UNIX 및 실습

13장 보충 – bash(2)

반목문 (1)

- for
  - 정해진 횟수의 반복

```
for 변수 in 단어목록
do
      명령(들)
done
```

25

```
[kgu@lily ch13]$ cat forloop.bash
#!/bin/bash
# Scriptname: forloop
for pal in Tom Dick Harry Joe
do
      echo "Hi $pal"
done
echo "Out of loop"
[kgu@lily ch13]$ chmod +x forloop.bash
[kgu@lily ch13]$ ./forloop.bash
Hi Tom
Hi Dick
Hi Harry
Hi Joe
Out of loop
```

```
[kgu@lily ch13]$ cat mylist
kgu
guest
lecafe
kea1024
```

```
[kgu@lily ch13]$ cat mailer.bash
#!/bin/bash
# Scriptname: mailer
for person in $(cat mylist)
do
      mail $person < letter
      echo $person was sent a letter.
done
echo "The letter has been sent."
[kgu@lily ch13]$ chmod +x mailer.bash
[kgu@lily ch13]$ ./mailer.bash
kgu was sent a letter.
guest was sent a letter.
lecafe was sent a letter.
kea1024 was sent a letter.
The letter has been sent.
```

26

27

```
[kgu@lily ch13]$ cat backup.bash
#!/bin/bash
# Scriptname: backup
# Purpose: Create backup files and
      store
# them in a backup directory.
#
dir=/home/kgu/2013U1/backup_sc
for file in *.bash
do
      if [ -f $file ]
      then
            cp $file $dir/$file.bak
            echo "$file is backed up in
      $dir"
      fi
done
```

반복문 (2)

- 단어목록에서의 $*와 $@ 변수

28

```
[kgu@lily ch13]$ cat greet.bash
#!/bin/bash
# Scriptname: greet
for name in $*        # same as for name in $@
do
      echo Hi $name
done
[kgu@lily ch13]$ chmod +x greet.bash
[kgu@lily ch13]$ ./greet.bash
[kgu@lily ch13]$ ./greet.bash kgu guest test
Hi kgu
Hi guest
Hi test
```

29

```
[kgu@lily ch13]$ cat permx.bash
#!/bin/bash
# Scriptname: permx

for file          # Empty wordlist
do

if [[ -f $file  && ! -x $file ]]
then
              chmod +x $file
              echo $file now has execute permission
fi
done
[kgu@lily ch13]$ chmod +x permx.bash
[kgu@lily ch13]$ ./permx.bash *.bash
```

반복문 (3)

- while
  - 다음에 따라오는 명령을 평가해서 종료상태가 0이면 반복문 본문 수행

```
while 명령
do
      명령(들)
done
```

30

```
[kgu@lily ch13]$ cat num.bash
#!/bin/bash
# Scriptname: num
num=0                    # Initialize num
while (( $num < 10 ))    # or  while [ num -lt 10 ]
do
      echo  -n "$num "
      let num+=1         # Increment num
done
echo -e "\nAfter loop exits, continue running here"
[kgu@lily ch13]$ chmod +x num.bash
[kgu@lily ch13]$ ./num.bash
0 1 2 3 4 5 6 7 8 9
After loop exits, continue running here
```

31

```
[kgu@lily ch13]$ cat quiz.bash
#!/bin/bash
# Scriptname: quiz
echo "해양컴퓨터공학과는 몇년도에 설립되었는가?"
read answer
while [[ "$answer" != "2012" ]]
do
      echo "Wrong try again!"
      read answer
done
echo You got it!
```

32

```
[kgu@lily ch13]$ cat sayit.bash
#!/bin/bash
# Scriptname: sayit
echo  Type q to quit.
go=start
while [ -n "$go" ]   # Make sure to double quote the variable
do
      echo -n "I love you. "
      read word
      if [[ $word == [Qq] ]]
      then    # [ "$word" = q -o "$word" = Q ]  Old style
              echo "I'll always love you!"
              go=
      fi
done
[kgu@lily ch13]$ ./sayit.bash
Type q to quit.
I love you. me, too
I love you.
I love you. q
I'll always love you!
```

