Shell Script

Nunu

1. Hello World!

- vim hello.sh

#!/usr/bin/env bash
echo “hello world"
printf "hello world ₩n"
printf "%s %s ₩n" hello world

* (실행이 안 될 시에는 chmod 777 hello.sh)

#!/usr/bin/env SOMETHING 을 사용하면 SOMETHING을 $PATH 변수에서 찾기 때문에, 절대경로나 직접 찾기 귀찮다면 이걸 쓰세요! (이걸 SHEBANG이라고 불러요.)

#!/usr/bin/bash
이걸 써도 되지만, 각각 장단점이 있어요!

자세한 내용은
https://stackoverflow.com/questions/16365130/what-is-the-difference-between-usr-bin-env-bash-and-usr-bin-bash

언제 Shebang을 써야 되는지는
https://stackoverflow.com/questions/7615430/do-you-need-shebang-in-all-bash-scripts

2. Functions

- function 생략 가능.

```
#!/usr/bin/bash
test_func() {
        echo "this is test function!"
}

function test_func2() {
        echo "this is test function 2!"
        echo "variable is ${@}"
}

test_func
test_func2 "hello world!"
```

```
root@kali:~# ./test.sh
this is test function!
this is test function 2!
variable is hello world!
```

함수에 **인자값**을 다음과 같이 넘겨줄 수 있습니다!

2. Functions

$0: script name,
$1: first argument,
$2: second argument,
$@: All arguments
$*: All arguments
$#: Number of arguments

⇨ 차이점?

대괄호는 있어도, 없어도 상관 x

```
#!/usr/bin/bash
test_func() {
        echo "this is test function!"
}

function test_func2() {
        echo "this is test function 2!"
        echo "variable is $0"
        echo "variable is $1"
        echo "variable is $2"
        echo "variable is ${@}"
}

test_func
test_func2 "apple!" "orange!"
```

```
root@kali:~# ./test.sh
this is test function!
this is test function 2!
variable is ./test.sh
variable is apple!
variable is orange!
variable is apple! orange!
```

3. Variables

1. 그냥 선언 시 전역 변수(global variable)
2. 함수 안에서만 지역 변수(local variable) 사용가능, 사용 시에 변수 명 앞에 ”local” 쓰기.
3. 변수 명 앞에 “export”를 붙여주면 환경 변수(environment variable)로 설정되어 자식 스크립트에서 사용 가능

   * Var = name (x) var=name (o)

Result

```
root@kali:~# ./test.sh
abc
localvar
abc
child called and received variable hello world!!
```

test.sh

```
str="abc"
echo ${str}

test_func() {
        local str="localvar"
        echo ${str}
}

function test_func2() {
        echo $1
}

test_func
test_func2 ${str}

export str2="hello world!"

# Called Child script!
./child.sh
```

child.sh

```
#!/usr/bin/env bash
echo "child called and received variable ${str2}!"
~
~
```

3. Variables

- 환경 변수 사용시, 예약 변수와 이름이 겹치면 안된다.

예약 변수(Reserved Variable)

문자	설명
HOME	사용자의 홈 디렉토리
PATH	실행 파일을 찾을 경로
LANG	프로그램 사용시 기본 지원되는 언어
PWD	사용자의 현재 작업중인 디렉토리
FUNCNAME	현재 함수 이름
SECONDS	스크립트가 실행된 초 단위 시간
SHLVL	쉘 레벨(중첩된 깊이를 나타냄)
SHELL	로그인해서 사용하는 쉘
PPID	부모 프로세스의 PID
BASH	BASH 실행 파일 경로

특수 매개 변수(Special Parameters)

문자	설명
$$	현재 스크립트의 PID
$?	최근에 실행된 명령어, 함수, 스크립트 자식의 종료 상태
$!	최근에 실행한 백그라운드(비동기) 명령의 PID
$-	현재 옵션 플래그
$_	지난 명령의 마지막 인자로 설정된 특수 변수

