**Shell Scripting Interview Questions And Answers:**

**Q: - What is Shell's Responsibilities ?**

The shell is responsible for the execution of all programs that you request from your terminal. Each time you type in a line to the shell, the shell analyzes the line and then determines what to do.The line that is typed to the shell is known more formally as the command line. The shell scans this command line and determines the name of the program to be executed and what arguments to pass to the program.

**Q: - What is "$#" Variable ?**  
  
The $# Variable  
Whenever you execute a shell program, the special shell variable $# gets set to the number of arguments that were typed on the command line.

**Q: - Explain "Exit Status" for a shell script ?**

Whenever any program completes execution under the Unix system, it returns an exit status back to the system. This status is a number that usually indicates whether the program successfully ran. By convention, an exit status of zero indicates that a program succeeded, and nonzero indicates that it failed. Failures can be caused by invalid arguments passed to the program, or by an error condition detected by the program. For example, the cp command returns a nonzero exit status if the copy fails for some reason (for example, if it can't create the destination file), or if the arguments aren't correctly specified (for example, wrong number of arguments, or more than two arguments and the last one isn't a directory). In the case of grep, an exit status of zero (success) is returned if it finds the specified pattern in at least one of the files; a nonzero value is returned if it can't find the pattern or if an error occurs (the arguments aren't correctly specified, or it can't open one of the files).

**Q: - What is "Command Substitution" ?**  
  
Command substitution is the process by which the shell runs a command and replaces the command substitution with the output of the executed command. That sounds like a mouthful, but it's pretty straightforward in practice.

**Q: - What is " eval" command ?**  
  
The eval command exists to supersede the normal command-line substitution and evaluation order, making it possible for a shell script to build up commands dynamically. This is a powerful facility, but it must be used carefully. Because the shell does so many different kinds of substitutions, it pays to understand the order in which the shell evaluates input lines.

**Q: - What is awk ?**  
  
An awk invocation can define variables, supply the program, and name the input files.

**Q: - What is "grep" programe ?**  
  
The grep program is the primary tool for extracting interesting lines of text from input datafiles. POSIX mandates a single version with different options to provide the behavior traditionally obtained from the three grep variants: grep, egrep, and fgrep.

**Q: - Name a new feature introduced with PHP 5.**  
  
PHP 5 introduces (among other things) SQLite support, improved XML support, and a significantly improved object model.

**Q: - explain "read" command ?**  
  
The read command is one of the most important ways to get information into a shell program:

$ x=abc ; printf "x is now '%s'. Enter new value: " $x ; read x

x is now 'abc'. Enter new value: PDQ

$ echo $x

PDQ

**Q: - What are the two files used by the shell to initialize itself?**

/etc/profile  
profile

**Q: - What is Interactive mode?**

Interactive mode means that the shell expects to read input from you and execute the commands that you specify. This mode is called interactive because the shell is interacting with a user. This is usually the mode of the shell that most users are familiar with: you log in, execute some commands, and log out. When you log out using the exit command, the shell exits.

**Q: - What is noninteractive mode?**

In this mode, the shell does not interact with you; instead it reads commands stored in a file and executes them. When it reaches the end of the file, the shell exits.

**Q: - what is local variable?**

A local variable is a variable that is present within the current instance of the shell. It is not available to programs that are started by the shell. The variables that you looked at previously have all been local variables.

**Q: - What is environment variable?**

An environment variable is a variable that is available to any child process of the shell. Some programs need environment variables in order to function correctly. Usually a shell script defines only those environment variables that are needed by the programs that it runs.

**Q: - What is shell variable?**

A shell variable is a special variable that is set by the shell and is required by the shell in order to function correctly. Some of these variables are environment variables whereas others are local variables.

**Q: - Explain the “Exit” command?**

Every program whether on UNIX or Linux should end at a certain point of time and successful completion of a program is denoted by the output 0. If the program gives an output other than 0 it defines that there has been some problem with the execution or termination of the problem. Whenever you are calling other function, exit command gets displayed.

**Q: - How do you find out what’s your shell?**

echo $SHELL

**Q: - How you will run a process in the background?**

./ProcessName &

**Q: - How do you write a while loop in shell?**

Use While Loop

**Q: - How do you read keyboard input in shell scripts?**

Use read command

**Q: - What is GUI Scripting?**

Graphical user interface provided the much needed thrust for controlling a computer and its applications. This form of language simplified repetitive actions. Support for different applications mostly depends upon the operating system. These interact with menus, buttons, etc.

**Q: - Explain the term “loops”?**

Loops enable you to execute a series of commands multiple times. Two main types of loops are the while and for loops.

**Q: - What is “Nested Loops”?**

When a loop is located inside the body of another loop it is said to be nested within another loop.

**Q: - What is “Infinite Loops”?**

Loops that execute forever without terminating.

**Q: - What is “File Descriptor”?**

An integer that is associated with a file. Enables you to read and write from a file using the integer instead of the file's name.

**Q: - Explain “STDIN”?**

STDINStandard Input. User input is read from STDIN. The file descriptor for STDIN is 0.

**Q: - Explain “STDOUT”?**

STDOUTStandard Output. The output of scripts is usually to STDOUT. The file descriptor for STDOUT is 1.

**Q: - Explain “STDERR”?**

STDERRStandard Error. A special type of output used for error messages. The file descriptor for STDERR is 2.

**Q: - Explain “Escape Sequence”?**

An escape sequence is special sequence of characters that represents another  
character.

**Q: - Explain “Output Redirection” in shell scripting?**

In UNIX or Linux, the process of capturing the output of a command and storing it in a file is called *output redirection* because it redirects the output of a command into a file instead of the screen.

**Q: Explain** “**Input Redirection” in shell scripting?**

In UNIX or Linux the process of sending input to a command from a file is called input redirection.

**Q: - What is** “**Field separator”?**

The field separator controls the manner in which an input line is broken into fields. In the shell, the field separator is stored in the variable IFS. In awk, the field separator is stored in the awk variable FS.

**Q: - What is** “**Library”?**

A file that contains only functions is called a library. Usually libraries contain no main code.

**1: What is a shell?**

Shell is a interface between user and the kernel. Even though there can be only one kernel ; a system can have many shell running simultaneously . Whenever a user enters a command through keyboard the shell communicates with the kernel to execute it and then display the output to the user.

**2: What are the different types of commonly used shells on a typical linux system?**

csh,ksh,bash,Bourne . The most commonly used and advanced shell used today is “Bash” .

**3:What is the equivalent of a file shortcut that we have on window on a Linux system?**

Shortcuts are created using “links” on Linux. There are two types of links that can be used namely “soft link” and “hard link”

**4:What is the difference between soft and hard links?**

Shell Scripting

Soft links are link to the file name and can reside on different filesytem as well; however hard links are link to the inode of the file and has to be on the same filesytem as that of the file. Deleting the orginal file makes the soft link inactive (broken link) but does not affect the hard link (Hard link will still access a copy of the file)

**5: How will you pass and access arguments to a script in Linux?**

Arguments can be passed as:  
scriptName “Arg1” “Arg2”….”Argn” and can be accessed inside the script as $1 , $2 .. $n

**6: What is the significance of $#?**

$# shows the count of the arguments passed to the script.

**7: What is the difference between $\* and $@?**

$@ treats each quoted arguments as separate arguments but $\* will consider the entire set of positional parameters as a single string.

**8: Use sed command to replace the content of the file (emulate tac command)**  
Eg:

if cat file1

ABCD

EFGH

Then O/p should be

EFGH

ABCD

sed '1! G; h;$!d' file1

Here G command appends to the pattern space,

h command copies pattern buffer to hold buffer

and d command deletes the current pattern space.

**9: Given a file, replace all occurrence of word “ABC” with “DEF” from 5th line till end in only those lines that contains word “MNO”**

sed –n ‘5,$p’ file1|sed ‘/MNO/s/ABC/DEF/’

**10: Given a file , write a command sequence to find the count of each word.**

tr –s “(backslash)040” <file1|tr –s “(backslash)011”|tr “(backslash)040 (backslash)011” “(backslash)012” |uniq –c

where “(backslash)040” is octal equivalent of “space”

”(backslash)011” is octal equivalent of “tab character” and

“(backslash)012” is octal equivalent of newline character.

**11: How will you find the 99th line of a file using only tail and head command?**

tail +99 file1|head -1

**12: Print the 10th line without using tail and head command.**

sed –n ‘10p’ file1

sed -n '10p' /etc/passwd

**13:In my bash shell I want my prompt to be of format ‘$”Present working directory”:”hostname”> and load a file containing a list of user defined functions as soon as I login , how will you automate this?**

In bash shell we can create “.profile” file which automatically gets invoked as soon as I login and write the following syntax into it.

export PS1=’$ `pwd`:`hostname`>’ .File1

Here File1 is the file containing the user defined functions and “.” invokes this file in current shell.

**14: Explain about “s” permission bit in a file?**

“s” bit is called “set user id” (SUID) bit.

“s” bit on a file causes the process to have the privileges of the owner of the file during the instance of the program.

Eg: Executing “passwd” command to change current password causes the user to writes its new password to shadow file even though it has “root” as its owner.

**15: I want to create a directory such that anyone in the group can create a file and access any person’s file in it but none should be able to delete a file other than the one created by himself.**

We can create the directory giving read and execute access to everyone in the group and setting its sticky bit “t” on as follows:

mkdir direc1

chmod g+wx direc1

chmod +t direc1

**16: How can you find out how long the system has been running?**

Command “uptime”

**17: How can any user find out all information about a specific user like his default shell, real life name, default directory,when and how long he has been using the sytem?**

finger “loginName” …where loginName is the login name of the  
user whose information is expected.

**18: What is the difference between $$ and $!?**

$$ gives the process id of the currently executing process whereas $! shows the process id of the process that recently went into background.

**19: What are zombie processes?**

These are the processes which have died but whose exit status is still not picked by the parent process. These processes even if not functional still have its process id entry in the process table.

**20: How will you copy file from one machine to other?**

We can use utilities like “ftp” ,”scp” or “rsync” to copy file from one machine to other.

Eg: Using ftp:

ftp hostname  
> put file1  
> bye

Above copies file file1 from local system to destination system whose hostname is specified.

**21: I want to monitor a continuously updating log file, what command can be used to most efficiently achieve this?**

We can use tail –f filename . This will cause only the default last 10 lines to be displayed on std o/p which continuously shows the updating part of the file.

**22: I want to connect to a remote server and execute some commands, how can I achieve this?**

We can use telnet to do this:

telnet hostname –l user  
> Enter password  
> Write the command to execute  
> quit

**23: I have 2 files and I want to print the records which are common to both.**

We can use “comm” command as follows:

comm -12 file1 file2 … 12 will suppress the content which are  
unique to 1st and 2nd file respectively.

**24: Write a script to print the first 10 elemenst of Fibonacci series.**

#!/bin/sh

a=1

b=1

echo $a

echo $b

for I in 1 2 3 4 5 6 7 8

do

c=a

b=$a

b=$(($a+$c))

echo $b

done

**25: How will you connect to a database server from linux?**

We can use isql utility that comes with open client driver as follows:  
isql –S serverName –U username –P password

**26: What are the 3 standard streams in Linux?**

Output stream , re presented as 0 , Input stream, represented as 1 and Error stream represented as 2.

**27: I want to read all input to the command from file1 direct all output to file2 and error to file 3, how can I achieve this?**

command <file1 0>file2 2>file3

**28: What will happen to my current process when I execute a command using exec?**

“exec” overlays the newly forked process on the current process ; so when I execute the command using exec a new process corresponding to the command will be created and the current process will die.  
Eg: Executing **“exec com1”** on command prompt will execute com1 and return to login prompt since my logged in shell is superimposed with the new process of the command .

**29: How will you emulate wc –l using awk?**

awk ‘END {print NR} fileName’

**30: Given a file find the count of lines containing word “ABC”.**

grep –c “ABC” file1

**31: What is the difference between grep and egrep?**

egrep is Extended grep that supports added grep features like “+” (1 or more occurrence of previous character),”?”(0 or 1 occurrence of previous character) and “|” (alternate matching)

**32: How will you print the login names of all users on a system?**

/etc/shadow file has all the users listed.

awk –F ‘:’ ‘{print $1} /etc/shadow’|uniq -u

**33: How to set an array in Linux?**

Syntax in ksh:  
Set –A arrayname= (element1 element2 ….. element)  
In bash  
A=(element1 element2 element3 …. elementn)

**34: Write down the syntax of “for “ loop**

Syntax:

for iterator in (elements)  
do  
execute commands  
done

**35:How will you find the total disk space used by a specific user?**

du -s /home/user1 ….where user1 is the user for whom the total disk  
space needs to be found.

**36: Write the syntax for “if” conditionals in linux?**

Syntax

If condition is successful  
then  
execute commands  
else  
execute commands  
fi

**37:What is the significance of $? ?**

$? gives the exit status of the last command that was executed.

**38: How do we delete all blank lines in a file?**

sed ‘^ [(backslash)011(backslash)040]\*$/d’ file1

where (backslash)011 is octal equivalent of space and

(backslash)040 is octal equivalent of tab

**39: How will I insert a line “ABCDEF” at every 100th line of a file?**

sed ‘100i\ABCDEF’ file1

**40: Write a command sequence to find all the files modified in less than 2 days and print the record count of each.**

find . –mtime -2 –exec wc –l {} \;

**41: How can I set the default rwx permission to all users on every file which is created in the current shell?**

We can use:

umask 777

This will set default rwx permission for every file which is created to every user.

**42: How can we find the process name from its process id?**

We can use **“ps –p ProcessId”**

**43: What are the four fundamental components of every file system on linux?**

bootblock, super block, inode block and datablock

**44: What is a boot block?**

This block contains a small program called “Master Boot record”(MBR) which loads the kernel during system boot up.

**45: What is a super block?**

Super block contains all the information about the file system like size of file system, block size used by it,number of free data blocks and list of free inodes and data blocks.

**46: What is an inode block?**

This block contains the inode for every file of the file system along with all the file attributes except its name.

**47: How can I send a mail with a compressed file as an attachment?**

*zip file1.zip file1|mailx –s “subject” Recepients email id*  
*Email content*  
*EOF*

**48: How do we create command aliases in shell?**

alias Aliasname=”Command whose alias is to be created”

**49: What are “c” and “b” permission fields of a file?**

“c “ and “b” permission fields are generally associated with a device file. It specifies whether a file is a character special file or a block special file.

**50: What is the use of a shebang line?**

Shebang line at top of each script determines the location of the engine which is to be used in order to execute the script.

Q.What is shell scripting?  
A.Shell scripting is used to program command line of an operating system. Shell Scripting is also used to program the shell which is the base for any operating system. Shell scripts often refer to programming UNIX. Shell scripting is mostly used to program operating systems of windows, UNIX, Apple, etc. Also this script is used by companies to develop their own operating system with their own features.  
  
Q.State the advantages of Shell scripting?  
A.There are many advantages of shell scripting some of them are, one can develop their own operating system with relevant features best suited to their organization than to rely on costly operating systems. Software applications can be designed according to their platform.  
  
Q.What are the disadvantages of shell scripting?  
A.There are many disadvantages of shell scripting they are  
1.Design flaws can destroy the entire process and could prove a costly error.  
2.Typing errors during the creation can delete the entire data as well as partition data.  
3.Initially process is slow but can be improved.  
4.Portbility between different operating system is a prime concern as it is very difficult to port scripts etc.  
  
Q.Explain about the slow execution speed of shells?  
A.Major disadvantage of using shell scripting is slow execution of the scripts. This is because for every command a new process needs to be started. This slow down can be resolved by using pipeline and filter commands. A complex script takes much longer time than a normal script.  
  
Q.Give some situations where typing error can destroy a program?  
A.There are many situations where typing errors can prove to be a real costly effort. For example a single extra space can convert the functionality of the program from deleting the sub directories to files deletion. cp, cn, cd all resemble the same but their actual functioning is different. Misdirected > can delete your data.  
  
Q.Explain about return code?  
A.Return code is a common feature in shell programming. These return codes indicate whether a particular program or application has succeeded or failed during its process. && can be used in return code to indicate which application needs to be executed first.  
  
Q.What are the different variables present in Linux shell?  
A.Variables can be defined by the programmer or developer they specify the location of a particular variable in the memory. There are two types of shells they are System variables and user defined variables. System variables are defined by the system and user defined variables are to be defined by the user (small letters).  
  
Q.Explain about GUI scripting?  
A.Graphical user interface provided the much needed thrust for controlling a computer and its applications. This form of language simplified repetitive actions. Support for different applications mostly depends upon the operating system. These interact with menus, buttons, etc.  
  
Q.Explain about echo command?  
A.Echo command is used to display the value of a variable. There are many different options give different outputs such as usage \c suppress a trailing line, \r returns a carriage line, -e enables interpretation, \r returns the carriage.  
  
  
Q.How do you stop a process?  
- kill pid  
  
Q.How do you find out about all running processes?  
A.- ps -ag  
  
Q.How do you stop all the processes, except the shell window?  
A.- kill 0  
  
Q.How do you fire a process in the background?  
A.- ./process-name &  
  
Q.How do you refer to the arguments passed to a shell script?  
A.- $1, $2 and so on. $0 is your script name.  
  
Q.What’s the conditional statement in shell scripting?  
A.- if {condition} then … fi  
  
Q.How do you do number comparison in shell scripts?  
A.- -eq, -ne, -lt, -le, -gt, -ge  
  
Q.How do you test for file properties in shell scripts?  
A.- -s filename tells you if the file is not empty, -f filename tells you whether the argument is a file, and not a directory, -d filename tests if the argument is a directory, and not a file, -w filename tests for writeability, -r filename tests for readability, -x filename tests for executability.  
  
Q.How do you do Boolean logic operators in shell scripting?  
A.- ! tests for logical not, -a tests for logical and, and -o tests for logical or.  
  
Q.How do you find out the number of arguments passed to the shell script?  
A.- $#  
  
Q.What’s a way to do multilevel if-else’s in shell scripting?  
A.- if {condition} then {statement} elif {condition} {statement} fi  
  
Q.How do you write a for loop in shell?  
A.- for {variable name} in {list} do {statement} done  
  
Q.How do you write a while loop in shell?  
A.- while {condition} do {statement} done  
  
Q.How does a case statement look in shell scripts?  
A.- case {variable} in {possible-value-1}) {statement};; {possible-value-2}) {statement};; esac  
  
Q.How do you read keyboard input in shell scripts?  
A.- read {variable-name}  
  
Q.How do you define a function in a shell script?  
A.- function-name() { #some code here return }  
  
Q.How does getopts command work?  
A.- The parameters to your script can be passed as -n 15 -x 20. Inside the script, you can iterate through the getopts array as while getopts n:x option, and the variable $option contains the value of the entered option.  
  
Q.What is $\*?  
A.Its mainly used for showing up all params. This show all parameter values passed in shell script  
  
Q.What does $# stand for?  
A.# will return the number of parameters that are passed as the command-line arguments.  
  
Q.What does $? Return?  
A.$? will return exit status of command .0 if command gets successfully executed ,non-zero if command failed.  
  
Q.What are Different types of shells?  
A.  
1.sh : the oldest shell   
2.csh : C shell   
3.ksh : Korn Shell   
4.bash : bourne again shell   
  
Q.How do you read arguments in a shell program – $1, $2?  
A.Shell script accepts parameters in following format…   
1.$1 would be the first command line argument, $2 the second, and so on   
2.$0 is the name of the script or function  
If your script has more than 9 params then accept in following way…   
${12} : 12th param   
${18} : 18th param.  
  
Q.If you have a string “one two three”, which shell command would you use to extract the strings?  
A.  
1.echo $string | cut -d” ” -f1   
2.echo $string | cut -d” ” -f2   
3.echo $string | cut -d” ” -f3

Q.Explain about @@@ lines?  
A.@@@ Lines are counted (but not printed), and as long as the count is not greater than the desired version, the editing commands are passed through. Two ed commands are added after those from the history file: $d deletes the single @@@ line that sed left on the current version.

Q.Explain about vis?  
A.Vis that copied its standard input to its standard output, except that it makes all non printing characters visible by printing them as \nnn, where nnn is the octal value of the character. Vis is invaluable for detecting strange or unwanted characters that may have crept into files.

Q.Is the function call to exit at the end of vis necessary?  
A.The call to exit at the end of vis is not necessary to make the program work properly, but it ensures that any caller of the program will see a normal exit status from the program when it completes. An alternate way to return status is to leave main with return 0; the return value from main is the program`s exit status.

Q.Explain about fgets?  
A.Fgets (buf, size, fp) fetches the next line of input from fp, up to and including a newline, into buf, and adds a terminating \0; at most size-1 characters are copied. A Null value is returned at the end of the file.

Q.Explain about efopen page?  
A.The routine efopen encapsulates a vey common operation: try to open a file; if it`s not possible, print an error message and exit. To encourage error messages that identify the offending program, efopen refers to an external string program containing the name of the program, which is set in main.

Q.Explain about yacc parser generator?  
A.Yacc is a parser generator that is a program for converting a grammatical specification of a language like the one above into a parser that will parse statements in the language.

Q.State and explain about features of UNIX?  
A.UNIX operating system originally was developed in 1969. This is an open source operating system developed by AT&T. It is widely used in work stations and servers. It is designed to be multi tasking, multi user and portable. UNIX has many several components packed together.

Q.Explain about sh?  
A.Sh is the command line interpreter and it is the primary user interface. This forms the programmable command line interpreter. After windows appeared it still retained the programmable characteristics.

Q.Explain about system and user utilities?  
A.There are two utilities they are system and user utilities. System utilities contain administrative tools such as mkfs, fsck, etc. Where as user utilities contain features such as passwd, kill, etc. It basically contains environment values.

Q.Explain about document formatting?  
A.UNIX systems were primarily used for typesetting systems and document formatting. Modern UNIX systems used packages such as Tex and Ghostscript. It uses some of the programs such as nroff, tbl, troff, refer, eqn and pic. Document formatting is very used because it forms the base of UNIX.

Q.Explain about communication features in UNIX?  
A.Early UNIX systems used inter user communication programs mail and write commands. They never contained a fully embedded inter user communication features. Systems with BSD included TCP/IP protocols.

Q.Explain about chmod options filename?  
A.This command allows you to change, write, read and execute permissions on your file. Changes can be done to the file system but at times you need to change permissions for the file systems. At times files should be executable for viewing the files.

Q.Explain about gzip filename?  
A.Gzip filename is used to compress the files so that those files take up less space. The size of the file actually gets reduced to half their size but they might also depend upon about the file size and nature of the file systems. Files using gzip file name end with .gz.

Q.Explain about refer?  
A.Refer was written in Bell Laboratories and it is implemented as a troff preprocessor. This program is used managing bibliographic references and it is used to cite them in troff documents. It is offered in most of the UNIX packages. It refers with text and reference file.

Q.Explain about lpr filename?  
A.This command is used to print a file. If you want to change the default print you can change the printer by using the P option. For double sided print you can use lpr-Pvalkyr-d. This is very useful command in UNIX present in many packages.

Q.Explain about lprm job number?  
A.This command is used to remove documents from the printer queue. The job number or the queue number can be found by using lpq. Printer name should be specified but this is not necessary if you want to use your default printer.

Q.Brief about the command ff?  
A.This command finds files present anywhere on the system. This command is used to find document location where you forgot the directory in which you kept the file but you do remember about the name. This command is not restricted in finding files it displays files and documents relevant to the name.

Q.Brief about finger username?  
A.This command is used to give information about the user; it gives out a profile about the user. This command is very useful for administrators as it gives the log information, email, current log information, etc. finger also displays information such as phone number and name when they use a file called .plan.

Q.Explain about the command elm?  
A.This command lets you to send email message from your system. This command is not the only one which sends email there are lots of other messenger systems which can facilitate the process of sending a mail. This command behaves differently on different machines.

Q.Brief about the command kill PID?  
A.This command ends the process to which it was assigned (ID). This command cannot be used in multi systems in the network. ID can be obtained by the command ps. This command ignores completely the state at which the process is it kills the process.

Q.Explain about the command lynx?  
A.This command helps you to browse web from an ordinary terminal. Text can be seen but not the pictures. URL can be assigned as an argument to the G command. Help section can be obtained by pressing H and Q makes the program to quit.

Q.Brief about the command nn?  
A.This command allows you to read the news. First you can read about the local news and then the remote news. "nnl” command makes or allows you to read local news and nnr command is used to read remote news. Manual and help information is available with many popular packages.

Q.Brief about ftp hostname?  
A.This command lets you download information, documents, etc from a remote ftp. First it is important to configure an FTP for the process to begin. Some of the important commands relevant to the usage of FTP are as follows get, put, mget, mput, etc. If you are planning to transfer files other than ASCII defined it is imperative to use binary mode.

