סיכום קורס לינוקס 2018

http://www.sweb.co.il/forums/%D7%9E%D7%93%D7%A8%D7%99%D7%9A-%D7%A4%D7%A7%D7%95%D7%93%D7%95%D7%AA-%D7%91%D7%A1%D7%99%D7%A1%D7%99%D7%95%D7%AA-%D7%91%D7%9C%D7%99%D7%A0%D7%95%D7%A7%D7%A1-linux/

http://www.computerhope.com/unix

REGX:
http://www.cyberciti.biz/faq/grep-regular-expressions/

grep:
http://www.cyberciti.biz/faq/howto-use-grep-command-in-linux-unix/

putty:
server ip: 192.168.22.22
user: b2_studentXX - XX is the 2 digits on the computer
password: afeka

password: nimda
terminal

מערכת ההפעלה UNIX מבוססת קבצים (כולל פקודות). מה שלא קובץ הוא process. ההבדל בין unix ל linux
הוא שב linux ניתן לשייך משתמש לכמה קבוצות
הספרייה הראשית נקראת ROOT (גם ה user name של האדמיניסטרטור).
ניתן ליצור משתמשים ללא סיסמה. לא ניתן לשחזר סיסמאות שנשכחו.

הספריות שמתחת :
/bin - פקודות בינאריות.
/sbin – פקודות יותר חדשות של הSHELL. הרחבה של ה- bin
/etc – שני סוגי קבצים:
1. הגדרות משתמש. בתוכה יש קובץ שנקרא 'passwd' המכיל רשימת משתמשים (user name) רשימת הסיסמאות נמצאת בקובץ הנקרא 'shadow' וקובץ שנקרא 'groups'
2. קבצי קונפיגורציה. קבצים שמתחילים/מסתיימים ב- CONF
/usr – נמצאות התקנות third – party. לא מתקינים בדר"כ ב main directory. תחת תיקייה זו גם יש תיקיית bin שקבצי ההרצה של התוכנות שהתקנו נמצאים שם.
/home – ספרייה של home directory. כל היוזרים נמצאים בדר"כ תחת תיקייה זו. יש אפשרות ביצירת יוזר ליצור ספרייה בשם המשתמש או שלכל המשתמשים תהיה תיקיית home אחת.
/dev – הגדרות של מכשירים. ז"א קובץ של מכשיר המכיל קונפיגורציה ודרייברים למכשיר
/var – מכילה קבצי temp.
/proc – קבצי system

Text editor

- Vi filname - open filname in editor
- Clear – clear the editor
- Whatis (command)- נותנת הסבר על הפקודה
- History- הצגת היסטוריה
- ls –a (show all)
  כל הקבצים שמתחילים ב . מוסתרים
- Kill- להרוג תהליך   kill -9\15 (9 perent , 15 the process)
- whoami = my user מי המשתמש
  w לראות את כל המשתמשים במערכת
- pwd = path work dir
- לראות מה הcurrent directory. איפה כרגע אנחנו נמצאים, הדפסת שם התיקייה הנוכחית
- Man (command) -
  פקודת help על פקודה מסוימת. קיימת רק ב linux. ניתן לחפש עפ"י מילת מפתח באופציה -k
  כדי לנווט יש 3 אופציות :
  אנטר - שורה הבאה
  רווח - הדף הבא
  q - יציאה
- i/I: insert
  I - beginig of row/end of file
  i - place of the cursor
  esc - back to command mode
- o/O: open line
  O - Open a new line before current line
  o - Open a new line after current line/under the cursor
- G: Move to the last line of the file
- yy: Yank the current line (Copy)
  3yy - Places 3 lines in the buffer-copies
  Xyy - Places X lines in the buffer-copies
  yG: copy all lines from cursor to end of the file
- p/P: Past
  P - Put before the position or before the line
  p - Put after the position or after the line
- dd: Delete current line
  3dd - Deletes 3 lines beginning with the current line
  Xdd - Deletes X lines beginning with the current line
  dG - delete all lines from cursor to end of the file
- x/X: Delete character
  x - Delete character to the right of cursor
  X - Delete character to the left of cursor/backspace
- cw: Change word
  c3w - change 3 words
  cXw - change X words
- r/R: Replace
  R - replace until press esc
  r - replace current place of the cursor (use r and than the value you want)
- b= go back one word
- :q! = exit without save
- :w = save
- :w FileName = save as FileName
- :q = exit

