

- Ch\_1 \_ 셸의 기본과 작성법
- Ch\_2 \_ 변수, 파라미터, 조건문
- Ch\_3 \_ 반복문, 함수, set 등
- Ch\_4 \_ 텔넷 서버
- Ch\_5 \_ OpenSSH 서버
- Ch\_6 \_ XRDP 서버

- 셸의 기본과 작성법

- 우분투의 bash 셸 (터미널)

- 기본 셸은 bash(Bourne Again Shell: '배시 셸')

- bash 셸의 특징

- ✓ Alias 기능(명령어 단축 기능)
    - ✓ History 기능(위/아래 화살표키)
    - ✓ 연산 기능
    - ✓ job Control 기능
    - ✓ 자동 이름 완성 기능 (탭키)
    - ✓ 프롬프트 제어 기능
    - ✓ 명령 편집 기능

- 셸의 명령문 처리 방법

- ✓ (프롬프트) 명령어 [옵션...][인자...]
    - ✓ 예) # rm -fg /mydir

## ● 환경 변수

- "echo \$환경변수이름" 으로 확인 가능
- "export 환경변수=값" 으로 환경 변수의 값을 변경
- 주요 환경변수

환경 변수	설명	환경 변수	설명
HOME	현재 사용자의 홈 디렉터리	PATH	실행 파일을 찾는 디렉터리 경로
LANG	기본 자원되는 언어	PWD	사용자의 현재 작업 디렉터리
TERM	로그인 터미널 타입	SHELL	로그인해서 사용하는 셸
USER	현재 사용자의 이름	DISPLAY	X 디스플레이 이름
COLUMNS	현재 터미널의 컬럼 수	LINES	현재 터미널 라인 수
PS1	1차 명령 프롬프트 변수	PS2	2차 명령 프롬프트(대개는 `>`)
BASH	bash 셸의 경로	BASH_VERSION	bash 버전
HISTFILE	히스토리 파일의 경로	HISTSIZE	히스토리 파일에 저장되는 개수
HOSTNAME	호스트의 이름	USERNAME	현재 사용자 이름
LOGNAME	로그인 이름	LS_COLORS	ls 명령의 확장자 색상 옵션
MAIL	메일을 보관하는 경로	OSTYPE	운영체제 타입

```

root@server:~/바탕 화면# echo $HOME
/root
root@server:~/바탕 화면# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
root@server:~/바탕 화면#

```

➔ HOME 과 PATH 사용 확인

## ● 셸 스크립트 프로그래밍

- c 언어와 유사하게 프로그래밍이 가능
- 변수, 반복문, 제어문 등의 사용이 가능
- 별도로 컴파일하지 않고 텍스트 파일 형태로 바로 실행
- vi 나 gedit 으로 작성이 가능
- 리눅스의 많은 부분이 셸 스크립트로 작성되어 있음

```
00_header
/etc/grub.d

131 if [ "$quick_boot" = 1 ]; then
132   cat <<EOF
133 function recordfail {
134   set recordfail=1
135 EOF
136
137 check_writable () {
138   abstractions="$(grub-probe --target=abstraction "${grubdir}")"
139   for abstraction in $abstractions; do
140     case "$abstraction" in
141       diskfilter |
142       btrfs | ext2 | ext3 | ext4 | fat | iso9660 | jfs | linux | minix | ntfs | reiserfs | squash4 | tar | vfat | xfs)
143       cat <<EOF
144       # GRUB lacks write support for $FS, so recordfail support is disabled.
145 EOF
146     return 1
147   ..
```

## ● 셸 스크립트의 작성과 실행

### • nano나 gedit으로 작성

```
*name.sh

1 #!/bin/sh
2 echo "사용자 이름: " $USER
3 echo "홈 디렉터리: " $HOME
4 exit 0
```

➤ 셸 스크립트 파일의 확장명은 되도록 \*.sh로 주는 것이 좋다.

➤ 셸 스크립트 파일을 /usr/local/bin/ 디렉토리에 복사하고, 속성을 755로 변경해 주면 모든 사용자가 스크립트를 사용할 수 있다.  
➤ 이 작업은 보안상 root만 수행함

### ➤ 실행방법

1. "sh <스크립트파일>"로 실행
2. "chmod +x <스크립트 파일>" 명령으로 실행 가능 속성으로 변경한 후에, "./<스크립트파일>" 명령으로 실행

```
1 #!/bin/sh
2 echo "사용자 이름: " $USER
3 echo "홈 디렉터리: " $HOME
4 exit 0
```

```
root@server:~# ls -l name.sh
-rw-r--r-- 1 root root 82 3월 17 17:58 name.sh
root@server:~# sh name.sh
사용자 이름: root
홈 디렉터리: /root
root@server:~#
```

➔ gedit 로 name.sh 을 생성 뒤 작성 후 실행

- 사용자이름과 홈 디렉터리가 나타나는 기능 코딩
- sh <스크립트파일> 로 실행 (sh name.sh)

```
root@server:~# ls -l name.sh
-rw-r--r-- 1 root root 86 1월 16 13:08 name.sh
root@server:~#
root@server:~# sh name.sh
사용자 이름: root
홈 디렉터리: /root
root@server:~#
root@server:~# chmod +x name.sh
root@server:~# ls -l
합계 40
-rwxr-xr-x 1 root root 86 1월 16 13:08 name.sh
```