Q. Explain about the shell variable IFS?  
A.The shell variable IFS (internal field separator) is a string of characters that separate words in argument lists such as back quotes and for statements. Normally IFS contains a blank, a tab, and a new line, but we can change it to anything useful, such as just a newline.

Q.Explain about the rules used in overwrite to preserve the arguments to the users command?  
A.Some of the rules are  
1.$\* and $@ expand into the arguments and are rescanned; blanks in arguments will result in multiple arguments.  
2.“$\*” is a single word composed of all the arguments to the shell file joined together with spaces.  
3.“$@” is identical to the arguments received by the shell file: blanks in arguments are ignored and the result is a list of words identical to the original arguments.

Q.Explain about the command overwrite?  
A.Overwrite is committed to changing the original file. If the program providing input to overwrite gets an error, its output will be empty and overwrite will dutifully and reliably destroy the argument file. Overwrite could ask for conformation before replacing the file, but making overwrite interactive would negate its efficiency. Overwrite could check that its input is empty.

Q.Explain about kill command?  
A.The kill command only terminates processes specified by process-id when a specific background process needs to be killed, you must usually run ps to find the process-id and then re type it as an argument to kill. Killing process is dangerous and care must be taken to kill the right processes.

Q.What exactly is UNIX?  
A.UNIX is written in C and it is portable. It runs on a range of computers from microprocessors to the largest mainframes. The source code is available and written in high level language which makes it easy to adapt for a particular requirement.

Q.Explain about the return key?  
A.Return Key signifies the end of a line of input; it must be presses before the system will interpret the characters you have types. It serves as an example of control character return key can be typed by holding the control key and typing M.

Q.Explain about DELETE and BREAK?  
A.These keys have significant meaning sometimes DELETE is called as RUBOUT key. Break is sometimes called INTERRUPT. In most SYSTEMS the delete key stops a program immediately, without waiting for it to finish on some systems, Ctrl-C provides this service. Break is a synonym for DELETE or CTRL-C.

Q.Explain about TYPE-ahead?  
A.The kernel reads what you type as you type it, even if it’s busy with something else, so you can type as fast as you want, whenever you want, even when some command is printing at you. Your I/P characters will appear intermixed with the O/P characters but they will be stored away and interpreted in the correct order.

Q.How to get a computer aided instruction?  
A.Your system may have a command called learn which provides computer aided instruction on the file system and basic commands, the editor, document preparation and even “C” programming. IF $learn exists on your system, it will teach you what to do from there.

Q.Explain about cat?  
A.Cat is the simplest of all the printing commands. Cat prints the contents of all the files named by its arguments. The named file or files are catenated the terminal one after another with nothing between. You have to be quick with CTRL-S to stop O/P from cat before it flows off your screen.

Q.How to search files for lines that match a pattern?  
A.Grep command searches for files for lines which match a pattern.  
E.g. g/regular expression/p.grep will also look for lines that don’t match the pattern, when the option-V is sued. grep can be used to search several files in that case it will prefix the filename.

Q.Describe about the root file system?  
A.The root file system has to be present for the system to execute /bin, /dev, and /etc are always kept on the root system because when the system starts only files in the root system are accessible and some files such as /bin/sh are needed to run at all. During the boot strap operation, all the file systems are checked for self consistency and attached to the root systems.

Q.Explain about ZAP?  
A.ZAP, which selectively kills processes, is another program. The main problem with that version is speed; it creates so many processes that it runs slowly, which is especially undesirable for a program that kills errant processes rewriting ZAP in C will make it faster.

Q.Suppose that you are using a terminal in which the screen size is bigger than the normal 24 lines. If you want to use p and take full advantage of your terminal capabilities what choices are open to you?  
A.You have to specify the screen size each time you use P. $P-36.  
Also you could put a shell file in your bin.  
$cat/usr/you/bin/p  
Exec/usr/bin/p-36$\*  
$  
Another solution is to modify p to use an environment variable that defines the properties of your terminal.

Q.Explain abut low-level I/O?  
A.The lowest of I/O is a direct entry into the O.S. your program reads or writes files in chunks of any convenient size. The kernel buffers your data into chunks that match the peripheral devices and schedule operations on the devices to optimize their performance over all users.

Q.What are the special arrangements to make a terminal i/p and o/p?  
A.When it is started by the shell, a program inherits three open files, with file descriptor 0, 1 and 2 called the standard i/p, the standard o/p, and the standard error. If the program reads “0” and writes descriptors 1 and 2, it can do I/O without opening files.

Q.Explain about read slow?  
A.One can call to read return 0 which signals the end of file life, if data is written on that file then a subsequent read will be able to find more bytes available. This observation is the basis of a program called readslow which continues to read its input, regardless of whether it got an end of file or not. Readslow is handy for watching the progress of the file.

Q.What function does “errno” do?  
A.Sometimes it is nice to know what specific error occurred; for this purpose all system calls, when appropriate, leave an error number in an external integer called calls, when appropriate, leave an error number in an external integer called errno. By using errno, your program can, for example, determine whether an attempt to open a file failed because it did not exist or because you lacked permission to read it.

Q.Describe the process of “spname”?  
A.The operation of spname is straightforward enough, although there are a lot of boundary conditions to get right. Suppose the file name is /d1/d2/f. The basic ideas is to peel off the first component (/), then search that directory for a name close to the nest component (dl), then search that directory for something near d2, and so on, until a match has been found for each component. If at any stage there isn’t a plausible candidate in the directory, the search is abandoned.

Q.Explain about fork?  
A.Splitting is done by a system call named fork. Proc\_id = fork (); splits the program into two copies, both of which continue to run. The only difference between the two is the value returned by fork, the process-id. Two copies of the program are made by the fork. In the child, the value returned by fork is zero, so it calls execlp, which does the command line and then dies. In the parent, fork returns non-zero so it skips the execlp.

Q.What is the easiest way to store variables and explain?  
A.The easiest way to store the values of the variables is in a 26-element array; the single-letter variable name can be used to index the array. But if the grammar is to process both variable names and values in the same stack, yacc has to be told that its stack contains a union of a double and an int, not just a double.

Q.Explain about the case statement.  
A.The case statement compares word to the patterns from top to bottom, and performs the commands associated with the first, and only the first, pattern that matches. The patterns are written using the shells pattern matching rules, slightly generalized.

Q..Explain the basic forms of each loop?  
A.There are three loops; for, while and until. For loop is by far the most commonly used form of loop. Basically like other programs it executes a given set of commands and instructions. While and until forms of loop use the exit status from a command based system. They control the execution of the commands in the body of the loop.

Q.Describe about awk and sed?  
A.The awk program processes this to report the changes in an easier to understand format. Sed output is always behind its input by one line; there is always a line of input that has been processed but not printed, and this would introduce an unwanted delay.

Q.Explain about signal argument?  
A.The sequence of commands is a single argument, so it must almost always be quoted. The signal numbers are small integers that identify the signal. For example, 2 is the signal generated by pressing the DEL key, and 1 is generated by hanging up the phone. Unless a program has taken explicit action to deal with signals, the signal will terminate it.

Q.Explain about exec?  
A.The exec is just for efficiency, the command would run just as well without it. Exec is a shell built-in that replaces the process running this shell by the named program, thereby saving one process- the shell that would normally wait for the program to complete. Exec could be used at the end of the enhanced cal program when it invokes /usr/bin/cal.

Q.Explain about trap command  
A.The trap command sequence must explicitly invoke exit, or the shell program will continue to execute after the interrupt. The command sequence will be read twice: once when the trap is set and once when it is invoked. Trap is used sometimes interactively, most often to prevent a program from being killed by the hangup signal.

Q.Explain about sort command?  
A.The sort command has an option –o to overwrite a file:  
$ sort file1 -0 file2  
Is equivalent to  
$ sort file1 > file2  
If file 1 and file 2 are the same file, redirection with > will truncate the input file before it is sorted. The –o option works correctly because the input is sorted and saved in a temporary file before the output file is created. Many other commands could also use a –o option.

Q.What are the Unix system calls for I/O?  
A.The following are the UNIX system calls for I/O:  
1.Open: to open a file. Syntax: open (pathname, flag, and mode).  
2.Create: To create a file. Syntax: create (pathname, mode).  
3.Close: To close a file. Syntax: close (filedes).  
4.Read: To read data from a file that is opened. Syntax: read (filedes, buffer, bytes)  
5.Write: To write data to a file that is opened. Syntax: write (filedes, buffer, bytes)  
6.Lseek: To position the file pointer at given location in the file.Syntax: lseek (filedes, offset, from).  
7.Dup: To make a duplicate copy of an existing file descriptor. Syntax: dup (filedes).  
8.Fcntl: To make the changes to the properties of an open file.Syntax: fcntl (filedes, cmd, arg).

Q.How are devices represented in UNIX?  
A.Devices in UNIX are represented by files. These are special files located in the )dev directory. Hence in UNIX, every piece of hardware is a file. This device file allows us to access the hardware.UNIX represents all devices as files. These files are located in the directory /dev. That is why the devices and other files are accessed in a similar way.Devices file which is specified as ‘block special file’ with some similar characters of a disk file. A device which is specified as a ‘character special file’ with some characteristics that is similar to a keyboard.For instance, the following command;  
Less –f /dev/hda is not a file in the ‘real’ sense. When read, it is actually reading directly from the first physical hard disk of your machine.

Q.Brief about the directory representation in UNIX  
A.A directory in UNIX is a special file that the kernel maintains. Only the kernel modifies the directory while the processes read them. A directory in UNIX is a association between file names and I node and hence the directory content is nothing but a list of filename and inode number pairs. When a directory is created by the kernel, two entries are made; '.' referring to the directory itself) and '..' referring to parent directory.A UNIX directory is representing a file that consists of a correspondence between file names and inodes. UNIX treats the directory as a special file. This file is maintained by the kernel. Processes can read the directories and only kernel modifies these directories. At the time of a directory creation, kernel makes two entries namely directory itself and parent directory.

Q.Discuss the mount and unmount system calls  
A.Mount system call makes a directory accessible by attaching a root directory of one file system to another directory. In UNIX directories are represented by a tree structure, and hence mounting would mean attaching them to the branches. This means the file system found on one device can be attached to the tree.The location in the system where the file is attached is called as a mount point.Example:-  
Mount –t type device dir  
This will attach or mount the file system found on device of type type to the directory dir.

Unmount system calls does the opposite. It un mounts or detaches the attached file front the target or mount point. If a file is opened or used by some process cannot be detached.The attaching of a file system to another file system is done by using mount system call. At the time of mounting, there is an essential splicing one directory tree onto a branch in another directory tree is done. The mount takes two arguments. One – the mount point, which is a directory in the current file naming system, two – the file system to mount to that point. At the time of inserting CDROM into the system, the corresponding CDROM file system will automatically mounts to the directory - /dev/cdrom in the system.The unmount system call is used to detach a file system.

Q.What are the process states in Unix?  
A.Process states in Linux:

1.Running: Process is either running or ready to run  
2.Interruptible: a Blocked state of a process and waiting for an event or signal from another process  
3.Uninterruptible:- a blocked state. Process waits for a hardware condition and cannot handle any signal  
4.Stopped: Process is stopped or halted and can be restarted by some other process  
5.Zombie: process terminated, but information is still there in the process table.

Q.What is use of sed command?  
A.Sed command in UNIX is commonly used for processing of files. Sed stands for Stream Editor which parses text files and used for making textual transformations to a file. The command specified to Sed, is applied on the file line by line.Sed command reads from a standard input and places it into the pattern space. It performs various editing commands on that pattern space in a sequential manner. Later the pattern space is written onto the STDOUT.  
Example:  
To replace all matching occurrences of some text to another.sed -e 's/olddata/newdata/g' inputFileName > outputFileName.  
Here, g is global which replaces ALL occurrences.

Q.What is 'inode'?  
A.Each file in UNIX has a unique number called as an inode. Using this number the file information like user, group, ownership and access mode information can be found. A files inode number can be found using the following command:Ls –i   
If the inode number is known, the following command can be used to get details of the file:Ls –l   
A file in UNIX is given a unique number. This unique number is known as ‘inode’. Every file and directory (treated as a file by UNIX) has an inode value.

Q.What are links and symbolic links in UNIX file system?  
A.A link is a pointer or reference to another file. A directory in UNIX has a list of file names and their corresponding inodes. A directory entry can have an Inode pointing to another file.This is a hard link. When a hard link is made, then the i-numbers of two different directory file entries point to the same inode.A symbolic link or a soft link is a special type of file containing links or references to another file or directory in the form of a path. The path may be relative or absolute.Link is a utility program in UNIX which establishes a hard link from one directory to another directory. A hard link is a reference to a directory or to file on storage media. A symbolic link is a type of file. It contains references to another file directory in the form of absolute or a relative path. To create a symbolic link, following command is used:  
Ln –s target link\_name  
Here, target is the path and link\_name is the name of the link. Symbolic links can be created to create a file system based on different views of the user.

Q.Explain fork() system call.  
A.A fork() system call in UNIX is used to create processes. It returns a process ID. The process created becomes the child process of the caller. After the child process is created both parent and child will execute the next instruction. Depending on the return values child process can be determined.fork() system call is used to create processes. It returns a process id. After calling fork() system call, it becomes a child process of the caller. After creation of child call, the instruction followed by fork() will be executed after a new child process is created.Fork() returns a negative value, if the child process creation is unsuccessful, zero if the new child process is created, returns a positive value which is the ‘process id’, to the parent.

Q.What is a zombie?  
A.Zombie is a process state when the child dies before the parent process. In this case the structural information of the process is still in the process table. Since this process is not alive, it cannot react to signals. Zombie state can finish when the parent dies. All resources of the zombie state process are cleared by the kernel.A zombie is a process which is completed the execution and still available in the process table. At the end of any process, all the memory resources pertaining to that process will be deallocated and can be used by other processes. The entry of the process will remain available.   
The exit status of a child can be read by the parent by using the wait system call, the stage at which the zombie is removed. Once the zombie is removed, then the process id in the process table be removed and can be reused.  
  
Q.How do you create special files like named pipes and device files?  
A.Using the mknod command, special files can be created.   
Mknod [options] name type[major minor]  
Here, if argument is “p” a named pipe of FIFO file is created.  
If the argument is “b”, a block file is created. Here, the major or minor device numbers needs to be specified.  
If the argument is “c or u”, a character file is created. Here, the major or minor device numbers needs to be specified.

Special files are created by the system call ‘mknod’. Upon using the following sequence of steps, a special file will be created.

1.new inode is assigned by kernel   
2.sets the file type as a pipe, directory or special file   
3.Major and minor device numbers are the two entries created if the file type is a device file.   
For example, for a disk, disk controller is the major device number and the disk is the minor device is the disk.  
Unix example: $ mknod p

Q.How do I use poll()?  
A.Poll() in UNIX is used for to wait for some event on a file descriptor. When poll() is used, the user must allocate an array of pollfd structures, and pass the number of entries in this array.poll() allows an event to wait on a file descriptor. A pointer is accepted by poll() to a list of ‘struct pollfd’. A bitwise mask is used to specify the events in the field of events of the structure. The structure’s instance will be filled at a later point of time and returns with events that were occurred.

int poll(struct pollfd \*ufds, unsigned int nfds, int timeout)

The parameter fd is a file descriptor to open the file while timeout is in milliseconds. If the timeout value is negative, it means negative timeout. The structure if the is as below:

struct pollfd {  
int fd; /\* file descriptor \*/  
short events; /\* requested events \*/  
short revents; /\* returned events \*/  
};

The struct pollfd has the following format

struct pollfd {  
int fd; /\* The descriptor. \*/  
short events; /\* The event(s) is/are specified here. \*/  
short revents; /\* Events found are returned here. \*/  
};

Q.How are devices represented in UNIX?   
A.All devices are represented by files called special files that are located in/dev directory. Thus, device files and other files are named and accessed in the same way. A 'regular file' is just an ordinary data file in the disk. A 'block special file' represents a device with characteristics similar to a disk (data transfer in terms of blocks). A 'character special file' represents a device with characteristics similar to a keyboard (data transfer is by stream of bits in sequential order).  
  
Q.What is a zombie?   
A.When a program forks and the child finishes before the parent, the kernel still keeps some of its information about the child in case the parent might need it - for example, the parent may need to check the child's exit status. To be able to get this information, the parent calls `wait()'; In the interval between the child terminating and the parent calling `wait()', the child is said to be a `zombie' (If you do `ps', the child will have a `Z' in its status field to indicate this.)

Q.What are the process states in Unix?   
A.As a process executes it changes state according to its circumstances. Unix processes have the following states: Running : The process is either running or it is ready to run . Waiting : The process is waiting for an event or for a resource. Stopped : The process has been stopped, usually by receiving a signal. Zombie : The process is dead but have not been removed from the process table  
  
Q.What is a Daemon?   
A.A daemon is a process that detaches itself from the terminal and runs, disconnected, in the background, waiting for requests and responding to them. It can also be defined as the background process that does not belong to a terminal session. Many system functions are commonly performed by daemons, including the sendmail daemon, which handles mail, and the NNTP daemon, which handles USENET news. Many other daemons may exist. Some of the most common daemons are: init: Takes over the basic running of the system when the kernel has finished the boot process. inetd: Responsible for starting network services that do not have their own stand-alone daemons. For example, inetd usually takes care of incoming rlogin, telnet, and ftp connections. cron: Responsible for running repetitive tasks on a regular schedule.

Q.What is 'ps' command for?   
A.The ps command prints the process status for some or all of the running processes. The information given are the process identification number (PID),the amount of time that the process has taken to execute so far etc  
  
Q.How would you kill a process?   
A.The kill command takes the PID as one argument; this identifies which process to terminate. The PID of a process can be got using 'ps' command.

Q.What is an advantage of executing a process in background?   
A.The most common reason to put a process in the background is to allow you to do something else interactively without waiting for the process to complete. At the end of the command you add the special background symbol, &. This symbol tells your shell to execute the given command in the background. Example: cp \*.\* ../backup& (cp is for copy)

Q.What is the difference between Swapping and Paging?   
A.Swapping: Whole process is moved from the swap device to the main memory for execution. Process size must be less than or equal to the available main memory. It is easier to implementation and overhead to the system. Swapping systems does not handle the memory more flexibly as compared to the paging systems. Paging: Only the required memory pages are moved to main memory from the swap device for execution. Process size does not matter. Gives the concept of the virtual memory. It provides greater flexibility in mapping the virtual address space into the physical memory of the machine. Allows more number of processes to fit in the main memory simultaneously. Allows the greater process size than the available physical memory. Demand paging systems handle the memory more flexibly.  
  
Q.What is a Region?   
A.A Region is a continuous area of a process’s address space (such as text, data and stack). The kernel in a ‘Region Table’ that is local to the process maintains region. Regions are sharable among the process.

Q.What is Fork swap?   
A.fork() is a system call to create a child process. When the parent process calls fork() system call, the child process is created and if there is short of memory then the child process is sent to the read-to-run state in the swap device, and return to the user state without swapping the parent process. When the memory will be available the child process will be swapped into the main memory.

Q.What is Expansion swap?   
A.At the time when any process requires more memory than it is currently allocated, the Kernel performs Expansion swap. To do this Kernel reserves enough space in the swap device. Then the address translation mapping is adjusted for the new virtual address space but the physical memory is not allocated. At last Kernel swaps the process into the assigned space in the swap device. Later when the Kernel swaps the process into the main memory this assigns memory according to the new address translation mapping  
  
Q.What is called a page fault?   
A.Page fault is referred to the situation when the process addresses a page in the working set of the process but the process fails to locate the page in the working set. And on a page fault the kernel updates the working set by reading the page from the secondary device.

Q.Difference between the fork() and vfork() system call?   
A.During the fork() system call the Kernel makes a copy of the parent process’s address space and attaches it to the child process. But the vfork() system call do not makes any copy of the parent’s address space, so it is faster than the fork() system call. The child process as a result of the vfork() system call executes exec() system call. The child process from vfork() system call executes in the parent’s address space (this can overwrite the parent’s data and stack ) which suspends the parent process until the child process exits.

Q.What is Page-Stealer process?   
A.This is the Kernel process that makes rooms for the incoming pages, by swapping the memory pages that are not the part of the working set of a process. Page-Stealer is created by the Kernel at the system initialization and invokes it throughout the lifetime of the system. Kernel locks a region when a process faults on a page in the region, so that page stealer cannot steal the page, which is being faulted in.

Q.How can a parent and child process communicate?   
A parent and child can communicate through any of the normal inter-process communication schemes (pipes, sockets, message queues, shared memory), but also have some special ways to communicate that take advantage of their relationship as a parent and child. One of the most obvious is that the parent can get the exit status of the child.

Q.How can you get/set an environment variable from a program?:   
A.Getting the value of an environment variable is done by using `getenv()'. Setting the value of an environment variable is done by using `putenv()'.

Q.List the system calls used for process management:   
A.System calls Description fork() To create a new process exec() To execute a new program in a process wait() To wait until a created process completes its execution exit() To exit from a process execution getpid() To get a process identifier of the current process getppid() To get parent process identifier nice() To bias the existing priority of a process brk() To increase/decrease the data segment size of a process

Q.What is 'inode'?   
A.All UNIX files have its description stored in a structure called 'inode'. The inode contains info about the file-size, its location, time of last access, time of last modification, permission and so on. Directories are also represented as files and have an associated inode. In addition to descriptions about the file, the inode contains pointers to the data blocks of the file. If the file is large, inode has indirect pointer to a block of pointers to additional data blocks (this further aggregates for larger files). A block is typically 8k. Inode consists of the following fields: File owner identifier File type File access permissions File access times Number of links File size Location of the file data  
  
Q.Brief about the directory representation in UNIX   
A.A Unix directory is a file containing a correspondence between filenames and inodes. A directory is a special file that the kernel maintains. Only kernel modifies directories, but processes can read directories. The contents of a directory are a list of filename and inode number pairs. When new directories are created, kernel makes two entries named '.' (refers to the directory itself) and '..' (refers to parent directory). System call for creating directory is mkdir (pathname, mode).

Q.What are the Unix system calls for I/O?   
A.open(pathname,flag,mode) - open file creat(pathname,mode) - create file close(filedes) - close an open file read(filedes,buffer,bytes) - read data from an open file write(filedes,buffer,bytes) - write data to an open file lseek(filedes,offset,from) - position an open file dup(filedes) - duplicate an existing file descriptor dup2(oldfd,newfd) - duplicate to a desired file descriptor fcntl(filedes,cmd,arg) - change properties of an open file ioctl(filedes,request,arg) - change the behaviour of an open file The difference between fcntl anf ioctl is that the former is intended for any open file, while the latter is for device-specific operations.

Q.How do you change File Access Permissions?   
A.Every file has following attributes: owner's user ID ( 16 bit integer ) owner's group ID ( 16 bit integer ) File access mode word 'r w x -r w x- r w x' (user permission-group permission-others permission) r-read, w-write, x-execute To change the access mode, we use chmod(filename,mode). Example 1: To change mode of myfile to 'rw-rw-r--' (ie. read, write permission for user - read,write permission for group - only read permission for others) we give the args as: chmod(myfile,0664) . Each operation is represented by discrete values 'r' is 4 'w' is 2 'x' is 1 Therefore, for 'rw' the value is 6(4+2).   
  
Q.What are links and symbolic links in UNIX file system?   
A.A link is a second name (not a file) for a file. Links can be used to assign more than one name to a file, but cannot be used to assign a directory more than one name or link filenames on different computers. Symbolic link 'is' a file that only contains the name of another file.Operation on the symbolic link is directed to the file pointed by the it.Both the limitations of links are eliminated in symbolic links. Commands for linking files are: Link ln filename1 filename2 Symbolic link ln -s filename1 filename2

Q.What is a FIFO?   
A.FIFO are otherwise called as 'named pipes'. FIFO (first-in-first-out) is a special file which is said to be data transient. Once data is read from named pipe, it cannot be read again. Also, data can be read only in the order written. It is used in interprocess communication where a process writes to one end of the pipe (producer) and the other reads from the other end (consumer).

Q.How do you create special files like named pipes and device files?   
A.The system call mknod creates special files in the following sequence. 1. kernel assigns new inode, 2. sets the file type to indicate that the file is a pipe, directory or special file, 3. If it is a device file, it makes the other entries like major, minor device numbers. For example: If the device is a disk, major device number refers to the disk controller and minor device number is the disk.

Q.Discuss the mount and unmount system calls   
A.The privileged mount system call is used to attach a file system to a directory of another file system; the unmount system call detaches a file system. When you mount another file system on to your directory, you are essentially splicing one directory tree onto a branch in another directory tree. The first argument to mount call is the mount point, that is , a directory in the current file naming system. The second argument is the file system to mount to that point. When you insert a cdrom to your unix system's drive, the file system in the cdrom automatically mounts to /dev/cdrom in your system.

