Contribution

Marathon

Innovation

엔터프라이즈 서버관리

12주차 : 셸 스크립트 1

2024년 1학기

목 차

1. 셸 스크립트 개요

1. 셸 스크립트 개요

셸

셸 스크립트 = 셸 스크립트 프로그램 = 셸 스크립트 파일

명령어 해석기

- 사용자에게 인터페이스를 제공하는 프로그램
- 셸은 명령어에 상응하는 프로그램을 실행시키고, 완료 후 결과를 스크린에 보여줌

스크립트 언어, 프로그래밍 언어

셸 스크립트 작성

- 함수, 변수 등의 프로그래밍 요소를 사용할 수 있음
- 반복문, 조건문 등의 제어 구조를 사용할 수 있음
- 셸은 셸 스크립트 프로그램에 있는 명령 행을 제어 구조에 따라 해석하고 실행시킴
- 파일(셸 스크립트)에 명령을 저장하여 자동, 반복적으로 실행 가능

정기적으로 특정한 시점이 되었을 때 정기적으로 반복해서 수행함 (금융사의 센터 컷, 정기적인 백업 작업)

셸의 종류

Bash 셸(리눅스 배포판의 기본 셸), Csh/Tcsh 셸, Ksh 셸, Zsh 셸, Fish 셸 등

텍스트 모드 또는 GUI 환경에서 터미널을 띄우고 셸 명령을 실행함

1. 셸 스크립트 개요

셸 스크립트

- 셸 명령어의 집합으로 이루어진 실행 가능한 프로그램(텍스트 파일)
  - 셸이 스크립트 파일의 내용을 읽어 처리함
  - 선택/반복 등의 프로그래밍 구조를 사용할 수 있음
  - 셸에서 사용할 수 있는 모든 기능을 포함할 수 있음
    - 함수나 변수의 선언, 파이프, 입출력 리다이렉션 등
      - 파이프 : 두 명령을 연결 (앞 명령의 출력이 뒷 명령의 입력)
      - 입출력 리다이렉션 : 입출력 내용이 파일에 저장됨
  - 긴 작업, 반복적으로 일어나는 작업을 셸 스크립트로 작성할 수 있음
  - 셸 스크립트를 새로운 하나의 명령어처럼 사용할 수 있음
- 예
  - /etc/profile, /etc/bashrc, ~/.bash_profile, ~/.bashrc 등의 셸 초기화 스크립트

1. 셸 스크립트 개요

```
[linux@localhost etc]$ cat profile
# /etc/profile

# System wide environment and startup programs, for login setup
# Functions and aliases go in /etc/bashrc

# It's NOT a good idea to change this file unless you know what you
# are doing. It's much better to create a custom.sh shell script in
# /etc/profile.d/ to make custom changes to your environment, as this
# will prevent the need for merging in future updates.

pathmunge () {
    case ":${PATH}:" in
        *:"$1":*)
            ;;
        *)
            if [ "$2" = "after" ] ; then
                PATH=$PATH:$1
            else
                PATH=$1:$PATH
            fi
    esac
}
```

1. 셸 스크립트 개요

```
[linux@localhost etc]$ cat bashrc
# /etc/bashrc

# System wide functions and aliases
# Environment stuff goes in /etc/profile

# It's NOT a good idea to change this file unless you know what you
# are doing. It's much better to create a custom.sh shell script in
# /etc/profile.d/ to make custom changes to your environment, as this
# will prevent the need for merging in future updates.

# Prevent doublesourcing
if [ -z "$BASHRCSOURCED" ]; then
  BASHRCSOURCED="Y"

  # are we an interactive shell?
  if [ "$PS1" ]; then
    if [ -z "$PROMPT_COMMAND" ]; then
      case $TERM in
      xterm*|vte*)
        if [ -e /etc/sysconfig/bash-prompt-xterm ]; then
            PROMPT_COMMAND=/etc/sysconfig/bash-prompt-xterm
        else
            PROMPT_COMMAND='printf "\033]0;%s@%s:%s\007" "${USER}" "${HOSTNAME%%.*}" "${PWD/#$HOME/\~}"'
        fi
        ;;
      screen*)
```