문자열이 null인지 확인

문자열을 null로 수정

반복문 (4)

- until
  - until 다음에 나오는 명령이 실패해야 반복문이 실행

```
until 명령
do
      명령(들)
done
```

33

```
[kgu@lily ch13]$ cat hour.bash
#!/bin/bash
# Scriptname: hour

let hour=0
until (( hour > 24 )) # or [ $hour -gt 24 ]
do
      case "$hour" in
      [0-9]|1[0-1])            echo "Good morning!"
                ;;
      12)       echo "Lunch time."
                ;;
      1[3-4])   echo "Siesta time."
                ;;
      1[5-8])   echo "Good afternoon."
                ;;
      19)       echo "Good evening."
                ;;
      *)        echo "Good night."
                ;;
      esac
      let hour+=1    # Don't forget to increment the hour
done
```

반복문 (5)

- select 명령과 메뉴
  - here 문서를 이용한 메뉴 만들기와 별도로 select 반복문 제공
  - 숫자(일련번호)를 선택항목으로 사용하는 메뉴를 표준 오류로 출력하고, PS3을 이용하여 사용자 입력을 기다림
  - 사용자가 입력한 값은 REPLY에 저장
  - Ctrl+C가 눌러질 때까지 반복 실행

34

```
[kgu@lily ch13]$ cat runit.bash
#!/bin/bash
# Scriptname: runit

PS3="Select a program to execute: "
select program  in 'ls -F' pwd date
do
     $program
done
[kgu@lily ch13]$ ./runit.bash
1) ls -F
2) pwd
3) date
Select a program to execute: 1
args.bash*     databasefile   greeting.bash*
…
backup.bash*   forloop.bash*  greeting2.bash*
Select a program to execute: 2
/home/kgu/2013U1/ch13
Select a program to execute: 3
2013. 06. 03. (월) 10:59:51 KST
Select a program to execute: 4
Select a program to execute: 5
Select a program to execute: c
Select a program to execute: ^C
[kgu@lily ch13]$
```

반복문 (6)

35

```
[kgu@lily ch13]$ cat goodboys.bash
#!/bin/bash
# Scriptname name: goodboys

PS3="Please choose one of the three boys : "
select choice in tom dan guy
do
      case $choice in
      tom)
                echo Tom is a cool dude!
                break;;         # break out of the select loop
      dan | guy )
                echo Dan and Guy are both wonderful.
                break;;
      *)
                echo "$REPLY is not one of your choices" 1>&2
                echo "Try again."
                ;;
      esac
done
[kgu@lily ch13]$ ./goodboys.bash
1) tom
2) dan
3) guy
Please choose one of the three boys : 4
4 is not one of your choices
Try again.
Please choose one of the three boys : 5
5 is not one of your choices
Try again.
Please choose one of the three boys : 2
Dan and Guy are both wonderful.
```

36

```
[kgu@lily ch13]$ cat ttype.bash
#!/bin/bash
# Scriptname: ttype
# Purpose: set the terminal type
# Author: Andy Admin

COLUMNS=60
LINES=1
PS3="Please enter the terminal type: "
select choice in wyse50 vt200 xterm sun
do
      case $REPLY in
      1)
                export TERM=$choice
                echo "TERM=$choice"
                break;;             # break out of the select loop
      2 | 3 )
                export TERM=$choice

                echo "TERM=$choice"
                break;;
      4)
                export TERM=$choice
                echo "TERM=$choice"
                break;;
      *)
                echo -e "$REPLY is not a valid choice. Try again\n"
      1>&2
                REPLY=     # Causes the menu to be redisplayed
                                ;;
      esac
done
[kgu@lily ch13]$ ./ttype.bash
1) wyse50
2) vt200
3) xterm
4) sun
Please enter the terminal type: 2
TERM=vt200
```

반복문 (7)

