mon Mar  7 19:41 (KST) on tty2   6 seconds idle
(messages off)
```

```
root@jyn-linux:~# usermod -c "Hong kil dong" myid
root@jyn-linux:~# finger myid
Login: myid                      Name: Hong kil dong
Directory: /home/myid           Shell: /bin/bash
On since Mon Mar  7 16:50 (KST) on tty4  1 hour 47 minutes idle
```

```
root@jyn-linux:~# userdel myid
root@jyn-linux:~# finger myid
finger: myid: no such user.
root@jyn-linux:~# _
```

# 사용자의 관리도구 (cont`d)

---

## □ groupadd

- /etc/group 파일에 그룹 추가

- Syntax

**groupadd** [-options] *groupname*

<b>-g</b> <i>GID</i>	새로 만드는 그룹의 <b>GID</b> 를 입력값으로 지정한다.
<b>-r</b>	추가하려는 그룹이 시스템 그룹일 때 주는 옵션 (499보다 작은 <b>GID</b> 값을 새 그룹에 부여할 때)

# 사용자의 관리도구 (cont`d)

---

## □ groupdel

- /etc/group 파일에 그룹 삭제

- Syntax

**groupdel** *group\_name*

---

# 사용자 관리도구 (cont`d)

---

## □ groupmod

- 그룹 등록 정보 변경

- Syntax

**groupmod** [-options] *group\_name*

<b>-g</b> <i>GID</i>	변경할 그룹의 <b>GID</b> 를 입력값으로 지정한다.
<b>-n</b> <i>GROUP_NAME</i>	그룹 이름 변경 시 사용



# 사용자 관리도구 (cont`d)

## □ 그룹 추가, 변경, 삭제

```
[root@localhost mail]# groupadd -g 7777 newgrp
[root@localhost mail]# cat /etc/group
root:x:0:root
bin:x:1:root,bin,daemon
```



```
park:x:504:
newgrp:x:7777:
[root@localhost mail]# groupmod -g 8888 newgrp
[root@localhost mail]# cat /etc/group
root:x:0:root
bin:x:1:root,bin,daemon
```



```
park:x:504:
newgrp:x:8888:
[root@localhost mail]# groupdel newgrp
[root@localhost mail]# cat /etc/group
root:x:0:root
bin:x:1:root,bin,daemon
```



```
grub:x:503:
park:x:504:
[root@localhost mail]#
```

# 권한 및 그룹 설정

## □ chmod

- 파일 또는 디렉토리의 접근권한을 변경
- change file modes

### ■ Syntax

**chmod** [options] [접근권한] *file or directory\_name*

<b>-f</b>	강제로 수행한다. (변경이 되지 않더라도 오류 메시지를 보여주지 않는다.)
<b>-R</b>	디렉토리 구조를 따라 내려가면서 서브 디렉터리의 모드도 순환적으로 모드를 변경한다.

# 권한 및 그룹 설정 (cont`d)

## □ chmod (cont`d)

- 기호모드 (symbolic mode)

**chmod** [ugoa] [+ -=][rwx] *file or directory\_name*

	옵션	설명
사용자	u	사용자 (User) : 파일 소유자
	g	사용자가 속한 그룹 (Group)
	o	기타 사용자 (Other user)
	a	위 모든 경우 (All user)
허가 여부	+	허가 첨가
	-	허가 삭제
	=	이 옵션 이외의 나머지는 모두 취소
허가 종류	r	읽기 (Read)
	w	쓰기 (Write)
	x	실행 허가 (Execute)

# 권한 및 그룹 설정 (cont`d)

## □ chmod (cont`d)

### ■ 8진수 모드 (Octal mode)

**chmod** [8진수 표기] 파일 및 디렉토리명

문자	허가권	값(8진수)	값(2진수)
r	읽기 (Read)	4	100
w	쓰기 (Write)	2	010
x	실행하기 (Execute)	1	001

문자	허가권	값(8진수)	값(2진수)
---	허가권 없음	0	000
r--	읽기만 가능	4	100
rw-	읽기, 쓰기 가능	6	110
rwX	읽기, 쓰기, 실행 가능	7	111
r-X	읽기, 실행 가능	5	101
--X	실행만 가능	1	001

# 권한 및 그룹 설정 (cont`d)

## □ chmod (cont`d)

