리눅스 시스템

4주차

목 차

Bash 스크립트 기초

- 생성 및 실행
  - 스크립트 파일 생성 (vi script.sh)
  - 실행권한 부여 (chmod +x script.sh)
  - 실행 (./script.sh)

- 쉐뱅(Shebang)
  - 스크립트 파일의 최상단에 어떤 스크립트를 사용할 지정하는 역할
    - #!/bin/bash
    - #!/bin/zsh
  - 만약 bash script.sh 처럼 명시적으로 사용한다면 필요 없으나, 일반적으로는 추가를 권장함

변수

- 변수 선언 및 출력

```
#!/bin/bash

name = “Jonggyu”
echo “Hello, $name!”
```

- 쉘 명령어 실행 결과를 변수에 저장

```
#!/bin/bash

today = $(date)
echo “Today is $today”
```

```
#!/bin/bash

flist = $(ls)
echo “file list are $flist”
```

```
#!/bin/bash

echo “enter your name: ”
read name
echo “Hello, $name”
```

제어문 if

- if 예제

```
#!/bin/bash

if [     조건     ]; then
    # 참일 때 실행
else
    거짓일 때 실행
fi
```

```
#!/bin/bash

if [ -e file.txt ]; then          // -e는 파일 존재 여부 확인
    echo “file.txt exists.”
else
    touch file.txt
    echo “file.txt created.”
fi
```

if␣[␣-e␣file.txt␣];

제어문 if

- if 예제

```
#!/bin/bash

if [ -r file.txt ]; then              // 파일이 읽기 가능한지 확인
    content=$(cat file.txt)
    echo "file content: $content"
else
    echo "file.txt is not readable or does not exist."
fi
```

```
#!/bin/bash

if [ -d my_dir ]; then              // -d는 디렉터리인지 확인
    echo "my_dir is a directory"
    cd my_dir
    ls
else
    echo "my_dir is not a directory"
fi
```

제어문 if

- if 예제

```
#!/bin/bash

str1 = “hello”
str2 = “world”

if [ “$str1” = “$str2” ]; then          // = 문자열 같은지 비교
    echo “same”                         // != 문자열 다른지 비교
else
    echo “different”
fi
```

- 조건문 안에서 쌍따옴표로 감싸는 이유

```
#!/bin/bash

str1 = “”

if [ $str1 = “hello” ]; then            // 조건문 안이 [ = “hello” ] 가 되면서 조건식에 구문 오류가 생긴다.

if [ “$str1” = “hello” ]; then          // 조건문 안이 [ “”=“hello”] 가 되면서 조건식 비교는 가능해진다.
```

제어문 if

- if 예제

```
#!/bin/bash

num1=3
num2=5

if [ “$num1” -lt “$num2” ]; then                    // 3 is less than 5, 3 < 5
    echo “$num1 is less than $num2”
fi
```

- 쉘 스크립트에서 “<, >”는 리다이렉션 기능으로 쓰이므로 숫자 비교에서는 아래 키워드를 사용해야함
  - -lt : 작다 (<)
  - -le: 작거나 같다 (<=)
  - -eq: 같다 (==)
  - -ne: 같지 않다(!=)
  - -gt: 크다(>)
  - -ge: 크거나 같다(>=)

제어문 case

- case 예제

```
#!/bin/bash

read = num

case $num in
        1)
                echo “your num is 1”
                ;;
        2)
                echo “your num is 2”
                ;;
        3)
                echo “your num is 3”
                ;;
        *)
                echo “your num is not in 1~3”
                ;;
esac
```

제어문 for

- for 예제

```
#!/bin/bash

for item in apple banana cherry; do
        echo “I like $item”
done
```

```
#!/bin/bash

for file in *.txt; do
        echo “text file: $file”
done
```

```
#!/bin/bash

for item in apple banana cherry; do
        echo “I like $item”
done 〉fruits.txt
```

제어문 for

- for 예제

```
#!/bin/bash

for i in {1..5}; do
        echo “Num: $i”
done
```

```
#!/bin/bash

for i in {1..10..2}; do
        echo “Num: $i”
done
```

제어문 while

- while 예제

```
#!/bin/bash

i=1
while [ $i -le 5 ]; do
        echo “i: $i”
        count=$((count+1))
done
```

- $((count+1)) 에서 괄호를 두 번 쓰는 이유
  - 스크립트에서 $( )는 명령어 치환을 위한 문법
  - $(( ))는 산술 연산을 위한 문법

제어문 while

- while 예제

```
#!/bin/bash

while IFS= read -r line; do
        echo “Line: $line”
done < fruits.txt
```

- `IFS=` : 공백과 탭을 그대로 유지
- `-r` : 역슬래시(₩)를 이스케이프 문자로 처리하지 않음

활용 예제

- 파일 확장자가 .log인 파일들을 모두 백업하고 싶을때

```
#!/bin/bash

for file in *.log; do
        mv “$file” “${file}_bak”
        echo “renamed $file to ${file}_bak”
done
```

Q&A