- 제어명령 : shift
  - 매개변수 목록에서 지정한 횟수만큼 왼쪽으로 이동

37

```
[kgu@lily ch13]$ cat shifter.bash
#!/bin/bash
# Scriptname: shifter
set joe mary tom sam
shift
echo $*

set $(date)
echo  $*

shift 5
echo  $*

shift 2
[kgu@lily ch13]$ ./shifter.bash
mary tom sam
2013. 06. 03. (월) 11:10:11 KST
KST
```

38

```
[kgu@lily ch13]$ cat dater.bash
#!/bin/bash
# Scriptname: dater
# Purpose: set positional parameters with the set command
# and shift through the parameters.

set $(date)
while (( $# > 0 )) # or  [ $# -gt 0 ]  Old style
do
      echo $1
      shift
done
[kgu@lily ch13]$ ./dater.bash
2013.
06.
03.
(월)
11:12:22
KST
```

39

```
[kgu@lily ch13]$ cat doit.bash
#!/bin/bash
# Name: doit
# Purpose: shift through command line arguments
# Usage: doit [args]
while (( $# > 0 ))  # or [ $# -gt 0 ]
do
      echo  $*
      shift
done
[kgu@lily ch13]$ ./doit.bash a b c d e f
a b c d e f
b c d e f
c d e f
d e f
e f
f
```

반복문 (8)

- 제어명령 break
  - 반복문을 즉시 탈출하고자 하는 경우
  - 프로그램 종료시 exit 이용

40

```
[kgu@lily ch13]$ cat loopberak.bash
#!/bin/bash
# Scriptname: loopbreak

while true; do
      echo Are you ready to move on\?
      read answer
      if [[  "$answer"  ==  [Yy] ]]
      then
           break
      else
         :  #  ....commands...
      fi
done
printf "Here we are\n"
[kgu@lily ch13]$ ./loopberak.bash
Are you ready to move on?
n
Are you ready to move on?
y
Here we are
```

- 제어명령 continue
  - 지정한 조건이 참이면 반복문의 시작 부분으로 이동

41

```
[kgu@lily ch13]$ cat mailm.bash
#!/bin/bash
# Scriptname: mailem
# Purpose: To send a list
for name in $(cat  mylist)
do
      if [[ $name == kgu ]]
      then
           continue
      else
           mail $name < letter
      fi
done
```

반복문 (9)

- 중첩 반복문 제어
  - break나 continue에 정수 매개변수를 주어 반복문 사이 이동 가능

42

```
[kgu@lily ch13]$ cat months.bash
#!/bin/bash
# Scriptname: months
for month in Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
do
      for week in  1 2 3 4
      do
                echo -n "Processing the month of $month. OK? "
                read ans
                if [ "$ans" = n  -o -z "$ans" ]
                then
                                continue 2
                else
                        echo -n "Process week $week of $month? "
                        read ans
                        if [ "$ans" = n -o -z "$ans"  ]
                        then
                                continue
                        else
                                echo "Now processing week $week of $month."
                                sleep 1
                                # Commands go here
                                echo  "Done processing..."
                        fi
                fi
      done
done
```

```
[kgu@lily ch13]$ ./months.bash
Processing the month of Jan. OK? y
Process week 1 of Jan? y
Now processing week 1 of Jan.
Done processing...
Processing the month of Jan. OK? y
Process week 2 of Jan? y
Now processing week 2 of Jan.
Done processing...
Processing the month of Jan. OK? y
Process week 3 of Jan? y
Now processing week 3 of Jan.
Done processing...
Processing the month of Jan. OK? y
Process week 4 of Jan? y
Now processing week 4 of Jan.
Done processing...
Processing the month of Feb. OK? n
Processing the month of Mar. OK? n
Processing the month of Apr. OK? n
Processing the month of May. OK? y
Process week 1 of May? y
Now processing week 1 of May.
Done processing...
Processing the month of May. OK? n
Processing the month of Jun. OK? y
Process week 1 of Jun? y
Now processing week 1 of Jun.
Done processing...
Processing the month of Jun. OK? y
Process week 2 of Jun? n
Processing the month of Jun. OK? y
Process week 3 of Jun? n
Processing the month of Jun. OK? n
Processing the month of Jul. OK? n
Processing the month of Aug. OK? n
Processing the month of Sep. OK? n
Processing the month of Oct. OK? n
Processing the month of Nov. OK? n
Processing the month of Dec. OK? n
```

