UNIX OS

Shell Scripting

Role of Shells

Generates Command Line Interface

Command Interpreter

Provides I/O Redirection

Provides Pipe Operator |

Executes commands from a file

Provides scripting constructs:

if statement

while/until statements

for statement

case/select statements

Popular Shells

/bin/sh /bin/csh

Bourne Shell C Shell

/bin/bash

Bourne(Born) Again Shell

/bin/ksh

Korn Shell

Shell Variables

Only data type is string

Assigning creates variables

$ X=hello

$ echo X world

$ echo $X world

$ X='hello world'

$ echo $X

Shell Variables

$ X=3+4

$ echo $X

$ Y=X

$ echo $Y

$ Y=$X

$ echo $Y

$ echo hello $ABCDEFGH world

Undefined variable??

Shell Variables

No Spaces allowed with =

$ X =hello ERROR

$ X= hello ERROR

No spaces between $ and variable

$ echo $ X world

What's the result?

Shell Variables

Display the list of variables

$ set

$ set | less

Undefine a variable

$ unset X

$ echo $X world

A $ dollar needed only if expansion is required

Predefined Variables

HOME

USER

UID

PWD

TERM

PATH

PS1

PS2

Variable Attributes

Integer Attribute

$ NUM=3+4

$ echo $NUM

$ typeset -i NUM

$ NUM=3+4

$ echo $NUM

$ NUM=3 + 4

What’s the result?

Variable Attributes

Integer Attribute variable list

$ typeset -i

$ declare -i

Removing Integer Attribute

$ typeset +i NUM

$ NUM=3+4

$ echo $NUM

Variable Attributes

Read Only Attribute

$ typeset -r MYEMPID=E100

$ echo $MYEMPID

$ MYEMPID=E999

$ unset MYEMPID

$ typeset +r MYEMPID

All read only variables

$ typeset -r

$ declare -r

Command Substitution

$ uname

$ hostname

$ echo hostname runs uname

$ echo `hostname` runs `uname`

$ umask

$ echo umask value is `umask`

Command Substitution

$ whoami

$ MyId=`whoami`

$ uname

$ OS=`uname`

$ uname –r

$ OS\_VERSION=`uname –r`

$ ls –l | grep ‘^d’ | wc – l

$ ND=`ls –l | grep ‘^d’ | wc – l`

Command Substitution

Wrong usage

$ echo hi `chmod o-x f1 ` world

$ ls –l f1

$ echo hi `cp /etc/passwd f1` world

$ ls –l f1

$ echo hi `rm a` world

$ ls –l f1

Quoting

Three expansions of shell

$ touch tab tall fall

$ echo $HOME `uname` ta\*

$ echo $HOME `uname` [ft]all

Quoting

$ echo $HOME `uname` ta\*

$ echo ‘ $HOME `uname` ta\* ‘

$ echo “ $HOME `uname` ta\* ”

$ echo $HOME `uname` [ft]all

$ echo ‘ $HOME `uname` [ft]all `

$ echo “ $HOME `uname` [ft]all “

Quoting With \

Completely Quotes the next char

$ echo f\*

$ echo f\\*

$ echo \*

$ echo \\*

Quoting With \

Completely Quotes the next char

$ echo hello > world

$ echo hello \> world

$ touch newfile &

$ touch newfile \&

Quoting

Single quotes – Complete quote

No characters are Meta ones

Double quotes – Partial Quotes

Meta Characters in “ qoutes

$ Variables expansion

` Command Substitution

\ Escaping $ and \

Back slash

Completely quote the next char

expr command

$ expr 3 + 4

$ expr 3 \* 4

$ expr 3 \\* 4

$ X=30 Y=40

$ expr $X \\* $Y

expr command

$ Z=`expr $X \\* $Y`

$ Z=`expr $X \\* \( $Y + 4 \)`

X \* ( Y + 4 )

$ echo “ X + Y = `expr $X + $Y` ”

Shell Process

Login Shell Process

Created by Kernel on user login

Sub Shell Process ( INTERACTIVE )

Created by bash command

Displays a prompt

Reads commands from the user & executes

Parent waits till child terminates

Parent resumes after child terminates

Shell Process

Sub Shell Process (NON-INTERACTIVE)

Created by running a shell script

Does NOT display a prompt

Reads commands from script file & executes them

Parent waits until child terminates

Parent resumes after child terminates

Command Exit Status

Commands/programs return an integer to OS on Termination, Called exit status.

Last command's exist status in $?

