Shell Scripting Guide

shell scripts

Ex
```
x=20
if [ $x -lt 50 ]
then
  echo "$x is less than 50"
fi

[shellscripts]bash if1.sh
20 is less than 50
[shellscripts]chmod u+x if1.sh
[shellscripts]ls
content.sh  forex.c  hell.sh  if1.sh  vars.sh
[shellscripts]./if1.sh
20 is less than 50
```

Ex

```
echo "enter x"
read x
if [ $x -lt 50 ]
then
  echo "$x is less than 50"
else
  echo "$x is not less than 50"
fi

[shellscripts]./if1.sh
20 is less than 50
[shellscripts]vi if1.sh
[shellscripts]./if1.sh
enter x
99
99 is not less than 50
[shellscripts]./if1.sh
enter x
```

```
3
3 is less than 50
```

```
Ex
echo "enter x"
read x
if [ $x -lt 50 ]
then
  echo "$x is less than 50"
elif [ $x -gt 50 ]
then
  echo "$x is greater than 50"
else
  echo "$x is  50"
fi


[shellscripts]vi if1.sh
[shellscripts]./if1.sh
enter x
89
89 is greater than 50
[shellscripts]./if1.sh
enter x
12
12 is less than 50
[shellscripts]./if1.sh
enter x
50
50 is  50
```

---

operators that can be used in if

-lt , -gt , -le , -ge , -eq , -ne

HW - WASS to accept a number and print whether it is odd or even

---

Loops

```
 for
 while
 until

for((x=1;x<10;x++))
do
 echo "$x"
done

while [ $x -lt 20 ]
do
 echo "$x"
 ((x=x+1))
done

until [ $x -gt 30 ]
do
 echo "$x"
 ((x=x+1))
done
```

Integer arithmetic
command expr

```
[shellscripts]expr 12 + 30
42
[shellscripts]expr 12 - 30
-18
[shellscripts]expr 12 * 30
expr: syntax error
[shellscripts]expr 12 \* 30
360
[shellscripts]expr 12 \ 30
expr: syntax error
[shellscripts]expr 12 / 30
0
```

WASS to accept two numbers and show the sum

```
echo "enter 2 numbers"
read a b
```

```
expr $a + $b
----------
echo "enter 2 numbers"
read a b

#using backquote to include the statement
sum=expr  $a  +  $b
echo "sum=$sum"

-----------------------------
WASS  to show a menu to the user
 + - *  /
 use if elif and show the result
 after every run ask do u want to continue
 y/n  }}  loop
-------------------------
 To compare string  = use  = , !=
echo "enter operator"
read s

if [ $s = '+' ]
then
 echo "adding"
fi
if [ $s != '+' ]
then
 echo "not adding"
fi
---------------------------------
Unary operator  = it has a single operand
   -f -e -d
binary operator =  -lt -gt = !=

echo "enter a file or folder name"
read f

if [ -f $f ]
then
 echo "$f is a file "
 cat $f
fi
```

```
if [ -d $f ]
then
  echo "$f is a folder"
  ls $f
fi

-----------------------------------
for f in ls
do
  echo "$f"
  if [ -f $f ]
  then
    cat $f
  fi

done
---------------------------------
command line argument

bash cmdline.sh one two three four
#positional parameters
echo "$1 $2 $3"
------
#positional parameters
echo "$1 $2 $3"

echo "$@"

sum=0
for x in $@
do
 ((sum=sum+x))
done
echo "sum=$sum"
_____________________________________
 Text manipulation commands
  sort
  head
  tail
  grep

_____________________________
```

```
[shellscripts]head -3 data

kohli 500
gukesh 1000
[shellscripts]tail -3 data
sachin 900
dhoni 800



head -lines  filename  }  top lines
tail -lines  filename }  bottom lines

------------------------
sort data  = sort by first col alphabetically/lexically
--------------
sort by second column numerically
sort -k 2 -n data

vaishali 400
kohli 500
prag 700
dhoni 800
sachin 900
gukesh 1000
----------------------
sort -k 2 -n -r data
sort by second col numerically desc (-r)
----------------------
PIPES   =  concatenate COMMANDS
command1 |  command2 | command3  }} output of
command3

[shellscripts]sort -k 2 -n -r data | head -1
gukesh 1000
[shellscripts]sort -k 2 -n -r data | head -2
gukesh 1000
sachin 900
[shellscripts]sort -k 2 -n -r data | head -3
gukesh 1000
sachin 900
dhoni 800
[shellscripts]sort -k 2 -n -r data | head -3 | tail -1
```

dhoni 800

_____

Redirection
copy the output of the command to a file
Two ways
> = Overwrite
>> = append