Q.How does the inode map to data block of a file?   
A.Inode has 13 block addresses. The first 10 are direct block addresses of the first 10 data blocks in the file. The 11th address points to a one-level index block. The 12th address points to a two-level (double in-direction) index block. The 13th address points to a three-level(triple in-direction)index block. This provides a very large maximum file size with efficient access to large files, but also small files are accessed directly in one disk read.  
  
Q.What is a shell?   
A.A shell is an interactive user interface to an operating system services that allows an user to enter commands as character strings or through a graphical user interface. The shell converts them to system calls to the OS or forks off a process to execute the command. System call results and other information from the OS are presented to the user through an interactive interface. Commonly used shells are sh,csh,ks etc.

Q.Brief about the initial process sequence while the system boots up.   
A.While booting, special process called the 'swapper' or 'scheduler' is created with Process-ID 0. The swapper manages memory allocation for processes and influences CPU allocation. The swapper inturn creates 3 children: the process dispatcher, vhand and dbflush with IDs 1,2 and 3 respectively. This is done by executing the file /etc/init. Process dispatcher gives birth to the shell. Unix keeps track of all the processes in an internal data structure called the Process Table (listing command is ps -el).  
  
Q.What are various IDs associated with a process?   
A.Unix identifies each process with a unique integer called ProcessID. The process that executes the request for creation of a process is called the 'parent process' whose PID is 'Parent Process ID'. Every process is associated with a particular user called the 'owner' who has privileges over the process. The identification for the user is 'UserID'. Owner is the user who executes the process. Process also has 'Effective User ID' which determines the access privileges for accessing resources like files. getpid() -process id getppid() -parent process id getuid() -user id geteuid() -effective user id

Q.Explain fork() system call.   
A.The `fork()' used to create a new process from an existing process. The new process is called the child process, and the existing process is called the parent. We can tell which is which by checking the return value from `fork()'. The parent gets the child's pid returned to him, but the child gets 0 returned to him.

Q.What is page fault? Its types?   
A.Page fault refers to the situation of not having a page in the main memory when any process references it. There are two types of page fault : Validity fault, Protection fault.

Q.In what way the Fault Handlers and the Interrupt handlers are different?   
A.Fault handlers are also an interrupt handler with an exception that the interrupt handlers cannot sleep. Fault handlers sleep in the context of the process that caused the memory fault. The fault refers to the running process and no arbitrary processes are put to sleep.

Q.Is it possible to create new a file system in UNIX?   
A.Yes, ‘mkfs’ is used to create a new file system  
  
Q.Is it possible to restrict incoming message?   
A.Yes, using the ‘mesg’ command.

Q.What is the use of the command "ls -x chapter[1-5]"   
A.ls stands for list; so it displays the list of the files that starts with 'chapter' with suffix '1' to '5', chapter1, chapter2, and so on.

Q.Is ‘du’ a command? If so, what is its use?   
A.Yes, it stands for ‘disk usage’. With the help of this command you can find the disk capacity and free space of the disk.

Q.Is it possible to count number char, line in a file; if so, How?   
A.Yes, wc-stands for word count. wc -c for counting number of characters in a file. wc -l for counting lines in a file.

Q.Name the data structure used to maintain file identification?   
A.‘inode’, each file has a separate inode and a unique inode number.

Q.How many prompts are available in a UNIX system?   
A.Two prompts, PS1 (Primary Prompt), PS2 (Secondary Prompt).

Q.How does the kernel differentiate device files and ordinary files?   
A.Kernel checks 'type' field in the file's inode structure.

Q.How to switch to a super user status to gain privileges?   
A.Use ‘su’ command. The system asks for password and when valid entry is made the user gains super user (admin) privileges.

Q.What are shell variables?   
A.Shell variables are special variables, a name-value pair created and maintained by the shell. Example: PATH, HOME, MAIL and TERM

Q.What is redirection?   
A.Directing the flow of data to the file or from the file for input or output. Example : ls > wc   
  
Q.How to terminate a process which is running and the specialty on command kill 0?   
A.With the help of kill command we can terminate the process. Syntax: kill pid Kill 0 - kills all processes in your system except the login shell.

Q.What is a pipe and give an example?   
A.A pipe is two or more commands separated by pipe char '|'. That tells the shell to arrange for the output of the preceding command to be passed as input to the following command. Example : ls -l | pr The output for a command ls is the standard input of pr. When a sequence of commands are combined using pipe, then it is called pipeline.

Q.What will the following command do?$ echo \*   
A.It is similar to 'ls' command and displays all the files in the current directory.

Q.Write a command to display a file’s contents in various formats?   
A.$od -cbd file\_name c - character, b - binary (octal), d-decimal, od=Octal Dump.

Q.Which command is used to delete all files in the current directory and all its sub-directories?   
A.rm -r \*

Q.Explain different types of Unix systems.   
A.The most widely used are: 1. System V (AT&T) 2. AIX (IBM) 3. BSD (Berkeley) 4. Solaris (Sun) 5. Xenix ( A PC version of Unix)

Q.Explain kernal and shell.   
A.Kernal: It carries out basic operating system functions such as allocating memory, accessing files and handling communications. Shell:A shell provides the user interface to the kernal.There are 3 major shells : C-shell, Bourne shell , Korn shell  
  
Q.What is ex and vi ?   
A.ex is Unix line editor and vi is the standard Unix screen editor.  
  
Q.Which are typical system directories below the root directory?   
A.(1)/bin: contains many programs which will be executed by users (2)/etc : files used by administrator (3)/dev: hardware devices (4)/lib: system libraries (5)/usr: application software (6)/home: home directories for different systems.

Q.What is the significance of the “tee” command?   
A.It reads the standard input and sends it to the standard output while redirecting a copy of what it has read to the file specified by the user.

Q.What does the command “ $who | sort –logfile > newfile” do?   
A.The input from a pipe can be combined with the input from a file . The trick is to use the special symbol “-“ (a hyphen) for those commands that recognize the hyphen as std input. In the above command the output from who becomes the std input to sort , meanwhile sort opens the file logfile, the contents of this file is sorted together with the output of who (rep by the hyphen) and the sorted output is redirected to the file newfile  
  
Q.What does the command “$ls | wc –l > file1” do?   
A.ls becomes the input to wc which counts the number of lines it receives as input and instead of displaying this count , the value is stored in file1.

Q.Which of the following commands is not a filter man , (b) cat , (c) pg , (d) head   
A.Ans: man A filter is a program which can receive a flow of data from std input, process (or filter) it and send the result to the std output.

Q.How is the command “$cat file2 “ different from “$cat >file2 and >> redirection operators ?   
A.is the output redirection operator when used it overwrites while >> operator appends into the file.  
  
Q.What difference between cmp and diff commands?   
A.cmp - Compares two files byte by byte and displays the first mismatch diff - tells the changes to be made to make the files identical

Q.What is the use of ‘grep’ command?   
A.‘grep’ is a pattern search command. It searches for the pattern, specified in the command line with appropriate option, in a file(s). Syntax : grep Example : grep 99mx mcafile

Q.What is the difference between cat and more command?   
A.Cat displays file contents. If the file is large the contents scroll off the screen before we view it. So command 'more' is like a pager which displays the contents page by page  
  
Q.Write a command to kill the last background job?   
A.Kill $!

Q. Explain kill() and its possible return values.   
A.There are four possible results from this call: ‘kill()’ returns 0. This implies that a process exists with the given PID, and the system would allow you to send signals to it. It is system-dependent whether the process could be a zombie. ‘kill()’ returns -1, ‘errno == ESRCH’ either no process exists with the given PID, or security enhancements are causing the system to deny its existence. (On some systems, the process could be a zombie.) ‘kill()’ returns -1, ‘errno == EPERM’ the system would not allow you to kill the specified process. This means that either the process exists (again, it could be a zombie) or draconian security enhancements are present (e.g. your process is not allowed to send signals to \*anybody\*). ‘kill()’ returns -1, with some other value of ‘errno’ you are in trouble! The most-used technique is to assume that success or failure with ‘EPERM’ implies that the process exists, and any other error implies that it doesn't. An alternative exists, if you are writing specifically for a system (or all those systems) that provide a ‘/proc’ filesystem: checking for the existence of ‘/proc/PID’ may work.

Q.Explain about Stdin, Stdout and Stderr?  
A.These are known as standard input, output and error. These are categorized as 0, 1 and 2. Each of these functions has a particular role and should accordingly functions for efficient output. Any mismatch among these three could result in a major failure of the shell.  
  
Q.Explain about sourcing commands?  
A.Sourcing commands help you to execute the scripts within the scripts. For example sh command makes your program to run as a separate shell. .command makes your program to run within the shell. This is an important command for beginners and for special purposes.  
  
Q.Explain about debugging?  
A.Shell can make your debugging process easier because it has lots of commands to perform the function. For example sh –ncommand helps you to perform debugging. It helps you to read the shell but not to execute it during the course. Similarly sh –x command helps you by displaying the arguments and functions as they are executed.  
  
Q.Explain about Login shell?  
A.Login shell is very useful as it creates an environment which is very useful to create the default parameters. It consists of two files they are profile files and shell rc files. These files initialize the login and non login files. Environment variables are created by Login shell.  
  
Q.Explain about non-login shell files?  
A.The non login shell files are initialized at the start and they are made to run to set up variables. Parameters and path can be set etc are some important functions. These files can be changed and also your own environment can be set. These functions are present in the root. It runs the profile each time you start the process.  
  
Q.Explain about shebang?  
A.Shebang is nothing but a # sign followed by an exclamation. This is visible at the top of the script and it is immediately followed by an exclamation. To avoid repetitive work each time developers use shebang. After assigning the shebang work we pass info to the interpreter.  
  
Q.Explain about the Exit command?  
A.Every program whether on UNIX or Linux should end at a certain point of time and successful completion of a program is denoted by the output 0. If the program gives an output other than 0 it defines that there has been some problem with the execution or termination of the problem. Whenever you are calling other function, exit command gets displayed.  
  
Q.Explore about Environment variables?  
A.Environment variables are set at the login time and every shell that starts from this shell gets a copy of the variable. When we export the variable it changes from an shell variable to an environment variable and these variables are initiated at the start of the shell.  
  
Q.How do you find out what’s your shell?  
A. - echo $SHELL  
  
Q.What’s the command to find out today’s date?  
A.- date  
  
Q.What’s the command to find out users on the system?  
A.- who  
  
Q.How do you find out the current directory you’re in?  
A.- pwd  
  
Q.How do you remove a file?   
A.- rm  
  
Q.How do you find out your own username?  
A.- whoami  
  
Q.How do you count words, lines and characters in a file?  
A.- wc  
  
Q.How do you search for a string inside a given file?.  
A.- grep string filename  
  
Q.How do you search for a string inside a directory?  
A.- grep string \*  
  
Q.How do you search for a string in a directory with the subdirectories recursed?  
A.- grep -r string \*  
  
Q.What are PIDs?  
A.- They are process IDs given to processes. A PID can vary from 0 to 65535.  
  
Q.How do you list currently running process?  
A.- ps  
  
Q.What are the different kinds of loops available in shell script?  
A.for, if, while, case  
  
Q.What is the difference between a shell variable that is exported and the one that is not exported?  
A.The Shell variable which is exported would available to all the programs outside the Shell also. And the shell variable which is not exported, would available for that shell or for the shell program only, in which the variable is declared.  
  
Export LANG=C : will make the variable LANG the global variable, put it into the global environment. All other processes can use it.   
  
LANG=C : will change the value only in the current script.  
  
Q.How will you list only the empty lines in a file (using grep)?  
A.grep “^$” filename.txt  
  
Q.How would you get the character positions 10-20 from a text file?  
A.cat filename.txt | cut -c 10-20  
or  
cut -c10-20   
  
Q.How would you replace the n character in a file with some xyz?  
A.sed ’s/n/xyz/g’ filename > new\_filename  
We can replace n characters by using the following command:  
1,$s/./xyz/g  
where 1 shows that the search string will start searching patterns from first line of the file. ‘.’ for any character.g for global replacemet.  
  
Q.What is the difference between a ‘thread’ and a ‘process’?  
A.A process is a collection of virtual memory space, code, data, and system resources. A thread is code that is to be serially executed within a process. A processor executes threads, not processes, so each application has at least one process, and a process always has at least one thread of execution, known as the primary thread. A process can have multiple threads in addition to the primary thread.Thread – is stream of executable code within process. They are light weight process. All thread with in a process share process instruction,code & data segment,open file descriptor,signal handler,userID and GroupID. Thread has its own set of register including program counter,stack pointer .  
  
Q.What is this line in the shell script do ?#!/bin/ksh?  
A.To invoke the shell indirectly this line is added as the first line in the file.This particular line invokes korn shell  
  
Q.What is use of “cut” command? Give some examples.   
A.Cut – Utility used to cut/Strip out the required data/text from the source.Cut can be used in three modes,  
1.Stripping by Character  
cut -c 1-3  
2.Striping by Byte length  
cut -b -1-72  
3.Stripping by delimiter and fields.  
cut -d “|” -f1  
Where   
-d “|” -> Delimiter used in input text to separate columns  
-f1 -> Field/Column number   
While processing huge input files, Cut’s performance is far better than awk.

**35. What is a Region?**   
**Ans:** A Region is a continuous area of a process’s address space (such as text, data and stack). The kernel in a ‘Region Table’ that is local to the process maintains region. Regions are sharable among the process.

**36. What are the events done by the Kernel after a process is being swapped out from the main memory?**   
**Ans:** When Kernel swaps the process out of the primary memory, it performs the following:  
Kernel decrements the Reference Count of each region of the process. If the reference count becomes zero, swaps the region out of the main memory,Kernel allocates the space for the swapping process in the swap device,Kernel locks the other swapping process while the current swapping operation is going on,The Kernel saves the swap address of the region in the region table.

**37. Is the Process before and after the swap are the same? Give reason.**   
**Ans:** Process before swapping is residing in the primary memory in its original form. The regions (text, data and stack) may not be occupied fully by the process, there may be few empty slots in any of the regions and while swapping Kernel do not bother about the empty slots while swapping the process out. After swapping the process resides in the swap (secondary memory) device. The regions swapped out will be present but only the occupied region slots but not the empty slots that were present before assigning. While swapping the process once again into the main memory, the Kernel referring to the Process Memory Map, it assigns the main memory accordingly taking care of the empty slots in the regions.

**38. What do you mean by u-area (user area) or u-block?**   
**Ans:** This contains the private data that is manipulated only by the Kernel. This is local to the Process, i.e. each process is allocated a u-area.

**39. What are the entities that are swapped out of the main memory while swapping the process out of the main memory?**   
**Ans:** All memory space occupied by the process, process’s u-area, and Kernel stack are swapped out, theoretically. Practically, if the process’s u-area contains the Address Translation Tables for the process then Kernel implementations do not swap the u-area.

**40. What is Fork swap?**   
**Ans:** fork() is a system call to create a child process. When the parent process calls fork() system call, the child process is created and if there is short of memory then the child process is sent to the read-to-run state in the swap device, and return to the user state without swapping the parent process. When the memory will be available the child process will be swapped into the main memory.

**41. What is Expansion swap?**   
**Ans:** At the time when any process requires more memory than it is currently allocated, the Kernel performs Expansion swap. To do this Kernel reserves enough space in the swap device. Then the address translation mapping is adjusted for the new virtual address space but the physical memory is not allocated. At last Kernel swaps the process into the assigned space in the swap device. Later when the Kernel swaps the process into the main memory this assigns memory according to the new address translation mapping.

**42. How the Swapper works?**   
**Ans:** The swapper is the only process that swaps the processes. The Swapper operates only in the Kernel mode and it does not uses System calls instead it uses internal Kernel functions for swapping. It is the archetype of all kernel process.

**43. What are the processes that are not bothered by the swapper? Give Reason.**  
**Ans:** Zombie process: They do not take any up physical memory.Processes locked in memories that are updating the region of the process.Kernel swaps only the sleeping processes rather than the ‘ready-to-run’ processes, as they have the higher probability of being scheduled than the Sleeping processes.

**44. What are the requirements for a swapper to work?**   
**Ans:** The swapper works on the highest scheduling priority. Firstly it will look for any sleeping process, if not found then it will look for the ready-to-run process for swapping. But the major requirement for the swapper to work the ready-to-run process must be core-resident for at least 2 seconds before swapping out. And for swapping in the process must have been resided in the swap device for at least 2 seconds. If the requirement is not satisfied then the swapper will go into the wait state on that event and it is awaken once in a second by the Kernel.

**45. What are the criteria for choosing a process for swapping into memory from the swap device?**   
**Ans:** The resident time of the processes in the swap device, the priority of the processes and the amount of time the processes had been swapped out.

**46. What are the criteria for choosing a process for swapping out of the memory to the swap device?**  
**Ans:** The process’s memory resident time,Priority of the process and the nice value.

**47. What do you mean by nice value?**   
**Ans:** Nice value is the value that controls {increments or decrements} the priority of the process. This value that is returned by the nice () system call. The equation for using nice value is: Priority = (“recent CPU usage”/constant) + (base- priority) + (nice value) Only the administrator can supply the nice value. The nice () system call works for the running process only. Nice value of one process cannot affect the nice value of the other process.

1. What is shell scripting?  
   Shell scripting is used to program command line of an operating system. Shell Scripting is also used to program the shell which is the base for any operating system. Shell scripts often refer to programming UNIX. Shell scripting is mostly used to program operating systems of windows, UNIX, Apple, etc. Also this script is used by companies to develop their own operating system with their own features.  
     
   2) State the advantages of Shell scripting?  
   There are many advantages of shell scripting some of them are, one can develop their own operating system with relevant features best suited to their organization than to rely on costly operating systems. Software applications can be designed according to their platform.  
     
   3) What are the disadvantages of shell scripting?  
   There are many disadvantages of shell scripting they are  
   \* Design flaws can destroy the entire process and could prove a costly error.  
   \* Typing errors during the creation can delete the entire data as well as partition data.  
   \* Initially process is slow but can be improved.  
   \* Portbility between different operating system is a prime concern as it is very difficult to port scripts etc.  
     
   4) Explain about the slow execution speed of shells?  
   Major disadvantage of using shell scripting is slow execution of the scripts. This is because for every command a new process needs to be started. This slow down can be resolved by using pipeline and filter commands. A complex script takes much longer time than a normal script.  
     
   5) Give some situations where typing error can destroy a program?  
   There are many situations where typing errors can prove to be a real costly effort. For example a single extra space can convert the functionality of the program from deleting the sub directories to files deletion. cp, cn, cd all resemble the same but their actual functioning is different. Misdirected > can delete your data.  
   Coding Related Shell Scripting Interview Questions ...  
     
   6) Explain about return code?  
   Return code is a common feature in shell programming. These return codes indicate whether a particular program or application has succeeded or failed during its process. && can be used in return code to indicate which application needs to be executed first.  
     
   7) What are the different variables present in Linux shell?  
   Variables can be defined by the programmer or developer they specify the location of a particular variable in the memory. There are two types of shells they are System variables and user defined variables. System variables are defined by the system and user defined variables are to be defined by the user (small letters).  
     
   8) Explain about GUI scripting?  
   Graphical user interface provided the much needed thrust for controlling a computer and its applications. This form of language simplified repetitive actions. Support for different applications mostly depends upon the operating system. These interact with menus, buttons, etc.  
   Shell Scripting Command Interview Questions ...  
     
   9) Explain about echo command?  
   Echo command is used to display the value of a variable. There are many different options give different outputs such as usage \c suppress a trailing line, \r returns a carriage line, -e enables interpretation, \r returns the carriage.  
     
   10) Explain about Stdin, Stdout and Stderr?  
   These are known as standard input, output and error. These are categorized as 0, 1 and 2. Each of these functions has a particular role and should accordingly functions for efficient output. Any mismatch among these three could result in a major failure of the shell.  
     
   11) Explain about sourcing commands?  
   Sourcing commands help you to execute the scripts within the scripts. For example sh command makes your program to run as a separate shell. .command makes your program to run within the shell. This is an important command for beginners and for special purposes.  
     
   12) Explain about debugging?  
   Shell can make your debugging process easier because it has lots of commands to perform the function. For example sh –ncommand helps you to perform debugging. It helps you to read the shell but not to execute it during the course. Similarly sh –x command helps you by displaying the arguments and functions as they are executed.  
     
   13) Explain about Login shell?  
   Login shell is very useful as it creates an environment which is very useful to create the default parameters. It consists of two files they are profile files and shell rc files. These files initialize the login and non login files. Environment variables are created by Login shell.  
     
   14) Explain about non-login shell files?  
   The non login shell files are initialized at the start and they are made to run to set up variables. Parameters and path can be set etc are some important functions. These files can be changed and also your own environment can be set. These functions are present in the root. It runs the profile each time you start the process.  
     
   15) Explain about shebang?  
   Shebang is nothing but a # sign followed by an exclamation. This is visible at the top of the script and it is immediately followed by an exclamation. To avoid repetitive work each time developers use shebang. After assigning the shebang work we pass info to the interpreter.  
     
   16) Explain about the Exit command?  
   Every program whether on UNIX or Linux should end at a certain point of time and successful completion of a program is denoted by the output 0. If the program gives an output other than 0 it defines that there has been some problem with the execution or termination of the problem. Whenever you are calling other function, exit command gets displayed.  
     
   17) Explore about Environment variables?  
   Environment variables are set at the login time and every shell that starts from this shell gets a copy of the variable. When we export the variable it changes from an shell variable to an environment variable and these variables are initiated at the start of the shell.

