OS-보충

학습 목표

- OS 명령어를 사용 할 수 있다.

리눅스 기본 관리
: 쉘과 프롬프트

- 콘솔 또는 원격 터미널을 통해 로그인하면 커맨드 프롬프트가 화면에 표시

```
[bituser@localhost ~] $ _
```

- 이를 커맨트 프롬프트(Prompt) 또는 쉘 프롬프트, 줄여서 프롬프트라 함
  - @ 앞 : 접속 계정
  - @ 뒤 : 접속한 시스템의 호스트 이름
  - 호스트명 뒤 : 현재 위치(경로) 표시. 틸트(~) 문자는 접속 계정 홈 디렉터리 (/home/bituser)의 shortcut
  - $: 프롬프트. 뒤에 커서가 깜빡이며 명령어 입력과 실행(엔터)을 기다림

리눅스 기본 관리
: 쉘과 프롬프트

- [실습] 계정 홈 디렉터리 다루기

```
[bituser@localhost ~]$ pwd
/home/bituser
[bituser@localhost ~]$ cd /
[bituser@localhost /]$ pwd
/
[bituser@localhost /]$ cd ~
[bituser@localhost ~]$ pwd
/home/bituser
[bituser@localhost ~]$
```

- Command 소개
  - pwd (Print Working Directory) : 현재 디렉터리 경로를 출력
  - cd (Change Directory) : 지정한 디렉터리르 이동. 경로를 지정하지 않거나 틸트(~)를 경로로 입력하면 사용자의 home 디렉터리로 이동
  - clear : 현재 사용중인 터미널 화면을 깨끗하게 지워줌

쉘 변수와 환경 변수

- 쉘 변수 (지역 변수)
  - 기본적으로 프로세스의 환경 변수와 관련 없음
  - (이름, 값)의 데이터베이스
  - 쉘 프로그래밍이나 쉘 제어에 사용
  - 보통 소문자 이름

```
[bituser@localhost ~]$ a=123
[bituser@localhost ~]$  x=Hello
[bituser@localhost ~]$  echo $x
```

쉘 변수와 환경 변수

- 환경 변수
  - 유닉스 프로세스 속성 중 하나
  - (이름, 값)의 데이터베이스
  - 쉘에서 제어 가능
  - 보통 대문자 이름
  - 중요 환경 변수
    - HOME
    - PATH
    - TERM
    - MAIL
    - SHELL
    - USER

리눅스 기본 관리
: 명령어 매뉴얼 보기 - man

- 리눅스에는 많은 명령어가 있으며, 각 명령어는 다양한 옵션을 가지고 있다
  - 이 명령어들과 옵션을 다 외우려 하는 것은 무리
- man : 리눅스에 포함된 체계화된 도움말
  - 사용법
    - man {명령어}
  - 페이징 관련 키보드
    - 상하 커서키 or J, K : 위쪽 행과 아래쪽 행으로 이동
    - PgUp(or B), PgDown(or SPACE) : 페이지 단위의 이동

- 대부분의 명령어에 –-help 를 붙여 실행하면 도움말을 보여준다

리눅스 기본 관리
: 사용자 관리

- 리눅스의 사용자 관리
  - 다중 사용자 시스템 : 여러 사용자가 동시 접속하여 사용할 수 있는 시스템
  - root 계정
    - 시스템 관리를 책임지는 계정
    - 시스템 변경의 모든 권한을 갖고 있어 편리하지만 보안상 이유로 꼭 필요할 때만 root로 로그인
  - 모든 사용자는 하나 이상의 그룹에 소속됨

리눅스 기본 관리
: 사용자 관리 – 사용자 추가(useradd)

- useradd : 계정을 추가할 때 사용하는 명령
- 사용법
  - useradd [옵션] 계정명

- 옵션
  - -o : 홈 디렉터리를 지정할 때 사용
  - -M : 홈 디렉터리 없이 계정을 추가할 때 사용
  - -g : 그룹을 지정할 때 사용
  - -G : 기본 그룹 이외에 추가로 지정할 그룹이 있는 경우 사용
  - -c : 계정 추가시 계정에 대한 설명을 설정
  - -s : 계정 추가시 이 계정으로 로그인한 사용자가 사용할 쉘을 지정
  - -D : /etc/default/useradd 파일에 설정되어 있는 useradd 명령의 기본 설정 내용을 표시

리눅스 기본 관리
: 사용자 관리 – 사용자 추가(useradd)

- [실습] user1 계정 추가

  - user1 계정이 추가, user1 그룹이 생성되어 user1의 그룹으로 설정됨
  - 계정 홈 디렉터리는 /home/user1 에 생성

```
[root@localhost ~]# useradd user1
```

- [실습] wheel 그룹에 속하는 user2 계정 추가

  - user2 계정이 추가, wheel 그룹에 user2를 소속시킴
  - -g 옵션으로 부여하는 그룹은 미리 만들어져 있어야 함

```
[root@localhost ~]# useradd –g wheel user2
```

리눅스 기본 관리
: 사용자 관리 – 현재 시스템의 사용자 확인

- 추가된 사용자 목록은 /etc/passwd 파일 내에서 관리

```
[root@localhost home]# cat /etc/passwd
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
...
...
...

sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
bituser:x:1000:10::/home/bituser:/bin/bash
user1:x:1001:1001::/home/user1:/bin/bash
user2:x:1002:100::/home/user2:/bin/bash
```

- 각 라인은 하나의 계정에 대한 정보를 다음과 같은 형식으로 보관

- 비밀번호는 보안상 이유로 x로 표시하고 /etc/shadow 파일에 암호회되어 관리

```
ID:비밀번호:UID:GID:설명:홈디렉터리:로그인 쉘
```

리눅스 기본 관리
: 사용자 관리

- 계정 추가시 생성된 홈 디렉터리의 내용

```
[root@localhost home]# ls -la /home/user1
합계 12
drwx------.  2 user1 user1   59   2월 13 13:59 .
drwxr-xr-x. 5 root   root     45   2월 13 14:11 ..
-rw-r--r--.  1 user1 user1    18 11월 20 14:02 .bash_logout
-rw-r--r--.  1 user1 user1  193 11월 20 14:02 .bash_profile
-rw-r--r--.  1 user1 user1  231 11월 20 14:02 .bashrc

[root@localhost home]#
```

  - /etc/skel 디렉터리의 내용을 복사하여 계정 홈 디렉터리를 생성함

- [실습] 홈 디렉터리 없이 user3 계정 추가하기

```
[root@localhost ~]# useradd –M user3
```

리눅스 기본 관리
: 사용자 관리 – 비밀번호 설정 (passwd)

- passwd : 사용자의 비밀번호를 생성/변경할 때 사용하는 명령
- 사용법
  - passwd 계정
- 비밀번호는 /etc/shadow 파일에 암호화되어 저장

```
[root@localhost mail]# cat /etc/shadow
root:$6$Vmxj9y1oEZzlg3DZ$Ew7ehXEubX7HyHzojlod9dVIBkyuk5M7m6x0gSsskOu.iUr4.vB5
6mcIoCJY/2ZkFD5kL83SYma4LJmU9fixn/::0:99999:7:::
bin:*:16659:0:99999:7:::
...
...
...
bituser:$6$9T6P43zD$wXfg9F2AUk9hejQV9aMIghm1PH2BOZAYAD8pIPa8a/jrOQ3ZVdUpUE
M3VzxOWkj9GuucX9CGEBJNvD1G3I4XE1:16842:0:99999:7:::
user1:!!:16844:0:99999:7:::
user2:!!:16844:0:99999:7:::
```

  - 비밀번호가 설정되어 있지 않으면 !!로 표시. 실제 로그인도 되지 않음

리눅스 기본 관리
: 사용자 관리 – 비밀번호 설정 (passwd)

- [실습] user1 계정에 비밀번호를 설정해 봅시다

```
[root@localhost mail]# passwd user1
user1 사용자의 비밀 번호 변경 중
새  암호:
새  암호 재입력:
passwd: 모든 인증 토큰이 성공적으로 업데이트 되었습니다.
[root@localhost mail]#
```

  - 비밀번호가 설정된 user1과 설정되지 않은 user2로 로그인 테스트를 해 봅니다

리눅스 기본 관리
: 사용자 관리 – 사용자 삭제(userdel)

- userdel : 사용자를 삭제할 때 사용하는 명령
- 사용법
  - userdel [옵션] 계정

- 옵션
  - -r : 사용자의 홈 디렉터리를 함께 삭제

리눅스 기본 관리
: 사용자 관리 – 사용자 삭제(userdel)

- [실습] user1 계정을 삭제해 봅니다

```
[root@localhost ~]# userdel user1
[root@localhost ~]# ls -l /home
합계 4
drwx------. 2  bituser   wheel  4096  2월 12 22:48 bituser
drwx------. 2      1001    1001     59  2월 13 13:59 user1
drwx------. 2     user2    users     59  2월 13 14:11 user2
[root@localhost ~]#
```

  - 계정과 그룹이 삭제되었으나 계정 홈 디렉터리는 삭제되지 않음
- [실습] user2 계정을 홈 디렉터리와 함께 삭제해 봅니다

```
[root@localhost ~]# userdel -r user2
[root@localhost ~]# ls -l /home
합계 4
drwx------. 2 bituser wheel 4096  2월 12 22:48 bituser
drwx------. 2    1001  1001   59  2월 13 13:59 user1
[root@localhost ~]#
```

리눅스 기본 관리
: 사용자 관리 – 사용자 정보 변경 (usermod)

- usermod : 사용자 정보 수정
  - 사용법 : usermod [옵션] 계정
  - 옵션
    - -c : 사용자의 설명을 수정
    - -d : 홈디렉토리를 변경
    - -m : 홈디렉토리 변경시 파일을 옮긴다.
    - -e : 계정종료일 변경
    - -s : 기본 쉘 변경
    - -u : UID변경
    - -g : 기본 그룹 변경
    - -G : 추가 그룹 변경
    - -l : 사용자명 변경
    - -L : 사용자 패스워드 LOCK (로그인 불가)
    - -U : 패스워드 LOCK을 푼다.

리눅스 기본 관리
: 사용자 관리 – 그룹 추가 (groupadd)

- groupadd : 새로운 그룹을 생성하는 명령
- 사용법
  - groupadd [옵션] 그룹명

- 옵션
  - -g GID : 특정 GID 번호로 그룹을 생성
  - -r : 0 ~ 1000번대 사이로 GID를 자동으로 생성

- group에 관련된 정보는 /etc/group 파일에 저장되어 관리됨

리눅스 기본 관리
: 사용자 관리 – 그룹 추가 (groupadd)

- [실습] group1이라는 그룹을 생성해 봅니다

```
[root@localhost ~]# groupadd group1
[root@localhost ~]# cat /etc/group | grep group
group1:x:1004:
[root@localhost ~]#
```

- [실습] –r 옵션을 이용, 1000미만의 수를 지정하도록 그룹 group2를 생성해 봅니다

```
[root@localhost ~]# groupadd -r group2
[root@localhost ~]# cat /etc/group | grep group
group1:x:1004:
group2:x:994:
[root@localhost ~]#
```

- [실습] –g 옵션을 이용하여 GID를 직접 지정한 그룹 group3을 생성해 봅니다

```
[root@localhost ~]# groupadd -g 1100 group3
[root@localhost ~]# cat /etc/group | grep group
group1:x:1004:
group2:x:994:
group3:x:1100:
[root@localhost ~]#
```

리눅스 기본 관리
: 사용자 관리 – 그룹 삭제 (groupdel)

- groupdel : 그룹을 삭제하는 명령. 삭제하려는 그룹에 속한 계정이 있다면 삭제되지 않음
- 사용법
  - groupdel 그룹명

- [실습] group3 그룹을 삭제해 봅니다

```
[root@localhost ~]# groupdel group3
[root@localhost ~]# cat /etc/group | grep group
group1:x:1004:
group2:x:994:
[root@localhost ~]#
```

