# 리눅스시스템 및 응용

## Week 8

# Shell Programming

# 학습목표

- 쉘 변수
- 쉘 프로그램의 제어문
- 쉘 프로그램에서 명령어

# 예제

```
sh      <enter>
$ for i in *                    : 현재 디렉토리의 모든 파일
> do                            : for의 do
> if grep -l root $i            : if문
> then
> more $i                       : 조건 만족시 실행
> fi                            : if의 닫음
> done                          : do 의 닫음
$
```

- 이전  예제를 파일로 작성하면 쉘프로그램이 된다

```
# 예제  1 #으로 라인을 시작하면 commen문 주석문

for i in *
do
if grep -l root $i then
more $i fi done
exit 0
```

# 쉘 스크립트 실행 방법

- 쉘 스크립트 파일 실행 방법
  - 1. 스크립트 실행 명령어로 실행

```
/bin/sh exam1.sh
```

  - 2. 스크립트파일을 실행가능으로 변경 후 실행

```
chmod +x exam1.sh
/exam1.sh <return>
```

# Shell 프로그램에서 사용 명령어

- Shell 프로그램에서 사용 명령어
  - Shell 내장명령어 + Linux 명령어
  - 프로그래밍 언어로서 일반적인 형식
  - 변수, 제어문,리스트, 함수, 내장명령어

# 쉘 변수 선언 및 호출

- 선언
  - 별도 선언 없음(최초에는 변수명만 표기)
  - 일반적으로 character string변수임

- 변수의 호출 Reference
  - $변수명
  - Ex:
    - foo='today'
    - echo $foo

```
#!/bin/sh

foo="Hello World"
echo $foo

#unset foo
foo=

echo $foo
```

# 쉘 변수 선언 및 호출

- 변수의 호출의 다른 예
  - $변수명, "$변수명"
  - '$변수명' ??
  - 변수 사용예

  – Ex:
    - myvar="Hi there"
    - echo $myvar
    - echo ${myvar}
    - echo "$myvar"
    - echo '$myvar'
    - echo ₩$myvar
    - echo Enter some text
    - read myvar
    - echo '$myvar' now equals $myvar

# 쉘 변수 선언 및 호출

- Ex: 출력
  - Hi there
  - Hi there Hi there
  - $myvar
  - $myvar
  - Enter some text Hello World
  - $myvar now equals Hello World

# Reserved Variable for System

- Reserved Variable for System
  - $HOME : 사용자의 홈 디렉토리
  - $PATH : 디렉토리 목록
  - $PS1, $PS2 : 프롬프트
  - $0 : Shell 스크립트/명령어 이름
  - $# :전달된 파라메터 수
  - $$ : Shell 스크립트 프로세스 번호
  - echo 명령어로 실행

  - 직접해보기 : echo 위변수명

# 쉘 프로그램 외부 파라메터

- Shell 외부 입력 파라메터
  - $1, $2, …
  - $0 : 쉘프로그램 이름(실행파일)
  - $* : 모든 파라메터
  - Ex:

```
#!/bin/sh salutation="Hello"
echo $salutation
echo "The program $0 is now running"
echo "The second parameter was $2"
echo "The first parameter was $1"
echo "The parameter list was $*"
echo "The user's home directory is $HOME"
echo "Please enter a new greeting"
read salutation
echo $salutation
echo "The script is now complete"
exit 0
```

# if 제어문

- if statement
  - 형식
    - **if** condition
    - then
    - statement
    - else
    - statement
    - **fi**
  - conditio문:
    - test statement
    - test –f fred.c
    - [ -f fred.c ]

- if 문의 예

```
echo "Is it morning? Please answer yes or no"
read timeofday

if [ $timeofday = "yes" ]
  then
        echo "Good morning"
   else
        echo "Good afternoon"
fi
exit 0
```

# 조건문 : test

- Test 문의 사용

| | |
|---|---|
| -b file | : file is a block special file |
| -c file | : file is a character special file |
| -d file | : file exists and is a directory |
| -f file | : file exists and is a file |
| -g file | : file has the set-group-id bit set |
| -k file | : file has the sticky bit set |
| -p file | : file is a named pipe |
| -r file | : file is readable |
| -s file | : file is greater than 0 byte |
| -t n | : n is a file descriptor, 0=keyboard input |
| -u file | : file has the set-user-id |
| -w file | : file is writable |
| -x file | : file is executable |

# 스트링 비교와 산술비교

- String 비교
  - string1 = string2
  - string1 != string2
  - -n string : null 아니면 참
  - -z string : null 이면 참

