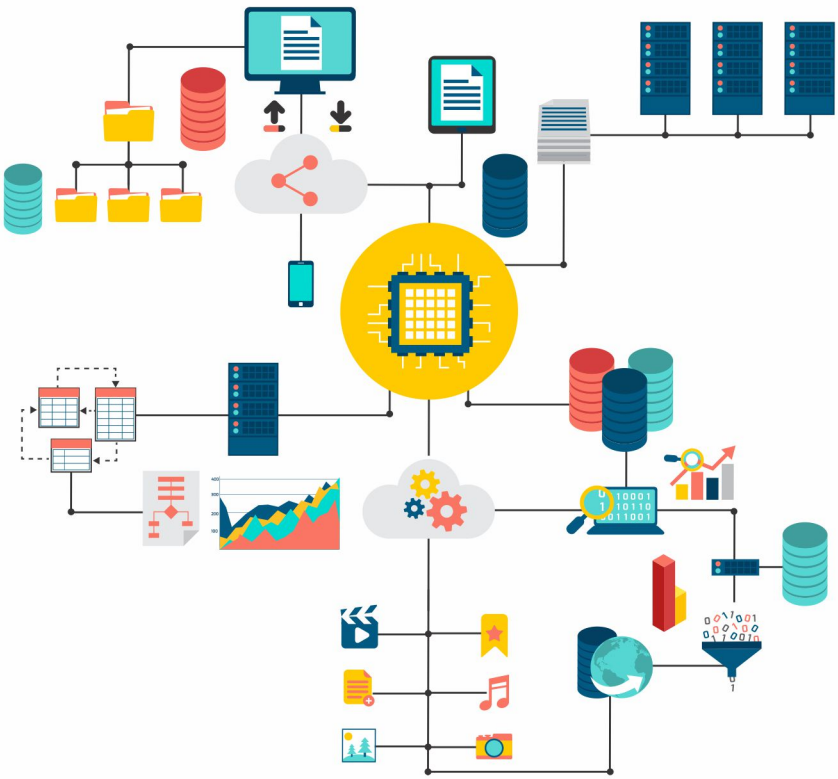
Job Interview Questions Series

UNIX® SHELL PROGRAMMING

INTERVIEW QUESTIONS YOU'LL MOST LIKELY BE ASKED



352

Interview Questions

VIBRANT PUBLISHERS

Job Interview Questions Series

UNIX® SHELL PROGRAMMING

INTERVIEW QUESTIONS YOU'LL MOST LIKELY BE ASKED

352
Interview Questions

VIBRANT
PUBLISHERS

UNIX® Shell Programming

Interview Questions You'll Most Likely Be Asked

ISBN-10: 1-949395-02-2
ISBN-13: 978-1-949395-02-0

Library of Congress Control Number: 2011911708

Table of Contents

Dear Reader,

Thank you for purchasing **UNIX Shell Programming Interview Questions You'll Most Likely Be Asked.** We are committed to publishing books that are content-rich, concise and approachable enabling more readers to read and make the fullest use of them. We hope this book provides the most enriching learning experience as you prepare for your interview.

Should you have any questions or suggestions, feel free to email us at **reachus@vibrantpublishers.com**

Thanks again for your purchase. Good luck with your interview!

– Vibrant Publishers Team

UNIX **Shell Programming** Interview Questions

Review these typical interview questions and think about how you would answer them. Read the answers listed; you will find best possible answers along with strategies and suggestions.

www.vibrantpublishers.com

This page is intentionally left blank

Chapter 1

C Shell – Beginner

1: What must you do before you are able to run your new script for the first time by its name or with an alias?

Answer:

You must make it executable, that is to execute the command:

chmod +x scriptname

2: The following command is included in the *.login* script of a user:

alias whois ´grep \!^ /etc/passwd´

What will be the output, when the user issues the following?

who is guru

Answer:

If there is a defined user account named "*guru*", or the string guru is contained elsewhere in */etc/passwd* file, then the output will be

the entry which contains the string "*guru*", otherwise, it will be an empty line.

3: If the condition If (*-r filename*)fails (returns false), what are the possible reasons?

Answer:

The possible reasons are:

a) filename is not readable by the owner of the process

b) filename does not exist

4: Which is the difference between the next two statements?

set var =99

@ var = 99

Answer:

Using the typical assignment form (set …), the value assigned in *var* is the string 99. Using the @, the value is the integer 99.

5: Given the code snippet:

```
@ n = 5
while ($n)
  # actions
...
end
```

What actions should be performed inside the loop, in order to get out of this loop?

Answer:

Any command that changes the value of variable *n*, for it to become 0 sometime. E.g., @ *n*—

6: What will the output of the following commands be? Explain.

set names =(Kathrin Chris Jacob)

shift names

echo $#names

Answer:

The output will be 2.

shift command gets rid of the first element of the array names. So, the *echo* command will display 2 as the number of elements of the array.

7: What does the command *rehash* do?

Answer:

Rehash recomputes the internal hash table for the PATH variable. If the new command resides in a directory not listed in PATH, add this directory to PATH, and then use rehash.

8: How could you ensure that a script will be run in *csh*?

Answer:

The first line of the script could be used to define the shell you want to use, as follows:

#!/bin/csh

This is sufficient to run the script in *csh*.

9: Given that *script1* is an executable C shell script situated in *directory /home/myhomedir/project1/data/dir1*, use three ways to run it, explaining the pros and cons.

