**How to Find out the version in Unix?**

uname –a

uname –r

uname -v

**What are Different Flavours in Unix?**

* AIX - developed by IBM for use on its mainframe computers
* HP-UX - developed by Hewlett-Packard for its HP 9000 series of business servers
* Solaris - developed by Sun Microsystems for the SPARC platform and the most widely used proprietary flavor for web servers
* [Linux](http://www.linfo.org/linuxdef.html) - the most popular and fastest growing of all the Unix-like operating systems
* Darwin - the new version of BSD that serves as the core for the Mac OS X

**What are Different Shells in Unix?**

* [Bourne shell (sh)](http://unixhelp.ed.ac.uk/shell/oview2.1.html)
* [C shell (csh)](http://unixhelp.ed.ac.uk/shell/oview2.2.html)
* [TC shell (tcsh)](http://unixhelp.ed.ac.uk/shell/oview2.3.html)
* [Korn shell (ksh)](http://unixhelp.ed.ac.uk/shell/oview2.4.html)
* [Bourne Again SHell (bash)](http://unixhelp.ed.ac.uk/shell/oview2.5.html)

**What are Positional parameters in Unix?**

Positional parameters in a shell script are nothing but the command line arguments passed to a shell script. The following are some of the positional parameters used:

$# -  Total number of arguments

$0 - Command or the script name

$1,$2, $3 - First, second and third args respectively.

$\* - All the command line arguments starting from $1.

The command line arguments can be accessed using $1, $2 till $9. However, if the number of command line arguments is more than 9, the same notation cannot be used. Instead, it should be used like ${10}, ${11} and so on. When $10 is used, the shell interprets it as $1 concatenated with 0, and hence you get the result as a0 ($1 is a).

**What is the Significance of first line in shell script in Unix?**

This will probably be more than you wanted to know, but here goes anyway....  
Originally, we only had one shell on unix. When you asked to run a command, the shell would attempt to invoke one of the exec() system calls on it. It the command was an executable, the exec would succeed and the command would run. If the exec() failed, the shell would not give up, instead it would try to interpet the command file as if it were a shell script.  
  
Then unix got more shells and the situation became confused. Most folks would write scripts in one shell and type commands in another. And each shell had differing rules for feeding scripts to an interpreter.  
This is when the "#! /" trick was invented. The idea was to let the kernel's exec() system calls succeed with shell scripts. When the kernel tries to exec() a file, it looks at the first 4 bytes which represent an integer called a magic number. This tells the kernel if it should try to run the file or not. So "#! /" was added to magic numbers that the kernel knows and it was extended to actually be able to run shell scripts by itself. But some people could not type "#! /", they kept leaving the space out. So the kernel was exended a bit again to allow "#!/" to work as a special 3 byte magic number.

So   
#! /usr/bin/ksh  
and  
#!/usr/bin/ksh  
now mean the same thing. I always use the former since at least some kernels might still exist that don't understand the latter.And note that the first line is a signal to the kernel, and not to the shell. What happens now is that when shells try to run scripts via exec() they just succeed. And we never stumble on their various fallback schemes.

**How to find out which shell you are working currently?**

ps -p $$

**What are different ways to execute the shell script?**

we can run shell script in the following methods assuming test.sh is the script name,

1. sh test.sh

Tells the command to use “sh” shell to execute test.sh.

1. ./test.sh

Tells the command to execute the script. The interpreter needs to be defined in the first line with something like #!/bin/sh or #!/bin/bash. Note ([thanks keltar](http://stackoverflow.com/questions/18331893/difference-between-different-ways-of-running-shell-script/18331933?noredirect=1#comment26906428_18331933)) that in this case the file test.sh needs to have execution rights for the user performing this command. Otherwise it will not be executed.In both cases, all variables used will expire after the script is executed.

1. . ./test.sh

Sources the code. That is, it executes it and whatever executed, variables defined, etc, will

persist in the session.

[**How to find exit status of script or command in Linux?**](http://www.linuxnix.com/2011/03/find-exit-status-script-command-linux.html)

In **Linux/Unix** when you execute a command or a script, they will exit with a meaning full **exit status** for your understanding purpose. So that we can take necessary actions on the out come(pass, failed or partially completed) of those commands. he exit status is stored in "$?" internal(builtin) variable. The exit status value varies from 0 to 255. Some of the commonly used exit status are as below.