## ➔ 파일 속성 변경 후 확인

- chmod +x [파일이름] 명령어로 파일 실행 속성 추가

## ● 변수의 기본

- 변수를 사용하기 전에 미리 선언하지 않으며, 변수에 처음 값이 할당되면서 자동으로 변수가 생성
- 모든 변수는 '문자열(String)'로 취급
- 변수 이름은 대소문자를 구분
- 변수를 대입할 때 '=' 좌우에는 공백이 없어야 함

```
ubuntu@server:~$ testval = hello
testval: 명령어를 찾을 수 없습니다
ubuntu@server:~$
ubuntu@server:~$ testval=hello
ubuntu@server:~$ echo $testval
hello
ubuntu@server:~$
ubuntu@server:~$ testval= Yes Sir
명령어 'Yes' 을 (를) 찾을 수 없습니다. 다음 명령어로 시도 하시겠습니까 :

deb mesmes의 명령어 ' (0.22-1)'
deb yescoreutils의 명령어 ' (8.30-3ubuntu2)'
deb lesatm-tools의 명령어 ' (1:2.5.1-4)'
deb nesfceaux의 명령어 ' (2.2.2+dfsg0-1build1)'
deb nesmednafen의 명령어 ' (1.22.2+dfsg-1build1)'
deb nesnestopia의 명령어 ' (1.50-1build1)'

Try: sudo apt install <deb name>

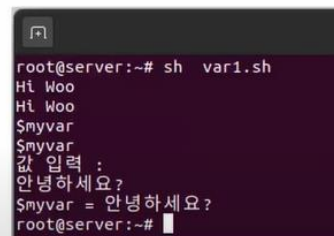
ubuntu@server:~$ testval="Yes Sir"
ubuntu@server:~$ echo $testval
Yes Sir
ubuntu@server:~$ testval=7+5
ubuntu@server:~$ echo $testval
7+5
ubuntu@server:~$
```

- '=' 오른쪽이 주입이 된다.
- 출력할때 \$ (대입할 때는 붙이지 않는다)
- 대입할 문자에 공백이 있을 경우 " " 사용
- 문자열로만 취급하기 때문에 testval=7+5 가 계산되지 않고 그대로 출력

## ● 변수의 입력과 출력

- '\$' 문자가 들어간 글자를 출력하려면 ' ' 로 묶어주거나 앞에 'w' 를 붙임
- " " 로 변수를 묶어줘도 된다.

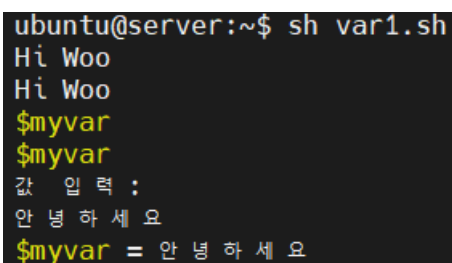
```
01 #!/bin/sh
02 myvar="Hi Woo"
03 echo $myvar
04 echo "$myvar"
05 echo '$myvar'
06 echo w$myvar
07 echo 값 입력 :
08 read myvar
09 echo '$myvar' = $myvar
10 exit 0
```



```
root@server:~# sh var1.sh
Hi Woo
Hi Woo
$myvar
$myvar
값 입력 :
안녕하세요?
$myvar = 안녕하세요?
root@server:~#
```

```
#!/bin/sh
myvar="Hi Woo"
echo $myvar
echo "$myvar"
echo '$myvar'
echo \ $myvar

echo 값 입력 :
read myvar
echo '$myvar' = $myvar
exit 0
```



```
ubuntu@server:~$ sh var1.sh
Hi Woo
Hi Woo
$myvar
$myvar
값 입력 :
안녕하세요?
$myvar = 안녕하세요?
```

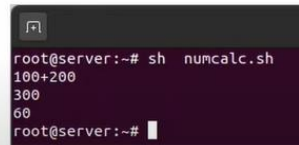
➔ vi 에디터 사용

## ● 숫자 계산

- 변수에 대입된 값은 모두 문자열로 취급

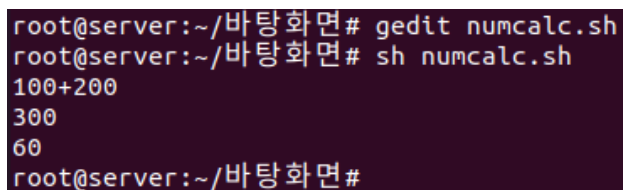
- 변수에 들어 있는 값을 숫자로 해서 +, -, \*, / 등의 연산을 하려면 expr 을 사용
- 역따옴표를 사용
- 수식에 괄호 또는 곱하기(\*)는 그 아페 꼭 역슬래쉬(\) 붙임

```
01 #!/bin/sh
02 num1=100
03 num2=$num1 + 200
04 echo $num2
05 num3=`expr $num1 + 200`
06 echo $num3
07 num4=`expr \$( $num1 + 200 \) / 10 \* 2`
08 echo $num4
09 exit 0
```



```
root@server:~# sh numcalc.sh
100+200
300
60
root@server:~#
```

```
1 #!/bin/sh
2 num1=100
3 num2=$num1+200
4 echo $num2
5 num3=`expr $num1 + 200`
6 echo $num3
7 num4=`expr \$( $num1 + 200 \) / 10 \* 2`
8 echo $num4
9 exit 0
```



```
root@server:~/바탕화면# gedit numcalc.sh
root@server:~/바탕화면# sh numcalc.sh
100+200
300
60
root@server:~/바탕화면#
```

➔ gedit 에디터 사용

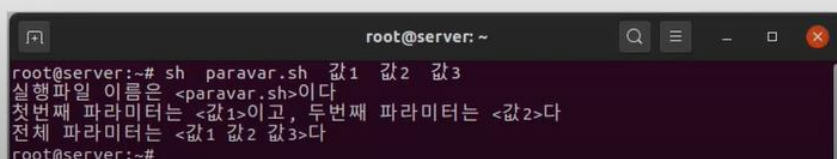
## ● 파라미터 변수

- 파라미터(Parameter) 변수는 \$0, \$1, \$2 ... 의 형태를 가짐
- 전체 파라미터는 \$\*로 표현

예)

명령	apt-get	-y	install	gftp
파라미터 변수	\$0	\$1	\$2	\$3

```
01 #!/bin/sh
02 echo "실행파일 이름은 <$0>이다"
03 echo "첫번째 파라미터는 <$1>이고, 두번째 파라미터는 <$2>다"
04 echo "전체 파라미터는 <$*>다"
05 exit 0
```



```
root@server:~# sh paravar.sh 값1 값2 값3
실행파일 이름은 <paravar.sh>이다
첫번째 파라미터는 <값1>이고, 두번째 파라미터는 <값2>다
전체 파라미터는 <값1 값2 값3>다
root@server:~#
```

```
#!/bin/sh
echo "실행 파일 이름은 <$0> 이다 "
echo "첫 번째 파라미터는 <$1>이고 , 두 번째 파라미터는 <$2>다 "
echo "전체 파라미터는 <${*}>다 "
exit 0
```

```
ubuntu@server:~$ vi paraver.sh
ubuntu@server:~$ sh paraver.sh
실행 파일 이름은 <paraver.sh> 이다
첫 번째 파라미터는 <>이고 , 두 번째 파라미터는 <>다
전체 파라미터는 <>다
ubuntu@server:~$
```

