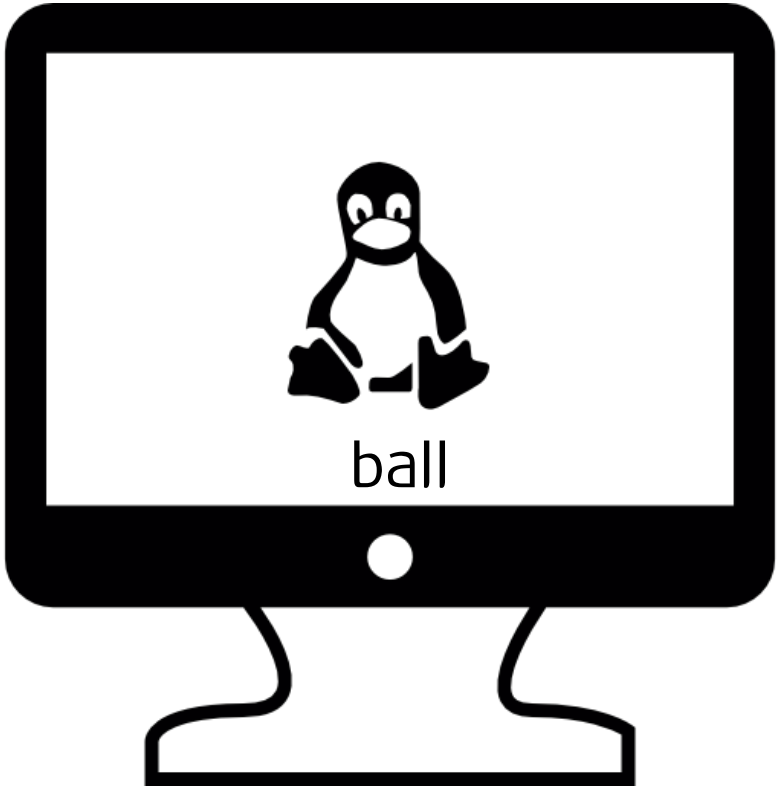
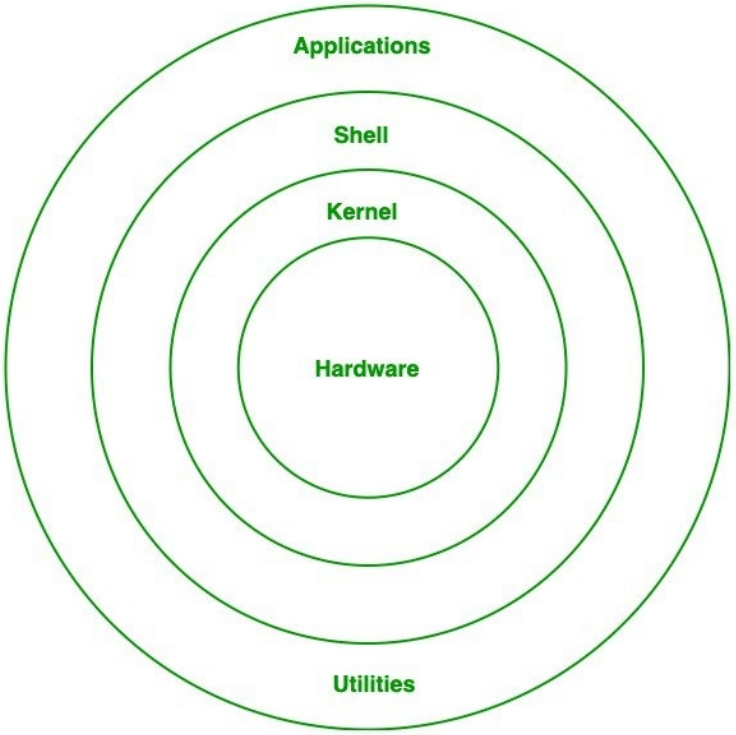
Linux Commands & Packages



Remind of OS



Kernel: 시스템의 보안을 담당하고,
자원관리(CPU process sceduling)를 한다.

Shell: User가 Kernal을 다루는 method =>
Kernel을 감싸고 있는 Shell 을 통해
Kernal에 명령을 내린다.

Shell의 명령어를 통해 실행.
이 명령어들은 bin file에 존재한다.

Environment Variable and PATH

```
sparcs@6ff112b5ad7a:~/ball$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
sparcs@6ff112b5ad7a:~/ball$ echo $HOME
/home/sparcs
sparcs@6ff112b5ad7a:~/ball$ echo $PWD
/home/sparcs/ball
sparcs@6ff112b5ad7a:~/ball$
```

Environment Variable: 컴퓨터의 프로세스의 동작에 영향을 끼치는 변수들의 모임

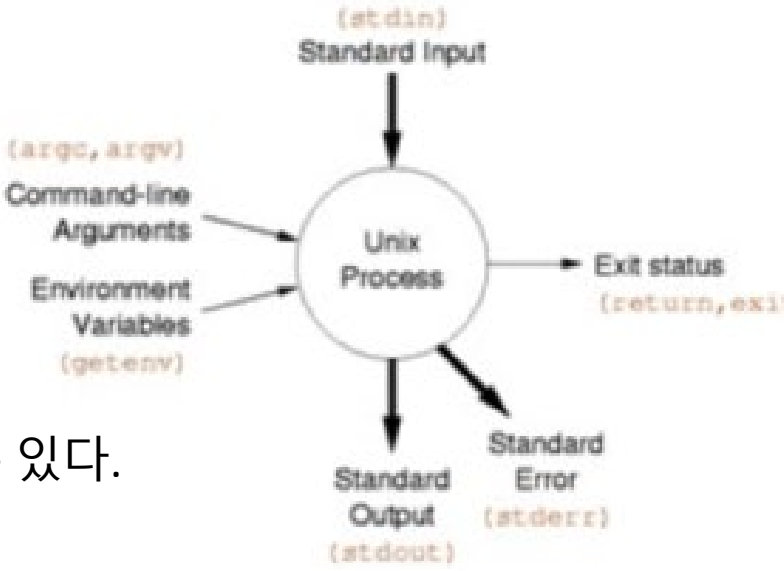
$PATH는 shell의 명령어들을 담고 있는 binary file 들의 디렉토리 주소를 담고 있다.
여러 주소들은 :로 구분되어 있다.
$ pwd 명령을 실행하려고 한다고 하자. -> Shell 에서 명령어를 Search한다.
/usr/local/sbin # 여기에 pwd binary file이 존재하면, 실행! 없으면 다음!
/usr/local/bin ## 여기에 pwd binary file이 존재하면, 실행! 없으면 다음!
/usr/sbin ### 여기에 pwd binary file이 존재하면, 실행! 없으면 다음!
…

$HOME은 홈 디렉토리의 주소를 나타낸다. cd ~ 를 shell 에 입력하면,
홈 디렉토리로 이동한다.
$PWD 는 현재의 디렉토리의 주소를 나타낸다.

Linux Shell Command – Basic

man [command] : [command]에 대한 매뉴얼 출력
[command] –help: [command]에 대한 설명을 출력.

which [command] : [command]의 절대 경로 출력
⇒ $PATH의 경로들 중 하나에 존재할 것이다.



echo [text]: standard output으로 [text] 출력
=>환경변수를 확인하거나, 간단히 파일을 편집할 수 있다.
touch hello.txt ; echo ‘hi, my name is ball’ > hello.txt

ctrl+c: command를 탈출하여 shell로 복귀: 오랜 시간 아무것도 출력되지 않으면 사용된다.