0 Successful execution of command

1 command fails because of an error during expansion or redirection, the exit status is greater than zero.

2 Incorrect command usage

126 Command found but not executable

127 Command not found

surendra@Krishna:~/Templates$ ls  
New.doc  
surendra@Krishna:~/Templates$ echo $?  
0

**what's the difference between absolute and relative path?**  
A path is a unique location to a file or a folder in a file system of an OS. A path to a file is a combination of / and alpha-numeric characters.

Absolute path: An absolute path is defined as the specifying the location of a file or directory from the root directory(/). In other words we can say absolute path is a complete path from start of actual filesystem from / directory.

Some examples of absolute path:

/var/ftp/pub  
/etc/samba.smb.conf  
/boot/grub/grub.conf

If you see all these paths started from / directory which is a root directory for every Linux/Unix machines.

Relative path:Relative path is defined as path related to the present working directory(pwd). Suppose I am located in /var/log and I want to change directory to /var/log/kernel. I can use relative path concept to change directory to kernel changing directory to /var/log/kernel by using relative path concept.

pwd  
/var/log  
cd kernel

Note: If you observe there is no / before kernel which indicates it's a relative directory to present working directory.

Changing directory to /var/log/kernel using absolute path concept.

cd /var/log/kernel

Note: We can use an absolute path from any location where as if you want to use relative path we should be present in a directory where we are going to specify relative to that present working directory.

**Different Types of files in Unix?**

The following is a table listing what characters represent what types of files:

- Ordinary or Regular File  
d Directory  
c Character special file  
b Block special file  
l Symbolic link  
p Named pipe  
s Socket

The file command is also helpful for determining file types. The syntax for this command is:

$ file filename

**Different processes in Unix?**

<http://www.theunixschool.com/2012/09/what-is-process-in-unix-linux.html>

<http://www.thegeekstuff.com/2010/05/unix-background-job/>

**How do you kill the process in Unix?**

Kill sends signal to the process with process ID as pid. If no option is specified ,it sends signal 15 (software termination signal) to the process.

Option - 1 Hang-up signal.

Option - 2 Interrupt signal.

Option - 9 terminate the process.

Option -15 Software termination signal.

kill -9 pid -- kills the process

**Output Redirection in Unix?**

<http://www.codecoffee.com/tipsforlinux/articles2/042.html>

**crontab in Unix?**

crontab command is used to schedule jobs. You must have permission to run this command by unix Administrator. Jobs are scheduled in five numbers, as follows.

Minutes 0-59

Hour 0-23

Day of month 1-31

month 1-12

Day of week 0-6 (0 is sunday)

so for example you want to schedule a job which runs from script named backup\_jobs in /usr/local/bin directory on sunday (day 0) at 11.25 (22:25) on 15th of month. The entry in crontab file will be. \* represents all values.

25 22 15 \* 0 /usr/local/bin/backup\_jobs

The \* here tells system to run this each month.

Syntax is

crontab file So a create a file with the scheduled jobs as above and then typecrontab filename .This will scheduled the jobs

**Mail Command in Unix?**

Mail command is used to send mails.

**Some basic Unix Command?**

uname -a

uname -r

uname -v

banner

who

who am i

echo shirish

pwd

cd /apps/users01/umredkas/shirish

cd ..

cd

ls

ls –lrt

ls –a ( To find hidden files )

ls –ldrt \*/ ( To list only directories )

ls –ldrt | grep ^d ( To list only directories )

find . -mindepth 1 -maxdepth 1 -type d -print

**Significance of –lrt in ls command in unix?**

-a displays all files

-l Displays the long format listing.

-t Displays newest files first. (based on timestamp). sort by modification time

-r Displays files in reverse order according to alphabet.

r->recursive sort – only when we do pattern search using ls –lrt shirish\*

ls -ltr : displays files based on timestamp in ascending order.

mv command.

------------------

mv command is used to move a file from one directory to another directory or to rename a file.

\* Some examples: mv oldfile newfile will rename oldfile to newfile.

\* mv -i oldfile newfile for confirmation prompt.

\* mv -f oldfile newfile will force the rename even if target file exists.

\* mv \* /usr/bajwa/ will move all the files in current directory to /usr/bajwa directory.

rm command.

------------------

To delete files use rm command.

\* Options: rm oldfile will delete file named oldfile.

