Linux Foundation

LFCT

Linux Foundation Certified Cloud Technician

For More Information – Visit link below:

https://www.examsempire.com/

Product Version

1. **Up to Date products, reliable and verified.**
2. **Questions and Answers in PDF Format.**

Latest Version: 6.0

Question: 1

You have two files, 'filel .txt' and 'file2.txt', which contain a list of IP addresses. You need to find the IP addresses that are present in 'filel .txt' but not in 'file2.txt'. Which command is the most efficient for this task?

A.
```
diff file1.txt file2.txt | grep '^>' | awk '{print $2}'
```
B.
```
grep -v -f file2.txt file1.txt
```
C.
```
comm -23 file1.txt file2.txt
```
D.
```
find file1.txt -exec grep -q '$(cat file2.txt)' {} ; -print
```
E.
```
awk '{print $1}' file1.txt | sort | uniq -u
```

Answer: C

Explanation:
The ‘comm’ command is specifically designed to compare sorted files and output lines that are unique to each file. Option C, ‘comm -23 filel .txt file2.txt’ , will only output lines that are present in Tilel .txt' but not in 'file2.txt’ . The other options may work but are less efficient or not as direct in their approach.

Question: 2

You have a file named 'log.txt' that contains thousands of lines. You need to extract all lines containing the string 'ERROR' and then replace all occurrences of the word 'database' with 'DB' in those lines. Which command combination can achieve this?

A.
```
grep 'ERROR' log.txt | sed 's/database/DB/g'
```
B.
```
grep 'ERROR' log.txt | awk '{gsub("database", "DB"); print}'
```
C.

```
sed '/ERROR/s/database/DB/g' log.txt
```

D.

```
awk '/ERROR/ {gsub("database", "DB"); print}' log.txt
```

E.

```
grep 'ERROR' log.txt | sed 's/database/DB/g' > log.txt
```

Answer: C,D

Explanation:
Both Option C and Option D correctly achieve the desired outcome- Option C uses ‘sed' to directly find lines containing 'ERROR' and replace 'database' with 'DB'. Option D uses 'awk' to similarly filter for lines with 'ERROR' and apply the substitution using 'gsub()' - Option E, while a valid syntax, will overwrite the original 'log txt' file, which may not be desired in all scenarios. Option A and Option B will only print the modified lines, not update the original file.

Question: 3

You are working on a large server configuration file ('server.conf) that has many sections. You need to find all lines containing the word 'port' and then print only the corresponding port number, which is always the last word on the line. What command sequence can accomplish this?

A.

```
grep 'port' server.conf | awk '{print $NF}'
```

B.

```
sed '/port/!d; s/. //' server.conf
```

C.

```
grep 'port' server.conf | cut -d' ' -f1
```

D.

```
awk '/port/ {print $NF}' server.conf
```

E.

```
find server.conf -exec grep 'port' {} ; | awk '{print $NF}'
```

Answer: A,B,D

Explanation:
Three options are correct Option A: Uses ‘grep' to filter for lines with 'port' and then Sawk' to print the last field CSNF). Option B: Uses 'sed' to delete lines not containing 'port' ('Iport/ld’) and then replaces everything before the last space with nothing ('s/. IF). Option D: Similar to Option A, uses ‘awk' to filter

for lines containing 'port' and directly print the last field using ‘SNF. Options C and E are incorrect. Option C will print the first field (not the last) if a space is the delimiter, and Option E will find all files named 'server-conf and process them, which is not the intention.

Question: 4

You have a file 'users.txt' containing usernames and their corresponding passwords on separate lines (username:password). You need to create a new file 'encrypted_users.txt' where each password is replaced with '' Which command combination can achieve this?

A.

```
awk '{print $1":" }' users.txt > encrypted_users.txt
```

B.

```
sed 's/://g' users.txt | awk '{print $1": "}' > encrypted_users.txt
```

C.

```
sed 's/:[^:] $/: /' users.txt > encrypted_users.txt
```

D.

```
grep ':' users.txt | sed 's/:[^:] $/: /' > encrypted_users.txt
```

E.

```
sed 's/:/: /' users.txt > encrypted_users.txt
```

Answer: C

Explanation:
Option C uses ‘sed' to replace everything from the colon to the end of the line with This effectively masks the password while keeping the username intact The other options are incorrect: Option A This will print only the username followed by ' : ' , losing the original password. Option B: This will remove all colons and then print only the username followed by ' : Option D: This will only process lines containing colons, not necessarily the username:password format Option E. This will simply replace all colons with ' which is not the desired behavior.