```
[root@localhost test]# ll
total 8
-rw-r--r-- 1 root root 8 2009-03-16 21:53 test.txt
[root@localhost test]# chmod 700 test.txt
[root@localhost test]# ll
total 8
-rwx----- 1 root root 8 2009-03-16 21:53 test.txt
[root@localhost test]# chmod g+x test.txt
[root@localhost test]# ll
total 8
-rwx--x--- 1 root root 8 2009-03-16 21:53 test.txt
[root@localhost test]# chmod u-x test.txt
[root@localhost test]# ll
total 8
-rw---x--- 1 root root 8 2009-03-16 21:53 test.txt
[root@localhost test]# chmod ugo+x test.txt
[root@localhost test]# ll
total 8
-rwx--x--x 1 root root 8 2009-03-16 21:53 test.txt
[root@localhost test]# _
```

권한설정, 링크수, 소유자, 소유그룹, 파일크기, 마지막 접근시간, 파일명

# 권한 및 그룹 설정 (cont`d)

## □ **mkdir** 디렉토리 생성 시 접근 권한 지정 가능

### ■ 구문(Syntax)

**mkdir** -m **mode** 디렉토리명

```
[root@localhost test]# mkdir T1
[root@localhost test]# ll
total 8
drwxr-xr-x 2 root root 4096 2009-03-16 23:23 T1
[root@localhost test]# mkdir -m 777 T2
[root@localhost test]# ll
total 16
drwxr-xr-x 2 root root 4096 2009-03-16 23:23 T1
drwxrwxrwx 2 root root 4096 2009-03-16 23:23 T2
[root@localhost test]#
```

# 권한 및 그룹 설정 (cont`d)

---

## □ chown

- 파일의 소유권을 변경
- change ownership

### ■ Syntax

**chown** [-options] user[:group] *file\_name*

-R	디렉토리 구조를 따라 아래로 순환적으로 수행한다.
-f	강제로 수행한다(오류 메시지를 보여주지 않는다.).

# 권한 및 그룹 설정 (cont`d)

## □ chgrp

- 파일의 그룹을 변경
- change group

### ■ Syntax

**chgrp** [options] group *file\_name*

-R	파일이나 디렉토리 구조를 따라 아래로 순환적으로 수행한다.
-f	강제로 수행한다(오류 메시지를 보여주지 않는다.).

```
[root@localhost T1]# ll
total 8
-rw-r--r-- 1 root root 6 2009-03-17 00:52 test
[root@localhost T1]# chown kim test
[root@localhost T1]# ll
total 8
-rw-r--r-- 1 kim root 6 2009-03-17 00:52 test
[root@localhost T1]# chgrp kim test
[root@localhost T1]# ll
total 8
-rw-r--r-- 1 kim kim 6 2009-03-17 00:52 test
[root@localhost T1]# _
```



# 권한 및 그룹 설정 (cont`d)

## □ umask

- 새로 만들어지는 파일 및 디렉토리의 **default** 권한을 지정한다
  - 셸의 기본적인 umask 값은 0022 이다.
  - umask 값 설정 비트(bit) 들은 요청된 허가 설정비트들과 mask된다.
  - 기본적으로 파일은 실행권한을 갖지 못한다. (디렉토리 : 777, 파일 : 666)

Umask	디렉토리 허가권	파일 허가권
0002	775	664
0007	770	660
0020	757	646
0070	707	606
0022	755	644

	r	w	x	r	w	x	r	w	x
원래값	1	1	0	1	1	0	1	1	0
마스크	0	0	0	0	1	0	0	1	0
최종값	1	1	0	1	0	0	1	0	0

# 사용자 관리도구 (cont`d)

## □ Step by Step : umask 변경하기

- 기존의 umask를 확인하고, umask에 따른 새로운 파일의 접근 권한을 확인한다.
- umask를 변경하고, 새로운 파일을 만들어 접근권한을 확인한다.

