💵

231023

사용자 전환

- su - : 하면 로그아웃하지 않고 사용자를 전환할 수 있다
- - 를 하면 해당 유저의 홈 디렉토리에 있는 설정들을 읽어와서 환경변수가 적용되기 때문에 위치가 그 유저의 홈디렉토리로 변경된다

```
[root@localhost user02]# pwd
/home/user02
[root@localhost user02]# su user01
[user01@localhost user02]$ pwd
/home/user02
[user01@localhost user02]$ su - user01
Password:
Last login: Mon Oct 23 10:32:56 KST 2023 on pts/2
[user01@localhost ~]$ pwd
/home/user01
[user01@localhost ~]$
```

- user02의 홈디렉토리에서 su user01로 유저1에 로그인하면 위치는 여전히 user02의 홈디렉토리
- 근데? su - user01로 유저1에 로그인하면 유저01의 홈디렉토리로 위치바뀜! → 해당 유저의 환경설정을 읽어온다

su VS su -

su	su -
passwd 필요 X	passwd 필요 O
위치 안바뀜	**로그인한 유저의 홈 디렉토리로** 위치 바뀜

- 오 근데 root에서 su -로 할때는 비밀번호 없이 마음대로 유저 전환 가능

su VS sudo

- su = switch user
- sudo = superuser do

▼ sudo 사용조건

1. 현재 사용자의 passwd
2. 사용자가 /etc/sudoer 파일에 등록되어있어야함

   → 원래는 sudoer이라는 파일에 등록해야하지만 **이제는 wheel 그룹에 사용자 넣어서 사용함**

- sudo는 root의 권한을 대행하지만 나의 passwd를 이용함!

🍄 root의 권한이 필요해! → 원래는 root의 passwd를 알아야하는데 공유하지 않음 → sudo를 이용해서 root의 권한을 잠깐 쓸게~ 하고 빌려옴 → root의 passwd를 가져오는 것은 아니기 때문에 내 passwd를 이용해서 사용함

고급 권한 관리

확장 권한

▼ 왜 씀?

- 읽기 쓰기 실행 말고 다른 권한이 필요?
- passwd 파일을 실행하게 되면 실행한 사람의 권한으로 실행해야 하는데, 파일을 소유한 사람의 권한으로 실행함 → 파일에 대한 소유권을 일시적으로 빌려줌