File/Directory Commands

pwd: print working directory: 현재의 디렉토리를 출력한다.

ls [dir]: list: [dir]에 있는 파일과 디렉토리의 리스트를 출력한다.
[dir]가 입력되지 않은 경우, default로 현재의 디렉토리의 ls를 출력한다.
⇒ ls [dir]-l: 디렉토리의 리스트를 깔끔하게 출력한다.(Permission을 포함)
⇒ ls [dir] –al: 디렉토리에 숨겨진 파일까지 ls –l 방법으로 출력한다.

cd [dir]: [dir]로 이동한다.
⇒ cd .. : 상위 디렉토리로 이동한다.
⇒ cd ~ : 홈 디렉토리($HOME)로 이동한다.

touch [file]: [file]의 변경날짜를 현재로 업데이트,
만약 현 디렉토리에 [file]이 존재하지 않은 경우, [file]을 생성한다.

File/Directory Commands

```
jinhochoi@DESKTOP-L2ACJJD:~/test$ pwd
/home/jinhochoi/test
jinhochoi@DESKTOP-L2ACJJD:~/test$ ls
a.log  b.log  c.text  hello.txt
jinhochoi@DESKTOP-L2ACJJD:~/test$ ls -l
total 0
-rw-rw-rw- 1 jinhochoi jinhochoi  0 Jul 11 18:52 a.log
-rw-rw-rw- 1 jinhochoi jinhochoi  0 Jul 11 18:52 b.log
-rw-rw-rw- 1 jinhochoi jinhochoi  0 Jul 11 18:52 c.text
-rw-rw-rw- 1 jinhochoi jinhochoi 19 Jul 11 18:40 hello.txt
jinhochoi@DESKTOP-L2ACJJD:~/test$ ls -al
total 0
drwxrwxrwx 1 jinhochoi jinhochoi 4096 Jul 11 18:52 .
drwxr-xr-x 1 jinhochoi jinhochoi 4096 Jul 11 18:40 ..
-rw-rw-rw- 1 jinhochoi jinhochoi    0 Jul 11 18:52 a.log
-rw-rw-rw- 1 jinhochoi jinhochoi    0 Jul 11 18:52 b.log
-rw-rw-rw- 1 jinhochoi jinhochoi    0 Jul 11 18:52 c.text
-rw-rw-rw- 1 jinhochoi jinhochoi   19 Jul 11 18:40 hello.txt
jinhochoi@DESKTOP-L2ACJJD:~/test$ cd ..
jinhochoi@DESKTOP-L2ACJJD:~$ touch ball.txt
jinhochoi@DESKTOP-L2ACJJD:~$ ls
Helloworld.html  date.log     hello.txt      perm
ball.txt         hellllllo.tt  hi-machine.sh  test
bellllo.txt      hello.html    lecture.txt
jinhochoi@DESKTOP-L2ACJJD:~$
```

File/Directory Commands

mv [dir1] [dir2] [file]: move : mv의 대표적인 사용법으로는 2가지가 있다.
파일의 이름을 변경하는 것과, 파일을 다른 디렉토리로 옮기는 것.
⇒ mv [file1] [file2] : 현재 디렉토리에서 [file1]의 이름을 [file2]로 바꾼다.
⇒ mv [file] [dir] : [file]의 디렉토리를 [dir]로 옮긴다.

cp [file1] [dir] : copy: 현 디렉토리에 있는 [file1]을 [dir]로 copy한다.

rm [option]: 파일을 삭제한다.
⇒ rm [file] : 파일을 삭제한다.
⇒ rm –r [dir] : 디렉토리를 재귀적으로 삭제한다.(다시말해, 디렉토리 내부의 디렉토리, 파일들을 남김없이 삭제한다.)
⇒ rm –f: 경고없이 삭제한다.
⇒ rm –rf : 절대로 하지말자.

File/Directory Commands

mkdir [dir] : 현재 디렉토리에 [dir]의 디렉토리를 생성한다.
⇒ mkdir –p [dir1]/[dir2]/[dir3]… : 현 디렉토리에 [dir1]생성. 이미 존재하거나 새로 생성한 경우, cd [dir1]해서 [dir2] 생성. 이미 존재하거나 새로 생성한 경우, cd [dir2]해서 [dir3] 생성… 다시 말해, 부모 디렉토리가 존재하지 않은 경우, 생성한다.

du [file] : 파일/디렉토리의 용량을 출력한다.
⇒ du –s : 해당 디렉토리만의 용량을 출력한다.
⇒ du –h : 사람이 읽기 편하게 써준다.

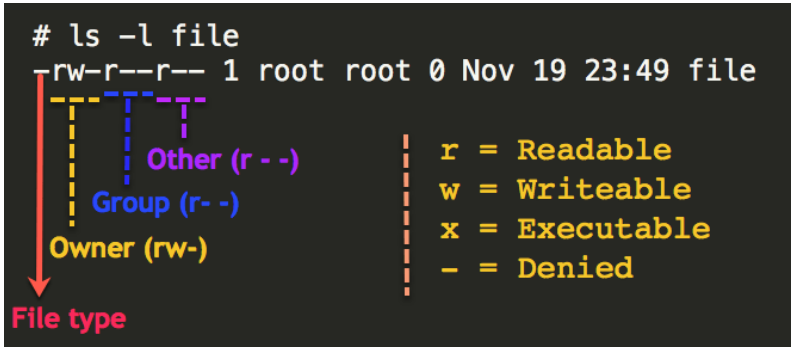
```
jinhochoi@DESKTOP-2BUMP27:~$ ls -l
total 0
drwxrwxrwx 1 jinhochoi jinhochoi 512 Jul 12 00:35 bak
-rw-rw-rw- 1 jinhochoi jinhochoi   0 Jul 12 00:39 hellllllo.txt
drwxrwxrwx 1 jinhochoi jinhochoi 512 Jul 12 00:45 hello
drwxrwxrwx 1 jinhochoi jinhochoi 512 Jul  8 18:46 script
jinhochoi@DESKTOP-2BUMP27:~$ du
0       ./.config/htop
0       ./.config/procps
0       ./.config
0       ./.elinks
0       ./.landscape
0       ./.local/share/nano
0       ./.local/share
0       ./.local
0       ./.ssh
0       ./bak
0       ./hello/pew/pow
0       ./hello/pew
0       ./hello
0       ./script/bak
0       ./script
80      .
jinhochoi@DESKTOP-2BUMP27:~$ du -s bak
0       bak
```

```
jinhochoi@DESKTOP-2BUMP27:~$ du -h
0       ./.config/htop
0       ./.config/procps
0       ./.config
0       ./.elinks
0       ./.landscape
0       ./.local/share/nano
0       ./.local/share
0       ./.local
0       ./.ssh
0       ./bak
0       ./hello/pew/pow
0       ./hello/pew
0       ./hello
0       ./script/bak
0       ./script
80K     .
jinhochoi@DESKTOP-2BUMP27:~$ ls
bak  hellllllo.txt  hello  script
jinhochoi@DESKTOP-2BUMP27:~$
```

File/Directory Permission Commands

리눅스에서 파일/디렉토리에 READ, WRITE, EXECUTE 하기 위해서는 권한이 필요하다.
파일 혹은 디렉토리의 Permission을 확인하기 위해서는 ls -al을 해보면 된다.

```
jinhochoi@DESKTOP-2BUMP27:~$ ls -al
total 80
drwxr-xr-x 1 jinhochoi jinhochoi   512 Jul 12 00:45 .
drwxr-xr-x 1 root      root        512 Jul  8 22:32 ..
-rw------- 1 jinhochoi jinhochoi  5768 Jul 11 02:56 .bash_history
-rw-r--r-- 1 jinhochoi jinhochoi   220 Jun  1 00:41 .bash_logout
-rw-r--r-- 1 jinhochoi jinhochoi  3771 Jun  1 00:41 .bashrc
drwx------ 1 jinhochoi jinhochoi   512 Jul  8 17:56 .config
drwx------ 1 jinhochoi jinhochoi   512 Jul 11 02:14 .elinks
drwxr-xr-x 1 jinhochoi jinhochoi   512 Jun  1 00:42 .landscape
-rw------- 1 jinhochoi jinhochoi    34 Jul  7 21:16 .lesshst
drwxrwxrwx 1 jinhochoi jinhochoi   512 Jul  8 17:29 .local
```