```
[root@localhost test]# umask
0022
[root@localhost test]# touch test1.txt
[root@localhost test]# ll
total 4
-rw-r--r-- 1 root root 0 2009-03-16 22:06 test1.txt
[root@localhost test]# umask 0000
[root@localhost test]# touch test2.txt
[root@localhost test]# ll
total 8
-rw-r--r-- 1 root root 0 2009-03-16 22:06 test1.txt
-rw-rw-rw- 1 root root 0 2009-03-16 22:06 test2.txt
[root@localhost test]# umask 0022
[root@localhost test]#
```

## □ Syntax

- umask [-s] 모드

```
[root@localhost test]# umask
0022
[root@localhost test]# umask -S
u=rwx,g=rx,o=rx
```

# 권한 및 그룹 설정 (cont`d)

## □ chsh

- 사용자가 사용하고 있는 로그인 셸을 바꾸는 명령어
- change shell
- `cat /etc/shells` 하면 현재 사용 가능한 셸 이름들을 볼 수 있다.
- Syntax

**chsh** [-options] *user\_name*

-s SHELL	지정한 <i>Shell</i> 이 앞으로 사용할 로그인 셸이 된다.
-h HELP	도움말

```
myid@jyn-linux:~$ cat /etc/shells
# /etc/shells: valid login shells
/bin/csh
/bin/sh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
/bin/tcsh
/usr/bin/esh
/bin/dash
/bin/bash
/bin/rbash
/usr/bin/screen
```

```
myid@jyn-linux:~$ chsh -s /bin/dash
Password:
myid@jyn-linux:~$ _
```

```
jyn-linux login: myid
Password:
Last login: Mon Mar  7 19:49:35 KST 2011 on tty2
Linux jyn-linux 2.6.35-27-generic #48-Ubuntu SMP
86 GNU/Linux
Ubuntu 10.10

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/
$ _
```

# SetUID와 SetGID

## □ SetUID 비트, SetGID 비트

- 프로그램이 파일 소유자 및 그룹의 권한으로 실행된다.
  - 실행 시 파일 소유자 및 그룹의 권한으로 변경 되기 때문에 root권한 파일을 실행하는 도중에는 root의 권한을 갖는다.
- 프로그램에 SetUID를 세팅하려면
  - 지정할 허가권 값의 앞에 4를 붙인다.
- 프로그램에 SetGID를 세팅하려면
  - 지정할 허가권 값의 앞에 2를 붙인다.

```
[root@localhost ~]$ chmod 4775 /bin/cat
```

```
[kim@localhost ~]$ cat /etc/shadow  
cat: /etc/shadow: Permission denied
```

```
[root@localhost /]# chmod 4755 /bin/cat  
[root@localhost /]# ll /bin/cat  
-rwsr-xr-x 1 root root 23360 2007-10-31 01:52 /bin/cat  
[root@localhost /]# _
```

```
[kim@localhost ~]$ cat /etc/shadow  
root:$1$nnhTqCCE$AGwXjd6/wcyzXoR3uKdUs/:14319:0:99999:7:::  
bin:!:14319:0:99999:7:::  
daemon:!:14319:0:99999:7:::  
adm:!:14319:0:99999:7:::
```

# SetUID와 SetGID (cont`d)

## □ SetUID 동작 (cont`d)

### ■ SetUID의 사용 예 : passwd 명령

- /etc/shadow 파일은 일반사용자가 접근 불가능하다.
- passwd 명령을 사용해서 패스워드를 설정하면, 패스워드에 대한 암호화나 해시된 값이 /etc/shadow에 저장된다.

