ADVANCED SHELL SCRIPT

UNIX Programming 2014 Fall by Euiseong Seo

for Loop

- Syntax
  - for ((condition))
    do
      operation
      operation
    done
- Example
  - for ((a=1; a <= LIMIT; a++))
    do
      echo -n “$a”
    done

for…in Loop

- Syntax
  - for arg in [List]
    do
      operation
      operation
    done
- Example
  - for planet in Mercury Venus Earth Mars Jupiter
    do
      echo $planet
    done

for…in Loop

- Another example

```
FILES="/usr/sbin/privatepw
/usr/sbin/pwck
/usr/sbin/go500gw
/usr/bin/fakefile
/sbin/mkreiserfs
/sbin/ypbind"
for file in $FILES
do
if [ ! -e "$file" ]
then
        echo "$file does not exist."
        continue
fi
done
```

for…in Loop

- Another example

```
for arg in $@
do
    echo $arg
done
```

- List
  - Elements are separated with a white space
  - Wrapped by double quotation marks, the whole line will be considered as an element
  - Each line becomes an element when wrapped by double quotation marks

Arithmetic

- All bash variables are string valued
  - Bash does not distinguish between 1 and “1”
  - ```
    a=1
    b=2
    c=$a+$b
    echo $c?
    ```
- To force arithmetic operation use $((…))
  - ```
    c=$(($a+$b))
    echo $c
    ```

Command Substitution

- Command substitution reassigns the output of a command
- Syntax
  - `command`
- Example
  - ```
    dir_list=`ls`
    for file in dir_list
    do
       if [ -x $file ]; then
       echo "$file in `pwd` is executable"
       fi
    done
    ```
    ```

Scripting Best Practices

- Scripts should print a usage message and exit
  - Implement --help
- Validate inputs and sanity-check derived values
- Return an appropriate exit code
  - Zero for success
  - Non-zero for failure - Don't feel compelled to give every failure mode a unique exit code
- Use appropriate naming conventions for variables, scripts, and routines

Perl and Python

- Shell script is only for automating shell works
- If you want to do complicated tasks use professional scripting languages such as Perl or Python

Regular Expressions

- A sequence of characters that forms a search pattern
- Character types
  - Literal characters
    - Normal alpha-numeric characters
    - Each literal character matches with a corresponding character in a search string
    - Matching is case sensitive
  - Special characters
    - A reserved character that expands to a search pattern

Special Characters

- .
  - Match any character
  - Period is denoted by "\."
  - "I am the ...\." matches "I am the man."
  - "I am the …\." does not match "I am the woman."
- [chars]
  - Match any character from a given set
  - "M[ou]ammar Gadhafi" matches both Moammar Gadhafi and Muammar Gadhafi

Special Characters

- char-char
  - Matches any characters in the given range
  - A-Z, a-z, 0-9, h-x and so on
- [^chars]
  - Matches any character not in a given set
- ^
  - Matches the beginning of a line
- $
  - Matches the end of a line
- \w
  - Matches any “word” character
  - Same as [A-Za-z0-9]

Special Characters

- \d
  - Matches any digit
  - Same as [0-9]
- |
  - Matches either the element to its left or the one to its right
  - To be|Not to be
- (expr)
  - Limits scope, group elements, allow matches to be captured
  - Programm(a|e)r

Special Characters

- ?
  - Allows zero or one match of the preceding element
  - I am the egg ?man\.
- *
  - Allows zero, one, or many matches of the preceding element
  - I am the egg *man\.
- +
  - Allows one or more matches of the preceding element

Special Characters

- {*n*}
  - Matches exactly *n* instances of the preceding element
  - (co){2}mong
- {*min*,}
  - Matches at least *min* instances
- {*min*,*max*}
  - Matches any number of instances from *min* to *max*

Regular Expression Quiz

- Regex that matches a date in the format of "YYYY-MM-DD"
- Regex that matches an IPv4 address
- Regex that matches an email address

Regular Expression in Action

- find . -regex '.*/\w*-2014-.*\.jpg'