시스템서버운영

교수 : 추연수 교수님
동양미래대학교
컴퓨터정보공학과 2-PC
20190747 기호정

강의계획서

Part 1

1.1 Umask란 무엇인가?

1.2 Umask는 어떻게 사용되는가?

1.3 Umask 사용할 때 주의사항은 무엇인가?

Part 2

2.1 SetUID의 기능은 무엇인가?

2.2 SetUID는 어떻게 사용되는가? (사용방법)

2.3 SetUID는 언제 어떤 상황에서 사용되는가? (사용타이밍)

2.4 SetUID 사용 시 주의사항

Part 3

3.1 SetGID의 기능은 무엇인가?

3.2 SetGID는 어떻게 사용되는가? (사용방법)

3.3 SetGID는 언제 어떤 상황에서 사용되는가? (사용타이밍)

3.4 SetGID 사용 시 주의사항

Part 4

4.1 Stickybit의 기능은 무엇인가?

4.2 Stickybit는 어떻게 사용되는가? (사용방법)

4.3 Stickybit는 언제 어떤 상황에서 사용되는가? (사용타이밍)

4.4 Stickybit 사용 시 주의사항

PART 1

1.1 Umask란 무엇인가?

Umask란 주어진 콘테스트에서 새로 만들어진 파일과 디렉토리 퍼미션에 적용될 마스크를 지정해주는 지시자이다. 프로세스의 파일 모드 작성 마스크이며 디렉토리 자체에 설정할 수 없다.

새로운 디렉토리와 파일들이 쓰기 권한을 가진 그룹이나 전체가 되는 것을 막기 위해 Umask를 합리적인 표준값인 002, 007, 020, 070, 022등으로 설정한다. 파일이라는 것은 원래 목적이 읽고 쓰는 것이다. 그러므로 일반적으로 실행을 하는 것은 드물다. 그러므로 값을 모두 짝수로 지정하게 되는 것이다.

대부분의 응용 프로그램은 실행 권한이 설정된 파일을 만들지 않으므로 기본값은 666, 디렉토리의 풀 퍼미션은 777이며 umask에 의해 수정된다.

계산을 하면

파일 퍼미션	디렉토리 퍼미션
666 -022 ------ 644	777 -022 ------ 755

umask를 입력하여 보면 현재 설정된 값을 확인할 수 있다. touch ddd를 해서 ddd란 파일을 생성 후 ls -l ddd를 통하여 확인을 하면 디폴트로 644가 된다. 디렉토리를 확인하고 싶다면 mkdir kbs라고 입력하여 디렉토리 생성 후 ls -l kbs를 통하여 확인을 하면 디폴트로 755가 되어 있다.

이제 한번 바꾸어 보면

```
[root@localhost ~]# grep umask /etc/*
/etc/bashrc:     umask 002
/etc/bashrc:     umask 022
/etc/csh.cshrc: umask 022
/etc/csh.cshrc: umask 002
/etc/ltrace.conf:octal umask(octal);
/etc/ltrace.conf:octal SYS_umask(octal);
[root@localhost ~]# umask
0022
[root@localhost ~]# umask 0024[root@localhost ~]# umask
0024
```

grep으로 etc 밑의 모든 파일에서 umask를 찾아보면 디폴트로 모두 설정되어있는 것을

확인할 수 있을 것이다. umask 명령어를 이용해서도 0022 라는 것을 확인하였다. umask 0024를 통하여 값을 바꾸어 주니 정상적으로 바뀌어져 있는 것을 확인할 수 있다.

단, 이때 etc 밑의 디폴트로 설정되어 있는 값들은 변경되지 않으니 참고하자.

