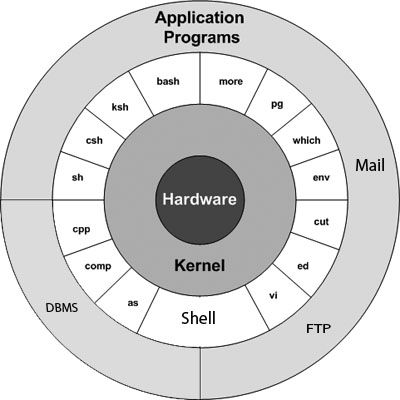
RIO\_01\_About the Unix Shell

Unix Arch



Various Types of Shell

sh csh ksh bash tcsh zsh rc es

Job control N Y Y Y Y Y N N

Aliases N Y Y Y Y Y N N

Shell functions Y(1) N Y Y N Y Y Y

"Sensible" Input/Output redirection Y N Y Y N Y Y Y

Directory stack N Y Y Y Y Y F F

Command history N Y Y Y Y Y L L

Command line editing N N Y Y Y Y L L

Vi Command line editing N N Y Y Y(3) Y L L

Emacs Command line editing N N Y Y Y Y L L

Rebindable Command line editing N N N Y Y Y L L

User name look up N Y Y Y Y Y L L

Login/Logout watching N N N N Y Y F F

Filename completion N Y(1) Y Y Y Y L L

Username completion N Y(2) Y Y Y Y L L

Hostname completion N Y(2) Y Y Y Y L L

History completion N N N Y Y Y L L

Fully programmable Completion N N N N Y Y N N

Mh Mailbox completion N N N N(4) N(6) N(6) N N

Co Processes N N Y N N Y N N

Builtin artithmetic evaluation N Y Y Y Y Y N N

Can follow symbolic links invisibly N N Y Y Y Y N N

Periodic command execution N N N N Y Y N N

Custom Prompt (easily) N N Y Y Y Y Y Y

Sun Keyboard Hack N N N N N Y N N

Spelling Correction N N N N Y Y N N

Process Substitution N N N Y(2) N Y Y Y

Underlying Syntax sh csh sh sh csh sh rc rc

Freely Available N N N(5) Y Y Y Y Y

Checks Mailbox N Y Y Y Y Y F F

Tty Sanity Checking N N N N Y Y N N

Can cope with large argument lists Y N Y Y Y Y Y Y

Has non-interactive startup file N Y Y(7) Y(7) Y Y N N

Has non-login startup file N Y Y(7) Y Y Y N N

Can avoid user startup files N Y N Y N Y Y Y

Can specify startup file N N Y Y N N N N

Low level command redefinition N N N N N N N Y

Has anonymous functions N N N N N N Y Y

List Variables N Y Y N Y Y Y Y

Full signal trap handling Y N Y Y N Y Y Y

File no clobber ability N Y Y Y Y Y N F

Local variables N N Y Y N Y Y Y

Lexically scoped variables N N N N N N N Y

Exceptions N N N N N N N Y

Key to the table above.

Y Feature can be done using this shell.

N Feature is not present in the shell.

F Feature can only be done by using the shells function

Eg1.sh – First shell script

echo 'hello world'

RIO\_02\_MetaCharacters

Eg2.sh – Strong and weak quotes

echo '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

echo ' Test 1 '

echo '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

echo $HOME file? `pwd`

echo '$HOME file? `pwd`'

echo "$HOME file? `pwd`

Eg3.sh– Strong and weak quotes

Var1="$HOME"

echo $Var1

Var2='$HOME'

echo $Var2

Var3=`ls -ltr`

echo $Var3

echo "$Var3"

Eg4.sh – Quotes within quotes

var="String's"; echo $var

var='String"s'; echo $var

var='String'\''s'; echo $var

gg.sh – check syntax error

echo 'he;;o'

