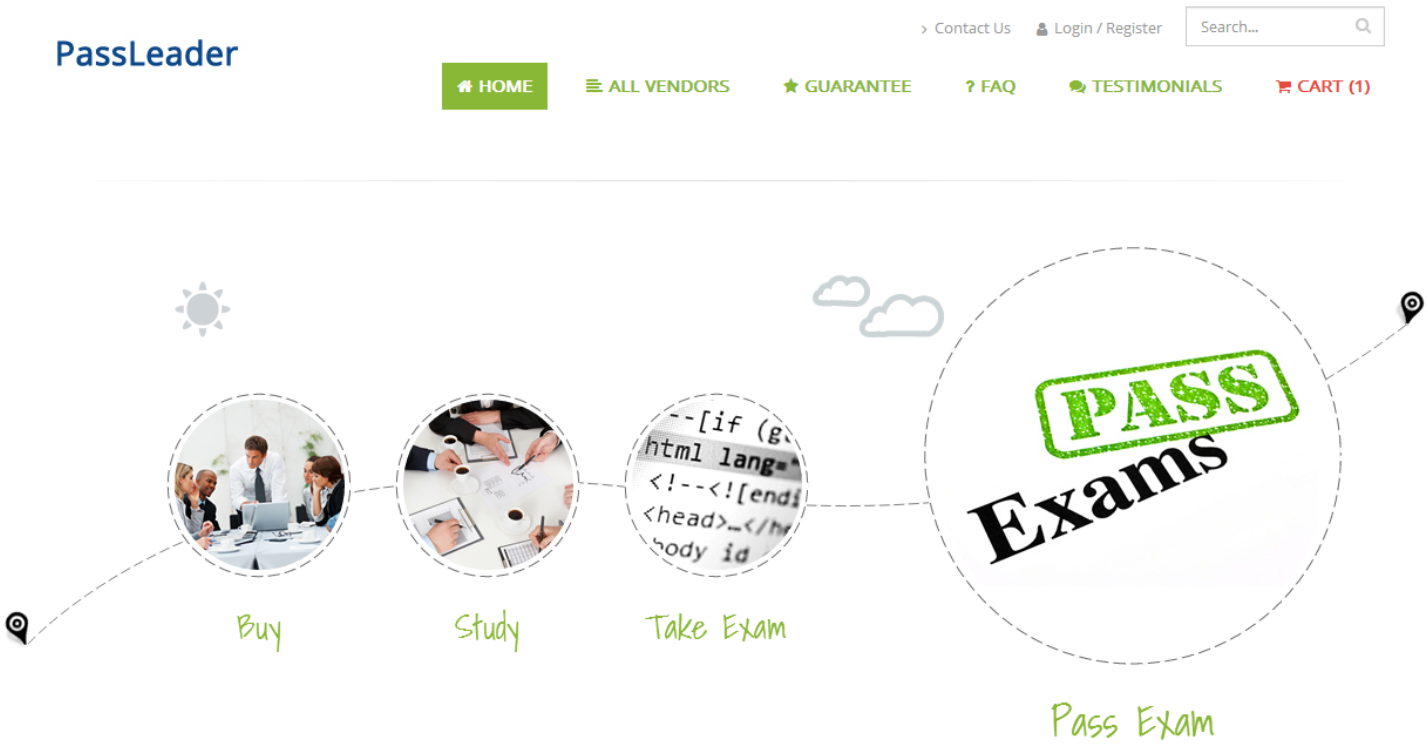
PassLeader



Try **PDF Demo** before you buy

We're not the only ones **happy** about PassLeader Practice Material ...

63159+ customers in 100+ countries use PassLeader Test Engine. Meet our customers.

