2.5.5 Practice Questions

Candidate: Ethan Bonavida (suborange)
Date: 11/8/2022 12:08:36 am • **Time Spent:** 01:10

Score: 100% Passing Score: 80%

Question 1: ✔ Correct

Which of the following commands shows the value of the **LANG** environmental variable currently set for the language the operating system uses?

- **echo %LANG**
- **echo $LANG**
- **echo %LANG%**
- **echo LANG**

Explanation

The **echo** command displays the results of an expression. An expression formed with a dollar sign (**$**) followed by a variable name results in the assigned valued of the variable.

The **echo LANG** command gives the results "LANG" without translating the variable to its assigned value.

The **echo %LANG** command gives the results "%LANG" without translating the variable to its assigned value. The percent (%) character is used for substitutions in Windows command line scripts.

The **echo %LANG%** command gives the results "%LANG%" without translating the variable to its assigned value. The percent (%) character is used for substitutions in Windows command line scripts.

References

- 2.5.1 Environment Variables
- 2.5.2 Manage Environment Variables
- 2.5.3 Environment Variable Facts
- 14.1.1 Bash Scripting Overview
- 14.1.2 Bash Script Execution
- 14.2.1 Bash Shell Environments and Shell Variables
- 14.2.2 Bash Shell Parameters, User Variables and Expansions
- 14.2.3 Bash Shell Variables and Parameters

14.2.4 User Variables and Shell Arithmetic

14.2.5 Arrays and Expansions

14.2.6 Shell Environments, Bash Variables and Parameters Facts

q_envvfct_lp5_01.question.fex

Question 2: ✔ Correct

You want to view the number of commands your bash shell is set to save by examining the current HISTSIZE environment variable. You don't want to have to scroll through all the environment variables.

Which of the following commands is the BEST way to determine the current value of the HISTSIZE variable?

- **cat /etc/profile | grep "HISTSIZE"**
- **cat $HISTSIZE**
- **echo $HISTSIZE**
- **HISTSIZE=**

Explanation

The **echo $HISTSIZE** command returns the value for the HISTSIZE environment variable.

The **HISTSIZE=[new value]** command is used to change the HISTSIZE environment variable value.

While the **cat /etc/profile | grep "HISTSIZE"** command displays the default HISTSIZE value, the value may have changed after it was set when the /etc/profile file was run.

Typically, the value of the HISTSIZE variable is 1000. If $HISTSIZE is replace with 1000 in **cat $HISTSIZE**, the command executed would be **cat 1000**. If there is a file named 1000, the contents of the file are displayed; otherwise, the cat command returns a "No such file or directory" message.

References

- 2.5.1 Environment Variables
- 2.5.2 Manage Environment Variables
- 2.5.3 Environment Variable Facts
- 14.1.1 Bash Scripting Overview
- 14.1.2 Bash Script Execution
- 14.2.1 Bash Shell Environments and Shell Variables

14.2.2 Bash Shell Parameters, User Variables and Expansions

14.2.3 Bash Shell Variables and Parameters

14.2.4 User Variables and Shell Arithmetic

14.2.5 Arrays and Expansions

14.2.6 Shell Environments, Bash Variables and Parameters Facts

q_envvfct_lp5_02.question.fex

Question 3: ✔ Correct

Which command displays all the environment variables defined in the shell?

env ✔

Explanation

Both the **printenv** and **env** commands display the values for environment variables defined in the shell.

References

- 2.5.1 Environment Variables
- 2.5.2 Manage Environment Variables
- 2.5.3 Environment Variable Facts
- 14.1.1 Bash Scripting Overview
- 14.1.2 Bash Script Execution
- 14.2.1 Bash Shell Environments and Shell Variables
- 14.2.2 Bash Shell Parameters, User Variables and Expansions
- 14.2.3 Bash Shell Variables and Parameters
- 14.2.4 User Variables and Shell Arithmetic
- 14.2.5 Arrays and Expansions
- 14.2.6 Shell Environments, Bash Variables and Parameters Facts

q_envvfct_lp5_03.question.fex

Question 4: ✔ Correct

You recently used the **HOST=FS4** command.

Which of the following commands should you use to change the HOST variable to a global variable that will be inherited by subsequent child shells and processes?

- **export HOST**
- **env HOST**
- **set HOST**
- **unset HOST**

Explanation

The **export HOST** command changes the HOST variable to an environment variable.