- :wq = save and exit
- shift+zz = save + exit
- cntrl+d (or type exit) =exit
- U = undo
- : commands:
- :number= jumps to line
- : set number = display line numbers
- : set nonumber = remove line numbers
- :r file = import text file to text editor
- :1 = first line in row
- J = לאחד 2 שורות
- $= move to the end of **line**
- ^a (look for words that start with a)
- $a (look for words that ends with a)
- A – append after line
- | = שרשור פקודות
- / = search
- :sh = back to terminal
- & ctrl+d = back to vim
- :4 - Move to the line 4 (can replace 4 with any line)
- cat filname - show last 24 lines of the file, see the content of file

  יכול לראות רק 24 שורות אחרונות. כדי לראות את כל התוכן נשתמש בפקודה :more

  - אנטר - שורה הבאה
  - רווח - דף הבא
  - q - יציאה

  cat f1 f2 > f3 - put f1 and f2 in f3
  cat f1 f2 >> f3 - append f1 and f2 to end of f3
- more filname
  Enter - next line
  Space - next page
  q - exit
- command1 | command2 | ... - The output of command1 is the input for command2
  example1: ls -l /etc | more
  example2: (do the same as example1) ls -l /ets > kk
- head -5 filname - show first 5 lines of the file

  אם לא מוגדר מספר יודפסו 10 שורות
  אם המספר שהוגדר **שלילי** יודפסו כל השורות מלבד # השורות האחרונות
- tail -5 filname - show last 5 lines of the file

  עבור **-n+#** מדפיסה החל מהשורה ה# -
  ברירת המחדל היא הדפסת 10 השורות האחרונות–
- head -8 filename | tail -2 filename - show only line 7 and 8 of the file
- .exrc - properties (Settings) file of vi
- :%s/OLD/NEW/g - replace OLD with NEW

  אם אין אחוז אז זה רק בשורה הנוכחית
- ls -l /etc | more
  ls -l /path
  ls -a (hidden files)
  ls -l (long format)
  ls -t (sort by timestep descending newest is up)
  ls [a-c]* (start with a b or c)
  ls a* (start with a)

la ??a* (a in 3rd char)
ls -l | more (all result in more mode)

- mkdir - create directory (folder)
  mkdir dir1
  mkdir ./dir2

אם לא נותנים נתיב, יוצר את התיקייה בנתיב שבו אנו נמצאים

- cd - change directory
  cd dir1
  .this dircetory
  cd .. - change to home directory
  cd ~ - change to home directory
  cd /home/alexL
- -r - recursive
- -i - Prompt
- rm -r dir1 = remove dir1 with all files that he contains
  rmdir dir1 = remove dir1 *** only if dir empty (empty dircetroy)
  rm file1 = remove file1
  rm -i file1 = are you sure to delete file1 ?
  rm -r (recursive remove, remove dir that not empty)

- cp file1 file2 = copy file1 to fil2 ,
  (**if f2 not exist it create f2)** if file2 exist need to do -i ,because he do overwrite
  cp ../a1 a2 = copy a1 to a2 , in this dir
- mv x y = move x to y and delete x (x and y are files) (rename method)
  mv -i filname1 filname2
- wc fileName - word couter, show how many characters words and lines in fileName
  wc -c - by characters
  wc -w - by words
  wc -l - by lines
  Who|wc –l לספור משתמשים מחוברים

- ln file1.txt file2.txt - create hard link (only in the same disc\partition)
  ln -s file1.txt file2.txt - create symbolic link, prefer to use symbolic link

כשמוחקים קובץ צריך קודם לנתק את החיבור
משתמשים בחיבור סימבולי כשמקשרים בין תיקיות. כשמשתמשים בחיבור זה, יש יותר מידע ויזואלי על הקובץ/התיקייה
חיבור קשיח, בין קבצים באותה התיקייה ובאותו המחשב