Var = 'gooD

echo $Var

eg5.sh – Wild card ,Special characters

echo `ls \*.sh`

echo `ls file?`

echo `ls file??`

Var='Learn'

echo ${Var}ing

echo $Varing

echo 'ravi';echo 'raj'

echo 'ravi'

echo 'raj'

echo $HOME

echo ~

ls -l > file3

# This is a comment

Eg6.sh- Escape sequence

var="ST'RIGH\"s"

echo $var

echo -e 'ravi\tkumar'

echo -e 'ravi\vkumar'

echo -e 'ravi\bkumar'

echo -e 'ravi\nkumar'

eg7.sh- Here document

ver=1.2.3

cat <<END

$0 (version $ver)

Usage:

$0 <start|stop|status>

\*.\*

END

RIO\_03\_Variables

Environment Variables

* **PATH** - Display lists directories the shell searches, for the commands.
* **HOME** - User's home directory to store files.
* **TERM** - Set terminal emulator being used by UNIX.
* **PS1** - Display shell prompt in the Bourne shell and variants.
* **MAIL** - Path to user's mailbox.
* **TEMP** - Path to where processes can store temporary files.
* **JAVA\_HOME** - Sun (now Oracle) JDK path.
* **ORACLE\_HOME** - Oracle database installation path.
* **TZ** - Timezone settings
* **PWD** - Path to the current directory.
* **HISTFILE** - The name of the file in which command history is saved
* **HISTFILESIZE** -The maximum number of lines contained in the history file
* **HOSTNAME** -The system's host name
* **LD\_LIBRARY\_PATH** -It is a colon-separated set of directories where libraries should be searched for.
* **USER** -Current logged in user's name.
* **DISPLAY** -Network name of the X11 display to connect to, if available.
* **SHELL** -The current shell.
* **TERMCAP** - Database entry of the terminal escape codes to perform various terminal functions.
* **OSTYPE** - Type of operating system.
* **MACHTYPE** - The CPU architecture that the system is running on.
* **EDITOR** - The user's preferred text editor.
* **PAGER** - The user's preferred text pager.
* **MANPATH** - Colon separated list of directories to search for manual pages.
* ###############################################

eg1.sh - Assignment

* ###############################################
* a=111
* echo "The value of \"a\" is $a."
* let a=16+5
* echo "The value of \"a\" is now $a."
* b=16+5
* echo "The value of \"b\" is now $b."
* echo -n "Values of \"a\" in the loop are: "
* for a in 7 8 9 11
* do
* echo -n "$a "
* done
* echo -n "Enter \"a\" "
* read a
* echo "The value of \"a\" is now $a."

eg2.sh - readonly

* declare -r var1=1
* echo "var1 = $var1"
* var1+=1

eg3.sh - typeset and declare

* declare -i num
* num=3
* echo "num = $num"
* num=three
* echo "num = $num"
* n=6/3
* echo "n = $n"
* declare -i n
* n=6/3
* echo "n = $n"
* typeset -u var2=ravi
* echo $var2

eg4.sh - substituion and unset

* A=2
* B=5
* let C1=$A+$B
* echo $C1
* a=2334
* let "a += 1"
* echo "a = $a "
* echo
* b=${a/23/BB}
* echo "b = $b"
* declare -i b
* echo "b = $b"
* let "b += 1"
* echo "b = $b"
* echo
* c=BB34
* echo "c = $c"
* d=${c/BB/23}
* echo "d = $d"
* let "d += 1"
* echo "d = $d"
* echo
* var4=10
* echo $var4
* unset var4
* echo $var4

eg5.sh - envor var

* cls
* echo $PATH
* env

eg6.sh - special variables

* echo '$0=' "$0"
* echo '$$=' "$$"
* echo '$?=' "$?"
* cls &
* echo '$!=' "$!"
* var=ravi
* echo $var
* echo '${#var}=' "${#var}"

eg7.sh - arithmatic & string operations

* a=10
* b=34
* let c=$a+$b
* echo '$c=' $c
* d=$(($a+$b))
* echo '$d=' $d
* e=`expr $a + $b`
* echo '$e=' $e