```
[root@localhost ~]$ ls -la /etc/shadow
```

```
[root@localhost ~]$ ls -la /usr/bin/passwd
```

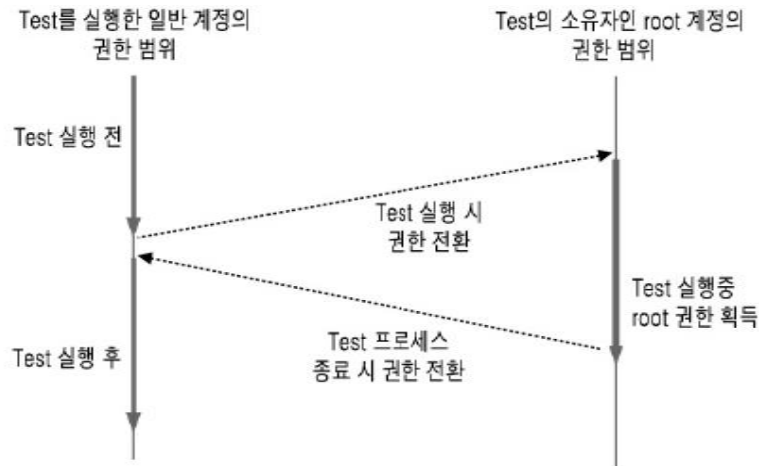
```
[root@localhost test]# ls -la /etc/shadow
-r----- 1 root root 1261 2009-03-16 21:12 /etc/shadow
[root@localhost test]# ls -la /usr/bin/passwd
-rwsr-xr-x 1 root root 25708 2007-09-26 06:32 /usr/bin/passwd
[root@localhost test]#
```

- passwd 파일에는 SetUID 권한이 주어져 있으며, 파일 소유자가 root 이므로 파일이 실행되는 프로세스는 실행 시간 동안 파일 소유자인 root 권한으로 실행된다.

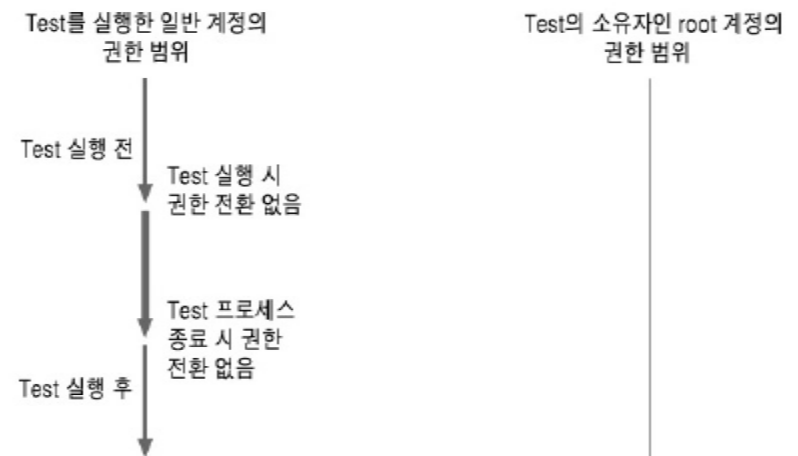
# SetUID와 SetGID (cont`d)

## □ SetUID 동작

Test에 SetUID 비트가 있을 경우



Test에 SetUID 비트가 없을 경우



# SetUID와 SetGID (cont`d)

## □ SetUID 설정 파일

- find 명령을 사용해 SetUID가 root로 설정된 파일들을 알 수 있다.