  - 같은 방법으로 group2, group1 그룹도 삭제해 봅니다

리눅스 기본 관리
: 사용자 관리 – 사용자 전환 (su)

- su : 다른 사용자 계정으로 쉘을 실행함. 로그아웃 없이 다른 사용자로 전환 가능
- 사용법
  - su [옵션] 계정

- 옵션
  - - : 사용자의 환경 변수를 읽어 옴

- [실습] root 계정으로 전환해 봅시다

```
[bituser@localhost ~]$ su -
암호:
마지막 로그인: 토  6월 13 15:39:46 KST 2016 일시 pts/0
[root@localhost ~]#
```

  - root 계정에서 다른 계정으로 전환할 때는 비밀번호를 묻지 않음

리눅스 기본 관리
: 디렉터리 관리 – 현재 경로 확인 (pwd), 파일 목록 표시(ls)

- pwd : 현재 위치한 디렉터리의 절대 경로를 출력

```
[root@localhost ~]# pwd
/root
[root@localhost ~]#
```

- ls : 디렉터리 내에 있는 파일의 목록을 표시
  - 사용법 : ls [옵션] [디렉터리]
  - 옵션
    - -l : 파일에 대한 권한, 생성 시간 등 보다 자세한 내용을 출력 (long)
    - -a : 숨긴 파일이나 디렉터리 등 현재 디렉터리의 모든 내용을 출력 (all)
    - -h : 파일 크기를 k, m, g 등 사람이 읽기 편한 단위르 출력 (human readable)
    - -F : 실행파일이나 디렉터리 등이 쉽게 구분될 수 있도록 출력
    - -R : 하위 디렉터리의 내용들도 함께 출력

리눅스 기본 관리

: 디렉터리 관리 – 현재 경로 확인 (pwd), 파일 목록 표시(ls)

- [실습] 간단히 현재 디렉터리의 내용을 살펴봅니다

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# ls
anaconda-ks.cfg  mongodb-linux-x86_64-rhel70-3.2.1.tgz
[root@localhost ~]#
```

- [실습] –al 옵션을 이용, 숨겨진 파일을 포함하여 모든 파일의 상세 정보를 나열해 봅니다

```
[root@localhost ~]# ls -al
합계 62120
dr-xr-x---.  2 root root     4096  2월 12 22:28 .
dr-xr-xr-x. 18 root root     4096  2월 12 22:29 ..
-rw-------.  1 root root     8867  2월 13 17:51 .bash_history
-rw-r--r--.  1 root root       18 12월 29  2013 .bash_logout
-rw-r--r--.  1 root root      176 12월 29  2013 .bash_profile
-rw-r--r--.  1 root root      176 12월 29  2013 .bashrc
-rw-r--r--.  1 root root      100 12월 29  2013 .cshrc
-rw-r--r--.  1 root root      129 12월 29  2013 .tcshrc
-rw-------.  1 root root     1066  2월 11 17:32 anaconda-ks.cfg
-rw-r--r--.  1 root root 63563953  1월 12 04:07 mongodb-linux-x86_64-rhel70-3.2.1.tgz
[root@localhost ~]#
```

리눅스 기본 관리

: 디렉터리 관리 – 현재 경로 확인 (pwd), 파일 목록 표시(ls)

- [실습] ll을 입력해 봅니다.

```
[root@localhost ~]# ll
합계 62080
-rw-------. 1 root root     1066  2월 11 17:32 anaconda-ks.cfg
-rw-r--r--. 1 root root 63563953  1월 12 04:07 mongodb-linux-x86_64-rhel70-3.2.1.tgz
[root@localhost ~]#
```

```
[root@localhost ~]# alias
alias cp='cp -i'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mv='mv -i'
alias rm='rm -i'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
[root@localhost ~]#
```

- [실습] /var/log 디렉터리의 내용을 살펴보고 어떤 내용들이 있는지 확인해 봅시다

리눅스 기본 관리
: 디렉터리 관리 – 디렉터리 생성 (mkdir)

- mkdir : 디렉터리를 생성
  - 사용법 : mkdir [옵션] 디렉터리명
  - 옵션
    - -m : 디렉터리 권한을 지정할 수 있음 (기본값 755)
    - -p : 상위 디렉터리가 존재하지 않으면 상위 디렉터리도 함께 생성

- [실습] 현재 경로에서 dir1 디렉터리를 생성해 봅니다

```
[bituser@localhost ~]$ pwd
/home/bituser

[bituser@localhost ~]$ mkdir dir1

[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel 6  2월 13 18:10 dir1

[bituser@localhost ~]$
```

리눅스 기본 관리
: 디렉터리 관리 – 디렉터리 생성 (mkdir)

- [실습] 현재 경로에서 dir2/subdir1 디렉터리를 –p 옵션을 이용, 한번에 생성해 봅니다

```
[bituser@localhost ~]$ pwd
/home/bituser

[bituser@localhost ~]$ mkdir -p dir2/subdir1

[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel  6  2월 13 18:10 dir1
drwxr-xr-x. 3 bituser wheel 20  2월 13 18:28 dir2

[bituser@localhost ~]$ cd dir2

[bituser@localhost dir2]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel 6  2월 13 18:28 subdir1

[bituser@localhost dir2]$
```

리눅스 기본 관리
: 디렉터리 관리 – 디렉터리 삭제 (rmdir)

- rmdir : 비어 있는 디렉터리를 삭제
  - 사용법 : rmdir [옵션] 디렉터리명
  - 옵션
    - -p : 상위 디렉터리도 삭제. 이때, 상위 디렉터리도 비어 있어야 함

- [실습] dir1 디렉터리를 지워봅니다

```
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel  6  2월 13 18:10 dir1
drwxr-xr-x. 3 bituser wheel 20  2월 13 18:28 dir2

[bituser@localhost ~]$ rmdir dir1

[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 3 bituser wheel 20  2월 13 18:28 dir2
[bituser@localhost ~]$
```

리눅스 기본 관리
: 디렉터리 관리 – 디렉터리 삭제 (rmdir)

- [실습] dir2 디렉터리를 지워봅니다

```
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 3 bituser wheel 20  2월 13 18:28 dir2

[bituser@localhost ~]$ rmdir dir2
rmdir: failed to remove `dir2': 디렉터리가 비어있지 않음

[bituser@localhost ~]$
```

- dir2 아래에는 subdir1 디렉터리가 있기 때문에 삭제가 되지 않음
- 이럴 경우, -p 옵션을 사용하여 subdir1을 삭제하면 상위 디렉터리인 dir2도 함께 삭제

```
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 3 bituser wheel 20  2월 13 18:28 dir2

[bituser@localhost ~]$ rmdir -p dir2/subdir1

[bituser@localhost ~]$ ls -l
합계 0
[bituser@localhost ~]$
```

리눅스 기본 관리
: 디렉터리 관리 – 디렉터리 이동(cd)

- cd : 디렉터리를 이동할 때 사용
  - 사용방법
    - cd [디렉터리]
  - 디렉터리 명이 생략되면 접속 계정의 홈 디렉터리로 이동 (= cd ~)
  - 디렉터리 경로는 상대경로와 절대 경로로 나타낼 수 있음
    - 절대 경로 : 루트 디렉터리(/) 부터 모든 경로를 표시하는 방법
    - 상대 경로 : 현재 디렉터리를 기준으로 특정 디렉터리의 경로를 표시
      유닉스에서는 상대 경로 표시를 위해 . (현재 디렉터리)와 .. (부모 디렉터리의 심볼을 제공
- [실습] 루트 디렉터리(/)로 이동해 봅니다

```
[root@localhost ~]# pwd
/root

[root@localhost ~]# cd /

[root@localhost /]# pwd
/

[root@localhost /]#
```

리눅스 기본 관리
: 파일 관리 – 파일 정보의 이해

- 파일의 종료
  - 유닉스에서 프로그램들은 주변의 장치(device)를 파일로 인식하기 때문에 여러 종류의 파일이 존재
  - ls 명령 결과를 보면, 각각의 파일들을 타입별로 구분해 놓은 정보가 있음
- 파일의 구분
  - - : 일반 파일
  - b : 블록 디바이스 파일
  - c : 문자열 디바이스 파일
  - d : 디렉터리
  - l : 심볼릭 링크
  - p 또는 = : 명명된 파이프(named pipe) / FIFO
  - s : 소켓(socket)

리눅스 기본 관리
: 파일 관리 – 파일 정보의 이해

- [실습] ls 명령을 이용, 파일의 타입을 살펴봅니다

```
[bituser@localhost ~]$ ls -la
합계 28
drwx------. 2 bituser wheel 4096  2월 14 09:26 .
drwxr-xr-x. 4 root    root    33  2월 13 16:07 ..
-rw-------. 1 bituser wheel 5871  2월 13 19:40 .bash_history
-rw-r--r--. 1 bituser wheel   18 11월 20 14:02 .bash_logout
-rw-r--r--. 1 bituser wheel  193 11월 20 14:02 .bash_profile
-rw-r--r--. 1 bituser wheel  231 11월 20 14:02 .bashrc
-rw-r--r--. 1 bituser wheel   44  2월 12 22:48 .dbshell
-rw-------. 1 bituser wheel    0  2월 12 22:35 .mongorc.js
-rw-r--r--. 1 bituser wheel    0  2월 14 09:27 test
[bituser@localhost ~]$ ls -l /dev
합계 0
crw-------. 1 root root     10, 235  2월 11 21:37 autofs
drwxr-xr-x. 2 root root          140  2월 11 21:37 block
drwxr-xr-x. 2 root root           80  2월 11 21:37 bsg
crw-------. 1 root root     10, 234  2월 11 21:37 btrfs-control
drwxr-xr-x. 3 root root           60  2월 12 00:04 bus
lrwxrwxrwx. 1 root root            3  2월 12 03:47 cdrom -> sr0
drwxr-xr-x. 2 root root        2640  2월 11 21:37 char
crw-------. 1 root root      5,   1  2월 11 21:37 console
lrwxrwxrwx. 1 root root           11  2월 11 21:36 core -> /proc/kcore
drwxr-xr-x. 3 root root           80  2월 12 00:04 cpu
crw-------. 1 root root     10,  61  2월 11 21:37 cpu_dma_latency
crw-------. 1 root root     10,  62  2월 11 21:37 crash
drwxr-xr-x. 6 root root          120  2월 12 03:47 disk
lrwxrwxrwx. 1 root root           13  2월 11 21:36 fd -> /proc/self/fd
crw-rw-rw-. 1 root root      1,   7  2월 11 21:37 full
crw-rw-rw-. 1 root root     10, 229  2월 11 21:37 fuse
crw-------. 1 root root     10, 228  2월 11 21:37 hpet
drwxr-xr-x. 2 root root            0  2월 11 21:37 hugepages
lrwxrwxrwx. 1 root root           25  2월 11 21:37 initctl -> /run/systemd/initctl/fifo
drwxr-xr-x. 3 root root          220  2월 11 21:37 input
crw-r--r--. 1 root root      1,  11  2월 11 21:37 kmsg
srw-rw-rw-. 1 root root            0  2월 11 21:36 log
```

리눅스 기본 관리
: 파일 관리 – 파일의 소유자와 그룹

- 파일 소유자와 그룹
  - 유닉스의 모든 파일(디렉터리 포함)에는 소유자와 그룹이 있음
  - 대부분 파일을 처음 생성한 계정과 그 계정이 속한 그룹이 그 파일의 소유자와 그룹이 되지만 chown 명령을 이용, 소유 계정과 그룹을 변경할 수 있음

- [실습] ls 명령을 사용해서 파일의 소유 계정과 그룹을 확인해 봅니다