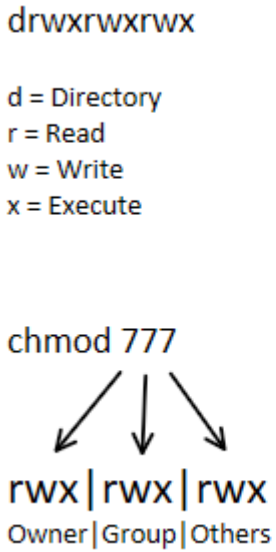


[type] rwx rwx rwx [owner] [group] [size] 의 형태로 Permission이 표시되어 있다.
⇒ [type]은 파일 혹은 디렉토리의 type을 나타낸다. file은 – 로 표기. dir 은 d 로 표기.
⇒ rwx rwx rwx 은 각각 owner, group, other의 Permission을 표기한 것이다.
⇒ r(READ): 파일 혹은 디렉토리를 읽을 수 있는 권한
⇒ w(WRITE): 파일 혹은 디렉토리를 쓸 수 있는 권한
⇒ x(EXECUTE): 파일 혹은 디렉토리를 실행할 수 있는 권한

File/Directory Permission Commands

chmod [privilege] [file or dir] : change file mode: [file or dir]의 접근 권한을 변경.
⇒ [privilege]: [privilege]를 표기할 수 있는 방법은 다양하다.
⇒ 1번째 방법은 o+x : other에 execute권한 부여 // o-x: other에 execute 권한 없애기
u+x: user(파일을 만든 사람)에 execute 권한 부여 // g+x: group에 execute 권한 부여

```
jinhochoi@DESKTOP-2BUMP27:~$ ls -l
total 0
drwxrwxrwx 1 jinhochoi jinhochoi 512 Jul 12 00:35 bak
-rw-rw-rw- 1 jinhochoi jinhochoi   0 Jul 12 00:39 hellllllo.txt
drwxrwxrwx 1 jinhochoi jinhochoi 512 Jul 12 00:45 hello
drwxrwxrwx 1 jinhochoi jinhochoi 512 Jul  8 18:46 script
jinhochoi@DESKTOP-2BUMP27:~$ chmod o-x hello
jinhochoi@DESKTOP-2BUMP27:~$ ls -l
total 0
drwxrwxrwx 1 jinhochoi jinhochoi 512 Jul 12 00:35 bak
-rw-rw-rw- 1 jinhochoi jinhochoi   0 Jul 12 00:39 hellllllo.txt
drwxrwxrw- 1 jinhochoi jinhochoi 512 Jul 12 00:45 hello
drwxrwxrwx 1 jinhochoi jinhochoi 512 Jul  8 18:46 script
```

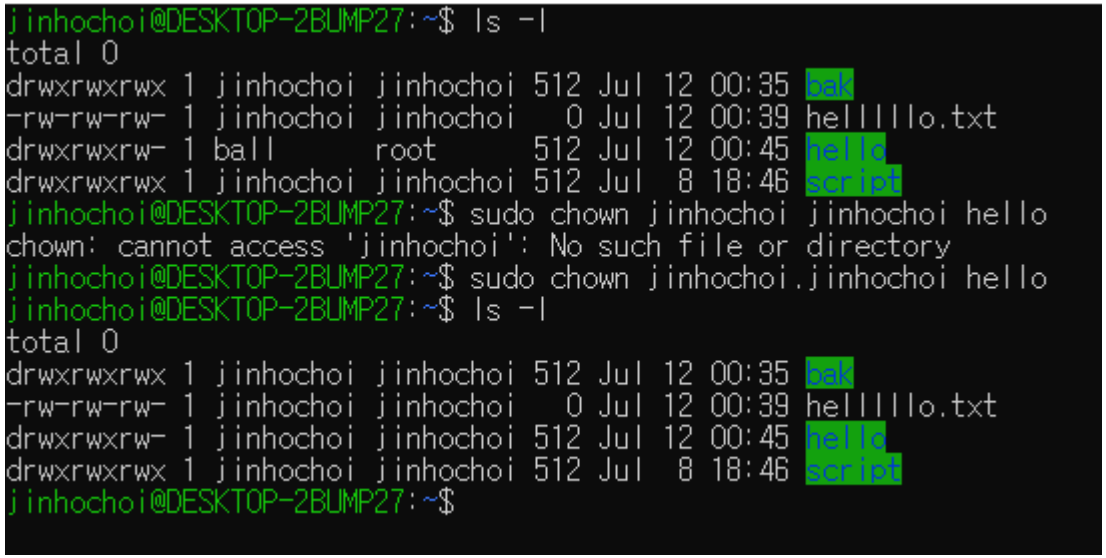


7	rwx	111
6	rw-	110
5	r-x	101
4	r--	100
3	-wx	011
2	-w-	010
1	--x	001
0	---	000

⇒ 2번째 방법은 세자리 8진수 숫자를 통해 user, group, other의 접근권한을 설정하는 것이다.

File/Directory Permission Commands

chown [user].[group] [file or dir] : change owner: [file or dir]의 소유자를 변경한다.
⇒ [group]을 안써주면, default는 기존의 group이다.



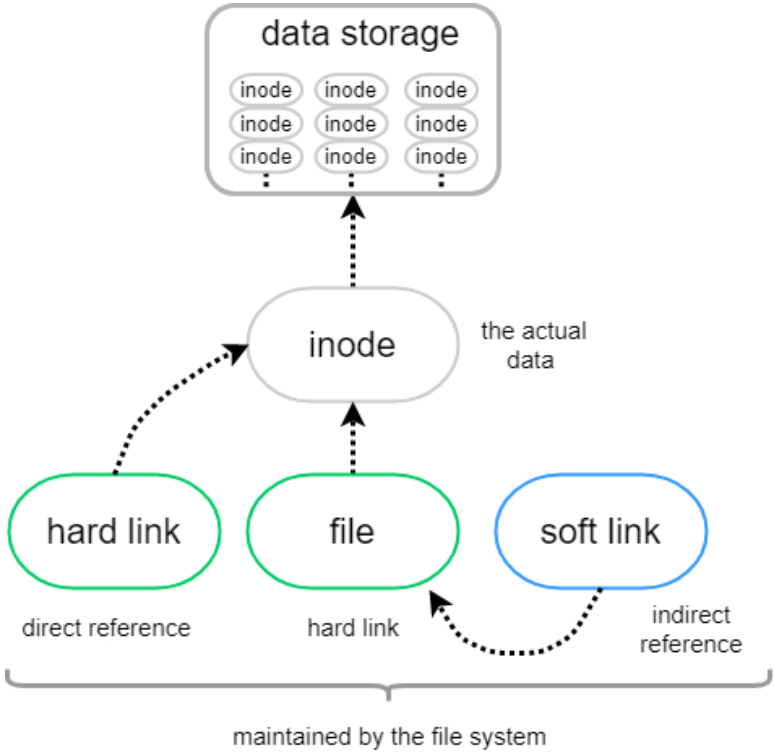
**chmod, chown에서 dir의 Permission을 수정할 때, -R(Recursive)을 사용하여, 재귀적으로 Permission을 수정하는 경우가 자주 발생한다.

