

목차

1. VMware를 활용한 Installation
2. Login & Logout
3. Prompt 구조
4. Basic Command
5. Browsing the file system
6. Filesystem & Hierarchy
7. vi
8. Regular Expression
9. grep
10. awk
11. cut

목차

- 12. 계정관리
- 13.그룹관리
- 14.암호관리
- 15.퍼미션 관린
- 16.Filesystem & Disk Management
- 17.mount / unmounts
- 18.inode
- 19.swap partition
- 20.RPM
- 21.YUM
- 22.Tar Zip Bzip

Linux Installation & Basic Command

Login & Logout

- 컴퓨터 시스템이나 네트워크 상에서 구분되는 사용자의 아이디와 암호를 입력하고 접근 권한이나 사용 권한을 얻어 시스템을 사용할 수 있게 되는 과정
- 리눅스 시스템은 Multi-Users 및 Multi-Tasking 환경이기 때문에 로그인 과정과 같은 사용자 구분이 되어있지 않으면 다른 사용자가 임의적인 작업을 할 수 있는 문제가 발생 할 수 있다
- 해당 사용자만이 작업할 수 있는 공간에 들어갈 수 있도록 하는 절차를 로그인 이라고 한다
- 로그 아웃을 하는 방법은 프롬프트 창에서 exit 명령어를 입력 하거나 CTRL + d 키를 입력한다

Login & Logout

● Login & Logout

● 로그인 하는 방법

- 대/소문자를 엄격히 구분하여 계정이름 및 패스워드를 입력한다
- 입력할 때 Caps Lock 키가 꺼져 있는지 주의한다
- 입력하는 패스워드는 보안상 콘솔 화면에 보여지지 않는다(GUI 환경에서는 예외)

● 로그아웃 하는 방법

- Shell(콘솔) 상에서 "**exit**" or "**logout**" or "**Ctrl + d**" 를 입력한다.

Shutdown - tui

- 리눅스 종료 명령

- shutdown 명령

- 다양한 방법으로 시스템을 종료하거나 재부팅하는 명령

- 현재 수행중인 프로세스들을 모두 종료하며 sync를 수행하여 아직 저장되어 있지 않은 데이터를 디스크에 저장한 후 모든 파일시스템을 umount시킨 후에 시스템을 종료한다.

- shutdown은 reboot 이나 halt 명령어에 비해 다양한 시스템 종료 방법을 제공

- 예약 종료

- 시스템에 접속 사용자들에게 다양한 방법으로 종료 사실을 알려줌

Shutdown - tui

- 리눅스 종료 명령
- shutdown 구동 과정

Step_1 - shutdown 명령 수행 후 sync작업 수행(sync)

Step_2 - 접속해 있는 사용자들에게 종료 메시지 전송

Step_3 - 새로운 사용자의 로그인 금지

Step_4 - 지정된 시간에 종료되지 않은 프로세스 강제종료(KILL)

Step_5 - 지정된 시간에 logout 하지 않은 계정 강제 logout (logout)

Step_6 - 메모리에 남아있는 데이터를 디스크에 저장(sync)

Step_7 - 시스템 종료에 관련 정보를 로그파일에 기록 (wtmp, utmp)

Step_8 - 마운트 되어 있는 디바이스 마운트 해제 (umount)

Step_9 - 시스템 종료 (System halt)

Shutdown - tui

리눅스 종료 명령

shutdown 명령

Syntax : shutdown [옵션] [시간] "사용자에게 전달할 메시지"

옵션	의미
-k	실질적인 shutdown을 하는게 아니라, 모든 사용자에게 종료 경고 메시지 만을 전송한다
-h	시스템 shutdown후 시스템 종료
-r	시스템 shutdown후 시스템 재시작
-f	빠른 재부팅시 사용(fsck 수행을 하지 않음)
-c	종료예약작업시 종료작업 취소 (Ctrl + c)
+m	현재 시간으로부터 종료시점 시간지정(분)
hh:mm	절대시간으로 종료시점 시간지정(시간:분)
now	명령어를 수행하는 순간 종료

Prompt

리눅스 Prompt 구조

구분자	의미
root	로그인한 사용자 계정명
localhost	리눅스 시스템의 호스트명(/etc/sysconfig/network)
~	현재 작업 디렉토리 위치
#	관리자계정(#) 일반계정(\$)

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

```
[ITBank@localhost ~]$
```

Basic Command

Basic Command

🌐 기본명령어

- 🌐 pwd : 현재 작업 디렉터리 경로 보기
 - 🌐 pwd 명령은 현재 작업중인 디렉터리의 절대 경로를 보여준다.
 - 🌐 Syntax : pwd
- 🌐 cd : 작업 디렉터리 변경 한다.
 - 🌐 디렉토리간 이동 명령
 - 🌐 Syntax : cd [인자값]

인자값	의미
directory	이동하기 원하는 디렉토리
.	현재 디렉토리
..	상위 디렉토리
\$변수	변수에 지정된 정보를 이용하여 디렉토리 이동
~	로그인 된 사용자의 홈디렉토리로 이동
~계정명	지정된 계정의 홈디렉토리로 이동

Basic Command

🌐 기본명령어

🌐 ls : 디렉토리의 목록 보기

도스의 dir과 같은 역할을 하며, 해당 디렉토리에 있는 내용을 출력한다.

Syntax : ls [option] [directory / file]

🌐 Option

옵션	의미
-a, --all	. 을 포함한 경로안의 모든 파일과 디렉토리 표시
-l, --format=long	지정한 디렉토리의 내용을 자세히 출력
-d, --directory	지정된 디렉토리의 정보 출력
-n, --numeric	파일 및 디렉토리 정보 출력시 UID, GID를 사용
-F, --classify	파일 형식을 알리는 문자를 각 파일 뒤에 추가 "*", "/", "@", " ", "=", "NULL"
-R, --recursive	하위 경로와 그 안에 있는 모든 파일들도 같이 나열

Basic Command

🌐 기본명령어

🌐 cp(copy) : 파일 / 디렉토리 복사

파일이나 디렉토리를 복사하는 명령어

Syntax : cp [-option] [sources] [target]

🌐 Option

옵션	의미
-i, --interactive	복사대상 파일이 있을 경우, 사용자에게 복사에 대한 실행 여부를 묻는다
-f, --force	복사대상 파일이 있을 경우, 사용자에게 확인없이 강제로 복사한다
-r, -R, --recursive	디렉토리를 복사할 경우 하위 디렉토리와 파일을 모두 복사한다
-v, --verbose	복사진행 상태를 출력한다
-d, --no-dereference	복사대상 파일이 심볼릭 파일이면, 심볼릭 정보를 그대로 유지한 상태로 복사한다
-p, --preserve	원본 파일의 소유주, 그룹, 권한, 시간정보를 보존 하여 복사한다
-a, --archive (-dpr)	원본 파일의 속성, 링크정보들을 그대로 유지하면서 복사한다.

Basic Command

🧠 기본 명령어

🧠 mv : 파일이동

파일이나 디렉토리를 이동하거나 이름을 바꿀때 사용한다.

Syntax : mv [-option] [sources] [target]

🧠 Option

옵션	의미
-i, --interactive	기본적으로 .bashrc에 alias되어 있는 옵션으로, 이동할 위치에 동일한 파일이 있을 경우 사용자에게 확인
-u, --update	이동할 파일이 이동할 위치에 있는 파일보다 최근 파일일 경우에만 이동한다
-b, --backup	대상 파일이 이미 있어, 지워지는 것을 대비해 백업파일을 생성한다
-f, --force	대상 파일이 이미 있어도 사용자에게 어떻게 처리할지를 묻지 않는다
-v, --verbose	파일 을 옮기는 과정을 자세하게 보여준다
-S, --suffix	-b 옵션을 이용하여 백업할 경우 백업파일에서 사용할 파일 이름의 꼬리 문자를 지정한다.

Basic Command

🧠 기본 명령어

🧠 mkdir(make directory) : 디렉토리 생성

디렉토리 생성 명령어

Syntax : mkdir [-option] [directory name]

🧠 Option

옵션	의미
-m, --mode	디렉토리 생성시 디렉토리의 기본 권한을 지정한다
-p, --parents	필요한 경우 상위 경로까지 생성한다
--help	도움말 표시
--version	버전 정보 표시

Basic Command

🌐 기본 명령어

🌐 rm : 파일 및 디렉토리 삭제
파일이나 디렉토리를 삭제하는 명령(권한이 있을 경우)
Syntax : rm [-option] [directory / file]

🌐 Option

옵션	의미
-f, --force	파일/디렉토리 삭제시 사용자에게 어떻게 처리할지 물어보지 않는다
-r, -R, --recursive	일반 파일이면 그냥 지우고, 디렉토리면 디렉토리를 포함한 하위 경로와 파일을 모두 지운다
-v, --verbose	각각의 파일 지우는 정보를 자세하게 보여준다
--version	버전 정보를 보여준다

Basic Command

🌐 기본 명령어

🌐 alias : 별칭지정 명령

- 🌐 복잡한 명령어와 옵션을 간단히 입력할 수 있는 문자열로 치환한다.
현 세션에서만 적용을 받는다.

🌐 alias 지정 / unalias 로 해제

```
# al i as / unal i as
```

🌐 명령어와 옵션 지정하여 사용하기

```
# al i as shut='shutdown -h now'
```

🌐 새로운 문자열에 기존 명령어 지정하기

```
# al i as end='cl ear'
```

Basic Command

🌐 기본 명령어

🌐 alias : 별칭지정 명령

- 🌐 시스템 수준에서 별칭을 정의 할 경우 /etc/bashrc 를 편집하고,
- 🌐 사용자 수준에서 별칭을 지정 할 경우 각 사용자의 홈 디렉터리에 있는 .bashrc 파일을 편집한다.

[적용예제]

```
1 # /etc/bashrc
2
3 # System wide functions and aliases
4 alias end='clear'
5 alias ls='ls -al'
6
```

Basic Command

🌐 기본 명령어

🌐 cat : 파일 내용 출력

🌐 텍스트 파일 내용을 표준 출력장치로 출력하는 명령

🌐 파일 내용 출력하기

```
# cat /etc/passwd
```

🌐 기존의 파일 내용을 다른 파일로 입력하기

```
# cat /etc/passwd > /testfile
```

🌐 기존 파일에 내용 추가하기

```
# cat >> /testfile
```

🌐 옵션 사용

```
# cat -n /etc/passwd
```

Basic Command

🌐 기본 명령어

🌐 touch : 파일 시간정보 변경 및 파일 생성

- 🌐 크기가 0인 새로운 파일을 생성 하거나 파일이 기존에 존재하는 경우 수정 시간을 변경하는 명령어
- 🌐 파일이 존재하지 않을 경우 0byte 파일 생성

[참고]stat 파일명을 사용하여 시간 정보를 자세히 볼 수 있다.

```
[root@localhost data]# stat AnnualReport
  File: 'AnnualReport'
  Size: 72          Blocks: 8          IO Block: 4096   regular file
Device: fd00h/64768d Inode: 914016       Links: 1
Access: (0644/-rw-r--r--)  Uid: (   0/   root)   Gid: (   0/   root)
Access: 2013-06-04 08:37:05.011996608 +0900
Modify: 2013-06-04 08:38:25.781994735 +0900
Change: 2013-06-04 08:38:25.781994735 +0900
```

Basic Command

🌐 touch -r

```
[root@localhost data]# touch -r AnnualReport GrandTotal
[root@localhost data]# stat GrandTotal
  File: 'GrandTotal'
  Size: 0                Blocks: 0                IO Block: 4096   regular empty file
Device: fd00h/64768d    Inode: 914017           Links: 1
Access: (0644/-rw-r--r--)  Uid: (   0/   root)   Gid: (   0/   root)
Access: 2013-06-04 08:37:05.011996608 +0900
Modify: 2013-06-04 08:38:25.781994735 +0900
Change: 2013-06-04 08:41:00.335013877 +0900
```

🌐 touch -t

```
[root@localhost data]# touch -t 203301010000.00 future
[root@localhost data]# stat future
  File: 'future'
  Size: 0                Blocks: 0                IO Block: 4096   regular empty file
Device: fd00h/64768d    Inode: 914018           Links: 1
Access: (0644/-rw-r--r--)  Uid: (   0/   root)   Gid: (   0/   root)
Access: 2033-01-01 00:00:00.000000000 +0900
Modify: 2033-01-01 00:00:00.000000000 +0900
Change: 2013-06-04 08:45:13.128998771 +0900
```

Basic Command

🌐 빈 파일 생성

```
[root@localhost data]# touch data1 data2 data3
[root@localhost data]#
[root@localhost data]# ls data*
-rw-r--r--. 1 root root 0 Jun  4 09:28 data1
-rw-r--r--. 1 root root 0 Jun  4 09:28 data2
-rw-r--r--. 1 root root 0 Jun  4 09:28 data3
```

Basic Command

🌐 기본 명령어

🌐 head

- 🌐 파일의 내용 중 처음부터 아래로 10줄 출력
- 🌐 head 명령어만 사용시 기본값인 위에서 10줄을 출력

```
# head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
.....
<중간 생략>
.....
news:x:9:13:news:/etc/news:
```

🌐 head -<n> /etc/passwd

```
# head -1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
```


Basic Command

🌐 기본 명령어

🌐 tail

- 🌐 파일의 내용중 마지막부터 위로 10줄 출력
- 🌐 tail 명령어만 사용시 기본값인 아래에서 10줄을 출력

```
# tail /etc/passwd
htt:x:100:101:||||MF Htt:/usr/lib/im:/sbin/nologin
.....
<중간 생략>
.....
smbtest:x:501:501::/home/smbtest:/bin/bash
xcurelab:x:502:502::/home/xcurelab:/bin/bash
```

- 🌐 tail -<n> /etc/passwd

```
# tail -1 /etc/passwd
xcurelab:x:502:502::/home/xcurelab:/bin/bash
```

- 🌐 tail -f /var/log/dmesg
(모니터링을 하고자 할 경우)

Basic Command

● 기본 명령어

● more

- 내용이 많은 파일을 화면단위로 끊어서 출력한다.
- more /etc/passwd

```
# more /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
--More--(7%)
```

● ls 명령어와 조합해서 사용하기

```
# ls -l /etc/ | more
합계 3264
-rw-r--r--  1 root root      2518  8월 13  2006 DIR_COLORS
-rw-r--r--  1 root root      2434  8월 13  2006 DIR_COLORS.xterm
drwxr-x---  2 root pegasus  4096 10월 10  2007 Pegasus
--More--
```

Basic Command

🌐 기본 명령어

🌐 rdate

- 🌐 타임서버에서 시간 정보를 얻어 시스템의 시간을 변경한다.

- 🌐 rdate 명령을 이용하여 타임서버의 현재시간 확인하기

```
# rdate -p time.bora.net 또는 203.248.240.140
```

- 🌐 rdate 명령을 이용하여 HOST 시간 타임서버와 동기화 하기

```
# rdate -s time.bora.net 또는 203.248.240.140
```

🌐 주요 타임 서버 리스트

- 🌐 time.bora.net
- 🌐 gps.bora.net
- 🌐 ntp1.cs.pusan.ac.kr
- 🌐 ntp.ewha.net

Basic Command

🌐 기본 명령어

🌐 file

- 🌐 확장자를 기본으로 사용하지 않는 리눅스의 파일 종류 확인하기
- 🌐 file 명령어를 이용한 파일 유형 확인

```
# file /bin/ls
/bin/ls: ELF 32-bit LSB executable, Intel 80386, version 1
(SYSV), for GNU/Linux 2.2.5, dynamically linked (uses shared
libs), stripped
```

- 🌐 file 명령어를 이용한 디스크 파일시스템 종류 확인

```
# file -s /dev/sda1
/dev/sda1: Linux rev 1.0 ext3 filesystem data (needs journal
recovery)
```

Basic Command

🌐 기본 명령어

🌐 find

- 🌐 파일 및 디렉토리 검색
- 🌐 파일 이름으로 검색

```
# find <경로> -name <파일명>
```

- 🌐 Access time 이 n 일 보다 작은

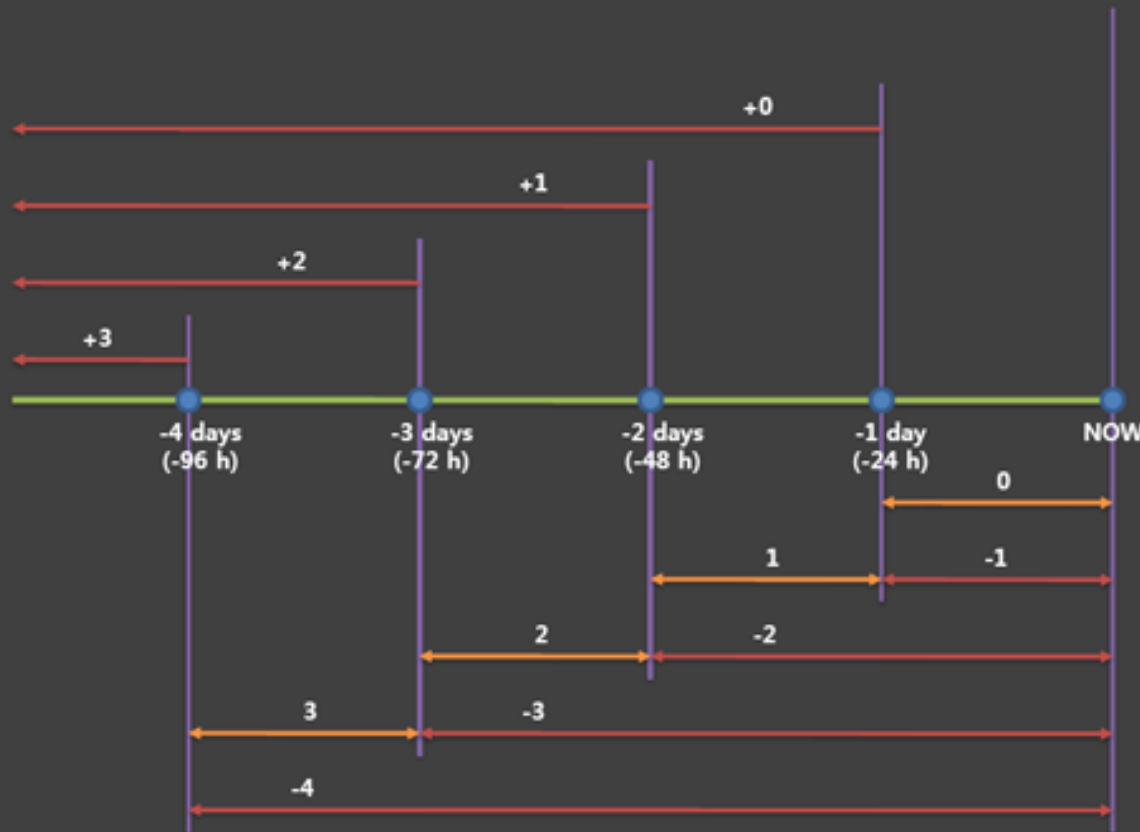
```
# find <경로> -atime -n
```

- 🌐 Access time 이 n 일 보다 큰 파일

```
# find <경로> -atime +n
```

Time line

-atime, -mtime, -ctime N



atime : 파일을 열어본
시간 또는 디렉토리에
cd 명령으로 접근한 시간

mtime : 파일의 내용이
변경된 시간 ls -l 에서
나오는 시간

ctime : 파일의 정보가
변경된 시간 chmod,
chown 등으로.