eg8.sh - eval statement

* a=10
* b=20
* c=$(($a+$b))
* echo $c
* d=`expr $a + $b`
* echo $d
* e=`expr $a \\* $b`
* echo $e

eg9.sh

* var=abcdev
* echo ${var?XYZ}
* #echo ${var1?XYZ}
* echo ${var-XYZ}
* echo ${var1-XYZ}
* echo ${var+XYZ}
* echo ${var1+XYZ}
* echo ${var1=xYZ}
* echo ${var1-XZZ}
* fund()
* {
* echo $1
* echo $2
* let c=$1+$2
* return $c
* }
* source /root/ravi/day2/scrp1.sh
* fund 10 20
* echo $?

eg10.sh - variable sub string

* var=ravikumar
* echo ${var:1}
* echo ${var/vi/mya}

eg11.sh - existance example

* a='chetan'
* echo ${a?mone}
* echo
* echo ${b?mone}
* echo
* echo ${a-mone}
* echo
* echo ${b-mone}
* echo
* echo ${a+mone}
* echo
* echo ${b+mone}
* echo
* echo ${a=mone}
* echo
* echo ${b=mone}
* echo
* echo $b

RIO\_04\_Functions

eg.12.sh - functions

fun()

{

echo 'THis is a fun function'

}

fu1()

{

echo 'This is fu1'

fun

}

fu1

fun

eg13.sh-function with arg and retrun value

func()

{

echo $1

return 12

}

funa()

{

exit 13

}

func 10

echo $?

funa

echo 'Hai'

scrp1.sh

fund()

{

echo $1

echo $2

let c=$1+$2

return $c

}

Scrp2.sh

source /root/ravi/basic/day2/scrp1.sh

fund 10 20

echo $?

eg14.sh

bb()

{

echo 'yoou are in bbhai'

}

cc()

{

echo 'you are in cc'

}

a=$1

eval $a

bash eg14.sh bb or bash eg14.sh cc

eg15.sh

var=1

echo $var

(

echo $var

var=$(($var+1))

echo $var

)

echo $var

{

echo $var

var=$(($var+1))

echo $var

}

echo $var

RIO\_05\_Control Structures and Loops

eg1.sh - if statement

a=$1

b=$2

if [ $a == $b ]

then

echo 'a is equal to b'

fi

if [ $a -gt $b ]

then

echo 'a is greater than b'

else

echo 'b is greater than or equal to a'

fi

if [ $a == $b ]

then

echo 'a is equal tO than b'

elif [ $a -lt $b ]

then

echo 'a is less than b'

else

echo 'b is less than a'

fi

if [ $a -lt $b ] && [ $a -gt 0 ]

then

echo 'OK'

fi

if [ $a -lt $b ] || [ $a -gt 0 ]

then

echo 'NOT OK'

fi

if [ : ]

then

echo 'always true'

fi

eg2.sh - expressions

str1=""

str2=$1

str3=$2

if [ -z $str1 ]

then

echo 'str1 is zero length'

fi

if [ -n $str2 ]

then

echo 'str2 is not zero length'

fi

if [ $str3 == $str2 ]

then

echo 'str3 = str2'

fi

if [ $str3 != $str2 ]

then

echo 'str3 NOT= str2'

fi

eg3.sh test file

if [ -f $1 ]

then

echo 'it is a file'

fi

if [ -d $1 ]

then

echo 'it is a directory'

fi

if [ -r $1 ]

then

echo 'it is readable file'

fi

eg4.sh - case

grade=$1

case $grade in

'first class')

echo "70% or above";;

'second class')

echo "between 50% to 70%";;

'third clss')

echo "between 30% to 50%";;