```
[bituser@localhost ~]$ ls -la
합계 28
drwx------.  2 bituser wheel  4096   2월 14 09:26 .
drwxr-xr-x. 4 root       root       33   2월 13 16:07 ..
-rw-------.  1 bituser wheel  5871   2월 13 19:40 .bash_history
-rw-r--r--.  1 bituser wheel    18 11월 20 14:02 .bash_logout
-rw-r--r--.  1 bituser wheel   193 11월 20 14:02 .bash_profile
-rw-r--r--.  1 bituser wheel   231 11월 20 14:02 .bashrc
-rw-r--r--.  1 bituser wheel    44   2월 12 22:48 .dbshell
-rw-------.  1 bituser wheel     0   2월 12 22:35 .mongorc.js
-rw-r--r--.  1 bituser wheel     0   2월 14 09:27 test
[bituser@localhost ~]$
```

리눅스 기본 관리
: 파일 관리 – 파일 권한

- Permission
  - 누가 파일에 접근해도 되는지, 접근해서 무엇을 할 수 있는지를 결정하는 것
  - ls 명령의 결과를 보면 개별 파일의 권한을 알 수 있음

```
[bituser@localhost ~]$ ls -la
합계 28
drwx------. 2 bituser wheel  4096  2월 14 09:26 .
drwxr-xr-x. 4 root    root     33  2월 13 16:07 ..
-rw-------. 1 bituser wheel  5871  2월 13 19:40 .bash_history
-rw-r--r--. 1 bituser wheel    18 11월 20 14:02 .bash_logout
-rw-r--r--. 1 bituser wheel   193 11월 20 14:02 .bash_profile
-rw-r--r--. 1 bituser wheel   231 11월 20 14:02 .bashrc
-rw-r--r--. 1 bituser wheel    44  2월 12 22:48 .dbshell
-rw-------. 1 bituser wheel     0  2월 12 22:35 .mongorc.js
-rw-r--r--. 1 bituser wheel     0  2월 14 09:27 test
[bituser@localhost ~]$
```

- 파일 타입 뒤 3개의 'rws'가 각각 파일에 대한 소유 계정, 소유 그룹, 모든 사용자에 대한 권한을 표시 (r: 읽기권한, w: 쓰기 권한, x: 실행 권한)

-	rwx	rwx	rwx
파일타입	user 권한	group 권한	other 권한

리눅스 기본 관리
: 파일 관리 – 파일 권한

- [실습] 다음 ls 명령 결과로 권한을 해석해 봅시다

```
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel 6  2월 14 10:24 dowork
-rw-r--r--. 1 bituser wheel 0  2월 14 09:27 test
[bituser@localhost ~]$
```

  - dowork 디렉터리는 rwx r-x r-x 권한을 가지고 있음
    - 파일 소유 계정 bituser는 디렉터리에 대해 읽기, 쓰기, 실행 권한을 가지고 있음
    - 파일 소유 그룹 wheel은 디렉터리에 읽기, 실행 권한을 가지고 있음
    - 다른 사용자는 디렉터리에 읽기, 실행 권한을 가지고 있음
- [실습] test 파일에 대한 권한도 해석해 봅니다

리눅스 기본 관리
: 파일 관리 – 파일 권한

```
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel 6  2월 14 10:24 dowork
-rw-r--r--. 1 bituser wheel 0  2월 14 09:27 test
[bituser@localhost ~]$
```

- 권한은 숫자로 표시할 수 있음 (r -> 4, w -> 2, x -> 1)
- 따라서 dowork 디렉터리는 4+2+1, 4+1, 4+1 즉 755로 표현

- [실습] text 파일의 권한을 숫자로 표시해 봅시다

리눅스 기본 관리
: 파일 관리 – 파일의 시간 정보 변경(touch)

- touch : 파일의 시간 정보를 변경하는 명령
  - 크기가 0인 파일을 생성하는 용도로 자주 사용
  - 사용법
    - touch [옵션] 파일명
  - 옵션
    - -c : 현재 시간으로 파일 시간을 변경 (파일이 없으면 생성하지 않음)
    - -d 시간 : 지정한 시간으로 파일 시간을 변경 (예: touch '2018-03-04 12:00:30' test
    - -r 파일 : 지정한 파일의 시간으로 파일 시간을 변경
    - -t MMDDHHMM[[CC]YY][.SS] : 지정한 시간으로 파일 시간을 변경
- [실습] 0byte 크기의 파일 test를 생성해 봅니다

```
[bituser@localhost ~]$ touch test
[bituser@localhost ~]$ ls -l
합계 0
-rw-r--r--. 1 bituser wheel 0  2월 13 23:55 test
[bituser@localhost ~]$
```

리눅스 기본 관리
: 파일 관리 – 파일의 시간 정보 변경(touch)

- [실습] –d 옵션을 사용하여 파일 시간을 변경해 봅니다

```
[bituser@localhost ~]$ touch -d '2016-01-01 10:00:30' test
[bituser@localhost ~]$ ls -l
합계 0
-rw-r--r--. 1 bituser wheel 0  1월  1 10:00 test
[bituser@localhost ~]$
```

- [실습] –t 옵션을 사용하여 파일 시간을 변경해 봅니다

```
[bituser@localhost ~]$ touch -t 201602011000.30 test
[bituser@localhost ~]$ ls -l
합계 0
-rw-r--r--. 1 bituser wheel 0  2월  1 10:00 test
[bituser@localhost ~]$
```

- [실습] 현재 시간으로 파일 시간을 변경해 봅니다

```
[bituser@localhost ~]$ date
2016. 02. 14. (일) 09:26:57 KST
[bituser@localhost ~]$ touch test
[bituser@localhost ~]$ ls -l
합계 0
-rw-r--r--. 1 bituser wheel 0  2월 14 09:27 test
[bituser@localhost ~]$
```

리눅스 기본 관리
: 파일 관리 – 파일의 복사(cp)

- cp : 파일을 복사하는 명령
  - 사용법
    - cp [옵션] 원본 사본
  - 옵션
    - -a : 원본 파일의 속성, 링크 정보들을 그대로 유지하면서 복사
    - -i : 만약 복사 대상에 같은 이름의 파일이 있으면 사용자에게 물어봄
    - -f : 만약 복사 대상에 같은 이름의 파일이 있으면 강제로 지우가 복사
    - -R : 디렉터리를 복사할 때, 내부에 포함된 모든 하위 디렉터리와 파일들을 모두 복사
- [실습] test 파일을 test.bak으로 복사해 봅니다

```
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel 6  2월 14 10:24 dowork
-rw-r--r--. 1 bituser wheel 0  2월 14 09:27 test
[bituser@localhost ~]$ cp test test.bak
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel 6  2월 14 10:24 dowork
-rw-r--r--. 1 bituser wheel 0  2월 14 09:27 test
-rw-r--r--. 1 bituser wheel 0  2월 14 10:43 test.bak
[bituser@localhost ~]$
```

리눅스 기본 관리
: 파일 관리 – 파일의 복사(cp)

- [실습] test 파일을 dowork 디렉터리로 복사해 봅니다

```
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel 6  2월 14 10:24 dowork
-rw-r--r--. 1 bituser wheel 0  2월 14 09:27 test
-rw-r--r--. 1 bituser wheel 0  2월 14 10:43 test.bak
[bituser@localhost ~]$ cp test dowork/
[bituser@localhost ~]$ ls -l dowork
합계 0
-rw-r--r--. 1 bituser wheel 0  2월 14 10:45 test
[bituser@localhost ~]$
```

- [실습] dowork 디렉터리를 dowork.bak 디렉터리로 복사해 봅니다
- [실습] –i 옵션을 이용, 이미 있는 이름으로 복사해 봅니다

```
[[bituser@localhost ~]$ cp -i test test.bak
cp: overwrite `test.bak'? n
[bituser@localhost ~]$ cp -i test test.bak
cp: overwrite `test.bak'? y
[bituser@localhost ~]$
```

- [tip] 안전을 위해 alias로 cp='cp –i'로 지정해 두는 것도 좋음

리눅스 기본 관리
: 파일 관리 – 파일의 복사(cp)

- [실습] alias로 cp = 'cp –i'로 지정해 봅니다
  - 로그인시 실행되는 스크립트 파일 중 보통 alias 관련 설정은 /etc/bashrc 파일에 지정

```
[bituser@localhost ~]$ su -
암호:
[root@localhost ~]# vi /etc/bashrc
fi
# vim:ts=4:sw=4

alias cp='cp -i'


:wq


[root@localhost ~]# exit
logout
[bituser@localhost ~]$ source /etc/bashrc
```

- [실습] alias로 설정된 -i 옵션이 불편하거나 의도적 덮어쓰기를 할 경우 –f 옵션으로 수행

```
[bituser@localhost ~]$ \ cp -f test test.bak
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel 17  2월 14 10:45 dowork
drwxr-xr-x. 2 bituser wheel 17  2월 14 10:47 dowork.bak
-rw-r--r--. 1 bituser wheel  0  2월 14 09:27 test
-rw-r--r--. 1 bituser wheel  0  2월 14 16:45 test.bak
[bituser@localhost ~]$
```

리눅스 기본 관리
: 파일 관리 – 파일의 이동(mv)

- mv : 파일을 이동하는 명령
  - 유닉스에는 rename 명령어가 따로 없음. 즉, 파일명을 바꿀 때도 mv를 이용
  - 사용법
    - mv [옵션] 원본 목적지
  - 옵션
    - -b : 덮어쓰게 되는 경우 백업 파일을 생성
    - -i : 덮어쓰게 될 경우 사용자에게 물어봄
    - -f : 덮어쓸게 될 경우 사용자에게 물어보지 않음
- [실습] test 파일을 test2 파일로 바꿔 봅니다

```
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel 17  2월 14 10:45 dowork
drwxr-xr-x. 2 bituser wheel 17  2월 14 10:47 dowork.bak
-rw-r--r--. 1 bituser wheel  0  2월 14 09:27 test
-rw-r--r--. 1 bituser wheel  0  2월 14 16:45 test.bak
[bituser@localhost ~]$ mv test test2
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel 17  2월 14 10:45 dowork
drwxr-xr-x. 2 bituser wheel 17  2월 14 10:47 dowork.bak
-rw-r--r--. 1 bituser wheel  0  2월 14 16:45 test.bak
-rw-r--r--. 1 bituser wheel  0  2월 14 09:27 test2
[bituser@localhost ~]$
```

리눅스 기본 관리
: 파일 관리 – 파일의 이동(mv)

- [실습] test2 파일을 dowork/test로 이동해 봅니다
  - dowork/test 파일은 이미 있으므로 –b 옵션을 사용, 백업파일을 남겨봅시다

```
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel 17  2월 14 10:45 dowork
drwxr-xr-x. 2 bituser wheel 17  2월 14 10:47 dowork.bak
-rw-r--r--. 1 bituser wheel  0  2월 14 16:45 test.bak
-rw-r--r--. 1 bituser wheel  0  2월 14 09:27 test2
[bituser@localhost ~]$ ls -l dowork
합계 0
-rw-r--r--. 1 bituser wheel 0  2월 14 10:45 test
[bituser@localhost ~]$ mv -b test2 dowork/test
[bituser@localhost ~]$ ls -l dowork
합계 0
-rw-r--r--. 1 bituser wheel 0  2월 14 09:27 test
-rw-r--r--. 1 bituser wheel 0  2월 14 10:45 test~
```

- [실습] 덮어 쓸 경우 사용자에게 물어보는 옵션 –i를 사용해 봅시다

```
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel 29  2월 14 18:12 dowork
drwxr-xr-x. 2 bituser wheel 17  2월 14 10:47 dowork.bak
-rw-r--r--. 1 bituser wheel  0  2월 14 16:45 test
[bituser@localhost ~]$ mv -i test dowork
mv: overwrite `dowork/test'? n
[bituser@localhost ~]$
```

리눅스 기본 관리
: 파일 관리 – 파일의 이동(mv)

- [실습] alias에 mv='mv –i'를 추가하고 안전하게 mv를 사용해 봅니다

```
[bituser@localhost ~]$ alias
alias cp='cp -i'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mv='mv -i'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
[bituser@localhost ~]$ mv test dowork
mv: overwrite `dowork/test'?
```

- [실습] 의도적으로 덮어쓰기를 하는 경우 옵션 –f를 사용, 메시지를 안나오게 할 수 있음