# Basic shell scripting questions

By [*admin*](http://www.techinterviews.com/author/admin/) | July 22, 2007

1. **How do you find out what’s your shell?** - echo $SHELL
2. **What’s the command to find out today’s date?** - date
3. **What’s the command to find out users on the system?** - who
4. **How do you find out the current directory you’re in?** - pwd
5. **How do you remove a file?** - rm
6. **How do you remove a <="" in="" files="" the="" all="" with=""> - rm -rf**
7. **How do you find out your own username? - whoami**
8. **How do you send a mail message to somebody? - mail** [**somebody@techinterviews.com**](mailto:somebody@techinterviews.com) **-s ‘Your subject’ -c** [**‘cc@techinterviews.com**](mailto:'cc@techinterviews.com)**‘**
9. **How do you count words, lines and characters in a file? - wc**
10. **How do you search for a string inside a given file? - grep string filename**
11. **How do you search for a string inside a directory? - grep string \***
12. **How do you search for a string in a directory with the subdirectories recursed? - grep -r string \***
13. **What are PIDs? - They are process IDs given to processes. A PID can vary from 0 to 65535.**
14. **How do you list currently running process? - ps**
15. **How do you stop a process? - kill pid**
16. **How do you find out about all running processes? - ps -ag**
17. **How do you stop all the processes, except the shell window? - kill 0**
18. **How do you fire a process in the background? - ./process-name &**
19. **How do you refer to the arguments passed to a shell script? - $1, $2 and so on. $0 is your script name.**
20. **What’s the conditional statement in shell scripting? - if {condition} then … fi**
21. **How do you do number comparison in shell scripts? - -eq, -ne, -lt, -le, -gt, -ge**
22. **How do you test for file properties in shell scripts? - -s filename tells you if the file is not empty, -f filename tells you whether the argument is a file, and not a directory, -d filename tests if the argument is a directory, and not a file, -w filename tests for writeability, -r filename tests for readability, -x filename tests for executability**
23. **How do you do Boolean logic operators in shell scripting? - ! tests for logical not, -a tests for logical and, and -o tests for logical or.**
24. **How do you find out the number of arguments passed to the shell script? - $#**
25. **What’s a way to do multilevel if-else’s in shell scripting? - if {condition} then {statement} elif {condition} {statement} fi**
26. **How do you write a for loop in shell? - for {variable name} in {list} do {statement} done**
27. **How do you write a while loop in shell? - while {condition} do {statement} done**
28. **How does a case statement look in shell scripts? - case {variable} in {possible-value-1}) {statement};; {possible-value-2}) {statement};; esac**
29. **How do you read keyboard input in shell scripts? - read {variable-name}**
30. **How do you define a function in a shell script? - function-name() { #some code here return }**
31. **How does getopts command work? - The parameters to your script can be passed as -n 15 -x 20. Inside the script, you can iterate through the getopts array as while getopts n:x option, and the variable $option contains the value of the entered option.**

**UNIX interview questions and best practices.....**

Q1.Write a shell script in Linux to shift all characters in a file forward by five characters. (Thus "a" becomes "f'").  
Solution: cat test tr '[a-z A-Z]' '[f-za-e F-ZA-E]'  
  
Q2.I have 2 files:-  
file1 and file2  
  
file1  
SEED  
RPTT  
TST8  
  
file2  
SEED:db:Y  
RPTT:db:Y  
SED8:db:N  
SEED:db:N  
TST8:db2:Y  
TRN8:db3:N  
CNV8:db4:Y  
  
I have to change third field of file2 to "y" for every entry in file1 matches first filed of file 2  
and rest to N  
  
I want file 2 like :-  
as file 1 has (SEED,RPTT,TST8) so in file2 I want to do third field "y" for those entries(which is first field in file2).  
In rest I want to convert to "N"  
SEED:db:Y  
RPTT:db:Y  
SED8:db:N  
SEED:db:Y  
TST8:db2:Y  
TRN8:db3:N  
CNV8:db4:N  
  
Solution: while IFS=: read a b \_ ; do echo -n "$a:$b:" ; ([[ $(grep -w $a file1.txt) ]] && echo Y) echo N ; done <>  
  
Q3. How u convert string "hi manu how are you?" to "Hi Manu How Are You?"   
Solution: echo "hi manu how are you"sed -e 's/ [a-z]/\U&/g'\;'s/^[a-z]/\U&/g'  
  
OR  
  
echo "hi manu how are you" sed 's/\<[a-z]\*\>/\u&/g'  
  
Q4. write a shell script that counts a number of unique word contained in the file and print them in alphabetical order line by line?  
Solution: $ rm -f res res1 ; while read line ; do cat uniqtest grep -wc $line >>res1 ; echo "$line :-> Count=" >>res ;done final\_temp ; cat final\_tempsort -u >final ; rm -f res res1 final\_temp1  
(Check file "final" for output)  
  
Q5. How to rename all the files in a folder having specific extension? Example: I have some files with extension (.txt) in a folder name 'Test'. I have to rename all the .txt files in a test and its subdirectories to .my extension.   
Solution: for file in `ls` ; do NEW=`echo $file sed 's/.txt/.my/g'` ; mv $file $NEW ; done  
  
Q6. write a shell script to identify the given string is palindrome or not?   
Solution:   
#! /usr/bin/sh  
len=0  
i=1  
tag=0  
echo -n "Enter a String: "  
read str  
len=`echo $str wc -c`  
len=`expr $len - 1`  
halfLen=`expr $len / 2`  
  
while [ $i -le $halfLen ]  
  
do  
c1=`echo $strcut -c$i`  
c2=`echo $strcut -c$len`  
if [ $c1 != $c2 ] ; then  
i=$halfLen  
tag=1  
fi  
i=`expr $i + 1`  
len=`expr $len - 1`  
done  
  
  
if [ $tag -eq 0 ]  
  
then  
echo "String is Palindrome"  
else  
echo "String is not Palindrome"  
  
fi  
  
Q7. one file is a.txt which has 4 fields/columns, where as each field has separated by space and that field contains some data like below  
  
ram usa 105 25  
rag uk 115 26  
pat ind 234 23  
sah 425 24  
tat usa 344 28  
rat brz 536 29  
  
now i want only 2nd field data, that should be like this  
  
usa  
uk  
ind  
  
usa  
brz  
  
(i.e. It should print blank line for the second field if it doesnt have country defined)  
Solution: awk 'NF < 4 {$2=""} { print $2 }' a.txt  
  
Q8. How to change all .my files to .txt in current directory??  
Solution: for i in `ls \*.my` ; do NEW=`echo $ised -e 's/.my/.txt/g'` ; mv $i $NEW ; done  
  
Q9. How to change all .extn files to .my in entire directory recursively??  
Solution:  
find ./ -type f grep ".extn$" >files.txt;for i in `cat files.txt`;do NEW=`echo $ised 's/.extn/.my/g'`;mv "${i}" "${NEW}";done;rm -f files.txt  
  
Q10: What is umask??  
Answer: umask is used to set default permission level in entire unix system. It can be set in .kshrc/.bashrc/.cshrc or init (So that to load at the time of login into shell)  
  
Q11. What is inode??  
Answer : inode represents any file/directory residing in your unix system with a numner known as inode number. It is unique for every file/dir. It is basically a reference that kernel use for any work related to that entity. "ls -i filename" will give you the inode number of that particu;ar file.  
  
Q12. Explain general commands like ps,df, export, env, ufsdump, tar, cron, system.  
Answer: EASY ONE  
  
Q13. Explain the make utility in unix?? what are the parameters of a makefile and how to use them. What are macros in makefile.  
Answer: Unix make utility is used to build a software package. It requires a makefile where all the bunch of the scripts/codes/programs are declared/defined. The main parameters of the make file are variable declaration, Macro declaration, Phony Target declaration and Clean-up block. Macros in a makefile represents the order in which the scripts would get executed.  
  
Q14. What are the benefits of makefile over shell script??  
Answer: Using makefile one can run the different scripts or programs or codes in one go. This can also be done by making a wrapper shell script which can execute the different scripts in one go. But then it would not be termed as a package because when we say a package it means that we have one installer (.exe) which can execute the bunch more efficiently and also this .exe will be in encrypted format. So in order to maintain efficiency and security makefile is given precedence over wrapper shell scripts.  
  
Q15. What are hard links in UNIX??  
Answer: A hard link is a reference to a file or directory that appears just like a file or directory, not a link. Hard links only work within a filesystem. In other words, don't use hard links between mounted filesystems. A hard link is only a reference to the original file, not a copy of the file. If the original file is deleted, the information will be lost.  
To create a hard link of the file /export/home/fred/stuff to /var/tmp/thing, use:  
  
ln /export/home/fred/stuff /var/tmp/thing  
  
The syntax for creating a hard link of a directory is the same. To create a hard link of /var/www/html to /var/www/webroot, use:  
  
ln /var/www/html /var/www/webroot  
  
Q16. What are the soft/symbolic links in unix ??  
Answer: A symbolic link is a pointer to another file or directory. It can be used just like the original file or directory. A symbolic link appears in a long listing (ls -l) with a reference to the original file/directory. A symbolic link, as opposed to a hard link, is required when linking from one filesystem to another and can be used within a filesystem as well.  
To create a symbolic link, the syntax of the command is similar to a copy or move command: existing file first, destination file second. For example, to link the directory /export/space/common/archive to /archive for easy access, use:  
  
ln -s /export/space/common/archive /archive  
  
To link the runtime control script /etc/init.d/httpd to /etc/rc2.d/S77httpd, use:  
  
cd /etc/rc2.d  
ln -s ../init.d/httpd S77httpd  
  
Q17. What is a command to kill the last background job??  
Answer: kill $!  
  
Q18. Bring a job to foreground by specifying its job number after the percent sign.  
Answer: fg %jobnumber  
(Note: jobnumber can be obtained by using jobs -l command)  
  
Q19. Explain top command.  
Answer: top provides an ongoing look at processor activity in real time. It displays a listing of the most CPU-intensive tasks on the system, and can provide an interactive interface for manipulating processes. It can sort the tasks by CPU usage, memory usage and runtime. can be better configured than the standard top from the procps suite. Most features can either be selected by an interactive command or by specifying the feature in the personal or system-wide configuration file.   
  
Q20. Write a script which give you those records which are having CPU utilization more than 80%.  
Answer: df -k awk 'NR==1{print $1"\t"$5"\t"$6}sub("%","",$5){if($5 >= 80) print $1"\t"$5"%\t"$NF}'  
Q21. How to pass arguments in an awk script??Answer: Using, cat a.txtawk -v awk\_var="$shell\_var" '{print $1 "\t" awk\_var}'  
e.g. #! /usr/bin/sh  
var=$\*  
cat ussd.txt >offpeak\_input.txt  
cat offpeaknawk -v var1="$var" '{print "\"1-"$1"\"""""\""$2"\"""""1006"""""$3"""0""""-1""""-1""""\""var1" .Account balance is Rs..Bonus minutes.\""}'>>offpeak\_input.txt

**What is Shell's Responsibilities ?**

The shell is responsible for the execution of all programs that you request from your terminal. Each time you type in a line to the shell, the shell analyzes the line and then determines what to do.The line that is typed to the shell is known more formally as the command line. The shell scans this command line and determines the name of the program to be executed and what arguments to pass to the program.

**Q: - What is "$#" Variable ?**  
  
The $# Variable  
Whenever you execute a shell program, the special shell variable $# gets set to the number of arguments that were typed on the command line.

**Q: - Explain "Exit Status" for a shell script ?**

Whenever any program completes execution under the Unix system, it returns an exit status back to the system. This status is a number that usually indicates whether the program successfully ran. By convention, an exit status of zero indicates that a program succeeded, and nonzero indicates that it failed. Failures can be caused by invalid arguments passed to the program, or by an error condition detected by the program. For example, the cp command returns a nonzero exit status if the copy fails for some reason (for example, if it can't create the destination file), or if the arguments aren't correctly specified (for example, wrong number of arguments, or more than two arguments and the last one isn't a directory). In the case of grep, an exit status of zero (success) is returned if it finds the specified pattern in at least one of the files; a nonzero value is returned if it can't find the pattern or if an error occurs (the arguments aren't correctly specified, or it can't open one of the files).

**Q: - What is "Command Substitution" ?**  
  
Command substitution is the process by which the shell runs a command and replaces the command substitution with the output of the executed command. That sounds like a mouthful, but it's pretty straightforward in practice.

**Q: - What is " eval" command ?**  
  
The eval command exists to supersede the normal command-line substitution and evaluation order, making it possible for a shell script to build up commands dynamically. This is a powerful facility, but it must be used carefully. Because the shell does so many different kinds of substitutions, it pays to understand the order in which the shell evaluates input lines.

**Q: - What is awk ?**  
  
An awk invocation can define variables, supply the program, and name the input files.

**Q: - What is "grep" programe ?**  
  
The grep program is the primary tool for extracting interesting lines of text from input datafiles. POSIX mandates a single version with different options to provide the behavior traditionally obtained from the three grep variants: grep, egrep, and fgrep.

**Q: - Name a new feature introduced with PHP 5.**  
  
PHP 5 introduces (among other things) SQLite support, improved XML support, and a significantly improved object model.

**Q: - explain "read" command ?**  
  
The read command is one of the most important ways to get information into a shell program:

$ x=abc ; printf "x is now '%s'. Enter new value: " $x ; read x

x is now 'abc'. Enter new value: PDQ

$ echo $x

PDQ

**Q: - What are the two files used by the shell to initialize itself?**

/etc/profile  
profile

**Q: - What is Interactive mode?**

Interactive mode means that the shell expects to read input from you and execute the commands that you specify. This mode is called interactive because the shell is interacting with a user. This is usually the mode of the shell that most users are familiar with: you log in, execute some commands, and log out. When you log out using the exit command, the shell exits.

**Q: - What is noninteractive mode?**

In this mode, the shell does not interact with you; instead it reads commands stored in a file and executes them. When it reaches the end of the file, the shell exits.

**Q: - what is local variable?**

A local variable is a variable that is present within the current instance of the shell. It is not available to programs that are started by the shell. The variables that you looked at previously have all been local variables.

**Q: - What is environment variable?**

An environment variable is a variable that is available to any child process of the shell. Some programs need environment variables in order to function correctly. Usually a shell script defines only those environment variables that are needed by the programs that it runs.

**Q: - What is shell variable?**

A shell variable is a special variable that is set by the shell and is required by the shell in order to function correctly. Some of these variables are environment variables whereas others are local variables.

**Q: - Explain the “Exit” command?**

Every program whether on UNIX or Linux should end at a certain point of time and successful completion of a program is denoted by the output 0. If the program gives an output other than 0 it defines that there has been some problem with the execution or termination of the problem. Whenever you are calling other function, exit command gets displayed.

**Q: - How do you find out what’s your shell?**

echo $SHELL

**Q: - How you will run a process in the background?**

./ProcessName &

**Q: - How do you write a while loop in shell?**

Use While Loop

**Q: - How do you read keyboard input in shell scripts?**

Use read command

**Q: - What is GUI Scripting?**

Graphical user interface provided the much needed thrust for controlling a computer and its applications. This form of language simplified repetitive actions. Support for different applications mostly depends upon the operating system. These interact with menus, buttons, etc.

**Q: - Explain the term “loops”?**

Loops enable you to execute a series of commands multiple times. Two main types of loops are the while and for loops.

**Q: - What is “Nested Loops”?**

When a loop is located inside the body of another loop it is said to be nested within another loop.

**Q: - What is “Infinite Loops”?**

Loops that execute forever without terminating.

**Q: - What is “File Descriptor”?**

An integer that is associated with a file. Enables you to read and write from a file using the integer instead of the file's name.

**Q: - Explain “STDIN”?**

STDINStandard Input. User input is read from STDIN. The file descriptor for STDIN is 0.

**Q: - Explain “STDOUT”?**

STDOUTStandard Output. The output of scripts is usually to STDOUT. The file descriptor for STDOUT is 1.

**Q: - Explain “STDERR”?**

STDERRStandard Error. A special type of output used for error messages. The file descriptor for STDERR is 2.

**Q: - Explain “Escape Sequence”?**

An escape sequence is special sequence of characters that represents another  
character.

**Q: - Explain “Output Redirection” in shell scripting?**

In UNIX or Linux, the process of capturing the output of a command and storing it in a file is called *output redirection* because it redirects the output of a command into a file instead of the screen.

**Q: Explain** “**Input Redirection” in shell scripting?**

In UNIX or Linux the process of sending input to a command from a file is called input redirection.

**Q: - What is** “**Field separator”?**

The field separator controls the manner in which an input line is broken into fields. In the shell, the field separator is stored in the variable IFS. In awk, the field separator is stored in the awk variable FS.

**Q: - What is** “**Library”?**

A file that contains only functions is called a library. Usually libraries contain no main code.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Re: Write a shell program to test whether a given number is even or odd? | | |
| **Answer** # [1](http://www.allinterview.com/viewpost/156634.html) | echo -n "Enter numnber : "  read n    rem=$(( $n % 2 ))    if [ $rem -eq 0 ]  then  echo "$n is even number"  else  echo "$n is odd number"  fi   |  |  |  | | --- | --- | --- | | **Is This Answer Correct ?** | **93 Yes** | 13 No | | 0 | **Fariha** |
|  | | | |
|  | Re: Write a shell program to test whether a given number is even or odd? | | |
| **Answer** # [2](http://www.allinterview.com/viewpost/159046.html) | echo "Enter the Number"  read a  b=`echo "$a % 2"|bc`  if [ $b -eq 0 ]  then  echo "Given Number is Even "  exit  fi  echo " Given Number is odd" |  |  |

1.What is $\*?  
Will display all the commandline arguments that are passed to the script

2. Different types of shells?  
sh : the oldest shell  
csh : C shell  
ksh : Korn Shell  
bash : bourne again shell

3.How do you read arguments in a shell program – $1, $2 ?

#!/bin/sh  
for i in $\*  
do  
echo $i  
done

On executig the above script with any number of command-line arguments it will display all the parametsrs.

4.What are the different kinds of loops available in shell script?

for  
while  
until

5.What does $# nad $? stand for?

$# returns the number of parameters that are passed to a shell script  
$? returns the exit code of the last executed command (0 : Successful, 1 or other: F)

6.What is the difference between a shell variable that is exported and the one that is not exported?

exported variable is visible to the child processes while the normal variables are not.

export LANG C  
will make the variable LANG the global variable put it into the global environment. all other processes can use it.

LANG C  
will change the value only in the current script.

7.If you have a string “one two three”, Which shell command would you use to extract the strings?

This article attempts to refresh your Unix skills in the form of a question/answer based Unix tutorial on Unix command lines. The commands discussed here are particulary useful for the developers working in the middle-tier (e.g. ETL) systems, where they may need to interact with several \*nx source systems for data retrieval.

### How to print/display the first line of a file?

There are many ways to do this. However the easiest way to display the first line of a file is using the [head] command.

$> head -1 file.txt

No prize in guessing that if you specify [head -2] then it would print first 2 records of the file.   
Another way can be by using [sed] command. [Sed] is a very powerful text editor which can be used for various text manipulation purposes like this.

$> sed '2,$ d' file.txt

How does the above command work? The 'd' parameter basically tells [sed] to delete all the records from display from line 2 to last line of the file (last line is represented by $ symbol). Of course it does not actually delete those lines from the file, it just does not display those lines in standard output screen. So you only see the remaining line which is the 1st line.

### How to print/display the last line of a file?

The easiest way is to use the [tail] command.

$> tail -1 file.txt

If you want to do it using [sed] command, here is what you should write:

$> sed -n '$ p' test

From our previous answer, we already know that '$' stands for the last line of the file. So '$ p' basically prints (p for print) the last line in standard output screen. '-n' switch takes [sed] to silent mode so that [sed] does not print anything else in the output.

### How to display n-th line of a file?

The easiest way to do it will be by using [sed] I guess. Based on what we already know about [sed] from our previous examples, we can quickly deduce this command:

$> sed –n '<n> p' file.txt

You need to replace <n> with the actual line number. So if you want to print the 4th line, the command will be

$> sed –n '4 p' test

Of course you can do it by using [head] and [tail] command as well like below:

$> head -<n> file.txt | tail -1

You need to replace <n> with the actual line number. So if you want to print the 4th line, the command will be

$> head -4 file.txt | tail -1

### How to remove the first line / header from a file?

We already know how [sed] can be used to delete a certain line from the output – by using the'd' switch. So if we want to delete the first line the command should be:

$> sed '1 d' file.txt

But the issue with the above command is, it just prints out all the lines except the first line of the file on the standard output. It does not really change the file in-place. So if you want to delete the first line from the file itself, you have two options.   
Either you can redirect the output of the file to some other file and then rename it back to original file like below:

$> sed '1 d' file.txt > new\_file.txt

$> mv new\_file.txt file.txt

Or, you can use an inbuilt [sed] switch '–i' which changes the file in-place. See below:

$> sed –i '1 d' file.txt

### How to remove the last line/ trailer from a file in Unix script?

Always remember that [sed] switch '$' refers to the last line. So using this knowledge we can deduce the below command:

$> sed –i '$ d' file.txt

### How to remove certain lines from a file in Unix?

If you want to remove line <m> to line <n> from a given file, you can accomplish the task in the similar method shown above. Here is an example:

$> sed –i '5,7 d' file.txt

The above command will delete line 5 to line 7 from the file file.txt

### How to remove the last n-th line from a file?

This is bit tricky. Suppose your file contains 100 lines and you want to remove the last 5 lines. Now if you know how many lines are there in the file, then you can simply use the above shown method and can remove all the lines from 96 to 100 like below:

$> sed –i '96,100 d' file.txt # alternative to command [head -95 file.txt]

But not always you will know the number of lines present in the file (the file may be generated dynamically, etc.) In that case there are many different ways to solve the problem. There are some ways which are quite complex and fancy. But let's first do it in a way that we can understand easily and remember easily. Here is how it goes:

$> tt=`wc -l file.txt | cut -f1 -d' '`;sed –i "`expr $tt - 4`,$tt d" test

As you can see there are two commands. The first one (before the semi-colon) calculates the total number of lines present in the file and stores it in a variable called “tt”. The second command (after the semi-colon), uses the variable and works in the exact way as shows in the previous example.

### How to check the length of any line in a file?

We already know how to print one line from a file which is this:

$> sed –n '<n> p' file.txt

Where <n> is to be replaced by the actual line number that you want to print. Now once you know it, it is easy to print out the length of this line by using [wc] command with '-c' switch.

$> sed –n '35 p' file.txt | wc –c

The above command will print the length of 35th line in the file.txt.

### How to get the nth word of a line in Unix?

Assuming the words in the line are separated by space, we can use the [cut] command. [cut] is a very powerful and useful command and it's real easy. All you have to do to get the n-th word from the line is issue the following command:

cut –f<n> -d' '

'-d' switch tells [cut] about what is the delimiter (or separator) in the file, which is space ' ' in this case. If the separator was comma, we could have written -d',' then. So, suppose I want find the 4th word from the below string: “A quick brown fox jumped over the lazy cat”, we will do something like this:

$> echo “A quick brown fox jumped over the lazy cat” | cut –f4 –d' '

And it will print “fox”

### How to reverse a string in unix?

Pretty easy. Use the [rev] command.

$> echo "unix" | rev

xinu

### How to get the last word from a line in Unix file?

We will make use of two commands that we learnt above to solve this. The commands are [rev] and [cut]. Here we go.   
Let's imagine the line is: “C for Cat”. We need “Cat”. First we reverse the line. We get “taC rof C”. Then we cut the first word, we get 'taC'. And then we reverse it again.

$>echo "C for Cat" | rev | cut -f1 -d' ' | rev

Cat

### How to get the n-th field from a Unix command output?

We know we can do it by [cut]. Like below command extracts the first field from the output of [wc –c] command

$>wc -c file.txt | cut -d' ' -f1

109

But I want to introduce one more command to do this here. That is by using [awk] command. [awk] is a very powerful command for text pattern scanning and processing. Here we will see how we may use of [awk] to extract the first field (or first column) from the output of another command. Like above suppose I want to print the first column of the [wc –c] output. Here is how it goes like this:

$>wc -c file.txt | awk ' '' {print $1}'

109

The basic syntax of [awk] is like this:

awk 'pattern space''{action space}'

The pattern space can be left blank or omitted, like below:

$>wc -c file.txt | awk '{print $1}'

109

In the action space, we have asked [awk] to take the action of printing the first column ($1). More on [awk] later.

### How to replace the n-th line in a file with a new line in Unix?

This can be done in two steps. The first step is to remove the n-th line. And the second step is to insert a new line in n-th line position. Here we go.   
Step 1: remove the n-th line

$>sed -i'' '10 d' file.txt # d stands for delete

Step 2: insert a new line at n-th line position

$>sed -i'' '10 i This is the new line' file.txt # i stands for insert

### How to show the non-printable characters in a file?

Open the file in VI editor. Go to VI command mode by pressing [Escape] and then [:]. Then type [set list]. This will show you all the non-printable characters, e.g. Ctrl-M characters (^M) etc., in the file.

### How to zip a file in Linux?

Use inbuilt [zip] command in Linux

### How to unzip a file in Linux?

Use inbuilt [unzip] command in Linux.

$> unzip –j file.zip

### How to test if a zip file is corrupted in Linux?

Use “-t” switch with the inbuilt [unzip] command

$> unzip –t file.zip

### How to check if a file is zipped in Unix?

In order to know the file type of a particular file use the [file] command like below:

$> file file.txt

file.txt: ASCII text

If you want to know the technical MIME type of the file, use “-i” switch.

$>file -i file.txt

file.txt: text/plain; charset=us-ascii

If the file is zipped, following will be the result

$> file –i file.zip

file.zip: application/x-zip

### How to connect to Oracle database from within shell script?

You will be using the same [sqlplus] command to connect to database that you use normally even outside the shell script. To understand this, let's take an example. In this example, we will connect to database, fire a query and get the output printed from the unix shell. Ok? Here we go –

$>res=`sqlplus -s username/password@database\_name <<EOF

SET HEAD OFF;

select count(\*) from dual;

EXIT;

EOF`

$> echo $res

1

If you connect to database in this method, the advantage is, you will be able to *pass Unix side shell variables value to the database*. See below example

$>res=`sqlplus -s username/password@database\_name <<EOF

SET HEAD OFF;

select count(\*) from student\_table t where t.last\_name=$1;

EXIT;

EOF`

$> echo $res

12

### How to execute a database stored procedure from Shell script?

$> SqlReturnMsg=`sqlplus -s username/password@database<<EOF

BEGIN

Proc\_Your\_Procedure(… your-input-parameters …);

END;

/

EXIT;

EOF`

$> echo $SqlReturnMsg

### How to check the command line arguments in a UNIX command in Shell Script?

In a bash shell, you can access the command line arguments using $0, $1, $2, … variables, where $0 prints the command name, $1 prints the first input parameter of the command, $2 the second input parameter of the command and so on.

### How to fail a shell script programmatically?

Just put an [exit] command in the shell script with return value other than 0. this is because the exit codes of successful Unix programs is zero. So, suppose if you write

exit -1

inside your program, then your program will thrown an error and exit immediately.

### How to list down file/folder lists alphabetically?

Normally [ls –lt] command lists down file/folder list sorted by modified time. If you want to list then alphabetically, then you should simply specify: [ls –l]

### How to check if the last command was successful in Unix?

To check the status of last executed command in UNIX, you can check the value of an inbuilt bash variable [$?]. See the below example:

$> echo $?

### How to check if a file is present in a particular directory in Unix?

Using command, we can do it in many ways. Based on what we have learnt so far, we can make use of [ls] and [$?] command to do this. See below:

$> ls –l file.txt; echo $?