## ● 기본 if 문

### ➤ 형식

if [조건]

then

    참일 경우 실행

fi

➤ "[ 조건 ]"의 사이의 각 단어에는 모두 공백이 있어야 한다

```
#!/bin/sh
if [ "woo" = "woo" ]
then
echo "참 입 니 다 "
fi
exit 0
```

```
ubuntu@server:~$ vi if1.sh
ubuntu@server:~$ sh if1.sh
참 입 니 다
```

➤ "[ 조건 ]"의 사이의 각 단어에는 모두 공백이 있어야 한다.

## ● if~else 문

### ➤ 형식

if [ 조건 ]

then

    참일 경우 실행

➤ 중복 if 문을 위해서 else if가 합쳐진 elif문도 사용할 수 있다.

else

거짓인 경우 실행

fi

```
#!/bin/sh
if [ "woo" != "woo" ]
then
    echo "참 입 니 다 "
else
    echo "거 짓 입 니 다 "
fi
exit 0
```

```
ubuntu@server:~$ vi if2.sh
ubuntu@server:~$ sh if2.sh
거 짓 입 니 다
ubuntu@server:~$
```

- 조건문에 들어가는 비교 연산자

문자열 비교	결과
"문자열1" = "문자열2"	두 문자열이 같으면 참
"문자열1" != "문자열2"	두 문자열이 같지 않으면 참
-n "문자열"	문자열이 NULL(빈 문자열)이 아니면 참
-z "문자열"	문자열이 NULL(빈 문자열)이면 참

산술 비교	결과
수식1 -eq 수식2	두 수식(또는 변수)이 같으면 참
수식1 -ne 수식2	두 수식(또는 변수)이 같지 않으면 참
수식1 -gt 수식2	수식1이 크다면 참
수식1 -ge 수식2	수식1이 크거나 같으면 참
수식1 -lt 수식2	수식1이 작으면 참
수식1 -le 수식2	수식1이 작거나 같으면 참
수식	수식이 거짓이라면 참

```
01 #!/bin/sh
02 if [ 100 -eq 200 ]
03 then
04     echo "100과 200은 같다."
05 else
06     echo "100과 200은 다르다."
07 fi
08 exit 0
```

```
root@server:~# sh if3.sh
100과 200은 다르다.
root@server:~#
```

```
#!/bin/sh
if [ 100 -eq 200 ]
then
    echo "100과 200은 같 다 "
else
    echo "100과 200은 다 르 다 "
fi
exit 0
```

```
ubuntu@server:~$ vi if3.sh
ubuntu@server:~$ sh if3.sh
100과 200은 다 르 다
```

- 파일과 관련된 조건



파일 조건	결과
-d 파일이름	파일이 디렉터리이면 참
-e 파일이름	파일이 존재하면 참
-f 파일이름	파일이 일반 파일이면 참
-g 파일이름	파일에 set-group-id가 설정되면 참
-r 파일이름	파일이 읽기 가능이면 참
-s 파일이름	파일 크기가 0이 아니면 참
-u 파일이름	파일에 set-user-id가 설정되면 참
-w 파일이름	파일이 쓰기 가능 상태이면 참
-x 파일이름	파일이 실행 가능 상태이면 참

```
#!/bin/sh
fname=lib/systemd/system/cron.service
if [ -f $fname ]
then
    head -5 $fname
else
    echo "cron 서버가 설치되지 않았습니다"
fi
exit 0
```

```
ubuntu@server:~$ vi if4.sh
ubuntu@server:~$ sh if4.sh
cron 서버가 설치되지 않았습니다
ubuntu@server:~$
```

## ● case~esac 문

- if 문은 참과 거짓의 두 경우만 사용 (2 중분기)
- 여러 가지 경우의 수가 있다면 case 문 (다중분기)

```
#!/bin/sh
case "$1" in
start)
    echo "시작 ~~";;
stop)
    echo "종지 ~~";;
restart)
    echo "다시 시작 ~~";;
*)
    echo "뭔지 모름 ~~";;
esac
exit 0
```

```
ubuntu@server:~$ sh case1.sh stop
종지 ~~
ubuntu@server:~$ sh case1.sh start
시작 ~~
ubuntu@server:~$
```

```
#!/bin/sh
echo "리눅스가 재미있나요? (yes/no)"
read answer
case $answer in
    yes | y | Y | YES | Yes)
        echo "다행입니다"
        echo "더 열심히하세요";;
    [nN]*)
        echo "안타깝게요 ππ";;
    *)
        echo "yes or no 만 입력하세요"
        exit 1;;
esac
exit 0
```

```
ubuntu@server:~$ vi case2.sh
ubuntu@server:~$ sh case2.sh start
리눅스가 재미있나요? (yes/no)
y
다행입니다
더 열심히하세요
ubuntu@server:~$ sh case2.sh start
리눅스가 재미있나요? (yes/no)
n
안타깝게요 ππ
ubuntu@server:~$ sh case2.sh start
리눅스가 재미있나요? (yes/no)
asdf
yes or no 만 입력하세요
ubuntu@server:~$
```