이제 0024 로 잘 되어있는지 확인해보면

umask 0022	umask 0024
[root@localhost ~]# umask 0022 [root@localhost ~]# touch bbb[root@localhost ~]# ls -l bbb-rw-r--r-- 1 root root 0 8월 1 20:53 bbb ==> | - | rw- | r-- | r-- | rw- : 4+2= 6 r-- : 4 r-- : 4 그러므로 644 로 되어있는 것을 확인 할 수 있다. 666-022 = 644 [root@localhost ~]# mkdir ccc[root@localhost ~]# ls -dl cccdrwxr-xr-x 2 root root 4096 8월 1 20:54 ccc ==> | d | rwx | r-x | r-x | rwx : 4+2+1 = 7 r-x : 4+1 = 5 r-x : 4+1 = 5 그러므로 755로 되어있는 것을 확인할 수 있다. 777-022 = 755	[root@localhost ~]# umask 0024 [root@localhost ~]# touch bbb [root@localhost ~]# ls -l bbb -rw-r---w- 1 root root 0 8월 1 20:57 bbb ==> | - | rw- | r-- | -w- | rw- : 4+2 = 6 r-- : 4 -w- : 2 642 로 변경되었다. 맨 뒤의 other 즉 모든 사용자가 파일의 수정이 가능하도록 한 것이다. 666 - 024 = 642 [root@localhost ~]# mkdir ccc [root@localhost ~]# ls -dl cccdrwxr-x-wx 2 root root 4096 8월 1 20:58 ccc ==> | d | rwx | r-x | -wx | rwx : 4+2+1 = 7 r-x : 4+1 = 5 -wx : 2+1 = 3 753 으로 변경되었다. 맨뒤의 other 즉 모든 사용자가 파일의 수정과 싫ㅇ이 가능하도록 설정되었다. 777 - 024 = 753

보통 사용을 할 때에는 0024를 절대 쓰지 않는다. 위의 예제는 연습용으로 0077로 하면 디렉토리 파일이 모두 안전하게 된다.

파일 --> 666-077 = 600 : | - | rw- | --- | --- |

디렉토리 ---> 777-077 = 700 : | d | rwx | --- | ---|

즉, 루트 이름을 가진 자만 해당 사항을 관리할 수 있는 것이다.

소유자에 대한 읽기 / 쓰기 비트와 다른 사용자에 대한 읽기 비트를 제거하도록 umask를 설정했으므로 777 응용 프로그램에서와 같은 기본값은 파일 권한이 133 된다. 이것은 나 (그리고 다른 사람들)이 파일을 실행할 수 있고 다른 사람들이 파일에 쓸 수 있음을 의미한다. 소유자가 아닌 다른 사람이 파일을 읽거나 쓰거나 실행하지 못하게 077 하려면 그룹 및 다른 사람에 대한 해당 권한을 끄는 것과 같은 umask를 사용해야 한다. 반대로 umask of 000 는 새로 만든 디렉토리를 모든 사람이 읽고 쓸 수 있고 내림차순으로 만들 수 있다(권한은 777). 이러한 umask는 안전하지 않으므로 umask로 설정해서는 안 된다 000.

Ubuntu의 기본 umask 022는 새로 만든 파일을 모든 사람이 읽을 수 있지만 소유자만 쓸 수 있음을 의미한다.

```
user@computer:~$ touch new-file-name
user@computer:~$ ls -dl new-file-name
-rw-r--r-- 1 user user 0 Apr  1 19:15 new-file-name
```

Ubuntu Oneiric (11.10)부터 기본 umask가 완화되어 002 소유자 그룹에 대한 쓰기 액세스가 확장되었다.

```
user@computer:~$ touch new-file-name
user@computer:~$ ls -dl new-file-name
-rw-rw-r-- 1 user user 0 Apr  1 19:15 new-file-name
```

- umask 보기 및 수정

현재 umask 설정을 보려면 터미널을 열고 다음 명령을 실행.

```
umask
```

현재 쉘의 umask 설정을 077과 같은 다른 것으로 변경하려면 다음을 실행.

```
umask 077
```

이 설정이 작동하는지 테스트하려면 새 파일을 작성하고 (기존 파일의 파일 권한에 영향을 미치지 않음) 파일에 대한 정보를 표시해야 함.

```
user@computer:~$ touch new-file-name
user@computer:~$ ls -dl new-file-name
-rw------- 1 user user 0 Apr  1 19:14 new-file-name
```

umask 설정은 동일한 쉘에서 시작된 프로세스에 의해 상속된다. 예를 들어, gedit 터미널에서 실행하여 텍스트 편집기 GEdit을 시작하고 gedit를 사용하여 파일을 저장하면 새로 만든 파일은 터미널에서와 동일한 umask 설정의 영향을 받는다.