출처: https://blog.gaerae.com/2015/01/bash-hello-world.html

4. Comparison

Integer Comparison
(정수 비교)

- -eq: 같음
- -ne: 같지 않음
- 그 외에 <, <=, >, >=

String Comparison (문자열 비교)

- =, ==: 같음 , !=: 같지 않음
- <, > 작음, 큼
- -z: 문자열이 NULL, 길이가 0
- -n: 문자열이 NULL이 아님
- ${변수}: 문자열이 NULL이 아님

5. Conditionals

- 실행 문장이 없으면 오류 발생

vim cond.sh

```
#!/usr/bin/env bash

str1="hello"
str2="world"

if [ $str1 == $str2 ]; then
        echo "success!"

else
        #receive user input
        read -p "Enter your name: " name

        # Same as:
        # echo -n "Enter your name: "
        # read name

        echo "hello $name"

fi
```

Result

```
root@kali:~# chmod 777 cond.sh && ./cond.sh
Enter your name: nunu
hello nunu
```

6. Loops

- For, while, until

loop.sh

```
#!/usr/bin/env bash

for num in 1 2 3 4 5
do
        echo $num
done


read -p "Enter your name: " name
while [ -z "$name" ];
do
        read -p "Enter your name: " name
done

echo "Hi $name"

# Also could write this as
# while [ -z $name ]; do read -p "Enter your name: " name; done


until [ $num -le 0 ]; do
        echo $num
        num=$(( $num-1 ))
done
```

result

```
root@kali:~# chmod 777 loop.sh && ./loop.sh
1
2
3
4
5
Enter your name: nunu
Hi nunu
5
4
3
2
1
```

7. Arrays

배열 변수는 반드시 괄호를 사용합니다.

```
#!/usr/bin/env bash

array=("ab" "cd" "ef" "gh")


# Number does not matter when inserting
array[100]="ij"

echo "array now: ${array[@]}"

array=(${array[@]} "kl")

echo "array now: ${array[@]}"

# It does when inserting in the middle.
array[2]="temp"
echo "array now: ${array[@]}"


unset array[2]
echo "array now: ${array[@]}"

unset array
echo "array now: ${array[@]}"
```

array.sh

Result

```
root@kali:~# chmod 777 array.sh && ./array.sh
array now: ab cd ef gh ij
array now: ab cd ef gh ij kl
array now: ab cd temp gh ij kl
array now: ab cd gh ij kl
array now:
```

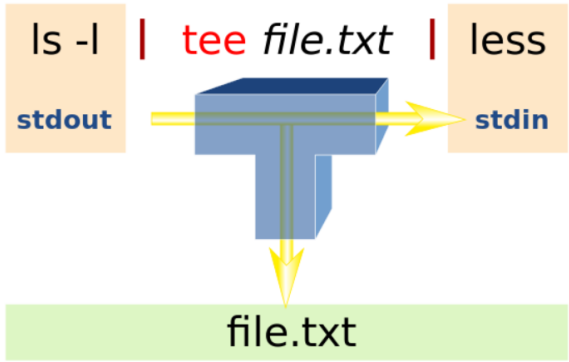
Aside- Logging

```
#!/usr/bin/env bash
exec 1>>log.txt
read -p "Enter your name: " name

# $nameabc would call varaible named nameabc.
# this appends "abc" to variable name.

echo "Hello ${name}abc" | tee log.txt
echo "$(date) bye!" | tee -a log.txt


# To terminal
exec 1> /dev/tty
echo "HI"
```

>, >>:
프로그램의 output을 stdout이 아닌 다른 곳으로 redirect시킬 때 사용 (>: overwrite, >>: append)

${name} 대괄호를 써서 string append

tee는 명령어의 출력 결과를 파일과 화면에 동시에 출력하게 해줍니다.



Aside- sed

Sed – 편집기를 명령어로 쓰듯 사용하는것과 비슷합니다.