If the file exists, the [ls] command will be successful. Hence [echo $?] will print 0. If the file does not exist, then [ls] command will fail and hence [echo $?] will print 1.

### How to check all the running processes in Unix?

The standard command to see this is [ps]. But [ps] only shows you the snapshot of the processes at that instance. If you need to monitor the processes for a certain period of time and need to refresh the results in each interval, consider using the [top] command.

$> ps –ef

If you wish to see the % of memory usage and CPU usage, then consider the below switches

$> ps aux

If you wish to use this command inside some shell script, or if you want to customize the output of [ps] command, you may use “-o” switch like below. By using “-o” switch, you can specify the columns that you want [ps] to print out.

$>ps -e -o stime,user,pid,args,%mem,%cpu

### How to tell if my process is running in Unix?

You can list down all the running processes using [ps] command. Then you can “grep” your user name or process name to see if the process is running. See below:

$>ps -e -o stime,user,pid,args,%mem,%cpu | grep "opera"

14:53 opera 29904 sleep 60 0.0 0.0

14:54 opera 31536 ps -e -o stime,user,pid,arg 0.0 0.0

14:54 opera 31538 grep opera 0.0 0.0

### How to get the CPU and Memory details in Linux server?

In Linux based systems, you can easily access the CPU and memory details from the /proc/cpuinfo and /proc/meminfo, like this:

$>cat /proc/meminfo

$>cat /proc/cpuinfo

Just try the above commands in your system to see how it works

### Beginners UNIX Interview Questions Answers

**1. Write command to list all the links from a directory?**

In this UNIX command interview questions interviewer is generally checking whether user knows basic use of "ls" "grep" and regular expression etc

You can write command like:

ls -lrt | grep "^l"

**2. Create a read-only file in your home directory?**

This is a simple UNIX command interview questions where you need to create a file and change its parameter to read-only by using chmod command you can also change your umask to create read only file.

touch file

chmod 400 file

read more about [**file and directory permission in unix and linux**](http://javarevisited.blogspot.com/2011/11/file-permissions-in-unix-linux-example.html) here.

**3. How will you find which operating system your system is running on in UNIX?**

By using command **"uname -a"** in UNIX

**4. How will you run a process in background? How will you bring that into foreground and how will you kill that process?**

For running a process in background use "&" in command line. For bringing it back in foreground use command "**fg jobid"** and for getting job id you use command "jobs", for killing that process find PID and use kill -9 PID command. This is indeed a good Unix Command interview questions because many of programmer not familiar with background process in UNIX.

**5. How do you know if a remote host is alive or not?**

You can check these by using either **ping** or **telnet** command in UNIX. This question is most asked in various Unix command Interview because its most basic networking test anybody wants to do it.

**6. How do you see command line history in UNIX?**

Very useful indeed, use history command along with[**grep command in unix**](http://javarevisited.blogspot.com/2011/06/10-examples-of-grep-command-in-unix-and.html)to find any relevant command you have already executed. Purpose of this Unix Command Interview Questions is probably to check how familiar candidate is from available tools in UNIX operation system.

**7. How do you copy file from one host to other?**

Many options but you can say by using "**scp**" command. You can also use **rsync** command to answer this UNIX interview question or even **sftp** would be ok.

**8. How do you find which process is taking how much CPU?**

By using "top" command in UNIX, there could be multiple follow-up UNIX command interview questions based upon response of this because “TOP” command has various interactive options to [sort](http://javarevisited.blogspot.com/2011/08/unix-sort-command-example-tutorial.html) result based upon various parameter.

**9. How do you check how much space left in current drive ?**

By using **"df"** command in UNIX. For example **"df -h ."** will list how full your current drive is. This is part of anyone day to day activity so I think this Unix Interview question will be to check anyone who claims to working in UNIX but not really working on it.

**10. What is the difference between Swapping and Paging?**

Swapping:

Whole process is moved from the swap device to the main memory for execution. Process size must be less than or equal to the available main memory. It is easier to implementation and overhead to the system. Swapping systems does not handle the memory more flexibly as compared to the paging systems.

Paging:

Only the required memory pages are moved to main memory from the swap device for execution. Process size does not matter. Gives the concept of the virtual memory. It provides greater flexibility in mapping the virtual address space into the physical memory of the machine. Allows more number of processes to fit in the main memory simultaneously. Allows the greater process size than the available physical memory. Demand paging systems handle the memory more flexibly.

### Intermediate UNIX Interview Questions Answers

**1. What is difference between ps -ef and ps -auxwww?**

This is indeed a good Unix Interview Command Question and I have faced this issue while ago where one culprit process was not visible by execute **ps –ef** command and we are wondering which process is holding the file.

ps -ef will omit process with very long command line while ps -auxwww will list those process as well.

**2. How do you find how many cpu are in your system and there details?**

By looking into file /etc/cpuinfo for example you can use below command:

**cat /proc/cpuinfo**

**3. What is difference between HardLink and SoftLink in UNIX?**

I have discussed this Unix Command Interview questions in my blog post difference between Soft link and Hard link in Unix

**4. What is Zombie process in UNIX? How do you find Zombie process in UNIX?**

When a program forks and the child finishes before the parent, the kernel still keeps some of its information about the child in case the parent might need it - for example, the parent may need to check the child's exit status. To be able to get this information, the parent calls 'wait()'; In the interval between the child terminating and the parent calling 'wait()', the child is said to be a 'zombie' (If you do 'ps', the child will have a 'Z' in its status field to indicate this.)

**Zombie : The process is dead but have not been removed from the process table.**

**5. What is "chmod" command? What do you understand by this line “r-- -w- --x?**

**6. There is a file some where in your system which contains word "UnixCommandInterviewQuestions” How will find that file in Unix?**

By using find command in UNIX for details see here 10 example of using find command in Unix

**7. In a file word UNIX is appearing many times? How will you count number?**

grep -c "Unix" filename

**8. How do you set environment variable which will be accessible form sub shell?**

By using **export** for example export count=1 will be available on all sub shell.

**9. How do you check if a particular process is listening on a particular port on remote host?**

By using telnet command for example “telnet hostname port”, if it able to successfully connect then some process is listening on that port. To read more about telnet read [networking command in UNIX](http://javarevisited.blogspot.com/2010/10/basic-networking-commands-in-linuxunix.html)

**10. How do you find whether your system is 32 bit or 64 bit?**

Either by using **"uname -a"** command or by using "**arch**" command.

### Advanced UNIX Questions and Answers

**1. How do you find which processes are using a particular file?**

By using **lsof** **command** in UNIX. It wills list down PID of all the process which is using a particular file.

**2. How do you find which remote hosts are connecting to your host on a particular port say 10123?**

By using **netstat command** execute netstat -a | grep "port" and it will list the entire host which is connected to this host on port 10123.

**3. What is nohup in UNIX?**

**4. What is ephemeral port in UNIX?**

Ephemeral ports are port used by Operating system for client sockets. There is a specific range on which OS can open any port specified by ephemeral port range.

**5. If one process is inserting data into your MySQL database? How will you check how many rows inserted into every second?**

Purpose of this Unix Command Interview is asking about **"watch" command** in UNIX which is repeatedly execute command provided with specified delay.

**6. There is a file Unix\_Test.txt which contains words Unix, how will you replace all Unix to UNIX?**

You can answer this Unix Command Interview question by using SED command in UNIX for example you can execute **sed s/Unix/UNIX/g fileName.**

**7. You have a tab separated file which contains Name, Address and Phone Number, list down all Phone Number without there name and Addresses?**

To answer this Unix Command Interview question you can either you AWK or CUT command here. CUT use tab as default separator so you can use

**cut -f3 filename.**

**8. Your application home directory is full? How will you find which directory is taking how much space?**

By using disk usage (DU) command in Unix for example du **–sh . | grep G** will list down all the directory which has GIGS in Size.

**9. How do you find for how many days your Server is up?**

By using **uptime** command in UNIX

**10. You have an IP address in your network how will you find hostname and vice versa?**

This is a standard UNIX command interview question asked by everybody and I guess everybody knows its answer as well. By using **nslookup** command in UNIX, you can read more about [**Convert IP Address to hostname in Unix**](http://javarevisited.blogspot.com/2011/09/find-hostname-from-ip-address-to.html) here.

**1. Write command to list all the links from a directory?**

In this UNIX command interview questions interviewer is generally checking whether user knows basic use of "ls" "grep" and regular expression etc

You can write command like:

ls -lrt | grep "^l"

**2. Create a read-only file in your home directory?**

This is a simple UNIX command interview questions where you need to create a file and change its parameter to read-only by using chmod command you can also change your umask to create read only file.

touch file

chmod 400 file

**3. How will you find which operating system your system is running on in UNIX?**

By using command "uname -a" in UNIX

**4. How will you run a process in background? How will you bring that into foreground and how will you kill that process?**

For running a process in background use "&" in command line. For bringing it back in foreground use command "fg jobid" and for getting job id you use command "jobs", for killing that process find PID and use kill -9 PID command. This is indeed a good Unix Command interview questions because many of programmer not familiar with background process in UNIX.

**5. How do you know if a remote host is alive or not?**

You can check these by using either ping or telnet command in UNIX. This question is most asked in various Unix command Interview because its most basic networking test anybody wants to do it.

**6. How do you see command line history in UNIX?**

Very useful indeed, use history command along with grep command in unix to find any relevant command you have already executed. Purpose of this Unix Command Interview Questions is probably to check how familiar candidate is from available tools in UNIX operation system.

**7. How do you copy file from one host to other?**

Many options but you can say by using "scp" command. You can also use rsync command to answer this UNIX interview question or even sftp would be ok.

**8. How do you find which process is taking how much CPU?**

By using "top" command in UNIX, there could be multiple follow-up UNIX command interview questions based upon response of this because “TOP” command has various interactive options to sort result based upon various parameter.

**9. How do you check how much space left in current drive?**

By using "df" command in UNIX. For example "df -h ." will list how full your current drive is. This is part of anyone day to day activity so I think this Unix Interview question will be to check anyone who claims to working in UNIX but not really working on it.

**10. What is the difference between Swapping and Paging?**

**Swapping:**

Whole process is moved from the swap device to the main memory for execution. Process size must be less than or equal to the available main memory. It is easier to implementation and overhead to the system. Swapping systems does not handle the memory more flexibly as compared to the paging systems.

**Paging:**

Only the required memory pages are moved to main memory from the swap device for execution. Process size does not matter. Gives the concept of the virtual memory. It provides greater flexibility in mapping the virtual address space into the physical memory of the machine. Allows more number of processes to fit in the main memory simultaneously. Allows the greater process size than the available physical memory. Demand paging systems handle the memory more flexibly.

**Intermediate UNIX Interview Questions Answers**

**1. What is difference between ps -ef and ps -auxwww?**

This is indeed a good Unix Interview Command Question and I have faced this issue while ago where one culprit process was not visible by execute ps –ef command and we are wondering which process is holding the file.

ps -ef will omit process with very long command line while ps -auxwww will list those process as well.

**2. How do you find how many cpu are in your system and there details?**

By looking into file /etc/cpuinfo for example you can use below command:

cat /proc/cpuinfo

**4. What is Zombie process in UNIX? How do you find Zombie process in UNIX?**

When a program forks and the child finishes before the parent, the kernel still keeps some of its information about the child in case the parent might need it - for example, the parent may need to check the child's exit status. To be able to get this information, the parent calls 'wait()'; In the interval between the child terminating and the parent calling 'wait()', the child is said to be a 'zombie' (If you do 'ps', the child will have a 'Z' in its status field to indicate this.)

Zombie : The process is dead but have not been removed from the process table.

**5. What is "chmod" command? What do you understand by this line “r-- -w- --x?**

6. There is a file some where in your system which contains word "UnixCommandInterviewQuestions” How will find that file in Unix?

**7. In a file word UNIX is appearing many times? How will you count number?**

grep -c "Unix" filename

**8. How do you set environment variable which will be accessible form sub shell?**

By using export for example export count=1 will be available on all sub shell.

**9. How do you check if a particular process is listening on a particular port on remote host?**

By using telnet command for example “telnet hostname port”, if it able to successfully connect then some process is listening on that port. To read more about telnet read networking command in UNIX

**10. How do you find whether your system is 32 bit or 64 bit ?**

Either by using "uname -a" command or by using "arch" command.

**Advanced UNIX Interview Questions and Answers**

**1. How do you find which processes are using a particular file?**

By using lsof command in UNIX. It wills list down PID of all the process which is using a particular file.

**2. How do you find which remote hosts are connecting to your host on a particular port say 10123?**

By using netstat command execute netstat -a | grep "port" and it will list the entire host which is connected to this host on port 10123.

**3. What is ephemeral port in UNIX?**

Ephemeral ports are port used by Operating system for client sockets. There is a specific range on which OS can open any port specified by ephemeral port range.

**4. If one process is inserting data into your MySQL database? How will you check how many rows inserted into every second?**

Purpose of this Unix Command Interview is asking about "watch" command in UNIX which is repeatedly execute command provided with specified delay.

**5. There is a file Unix\_Test.txt which contains words Unix, how will you replace all Unix to UNIX?**

You can answer this Unix Command Interview question by using SED command in UNIX for example you can execute sed s/Unix/UNIX/g fileName.

**6. You have a tab separated file which contains Name, Address and Phone Number, list down all Phone Number without there name and Addresses?**

To answer this Unix Command Interview question you can either you AWK or CUT command here. CUT use tab as default separator so you can use

cut -f3 filename.

**7. Your application home directory is full? How will you find which directory is taking how much space?**

By using disk usage (DU) command in Unix for example du –sh . | grep G will list down all the directory which has GIGS in Size.

**8. How do you find for how many days your Server is up?**

By using uptime command in UNIX

**9. You have an IP address in your network how will you find hostname and vice versa?**

This is a standard UNIX command interview question asked by everybody and I guess everybody knows its answer as well. By using nslookup command in UNIX

--

## Friday, 5 October 2012

### Bash shell scripting interview questions and answer

**1. What is Shell Scripting ?**

Shell scripting, in Linux or Unix, is programming with the shell using which you can automate your tasks. A shell is the command interpreter which is the interface between the User and the kernel. A shell script allows you to submit a set of commands to the kernel in a batch. In addition, the shell itself is very powerful with many properties on its own, be it for string manipulation or some basic programming stuff.

**2. The command "cat file" gives error message "--bash: cat: Command not found". Why?**   
It is because the PATH variable is corrupt or not set appropriately. And hence the error because the cat command is not available in the directories present PATH variable.  
  
  
**3. How to find the length of a string in Linux?**

$ x="welcome"

$ echo ${#x}

7

**4. What are the different timestamps associated with a file?**

* Modification time:- Refers to the time when the file is last modified.
* Access time :- The time when the file is last accessed.
* Changed time :- The time when the attributes of the file are last changed.

**5. How to get the list of files alone in a directory in Linux?**

$ ls -lrt | grep ^-

**6. How to find the last modified file or the newest file in a directory?**

$ ls -lrt | grep ^- | awk 'END{print $NF}'

**7. How to access the 10th command line argument in a shell script in Linux?**

$1 for 1st argument, $2 for 2nd, etc... For 10th argument, ${10}, for 11th, ${11} and so on.

**8. How to find the sum of all numbers in a file in Linux?**

$ awk '{x+=$0}END{print x}' file

**9. How to delete a file which has some hidden characters in the file name?**  
Since the rm command may not be able to delete it, the easiest way to delete a file with some hidden characters in its name is to delete it with the find command using the inode number of the file.

$ ls -li

total 32

9962571 -rw-r--r-- 1 guru users 0 Apr 23 11:35

$ find . -inum 9962571 -exec rm '{}' \;

**10. Using the grep command, how can you display or print the entire file contents?**

$ grep '.\*' file

**11. What is the difference between a local variable and environment variable in Linux?**  
A local variable is the one in which the scope of the variable is only in the shell in which it is defined. An environment variable has scope in all the shells invoked by the shell in which it is defined.  
  
  
**12. What does the 'execute' permission in a directory stand for?**  
Without the execute permission on a directory, the user will not be able to traverse or in other words, do a "cd" to the directory.  
  
**13. How to find the total number of arguments in a shell script in Linux?**  
The shell special variable, $# ,contains the total number of arguments passed to a shell script.  
  
**14. How to remove the Control-M character from a file in Linux?**

$ dos2unix file

**15. In which file should a variable be set in order to make the setting permanent?**  
The variable should be set in the profile file to make the setting permanent. The appropriate profile depends on the default shell being set for the user.  
  
**16. What is a she-bang line in a shell script?**

She-bang line in a shell script is the first line, if present. It starts with '#!' and followed up with a full path of a shell. The shell specified indicates the shell in which this script will be run. The entry of she-bang is not mandatory, however, if present, should be the first line of the script. With a number of shells available and syntax being specific for a given shell, it is always good to specify the she-bang line in a shell script.

**17. A file contains many lines, and each line containing multiple words. How to find out the unique words and the word count of each of the words?**

$ cat file

apple orange

banana apple orange

papaya

$ awk '{for(i=1;i<=NF;i++)a[$i]++;}END{for(i in a){print i, a[i];}}' file

banana 1

apple 2

orange 2

papaya 1

**18. What is an internal command in Linux?**

Internal commands are also called shell built-in commands. Example: cd,fg. Since these are shell built-in, no process is created while executing these commands, and hence are considered to be much faster.

**19. x and y are two variables containing numbers? How to add these 2 numbers?**

$ expr $x + $y

**20. How to add a header record to a file in Linux?**

$ sed -i '1i HEADER' file

**21. How to find the list of files modified in the last 30 mins in Linux?**

$ find . -mmin -30

**22. How to find the list of files modified in the last 20 days?**

$ find . -mtime -20

**23. How to find the files modified exactly before 30minutes?**

$ find . -mmin 30

**24. A string contains a absolute path of a file. How to extract the filename alone from the absolute path in Linux?**

$ x="/home/guru/temp/f1.txt"

$ echo $x | sed 's^.\*/^^'

**25. How to find all the files created after a pre-defined date time, say after 10th April 10AM?**  
This can be achieved in 2 steps:  
1. Create a dummy file with the time stamp, 10th April 10AM.  
2. Find all the files created after this dummy file.

$ touch -t 1004101000 file

$ find . -newer file

**26. How to print the contents of a file line by line in Linux?**

$ while read line

> do

> echo $line

> done < file

**27. The word "Unix" is present in many .txt files which is present across many files and also files present in sub directories. How to get the total count of the word "Unix" from all the .txt files?**

$ find . -name \*.txt -exec grep -c Unix '{}' \; | awk '{x+=$0;}END{print x}'

**28. How to get tomorrow's date in Linux?**

$ date -d "1 day"

**29. How to join all lines in a file using comma?**

$ paste -s -d, file

**30. How to join all lines in a file without any delimiter?**

$ paste -s --delimiter="" file

**31. How to join every 2 lines in a file in Linux?**

$ sed 'N;s/\n//' file

**32. A shell script will ask for 3 inputs. The user will not be physically present to be able to give it manually. How can the script be run without having to manually give the input?**

Put those 3 input values in a file, and make the script to read this file as input. For example:

Assume the 3 values to be : 3, 10 and 20:

$ cat file

3

10

20

and assuming the script is hello.sh, run it like:

./hello.sh < file

**33. How to find the total number of a lines in a file in Linux?**

$ wc -l file | awk '{print $1}'

**34. How to print the first line or the header record in a file?**

$ head -1 file

**35. How to replace all occurrences of "Unix" to "Linux" in a file?**

$ sed 's/Unix/Linux/g' file

**36. How to make the above changes permanent in the file?**

$ sed -i 's/Unix/Linux/g' file

**37. How to replace only the first occurrence of "Unix" to "Linux" in a string in Linux?**

$ sed 's/Unix/Linux/' file

**38. How to replace only the second occurrence of "Unix" to "Linux" in a string in Linux?**

$ sed 's/Unix/Linux/2' file

In fact, to replace nth occurrence of a string in a file, it is:

$ sed 's/Unix/Linux/*n*' file #where n is the nth occurrence

**39. How to add leading zeros to every line in a file in Linux?**

$ sed 's/^/0000/' file

**40. How to add trailing zeros to every line in a file in Linux?**

$ sed 's/$/00/' file

**41. How to get yesterday's date in Linux?**

$ date -d "1 day ago"

**42. I have a file with SQL commands. How can I open a sqlplus session in Linux and run this SQL file?**

$ sqlplus guru/unix11@XE @file.txt

where file.txt is the ASCII file containing the sql instructions.  
  
**43. The ps command will disclose the sqlplus connect string if any sqlplus session is ON. How to prevent the sqlplus connect string from appearing in the ps command in Linux?**  
While connecting to sqlplus, instead of connecting in the normal way, connect as below:

$ sqlplus /nolog

> connect guru/unix11@XE

**44. How to rename a group of files from .txt to .exe in Linux?**

for i in \*.txt

do

x=`basename $i .txt`

mv $i $x.exe

done

**45. After logging in to your account in Linux, you did "cd log". There was no "log" directory under the current directory, still the "cd" command traversed to a log directory under a different location? How it happened?**

It is because the CDPATH variable is set.  
  