반복문 (10)

- 반복문의 출력을 파일로 리다이렉션

43

```
[kgu@lily ch13]$ cat numberit.bash
#!/bin/bash
# Program name: numberit
# Put line numbers on all lines of memo
if (( $# <  1 ))
then
      echo "Usage: $0 filename " >&2
      exit 1
fi

count=1            # Initialize count

cat $1 | while read line
# Input is coming from file provided at command line
do
   ((count == 1)) && echo "Processing file $1..." >
      /dev/tty
   echo -e "$count\t$line"
   let count+=1
done > tmp$$       # Output is going to a temporary file
mv tmp$$ $1
[kgu@lily ch13]$ ./numberit.bash mylist
Processing file mylist...
[kgu@lily ch13]$ cat mylist
1     kgu
2     guest
3     lecafe
[kgu@lily ch13]$ cat tmp$$
cat: tmp14190: 그런 파일이나 디렉터리가 없습니다
```

자동으로 만든 임시 파일

- 반복문의 출력을 유닉스/리눅스 명령으로 파이프하기

44

```
[kgu@lily ch13]$ cat loop_pipe.bash
#!/bin/bash
for i in 7 9 2 3 4  5
do
     echo  $i
done | sort -n
[kgu@lily ch13]$ ./loop_pipe.bash
2
3
4
5
7
9
```

반복문 (11)

- 백그라운드에서 반복문 사용

45

```
[kgu@lily ch13]$ cat bg_loop.bash
#!/bin/bash
for person in bob jim joe sam
do
     mail $person < letter
done &
```

- IFS와 반복문

46

```
[kgu@lily ch13]$ cat runit2.bash
#/bin/bash
# Script is called runit.
# IFS is the internal field separator and defaults to
# spaces, tabs, and newlines.
# In this script it is changed to a colon.
names=Tom:Dick:Harry:John
oldifs="$IFS"       # Save the original value of IFS

IFS=":"

for persons in $names
do
      echo  Hi $persons
done

IFS="$oldifs"        # Reset the IFS to old value

set Jill Jane Jolene # Set positional parameters
for girl in $*
do
      echo Howdy $girl
done
[kgu@lily ch13]$ ./runit2.bash
Hi Tom
Hi Dick
Hi Harry
Hi John
Howdy Jill
Howdy Jane
Howdy Jolene
```

함수 (1)

- 함수에 관한 주요 규칙
  - 쉘에서는 다음 순서로 사용자의 입력을 판단한다.
    - 별명
    - 함수
    - 내장 명령
    - 실행 프로그램
  - 함수는 사용하기 전에 정의되어야 함
  - 함수는 현재 환경하에서 실행
    - 스크립트, 변수 공유
  - 함수 안에서 exit을 실행하면 스크립트 자체가 종료됨
  - 함수가 종료하면 함수를 호출한 곳으로 복귀
- 함수 내의 return 문은 가장 최근에 실행한 명령의 종료 상태값이거나 지정한 매개변수 값
- 내장명령 export –f로 함수를 자식 쉘에게 export 할수 있음
- declare –f로 함수와 함수 정의 출력 가능. declare –F는 함수 이름만 출력
- 트랩(trap)은 변수처럼 함수 간에도 전역
- 함수가 다른 파일에 정의되어 있다면 . (source) 명령으로 불러들일 수 있음
- 함수는 자신을 호출할 수도 있음

함수 (2)

```
function 함수이름 { 명령 ; 명령 ; }
```

```
function dir { echo "Directories: "; ls -l |
      awk '/^d/ {print $NF}';}
```

```
unset -f 함수이름
```

```
export -f 함수이름
```