Basic Command

🔹 기본 명령어

🔹 find(계속)

- 🔹 test 파일 이후에 수정된 모든 파일을 찾기

```
# find /home/ -newer test
```

🔹 명령수행

```
# find . -name "test*" -exec rm {} \;
```

- 🔹 root 권한으로 실행되는 파일 찾기

```
# find . -user root -perm +4000 2> /dev/null
```

🔹 xargs

```
# find . -name "test*" -type f | xargs rm -f
```

Basic Command

🔍 빈 파일 찾기

```
[root@localhost data]# find . -empty -type f
./data3
./data1
./data2
[root@localhost data]# find . -empty -type f -exec rm -rf {} \;
[root@localhost data]#
[root@localhost data]# find . -empty -type f
```

🔍 백업파일 찾아서 지우기

```
[root@localhost data]# find . -name "*~" | xargs rm -f
```


실습

📍 기본 명령어

📍 실습

- 📍 /down 디렉토리를 생성하시오
- 📍 /down 디렉토리에 test1, test2 파일을 생성 하시오
- 📍 /down 디렉토리에 있는 test1, test2 파일을 각각 test3, test4 파일로 /tmp 디렉토리에 복사하시오
- 📍 /down 디렉토리에 test 디렉토리를 생성하시오
- 📍 /down/test 디렉토리를 /tmp 디렉토리로 이동 하시오
- 📍 Find 명령어를 통해 파일명 앞부분에 "test" 문자열이 섞인 파일을 검색하면서 삭제 하시오
(/tmp 디렉토리와 /down 디렉토리로 만 부분검색을 하고 명령어 한 줄로)

Browsing the Filesystem

Filesystem & Hierarchy

🌐 파일유형(type)

기호	의미
-	일반 파일
b	블록형 특수 장치파일(주로 디스크, /dev/sda 등)
c	문자형 특수 장치파일 (주로 입출력에 사용, /dev/console등)
d	디렉토리를 의미
p	파이프 파일임을 의미
s	소켓 파일임을 의미
l	심볼릭 링크파일임을 의미

Filesystem & Hierarchy

● 파일유형(type)

● 블록 장치 파일(b)

- 일정 크기의 블록 단위로 입출력을 하고 커널 내에 입출력 버퍼 캐쉬 기능이 있음
 - EX) hd? : IDE 하드 디스크, 플로피 디스크, CDROM
 - sd? : SCSI, SATA, or USB Storage
 - md? : Software RAID

● 문자 장치 파일(c)

- 블록 장치가 아닌 모든 입출력 장치를 문자 장치로 정의함
 - EX) 터미널 장치, 네트워크 장치, NULL

● 특수 파일(p)

- 보통 파이프라 불리는 프로세스간 통신을 위한 특수 파일
- 사용중이 아닐 때는 크기가 0 임

Filesystem & Hierarchy

- 파일 유형(type)

- 소켓 특수 파일(s)

- 한 컴퓨터 내에 있는 프로세스간 통신을 위한 특수 파일
- 자료를 저장하지 않기 때문에 크기는 항상 0 임

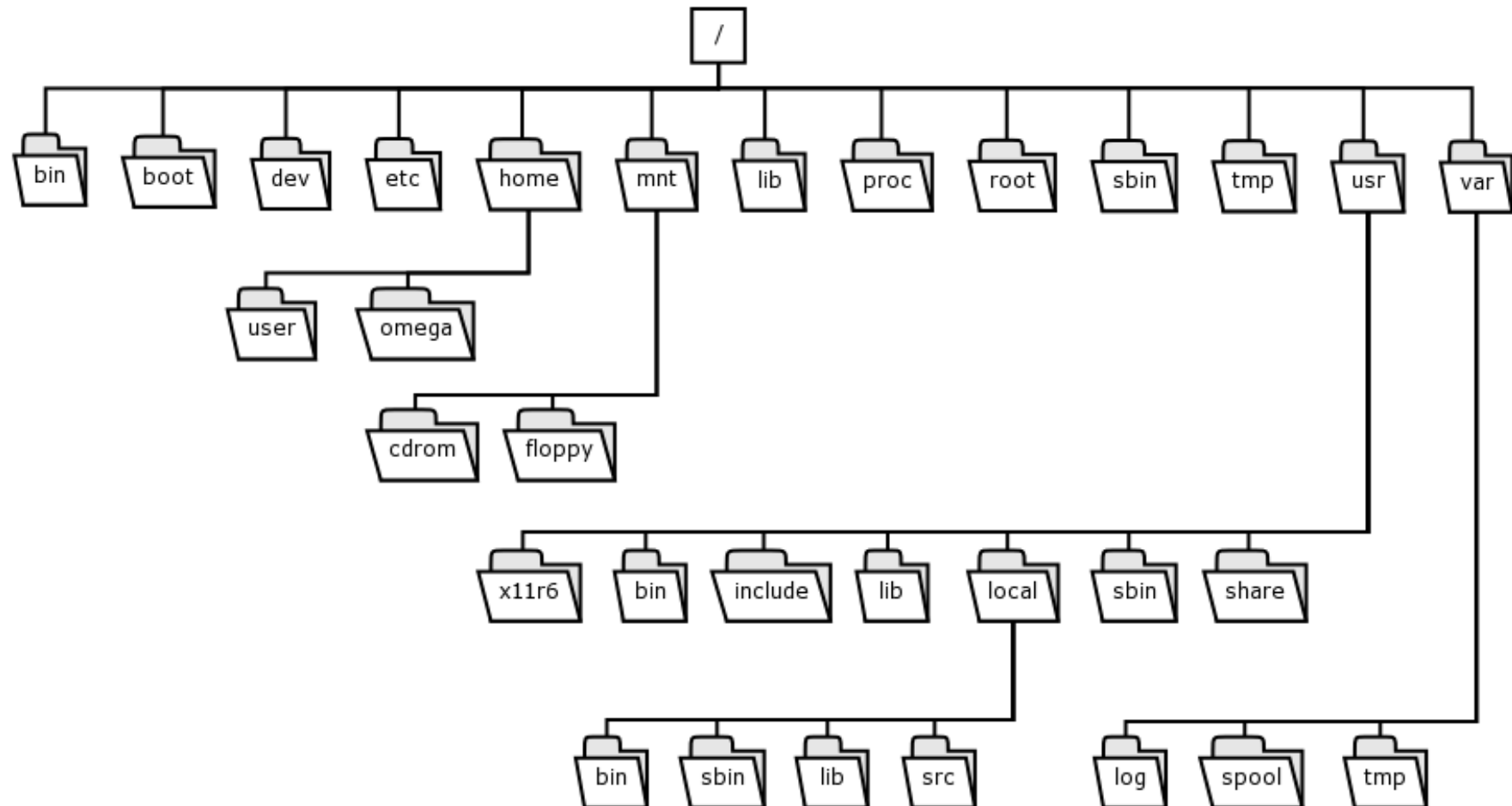
- 심볼릭 링크(l)

- 마이크로소프트의 "바로가기" 개념과 동일하며, 이는 심볼릭 링크를 모방한 것임

Filesystem & Hierarchy

● 기본 파일 시스템 및 계층구조

● 주요 디렉토리 계층구조



Filesystem & Hierarchy

📍 주요 디렉토리

📍 /

- 📍 리눅스 상에 존재하는 모든 파일과 디렉토리의 최상위에 위치하는 최상위 디렉토리 이다

📍 /boot

- 📍 리눅스 커널의 메모리 이미지와 부팅 과정에서 필요한 정보 파일들이 있다

📍 /bin

- 📍 기본적인 명령어들이 있다

📍 /root

- 📍 슈퍼 유저 root의 홈 디렉토리이다

Filesystem & Hierarchy

📍 주요 디렉토리

📍 /home

- 📍 관리자 이외의 사용자의 홈 디렉토리가 생성되는 위치이다

📍 /etc

- 📍 시스템이나 사용자 관리를 위한 자료 파일이나 관리자용 명령들을 가지고 있다

📍 /dev

- 📍 장치 파일이라고 하는 파일시스템과 하드웨어간의 인터페이스를 담당하는 파일이 들어 있다
(/dev/modem, /dev/console)

📍 /lib

- 📍 각종 언어를 위한 라이브러리를 가진다

Filesystem & Hierarchy

🌐 주요 디렉토리

🌐 /mnt

- 🌐 Remote Device 를 이용하기 위한 디렉토리

🌐 /media

- 🌐 Local Device 를 이용하기위한 디렉토리

🌐 /sbin

- 🌐 시스템운영 및 관리를 위한 명령 파일들이 있다. 부팅과정에 필요한 명령들은 여기에 있고 정상 동작 상태에서 필요한 크기가 큰 명령들은 /usr/sbin 에 있다

🌐 /usr

- 🌐 루트 파일 시스템과 구조는 유사하고 용량이 크고 자주 사용되지 않는 파일들이 있다

Filesystem & Hierarchy

🌐 주요 디렉토리

🌐 /proc

- 🌐 커널과 프로세스 정보를 얻을 수 있는 가상 파일 시스템이다.
- 🌐 관리자는 이 디렉토리에 대해서 잘 알고 있어야 한다

🌐 /tmp

- 🌐 잠시 사용되는 임시 파일들을 위한 디렉토리 이다

🌐 /var

- 🌐 /var/log 나 /var/adm 과 같이 자주 변경되는 시스템 파일들을 가지고 있다

Filesystem & Hierarchy

● Absolute & Relative Pathnames

● 절대경로

- 리눅스 파일 시스템에서 절대 경로는 파일 시스템 전체를 기준으로 파일이나 디렉토리의 절대적인 위치를 기준으로 시작하는 경로를 말한다.

● 상대경로

- 리눅스 파일 시스템에서 상대 경로는 현재 작업하고 있는 디렉토리 에서 파일이나 디렉토리의 상대적인 위치를 말한다

Vi Editor

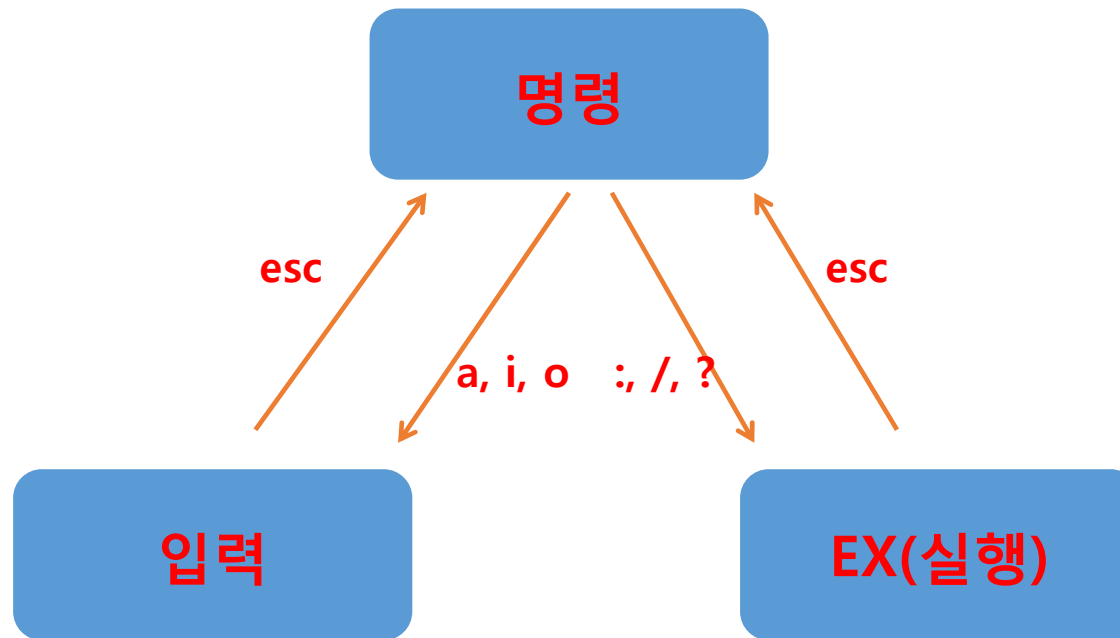
Vi and Vim text editor

🌐 What is VI editor?

- 🌐 Emacs와 함께 유닉스 환경에서 많이 쓰이는 문서 편집기
- 🌐 1976년 빌 조이가 초기 BSD 릴리즈에 포함될 편집기로 제작
- 🌐 줄 단위 편집기가 아닌 한 화면을 편집하는 Visual editor 이라는 뜻에서 유래
- 🌐 명령, 실행(EX), 입력 모드가 있음

Vi and Vim text editor

Mode



Vi and Vim text editor

- 명령모드
- Vi의 기본모드
- Vi가 처음 실행되거나, 입력 모드에서 ESC 키를 누른경우
- 커서이동, 문자열 수정, Copy & Paste 등
- 엔터키를 누를 필요 없이 각각의 명령에 따른 버튼을 누르면 바로 실행됨

Vi and Vim text editor

● 입력모드

- 버퍼에 내용을 입력할 수 있는 모드
- 명령 상태에서 a, i, o 등의 키를 누르면 진입
- 하단에 --INSERT-- 라고 표시됨
- R을 누른 경우에는 --REPLACE-- 라고 표시됨

Vi and Vim text editor

🔵 명령모드

🔵 커서이동

KEY	동작
h	커서를 왼쪽으로 이동
j	커서를 아래로 이동
k	커서를 위로 이동
l	커서를 오른쪽으로 이동

Vi and Vim text editor

🔵 명령모드

🔵 커서이동(단어단위)

KEY	동작
W,w	다음 단어의 처음으로 이동
E,e	단어의 끝으로 이동
B,b	단어의 처음으로 이동

Vi and Vim text editor

🔵 명령모드

🔵 커서이동(행단위)

🔵 커서이동(문서단위)

KEY	동작
0(zero)	행의 처음으로 이동
\$	행의 마지막으로 이동
G	문서의 마지막으로 이동
gg	문서의 처음으로 이동

Vi and Vim text editor

🔵 명령모드

🔵 마크이동

🔵 책갈피 기능

🔵 설정방법

EX1) m? : ?는 a~z, A~Z 중 하나

🔵 이동방법

EX1) back quote + ? : 마크된 정확한 위치로 이동

EX2) quote + ? : 마크된 줄의 처음으로 이동

EX3) ": 원래 위치로 되돌아 감

Vi and Vim text editor

명령모드

KEY	동작
i	현재 위치에서 입력 모드로 변경
I	행의 제일 처음에서 입력 모드로 변경
a	현재 위치에서 우측으로 한 칸 이동 후 입력 모드로 변경
A	행의 제일 마지막에서 입력모드로 변경
o	커서 아래에 새로운 행을 추가하고 입력모드로 변경
s	현재 문자를 지우고 입력모드로 변경
S	현재 행의 모든 문자를 지우고 입력모드로 변경

Vi and Vim text editor

명령모드

KEY	동작
x	커서가 있는 문자 삭제
X	커서가 있는 앞 문자 삭제
dd	현재 커서의 행 삭제
숫자 + dd	현재 커서부터 숫자만큼 행 삭제
yy	현재 커서가 있는 라인을 복사
숫자 + yy	현재 커서부터 숫자만큼의 행을 복사
p	복사한 내용을 현재 라인 이후에 붙여넣기
P	복사한 내용을 현재 라인 이전에 붙여넣기
u	바로 전 수행 명령 취소
숫자 + u	숫자 만큼 뒤로 취소

Vi and Vim text editor

● EX 모드

● 검색

● :/

● 패턴이 검색 된 후 n 키를 통해 아래 방향으로 계속 찾기

● 패턴이 검색 된 후 N 키를 통해 위 방향으로 계속 찾기

● :?

● 패턴이 검색 된 후 N 키를 통해 아래 방향으로 계속 찾기

● 패턴이 검색 된 후 n 키를 통해 위 방향으로 계속 찾기

Vi and Vim text editor

- EX 모드

- 치환

- [범위]s/[Old]/[New]/[옵션]

- 범위는 n 혹은 n,n 혹은 % 를 넣을 수 있다
- g 옵션을 주면 적용되는 라인의 모든 부분 변경
- g 옵션을 주지 않으면 처음 찾은 부분만 변경

Vi and Vim text editor

- EX 모드

- Shell 명령어

 - :! [command]

 - vi 를 잠시 중단하고 명령어 수행.

 - ::! [command]

 - 수행한 명령의 결과를 vi 편집기로 출력한다.

Vi and Vim text editor

- EX 모드

- 수평 나누기

[n]split[filename]

VIM에서는 :new

- 수직 나누기

[n]vs[filename]

VIM에서는 :vs

- 명령모드에서 Ctrl + wn 은 현재 화면을 수평으로 나누기
- 명령모드에서 Ctrl + wv 는 현재 화면을 수직으로 나누기
- n 은 창의 크기를 의미
- Ctrl + ww 창간의 이동

Vi and Vim text editor

🔵 EX 모드

🔵 파일관련

KEY	동작
:e [filename]	파일열기
:enew	현재 창을 닫고 빈 문서를 연다
:q	종료(변경된 내용이 없는 경우)
:q!	강제 종료(변경된 내용이 있어도 무시)
:w	파일 저장
:wq	파일 저장 후 종료

Vi and Vim text editor

EX 모드

파일 및 실행 관련

KEY	동작
:w > > [filename]	Filename에 내용추가
:f	현재 작업중인 파일의 이름과, 라인 수
:[n]r[filename]	filename 파일의 내용을 현재 편집중인 파일의 n 라인부터 삽입
:[n]r![command]	Command 실행결과를 파일의 n 라인부터 삽입

Vi and Vim text editor

🌐 EX 모드

🌐 ETC

KEY	동작
:set ts=4	Tab 사이즈 조절
:set nu	Line number 활성화
:set nonu	Line number 비활성화
:nohl	High light 비활성화

Vi and Vim text editor

🔹 설정파일

🔹 /usr/share/vim

- 🔹 vim 설정파일이 위치한 디렉토리

🔹 /etc/vimrc

- 🔹 시스템에 로그인 하는 모든 사용자가 공통적으로 적용받는 vim 설정파일

🔹 사용자 개별 설정파일

- 🔹 /usr/share/vim/vim70/vimrc_example.vim 을 사용자의 홈 디렉토리에 .vimrc 로 복사한 후 수정한다

```
# cp /usr/share/vim/vim70/vimrc_example.vim ~/.vimrc
```

Shell

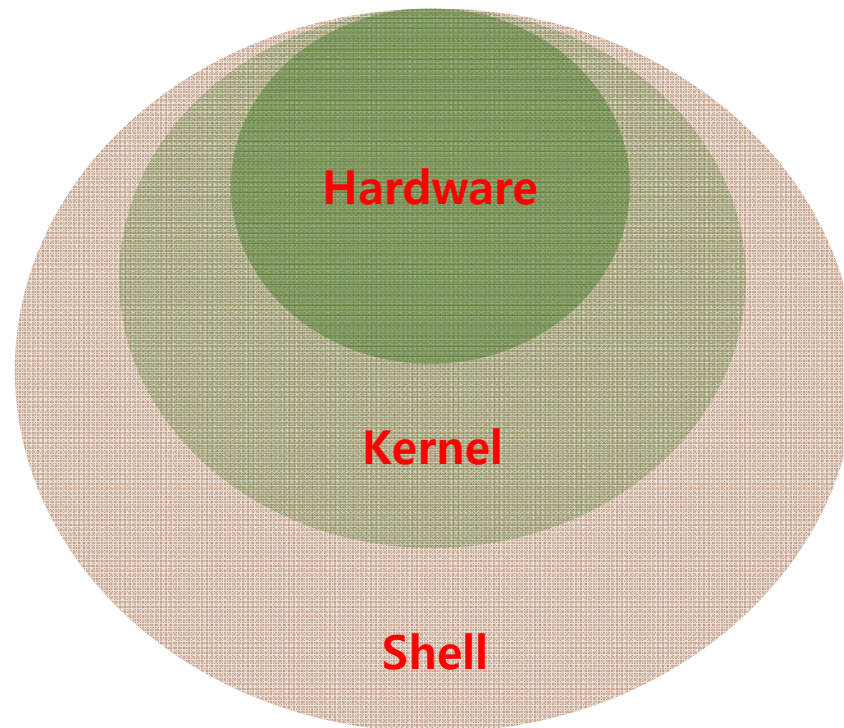
Shell

🔵 What is Shell?

- 🔵 유닉스에서 대화형 사용자 인터페이스를 말함
- 🔵 사용자가 입력하는 명령어를 이해하고, 실행하는 역할
- 🔵 운영체계의 바깥 계층에 위치하며, 사용자와 커널의 의사소통을 담당
- 🔵 cshell, bourneshell, bashshell, tcshell, kornshell, 등 다양한 종류가 있음

Shell

🌐 Where is Shell?