  
**46. How to zero pad a number in Linux?**  
Say, to zero pad a number to 4 places:

$ x=20

$ printf "%04d\n" $x

**46. How to find all the .c and .h files in Linux?**

$ find . -name "\*.[ch]"

**47. How to find the list of all the .c files and display only the file name, instead of the default find output which includes the relative path?**

$ find . -name \*.c | sed 's^.\*/^^'

**48. How to copy a file with the same time stamp as the source file in Linux?**

$ cp --preserve=timestamp file1 file2

**49. How to copy a file "file1" to "file2" by passing only one argument to cp command?**

$ cp file{1,2}

|  |  |  |  |
| --- | --- | --- | --- |
| **Question** | **why did you apply to shell** | **Rank** | **Answer Posted By** |
| ***Question Submitted By* :: Guest** | | | |
| This Interview Question Asked @ Shell | | | | | |
| [I also faced this Question!!](http://www.allinterview.com/showqa/Shell-Script/page6.html#aid) | | © ALL Interview .com | |  |  |
| **Answer** | i think there was confusion in this area =)  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **James Galaura** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | i think that shell is the top most industry in india,& it  will be matter of great honour for me if i will be apart of  it.  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Sanjay Sahu** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | With Shell Being an MNC, the opportunities here are far greater than anywhere else. It promises an avenue for me to grow, depending on my strengths. With shell, i know that whichever area of interest i pursue, i will have more opportunities to choose from in my career here.  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Karan** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| |  | | --- | |  | | | | |  |  |
|  | | | |  |  |
| **Question** | **There is a record with fields namely name,roll**  **no.,salary,grade etc.Now,write a script to create a file**  **with multiple records have same combination of fields but**  **with unique roll numbers.The script should work for**  **different names in the input file.** | **Rank** | **Answer Posted By** |  |  |
| ***Question Submitted By* :: Irfan** | | | |  |  |
| This Interview Question Asked @ Wipro , Wipro | | | | | |
| [I also faced this Question!!](http://www.allinterview.com/showqa/Shell-Script/page6.html#aid) | | © ALL Interview .com | |  |  |
| **Answer** | Don't mind my answer is lengthy....  Let file name be employee  name, rollno., salary, grade  cut each of records in each field to temp files as follows:  -----------------------------------------------------------  cut -f1 employee > /sm.tmp.$$  cut -f2 employee > /sn.tmp.$$  cut -f3 employee > /so.tmp.$$  cut -f4 employee > /sp.tmp.$$  Paste them as required in to another file as follows  ----------------------------------------------------  paste /sn.tmp.$$ /sm.tmp.$$  paste /sn.tmp.$$ /so.tmp.$$  paste /sn.tmp.$$ /sp.tmp.$$  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Sabhapathi** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Question** | **There are three departments A,B and C.Write a query to**  **display the names of all the persons( in departments other**  **than A) who are paid higher than the person receiving the**  **lowest salary in DEPT A** | **Rank** | **Answer Posted By** |  |  |
| ***Question Submitted By* :: Irfan** | | | |  |  |
| This Interview Question Asked @ Wipro | | | | | |
| [I also faced this Question!!](http://www.allinterview.com/showqa/Shell-Script/page6.html#aid) | | © ALL Interview .com | |  |  |
| **Answer** | select E.name from department D, emp E  where D.name = E.dname  and E.dname <> 'A'  and E.sal > (select min(E.sal) from department A, emp E  where A.name = E.dname  And E.dname = 'A')  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Ofss** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Question** | **A file has multiple records each having three 30-bit long**  **fields(field1,field2,field3).There is also a lookup**  **file,LOOK\_UP.dat.Now, we need to consider only the**  **last ten digits of field1 and lookup the file LOOK\_UP.dat.**  **If there a match then field2 and field3 should replaced**  **with corresponding data from the lookup file. otherwise**  **that particular record,for which there is no match, should**  **be stored in a seperate file.** | **Rank** | **Answer Posted By** |  |  |
| ***Question Submitted By* :: Irfan** | | | |  |  |
| This Interview Question Asked @ Wipro | | | | | |
| [I also faced this Question!!](http://www.allinterview.com/showqa/Shell-Script/page6.html#aid) | | © ALL Interview .com | |  |  |
| **Answer** | If we would knew the structure of LOOK\_UP.dat (like the  structure of file is mentioned) then the script could be  more precisely written.  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Tazz** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | If we would knew the structure of LOOK\_UP.dat (like the  structure of file is mentioned) then the script could be  more precisely written.  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Tazz** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Question** | **The information of the two files should be redirect to**  **Third file in such a way that, the third file contain the**  **information like this.**  **1st line in third file should be from 1st file**  **2nd line in Third file should be from 2nd file**  **3rd line in Third file should be from 1st file**  **4th line in Third file should be from 2nd file**  **-**  **- so on** | **Rank** | **Answer Posted By** |  |  |
| ***Question Submitted By* :: Praveen** | | | |  |  |
| This Interview Question Asked @ Caritor | | | | | |
| [I also faced this Question!!](http://www.allinterview.com/showqa/Shell-Script/page6.html#aid) | | © ALL Interview .com | |  |  |
| **Answer** | Unix utility sed can be used serve the purpose  Assuming file1.txt and file2.txt are two input files and  file3.txt is the third file.  Contents of the file1.txt  line 1  line 2  line 3  Contents of the file2.txt  LINE 1  LINE 2  LINE 3  After executing below mentioned command  sed -n -e '{  > R file2.txt  > p  > }' file1.txt > file3.txt  The third file file3.txt will be  line 1  LINE 1  line 2  LINE 2  line 3  LINE 3  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Amol** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | paste -d"\n" file1 file2 > file3  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  5  Bottom of Form | **Karthik C N** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Question** | **What is the command for " how many shell is running in your**  **system or how many shell supported by your system " ?.** | **Rank** | **Answer Posted By** |  |  |
| ***Question Submitted By* :: Ravendar** | | | |  |  |
| [I also faced this Question!!](http://www.allinterview.com/showqa/Shell-Script/page6.html#aid) | | © ALL Interview .com | |  |  |
| **Answer** | how many shell supported by your system  check the file /etc/shells  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Sandeep Jaiswal** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | ps| grep -E "\*shh\*" | wc -l  This gives the no of shell currently running.  Though this answer is correct but I don't think this is the  right approach.  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Nayakss** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | > ps -a |grep -E 'sh$' |wc -l  -a is for to list process of all users  -E extended grep to filter the shells (since they end with sh like bash, tcsh, ksh, sh  -$ to match only at the end of the line where the process is printed  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Joker** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Question** | **How to rename all the files in a folder having specific**  **extension?**  **Example: I have some files with extension (.txt) in a folder**  **name 'Test'. I have to rename all the .txt files in a test**  **and its subdirectories to .my extension.** | **Rank** | **Answer Posted By** |  |  |
| ***Question Submitted By* :: Ramit** | | | |  |  |
| This Interview Question Asked @ Interra-IT | | | | | |
| [I also faced this Question!!](http://www.allinterview.com/showqa/Shell-Script/page6.html#aid) | | © ALL Interview .com | |  |  |
| **Answer** | It can be done using the xargs command of Unix ...  cd to the dir containing the \*.txt files  $ls \*.txt | xargs -i mv \{\} \{\}.my  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Jks** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | for file in \*.txt  do  mv $file $file.my  done  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Nagesh** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | Hi Nagesh,  Thanks for the reply but I dont think this will work  as $file= myFile.txt  and mv $file $file.my will rename to myFile.txt.my  I found the solution by extracting basedirectory and  filename then renaming it to req. extension.  thanks  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Ramit** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | for i in `find . -name "\*.txt"`  > do  > mv $i ${i%.txt}.my  > done  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  4  Bottom of Form | **Akshay Telang** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | basename is the appropriate command to cut the unwanted  string from a filename  for example  $basename unix.txt txt #txt stripped off  unix.  -----------------------  for your problem the solution is  for i in `find . -name '\*.txt'`|xargs ls; do  leftname=`basename $i txt`  mv $i ${leftname}doc  done  -----------------------  basename isn't a shell bulletin, but an external command.  The reason why we deal with it here is that it's most  effective when used with the for loop.  Ref: Unix Concepts & Applications - Sumitabha Das  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Aravind S** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | #!/bin/sh  for filename in \*$1\*  do  echo "$filename"  mv -f "$filename" `echo $filename|sed "s/$1/$2/"`  done  run script like  $>scrpt1 .txt .my  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Prav Gir** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | for file in `ls` ; do NEW=`echo $file | sed 's/.txt/.my/g'`  ; mv $file $NEW ; done  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Manuswami** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | The script given below may work properly.  find command is used to search in all the subdirectories.  basename will extract only filename with .txt extension from  absolute path of file.  cut command will print opnly filename without extension.  mv command is uset to rename.  #!/bin/bash  for i in $(find -name \\*.txt)  do  mv $i $(echo `basename $i` | cut -d . -f 1).my  done  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Vipin** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | foreach x (`ls \*.txt')  set y = `echo $x | cut -d . -f 1`  mv y y.my  done  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Asit Pal** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | #!/bin/bash  for x in $(find $1 -name '\*.txt' -type f )  do  OUT=$(echo $x | sed -e "s/\.txt$/.my/")  mv $x $OUT  done  exit 0;  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Geichel** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | for file in `ls \*.txt`; do new\_file=`echo $file | sed  's/txt/my/g'`; mv $file $new\_file; done  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Venugopal Adep** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | its very simple  try this out  rename .txt .my \*.txt  all the files with .txt extension will be converted into .my  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Bindu** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | find . -name "\*.txt" -exec ls {} \; | xargs -n1 -I{} sh -c 'mv "{}" `echo "{}"|basename "{}" .txt`.my'  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Indu Sharma** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | for i in \*  do  p=`basename $i c`  q=$p "txt"  mv $i $q  done  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Sushanta** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Question** | **what are bootlevel in linux?which level is booting by**  **default.** | **Rank** | **Answer Posted By** |  |  |
| ***Question Submitted By* :: Guest** | | | |  |  |
| This Interview Question Asked @ Symphony | | | | | |
| [I also faced this Question!!](http://www.allinterview.com/showqa/Shell-Script/page6.html#aid) | | © ALL Interview .com | |  |  |
| **Answer** | 6 levels, by default which level is in inittab that level  is by default is working.  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Guest** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | 6 levels, by default level is 2(i.e: INIT 2)  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Ravindar** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | 6 levels,by default level is 3  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Sonu** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | 6 levels is correct,XIn Red Hat Linux, the default boot  level is 3, unless booting into an X Windows login, the  default boot level then is 5.  Runlevels Generic mode  0 halt  1 Single user mode  2 Multiuser, without Networking  3 Full multiuser mode  4 unused  5 X11  6 reboot  You can view the settings of default runlevel  in /etc/inittab. The command /sbin/chkconfig --list is used  to display a list of all runlevels and services.  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Jkohlend** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Question** | **How can I Debug a shell scripts and Perl scripting??**  **or How do you debug a shell scripting and perl scripting (**  **at the compile time error or run**  **time error) in Unix environment ?** | **Rank** | **Answer Posted By** |  |  |
| ***Question Submitted By* :: Guest** | | | |  |  |
| [I also faced this Question!!](http://www.allinterview.com/showqa/Shell-Script/page6.html#aid) | | © ALL Interview .com | |  |  |
| **Answer** | We can debug only by $? which contains the returned value  of last executed command.  If $? contains 0 value then command run successfully other  wise error occured while running command  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Manas** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | Using sh -x script argument you can debug(-x option).  Bash shell offers debugging options which can be turn on or  off using  set command.  => set -x : Display commands and their arguments as they are  executed.  => set -v : Display shell input lines as they are read.  Hope this answers your question.  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  5  Bottom of Form | **Hema** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | In addition to Hema's ans :  We can have some $? statement, echo statements embadded in  script.  In pern we can use -w option to print the warning or erros  generated during compilation, can make use os strict to force  to use good programming practice...  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Sushil** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | We can also use below command to debug the script  bash -xv <scriptname.sh>  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Ram** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Question** | **how did u debugging in unix/ linux platform ??( project**  **Related)** | **Rank** | **Answer Posted By** |  |  |
| ***Question Submitted By* :: Guest** | | | |  |  |
| This Interview Question Asked @ Symphony | | | | | |
| [I also faced this Question!!](http://www.allinterview.com/showqa/Shell-Script/page6.html#aid) | | © ALL Interview .com | |  |  |
| **Answer** | For gdb in linux/unix Gnu Debugger is available. Please  read the mannuals for gdb.  man gdb  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Siba Sankar Nayak** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | If it is a shell script use #!/bin/ksh -x  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Santana20142003** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Question** | **How to handle the delimiter unit seperator in Unix** | **Rank** | **Answer Posted By** |  |  |
| ***Question Submitted By* :: Devi** | | | |  |  |
| [I also faced this Question!!](http://www.allinterview.com/showqa/Shell-Script/page6.html#aid) | | © ALL Interview .com | |  |  |
| **Answer** | create script test.txt with | seperated  The below commands can be used to get the fields values:  awk -F"|" '{print $1}' test.txt  cat sri.txt | cut -d"|" -f1  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Srikanth** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Question** | **If one dont know how to create a script then how he/she can**  **use the QTP?** | **Rank** | **Answer Posted By** |  |  |
| ***Question Submitted By* :: Dart** | | | |  |  |
| [I also faced this Question!!](http://www.allinterview.com/showqa/Shell-Script/page6.html#aid) | | © ALL Interview .com | |  |  |
| **Answer** | use record/playback feature but won't be able to go too far  with it  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **P** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Question** | **Hi Friends,**  **I am currently Undergoing Course On Testing.I am Planning**  **To Keep Fake Resume.Can any One tell me the ways to Prepare**  **i.e, Real Time experience For Manual Testing.**  **With Regards,**  **Vikram** | **Rank** | **Answer Posted By** |  |  |
| ***Question Submitted By* :: Vikram.bheem** | | | |  |  |
| [I also faced this Question!!](http://www.allinterview.com/showqa/Shell-Script/page6.html#aid) | | © ALL Interview .com | |  |  |
| **Answer** | this site is for knowledge sharing friend. not for fake  things. use this site as it meant for.  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Raviabcdefg** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Question** | **Why we are writting shell scripts?**  **Plz if possible explain it briefly.** | **Rank** | **Answer Posted By** |  |  |
| ***Question Submitted By* :: Sahoo.sasmita** | | | |  |  |
| This Interview Question Asked @ ITC-Infotech | | | | | |
| [I also faced this Question!!](http://www.allinterview.com/showqa/Shell-Script/page6.html#aid) | | © ALL Interview .com | |  |  |
| **Answer** | 1>Interactive mood:--  As Shell scripting is working like as interpreter (not  likely compiler) in a interactive mood.  2> Less time:--  it will take less time than C or C++ or other file to process.  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Amaresh\_83** |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | Shell is a command interpeter. If we want to run multiple  commands at a time through a script file then it is called  shell scripts.  Generally we will write these to set certain environmental  variables in unix and run some other programs.  The shell scripts utility programs to perform certain tasks  in unix environment.  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  5  Bottom of Form | **Ashok**  [QSS] |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | Shell is command line interpreter, in shell script we can  write multiple commands to do particular task by executing  the shell script.  shell is used to set our environment.  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Nagireddyunix**  [QSS] |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | Another one would be to automate routinary/housekeeping  tasks especially those long commands we keep on typing  everyday or weekly - we could write all those series of  commands in a script and just run that script anytime we  need it instead of writing all those hard to memorize  commands all the time.  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Jrgalaura**  [QSS] |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | Can any one explain Automation using Shell script  [http://www.allinterview.com/images/top.gif](javascript:history.back()) | Top of Form  0  Bottom of Form | **Sri**  [QSS] |  |  |
|  | | | |  |  |
|  | | | |  |  |
| **Answer** | Shell script allows the user to perform his day-to-day  tasks sooner.Even a Shell Environment doesnt need a  graphical environment to write a script. So what happens is  it will be more easy for programmer to write shell scripts  on Command line environment , where it doesnt costs more  resources comparable to Graphical environment.    For example : Backup Scripts , Keeping Log files of system.  shell scripts makes you the task easier !!  And to automate the script in Linux we perform cron tasks.  Once cron has to be set. It performs operations automatically |  |  |  |  |

## Friday, 24 August 2012

### Interview Questions and Answers Unix & Shell

**Q. How will you kill all the child process of a parent process?**

For example a.sh is parent shell and it execute many other shell files then first need to kill all the child and then main shell. How can we do this in unix shell scripting?

Q.**What is autosys? Why do we need to use autosys?**

Ans - Autosys is a job scheduling software like Control - M and cron, with the help of autosys we can define the run time,day,date,week and which script or program needs to be run.   
In real business, you need to fire jobs not just based on scheduled time, but also based on events (like an arrival of a file), and, based on the success or failure of other jobs. Autosys is a total job scheduling solution that allows you to define these events, dependencies, time schedules, alerts, etc, making it a complete data center automation tool.   
The main advantage of using autosys w.r.t crontab is that it is has a Java front end too, so a person do not need to be a Unix champ to create or change a job in autosys.

**Q. How to schedule a job in autosys?**

Ans - There are the two methods you can use to create job definitions:   
¦ Using the AutoSys Graphical User Interface (GUI).   
¦ Using the AutoSys Job Information Language (JIL) through a command-line interface.

**Q. Have you ever used any other scheduling tool, other than Cron Tab?**

Ans – Yes I have used Autosys and Control M.

**Q. How can you identify the process currently running?**

Ans - ps -eaf | grep RU -- will list out the Processes currently running in the System

Q. **Explain about sourcing commands?**

Sourcing commands help you to execute the scripts within the scripts. For example sh command makes your program to run as a separate shell. .command makes your program to run within the shell. This is an important command for beginners and for special purposes.

Q. How do you read arguments in a shell program - $1, $2 ?

A. Shell script accepts parameters in following format...

$1 : first

$2 : second....so on upto

$9 : 9th param

whereas $0 : gives script/function name

If your script has more than 9 params then accept in following way...

${12} : 12th param

${18} : 18th param

Q. What are the different kinds of loops available in shell script?

A. Broadly categorised in 3

for

while

until

Q. What does $# stand for?

A. $# returns the number of parameters that are passed to a shell script

$? returns the exit code of the last executed command (0 : Successful, 1 or other: Failed)

Q. How do u open a read only file in Unix?

A. "vi -R filename"

In a shell script you can open a file in read only mode by using O\_RDONLY

Syntax - open(<filename> O\_RDONLY)

more <file>

cat <file>

less <file>

Q. What is the difference between a shell variable that is exported and the one that is not exported?

A. exported variable is visible to the child processes while the normal variables are not.

Export is to set the vaibales globally

Q. If you have a string "one two three", Which shell command would you use to extract the strings?

A. echo "one two three" | awk '{print $1 $2 $3}'

echo $string | cut -d -f1

echo $string | cut -d -f2

echo $string | cut -d -f3

Q. How do you schedule a command to run at 4:00 every morning?

A. Schedule the Job using crontab..From the command prompt perform a man on crontab and then type in crontab-e to create a new job

Crontab format

\* \* \* \* \* <command>

< minute> <hour> <day>< month> <day\_of\_the\_week> command

< 0-59> <0-23> <1-31>< 1-12> <0-7> command

Q. How will you list only the empty lines in a file (using grep)?

A. We can do this efficiently through awk script

awk '{

if (NF 0)

{

print "Here comes the empty line"

print $0

}

}' filename

NF number of fields in the current line

$0 current line

Q. How would you get the character positions 10-20 from a text file?

A. cat filename.txt | cut -c 10-20

TAB is default field separator for in a file for cut command.

cut -d"," -f2 cdr.txt

-d is for delimeter.

-f is for extracting the fields.

cut [OPTION]... [FILE]...

DESCRIPTION

Print selected parts of lines from each FILE to standard output.

Mandatory arguments to long options are mandatory for short options too.

-b, --bytes=LIST

output only these bytes

-c, --characters=LIST

output only these characters

-d, --delimiter=DELIM

use DELIM instead of TAB for field delimiter

-f, --fields=LIST

output only these fields; also print any line that contains no delimiter character, unless the -s option is specified.

Q. How would you print just the 25th line in a file (smallest possible script please)?

A. head -25 cdr.txt | tail -1

sed -n '25p' cdr.txt

using awk we can do the same also.

Q. How would you replace the n character in a file with some xyz?

A. sed 's/n/xyz/g' filename > new\_filename

Q. What is chmod in unix?

A. This is for file permissions on unix- first is for self, second for group, and third for others.

And this rwx for each with 421.

Triplet for u: rwx => 4 + 2 + 1 = 7

Triplet for g: r-x => 4 + 0 + 1 = 5

Tripler for o: r-x => 4 + 0 + 1 = 5

Which makes : 755

UNIX chmod command

To change permissions on a UNIX file or directory, use the chmod command. chmod category+permissions filename category can be omitted, in which case the permissions specified are granted to all three categories (user, group and other). If + is replaced by a -, then those permissions are taken away from the categories.

As an example, consider this statement:

chmod o+r data

This grants other read permission to the file data. The command

chmod +x data

grants everyone (user, group and other) execute permission, and the command

chmod g+rwx data

gives category group read, write and execute permission.

Permissions change cumulatively, in that the commands

chmod +r file

chmod +w file

chmod +x file

are equivalent to

chmod +rwx file

Q. What is the difference between a 'thread' and a 'process'?

A. The key difference is that processes are fully isolated

from each other; threads share (heap) memory with other

threads running in the same application.

OR

A process is an instance of an running application.

And a Thread is the execution stream of the process.

A process can have multiple threads.

OR

Changes to the main thread (cancellation, priority

change, etc.) may affect the behavior of the other threads

of the process; changes to the parent process does not

affect child processes. If we consider running a word

processing program to be a process, then the auto-save and

spell check features that occur in the background are

different threads of that process which are all operating

on the same data set (your document).

OR

A process runs in it’s own memory area, where the thread always run in the parent process memory.

This is just like parent child.

Q. Write a shell script to identify the given string is palindrome or not?

A. var hai

var1 `echo $var|rev`

if [ $var ! $var1 ]

then

echo $var is not palindrome.

else

echo $var is palyndrome

fi

Q. How Connect to a Database in Shell Programming?Please tell me Step by Step?

A. Hi

In order to connect to a database using a shell script it is always necessary to set the environment first and then connect to the database and execute whatever the statements u need to and come out of the script gracefully.

ex:

#!/bin/sh

export ORACLE\_SID $1

export ORACLE\_HOME more /etc/oratab|grep -v '#'|grep -i $1|cut -d":" -f2

export LD\_LIBRARY\_PATH $ORACLE\_HOME/lib:/usr/lib

export PATH $ORACLE\_HOME/bin:$HOME/bin:$PATH

sqlplus -s sys/<passwd> as sysdba <EOF

select name from v$database;

archive log list;

exit

EOF

The above script echoes the name of the database you are connected along with the archive log mode information.

Q. What is this line in the shell script do ?#!/bin/ksh?

A. This line is called as "Hash Bang" Statement.

It gives the environment or the shell in which your script would be run

Q. What is the basic difference u find between a shell script and perl.I mean the advantages of one over the other?

A. 1) PERL scripts can be used on both UNIX and windows systems unless some OS specific commands are used. But the same case is not with Shell scripting.

2) PERL scripts are used for web based applications but shell scripts can not be used for the same purpose.

3) PERL modules gives PERL extra edge over Shell scripts. PERL modules are extensive and can be used for n number of purposes.

4) Data base drivers are available for perl also.

Q. How to take input values from the user?

A. 1. using read command

2. reading input from a file

Q. How to compress files by using shell scripting?

A.

Command Syntax File Produced

gzip gzip .gz

zip zip .zip

compress compress .Z

pack pack .z

Gzip, zip, compress, pack are used for compression of files.

Tar is also used to compress.

Q.how todelete a word from a file using shell scripting???

A. sed -e 's/word//g' filename

Q. How do you search the string for vowel's occurrence and number of occurrences of each vowel.

A. grep -io [aeiou]filename | wc –w

Q. You havecurrent directorycontaining set of directories which contain files.  
One file can reside in many directories.  
Write script which returns number of unique file names in  
all the subdirectories of a current dir.

A. ls -R|sort|uniq

Q. What are the different types of shells available in UNIX?

A. bourne (sh)  
c shell (csh)  
korn (ksh)  
bourne again shell (bash)  
TC shell (tcsh)

Q. What are the different security mechanisms available in UNIX?

A. Unix is having 3 ways of Security Mechanism:  
1. By granting or revoking File permissions. Owner or Admin can change permissions to be given to group or others by using chmod command in Unix.  
2. Login is restricted using login credentials ( User name and Password ).  
3. Password is kept in encrypted format in the file /etc/passwd

Q. What are the type of files in unix?

A. File is of type c:

b block (buffered) special

c character (unbuffered) special

d directory

p named pipe (FIFO)

f regular file

l symbolic link

s socket

D door (Solaris)

Q. what is find command?

A. find is used for searching dir for files.

Examples –

find /oracle/oracle9i/Import -perm 755-name "\*.htm\*" -mtime-1 -size 100k

Q. What is a named pipe?

A **named pipe** is a special file that is used to transfer data   
between unrelated processes. One (or more) processes write to it, while   
another process reads from it. Named pipes are visible in the file   
system and may be viewed with `ls'like any other file. (Named   
pipes are also called **fifo**s; this term stands for `First In, First   
Out'.)

To create a named pipe, use the Unix command mknod(1)or on some systems, mkfifo(1).These may not be in your normal path.

# system return val is backwards, so && not ||

#

$ENV{PATH} .= ":/etc:/usr/etc";

if ( system('mknod', $path, 'p')

&& system('mkfifo', $path) )

{

die "mk{nod,fifo} $path failed;

}

Afifo is convenient when you want to connect a process to an unrelated one. When you open a fifo, the program will block until there's something on the other end.

For example, let's say you'd like to have your .signature file be a named pipe that has a Perl program on the other end. Now every time any program (like a mailer, news reader, finger program, etc.) tries to read from that file, the reading program will block and your program will supply the new signature. We'll use the pipe-checking file test**-p** to find out whether anyone (or anything) has accidentally removed our fifo.

chdir; # go home

$FIFO = '.signature';

$ENV{PATH} .= ":/etc:/usr/games";

while (1) {

unless (-p $FIFO) {

unlink $FIFO;

system('mknod', $FIFO, 'p')

&& die "can't mknod $FIFO: $!";

}

# next line blocks until there's a reader

open (FIFO, "> $FIFO") || die "can't write $FIFO: $!";

print FIFO "John Smith (smith\@host.org)\n", `fortune -s`;

close FIFO;

sleep 2; # to avoid dup signals

}

Q. What is relative path and absolute path?

Absolute path : Exact path from root directory.

Relative path : Relative to the current path.

Q. Explain kill() and its possible return values?

There are four possible results from this call:

‘kill()’ returns 0. This implies that a process exists with the given PID, and the system would allow you to send signals to it. It is system-dependent whether the process could be a zombie.

‘kill()’ returns -1, ‘errno == ESRCH’ either no process exists with the given PID, or security enhancements are causing the system to deny its existence. (On some systems, the process could be a zombie.)

‘kill()’ returns -1, ‘errno == EPERM’ the system would not allow you to kill the specified process. This means that either the process exists (again, it could be a zombie) or draconian security enhancements are present (e.g. your process is not allowed to send signals to \*anybody\*).

‘kill()’ returns -1, with some other value of ‘errno’ you are in trouble! The most-used technique is to assume that success or failure with ‘EPERM’implies that the process exists, and any other error implies that it doesn't.

An alternative exists, if you are writing specifically for a system (or all those systems) that provide a ‘/proc’ filesystem: checking for the existence of ‘/proc/PID’may work.

Q. What is a pipe and give an example?

A pipe is two or more commands separated by pipe char '|'. That tells the shell to arrange for the output of the preceding command to be passed as input to the following command.

Example : ls -l | pr

The output for a command ls is the standard input of pr.

When a sequence of commands are combined using pipe, then it is called pipeline.

Q. How to terminate a process which is running and the specialty on command kill 0?

To find out weather a process is running or not :

We can use ps command with gives the current running process.

And after that we can apply grep command to filter out the process name.