- Cut

לפי מיקום אותיות Cut_–c1-2 _filename will cut 2 first chars -c
לפי דלימיטר Cut_–f2-3 –d_" "_filename (-d " " delimiter) -f

- Echo

מחזיר את הערך של הביטוי בפנים
Echo abcdef|cut –c2-4
will return bcd
דוגמא להראות ערך של איבר:
n=10
echo n show n
echo $n show 10

echo $SHALL will return the shell

- touch "file name"-
  create file in size 0 if no exists else changes the modify date whit out change the file
    - why size 0 ? - in past used as flags
    - ls –k* -
    - ls /etc
    - ls ?u*   *to exit
- more /etc/group - show all groups
- Alias - shortcuts for commands
  Alias ld ="ls –l /etc | grep ^d"
  (alias lk="ls -lst")
  נותן שם מקוצר לפקודה מסוימת בדוגמא:
  lk מסמן את הפקודה ls -lst
  alias instructs the shell to replace one string with another when executing commands.
  alias lk='ls -lst'
  alias rm='rm -i' - set the remove prompt every time
- Grep – מחפשת בתוך קובץ, חיפוש שורות שמכילות את המופיעים
  grep -c SOMETHING (count rows that contains SOMETHING)
  grep -l SOMETHING (return rows that NOT contains SOMETHING)
  -i not matter uppercase or lowercase
  -w find completes words
  -r read in files under the current dir
  -n shows line numbers
  ^a = find lines start with a
  $a = find lines ends with a
  grep ^[a-c] (return rows that start with a b or c)
  grep ??a* (return rows that has a in the 3rd char)

- ***Find***
  find . - from current dir and under
  find / - from root and undr
  find /dir - from dir and under
  find -name "*.txt" - all txt files
  find -size +xx (size over xx in kb)
  find -mtime +xx (modified in the last xx days)
  find -ctime +xx (created in the last xx days)
  find -amin -xx (modified in the last xx minutes)
  find -perm x (with permition x)
  find -type dosxx (from type dosxx)
  find -exec rm {} \; (remove all files that founds)

- **Sort**
  sort -o FILENAME (put result in FILENAME)
  sort -m (merge)
  sort -m “filename1” “filename2” will create one sorted file
  sort -n (first numeric column)
  sort -f (not case sensative)
  sort -kX (according column X)
  sort from 2nd column -k2
  sort -u
  sort|uniq

הפקודה uniq מדפיסה עותק יחיד של שורות זהות סמוכות

> uniq[options] [input [output]]

–עבור קובץ ממוין שורות זהות לא יודפסו יותר מפעם אחת

Sort –file = sort by default first columns

Sort –k2 -file = sort by second columns

Sort –r –file = reverse sort

Sort –n1 –file = sort by number colums

cat filename1 filename2 |sort|uniq will marge 2 files and then cat the not uniq ones

-r reverse sort

- **cal** - showing calender on shell.

Permisions:

in unix you give permisions for concret file only

there 3 permissions personas:

for user => **-u**. בעל הקובץ

for group => **-g**. משתמש אשר שייך לקבוצה של בעל הקובץ

for others => **-o**. שאר המשתמשים במערכת

Permissions determine who can access the file.

view etc/passwd - will show each user info

view etc/group - will show the groups info

drwxr-xr-x

1st 3 are U (user)

2nd 3 are G (group)

3rd *3* are O (others)

9 last positions are permitions

r - read a file.

w - write to a file or create.

x - permission to execute a program file. הרצת הקובץ

chmod (change mod)_“filename”:

for my self - u+x

for my group - g+w

for other - o-e

** in linux u can give all 3 permisions in one command like g+xwr

also

4 - read

2- write

1 - exc

and then you can use the cmmod 664 (6 = 4+2 [r+w]) when using this syntax you muse give the persmision to all U+G=O

umask - define defoult (for my self) in unix 3 parameters

in linux 4 parameters

** 1st paramter not used by us - when u want to run afile with root permision

2nd user 3 group 4 others

**umask will be alive just for this session

example: -rwxrw-r-- 1 userName ……

the **user** gets r, w, x permissions.

the **group** gets r, w permissions.

the **others** gets r permissions.

Scripts:

$n - this will be the nth input that entered.

