

Shell Scripting

Outline

- What is shell?
- Basic
- Syntax
 - Lists
 - Functions
 - Command Execution
 - Here Documents
 - Debug
- Regular Expression
- Find

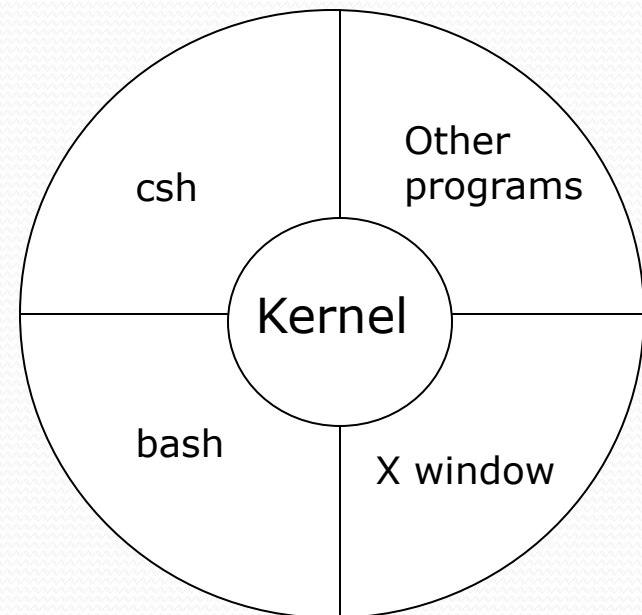
Why Shell?

- The commercial UNIX used Korn Shell
- For Linux, the Bash is the default
- Why Shell?
 - For routing jobs, such as system administration, without writing programs
 - However, the shell script is not efficient, therefore, can be used for prototyping the ideas
- For example,
 - % **ls -al | more** (better format of listing directory)
 - % **man bash | col -b | lpr** (print man page of man)

What is Shell?

- Shell is the interface between end user and the Linux system, similar to the commands in Windows
- Bash is installed as in `/bin/sh`
- Check the version

```
% /bin/sh --version
```



Pipe and Redirection

- Redirection (< or >)

- % **ls -l > lsoutput.txt** (save output to lsoutput.txt)

- % **ps >> lsoutput.txt** (append to lsoutput.txt)

- % **more < killout.txt** (use killout.txt as parameter to more)

- % **kill -l 1234 > killouterr.txt 2 >&1** (redirect to the same file)

- % **kill -l 1234 >/dev/null 2 >&1** (ignore std output)

- Pipe (|)

- Process are executed *concurrently*

- % **ps | sort | more**

- % **ps -xo comm | sort | uniq | grep -v sh | more**

- % **cat mydata.txt | sort | uniq | > mydata.txt** (generates an empty file !)

Shell as a Language

- We can write a script containing many shell commands
- Interactive Program:

- grep files with POSIX string and print it

```
% for file in *
```

```
> do
```

```
> if grep -l POSIX $file
```

```
> then
```

```
> more $file
```

```
➤ fi
```

```
➤ done
```

```
Posix
```

```
There is a file with POSIX in it
```

- '*' is wildcard

```
% more `grep -l POSIX *`
```

```
% more $(grep -l POSIX *)
```

```
% more -l POSIX * | more
```

Writing a Script

- Use text editor to generate the “first” file

```
#!/bin/sh
# first
# this file looks for the files containing POSIX
# and print it
for file in *
do
    if grep -q POSIX $file
    then
        echo $file
    fi
done
exit 0
```

% /bin/sh ← first exit code, 0 means successful

% chmod +x first

% ./first (make sure . is include in PATH parameter)

Syntax

- Variables
- Conditions
- Control
- Lists
- Functions

Variables

- Variables needed to be declared, note it is case-sensitive (e.g. foo, FOO, Foo)
- Add '\$' for storing values

```
% salutation=Hello
```

```
% echo $salutation
```

```
Hello
```

```
% salutation=7+5
```

```
% echo $salutation
```

```
7+5
```

```
% salutation="yes dear"
```

```
% echo $salutation
```

```
yes dear
```

```
% read salutation
```

```
Hola!
```

```
% echo $salutation
```

```
Hola!
```

Quoting

- Edit a "vartest.sh" file

```
#!/bin/sh
```

```
myvar="Hi there"
```

```
echo $myvar
```

```
echo "$myvar"
```

```
echo ` $myvar `
```

```
echo \ $myvar
```

```
echo Enter some text
```

```
read myvar
```

```
echo ` $myvar ` now equals $myvar
```

```
exit 0
```

Output

```
Hi there
```

```
Hi there
```

```
$myvar
```

```
$myvar
```

```
Enter some text
```

```
Hello world
```

```
$myvar now equals Hello world
```

Environment Variables

- \$HOME home directory
- \$PATH path
- \$PS1 第一層提示符號 (normally %)
- \$PS2 第二層提示符號 (normally >)
- \$\$ process id of the script
- \$# number of input parameters
- \$0 name of the script file
- \$IFS separation character (white space)
- Use 'env' to check the value

Parameter

```
% IFS = '\'
```

```
% set foo bar bam
```

```
% echo "$@"
```

```
foo bar bam
```

```
% echo "$*"
```

```
foo bar bam
```

```
% unset IFS
```

```
% echo "$*"
```

```
foo bar bam
```

← doesn't matter IFS

Parameter

Edit file 'try_var'

```
#!/bin/sh
salutation="Hello"
echo $salutation
echo "The program $0 is now running"
echo "The parameter list was $*"
echo "The second parameter was $2"
echo "The first parameter was $1"
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
```

%./try_var foo bar baz

Hello

The program ./try_var is now running

The second parameter was bar

The first parameter was foo

The parameter list was foo bar baz

The user's home directory is /home/ychuang

Please enter a new greeting

Hola

Hola

The script is now complete

Condition

need space !