STORING the top3 names in a file top3
sort -k 2 -n -r data | head -3 >> top3

_____
wc = word count
echo "alphonso"| wc -c

[shellscripts]wc -c data
67 data
[shellscripts]wc -l data
8 data
[shellscripts]wc -w data
12 data

WASS to accept a folder path and show how many entries are in it

ls | wc -l

_____

```
x=ps -e | wc -l
((x=x-1))
echo "number of processes are $x"
```

_____

WASS to accept numbers from user till user enters -1
save all numbers to a file numdata
LOOP
enter a num
read n

```
    if ..
    ...
    echo "$n" >> numdata

sort the numbers in ascending
      sort -n  numdata
show count of total numbers added in the file
      cat numdata | wc -l
add all numbers in the file
      for x in cat  numdata
      do
       ((sum=sum+x))
      done
---------------------------------
 grep  =  grep kya karta hai  = "grep searches"

 Regular Expression --- it is a way to describe PATTERNs

  grep uses the pattern as input and gives the matching
output

grep "a" data
grep "echo" *

[shellscripts]grep "^$" data | wc -l
2
[shellscripts]grep -v  "^$" data | wc -l
6
[shellscripts]wc -l data
8 data

--------------------------------
[shellscripts]echo "hello" | grep "e"
hello
[shellscripts]echo "hello" | grep "ee"
[shellscripts]echo "hello" | grep -E  "[a-z]+"
------------
+  =  1 or more occurences
*  =  0 or more occurences
?  =  single occurence
```

GREP searching for title case
[shellscripts]echo "Prachi" | grep -E "[A-Z][a-z]+"
Prachi
[shellscripts]echo "prachi" | grep -E "[A-Z][a-z]+"

GREP searching for 0 to 3 occurences of chars
[shellscripts]echo "ww" | grep -E "[a-z]{3}"
[shellscripts]echo "www" | grep -E "[a-z]{3}"
www
[shellscripts]echo "wwwww" | grep -E "[a-z]{3}"
wwwww
[shellscripts]echo "wwwww" | grep -E "[a-z]{3}$"
wwwww
[shellscripts]echo "wwwww" | grep -E "^[a-z]{3}$"
[shellscripts]echo "ww" | grep -E "^[a-z]{3}$"
[shellscripts]echo "wws" | grep -E "^[a-z]{3}$"
wws

_____

grep -E "^[A-Z].*" 0 or more occurences of any char after a capital

grep -E "^[A-Z][a-z]+"

. = any single char
* = 0 or more times occuring
+ = 1 or more occurences
? = 0 or 1 occurence
[A-Z]
[a-z]
[px] = either p or x
l{2} = 2 times occurence of l
r{2,4} = 2 , 3 or 4 times r has occured

find out a regular expression for valid mail ID
prachi@gmail.com

@ should be preceded by any number of chars or numbers
@ should be followed by any number of chars followed by . followed by 3 letter

grep -E "[a-z A-Z 0-9]+@[a-z A-Z]+.[a-z]{2,3}"

Regular expression for a phone number
020-12345678

grep -E "[0-9]{3}-[0-9]{8}"===make me exam ready

Here's your **Shell Scripting Exam-Ready Notes**—neatly formatted with all key topics, examples, commands, and output patterns. You can save or convert this into a PDF later.

🐚 Shell Scripting – Quick Revision Guide

✅ Basics

```bash
x=20
if [ $x -lt 50 ]; then
  echo "$x is less than 50"
fi
```

Run & Permission

```bash
bash if1.sh      # Runs script
chmod u+x if1.sh  # Grants execute permission
./if1.sh         # Executes the script
```

🧮 Conditional Statements

`if-else`

```bash
```

```
echo "Enter x"
read x
if [ $x -lt 50 ]; then
  echo "$x is less than 50"
else
  echo "$x is not less than 50"
fi
```