$# - shows the number of paramenters inputted.

$* - shows the parameters that inputted.

$0 - shows the scripts name.

$? - shows “0” if the last command succeeded, shows “1” if the last command failed.

the script:

```
echo "first is:" $1
echo "second is:" $2
echo "total number of prams:" $#
echo "the parms are:" $*
echo  "the sum is " `expr $1 +
$2`
echo  "the diff is " `expr $1 - $2`
```

the output: 6 3

```
first is: 6
second is: 3
total number of prams: 2
the parms are: 6 3
the sum is  9
the diff is  3
```

Math- ematical Operator in Shell Script	Meaning	Normal Arithmetical/ Mathematical Statements	But in Shell	
-eq	is equal to	5 == 6	if test 5 -eq 6	if expr [5 -eq 6]
-ne	is not equal to	5 != 6	if test 5 -ne 6	if expr [5 -ne 6]
-lt	is less than	5 < 6	if test 5 -lt 6	if expr [5 -lt 6]
-le	is less than or equal to	5 <= 6	if test 5 -le 6	if expr [5 -le 6]
-gt	is greater than	5 > 6	if test 5 -gt 6	if expr [5 -gt 6]
-ge	is greater than or equal to	5 >= 6	if test 5 -ge 6	if expr [5 -ge 6]

Operator	Meaning
string1 = string2	string1 is equal to string2
string1 != string2	string1 is NOT equal to string2
string1	string1 is NOT NULL or not defined
-n string1	string1 is NOT NULL and does exist
-z string1	string1 is NULL and does exist

Test	Meaning
-s file	Non empty file
-f file	Is File exist or normal file and not a directory
-d dir	Is Directory exist and not a file
-w file	Is writeable file
-r file	Is read-only file
-x file	Is file is executable

Operator	Meaning
! expression	Logical NOT
expression1 -a expression2	Logical AND
expression1 -o expression2	Logical OR

If

```
if <expression>; then
      <commands>
else
      <commands>
fi
```

While

```
while <expression>; do
      <command1>
      <command2>
      ...
done
```

For

```
for loop-index; do
      <command1>
      <command2>
      ...
done
```

Case

```
case  test_string in
 pattern-1 )
          commands_1
         ;;
 pattern-2 )
      commands_2
         ;;
 … …
esac
```

תרגילי QUIZZ

- *grep -l '^#include' /usr/include/**
  רשימת הקבצים שמכילים את הביטוי בספריה
- *grep -c /bin/tcsh /etc/passwd*
  רשימת המשתמשים שמשתמשים ב TCSH
- *grep -c pattern files | grep :0*
  רשימת הקבצים שלא מכילים את הפטרן
- *paste -s -d"\t\n" file_name*
  מאחד שתי שורות לשורה אחת
- *cut -c4 file | paste - file*
  גוזר את התו הרביעי ומדביק בתחילת השורה
- *$ cat f1*
  peach apple cherry
  cat f1 | tr "" "\n" | sort
  מחליף רווח בירידת שורה וממיין

  answer:
  apple
  cherry
  peach

- *command cp x y && echo "ok" || "else" // print "ok" || "else" according to cp command result*

- ***cmp -s old new && echo 'no changes'***
  מדפיס הודעה אם שני הקבצים זהים
- ***find .\! -name '[A-Z] *' -exec lpr {}\;***
  שולח להדפסה את כל שמות הקבצים בתיקייה הזאת שלא מתחילים באותיות גדולות
- ***find / -size 0 -ok rm {} \;***
  מוחק את כל הקבצים הריקים במערכת (prompting first)
- ***ls olddir | xargs -i -t mv olddir/{} newdir/{}***
  מעביר מהספריה הישנה לחדשה ומראה כל פקודה
- ***cat file | tr -s "" " " > new.file***
  מוריד את כל הרווחים ושומר בקובץ החדש
- ***tail -2b bigfile***
  מדפיס את שני הבלוקים האחרונים של הקובץ
- ***find . -maxdepth 1 -type f -newer first_file***
  חיפוש קבצים רק בספריה המקומית בלי תת הספריות
- ***grep -n '[dD]on\'t' tasks***
  לחפש בקובץ טאסק את המילה דונט
- ***ls -a /etc | grep ^[.].* > file2***
  להכניס את כל הקבצים הנסתרים לתוך הקובץ
- ***grep -v -c this demo_file***
  כמה שורות לא תואמות ל PATTERN
- ***grep -w 'word1 | word2' /path/to/file***
  לחפש 2 מילים שונות
- ***find . -name "rc.conf" -exec chmod o+r '{}' \;***
  לחפש בתיקייה הנוכחית ובתתי התיקיות. כל הקבצים עם השם הזה יבוצעו על ידי סצ' מוד או+ר קומנד
- ***find . -name "*.tmp" -size +2000 -mtime +5 -exec rm {} \;***
  (remove all tmp file that over 2000kb (2Mb) that are modified in the last 5 days)
- ***grep textToFind fileName - find all rows with textToFind in fileName***
  grep Alex kuku - find all rows with Alex in kuku
  grep -c Alex kuku - count how many row with Alex in kuku