- test or '['

```
if test -f fred.c
then
...
fi

If [ -f fred.c ]
then
...
fi

if [ -f fred.c ]; then
...
fi
```

expression1 -eq expression2	-d file	if directory
expression1 -ne expression2	-e file	if exist
expression1 -gt expression2	-f file	if file
expression1 -ge expression2	-g file	if set-group-id
expression1 -lt expression2	-r file	if readable
expression1 -le expression2	-s file	if size >0
!expression	-u file	if set-user-id
	-w file	if writable
	-x file	if executable


```
String1 = string2
String1 != string 2
-n string (if not empty string)
-z string (if empty string)
```

Control Structure

Syntax

```
#!/bin/sh
if condition
then
    statement
else
    statement
fi
exit 0
```

Is it morning? Please answer yes or no

yes

Good morning

Condition Structure

```
#!/bin/sh
```

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

```
read timeofday
```

```
if [ $timeofday = "yes" ]; then
```

```
    echo "Good morning"
```

```
elif [ $timeofday = "no" ]; then
```

```
    echo "Good afternoon"
```

```
else
```

```
    echo "Sorry, $timeofday not recongnized. Enter yes or no"
```

```
    exit 1
```

```
fi
```

```
exit 0
```

Condition Structure

```
#!/bin/sh
```

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

```
read timeofday
```

```
if [ "$timeofday" = "yes" ]; then
```

```
    echo "Good morning"
```

```
elif [ $timeofday = "no" ]; then
```

```
    echo "Good afternoon"
```

```
else
```

```
    echo "Sorry, $timeofday not recongnized. Enter yes or no"
```

```
    exit 1
```

```
fi
```

```
exit 0
```

If input "enter" still returns Good morning

Loop Structure

Syntax

```
for variable  
do  
    statement  
done
```

```
#!/bin/sh
```

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

```
bar  
fud  
43
```

How to output as bar fud 43?

Try change for foo in "bar fud 43"

This is to have space in variable

Loop Structure

- Use wildcard '*'

```
#!/bin/sh
```

```
for file in $(ls f*.sh); do  
    lpr $file  
done  
exit 0
```

Print all f*.sh files

Loop Structure

Syntax

```
while condition
do
    statement
done
```

```
#!/bin/sh
for foo in 1 2 3 4 5 6 7 8 9 10
do
    echo "here we go again"
done
exit 0
```

Syntax

```
until condition
do
    statement
done
```

```
#!/bin/sh
foo = 1
while [ "$foo" -le 10 ]
do
    echo "here we go again"
    foo = $foo(($foo+1))
done
exit 0
```

Note: condition is
Reverse to while
How to re-write
previous sample?

Case Statement

Syntax

```
case variable in\  
    pattern [ | pattern ] ...) statement;;  
    pattern [ | pattern ] ...) statement;;  
...  
esac
```

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

Case Statement

- A much “cleaner” version

```
#!/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 recongnized";;
```

```
esac
```

```
exit 0
```

But this has a problem, if we enter 'never' which obeys n* case and prints "Good Afternoon"

Case Statement

```
#!/bin/sh
echo "Is it morning? Please answer yes or no"
read timeofday
case "$timeofday" in
    yes | y | Yes | YES )
        echo "Good Morning"
        echo "Up bright and early this morning"
        ;;
    [nN]*)
        echo "Good Afternoon";;
    *)
        echo "Sorry, answer not recongnized"
        echo "Please answer yes of no"
        exit 1
        ;;
esac
exit 0
```

List

- AND (&&)

statement1 && statement2 && statement3 ...

```
#!/bin/sh
```

```
touch file_one
```

```
rm -f file_two
```

Check if file exist if not then create one



Remove a file



```
if [ -f file_one ] && echo "Hello" && [-f file_two] && echo " there"
then
```

```
    echo "in if"
```

```
else
```

```
    echo "in else"
```

```
fi
```

```
exit 0
```

Output

Hello

in else

List

- OR (||)

statement1 || statement2 || statement3 ...

```
#!/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
```

Output

Hello

in else

Statement Block

- Use multiple statements in the same place

```
get_confirm && {  
    grep -v "$cdcatnum" $stracks_file > $temp_file  
    cat $temp_file > $stracks_file  
    echo  
    add_record_tracks  
}
```

Function

- You can define functions for “structured” scripts

```
function_name() {  
    statements  
}  
#!/bin/sh  
foo() {  
    echo "Function foo is executing"  
}  
echo "script starting"  
foo  
echo "script ended"  
exit 0
```

Output

```
script starting  
Function foo is executing  
Script ended
```

You need to define a function before using it

The parameters `$*`, `$@`, `$#`, `$1`, `$2` are replaced by local value

if function is called and return to previous after function is finished²⁷

Function

define local
variable



```
#!/bin/sh
sample_text="global variable"
foo() {
    local sample_text="local variable"
    echo "Function foo is executing"
    echo $sample_text
}
echo "script starting"
echo $sample_text
foo

echo "script ended"
echo $sample_text

exit 0
```

Output?

**Check the
scope of
the
variables**

Function

- Use return to pass a result

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

```
echo "Original parameters are $*"
if yes_or_no "$1"
then
    echo "Hi $1, nice name"
else
    echo "Never mind"
fi
exit 0
```

Output

./my_name John Chuang

Original parameters are John Chuang

Is your name John?

Enter yes or no: **yes**

Hi John, nice name.