```
[root@localhost ~]$ find / -user root -perm +4000
```

```
/usr/lib/squid/ncsa_auth  
/usr/lib/squid/pam_auth  
/usr/kerberos/bin/ksu  
/usr/bin/chsh  
/usr/bin/rlogin  
/usr/bin/chfn  
/usr/bin/rcp  
/usr/bin/pulseaudio  
/usr/bin/Xorg  
/usr/bin/at  
/usr/bin/newgrp  
/usr/bin/crontab  
/usr/bin/gpasswd  
/usr/bin/sudo  
/usr/bin/sudoedit  
/usr/bin/rsh  
/usr/bin/chage  
/usr/bin/passwd  
/bin/mount  
/bin/su  
/bin/fusermount  
/bin/umount  
/bin/ping  
/bin/ping6  
[root@localhost test]#
```

# SetUID와 SetGID (cont`d)

## □ Sticky bit의 동작

- 스티키 비트는 디렉토리에만 주어지는 권한
  - 스티키 비트가 부여된 디렉토리에 있는 파일은 접근 권한과 상관없이 파일의 소유자와 관리자만이 파일을 삭제할 수 있다.
  - 스티키 비트가 주어진 대표적인 디렉토리로는 /tmp 디렉토리가 있다
  - 스티키 비트를 설정하려면 지정할 허가권 값의 앞에 1을 준다.

(chmod 1777 directory\_name)

```
drwxr-xr-x  3 root root  4096 2009-03-16 19:48 media
drwxr-xr-x  2 root root    0 2009-03-16 16:32 misc
drwxr-xr-x  2 root root  4096 2007-08-13 23:47 mnt
drwxr-xr-x  2 root root    0 2009-03-16 16:32 net
drwxr-xr-x  2 root root  4096 2007-08-13 23:47 opt
dr-xr-xr-x 133 root root    0 2009-03-16 16:30 proc
drwxr-x--  25 root root  4096 2009-03-16 19:06 root
drwxr-xr-x  2 root root 12288 2009-03-16 17:54 sbin
drwxr-xr-x  6 root root    0 2009-03-16 16:30 selinux
drwxr-xr-x  3 root root  4096 2009-03-16 22:50 srv
drwxr-xr-x 12 root root    0 2009-03-16 16:30 sys
drwxrwxrwt 17 root root  4096 2009-03-16 22:12 tmp
drwxr-xr-x 13 root root  4096 2009-03-16 22:18 usr
drwxr-xr-x 21 root root  4096 2009-03-16 22:57 var
[root@localhost ~]#
```



# SetUID와 SetGID (cont`d)

## □ Sticky bit의 동작 (cont`d)

- 스티키 비트가 설정된 /tmp 디렉토리

```
[root@localhost tmp]# touch 1234
[root@localhost tmp]# ls -la 1234
-rw-r--r-- 1 root root 0 2009-03-16 22:26 1234
[root@localhost tmp]# chmod 777 1234
[root@localhost tmp]# ls -la 1234
-rwxrwxrwx 1 root root 0 2009-03-16 22:26 1234
[root@localhost tmp]# su - kim
[kim@localhost ~]$ cd /tmp/
[kim@localhost tmp]$ ls -la 1234
-rwxrwxrwx 1 root root 0 2009-03-16 22:26 1234
[kim@localhost tmp]$ rm 1234
rm: cannot remove `1234': Operation not permitted
[kim@localhost tmp]$ _
```

□ 사용자 및 그룹 관리

□ 정보검색 명령어

- which, whereis

- find

- grep

---

## □ which

- 환경설정의 PATH에서 해당 명령어의 위치를 절대경로로 알려준다

- Syntax

**which** [-options] *command\_name*

```
[root@localhost tmp]# which perl
/usr/bin/perl
[root@localhost tmp]# which ls
alias ls='ls --color=tty'
        /bin/ls
[root@localhost tmp]# which which
alias which='alias ; /usr/bin/which --tty-only --read-alias --show-dot --show-ti
lde'
        /usr/bin/which
[root@localhost tmp]# _
```

# 정보검색 명령어 (cont`d)

## □ whereis

- 환경변수에 등록되어 있는 각종 경로를 검색하여 프로그램 이름과 그 프로그램이 저장되어 있는 절대 디렉토리, 관련 소스 파일(있는 경우) 그리고 명령어에 대한 man페이지(있는 경우) 위치 등의 정보를 출력한다.

## ■ Syntax

**whereis** *command*