1. 셸 스크립트 개요

셸 스크립트의 실행 방법 명령 행에서 *script_file* 다음에 인수를 추가할 수 있음

- bash *script_file* **편집작업은 vi 에디터를 이용하거나 cat > ooo.sh 내용 입력 후 ctrl +d로 저장**
  - **bash** 명령을 사용하며, 스크립트 파일에 실행 권한을 추가할 필요가 없음
  - 서브 셸을 새로 생성하여 스크립트를 실행함
- *./script_file*
  - 스크립트 파일의 이름을 명령어처럼 사용하는 방법
  - 스크립트 파일에 실행 권한을 추가해야 함
  - **PATH** 환경 변수에 설정된 디렉터리에서 스크립트 파일을 찾음
  - 스크립트 파일의 첫 행에서 **#!** 다음에 해석기(/bin/bash)를 지정함
  - 서브 셸을 새로 생성하여 스크립트를 실행함
- source *script_file* 또는 . *script_file*
  - **source**는 셸의 내장 명령이며 현재 사용 중인 셸 환경에서 스크립트 파일을 실행 함
  - **source** 명령 대신에 도트(.) 명령을 사용할 수 있음

#! : 셔뱅(shebang)
어떤 셸을 이용해서 셸 스크립트를 실행 시킬지 선언하는 부분

1. 셸 스크립트 개요

셸 스크립트 문법

- # 이후 (같은 행에서) 나오는 내용은 주석으로 처리됨
- $0은 스크립트 파일의 이름으로 확장됨
  - source 명령으로 실행하는 경우는 셸의 이름(bash)으로 확장됨
- $1은 첫 번째 인수, $2는 두 번째 인수로 확장됨
- 스크립트가 복잡하다면 , 중간중간에 적절한 echo 명령을 넣는 것이 좋음
- *변수=값* 을 사용하여 변수에 값을 지정하고 *$변수* 를 사용하여 값을 추출할 수 있음
  - 등호(=)의 좌우에 공백이 있으면 안됨
- 참고
  - bash -x *script_file* 과 같이 실행하면 실행 전에 명령 자체가 화면에 출력됨

```
$ ./myScript.sh Seoul pusan
  $0          $1    $2
```

입력한 명령이 각각 순서대로 $0, $1, $2로 확장됨
$#은 $0를 제외한 개수
$*는 $0를 제외하고 모두 출력

중간중간에 진행 내용을 확인할 수 있음

```
a=3
echo $a
```

디버깅 모드로 수행

1. 셸 스크립트 개요

예

```
$ cat myScript.sh
#!/bin/bash
echo Hello Linux
echo $0 $1 $2
ls -l $0
$ chmod u+x myScript.sh
$ ./myScript.sh Seoul pusan
Hello Linux
./myScript.sh Seoul pusan
-rwxrw-r--. 1 kdhong kdhong 52  4월  7 20:11 ./myScript.sh
```

- #! : 셔뱅(shebang)
  - 어떤 셸을 이용해서 셸 스크립트를 실행시킬지 선언하는 부분

처리기로 사용할 셸을 지정

사용자가 실행할 수 있도록 권한을 줌

$0 $1 $2

확장

```
[root@localhost home]# cat myScript.sh
#! /bin/bash
echo Hello Linux
echo $0 $1 $2
ls -l $0
```

셸 스크립트

```
[root@localhost home]# ./myScript.sh Seoul pusan
Hello Linux
./myScript.sh Seoul pusan
-rwxr--r--. 1 root root 53  5월  20 20:03 ./myScript.sh
```

```
$ bash -x myScript.sh Daegu Kwangju
+ echo Hello Linux
Hello Linux
+ echo myScript.sh Daegu Kwangju
myScript.sh Daegu Kwangju
+ ls -l myScript.sh
-rwxrw-r--. 1 kdhong kdhong 52  4월  7 20:11 myScript.sh
```

+ 표시
결과가 나오기 전에
실행할 명령을 출력

결과

```
[root@localhost home]# bash -x myScript.sh Daegu Kwangju
+ echo Hello Linux
Hello Linux
+ echo myScript.sh Daegu Kwangju
myScript.sh Daegu Kwangju
+ ls -l myScript.sh
-rwxr--r--. 1 root root 53  5월  20 20:03 myScript.sh
```

1. 셸 스크립트 개요 (리눅스 마스터 기출문제)

리눅스 마스터 2급

다음 중 (괄호) 안에 들어갈 용어로 알맞은 것은?