\* rm -f option will remove write-protected files without prompting.

\* rm -r option will delete the entire directory as well as all the subdirectories, very dangerous command.

rmdir command.

------------------

rmdir command will remove directory or directories if a directory is empty.

\* Options: rm -r directory\_name will remove all files even if directory is not empty.

\* rmdir sandeep is how you use it to remove sandeep directory.

\* rmdir -p will remove directories and any parent directories that are empty.

\* rmdir -s will suppress standard error messages caused by -p.

touch command.

------------------

touch command modifieds the time of existing file to current time or creates the empty file if file

does not exists

mkdir shirish

touch shirish.txt

mv shirish.txt shirish.txt1

cp -p shirish.txt1 shirish.txt

rm -f \*

rm -r /apps/users01/umredkas/shirish

rmdir /apps/users01/umredkas/shirish

chmod 777 shirish

chmod +rwx shirish

chown umredkas:tdb shirish

id umredkas

groups umredkas

**Which Editor you use to Edit the Unix file & Basic Commands?**

VI Editor

<http://www.cs.colostate.edu/helpdocs/vi.html>

**How to Find Unique lines in a file?**

To remove adjacent repeated lines**.**

Option -u display only the lines not repeated in the file named input.

Option -d display only the lines that are repeated in the file input.

Option -c precede each line displayed by the number of times it occurs.

+n ignore the first n characters including white space

-m ignore the first m fields and any blanks.

unique.txt

**SHIRISH**

**SHIRISH**

**AMIT**

**NIKHIL**

uniq unique.txt

**Different ways to view the file in unix?**

cat unique.txt

less unique.txt

more unique.txt

view unique.txt

vi unique.txt

**Counting the number of lines/characters/word in a file in unix?**

wc -l unique.txt

wc -c unique.txt

wc -w unique.txt

wc command counts the characters, words or lines in a file depending upon the option.

\* Options wc -l filename will print total number of lines in a file.

\* wc -w filename will print total number of words in a file.

\* wc -c filename will print total number of characters in a file.

**Sorting data in a file?**

sort command sort the lines of a file or files, in alphabetical order. for example if you have a file named testfile with these contents

zzz

aaa

1234

yuer

wer

qww

wwe

Then running

sort testfile

will give us output of

1234

aaa

qww

wer

wwe

yuer

zzz

\* Options: -b ignores leading spaces and tabs.

\* -c checks whether files are already sorted.

\* -d ignores punctuation.

\* -i ignores non-printing characters.

\* -n sorts in arithmetic order.

\* -ofile put output in a file.

\* +m[-m] skips n fields before sorting, and sort upto field position m.

\* -r reverse the order of sort.

\* -u identical lines in input file apear only one time in output.

sort unique.txt

sort –r unique.txt

**Process Command in Unix?**

PS command

ps command is probably the most useful command for systems administrators. It reports information on active processes.

ps options

\* options. -a Lists all processes in system except processes not attached to terminals.

\* -e Lists all processes in system.

\* -f Lists a full listing.

\* -j print process group ID and session ID.

ps -e -- lists all proceses with information about process

ps -ef

**How to find out which processes/CPU is taking time in unix?**

Top

**Head / Tail Command in Unix?**

Head command.

------------------

head filename by default will display the first 10 lines of a file. If you want first 50 lines you can use head -50 filename or for 37 lines head -37 filename and so forth.

Tail command.

------------------

tail filename by default will display the last 10 lines of a file. If you want last 50 lines then you can use tail -50 filename.