```
[root@localhost tmp]# whereis tar
tar: /bin/tar /usr/include/tar.h /usr/share/man/man1/tar.1.gz
[root@localhost tmp]# whereis man
man: /usr/bin/man /etc/man.config /usr/share/man /usr/share/man/man1/man.1.gz /u
sr/share/man/man7/man.7.gz /usr/share/man/man1p/man.1p.gz
[root@localhost tmp]#
```

# 정보검색 명령어 (cont`d)

## □ find

- 리눅스 시스템 내에 존재하는 파일을 찾고자 할 때 사용

- Syntax

**find** [path] [-options] *file\_name*

<b>-name</b> <i>filename</i>	찾고자 하는 파일의 이름 지정
<b>-perm</b> <i>mode</i>	파일 권한(permission)이 일치되는 것 찾기
<b>-type</b> <i>filetype</i>	파일종류, 심볼릭 링크, 디렉토리 등에 따라 검색 디렉토리는 <b>d</b> , 파이프는 <b>p</b> , 심볼릭 링크는 <b>l</b> , 소켓은 <b>s</b> , 블록 파일은 <b>b</b> , 일반 파일은 <b>f</b> 등의 기호를 사용
<b>-size</b> <i>n</i>	파일 크기가 일치하는 것을 검색 파일 크기는 블록단위로 지정(1블록 == 512KB) 블록 숫자 뒤에 <b>k</b> 를 붙이면 1KB 크기의 블록 숫자로 간주
<b>-links</b> <i>n</i>	특정 개수의 링크를 가진 파일을 찾기 (링크의 개수 <i>n</i> 인 것)

# 정보검색 명령어 (cont`d)

## □ find (cont`d)

<b>-user</b> <i>name</i>	파일 사용자의 <b>name</b> (ID) 에 따라서 검색
<b>-group</b> <i>name</i>	그룹 소유주가 <b>name</b> 인 파일을 찾는다.
<b>-mtime</b> <i>n</i>	파일을 변경한 시간이 <b>n</b> 일이 지난 파일을 찾는다.
<b>-atime</b> <i>n</i>	<b>n</b> 일 전에 마지막 액세스한 파일을 찾는다.
<b>-amin</b> <i>n</i>	<b>n</b> 분 전에 마지막 액세스한 파일을 찾는다.
<b>-ctime</b> <i>n</i>	<b>n</b> 일 전에 마지막 변경한 파일을 찾는다.
<b>-cmin</b> <i>n</i>	<b>n</b> 분 전에 마지막 변경한 파일을 찾는다.
<b>-print</b>	찾는 파일을 화면에 출력

# 정보검색 명령어 (cont`d)

## □ find 활용 예

### ■ 와일드 카드 이용 검색

```
[root@localhost ~]$ find . -name '*.txt'
[root@localhost ~]$ find . -name "[a-z]*.jpg"
```

### ■ 마지막 수정한 일수로 검색

// (+) 기호는 “~보다 이상”을 의미하고 (-) 기호는 “~보다 이하”를 의미한다.

//5일 이전에 마지막으로 수정한 파일 찾기

```
[root@localhost ~]$ find . -mtime +5 -print
```

//20일 전부터 오늘까지 마지막으로 수정한 파일 찾기

```
[root@localhost ~]$ find . -type f -atime -20 -print
```

# 정보검색 명령어 (cont`d)

## □ find 활용 예 (cont`d)

### ■ 퍼미션을 이용한 검색

//파일 허가권이 521인 파일 찾기

```
[root@localhost ~]$ find . -perm -521 -print
```

//모든 사용자에게 모든 허가권이 주어진 디렉토리 찾기

```
[root@localhost ~]$ find . -type d -perm 777 -print
```

//루트 계정으로 setuid 파일 찾기