> (괄호)는/은 사용자로 부터 명령을 받아 해석하고 프로그램을 실행하는 역할을 한다. 또한 프로그램할 수 있는 여러가지 기능을 가지고 있다.

① 셸(Shell)
② 커널
③ 네트워크
④ 환경변수

1. 셸 스크립트 개요 (리눅스 마스터 기출문제)

리눅스 마스터 2급

다음 중 (괄호) 안에 들어갈 용어로 알맞은 것은?

> (괄호)는/은 사용자로 부터 명령을 받아 해석하고 프로그램을 실행하는 역할을 한다. 또한 프로그램할 수 있는 여러가지 기능을 가지고 있다.

① 셸(Shell)　② 커널

③ 네트워크　④ 환경변수

셸　셸 스크립트 = 셸 스크립트 프로그램 = 셸 스크립트 파일

- 명령어 해석기
  - 사용자에게 인터페이스를 제공하는 프로그램
  - 셸은 명령어에 상응하는 프로그램을 실행시키고, 완료 후 결과를 스크린에 보여줌

1. 셸 스크립트 개요 (리눅스 마스터 기출문제)

리눅스 마스터 2급

다음 명령의 결과로 알맞은 것은?

```
$ user=lin
$ echo user
(   )
```

① lin　　② echo

③ user　　④ 화면에 아무것도 출력되지 않는다.

1. 셸 스크립트 개요 (리눅스 마스터 기출문제)

리눅스 마스터 2급

다음 명령의 결과로 알맞은 것은?

```
$ user=lin
$ echo user
(   )
```

① lin　　② echo

③ user　　④ 화면에 아무것도 출력되지 않는다.

- 스크립트가 복잡하다면 , 중간중간에 적절한 echo 명령을 넣는 것이 좋음

중간중간에 진행 내용을 확인할 수 있음

2. 변수의 사용

2. 변수의 사용

셸 스크립트에서 변수의 사용

선언 없이 변수를 사용할 수 있음

- 변수에 값을 지정하는 방법은 *변수이름=값* (공백이 있으면 안 됨)
- 예: MYCOLOR=blue 변수의 선언과 동시에 사용
- 변수 확장 $*변수* 는 변수의 값을 추출하는 것
- 예: echo $PATH

```
[root@localhost home]# cat > aaa.sh
a=3
b=4
echo a+b
[root@localhost home]# bash aaa.sh
a+b
```

변수의 값은 기본적으로 문자열로 취급됨

- 연산이 필요하고, 변환이 가능한 경우에 정수로 다루어짐

```
[root@localhost home]# cat aaa.sh
a=3
b=4
echo $a+$b
[root@localhost home]# bash aaa.sh
3+4
```

변수의 이름 작명

- 대소문자의 구별
- 첫 자는 숫자가 될 수 없음
- 영문자, 숫자, 언더스코어(_) 문자로 구성

```
[root@localhost home]# cat aaa.sh
a=3
b=4
echo $[$a+$b]
[root@localhost home]# bash aaa.sh
7
```

```
[root@localhost home]# cat aaa.sh
a=3
b=4
echo $[a+b]
[root@localhost home]# bash aaa.sh
7
```

2. 변수의 사용

변수의 사용 예

사용 예	설명
a=ls	변수 a에 문자열 “ls”를 대입
b="A string"	따옴표를 사용하여 공백을 가진 문자열을 대입
c="A string and $b"	$b를 ‘변수 확장’한 후 문자열을 대입
d=$(ls | wc - l)	$(...)는 ‘명령 치환’을 의미. 괄호 안의 명령을 실행한 후 결과를 대입
e=$[5*7]	$[...]은 ‘수식 확장’을 의미. 대괄호 안의 수식을 계산한 후 결과를 대입
f="\t\tA string\n"	이스케이프 문자인 백 슬래시(\)를 사용한 예
g=5 h="A string"	한 행에 2 개의 변수를 사용하는 예
i=${b}1	변수 이름을 중괄호로 묶으면 뒤따라 나오는 문자를 분리할 수 있음. 즉, “A string1"이 대입됨. 실제 $b는 ${b}를 줄여 쓴 형태
$a	먼저 변수 확장이 일어나고, 셸은 이것을 하나의 명령어로 해석하여 처리
$a=$e	변수 확장의 결과는 ls=35. 셸은 이 자체를 하나의 명령어로 처리하므로 오류가 발생. eval " $a=$e " 를 실행할 수 있음