VOREED GetCustom JET ORANGE iCompany Paradoxx iMessenger

http://www.passleader.top/

Latest Exam Guide & Learning Materials

Exam : **010-160-Deutsch**

Title : Linux Essentials Certificate Exam, version 1.6 (010-160 Deutsch Version)

Vendor : Lpi

Version : DEMO

QUESTION NO: 1
Was ist durch eine Lizenz für freie Software definiert?
A. Details der technischen Dokumentation, die jeder Mitwirkende bereitstellen muss.
B. Die Programmiersprachen, mit denen das Lizenzprogramm erweitert werden kann.
C. Eine vollständige Liste der Bibliotheken, die zum Kompilieren der lizenzierten Software erforderlich sind.
D. Beschränkungen für die Zwecke, für die die lizenzierte Software verwendet werden darf.
E. Bedingungen zum Ändern und Verteilen der lizenzierten Software.
Answer: E
Explanation:
A free software license is a notice that grants the recipient of a piece of software extensive rights to modify and redistribute that software. These actions are usually prohibited by copyright law, but the rights-holder (usually the author) of a piece of software can remove these restrictions by accompanying the software with a software license which grants the recipient these rights. Software using such a license is free software (or free and open-source software) as conferred by the copyright holder. Free software licenses grant users the freedom to use it for any purpose, study and change the source code and copy and redistribute the software with or without modifications. Free software must come with source code or provide access to it, while the freedom to redistribute includes the right to give away copies gratis as well as sell copies1 Reference: 1: Free-software license - Wikipedia

QUESTION NO: 2
Welches der folgenden Beispiele zeigt die allgemeine Struktur einer for-Schleife in einem Shell-Skript?
A. for *.txt as file => echo $file
B. for *.txt (echo $i)
C. for file in *.txt do
echo $i done
D. for ls *.txt exec {} \;
E. foreach @{file} { echo $i
}}
Answer: C
Explanation:
The general structure of a for loop in a shell script is as follows12:
for variable in list do commands done
The variable is the name of a loop counter or iterator that takes on the values of the items in the list. The list can be a sequence of words, numbers, filenames, or the output of a command. The commands are the body of the loop that are executed for each value of the variable. The do and done keywords mark the beginning and the end of the loop body.
The option C. for file in *.txt do echo $i done follows this structure, with the variable being file, the list being *.txt (which matches all the files with the .txt extension in the current directory), and the command being echo $i (which prints the value of the variable i, which is presumably set somewhere else in the script).
The other options are incorrect because:

A . for *.txt as file => echo $file uses an invalid syntax for a for loop. The as keyword is not part of the shell script syntax, and the => symbol is not a valid operator. The correct way to write this loop would be:
for file in *.txt do echo $file done
B . for *.txt (echo $i) uses an invalid syntax for a for loop. The parentheses are not part of the shell script syntax, and the loop body is missing the do and done keywords. The correct way to write this loop would be:
for i in *.txt do echo $i done
D . for ls *.txt exec {} ; uses an invalid syntax for a for loop. The ls command is not a valid variable name, and the exec {} ; is not a valid command. This looks like a mix of a for loop and a find command. The correct way to write this loop would be:
for file in *.txt do exec $file done
E . foreach @{file} { echo $i } uses an invalid syntax for a for loop. The foreach keyword is not part of the shell script syntax, and the @{file} and { echo $i } are not valid expressions. This looks like a mix of a for loop and a Perl syntax. The correct way to write this loop would be:
for file in * do echo $file done
Reference:
Looping Statements | Shell Script - GeeksforGeeks
How do I write a 'for' loop in Bash? - Stack Overflow

QUESTION NO: 3
Die Datei script.sh im aktuellen Verzeichnis enthält den folgenden Inhalt:
#! / bin / bash echo $ MYVAR
Die folgenden Befehle werden verwendet, um dieses Skript auszuführen:
MYVAR = Wert
./script.sh
Das Ergebnis ist eine leere Zeile anstelle des Inhalts der Variablen MYVAR. Wie sollte MYVAR eingestellt werden, damit script.sh den Inhalt von MYVAR anzeigt?
A. !MYVAR=value
B. env MYVAR=value
C. MYVAR=value
D. $MYVAR=value
E. export MYVAR=value
Answer: E
Explanation:
The reason why the script.sh does not display the content of the variable MYVAR is that the variable is not exported to the environment of the script. When a script is executed, it runs in a separate process that inherits the environment variables from the parent process, but not the shell variables. A shell variable is a variable that is defined and visible only in the current shell session, while an environment variable is a variable that is exported to the environment and visible to all processes that run in that environment1.
To make a shell variable an environment variable, we need to use the export command. The export command takes a shell variable name and adds it to the environment of the current shell and any subshells or processes that are created from it2. For example, to export the variable MYVAR with the value value, we can use:

export MYVAR=value
This will make the variable MYVAR available to the script.sh when it is executed, and the script will print the value of MYVAR as expected. Alternatively, we can also use the export command with the -n option to remove a variable from the environment, or with the -p option to list all the environment variables2.
The other options are not valid ways to set MYVAR as an environment variable. The !MYVAR=value option is not a valid syntax for setting a variable in bash. The env MYVAR=value option will run the env command with the MYVAR=value argument, which will print the environment variables with the addition of MYVAR=value, but it will not affect the current shell or the script.sh3. The MYVAR=value option will set MYVAR as a shell variable, but not as an environment variable, so it will not be visible to the script.sh1. The $MYVAR=value option will try to set the variable whose name is the value of MYVAR to the value value, which is not what we want4. Reference:
Linux Essentials Exam Objectives, Version 1.6, Topic 103.1, Weight 2
Linux Essentials Certification Guide, Chapter 3, Page 51-52
env(1) - Linux manual page
Bash Variables - LinuxConfig.org

QUESTION NO: 4
Welcher der folgenden Befehle findet alle Zeilen in der Datei operation-systems.txt, die den Begriff Linux enthalten, unabhängig vom Fall?
A. igrep linux operation-systems.txt
B. weniger -i Linux-Betriebssysteme.txt
C. grep -i Linux-Betriebssysteme.txt
D. Linux-Betriebssysteme.txt ausschneiden
E. schneide [Ll] [Ii] [Nn] [Uu] [Xx] Betriebssysteme.txt
Answer: C
Explanation:
The grep command is used to search for a pattern in a file or input. The -i option makes the search case-insensitive, meaning that it will match both uppercase and lowercase letters. The grep command takes the pattern as the first argument and the file name as the second argument. Therefore, the command grep -i linux operating-systems.txt will find all lines in the file operating-systems.txt which contain the term linux, regardless of the case. Reference:
Linux Essentials - Topic 103: Finding Linux Documentation and Linux Essentials - Topic 104 : Command Line Basics

QUESTION NO: 5
Welcher der folgenden Befehle setzt die Variable USERNAME auf den Wert bob?
A. set USERNAME bob
B. $USERNAME==bob
C. var USERNAME=bob
D. USERNAME<=bob
E. USERNAME=bob
Answer: E

Explanation:
The correct command to set the variable USERNAME to the value bob is USERNAME=bob. This command assigns the string bob to the variable name USERNAME, using the equal sign (=) as the assignment operator. There is no space around the equal sign, and the variable name and value are case-sensitive. This command sets a shell variable, which is only valid in the current shell session. To make the variable an environment variable, which can be inherited by child processes and subshells, you need to use the export command, such as export USERNAME=bob. The other commands are not valid for setting variables in Linux. The set command is used to set or unset shell options and positional parameters, not variables. The $ sign is used to reference the value of a variable, not to assign it. The == sign is used for comparison, not assignment. The var keyword is not used in Linux, but in some other programming languages. The <= sign is used for redirection, not assignment.
Reference:
Linux Essentials - Linux Professional Institute (LPI)
How to Set and List Environment Variables in Linux | Linuxize