File/Directory Commands

ln [target] [name] : link: [target] 파일로 [name]의 이름을 가진 바로가기를 생성한다.
⇒ -s :symbolic : 심볼릭 링크/소프트 링크를 생성한다.
⇒ Inode: index node로, 간단히 말해서 디스크 상에서, 파일의 정보가 담겨진 곳의 index
⇒ Hard link vs Soft link
⇒ Soft link는 inode로 매핑되는 file을 가르키는 링크이다.
⇒ Hard link는 indode로 직접 매핑되는 링크이다.

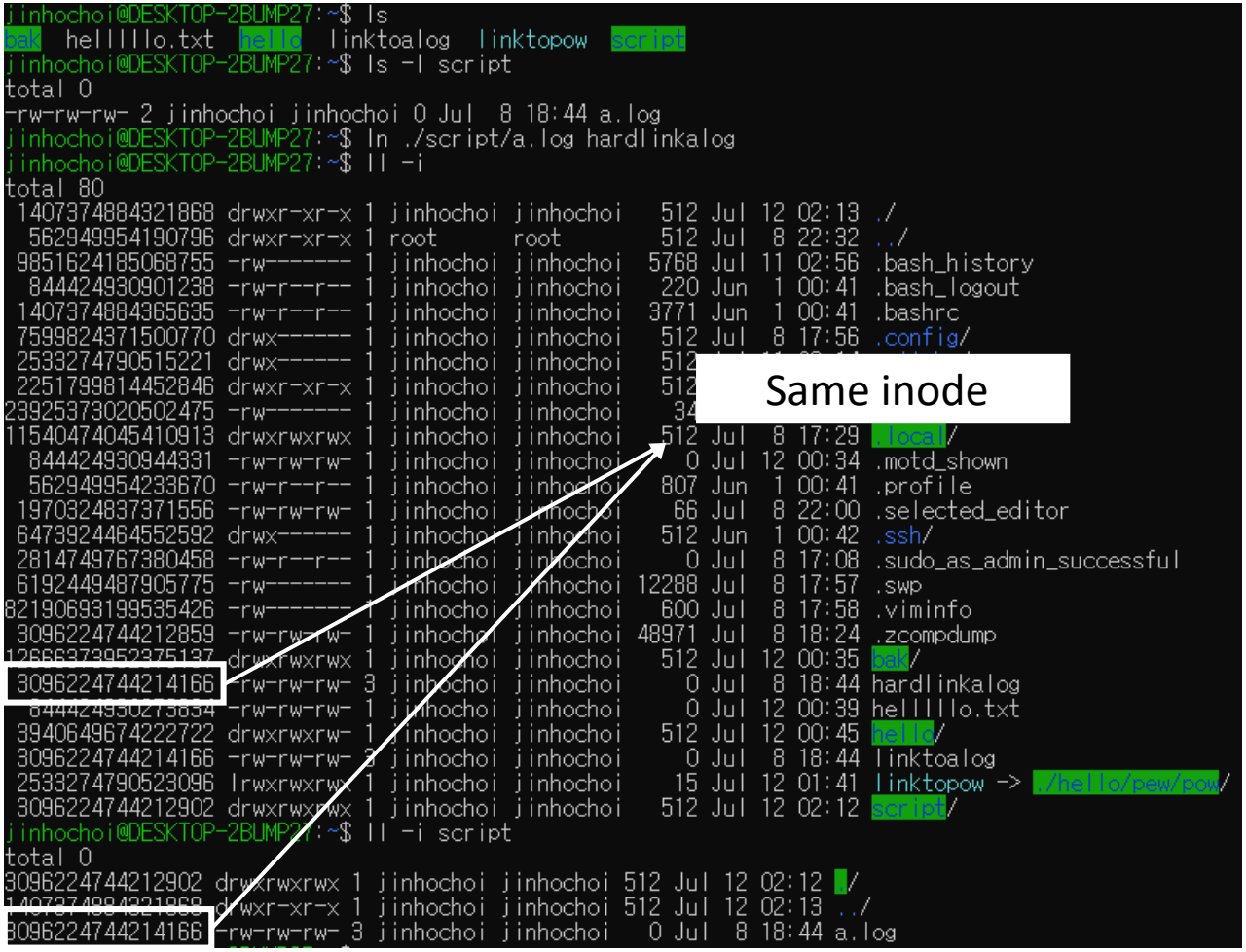
```
jinhochoi@DESKTOP-2BUMP27:~$ ls -l
total 0
drwxrwxrwx 1 jinhochoi jinhochoi 512 Jul 12 00:35 bak
-rw-rw-rw- 1 jinhochoi jinhochoi   0 Jul 12 00:39 hellllllo.txt
drwxrwxrw- 1 jinhochoi jinhochoi 512 Jul 12 00:45 hello
drwxrwxrwx 1 jinhochoi jinhochoi 512 Jul  8 18:46 script
jinhochoi@DESKTOP-2BUMP27:~$ ln ./hello/pew/pow linktopow
ln: ./hello/pew/pow: hard link not allowed for directory
jinhochoi@DESKTOP-2BUMP27:~$ ln -s ./hello/pew/pow linktopow
jinhochoi@DESKTOP-2BUMP27:~$ ls -l
total 0
drwxrwxrwx 1 jinhochoi jinhochoi 512 Jul 12 00:35 bak
-rw-rw-rw- 1 jinhochoi jinhochoi   0 Jul 12 00:39 hellllllo.txt
drwxrwxrw- 1 jinhochoi jinhochoi 512 Jul 12 00:45 hello
lrwxrwxrwx 1 jinhochoi jinhochoi  15 Jul 12 01:41 linktopow -> ./hello/pew/pow
drwxrwxrwx 1 jinhochoi jinhochoi 512 Jul  8 18:46 script
jinhochoi@DESKTOP-2BUMP27:~$
```

Soft link의 예시



File/Directory Commands

ll –i : 파일/디렉토리의 inode를 확인하는 명령어이다.



Hard link의 예시

File/Directory Commands

Hard link는 잘 사용되지 않으며, soft link가 주로 사용된다.
명령어를 버전별로 관리하거나, 목적을 지정하기에 용이하다.

