**Essential Shell Programming - 3**

Date: 09/04/2020

Topics to be covered:

* **set and shift**
* **set --**
* **here document**
* **trap**
* **debugging shell scripts**

**set and shift:**

set command converts its arguments to positional parameters

$set 100 200 300 400

$echo "The $# arguments are $\*"

The 4 arguments are 100 200 300 400

$echo "\$1 is $1, \$2 is $2, \$3 is $3 and \$4 is $4"

$1 is 100, $2 is 200, $3 is 300 and $4 is 400

**set can be used with command substitution as follows:**

$set `date`

$echo $\*

Thu Apr 9 11:09:09 IST 2020

$echo "The date today is $2 $3, $6"

The date today is Apr 9, 2020.

**shift: Shifting arguments left**

shift transfers the contents of positional parameters to its lower numbered one.

$set `date`

$echo $\*

Thu Apr 9 11:09:09 IST 2020

$echo $1 $2 $3

$shift

$echo $1 $2 $3

Apr 9 11:09:09

$shift 2

$echo $1 $2 $3

11:09:09 IST 2020

**Exercise: Execute the script emp7.sh and observe how shift works.**

**set --:**

We know that set is used with command substitution as observed in the previous section. set was used with date command substitution.

set when used with command substitution gives problem in two situations:

**Situation 1:**

**When the output of the command begins with a -.**

For Eg.

$ set `ls -l t1.c`

-rw-r--r--: bad options

**Situation 2:**

**When the output evaluates to null string**

For Eg.

$set `grep xxx /etc/passwd`

xxx is a non-existing user. Thus grep evaluates to null.. This makes set to run with no arguments. The output will be display of variables on the terminal(default behaviour of set).

**Solution: Use -- after set.**

**The above two cases can be solved as follows:**

**$ set -- `ls -l t1.c`**

**$set --`grep xxx /etc/passwd`**

**The two hyphens direct set to understand that the arguments following -- are not to be treated as options and also suppress the default behaviour if the arguments evaluate to a null string.**

**here document:**

**here document is used in two scenarios:**

**Scenario 1:**

**If the data required by the script is fixed and limited, data can be placed in the script itself without using any additional file.**

grep "pattern" << mark

----any content

----any content

mark

Note : mark is user defined. It can be replaced by any other word.

**Exercise : Execute valcode.sh available in text book under Sample Validation and Data entry scripts. Observe how here document is used to enter department details.**

**Scenario 2:**

**To make an interactive program, non-interactive.**

Recollect the program emp1.sh that we executed to understand read statement. Whenever read statement executed, we were able to enter data through keyboard. We had entered pattern and filename to search the pattern i the file uing grep.

Instead of waiting for each read statement to be executed, we can use here document in the following way:

$ ./emp1.sh << mark

Anil

emp.lst

mark

Thus we are able to run an interactive program non-interactively.

**Exercise : Create emp.lst file with atleast 5 records containing SSN, Employee Name, Designation, Department, DOB, Salary separated by '|' symbol.**

**Use the same script emp1.sh created for read statement. Run the script as follows using here document.**

**$ ./emp1.sh << mark**

**any employee name**

**emp.lst**

**mark**

**Observe that even though read statement is there, the script doesn't wait for user to enter the data but instead takes from the here document following the script.**

**trap:**

Shell scripts terminate whenever the interrupt key (^c) is pressed.

**Exercise : Execute any menu driven program. When you press ^c (interrupt signal) the script gets terminated.**

trap lets the programmer to do what he wishes whenever such signals are received instead of terminating the script.

**Syntax:**

trap 'command list' signal\_list

For eg.

trap ' rm \*.tmp; echo "Program interrupted"; exit' HUP INT TERM

HUP is hangup signal (Signal no. is 1 )

INT is interrupt signal (Signal no. is 2)

TERM is terminate signal (Signal no. is 15)

HUP INT TERM can be replaced by 1 2 15

**The above signals can be sent to the script using kill statement.**

**Exercise :**

**Place the trap statement at the beginning of any of your earlier scripts. Press ^c and observe that the statements eneterned in quotes after trap will be executed.**

**Debugging shell scripts with set -x:**

set is also used as a debugging tool.

set -x starts debugging. it can be turned off by set +x.

Therefore, place set -x at the beginning of the script and set +x at the end of the script.

**Exercise : In any script place set -x at the beginning of the script and observe how a shell script is debugged.**