Shell requires

Zero on success

Non Zero on failure

Check the Exit Status

$ touch f1

$ ls -l f1

$ ls -l NoFile

$ NoCommand

Check the Exit Status

$ cp f1 f2

$ cp NoFile f2

$ rm f1

$ rm NoFile

List Commands Exit Status

$ grep root /etc/passwd

$ grep NoId /etc/passwd

$ echo hello; uname; ls NoFile

$ echo hello; uname; ls f1

$ echo hello uname; ls f1

If Compound Command

if Control\_Command

then

Commands to be executed

fi

If ……

$ if cp f1 f2

then

echo “Copy Successfull”

fi

$ if cp NoFile f2

then

echo “Copy Successfull”

fi

If ……

if grep NoId /etc/passwd

then

echo “/etc/passwd has NoId”

fi

If ……

if grep root /etc/passwd

then

echo “/etc/passwd has root”

fi

If ……

if grep root /etc/passwd >/dev/null

then

echo “/etc/passwd has root”

fi

If else

if Control\_Command

then

Commands on Success

else

Commands on Failure

fi

If else

$if grep NoId /etc/passwd

then

echo “/etc/passwd has NoId”

else

echo “/etc/passwd doesn't have”

fi

Comparing Files

$ echo hello > f1

$ cp f1 f2

$ cmp f1 f2

$ echo $?

Comparing Files

$ echo heap > f3

$ cmp f1 f3

f1 f3 differ: byte 3, line 1

$ echo $?

If else

$ if cmp f1 f2

then

echo “f1 and f2 don't differ”

else

echo “f1 and f2 differ”

fi

If else

$ if cmp f1 f3

then

echo “f1 and f3 don't differ”

else

echo “f1 and f3 differ”

fi

If else

$ if cmp f1 f3 > /dev/null

then

echo “f1 and f3 don't differ”

else

echo “f1 and f3 differ”

fi

Elif

if Control\_Command1

then

Commands1

elif Control\_Command2

then

Commands2

elif Control\_Command3

then

Commands3

else

Commands4

fi

Elif

read –p “Languages list:” LANGS

if echo “$LANGS” | grep –i ‘ c++ ’

then

echo ”C++ included”

elif echo “$LANGS”| grep –i ‘ c ‘

then

echo ”C included”

else

echo “C and C++ missing”

fi

What’s wrong?

$ X=20

$ if $X > 10

then

echo X is greater than 10

fi

bash: 20: command not found

$ ls -l 10

[ and test commands

$ which [

$ which test

Exits status based on arguments

To be used for

String tests

Numeric tests

File tests

Numeric tests

$ X=90

$ [ “$X” -gt 200 ]

$ [ “$X” -lt 200 ]

$ [ “$X” -ge 90 ]

$ [ “$X” -le 90 ]

$ [ “$X” -ne 200 ]

$ [ “$X” -eq 200 ]

Numeric tests

$ X=90

$ [ “$X” \> 200 ]

Whats the result?

String comparison for numbers?

Os is Linux?

OS=`uname `

if [ “$OS” = Linux ]

then

echo “OS is Linux”

elif [ “$OS” = HPUX ]

then

echo “OS is HPUX”

else

echo “OS is neither Linux nor HPUX”

fi

Copy wrapper

# Copies source file to a directory

read -p “Source file name:” SNAME

if [ -z "$SNAME” ]

then

echo “Source file name error"

exit 1

fi

if [ ! -f “$SNAME” ]

then

echo “Source not regular file”

exit 2

fi

Copy wrapper

read -p “Directory name:” DNAME

if [ -z "$DNAME” ]

then

echo “Directory name error"

exit 3

fi

if [ ! -d “$DNAME” ]

then

echo “Directory doesn’t exist”

exit 4

fi

Copy wrapper

if [ ! -r $SNAME ]

then

echo “Source file unreadable!!”

exit 5

fi

if [ ! -w $DNAME ]

then

echo “Directory Unwritable!”

exit 6

fi

Copy wrapper

if cp “$SNAME” “$DNAME”

then

echo “Copied successfully”

else

echo “error copying. Possibly”

echo “no space on device!!”

fi

Grades..

# Decide grade for marks given

# argument.

# A 75 – 100, B 50 – 74

# Faliure 0 - 49

if [ $# -ne 1 ]

then

echo ”Usage: $0 marks”; exit 1;

elif [ “$1” –gt 100 –o “$1” –lt 0 ]

then

echo “Marks not valid”; exit 2;

fi

Grades..

if [ $1 -ge 75 ]

then

echo “A Grade”

elif [ $1 –lt 75 –a $1 -ge 50 ]

then

echo “B Grade”

else

echo “Failure”

fi

Compare file contents

# Compare contents of two files

# given as arguments

if [ $# -ne 2 ]

then

echo ”Usage: $0 <file1> <file2>;”

exit 1;

fi

if [ “$1” = “$2” ]

then

echo “Both arguments same!!!”

exit 2;

fi

Compare file contents

if [ -f “$1” –a –f “$2” ]

then

if diff “$1” “$2” >/dev/null

then

echo “Same contents in $1 and $2”

else

echo “$1 and $2 contents differ”

fi

else

echo “Both arguments should be

regular files”

fi

Copy Script

Develop a script that copies the file from first argument to second argument.

