3 쉘 프로그래밍

3-1.쉘 프로그래밍 개요

3-2.쉘프로그램의 전반적 구조

변수
흐름제어
함수
내장 쉘명령

3-3.변수

ex)echoex.sh

```
MYNAME="Sang-Yeob Lee"
echo '${MYNAME} yields: ' ${MYNAME}_
echo 'SMYNAME_yields: ' $MYNAME_
```

출력

```
${MYNAME} yields:  Sang-Yeob Lee_
SMYNAME_yields:
```

ex)echoex2.sh

```
MYNAME="Sang-Yeob Lee"
echo \$MYNAME
echo '${MYNAME} yields: ' ${MYNAME}_
echo 'SMYNAME_yields: ' $MYNAME_
```

출력

```
$MYNAME
${MYNAME} yields:  Sang-Yeob Lee_
SMYNAME_yields:
```

표1 환경변수 리스트

환경변수	설명
$HOME	현재 사용자의 홈디렉토리
$PATH	명령을 검색하는 디렉토리들의 목록
$PS1	대개 $인 명령 프롬포트
$PS2	추가적인 입력을 요구할 때 사용되는 2차 프롬포트 일반적으로 >
$IFS	입력 필드 구분자, 쉘이 입력을 받아들일 때 단어를 구분하는데 사용되는 문자의 목록으로, 빈칸,탭,new line
$#	전달되는 파리키터의수
$*	모든 파라미터
$@	IFS 환경변수를 사용하지 않는 $* 다른 형태
$1, $2...	파라미터 첫 번째,두번째 .. 문자열

ex)echoex3.sh

```
echo HOME= $HOME
echo PATH= $PATH
echo PS1= $PS1
echo PS2= $PS3
echo IFS= $IFS
```

출력

```
HOME= /root
PATH= ::/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/usr/kerberos/bin:/home/syle
e/bin:/usr/kerberos/bin
PS1=
PS2=
IFS=
```

ex)exparm1.sh

```
cho -e "first parameter string is $1"
echo -e "second parameter string is $2"
echo -e "all parameter string is $*"
echo -e "parameter num is $#"
```

출력

```
bash# ./exparm1.sh my name
first parameter string is my
second parameter string is name
all parameter string is my name
parameter num is 2
```

ex)exparm2.sh

```
function analysisparm
{
```

```
        echo -e "parameter num $#"
}

analysisparm "$*"
analysisparm "$@"
echo -e "parameter $@"
```

출력

```
bash# ./exparm2.sh my name
parameter num 1
parameter num 2
parameter my name
```

중괄호 확장

```
bash# echo c{ar,at,an,on}s
cars cats cans cons

bash# echo c{a{r,t,n},on}s
cars cats cans cons
```

3-4.흐름제어

3-4-1 if 구문

```
if 조건
then
합당구문
else
그밖에 구문
fi
```

ex)ifex.sh

```
if test -f test.c
then
cat ./test.c
else
echo "test.c file not found"
fi
```

출력

[root@cyberkorean bashprogramming]# ./ifex1.sh
hi! my name is Sang-Yeob Lee

ex)ifex2.sh
```
if [ -f test.c ]
then
cat ./test.c
else
echo "test.c file not found"
fi
```

출력
[root@cyberkorean bashprogramming]# ./ifex2.sh
hi! my name is Sang-Yeob Lee

표2 문자열 비교

형식	설명
string1 = string2	두문자열이 같으면
string1 != string2	두분자열이 같지 않으면
-n string	문자열이 NULL 아니면
-z string	문자열이 NULL 이면

표3 파일조건

형식	설명
-d file	파일이 디렉토리 이면
-e file	파일이 존재하면
-f file	파일이 일번적인 파일이면
-g file	파일에서 set-group-id 가 설정되어 있으면
-r file	파일이 읽기 가능상태이면
-s file	파일이 0이 아닌 크기를 가지면
-u file	파일에서 set-user-id 가 설정되면
-w file	파일이 쓰기 가능 상태이면
-x file	파일이 실행가능 상태이면