Shell

🧠 Shell 의 역할

- 🧠 입력을 읽고 해당 명령행을 분석
- 🧠 특수 문자들을 평가함
- 🧠 파이프, 리다이렉션, 백그라운드 프로세스를 설정
- 🧠 시그널을 처리

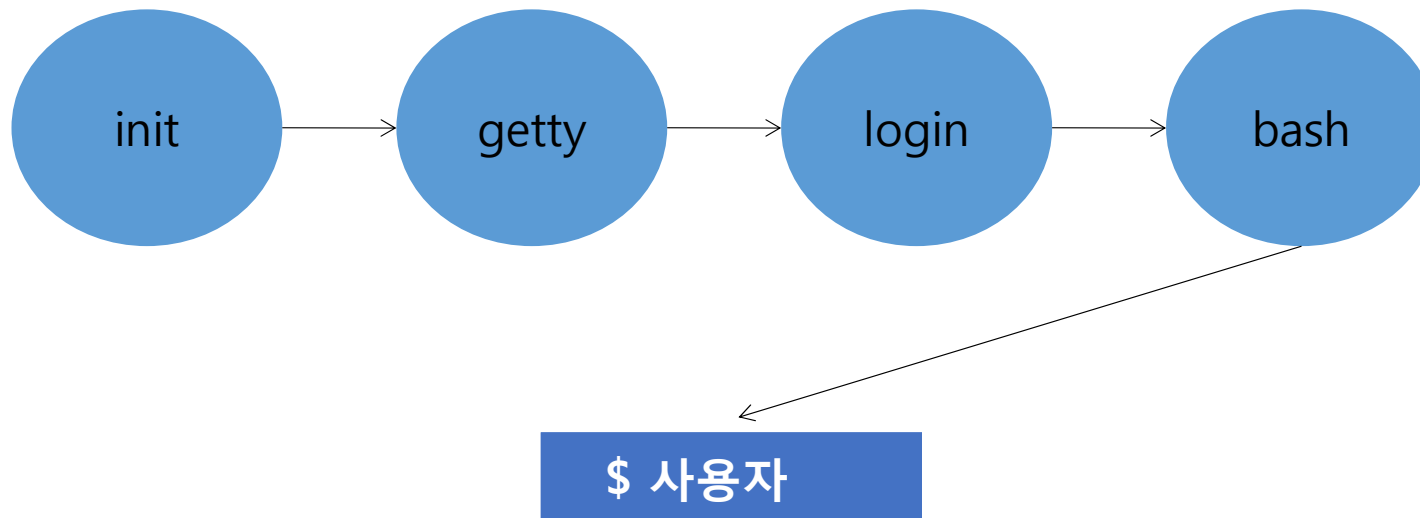
Shell

🌐 Bash Shell

- 🌐 다양한 내장 명령어, 히스토리, 별명, 파일, 명령어 완성 기능, 명령행 편집등의 기능을 지원
- 🌐 2.X 버전에 접어들면서 Corn Shell 과 C Shell에서 제공하는 기능들도 많이 추가되어 있음
- 🌐 Bourn Shell 과 하위 호환성도 유지하고 있음

Shell

🌐 Bash Shell 구동 과정



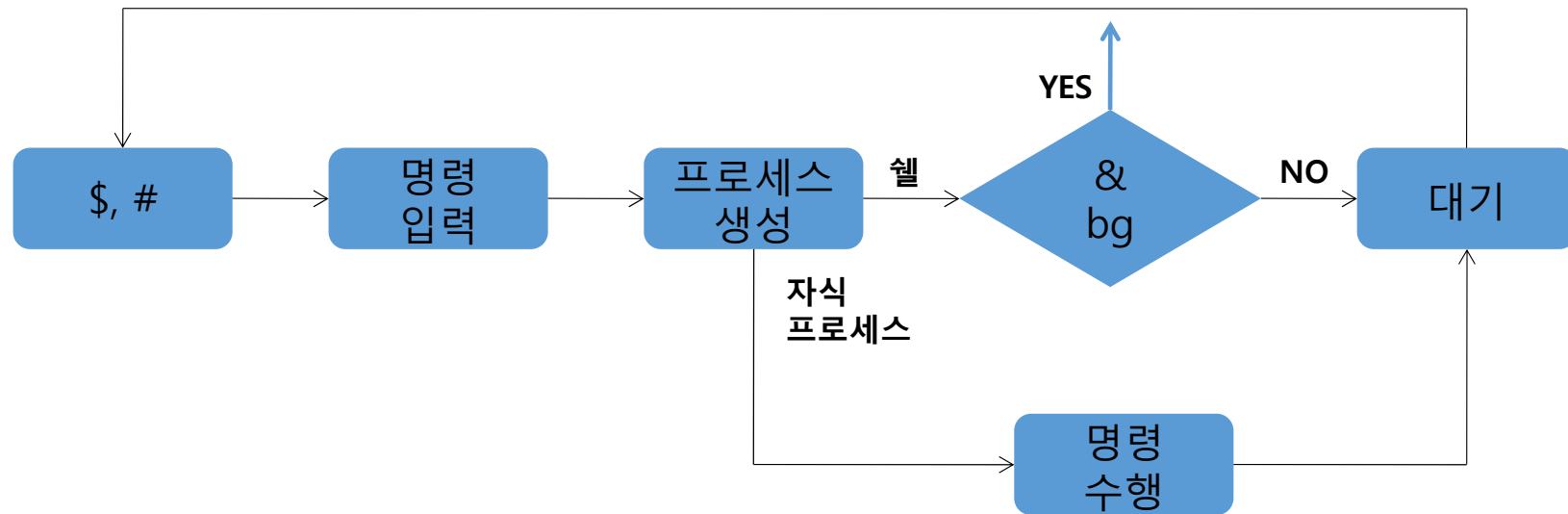
Shell

🐼 Bash Shell 구동 과정

- 🐼 최초로 구동되는 프로세스는 PID 1 의 init 프로세스
- 🐼 init 프로세스가 /bin/mingetty 프로세스를 실행시킴
- 🐼 사용자 및 패스워드 입력 후 /etc/passwd 에서 해당 사용자에게 설정된 shell(Bash shell) 을 실행시킴
- 🐼 시스템 파일은 /etc/profile 을 찾아서 그 안의 명령어 들을 실행 시킴
- 🐼 사용자의 홈 디렉토리에서 .bash_profile 의 내용을 읽음

Shell

💡 Shell 의 명령 수행과정



Shell

🌐 Shell 의 기능



Shell

● 변수

● 변수의 종류

● 지역 변수

- 자신을 생성한 셸에서만 사용할 수 있다.

● 환경 변수

- 자신을 생성한 셸뿐만 아니라, 이로부터 파생되어 나온 자식 프로세스에서도 사용할 수 있다.

● 변수 명칭 표준

- 변수 이름에는 알파벳 a~z, A~Z, 0~9, 밑줄문자(_) 63개의 문자를 사용 할 수 있으며 첫 글자는 알파벳 혹은 밑줄문자(_) 만 쓸 수 있다.

● 변수에 값 할당하기

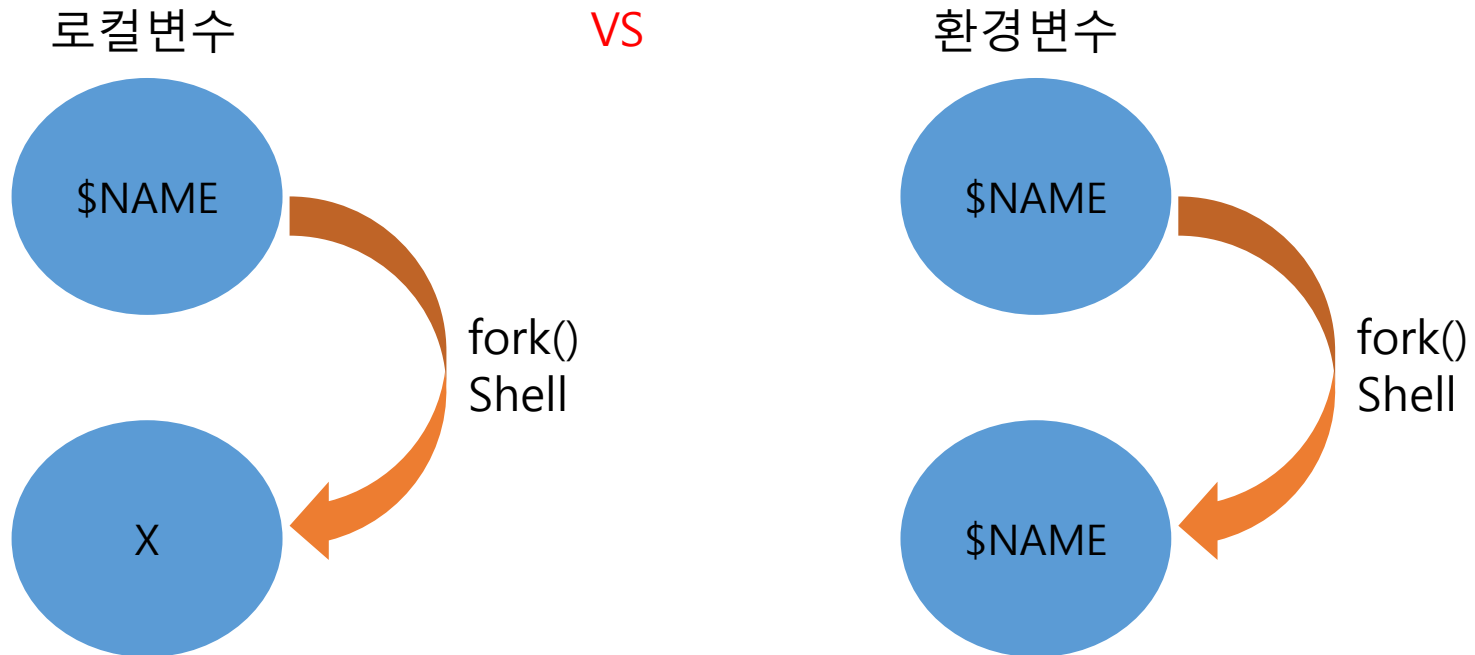
- 등호(=) 를 사용하며 등호(=) 양쪽에는 공백이 올 수 없다.

Variable=value / Variable="value value value"

Shell

● 변수

● 변수치환



Shell

- 지역 변수

- 지역변수 설정

- 형식

```
변수명=변수값
```

- 지역 변수 A에 ITBank라는 값을 지정

```
# A=I TBank or # decl are A=I TBank
```

- 지역 변수 호출

- 지역 변수값 호출

```
# echo $A / # echo $B
```

Shell

🌐 환경 변수

🌐 환경 변수 설정

- 🌐 환경 변수는 POSIX 표준에 의해 대문자를 사용한다.

🌐 형식

```
# 변수명=변수값 ; export 변수명  
# export 변수명=변수값  
# declare -x variable=value
```

🌐 환경 변수 NAME 생성

```
# export NAME="Choi Young"
```

🌐 환경 변수 를 지역 변수로 변환

```
# export -n 환경변수  
# declare -a 환경변수
```

Shell

🔹 읽기 전용 변수

- 🔹 재정의나 설정해제를 할 수 없는 특수 변수이다.
- 🔹 C 라는 변수를 readonly 명령을 사용하여 읽기전용 변수로 지정

```
# C="ReadOnly Local Variable"
# readonly C
# C="Update"

# declare -r C="ReadOnly Local Variable"
# declare C="Update"
```

🔹 설정변수 확인 및 삭제

- 🔹 설정 변수 확인

```
# env # printenv # declare
```

- 🔹 변수 해제

```
# unset 변수명
```

Shell

BASH 주요 환경변수

환경 변수	의 미
ENV	.bashrc 파일에서 설정하며 함수와 별명 등을 설정하는 환경 파일의 이름을 설정
EUID	Shell이 시작할 때 현재 사용자의 유효 ID를 확장한다.
HISTFILE	명령어 히스토리를 저장할 파일이름. 기본값은 ~/.bash_history이다.
HOME	사용자의 홈 디렉토리의 값을 가지고 있으며 특정 디렉토리를 지정하지 않고 cd 명령어를 사용할 때 이용한다.
LANG	LC_로 시작하는 변수들에 포함되지 않는 로케일을 설정
PATH	명령어의 검색 경로, 콜론으로 구분한 디렉토리 목록
PPID	부모프로세스의 PID
PWD	현재 작업 디렉토리
SHELL	현재 사용되어지고 있는 SHELL
UID	현재 사용자의 UID

Shell

● 치환

● 변수치환

- 변수는 리눅스 사용환경 구성을 위한 설정에 사용됨
- Windows 의 경우 환경 설정을 위해 환경설정 파일, 레지스트리, 환경변수 등을 사용한다
- Unix 의 경우 환경 설정을 위해 환경설정파일, 환경변수 등을 사용한다

Shell

● 치환

● 변수치환

- 기존의 환경변수에 추가정보 구성하기

```
# export 변수명=$변수명: 추가할 디렉토리
```

```
# export PATH=$PATH: /home/test/bi n
```

- alias와 유사한 기능으로 명령어와 옵션을 묶어서 사용

```
# M="/bi n/mkdi r"  
# T="/bi n/touch"  
# R="/bi n/rm -rf"  
  
# $M /down/test_di r  
# $T /down/test_fi l e  
# $R /down/{test_di r, test_fi l e}
```

실습

🌐 치환

🌐 변수치환 실습

🌐 명령어 들을 다음과 같은 변수명으로 정의 하시오

🌐 cp -> MYCP

🌐 mv -> MYMV

🌐 rm -> MYRM

🌐 cat -> MYCAT

🌐 touch -> MYTOUCH

🌐 정의된 변수만을 사용하여 testfile1을 생성하고, 이 파일을 testfile2 로 복사 하시오

🌐 정의된 변수만을 사용하여 생성된 두 개의 파일을 삭제 하시오

🌐 HISTFILE의 변수를 이용하여, MYCAT 변수를 이용해 .bash_history 파일의 내용을 확인 하시오

Shell

❶ 치환

❶ 명령어 치환

- ❶ Back quote(`) 사이에 있는 command는 수행 결과 값으로 치환

```
# 명령어 `명령어`
```

❶ 서비스 변경내용 적용하기

```
# kill -HUP `cat /var/run/xinetd.pid`
```

❶ echo 명령이 수행되어 test 만 출력이됨

```
# echo `echo test`
```

실습

🔵 치환

🔵 명령어치환 실습

- 🔵 echo 명령어와 명령어 치환을 사용하여 다음과 같이 날짜 및 시간정보를 출력 하시오

```
# echo XXXXXXXXXX  
Now Date is : 2008. 08. 27. (수) 20:01:28 KST
```

Shell

🐚 치환

🐚 틸드(~) 치환

- 🐚 ~(tilde) 는 사용자의 home 디렉토리를 의미한다

🐚 홈 디렉토리 출력

```
# echo ~
```

🐚 지정한 사용자의 홈 디렉토리로 이동

```
# cd ~사용자계정명
```

🐚 바로 이전의 작업 디렉토리로 이동

```
# cd ~- // $OLDPWD
```

Shell

🧠 쿼팅

🧠 What is Quoting

치환을 억제 하거나 문자의 특수한 의미를 제거 한다

	변수치환	명령치환	Tilde 치환	범위
" (Double Quote)	허용	허용	허용안함	" ~ "
' (Single Quote)	허용안함	허용안함	허용안함	' ~ '
₩(Back Slash)	허용안함	허용안함	허용안함	₩ 뒤 한문자

Shell

❶ 쿼팅

❶ 쿼팅 사용예제

❶ 쿼터

```
# echo ' $HOSTNAME'
```

❷ 이중쿼터

```
# echo " $HOSTNAME"
```

❸ Back slash

```
# echo \ $HOSTNAME
```

Standard I/O Redirection

Standard I/O Redirection

표준 입력과 출력

- 표준 입출력 기능은 입력/출력을 다루는 기본적인 방법을 제공
기본 입출력 스트림(I/O stream)에는 **표준 입력**, **표준 출력**, **표준 에러** 가 있다

표준 입력 (0) ---> 키보드

표준 출력 (1) ---> 모니터

표준 에러 (2) ---> 모니터

표준 입출력에 사용하는 기호

기 호	기 능
>	쓰기
<	읽기
>>	추가
2>	에러의 방향을 바꿈
1>&2	출력을 에러로 보냄
2>&1	에러를 출력으로 내보냄

Standard I/O Redirection

- 표준 입력과 출력

- 표준 입력 사용

- 일반적으로 키보드를 주로 사용하며, 파일을 직접 열수 있는 명령은 표준입력 스트림을 지정하지 않아도되지만 파일을 직접 열수 없는 명령어에서는 표준 입력 스트림을 유용하게 사용할 수 있다

```
# cat test_file  
  
# mail user1 < send.txt
```

- 지정된 문자열 "C" 가 입력되면 입력 중지

```
# cat << c > test_file
```


Standard I/O Redirection

표준 입력과 출력

표준 출력 사용

- 키보드나 파일로 부터 입력 받은 값을 표준 출력 장치(모니터)로 출력

```
# cat  
# cat test_file  
# cat < test_file
```

- 리다이렉션 기호를 사용하면 표준 출력 장치가 아닌 파일로 출력

```
# cat > output  
# cat test_file > output1  
# cat < test_file > output2
```

- 더블 리다이렉션 기호를 사용하면 표준 출력 되어질 파일에 내용 추가

```
# cat >> output  
# cat test_file >> output1  
# cat < test_file >> output2
```

Standard I/O Redirection

🌐 Redirection 사용예제

🌐 명령어 예제

- 🌐 루트 디렉토리에서 .c 로 끝나는 파일 이름들을 검색하여, 찾아진 이름들을 foundit 파일로 출력하며, find 명령어의 에러는 /dev/null 파일로 보내진다

```
# find / -name "*.c" > foundit 2> /dev/null
```

- 🌐 루트 디렉토리에서 .c 로 끝나는 파일 이름들을 검색하여 foundit 파일로 출력하며 발생하는 에러 메시지는 표준출력 (파일식별자 1)과 같은 곳(foundit)으로 보낸다

```
# find / -name "*.c" > foundit 2>&1
```

Standard I/O Redirection

🌐 Redirection 사용예제

🌐 명령어 예제

- 🌐 Command 수행 결과 에러메시지를 표준출력(화면) 으로 보내고 나머지 수행 결과 result.txt 파일로 출력

```
# ls /etc/passwd /etc/password 2>&1 > result.txt
```

- 🌐 순서를 바꾸면 동작하지 않는다. 에러 메시지 및 수행 결과 모두 result.txt 파일로 출력됨

```
# ls /etc/passwd /etc/password > result.txt 2>&1
```

Standard I/O Redirection

● 파이프(pipe)

- 한 프로그램의 출력을 중간 파일 없이 다른 파일의 입력으로 바로 보내는 유닉스 매커니즘
 - 파이프는 파이프(|) 기호 왼쪽 명령어의 출력을 오른쪽 명령어의 입력으로 보낸다
 - 파이프라인은 하나 이상의 파이프로 구성된다

● 명령어 예제

- /etc/profile 의 내용을 한페이지 단위로 출력

```
# cat /etc/profile | more
```

- /etc/passwd 파일의 내용을 역으로 정렬하면서 한 페이지 단위로 출력

```
# cat /etc/passwd | sort -r | more
```

Standard I/O Redirection

🔵 파이프(pipe)

🔵 Pipe 와 Redirection 사용예제

- 🔵 파일 목록을 역으로 정렬하며 목록 중 "S"를 포함하는 라인을 추출하여 result.txt에 출력

그 결과를

```
# ls /etc/rc5.d | sort -r | grep S > result.txt
```

- 🔵 출력이 result.txt 로 향해 있기 때문에 grep 명령어의 입력으로 전달되지 않는다(X)

```
# ls /etc/rc5.d | sort -r > result.txt | grep S
```

Standard I/O Redirection

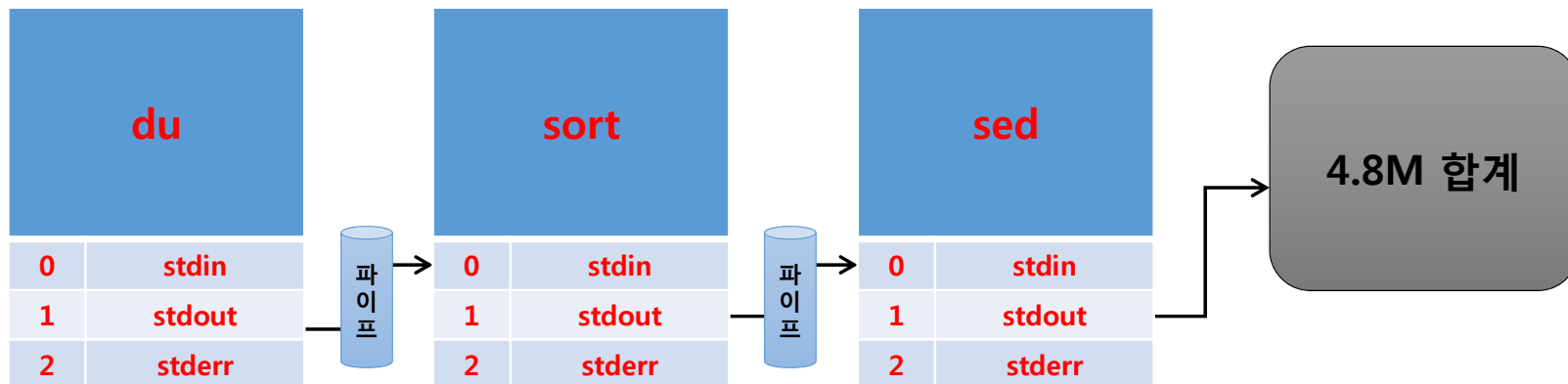
🔗 파이프(pipe)

🔗 명령어 예제

- 🔗 시스템에 로그인한 사용자수 확인

```
# who | wc -l  
3
```

```
# du -ch | sort -n | sed -n '$p'  
4.8M     합계
```



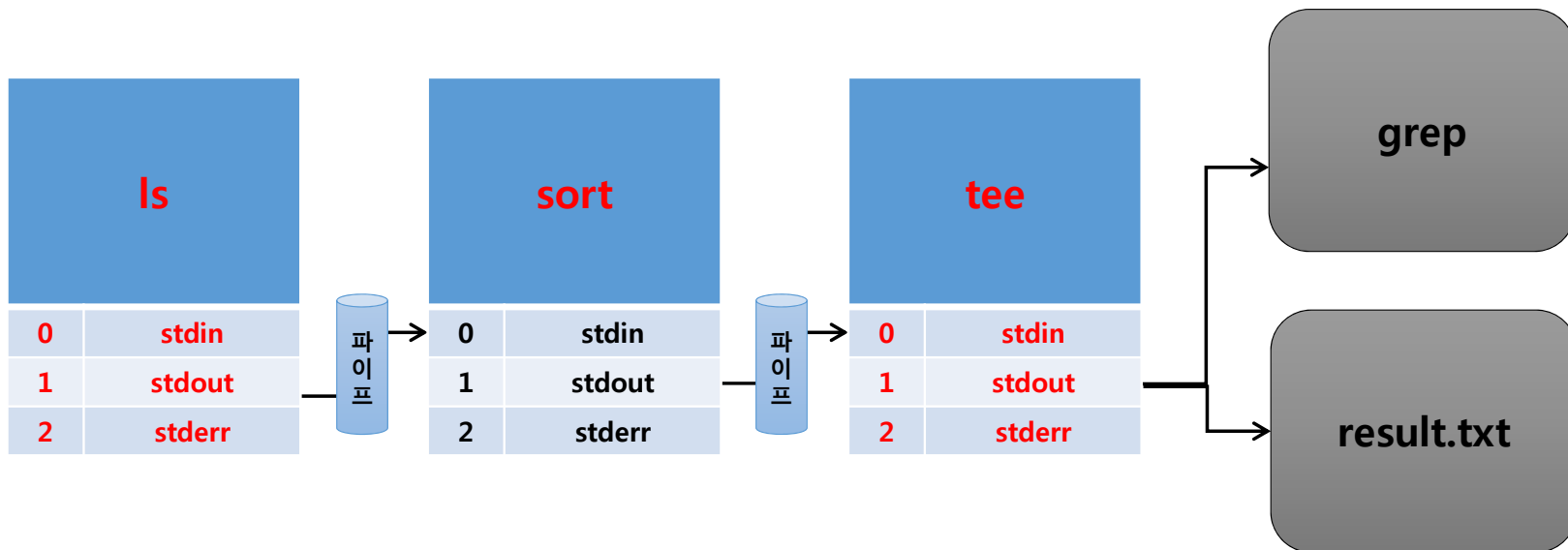
Standard I/O Redirection

파이프(pipe)

Pipe 와 Redirection 사용예제

- Tee 명령어는 표준 입력으로부터 읽은 결과를 표준 출력 및 파일에 쓰기를 하므로 grep 까지 결과값이 도달한다

```
# ls /etc/rc5.d | sort -r | tee result.txt | grep S
```



Standard I/O Redirection

- 이름있는 파이프(FIFO : First In First Out)
- 서로 연관되지 않은 두 프로세스 사이에서 데이터를 주고받을 수 있는것이 이름있는 파이프라고 한다
- 명령행에서 파이프 기호(|)를 사용할 때는 파이프를 통해서 데이터를 주고받는 두 프로세스가 같은 셸에서 시작된 것이어야 한다

FIFO 예제

Process 1

```
# mkfifo /down/fifo
# cat < /down/fifo

2009. 05. 22. (금) 22:21:45 KST
2009. 05. 22. (금) 22:21:48 KST
2009. 05. 22. (금) 22:21:51 KST
2009. 05. 22. (금) 22:21:54 KST
2009. 05. 22. (금) 22:21:57 KST
2009. 05. 22. (금) 22:22:00 KST
```

Process 2

```
# cat > /down/fifo.test
while sleep 3
do date
done
Ctrl +d

# ./fifo.test > /down/fifo
```


String Process with Regular Expressions

Regular Expression

🌐 What is Regular Expressions?

