Mahasarakham University

มหาวิทยาลัยมหาสารคาม

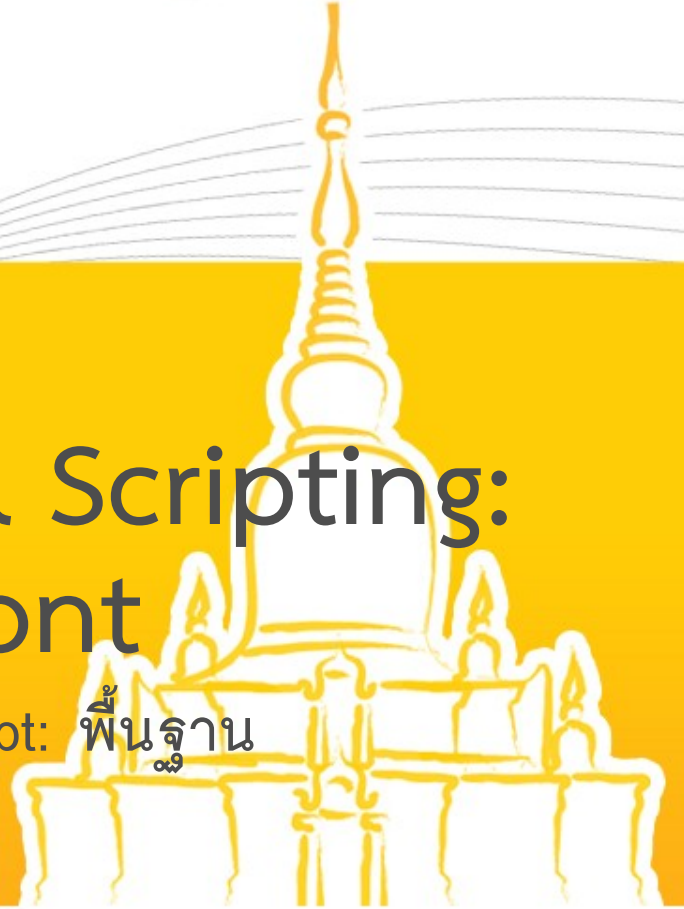
Heart of
the Northeast

www.msu.ac.th

Mahasarakham University
มหาวิทยาลัยมหาสารคาม

Introduction to Shell Scripting: The Basics: cont

แนะนำการทำงานของ Shell Script: พื้นฐาน



Heart of the Northeast

www.msu.ac.th

โดย อาจารย์โอฬาริก สุรินต๊ะ

Jan 18, 2017

Agenda

- Comments
- Basic bash command
  - find
  - cat, tac, rev
- Wildcards
- Case statement
- Basic Functions

Defined & Empty Variables

Defined variables

```
$ number=1
$ echo $number
=======
```

Empty variables

```
$ number
$ echo $number
```

Single and Multi Line Comments

- **Single line comment**

  # comment

  : comment

- **Multiline comment**

  : '

  Don't forget “space”

  Comment 1

  Comment 2

  '

Single and Multi Line Comments

- **Multiline comment**

<<COMMENT1 # or <<comment1

Your comment

Your comment

COMMENT1

Basic bash command: find

- Help

$ find --help

Usage: find [-H] [-L] [-P] [-Olevel] [-D help|tree|search|stat|rates|opt|exec] [path...] [expression]

default path is the current directory; default expression is -print

expression may consist of: operators, options, tests, and actions:

Basic bash command: find

Finding by name in current directory

- To find a file by name. This will be case sensitive

$ ls -a

```
.    function-return.sh    switch-input-argu.sh
..   switch-ex.sh          switch-prompt-user.sh
```

$ find -name 'function-return.sh'

```
./function-return.sh
```

Basic bash command: find

- To find a file by name, but ignore the case of query, type:

$ find -iname 'function-return.SH'

./function-return.sh

Basic bash command: find

- If you want to find all files that don't adhere to a specific pattern, you can invert the search with "-not" or "!"

$ find -not -name "switch*"

.