표4 산술비교

형식	설명
수식1 -eq 수식2	두 수식이 같으면
수식1 -ne 수식2	두 수식이 같지 않으면
수식1 -gt 수식2	수식 1 이 수식 2보다 크면
수식1 -ge 수식2	수식 1 이 수식 2보다 크거나 같으면
수식1 -lt 수식2	수식 1 이 수식 2보다 작으면
수식1 -le 수식2	수식 1 이 수식 2보다 작거나 같으면
! 수식	수식이 참이 아니면

ex)ifex3.sh
```
STR=$1
```

```
if [ $STR = "lsy" ]
then
cat ./test.c
else
echo "test.c file not found"
fi
```

출력

```
[root@cyberkorean bashprogramming]# ./ifex3.sh lsy
hi! my name is Sang-Yeob Lee
```

3-4-2 elif 구문

비교후에 결과중에서 조건에 맞지 않는 것을 재비교

ex)ifex4.sh

```
STR=$1
if [ $STR = "lsy" ]
then
cat ./test.c
elif [ $STR = "sylee" ]
then
echo "hi! sylee"
else
echo "test.c file not found"
fi
```

출력

```
bash# ./ifex4.sh sylee
hi! sylee
```

3-4-3 for 구문

```
for 변수 in 설정값들 ...
do
  명령문
done
```

ex)forex1.sh

```
for count in one two three four
do
  echo $count
done
```

출력
bash# ./forex1.sh
one
two
three
four

ex)forex2.sh
```
for file in $(ls ex*.sh)
do
  echo $file
done
```

출력
bash# ./forex2.sh
exparm1.sh
exparm2.sh

3-4-4 while 구문
while 조건 do
명령문
done

ex)whileex1.sh
```
echo "Enter password"
read passtr

while [ "$passtr" != "aiai" ]
do
echo "Sorry! try again"
read passtr
done
```

출력
ash# ./whileex1.sh
Enter password
sdkajf
Sorry! try again
sadlkfjla
Sorry! try again
aiai

3-4-5 untile 구문

while 와 조건이 반대 형태

ex)untilex1.sh

```
echo "Enter password"
read passtr

until [ "$passtr" = "aiai" ]
do
echo "Sorry! try again"
read passtr
done
```

출력

```
[root@cyberkorean bashprogramming]# ./untilex1.sh
Enter password
skladfj
Sorry! try again
aiai
```

3-4-6 case 구문

형식

```
case 변수 in
pattern )
:
esac
```

ex)caseex1.sh

```
echo "Delete this file (yes,no)"
read passtr

case "$passtr" in
        yes) echo "delete file";;
        no) echo "not delete";;
        *) echo "only yes or no";;
esac
```

결과

```
[root@cyberkorean bashprogramming]# ./caseex1.sh
Delete this file (yes,no)
```

```
yes
delete file

ex)caseex2.sh
echo "Delete this file (yes,no)"
read passtr

case "$passtr" in
        yes | y | Yes | YES ) echo "delete file";;
        no) echo "not delete";;
        *) echo "only yes or no";;
esac
```

결과

```
[root@cyberkorean bashprogramming]# ./caseex1.sh
Delete this file (yes,no)
yes
delete file
[root@cyberkorean bashprogramming]# ./caseex2.sh
Delete this file (yes,no)
y
delete file
[root@cyberkorean bashprogramming]# ./caseex2.sh
Delete this file (yes,no)
Yes
delete file
[root@cyberkorean bashprogramming]# ./caseex2.sh
Delete this file (yes,no)
dskafj
only yes or no
```

3-4-7 논리 연산

AND 연산 : &&
OR 연산 : ||

```
ex)andex1.sh
echo "insert string"
read passtr
echo "insert string"
read passtr1
if [ "$passtr" = "yes" ] && [ "$passtr1" = "no" ]
```

```
then
echo "you writed yes and no";
else
echo $passtr
echo $passtr1
fi
```

출력

```
bash# ./andex1.sh
insert string
yes
insert string
no
you writed yes and no
```

ex)

```
echo "insert string"
read passtr
echo "insert string"
read passtr1
if [ "$passtr" = "yes" ] || [ "$passtr1" = "no" ]
then
echo "good!";
else
echo $passtr
echo $passtr1
fi
```

출력

```
bash# ./orex1.sh
insert string
yes
insert string
lskdajf
good!
```

3-4-8 break, continue

3-5.함수만들기와 함수들

3-5-1 함수만들기

파라미터 예제 참조
하나 만들어 보기

3-5-2 내장함수들

echo
화면에 문자열 출력

exec
다은 프로그램을 실행

exit
프로그램 종료

set
파라미터 설정