less unique.txt

SHIRISH

SHIRISH

AMIT

NIKHIL

**First two record in a file ?**

less unique.txt | head -2

SHIRISH

SHIRISH

**Last two record in a file ?**

less unique.txt | tail -2

AMIT

NIKHIL

**3rd last record in a file ?**

less unique.txt | tail -3 | head -1

SHIRISH

**grep/zgrep Command in Unix?**

less unique.txt

SHIRISH

SHIRISH

AMIT

NIKHIL

grep 'SHIRISH' unique.txt

SHIRISH

SHIRISH

grep -i 'Shirish' unique.txt

SHIRISH

SHIRISH

grep -li 'Shirish' unique.txt

unique.txt

grep -li 'Shirish' \*

unique.txt

grep -i 'Shirish' unique.txt | wc -l

2

less unique.txt.gz

SHIRISH

SHIRISH

AMIT

NIKHIL

zgrep 'SHIRISH' unique.txt.gz

SHIRISH

SHIRISH

zgrep -i 'Shirish' unique.txt.gz

SHIRISH

SHIRISH

zgrep -li 'Shirish' unique.txt.gz

unique.txt.gz

zgrep -li 'Shirish' \*

unique.txt.gz

zgrep -i 'Shirish' unique.txt.gz | wc -l

2

**Compressing/Uncompressing/zipping/tarring files in Unix?**

gzip –f unique.txt

gunzip –f unique.txt.gz

compress -f unique.txt

uncompress -f unique.txt.Z

zip unique.txt.zip unique.txt

unzip -f unique.txt.zip

unix2dos unique.txt

dos2unix unique.txt

tar -cvf unique.tar unique.txt.zip unique.txt

tar -tvf unique.tar

tar -xvf unique.tar

tar -cvzf unique.tar.gz unique.txt.zip unique.txt

tar -tvzf unique.tar.gz

tar -xvzf unique.tar.gz

**Find command in Unix?**

How do you find files which were updated last monday from a directory

You can use -mtime option. It returns list of file if the file was last accessed N\*24 hours ago.

For example to find file in last 2 months (60 days) you need to use -mtime +60 option.

-mtime +60 means you are looking for a file modified 60 days ago.

-mtime -60 means less than 60 days.

-mtime 60 If you skip + or - it means exactly 60 days.

So to find text files that were last modified 60 days ago, use

$ find /home/you -iname "\*.txt" -mtime -60 -print

Display content of file on screen that were last modified 60 days ago, use

$ find /home/you -iname "\*.txt" -mtime -60 -exec cat {} \;

Count total number of files using wc command

$ find /home/you -iname "\*.txt" -mtime -60 | wc -l

<http://www.cyberciti.biz/faq/howto-finding-files-by-date/>

**AWK & SED command in Unix?**

Both are tools that transform text. BUT awk can do more things besides just manipulating text. Its a programming language by itself with most of the things you learn in programming, like arrays, loops, if/else flow control etc You can "program" in sed as well, but you won't want to maintain the code written in it.

Conclusion: Use sed for very simple text parsing. Anything beyond that, awk is better. In fact, you can ditch sed altogether and just use awk. Since their functions overlap and awk can do more, just use awk. You will reduce your learning curve as well.

**SED:**

<http://www.grymoire.com/Unix/Sed.html>

**AWK:**

<http://www.thegeekstuff.com/2010/01/awk-introduction-tutorial-7-awk-print-examples/>

**CUT & PASTE command in Unix?**

**cut** - Cuts selected fields of each line of a file.

List following -f is a list of fields assumed to be separated by a delimiter character.

Character following -d is the delimiter.

Example –

$> cat infydata

1000:sundar:mysore

1001:eugene:chennai

1002:ajit:mysore

1004:hanumesh::::bangalore

1005:girish:mangalore

$> cut –d”:” –f2 infydata

sundar

eugene

ajit

hanumesh

girish

In the file “infydata”, the de-limiter (field separator) used is “:”.

The above example extracts the 2nd field (specified as –f2) from the file.

$> cut -d":" -f1,2 infydata

1000:sundar

1001:eugene

1002:ajit

1004:hanumesh

1005:girish

In the above example, the 1st and 2nd fields are extracted.

$> cut -c6 infydata

s

e

a

h

g

The option “–c” is used for specifying the column which needs to be extracted.

**paste** - Merging the corresponding lines of given files.This command allows horizontal pasting of files.

Example –

$> cat textdata1

Infosys Technologies Ltd

Mysore

India

$> cat textdata2

Infosys Technologies Ltd

Bangalore

India

$> paste textdata1 textdata2

Infosys Technologies Ltd Infosys Technologies Ltd

Mysore Bangalore

India India

In the above example, the contents of the two files are horizontally pasted.

**tr Command in Unix?**

tr - Translate characters

Translates character/characters.

tr [option...] string1 [string2]

Example –

$> cat infydata | tr "a" "b"

1000:sundbr:mysore

1001:eugene:chennbi

1002:bjit:mysore

1004:hbnumesh::::bbngblore

1005:girish:mbngblore

In the above example, all occurrences of “a” have been changed to “b”.

**Command to find out number of person connected to internet in Unix?**

netstat