  → 이럴때 쓰는게 **확장권한**

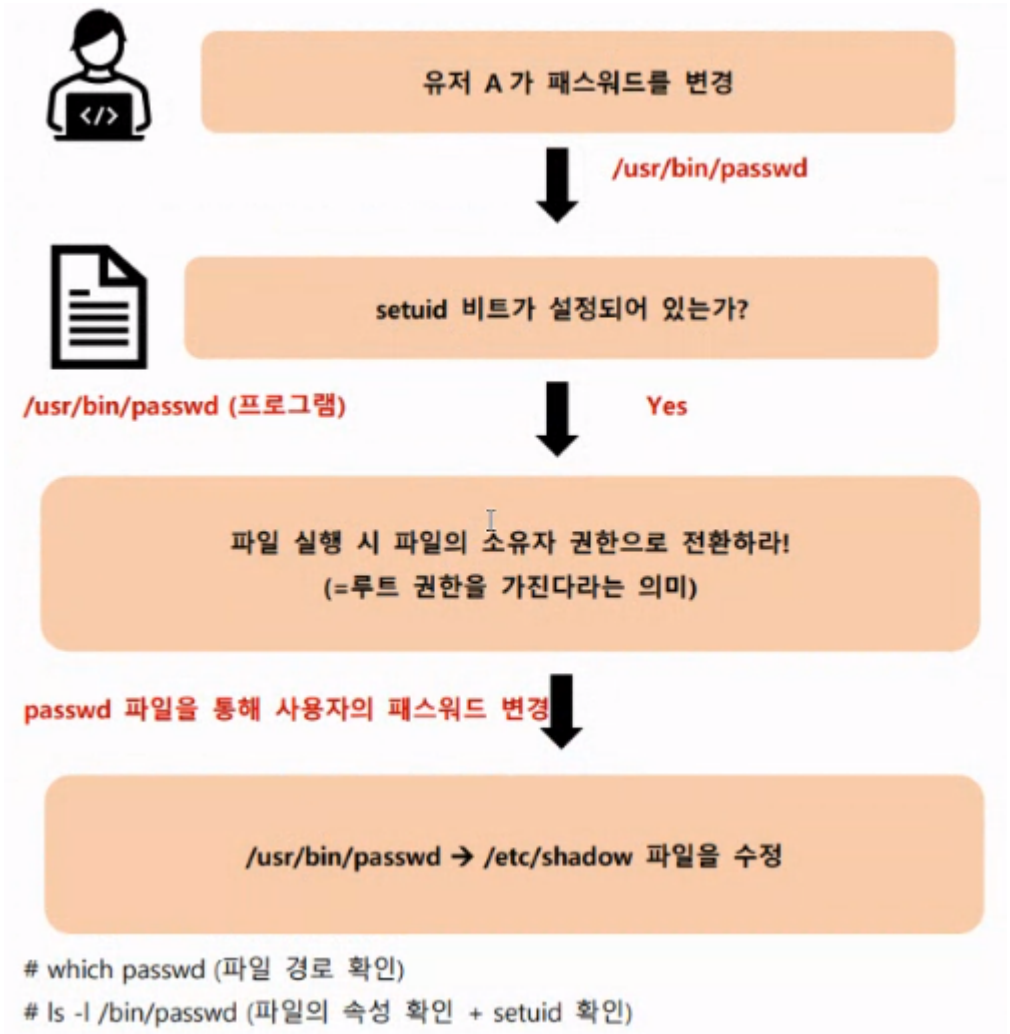
- 소유권이 없는 사용자도 파일에 대한 권한을 일시적으로 행사해야 할 때, 일시적으로 파일에 대한 소유권을 빌려줄때 쓰는게 확장권한

setuid

```
[user02@localhost ~]$ which passwd
/bin/passwd
[user02@localhost ~]$ ls -al /bin/passwd
-rwsr-xr-x. 1 root root 27856 Apr  1  2020 /bin/passwd
```

- x 자리에 s가 있음!! → 이거는 setuid가 설정된 파일이야!

🦄 user02가 로그인할때 passwd에 접근해야함 → 근데? 소유자 root라서 원래 접근 못함! → 지금 other은 권한이 x 실행밖에 없음 → 어떡하지? → 나에게 소유권은 없지만 일시적으로 써야해 → 권한 좀 잠깐 쓸게~ ⇒ **확장 권한 s**



- setuid가 설정된 파일 : 파일을 실행하는 사용자가 주체가 되는게 아니라 파일을 소유하고 있는 사용자의 권한으로 프로세스를 실행하는 것

setgid

- setgid는 보통 디렉토리에 설정함
- 디렉토리를 열고 들어가면 여러 파일이 있을 수 있음 → 디렉토리 소유자와 그 밑에 파일의 소유자가 다를 수 있음 → setgid 안주면 ? 아래 파일에 권한 없음

💧 test01이랑 test02를 learning 그룹에 둘다 넣음
→ workspace라는 디렉토리 만들고 learning 그룹에 w 권한 줌
→ test01로 test1 파일을 만듬 → 이 파일의 소유자는 test01
→ test02로 로그인
→ test1 파일에 접근
→ learning 그룹에 있어서 workspace 디렉토리에는 접근 가능하지만, test1 파일은 test01 소유라서 readonly라고 뜸!
→ test01이 만든 파일은 test02가 수정 불가 / test02가 만든 파일은 test01이 수정 불가!
→ 매번 root로 권한 줄 수 없음 → setgid 이용하기
근데?? 나갈때 :wq! 로 나가면 수정 가능함……. 개무서움