Wrong usage of [

if [ cp $1 $2 ]

then

echo “Copy successful”

else

echo “Copy failed”

fi

if [ `cp $1 $2` ]

then

echo “Copy successful”

else

echo “Copy failed”

fi

Copy script

if cp $1 $2

then

echo “Copy successful”

else

echo “Copy failed”

fi

While and until loops

while Control\_Command

do

Commands

done

While loops

$ while read X

do

echo “--$X--”

done

Working with Records

while read -p "EMPID=" EMPID

do

read -p "EMPNAME=" EMPNAME

echo $EMPID:$EMPNAME

echo

done

Working with Records

while read -p "EMPID=" EMPID

do

read -p "EMPNAME=" EMPNAME

echo $EMPID:$EMPNAME >> IdNames

echo

done

emp file

cat > emp

M285:Raghu:mrktng:sales:12000

I024:Vasu:software:testing:9000

I392:John:software:design:15000

M352:Shenoy:mrktng:adv:11000

I004:Bhanu S:software:design:8000

M046:Bharath:mrktng:adv:9000

I020:Sam:software:testing:12000

M004:Hari:mrktng:sales:9000

Ctrl-d

Print Numbers..

# Reads a number and prints from

# zero to that number

echo -n "Input a non -ve number"

read NUM

if [ -z "$NUM" –o “$NUM” –le 0 ]

then

echo "$0:$NUM:invalid number "

exit 1;

fi

Print N numbers

# IN for Iteration Number

IN=1

while [ "$IN" -le "$NUM" ]

do

echo “Iteration : $IN”

IN=`expr $IN + 1`

done

Sum salaries

# Print the sum of salaries

TOTALLINES=`cat emp| wc -l`

SUM=0;

LINE=1

while [ "$LINE" -le "$TOTALLINES" ]

do

SAL=`head -"$LINE“ emp | tail -1

| cut -d: -f5`

SUM=`expr "$SUM" + "$SAL"`

LINE=`expr $LINE + 1`

done

echo “SUM=$SUM”

&& and ||

$ touch f1

$ ls -l f1 && echo Hello world

$ ls -l NoFile && echo Hello world

$ [ -f f1 ] && { cp f1 f1\_copy

chmod o-rwx f1\_copy

echo copied

}

&& and ||

$ cat f1 || echo Hello world

$ cat NoFile|| echo Hello world

$ [ -d “$1” ] || {

echo “$1 is not”

echo “directory”

exit 4

}

Processing arguments

[ $# -eq 0 ] && {

echo “No args”; exit 1; }

while [ $# -ne 0 ]; do

echo ‘$1 is’ = $1

shift

done

Non zero sized files

# From the argument list

[ $# -eq 0 ] && {

echo “No args”; exit 1; }

COUNT=0

while [ $# -ne 0 ]; do

if [ -s "$1" ];

then echo "$1"

COUNT=`expr $COUNT + 1`

fi

shift

done

echo “Num of files size>0 = $COUNT"

For Loop

for loop\_var in list\_of\_words

do

Commands

done

$ for WORD in hello world shell

do

echo “WORD = $WORD”

done

For Loop

$ for FNAME in \*

do

echo “FNAME = $FNAME”

done

$ for ARG in $\*

do

echo “ARG = $ARG”

done

For Loop

$ for ARG in $@

do

echo “ARG = $ARG”

done

$ for ARG in “$@”

do

echo “ARG = $ARG”

done

For Loop

$ for USER in `who|cut –d ‘ ’ –f1`

do

echo “USER = $USER”

done

For Loop

$ SUM=0

$ for SAL in `cut –d : –f5 emp`

do

SUM=`expr $SUM + SAL`

done

$ echo “Sum = $SUM”

Interactive backup

for ARG in "$@"

do

echo -n "$ARG“

[ -f "$ARG" ] && {

echo -n " a file “

[ -r "$ARG" ] && echo –n "Readable“

}

File or Directory?

[ -d "$ARG" ] && {

echo -n " a directory “

[ -w "$ARG" ] && echo "Writable“

}

echo

done

Case statement

case CASE\_VAR in

Pattern1)

Commands1

;;

Pattern2)

Commands2

;;

Pattern3)

Commands3

;;

esac

Case statement

$ read -p 'copy/list?' COMMAND

$ case $COMMAND in

copy) read -p "Source:" SF

read -p "Destination:" DF

cp "$SF" "$DF"; echo

;;

list) read -p "File:" F

ls -ld $F; echo

;;

\*) echo "$COMMAND not known"

;;

esac