`if-elif-else`

bash

```
echo "Enter x"
read x
if [ $x -lt 50 ]; then
  echo "$x is less than 50"
elif [ $x -gt 50 ]; then
  echo "$x is greater than 50"
else
  echo "$x is 50"
fi
```

🔢 Operators in Conditions

- `-lt` → less than
- `-gt` → greater than
- `-le` → less than or equal
- `-ge` → greater than or equal
- `-eq` → equal
- `-ne` → not equal

📝 Homework Practice

WASS – Write A Shell Script to check if a number is odd or even:

```bash
read x
(( x % 2 == 0 )) && echo "Even" || echo "Odd"
```

🔁 Loops

`for` loop

```bash
for((x=1; x<10; x++)); do
  echo "$x"
done
```

`while` loop

```bash
x=1
while [ $x -lt 20 ]; do
  echo "$x"
  ((x++))
done
```

`until` loop

```bash
x=1
until [ $x -gt 30 ]; do
  echo "$x"
  ((x++))
done
```

➕ Arithmetic Using `expr`

```bash
expr 12 + 30     # 42
expr 12 \* 30    # 360 (note: \* is escaped)
expr 12 / 30     # 0
```

Sum of Two Numbers

```bash
read a b
sum=`expr $a + $b`
echo "sum=$sum"
```

🗒️ Menu-Based Calculator

```bash
while true; do
 echo "Enter two numbers:"
 read a b
 echo "Enter operator (+ - * /):"
 read op

 if [ $op = "+" ]; then
  res=`expr $a + $b`
 elif [ $op = "-" ]; then
  res=`expr $a - $b`
 elif [ $op = "*" ]; then
  res=`expr $a \* $b`
 elif [ $op = "/" ]; then
  res=`expr $a / $b`
 else
  echo "Invalid operator"
  continue
 fi
 echo "Result: $res"
```

```
  echo "Continue? (y/n)"
  read ans
  [ $ans = "n" ] && break
done
```

📁 File Tests

bash

```
read f
if [ -f $f ]; then
  echo "$f is a file"
  cat $f
elif [ -d $f ]; then
  echo "$f is a directory"
  ls $f
fi
```

📂 Loop Through Files

bash

```
for f in `ls`; do
  echo "$f"
  if [ -f $f ]; then
    cat $f
  fi
done
```

📜 Command Line Arguments

bash

```
# Save as cmdline.sh
echo "$1 $2 $3"
sum=0
for x in "$@"; do
 ((sum += x))
done
echo "sum=$sum"
```

📄 Text Processing Commands

- `head -3 file` – first 3 lines
- `tail -3 file` – last 3 lines
- `sort file` – sort lexically
- `sort -k 2 -n file` – sort numerically by 2nd col
- `sort -k 2 -n -r file` – numeric descending
- `command1 | command2` – pipe

bash

```
sort -k 2 -n -r data | head -3 > top3
```

🔀 Redirection

- `>` – overwrite
- `>>` – append

📊 Word Count

bash

```
wc -c file  # bytes
wc -l file  # lines
wc -w file  # words
```

Count processes

```bash
x=`ps -e | wc -l`
((x--))
echo "Processes: $x"
```

📁 Count Entries in Folder

```bash
ls | wc -l
```

🧮 Accept Numbers Until -1, Then Process

```bash
> numdata
while true; do
 echo "Enter a number (-1 to stop):"
 read n
 [ $n -eq -1 ] && break
 echo "$n" >> numdata
done

sort -n numdata
echo "Count:" `cat numdata | wc -l`

sum=0
for x in `cat numdata`; do
```

```
 ((sum+=x))
done
echo "Sum=$sum"
```

🔎 grep – Pattern Search

```bash
grep "a" file
grep -E "[a-z]{3}" file
grep -E "^[A-Z][a-z]+" file
grep -v "^$" file  # exclude blank lines
```

Common Regex

- `.` – any single char
- `*` – 0 or more
- `+` – 1 or more
- `?` – 0 or 1
- `[a-z]` – lowercase
- `[0-9]{3}` – 3 digits

📧 Regex for Valid Email

```bash
grep -E "[a-zA-Z0-9._]+@[a-zA-Z]+\.[a-z]{2,3}" file
```

☎ Regex for Phone Number

```bash
```

```
grep -E "[0-9]{3}-[0-9]{8}" file
```

Let me know if you'd like this content exported as a **PDF** or you want a **printable version with practice questions**.