- 산술 비교
  - exp1 –eq exp2
  - exp1 –ne exp2
  - exp1 –gt exp2  ef: -ge, -lt –le
  - ! expression

# for 제어문

- for statement
  - 형식
    - **for** variable in values
    - **do**
    - statement
    - …
    - **done**

  - Ex:

```
for foo in bar fud 43
do
  echo $foo
done
exit 0
```

*화면출력*

```
bar
fud
43
```

# while 제어문

- while statement
  - 형식
    - **while** condition
    - **do**
    - statement
    - …
    - **done**
  - Ex:

```
echo "Enter Password"
read trythis

while [ "$trythis" != "secret" ]
do
 echo "Sorry, try again" read trythis
done
exit 0
```

# until 제어문

- until statement
  - 형식
    - until condition
    - do
    - statement
    - …
    - done

  - Ex:

```
until who | grep "$1" > /dev/null
do
 sleep 60 done
# Now ring the bell and announce the
unexpected user.
echo -e \\a
echo "***** $1 has just logged in *****"
exit 0
```

# case 제어문

- case statement
  - 형식
    - **case** variable **in**
    - pattern [ | pattern] ... ) statement **;;**
    - pattern [ | pattern] ... ) statement **;;**
    - ...
    - **esac**
  - Ex:

- Ex:

```
#!/bin/sh
echo "Is it morning? Please answer yes or no"
read timeofday
case $timeofday in
  "yes") echo "Good Morning";;
  "no" ) echo "Good Afternoon";;
  "y"  ) echo "Good Morning";;
  "n"  ) echo "Good Afternoon";;
  *  ) echo "Sorry, answer not recognised";;
esac exit 0
```

– Ex:

```
#!/bin/sh
echo "Is it morning? Please answer yes or no" read
timeofday
case $timeofday in
  "yes" | "y" | "Yes" | "YES" ) echo "Good Morning";;

  "n*" | "N*" ) echo "Good Afternoon";;
  * ) echo "Sorry, answer not recognised";;
esac
exit 0
```

# 명령어 AND 결합

- 명령어 AND 결합
  - 앞명령어 성공시 뒷명령어 수행
  - 형식: statement1 && statement2 && …
  - Ex:

```
#!/bin/sh
touch file_one
rm -f file_two
if [ -f file_two ] && echo "hello" || echo "there"
then
    echo "in if"
else
    echo "in else"
fi
exit 0
```

# 명령어 OR 결합

- 명령어 OR 결합
  - 형식: statement1 || statement2 || …
  - Ex:

```
#!/bin/sh
rm -f file_one
if [ -f file_one ] || echo "hello" || echo "there"
then
    echo "in if"
else
     echo "in else"
fi
exit 0
```

# 쉘 프로그램의 함수

- 함수문 function statement
  - 형식
    - function_name ( ) {
    - statement
    - …
    - }

# 쉘 프로그램의 함수

- Ex:

```
#!/bin/sh
yes_or_no() {
 echo "Parameters : $*"
 while true
 do
  echo  "Enter yes or no“
  read x
  case "$x" in
   “ y” | ”yes” ) return 0;;
   “n” | “no” )  return 1;;
   * )  echo "Answer yes or
  no”;;
  esac
  done
  }#yes_or_no()
```

```
echo "Original parameters are: $*"
if yes_or_no "k1 k2 k3 k4 5k"
then
 echo "Hi $1"
 else
 echo "Never mind"
fi
exit 0
```

# 쉘 프로그램내 내장명령어

- break statement
  - 형식:

    break
  - Shell 스트립트 수행시 종료

- Null statement
  - 형식 :

    :
  - Null 명령어, true로 실행 결과

# 쉘 프로그램내 내장명령어

- • (dot) 명령어
  - 명령어 수행:
  - Ex: **`. ./shell_script`**

- echo statement
  - String 프린트
  - C언어에서 printf와 유사

# 쉘 프로그램내 내장명령어

- eval statement
  - 변수에 대한 평가
  - Ex:

```
foo=10
echo $foo                  10
x=foo
y='$'$x
echo $y
eval y='$’$x
echo $y                    10
```

# 쉘 프로그램내 내장명령어

- exec statement
  - Shell 스크립트 내에서 명령어 수행
  - 스트립트의 다음 행부터 수행 안함
  - Ex:

```
#!/bin/sh
echo "first"
exec wall "Hi there this is test”
echo "second"
exit 0
```

# 쉘 프로그램내 내장명령어

- exit statement
  - 형식: **`exit n`**
  - Shell 스크립트 수행 종료
    - n=0 : 성공적인 종료
    - 1<n<125 : 사용 가능
    - n=126 : 파일이 실행 불가능
    - n=127 : 명령이 발견되지 못함
    - n<128 : 시그널 발생