🌐 어떤 문자열의 집합을 묘사하는데 사용되는 텍스트 스트링

🌐 정해진 구문 규칙에 따른다

🌐 Editor, Utility, Programming 언어에서 텍스트 패턴을 기준으로 검색 혹은 조작하는데 사용된다

Regular Expression

Metacharacters

문자	의미
^	줄의 시작을 나타내지만 문맥에 따라서는 정규 표현식에서는 문자 집합의 의미를 반대로 해석해 줌
*	바로 앞의 문자열이나 정규 표현식에서 0 개 이상 반복되는 문자를 나타냄

Regular Expression

Metacharacters

문자	의미
.	새로운 라인을 제외한 오직 한 개의 글자와 일치
\$	줄의 끝과 일치하며, ^\$ 의 경우 빈 줄과 일치함
[..]	문자들을 집합으로 묶어줌
\w	특수 문자를 원래의 문자 의미 그대로 해석 하도록 해줌
\w(.....\w)	다음 사용을 위해 태그를 붙인다. 최대 9개 까지 사용 가능하다.

Regular Expression

Metacharacters

문자	의미
$\backslash <$	단어의 시작 지시자
$\backslash >$	단어의 끝 지시자
$X\{n\}$	문자 X를 n번 반복
$X\{n,\}$	문자 X를 적어도 n번
$X\{m,n\}$	문자 X를 m번에서 n번 반복

Regular Expression

🌐 사용예제

🌐 "1133*"

🌐 3이 0개 이상 반복됨을 의미한다

EX) 113, 1133, 11333, 113

🌐 "13."

🌐 빈칸을 포함한 최소 한 글자를 나타낸다(space,tab 등 포함)

EX) 1133, 11333, 13 , 13(X)

🌐 "[xyz]"

x, y, z 중에 한 글자와 일치

🌐 "[c-n]"

c 에서 n 사이에 들어 있는 한 문자와 일치

Regular Expression

🔵 사용예제

🔵 "[B-Pk-y]"

- 🔵 B에서 P 까지 사이 혹은 k 에서 y 까지 사이 중의 한 글자와 일치

🔵 "[a-z0-9]"

- 🔵 소문자 혹은 숫자와 일치

🔵 "[^b-d]"

- 🔵 b 에서 d 사이의 문자를 제외한 모든 문자를 나타냄

🔵 "₩\$"

- 🔵 정규 표현식에서 줄 끝(end-of-line) 을 나타내는 의미 대신 문자 그대로 해석하게 해줌

Regular Expression

- 모두 []로 묶어서 사용한다.(예 [[[:alnum:]]])
- **[[:alnum:]]** 는 알파벳이나 숫자와 일치하고 **[A-Za-z0-9]** 와 같은 표현입니다.
- **[[:alpha:]]** 는 알파벳과 일치하고 **[A-Za-z]** 와 같은 표현입니다.
- **[[:blank:]]** 는 빈 칸이나 탭과 일치합니다.
- **[[:cntrl:]]** 는 제어 문자들과 일치합니다.
- **[[:digit:]]** 는 10진 숫자들과 일치하고 **[0-9]** 와 같은 표현입니다.
- **[[:graph:]]** (출력가능한 그래픽 문자들). 아스키 33 - 126 의 문자들과 일치합니다.
빈 칸 문자가 포함되지 않는다는 것만 제외하고는 밑에서 설명할 **[[:print:]]** 와 같습니다.
- **[[:lower:]]** 는 알파벳 소문자와 일치하고 **[a-z]** 와 같은 표현입니다.
- **[[:print:]]** (출력 가능한 문자들). 아스키 32 - 126 까지의 문자들과 일치합니다. 위에서 설명한 **[[:graph:]]** 와 같지만 빈 칸 문자가 포함되어 있습니다.
- **[[:space:]]** 는 공백문자들과 일치합니다(빈 칸, 수평탭).
- **[[:upper:]]** 는 알파벳 대문자와 일치하고 **[A-Z]** 와 같은 표현입니다.
- **[[:xdigit:]]** 는 16진수 숫자와 일치하고 **[0-9A-Fa-f]** 와 같은 표현입니다.

Grep

💡 What is grep?

- 💡 파일 전체를 뒤져 정규 표현식에 대응하는 모든 행들을 출력
- 💡 기본형식

```
# grep <word> <filename>
```

Grep

Grep option

문자	의미
-b	검색결과의 각 행 앞에 검색된 위치의 블록 번호 표시. 검색 내용이 디스크의 어디쯤 있는지 위치를 알아내는데 유용
-c	검색 결과를 출력하는 대신, 찾아낸 행의 총수를 출력
-h	파일 이름을 출력하지 않음
-i	대소문자를 구분하지 않음 (대문자와 소문자 동일취급)
-l	패턴이 존재하는 파일의 이름만 출력

Grep

Grep option

문자	의미
-n	파일 내에서 행 번호를 함께 출력
-s	에러 메시지 외에는 출력하지 않음. 종료상태를 검사할 때 유용하게 사용
-v	패턴이 존재하지 않는 행만 출력
-w	패턴 표현식을 하나의 단어로 취급하여 검색.

Grep

💡 grep 사용예제

- 💡 /etc/passwd 파일 에서 root 라는 패턴을 찾음

```
# grep root /etc/passwd
```

- 💡 Ps 명령어의 출력 결과에서 root 를 포함하는 모든 행들이 화면에 출력됨

```
# ps -ef | grep root
```

- 💡 root 로 시작하는 행과 행 번호(n) 를 출력

```
# grep -n '^root' /etc/passwd
```

Grep

💡 grep 사용예제

- 💡 .bak 로 끝나는 행을 출력. 작은 따옴표는 달러 기호가 해석되는 것을 막는다

```
# grep '.bak$' FILE
```

- 💡 대문자로 시작하고 숫자로 끝나는 다섯 문자의 열이 포함된 행을 출력

```
# grep '[A-Z]...[0-9]' FILE
```

- 💡 User1 으로 시작하는 행을 검색하며, 문자열 앞에 0개 혹은 임의 개수의 공백이 와도 무관하다

```
# ps -ef | grep "^ *user1"
```

Grep

🔗 grep with option

- 🔗 -n 옵션은 패턴이 찾아진 행 번호를 함께 출력

```
# grep -n '^south' datafile
```

- 🔗 -v 옵션은 해당 패턴이 포함되지 않은 모든 행을 출력하며, 파일에서 특정 내용의 이력을 삭제하는데 쓰이기도 한다

```
# grep -v 'root' datafile > temp  
# mv temp datafile
```

Grep

• grep with Regular Expression

- 소문자가 적어도 아홉 개가 연속적으로 나오는 문자열을 포함한 모든 행을 출력

```
# grep '[a-z]\{9\}' datafile
```

- 소문자 하나로 시작하고, 이어서 임의 개수의 여러 문자가 나오며, n으로 끝나는 단어가 포함된 모든 행을 출력

(.*) 기호는 공백을 포함한 임의의 문자들을 의미

```
# grep '\<[a-z].*h\>' datafile
```

Grep

🔗 grep with option

- 🔗 -n 옵션은 패턴이 찾아진 행 번호를 함께 출력

```
# grep -n '^south' datafile
```

- 🔗 -v 옵션은 해당 패턴이 포함되지 않은 모든 행을 출력하며, 파일에서 특정 내용의 이력을 삭제하는데 쓰이기도 한다

```
# grep -v 'root' datafile > temp  
# mv temp datafile
```


Grep

🔗 grep with option

- 🔗 -i 옵션은 대소문자를 구별하지 않음. 그러므로 문자열 pat가 어떠한 대소문자 조합으로 이루어져 있더라도 무관

```
# grep -i 'pat' datafile
```

- 🔗 -l 옵션은 패턴이 찾아진 파일의 행 번호 대신, 단지 파일 이름만 출력

```
# grep -l 'SE' *  
datafile  
databook
```

Grep

💡 grep with option

- 💡 -c 옵션은 패턴이 찾아진 행의 총 수를 출력. 패턴이 발견된 횟수가 아니라 '행의 수'가 출력

```
# grep -c 'west' datafile
3
```

- 💡 -w 옵션은 패턴이 다른 단어의 일부가 아닌 하나의 단어가 되는 경우만 찾음

```
# grep -w 'north' datafile
north      NO      Margot Weber      4.5      5
```

실습

📍 실습

📍 핸드폰(집전화) 번호 패턴을 정규표현을 통해 정의

- 📍 지역번호 및 통신사 번호는 '0'으로 시작하는 2 혹은 3자리 숫자 (02,032,010)
- 📍 중간자리 번호 패턴은 3자리 혹은 4자리가 반드시 한 번은 나와야 한다
- 📍 끝자리 번호 패턴은 4자리가 반드시 한 번은 나와야 한다

Egrep

💡 What is egrep?

💡 Grep에서 제공하지 않는 확장된 정규표현식 메타문자를 사용할 수 있음

```
# egrep <word> <filename>
```

Egrep

🔵Egrep이 지원하는 확장 Metacharacters

문자	의미
+	선행문자와 같은 문자의 1개 혹은 임의 개수와 대응
?	선행문자와 같은 문자의 0개 혹은 1개와 대응
a b	a 혹은 b와 대응
()	정규 표현식을 묶어줌

Egrep

💡 egrep 사용예제

- 💡 정규표현식 NW나 EA가 포함된 행을 출력

```
# egrep 'NW|EA' datafile
```

- 💡 숫자 3이 한 번 이상 등장하는 행을 출력

```
# egrep '3+' datafile
```

- 💡 숫자 2 다음에 마침표가 없거나 한 번 나오고, 다시 숫자 하나가 오는 행을 출력

```
# egrep '2\.[0-9]' datafile
```

Egrep

🔗 egrep 사용예제

- 🔗 패턴 no가 한 번 이상 연속해서 나오는 행을 출력한다.

```
# egrep '(no)+' datafile
```

- 🔗 문자 S 다음에 h나 u가 나오는 행을 출력

```
# egrep 'S[h|u]' datafile
```

- 🔗 패턴 Sh 나 u를 포함한 행을 출력

```
# egrep 'Sh|u' datafile
```

Fgrep

What is fgrep?

- Fgrep은 grep 명령어와 동일하게 동작. 하지만 정규표현식 메타 문자들을 특별히 취급하지 않음

```
# fgrep '문자열' <file name>
```

Ex)

```
# fgrep '[A-Z]...[0-9]' <file name>
```

- [A-Z]...[0-9] 를 정규표현식 메타문자로 특별히 취급하지 않고 문자열 자체로 인식
- [A-Z]...[0-9] 자체를 포함하는 문자열을 찾는다.

Awk

What is Awk?

- 자료 처리 및 리포트 생성에 사용하는 프로그래밍 언어
- 입력 데이터로는 표준 입력이나 여러 개의 파일 또는 다른 프로세스의 결과를 사용할 수 있다.
- 사용자가 지정한 패턴 검색이나 특별한 작업을 수행하기 위해 파일을 행 단위로 조사

기본형식

```
# awk 'pattern' filename  
# awk '{action}' filename  
# awk 'pattern {action}' filename
```

Awk

🌐 awk 사용예제

- 🌐 파일의 첫 번째 필드를 출력한다. 첫 번째 필드는 각 행의 맨 왼쪽 경계에서 시작하고 공백 문자로 구분되는 영역을 의미한다

```
# awk '{print $1}' datafile
```

- 🌐 파일에서 root 를 포함하는 행들의 첫 번째와 두 번째 필드를 출력한다

```
# awk '/root/{print $1, $2}' datafile
```

Awk

🧠 awk 사용예제

- 🧠 df 명령어를 통해 출력되는 내용 중 두 번째 필드가 80000 보다 큰 행이 출력된다

```
# df | awk ' $2 > 80000 '
```

- 🧠 date 명령을 통해 출력되는 두 번째 필드(month) 와 여섯 번째 필드(year)를 print 함수를 이용하여 출력

```
# date | awk '{print "Month : " $2 "\nYear: " $6}'
```

Awk

🔗 awk 사용예제

- 🔗 NR 변수(하나의 레코드를 처리한 후 1이 증가하는 변수) 를 사용하여 레코드의 번호와 함께 파일의 내용대로 출력

```
# awk '{print NR, $1, $3}' datafile
```

- 🔗 NF 변수(필드의 개수를 출력라는 변수) 를 사용하여 레코드의 번호와 함께 각 행의 필드의 개수도 함께 출력

```
# awk '{print $1, $3, NF}' datafile
```

Awk

🎧 awk 사용예제

- 🎧 ":"구분자를 기준으로 필드를 나누며 root 를 포함하는 행을 출력

```
# awk -F : ' /root/{print $0}' /etc/passwd
```

- 🎧 n 이나 s 로 시작하는 행의 첫 번째 필드를 출력

```
# awk -F : ' /^[ns]/{print $1}' datafile
```

Awk

🌐 awk 사용예제

- 🌐 TAB 으로 필드를 구분하여 나누며 첫 번째, 두 번째, 세 번째 필드를 출력한다. 작은 따옴표로 묶은 이유는 쉼이 대괄호를 해석해 버리는 것을 방지하기 위해서 이다

```
# awk -F '[\t]'{print $1, $2, $3}' employees
```

- 🌐 Match 연산자(~, tilde)는 특정 레코드나 필드 내에서 일치하는 정규표현식 패턴이 존재 하는지 검사하는 데 쓰인다

```
# awk '$1 ~ /[bB]ill/' employees
```

Awk

• awk 사용예제

- 첫 번째 필드가 ly 로 끝나지 않는 행들을 출력한다

```
# awk ' $1 !~ /ly$/ ' empl oyees
```

- 네 번째 필드가 Chin 으로 끝나면 문자열 The price is \$와 여덟 번째 필드(\$8) 및 마침표를 함께 출력한다

```
# awk ' $4 ~ /Chi n$/{pri nt "The pri ce i s $" $8 " . "}' FILE
```

실습

💡 실습

- 💡 /etc/passwd 파일에서 bash 셸을 사용하는 유저의 계정명과 홈 디렉토리를 다음과 같은 형식으로 출력하시오

EX) USER : [계정명] HOME : [홈 디렉토리]

CUT

🌐 cut은 각 입력파일의 각 라인의 선택된 부분을 표준출력 한다. 또는 주어진 파일이 없거나 파일이름이 없으면 표준 입력한다.

🌐 옵션

- b 오직 설정된 바이트를 출력한다.
- c 오직 설정된 char.를 출력한다.
- d 필드 구별자나 탭 대신에 DELIM을 사용한다.
- f 오직 선택된 필드만 출력한다.
- n -b와 함께사용. 멀티바이트 문자를 분리하지 말고 출력.
- S 구별자가 없는 라인을 출력하지 말라.
- b,-c,-f 중 오직 한가지만 사용.

각 리스트는 범위, 콤마에 의한 많은 범위 분리할 수 있다.

- N N번째 바이트만 출력
- N- N번째 바이트부터 출력
- N-M N에서 M까지 출력
- M 처음부터 M까지 출력

User Account Management

User Account Management

🌐 사용자 계정/그룹 관리

🌐 계정생성(useradd) 명령어 관련 파일

위 치	의 미
/etc/passwd	사용자 계정 정보의 저장 장소
/etc/shadow	안전하게 보호될 사용자 계정 정보가 들어있음
/etc/group	계정이 속한 그룹 ID 및 그룹 목록들에 대한 정보
/etc/gshadow	그룹의 암호 정보 저장 장소
/home	사용자별 개별 홈 디렉토리가 생성되는 기본 디렉토리
/etc/login.defs /etc/default/useradd /etc/skel	계정 생성시 참조하는 기본설정 정보

User Account Management

🌐 /etc/passwd

```
Chris : x : 500 : 500 : : /home/Chris : /bin/bash
```

필드	의미
Chris	사용자 계정명
x	사용자에게 부여된 패스워드
500	숫자로 표현되는 사용자 id(UID)
500	이 사용자에게 대한 숫자로 표현되는 주 그룹 id(GID)
NULL	사용자에게 부여되는 임의의 정보(Comment)
/home/test	사용자계정에 개별 홈디렉토리 위치
/bin/bash	사용자가 시스템에 로그인시 사용하는 Shell의 위치

User Account Management

🌐 /etc/shadow

```
Chi rs : !! : 14349 : 0 : 99999 : 7 : : :
```

필드	의미
Chris	사용자 계정명
!!	사용자에게 부여된 기본암호(인증불가)
14349	test 계정에 부여된 암호 생성일자
0	test 계정의 암호를 변경할 수 있는 최소 기간
99999	test 계정에 부여된 암호를 변경없이 사용할 수 있는 유효기간
7	만료일 지정시 만료 경고일수
Null	계정 만료일자와 비활성화 일수
Null	계정의 만료일 (기본값은 지정되어 있지 않다)

User Account Management

🌐 /etc/group

```
test : x : 500 :
```

필드	의미
test	그룹의 이름
x	그룹 패스워드
500	숫자로 된 그룹id
Null	그룹 구성원 모두의 사용자 이름 ,(кома)로 구분된다

User Account Management

🌐 /etc/gshadow

Chris : ! : :

필드	의미
test	그룹의 이름
!	그룹에 부여된 기본암호(인증불가)
500	숫자로 된 그룹id
Null	그룹의 소유주
Null	그룹 구성원 모두의 사용자 이름 ,(콤마)로 구분된다

User Account Management

- /etc/login.defs(shadow password suite configuration)

- 해당 계정의 패스워드 유효기간
- 계정 생성시 자동으로 할당되는 UID/GID 범위

- /etc/default/useradd(Default values for account creation)

- /etc/passwd 파일에 생성되는 대부분의 기본 값
- SKEL 디렉토리 경로

- /etc/skel(Directory containing default files)

- 홈 디렉토리가 생성 되면서 복사될 기본 파일들이 들어있음
- 만약 계정이 생성된 후 생성되는 계정의 사용자들에게 공지할 내용이 있다면 /etc/skel 디렉토리에 내용을 포함해 둘 수 있음

User Account Management

🌐 USERADD 계정 생성 명령

```
# useradd [Option] [Argument] ... <Account_Name>
```

```
Chris : x : 500 : 500 : : /home/Chris : /bin/bash
```