The **set** command with no options or arguments displays the names and values of all variables and functions. Otherwise, the **set** command is used to manipulate shell options and positional parameters that are mostly used in shell scripting. The **set HOST** command essentially sets the first positional parameter ($1) to HOST, which can be verified by subsequently running the **echo $1** command.

The **env HOST** command attempts to run the HOST command. Normally, environment variables (not present in this command) are temporarily set before the command is run. Consequently, this command will likely give the message "No such file or directory," since HOST is not a command and there is no executable file named HOST.

The **unset HOST** command attempts to remove the HOST variable. If the variable exists, it is removed. If not, no warning message is displayed.

References

- 2.5.1 Environment Variables
- 2.5.2 Manage Environment Variables
- 2.5.3 Environment Variable Facts
- 14.1.1 Bash Scripting Overview
- 14.1.2 Bash Script Execution
- 14.2.1 Bash Shell Environments and Shell Variables

14.2.2 Bash Shell Parameters, User Variables and Expansions

14.2.3 Bash Shell Variables and Parameters

14.2.4 User Variables and Shell Arithmetic

14.2.5 Arrays and Expansions

14.2.6 Shell Environments, Bash Variables and Parameters Facts

q_envvfct_lp5_04.question.fex

Question 5: ✔ Correct

Which environment variable affects the number of past commands stored in memory in the current shell session?

HISTSIZE ✔

Explanation

The HISTSIZE environment variable specifies the number of past commands stored in memory in the current shell session.

The HISTFILESIZE shell variable holds the number of past commands remembered between sessions.

References

- 2.5.1 Environment Variables
- 2.5.2 Manage Environment Variables
- 2.5.3 Environment Variable Facts
- 14.1.1 Bash Scripting Overview
- 14.1.2 Bash Script Execution
- 14.2.1 Bash Shell Environments and Shell Variables
- 14.2.2 Bash Shell Parameters, User Variables and Expansions
- 14.2.3 Bash Shell Variables and Parameters
- 14.2.4 User Variables and Shell Arithmetic
- 14.2.5 Arrays and Expansions
- 14.2.6 Shell Environments, Bash Variables and Parameters Facts

q_envvfct_lp5_05.question.fex

Question 6: ✔ Correct

Which of the following commands configures the shell to retain 300 recently used commands in the **~/.bash_history** file for multiple shell sessions?

- **HISTSIZE=300**
- **HISTFILE=300**
- **BASH=300**
- **HISTFILESIZE=300**

Explanation

The **HISTFILESIZE=300** command sets the number of past commands remembered between multiple sessions and stored in the **~/.bash_history** file.

The HISTSIZE environment variable specifies the number of past commands remembered for the current shell session.

The HISTFILE shell variable specifies the filename where past commands are stored, which is **~/.bash_history** by default.

The BASH environment variable specifies the location of the bash executable file, which is normally **/user/bin/bash**.

References

- 2.5.1 Environment Variables
- 2.5.2 Manage Environment Variables
- 2.5.3 Environment Variable Facts
- 14.1.1 Bash Scripting Overview
- 14.1.2 Bash Script Execution
- 14.2.1 Bash Shell Environments and Shell Variables
- 14.2.2 Bash Shell Parameters, User Variables and Expansions
- 14.2.3 Bash Shell Variables and Parameters
- 14.2.4 User Variables and Shell Arithmetic

14.2.5 Arrays and Expansions

14.2.6 Shell Environments, Bash Variables and Parameters Facts

q_envvfct_lp5_06.question.fex

Question 7: ✔ Correct

You need to set the COMP variable to the value 1745.

Which of the following commands sets the variable so it is inherited by subsequent child shells?

- set COMP to 1745
- **export COMP=1745**
- COMP=1745
- set COMP=1745

Explanation

The **export COMP=1745** command creates the COMP variable, assigns it the value of 1745, and sets the export attribute to make it an environment variable that will be inherited by subsequent child shells.

The **set COMP to 745** command sets the value of the $1 positional variable to "COMP", the value of the $2 positional variable to "to", and the value of the $3 positional variable to "1745".

The **set COMP=1745** command sets the value of the $1 positional variable to "COMP=1745".

The **COMP=1745** command simply creates a shell variable named COMP and sets its value to 1745. This shell variable is not inherited by subsequent child shells.