47

```
[kgu@lily ch13]$ cat checker2.bash
#!/bin/bash
# Scriptname: checker2
# Purpose: Demonstrate function and arguments

function Usage { echo "error: $*" 2>&1; exit 1; }

if (( $# != 2 ))
then
      Usage "$0: requires two arguments"
fi
if [[ ! ( -r $1 && -w $1 ) ]]
then
      Usage "$1: not readable and writeable"
fi
echo The arguments are: $*
# <  Program continues here >
[kgu@lily ch13]$ ./checker2.bash
error: ./checker2.bash: requires two arguments
[kgu@lily ch13]$ ./checker2.bash checker.bash ttt
The arguments are: checker.bash ttt
```

48

```
[kgu@lily ch13]$ cat do_inc.bash
#!/bin/bash
# Scriptname: do_inc
 increment ()  {
    local sum;      # sum is known only in this function
    let "sum=$1 + 1"
    return $sum     # Return the value of sum to the script
}
echo  -n "The sum is "
increment 5     # Call function increment; pass 5 as a
                # parameter; 5 becomes $1 for the increment
                # function

echo $?         # The return value is stored in $?
echo  $sum      # The variable "sum" is not known here
[kgu@lily ch13]$ ./do_inc.bash
The sum is 6
```

49

```
[kgu@lily ch13]$ cat do_square.bash
#!/bin/bash
# Scriptname: do_square
function square {
  local sq      # sq is local to the function
  let "sq=$1 * $1"
  echo "Number to be squared is $1."
  echo "The result is $sq "
}
echo "Give me a number to square. "
read number
value_returned=$(square $number) # Command substitution
echo    "$value_returned"
[kgu@lily ch13]$ ./do_square.bash
Give me a number to square.
17
Number to be squared is 17.
The result is 289
```

함수 (3)

50

```
[kgu@lily ch13]$ cat dbfunctions.bash
function addon () {     # Function defined in file dbfunctions
while true
do
echo "Adding information "
echo "Type the full name of employee "
read name
echo "Type address for employee "
read address
echo "Type start date for employee (4/10/88 ) :"
read startdate
echo $name:$address:$startdate
echo -n "Is this correct? "
read ans
case "$ans"  in
[Yy]*)
       echo "Adding info..."
       echo $name:$address:$startdate>>datafile
       sort -u datafile -o datafile
       echo -n "Do you want to go back to the main menu? "
       read ans
       if [[ $ans == [Yy] ]]
       then

              return     # Return to calling program
       else

              continue   # Go to the top of the loop
       fi
       ;;
*)
       echo "Do you want to try again? "
       read answer
       case "$answer" in
       [Yy]*) continue;;
       *) exit;;
       esac

       ;;
esac
done
}       # End of function definition
```

51

```
[kgu@lily ch13]$ cat mainprog.bash
#!/bin/bash
# Scriptname: mainprog
# This is the main script that will call the function, addon

# datafile=$HOME/bourne/datafile
datafile=./mylist
source dbfunctions.bash    # The file is loaded into memory

if [ ! -e $datafile ]
then
       echo "$(basename $datafile) does not exist" >&2
       exit 1
fi

echo "Select one: "
cat << EOFMENU
       [1] Add info
       [2] Delete info
       [3] Update info
       [4] Exit
EOFMENU

read choice
case $choice in
1)     addon       # Calling the addon function
              ;;
2)     delete      # Calling the delete function
              ;;
3)     update
              ;;
4)
       echo Bye
       exit 0
              ;;
*)     echo Bad choice
       exit 2
              ;;
esac
echo Returned from function call
echo The name is $name
# Variable set in the function are known in this shell.
```

디버깅

디버깅 옵션

명령	옵션	기능
bash –x scriptname	echo 옵션	스크립트 각 행을 변수 치환하여 출력한 후 실행
bash –v scriptname	verbose 옵션	스크립트 각 행을 내용 그대로 출력한 후 실행
bash –n scriptname	noexec 옵션	명령을 실행시키지 않고 스크립트 검사
set -x	echo 켜기	스크립트 실행과정을 추적
set +x	echo 끄기	추적 기능 끄기