```
[root@localhost ~]$ find . -user root -perm -4000 -print
```

//그룹 members에 대한 setgid 파일 찾기

```
[root@localhost ~]$ find . -group members -perm -2000 -print
```

//-o 옵션을 이용하여 OR 결합되고 괄호를 이용하여 그룹을 지음

```
[root@localhost ~]$ find / -type f \( -perm -2000 -o -perm -4000 \)  
- print
```



# 정보검색 명령어 (cont`d)

## □ grep

- 문자열 및 문자열 패턴 검색, 조건에 맞는 문자열을 찾아서 출력
  - 주로 find나 ps등 결과물이 많은 경우 파이프라인(|)을 이용하여 검색 범위를 줄이는데 사용됨

## ■ Syntax

**grep** [-options] *pattern file\_name*

<b>-c</b>	파일 이름 다음에 숫자를 함께 출력, 패턴이 들어 있는 라인의 개수
<b>-h</b>	파일의 이름은 출력하지 않는다.
<b>-i</b>	대소문자 구분 안 함
<b>-l</b>	파일 이름만 출력하고 패턴이 있는 라인은 출력하지 않는다.
<b>-n</b>	각 라인 앞에 라인 번호를 붙인다.
<b>-w</b>	패턴이 원하는 단어로 독립되어 있을 경우만 찾는다.
<b>-v</b>	패턴이 없는 (inverse) 라인만 출력

예 ) `grep aaa aaa.txt, grep testid httpd.conf`

# 정보검색 명령어 (cont`d)

## □ 표현식 (experssion)

### ■ 간단한 문법

^	라인의 첫 시작 ^a 는 a로 시작하는 라인
\$	라인의 끝 a\$ 는 a로 끝나는 라인
.	한글자 a...b 사이에 3글자
*	이전의 글자나 정규식이 0회 이상 반복 $abc^* = abccccc$
+	*는 0회, +는 1회 이상 반복
[]	한 글자의 대체 글자의 목록 [abc] 는 abc 중 한 글자라도 반드시 있는 단어
[^]	[^abc] = abc는 반드시 없는 라인

# 정보검색 명령어 (cont`d)

## □ 정규 표현식과 **grep**의 사용 예제

```
[root@localhost test]# ps -ef | more
UID          PID    PPID  C STIME TTY          TIME CMD
root          1         0  0 16:30 ?          00:00:02 init [5]
root          2         0  0 16:30 ?          00:00:00 [kthreadd]
root          3         2  0 16:30 ?          00:00:00 [migration/0]
root          4         2  0 16:30 ?          00:00:00 [ksoftirqd/0]
root          5         2  0 16:30 ?          00:00:00 [watchdog/0]
root          6         2  0 16:30 ?          00:00:02 [events/0]
root          7         2  0 16:30 ?          00:00:00 [khelper]
root         58         2  0 16:30 ?          00:00:01 [kblockd/0]
root         61         2  0 16:30 ?          00:00:00 [kacpid]
root         62         2  0 16:30 ?          00:00:00 [kacpi_notify]
root        234         2  0 16:30 ?          00:00:00 [cqueue/0]
root        236         2  0 16:30 ?          00:00:00 [ksuspend_usbd]
```

```
[root@localhost test]# ps -ef | grep http
root        10717   8539  0 22:48 tty2          00:00:00 grep http
[root@localhost test]#
```

# 정보검색 명령어 (cont`d)

## □ 정규 표현식과 **grep**의 사용 예제 (cont`d)

■ `[root@localhost ~]$ grep NW t*`

□ t로 시작하는 모든 파일에서 NW를 포함하는 모든 행을 찾음

```
[root@localhost test]# cat test.txt
1234
NWKA
abdc cccc
[root@localhost test]# grep NW t*
test.txt:NWKA
[root@localhost test]# _
```

■ `[root@localhost ~]$ grep '^a' datafile`

□ a로 시작하는 모든 행을 출력