Question: 5

You have a file named 'server_logs.txt' containing lines like this:

```
2023-10-26 14:30:01  INFO  User1 logged in
2023-10-26 14:31:02  ERROR  Database connection failed
2023-10-26 14:32:03  INFO  User2 logged in
2023-10-26 14:33:04  INFO  User1 logged out
2023-10-26 14:34:05  WARNING  Low disk space
```

You want to extract the timestamps and log levels from this file using the 'cut' command and store them in a new file named 'extracted_logs.txt'. Which of the following commands achieves this?

A.
```
cut -d' ' -f1,2 server_logs.txt > extracted_logs.txt
```

B.
```
cut -d' ' -f1-2 server_logs.txt > extracted_logs.txt
```

C.
```
cut -d' ' -f1,3 server_logs.txt > extracted_logs.txt
```

D.
```
cut -d' ' -f1,4 server_logs.txt > extracted_logs.txt
```

E.
```
cut -d' ' -f1,5 server_logs.txt > extracted_logs.txt
```

Answer: B

Explanation:
cut -d' ' -f1-2 server _ logs . txt > extracted _ logs .txt
. This command uses 'cut' to extract fields from the 'server_logs.txt' file based on spaces as delimiters. The option '-d' specifies the delimiter as a space C '), and '-f1-2' indicates extracting fields 1 and 2, which correspond to the timestamp and log level respectively. The output is redirected to the 'extracted_logs.txt' file. The other options are incorrect because they either extract the wrong fields or use incorrect syntax.

Question: 6

You have a file containing a list of user IDs. You need to remove duplicate user IDs and sort the remaining unique IDs in ascending order. Which of the following commands will achieve this?

A.
```
sort | uniq
```

B.
```
uniq | sort
```

C.
```
sort -u
```

D.
```
uniq -u
```

E.
```
sort | uniq -u
```

Answer: C

Explanation:
This command sorts the input file in ascending order and then removes duplicate lines using the '-u' option. It accomplishes both tasks in a single command. Options A and B are incorrect because they use 'uniq' before 'sort', resulting in a sorted list but not necessarily unique. Option D uses '-u' with 'uniq' which displays only unique lines without sorting. Option E uses '-u' with 'uniq' after sorting, which is redundant and will remove unique lines as well.

Question: 7

You have a file named 'user_data.txt' containing user names and their corresponding roles separated by colons. For example:
John Doe : Admin
Jane Smith : User
Peter Jones :Admin
You need to extract only the usernames and store them in a new file named 'usernames.txt'. Which of the following commands will achieve this?

A.
```
cut -d':' -f1 user_data.txt > usernames.txt
```
B.
```
cut -d':' -f2 user_data.txt > usernames.txt
```
C.
```
tr ':' ' ' < user_data.txt > usernames.txt
```
D.
```
tr ' ' ':' < user_data.txt > usernames.txt
```
E.
```
cut -d':' -f1,2 user_data.txt > usernames.txt
```

Answer: A

Explanation:
This command uses 'cut' to extract fields from the 'user_data txt' file based on colons as delimiters. The option '-d' specifies the delimiter as a colon and '-fl' indicates extracting the first field, which corresponds to the username. The output is redirected to the 'usernames.txt' file. The other options are incorrect because they either extract the wrong field (option B, E) or use incorrect commands (option C, D) that don't achieve the desired output.

Question: 8

You are anal zin a 10 file named 'access 10 .txt' which contains lines like this:

```
192.168.1.10 - - [20/Oct/2023:14:30:01 +0000] "GET /index.html HTTP/1.1" 200 1234
192.168.1.20 - - [20/Oct/2023:14:31:02 +0000] "GET /about.html HTTP/1.1" 404 5678
192.168.1.10 - - [20/Oct/2023:14:32:03 +0000] "GET /index.html HTTP/1.1" 200 1234
```

You need to count the occurrences of each IP address in the log file and sort the results in descending order. Which of the following commands will achieve this?

A.

```
cut -d' ' -f1 access_log.txt | sort | uniq -c | sort -r
```

B.

```
cut -d' ' -f1 access_log.txt | uniq -c | sort -r
```

C.

```
cut -d' ' -f1 access_log.txt | sort -r | uniq -c
```

D.

```
cut -d' ' -f1 access_log.txt | uniq -c | sort
```

E.

```
cut -d' ' -f1 access_log.txt | sort | uniq -c | sort
```

Answer: A

Explanation:
This command first uses 'cut' to extract the IP addresses from the 'access_log.txt' file, then sorts the extracted IPs using 'sort'. Next, 'uniq -c' counts the occurrences of each unique IP, and finally, 'sort -r' sorts the results in descending order by count. The other options are incorrect because they either use 'uniq' before sorting the IPs or use incorrect sorting options.