## ● AND, OR 관계 연산자

- and 는 '-a' 또는 '&&'를 사용
- or 는 '-o' 또는 '||'를 사용

```
01 #!/bin/sh
02 echo "보고 싶은 파일명을 입력하세요."
03 read fname
04 if [ -f $fname ] && [ -s $fname ]; then
05     head -5 $fname
06 else
07     echo "파일이 없거나, 크기가 0입니다."
08 fi
09 exit 0
```

```
root@server: ~
root@server:~# sh andor.sh
보고 싶은 파일명을 입력하세요.
/lib/systemd/system/nofileservice
파일이 없거나, 크기가 0입니다.
root@server:~# sh andor.sh
보고 싶은 파일명을 입력하세요.
/lib/systemd/system/cron.service
[Unit]
Description=Regular background program processing daemon
Documentation=man:cron(8)
After=remote-fs.target nss-user-lookup.target
root@server:~#
```

```
#!/bin/sh
echo "보고 싶은 파일명을 입력하세요"
read fname
if [ -f $fname ] && [ -s $fname ]; then
    head -5 $fname
else
    echo "파일이 없거나 크기가 0입니다."
fi
exit 0
```

```
ubuntu@server:~$ sh andor.sh start
보고 싶은 파일명을 입력하세요
/lib/systemd/system/nofileservice
파일이 없거나 크기가 0입니다.
ubuntu@server:~$
```

## ● 반복문 – for~in 문

- 형식

for 변수 in 값 1 값 2 값 3 ...

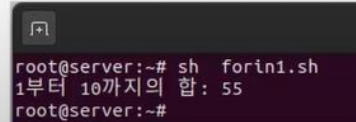
- 3행은  
for((i=1;i<=10;i++)) 또는 for i in 'seq 1 10'  
로 변경할 수 있음

do

반복할 문장

done

```
01 #!/bin/sh
02 hap=0
03 for i in 1 2 3 4 5 6 7 8 9 10
04 do
05     hap=`expr $hap + $i`
06 done
07 echo "1부터 10까지의 합: "$hap
08 exit 0
```



```
root@server:~# sh forin1.sh
1부터 10까지의 합: 55
root@server:~#
```

- 현재 디렉터리에 있는 셸 스크립트 파일(\*.sh)의 파일명과 앞 3 줄을 출력하는 프로그램

```
01 #!/bin/sh
02 for fname in $(ls *.sh)
03 do
04     echo "-----$fname-----"
05     head -3 $fname
06 done
07 exit 0
```



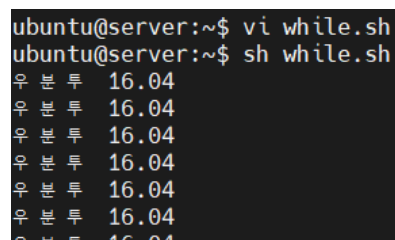
```
root@server:~# sh forin2.sh
-----andor.sh-----
#!/bin/sh
echo "보고 싶은 파일명을 입력하세요."
read fname
-----bce.sh-----
#!/bin/sh
echo "무한반복 입력을 시작합니다. (b: break, c: continue, e: exit)"
while [ 1 ]; do
-----case1.sh-----
```

- 반복문 – while 문

- 조건식이 참인 동안에 계속 반복

➤ [1] 또는 [:]가 오면 항상 참이 됨. 그러므로 4행을 무한 루프로 반복함.

```
#!/bin/sh
while [ 1 ]
do
    echo "우 분 투 16.04"
done
exit 0
```



```
ubuntu@server:~$ vi while.sh
ubuntu@server:~$ sh while.sh
우 분 투 16.04
우 분 투 16.04
우 분 투 16.04
우 분 투 16.04
우 분 투 16.04
우 분 투 16.04
우 분 투 16.04
우 분 투 16.04
```

- 1 에서 10 까지의 합계를 출력 ('반복문 - for' 내용과 동일)

```
01 #!/bin/sh
02 hap=0
03 i=1
04 while [ $i -le 10 ]
05 do
06     hap=`expr $hap + $i`
07     i=`expr $i + 1`
08 done
09 echo "1부터 10까지의 합 : "$hap
10 exit 0
```

> until 문은 조건식이 참일 때까지(=거짓인 동안) 계속 반복  
> 4행을 until 문으로 바꾸면, until [ \$i -gt 10 ]

```
root@server:~# sh while2.sh
1부터 10까지의 합 : 55
root@server:~#
```

- 비밀번호를 입력받고, 비밀번호가 맞을 때까지 계속 입력받는 스크립트

```
01 #!/bin/sh
02 echo "비밀번호를 입력하세요."
03 read mypass
04 while [ $mypass != "1234" ]
05 do
06     echo "틀렸음. 다시 입력하세요."
07     read mypass
08 done
09 echo "통과~~"
10 exit 0
```

```
root@server:~# sh while3.sh
비밀번호를 입력하세요.
3333
틀렸음. 다시 입력하세요.
4444
틀렸음. 다시 입력하세요.
1234
통과~~
root@server:~#
```

```
#!/bin/sh
echo "비밀번호를 입력하세요"
read mypass
while [ $mypass != "1234" ]
do
    echo "틀림 다시 입력하세요"
    read mypass
done
echo "통과 ~"
exit 0

ubuntu@server:~$ vi while3.sh
ubuntu@server:~$ sh while3.sh start
비밀번호를 입력하세요
3333
틀림 다시 입력하세요
1255
틀림 다시 입력하세요
1234
통과 ~
ubuntu@server:~$
```

## ● until 문

- while 문과 용도가 거의 같지만, until 문은 조건식이 참일때까지(=거짓인 동안) 계속 반복한다
- while2.sh 를 동일한 용도로 until 문으로 변경하려면 4 행을 다음과 같이 변경하면 된다.