```
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel 29  2월 14 18:12 dowork
drwxr-xr-x. 2 bituser wheel 17  2월 14 10:47 dowork.bak
-rw-r--r--. 1 bituser wheel  0  2월 14 16:45 test
[bituser@localhost ~]$ mv -f test dowork
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel 29  2월 14 18:27 dowork
drwxr-xr-x. 2 bituser wheel 17  2월 14 10:47 dowork.bak
[bituser@localhost ~]$
```

리눅스 기본 관리
: 파일 관리 – 파일의 삭제(rm)

- rm : 파일을 삭제하는 명령
  파일을 삭제하면 복구가 불가능하기 때문에 파일 삭제에는 항상 주의
  - 사용법
    - rm [옵션] 파일명
  - 옵션
    - -r, -R : 디렉터리인 경우, 하위 디렉터리와 파일을 모두 삭제
    - -i : 파일을 삭제할 것인지 사용자에게 물어봄
    - -f : -i 옵션을 무시하고 강제로 삭제
- [실습] dowork 디렉터리 안의 test~ 파일을 삭제해 봅니다

```
[bituser@localhost ~]$ cd dowork
[bituser@localhost dowork]$ ls -l
합계 0
-rw-r--r--. 1 bituser wheel 0  2월 14 16:45 test
-rw-r--r--. 1 bituser wheel 0  2월 14 10:45 test~
[bituser@localhost dowork]$ rm test~
[bituser@localhost dowork]$ ls -l
합계 0
-rw-r--r--. 1 bituser wheel 0  2월 14 16:45 test
[bituser@localhost dowork]$
```

리눅스 기본 관리
: 파일 관리 – 파일의 삭제(rm)

- [실습] /etc/bashrc에 rm 명령을 'rm –i'로 alias 설정을 하고 삭제해 봅니다

```
[bituser@localhost dowork]$ su -
암호:
[root@localhost ~]#
[root@localhost ~]# vi /etc/bashrc
[root@localhost ~]# exit
logout
[bituser@localhost dowork]$ source /etc/bashrc
[bituser@localhost dowork]$ ls -l
합계 0
-rw-r--r--. 1 bituser wheel 0  2월 14 16:45 test
[bituser@localhost dowork]$ rm test
rm: remove 일반 빈 파일 `test'?
```

- [실습] 많은 파일을 지울 때 매번 확인 메시지가 나타나는 것을 의도적으로 피하려면 –f 옵션을 사용

```
[bituser@localhost dowork]$ ls -l
합계 0
-rw-r--r--. 1 bituser wheel 0  2월 14 16:45 test
[bituser@localhost dowork]$ rm -f test
[bituser@localhost dowork]$ ls -l
합계 0
[bituser@localhost dowork]$
```

리눅스 기본 관리
: 파일 관리 – 파일의 삭제(rm)

- [실습] 디렉터리 삭제
  - 디렉터리 삭제시 rmdir 보다 rm –rf를 사용하는 것이 일반적
  - 하지만 –f 옵션은 사용자에게 묻지 않고 삭제하기 때문에 주의가 필요

```
[bituser@localhost ~]$ ls -l dowork.bak/
합계 0
-rw-r--r--. 1 bituser wheel 0  2월 14 10:47 test
[bituser@localhost ~]$ rm -rf dowork.bak
[bituser@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 bituser wheel 6  2월 14 19:34 dowork
[bituser@localhost ~]$
```

리눅스 기본 관리
: 파일 관리 – 파일 내용 확인(cat)

- cat : 파일의 내용을 화면에 출력
  - 사용법
    - cat [옵션] 파일명
  - 옵션
    - -n : 줄 번호를 표시
    - -b : 빈 행은 제외하고 줄 번호를 표시
    - -E : 각 행마다 끝에 $ 문자를 표시
- [실습] 계정 내 .bashrc 파일의 내용을 확인해 봅니다

```
[bituser@localhost ~]$ cat .bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
[bituser@localhost ~]$
```

리눅스 기본 관리
: 파일 관리 – 파일 내용 확인(cat)

- [실습] .bashrc의 내용에 줄 번호를 표시해 출력해 봅시다

```
[bituser@localhost ~]$ cat -n .bashrc
     1	# .bashrc
     2
     3	# Source global definitions
     4	if [ -f /etc/bashrc ]; then
     5		. /etc/bashrc
     6	fi
     7
     8	# Uncomment the following line if you don't like systemctl's auto-paging feature:
     9	# export SYSTEMD_PAGER=
    10
    11	# User specific aliases and functions
[bituser@localhost ~]$
```

- [실습] 옵션 –b, -E를 사용하여 .bashrc의 내용을 확인해 봅시다

리눅스 기본 관리
: 파일 관리 – 화면 단위로 분할하여 파일 내용 확인(more)

- more : 화면 단위로 분할하여 파일 내용을 출력
  - 사용법
    - more [옵션] 파일명
  - 옵션
    - -d : 안내 메시지를 화면에 출력
    - -s : 연속되는 공백 행을 하나의 행으로 줄여서 출력
  - 페이징을 위한 키보드
    - SPACE : 다음 페이지로 이동
    - b : 이전 페이지로 이동
    - q : 종료
- [실습] /var/log/boot.log 내용을 more 명령으로 확인해 봅시다
  - 엔터와 스페이스 키를 눌러가며 내용을 스크롤해 봅시다
    사용을 하면서 무엇이 불편한지 생각해 봅시다

리눅스 기본 관리
: 파일 관리 – 화면 단위로 분할하여 파일 내용 확인(less)

- less : more와 용도는 거의 동일하지만 기능이 좀 더 확장
  - 사용법
    - less [옵션] 파일명
  - 페이징을 위한 키보드
    - more에서 사용하는 키 입력
    - 화살표 키, PgUp, PgDown, J, K 등
- [실습] /var/log/boot.log 내용을 less 명령으로 확인해 봅시다
  - 엔터와 스페이스 키를 눌러가며 내용을 스크롤해 봅시다 more에 비해 편리해진 기능을 확인해 봅니다
- [실습] less --help 명령으로 사용할 수 있는 키 조합을 확인해 봅니다

리눅스 기본 관리
: 파일 관리 – 파일 내용 일부 확인 (head, tail)

- 텍스트 형식으로 작성된 파일의 앞 또는 마지막 일부만 보여준다 (기본값: 10행)
  - 사용법
    - head [옵션] 파일명 : 파일의 앞 부분을 보여줌
    - tail [옵션] 파일명 : 파일의 마지막 부분을 보여줌
  - 옵션
    - -{라인수} : 보여줄 라인 수를 지정한다
    - -f (tail) : 파일의 변경 내용을 추적한다
- [실습] /var/log/secure 내용을 head, tail 명령으로 각각 확인해 봅시다
- [실습] /var/log/secure 내용을 tail –f 명령으로 변경 추적해 봅니다

리눅스 기본 관리
: 파일 관리 –파일 찾기 (find)

- find : 다양한 조건에 맞는 파일을 찾음
  - 사용법
    - find [시작디렉터리] [조건]
  - 설명
    - [시작디렉터리]부터 시작, 하위 디렉터리와 모든 파일 중 [조건]에 맞는 파일을 검색
    - [시작디렉터리]를 /로 지정하면 시스템 내의 모든 파일을 검색
- 조건 1: -name "문자열"
  - 파일명이 문자열과 일치하는 파일을 검색. ?(임의의 한 개 문자), *(임의의 여러 개 문자) 사용 가능

"log" : 파일이름이 log인 파일을 찾음

"*log" : 파일 이름이 log로 끝나는 모든 파일을 찾음

"log*" : 파일 이름이 log로 시작하는 모든 파일을 찾음

"*log*" : 파일 이름 중간에 log가 들어가 있는 파일을 찾음

"?log" : 파일 이름에서 첫 글자는 어떤 문자라도 상관없고 log로 끝나는 4개 문자 이름의 파일

"log??" : 파일 이름 시작이 log로 시작하고 끝의 두 문자가 어떤 것이어도 상관없는 5개 문자 이름의 파일

리눅스 기본 관리
: 파일 관리 –파일 찾기 (find)

- 조건 2: -user "사용자명"
  - 특정 유저가 소유한 파일들을 모두 찾음
- [실습] 현재 로그인 계정이 소유하고 있는 모든 파일을 찾아봅시다

```
[bituser@localhost log]$ find / -user "bituser"
find: '/boot/grub2': 허가 거부
/dev/pts/0
find: '/proc/tty/driver': 허가 거부
find: '/proc/1/task/1/fd': 허가 거부
...
...
/proc/6921/task/6921/fd/4
/proc/6921/task/6921/fd/5
find: '/proc/6921/task/6921/fd/6': 그런 파일이나 디렉터리가 없습니다
...
...
/home/bituser/.bash_history
/home/bituser/.mongorc.js
/home/bituser/.dbshell
/home/bituser/dowork
/home/bituser/.lesshst
```

```
[bituser@localhost log]$ find / -user "bituser" 2>/dev/null
```

리눅스 기본 관리
: 파일 관리 –파일 찾기 (find)

- 조건 3: -perm "퍼미션"
  - 명시된 퍼미션으로 된 파일을 찾을 때 사용
- [실습] /home 디렉터리 내 755 퍼미션을 가지고 있는 파일을 찾아봅시다

```
[bituser@localhost log]$ su -
암호:
마지막 로그인: 일  2월 14 21:58:57 KST 2016 일시 pts/0
[root@localhost ~]# find /home -perm 755
/home
/home/bituser/dowork
[root@localhost ~]#
```

- 조건 4: -type ?
  - ? 형태의 파일을 찾음
- [실습] /dev 디렉터리 내부에 문자열 디바이스 파일을 찾아봅시다

```
[root@localhost ~]# find /dev -type c
...
/dev/sg1
/dev/sg0
/dev/ppp
/dev/loop-control
/dev/uhid
/dev/btrfs-control
/dev/mapper/control
/dev/net/tun
/dev/vfio/vfio
```

리눅스 기본 관리
: 파일 관리 – 파일 내부 특정 패턴 검색(grep)

- grep : 파일 내 또는 입력 값으로부터 특정 패턴을 검색
  - 사용법
    - grep [옵션] 표현 [파일(들)]
  - 옵션
    - -v : 일치되는 내용이 없는 라인을 표시
    - -c : 일치되는 내용이 있는 행의 개수를 표시
    - -n : 일치되는 내용이 있는 행은 행 번호와 함께 표시
- [실습] /var/log/secure에서 root와 관련된 로그의 개수를 세어 봅니다

```
[bituser@localhost ~]$ su -
암호:
마지막 로그인: 화  2월 16 00:47:59 KST 2016 일시 pts/0
[root@localhost ~]# grep -c root /var/log/secure
45
[root@localhost ~]#
```

리눅스 기본 관리
: 파일 관리 – 파일 내부 특정 패턴 검색(grep)

- [실습] /var/log/secure 에서 root를 포함하는 라인을 출력해 봅시다

```
[root@localhost ~]# grep root /var/log/secure
Feb 14 10:57:34 localhost su: pam_unix(su-l:session): session opened for user root by bituser(uid=1000)
Feb 14 10:58:31 localhost su: pam_unix(su-l:session): session closed for user root
Feb 14 11:53:29 localhost su: pam_unix(su-l:session): session opened for user root by bituser(uid=1000)
...
...
...
...
Feb 15 22:41:00 localhost su: pam_unix(su-l:session): session opened for user root by bituser(uid=1000)
Feb 16 00:52:04 localhost su: pam_unix(su-l:session): session closed for user root
Feb 16 00:52:14 localhost su: pam_unix(su-l:session): session opened for user root by bituser(uid=1000)
[root@localhost ~]#
```

리눅스 기본 관리
: 파일 관리 – 파일 내부 특정 패턴 검색(grep)

- [실습] /etc/profile.d 디렉터리의 모든 파일에서 alias 설정되어 있는 행을 행번호와 함께
  출력해 봅니다

