Shell variables

Tutor: Lưu Thanh Trà

Email: lttra@hoasen.edu.vn

Outline

- Shell variable
- Embedding documentation
- Surrounding text
- Exporting variables
- Default values
- Arrays

$(( )) , let

- Example
  - COUNT=$((COUNT + 5 + MAX * 2))
  - let COUNT+=5+MAX*2
- No space after the “=“
  - COUNT = $((COUNT + 5))
  - => running the program COUNT with 2 parameters “=“ and value of (COUNT+2)

Operators

Operator	Operation with assignment	Use	Meaning
=	Simple assignment	a=b	a=b
*=	Multiplication	a*=b	a=(a*b)
/=	Division	a/=b	a=(a/b)
%=	Remainder	a%=b	a=(a%b)
+=	Addition	a+=b	a=(a+b)
-=	Subtraction	a-=b	a=(a-b)
<<=	Bit-shift left	a<<=b	a=(a<<b)
>>=	Bit-shift right	a>>=b	a=(a>>b)
&=	Bitwise “and”	a&=b	a=(a&b)
^=	Bitwise “exclusive or”	a^=b	a=(a^b)
|=	Bitwise “or”	a|=b	a=(a|b)

- COUNT=$((COUNT + $2 + OFFSET))
- COUNT=$((COUNT + 2 + OFFSET))
- let COUNT=COUNT+5
- let COUNT+=5
- echo $((X+=5 , Y*=3))
- let X+=5 Y*=3
- let Y=(X+2)*10 # no quote
- Y=$(((X + 2) * 10))

Conditions

```
1. if [ $# -lt 3 ]
2. then
3. printf "%b" "Error. Not enough arguments.\n"
4. printf "%b" "usage: myscript file1 op file2\n"
5. exit 1
6. fi
```

- or

```
1. if (( $# < 3 ))
2. then
3. printf "%b" "Error. Not enough arguments.\n"
4. printf "%b" "usage: myscript file1 op file2\n"
5. exit 1
6. fi
```

- if (($# < 3))
  - then
  - printf "%b" "Error. Not enough arguments.\n"
  - printf "%b" "usage: myscript file1 op file2\n"
  - exit 1
- elif (($# > 3))
  - then
  - printf "%b" "Error. Too many arguments.\n"
  - printf "%b" "usage: myscript file1 op file2\n"
  - exit 2
- else
  - printf "%b" "Argument count correct. Proceeding...\n"
- fi

- [$result = 1] \
  && { echo "Result is 1; excellent." ; exit 0; } \
  || { echo "Uh-oh, ummm, RUN AWAY! " ; exit 120; }

General format of “if”

- if list; then list; [elif list; then list;] ... [else list;] fi

Testing for file permissions

- #!/usr/bin/env bash
- # cookbook filename: checkfile
- #
- DIRPLACE=/tmp
- INFILE=/home/yucca/amazing.data
- OUTFILE=/home/yucca/more.results
- if [-d "$DIRPLACE"]
- then
  - cd $DIRPLACE
  - if [-e "$INFILE"]
  - then
    - if [-w "$OUTFILE"]
    - then
      - doscience < "$INFILE" >> "$OUTFILE"
    - else
      - echo "can not write to $OUTFILE"
    - fi
- else
  - echo "can not read from $INFILE"
- fi
- else
  - echo "can not cd into $DIRPLACE"
- fi

Test file with more than one thing

- if [-r $FILE **-a** -w $FILE]
- if [-r "$FN" -a **\(** -f "$FN" -o -p "$FN" **\)**]

String testing

- VAR="$1“
- if ["$VAR"]
- then
  - echo has text
- else
  - echo zero length
- fi
- #
- if [-z "$VAR"]
- then
  - echo zero length
- else
  - echo has text
- fi

String – number comparison

Numeric	String	Meaning
-lt	<	Less than
-le	<=	Less than or equal to
-gt	>	Greater than
-ge	>=	Greater than or equal to
-eq	=, ==	Equal to
-ne	!=	Not equal to

Patterns

- if [["${MYFILENAME}" == *.jpg]]
- Or
  - shopt -s extglob
  - if [["$FN" == *.@(jpg|jpeg)]]
  - then

```
1. case $FN in
2. *.gif) gif2png $FN
3. ;;
4. *.png) pngOK $FN
5. ;;
6. *.jpg) jpg2gif $FN
7. ;;
8. *.tif | *.TIFF) tif2jpg $FN
9. ;;
10. *) printf "File not supported: %s" $FN
11. ;;
12. esac
```

Extended pattern matching

Grouping	Meaning
@(...)	Only one occurrence
*(...)	Zero or more occurrences
+(...)	One or more occurrences
?(...)	Zero or one occurrences
!(...)	Not these occurrences, but anything else

Loop with a count

- for ((expr1 ; expr2 ; expr3)) ; do list ; done
- example
  - $ for **((** i=0 ; i < 10 ; i++ **))** ; do echo $i ; done

Loop with real number

- for fp in $(seq 1.0 .01 1.1)
- do
  - echo $fp; other stuff too
- done