✓ until [ \$i -gt 10 ]

- break, continue, exit, return 문

- break 는 주로 반복문을 종료할 때 사용되며, continue 는 반복문의 조건식으로 돌아가게 함. exit 는 해당 프로그램을 완전히 종료함. return 은 함수 안에서 사용될 수 있으며 함수를 호출한 곳으로 돌아가게 함

- 사용자 정의 함수

- 형식  
함수이름 () { → 함수를 정의  
내용들...  
}  
함수이름 → 함수를 호출

```
01 #!/bin/sh
02 myFunction () {
03     echo "함수 안으로 들어 왔음"
04     return
05 }
06 echo "프로그램을 시작합니다."
07 myFunction
08 echo "프로그램을 종료합니다."
09 exit 0
```

- 함수의 파라미터 사용

- 형식

함수이름 () { → 함수를 정의

\$1, \$2 ... 등을 사용

}

함수이름 파라미터 1 파라미터 2 ... → 함수를 호출

- eval

- 문자열을 명령문으로 인식하고 실행

```
#!/bin/sh
str="ls -l eval.sh"
echo $str
eval $str
exit 0
```

```
ubuntu@server:~$ vi eval.sh
ubuntu@server:~$ sh eval.sh start
ls -l eval.sh
-rw-rw-r-- 1 ubuntu ubuntu 57  3월 18 14:02 eval.sh
ubuntu@server:~$
```

- export

- 외부 변수로 선언해 준다. 즉, 선언한 변수를 다른 프로그램에서도 사용할 수 있도록 해줌

```
• exp1.sh
01 #!/bin/sh
02 echo $var1
03 echo $var2
04 exit 0

• exp2.sh
01 #!/bin/sh
02 var1="지역 변수"
03 export var2="외부 변수"
04 sh exp1.sh
05 exit 0
```

```
root@server:~# sh exp2.sh
외부 변수
root@server:~#
```

- printf

- c 언어의 printf() 함수와 비슷하게 형식을 지정해서 출력

```
#!/bin/sh
var1=100.5
var2="재 미 있 는 이 룩 스 "
printf "%5.2f \n\n \t %s \n" $var1 "$var2"
exit
```

```
ubuntu@server:~$ vi printf.sh
ubuntu@server:~$ sh printf.sh start
100.50
재 미 있 는 이 룩 스
ubuntu@server:~$
```

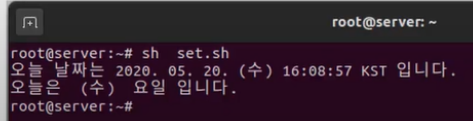
- set 과 \$(명령어)

- 리눅스 명령어를 결과로 사용하기 위해서는 \$(명령어) 형식을 사용
- 결과를 파라미터로 사용하고자 할 때에는 set 과 함께 사용

```

01 #!/bin/sh
02 echo "오늘 날짜는 $(date) 입니다."
03 set $(date)
04 echo "오늘은 $4 요일 입니다."
05 exit 0

```



```

root@server: ~
root@server:~# sh set.sh
오늘 날짜는 2020. 05. 20. (수) 16:08:57 KST 입니다.
오늘은 (수) 요일 입니다.
root@server:~#

```

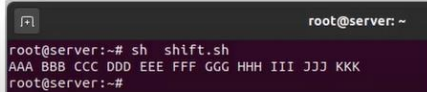
- shift

- 파라미터 변수를 왼쪽으로 한 단계씩 아래로 쉬프트(이동) 시킨다

```

01 #!/bin/sh
02 myfunc() {
03     str=""
04     while [ "$1" != "" ]; do
05         str="$str $1"
06         shift
07     done
08     echo $str
09 }
10 myfunc AAA BBB CCC DDD EEE FFF GGG HHH III JJJ KKK
11 exit 0

```



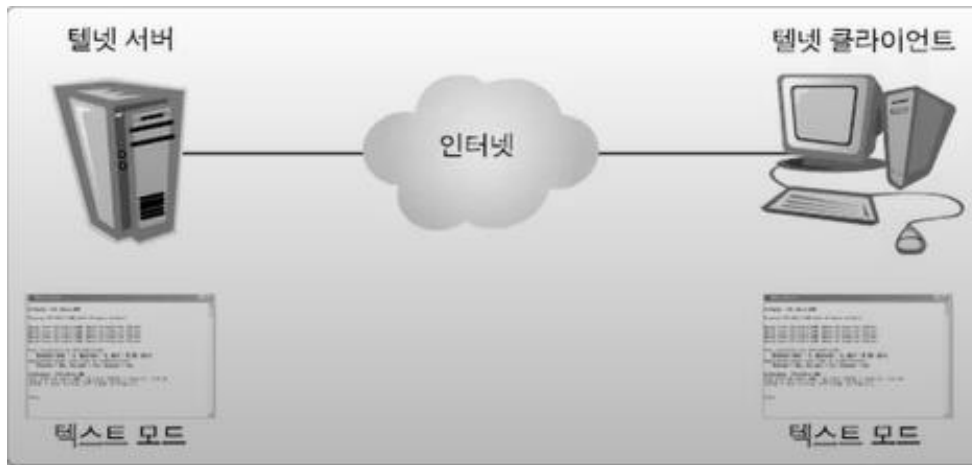
```

root@server: ~
root@server:~# sh shift.sh
AAA BBB CCC DDD EEE FFF GGG HHH III JJJ KKK
root@server:~#

```

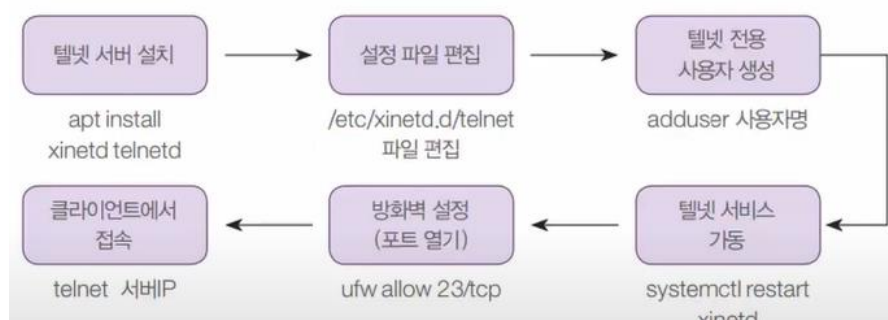
- 텔넷 서버 개요

- 오랫동안 전통적으로 사용되어 온 원격 접속 방법
- 보안에 취약
- 리눅스 서버에 텔넷 서버를 설치하고 나면, 원격지에서 접속할 pc 에는 텔넷 클라이언트 프로그램이 필요
- 원격지의 pc(텔넷 클라이언트)에서 접속하게 되면 서버 앞에 앉아서 직접 텍스트 모드로 작업하는 것과 완전히 동일한 효과