```
[root@localhost ~]# grep -n alias /etc/profile.d/*
/etc/profile.d/colorgrep.csh:9:alias grep 'grep --color=auto'
/etc/profile.d/colorgrep.csh:10:alias egrep 'egrep --color=auto'
/etc/profile.d/colorgrep.csh:11:alias fgrep 'fgrep --color=auto'
/etc/profile.d/colorgrep.sh:5:alias grep='grep --color=auto' 2>/dev/null
/etc/profile.d/colorgrep.sh:6:alias egrep='egrep --color=auto' 2>/dev/null
/etc/profile.d/colorgrep.sh:7:alias fgrep='fgrep --color=auto' 2>/dev/null
/etc/profile.d/colorls.csh:13:alias ll 'ls -l'
/etc/profile.d/colorls.csh:14:alias l. 'ls -d .*'
/etc/profile.d/colorls.csh:66:alias ll 'ls -l --color=auto'
/etc/profile.d/colorls.csh:67:alias l. 'ls -d .* --color=auto'
/etc/profile.d/colorls.csh:68:alias ls 'ls --color=auto'
/etc/profile.d/colorls.sh:9:  alias ll='ls -l' 2>/dev/null
/etc/profile.d/colorls.sh:10:  alias l.='ls -d .*' 2>/dev/null
/etc/profile.d/colorls.sh:55:alias ll='ls -l --color=auto' 2>/dev/null
/etc/profile.d/colorls.sh:56:alias l.='ls -d .* --color=auto' 2>/dev/null
/etc/profile.d/colorls.sh:57:alias ls='ls --color=auto' 2>/dev/null
/etc/profile.d/which2.csh:5:# alias which 'alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
/etc/profile.d/which2.sh:4:alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
[root@localhost ~]#
```

리눅스 기본 관리
: 파일 관리 – 파이프(Pipe)

- 프로그램의 실행결과를 다른 프로그램의 입력으로 연결 (|)
  둘 이상의 명령을 함께 사용하고 한 명령어의 출력 결과를 다른 명령어의 입력으로 전환
- [실습]

```
[root@localhost ~]# grep root /var/log/secure | less
```

```
Feb 14 21:14:51 localhost su: pam_unix(su-l:session): session closed for user root
Feb 14 21:19:16 localhost su: pam_unix(su-l:session): session opened for user root by bituser(uid=1000)
Feb 14 21:22:25 localhost su: pam_unix(su-l:session): session closed for user root
Feb 14 21:22:32 localhost su: pam_unix(su-l:session): session opened for user root by bituser(uid=1000)
Feb 14 21:52:59 localhost su: pam_unix(su-l:session): session closed for user root
Feb 14 21:53:54 localhost su: pam_unix(su-l:session): session opened for user root by bituser(uid=1000)
Feb 14 21:57:29 localhost su: pam_unix(su-l:session): session closed for user root
Feb 14 21:58:57 localhost su: pam_unix(su-l:session): session opened for user root by bituser(uid=1000)
Feb 14 22:01:28 localhost su: pam_unix(su-l:session): session closed for user root
Feb 14 22:15:12 localhost unix_chkpwd[6933]: password check failed for user (root)
Feb 14 22:15:12 localhost su: pam_unix(su-l:auth): authentication failure; logname=bituser uid=1000 euid=0 tty=pts/0 ruser=bituser
rhost=  user=root
Feb 14 22:15:12 localhost su: pam_succeed_if(su-l:auth): requirement "uid >= 1000" not met by user "root"
Feb 14 22:15:19 localhost su: pam_unix(su-l:session): session opened for user root by bituser(uid=1000)
Feb 14 22:51:12 localhost su: pam_unix(su-l:session): session closed for user root
Feb 15 21:35:34 localhost su: pam_unix(su-l:session): session opened for user root by bituser(uid=1000)
:
```

리눅스 기본 관리
: 파일 관리 – 리다이렉션 (Redirection)

- 리다이렉션을 이용하면 명령의 출력을 변경할 수 있음
  - 명령어 출력의 기본 방향은 터미널(stdout)
  - 리다이렉션을 이용하면 파일로 출력을 돌릴 수 있음
- 연산자
  - 명령어 > 파일 : 파일이 없으면 생성하고, 있다면 기존의 내용을 지움
  - 명령어 >> 파일 : 파일이 없으면 생성하고, 있다면 기존의 내영에 추가
  - 명령어 < 파일 : 파일로부터 표준 입력 (stdin)을 받음

- [실습] echo는 주어진 문장을 현재 터미널 화면에 출력함

```
[bituser@localhost ~]$ echo "hello"
hello
[bituser@localhost ~]$
```

리눅스 기본 관리
: 파일 관리 – 리다이렉션 (Redirection)

- [실습] echo로 출력할 문자열을 > 연산자를 이용 파일에 저장

```
[bituser@localhost ~]$ echo "hello" > hello.txt
[bituser@localhost ~]$ cat hello.txt
hello
[bituser@localhost ~]$
```

- [실습] hello.txt 파일에 'World'라는 단어를 추가해 봅니다

```
[bituser@localhost ~]$ echo "world" >> hello.txt
[bituser@localhost ~]$ cat hello.txt
hello
world
[bituser@localhost ~]$
```

- [실습] 몇 개의 단어를 더 추가해 봅니다
  - apple, linux, java, red, hat, gnu, unix

리눅스 기본 관리
: 파일 관리 – 리다이렉션 (Redirection)

- [실습] hello.txt를 표준 입력으로 받아 sort 명령에 입력시키고 결과를 확인해 봅니다

```
[bituser@localhost ~]$ sort < hello.txt
apple
gnu
hat
hello
java
linux
red
unix
world
[bituser@localhost ~]$
```

- [실습] 위 sort된 결과를 다시 파일 sorted.txt로 출력해 봅시다

```
[bituser@localhost ~]$ sort < hello.txt > sort.txt
[bituser@localhost ~]$ cat sort.txt
apple
gnu
hat
hello
java
linux
red
unix
world
[bituser@localhost ~]$
```

리눅스 기본 관리
: 파일 관리 – 파일 소유 변경 (chown)

- chown : 파일 소유자나 소유 그룹을 변경하기 위한 명령
  - 사용법
    - chown [옵션] 소유자:소유그룹 파일명
  - 옵션
    - -R : 경로와 그 하위 파일(혹은 디렉터리) 소유자나 소유그룹을 모두 변경
- [실습] root로 새로 파일 root.file을 생성하고 /home/bituser로 복사한 후, 소유계정과 그룹을 바꿔 봅니다

```
[root@localhost ~]# touch root.file
[root@localhost ~]# cp root.file /home/bituser
[root@localhost ~]# chown bituser /home/bituser/root.file
[root@localhost ~]# ls -la /home/bituser/root.file
-rw-r--r--. 1 bituser root 0  2월 16 03:32 /home/bituser/root.file
[root@localhost ~]# chown root:wheel /home/bituser/root.file
[root@localhost ~]# ls -la /home/bituser/root.file
-rw-r--r--. 1 root wheel 0  2월 16 03:32 /home/bituser/root.file
[root@localhost ~]# chown bituser:users /home/bituser/root.file
[root@localhost ~]# ls -la /home/bituser/root.file
-rw-r--r--. 1 bituser users 0  2월 16 03:32 /home/bituser/root.file
```

리눅스 기본 관리
: 파일 관리 – 파일 묶기 (tar)

- 압축 유틸리티 중 가장 많이 사용하는 것은 tar, gzip, bzip2
- tar : 파일을 묶어주는 역할(archive)을 하는 유틸리티 (압축은 하지 않음)
  - 사용법
    - 묶을 때 : tar [옵션] 생성파일.tar 묶을 파일
    - 해제할 때 : tar [옵션] 파일.tar
  - 옵션
    - -c : 새 저장 파일을 만듦. 즉, 파일을 묶을 때 사용
    - -x : 묶인 파일을 해제
    - -v : 처리중인 파일을 자세하게 보여줌
    - -f : 파일을 지정
    - -z : gzip 압축 또는 해제
    - -j : bzip2 압축 또는 해제
- [실습] tar –cvf 옵션을 이용, /home/bituser 폴더를 bituser.tar로 묶어 봅시다

리눅스 기본 관리
: 파일 관리 – 파일 묶기 (tar)

- 일반적으로 tar를 이용, 파일을 묶을 때는 –cvf 옵션을 사용, 파일을 풀 때는 –xvf 옵션을 사용
- [실습] tar –cvf 옵션을 이용, /home/bituser 폴더를 bituser.tar로 묶어 봅시다

```
[root@localhost ~]# tar -cvf bituser.tar /home/bituser
tar: Removing leading `/' from member names
/home/bituser/
/home/bituser/.bash_logout
/home/bituser/.bash_profile
/home/bituser/.bashrc
/home/bituser/.bash_history
/home/bituser/.mongorc.js
/home/bituser/.dbshell
/home/bituser/dowork/
/home/bituser/.lesshst
/home/bituser/hello.txt
/home/bituser/sort.txt
/home/bituser/root.file
```

-

리눅스 기본 관리
: 파일 관리 – 파일 압축 (gzip)

- tar는 압축을 하지 않기 때문에 사이즈를 줄이기 위해서는 gzip을 이용합니다.
  압축된 파일의 확장자는 .gz가 됩니다
- [실습] bituser.tar 파일을 gzip으로 압축해 봅니다

```
[root@localhost ~]# gzip bituser.tar
[root@localhost ~]# ls -l
합계 62088
-rw-------. 1 root root     1066  2월 11 17:32 anaconda-ks.cfg
-rw-r--r--. 1 root root     4539  2월 16 03:53 bituser.tar.gz
-rw-r--r--. 1 root root        0  2월 16 03:32 root.file
```

- .gz 파일을 압축 해제할 때는 gzip –d 또는 gunzip명령어를 사용
- [실습] bituser.tar.gz 파일을 압축 해제해 봅시다

```
[root@localhost ~]# gzip -d bituser.tar.gz
[root@localhost ~]# ls -l
합계 62112
-rw-------. 1 root root     1066  2월 11 17:32 anaconda-ks.cfg
-rw-r--r--. 1 root root    30720  2월 16 03:53 bituser.tar
-rw-r--r--. 1 root root        0  2월 16 03:32 root.file
[root@localhost ~]#
```

리눅스 기본 관리
: 파일 관리 – 파일 압축 (gzip)

- [실습] tar의 –z 옵션을 이용, /home/bituser 디렉터리를 bituser.tar.gz로 묶어 봅시다

```
[root@localhost ~]# tar -cvzf bituser.tar.gz /home/bituser
tar: Removing leading `/' from member names
/home/bituser/
/home/bituser/.bash_logout
/home/bituser/.bash_profile
/home/bituser/.bashrc
/home/bituser/.bash_history
/home/bituser/.mongorc.js
/home/bituser/.dbshell
/home/bituser/dowork/
/home/bituser/.lesshst
/home/bituser/hello.txt
/home/bituser/sort.txt
/home/bituser/root.file
[root@localhost ~]# ls -l
합계 62120
-rw-------. 1 root root    1066  2월 11 17:32 anaconda-ks.cfg
-rw-r--r--. 1 root root    4526  2월 16 04:03 bituser.tar.gz
-rw-r--r--. 1 root root       0  2월 16 03:32 root.file
```

- [실습] tar –zxvf를 이용, 압축 해제와 묶음 해제를 동시에 해 봅시다

```
[root@localhost ~]# tar -xvzf bituser.tar.gz
```

리눅스 기본 관리
: 파일 관리 – 파일 압축 (tar with bzip2)

- [실습] tar의 –j 옵션을 이용, /home/bituser 디렉터리를 bituser.tar.bz2로 묶어 봅시다