1.2 umask는 어떻게 사용되는가?

umask는 새로운 파일 및 디렉토리를 만들 때 자종으로 접근(엑세스:읽기/쓰기/실행)권한을 설정해 준다.

umask 명령어의 일반적인 형태는 다음과 같다.

```
umask nnn
```

위에서 nnn은 000 부터 777 까지의 숫자를 의미한다. umask의 설정은 일반적으로 /etc/profile에서 설정하여 시스템의 전체를 기본 설정하거나 개인적으로 설정을 원할 경우에는 .profile이나 혹은 .bashrc (Bash 쉘을 이용할 경우)나 .cshrc (C 쉘을 이용할 경우) 등 자신이 사용하는 쉘의 환경 파일에 등록을 해두면 로그인과 동시에 설정이 되어 편리하게 이용할 수 있다.

자, 그럼 umask를 어떻게 사용하는지 알아보자. umask 명령어를 사용할 때 함께 사용한 숫자는 팔진수의 보수(complement)로써 파일과 디렉토리에 작용한다. 단, 알아두어야 할 것은 유닉스 시스템에서는 기본적으로 파일이 실행 권한('x')을 갖고 생성되지 못하도록 한다는 점이다. 디렉토리의 경우에는 실행 권한이 그 디렉토리의 접근 권한을 나타내는 것이기 때문에 실행 권한을 생성과 동시에 가질 수 있다. 다음의 표는 umask의 명령으로 설정된 파일과 디렉토리의 권한을 나타낸 것이다.

다음의 표는 umask의 명령으로 설정된 파일과 디렉토리의 권한을 나타낸 것이다.

Table 2.2: umask permission table

umask	File	Directory
0	6	7
1	6	6
2	4	5
3	4	4
4	2	3
5	2	2
6	0	1
7	0	0

다음의 예를 보면서 이해를 해보면

umask 명령	권한 모드	
(숫자)	File	Directory
077	rw- --- --- (600)	rwx --- --- (700)
007	rw- rw- --- (660)	rwx rwx --- (770)
022	rw- r-- r-- (644)	rwx r-x r-x (755)

umask와 십진수 (Octal complement)

리눅스 파일에는 "소유자(user), 그룹(group), 그 외(other)"라는 세 종류의 사용자가 항시 존재한다. 각 사용자 유형에는 다시 "읽기 허가권, 쓰기 허가권, 실행 허가권"의 세 가지 허가권이 항시 정의되어 있다. 각 사용자 유형의 세 가지 허가권은 이진수로 표현되어 있는데, 최솟값인 이진수 000(읽기, 쓰기, 실행권 없음)에서부터 최댓값인 이진수 111(세 가지 모두 있음)까지다. (최곳값인 이진수 111은 십진수로 7이므로 어떤 파일에 대한 허가권이 "111.111.111"이라면 이것은 일반적으로 십진수 "777"로 부른다.)

umask는 새로 만들어질 파일과 디렉토리들의 디폴트 소유권 값을 정할 때 사용한다. umask에는 새로 만들어지는 모든 파일의 기준값에 반한 십진수를 사용한다. 예로써 한 사용자의 umask 값을 022에서 077로 바꿔주면, 이 순간부터 앞으로 이 사용자가 만드는 파일과 디렉토리는 111.000.000의 가장 제한적인 허가권을 자동으로 가지게 된다.

umask와 십진 보수의 예:

파일 값	십진수	십진 전수	
-.uuu.ggg.ooo	u.g.o	u.g.o	umask
-.rwx.rwx.rwx			
0.111.111.111	7.7.7	0.0.0	umask 0
0.111.111.100	7.7.4	0.0.3	umask 3
0.111.101.101	7.5.5	0.2.2	umask 22
0.111.000.000	7.0.0	0.7.7	umask 77

파일 허가권의 십진수 사용

예를 들어서, 이진수 "110.100.000"은 십진수 "640"으로 표시된다. (이것은 "rw-.r--.---"이다.) 이진수인 파일 허가권을 "640" 등의 십진수로 부르는 것은 초보자로서는 헷갈릴 수 있지만 "110.100.000" 등으로 부르는 것보다 "640"으로 부르는 것이 훨씬 편하다.