(쉘 리다이렉션을 통해 편집 결과를 저장하기 전 까지는 원본을 손상시키지 않아요.)

만약 아래 글이 a.txt에 저장되어 있다면,

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.”

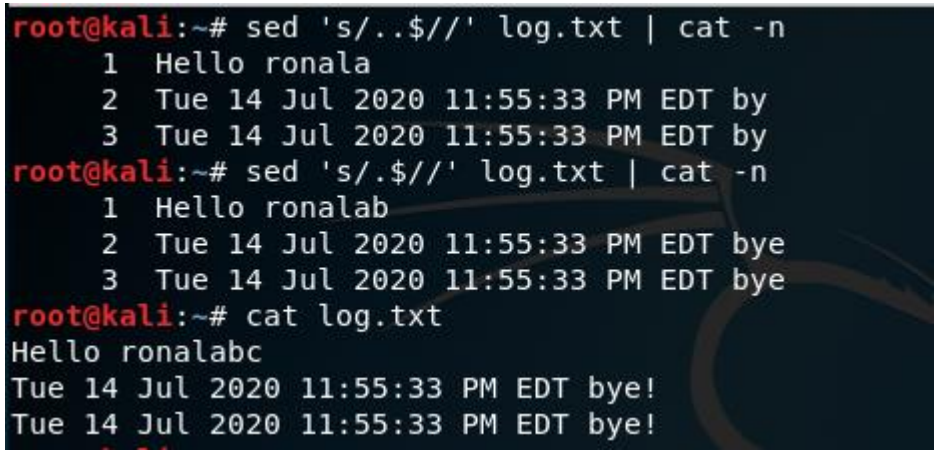
1. 치환 (substitute)

sed ‘s/lorem/changed/’ a.txt **sed ‘s/<original>/<new>/’ <filename>**

2. 삭제 (Delete)

sed ‘/word/d’ a.txt -> word가 포함된 줄을 삭제합니다.

sed ‘3d’ a.txt -> 세번째 줄을 삭제합니다.



3. 이 외에도…

sed ‘a₩₩<Inserted at the end of every line>₩ <filename>

sed ‘c ₩₩<Replace every line with this>’ <filename>

sed ‘<num>q’ <filename line number <num> 에서 멈춘다.

sed –n ‘<num>s/<original>/<new>’ <filename>

<num>번째 라인에서 특정 문자 치환

- 이렇게 쓰일 수 있습니다.

1. 모든 공백 라인 제거
sed ‘/^$/d’ <filename>

2. 각 line마다 공백라인 추가
sed ‘a₩₩’ <filename>

3. 각 line의 시작을 3 space로 대체
sed ‘s/^/ /’ <filename>

편집된 내용을 저장할려면 명령어 뒤에 “> new_filename” 만 붙여주시면 됩니다.