```
[root@localhost ~]# tar -cvjf bituser.tar.bz2 /home/bituser/
tar: Removing leading `/' from member names
/home/bituser/
/home/bituser/.bash_logout
/home/bituser/.bash_profile
/home/bituser/.bashrc
/home/bituser/.bash_history
/home/bituser/.mongorc.js
/home/bituser/.dbshell
/home/bituser/dowork/
/home/bituser/.lesshst
/home/bituser/hello.txt
/home/bituser/sort.txt
/home/bituser/root.file
[root@localhost ~]# ls -l
합계 62088
-rw-------. 1 root root     1066  2월 11 17:32 anaconda-ks.cfg
-rw-r--r--. 1 root root     4204  2월 16 04:12 bituser.tar.bz2
-rw-r--r--. 1 root root        0  2월 16 03:32 root.file
[root@localhost ~]#
```

- [실습] tar –jxvf를 이용, 압축 해제와 묶음 해제를 동시에 해 봅시다

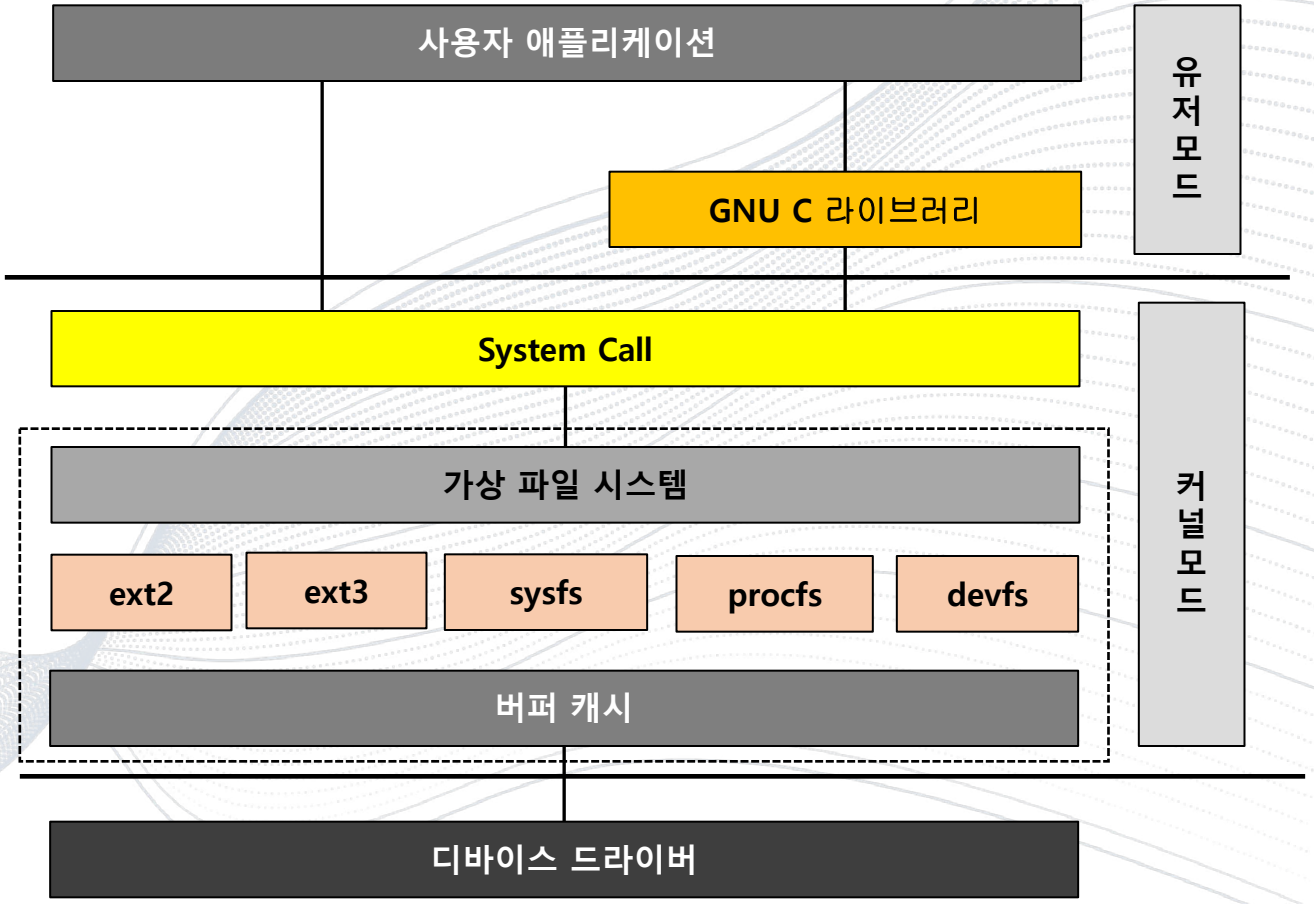
```
[root@localhost ~]# tar -xvjf bituser.tar.gz
```

리눅스 내부 구조
: 파일 시스템의 이해

- 파일 시스템
  - 운영체제는 커널 이미지, 시스템 실행과 관련된 시스템 파일, 그리고 유틸리티 파일 등을 제공
  - 사용자의 데이터 저장을 위해 사용됨
  - 파일 시스템을 통해 이러한 파일들을 관리
  - 파일 시스템은 파일의 저장, 읽기, 삭제 등의 파일 관리 기능과 파일에 대한 접근 제어 기능을 제공
  - 윈도우에서는 FAT32, NTFS 등의 파일 시스템을 제공, 리눅스에서는 EXT3, EXT4 등의 파일 시스템을 제공
  - 디렉터리 안에 디렉터리를 저장할 수 있는 트리형 구조
  - 루트 디렉터리 : 장치의 메인 디렉터리
    - 여러 장치(하드디스크)를 사용하면 루트 디렉터리 구분에 문제가 생김
    - 윈도우에서는 분리형 루트라 불리는 방식으로 이를 해결
    - 유닉스 계열에서는 통합형 파일 시스템을 사용

리눅스 내부 구조
: 가상 파일 시스템(VFS)

- 유닉스는 디스크, 터미널, 네트워크 카드 등 모든 주변장치들을 파일로 취급
- 디스크상의 파일 시스템 외에도 다양한 기능의 특수 파일 시스템이 존재
- 이러한 다양한 파일 시스템을 하나의 파일 시스템처럼 사용할 수 있도록 가상 파일 시스템(VFS) 구조를 사용



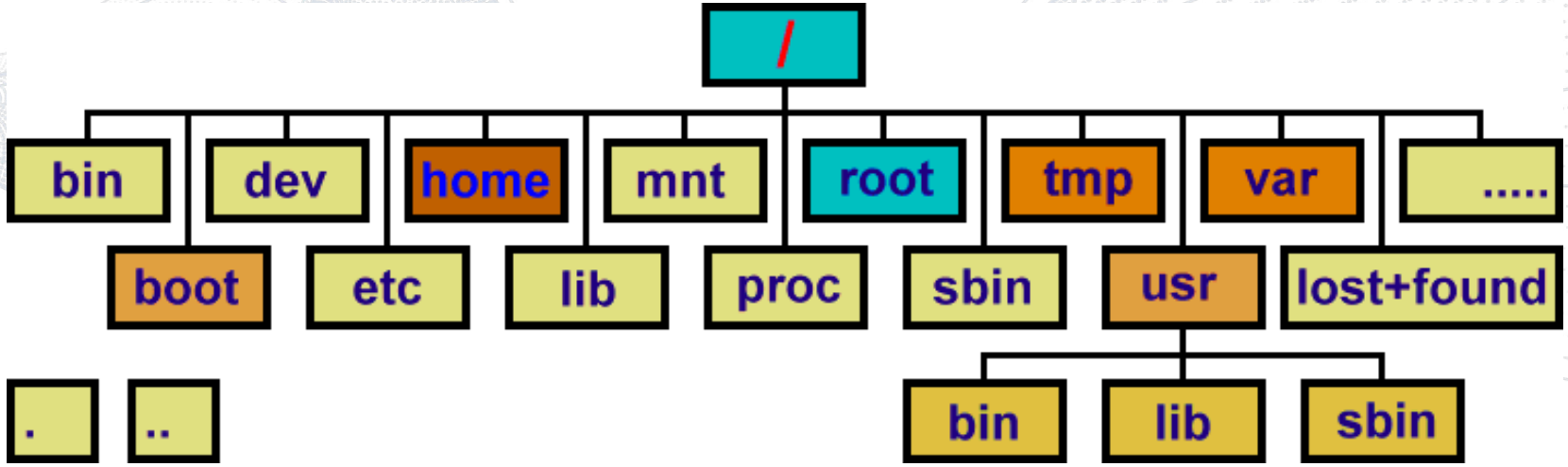
리눅스 내부 구조
: 리눅스의 가상 파일 시스템

- 리눅스 커널의 특징 중 하나이며 유닉스에 비해 초창기부터 가상 파일 시스템을 지원
- 모든 파일 시스템을 하나의 파일 시스템으로 보이게 하는 계층(layer)이라 할 수 있음
- 파일, 디렉터리, 특수 파일 등을 파일 처리 시스템 호출(System Call)을 통해 일관적으로 조작할 수 있음

- 주요 특수 파일 시스템
  - procfs : 커널 및 커널 모듈(디바이스 드라이버) 정보를 참조하거나 설정 변경을 위한 파일 시스템 -> /proc 에 마운트됨
  - sysfs : 시스템에 접속된 디바이스 정보를 참조하거나 설정 변경을 위한 파일 시스템
    -> /sys 에 마운트
  - devfs : 물리 디바이스에 억세스하기 위한 디바이스 파일을 배치하는 파일 시스템
    -> /dev 에 마운트

리눅스 내부 구조
: 리눅스의 디렉터리 구조

- 리눅스 배포판은 FHS 표준에 따르도록 권장
- 강제사항은 아니지만, 대부분 배포판은 이 표준을 준수
- 각각의 디렉터리는 그에 맞는 용도가 있음
- 안전하고 편리한 시스템 운용을 위해서는 용도를 잘 알고, 그에 맞게 활용해야 함



리눅스 내부 구조
: 리눅스의 디렉터리 구조

- / : 루트 디렉터리
  - 시스템의 근간이 되는 가장 중요한 디렉터리. 모든 파티션, 디렉터리는 루트 디렉터리 아래 위치하므로 반드시 있어야 함
- /bin
  - 시스템 관리자 혹은 일반 사용자가 실행할 수 있는 명령어들이 위치
  - 예) cat, chmod, date, ls, mkdir, rm, touch, vi 등
- /sbin
  - 시스템 관리자가 사용할 수 있는 명령어들이 위치.
  - 시스템 수정, 복구에 관한 많은 명령어들이 포함되므로 일반 사용자의 실행 권한 제한 등 보안에 신경을 써야 함
  - 예) ifconfig, reboot, shutdown, halt, mount, fsck 등

리눅스 내부 구조
: 리눅스의 디렉터리 구조

- /boot
  - 부트로더와 부팅에 관련된 파일들을 포함
  - 손상되면 시스템이 부팅되지 않으므로 특별한 목적이 아니면 접근하지 말아야 함
- /home
  - 사용자들의 계정 홈 디렉터리가 하위 디렉터리로 존재
- /dev
  - 디바이스 파일들이 위치
  - 시스템의 모든 장치가 파일로 표현되어 있으며 udev라는 데몬이 이곳의 장치 파일을 관리
  - 예) /dev/sda, /dev/hda, /dev/tty1, /dev/pts/0 등
- /lib
  - 시스템의 프로그램이 실행될 때 필요한 공유 라이브러리들을 포함
  - 특별한 일이 없으면 변경하거나 삭제하지 않는 것이 좋음

리눅스 내부 구조
: 리눅스의 디렉터리 구조

- /etc
  - 시스템 혹은 각종 프로그램들의 환경 설정 파일들이 위치
  - 시스템 관리에서는 주로 이곳의 파일들을 수정(백업 권장)
  - 예) /etc/fstab, /etc/group, /etc/passwd, /etc/sysconfig/i18n 등
- /mnt
  - 마운트를 위한 임시 디렉터리가 위치. 광학 드라이브나 USB 등 이동 디스크를 마운트할 때 이용
- /root
  - root 계정의 홈 디렉터리. root 계정만 접근 가능
- /var
  - log 파일 등 수시로 업데이트 되는 파일들이 위치
  - 시스템 운영에 필요한 파일들도 위치하므로 수정과 삭제에 주의해야 함

리눅스 내부 구조
: 리눅스의 디렉터리 구조

- /proc
  - 실행중인 프로세스 정보, CPU, 메모리 등 시스템 정보가 가상 파일로 저장
  - 대부분 읽기 전용이지만 일부 쓰기가 가능한 파일들이 있는데, 이런 파일들은 커널의 기능을 변경할 수 있음
- [실습] CPU 정보 출력

리눅스 내부 구조
: 리눅스의 디렉터리 구조

- [실습] 파티션 정보를 출력해 봅니다