## ● 텔넷 서버 구축

- 원격지에서 서버 접속할 경우 필요
- 텔넷 서버 설치 과정 요약



- 서버에 접속하기 위해서는 꼭 클라이언트 프로그램 필요
- 서버가 리눅스라고 클라이언트도 리눅스일 필요는 없음
- 각각의 서버 프로그램은 자신에 맞는 별도의 클라이언트 프로그램이 필요w

```

root@server:~/바탕화면# apt -y install xinetd telnetd
패키지 목록을 읽는 중입니다... 완료
이중 선택을 만드는 중입니다
  
```

➔ 텔넷 서버 설치 (apt -y install xinetd telnetd)



```

telnet
/etc/xinetd.d
1 service telnet
2 {
3     disable = no
4     flags = REUSE
5     socket_type = stream
6     wait = no
7     user = root
8     server = /usr/sbin/in.telnetd
9     log_on_failure += USERID
10 }

```

➔ 설정 파일 편집 (/etc/xinetd.d/telnet 파일 편집)

```

root@server:/etc/xinetd.d# adduser teluser
'teluser' 사용자를 추가 중...
새 그룹 'teluser' (1001) 추가 ...
새 사용자 'teluser' (1001) 을(를) 그룹 'teluser' (으)로 추가 ...
'/home/teluser' 홈 디렉터리를 생성하는 중...
'/etc/skel'에서 파일들을 복사하는 중...
새 암호:
새 암호 재입력:
passwd: 암호를 성공적으로 업데이트했습니다

```

➔ 텔넷 전용 사용자 생성

➤ root 사용자로 이용하면 위험하기 때문에 adduser 명령을 사용하여 생성 (teluser, passwd= teluser 생성)

```

root@server:/etc/xinetd.d# systemctl restart xinetd
root@server:/etc/xinetd.d#

```

➔ 텔넷 서비스 가동 // systemctl restart xinetd

```

root@server:/etc/xinetd.d#
root@server:/etc/xinetd.d# ufw allow 23/tcp
규칙이 추가되었습니다
규칙이 추가되었습니다 (v6)
root@server:/etc/xinetd.d#

```

➔ 방화벽 설정(포트 열기) // ufw allow 23/tcp (텔넷은 23 번 포트 사용)

```

root@server:/etc/xinetd.d# telnet 192.168.111.100
Trying 192.168.111.100...
Connected to 192.168.111.100.
Escape character is '^]'.
Ubuntu 20.04 LTS
server login: teluser
Password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.13.0-30-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 updates can be installed immediately.
0 of these updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2025.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

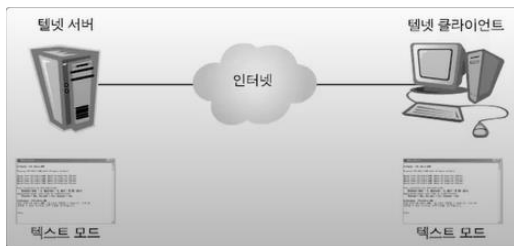
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

teluser@server:~$

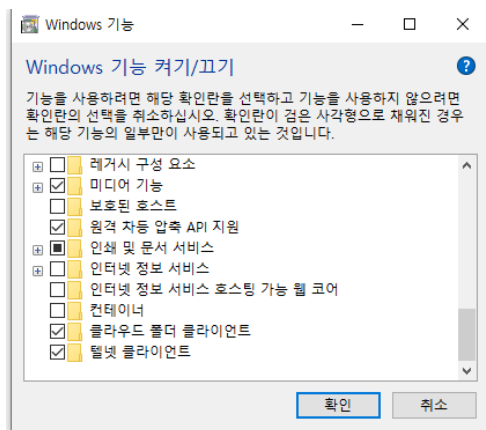
```

## ➔ 서버 컴퓨터로 접속 테스트

- 서비스 작동 상태를 확인하기 위해 서버 자기 자신에게 접속 테스트를 진행 함으로써 문제가 발생할 때 폭을 줄일 수 있다.



- 서버에 클라이언트가 접속하기 위해선 그 서버 전용의 클라이언트가 필요하다  
(ex, 텔넷서버 < 텔넷클라이언트 웹서버 < 웹클라이언트 , db 서버 < db 클라이언트 등)



## ➔ winclient 에서 텔넷 클라이언트 적용

- 앱 및 기능 > 프로그램 기능 > windows 기능에서 텔넷 클라이언트 확인 후 재부팅

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:\Windows\system32> ping 192.168.111.100

Ping 192.168.111.100 32바이트 데이터 사용:
192.168.111.100의 응답: 바이트=32 시간<1ms TTL=64
192.168.111.100의 응답: 바이트=32 시간<1ms TTL=64
192.168.111.100의 응답: 바이트=32 시간<1ms TTL=64
192.168.111.100의 응답: 바이트=32 시간<1ms TTL=64

192.168.111.100에 대한 Ping 통계:
    패킷: 보낸 = 4, 받음 = 4, 손실 = 0 (0% 손실),
    왕복 시간(밀리초):
        최소 = 0ms, 최대 = 0ms, 평균 = 0ms
PS C:\Windows\system32>
```

## ➔ windows powershell 에서 텔넷 서버로 ping 응답 확인

- ping [ip 주소]

```
PS C:\Windows\system32> telnet 192.168.111.100
```

➔ 클라이언트에서 접속

```

ubuntu 20.04 LTS
server login: teluser
Password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.13.0-30-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 updates can be installed immediately.
0 of these updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Fri Mar 18 15:43:55 KST 2022 from server on pts/1
teluser@server:~$
teluser@server:~$ whoami
teluser
teluser@server:~$ ifconfig ens32
ens32: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.111.100 netmask 255.255.255.0 broadcast 192.168.111.255
    inet6 fe80::e5c8:f207:19c5:749a prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:ac:0b:68 txqueuelen 1000 (Ethernet)
    RX packets 405382 bytes 605195059 (605.1 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 19949 bytes 1301898 (1.3 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

