10강. 쉘 스크립트

● 쉘 스크립트
• 쉘은 명령어들을 연속적으로 실행하는 인터프리터 환경을 제공
• 쉘 스크립트는 제어문과 변수 선언 등이 가능하며 프로그래밍 언어와 유사
• 프로그래밍 언어와 스크립트 언어
-프로그래밍 언어를 사용하는 경우 소스 코드를 컴파일 하여 실행 가능한 파일로 만들어야 함
-일반적으로 실행 파일은 다른 운영 체제로 이식되지 않음
-스크립트 언어를 사용하면 컴파일 과정이 없고 인터프리터가 소스 파일에서 명령문을 판독하여 각각의 명령을 수행

● 쉘 스크립트 언어
• 일반적으로 '.sh'라는 확장자를 가짐
01 : #!/bin/bash
02 : echo Hello Linux
• 01 : #!/bin/bash 는 해당 스크립트를 실행하기 위한 쉘의 경로
• 02 : 실제 명령
• 스크립트 파일에 실행 권한을 부여해야 실행 가능
◦ chmod u+x myscript.sh
• './파일명'으로 실행
◦ 예: ./myscript.sh

● 쉘 스크립트 예
• mydir 파일을 작성한 후 실행권한을 부여하고 실행함
• 'vi mydir'로 파일을 편집 (i로 입력모드, <Esc>로 명령모드, ZZ)
```
#!/bin/bash
ls -l * > ~/mydir.txt
cat ~/mydir.txt | more
```
• 실행권한 부여
◦ chmod 'u+x,g+x' mydir 또는 chmod +x mydir
• 실행
◦ ./mydir

● 리다이렉션(redirection)
• 리다이렉션은 입력과 출력의 방향을 파일로 바꾸는 것
◦ 표준입력(stdin), 표준출력(stdout), 표준에러(stderr)
• 표준입력을 파일로 지정 : < 또는 << 사용
• 표준출력을 파일로 지정 : > 또는 >> 사용
◦ 예: cat x y > hold

- 표준에러를 파일로 지정
  - 예: cat x y 2> hold
- 표준에러를 파일로, 표준출력을 표준에러로 보내기
  - 예: cat x y 2> hold 1>&2

● 쉘 스크립트와 변수
- 환경 변수
  - 초기 설정에 대한 정보를 저장하는 변수 : HOME, PATH 등
- 특수 변수
  - 쉘 프로그램에 전달되는 특수 변수
  - $ {인수번호} : 명령 행 인수(argument)
  - $? : 직전 명령의 상태 값
  - $# : 명령 행 인수의 개수
  - 예 : echo "$0 $1" 또는 ls -l $ {1 } > ~/mydir.txt
- 프로그램 변수
  - 쉘 스크립트의 변수는 필요 시 선언하여 사용할 수 있음
  - 변수의 데이터 타입(숫자, 문자, 문자열)을 정할 필요 없음

● 변수 선언
- 변수명 = 값

```
#!/bin/bash
STR="Have a good day!"
echo $STR
```

● 조건문
- 조건이 만족되면 수행(if~)

```
#!/bin/bash
a=10
b=20
if [ $a -lt $b ]; then
      echo $b
fi
```

- 조건이 만족되거나 조건이 만족되지 않을 때(if~ else)

```
#!/bin/bash
if [ $ {1 } -gt $ {2 } ]; then
     echo $ {1 }
else
     echo $ {2 }
fi
```

- 여러 조건이 주어지거나 아무 조건도 만족되지 않을 때(if~ elif~ else)

```
#!/bin/bash
```

```
if [ $ {1 } -lt 1 ]; then
    echo "Less than 1"
elif [ $ {1 } -gt 3 ]; then
    echo "Greater than 3"
else
    if [ $ {1 } -eq 1 ]; then
        echo "one"
    elif [ $ {1 } -eq 2 ]; then
        echo "two"
    elif [ $ {1 } -eq 3 ]; then
         echo "three"
    fi
fi
```

● 조건 검사
• test 조건식 또는 [조건식]

● 정수 비교 조건식
• A -eq B : A와 B의 값이 같음
◦ 예: if [$# -eq 0] ($# 은 명령행 인수의 개수)
• A -ne B : A와 B의 값이 같지 않음
• A -gt B : A가 B보다 큼
• A -lt B : A가 B보다 작음
• A -ge B : A가 B보다 크거나 같음
• A -le B : A가 B보다 작거나 같음

● 문자열 비교 조건식
• string1 = string2 : string1과 string2가 같음 (또는 ==)
◦ 예1 : if [$1 = hi]
◦ 예2 : if test $1 == "hi"
• string1 != string2 : string1과 string2가 같지 않음
◦ 예 : if [$1 != 'bye']
• string : string이 정의되어 있고 NULL 이 아님
• -z string : string이 정의되어 있지 않거나 NULL 임

● 파일 비교 조건식
• -e file : file이 존재함
• -f file : file이 존재하며 일반 파일이고 디렉터리가 아님
• -d dir : dir이 존재하며 디렉터리임
• -r(또는 w, x) file : file이 존재하며 읽기(또는 쓰기, 실행) 가능

