



# 13장. 배시셀 프로그래밍(1)

## ❖ 학습목표

- 다양한 쉘 변수를 이해하고 활용하는 방법을 익힌다
- 사용자로부터 입력을 받아 스크립트 파일에서 처리하는 방법을 익힌다
- 다양한 연산자와 문자열 테스트, 파일 테스트를 활용하는 방법을 익힌다
- 조건문과 반복문의 사용 방법을 익힌다
- 스크립트의 실행 오류를 찾아 수정하는 방법을 익힌다

## ❖ 내용

- 쉘 스크립트
- 쉘 변수 사용하기
- 사용자로부터 입력받기
- 연산자
- 제어문
- 디버깅

# 01. 쉘 스크립트

## ❖ 스크립트?

- 인터프리터라 불리는 다른 프로그램에 의해 실행되는 프로그램
- 자바 스크립트, Perl, 파이썬,...

## ❖ 쉘 스크립트

- 쉘이 실행하는 프로그램
- 유닉스 명령 + 쉘이 제공하는 프로그램 구성 요소
- 쉘 스크립트 파일 이름은 키워드나 앨리어스(alias) , 내장 명령과 같은 이름을 쓰지 않는 것이 좋다.

# 셸 스크립트 만들기

## ❖ vi 에디터로 작성

■ 예 :

```
$ vi test_script
```

```
#!/bin/bash  
#  My First Script Program  
  
echo I love UNIX !  
pwd
```

# 셸 스크립트 실행하기

## ❖ 스크립트 실행방법

- 셸을 실행하면서 인자로 스크립트 이름 지정( bash 파일명)

```
$ bash test_script
I love UNIX!
/home/user1
$
```

- 파일을 직접 실행
  - 텍스트파일에 실행 권한을 부여하여 실행파일로 만들어서 직접 실행시킴.

```
$ chmod +x test_script
$ ./test_script
I love UNIX!
/home/user1
$
```

PATH 환경 변수에 현재 디렉토리 (.)가 포함되어 있지 않으면 ./를 꼭 지정해야 함!!

# 명령어 경로(PATH) 설정(1)

## ❖ 현재 path 관련 변수 확인

- echo \$PATH

```
$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:
$
```

## ❖ 명령어로 경로 path 디렉토리 추가

- PATH=\$PATH:<추가할 디렉토리>
- 현재 로그인 상태에서만 추가 (로그아웃 시 소멸)

```
$ PATH=$PATH:.
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:
$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin: .
```

## ❖ 설정파일에 경로 path 디렉토리 추가

- 전체 사용자에게 추가 : /etc/profile 편집
- 특정 사용자에게 추가 : ~/.bash\_profile 편집

# 명령어 경로(PATH) 설정(2)

## ❖ 설정파일에 경로 path 디렉토리 추가

- 전체 사용자에게 추가 : /etc/profile 편집
- 특정 사용자에게 추가 : ~/.bash\_profile 편집

```
$ vi .bash_profile
```

```
# .bash_profile
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs
PATH=$PATH:$HOME/.local/bin:$HOME/bin:.
export PATH
```

PATH 끝부분에 . 추가

```
$ echo $PATH
```

```
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:.
```

# 스크립트 파일의 구성요소 - #!

❖ **#! : 매직 넘버**

❖ **파일의 가장 처음에 위치**

❖ **스크립트를 실행할 프로그램 지정**

- 각 셸마다 제공하는 스크립트 언어의 문법이 조금씩 다르므로, 스크립트를 실행할 셸을 지정해 주어야 올바르게 실행됨
  - 예 : `#!/bin/bash`, `#!/bin/csh`
- 셸이 아닌, 다른 실행 가능한 명령을 지정해 주어도 됨
  - 예: `#!/bin/more`, `#!/bin/rm`