UMASK 계산원리

umask 022 이라면

Directory	File	계산 방법
Default Permission 7 7 7 111 111 111	Default Permission 6 6 6 110 110 110	022의 보수를 취함 000 010 010 **111 101 101**
111 111 111 111 101 101 ---------- 111 101 101 → 755	110 110 110 111 101 101 ---------- 110 100 100 → 644	보수를 취한 값과 AND 연산을 함

1.3 Umask 사용할 때 주의사항은 무엇인가?

umask 값을 022로 수정하면 755로 파일 / 디렉토리가 생성 되야할 것 같지만 그렇지 않다. 디렉토리에 대해서는 해당 설정이 그대로 적용되지만, 파일에 대해서는 수동으로 권한을 지정해야 한다. (chmod 를 사용)

umask 122 로 지정하면 chmod 를 생각했을 때 655 로 지정됨을 의미한다. owner 에 적용되는 권한은 read 와 write 이다. 폴더에 실행권한이 있다는 것은 폴더의 이동이 가능하다는 것을 의미한다. 위와같이 설정하면 폴더에 접근할 수 없게 된다.

```
ocp@orcl : /home/oracle> mkdir 1
ocp@orcl : /home/oracle> cd 1
bash: cd: 1: Permission denied
ocp@orcl : /home/oracle> ls -lad 1
drw-r-xr-x 2 oracle oinstall 4096 Feb 28 10:46 1
ocp@orcl : /home/oracle> chmod 755 1
ocp@orcl : /home/oracle> cd 1
```

umask 1?? 와 같이 줄 일은 없다는 것을 알 수 있다.

umask 값을 022로 원복해서 directory / file에 대해 권한 할당 정보를 확인해 보면

```
ocp@orcl : /home/oracle> mkdir 1
ocp@orcl : /home/oracle> ls -lad 1
drwxr-xr-x 2 oracle oinstall 4096 Feb 28 10:52 1
ocp@orcl : /home/oracle> cd 1
ocp@orcl : /home/oracle/1> touch 1
ocp@orcl : /home/oracle/1> ls -la
total 20
drwxr-xr-x  2 oracle oinstall 4096 Feb 28 10:52 .
drwxr-xr-x 18 oracle dba    4096 Feb 28 10:52 ..
-rw-r--r--  1 oracle oinstall    0 Feb 28 10:52 1
```

위 결과를 보면 디렉토리에 대해서는 umask 설정이 그대로 적용되었지만, 파일에 대해서는 execute 권한이 빠져있는 걸 확인할 수 있다.

PART 2

2.1 Set UID의 기능은 무엇인가?

SetUID는 Set UID의 약자로서 프로세스가 실행 중인 동안 일시적으로 해당 실행 파일의 소유자, 소유그룹의 권한으로써 자원에 접근할 수 있도록 하는 권한 설정이다. 즉, SetUID가 설정된 프로그램을 실행시키면 프로세스로 동작하는 동안에 RUID는 실행시킨 사용자의 UID로 설정되고 EUID는 실행파일 소유자의 UID로 설정된다. 따라서 실행시킨 사용자(user)와는 무관하게 프로세스가 실행중인 동안에는 실행파일 소유자의 권한으로 자원에 접근하게 된다. 예를 들어 root 권한으로 지정된 프로그램에 SetUID가 지정되어 있다면 실행할 때 root 권한으로 실행된다.

SetUID가 적용되어 있는 파일 중 가장 대표적인 것은 /usr/bin/passwd 파일이다. 해당 파일은 계정의 비밀번호를 변경할 수 있도록 하는 명령어 실행 파일로 /etc/passwd로 접근하여 비밀번호를 변경하도록 한다. 실제 /etc/passwd의 권한은 소유주인 root만이 변경이 가능하도록 설정이 되어있다. 만약 /usr/bin/passwd에 SetUID가 적용되어있지 않다면 일반 사용자들은 항상 관리자를 거쳐 자신의 비밀번호를 변경해야 한다. 이런 번거로움을 줄이기 위해 일반 사용자들도 root의 권한으로 /etc/passwd 파일을 수정 가능하도록 설정한 것이다.

2.2 어떻게 사용되는가?