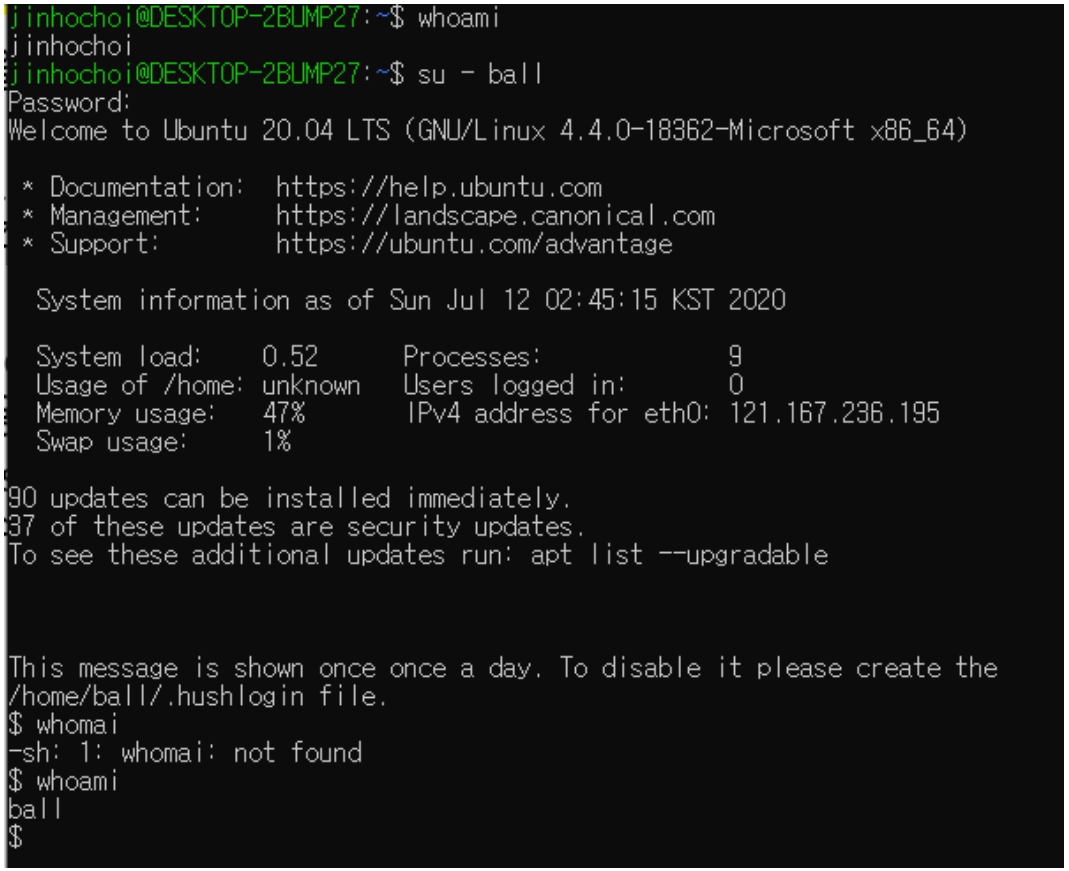
```
lrwxrwxrwx 1 root root 16 Jul  5 17:49 K89iscsid -> ../init.d/iscsid
lrwxrwxrwx 1 root root 18 Jul  5 17:48 K89netplugd -> ../init.d/netplugd
lrwxrwxrwx 1 root root 14 Jul  5 17:50 K89pand -> ../init.d/pand
lrwxrwxrwx 1 root root 15 Jul  5 17:47 K89rdisc -> ../init.d/rdisc
lrwxrwxrwx 1 root root 19 Jul  5 17:50 K90bluetooth -> ../init.d/bluetooth
lrwxrwxrwx 1 root root 17 Jul  5 17:49 K90network -> ../init.d/network
lrwxrwxrwx 1 root root 14 Jul  5 17:52 K91capi -> ../init.d/capi
lrwxrwxrwx 1 root root 14 Jul  5 17:52 K91isdn -> ../init.d/isdn
lrwxrwxrwx 1 root root 19 Jul  5 17:47 K92ip6tables -> ../init.d/ip6tables
lrwxrwxrwx 1 root root 18 Jul  5 17:47 K92iptables -> ../init.d/iptables
lrwxrwxrwx 1 root root 19 Jul  5 17:52 K95firstboot -> ../init.d/firstboot
lrwxrwxrwx 1 root root 15 Jul  5 17:52 K95kudzu -> ../init.d/kudzu
lrwxrwxrwx 1 root root 18 Jul  5 17:47 K99cpuspeed -> ../init.d/cpuspeed
lrwxrwxrwx 1 root root 23 Jul  5 17:50 K99microcode_ctl -> ../init.d/microcode_c
tl
lrwxrwxrwx 1 root root 25 Jul  5 17:50 K99readahead_early -> ../init.d/readahead
_early
lrwxrwxrwx 1 root root 25 Jul  5 17:50 K99readahead_later -> ../init.d/readahead
_later
lrwxrwxrwx 1 root root 22 Jul  7 17:26 K99vmware-tools -> ../init.d/vmware-tools
lrwxrwxrwx 1 root root 17 Jul  5 17:49 S00killall -> ../init.d/killall
lrwxrwxrwx 1 root root 14 Jul  5 17:49 S01halt -> ../init.d/halt
[root@localhost rc0.d]# ls -l /usr/bin/locate
-rwx--s--x 1 root slocate 23856 Sep  4  2009 /usr/bin/locate
[root@localhost rc0.d]# _
```

목적별로 소프트링크를 지정해준 예시

User / Privilege Commands

whami: 현재 shell 을 사용하고 잇는 유저의 이름을 출력

su – [user] : [user]로 사용자 전환
⇒ [user]를 지정해주지 않은 경우, default로 root 사용자로 전환



User / Privilege Commands

sudo [command]: root 사용자의 권한으로 command를 실행한다.
⇒ sudo를 모든 사용자가 막 사용할 수는 없다.
(막 사용하면 permission을 지정해준 이유가 없으므로…)
⇒ sudo를 사용할 수 있는 user(sudoers)가 되기 위해서는,
/etc/sudoers 파일에 user를 추가해야 한다. or sudo 그룹에 유저를 추가해야 한다.

```
jinhochoi@DESKTOP-2BUMP27:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.25.1-1ubuntu3).
The following package was automatically installed and is no longer required:
  zsh-common
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 91 not upgraded.
jinhochoi@DESKTOP-2BUMP27:~$
```

apt-get install 명령어는 root 의 권한이 필요한 명령어이다.
root 계정으로 전환하기보다, 이 command를 root의 권한으로 실행시키는 명령어인 sudo를 사용한다.

User / Privilege Commands

```
jinhochoi@DESKTOP-2BUMP27:~$ cd /etc
jinhochoi@DESKTOP-2BUMP27:/etc$ ls -l sudoers
-r--r----- 1 root root 755 Feb  3 23:32 sudoers
jinhochoi@DESKTOP-2BUMP27:/etc$ cat sudoers
cat: sudoers: Permission denied
jinhochoi@DESKTOP-2BUMP27:/etc$ sudo cat sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
jinhochoi@DESKTOP-2BUMP27:/etc$
```

/etc/sudoers 의 file은 다음과 같다.
일반 user를 sudoer로 만들기 위해서는 이 파일을 변경해야 한다.
https://sseungshin.tistory.com/82 를 참고하자.

User / Privilege Commands

passwd : password: User의 비밀번호를 변경하고자 할 때 사용된다.

adduser [name]: [name]의 이름을 가진 user를 추가한다.
⇒ adduser [name] [group] : [name]의 이름을 가진 user를 [group]에 추가한다.

deluser [name] : [name]을 가진 user를 삭제한다.

usermod [name] : [name] user 정보를 편집한다.

종종 위 세 command는 $PATH에 존재하지 않는 경우가 있다.
=> 보통 /usr/sbin 등에 존재한다. 따라서 $PATH에 경로를 추가하면 된다.

Utility Commands

cat [file] : [file]의 내용을 shell에 출력한다.
⇒ head [file]: 파일 시작부터 10줄을 출력한다.
⇒ tail [file]: 파일의 끝 10줄을 출력한다.
⇒ tail –f [file]: 파일을 계속 관찰하고, 파일의 끝에 무언가가 추가될때마다 출력한다.

wc [file] : word count: [file]의 줄 수, 단어 수, 바이트 수를 출력한다.

less [file] : [file]의 내용을 깔끔하게 출력한다. (vim 조작법과 유사하다.)

nano [file]: [file]을 편집하는 CLI 에디터를 실행한다. 조작법은 모두 화면에 표시된다.
⇒ 조작법이 모두 표시되어 다루기 쉽다.

tar [target] : 압축/압축해제를 한다.
⇒ tar –c [source] [target] : 압축한다
⇒ tar –x [target] : 압축을 푼다.
⇒ tar –v [target] : 진행상황을 출력한다.
⇒ tar –z [target] : gzip으로 압축/해제
⇒ tar –f [target] : 압축/해제의 결과를 파일로 출력한다.
⇒ ** 정리를 하자면, tar –cvzf [source] [target]으로 압축, tar –xvzf [target]로 압축해제!!

Utility Commands

find [dir] –name [file] : [dir]에서 [file] 검색.
⇒ [dir] 이 없는 경우, default값은 현재 디렉토리이다.