필드별 옵션	의 미
Null	사용자 계정명은 직접 지정
-p	사용자의 패스워드 생성(MD5_Crypt)
-u	사용자의 uid 정보를 임의로 변경
-g	사용자의 기본그룹 지정시 사용
-c	사용자에게 설명 부여
-d	사용자의 기본 홈 디렉토리 변경
-s	사용자의 로그인 셸 변경
-r	사용자의 uid 정보를 1~499번 사이의 값으로 생성
-o (non-unique)	동일한 uid값을 갖는 계정생성

User Account Management

🌐 USERADD 추가 기능

- 🌐 계정 생성과 관련된 기본 정보를 변경할 수 있다

```
# useradd -D [Option] [Argument]
```

옵 션	의 미
-b	계정생성시 생성되는 홈디렉토리의 기본생성 위치 변경
-e	계정의 기본 만료일자 변경
-f	계정 만료일자에 대한 Inactive Days 변경
-g	계정생성시 지정되는 기본그룹 변경
-s	계정의 기본 로그인 쉘 변경

User Account Management

🌐USERMOD 계정 수정 명령

```
# usermod [Option] [Argument] ... <Account_Name>
```

```
Chris : x : 500 : 500 : : /home/Chris : /bin/bash
```

필드별 옵션	의 미
-l	사용자의 계정명 변경
-p	사용자의 패스워드 변경(MD5_Crypt)
-u	사용자의 uid 정보를 임의로 변경
-g	사용자의 기본그룹 변경시 사용
-G	사용자에게 추가그룹 지정 (-a 옵션과 함께사용)
-c	사용자에게 설명 부여
-d	사용자의 기본 홈 디렉토리 변경(-m 옵션과 함께사용)
-s	사용자의 로그인 쉘 변경
-o (non-unique)	동일한 uid값을 갖는 계정생성

User Account Management

🌐 USERDEL 계정 삭제 명령

🌐 계정 삭제

```
# userdel [Option] <Account_Name>
```

옵 션	의 미
-f	로그인 되어있거나, 삭제하려는 계정의 그룹을 다른계정의 기본그룹으로 사용하고 있을 때 강제로 삭제할수 있다
-r	계정생성시 함께 만들어진 모든 정보를 삭제
-h	도움말 표시

User Account Management

🌐 PASSWD 암호 생성

🌐 암호생성

```
# passwd [Option] <Account_Name>
```

옵 션	의 미
-d	사용자의 패스워드를 Null 값으로 지정(암호삭제)
-l	사용자의 패스워드를 잠금설정>Password Lock
-u	사용자의 패스워드를 잠금해제>Password Unlock
-S	사용자의 패스워드 상태출력
--stdin	사용자의 패스워드를 표준출력장치로 입력받아 생성

User Account Management

🌐 GROUPADD 그룹생성 명령

```
# groupadd [Option] [Argument] <Group_Name>
```

옵 션	의 미
-f	이미 존재하는 그룹과 동일한 그룹을 강제 생성
-g	그룹생성시 GID 값 지정
-r	그룹의 gid 정보를 1~499번 사이의 값으로 생성

User Account Management

🌐 GROUPMOD 그룹수정 명령

```
# groupmod [Option] [Argument] <Group_Name>
```

옵 션	의 미
-g	기존그룹의 GID 값 지정
-n	그룹의 이름 변경

🌐 GROUPDEL 그룹삭제 명령

```
# groupdel [Group_Name]
```

Ownership / Permissions

Permissions

Ownership/Permission

관련된 구성요소

-  파일 유형

-  파일 허가권

-  링크 수

-  파일 소유자 이름

-  파일 소유 그룹 이름

Permissions

Ownership/Permission

관련된 구성요소

```
drwxr-xr-x    2 root root 4096 3월 11 18:23 Desktop
-rw-----    1 root root 1655 3월 11 06:02 anaconda-ks.cfg
-rw-r--r--    1 root root 46643 3월 11 06:01 install.log
-rw-r--r--    1 root root 6863 3월 11 05:59 install.log.syslog
-rw-r--r--    1 root root 74625 3월 17 23:17 iptables.txt
-rw-r--r--    1 root root 199 3월 11 18:06 scsrn.log
```

Owner/Group/Others

Owner			Group			Other		
r	w	x	r	w	x	r	w	x
4	2	1	4	2	1	4	2	1
7			7			7		

Permissions

Ownership/Permission

자주 사용되는 명령어 예제(Permission)

- sales 파일의 사용자 권한은 rw, 그룹의 권한은 r, 그 외 사용자의 권한은 r 로 설정 (numeric method)

```
# chmod 644 sales
```

- sales 파일의 그룹에 읽기와 실행권한을 허가(symbolic method)

```
# chmod g+rx sales
```

- sales 파일의 그 외 사용자에게 읽기/쓰기/실행 권한 허가

```
# chmod o+rw sales
```

Permissions

Ownership/Permission

파일에서의 r 권한

- cat , more , vim , cp 등과 같은 파일의 내용을 읽어 들이는 것과 관련이 있다.

파일에서의 w 권한

- cat , echo , vim 등과 같은 파일의 내용을 수정,변경 하는 것과 관련이 있다.
- vim 같은 경우는 :wq!로 저장이 가능하긴 하다. (소유주만 적용)
- 파일 권한에는 삭제 권한이 없다.

파일에서의 x 권한

- 실행 파일 같은 경우 실행 여부와 관련이 있다.

Permissions

Ownership/Permission

디렉토리 에서의 r 권한

- ls , dir등과 같은 디렉토리의 내부 내용을 읽어 들이는 것과 관련이 있다.

디렉토리 에서의 w 권한

- mkdir , touch , rm , mv등과 같은 디렉토리 내부에 생성,삭제와 관련이 있다

디렉토리 에서의 x 권한

- cd 로 디렉토리의 접근을 할 수 있다.
- 디렉토리에 x가 없다는것은 대부분의 명령어를 이용 할 수가 없게 된다.

Permissions

Ownership/Permission

자주 사용되는 명령어 예제(Changing Ownership)

- sales 파일의 소유자를 chris 으로 바꿈

```
# chown chri s sal es
```

- sales 파일의 소유자와 그룹을 chris 으로 바꿈

```
# chown chri s:chri s sal es
```

- sales 파일의 소유자는 chris 그룹은 root 로 바꿈

```
# chown chri s:root sal es
```

Group Ownership

자주 사용되는 명령어 예제(Changing Group Ownership)

- sales 파일의 소유그룹 을 chris 으로 바꿈

```
# chgrp chri s sal es
```

Permissions

Ownership/Permission

SetUID

- S나 s 로 표현되고 8진수 모드로는 4000 으로 표현됨
- 사용자가 SetUID 권한이 설정되어 있는 실행파일을 실행 할 경우 그 파일의 소유자 권한을 가지게 된다

```
# chmod 4XXX <파일>
```

find 로 setuid 찾기

```
[root@sec ~]# find / -perm +4000 -user root 2>/dev/null | xargs ls -l
-rwsr-xr-x. 1 root root 74764 Feb 22 14:00 /bin/mount
-rwsr-xr-x. 1 root root 36892 Jul 19 2011 /bin/ping
-rwsr-xr-x. 1 root root 32016 Jul 19 2011 /bin/ping6
-rwsr-xr-x. 1 root root 30092 Jun 22 2012 /bin/su
-rwsr-xr-x. 1 root root 49796 Feb 22 14:00 /bin/umount
```

Permissions

Ownership/Permission

SetGID

- S 나 s로 표현되고 8진수 모드로는 2000 으로 표현된다
- SetUID와 동일한 내용을 가지며 다만 그룹의 권한으로 실행 된다는 것만이 다름

```
# chmod 2XXX <파일>
```

find setgid 설정 파일 찾기

```
[root@sec ~]# find / -perm +2000 -user root 2>/dev/null | xargs ls -l
-rwxr-sr-x. 1 root root      4108 Feb 22 20:18 /sbin/netreport
-r-xr-sr-x. 1 root tty       8524 Jul 19  2011 /usr/bin/wall
-rwxr-sr-x. 1 root tty     10124 Feb 22 14:00 /usr/bin/write
-rwx--s--x. 1 root utmp      4772 Aug 19  2010 /usr/libexec/utempter/utempter
-rwxr-sr-x. 1 root postdrop 174856 Dec  3  2011 /usr/sbin/postdrop
-rwxr-sr-x. 1 root postdrop 207708 Dec  3  2011 /usr/sbin/postqueue
```


Permissions

Ownership/Permission

Sticky bit

- T 나 t로 표현되며 8진수 모드로는 1000 으로 표현된다
- 해당 권한이 설정되어 있는 디렉토리의 모든 유저는 파일을 생성하고 삭제하는 것이 자유롭지만, 삭제시에는 소유권자 혹은 슈퍼유저만 지울 수 있다

```
# chmod 1XXX<파일>
```

find로 찾기

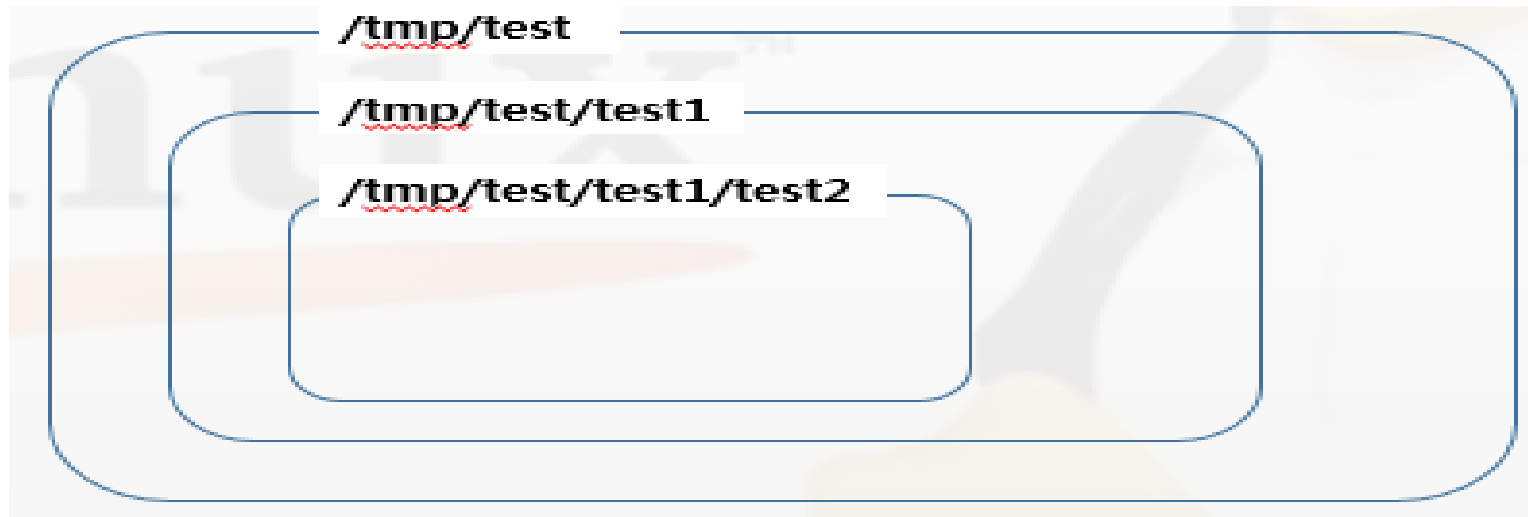
```
[root@sec ~]# find / -perm +1000 -user root 2>/dev/null | xargs ls -dl
drwxrwxrwt. 2 root root 40 Jun 5 2013 /dev/shm
drwxrwxrwt. 3 root root 4096 Jun 4 19:14 /tmp
drwxrwxrwt. 2 root root 4096 Jun 4 19:14 /tmp/.ICE-unix
drwxrwxrwt. 2 root root 4096 Jun 4 09:36 /var/tmp
```

실습

Ownership/Permission

실습

- 옵션을 찾아서 아래 그림의 모양대로 한 번의 명령어로 디렉토리를 생성한다
- /tmp/test/ 디렉토리는 Sticky bit 를 가지며 모든 사용자가 모든 권한을 가진다
- /tmp/test/test1/ 디렉토리는 소유자는 모든 권한을, 그룹은 읽기 및 실행 권한을, 다른 유저는 실행 권한을 가진다
- /tmp/test/test1/test2/ 디렉토리는 소유자는 모든 권한을, 그룹은 읽기 권한을, 다른 유저는 아무런 권한도 가지지 않는다



Permissions

Ownership/Permission

chattr 명령어

- 리눅스의 second extended 파일 시스템에서 제공되는 파일 속성 변경 명령어
- 사용 형식

```
# chattr [option] [+ -=] [속성값] <File / Directory>
```

옵션

- a(append) : 추가 모드만 지원함(소유주만 가능)
- c(compressed) : 압축상태로 저장함
- d(no dump) : 덤프를 하지 않음
- i (immutable) : 변경 불가능 상태로 설정
- j(journaling) : 저널링으로 기록하게 함(ext3만 가능)
- s(secure deletion) : 안전한 쓰기를 제공
- u(undelete) : 삭제한 데이터를 debugfs 명령어로 복구 함
- e : ext3를 ext4로 이주 가능

chattr / lsattr

```
[root@sec data]# lsattr AnnualReport
-----e- AnnualReport
[root@sec data]#
[root@sec data]# chattr +a AnnualReport
[root@sec data]#
[root@sec data]# echo "Adding Data" >> AnnualReport
[root@sec data]#
[root@sec data]# lsattr AnnualReport
-----a-----e- AnnualReport
[root@sec data]#
[root@sec data]# cat AnnualReport
1/1 ; Total:1500$
1/2 ; Total:2000$
1/3 ; Total:1300$
1/4 ; Total:3000$
Adding Data
```

chattr / lsattr

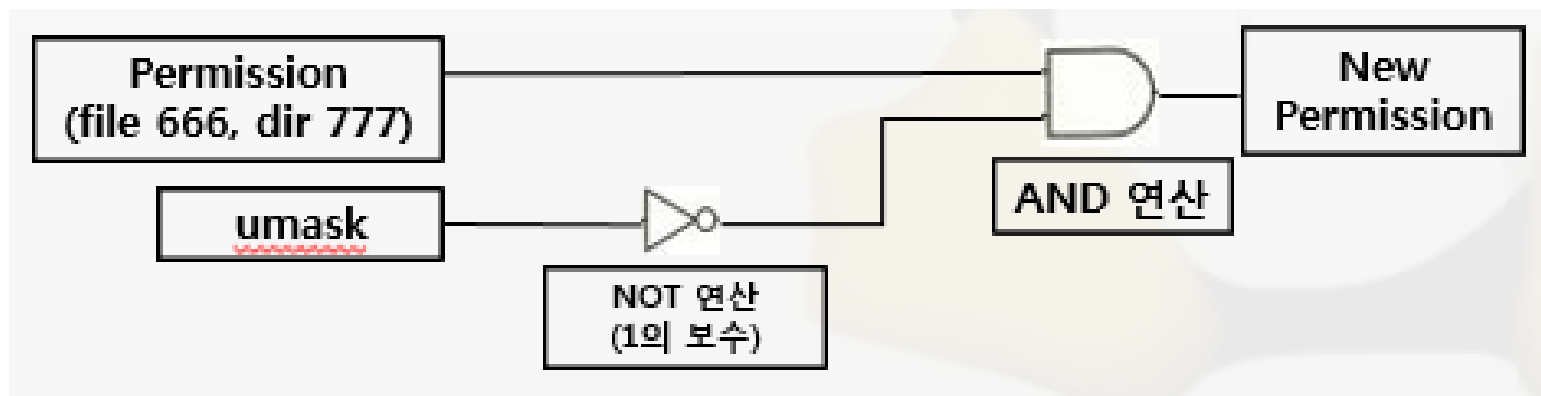
```
[root@sec data]# chattr +i Å
chattr: No such file or directory while trying to stat Å
[root@sec data]#
[root@sec data]# chattr +i AnnualReport
[root@sec data]# lsattr AnnualReport
----ia-----e- AnnualReport
[root@sec data]# echo "Adding Data2" >> AnnualReport
-bash: AnnualReport: Permission denied
```

Permissions

● UMASK



# umask [mode]		
mode	File	Directory
0000	666	777
0001	666	776
0002	664	775
0022	644	755

● UMASK 구동방식 (계산법)



Permissions

ACL (Access Control List)

-  ACL은 특별한 권한 설정기능을 제공한다. 우리는 기존에 파일이나 디렉토리에 부여하던 소유자, 그룹, 기타 사용자의 권한을 특정 사용자, 특정 그룹에게 추가해서 설정할 수 있다.
-  ACL은 그룹에게는 읽기 권한만 주고, 특정 유저에게는 읽기/쓰기 권한을 줄 수도 있다.

Permissions

SetfacI

```
# setfacI [opti on] [u, g, o: ---] [Fi le / Di rectory]
```

옵 션	의 미
-m	Modify의 약자로 권한을 지정하거나 수정할때 사용
-x	권한을 삭제할 때 사용
-R	디렉토리의경우 하위 디렉토리와 파일까지 권한을 변경
-b	권한 및 mask 등 지정한 권한을 전부 제거
-d	디렉토리는 자신의 디렉토리내에서 생성되는 파일에 자동으로 적용되는 acl

GetfacI

```
# getfacI [opti on] [Fi le / Di rectory]
```

옵 션	의 미
-R	디렉토리 이면 하위까지 전부권한을 보여준다

setfacl / getfacl

```
[root@sec data]# getfacl GrandTotal
# file: GrandTotal
# owner: root
# group: root
user::rw-
group::r--
other::r--
```

```
[root@sec data]# setfacl -m user:sa:rw ./GrandTotal
```

```
[root@sec data]# getfacl GrandTotal
# file: GrandTotal
# owner: root
# group: root
user::rw-
user:sa:rw-
group::r--
mask::rw-
other::r--
```

Investigating and Managing Processes

Process Management

What is Process?

- 컴퓨터의 CPU에서 실행되는 모든 프로그램을 프로세스라고 한다

Process의 종류

- 프로세스에는 대화형 프로세스, 배치 프로세스, 데몬 프로세스로 구분되어 진다.

각 Process의 기본규칙

- 각각의 프로세스마다 고유 번호의 프로세스 ID(PID)를 하나씩 증가시키면서 부여한다.
 - 더이상 할당할 PID가 없으면, 사용되지 않는 가장 낮은 숫자로 되돌아가 다시 할당한다
- 프로세스는 파일의 소유권과 유사한 방식의 소유권을 갖는다
 - 프로세스를 실행하는 사용자의 UID가 프로세스의 실제 사용자 UID로 할당된다
 - SetUID와 같은 특수한 경우 실행하는 사용자의 UID가 아닌 파일의 소유자가 실행한것처럼 실행된다(EUID/EGID)

Process Management

● 대화형 프로세스

- 터미널과 세션을 통하여 사용자와 정보를 주고받으며 실행하는 프로세스를 말한다
 - 프로그램과 정보를 주고받는 동안 프로세스는 포그라운드로 실행되며 터미널과의 입출력 교류도 이루어진다
- 포그라운드에서 실행되고 있는 프로세스는 그 프로세스가 종료되거나 일시 중지될 때까지 터미널에 대한 완벽한 제어권을 가지며, 포그라운드 프로세스가 일시 중지된 후 백그라운드 프로세스로 전환되면, 터미널의 제어권은 백그라운드 프로세스의 부모 프로세스(일반적으로 셸)에게 넘어간다
- 터미널 상에서 입출력 작업이 필요하지 않다면 백그라운드 상태에서 수행을 계속할 수 있다.
- 대화형 프로세스 제어 또는 작업 제어(job control)는 프로세스를 포그라운드나 백그라운드로 전환하는 역할을 하고, 포그라운드에서 실행을 계속할 수 있도록 제어한다

Process Management

● 배치 프로세스

- 일련의 작업을 몰아서 특정 시각에 실행 시키는 것이며, 일반적으로 터미널과의 입/출력 교류가 전혀 없다
 - 중요하지 않은 작업에 대해 시스템의 사용률이 낮을때 처리하는데 매우 유용
 - batch 명령어와 at 명령어를 사용한다

● 데몬

- 데몬은 특정 서비스를 위해 백그라운드 상태에서 계속 실행되는 서버 프로세스이다
 - 일반적인 서비스는 각각의 서비스가 사용하는 포트를 관리하는 데몬이 존재한다
 - 다른 데몬들에게 할당된 포트를 관리하는 특별한 용도의 데몬도 존재한다

Process Management

🐼 좀비 프로세스

🐼 Zombie process?

- 🐼 자식 프로세스가 종료 되었는데, 부모 프로세스에게 SIGCHLD 전달이 안되었을 때 혹은 손실 (lost) 되었을 때 발생
- 🐼 자식 프로세스는 종료 되었는데 부모 프로세스는 종료되지 않았거나, wait() 계열 함수를 호출 해서 자식 프로세스를 정리하지 않았을 때 발생
- 🐼 파일 등의 모든 자원 및 메모리 이미지 역시 해제 하였으며 프로세스 테이블만 유지하고 있는 상태이다

Process Management

• Foreground & Background Process

• Foreground Process?