● 논리 연산
- !표현식 : NOT을 표현
- 조건식1 -a 조건식2 : AND을 표현
- 조건식1 -o 조건식2 : OR을 표현

● 따옴표의 의미
- '…' : 안의 내용을 해석하지 않음. 특수 문자를 일반 문자로 취급
  - 예 : echo '"$PATH"' (“ ”안에 적힌 문자열이 그대로 출력됨)
- "…" : 마찬가지로 특수 문자를 일반 문자로 취급. 단, 특수 문자 $, \, ` 는 해석함
  - 예: echo "'$PATH'" (PATH에 설정된 경로가 ‘ ’ 내부에 출력됨)
- `…` 또는 $(…) : 안의 내용을 실행(command substitution)
  - 예1 : echo "Today is `date`“ 또는 : echo "Today is $(date)"
  - 예2 : echo $(expr 3 * 7)

● 연산자
- 산술 연산자
- +, -, *, /, % : 더하기, 빼기, 곱하기, 나누기, 나머지
  - 예: sum = $(expr $a + $b)
  - 예: mul = $(expr $a * $b) <- 백슬래시를 사용해야 함
  - 예: mod = `expr $a % $b`
- 논리 연산자
- !, &&, || : NOT, AND, OR
  - 예: if [$# -gt 0] && [$# -lt 4]
  - 예: if [$# -gt 0 -a $# -lt 4] <- 대괄호 안에서 -a를 사용함

● 순환문
- do ~ done 사이의 구문을 지정한 조건만큼 반복
- 순환문의 종류 : for, while, until
- for 문 : 각 원소에 대해 구문을 수행
- while 문 : 조건이 참인 동안 구문을 수행
- until 문 : 조건이 거짓인 동안 구문을 수행

● 순환문 - for 문

```
for 변수 in list1 list2 list3 또는 for ((초기값; 조건; 증가식))
do
    문장(들)
done
```

- for 문 조건의 예
  - 예1 : for num in 1 2 3 4 5 6 7 8 9
  - 예2 : for ((num=1; num<=9; num++))
- for 순환문 사용 예

```
#!/bin/bash
if [ $# -eq 0 ]; then
    echo "Usage: $0 integer_number"
else
    # for num in 1 2 3 4 5 6 7 8 9
    for ((num= 1; num<= 9; num++) )
    do
        echo " $1 * $num = ` expr $1 \* $num ` "
    done
fi
```

● 순환문 - while 문

```
while 조건
do
    문장(들)
done
```

- while 문 조건의 예
  - 예1 : while [$num -le 9]
  - 예2 : while test $num -le 9
- while 순환문 사용 예

```
#!/bin/bash
if [ $# -eq 0 ]; then
    echo "Usage: $0 integer_number"
else
    num=1
    while [ $num -le 9 ]
    do
        echo " $1 * $num = $(expr $1 \* $num) "
        num=`expr $num + 1`
    done
fi
```

● 순환문 - until 문

```
until 조건
do
    문장(들)
done
```

- until 문 조건의 예
  - 예1 : until [$num -gt 9]
  - 예2 : until test $num -gt 9
- until 문 사용 예

```
#!/bin/bash
if [ $# -eq 0 ]; then
    echo "Usage: $0 integer_number"
else
    num=1
    until [ $num -gt 9 ]
    do
        echo " $1 * $num = $(expr $1 \* $num) "
        num=` expr $num + 1 `
    done
fi
```

● 함수

• 특정한 일을 수행하는 기능이 구현된 코드 블록

• 함수 정의

```
function 함수명
{
        문장(들)
}
```

• 함수 호출 방법

◦ 함수명 또는 함수명 인자(들)

• 함수 사용 예

```
#!/bin/bash
function mult()
{
    for ((i=1; i<=9; i++))
    do
        echo " $1 * $i = `expr $1 \* $i` "
    done
}
if [ $# -eq 0 ]; then
    for ((num=1; num<=9; num++))
    do
        mult $num
    done
else
    mult $1
fi
```

● 디버깅

• 쉘 스크립트의 첫 번째 행에 -x를 추가

◦ #!/bin/bash -x

- 실행 과정을 자세히 보여줌

● 다음 문제의 정답을 고르시오.

13. BASH 쉘에서 다음 중 abc라는 쉘 스크립트를 디버깅 모드에서 실행하는 옵션은?

① sh -d abc ② sh -c abc

③ sh -s abc ④ sh -x abc

● 다음 문제에 대한 정답을 서술하시오.

16. 다음과 같이 입력한 숫자대로 삼각형을 출력하는 쉘 스크립트를 작성하시오.

Number : 4

****

***

**

*

```
#!/bin/bash
echo -n "Input : "
read num
while [ 1 -le $num ]
do
num2=1
while [ $num2 -le $num ]
do
echo -n "*"
let num2=num2+1
done
echo
let num=num-1
done
```

17. 점수를 입력받아 입력한 점수에 대하여 학점(A, B, C, D, F)으로 환산해 주는 쉘 스크립트를 작성하시오.

※ 점수의 처리 조건

A 학점 : 90점 이상

B 학점 : 80점 이상

C 학점 : 70점 이상

D 학점 : 60점 이상

F 학점 : 60점 미만

※ 처리 형태

input : 93

A

thank you! bye~!

```
#!/bin/bash
echo -n "input : "
read num
if [ $num -ge 90 ]; then
echo A
elif [ $num -ge 80 ]; then
echo "B"
elif [ $num -ge 70 ]; then
echo "C"
elif [ $num -ge 60 ]; then
echo "D"
else
echo "F"
fi
echo "thank you! bye~"
```