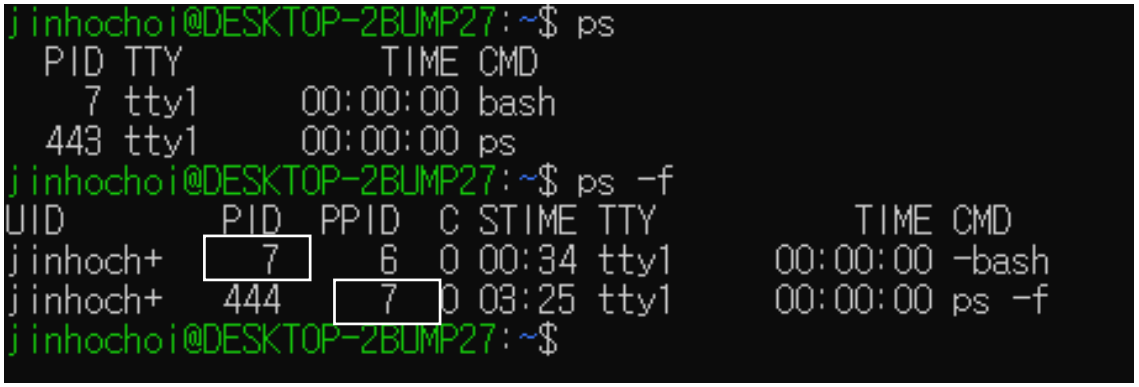
grep [regex] [file]: regex(regular expression)에 해당하는 줄을 파일에서 찾아 출력한다.
⇒ grep –i : 대소문자 구분을 하지 않는다.
⇒ grep –n : 파일에서 줄 번호도 출력한다.
⇒ grep -r : 재귀적으로 하위 파일을 탐색한다.

```
jinhochoi@DESKTOP-2BUMP27:/usr/bin$ ls -l | grep python
lrwxrwxrwx 1 root   root          23 Mar 13 19:14 pdb3.8 -> ../lib/python3.8/pdb.py
lrwxrwxrwx 1 root   root          31 Mar 13 21:20 py3versions -> ../share/python3/py3versions.py
lrwxrwxrwx 1 root   root           9 Mar 13 21:20 python3 -> python3.8
-rwxr-xr-x 1 root   root     5457568 Mar 13 19:14 python3.8
jinhochoi@DESKTOP-2BUMP27:/usr/bin$
```

여기서는 [regex] 가 pytho이고, [file]이 ls –l의 결과값이다. 따라서, 이 command는 파일/디렉토리 리스트에서 python이 포함된 파일/디렉토리를 출력한다.
여기서 사용한 pipeline은 조금 뒤에서 설명할 것이다.

Process Commands

ps : process: 현재 실행되고 있는 사용자의 프로세스를 출력한다.
⇒ ps –e : 모든 프로세스 출력
⇒ ps –f: 자세히 출력
⇒ PID : 프로세스의 ID. 프로세스에게 부여된 고유의 번호
⇒ PPID: 프로세스를 실행한 부모 프로세스의 PID



Process Commands

fg [process] : foreground: [process]를 foreground로 옮긴다.
=> user 간섭을 해야 하는 프로세스를 주로 foreground 에 놓는다.
bg [process] : background : [process]를 background로 옮긴다.
⇒ user 간섭 없이 실행되게 하고 싶은 프로세스를 background에 놓는다.
예를 들어, 백업을 하는 프로세스는 background에 놓는게 좋다.

[command] & : [command]를 background에서 실행한다.
다시 말해, background 프로세스를 실행한다.

nohup [command] : [command]의 stdout를 ./nohup.out 파일로
리다이렉션 후 shell과 독립적으로 실행한다.
(shell과 독립적으로 실행 = shell 이 꺼져도, 계속 process 는 실행된다.)
이러한 process 종료 방법: ps –ef | grep [file] 로 process PID를 알아낸 뒤, kill -9 PID

nohup [command] & : [command]를 background에서 shell과 독립적으로 실행한다.

Ctrl + Z : 현재 shell을 점유하고 있는 process를 background로 보낸다.

jobs : background process를 모두 출력한다.

Process Commands

Network Commands

ssh [user]@[address] : secure shell : 원격 서버컴퓨터에 ssh 포트로 접속.
⇒ ssh 포트로 접속하면 안전하다.'
⇒ [user] 생략 시, 현재 장치의 user명으로 접속한다.
⇒ -p [port] : 지정한 [port]로 ssh 접속: 서버 컴퓨터의 ssh 포트가 22가 아닌 경우에 사용
(현재, old sparcs 서버의 22번 포트에 문제가 있어서 ssh 포트로 다른 포트를 사용 중이다.)

scp [file] [user]@[address]:[path-to-file] : [file]을 원격 서버컴퓨터 [address]의 [path]에 [user] 권한으로 복사(로컬에서 원격 서버컴퓨터로 파일 업로드)

scp [user]@[address]:[path-to-file] [path]: 현재 장치의 [path]에 [user] 권한으로 [address]의 [path-to-file] 의 파일을 복사한다.(원격 서버컴퓨터의 파일을 로컬에 다운로드한다.)
⇒ [user]@ 생략시, 현재 장치의 user 명으로 접속한다.
⇒ -P [port]: 지정한 [port]를 사용한다. ([port]는 원격 서버컴퓨터의 ssh 포트 넘버)
⇒ -R: 재귀적으로 복사한다.

사실, scp를 사용하지 않더라도, file(코드파일)을 서버에 올리는 방법이 있다.
=> git ~!~!~!~! 하지만, 서버컴퓨터에 git 이 설치되 있어야 한다는 전제하에…

Network Commands

wget [http-address] : web get : [http-address] 파일을 현재 디렉토리에
같은 이름으로 다운로드 한다.

ping [address] : [address]로 ping을 전송하여 되돌아오는 시간을 반복적으로 측정한다.
⇒ 보통, DNS 설정이 이상한 경우 ping test를 실시한다.

ifconfig : 장치의 네트워크 연결정보를 출력한다.

Package & Service Management Commands

리눅스 배포판들은 응용 프로그램들(ex. git, nodejs, …) 을 관리하기 위해
package manager 명령어를 갖고 있다.
⇒ Debian 계열(Ubuntu): dpkg, apt, apt-get
⇒ Redhat 계열(CentOS): rpm, yum

apt: (advanced package tool): Ubuntu 에서 사용하는 package manager
=> apt update: package 목록을 업데이트: 다운로드 하기전에, 습관적으로 하자.
⇒ apt install [package]: [package]와 dependency 를 설치
⇒ apt remove [package]: [package]와 dependency를 제거
⇒ apt purge [package] : [package]와 dependency, 환경설정 파일까지 삭제한다.
⇒ apt search [regex]: [regex]와 매칭되는 package 을 검색하여 출력.

```
jinhochoi@DESKTOP-2BUMP27:~/hello$ rpm --version

Command 'rpm' not found, but can be installed with:

sudo apt install rpm

jinhochoi@DESKTOP-2BUMP27:~/hello$ dpkg --version
Debian 'dpkg' package management program version 1.19.7 (amd64).
This is free software; see the GNU General Public License version 2 or
later for copying conditions. There is NO warranty.
```

Ubuntu20:04 LTS에는 dpkg, apt, apt-get이 내장되어 있으며,
rpm, yum은 default로 설치되어 있지 않다.