- ***ls -l|grep ^d - find all directories in cuurnet directory in long format***
- ***ls -la|grep ^d - find all directories in cuurnet directory in long format (include hidden files)***

- grep -n '[dD]on\'t' tasks → (Looks for the word Don\don in tasks)
- ls -l | grep '^d…………x' → (shows folder that contains permisions of others משהו
- xargs - used to help in editing results from grab and find

Shell Script Exercises

http://www.freeos.com/guides/lsst/ch08.html

1. Write shell script that will add two numbers, which are supplied as command line argument, and if this two numbers are not given show error and its usage

```
if [ $# -ne 2 ]
then
    echo "Usage - $0   x    y"
    echo "        Where x and y are two nos for which I will print 
sum"
    exit 1
fi
    echo "Sum of $1 and $2 is `expr $1 + $2`"
```

2.Write Script to find out biggest number from given three numbers. Numbers are supplies as command line argument. Print error if sufficient arguments are not supplied

```
if [ $# -ne 3 ]

then
    echo "$0: number1 number2 number3 are not given" >&2
    exit 1
fi
n1=$1
n2=$2
n3=$3
if [ $n1 -gt $n2 ] && [ $n1 -gt $n3 ]
then
    echo "$n1 is Bigest number"
elif [ $n2 -gt $n1 ] && [ $n2 -gt $n3 ]
then
    echo "$n2 is Bigest number"
elif [ $n3 -gt $n1 ] && [ $n3 -gt $n2 ]
then
    echo "$n3 is Bigest number"
elif [ $1 -eq $2 ] && [ $1 -eq $3 ] && [ $2 -eq $3 ]
then
    echo "All the three numbers are equal"
else
    echo "I can not figure out which number is biger"
fi
```

3.Write script to print numbers as 5,4,3,2,1 using while loop.

```
i=5
while test $i != 0
do
    echo "$i"
    i=`expr $i - 1`
done
```

4. Write Script, using case statement to perform basic math operation as

+ addition
- subtraction
x multiplication
/ division

```
echo -----------------------------------------------------
echo '\tEvaluation of Arithmetic expression'
echo -----------------------------------------------------
echo Enter the a value
read a
echo  Enter the b value
read b
echo 1.Addition
echo 2.Subtraction
echo 3.Multiplication
echo 4.Division
echo 5.Modules
echo Enter your choice
read choice
case $choice in
	1)echo Addition       : $(expr $a + $b);;
	2)echo Suubtraction   : $(expr $a - $b);;
	3)echo Multiplication : $(expr $a \* $b);;
	4)echo Division       : $(expr $a / $b);;
	5)echo Modules        : $(expr $a % $b);;
	*)echo This is not a choice
esac
```

5.Write Script to see current date, time, username, and current directory

```
echo "Hello, $LOGNAME"
echo "Current date is `date`"
echo "User is `who i am`"
echo "Current directory `pwd`"
```