라인 수를 카운트하여 변수 d에 대입

ls 명령 실행

eval : 인자로 주어진 것을 하나의 셸 명령으로 해석함
eval “$a=$e”를 실행하면 ls에 35가 대입됨

2. 변수의 사용

```
a=ls
echo $a
b="A string"
echo $b
c="A string and $b"
echo $c
d=$(ls | wc -l)
echo $d
e=$[5*7]
echo $e
f="\t\tA string \n"
echo $f
g=5 h="A string":
echo $g
echo $h
i=${b}1
echo $i
$a
eval "$a=$e"
echo $ls
j="pwd"
eval "$j"
```

```
echo $a
echo $b
echo $c
echo $d
echo $e
echo $f
echo $g
echo $h
echo $i
$a
echo $ls
eval "$j"
```

```
[root@localhost inha]# bash inha.sh
ls
A string
A string and A string
3
35
\t\tA string \n
5
A string:
A string1
arg.sh  inha.sh  whoson.sh
35
/home/inha
```

2. 변수의 사용

명령행 인수

- 셸이 스크립트를 처리할 때 사용되는 인수
  - 스크립트 실행에 필요한 입력값이 있을 수 있음
- $0은 스크립트 파일의 이름으로 확장됨
- 나머지를 위치 매개변수(positional parameter)라 하며 $1, $2, $3 등은 이러한 인수를 의미함

```
$ cat > arg.sh
echo "This script's name is : $0"
echo Argument 1: $1
echo Argument 2: $2
$ chmod u+x arg.sh
$ ./arg.sh first second
This script's name is : ./arg.sh
Argument 1: first
Argument 2: second
$ /home/kdhong/arg.sh One Two
This script's name is : /home/kdhong/arg.sh
Argument 1: One
Argument 2: Two
```

```
[root@localhost inha]# cat arg.sh
echo "This script's name is : $0"
echo Argument 1: $1
echo Argumaet 2: $2
[root@localhost inha]# ./arg.sh first second
This script's name is : ./arg.sh
Argument 1: first
Argumaet 2: second
[root@localhost inha]# /home/inha/arg.sh One Two
This script's name is : /home/inha/arg.sh
Argument 1: One
Argumaet 2: Two
```

2. 변수의 사용

```
[root@localhost inha]# cat arg.sh
echo "This script's name is : $0"
echo Argument 1: $1
echo Argumaet 2: $2
[root@localhost inha]# ./arg.sh first second
This script's name is : ./arg.sh
Argument 1: first
Argumaet 2: second
[root@localhost inha]# /home/inha/arg.sh One Two
This script's name is : /home/inha/arg.sh
Argument 1: One
Argumaet 2: Two
```

2. 변수의 사용

특별한 매개 변수

- 명령 행 인수와 관계가 있는 특별한 변수
- 변수의 값을 참조만 할 수 있음

특수 변수	설명
$*	"$*"는 모든 위치 매개변수($0 은 제외)를 포함하는 1 개의 큰 따옴표, 즉 1 개의 문자열로 확장됨. 즉, "$1 $2 $3..." 와 같음
$@	"$@"는 여러 개별 큰 따옴표, 즉 분리된 문자열로 확장된다. 즉, "$1" "$2" "$3"... 와 같음
$$	셸의 프로세스 ID(PID)로 확장된다.
$#	위치 매개변수의 개수로 확장된다.
$?	최근에 실행된 포어그라운드 명령의 종료 상탯값으로 확장된다. 성공적으로 종료된 경우 0임
$!	최근 실행된 백그라운드 명령의 프로세스 ID로 확장된다.