- 셸 상태에서 명령을 내리면 사용자는 해당 프로세스가 종료될 때까지 기다려야 함

```
# cp -R /usr/sbin/ /tmp/
```



• Background Process?

- 일반적으로 명령뒤에 '&'를 붙여서 동작시키며, 명령을 내린 사용자는 자신이 하고자 하는 다른 명령어를 계속 실행 할 수 있음

```
# cp -R /usr/sbin/ /tmp/ &
[1] 26662
#
[1]+  Done
2 cp -i -R /usr/sbin /tmp/ &
#
```

Process Management

🔹 관련 명령어(jobs)

🔹 jobs

- 🔹 백그라운드로 실행중인 프로세스나 현재 중지된 프로세스 목록을 출력해줌
- 🔹 사용예제(-l 옵션은 PID 와 같이 출력)

```
# j o b s  -l
```


Process Management

● 관련 명령어(fg)

● fg

- fg는 Background Process 를 foreground Process로 전환하는 명령어
- Background Process 가 여러 개 존재할 경우 별도의 작업번호를 부여하지 않으면 현재 수행중인(+기호가 붙은) 작업을 전환
- 사용예제

```
# fg %작업번호
```

Process Management

● 관련 명령어(bg)

● bg

- Foreground Process를 Background Process 로 전환하는 명령
- 보통 실행시키고 있는 Foreground Process 에서 CTRL + Z 키를 눌러 작업을 잠시 중지시킨 후에 bg 명령어를 사용하여 작업을 백그라운드로 전환 함
- 사용예제

```
# find / -name '*.txt' 2> /dev/null >list.txt  
CTRL + Z  
# bg
```

```
# find / -name '*.txt' 2> /dev/null >list.txt &  
[1] 35432  
#
```

Process Management

• at / batch 를 이용한 배치 프로세싱

• at와 batch는 /bin/sh를 이용하여 지정된 시점에 실행할 명령을 표준 입력이나 지정한 파일에서 읽어들이다

명령	의 미
at	지정한 시간에 수행할 명령어 설정 및 파일 지정
atq	현재 사용자의 등록된 작업 목록을 출력하며, 관리자의 경우 모든 사용자의 작업 목록을 보여준다. 출력 형식은 작업번호, 날짜, 시간, 작업 구분(at, batch)
atrm	지정된 작업 번호에 해당하는 작업을 삭제한다
batch	시스템 부하 수준이 낮을 때 명령을 수행한다 부하 평균이 0.8이하 또는 atrun으로 지정한 값 이하로 내려갔을 때 수행한다

Process Management

● 부모 프로세스와 자식 프로세스

- 프로세스는 여러가지 기능을 수행함으로써 주어진 작업을 완료하며, 모든 기능이 완료되기 전까지는 종료될 수 없다
- 부모 프로세스는 여러 개의 자식 프로세스를 실행하여 다수의 작은 작업들을 동시에 처리하도록 할 수 있다

● exec

- 한 프로세스가 다른 프로세스를 생성(spawn)할 경우 원래의 프로세스가 더는 남아 있을 필요가 없다면 exec호출을 통해서 다른 프로그램을 실행하고 새로운 프로세스로 자신을 대체 할 수 있다

● fork

- exec에 반해 원래 프로세스가 계속 존재해야 한다면 fork를 호출하여 자신의 복사본 프로세스를 먼저 만들고 복사본 프로세스에서 다시 exec를 호출하여 다른 프로그램을 실행하는 새로운 프로세스로 자신을 대체한다

Process Management

• init 프로세스

- 리눅스 시스템에서 가장 중요한 프로세스이며, 부트 과정에서 커널이 수행하는 마지막 작업이다

• init 프로세스의 역할

- 파일시스템의 점검과 마운트
- 필요한 데몬 동작
- 시스템 구성에 필요한 기타설정
- 부모 프로세스가 없는 자식 프로세스에 대한 부모 프로세스 역할

Process Management

● 관련 명령어(ps)

● ps

- 현재 동작하고 있는 프로세스들의 상황(스냅샷)을 보여줌
- 실시간으로 프로세스의 동작 상황을 확인하고 싶을 때는 top 명령어를 사용
- Process 정보의 기본적인 내용은 proc 파일 시스템에 있는 stat 의 내용에서 가져온다

Process Management

- 관련 명령어(ps)
- ps 명령어 주요 옵션

옵션	의미
a	현재 실행중인 전체 사용자의 모든 프로세스 출력
e	모든 프로세스 정보 출력
l	Long Format
u	프로세스 사용자 정보와 시작 시간 등을 출력
x	제어 터미널을 갖지 않는 프로세스 출력
f	Full format 정보

Process Management

🔗 관련 명령어(ps)

🔗 ps 명령어 주요 필드

필드	의미
USER	프로세스를 실행시킨 사용자
PID	프로세스 ID
%CPU	최근 1분간 프로세스가 사용한 CPU 시간 백분율
%MEM	최근 1분간 프로세스가 사용한 실제 메모리의 백분율
RSS	현재 프로세스가 사용하고 있는 실제 메모리 크기
TTY	프로세스를 제어하고 있는 터미널
STAT	프로세스 상태 코드
START	프로세스의 시작 시간

Process Management

🔗 관련 명령어(ps)

🔗 STAT 필드 구성 요소

속성	의미
R	실행 중 혹은 실행할 수 있는 상태
S	인터럽트 가능한 수면 상태
I	휴식 상태
T	정지 상태
W	스왑 아웃된 상태
P	페이지 대기
D	인터럽트 불가능한 수면상태 (보통 디스크 I/O)
N	nice 로 실행 우선순위가 낮아진 상태
Ss	Session leader
+	Foreground process 그룹
Z	Defunct(좀비) 프로세스

Process Management

● 관련 명령어(ps)

● ps 명령어 예제

- 현재 실행 중인 전체 사용자 정보와 프로세스 시작 시간 등 출력

```
# ps -au
```

```
# ps -au
USER  PID  %CPU  %MEM  VSZ   RSS  TTY   STAT  START  TIME  COMMAND
root  10842  0.0   0.2   1668  464  tty2   Ss+   06:53   0:00  /sbin/mingetty tty2
root  10843  0.0   0.1   1668  440  tty3   Ss+   06:53   0:00  /sbin/mingetty tty3
root  10867  0.0   0.1   1668  440  tty4   Ss+   06:53   0:00  /sbin/mingetty tty4
root  10868  0.0   0.1   1668  436  tty5   Ss+   06:53   0:00  /sbin/mingetty tty5
root  10870  0.0   0.1   1668  440  tty6   Ss+   06:53   0:00  /sbin/mingetty tty6
root  25362  0.0   0.6   5908  1536  tty1   Ss+   13:44   0:00  -bash
root  26551  1.4   0.6   5668  1456 pts/1  Ss    14:20   0:00  -bash
root  26584  0.0   0.3   5288  888  pts/1  R+    4:20   0:00  ps au
```

Process Management

● 관련 명령어(ps)

● ps 명령어 예제

- 사용자가 정의한 형식(o) 으로 모든 프로세스(e) 를 보여줌

```
# ps -eo pi d, state, ni ce, args
```

```
# ps -eo pi d, state, ni ce, args | more
```

PID	S	NI	COMMAND
1	S	0	i ni t [5]
2	S	-	[mi grati on/0]
3	S	19	[ksofti rqd/0]
4	S	-	[watchdog/0]
5	S	-	[mi grati on/1]
6	S	19	[ksofti rqd/1]
7	S	-	[watchdog/1]
8	S	-5	[events/0]
9	S	-5	[events/1]
10	S	-5	[khel per]
11	S	-5	[kthread]

Process Management

🔹 관련 명령어(pstree)

🔹 pstree

- 🔹 현재 프로세스의 상황을 트리 형식으로 보여줌
- 🔹 pstree 명령어 주요 옵션

옵션	의미
-n	PID 순으로 정렬
-p	프로세스명과 PID 함께 출력

Process Management

● 관련 명령어(pstree)

● pstree 명령어 예제

- PID 순으로 정렬하며, 프로세스명 과 PID 출력

```
# pstree -np
```

```
# pstree -np

i ni t(1)---mi grati on/0(2)
|   -ksofti rqd/0(3)
|   -watchdog/0(4)
|   -events/1(9)
|   -khel per(10)
|   -kthread(11)---xenwatch(13)
|                   | -xenbus(14)
|                   | -kbl ockd/0(17)
|   -udevd(585)
|   -audi td(9072)---{audi td}(9073)
|                   ` -audi spd(9074)---{audi spd}(9075)
|   -sysl ogd(9097)
|   -i rqbal ance(9112)
|   -portmap(9134)
|   -rpc. statd(9163)
|   -rpc. i dmapd(9200)
|   -vmware-guestd(9591)
```

Process Management

🌐 SIGNAL

🌐 What is SIGNAL?

- 🌐 프로세스에게 발생하는 비동기적인 사건(asynchronous event)
- 🌐 한 프로세스가 다른 프로세스에게 보낼 수도 있고, 커널이 프로세스에게 보낼 수도 있음
- 🌐 비동기적이므로 어느 시점에서 시그널이 발생할지 미리 예측할 수 없음

Process Management

🌐 SIGNAL

🌐 SIGNAL 이 발생하는 경우

- 🌐 키보드를 통해 발생
 - 🌐 CTRL + C(SIGINT), CTRL + Z(SIGSTOP)
- 🌐 다른 프로세스에 의해서 발생
- 🌐 커널 내부에서 발생
 - 🌐 Kill(), alarm() 과 같은 시스템 콜

Process Management

🔵 관련 명령어(kill)

🔵 kill

- 🔵 프로세스나 프로세스 그룹에게 지정된 시그널을 보내줌

🔵 주요 시그널

시그널	의미
SIGHUP(HUP) <1>	로그아웃, 설정 파일 다시 읽기
SIGINT(INT) <2>	키보드에 의한 실행 중지, CTRL + C
SIGKILL(KILL) <9>	강제 종료, emergency kill
SIGTERM(TERM) <15>	실행 종료, 안전한 종료
SIGSTOP (STOP) <19>	실행 정지, CTRZ + Z

Process Management

- 관련 명령어(kill)

- kill 명령어 사용예제

- PID 1234 프로세스에게 SIGTERM(기본값) 신호를 보냄

```
# kill PID
```

- 다수의 프로세스 강제종료(SIGKILL 옵션을 주어도 동일)

```
# kill -9 PID1 PID2 PID3 PID4
```

- 프로세스를 정보를 다시 읽어들임

```
# kill -HUP 1234
```

Process Management

● 관련 명령어(kill)

- kill 명령어 사용예제
 - 작업번호 단위로 종료

```
# kill %작업번호
```

● killall

- kill 명령은 다수의 경우 각각의 PID 값을 알아내고 모두 입력해야 하나 데몬에 의해 동작되는 모든 프로세스에게 한번에 같은 시그널을 보낼 경우에 사용됨
 - 프로세스명 또는 데몬명으로 프로세스 종료시킴

```
# killall %작업번호
```

```
# killall -s SIGHUP sshd
```

Process Management

🔵 관련 명령어(skill)

🔵 skill

🔵 시스템에 접속해 있는 사용자, 터미널, PID, command를 종료 시키는데 사용됨

🔵 옵션

🔵 -t (tty or pty).

🔵 -u username.

🔵 -p process ID number.

🔵 -c command name.

Process Management

● 관련 명령어(skill)

● skill 명령어 사용예제

- testuser1 이라는 사용자를 시스템에서 제거

```
# skill -KILL -u testuser1
```

- Testuser1 사용자의 특정 프로세스(top)만 제거

```
# skill -KILL -u testuser1 top
```

- 특정 터미널(pts/0)에 접속해 있는 사용자 추방

```
# skill -KILL -t pts/0
```

- 특정 사용자를 정지 시킴

```
# skill -KILL -p PID번호
```

File System & Disk management

File System

What is file system

- 컴퓨터의 파일 시스템은 파일과 그 안에 든 자료를 저장하고 찾기 쉽도록 유지 관리하는 방법이다.
- 파일 시스템은 하드 디스크나 CD-ROM과 같은 물리적 저장공간을 저장장치로 활용할 수도 있지만, *NFS*와 같은 파일 시스템에서는 네트워크 상에 존재하는 파일에 접근하기 위한 가상적 인터페이스인 경우도 있다.
- **학술적으로**, 파일 시스템은 자료를 계층적으로 **저장, 탐색, 접근, 조작**하기 위한 추상적 자료구조의 집합으로 정의된다.

File System

- 리눅스가 지원하는 파일시스템

- 리눅스가 지원하는 파일시스템은 매우 뛰어나기는 하지만 완벽하지는 않다

- 지원되는 파일시스템 찾기

- 대부분의 리눅스 파일시스템은 커널에 포함되어 있으며, 레드햇의 경우 대부분의 파일시스템을 기본 구성에 포함시키고 있다

- 추가적인 파일시스템들은 모듈형태로 제공 되어지고 있으며 다음의 경로에서 확인할 수 있다.

```
/lib/modules/x.y.z/kernel/fs
```

File System

● 파티션이란?

- 연속된 저장 공간을 하나 이상의 연속되고 독립된 영역으로 나누어서 사용할 수 있도록 정의한 규약이다
 - 파티션을 나누기 위해서는 저장장치에 연속된 공간이 있어야한다
 - 하나의 하드디스크에는 여러 파티션을 나눌 수 있지만, 두 개의 하드디스크를 이용해서 하나의 파티션을 만들 수는 없다
- 단일 파티션과 다중 파티션의 가장 큰 차이점은 MBR의 유무이다
 - DOS파티션에서만 해당되는 것으로 파티션이 나뉜 경우 각 파티션들을 관리할 수 있는 파티션 테이블과 부팅이 가능하도록 하는 부트 프로그램이 있는 영역이다.
 - 물리적으로는 디스크의 첫 번째 섹터를 의미

File System

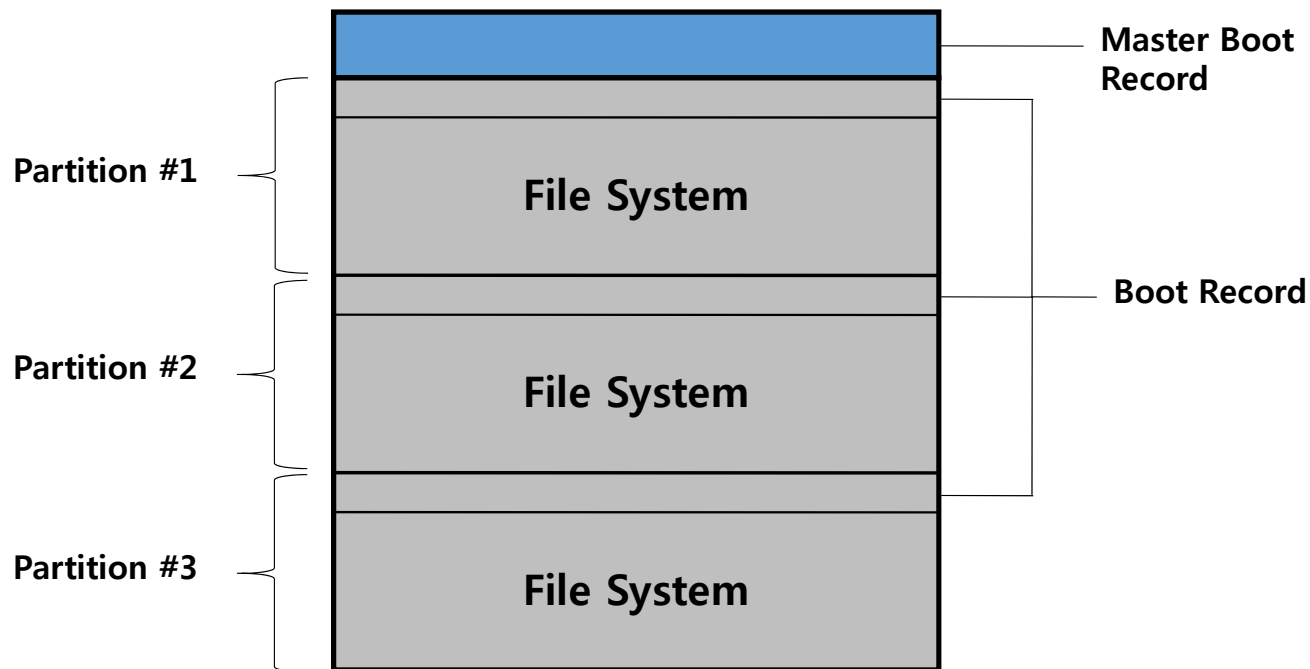
● 파티션의 용도

- 파티션을 나누는 용도는 개인의 사용용도에 따라 차이가 있다
 - 하나의 물리적인 디스크를 여러 논리 영역으로 나누어 관리를 용이하게 하기 위해
 - OS영역과 Data 영역으로 나누어 OS 영역만 따로 포맷 및 관리하기 위해
 - 여러 OS를 설치하기 위해
 - 하드디스크의 물리적인 배드 섹터로 특정 영역을 잘라서 사용하기 위해

File System

● MBR이란?

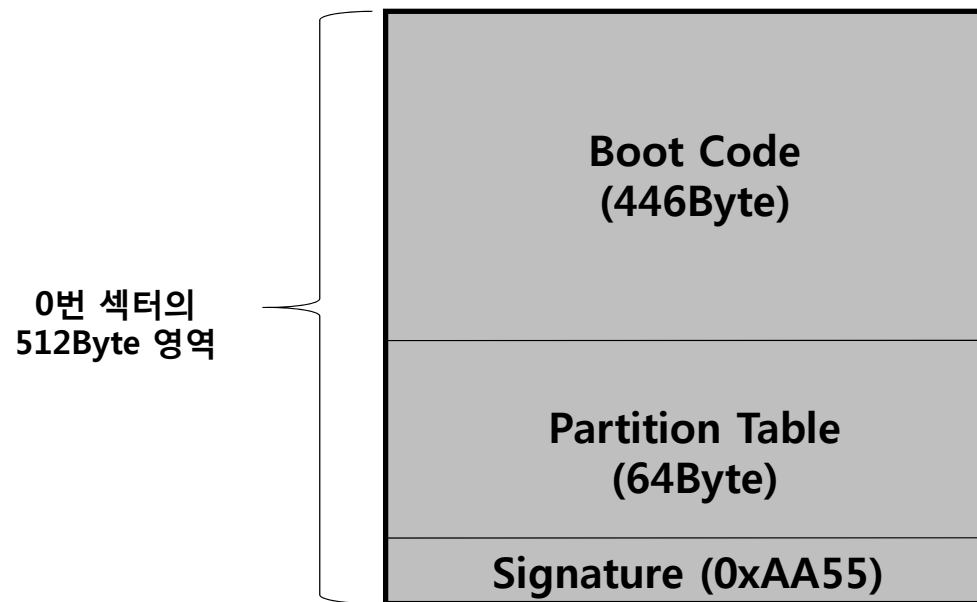
- MBR(Master Boot Record, 마스터 부트 레코드)은 말 그대로 Boot Record들의 메인 Boot Record 라고 할 수 있다



File System

● MBR의 구조

- MBR의 구조는 크게 두 영역으로 나눌수 있다. 부트코드(Boot Code)영역과 파티션 정보를 저장하고 있는 파티션 테이블(Partition Table)영역 이다



File System

● 파티션의 종류

- 파일시스템의 종류는 벤더에 따라 그 수가 굉장히 많다. 그 만큼 파티션의 종류도 다양각색이다.

