Interview Questions Sets : Shell Script Descriptive

Interview Questions Sets : Shell Script Descriptive Questions Sets

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.

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.

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.

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.

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 ...

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.

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).

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 ...

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.

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.

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.

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.

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.

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.

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.

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.

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.

How can you tell what shell you are running on a UNIX system?

Answer :

You can do the Echo $RANDOM. It will return a undefined variable if you are from the **C-Shell,** just a return prompt if you are from the **Bourne shell**, and a 5 digit random numbers if you are from the Korn shell.

You could also do a ps -l and look for the shell with the highest PID.

What are conditions on which deadlock can occur while swapping the processes?

All processes in the main memory are asleep. All 'ready-to-run' processes are swapped out.

There is no space in the swap device for the new incoming process that are swapped out of the main memory. There is no space in the main memory for the new incoming process.

How do you change File Access Permissions?

Answer :

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:

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).

Example 2:

To change mode of myfile to 'rwxr–r–' we give the args as:

chmod(myfile,0744).

List the system calls used for process management.

Answer :

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

What is the difference between Swapping and Paging?

Answer:

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.

What is the difference between cmp and diff commands?

Answer :

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

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

What is meant by the nice value?

Answer :

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.

What is a daemon?

Answer :

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.

What are the process states in UNIX?

Answer :

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.

How are devices represented in UNIX?

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).

What is 'inode'?

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

Brief about the directory representation in UNIX

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).

What are the Unix system calls for I/O?

- 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.

How do you change File Access Permissions?

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).

Example 2:

To change mode of myfile to 'rwxr--r--' we give the args as:

chmod(myfile,0744).

What are links and symbolic links in UNIX file system?

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

What is a FIFO?

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).

How do you create special files like named pipes and device files?

The system call mknod creates special files in the following sequence.

kernel assigns new inode,

sets the file type to indicate that the file is a pipe, directory or special file,

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.

Discuss the mount and unmount system calls

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.

How does the inode map to data block of a file?

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.

What is a shell?

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.

Brief about the initial process sequence while the system boots up.

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).

What are various IDs associated with a process?

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

Explain fork() system call.

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.

Predict the output of the following program code

```
main()
{
fork();
printf("Hello World!");
}
```

Answer:

Hello World!Hello World!

Explanation:

The fork creates a child that is a duplicate of the parent process. The child begins from the fork().All the statements after the call to fork() will be executed twice.(once by the parent process and other by child). The statement before fork() is executed only by the parent process.

Predict the output of the following program code

```
main()
{
fork(); fork(); fork();
printf("Hello World!");
}
```

Answer:

"Hello World" will be printed 8 times.

Explanation:

2^n times where n is the number of calls to fork()

List the system calls used for process management:

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

How can you get/set an environment variable from a program?

Getting the value of an environment variable is done by using `getenv()'.

Setting the value of an environment variable is done by using `putenv()'.

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.

What is a zombie?

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.)

What are the process states in Unix?

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.

What Happens when you execute a program?

When you execute a program on your UNIX system, the system creates a special environment for that program. This environment contains everything needed for the system to run the program as if no other program were running on the system. Each process has process context, which is everything that is unique about the state of the program you are currently running. Every time you execute a program the UNIX system does a fork, which performs a series of operations to create a process context and then execute your program in that context. The steps include the following:

• Allocate a slot in the process table, a list of currently running programs kept by UNIX.

• Assign a unique process identifier (PID) to the process.

• iCopy the context of the parent, the process that requested the spawning of the new process.

• Return the new PID to the parent process. This enables the parent process to examine or control the process directly.

After the fork is complete, UNIX runs your program.

What Happens when you execute a command?

When you enter 'ls' command to look at the contents of your current working directory, UNIX does a series of things to create an environment for ls and the run it:

The shell has UNIX perform a fork. This creates a new process that the shell will use to run the ls program.

The shell has UNIX perform an exec of the ls program. This replaces the shell program and data with the program and data for ls and then starts running that new program.

The ls program is loaded into the new process context, replacing the text and data of the shell. The ls program performs its task, listing the contents of the current directory.

What is a Daemon?

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.

What is 'ps' command for?

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.

How would you kill a process?

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.

What is an advantage of executing a process in background?

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)

How do you execute one program from within another?

The system calls used for low-level process creation are execlp() and execvp(). The execlp call overlays the existing program with the new one , runs that and exits. The original program gets back control only when an error occurs.

execlp(path,file_name,arguments..); //last argument must be NULL

A variant of execlp called execvp is used when the number of arguments is not known in advance.

execvp(path,argument_array); //argument array should be terminated by NULL

What is IPC? What are the various schemes available?

The term IPC (Inter-Process Communication) describes various ways by which different process running on some operating system communicate between each other. Various schemes available are as follows:

Pipes:

One-way communication scheme through which different process can communicate. The problem is that the two processes should have a common ancestor (parent-child relationship). However this problem was fixed with the introduction of named-pipes (FIFO).

Message Queues :

Message queues can be used between related and unrelated processes running on a machine.

Shared Memory:

This is the fastest of all IPC schemes. The memory to be shared is mapped into the address space of the processes (that are sharing). The speed achieved is attributed to the fact that there is no kernel involvement. But this scheme needs synchronization.

State and explain about features of UNIX?

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.

Explain about sh?

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.

Explain about system and user utilities?

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.

Explain about document formatting?

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.

Explain about communication features in UNIX?

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.

Explain about chmod options filename?

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.

Explain about gzip filename?

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.

Explain about refer?

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.

Explain about lpr filename?

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.

Explain about lprm job number?

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.

Brief about the command ff?

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.

Brief about finger username?

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.

Explain about the command elm?

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.

Brief about the command kill PID?

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.

Explain about the command lynx?

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.

Brief about the command nn?

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.

Brief about ftp hostname?

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.

Explain about the case statement.

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.

Explain the basic forms of each loop?

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.

Describe about awk and sed?

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.

Explain about signal argument?

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.

Explain about exec?

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.

Explain about trap command

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.

Explain about sort command?

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.

Explain about the command overwrite?

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.

Explain about kill command?

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.

Explain about the shell variable IFS?

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.

Explain about the rules used in overwrite to preserve the arguments to the users command?

Some of the rules are

• $* and $@ expand into the arguments and are rescanned; blanks in arguments will result in multiple arguments.

• "$*" is a single word composed of all the arguments to the shell file joined together with spaces.

• "$@" 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.

Explain about @@@ lines?

@@@ 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.

Explain about vis?

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.

Is the function call to exit at the end of vis necessary?

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.

Explain about fgets?

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.

Explain about efopen page?

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.

Explain about yacc parser generator?

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.

What is $*?

Will display all the commandline arguments that are passed to the script

Different types of shells?

Bourne Shell (bash)

Korn Shell (ksh)

C Shell (csh)

What is difference between a wild-card and regular expression?