Like to find out sshd process is running or not : $ ps -ewwo pid,args | grep [s]sh

Output :

5341 /usr/sbin/sshd

5864 /usr/bin/ssh-agent x-session-manager

6289 ssh oldbox

7126 ssh admin@core.r1.vsnl.router

Where

* **ps** : Command name
* **-ewwo pid,args** : -e option force to select all running processes. -o option is used to specify user-defined format. In our case we are forcing to display only program pid and its arguments. Finally -w option specifies wide output. Use this option twice for unlimited width.
* **grep [s]sh** : We are just filtering out sshd string

**Once we got the process id we can kill the process with kill command:**

Syntax is : kill -NPID

Where,

* **N** is a signal number
* **PID** is the Process Identification Number. If you do not know the PID, it can be learned through the ps command.

### Some of the more commonly used signals:

|  |  |
| --- | --- |
| **signal #** | **Usage** |
| 1 | HUP (hang up) (Default) |
| 2 | INT (interrupt) |
| 3 | QUIT (quit) |
| 6 | ABRT (abort) |
| 9 | KILL (non-catchable, non-ignorable kill) |
| 14 | ALRM (alarm clock) |
| 15 | TERM (software termination signal) |

**Kill 0 - kills all processes in your system except the login shell.**

Q. What is redirection?

Directing the flow of data to the file or from the file for input or output.

Example : ls > wc

Q. What are shell variables?

Shell variables are special variables, a name-value pair created and maintained by the shell.

Example: PATH, HOME, MAIL and TERM

Q. How to switch to a super user status to gain privileges?

Use ‘su’ command. The system asks for password and when valid entry is made the user gains super user (admin) privileges.

Q. How does the kernel differentiate device files and ordinary files?

Kernel checks 'type' field in the file's inode structure.

Q. How many prompts are available in a UNIX system?

Two prompts, PS1 (Primary Prompt), PS2 (Secondary Prompt).

Q. Name the data structure used to maintain file identification?

‘inode’, each file has a separate inode and a unique inode number.

Q. Is it possible to count number char, line in a file; if so, How?

Yes, wc-stands for word count.

wc -c for counting number of characters in a file.

wc -l for counting lines in a file.

Q. Is ‘du’ a command? If so, what is its use?

Yes, it stands for ‘disk usage’.With the help of this command you can find the disk capacity and free space of the disk.

Q. What is the use of the command "ls-x chapter[1-5]"?

ls stands for list; so it displays the list of the files that starts with 'chapter' with suffix '1' to '5', chapter1, chapter2, and so on.

Q. Is it possible to restrict incoming message?

Yes, using the ‘mesg’ command.

Q. Is it possible to create new a file system in UNIX?

Yes, ‘mkfs’ is used to create a new file system.

Q. What will the following command do?

$ echo \*

It is similar to 'ls' command and displays all the files in the current

directory.

Q. Write a command to display a file’s contents in various formats?

$od -cbd file\_name

c - character, b - binary (octal), d-decimal, od=Octal Dump.

Q. Which command is used to delete all files in the current directory and all its sub-directories?

rm -r \*

Q. Write a command to kill the last background job?

Kill $!

Q. What is the difference between cat and more command?

Cat displays file contents. If the file is large the contents scroll off the screen before we view it. So command 'more' is like a pager which displays the contents page by page.

Q. What is the use of ‘grep’command?

‘grep’ is a pattern search command. It searches for the pattern, specified in the command line with appropriate option, in a file(s).

Syntax : grep

Example : grep 99mx mcafile

Q. What difference between cmp and diff commands?

cmp - Compares two files byte by byte and displays the first mismatch

diff - tells the changes to be made to make the files identical

Q. Explain the steps that a shell follows while processing a command?

After the command line is terminated by the key, the shel goes ahead with processing the command line in one or more passes. The sequence is well defined and assumes the following order.

Parsing: The shell first breaks up the command line into words, using spaces and the delimiters, unless quoted. All consecutive occurrences of a space or tab are replaced here with a single space.

Variable evaluation: All words preceded by a $ are avaluated as variables, unless quoted or escaped.

Command substitution: Any command surrounded by backquotes is executed by the shell which then replaces the standard output of the command into the command line.

Wild-card interpretation: The shell finally scans the command line for wild-cards (the characters \*, ?, [, ]). Any word containing a wild-card is replaced by a sorted list of filenames that match the pattern. The list of these filenames then forms the arguments to the command.

PATH evaluation: It finally looks for the PATH variable to determine the sequence of directories it has to search in order to hunt for the command.

Q. What is the difference between > and >> redirection operators ?

is the output redirection operator when used it overwrites while >> operator appends into the file.

Q. Which of the following commands is not a filter?(a)man , (b) cat , (c) pg , (d) head?

Ans: man

A filter is a program which can receive a flow of data from std input, process (or filter) it and send the result to the std output.

Q. Construct pipes to execute the following jobs?

1. Output of who should be displayed on the screen with value of total number of users who have logged in displayed at the bottom of the list.

2. Output of ls should be displayed on the screen and from this output the lines containing the word ‘poem’should be counted and the count should be stored in a file.

3. Contents of file1 and file2 should be displayed on the screen and this output should be appended in a file .

From output of ls the lines containing‘poem’ should be displayed on the screen along with the count.

4. Name of cities should be accepted from the keyboard . This list should be combined with the list present in a file. This combined list should be sorted and the sorted list should be stored in a file ‘newcity’.

5. All files present in a directory dir1 should be deleted any error while deleting should be stored in a file ‘errorlog’.

Q. How to find free space in unix/linux?

df -h

Q. What is the difference between soft link and hard link in unix operating system ?

Hard Links :

1. All Links have same inode number.

2.ls -l command shows all the links with the link column(Second) shows No. of links.

3. Links have actual file contents

4.Removing any link just reduces the link count but doesn't affect other links.

Soft Links(Symbolic Links) :

1.Links have different inode numbers.

2. ls -l command shows all links with second column value 1 and the link points to original file.

3. Link has the path for original file and not the contents.

4.Removing soft link doesn't affect anything but removing original file the link becomes dangling link which points to nonexistant file.

Q. What are the differences between Shared and Dynamic libraries?

Shared libraries are loaded into memory before compilation during parsing whereas dynamic libraries are loaded during compilation time itself.

Q. How would you create shared and dynamic libraries?

Well shared libraries have 2 types

1) Static

2) Dynamic.

u can create library by

ar cr -o sharedobj.a file1.o file2.o

while file1 and file2 are headfiles (obj)

now put this sharedobj.a into /usr/lib directory

**Q. What type of scheduling is used in Unix?**

Multi Level Feedback Queue Scheduling with each queue in round robin

Q. What is stty used for?

stty is used to set your terminal options. Following are some of the typical examples1. stty echo-iuclc The above command will switch off character display on your screen and ignore any case differences (ignore upper case and lower case). This can be used in situation where password is read and should not get printed on screen2. stty -erase '^H' This will set the erase character for your terminal3. stty -intr '^c' This will set the execution interrupt as control+c key 4. stty -eof '^d' This will set the end of file character as control+d key

Q. How to read and write of a open file?

Read from file:

cat file\_name

Or on terminal pagewise read as :

more file\_name.

Write to file :

echo "`cmd`" > file\_name.

Where cmd= any executable UNIX command.

> = redirect the o/p.

>> = append to file\_name.

Q. What are the directory commands available in UNIX programming?

mkdir to create directory

cd to change directory

rmdir to remove directory (must be empty)

pwd to show present working directory

ls to list contents of a directory

**Q. What is the significance of the “tee” command?**

It reads the standard input and sends it to the standard output while redirecting a copy of what it has read to

the file specified by the user.

**Q. Write a command to display a file’s contents in various formats?**

$od-cbd file\_name

c- character, b - binary (octal), d-decimal, od=Octal Dump.

**Q. How does the kernel differentiate device files and ordinary files?**

Kernel checks 'type' field in the file's inode structure.

## Q. In Unix, what is a symbolic link, and how do I create one?

A symbolic link, also termed a soft link, is a special kind of file that points to another file, much like a shortcut in Windows or a Macintosh alias. Unlike a hard link, a symbolic link does not contain the data in the target file. It simply points to another entry somewhere in the file system. This difference gives symbolic links certain qualities that hard links do not have, such as the ability to link to directories, or to files on remote computers networked through NFS. Also, when you delete a target file, symbolic links to that file become unusable, whereas hard links preserve the contents of the file.

To create a symbolic link in Unix, at the Unix prompt, enter:

ln -s source\_file myfile

Replacesource\_filewith the name of the existing file for which you want to create the symbolic link (this file can be any existing file or directory across the file systems). Replace myfilewith the name of the symbolic link. The lncommand then creates the symbolic link. After you've made the symbolic link, you can perform an operation on or execute myfile, just as you could with the source\_file. You can use normal file management commands (e.g., cp,rm) on the symbolic link.

**Note:**If you delete the source file or move it to a different location, your symbolic file will not function properly. You should either delete or move it. If you try to use it for other purposes (e.g., if you try to edit or execute it), the system will send a "file nonexistent" message.

# Q.Where is Socket used?

A Unix Socket is used in a client server application frameworks. A server is a process which does some function on request from a client. Most of the application level protocols like FTP, SMTP and POP3 make use of Sockets to establish connection between client and server and then for exchanging data.

# Socket Types:

There are four types of sockets available to the users. The first two are most commenly used and last two are rarely used.

* **Stream Sockets:** Delivery in a networked environment is guaranteed. If you send through the stream socket three items "A,B,C", they will arrive in the same order - "A,B,C". These sockets use TCP (Transmission Control Protocol) for data transmission. If delivery is impossible, the sender receives an error indicator. Data records do no have any boundaries.
* **Datagram Sockets:** Delivery in a networked environment is not guaranteed. They're connectionless because you don't need to have an open connection as in Stream Sockets - you build a packet with the destination information and send it out. They use UDP (User Datagram Protocol).
* **Raw Sockets:** provides users access to the underlying communication protocols which support socket abstractions. These sockets are normally datagram oriented, though their exact characteristics are dependent on the interface provided by the protocol. Raw sockets are not intended for the general user; they have been provided mainly for those interested in developing new communication protocols, or for gaining access to some of the more esoteric facilities of an existing protocol.
* **Sequenced Packet Sockets:** They are similar to a stream socket, with the exception that record boundaries are preserved. This interface is provided only as part of the Network Systems (NS) socket abstraction, and is very important in most serious NS applications. Sequenced-packet sockets allow the user to manipulate the Sequence Packet Protocol (SPP) or Internet Datagram Protocol (IDP) headers on a packet or a group of packets either by writing a prototype header along with whatever data is to be sent, or by specifying a default header to be used with all outgoing data, and allows the user to receive the headers on incoming packets.

Q. What is door in unix?

Ans. - Doors are an [inter-process communication](http://en.wikipedia.org/wiki/Inter-process_communication)facility for [Unix](http://en.wikipedia.org/wiki/Unix)computer systems. They provide a form of [procedure call](http://en.wikipedia.org/wiki/Subroutine). Doors were developed by [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems)as a core part of the [Spring operating system](http://en.wikipedia.org/wiki/Spring_%28operating_system%29), then added to [Solaris](http://en.wikipedia.org/wiki/Solaris_Operating_System)

**Crontab – Quick Reference**

**Setting up cron jobs in Unix and Solaris**

cron is a unix, solaris utility that b which tasks run automatically in the background at regular intervals. Crontab (CRON TABle) is a file which contains the schedule of cron entries to be run and at specified times.

This document covers following aspects of Unix cron jobs  
1. Crontab Restrictions  
2. Crontab Commands  
3. Crontab file – syntax  
4. Crontab Example  
5. Crontab Environment  
6. Disable Email  
7. Generate log file for crontab activity

**1. Crontab Restrictions**  
You can execute crontab if your name appears in the file /usr/lib/cron/cron.allow. If that file does not exist, you can use crontab if your name does not appear in the file /usr/lib/cron/cron.deny. **If only cron.deny exists and is empty, all users can use crontab.** If neither file exists, only the root user can use crontab.

**2. Crontab Commands**

export EDITOR=vi ;to specify a editor to open crontab file.

crontab -e Edit your crontab file, or create one if it doesn’t already exist.  
crontab -l Display your crontab file.  
crontab -r Remove your crontab file.  
crontab -v Display the last time you edited your crontab file. (This option is only available on a few systems.)

**3. Crontab file**  
A crontab file has five fields for specifying day , date and time followed by the command to be run at that interval.

|  |
| --- |
| \* \* \* \* \* command to be executed  - - - - -  | | | | |  | | | | +----- day of week (0 - 6) (Sunday=0)  | | | +------- month (1 - 12)  | | +--------- day of month (1 - 31)  | +----------- hour (0 - 23)  +------------- min (0 - 59) |

\* in the value field above means all legal values as in braces for that column.  
The value column can have a \* or a list of elements separated by commas. An element is either a number in the ranges shown above or two numbers in the range separated by a hyphen (meaning an inclusive range).

**4. Crontab Example**  
A line in crontab file like below removes the tmp files from /home/someuser/tmp each day at 6:30 PM.

30 18 \* \* \* rm /home/someuser/tmp/\*

Changing the parameter values as below will cause this command to run at different time schedule below :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| min | hour | day/month | month | day/week | Execution time |
| 30 | 0 | 1 | 1,6,12 | \* | – 00:30 Hrs on 1st of Jan, June & Dec. |
|  | | | | | |
| 0 | 20 | \* | 10 | 1-5 | –8.00 PM every weekday (Mon-Fri) only in Oct. |
|  | | | | | |
| 0 | 0 | 1,10,15 | \* | \* | – midnight on 1st ,10th & 15th of month |
|  | | | | | |
| 5,10 | 0 | 10 | \* | 1 | – At 12.05,12.10 every Monday & on 10th of every month |
| : | | | | | |

Note : If you inadvertently enter the crontab command with no argument(s), do not attempt to get out with Control-d. This removes all entries in your crontab file. Instead, exit with Control-c.

**5. Crontab Environment**  
cron invokes the command from the user’s HOME directory with the shell, (/usr/bin/sh).  
cron supplies a default environment for every shell, defining:  
HOME=user’s-home-directory  
LOGNAME=user’s-login-id  
PATH=/usr/bin:/usr/sbin:.  
SHELL=/usr/bin/sh

**6. Disable Email**  
By default cron jobs sends a email to the user account executing the cronjob. If this is not needed put the following command At the end of the cron job line .

> /dev/null 2>&1

**7. Generate log file**  
To collect the cron execution execution log in a file :

30 18 \* \* \* rm /home/someuser/tmp/\* > /home/someuser/cronlogs/clean\_tmp\_dir.log

**Basic UNIX commands**

## Files

**Ls**command   
ls command is most widely used command and it displays the contents of directory.

options

ls will list all the files in your home directory, this command has many options.

ls -l will list all the file names, permissions, group, etc in long format.

ls -a will list all the files including hidden files that start with . .

ls -lt will list all files names based on the time of creation, newer files bring first.

ls -Fxwill list files and directory names will be followed by slash.

ls -Rwill lists all the files and files in the all the directories, recursively.

ls -R | more will list all the files and files in all the directories, one page at a time.

* **more *filename*** --- shows the first part of a file, that can fit on one screen. Just hit the space bar to see more or **q** to quit.
* **emacs *filename***--- is an editor that lets you create and edit a file.
* **mv *filename1 filename2*** --- moves a file
* **cp *filename1 filename2*** --- copies a file
* **rm *filename***--- removes a file. It is wise to use the option rm -i, which will ask you for confirmation before actually deleting anything.
* **diff *filename1 filename2*** --- compares files, and shows where they differ
* **wc *filename*** --- tells you how many lines, words, and characters there are in a file -l for lines, -c for char, -w for words.
* **chmod *options filename*** --- lets you change the read, write, and execute permissions on your files. The default is that only you can look at them and change them. For example, **chmod o+r *filename*** will make the file readable for everyone, and **chmod o-r *filename*** will make it unreadable for others again.
* File Compression
  + **gzip *filename*** --- compresses files. Usually text files compress to about half their original size, but it depends very much on the size of the file and the nature of the contents. There are other tools for this purpose, too (e.g. **compress**), but gzip usually gives the highest compression rate. Gzip produces files with the ending '.gz' appended to the original filename.
  + **gunzip *filename*** --- uncompresses files compressed by gzip.
  + **gzcat *filename*** --- lets you look at a gzipped file without actually having to gunzip it (same as **gunzip -c**). You can even print it directly, using **gzcat *filename* | lpr**
* printing
  + **lpr *filename*** --- print. Use the -P option to specify the printer name if you want to use a printer other than your default printer. For example, if you want to print double-sided, use 'lpr -Pvalkyr-d', or if you're at CSLI, you may want to use 'lpr -Pcord115-d'. See 'help printers' for more information about printers and their locations.
  + **lpq** --- check out the printer queue, e.g. to get the number needed for removal, or to see how many other files will be printed before yours will come out
  + **lprm *jobnumber*** --- remove something from the printer queue. You can find the job number by using lpq. Theoretically you also have to specify a printer name, but this isn't necessary as long as you use your default printer in the department.
  + **genscript** --- converts plain text files into postscript for printing, and gives you some options for formatting. Consider making an alias like **alias ecop 'genscript -2 -r \!\* | lpr -h -Pvalkyr'** to print two pages on one piece of paper.
  + **dvips *filename*** --- print **.dvi** files (i.e. files produced by LaTeX). You can use **dviselect** to print only selected pages. See the [LaTeX page](http://mally.stanford.edu/%7Esr/computing/latex.html) for more information about how to save paper when printing drafts.

## Directories

Directories, like folders on a Macintosh, are used to group files together in a hierarchical structure.

* mkdir dirname --- make a new directory
* cd dirname --- change directory. You always start out in your 'home directory', and you can get back there by typing 'cd' without arguments. 'cd ..' will get you one level up from your current position.
* **pwd** --- tells you where you currently are.

## About other people

* **w** --- tells you who's logged in, and what they're doing. Especially useful: the 'idle' part. This allows you to see whether they're actually sitting there typing away at their keyboards right at the moment.

$w

11:47:16 up 28 days, 8:22, 2 users, load average: 1.01, 1.03, 1.26

USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT

whdcaxkc pts/0 10.57.42.146 Thu17 20:11m 0.35s 0.01s login -- whdcaxkc

whdcaxkc pts/1 10.57.12.174 10:38 0.00s 0.13s 0.01s login -- whdcaxkc

* **who** --- tells you who's logged on, and where they're coming from (IP add).

$who

whdcaxkc pts/0 Aug 5 17:05 (10.57.42.146)

whdcaxkc pts/1 Aug 9 10:38 (10.57.12.174)

* **finger *username*** --- gives you lots of information about that user, e.g. when they last read their mail and whether they're logged in. Often people put other practical information, such as phone numbers and addresses, in a file called **.plan**. This information is also displayed by 'finger'.

$finger whdcaxkc

Login: whdcaxkc Name: Kannan Chandran

Directory: /home/whdcaxkc Shell: /bin/bash

On since Thu Aug 5 17:05 (IST) on pts/0 from 10.57.42.146

20 hours 7 minutes idle

On since Mon Aug 9 10:38 (IST) on pts/1 from 10.57.12.174

No mail.

No Plan.

* **last -1 *username*** --- tells you when the user last logged on and off and from where. Without any options, **last** will give you a list of everyone's logins.
* **talk *username*** --- lets you have a (typed) conversation with another user
* **write *username*** --- lets you exchange one-line messages with another user
* **elm** --- lets you send e-mail messages to people around the world (and, of course, read them). It's not the only mailer you can use, but the one we recommend. It is like mailx on sun.

## About your (electronic) self

* **whoami** --- returns your username. Sounds useless, but isn't. You may need to find out who it is who forgot to log out somewhere, and make sure \*you\* have logged out.
* **finger** & .plan files   
  of course you can finger yourself, too. That can be useful e.g. as a quick check whether you got new mail. Try to create a useful .plan file soon. Look at other people's .plan files for ideas. The file needs to be readable for everyone in order to be visible through 'finger'. Do 'chmod a+r .plan' if necessary. You should realize that this information is accessible from anywhere in the world, not just to other people on turing.
* **passwd** --- lets you change your password, which you should do regularly (at least once a year).
* **ps -u *yourusername*** --- lists your processes. Contains lots of information about them, including the process ID, which you need if you have to kill a process.

Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.3/FAQ

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

whdcaxkc 4254 0.0 0.0 3480 744 pts/1 R+ 11:56 0:00 ps -u

whdcaxkc 9393 0.0 0.0 6004 1480 pts/0 S+ Aug06 0:00 bash

whdcaxkc 21998 0.0 0.0 5000 1508 pts/0 Ss Aug05 0:00 -bash

whdcaxkc 27731 0.0 0.0 5628 1488 pts/1 Ss 10:38 0:00 -bash

whdcaxkc 31295 0.0 0.0 4012 856 pts/1 T 11:08 0:00 man find

* **kill *PID*** --- kills (ends) the processes with the ID you gave. **This works only for your own processes, of course**. Get the ID by using **ps**. If the process doesn't 'die' properly, we can use the option -9. But we should attempt without that option first, because it doesn't give the process a chance to finish possibly important business before dying.
* **quota -v** --- show what your disk quota is (i.e. how much space you have to store files), how much you're actually using, and in case you've exceeded your quota (which you'll be given an automatic warning about by the system) how much time you have left to sort them out (by deleting or gzipping some, or moving them to your own computer).
* **du *filename*** --- shows the disk usage of the files and directories in *filename* (without argument the current directory is used). **du -s** gives only a total.
* **last *yourusername*** --- lists your last logins. Can be a useful memory aid for when you were where, how long you've been working for.

## Connecting to the outside world

* **nn** --- allows you to read news. It will first let you read the news local to turing, and then the remote news. If you want to read only the local or remote news, you can use **nnl** or **nnr**, respectively.
* **rlogin *hostname*** --- lets you connect to a remote host
* **telnet *hostname*** --- also lets you connect to a remote host. Use **rlogin** whenever possible.
* **ftp *hostname*** --- lets you download files from a remote host which is set up as an ftp-server. The most important commands within ftp are **get** for getting files from the remote machine, and **put** for putting them there (**mget** and **mput** let you specify more than one file at once). Be sure not to confuse the two, especially when your physical location doesn't correspond to the direction of the ftp connection you're making. ftp just overwrites files with the same filename. If you're transferring anything other than ASCII text, use binary mode.
* **lynx** --- lets you browse the web from an ordinary terminal.

## Miscellaneous tools

* **date** --- shows the current date and time.
* **cal** --- shows a calendar of the current month. Use e.g., 'cal 10 1995' to get that for October 95, or 'cal 1995' to get the whole year.

**Nohup**command.   
nohup command if added in front of any command will continue running the command or process even if you shut down your terminal or close your session to machine. For example, if I want to run a job that takes lot of time and must be run from terminal and is called update\_entries\_tonight .

**Ln**command.   
Instead of copying you can also make links to existing files using ln command.   
If you want to create a link to a file called coolfile in /usr/local/bin directory then you can enter this command.   
ln mycoolfile /usr/local/bin/coolfile

Some examples:

ln -s fileone filetwo will create a symbolic link and can exist across machines.

ln -n option will not overwrite existing files.

ln -f will force the link to occur.

**Diff**command.   
diff command will compare the two files and print out the differences between.

**Cmp**command.   
cmp command compares the two files. For exmaple I have two different files fileone and filetwo.   
cmp fileone filetwo will give me

fileone filetwo differ: char 80, line 4

if I run cmp command on similar files nothing is returned.  
-s command can be used to return exit codes. i.e. return 0 if files are identical, 1 if files are different, 2 if files are inaccessible.   
This following command prints a message 'no changes' if files are same   
cmp -s fileone file1 && echo 'no changes' .

**Find**command.   
Find command is a extremely useful command. you can search for any file anywhere using this command provided that file and directory you are searching has read write attributes set to you ,your, group or all. Find descends directory tree beginning at each pathname and finds the files that meet the specified conditions. Here are some examples.  
  
Some Examples:   
find $HOME -print will lists all files in your home directory.   
find /work -name chapter1 -print will list all files named chapter1 in /work directory.   
find / -type d -name 'man\*' -print will list all manpage directories.  
find / -size 0 -ok rm {} \; will remove all empty files on system.

conditions of find

-atime +n |-n| n will find files that were last accessed more than n or less than -n days or n days.

-ctime +n or -n will find that were changed +n -n or n days ago.

-depth descend the directory structure, working on actual files first and then directories. You can use it with cpio command.

-exec commad {} \; run the Unix command on each file matched by find. Very useful condition.

-print print or list to standard output (screen).

-name pattern find the pattern.

-perm nnnfind files whole permission flags match octal number nnn.

-size n find files that contain n blocks.

-type c Find file whole type is c. C could be b or block, c Character special file, d directory, p fifo or named pipe, l symbolic link, or f plain file.