QUESTION NO: 6
Das Ausführen des Befehls rm Downloads führt zu folgendem Fehler:
rm: 'Downloads /' kann nicht entfernt werden: Ist ein Verzeichnis
Welcher der folgenden Befehle kann stattdessen zum Entfernen von Downloads verwendet werden, sofern Downloads leer sind? (Wählen Sie zwei richtige Antworten.)
A. undir Downloads
B. rmdir Downloads
C. dir -r Downloads
D. rem Downloads
E. rm -r Downloads
Answer: B,E
Explanation:
To remove a directory, you need to use a command that can delete directories, not just files. The rm command can only remove files by default, unless you use the -r option, which stands for recursive. This option tells rm to delete the directory and all of its contents, including subdirectories and files. The rmdir command can also remove directories, but only if they are empty. If the directory contains any files or subdirectories, rmdir will fail and display an error message. Therefore, the correct commands to remove Downloads, assuming it is empty, are rmdir Downloads and rm -r Downloads. The other commands are either invalid or do not work on directories. Reference:
Linux Essentials - Linux Professional Institute (LPI), section 2.3.1
LPI Linux Essentials Study Guide: Exam 010 v1.6, 3rd Edition, chapter 4, page 93.

QUESTION NO: 7
Welche Dateien sind die Informationsquelle in der folgenden Ausgabe? (Wählen Sie zwei aus.) Uid = 1000 (Bob) gid = 1000 (Bob) Gruppen = 1000 (Bob), 10 (Rad), 150 (Docker), 1001 (Libvirt) (Wireshark), 989
A. / etc / id
B. / etc / passwd

C. / etc / group
D. / home / index
E. / var / db / users
Answer: B,C
Explanation:
The files /etc/passwd and /etc/group are the source of the information in the following output: uid=1000 (bob) gid=1000 (bob) groups=1000 (bob), 10 (wheel), 150 (docker), 1001 (libvirt) (wireshark), 989 The /etc/passwd file contains information about user accounts, such as the username, password, user ID (UID), group ID (GID), full name, home directory, and login shell1. The /etc/group file contains information about groups, such as the group name, password, group ID (GID), and members2.
The output shows the UID, GID, and group membership of the user bob. The UID and GID of bob are 1000, which can be found in the /etc/passwd file. The groups that bob belongs to are bob, wheel, docker, libvirt, wireshark, and 989, which can be found in the /etc/group file. The group names are shown in parentheses after the GID, except for the last group, which has no name.
The other options are not files that store user and group information in Linux. The /etc/id file does not exist by default. The /home/index file is not a standard file and has no relation to user and group information. The /var/db/users file is not a standard file and has no relation to user and group information. Reference:
Linux Essentials Exam Objectives, Version 1.6, Topic 103.1, Weight 2
Linux Essentials Certification Guide, Chapter 3, Page 51-52
Linux Filesystem Hierarchy, Chapter 3, Page 17-18
Linux Users and Groups, Chapter 2, Page 9-10

QUESTION NO: 8
Ein Verzeichnis enthält die folgenden drei Dateien:
Texte 1.txt
Texte 2.txt
Texte 3.csv
Welcher Befehl kopiert die beiden Dateien, die mit .txt enden, in das Verzeichnis / tmp /?
A. cp ??. Txt / tmp /
B. cp * .txt / tmp /
C. vgl. \ .txt / tmp /
D. cp? .Txt / tmp /
E. cp $?. Txt / tmp /
Answer: B
Explanation:
The correct command to copy the two files ending in .txt to the /tmp/ directory is cp *.txt /tmp/. This command uses the wildcard character * to match any number of characters before the .txt extension. Therefore, it will copy both texts 1.txt and texts 2.txt to the destination directory /tmp/. The other options are incorrect because they use different wildcard characters or syntax that do not match the desired files. For example, option A uses ?? to match exactly two characters before the .txt extension, but the files have a space and a number, which are

not considered as one character. Option C uses a backslash \ to escape the dot . before the .txt extension, but this is unnecessary and will cause the command to fail. Option D uses ? to match exactly one character before the .txt extension, but the files have more than one character. Option E uses $? to match the exit status of the previous command before the .txt extension, but this is not relevant and will cause the command to fail123 Reference: 1: Linux wildcards | How do wildcards work in Linux with examples? - EDUCBA 2: Wildcards in Linux explained with 10 examples | FOSS Linux 3: What are wildcard characters in Linux? - Sage -Answers