8. 실습

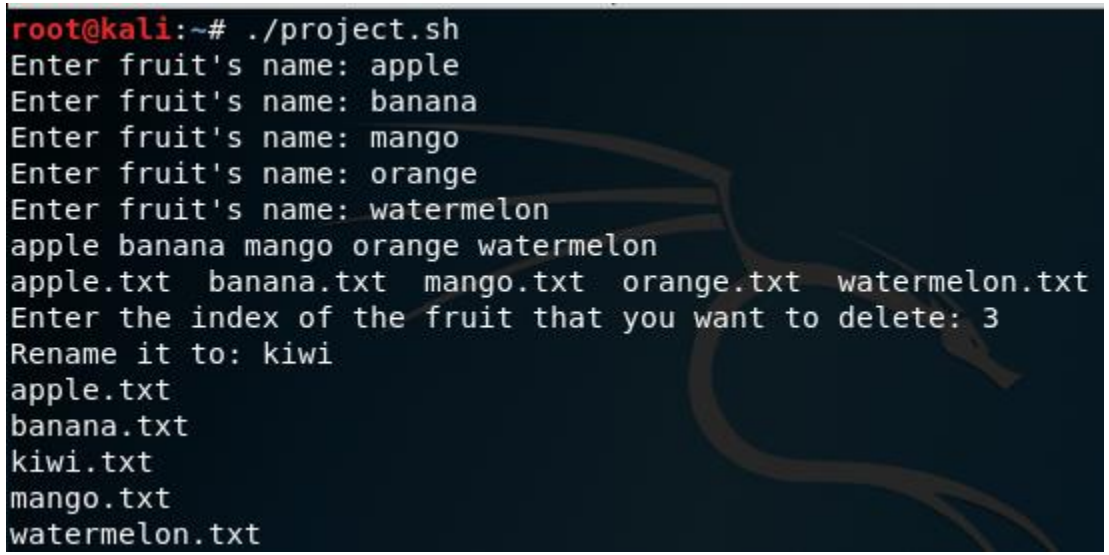
0. vim project.sh
1. User input으로 과일 5개를 받고 배열에 저장합니다.
   (Tip* empty array 선언은 unset $array로 할 수 있습니다.)
2. 이때, User가 empty string을 넣었을 경우 프로그램을 종료시킵니다. (exit)
3. ①에서 5개의 과일을 성공적으로 저장했다면, function make()에 배열을 인자값으로 넘깁니다.
4. make() 함수에서는 fruits 폴더를 생성한 뒤 폴더 안에 인자값에 해당하는 5개의 “fruit_name.txt”를 만듭니다. 이후, fruits 폴더 내에 있는 모든 파일을 출력합니다. (ls를 사용하시면 됩니다!)
5. 이후 다시 한번 User input으로 숫자를 받습니다.
6. Export를 사용하여 ${array[$[숫자]}를 환경변수로 설정 후 ../rename.sh를 실행시킵니다.

8. 실습

7. vim rename.sh
8. User input으로 new_name을 받습니다.
9. 환경변수에 해당하는 old_name 파일을 new_name으로 rename 합니다.
10. 마지막으로, fruits 파일에 있는 모든 파일을 출력한 뒤,

(이때는 각각 new line에 나와야 되요! => for loop을 이용해서 출력해주시면 됩니다. Ex. For filename in *f) 이후 터미널의 사진을 스크린샷 찍어 slack에 올려주시면 됩니다.

제출 예시


root@kali:~# ./project.sh
Enter fruit's name: apple
Enter fruit's name: banana
Enter fruit's name: mango
Enter fruit's name: orange
Enter fruit's name: watermelon
apple banana mango orange watermelon
apple.txt banana.txt mango.txt orange.txt watermelon.txt
Enter the index of the fruit that you want to delete: 3
Rename it to: kiwi
apple.txt
banana.txt
kiwi.txt
mango.txt
watermelon.txt

감사합니다!

Reference

- https://linuxstory1.tistory.com/entry/SED-%EB%AA%85%EB%A0%B9%EC%96%B4-%EC%82%AC%EC%9A%A9%EB%B2%95
- *https://blog.gaerae.com/2015/01/bash-hello-world.html*
- https://twpower.github.io/135-tee-command-usage

project.sh

```
#!/usr/bin/env bash
make() {
        mkdir -p fruits && cd fruits
        echo $@
        for str in $@
        do
                echo "fruit" > $str.txt
        done
        ls
}

num=5
unset $array
until [ $num -le 0 ]; do
        read -p "Enter fruit's name: " fruit
        if [ -z ${fruit} ]
        then
                echo "You put invalid fruit name. Terminating..."
                exit
        fi
        array=(${array[@]} $fruit)
        num=$(( $num - 1 ))
done

make ${array[@]}
read -p "Enter the index of the fruit that you want to delete: " num
export str=${array[$num]}
../rename.sh
```

rename.sh

```
#!/usr/bin/env bash
read -p "Rename it to: " name

mv "$str.txt" "$name.txt"


for fname in *
do
        echo "$fname"
done
```