**Shell Scripting**

**File Transfer (scp, rsync etc)**

**Grep command**

**TOP Command**

**What is Process and Process Managements**

**Cron Jobs**

**Group, users, permissions**

**Filters**

**File creation, deletion**

**Search a file**

**What is kernel, shell, features of ksh shell**

**Difference between multiuser and multitask**

**Way to transfer fill from windows to unix and vice-versa**

**Command Substitution**

**Inode**

**File System in Unix**

**Relative Path and Absolute Path**

**Difference Between System Call, library Function and Unix Command**

**Network Commands --- telnet, ping, su, ftp, finger**

**Difference between cmp and diff command**

**Use of –l , let it’s options**

**Determine path and set Path in unix**

**Process concept and it’s commands**

**Tee command**

**Tee command** is used to store and view (both at the same time) the output of any other command.

**Difference between more and cat**

**Parsing, pid**

**Sort command**

**Find command**

**How to switch from any user super user**

**Uniq command**

**Unix shut down and reboot**

**Touch command**

**Which OS you are working on “uname -a”**

**Running process in background “&” command and fg command**

**History command**

**How do you check how much space is left in current device**

**Difference between Page Swapping and Pagging**

**Difference between ps – ef and ps –auxwww**

**How to find how many CPU are in your system and their details**

**Cat /proc/cpuinfo**

**Hardlink and Softlink concept**

**What is Zombie process in Unix**

**There is a file in your system which contains a word “\_\_\_\_\_” how will you find that file in unix**

**How to set environment variable**

**How do you find your system is 32 bit or 64 bit**

**“uname -a” or arch command**

**Which process are using a particular file?**

**Find remote hosts are connecting to your host on a particular port say 101232**

**Netstat command**

**Netstat –a | grep “port”**

**Nohup in unix**

**Difference between nohup and &**

**Watch command**

**DU command**

**How to find since how many days server is UP ?**

**Nslookup**

**What is ephermel port ?**

**=======================================================**

**sed command** is used to perform search and replace operation on a file.

Syntax: sed –options script filename

-options

n: used to suppress the printing to standard output

e: used to execute commands basically when performing multiple search and replace on single file.

i: used to print output to same source file (risky)

script

script is nothing but operation specification to be performed on a file

s: substitute

‘s/pattern/substitute/flag’

Flag: p used to print the lines which got replaced

g: used to perform substitute operation on each occurrence of pattern

i: used to ignore case

h: used to replace contents of hold space with pattern space

pattern: ^ represent start search from start of line

$ represents

sed –n ‘s/patter/substitute/g’ test.file

sed –n ‘s/^inter\*in$/g’ test.file

sed ‘d’ test.file (deletes all the lines)

sed ‘/pattern/d’ test.file (searches for pattern and deletes)

sed –i ‘1d’ test.file (deletes first line and updates test.file)

**CUT Command**

cut -c [RANGE] [FILENAME]

-c option is used to capture character

-f to capture field

-d specify delimiter to search for (e.g. –d “|”)

--output-delimiter=”:” (replace delimiter with :)

--compliment –f 1 (print all fields except 1st field)

[Range]: from to column, **each character is considered as column here**. E.g. 1-4 (even white space is considered as character/column)

[FILENAME]: source file

To display 4th character from each row

cut -c 4 source.txt

to display characters from 5th to 10th

cut –c 5-10 source.txt

to print till end

cut –c 6- source.txt

$ cut -c 6- cuttest.txt

is line #1

line #2

is line #3

, this is line #4

line #5

line #6

f line #7

me, line #8

, I am line #9

line, line #10

to get first six characters

cut –c -6 source.txt

$ cut -c -6 cuttest.txt

This i

It is

That i

While,

It's l

I am l

Myself

It's m

Hello,

Last l

**Extracting Field from a File**

In order to understand this usage of cut command, lets consider a csv file as follows:

$ cat employees.txt

Employee ID, Employee Name, Age, Gender, Department, Salary

101, John Davies, 35, M, Finance, $4000

102, Mary Fernandes, 29, F, Human Resources, $3000

103, Jacob Williams, 40, M, Sales, $4700

104, Sean Anderson, 25, M, Production, $2700

105, Nick Jones, 42, M, Finance, $7500

106, Diana Richardson, 29, F, Finance, $3200

Remember, in order to extract a field from a file, we would need a delimiter (i.e. a column separator), based on which the file will be divided into columns and we can extract any of them. In this case, the syntax would be-

cut -d [DELIMITER] -f [RANGE] [FILENAME]

Here, we are instructing cut command to use a particular delimiter with option -d and then extract certain fields using option -f.