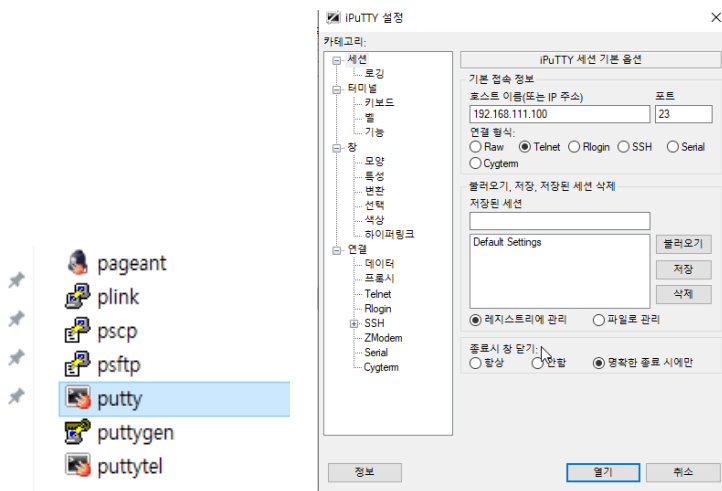
teluser@server:~$

```

➔ 접속 후 사용자 정상 확인 // username: teluser passwd: teluser

```
teluser@server:~$ exit
濡엇렬? 끝쑔
호스트에 대한 연결을 잃었습니다.
PS C:\Windows\system32>
```

➔ 텔넷 클라이언트를 windows 용을 사용하면 한글이 깨지는 상황



```

teluser@server: ~
Ubuntu 20.04 LTS
server login: teluser
Password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.13.0-30-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 updates can be installed immediately.
0 of these updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Fri Mar 18 16:25:29 KST 2022 from 192.168.111.131 on pts/1
teluser@server:~$ ls
bin  cdrom  etc  lib  lib64  lost+found  mnt  proc  run  snap  sys  usr
boot  dev  home  lib32  libx32  media  opt  root  sbin  srv  tmp  var
teluser@server:~$

```

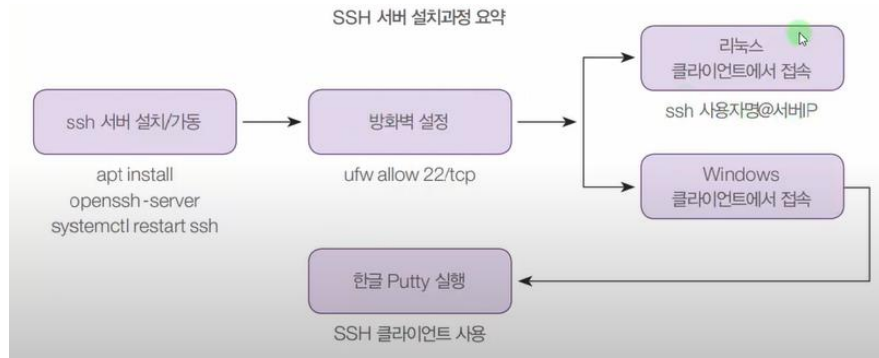
➔ putty 사용하여 한글 깨짐 없이 사용 가능 (필수 x)

## ● OpenSSH 서버

- 텔넷과 용도는 동일하지만, 보안이 강화
- 텔넷과 거의 동일하지만 데이터를 전송할 때 암호화를 한다는 점이 다름



- 원격지에서 보안이 강화된 서버 접속할 경우 필요
- Opeenssh 서버 설치 과정 요약



```
root@server:~# apt install openssh-server
```

➔ ssh 서버 설치 // apt install openssh-server

```
root@server:~# systemctl restart ssh
root@server:~#
```

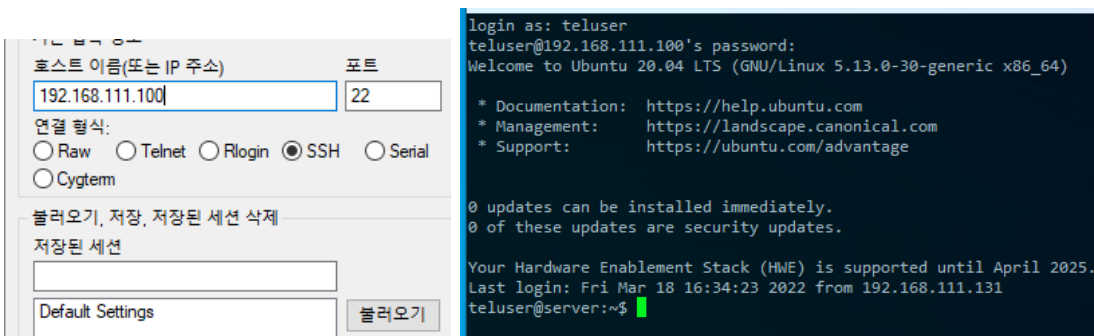
➔ ssh 서버 재부팅(적용) // systemctl restart ssh

```
root@server:~# systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.s
   Active: active (running) since Fri 2022-0
     Docs: man:sshd(8)
           man:sshd_config(5)
   Process: 5972 ExecStartPre=/usr/sbin/sshd
           5973 /usr/sbin/sshd
```

➔ ssh 서버 상태 확인 // systemctl status ssh

```
root@server:~# ufw allow 22/tcp
규칙이 추가되었습니다
규칙이 추가되었습니다 (v6)
root@server:~#
```

➔ 방화벽 설정 // ufw allow 22/tcp (openssh 은 22 번 포트)



➔ windows 클라이언트에서 접속 (본인은 putty 사용했으나 터미널도 같음)

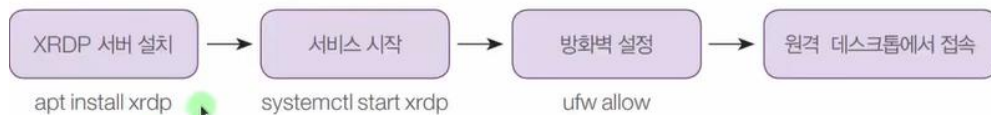
## ● XRDP 서버

➤ X 윈도우 환경으로 원격접속을 사용하고 싶을 때

- Windows 의 '원격 데스크톱 연결' 프로그램을 사용해서 리눅스에 그래픽 환경으로 접속



- 원격지에서 X 윈도우 모드로 접속할 경우 필요
- XRDp 서버 설치 과정 요약