Question: 9

You are tasked with creating a script that identifies and removes all files in the /tmp directory older than 30 days. Which command combination effectively accomplishes this using pipes and redirection?

A.

```
find /tmp -mtime +30 | xargs rm -f
```

B.

```
find /tmp -mtime +30 -delete
```

C.

```
find /tmp -mtime +30 > /dev/null; rm -f /tmp/
```

D.

```
find /tmp -mtime +30 | rm -f
```

E.

```
find /tmp -mtime +30 -exec rm -f {} +
```

Answer: A,E

Explanation:
Both options A and E are correct solutions. Option A uses the "xargs" command to process the output of "find" in batches, ensuring that the "rm" command receives a reasonable number of files at a time. Option E utilizes the ' exec" option within "find" to directly execute the "rm" command on each matched file, offering a slightly more efficient
approach compared to "xargs". Option B, while using the "-delete" option for direct removal, might not be suitable for large directories due to the potential for exceeding the maximum command line length. Option C uses redirection to send the output to /dev/null, but does not perform the actual deletion. Option D is incorrect because it lacks the necessary arguments to the "rm" command for deletion.

Question: 10

You need to create a script that displays the top 5 processes consuming the most CPU resources, along with their corresponding user IDs. Which combination of commands and redirection effectively achieves this?

A.

```
ps aux | sort -k3 -r | head -5
```

B.

```
top -b -n 1 | awk '{print $2,$11}' | sort -k2 -r | head -5
```

C.

```
ps aux | grep -v 'USER' | sort -k3 -r | head -5
```

D.

```
ps aux | awk '{print $1,$3}' | sort -k2 -r | head -5
```

E.

```
top -b -n 1 | sort -k3 -r | head -5
```

Answer: D

Explanation:
Option D is the correct approach. It utilizes ps aux to list processes, then pipes the output to awk to extract the user ID ($1) and CPU usage ($3). The result is sorted in descending order by CPU usage (-k2 -r) and then limited to the top 5 processes using head -5. Option A only sorts by CPU usage, not by user

ID. Option B uses top but does not correctly extract user ID information. Option C filters out a specific line, not necessarily related to process information. Option E sorts by a different metric, not CPU usage.

Question: 11

You are given a text file named "users.txt" containing a list of usernames, one per line. You need to extract usernames starting with the letter 'j' and redirect them to a new file named "jusers.txt". Which command combination effectively accomplishes this?

A.
```
grep '^j' users.txt > jusers.txt
```

B.
```
cat users.txt | grep 'j' > jusers.txt
```

C.
```
sed '/^j/p' users.txt > jusers.txt
```

D.
```
awk '/^j/' users.txt > jusers.txt
```

E.
```
grep 'j' users.txt > jusers.txt
```

Answer: A,D

Explanation:
Both option A (using grep) and option D (using awk) are correct solutions- Both commands effectively filter usernames starting with 'j' from "users-txt" and redirect them to "jusers.txt' Option B would match any line containing 'j', not just those starting with 'j'. Option C uses sed, but would print all lines instead of just those starting with 'j'. Option E would match any line containing 'j', not just those starting with 'j'

Question: 12

You are creating a script to update a file named "config.txt" with a new value for the "port" parameter. The original file contains a line "port=8080". You need to replace the value "8080" with "9090". Which command combination effectively modifies the file using pipes and redirection?

A.
```
sed 's/port=8080/port=9090/g' config.txt > temp.txt; mv temp.txt config.txt
```

B.
```
cat config.txt | sed 's/8080/9090/g' > config.txt
```

C.

```
sed -i 's/port=8080/port=9090/g' config.txt
```

D.

```
awk '{gsub("8080","9090"); print}' config.txt > config.txt
```

E.

```
grep 'port=' config.txt | sed 's/8080/9090/g' > config.txt
```

Answer: A,C,D

Explanation:
Options A, C, and D are all valid ways to update the "config.txt" file. Option A uses a temporary file ('temp txt") to avoid overwriting the original file, making it safer in case of errors. Option C uses the "-i" flag with sed to perform in-place editing; directly modifying the original file. Option D uses awk to replace the value "8080" with "9090" within the "config txt" file, also directly updating the original file. Option B uses cat to print the content of "config txt", redirecting the output to the same file, potentially causing unexpected results. Option E only processes lines containing "port= " and may not replace the value correctly for other instances within the file.

Thank You for Trying Our Product

Special 16 USD Discount Coupon: NSZUBG3X

Email: support@examsempire.com

Check our Customer Testimonials and ratings available on every product page.

Visit our website.

https://examsempire.com/