- setgid를 주면 권한을 상속받는다

```
[root@localhost workspace]# ls -lod /workspace
drwxrwxr-x. 2 root 20 Oct 23 12:12 /workspace
[root@localhost workspace]# chmod g+s /workspace
[root@localhost workspace]# ls -ld /workspace
drwxrwsr-x. 2 root learning 20 Oct 23 12:12 /workspace
```

- **chmod g+s /workspace** : workspace라는 디렉토리에 s 권한을 주겠다 → setgid를 설정하겠다
- chgrp 그룹이름 파일/디렉토리이름 : 이 파일이나 디렉토리의 그룹을 그룹이름으로 바꾸겠다
- workspace 디렉토리에서 group의 권한이 rwx에서 rws로 바뀜

```
-rw-rw-r--. 1 test01 learning  11 Oct 23 12:15 AAA
```

- setgid를 하고 파일을 만들면 그룹에 속한 아무 유저나 로그인해서 파일을 만들어도 **디렉토리의 권한이 그대로 상속**된다 → 그 그룹에 속한 누구던지 그 파일을 편집할 수 있다

소문자 s VS 대문자 S

```
drwxrw-r-x. 2 root learning 6 Oct 23 12:28 /workspace2
[root@localhost ~]# chmod g+s /workspace2
[root@localhost ~]# ls -ld /workspace2
drwxrwSr-x. 2 root learning 6 Oct 23 12:28 /workspace2
```

- x 권한이 **있을때** setgid → 소문자 s → 이 디렉토리에 들어갈 수 있음
- x 권한이 **없을때** setgid → 대문자 S → 이 디렉토리에 못들어감! (실행권한이 없어서→디렉토리는 실행권한이 없으면 못들어감)

sticky bit

- 원래는 root나 파일을 소유한 사용자만 파일을 삭제나 생성 할 수있는 권한을 가짐
- 근데 sticky bit를 가지면 어느 누구나 파일을 생성할 수 있지만, 삭제는 소유자와 root만 가능
- **생성은 누구나 가능 / 삭제는 아무나 X 소유자나 root만 가능** → 인터넷 게시판이랑 비슷

```
drwxrwxrwt. 18 root root 4096 Oct 23 12:28 /tmp
```

- other의 x권한 자리에 t가 있으면 여기에 파일을 생성하는건 누구나 가능하지만 삭제는 소유자나 root만 가능하다!

🍭 sticky bit가 설정된 tmp 파일에서 user01로 파일 만들고 user02로 와서 삭제하려고 하면 못지움

```
[user01@localhost tmp]$ rm -f user02test
rm: cannot remove ‘user02test’: Operation not permitted
```

- 파일을 소유한 사람과 root만 삭제가능
- -f로 강제로 지우려고 해도 불가능

▼ 이런거 왜??

멀티유저니까

> setuid, setgid, sticky bit 모두 확장권한을 설정할때 각각 바뀌는 권한의 x 자리가 다른 알파벳으로 바뀜!
>
> - setuid → user의 x 자리가 s로 바뀜
> - setgid → group의 x 자리가 s로 바뀜
> - sticky bit → other의 x 자리가 t로 바뀜

ACL

- Other에 다 넣고 퉁치는게 아니라 **특정 유저에게 특정 파일과 디렉토리**에 대한 권한을 세부적으로 줘야할 경우에 씀
- ACL 적용 유무를 확인하려면 ls -l로 확인해서 맨 끝에 + 의 유무로 확인가능

```
[user01@localhost ~]$ setfacl -m u:test01:rwx ACLtest
```

- setfacl -m u/g/o:이름:줄권한 파일이름 : 이 이름의 u/g/o에게 이 파일에 대해 r/w/x 권한을 줄거야!

```
-rw-rw-r--. 1 user01 user01     0 Oct 23 14:27 ACLtest
```

ACL 주기 전

```
-rw-rwxr--+ 1 user01 user01     0 Oct 23 14:27 ACLtest
```

ACL 준 후

- ACL 주기 전에는 맨 뒤가 . ACL 준 후에는 맨 뒤가 +
- getfacl 파일이름 : 이 파일에 대해 누구한테 어떤 권한 있는지 알려줘

작업 스케줄링

- 작업의 **주기**(단일, 반복)
- **누가** 작업을 걸거야?