./function-return.sh

$ find -name "switch*"

./switch-prompt-user.sh

./switch-input-argu.sh

./switch-ex.sh

Basic bash command: find

- find in the current directory

$find . -name "switch*"

./switch-prompt-user.sh

./switch-input-argu.sh

./switch-ex.sh

Basic bash command: find

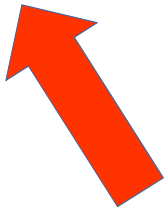
- find in specific directory

$find /home/pro/Desktop/ -name "switch*"

./switch-prompt-user.sh

./switch-input-argu.sh

./switch-ex.sh



Specific directory

Basic bash command: find

- Searching a file on your system

$ sudo find / -name "switch-ex.sh"

/home/pro/Desktop/1201377-Linux-Shell-Script/week3/switch-ex.sh

Basic bash command: find

- Finding by type
  - Some of the most common descriptors that you can use to specify the type of file are:
    - f: regular file
    - d: directory
    - l: symbolic link
    - c: character devices
    - b: block devices

Basic bash command: find

- f: regular file

$ ls -a

. function-return.sh switch-ex.sh switch-promt-user.sh

.. switch switch-input-argu.sh

$ find -name "switch*"

./switch-prompt-user.sh

./switch-input-argu.sh

./switch-ex.sh

./switch

$ find -name "switch*" -type f

./switch-prompt-user.sh

./switch-input-argu.sh

./switch-ex.sh

What is the difference?

Can you try this command

$ find -name "switch*" -type d

Basic bash command: find

- d: directory

$ find -type d

.

./prompt

./switch

$ find -type d -name "prompt"

./prompt

Basic bash command: find

```
$ find -not -name "switch*"
.
./test.txt
./prompt.txt
./prompt
./function-return.sh
```

Basic bash command: find

- Find with operator
  - And -a, -and
  - Or -o, or
  - Not -not

Basic bash command: find

```
$ ls
function-return.sh    switch                  switch-prompt-user.sh
prompt                switch-ex.sh            switch-test-user.sh
prompt.txt            switch-input-argu.sh    test.txt
```

** ค้นหา ไฟล์ที่ มีคำว่า "switch" และ "user" และเป็นนามสกุล ".sh" และต้องไม่มีคำว่า "prompt" ปรากฏในการค้นหา

คำตอบที่ต้องการ: **switch-test-user.sh**

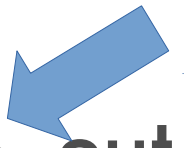
Basic bash command: find

```
$ find -name "switch*user*"
./switch-prompt-user.sh
./switch-test-user.sh
```

```
$ find -not -name "*prompt*"
./test.txt
./switch-input-argu.sh
./switch-test-user.sh
./function-return.sh
./switch-ex.sh
./switch
```

```
$ find -not -name "*prompt*" -and -not -name "*prompt*"
./switch-test-user.sh
```

Basic bash command: save output



This “>” sign

```
$ ping google.com > out_google.txt
$ cat out_google.txt
PING google.com (202.28.85.245) 56(84) bytes of data.
64 bytes from 202.28.85.245: icmp_seq=1 ttl=56 time=4.02 ms
64 bytes from 202.28.85.245: icmp_seq=2 ttl=56 time=3.98 ms
64 bytes from 202.28.85.245: icmp_seq=3 ttl=56 time=4.08 ms

--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 3.985/4.031/4.082/0.083 ms
```

Basic bash command: cat

- Cat, an acronym for concatenate, lists a file to *stdout*.
- When combined with redirection (> or >>), it is commonly used to concatenate files.

```
$ cat filename      # Lists the file
$ cat file.1 file.2 file.3 > file.123
# Combines three files into one
```

Basic bash command: cat, rev (reverse), tac (inverse)

```
$ cat file1.txt
This is line 1.
 This is line 2.


$ tac file1.txt
This is line 2.
 This is line 1.

$ rev file1.txt
.1 enil si sihT
 .2 enil si sihT
```

Wildcards

- A wildcard is a character that can be used as a substitute for any of a class of characters in a search, thereby greatly increasing the flexibility and efficiency of searches

Wildcards

- 3 Type of wildcards are used with Linux commands.
  - Star wildcard or asterisk (*)
  - Question mark wildcard (?)
  - Square Brackets wildcard ([ ])

Star wildcard (*)

- Star or asterisk (*) wildcard has the broadest meaning of any of the wildcards, as it can represent zero characters, all single characters or any string.