SetUID를 적용하기 위해서는 기존의 허가권 앞에 4를 붙이면 된다.

```
[root@219226 cofls]# touch setuid
[root@219226 cofls]# ll
total 0
-rw-r--r-- 1 root root 0 May 22 23:00 setuid
[root@219226 cofls]# chmod 4644 ./setuid
[root@219226 cofls]# ll
total 0
-rwSr--r-- 1 root root 0 May 22 23:00 setuid
```

ex) # chmod 4644 파일이름

<기존 권한에 실행권한이 없으면 대문자 S, 있으면 소문자 s로 표시된다.>

기호모드로는 **u+s**, 숫자모드로는 **4**이다. 예를 들어

chmod u+s test1이라고 하거나 chmod 4644 test2

라고 쓰면 소유주의 실행 권한, 첫 번째 execute(x)가 있어야 될 자리에 s가 보일 것이다.

2.3 SetUID는 언제 어떤 상황에서 사용되는가?

슈퍼유저 root만 접근할 수 있는 파일이나 명령에 대해, 일반 사용자로 접근하는 것이 필요한 경우 사용한다.

예를 들어 root가 아닌 A라는 계정 일반 사용자가 아래 리눅스 명령으로써 패스워드를 변경한다고 생각해보면 당연히 변경된 패스워드는 리눅스의 패스워드 파일인 /etc/shadow에 저장됨을 확인할 수 있다. 그렇지만 /etc/shadow 파일의 퍼미션은 다음과 같이 되어있다.

-rw------- 1 root root 24736 Oct 23 17:54 /etc/shadow

현재 /etc/shadow 파일은 root는 읽고 쓰기, root그룹과 others는 아무 권한이 없는 상태이다. 그런데 어떻게 A라는 계정 사용자가 명령을 내려서 /etc/shadow 파일이 갱신이 될 수 있을까? 바로 여기에 SetUID의 비밀이 숨어있다.

chmod u+s test1이라고 하거나 chmod 4644 test2라고 쓰면

-rwS—x—x 1 rot root 32888 Mar 11 2009 /bin/passwd

소유주의 실행 권한, 첫 번째 execute(x)가 있어야 될 자리에 s가 보일 것이다.

쉽게 말하면 시스테의 어떠한 계정이라고 /bin/passwd 파일을 실행하는 동안에는 소유주 즉, root의 권한을 갖는다는 것이다. 물론 실행이 종료되면 다시 A라는 계정의 권한으로 돌아온다.

2.4 SetUID 사용 시 주의사항

SetUID가 설정된 파일(특히, root 소유의 파일인 경우)은 특정 명령어를 실행하여 root 권한 획득 및 정상 서비스 장애를 발생시킬 수 있으며, 로컬 공격에 많이 이용되므로 보안상 철저한 관리가 필요하다. root 소유의 SUID 파일의 경우에는 꼭 필요한 파일을 제외하고는 SUID 속성을 제거해주고, 잘못 설정되어 보안위협이 되고 있는지 주기적인 진단 및 관리가 요구된다. SUID 설정된 파일 실행 시, 특정 수행을 위하여 일시적으로 파일 소유자의 권한을 얻게 된다. 판단 기준은 주요 파일의 권한에 SUID에 대한 설정이 부여되어 있는 경우 취약하다고 판단하며, 조치 방법은 불필요한 SUID 파일 제거하고, 아래의 목록 이외에 애플리케이션에서 생성한 파일이나 사용자가 임의로 생성한 파일 등 의심스럽거나 특이한 파일의 발견 시 SUID 제거가 필요하다.

PART 3

3.1 SetGID의 기능은 무엇인가?

SetGID는 SetUID와 마찬가지로 유효 그룹 ID(EGID)를 사용자의 실제 그룹 ID에서 파일 소유자의 그룹 ID로 변경한다. RUID와 EUID가 해당 UID로 설정되는 것과 같이, SetGID로 인해 RGID와 EGID가 해당 GID로 설정된다. 이러한 SetGID가 설정되지 않은 프로세스를 실행시키면 RUID와 EUID, RGID와 EGID가 동일하게 설정된다. 즉, 프로세스를 실행시킨 사용자의 UID와 GID로 RUID EUID와 RGID, EGID가 설정되어 해당 권한으로 자원에 접근하게 되는 것이다. 그러나 실행 권한이 없을 경우 대문자 S로 표시된다.