# 쉘 프로그램내 내장명령어

- export statement
  - 지정된 변수를 다른 Shell 프로그램에서 유효하게 사용하도록 설정
  - 형식 : **export** shell변수
  - Ex: 두개의 프로그램 작성 후 export2.sh 테스트 !!
    - export1.sh 프로그램

```
echo Shell Program export1.sh
foo="The first meta-syntactic variable"
export bar="The second meta-syntactic variable"
./export2.sh
```

    - export2.sh 프로그램

```
echo Shell Program export2.sh
echo "$foo"
echo "$bar"
```

# 쉘 프로그램내 내장명령어

- expr statement
  - 인수를 수식으로 평가
  - Ex:
    - expr 327 + 431
    - 출력: 758

- set statement
  - Shell을 위한 파라메터 변수 설정
  - Ex:

```
#!/bin/sh
echo The date today date is $(date) set $(date)
echo The month is $2
exit 0
```

# 쉘 프로그램내 내장명령어

- shift statement
  - shell 프로그램 외부 변수(파라메타 변수) 값을 한단계씩 왼쪽으로
  - $2는 $1으로 , $3는 $2로, …
  - $0는 변경 불가
  - Ex: **`shift`**

- trap statement
  - Signal이 수신될 때 수행
  - shell 프로그램의 가장 앞부분에 미리 정의해 둬야 함
  - 반복문 수행시 혹은 장시간 수행시 signal수신하면 정의한 trap문 수행
  - 형식: **`trap command`** ***`signal`***

# trap 사용 예

```
# 이 예제는 반드시 sh 프로그램파일명 으로 실행 !!
trap 'rm –f /tmp/my_tmp_file_$$' INT
echo creating file /tmp/my_tmp_file_$$
date > /tmp/my_tmp_file_$$
echo "Press interrupt (Ctrl-C) to interrupt…."
while [ -f /tmp/my_tmp_file_$$ ]
do
  echo File exists sleep 1
done
echo The file no longer exists

trap –INT    # 이 부분을 trap : INT 로 수정해서 다시 run!!
echo creating file /tmp/my_tmp_file_$$

date > /tmp/my_tmp_file_$$
echo "Press interrupt (Ctrl-C) to interrupt...."
while [ -f /tmp/my_tmp_file_$$ ]
do
   echo File exists sleep 1
done
echo We never get here exit 0
```

# 쉘 프로그램 debugging

- 쉘 프로그램 debugging
  - 특별한 디버깅 도구 없음
  - 에러 있는 행의 번호를 출력
  - 그외 방법 : 명령라인 옵션
    - sh -n <script> : 형식 검사만,실행안함
    - sh -v <script> : 실행전 스크립트 출력
    - sh -x <script> : 처리후 명령라인 출력
  - 스크립트 내 set 옵션
    - set -o noexec  or set -n
    - set -o verbose or set -v
    - set -o xtrace  or set –x s
    - set -o nounset or set -u

# 자주 사용하는 쉘 내장 명령어

- cut
  - 파일의 특정 필트 추출
  - -d 에 delimeter 정의 가능
  - Ex:

    **grep root /etc/passwd | cut –f1,5 –d:**

    **ls –l | cut –c1-15,55-**

- paste
  - 출력 형식의 변경
  - Ex:

    **ls –a | paste - -**

    - 출력 라인 2개씩

# 자주 사용하는 쉘 내장 명령어

- sort
  - 하나 혹은 둘 이상 파일의 내용의 순서화
  - Ex:

    **`sort -t: +0 -2 /etc/passwd`**

    - delimeter **:** , 1번째부터 2번째 필드까지 출력

- 파일의 결합
  - join, merge, uniq

# 자주 사용하는 쉘 내장 명령어

- tr
  - 문자의 변환
  - Ex:

    **tr “[ ]” “[\012]” < chapter1 | sort| uniq –c**

    - **\012** : form feed, new line

# Programming Homework

- 과제
  - 다음과 같은 프로그램을 작성하시오
  - shell 명령어가 실행되는 디렉토리 아래의 파일과 디렉토리에 대하여
  - 파일만 그 이름을 대문자에서 소문자 소문자에서 대문자로 변경하는 프로그램
  - **filetr –F**
    - **fIlE1 -> FILE1 :** 모든 파일명을 대문자로 변경
  - **filetr –f**
    - **FilE2 -> file2 :** 모든 파일명을 소문자로 변경

# Question?