● DOS Partition Table

- 현재 가장 많은 OS가 채택하여 사용하고 있는 파티션이며, 0번 섹터 중 Boot Code가 사용하고 남은 64Byte를 파티션 테이블로 사용한다
- 파티션 1개의 정보는 16Byte 이며 물리적으로 생성할 수 있는 개수는 4개이다

● BSD Disk Label

- 하나의 섹터 안에 파티션 정보를 포함한 디스크 정보를 모두 담아놓으며, 1번 섹터 148번지부터 403번지까지 총 256Byte를 파티션 테이블로 사용한다
- 파티션 1개의 정보는 16Byte 이며 물리적으로 생성할 수 있는 개수는 16개이다

File System

Apple Partition Map

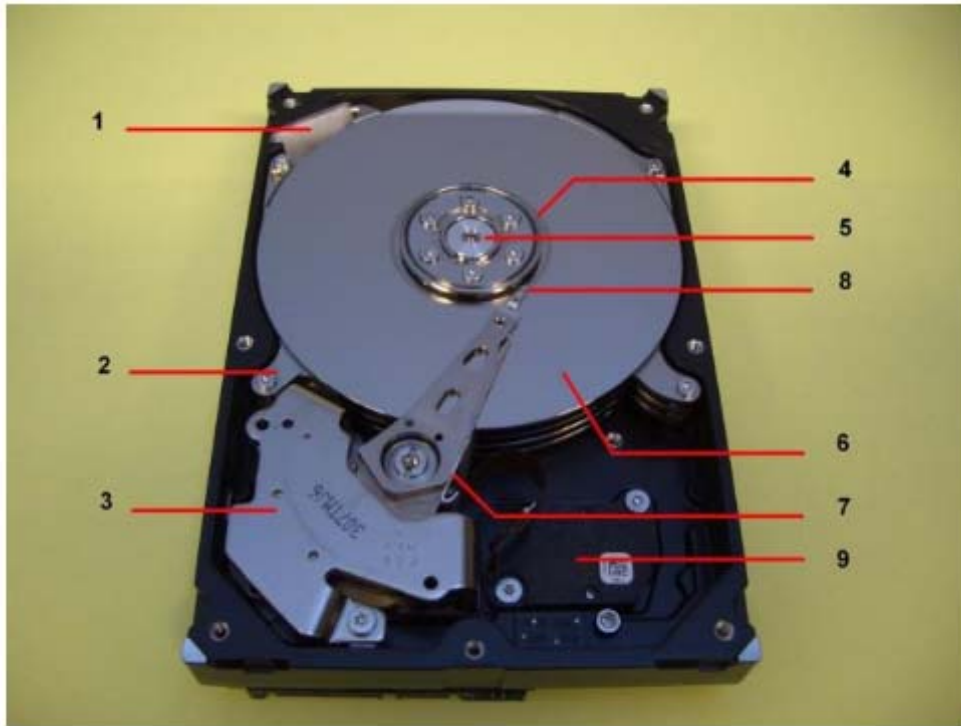
- BSD 계열의 파티션 기록 방식을 채택하여 사용하며, 다른 OS와 달리 0번 섹터를 사용하지 않으며 1번 섹터 512Byte 전체가 파티션 테이블로 사용한다
- 파티션 1개의 정보는 16Byte 이며 물리적으로 생성할 수 있는 개수는 32개이다
- MBR정보는 Firmware에 기록되어 있다

Solaris Disk Label

- Sun Sparc 에서는 0번 섹터에 Disk Label이 위치하며, 444번지 부터 64Byte에 걸쳐 8개의 파티션이 생성 가능하다(8Byte 사용)
- 파티션의 타입, 상태정보, Time stamp등의 추가 적인 정보는 VTOC(Volume Table of Contents)라는 데이터 영역에 기록되어 있으며, Disk Label의 128번지 부터 134Byte를 사용 하고 있으며 파티션 정보 뿐만 아니라 Disk의 볼륨명, 사용하는 파티션의 개수, 부팅정보 등이 추가로 기록된다.

IBM 호환 장비에서는 DOS Partition Table을 사용한다

- 기존의 DOS Partition Table 과는 조금 다른 변형된 형태를 사용하며 총 16개의 파티션을 생성할 수 있다.(Solaris File System Partition사용기준)



Key to Illustration Inside a Hard Drive:

1. Air filter.
2. Platter spacers.
3. Permanent magnets.
4. Platter landing zone.
5. Platter motor.
6. Data platter.
7. Read/Write heads' pre-amplifier circuit (in this case mounted on the side of the arm assembly).
8. Read/write heads.
9. Connections from the read/write heads through to the printed circuit board at the back.

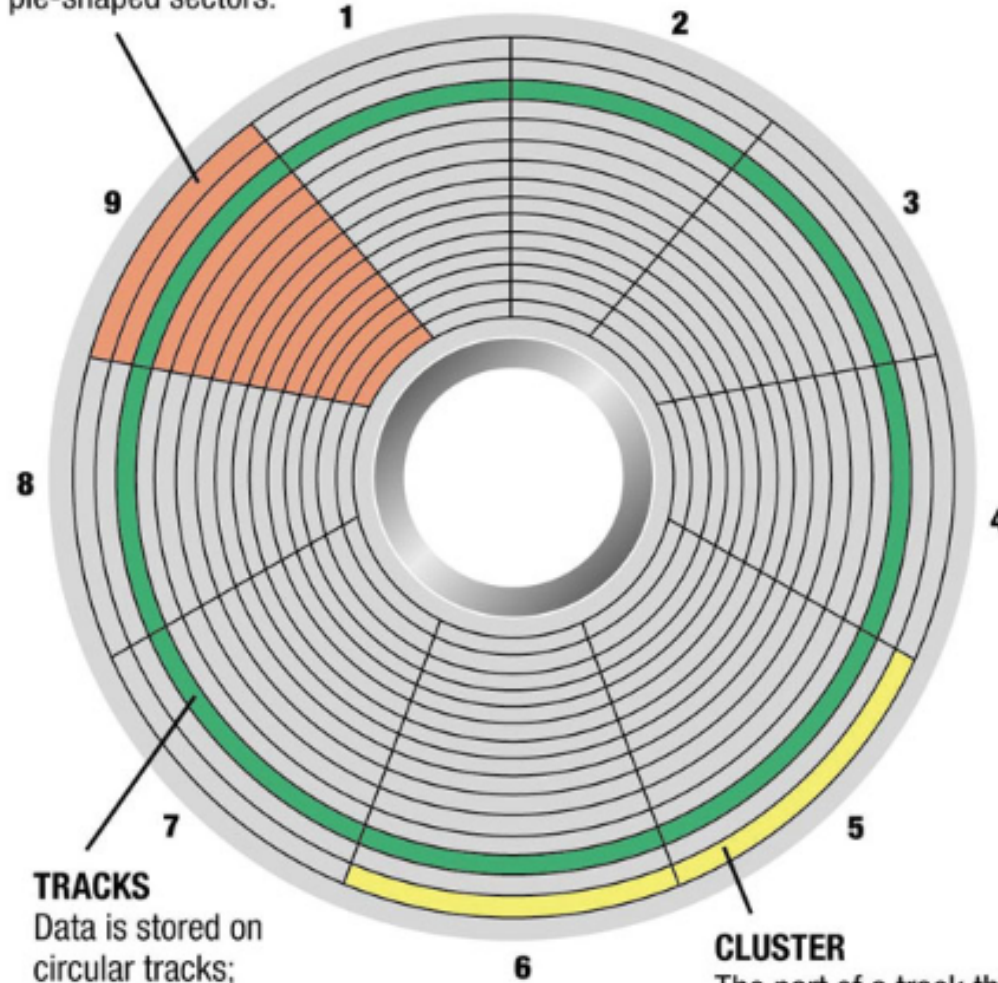


Key to Illustration of Hard Drive PCB:

1. Electrical connections to platter motor.
2. Microprocessor chip.
3. Programmable flash memory chip (stores the PCB portion of the drive's firmware-see below).
4. Platter motor controller chip.
5. S-ATA data connector (the connection between the hard drive and the outside world).
6. S-ATA power connector (as the name suggests, provides the drive with DC power).

SECTORS

A disk is divided into pie-shaped sectors.



TRACKS

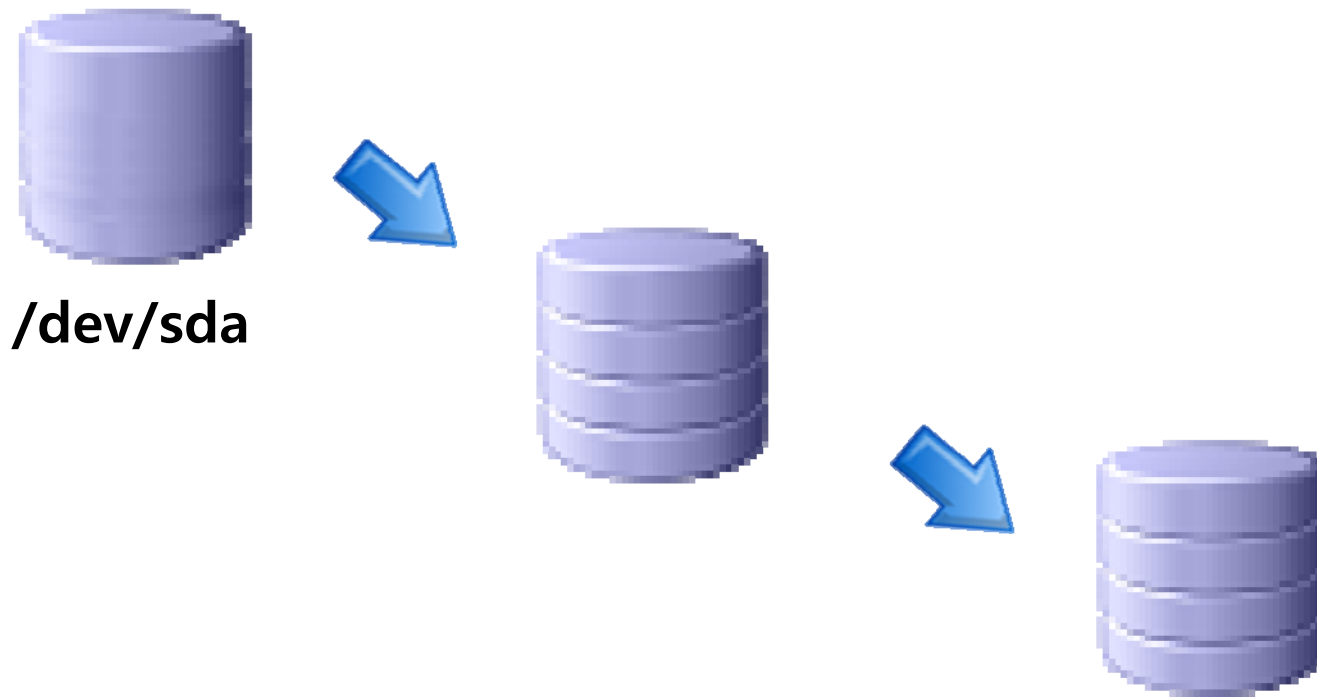
Data is stored on circular tracks; the 0s and 1s are represented magnetically.

CLUSTER

The part of a track that crosses two or more adjacent sectors forms a cluster, the smallest addressable unit of disk storage.

Disk Partitioning

- 리눅스의 디스크 파티션 생성 소프트웨어
- 파티션은 Primary, Extended, Logical Partition 으로 구분된다.



Disk Partitioning

🌐 Fdisk

🌐 fdisk 명령어 사용 예제

- 🌐 시스템에 연결된 디스크 정보 확인

```
# fdisk -l
```

- 🌐 특정 디스크 정보 확인

```
# fdisk -l /dev/sda
```

- 🌐 디스크 파티션 구성하기

```
# fdisk /dev/sdb
```

Creating Filesystem

Creating Filesystem

● 기본 파일 시스템 및 계층구조

● What is File System?

- 파일과 그 안에 든 자료를 저장하고 찾기 쉽도록 유지 관리하는 방법
- OS 가 파일을 시스템의 디스크상에 구성하는 방식
- OS 는 해당 파티션을 기반으로 시스템의 디스크 파티션 상에 파일들을 연속적이고 일정한 규칙을 가지게 저장
- 파일 시스템은 이러한 규칙들의 방식을 제시하는 역할을 함

Creating Filesystem

● 기본 파일 시스템 및 계층구조

● 파티션 VS 파일 시스템

- 파일 시스템은 파티션을 구성해 주는 역할을 한다
- 파일 시스템을 포함하지 못한 파티션은 파일 시스템이 사용될 수 있도록 초기화되고 파일 정보를 기록하기 위한 형식을 만들어야 한다
- 위의 두 과정을 거쳐야 파티션은 파일 시스템으로 사용될 수 있다

Creating Filesystem

● 기본 파일 시스템 및 계층구조

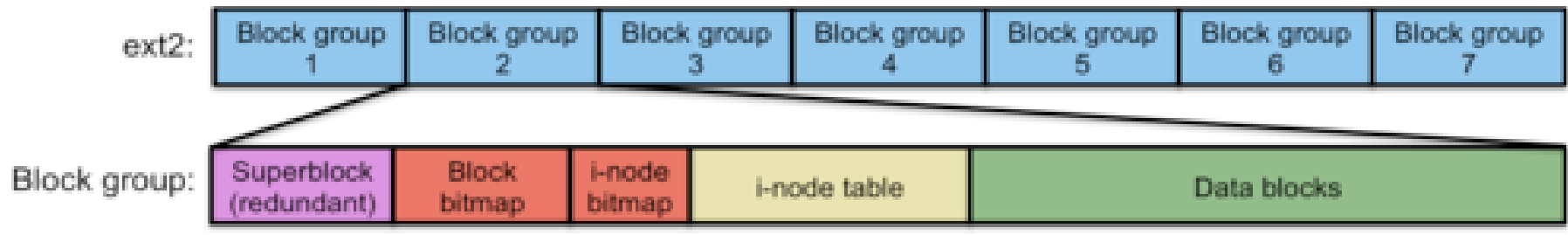
● 기본적인 파일 시스템의 구성 요소

- Boot Block(MBR) : 부팅할 때 실행되어야 할 코드가 담긴 부분
- Super Block : 파일 시스템 정보를 가지며 파일 시스템의 크기 등과 같은 파일 시스템의 전체적인 정보를 가진다
EX) 파일시스템이 생성된 시각, Inode 개수, mount 횟수
- Inode(meta data) : 파일의 이름을 제외한 해당 파일의 모든 정보를 가진다
- Data Block : data가 존재하는 Block

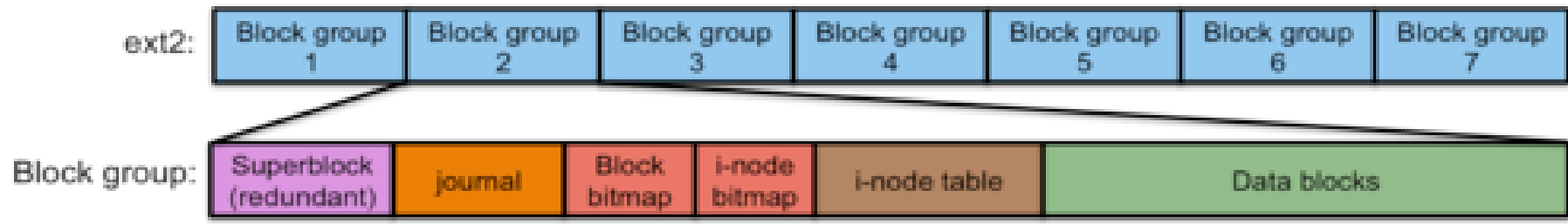
Creating Filesystem

● 기본 파일 시스템 및 계층구조

● ext2



● ext3



UFS / EXT2,3

● UFS (Unix File System)

- 유닉스의 대표적인 File System
- 대부분의 유닉스들(BSD계열, HP-UX, Apple OS X, Sun Solaris)이 UFS를 각각의 OS에 맞게 변형해서 사용

● EXT2

- UFS에서 불필요한 구조들을 제거
- UFS 수준의 속도와 안정성을 가지고 있음

● EXT3

- EXT2에 저널링 기능을 추가한 File System
- 리눅스의 기본 File System

Creating Filesystem

🌐 file system 생성

🌐 mkfs, mke2fs 를 이용하여 원하는 파일 시스템을 생성 한다.

🌐 mkfs 를 이용하여 ext2 파일 시스템 생성

```
# mkfs -t ext2 /dev/sdb1
```

🌐 mkfs 를 이용하여 ext3 파일 시스템 생성

```
# mkfs -t ext3 /dev/sdb2
```

🌐 mke2fs 를 이용하여 ext2 파일 시스템 생성

```
# mke2fs /dev/sdb1
```

🌐 mke2fs 를 이용하여 ext3 파일 시스템 생성

```
# mke2fs -j /dev/sdb2
```


Mount

Mount

🌐 What is Mount?

- 🌐 파일 시스템 구조 내에 있는 일련의 파일들을 사용자나 사용자 그룹들이 이용할 수 있도록 만드는 것
- 🌐 리눅스 환경에서 "mount"라는 명령은 논리적으로 디스크와 디렉토리를 붙이는 것을 말함
- 🌐 지역이나 원격으로 행해질 수 있으며, 지역 마운트는 하나의 논리적 시스템으로서 행동하도록 여러대의 디스크 드라이브들을 하나의 컴퓨터 상에 접속한다
- 🌐 원격 마운트는 다른 컴퓨터 상의 디렉토리들에 접속하기 위해 NFS(Network File System) 을 사용한다

Mount

🌐 파일 시스템 사용하기

- 🌐 Windows시스템에서는 파티션을 생성하면 자동으로 "드라이브명" 이라고 하는 특정문자에 자동으로 연결되어 지지만 Unix/Linux는 관리자가 직접 파티션을 특정 문자(디렉토리)에 연결하여야 하는데 그것을 마운트 라 한다.

🌐 파일시스템의 마운트와 언마운트

- 🌐 리눅스에서는 파티션을 연결하는데 두 가지 방법이 있다.
 - 🌐 mount 명령어를 통해 수동으로 연결하는 방법
 - 🌐 /etc/fstab 파일 내에 마운트할 항목을 추가하여 부트시에 자동으로 마운트 하는 방법
- 🌐 파일시스템의 사용을 중지할때는 umount 명령어를 사용한다.

Mount

📍 Mount 명령어 사용형식

📍 Syntax : mount [Option] [Device] [Directory]

📍 Option

- 📍 -a : /etc/fstab에 기술되어있는 모든 파일시스템을 마운트 한다.
- 📍 -t : 파일시스템의 형식을 지정한다.
- 📍 -n : /etc/mtab 파일에 정보를 남기지 않고 마운트한다.
- 📍 -o : -o [Option] 뒤에 콜론(",") 으로 다음 옵션을 사용할 수 있다.
 - 📍 async : 파일시스템에 대한 I/O가 비동기적으로 이뤄지도록 한다.
 - 📍 defaults : rw, suid, dev, exec, auto, nouser, async를 기본 옵션으로
 - 📍 nosuid : SetUID, SetGID를 무시하게 된다.
 - 📍 suid : SetUID, SetGID가 효력을 발휘할 수 있게 해준다.
 - 📍 remount : 이미 마운트된 파일시스템을 다시 마운트 한다.

사용한다.

📍 Ex)

📍 # mount -t ext3 /dev/sda1 /test

Umount

- umount : 장치연결 해제하기

- 파일시스템의 연결을 해제하여 사용을 중지한다.

- Syntax : umount [Option] [Device] or [Directory]

- Option

- -a : /etc/fstab에 기술되어있는 모든 파일시스템의 마운트를 해제할 때 사용하지만, 정상적인 상황에서 실행해서는 안된다.
- -f : 강제로 연결해제 할때 사용할 수 있으며 NFS서비스에 연결되어있는 파일시스템을 연결해제 할 때 이용할 수 있다.

- Ex)

- # umount /dev/sda1

- # umount /test

Mount

- Mount 명령어 사용 예
- 현재 마운트된 정보를 볼 수 있다

```
# mount
```

```
# mount
/dev/sda3 on / type ext3 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gi d=5,mode=620)
/dev/sda6 on /usr type ext3 (rw)
/dev/sda2 on /tmp type ext3 (rw)
/dev/sda1 on /boot type ext3 (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
none on /proc/fs/vmblock/mountPoint type vmblock (rw)
```

Mount

🌐 Mount 명령어 사용 예

🌐 Mounting

🌐 로컬 파일시스템 연결하기

```
# mount -t ext3 /dev/sdb2 /mount_test
```

🌐 CD-ROM 을 /mnt/cdrom 디렉토리에 연결

```
# mount -o ro /dev/cdrom /mnt/cdrom
```

🌐 원격 nfs 파일 시스템 마운트

```
# mount -t nfs [IP: /EXPORT DIR] [LOCAL DIR]
```

🌐 마운트 해제

```
# umount [DIR] / [Device Name]
```

File System Table

- 부팅시 자동 마운트 설정
 - /etc/fstab 파일에서 설정
 - 총 6개의 필드로 구성

필드	의미
첫 번째(fs_spec)	디바이스명
두 번째(fs_file)	마운트 포인트
세 번째(fs_vfstype)	파일 시스템 타입
네 번째(fs_mntops)	자동 마운트 관련
다섯 번째(fs_freq)	파일 시스템 덤프 관련
여섯 번째(fs_passno)	재부팅기간 파일 시스템 체크

File System Label

🔗 Filesystem Labels

🔗 명령어 사용 예제

- 🔗 /dev/sda1 파티션의 라벨을 조회

```
# e2label /dev/sda1
```

- 🔗 /dev/sda1 파티션의 라벨을 설정

```
# e2label /dev/sda1 bootsection
```

- 🔗 라벨을 이용한 mount

```
# mount LABEL=bootsection /boot
```

File System Table

● 부팅시 자동 마운트 설정

● /etc/fstab 파일 예제

LABEL=/boot	/boot	ext3	defaults	1	2
--------------------	--------------	-------------	-----------------	----------	----------

- LABEL=/boot : boot 디바이스를 의미
- /boot : 마운트될 디렉토리
- ext3 : 파일 시스템 타입
- Defaults: 자동 마운트 여부
- 1 : dump명령어에 의한 덤프가 필요할지에 대한 여부
- 2 : fsck 프로그램이 재부팅시 파일 시스템 체크에 대한 여부를 지정하며 1은 root(/) 파일 시스템이, 2는 그밖의 파일 시스템이다

File System Label

🔵 What is Filesystem Labels?