SetGID가 설정된 파일을 실행할 때 일시적으로 파일 소유그룹의 권한을 얻어 실행하도록 한다. 즉, SetGID가 디렉토리에 설정되어 있으면, 이 디렉토리에 새로 설정된 파일들은 디렉토리 그룹 소유권보다 파일 생성자의 그룹 소유권을 얻게 될 것이다.

3.2 SetGID는 어떻게 사용되는가?

SetGID를 적용하기 위해서는 기존의 허가권 앞에 2를 붙이면 된다.

```
[root@19226 cofls]# touch setgid
[root@19226 cofls]# chmod 2644 ./setgid
[root@19226 cofls]# ll
total 0
-rw-r-Sr-- 1 root root 0 May 22 23:07 setgid
```

ex) # chmod 2644 파일이름

<기존 권한에 실행권한이 없으면 대문자 S, 있으면 소문자 s로 표시된다.>

기호모드로는 **g+s**(setgid 설정 제거 g-s), 숫자모드로는 **2**이다. 예를 들어

chmod g-s test1이라고 하거나 chmod 2755 test2

라고 쓰면 소유주의 실행 권한, 첫 번째 execute(x)가 있어야 될 자리에 s가 보일 것이다.

3.3 SetGID는 언제 어떤 상황에서 사용되는가?

예를 들어 부서원들이 많은 그룹에서 관리할 디렉터리에 setGID를 사용하면 모든 사용자는 같은 소유그룹의 권한으로 파일을 관리할 수 있다.

SetGID 권한은 SetUID와 동일하지만 일반적으로 디렉터리에 설정되고 소유그룹(group)의 권한을 갖는다는 차이만 있다.

-r-xr-sr-x 1 root mail /user/bin/mail

소유그룹의 실행 권한, 두 번째 execute(x)가 있어야 될 자리에 s가 보일 것이다. 하지만 역시 실행 권한이 없을 경우 대문자 S로 표시된다. 만약 root 소유의 자원에 대하여 일반 사용자의 직접 접근을 차단하면서 기능상 필요한 부분만을 접근하도록 허용할 필요

가 있는 경우 별도의 root 소유 프로그램을 통해 SGID를 설정하여 사용하는 것이 보안적인 측면과 운영적인 측면에서 매우 효율적이다,

3.4 SetGID 사용 시 주의사항

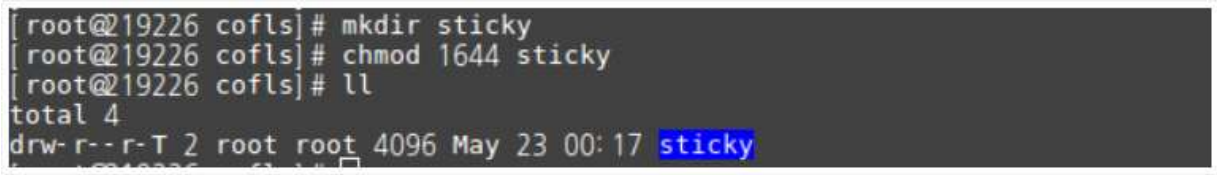
SetGID가 설정된 파일(특히, root 소유의 파일인 경우)은 특정 명령어를 실행하여 root 권한 획득 및 정상 서비스 장애를 발생시킬 수 있으며, 로컬 공격에 많이 이용되므로 보안상 철저한 관리가 필요하다. 잘못 설정되어 보안 위협이 되고 있는지 주기적인 진단 및 관리가 요구된다. 판단 기준은 주요 파일의 권한에 SGID에 대한 설정이 부여되어 있는 경우 취약하다고 판단하며, 조치 방법은 불필요한 SGID 파일 제거한다.