\*)

echo "failed";;

esac

eg5.sh

for a in \*.sh

do

cat $a

done

eg6.sh - while/until loop

a=10

while [ $a -gt 0 ]

do

echo '$a=' $a

a=$(($a - 1))

done

echo

until [ $a -gt 10 ]

do

echo '$a=' $a

a=$(($a + 1))

done

eg6.sh - Break / Continue

a=10

while [ $a -gt 0 ]

do

echo '$a=' $a

if [ $a -lt 5 ]

then

break

fi

a=$(($a - 1))

done

echo

until [ $a -gt 10 ]

do

a=$(($a + 1))

if [ $a -lt 7 ]

then

continue

fi

echo '$a=' $a

done

eg8.sh - select option

medals="gold silver bronze"

select medal in $medals

do

echo $medal

done

RIO\_06\_Parameter Passing

eg9.sh- Passsing Arguments

echo 'You have passed' $# 'arguments'

echo 'Calling module:' $0

for a in $\*

do

echo $a

done

eg10.sh - Using Shift

a=$#

while [ $a -gt 0 ]

do

echo $1

a=$(($a - 1))

shift

done

eg11.sh - Accept user input

read -p "Enter your age and gendar :" age gendar

echo 'Age=' $age

echo 'Gendar=' $gendar

RIO\_07\_Redirection and Pipes

eg1.sh - Redirection

i=1

while true

do

read line

if [ $? -ne 0 ]

then

echo '----------------'

exit 0

fi

echo line $i : ${#line} chars

i=$(($i+1))

done

eg2.sh - STDOUT/STD ERROR

i=$1

if [ $i -gt 100 ]

then

echo aa

else

echo `aa`

fi

cat $1 |grep $2 |sort

RIO\_08\_Arrays

eg4.sh - Arrays

arr[10]='ravi'

# Array can be accessed using {}

echo -n "arr[10]="

echo ${arr[10]}

# COntent of uninitialize array is balnk

echo '${arr[11]}=' ${arr[11]}

# Another way to initialize

arr=('first element' 'second element' 'third element')

echo ${arr[@]:0}

eg5.sh#####

array=(zero one two three four five)

echo ${array[0]}

echo ${array:0}

echo ${array:1}

echo "--------------"

echo ${#array[0]}

echo ${#array}

echo ${#array[1]}

echo ${#array[\*]}

echo ${#array[@]}

echo "------------------"

eg6.sh

filename=$1

declare -a array1

array1=(`cat "$filename"`)

echo ${array1[@]}

echo ${array1[@]:3}

echo ${#array1[\*]}

echo ${array1[5]}

declare -a mat

r=4

c=5

for ((i=1;i<=r;i++)) do

for((j=1;j<=c;j++)) do

mat[$i,$j]=$RANDOM

done

done

RIO\_09\_Traps and Signals

eg1.sh - SIGINT

trap "echo I will ignore SIGINT" SIGINT

a=60

i=1

while [ $i -le $a ]

do

echo "i =$i"

i=$(($i+1))

sleep 1

done

eg2.sh - SIGTERM

trap "echo I will ignore SIGTERM " SIGTERM

trap "echo I will try to ignore " SIGKILL

i=1

a=60

while [ : ]

do

i=$(($i+1))

echo 'Hai' $i > file1

#if [ $i -gt 5000 ]

#then

#exit

#fi

Done

RIO\_10\_Good Practices

Case Study – Sample 1

CASE STUDY - EXAMPLE

source ~/ravi/day5/disp.sh

source ~/ravi/day5/search.sh

opts="Display Search Exit"

select opt in $opts

do

case $opt in

'Display')

display ;;

'Search')

search ;;

'Exit')

exit;;

'')

echo " invalid option";;

esac

done

Function 1 - Display

display()

{

echo 'Enter the source file name'

read fname

if [ -z $fname ]

then

echo ' in-valid filename'

return

else

cat $fname

return

fi

}

Function 2 - Search

search()

{

echo 'Enter the source file name'

read fname

echo 'Enter the string to search'

read str

if [[ -z $fname ]] || [[ -z $str ]]

then

echo ' in-valid filename or string'

return

else

cat $fname | grep $str

return

fi

}