▼ at 시간 : 딱 그 시간에 작업해라

- 단일성 작업을 예약할 때 사용
- 한번 실행하고 나면 직업이 삭제됨
- atd : at daemon / 데몬 프로세스이랑 같은 데몬이라는 뜻 (뒤에서 돌고있다)

```
drwx------.  3 root root  31 May 19  2022 at
```

- 이 at 안에 있다가 한번 실행되고 사라짐
- systemctl stop / systemctl start

```
[root@localhost ~]# at now +5min
at> ps -ef > /root/pslist
at> <EOT>
job 1 at Mon Oct 23 15:32:00 2023
```

- at now +~ : 지금부터 ~ 후에 작업해
- at> 로 바뀜
- ps -ef : 명령을 실행한 결과값을 /root/pslist 파일에 저장
- atq : at명령으로 등록한 작업 확인하기

▼ 주기적인 작업 예약

- 특정 주기마다 실행되는 작업
- crontab 사용 → crond 데몬에 의해서 작업이 실행
- crontab -e : 내가 작업을 걸려고하는 유저로 로그인한 상태에서 해야함
- 분 시 일 월 요일 명령어 → 순서대로 써야함
- 매 번은 * 로 표기함

> 매일 매시 45분마다 clock.txt라는 파일을 만들어!
> → 45 * * * * touch clock.txt
> 근데 만약 이미 있으면??
> → 그냥 덮어씀.. → 근데 내용은 안바뀐다???????? 파일 생성 시간만 바뀜

```
-rw-rw-r--.  1 user01 user01     0 Oct 23 15:45 1234.txt
[user01@localhost ~]$ ls -al | grep '1234'
-rw-rw-r--.  1 user01 user01     0 Oct 23 15:46 1234.txt
```

crontab 응용문제

새로운 유저(test03)을 생성하고 그 유저로 크론탭을 거는 것이 목표

date : 지금 시간을 보여줌

date라는 명령어를 사용한 결과값을 리다이렉션을 이용하여 date.log라는 파일로 출력

주기는 2분마다

1. test03 유저 생성하고 패스워드 등록
2. **test03으로 로그인 → 누가 작업을 거는지 중요함**
3. date.log라는 파일 만듬
4. crontab -e로 편집모드 들어감
5. */2 * * * * date >> 파일경로

*/2 가 2분마다! 라는 뜻 → 1일마다면 * */1 * * * 이런식

매달 10일 03시 05분마다

5 3 10 * *

> 는 덮어쓰기 >> 는 추가하기

```
[root@localhost ~]# cat /root/date.log
Mon Oct 23 16:11:01 KST 2023
Mon Oct 23 16:12:01 KST 2023
Mon Oct 23 16:13:01 KST 2023
Mon Oct 23 16:14:01 KST 2023
```

> 유저 생성할때 주석달기
>
> useradd 유저이름 -c '주석'
>
> 하면 이 유저에 주석이 달림! 무슨 일을 하는 유저인지 알아보기 쉽게해줌
>
> passwd 파일에서 주석 확인 가능

```
[root@localhost ~]# useradd test04 -c 'a user go home'
[root@localhost ~]# tail -3 /etc/passwd
test02:x:1003:1004::/home/test02:/bin/bash
test03:x:1004:1005::/home/test03:/bin/bash
test04:x:1005:1006:a user go home:/home/test04:/bin/bash
```