```
[root@lx ~]# cat /proc/partitions
major minor  #blocks  name

  11        0    1048575 sr0
   8        0   25165824 sda
   8        1     512000 sda1
   8        2    2098176 sda2
   8        3   22554624 sda3
[root@lx ~]#
power management:
```

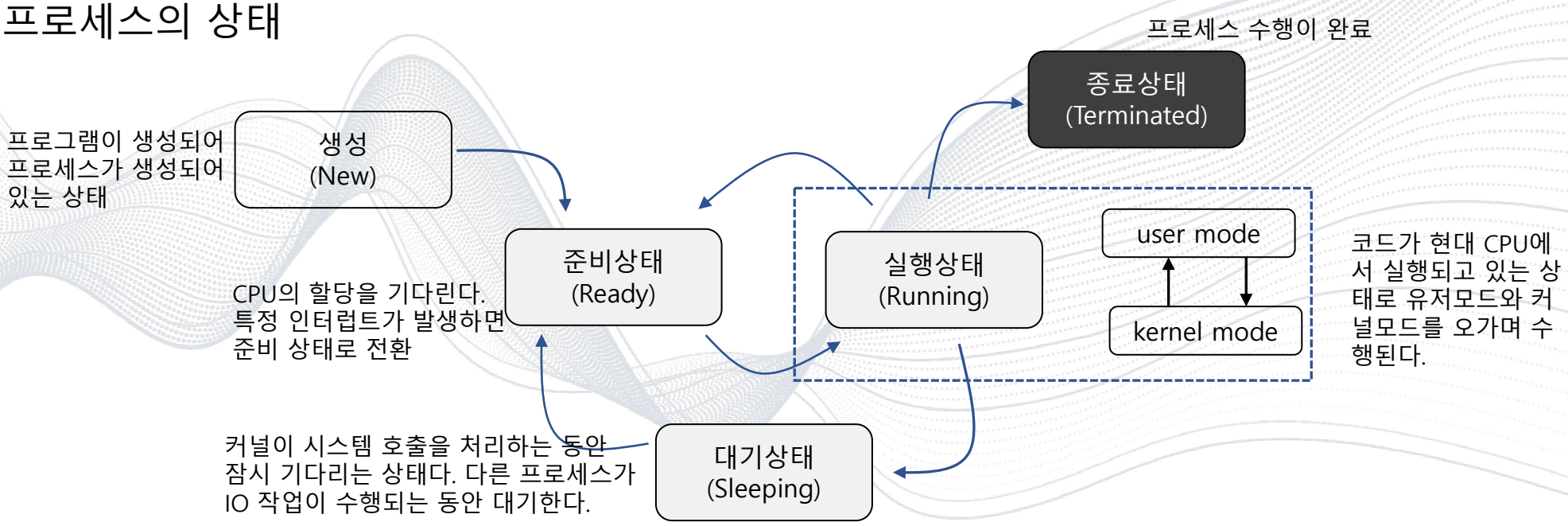
- [실습] 다음 정보도 출력해 봅시다
  - /proc/meminfo
  - /proc/uptime
  - /proc/filesystems
  - /proc/version
  - /proc/modules
  - /proc/loadavg

리눅스 내부 구조
: 리눅스의 디렉터리 구조

- /usr/bin
  - 응용 프로그램들의 실행파일들이 위치
- /usr/sbin
  - 시스템 관리를 위한 명령어들이 위치
- /usr/include
  - 시스템, 네트워크 프로그래밍을 위한 C 헤더 파일
- /usr/lib
  - /usr/bin, /usr/sbin 에 있는 실행 파일들을 위한 라이브러리들이 위치
- /tmp
  - 임시로 파일을 만들고 삭제하는 공간
- /lost+found
  - 부팅시 파일 시스템에 문제가 생길 경우 fsck 명령어로 복구할 때 사용하는 디렉터리

리눅스 내부 구조
: 프로세스

- 프로세스
  - 유닉스는 시분할 시스템으로 여러 개의 프로그램을 동시에 실행(멀티 태스킹)
  - 컴퓨터 내에서 실행 중인 프로그램을 프로세스(process) 또는 태스크(Task)라 함
  - 여러 프로세스가 동시에 실행되는 것을 멀티 프로세스라 함
- 프로세스의 상태



리눅스의 내부 구조
: 프로세스 정보

- [실습] ps 명령을 이용, PID를 확인하고 /proc 디렉터리 안의 status 정보를 확인해 봅니다

```
[root@lx ~]# ps
PID TTY          TIME CMD
14877 pts/0    00:00:00 su
14881 pts/0    00:00:00 bash
14896 pts/0    00:00:00 ps

[root@lx ~]# cat /proc/14881/status
Name:           bash
State:          S (sleeping)
Tgid:           14881
Ngid:           0
Cpus_allowed_list:              0
Mems_allowed:
                00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00
000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,000000
00,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000001
Mems_allowed_list:              0
voluntary_ctxt_switches:        55
nonvoluntary_ctxt_switches:             0
[root@lx ~]# ^C
```

리눅스의 내부 구조
: 프로세스 관리 명령어 - ps

- ps : 현재 실행되고 있는 프로세스의 목록을 보여줌
  - 사용법
    - ps [옵션]
  - 옵션
    - -l : 자세한 정보를 출력
    - -a : 다른 사용자들의 프로세스도 보여줌
    - -u : 프로세스의 사용자 이름과 시작 시간을 출력
    - -x : 터미널과 연결되지 않은 프로세스도 출력
    - -e : 환경을 보여줌
    - -f : 프로세스의 정보를 한 줄로 자세히 출력
    - -r : 현재 실행중인 프로세스를 출력
    - -j : 작업 중심의 형태로 출력
    - -c : 커널 task_struct 구조체 형태로 보여줌
  - 보통 –aux 또는 –ef 옵션을 사용하여 프로세스 상태를 확인

리눅스의 내부 구조
: 프로세스 관리 명령어 - ps

- [실습] –ef 옵션을 사용하여 프로세스 상태를 확인해 봅니다

```
[root@lx ~]# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 2월17 ?        00:00:01 /usr/lib/systemd/systemd --switched-root --system --deseri
root         2     0  0 2월17 ?        00:00:00 [kthreadd]
root         3     2  0 2월17 ?        00:00:00 [ksoftirqd/0]
root         6     2  0 2월17 ?        00:00:00 [kworker/u2:0]
root         7     2  0 2월17 ?        00:00:00 [migration/0]
root         8     2  0 2월17 ?        00:00:00 [rcu_bh]
.
.
.
root     14877 14858  0 05:26 pts/0    00:00:00 su -
root     14881 14877  0 05:26 pts/0    00:00:00 -bash
root     14898     2  0 05:28 ?        00:00:00 [kworker/0:2H]
postfix  14900   857  0 05:32 ?        00:00:00 pickup -l -t unix -u
root     14901     2  0 05:33 ?        00:00:00 [kworker/0:0H]
root     14913     2  0 05:53 ?        00:00:00 [kworker/0:1]
root     14914     2  0 05:58 ?        00:00:00 [kworker/0:0]
postfix  14942   857  0 06:01 ?        00:00:00 cleanup -z -t unix -u
postfix  14944   857  0 06:01 ?        00:00:00 trivial-rewrite -n rewrite -t unix -u
postfix  14945   857  0 06:01 ?        00:00:00 local -t unix
root     14946 14881  0 06:01 pts/0    00:00:00 ps -ef
```

USER: 프로세스 소유자의 계정 **PID:** 프로세스를 구분하는 프로세스 아이디 **PPID:** 부모 프로세스 PID
STIME: 프로세스 시작 시간 **TTY:** 프로세스의 표준 입출력을 담당하는 터미널 **TIME:** 프로세스의 CPU 점유시간
CMD: 실행 명령어

리눅스의 내부 구조
: 프로세스 관리 명령어 - ps

- [실습] –aux 옵션을 사용하여 프로세스 상태를 확인해 봅니다

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.3  44364  7056 ?        Ss   2월17   0:01 /usr/lib/systemd/systemd --switched-roo
root         2  0.0  0.0      0     0 ?        S    2월17   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    2월17   0:00 [ksoftirqd/0]
root         6  0.0  0.0      0     0 ?        S    2월17   0:00 [kworker/u2:0]
root         7  0.0  0.0      0     0 ?        S    2월17   0:00 [migration/0]
root         8  0.0  0.0      0     0 ?        S    2월17   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    2월17   0:00 [rcuob/0]
root        10  0.0  0.0      0     0 ?        R    2월17   0:00 [rcu_sched]
root        11  0.0  0.0      0     0 ?        S    2월17   0:02 [rcuos/0]
root        12  0.0  0.0      0     0 ?        S    2월17   0:00 [watchdog/0]
.
.
.
root     14881  0.0  0.1 115504  2160 pts/0    S    05:26   0:00 -bash
root     14898  0.0  0.0      0     0 ?        S<   05:28   0:00 [kworker/0:2H]
postfix  14900  0.0  0.1  91232  3892 ?        S    05:32   0:00 pickup -l -t unix -u
root     14901  0.0  0.0      0     0 ?        S<   05:33   0:00 [kworker/0:0H]
root     14947  0.0  0.0      0     0 ?        S    06:03   0:00 [kworker/0:1]
root     14949  0.0  0.0      0     0 ?        S    06:08   0:00 [kworker/0:0]
root     14950  0.0  0.0 139496  1656 pts/0    R+   06:08   0:00 ps -aux
```

USER: 프로세스 소유자의 계정 **PID:** 프로세스를 구분하는 프로세스 아이디 **%CPU:**마지막 분 동안 사용한 CPU의 %
%MEM: 마지막 분 동안 사용한 메모리 양의 % **VSZ:** 프로세스 데이터 스택의 크기 **RSS:** 실제 메모리 양
COMMAND: 실행 명령어 STAT: 프로세스의 상태 **START:** 프로세스가 시작된 시간

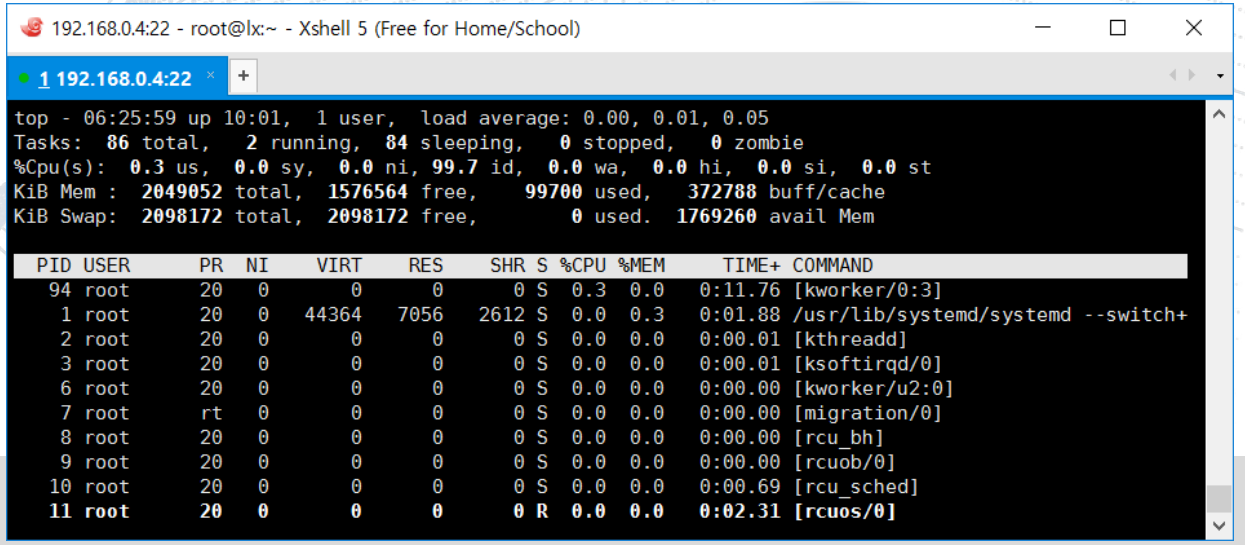
stat:
p: 수행가능, T:일시 정지, D: 디스크 입출력 대기, S: 20초 미만의 짧은 휴식, I :20초 이상의 긴 휴식, Z:좀비 상태