2. 변수의 사용

```
[root@localhost home]# cat arg.sh
echo $0
echo $1
echo $2
echo $*
echo $@
echo $$
echo $#
echo $?
echo $!
[root@localhost home]# ./arg.sh arg1 arg2
./arg.sh
arg1
arg2
arg1 arg2
arg1 arg2
9986
2
0
9919
```

2. 변수의 사용

read 명령

- 키보드로부터 한 라인을 읽은 후, 개별 단어를 상응하는 변수에 저장함
  - 대화식으로 스크립트를 실행시킬 수 있음
- read [*options*] [*variable…*]
  - 첫 번째 변수에 첫 번째 단어를, 두 번째 변수에 두 번째 단어를 저장
  - 변수 개수가 적으면, 마지막 변수에 나머지 모두를 저장
  - 남는 변수가 있으면 빈 문자열이 됨
  - 변수의 이름을 사용하지 않으면 셸 변수 **REPLY**에 저장됨
  - 옵션 **-p *prompt***는 입력을 위한 프롬프트를 지정함

```
$ read -p "Type your first name, last name, and address:" first last address
Type your first name, last name, and address:Kildong Hong Songpa-gu, Seoul
```

2. 변수의 사용

```
[root@localhost home]# cat readtest.sh
read -p "Type your name, tel_no, address : " name tel_no address
echo name [$name]
echo tel_no [$tel_no]
echo address [$address]

[root@localhost home]# ./readtest.sh
Type your name, tel_no, address : kil-dong Hong, 010-1234-5678, Seoul Korea
name [kil-dong]
tel_no [Hong,]
address [010-1234-5678, Seoul Korea]
```

```
[root@localhost home]# ./readtest.sh
Type your name, tel_no, address : kil-dong-Hong 010-1234-5678 Seoul-Korea
name [kil-dong-Hong]
tel_no [010-1234-5678]
address [Seoul-Korea]
```

2. 변수의 사용 (리눅스 마스터 기출문제)

리눅스 마스터 2급

다음 중 (괄호) 안에 들어갈 명령의 결과로 알맞은 것은?

```
[ihduser@ihd ~]$ user=kaitman
[ihduser@ihd ~]$ echo $USER
( 괄호 )
```

① 아무것도 출력되지 않는다.
② $user
③ ihduser
④ kaitman

2. 변수의 사용 (리눅스 마스터 기출문제)

리눅스 마스터 2급

다음 중 (괄호) 안에 들어갈 명령의 결과로 알맞은 것은?

```
[ihduser@ihd ~]$ user=kaitman
[ihduser@ihd ~]$ echo $USER
( 괄호 )
```

① 아무것도 출력되지 않는다.
② $user
③ ihduser
④ kaitman

user=kaitman **이므로** echo $user **인 경우만 괄호안에** kautman **이 출력됨**

하지만 위에 예제는 $USER **이므로 원래 계정인** ihduser **이 출력됨**

2. 변수의 사용 (리눅스 마스터 기출문제)

리눅스 마스터 2급

다음 중 (괄호) 안에 들어갈 스크립트 출력 내용으로 틀린 것은?

```
$ cat arg.sh
echo $0
echo $1
echo $#
echo $*

$ ./arg.sh arg1 arg2
(     ㉠     )
(     ㉡     )
(     ㉢     )
(     ㉣     )
```

① ㉠ : ./arg.sh　　② ㉡ : arg1

③ ㉢ : 3　　④ ㉣ : arg1 arg2

2. 변수의 사용 (리눅스 마스터 기출문제)

리눅스 마스터 2급

다음 중 (괄호) 안에 들어갈 스크립트 출력 내용으로 틀린 것은?

```
$ cat arg.sh
echo $0
echo $1
echo $#
echo $*

$ ./arg.sh arg1 arg2
(     ㉠     )
(     ㉡     )
(     ㉢     )
(     ㉣     )
```

```
[root@localhost home]# cat > arg.sh
echo $0
echo $1
echo $#
echo $*
```

```
[root@localhost home]# ./arg.sh arg1 arg2
bash: ./arg.sh: 허가 거부
[root@localhost home]# chmod u+x arg.sh
[root@localhost home]# ./arg.sh arg1 arg2
./arg.sh
arg1
2
arg1 arg2
```