**Sort**command.   
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.

**Uniq**command.   
uniq command removes duplicate adjacent lines from sorted file while sending one copy of each second file.  
Examples  
  
sort names | uniq -d will show which lines appear more than once in names file.

Options:

-c print each line once, counting instances of each.

-d print duplicate lines once, but no unique lines.

-u print only unique lines.

**Sed**command.   
sed command launches a stream line editor which you can use at command line.   
you can enter your sed commands in a file and then using -f option edit your text file. It works as   
sed [options] files

options:

-e 'instruction' Apply the editing instruction to the files.

-f script Apply the set of instructions from the editing script.

-n suppress default output.

**Df**command.   
df command displays information about mounted filesystems. It reports the number of free disk blocks. Typically a Disk block is 512 bytes (or 1/2 Kilobyte).   
syntax is   
df options name

Options

-b will print only the number of free blocks.

-e will print only the number of free files.

-f will report free blocks but not free inodes.

-F type will report on an umounted file system specified by type.

-k will print allocation in kilobytes.

-l will report only on local file systems.

-n will print only the file system name type, with no arguments it lists type of all filesystems

**Du**command.   
du command displays disk usage.

**Touch**– is used to change a [file](http://en.wikipedia.org/wiki/Computer_file)'s access and modification [timestamps](http://en.wikipedia.org/wiki/System_time). It is also used to create a new empty file.

-a, change the access time only

-c, if the file does not exist, do not create it and do not report this condition

-m, change the modification time only

-r file, use the access and modification times of file

-t time, use the time specified (in the format below) to update the access and modification times

The simplest [use case](http://en.wikipedia.org/wiki/Use_case) for touch is thus:

# touch myfile.txt

**Alias**- It is mainly used for abbreviating a system command, or for adding default arguments to a regularly used command. An alias will last for the life of the shell session. Regularly used aliases can be set from the shell's configuration so that they will be available upon the start of the corresponding shell session.

Syntax

alias [name=['command']]

An example of the [Bash](http://en.wikipedia.org/wiki/Bash_%28Unix_shell%29)shell syntax is:

alias copy="cp"

To view defined aliases the following commands can be used:

alias # Used without arguments; displays a list of all current aliases

alias -p # Analogous to the above; not available in 4DOS/4NT and PowerShell

alias myAlias # Displays the command for a defined alias

In Unix shells, if an alias exists for a command, it is possible to override the alias by surrounding the command with quotes or prefixing it with a backslash. For example, consider the following alias definition:

alias ls='ls -la'

To override this alias and execute the ls command as it was originally defined, the following syntax can be used:

'ls' or \ls

In Unix shells and 4DOS/4NT, aliases can be removed by executing the unalias command:

unalias copy # Removes the copy alias

unalias -a # The -a switch will remove all aliases; not available in 4DOS/4NT

unalias \* # 4DOS/4NT equivalent of `unalias -a` - wildcards are supported

**Tee- tee** is normally used to *split* the output of a program so that it can be seen on the display and also be saved in a file. The command can also be used to capture intermediate output before the data is altered by another command or program. The tee command reads [standard input](http://en.wikipedia.org/wiki/Standard_input), then writes its content to [standard output](http://en.wikipedia.org/wiki/Standard_output) and simultaneously copies it into the specified file(s) or variables.

The ***grep***command allows you to search one file or multiple files for lines that contain a pattern. Exit status is 0 if matches were found, 1 if no matches were found, and 2 if errors occurred.

The syntax for the ***grep*** command is:

grep [options] pattern [files]

options:

|  |  |
| --- | --- |
| -b | Display the block number at the beginning of each line. |
| -c | Display the number of matched lines. |
| -h | Display the matched lines, but do not display the filenames. |
| -i | Ignore case sensitivity. |
| -l | Display the filenames, but do not display the matched lines. |
| -n | Display the matched lines and their line numbers. |
| -s | Silent mode. |
| -v | Display all lines that do NOT match. |
| -w | Match whole word. |

Examples:

grep -c tech file1

**Some one Liners of Unix**

1. **How do you find out what’s your shell?** - echo $SHELL
2. **What’s the command to find out today’s date?** - date
3. **What’s the command to find out users on the system?** - who
4. **How do you find out the current directory you’re in?** - pwd
5. **How do you remove a file?** - rm
6. **How do you find out your own username?** - whoami
7. **How do you count words, lines and characters in a file?** – wc –l or –w or -c
8. **How do you search for a string inside a given file?** - grep string filename
9. **How do you search for a string inside a directory?**- grep string \*
10. **How do you search for a string in a directory with the subdirectories recursed?** - grep -r string \*
11. **What are PIDs?** - They are process IDs given to processes. A PID can vary from 0 to 65535.
12. **How do you list currently running process?** - ps
13. **How do you stop a process?**- kill pid
14. **How do you find out about all running processes?** - ps –ae : a for all user process, e for system process
15. **How do you stop all the processes, except the shell window?** - kill 0
16. **How do you fire a process in the background?** - ./process-name &
17. **How do you refer to the arguments passed to a shell script?** - $1, $2 and so on. $0 is your script name.
18. **What’s the conditional statement in shell scripting?** - if {condition} then … fi
19. **How do you do number comparison in shell scripts?** - -eq, -ne, -lt, -le, -gt, -ge
20. **How do you test for file properties in shell scripts?** - -s filename tells you if the file is not empty, -f filename tells you whether the argument is a file, and not a directory, -d filename tests if the argument is a directory, and not a file, -w filename tests for writeability, -r filename tests for readability, -x filename tests for executability
21. **How do you do Boolean logic operators in shell scripting?** - ! tests for logical not, -a tests for logical and, and -o tests for logical or.
22. **How do you find out the number of arguments passed to the shell script?** - $#
23. **What’s a way to do multilevel if-else’s in shell scripting?** - if {condition} then {statement} elif {condition} {statement} fi
24. **How do you write a for loop in shell?** - for {variable name} in {list} do {statement} done
25. **How do you write a while loop in shell?** - while {condition} do {statement} done
26. **How does a case statement look in shell scripts?** - case {variable} in {possible-value-1}) {statement};; {possible-value-2}) {statement};; esac
27. **How do you read keyboard input in shell scripts?** - read {variable-name}
28. **How do you define a function in a shell script?** - function-name() { #some code here return }
29. **How does getopts command work?** - The parameters to your script can be passed as -n 15 -x 20. Inside the script, you can iterate through the getopts array as while getopts n:x option, and the variable $option contains the value of the entered option.

**Q. State the advantages of Shell scripting?**There are many advantages of shell scripting some of them are, one can develop their own operating system with relevant features best suited to their organization than to rely on costly operating systems. Software applications can be designed according to their platform.

**Q. What are the disadvantages of shell scripting?**  
There are many disadvantages of shell scripting they are  
\* Design flaws can destroy the entire process and could prove a costly error.  
\* Typing errors during the creation can delete the entire data as well as partition data.  
\* Initially process is slow but can be improved.  
\* Portbility between different operating system is a prime concern as it is very difficult to port scripts etc.

**Q. Explain about the slow execution speed of shells?**  
Major disadvantage of using shell scripting is slow execution of the scripts. This is because for every command a new process needs to be started. This slow down can be resolved by using pipeline and filter commands.

**Q. Give some situations where typing error can destroy a program?**  
There are many situations where typing errors can prove to be a real costly effort. For example a single extra space can convert the functionality of the program from deleting the sub directories to files deletion. cp, cn, cd all resemble the same but their actual functioning is different. **Misdirected > can delete your data**.

**Q. Explain about Stdin, Stdout and Stderr?**  
These are known as standard input, output and error. These are categorized as 0, 1 and 2. Each of these functions has a particular role and should accordingly functions for efficient output. Any mismatch among these three could result in a major failure of the shell.

**Q. Explain about sourcing commands?**  
Sourcing commands help you to execute the scripts within the scripts. For example sh command makes your program to run as a separate shell. .command makes your program to run within the shell.

**Q. Explain about Login shell?**  
Login shell creates an environment and default parameters. It consists of two files they are profile files and shell rc files. These files initialize the login and non login files. Environment variables are created by Login shell.

**Q. Explain about non-login shell files?**  
The non login shell files are initialized at the start and they are made to run to set up variables. Parameters and path can be set etc are some important functions. These files can be changed and also your own environment can be set. These functions are present in the root. It runs the profile each time you start the process.

**Q. Explore about Environment variables?**  
Environment variables are set at the login time and every shell that starts from this shell gets a copy of the variable. When we export the variable it changes from an shell variable to an environment variable and these variables are initiated at the start of the shell.

**Q. What type of variables are in shell?**

There are two types of variable:  
(1)**System variables** - Created and maintained by Linux itself. This type of variable defined in CAPITAL LETTERS.  
(2) **User defined variables (UDV)** - Created and maintained by user.

|  |  |
| --- | --- |
| **System Variable** | **Meaning** |
| BASH=/bin/bash | Our shell name |
| BASH\_VERSION=1.14.7(1) | Our shell version name |
| COLUMNS=80 | No. of columns for our screen |
| HOME=/home/vivek | Our home directory |
| LINES=25 | No. of columns for our screen |
| LOGNAME=students | students Our logging name |
| OSTYPE=Linux | Our Os type |
| PATH=/usr/bin:/sbin:/bin:/usr/sbin | Our path settings |
| PS1=[\u@\h \W]\$ | Our prompt settings |
| PWD=/home/students/Common | Our current working directory |
| SHELL=/bin/bash | Our shell name |
| USERNAME=vivek | User name who is currently login to this PC |

**Q.** [How to define User defined variables (UDV)](http://www.freeos.com/guides/lsst/ch02sec03.html) ?

*Syntax:*   
variable name=value

*Example:*

$ n=10

**Q.** [Rules for Naming variable name (Both UDV and System Variable)](http://www.freeos.com/guides/lsst/ch02sec04.html)?

(1) Variable name must begin with Alphanumeric character or underscore character (\_), followed by one or more Alphanumeric character.

(2) Variables are case-sensitive, just like filename in Linux.

(4) You can define NULL variable as follows (NULL variable is variable which has no value at the time of definition) For e.g.  
$ vech=  
$ vech=""

Q. [How to print or access value of UDV (User defined variables)](http://www.freeos.com/guides/lsst/ch02sec05.html)?

To print or access UDV use following syntax  
*Syntax:*   
$variablename

Define variable vech as follows:  
$ vech=Bus  
To print contains of variable 'vech' type  
$ echo $vech

**Q.** [How](http://www.freeos.com/guides/lsst/ch02sec05.html)to perform arithmetic in shell?

Use to perform arithmetic operations.

*Syntax:*  
expr op1 math-operator op2  
  
*Examples:*   
$echo `expr 6 + 3`

For the last statement not the following points

(1) First, before expr keyword we used ` (back quote) sign not the (single quote i.e. ') sign. Back quote is generally found on the key under tilde (~) on PC keyboard OR to the above of TAB key.

(2) Second, expr is also end with ` i.e. back quote.

(3) Here expr 6 + 3 is evaluated to 9, then echo command prints 9 as sum

(4) Here if you use double quote or single quote, it will NOT work

**Q. How to read values from user?**

*Syntax:*   
read variable1, variable2,...variableN

Example:

echo "Your first name please:"  
read fname  
echo "Hello $fname, Lets be friend!"

**Q. How can we write more than one command in one line?**

*Syntax:*  
command1;command2

*Examples:*  
**$ date;who**

Will print today's date followed by users who are currently login. Note that You can't use

Q. test command or [expr]?

test command or [ expr ] is used to see if an expression is true, and if it is true it return zero(0), otherwise returns nonzero for false.  
*Syntax:*   
test expression OR [ expression ]

Example:

#!/bin/sh  
if test $1 -gt 0  
then  
echo "$1 number is positive"  
fi

**Shell also test for file and directory types**

|  |  |
| --- | --- |
| **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 |

**Q. if else syntax ;**

if condition

then

condition is zero (true - 0)

execute all commands up to else statement

elif

else

if condition is not true then

execute all commands up to fi

fi

**Example :**

$ vi isnump\_n  
#!/bin/sh  
#  
# Script to see whether argument is positive or negative  
#  
if [ $# -eq 0 ]  
then  
echo "$0 : You must give/supply one integers"  
exit 1  
fi

if test $1 -gt 0  
then  
echo "$1 number is positive"  
else  
echo "$1 number is negative"  
fi

**Q. for loop in shell?**

*Syntax:*

*for { variable name } in { list }*

*do*

*execute one for each item in the list until the list is*

*not finished (And repeat all statement between do and done)*

*done*

**Example:**

for i in 1 2 3 4 5  
do  
echo "Welcome $i times"  
done

Even you can use following syntax:

*Syntax:*

for (( expr1; expr2; expr3 ))  
do  
.....

...

repeat all statements between do and

done until expr2 is TRUE

Done

**Example:**

for (( i = 0 ; i <= 5; i++ ))  
do  
echo "Welcome $i times"  
done

**Q. While loop in shell?**

*Syntax:*

*while [ condition ]*

*do*

*command1*

*command2*

*command3*

*..*

*....*

*Done*

*Example:*

while [ $i -le 10 ]  
do  
echo "$n \* $i = `expr $i \\* $n`"  
i=`expr $i + 1`  
done

**Q. How to debug shell script?**

For this purpose you can use -v and -x option with sh or bash command to debug the shell script. General syntax is as follows:  
*Syntax:*  
sh option { shell-script-name }  
**OR**  
bash option { shell-script-name }  
Option can be  
**-v** Print shell input lines as they are read.  
**-x** After expanding each simple-command, bash displays the expanded value of PS4 system variable, followed by the command and its expanded arguments.

**Q. What is test command in unix shell?**

The unix test command can test for various conditions, and then returns 0 for true and 1 for false. Usually, test is used in if...then or whileloops.

Let start with a simple unix test or if examples for comparing two number equal or not  
if [ 9 -eq 9 ]  
then  
echo equal  
else  
echo no equal  
fi  
We can do the same thing by using test command as  
test 9 -eq 9  
if [ $? -eq 0 ]  
then  
echo equal  
else  
echo no equal  
fi

**Q. What is /dev/null in unix?**

This is special Linux file which is used to send any unwanted output from program/command.  
*Syntax:*  
command > /dev/null

*Example:*  
**$ ls > /dev/null**  
Output of above command is not shown on screen its send to this special file. The /dev directory contains other device files. The files in this directory mostly represent peripheral devices such disks like floppy disk, sound card, line printers etc

**Q. What is conditional execution i.e. && and || in shell?**

The control operators are && (read as AND) and || (read as OR). The syntax for AND list is as follows  
*Syntax:*   
command1 && command2  
command2 is executed if, and only if, command1 returns an exit status of zero.

The syntax for OR list as follows  
*Syntax:*  
command1 || command2  
command2 is executed if and only if command1 returns a non-zero exit status.  
  
You can use both as follows  
*Syntax:*  
command1&& comamnd2 if exist status is zero || command3 if exit status is non-zero  
if command1 is executed successfully then shell will run command2 and if command1 is not successful then command3 is executed.

*Example:*  
**$ rm myf && echo "File is removed successfully" || echo "File is not removed"**

If file (myf) is removed successful (exist status is zero) then "*echo File is removed successfully*" statement is executed, otherwise "*echo File is not removed*" statement is executed (since exist status is non-zero)

**Q. how to write functions in shell?**

*Syntax:*

function-name ( )

{

command1

command2

.....

...

commandN

return

}

Where function-name is name of you function, that executes series of commands. A return statement will terminate the function.*Example:*  
Type SayHello() at $ prompt as follows  
**$ SayHello()  
{  
echo "Hello $LOGNAME, Have nice computing"  
return  
}**

### Arithmetic Operations in Shell Scripting

For arithmetic operations such as addition, subtraction, multiplication, and division. We uses a function called expr; e.g., "expr a + b" means 'add a and b'.

e.g.:

sum=`expr 12 + 20`

Similar syntax can be used for subtraction, division, and multiplication.

There is another way to handle arithmetic operations; enclose the variables and the equation inside a square-bracket expression starting with a "$" sign. The syntax is

$[expression operation statement]

e.g.:

echo $[12 + 10]

### Conditional Statements

The syntax is show below:

if [ conditional statement ]

then

... Any commands/statements ...

fi

The script cited below will prompt for a username, and if the user name is "blessen", will display a message showing that I have successfully logged in. Otherwise it will display the message "wrong username".

#!/bin/sh

echo "Enter your username:"

read username

if [ "$username" = "blessen" ]

then

echo 'Success!!! You are now logged in.'

else

echo 'Sorry, wrong username.'

Fi

**Remember to always enclose the variable being tested in double quotes; not doing so will cause your script to fail due to incorrect syntax when the variable is empty.** Also, the square brackets (which are an alias for the 'test' command) must have a space following the opening bracket and preceding the closing one.

### Variable Comparison

If the values of variables to be compared are numerical, then you have to use these options:

-eq Equal to  
-ne Not Equal to  
-lt Less than  
-le Less than or equal to  
-gt Greater than  
-ge Greater then or equal to

If they are strings, then you have to use these options:

= Equal to  
!= Not Equal to  
< First string sorts before second  
> First string sorts after second

### Loops

#### The "for" Loop

The most commonly used loop is the "for" loop. In shell scripting, there are two types: one that is similar to C's "for" loop, and an iterator (list processing) loop.

Syntax for the first type of "for" loop (again, this type is only available in modern shells):

for ((initialization; condition; increment/decrement))

do

...statements...

done

Example:

#!/bin/sh

for (( i=1; $i <= 10; i++ ))

do

echo $i

done

This will produce a list of numbers from 1 to 10. The syntax for the second, more widely-available, type of "for" loop is:

for <variable> in <list>

do

...statements...

done

This script will read the contents of '/etc/group' and display each line, one at a time:

#!/bin/sh

count=0

for i in `cat /etc/group`

do

count=`expr "$count" + 1`

echo "Line $count is being displayed"

echo $i

done

echo "End of file"

Another example of the "for" loop uses "seq" to generate a sequence:

#!/bin/sh

for i in `seq 1 5`

do

echo $i

done

### While Loop

The "while" loop is another useful loop used in all programming languages; it will continue to execute until the condition specified becomes false.

while [ condition ]

do

...statement...

done

The following script assigns the value "1" to the variable num and adds one to the value of num each time it goes around the loop, as long as the value of num is less than 5.

#!/bin/sh

num=1

while [$num -lt 5]; do num=$[$num + 1]; echo $num; done

### Select and Case Statement

Similar to the "switch/case" construct in C programming, the combination of "select" and "case" provides shell programmers with the same features. The "select" statement is not part of the "case" statement, but I've put the two of them together to illustrate how both can be used in programming.

Syntax of select:

select <variable> in <list>

do

...statements...

Done

Syntax of case:

case $<variable> in

<option1>) statements ;;

<option2>) statements ;;

\*) echo "Sorry, wrong option" ;;

esac

The example below will explain the usage of select and case together, and display options involving a machine's services needing to be restarted. When the user selects a particular option, the script starts the corresponding service.

#!/bin/bash

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

select opt in apache named sendmail

do

case $opt in

apache) /etc/rc.d/init.d/httpd restart;;

named) /etc/rc.d/init.d/named restart;;

sendmail) /etc/rc.d/init.d/sendmail restart;;

\*) echo "Nothing will be restarted"

esac

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

# If this break is not here, then we won't get a shell prompt.

break

done

[ Rather than using an explicit 'break' statement - which is not useful if you want to execute more than one of the presented options - it is much better to include 'Quit' as the last option in the select list, along with a matching case statement. -- Ben ]

### Functions

In the modern world where all programmers use the OOP model for programming, even we shell programmers aren't far behind. We too can break our code into small chunks called functions, and call them by name in the main program. This approach helps in debugging, code re-usability, etc.

Syntax for "function" is:

< name of function> ()

{ # start of function

statements

} # end of function

Functions are invoked by citing their names in the main program, optionally followed by arguments. For example:

#!/bin/sh

sumcalc ()

{

sum=$[$1 + $2]

}

echo "Enter the first number:"

read num1

echo "Enter the second number:"

read num2

sumcalc $num1 $num2

echo "Output from function sumcalc: $sum"

### Debugging Shell Scripts

Now and then, we need to debug our programs, we use the '-x' and '-v' options of the shell. The '-v' option produces verbose output. The '-x' option will expand each simple command, "for" command, "case" command, "select" command, or arithmetic "for" command, displaying the expanded value of PS4, followed by the command and its expanded arguments or associated word list. Try them in that order - they can be very helpful when you can't figure out the location of a problem in your script.

**Shell Variable:**

### 1. The PATHvariable

One of the more important variables is the path variable. The path variable controls where the shell searches for commands, when you type them to at the prompt.

2. HOME

This is the home directory of your account. It is also used as the default when typingcdwithout any arguments.

3. PS1

The primary prompt string, normally set to `$'. This is what the computer prints whenever it is ready to process another command.

4. PS2

The secondary prompt string, normally set to `>'. The shell prints this prompt whenever you have type an incomplete line, e.g., if you are missing a closing quote. For example:

$**echo 'hello**

>**world'**

hello

world

5. SHELL

This variables give the path of the shell you are currently executing, e.g., if you are executing the standard Bourne shell, the SHELLvariables is usually set to /bin/sh.

6. IFS

This is set to theinternal field separators, normally set to space, tab, and blank. We will talk more about this later.

# One-liners

Here are some one-liners that might come in handy some time. These one-liners serve both as educational examples that solves "real problems". If you have suggestions for more one-liners, please [mail me](mailto:matkin@docs.uu.se).

#### Renaming several files at the same time

If you have a number of files namedfoo.C,bar.C.gz, etc. and want to rename them to foo.cc,bar.cc.gz, etc. This line will do the trick.

\ls \*.C\* | sed 's/\(.\*\).C\(.\*\)/mv & \1.cc\2/' | sh

The backslash before the lscommand is to prevent is from being expanded, in the case that is is an alias and you are using shell that has aliases (such as Bash). We want to prevent the shell from doing this expansion since lsmight come out as ls -F (which would behave strange) or ls -l which is really bad.

An alternative is to install the [rename](http://user.it.uu.se/%7Ematkin/documents/PERL/#rename)script, which is written in Perl.

#### Remove processes matching some regular expression

If you have a number of processes that you want to kill, one of the following one-liners might be useful:

kill `ps xww | grep "sleep" | cut -c1-5` 2>/dev/null

ps xww | grep "sleep" | cut -c1-5 | xargs kill 2>/dev/null

This will kill any processes that has the word "sleep" in the calling command. If your killdoes not handle multiple pids' you can either use the one-liner

ps xww | grep "sleep" | cut -c1-5 | xargs -i kill {} 2>/dev/null

or use a for-loop:

for x in `ps xww | grep "sleep" | cut -c1-5`

do

kill $x 2>/dev/null

done

But then it is not a one-liner any more.

What does $# stand for?  
  
$# means, number of positional parameters set  
  
What is $\*  
  
$1: the 1st parameter  
$2: the 2nd parameter  
...  
$\*  
All parameters in the command line  
  
If you have a string "one two three", which shell command would you use to extract the   
  
strings  
  
echo $string | cut -d" " -f1  
echo $string | cut -d" " -f2  
echo $string | cut -d" " -f3   
  
What is the difference between a shell variable that is exported and the one that is not   
  
exported?  
  
export LANG=C  
will make the variable LANG the global variable, put it into the global environment. all   
  
other processes can use it.  
  
LANG=C  
will change the value only in the current script.  
  
How will you list only the empty lines in a file (using grep)  
  
grep "^[ ]\*$" filename.txt  
  
In character set (between [ and ] one space and tab is given)  
  
this command will gives all the blank line including those having space and tabs (if   
  
pressed)only  
  
  
The grep utility searches text files for a pattern and prints all lines that contain that   
  
pattern. It uses a com-pact non-deterministic algorithm.   
  
How do you schedule a command to run at 4:00 every morning?  
  
Step 1. Set environemen variable EDITOR=vi if not set  
  
Step 2. Give command crontab -e  
  
Step 3. Add following entry at the end  
  
0 4 \* \* \*   
  
How do u open a read only file in Unix?  
  
vi filename  
e.g vi abc.txt but you cannot write to it  
  
easy way if file is small  
cat filename and contents is display on screen  
  
  
more filename.txt   
  
What are the different kinds of loops available in shell script  
  
Broadly categorised in 3  
for  
while  
until   
  
How do you read arguments in a shell program - $1, $2 ...  
  
Shell script accepts parameters in following format.  
$1 : first  
$2 : second so on upto  
$9 : 9th param  
whereas $0 : gives script/function name  
If your script has more than 9 params then accept in following way.  
${12} : 12th param  
${18} : 18th param

Links

https://sites.google.com/site/sumedhshende/shellscripting