PART 4

4.1 Stickybit의 기능은 무엇인가?

일반적으로 유닉스 운영체제는 파일이나 디렉토리의 소유자, 소유그룹만이 삭제, 수정을 할 수 있도록 권한을 지정하지만, 모든 사용자들이 파일 생성, 수정, 삭제할 수 있는 디렉토리가 있다. /tmp와 /var/tmp 디렉토리는 모든 사용자가 사용하는 공용 디렉토리이다. 그렇다 보니 파일의 소유자가 아닌 다른 사용자가 777권한인 파일을 삭제, 수정하는 부분에서 문제가 발생하게 된다. 이를 방지하기 위해 Sticky Bit가 설정된 디렉토리에 파일을 생성하면 해당 파일은 생성한 사람의 소유가 되며, 소유자와 root만이 해당 파일에 대한 삭제 및 수정에 대한 권한을 가질 수 있다. 즉, Sticky Bit가 설정된 디렉토리 안에 누구나 파일을 생성하고 사용할 수는 있지만, 삭제나 이름 변경은 본인과 관리자만 가능하게 되는 것이다. 참고로 특수 권한은 해당 자리의 기존 권한에 실행 권한이 있어야 정상적으로 적용이 된다.

4.2 Stickybit는 어떻게 사용되는가?



Sticky Bit를 적용하기 위해서는 기존의 허가권 앞에 1를 붙이면 된다.

ex) # chmod 1644 디렉토리 이름

<기존 권한에 실행 권한이 없으면 대문자 T, 있으면 소문자 t로 표시된다.>

기호모드로는 **o+t 또는 u+t** 숫자모드로는 **1**이다. 예를 들어

chmod o+t test1이라고 하거나 chmod 1754 test2

라고 쓰면 소유주의 실행 권한, 첫 번째 execute(x)가 있어야 될 자리에 t가 보일 것이다.

4.3 Stickybit는 언제 어떤 상황에서 사용되는가?

예를 들어 폴더에 올라온 3개의 자료를 삭제하려는 상황일 때 우선 파일을 공유하기 위하여 permission이라는 폴더를 생성하고 다른 사용자들도 공유를 할 수 있도록 permission의 권한을 777로 설정한다. 그리고 Alice, Bob, Chris, David라는 계정을 생성하고, Bumsun 계정으로 접속하여 test.txt. 파일을 생성한다. 현재, Alice, Bob, Chris가 test.txt 파일을 공유하고 있다는 가정 하에 David 계정으로 접속하여 해당 파일을 삭제해보면 분명 Bumsun이라는 계정으로 test.txt.파일을 생성하였는데 David 계정으로 삭제가 된 것을 확인할 수 있다. 이것은 공유 디렉터리에 있는 파일을 다른 사용자가 삭제할 수 있기 때문에 굉장히 심각한 문제이다. 이런 문제를 해결하기 위해 Stickybit를 적용하면 다른 사용자가 파일의 삭제를 막기 위한 것임으로 다른 사용자(제 3자)필드의 실행권

한이 바뀌게 딘다. 단, 실행권한이 x에서 s가 아닌 t로 바뀌게 된다. 그리고 다시 David 계정으로 접속하여 test.txt.파일을 삭제해보면 삭제할 수 없게 된다.

4.4 Stickybit 사용 시 주의사항

대문자인 경우는 기존 파일의 권한에서 실행 권한이 없는 상태에서 설정했기 때문에 SUID, SGID 등을 설정해도 실행은 되지 않는다. 즉, 대문자로 나타나는 경우는 동작되지 않음을 의미한다.

마무리글...

수업을 듣기 전 예습의 의미로 이번 레포트를 작성하게 되었다.

수업 교재뿐만 아니라 유튜브, 블로그 등을 통해 다양한 자료들을 통해 조사하고, 참고하여 정리하였다.

또한, 정리에 그치지 않고, 정리한 것을 다시 읽으며 밑줄을 침으로써 중요한 부분들이 더 쉽게 눈에 띄어서 더 많이 읽으며 복습할 수 있도록 했다.

물론 중간고사 대체 과제이긴 하지만 과제라는 생각은 잊고, 블로그들을 참고하며 작성하는 것부터 도움이 되었다.

남은 시간 동안 이해가 안 된 부분은 반복적으로 보며, 이해하고 다음 부분으로 넘어갈 수 있도록 할 것이다.

이번 포트폴리오를 계기로 머릿속에 엉켜있던 앞서 배웠던 내용들도 조금이나마 정리할 수 있었다.

참고 : Google