```
[root@localhost test]# cat test.txt
1234
NWKA
abdc cccc
[root@localhost test]# grep '^b' test.txt
[root@localhost test]# grep '^a' test.txt
abdc cccc
[root@localhost test]# _
```

# 정보검색 명령어 (cont`d)

## □ 정규 표현식과 **grep**의 사용 예제 (cont`d)

■ `[root@localhost ~]$ grep '4$' datafile`

□ 4로 끝나는 모든 행을 출력

```
[root@localhost test]# cat test.txt
1234
NWKA
abdc cccc
[root@localhost test]# grep '4$' test.txt
1234
[root@localhost test]# _
```

■ `[root@localhost ~]$ grep '[^0-9]' datafile`

□ 숫자가 아닌 문자를 하나라도 포함하는 모든 행을 출력

```
[root@localhost test]# cat test.txt
1234
NWKA
abdc cccc
[root@localhost test]# grep '[^0-9]' test.txt
NWKA
abdc cccc
[root@localhost test]#
```

# 정보검색 명령어 (cont`d)

## □ 정규 표현식과 **grep**의 사용 예제 (cont`d)

■ `[root@localhost ~]$ grep 'ab*' datafile`

- a가 나온 후, b가 0번 또는 여러 번 나오는 문자열을 포함한 모든 행을 출력

```
[root@localhost test1# cat test.txt
1234
NWKA
abdc cccc
[root@localhost test1# grep 'ab*' test.txt
abdc cccc
[root@localhost test1# _
```

```
[root@localhost test1# cat test1.txt
1234
NWKA
accc dddd
[root@localhost test1# grep 'ab*' test1.txt
accc dddd
[root@localhost test1#
```

# 정보검색 명령어 (cont`d)

## □ 정규 표현식과 **grep**의 사용 예제 (cont`d)

■ `[root@localhost ~]$ grep -n '^ab' datafile`

□ 행 번호를 함께 출력

```
[root@localhost test]# grep -n '^ab' test.txt
3:abdc cccc
[root@localhost test]#
```

■ `[root@localhost ~]$ grep -i 'ab' datafile`

□ 대소문자를 구별하지 않음

```
[root@localhost test]# cat test.txt
1234
NWKA
abdc cccc
ABCD BBBB
[root@localhost test]# grep 'ab' test.txt
abdc cccc
[root@localhost test]# grep -i 'ab' test.txt
abdc cccc
ABCD BBBB
[root@localhost test]#
```

# 정보검색 명령어 (cont`d)

## □ 정규 표현식과 **grep**의 사용 예제 (cont`d)

■ [root@localhost ~]\$ **grep -v 'ab' datafile**

□ 문자열 ab가 포함되지 않은 모든 행을 출력

```
[root@localhost test]# cat test.txt
1234
NWKA
abdc cccc
ABCD BBBB
[root@localhost test]# grep -v 'ab' test.txt
1234
NWKA
ABCD BBBB
[root@localhost test]# _
```

■ [root@localhost ~]\$ **grep -l 'ab' \***

□ 패턴이 찾은 파일의 행 번호 대신 단지 파일이름만 출력

```
[root@localhost test]# grep -l 'ab' *
test.txt
[root@localhost test]# _
```



# 정보검색 명령어 (cont`d)

## □ 정규 표현식과 **grep**의 사용 예제 (cont`d)

■ `[root@localhost ~]$ grep -w 'north' datafile`

- 패턴이 다른 단어의 일부가 아닌 하나의 단어가 되는 경우만 찾음.  
northwest나 northeast 등의 단어가 아니라, north라는 단어가 포함된  
행만 출력

```
[root@localhost test]# cat test.txt
northwest
north
northeast
[root@localhost test]# grep 'north' test.txt
northwest
north
northeast
[root@localhost test]# grep -w 'north' test.txt
north
[root@localhost test]# _
```