🔵 Label은 장치를 참조하는 방법중 하나이다

🔵 e2label 명령어를 통해 ext2/ext3 파일시스템의 라벨을 변경하거나 조회할 수 있다

🔵 e2label 명령어는 superblock 안에 label을 기록한다

Inode

Inode

🌐 What is Inode?

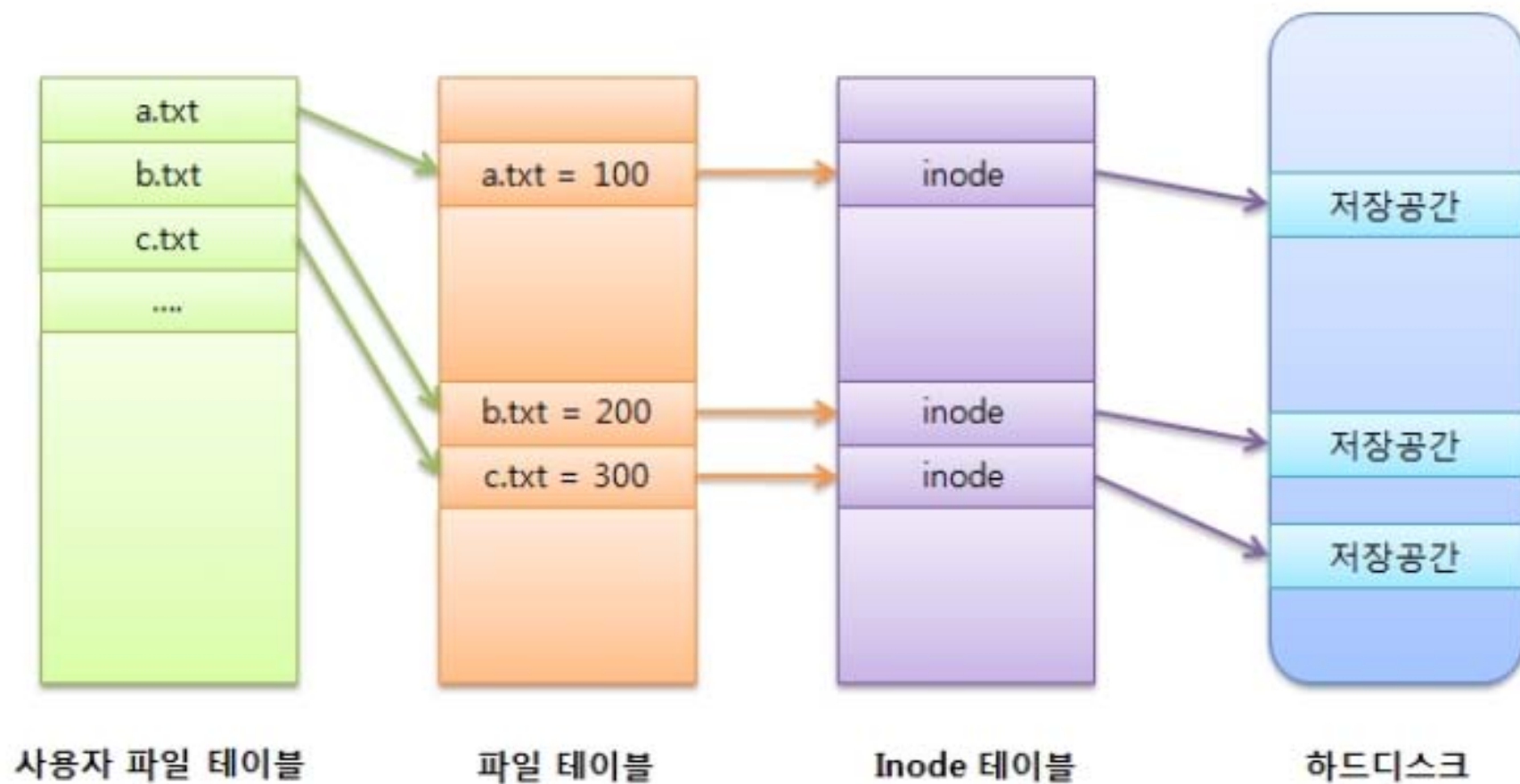
- 🌐 사용자 프로세스는 커널을 통하여 디스크의 파일을 접근한다
- 🌐 이를 위해 커널 내에는 현재 사용 중인 파일에 대한 자료 구조를 유지한다
- 🌐 커널 내에서 파일 시스템과 관련하여 중요한 자료 구조 중 하나는 Inode 이다
- 🌐 Inode는 해당 파일에 대한 대부분의 정보를 가진다

Inode

● Inode의 내용

- 파일의 소유권 및 이용할 수 있는 여부에 대한 정보
- 파일 내용이 들어있는 디스크 내의 물리적 주소
- 파일의 링크 수
- 파일의 형태
- 파일의 크기
- 파일의 만들어진 시간, 최근 사용시간, 최근 수정시간
- Inode 의 최근 수정시간

inode



Inode

● Inode & File

● 파일의 생성

- 새로운 파일이 만들어 지면 그에 해당하는 inode가 i-list 안에 만들어지며, 그 inode의 inumber 와 파일이름이 디렉토리에 등록된다

● 파일의 삭제

- 파일을 삭제하면 그 파일에 대한 inode 의 파일 링크수가 하나 감소되고 디렉토리 entry 에서는 해당 파일의 inumber 가 zero 로 변한다. Inode의 파일링크수가 zero 가 되면 파일의 디스크 블록은 free 가 되며 inode 는 dellocate 상태가 된다

Inode

🌐 Inode & File

🌐 파일의 링크(Hard Link)

- 🌐 Hard Link 형식으로 파일을 링크시킬 경우는 디렉토리에 그 파일에 대한 새로운 이름이 등록되고, inumber는 본래 있던 파일의 inumber가 복사(동일한 Inode 사용) 된다. 이때 복사되는 파일의 inode 에서 파일의 링크수가 하나 증가된다

```
# ls -li
합계 0
384387 -rw-r--r-- 2 root root 0 4월 6 14:39 hardlink
384387 -rw-r--r-- 2 root root 0 4월 6 14:39 inode_test
```

Inode

🌐 Inode & File

🌐 파일의 링크(Symbol Link)

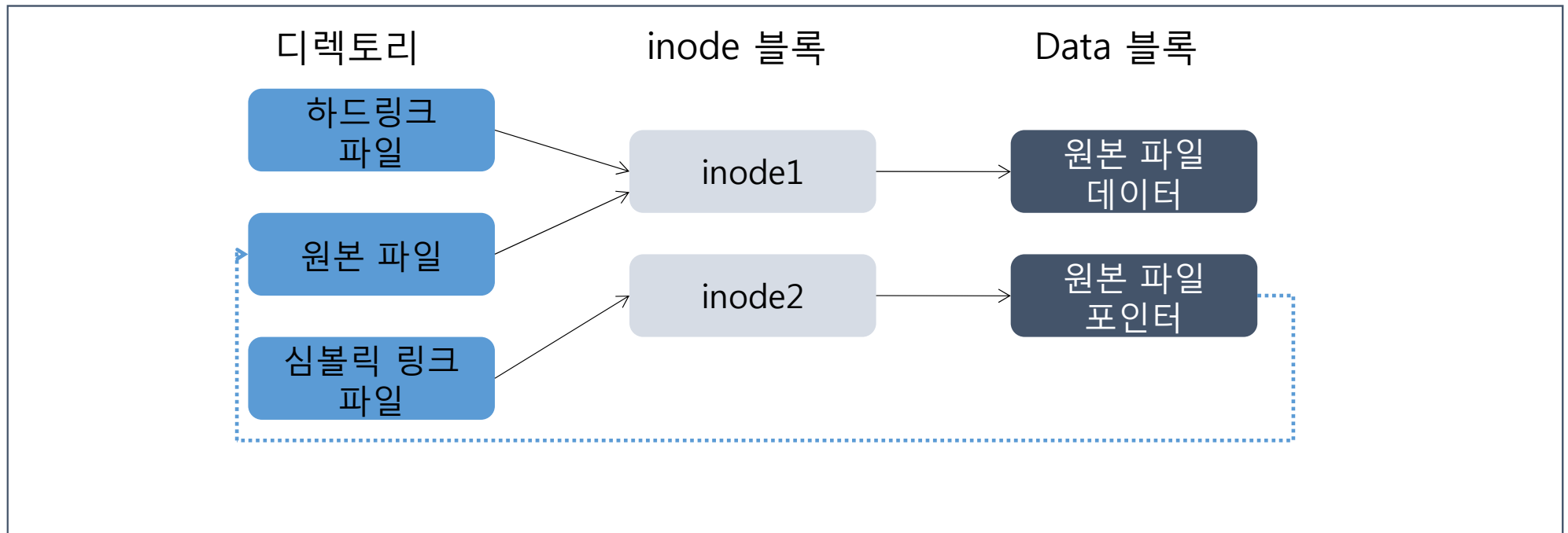
- 🌐 Symbol Link 형식으로 파일을 링크시킬 경우는 디렉토리에 그 파일에 대한 새로운 이름이 등록되고, 새로운 inode 를 생성하며 해당 inode는 원본 파일을 연결한다

```
# ls -li
합계 0
384387 -rw-r--r--  2 root root  0  4월  6 14:39 hard link
384387 -rw-r--r--  2 root root  0  4월  6 14:39 inode_test
384388 lrwxrwxrwx  1 root root 10  4월  6 14:40 symbol link -
> inode_test
```

Inode

🌐 Inode & File

🌐 Hard Link & Symbol Link



Inode

💡 Inode & File

💡 명령어 형식(Symbol)

```
# ln -s <link 대상파일명> <link 파일명>
```

💡 명령어 형식(Hard)

```
# ln <link 대상파일명> <link 파일명>
```

Inode

🌐 Inode & File

🌐 실습

- 🌐 /tmp/inodetest 디렉토리를 생성 후 inodetest 파일을 생성한다
- 🌐 Inodetest 파일 내에 "I am test file" 내용을 입력한다
- 🌐 Inodetest 파일에 대한 Hard Link 파일인 hardlnk 를 생성한다
- 🌐 Inodetest 파일에 대한 Symbol Link 파일인 symlnk 파일을 생성한다
- 🌐 Symlnk 파일에 대한 Symbol Link 파일인 symlnk2 파일을 생성한다
- 🌐 symlnk2 파일의 내용을 읽어서 원본 파일에 쓰인 내용이 출력 되는지 확인 한다

Swap Partition

Swap Partition

🌐 What is swap?

- 🌐 프로세스 메모리 이미지를 가지고 있는 임시 디스크 공간
- 🌐 물리 메모리공간의 요구가 크지 않다면, 프로세스 메모리의 이미지는 디스크에 있는 swap 공간에서 물리 메모리로 다시 가져옴
- 🌐 Swap 공간을 통해 시스템은 많은 물리적 메모리 공간을 확보할 수 있음

Swap Partition

🔵 Swap의 종류

🔵 Device swap

- 🔵 Logical 볼륨이나 파티션을 점유하고 있음
- 🔵 스와핑을 목적으로 따로 떼어둔 공간
- 🔵 덤프공간으로 사용되기 위해 설정되는 경우도 있음
- 🔵 로컬에서만 사용되어 지며, 원격으로 접근할 수 없음
- 🔵 빠르게 접근이 가능함

Swap Partition

- Swap의 종류

- File system swap

- 파일 시스템 안에 가용한 공간을 swap 공간으로 추가적으로 사용
- 가끔 Device swap 공간보다 더 많은 공간을 필요로 할 때 사용
- 시스템이 상당한 프로세싱을 수행해야 하기 때문에 device swap에 비해서 느림
- Local 혹은 remote 로 사용되어짐

Swap Partition

🌐 Swap 파티션 생성

🌐 Device swap 추가과정

- 🌐 적절한 크기의 새로운 디스크 추가
- 🌐 파티션 생성

🌐 추가한 파티션을 swap 장치로 만듦

```
# mkswap /dev/[SWAP파티션]
```

🌐 swap 활성화

```
# swapon /dev/[SWAP 파티션]
```

🌐 swap 공간 확인

```
# free
```

swap partition

- `dd if=/dev/zero of=swapdisk bs=1k count=100000`
- `losetup /dev/loop0 ./swapdisk`
- `fdisk /dev/loop0 (n,p,1,Enter,Enter,t,82,w)`

```
[root@sec data]# fdisk -l /dev/loop0
```

```
Disk /dev/loop0: 102 MB, 102400000 bytes
255 heads, 63 sectors/track, 12 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x19991f48
```

Device	Boot	Start	End	Blocks	Id	System
/dev/loop0p1		1	12	96358+	82	Linux swap / Solaris

- `mkswap /dev/loop0`
- `swapon /dev/loop0`

swap partition

```
[root@sec data]# mkswap /dev/loop0
mkswap: /dev/loop0: warning: don't erase bootbits sectors
        (dos partition table detected). Use -f to force.
Setting up swapspace version 1, size = 99996 KiB
no label, UUID=d69d6e21-512e-42d9-b735-32b37655861e
[root@sec data]# swapon /dev/loop0
[root@sec data]# swapon -s
```

Filename	Type	Size	Used	Priority
/dev/dm-1	partition	2064376	0	-1
/dev/loop0	partition	99992	0	-2

```
[root@sec data]# free
```

	total	used	free	shared	buffers	cached
Mem:	1030428	137408	893020	0	20612	34464
-/+ buffers/cache:		82332	948096			
Swap:	2164368	0	2164368			

Redhat Package Manager

Redhat Package Manager

- RedHat Package Manager
- 초창기 리눅스에는 없었던 패키지 개념을 도입하여 설치와 삭제, 업그레이드 등을 편리 하도록 만들어, 리눅스를 성장 시키는데 큰 도움을 제공

Redhat Package Manager

● RPM 패키지 파일의 구성

Vim-enhanced-7.0.109-3.i386.rpm

● Vim-enhanced : 패키지명

● 7.0.109-3 : 버전

● 7 : major 버전, 프로그램 자체가 완전히 변경 되었을 경우 변경

● 0 : minor 버전, 기능의 추가가 있을 경우에 올라간다

● 109 : patch 버전, 기존 기능에서 버그가 수정되는 경우 올라감

● 3 : release 버전, 배포하는 측에서 몇 번째로 만든 배포판 인지 를 나타낸다

● i386 : 아키텍처

Redhat Package Manager

Architecture	사용 가능한 CPU 및 설명
i386	intel 호환 386 CPU급 이상.
i586	intel 호환 586 펜티엄급 CPU 이상.
i686	intel 호환 686 펜티엄 프로급 CPU 이상
x86_64	x86호환 64bit CPU급에 사용가능. e.g. AMD 64bit CPU, Intel EM64T 지원 CPU 등등
noarch	플랫폼에 상관없이 설치되는 파일들 e.g. 글꼴 파일, 아이콘, 배경. 그림 파일 등등
sparc	SunMicro System 의 스팍 계열 CPU용
ppc	Power PC CPU용
src	Source 패키지, rpmbuild 프로그램을 사용하여 바이너리 패키지인 위의 i386, noarch, ppc 등의 패키지를 만들 수 있다.

Redhat Package Manager

모 드	설 명
install	<pre>rpm -i [rpm file]</pre> <p>일반적인 패키지 설치 방법이며 동일한 패키지가 설치되어 있다면 버전에 상관 없이 설치할 수 없다.</p>
Upgrade	<pre>rpm -U [rpm file]</pre> <p>동일 패키지가 설치되어 있지 않다면 install과 동일하다. 동일 패키지가 설치되어 있다면 버전 비교 후 이전 버전은 삭제하고 새 버전으로 업그레이드 한다.</p>

Redhat Package Manager

Mode

모 드	설 명
Freshen	<code>rpm -F [rpm file]</code> 동일 패키지가 설치되어 있는 경우에만 버전 비교 후 업그레이드 한다. 동일 패키지가 설치되어 있지 않다면 아무 일도 하지 않는다.
erase	<code>rpm -e [rpm 패키지 이름]</code> 패키지를 삭제한다.

Redhat Package Manager

🌐 설치 및 삭제모드의 추가옵션

옵 션	설 명
-v	verbose, 작업을 하는 과정을 자세히 보여준다.
-h	hash, 설치 작업시 진행 상황을 해쉬 마크(#)의 바(bar) 형태로 보여준다.
--test	작업을 테스트만 한다. 실제로 설치/삭제 하지는 않는다.
--force	작업을 강제로 한다. 무시할 수 있는 에러의 경우에는 무시하며, 설치시 같은 패키지가 있다면 덮어씌운다.
--nodeps	no dependancy, 패키지 의존성을 검사하지 않는다.

Redhat Package Manager

💡 명령어 예제

💡 Upgrade, verbose, hash

```
# rpm -Uvh vim-enhanced-7.0.109-3.i386.rpm
```

💡 의존성 무시하고 설치

```
# rpm -Uvh vim-enhanced-7.0.109-3.i386.rpm --nodeps
```

💡 삭제

```
# rpm -e vim-enhanced-7.0.109-3
```

Redhat Package Manager

질의 옵션(-q)

질의 옵션	설 명
-q [name]	name 패키지의 전체 이름(버전 포함)을 조회
-qa	시스템에 설치된 패키지 전체 목록을 조회
-ql [name]	name 패키지의 파일 리스트를 조회(설치 파일)
-qd [name]	name 패키지의 파일 리스트 중에 문서 파일만 조회
-qc [name]	name 패키지의 파일 리스트 중에 설정 파일만 조회
-qf [file]	file 이 어느 패키지에 속해 있는지 조회
-qs [name]	name 패키지의 현재 파일 목록 상태 조회 normal(정상), not installed(설치되지 않았음), replaced(파일은 있는데 내용이 다름)
-qR [name]	name 패키지가 필요로 하는 의존성을 조회
-qi [name]	name 패키지의 패키지 정보 조회 패키지명, 종류, 용량, 설치 날짜, 간단한 설명 등등..

YUM

YUM

● YUM - Yellowdog Updater Manager

● RedHat Linux에서는 제공하지 않다가 Fedora Core Project로 넘어오면서 RedHat , RedHat계열의 리눅스에 포팅 되어졌다.

● Yum은 **지정된 서버주소로부터 업데이트된 패키지들을 검사하여 다운로드하고 설치까지 처리**해주는 텍스트 기반의 업데이트 프로그램이다. 또한 **의존성 문제도 같이 검사하여 관련 패키지들을 자동으로 설치**해주기 때문에 이전에 다소 불편하던 RPM 기반의 프로그램 설치 및 업데이트를 대폭 개선한 패키지 관리자이다.

YUM

● YUM 설정 파일

- yum의 설정 파일은 다른 패키지와 마찬가지로 /etc 밑에 존재하며 설정파일의 위치는 다음과 같다.

● /etc/yum.conf

- yum의 환경설정이 들어있는 파일.
 - 업데이트 서버의 URL과 기타 세부적인 설정사항.

YUM

🐼 yum 사용법

🐼 yum -y [install / update / remove] [package_name]

🐼 업데이트 가능한 패키지 확인

```
# yum list
```

🐼 최신 패키지로 업데이트

```
# yum update
```

🐼 새로운 패키지 설치

```
# yum -y install [ package_name ]
```

🐼 패키지 삭제

```
# yum -y remove [ package_name ]
```

Tar, Gzip and Bzip

Tar, Gzip and Bzip

🌐 Tar / gzip / bzip

🌐 리눅스에서 자주 사용되는 파일 묶음 및 압축 도구들

🌐 형식 및 파일명

형식	파일명
tar	파일명.tar
gzip & gunzip	파일명.gz
bzip2 & bunzip2	파일명.bz2

Tar, Gzip and Bzip

🌐 Tar / gzip / bzip

🌐 Tar 의 주요 옵션

옵션	의미
c(create)	새로운 묶음을 생성
x(extract)	묶인 파일을 풀어줌
t(list)	묶음을 풀기 전에 목록을 보여줌
f(file)	묶음 파일명을 지정해줌
v(visual)	파일이 묶이거나 풀리는 과정을 보여줌
z	Tar + gzip
j	Tar + bzip2

Tar, Gzip and Bzip

- Tar / gzip / bzip

- 자주 사용되는 명령어 및 옵션

- Archive 생성

```
# tar cvf xi netd. tar /etc/xi netd. d/
```

- Archive 생성 + gzip 압축

```
# tar cvzf xi netd. tar. gz /etc/xi netd. d/
```

- Archive 생성 + bzip2 압축

```
# tar cvj f xi netd. tar. bz2 /etc/xi netd. d/
```

Tar, Gzip and Bzip

- Tar / gzip / bzip

- 자주 사용되는 명령어 및 옵션

- 확인

```
# tar tvf xi netd. tar
```

- 풀기

```
# tar xvf xi netd. tar
```

- gzip 압축 해제 + tar 풀기

```
# tar xvzf xi netd. tar. gz
```

Tar, Gzip and Bzip

- Tar / gzip / bzip

- 실습

- /bin 디렉토리의 모든 파일을 gzip압축 및 tar 로 묶어서
- /home 디렉토리 안에 binlist.tar.gz 라는 파일로 생성