Answer:

a) *cd/home/myhomedir/project1/data/dir1/script1*

(You should first *cd* to the directory path)

b) */home/myhomedir/project1/data/dir1/script1*

(You should include the absolute directory path)

c) *script1* (shortest form, but it works only if the directory path is added to the PATH environment variable of the user)

10: What will be the value of the *sixrem* variable, after executing this command?

@ sixrem = $data[2] % 6

Answer:

The expression divides the value of second element of the data array by 6 and assigns the remainder of the division to the *sixrem* variable.

11: Name two ways to obtain the length of a string, giving a simple example for each one.

Answer:

The two ways to obtain the length of a string are:

a) Using the *wc* command:

```
set string = "any string"
@ ln = `echo $string | wc -c` -1
```

b) Using the *awk* function length:

```
set string = "any string"
set ln = `echo $string | awk '{print length($0)}'`
```

12: Create a script that displays a list of regular files from the current directory.

Answer:

```
#!/bin/csh -f
foreach i (*)
   if( -f  $i ) then
      print $i
      endif
end
```

13: Describe in short, the word-completion feature of the *tcsh* shell.

Answer:

If you want word completion to work, whether on commands or filenames, you just need to type the beginning of the word and press the Tab key. The Shell will substitute the unfinished word with a complete one which is available in the input buffer and also adds a "/" to its end if it's a directory and a space if it's a word. The shell will check the buffer to identify whether the completed word is a word or a command, variable or filename. The buffer considers the first word in it and the words following {'|', '|&', ';', '&&', '||'} as a command. If the word starts with a $ it's a variable and everything else is a filename. If there's an empty line, it is completed as a filename.

14: In *tcsh*, how are the remaining choices (if any) listed whenever the word completion fails?

Answer:

The remaining choices (if any), are listed only if the shell variable *autolist* is set.

15: In *tcsh*, how do you disable filename substitution?

Answer:

noglob shell variable can be set to disable the filename substitution feature.

16: Compare the *sched tcsh* built-in command with the UNIX/Linux at command.

Answer:

These commands are similar but not the same. *sched* command runs directly from the shell, so it has access to shell variables and settings at command can run a scheduled command at exactly the specified time.

17: Schedule a *prompt* change at 10:55 as a reminder for an oncoming event.

Answer:

sched 10:55 set prompt = 'It\'s time for the important meeting: >'

18: What is the impact of *-f* option in the first line of a *csh* script?

(#!/bin/csh

versus

#!/bin/csh -f)

Answer:

When you add the *-f* option to *csh*, the shell will skip loading the startup files *.cshrc* and resources. It will also skip the hashing process. The shell will load quicker because of that.

19: How can you start a job in the background, and then terminate your login session, without terminating the background job?

Answer:

We can start a job in the background and then terminate our login session, without terminating the background job using "no hangup" command, *nohup:*

nohup command > output_file &

20: Which is the difference between

echo c{1,4,2,5,1}

and

echo [c]{1,4,2,5,1}?

Answer:

The first *echo* will display *c1 c4 c2 c5 c1*, while the second displays only *c1*.

21: Display the first and last arguments of a script, regardless of the number of arguments, and without a loop.

Answer:

my_var3 = $#argv

echomy_var1: $argv[$1]

echomy_last_var: $argv[$my_var3]

22: How will you set the 'search path' in *csv*?

Answer:

Search path in *csv* can be set as follows:

a) *setenv PATH"/myfolder1 /bin: /myfolder2 /myfile3"*

b) using list
set path =(/myfolder1 /bin /myfolder2 /myfile3)

23: Create a tar archive into /*home/user1/myarch.tar*, including all files ending in *.c, .h, .l, .y,.o* and *.cc* and also the *Makefile* from two directories, ~/*dir1* and ~/*dir2*.
Answer:
tar cvf /home/user1/myarch.tar ~/{dir1,dir2}/{Makefile,.{c,h,l,o,y,cc}}*
or
tar cvf /home/user1/myarch.tar ~/dir1/Makefile ~/dir1/.[chloy] ~/dir1/*.cc ~/dir2/Makefile ~/dir2/*.[chloy] ~/dir2/*.cc*

24: Your script must be executed with exactly two arguments, otherwise would be terminated. Write a code to implement these checks.
Answer:
if ($#argv <> 2) then
echo "Usage: $0 arg1 arg2"
echo "You must give exactly two parameters"
exit 20
endif

25: Write a pipeline that reads from the *j-th* line up to the *k-th* line of a text file, without using *awk*.
Answer:
set total = `cat textfile | wc -l`
set j = 10
set k = 18
@ count = $k - $j
head -$k textfile | tail -$count

Job Interview Questions Series

UNIX® SHELL PROGRAMMING

INTERVIEW QUESTIONS YOU'LL MOST LIKELY BE ASKED

UNIX Shell Programming Interview Questions You'll Most Likely Be Asked is a perfect companion to stand ahead above the rest in today's competitive job market. Rather than going through comprehensive, textbook-sized reference guides, this book includes only the information required immediately for job search to build an IT career. This book puts the interviewee in the driver's seat and helps them steer their way to impress the interviewer.

The following is included in this book:

- 276 UNIX Shell Programming Interview Questions, Answers and proven strategies for getting hired as an IT professional
- Dozens of examples to respond to interview questions
- 76 HR Questions with Answers and proven strategies to give specific, impressive, answers that help nail the interviews
- 2 Aptitude Tests download available on www.vibrantpublishers.com

www.vibrantpublishers.com

ISBN 978-1-949395-02-0