```
root@server:~# apt -y install xrdp
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
```

- ➔ XRDp 서버 설치 // apt install xrdp

```
root@server:~# systemctl restart xrdp
root@server:~# systemctl status xrdp
● xrdp.service - xrdp daemon
   Loaded: loaded (/lib/systemd/system/xrdp.service; vendor preset: enabled)
   Active: active (running) since Fri 2023-08-11 10:00:00 KST; 1min ago
     Docs: man:xrdp(8)
```

- ➔ 서비스 시작

- xrdp 서버 재부팅 (systemctl restart xrdp)
- xrdp 서버 상태 확인 (systemctl status xrdp)

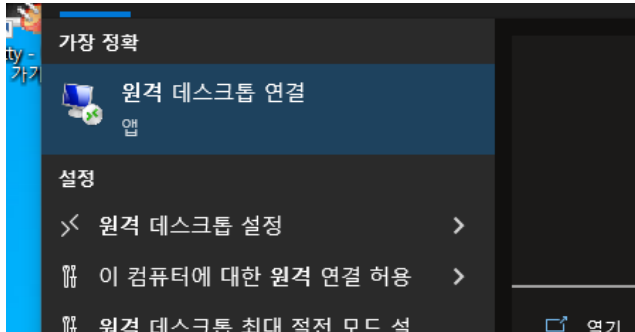
```
root@server:~# systemctl enable xrdp
Synchronizing state of xrdp.service with SysV init script by running 'systemctl start xrdp'
Executing: /lib/systemd/systemd-sysv-in
root@server:~#
```

- ➔ 항상 가동 적용

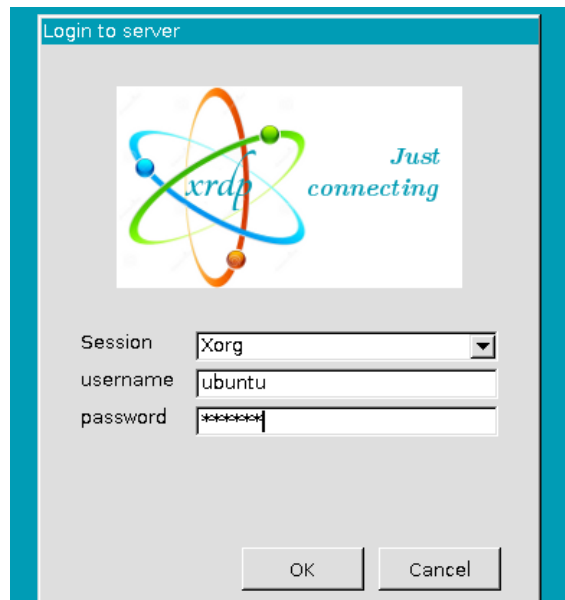
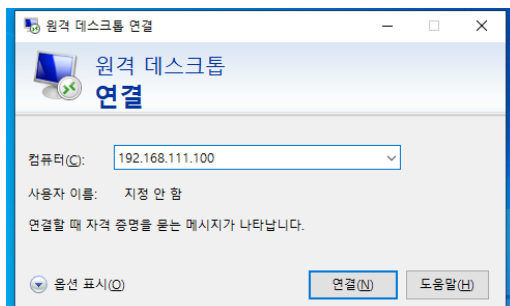
- 전원이 꺼지면 서버도 off 되어 다시 위와 같은 과정을 거쳐야 하기 때문에 (보통 서버는 항상 가동) systemctl enable xrdp 명령어로 항상 가동을 적용

```
root@server:~# ufw allow 3389/tcp
규칙이 추가되었습니다
규칙이 추가되었습니다 (v6)
root@server:~#
```

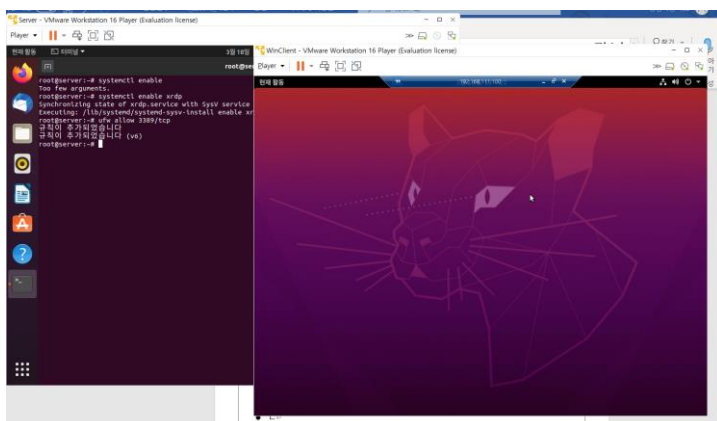
➔ 방화벽 설정 (ufw allow 3389/tcp) // xrdp 는 3389 포트



➔ Windows 에서 원격 데스크톱 연결



➔ ip 연결 후 로그인 (name: ubuntu passwd: ubuntu)



➔ 원격 연결 성공

- 3 가지 원격 접속 서버의 비교

- 비교표

	텔넷 서버	SSH 서버	XRDP 서버
속도	빠르다.	빠르다.	약간 느리다.
그래픽 지원	지원하지 않는다.	지원하지 않는다.	지원한다.
보안	취약하다.	강하다.	
사용 가능 명령	텍스트 모드의 명령만 사용할 수 있다.	텍스트 모드의 명령만 사용할 수 있다.	제한 없다.
클라이언트 프로그램	대부분의 운영체제에 기본적으로 있다.	리눅스는 기본적으로 있다. Windows는 별도 설치해야 한다.	Windows에 포함되어 있다.

- 결론

- ✓ SSH 를 기본적으로 사용하고, XRDP 서버는 설정만 해놓고 가동하지 않는다. 원격지에서 SSH 로 서버를 관리하다가, X 윈도우 접속이 필요할 경우에는 접속된 SSH 접속 창에서 XRDP 서버를 구동하고 Windows 의 원격 데스크톱으로 접속해서 사용
- ✓ 텔넷 서버는 보안이 철저한 회사 내의 네트워크에서만 사용하면 무난함