**1. Display a specific field from a file**

In case of a *csv* file, it is crystal clear that our delimiter will be a comma (,). Now, we need to enlist the names of the employees working in our organization, i.e. field number 2.

**Example:**

$ cut -d ',' -f 2 employees.txt

Employee Name

John Davies

Mary Fernandes

Jacob Williams

Sean Anderson

Nick Jones

Diana Richardson

**Displaying Multiple Fields from a File**

Moving forward now, lets display more than one field now. Suppose, we need to include 'Age' and 'Gender' fields also. For this, we must specify the range - again, a start and an end.

$ cut -d ',' -f 2-4 employees.txt

Employee Name, Age, Gender

John Davies, 35, M

Mary Fernandes, 29, F

Jacob Williams, 40, M

Sean Anderson, 25, M

Nick Jones, 42, M

Diana Richardson, 29, F

Conclusion, in this case, is that, *Input Delimiter = Output Delimiter*.  
  
Lets have a look at a variant in this case. Suppose, we need to extract *'Employee ID'*, *'Employee Name'*, *'Department'* and *'Salary'*. In that case, we need to specify two ranges as below:  
  
**Example:**

$ cut -d ',' -f 1-2,5-6 employees.txt

Employee ID, Employee Name, Department, Salary

101, John Davies, Finance, $4000

102, Mary Fernandes, Human Resources, $3000

103, Jacob Williams, Sales, $4700

104, Sean Anderson, Production, $2700

105, Nick Jones, Finance, $7500

106, Diana Richardson, Finance, $3200

This is just awesome!

**3. Change the Delimiter in the Output**

As we just saw in one of the examples above, by default, Input Delimiter = Output Delimiter. What if I wish to change the output delimiter? Just have a look at the example below:  
  
**Example:**

$ cut -d ',' -f 2-4 --output-delimiter='|' employees.txt

Employee Name| Age| Gender

John Davies| 35| M

Mary Fernandes| 29| F

Jacob Williams| 40| M

Sean Anderson| 25| M

Nick Jones| 42| M

Diana Richardson| 29| F

**4. Do not Display Certain Columns**

Just like above example, if we **use --complement** as an option, cut command will display all the fields, but the specified field.  
  
**Example:**

$ cut -d ',' --complement -f 6 employees.txt

Employee ID, Employee Name, Age, Gender, Department

101, John Davies, 35, M, Finance

102, Mary Fernandes, 29, F, Human Resources

103, Jacob Williams, 40, M, Sales

104, Sean Anderson, 25, M, Production

105, Nick Jones, 42, M, Finance

106, Diana Richardson, 29, F, Finance

**AWK command**

**sort command**

**find**

**find <start directory/specify . for root> –name “file-name pattern”**

**grep**

**grep is used to search for a particular content in a file**

**we can also use grep command to find all files containing a string**

**Double Quotes**

* Use when you want to enclose variables or use shell expansion inside a string.
* All characters within are interpreted as regular characters except for $ or ` which will be expanded on the shell.

**Single Quotes**

* All characters within single quotes are interpreted as a string character.

And thus ends the lesson of the quotes. Use them wisely.

**Operators used in Conditions**

= is used to compare between strings

-gt –lt –ge –le –eq –ne used to compare numbers

== used to compare numbers $a == $b

!= represents not equal

! not

-o or operator (&&)

-a and operator (||)

-f to check if file present

-z to check if string length is zero

-n to check if string length nonzero

-d to check if directory present

Disk used

du –sc \*

Disk free

df –h (h represents display in human readable format)