Package & Service Management Commands

Daemon: 시스템에 상주하며, 특정 상태(주로 부팅)되면 자동으로 동작하는 프로세스
⇒ Shell이 꺼지든, 말든 상관없이 동작하는 background process를 Daemon이라 한다.
⇒ 지속적인 서비스 제공을 위해 OS에서 직접 관리한다.
⇒ 부팅 시 작동해야 하는 Daemon들은 /etc/init.d에 정의되어 있다.
⇒ Daemon으로는 sshd, httpd, apache2, mysqld 등이 있다.

systemctl 또는 service: system control: Daemon을 관리하는 명령어
⇒ service [daemon] start : start daemon
⇒ service [daemon] stop : stop daemon
⇒ service [daemon] restart : restart daemon

systemctl 은 command가 살짝 다르다.
https://galid1.tistory.com/35 참고.

Shell Utilities

[process1]|[process2] : pipeline: [process1]의 stdout를 [process2]의 stdin으로 연결한다.
=> 함수로 생각하면 편하다. process의 output을 process2의 input으로 받는다고 생각하자.

```
jinhochoi@DESKTOP-2BUMP27:/etc$ ls -al | grep python
drwxr-xr-x 1 root root        512 Apr 23 15:41 python3
drwxr-xr-x 1 root root        512 Apr 23 15:40 python3.8
```

Redirection: stdin/stdout/stderr을 특정 파일로 redirec한다.

⇒ [command] > [file] : [command]의 stdout이 [file]의 stdin으로

⇒ [command < [file] : [file]이 [command]의 stdin으로

⇒ [command] 2> [file] : [command] 의 stderr이 [file]으로

⇒ ** Redirection을 통해 user 로그를 남길 수 있다.

```
jinhochoi@DESKTOP-2BUMP27:~$ ls
bak  hardlinkalog  hellllllo.txt  hello  linktoalog  linktopow  script
jinhochoi@DESKTOP-2BUMP27:~$ echo "hello, my name is ball. I'm going to talk about Redirection." > hello.txt
jinhochoi@DESKTOP-2BUMP27:~$ cat hello.txt
hello, my name is ball. I'm going to talk about Redirection.
jinhochoi@DESKTOP-2BUMP27:~$
```

```
jinhochoi@DESKTOP-2BUMP27:~$ ls
bak  hardlinkalog  hellllllo.txt  hello  hello.txt  linktoalog  linktopow  script
jinhochoi@DESKTOP-2BUMP27:~$ rm hello 2> error.txt
jinhochoi@DESKTOP-2BUMP27:~$ cat error.txt
rm: cannot remove 'hello': Is a directory
jinhochoi@DESKTOP-2BUMP27:~$
```

실습 준비 – Git hub repo

이번 실습은 github을 이용하겠습니다.

1. 디렉토리를 하나 생성하고, git init을 합니다.(Git이 없다면, sudo apt-get install git)
2. echo “hello, my name is <ball>.” > hello.txt를 하여 textfile 하나를 생성합니다.
3. git add .
4. git status
5. git commit –m “commit initial file.”

6. github에 가서 WheelSeminar repo를 한 개 파자.
7. bash 에 git remote add origin <remote repo url>
8. git push origin master (만약, git push에서 “permission denied. public key” 에러가 뜬다면, https://git-scm.com/book/ko/v2/Git-%EC%84%9C%EB%B2%84-SSH-%EA%B3%B5%EA%B0%9C%ED%82%A4-%EB%A7%8C%EB%93%A4%EA%B8%B0 참고.)

이제 wheel seminar 용 github가 완성되었다.

실습 준비

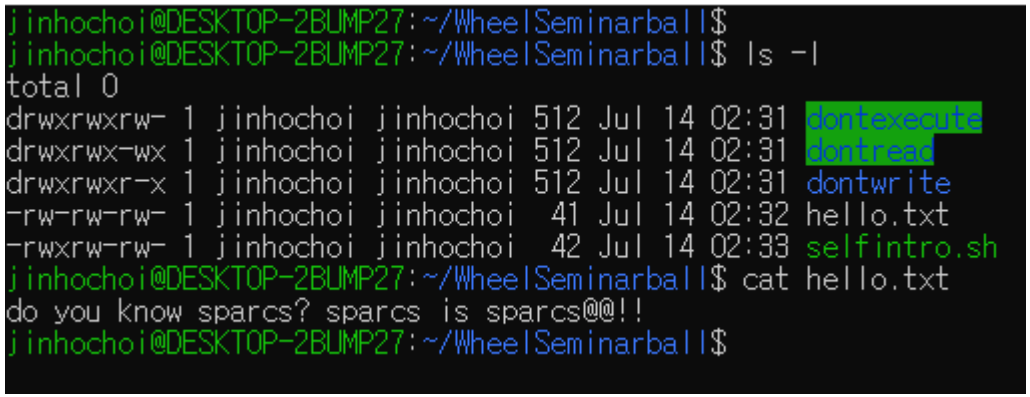
실습하기에 앞서, 먼저 설명드릴 것이 있습니다. 오늘의 실습은 container에 scp를 통해 제출하는 것이 목표입니다.(sparcs whale container)을 이용할 것입니다. 그래서 실습 제출을 위해서는 쌀 서버에 접근할 수 있는 private key 가 필요합니다.
(저는 old sparcs server에 있습니다.)
혹시 이게 없으신 분들은 git repo를 파셔서, 나중에 git repo의 링크를 제출해주시면 감사하겠습니다.

그럼 우선, 쌀 서버 private key가 있는 디렉토리로 이동합니다. 그리고
$ mkdir WheelSeminar<name> 으로 디렉토리를 만듭니다.
$ cd WheelSeminar<name>

```
jinhochoi@DESKTOP-2BUMP27:~$ ls -l |grep Wheel
drwxrwxrwx 1 jinhochoi jinhochoi  512 Jul 14 02:40 WheelSeminarball
jinhochoi@DESKTOP-2BUMP27:~$ ls -l |grep priv
-rw------- 1 jinhochoi jinhochoi 1766 Jul 14 03:01 ubuntu_sparcs_privkey
jinhochoi@DESKTOP-2BUMP27:~$ cd WheelSeminarball/
jinhochoi@DESKTOP-2BUMP27:~/WheelSeminarball$
```

실습

1. 3가지 디렉토리를 만들어줍니다. dontread/ dontwrite/ dontexecute/
2. 3가지 디렉토리의 permission을 변경하여, other가 dontread 디렉토리에서는 read, dontwrite 디렉토리에서는 write, dontexecute 디렉토리에선는 execute 할 수 없도록 permission을 변경해줍니다. (힌트: chmod <privilege> <file or dir>)
3. CLI editor(nano or vim)를 사용하여 hello.txt의 내용을 다음과 같이 구성해줍니다. “do you know sparcs? sparcs is sparcs@@!!”



실습하기

1. 이번에는 “hello, my name is ball” 을 출력하는 프로그램을 만들 것입니다.
2. WheelSeminar<name> 디렉토리에서 nano(vim)로 selfintro.sh 파일을 생성해줍니다.
3. 그림과 같이 자기소개의 내용을 적어줍니다.
4. 마지막으로 프로그램을 실행시켜봅니다.(에러가 떠야 정상)
5. 왜 에러가 뜨냐면, user에게 execute할 권한이 없기 때문이다.