6.Write script to print given number in reverse order, for eg. If no is 123 it must   print as 321.

```
# Algo:
#       1) Input number n
#       2) Set rev=0, sd=0
#      3) Find single digit in sd as n % 10 it will give (left most
digit)
#       4) Construct revrse no as rev * 10 + sd
#       5) Decrment n by 1
#       6) Is n is greater than zero, if yes goto step 3, otherwise
next step
#       7) Print rev
#
if [ $# -ne 1 ]
```

```
then
    echo "Usage: $0   number"
    echo "       I will find reverse of given number"
    echo "       For eg. $0 123, I will print 321"
    exit 1
fi

n=$1
rev=0
sd=0

while [ $n -gt 0 ]
do
    sd=`expr $n % 10`
    rev=`expr $rev \* 10  + $sd`
    n=`expr $n / 10`
done
    echo  "Reverse number is $rev"
```

7.Write script to print given numbers sum of all digit, For eg. If no is 123 it's sum of all digit will be 1+2+3 = 6.

```
# Algo:
#       1) Input number n
#       2) Set sum=0, sd=0
#        3) Find single digit in sd as n % 10 it will give
(left most digit)
#       4) Construct sum no as sum=sum+sd
#       5) Decrment n by 1
#       6) Is n is greater than zero, if yes goto step 3,
otherwise next step
#       7) Print sum
#
if [ $# -ne 1 ]
then
    echo "Usage: $0   number"
    echo "       I will find sum of all digit for given number"
    echo "       For eg. $0 123, I will print 6 as sum of all
digit (1+2+3)"
    exit 1
fi

n=$1
sum=0
sd=0
while [ $n -gt 0 ]
do
    sd=`expr $n % 10`
    sum=`expr $sum + $sd`
    n=`expr $n / 10`
done
    echo  "Sum of digit for numner is $sum"
```

8. Write script to print contains of file from given line number to next given number of lines. For e.g. If we called this script as test8 and run as

$ ./test8 5 5 myf , Here print contains of 'myf' file from line number 5 to   next 5 line of that file.

```
# Print error / diagnostic for user if no arg's given
#
if [ $# -eq 0 ]
then
    echo "$0:Error command arguments missing!"
    echo "Usage: $0 start_line   uptoline   filename"
    echo "Where start_line is line number from which you would
like to print file"
    echo "uptoline is line number upto which would like to
print"
    echo "For eg. $0 5 5 myfile"
    echo "Here from myfile total 5 lines printed starting from
line no. 5 to"
    echo "line no 10."
    exit 1
fi
# Look for sufficent arg's

 if [ $# -eq 3 ]; then
  if [ -e $3 ]; then
            tail +$1 $3 | head -n$2
         else
            echo "$0: Error opening file $3"
      exit 2
  fi
    else
        echo "Missing arguments!"
    fi
```

9. Write script called sayHello, put this script into your startup file called .cshrc,   the script should run as soon as you logon to system, and it print any one of   the following message according to system time:

Good Morning
Good Afternoon
Good Evening ,.

```
temph=`date | cut -c12-13`
dat=`date +"%A %d in %B of %Y (%r)"`

if [ $temph -lt 12 ]
then
    mess="Good Morning $LOGNAME, Have nice day!"
fi

if [ $temph -gt 12 -a $temph -le 16 ]
then
    mess="Good Afternoon $LOGNAME"
fi

if [ $temph -gt 16 -a $temph -le 18 ]
then
    mess="Good Evening $LOGNAME"
fi
```

```
if which dialog > /dev/null
then
    dialog --backtitle "Linux Shell Script Tutorial"\
    --title "(-: Welcome to Linux :-)"\
    --infobox "\n$mess\nThis is $dat" 6 60
    echo -n "                             Press a key to continue. . .
"
    read
    clear
else
    echo -e "$mess\nThis is $dat"
fi
```

10. How to write script, that will print, Message "Hello World" , in Bold and Blink effect, and in different colors like red, brown etc using echo command.