$ file *

function-return.sh: Bourne-Again shell script, ASCII text executable
prompt: directory
prompt.txt: empty
switch-input-argu.sh: Bourne-Again shell script, ASCII text executable
switch-test-user.sh: empty

Star wildcard (*)

```
$ ls *.txt *.sh
function-return.sh    switch-ex.sh    switch-prompt-user.sh    test.txt
prompt.txt            switch-input-argu.sh    switch-test-user.sh
```

```
$ rm *.txt (rm = remove)
```

Question mark wildcard (?)

- The question mark (?) is used as a wildcard character in shell commands to represent exactly one character, which can be any single character.

$ ls promp?.txt

prompt.txt

Square brackets wildcard ([ ])

- The square brackets can represent any of the characters enclosed in the brackets.

$ file *[prompt]*

prompt: directory

prompt.txt: empty

switch-ex.sh: Bourne-Again shell script, ASCII text executable

Square brackets wildcard ([ ])

- When a hyphen (-) is used between two characters in the square brackets wildcard, it indicates a range inclusive of those two characters.
  - $ file [a-f]*
  - $ file *[0-9]*
  - $ file [a-cst]*
    - names begin with any letter from **a** through **c** or begin with **s** or **t**
  - $ file [a-cx-z]*
    - Names begin with the first three or the final three lower case letters
  - $ file jones[0-9][0-9][0-9]
    - display all filenames that consist of jones followed by a three-digit number

Curly brackets wildcard ({ })

- Terms are separated by commas and each term must be the name of something or a wildcard.

```
$ ls
function-return.sh  switch  switch-prompt-user.sh
prompt      switch-ex.sh  switch-test-user.sh
prompt.txt      switch-input-argu.sh  test.txt
```

```
$ ls *.{sh,txt}
function-return.sh    switch-ex.sh              switch-prompt-user.sh    test.txt
prompt.txt            switch-input-argu.sh      switch-test-user.sh
```

Retrieve a package from Internet

- Retrieve a package and install it on your system

  $ sudo apt-get install vlc

- Uninstall an application and all dependencies package

  $ sudo apt-get autoremove vlc

Download web pages, files and images from the web

- wget stands for “web get”.
- It is a command-line utility which downloads files over a network.

- Installing wget

$ sudo apt-get install wget

Download web pages, files and images from the web

- Single file download

$ wget http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz

- Download file with different name (-O)

$ wget -O wget.zip http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz

- Download multiple file with http and ftp protocol

$ wget http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz ftp://ftp.gnu.org/gnu/wget/wget-1.10.1.tar.gz.sig

Download web pages, files and images from the web

- Resume uncompleted download

$ wget -c http://mirrors.hns.net.in/centos/6.3/isos/x86_64/CentOS-6.3-x86_64-LiveDVD.iso

- Download files in background

$ wget -b http://mirrors.hns.net.in/centos/6.3/isos/x86_64/CentOS-6.3-x86_64-LiveDVD.iso

Case Statement

- **Syntax of bash case statement**

```
case expression in
        pattern1 )
                statements ;;
        pattern2 )
                statements ;;
        ...
esac
```

Case Statement

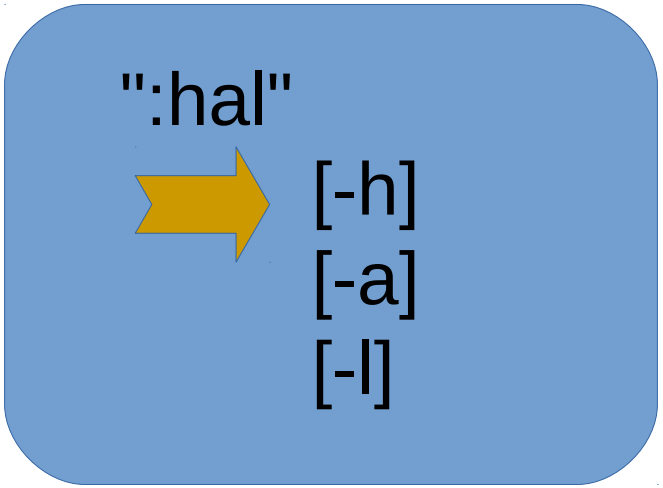
- Following are the key points of bash case statements:
  - Case statement first expands the expression and tries to match it against each pattern.
  - When a match is found all of the associated statements until the double semicolon (;;) are executed.
  - After the first match, case terminates with the exit status of the last command that was executed.
  - If there is no match, exit status of case is zero.