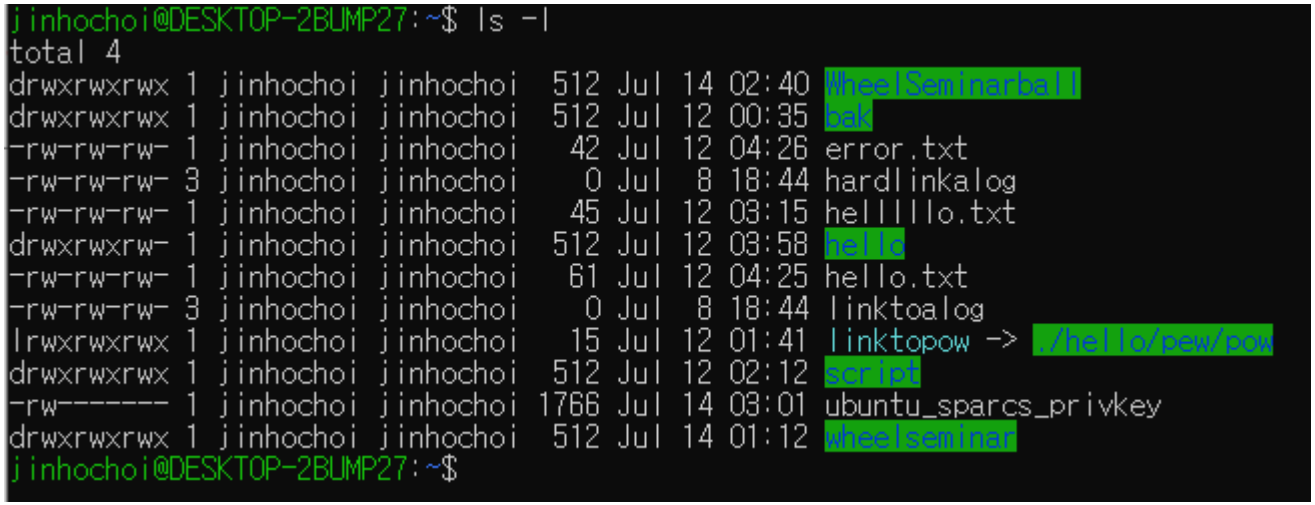
-> chmod 로 user에게 permission을 부여하자.

6. permission을 부여한 뒤, 실행하면 정상적으로 실행되는 것을 볼 수 있다.


```
jinhochoi@DESKTOP-2BUMP27:~/WheelSeminarball$ nano selfintro.sh
jinhochoi@DESKTOP-2BUMP27:~/WheelSeminarball$ ./selfintro.sh
-bash: ./selfintro.sh: Permission denied
jinhochoi@DESKTOP-2BUMP27:~/WheelSeminarball$ ls -l
total 0
drwxrwxrwx 1 jinhochoi jinhochoi 512 Jul 14 02:31 dontexecute
drwxrwx-wx 1 jinhochoi jinhochoi 512 Jul 14 02:31 dontread
drwxrwxrwx 1 jinhochoi jinhochoi 512 Jul 14 02:31 dontwrite
-rw-rw-rw- 1 jinhochoi jinhochoi  41 Jul 14 02:32 hello.txt
-rw-rw-rw- 1 jinhochoi jinhochoi  42 Jul 14 02:33 selfintro.sh
jinhochoi@DESKTOP-2BUMP27:~/WheelSeminarball$ /bin/bash selfintro.sh
hello my name is ball
jinhochoi@DESKTOP-2BUMP27:~/WheelSeminarball$ chmod u+x selfintro.sh
jinhochoi@DESKTOP-2BUMP27:~/WheelSeminarball$ ./selfintro.sh
hello my name is ball
jinhochoi@DESKTOP-2BUMP27:~/WheelSeminarball$
```

실습한 것 제출하기

이제 WheelSeminar<name> 파일을 ssh를 이용해 container에 제출하겠습니다
ssal server의 private key는 newbie project에서 한번쯤은 다뤄 봤을 것입니다.
ssal server의 private key의 위치가 어디있는지 잘 모르는 경우, git repo를 파서 제출하셔도 됩니다.
일단, 제출하시기 전에, $ls -l을 해주셔서 dontread, dontwrite, dontexecute의 디렉토리의 권한을 확인하시고, 캡처해주세요.(scp를 이용해서 파일을 전송하게 되면, permission이 살짝 변해서 캡처를 해주시면 감사하겠습니다.)
다음으로 이제 scp를 이용해 WheelSeminar<name> 디렉토리를 container에 보내도록 하겠습니다.
일단, ssal 서버의 privatekey가 있는 디렉토리로 이동합니다.



실습한 것 제출하기

다음으로, user가 sparcs이고, 도메인 이름(IP adress의 다른 표기법)가 ssal.sparcs.org 이고, port가 45001 인 도메인에 파일을 보내주도록 합시다.

```
jinhochoi@DESKTOP-2BUMP27:~$ ls
WheelSeminarball  error.txt     hellllllo.txt  hello.txt   linktopow  ubuntu_sparcs_privkey
bak               hardlinkalog  hello          linktoalog  script     wheelseminar
jinhochoi@DESKTOP-2BUMP27:~$ scp -p 45001 -i "./ubuntu_sparcs_privkey" ./WheelSeminarball/ sparcs@ssal.sparcs.org:~/
^Cjinhochoi@DESKTOP-2BUMP27:~$ scp -i "./ubuntu_sparcs_privkey" -p 45001 ./WheelSeminarball/ sparcs@ssal.sparcs.org:~/
^Cjinhochoi@DESKTOP-2BUMP27:~$ scp -i "./ubuntu_sparcs_privkey" -p 45001 ./WheelSeminarball sparcs@ssal.sparcs.org:~/
^Cjinhochoi@DESKTOP-2BUMP27:~$ scp -i "./ubuntu_sparcs_privkey" -P 45001 ./WheelSeminarball sparcs@ssal.sparcs.org:~/
Enter passphrase for key './ubuntu_sparcs_privkey':
Enter passphrase for key './ubuntu_sparcs_privkey':
./WheelSeminarball: not a regular file
jinhochoi@DESKTOP-2BUMP27:~$ scp -r -i "./ubuntu_sparcs_privkey" -P 45001 ./WheelSeminarball sparcs@ssal.sparcs.org:~/
Enter passphrase for key './ubuntu_sparcs_privkey':
hello.txt                                                                   100%   41     8.1KB/s   00:00
selfintro.sh                                                                100%   42     9.0KB/s   00:00
jinhochoi@DESKTOP-2BUMP27:~$
```

Container details

Image	sha256:bb30a76478c37e29addac61972b76785317912aace364d98b889c757e9b5ad83
Port configuration	22/tcp → 0.0.0.0:45001 3000/tcp → 0.0.0.0:40303 3001/tcp → 0.0.0.0:42004 80/tcp → 0.0.0.0:40042
CMD	`/usr/sbin/sshd -D`
ENV	PATH: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

실습한 것 제출하기

```
sparcs@a8830dda5e20:~$ ls -l
total 8
drwxrwxr-x 5 sparcs sparcs 4096 Jul 13 18:48 WheelSeminarball
drwxrwxr-x 2 sparcs sparcs 4096 Jul 13 17:52 hellllllo
sparcs@a8830dda5e20:~$ ls -l WheelSeminarball/
total 20
drwxrwxr-- 2 sparcs sparcs 4096 Jul 13 18:48 dontexecute
drwxrwx--x 2 sparcs sparcs 4096 Jul 13 18:48 dontread
drwxrwxr-x 2 sparcs sparcs 4096 Jul 13 18:48 dontwrite
-rw-rw-r-- 1 sparcs sparcs   41 Jul 13 18:48 hello.txt
-rwxrw-r-- 1 sparcs sparcs   42 Jul 13 18:48 selfintro.sh
sparcs@a8830dda5e20:~$
```

짠!

실습한 것 제출하기 – git ver.

만약, private key의 위치가 어딨는지 까먹은 경우, git 을 통해서 제출해주세요!
(쌀 서버는 중요하니, private key를 꼭 찾도록 합시다.)
만약, git을 통해 제출할 경우, slack에 깃 링크를 올려주세요.
그리고 scp를 이용해서 제출하셨던 분과 git을 이용해서 제출하셨던 분들은
dontread dontwrite dontexecute의 permissio을 캡처해주셔서($ls –l)
slack에 올려주시면 정말 감사하겠습니다!!

Q&A