Answer: See Q16 shell Script

```
# echo command with escape sequance to give differnt effects
#
# Syntax: echo -e "escape-code your message, var1, var2 etc"
# For eg. echo -e "\033[1m  Hello World"
#                    |          |
#                    |          |
#                Escape code   Message
#

clear
echo -e "\033[1m Hello World"
 # bold effect
echo -e "\033[5m Blink"
       # blink effect
echo -e "\033[0m Hello World"
 # back to noraml
echo -e "\033[31m Hello World"
 # Red color
echo -e "\033[32m Hello World"
 # Green color
echo -e "\033[33m Hello World"
 # See remaing on screen
echo -e "\033[34m Hello World"
echo -e "\033[35m Hello World"
echo -e "\033[36m Hello World"
echo -e -n "\033[0m "
  # back to noraml

echo -e "\033[41m Hello World"
echo -e "\033[42m Hello World"
echo -e "\033[43m Hello World"
echo -e "\033[44m Hello World"
echo -e "\033[45m Hello World"
echo -e "\033[46m Hello World"

echo -e "\033[0m Hello World"
  # back to noraml
```

11. Write shell script to show various system configuration like

1) Currently logged user and his logname

2) Your current shell
3) Your home directory
4) Your operating system type
5) Your current path setting
6) Your current working directory
7) Show Currently logged number of users
8) File system (Mounted)

```
    nouser=`who | wc -l`
    echo -e "User name: $USER (Login name: $LOGNAME)"
    echo -e "Current Shell: $SHELL"
    echo -e "Home Directory: $HOME"
    echo -e "Your O/s Type: $OSTYPE"
    echo -e "PATH: $PATH"
    echo -e "Current directory: `pwd`"
    echo -e "Currently Logged: $nouser user(s)"

    if [ -f /etc/redhat-release ]
    then
        echo -e "OS: `cat /etc/redhat-release`"
    fi

    if [ -f /etc/shells ]
    then
        echo -e "Available Shells: "
        echo -e "`cat /etc/shells`"
    fi

    echo -e "--------------------------------"
    echo -e "File System (Mount):"
    echo -e "-------------------------------------------------------
-------"
    cat /proc/mounts
```

12. to determine whether given command line argument ($1) contains "*" symbol or not, if $1 does not contains "*" symbol add it to $1, otherwise show message "Symbol is not required". For e.g. If we called this script test9 then after giving ,
$ ./test /bin
Here $1 is /bin, it should check whether "*" symbol is present or not if not it should print Required i.e. /bin/*, and if symbol present then Symbol is not required must be printed.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Q12
# Script to check whether "/*" is included, in $1 or not
#
```

```
cat "$1" > /tmp/file.$$   2>/tmp/file0.$$

grep "*"  /tmp/file.$$    >/tmp/file0.$$

if [ $? -eq 1 ]
then
    echo "Required i.e. $1/*"
else
    echo "Symbol is Not required"
fi

rm -f /tmp/file.$$
rm -f /tmp/file0.$$
#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using
this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

13. To Generate Fibonacci Series

Explanation:
0,1,1,2,3,5,8,13,21,34,55,89,144, ...
By definition, the first two Fibonacci numbers are 0 and 1, and each
subsequent number is the sum of the previous two.

```
##############################################################
#                        Script Starts Here
#
##############################################################

if [ $# -eq 1 ]
then
    Num=$1
else
    echo -n "Enter a Number :"
    read Num
fi

f1=0
f2=1

echo "The Fibonacci sequence for the number $Num is : "

for (( i=0;i<=Num;i++ ))
do
     echo -n "$f1 "
     fn=$((f1+f2))
     f1=$f2
     f2=$fn
done
echo
```

14) Decimal to Binary Conversion (Takes input as command line arguments)

```
function convertIntvalToBase () # (Val Base)
{
  val=$1
  base=$2
  result=""
  while [ $val -ne 0 ] ; do
     result=$(( $val % $base ))$result #residual is next digit
     val=$(( $val / $base ))
  done
  echo -n $result
}
```

15) To Check Whether a String is Palindrome or not

- **Examples:**

  Phrases: Dammit, I'm mad!

  Quotations: Able was I ere I saw Elba.

  Numbers: 5335, 123454321

  Dates: 01/02/2010 (dd/mm/yyyy format)

Tip: compare first character with last character, up to middle of the string.

```
S="mtest
"

R='';
P=''
for
((i=${#S
}; i>=0;
i--))
  do
R="$R"${
S:$i:1}
done
#echo $R
[[ $S =
$R ]] ||
P="No "
echo
"${P}Pal
indrome
```