Case Statement

```
#!/bin/bash

printf 'Which Linux distribution do you know? '
read DISTR

case $DISTR in
        ubuntu)
                echo "I know it! It is an operating system based on Debian."
                ;;
        centos|rhel)
                echo "Hey! It is my favorite Server OS!"
                ;;
        *)
                echo "Hmm, seems i've never used it."
                ;;
esac
```

Multiple pattern

Case Statement: Sample 1




```
#!/bin/bash

all=false
long=false

while getopts ":hal" option; do
   case $option in
      h) echo "usage: $0 [-h] [-a] [-l] file ..."; exit ;;
      a) all=true ;;
      l) long=true ;;
      ?) echo "Invalid option: -$OPTARG"; exit ;;
   esac
done

echo $all
```

Case Statement: Sample 2 Prompt user with Yes or No

```
#!/bin/bash

echo -n "Do you agree with this? [yes or no]: "
read yno
case $yno in

        [yY] | [yY][Ee][Ss] )
                echo "Agreed"
                ;;

        [nN] | [nN][Oo] )
                echo "Not agreed, you can't proceed the installation";
                exit 1
                ;;
        *) echo "Invalid input"
             ;;
esac
```

Case Statement: Sample 3

```
#!/bin/bash

# option with argument
while getopts ":a:b:" opt; do
   case $opt in
      a)
         echo "-$opt was triggered, Parameter: $OPTARG" >&2
         ;;
      b)
         echo "-$opt was triggered, Parameter: $OPTARG" >&2
         ;;
```

Case Statement: Sample 3 (cont)

```
    \?)
        echo "Invalid option: -$OPTARG" >&2
        exit 1
        ;;
    :)
        echo "Option -$OPTARG requires an argument." >&2
        echo "usage: $0 [-a] arg [-b] arg"
        exit 1
        ;;
  esac
done
```

Basic Functions in bash

```
function function_name {
        Command...
}
```

Or

```
function_name() {
        Command...
}
```

To call a function with arguments:

```
function_name $arg1 $arg2
```

Case Statement: calling functions with arguments

```
#!/bin/bash

function func1() {
        echo "Function 1"
        val=$(($1 * $2))
        echo "$1 * $2 = $val"
}

echo "param1 = $1"
echo "param2 = $2"
func1 $1 $2
```

Returning value from bash functions

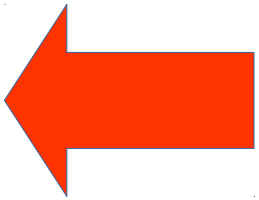
```
#!/bin/bash

# Setting a return status for a function
print_something() {
        echo "Hello $1"
        return 5
}

name="Mars"
print_something $name
print_something Jupiter
echo The previous function has a return value of $?
```

Returning value from bash functions

```
#!/bin/bash

# Setting a return value to a function
lines_in_file () {
        cat $1 | wc -l
}

num_lines=$( lines_in_file $1 )
echo The file $1 has $num_lines lines in it.
```

Returning value from bash functions

```
#!/bin/bash

# Setting a return status for a function
print_something() {
        echo "Hello $1"
        var=$1
        var1="Olarik"
        echo $var1
}

name="Mars"
newname=$(print_something $name)
echo $newname
```



What is the result of the program?
and
How many values can it returns?

Combining switch case and functions: sample 1

```
#!/bin/bash


function func1() {
    param1=$OPTARG
    echo "Function 1"
    echo "-$opt, param = $param1"
}
```

Combining switch case and functions: sample 1 (cont)

```
# option with argument
while getopts ":a:b:c:" opt; do
   case $opt in
      a)
         echo "-$opt was triggered, Parameter: $OPTARG" >&2
         echo "optarg = $OPTARG"
         func1 $opt $OPTARG
         ;;
...
...
```

References:

- http://wiki.bash-hackers.org/syntax/ccmd/case
- http://stackoverflow.com/questions/4554718/patterns-in-case-statement-in-bash-scripting
- http://unix.stackexchange.com/questions/20975/how-do-i-handle-switches-in-a-shell-script

References:

- https://www.digitalocean.com/community/tutorials/how-to-use-find-and-locate-to-search-for-files-on-a-linux-vps
- http://www.linfo.org/wildcard.html
- http://www.tecmint.com/10-wget-command-examples-in-linux/

http://linuxcommand.org/lc3_wss0090.php