리눅스 내부 구조
: 프로세스 관리 명령어 - pstree

- pstree : 프로세스 정보를 트리 형태로 보여줌
  - 사용법
    - pstree [옵션]
  - 옵션
    - -n : PID 순으로 정렬
    - -p : 프로세스명 + PID
- [실습] pstree 명령으로 프로세스의 정보를 확인해 봅시다

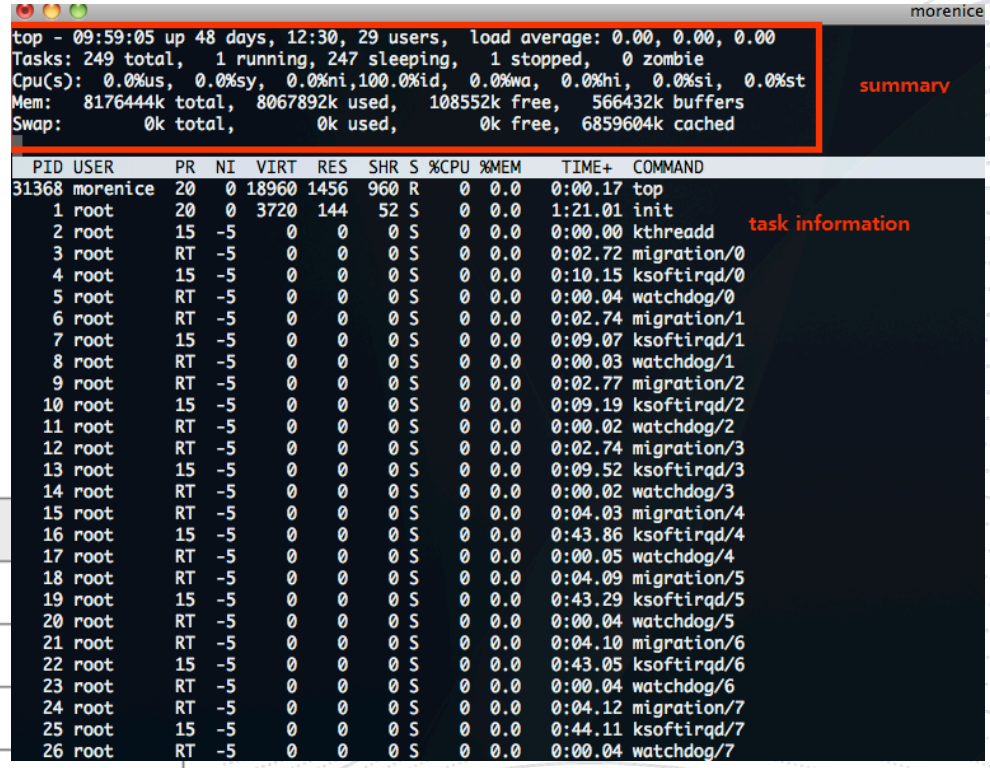
리눅스 내부 구조
: 프로세스 관리 명령어 - top

- top : 프로세스의 CPU, Memory 사용량 등 전반적 상황을 실시간으로 모니터링
  - 사용법
    - top [옵션]
  - 옵션
    - -d : 시간, 화면 갱신 시간 지정
    - -c : 명령행 전체를 보여줌
    - -q : 화면을 계속 갱신
- [실습] top 명령으로 프로세스를 확인해 봅니다



리눅스 내부 구조

: 프로세스 관리 명령어 - top

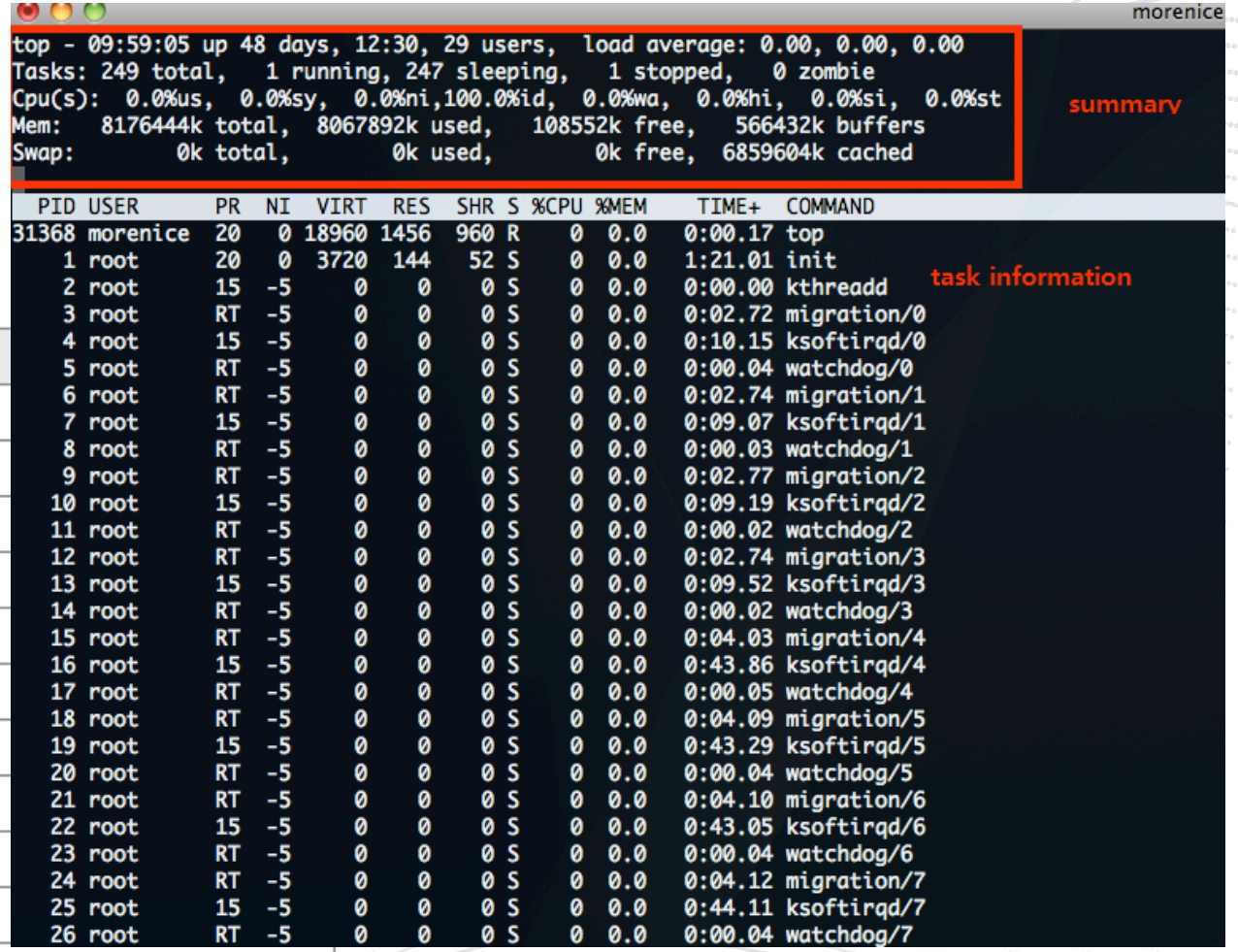


top 세 번째 줄 - Cpu(s)

이름	설명
us	사용자 어플리케이션에 할당 된 CPU 비중
sy	시스템 어플리케이션에 할당 된 CPU 비중
ni	CPU 우선순위를 낮추기 위해 (nice) 할당 된 CPU 비중
id	idle (휴식) 상태의 CPU 비중
wa	I/O를 기다리는 프로세스에 할당 된 CPU 비중
hi	하드웨어 인터럽트를 기다리는 프로세스에 할당 된 CPU 비중
si	소프트웨어 인터럽트를 기다리는 프로세스에 할당 된 CPU 비중
st	하이퍼바이저 (가상플랫폼을 실행하는 소프트웨어)에 할당 된 CPU 비중

리눅스 내부 구조

: 프로세스 관리 명령어 - top



Process Table

이름	설명
PID	프로세스의 ID 번호
USER	프로세스를 소유한 사용자
PR	프로세스의 우선 순위
NI	프로세스의 nice 값
VIRT	프로세스가 소비하는 가상 메모리의 양
RES	실제 상주하는 가상 메모리의 크기
SHR	프로세스가 사용하고 있는 공유 메모리의 양
S	프로세스 상태 (ex 잠자기 상태, 실행 중 상태 등)
%CPU	CPU 사용 률
%MEM	메모리 사용 률
TIME+	Task 가 시작된 이후 사용한 시간
COMMAND	명령어 이름

리눅스 내부 구조
: 프로세스 관리 명령어 - top

- top은 다른 프로세스 관리 명령(ps, pstree 등)과는 달리, 인터랙티브 쉘 프로그램
- 주요 단축키
  - q : 모니터링 종료
  - m : 메모리 사용량 순으로 정렬
  - t : 실행시간이 긴 순서로 정렬
  - r : 정렬 순서 변경
- [실습] top으로 모니터링 도중 단축키를 입력, 정렬 순서를 변경해 봅니다

리눅스 내부 구조
: 프로세스 관리 명령어 - kill

- kill : 지정한 프로세스에 시그널을 보내는 명령
  - 사용법
    - kill [옵션] PID
  - 옵션
    - -l : 사용할 수 있는 시그널을 출력
    - -Signal_ID : 프로세스에 Signal_ID에 해당하는 시그널을 보냄
- [실습] –l 옵션을 이용, 사용할 수 있는 시그널 목록을 확인해 봅니다

```
[root@lx ~]# kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-1451) SIGRTMAX-1352) SIGRTMAX-12
53) SIGRTMAX-1154) SIGRTMAX-1055) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

리눅스 내부 구조
: 프로세스 관리 명령어 - kill

- [실습] -9 (-KILL) 시그널을 프로세스에 전송, 프로세스를 죽여 봅니다

```
[root@lx ~]# ps -ef | grep hhtpd
root     14999 14881  0 06:39 pts/0    00:00:00 grep --color=auto hhtpd
[root@lx ~]# ps -ef | grep httpd
root     13691     1  0  2월17 ?     00:00:01 /usr/local/apache/bin/httpd -k start
daemon   13692 13691  0  2월17 ?     00:00:00 /usr/local/apache/bin/httpd -k start
daemon   13693 13691  0  2월17 ?     00:00:00 /usr/local/apache/bin/httpd -k start
daemon   13694 13691  0  2월17 ?     00:00:00 /usr/local/apache/bin/httpd -k start
daemon   13695 13691  0  2월17 ?     00:00:00 /usr/local/apache/bin/httpd -k start
daemon   13779 13691  0  2월17 ?     00:00:00 /usr/local/apache/bin/httpd -k start
root     15001 14881  0 06:39 pts/0    00:00:00 grep --color=auto httpd
[root@lx ~]# kill -9 13691
[root@lx ~]# ps -ef | grep httpds
root     15003 14881  0 06:40 pts/0    00:00:00 grep --color=auto httpds
[root@lx ~]#
```

- [Tip] KILL 시그널을 받게 되면 프로그램은 Clean Up 코드를 거치지 않게 됩니다. 바로 -9(-KILL) 시그널을 보내기보다는 두 세번 정도 -15(-TERM) 시그널을 보내고 정상종료 되지 않을 때 KILL 시그널을 보내는 것이 좋습니다