① ㉠ : ./arg.sh
② ㉡ : arg1
③ ㉢ : 3
④ ㉣ : arg1 arg2

3. 함수의 사용

3. 함수의 사용

함수

- 함수는 셸 스크립트에서 반복적으로 사용되는 명령의 묶음
- 셸 스크립트에서 함수를 정의할 수 있음
  - 정의되어 있는 함수를 호출하여 사용함
- 셸에서 함수를 호출하면, 호출한 셸 내에서 실행됨

함수의 사용(호출) 방법

- 함수가 정의된 셸 스크립트 내에서 함수를 호출하여 사용할 수 있음
- source 명령이나 도트(.) 명령으로 함수가 정의된 셸 스크립트를 실행하면 셸 환경에 함수 정의가 추가됨. 그러면 같은 셸에서 함수 이름을 셸 명령어처럼 사용
  - set 명령으로 함수 정의를 확인하고 unset *name* 으로 삭제할 수 있음

3. 함수의 사용

함수 정의 방법

- function *name* {
  command ...
  return
  }

  또는

  name ( ) {
  command ...
  return
  }

- return 또는 return *n* 문은 함수를 종료하는 문장으로 종료 상탯값을 리턴할 수 있음(n이 생략되면 직전 명령의 종료 상탯값을 리턴)
- 함수 정의를 항상 사용하려면 .bashrc에 넣는 것이 좋음
  - 에일리어스 설정도 마찬가지임
  - 셸이 시작될 때 항상 **. ~/.bashrc** 가 실행되기 때문임

3. 함수의 사용

함수의 실행 예(1)

```
$ cat whoson.sh
#!/bin/bash

whoson() {
 date
 user=$USER
 echo "$user currently logged on"
}

echo "Step 1"
whoson   함수 호출
echo "Step 3"
```

```
$ . whoson.sh
Step 1
2023. 04. 07. (금) 20:40:11 KST
kdhong currently logged on
Step 3
$ whoson
2023. 04. 07. (금) 20:40:41 KST
kdhong currently logged on
$ echo $user
kdhong
```

3. 함수의 사용

```
[root@localhost inha]# cat whoson.sh
#! /bin/bash

whoson() {
 date
 user=$USER
 echo "$user currently logged on"
}

echo "Step 1"
whoson
echo "Step 3"
```

```
[root@localhost inha]# . whoson.sh
Step 1
2024. 05. 21. (화) 03:46:44 KST
linux currently logged on
Step 3
```

```
[root@localhost inha]# whoson
2024. 05. 21. (화) 03:46:55 KST
linux currently logged on
[root@localhost inha]# echo $user
linux
```

3. 함수의 사용

함수의 실행 예(2)

```
$ cat .bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific aliases and functions
arg1 () {
  echo "$1"
}

ds () {
  echo "Disk space utilization for $USER"
  du -sh ~
}
```

```
$ . ~/.bashrc
$ arg1 first
first
$ ds
Disk space utilization for kdhong
637M    /home/kdhong
```

```
[linux@localhost ~]$ arg1 first
first
[linux@localhost ~]$ ds
Disk space utilization for linux
59M     /home/linux
```

3. 함수의 사용 [리눅스 마스터 기출문제]

리눅스 마스터 2급

다음 (괄호) 안에 출력되는 내용으로 알맞은 것은?

```
[ihduser@kait ~]$ user=lin
[ihduser@kait ~]$ echo $USER
( 괄호 )
```

① lin
② USER
③ ihduser
④ 아무것도 출력되지 않는다.

3. 함수의 사용 (리눅스 마스터 기출문제)

리눅스 마스터 2급

다음 (괄호) 안에 출력되는 내용으로 알맞은 것은?

```
[ihduser@kait ~]$ user=lin
[ihduser@kait ~]$ echo $USER
( 괄호 )
```

① lin
② USER
③ ihduser
④ 아무것도 출력되지 않는다.

```
ds () {
  echo "Disk space utilization for $USER"
  du -sh ~
}
```

```
[linux@localhost ~]$ arg1 first
first
[linux@localhost ~]$ ds
Disk space utilization for linux
59M     /home/linux
```

Contribution

Marathon

Innovation

The End !