# 스크립트 파일의 구성요소 - #!

## ❖ 예제

test\_script

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

```
# My First Script Program
```

```
echo I love UNIX !
```

```
pwd
```

test\_sharp

```
#!/bin/more
```

```
# test_sharp: 스스로를 출력하는 스크립트
```

```
# 이 스크립트를 실행시키면 자기 자신을 화면에 출력합니다.
```

```
# 주석문도 모두 출력되지요.
```

```
echo "This line is printed. "
```

## 실행

```
$ bash test_script
```

```
I love UNIX!
```

```
/home/user1
```

```
$ more test_sharp
```

```
#!/bin/more
```

```
# test_sharp: 스스로를 출력하는 스크립트
```

```
# 이 스크립트를 실행시키면 자기 자신을 화면에 출력합니다.
```

```
# 주석문도 모두 출력되지요.
```

```
echo "This line is printed..."
```

```
$
```

# 스크립트 파일의 구성요소 - #

- ❖ # : 주석 (comment)
- ❖ 프로그램에 대한 설명
- ❖ 행 전체, 또는 행의 일부를 주석으로 처리할 수 있음
- ❖ 예제

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

```
# My First Script Program
```

```
echo I love UNIX !
```

```
pwd
```

# 스크립트 파일의 구성요소 - 셸 명령

- ❖ 셸이 실행할 수 있는 모든 명령어 사용 가능
- ❖ 여러 명령을 반복 수행해야 할 때 스크립트 파일로 저장하여 실행
- ❖ 예제

```
#!/bin/bash  
# My First Script Program
```

```
echo I love UNIX !  
pwd
```

bash 셸로 실행

```
$ bash test_script  
I love UNIX!  
/home/user1  
$ chmod +x test_script  
$ ./test_script  
I love UNIX!  
/home/user1
```

실행파일로 권한변경 후에  
직접실행

# 스크립트 파일의 구성요소 - 쉘 프로그램

- ❖ 각 쉘이 제공하는 프로그램을 위한 구문
- ❖ 쉘 변수, 인자처리, 각종 연산자, 제어문 등 포함
- ❖ 시스템이 사용하는 스크립트의 예 :

배시셸 환경설정 스크립트 파일  
/etc/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.
# are we an interactive shell?
if [ "$PS1" ]; then
  if [ -z "$PROMPT_COMMAND" ]; then
    case $TERM in
      xterm*|vte*)
        if [ -e /etc/sysconfig/bash-prompt-xterm ]; then
          PROMPT_COMMAND=/etc/sysconfig/bash-prompt-xterm
        elif [ "${VTE_VERSION:-0}" -ge 3405 ]; then
          PROMPT_COMMAND="__vte_prompt_command"
        else
          PROMPT_COMMAND='printf "\033]0;%s@%s:%s\007" "${USER}" "${HOSTNAME%%.*}" "${PWD/#$HOME/~}"'
```

## 02. 쉘 변수 사용하기

❖ 변수 : 프로그램에서 처리하는 다양한 정보를 저장하는 곳

❖ 종류

- 쉘 변수 : 현재 쉘에서만 사용 가능
- 환경변수 : 모든 쉘에서 사용 가능

❖ 지정 방법

- 변수명=값
- 변수명과 값 사이에 공백이 있어서는 안된다.

```
$ a=5  
$ echo $a  
5  
$ export a
```

쉘 변수

환경 변수

# 셸 특수문자 및 명령 처리

## ❖ 인용부호 : 셸 특수문자의 의미를 없애기 위해 사용

인용 부호	기능	사용법
작은 따옴표 ( ' '	모든 특수문자들이 해석되는 것을 막음	\$ test=100 \$ echo '\$test' \$test
큰 따옴표 ( " "	변수나 명령의 대체만 허용	\$ echo "\$test" 100
역슬래시 (\)	단일 문자가 해석되는 것을 막음	\$ echo \\$test \$test

## ❖ 명령 대체 : 명령 실행 결과를 문자열로 변환

기호	사용법
역따옴표 ( ` ` )	\$ echo `date` Sunday, April 15, 2012 11:05:06 AM KST
\$(명령)	\$ echo \$(date) Sunday, April 15, 2012 11:15:11 AM KST



---

*다음 강좌에 계속*