References

- 2.5.1 Environment Variables
- 2.5.2 Manage Environment Variables
- 2.5.3 Environment Variable Facts
- 14.1.1 Bash Scripting Overview
- 14.1.2 Bash Script Execution
- 14.2.1 Bash Shell Environments and Shell Variables
- 14.2.2 Bash Shell Parameters, User Variables and Expansions
- 14.2.3 Bash Shell Variables and Parameters

14.2.4 User Variables and Shell Arithmetic

14.2.5 Arrays and Expansions

14.2.6 Shell Environments, Bash Variables and Parameters Facts

q_envvfct_lp5_07.question.fex

Question 8: ✔ Correct

Two users should have identical settings, yet one is having problems with the display on his screen, and you suspect there is a difference in their environment variables.

Which of the following commands displays all of the environment variables?

- **env**
- **echo**
- **display**
- **print**

Explanation

The **env** command displays a list of all the environment variables and their values.

The **echo** command with no arguments simply displays a blank line.

The **display** command will most likely display the "command not found" message.

The **print** command will most likely display the "command not found" message. However, the **printenv** command will display a list of all the environment variables and their values.

References

2.5.3 Environment Variable Facts

q_envvfct_lp5_08.question.fex

Question 9: ✔ Correct

Which of the following statements BEST describes the PATH environment variable?

- ➡ It contains the directory prefixes used to search for programs and files.
- It contains the path of the current working directory.
- It specifies the filename where past commands are stored.
- It specifies the characters the shell uses to indicate normal user ($), root user (#), and similar items.

Explanation

The PATH environment variable contains the directory prefixes used to search for programs and files. Use a colon to separate multiple directories in the PATH variable.

The HISTFILE shell variable specifies the filename where past commands are stored.

The PS1 shell variable specifies the characters the shell prompt that indicates either a normal user ($) or root user (#).

The PWD environment variable contains the path of the current (or present) working directory.

References

- **2.5.1 Environment Variables**
- **2.5.2 Manage Environment Variables**
- **2.5.3 Environment Variable Facts**
- **14.1.1 Bash Scripting Overview**
- **14.1.2 Bash Script Execution**
- **14.2.1 Bash Shell Environments and Shell Variables**
- **14.2.2 Bash Shell Parameters, User Variables and Expansions**

14.2.3 Bash Shell Variables and Parameters

14.2.4 User Variables and Shell Arithmetic

14.2.5 Arrays and Expansions

14.2.6 Shell Environments, Bash Variables and Parameters Facts

q_envvfct_lp5_09.question.fex

Question 10: ✔ Correct

You want the directory /sbin/special to be include in the PATH environment variable. You also want to keep all the current directory entries currently in the PATH variable.

Which of the following commands would you use?

- **PATH=PATH&/sbin/special**
- **PATH=$PATH:/sbin/special**
- **PATH=/sbin/special**
- **PATH=$PATH:$/sbin/special**

Explanation

A colon separates entries in the PATH statement. The **PATH=$PATH:/sbin/special** command appends a colon (:) and the /sbin/special directory to the existing PATH ($PATH).

The **PATH=$PATH:$/sbin/special** command corrupts the PATH variable by append a dollar sign ($) and the /sbin/special directory to it.

The **PATH=/sbin/special** command replaces the current PATH with only the /sbin/special directory.

The ampersand (&) in the **PATH=PATH&/sbin/special** command causes the **PATH=PATH** command to be run in a child process and then the **/sbin/special** command to run. While this command is interpreted by the bash shell, it does not modify the PATH variable.

References

- 2.5.1 Environment Variables
- 2.5.2 Manage Environment Variables
- 2.5.3 Environment Variable Facts
- 14.1.1 Bash Scripting Overview
- 14.1.2 Bash Script Execution
- 14.2.1 Bash Shell Environments and Shell Variables
- 14.2.2 Bash Shell Parameters, User Variables and Expansions

14.2.3 Bash Shell Variables and Parameters

14.2.4 User Variables and Shell Arithmetic

14.2.5 Arrays and Expansions

14.2.6 Shell Environments, Bash Variables and Parameters Facts

q_envvfct_lp5_10.question.fex

Question 11: ✔ Correct

Tim, a technician, created a local variable named val and set it to 5000 at the bash prompt. Tim wants to use the variable in a script. But when the script is executed, the value of val is not set to 5000.

Which of the following commands would allow Tim to set a global variable that would be available to the script?

- **declare val**
- **export val=5000**
- **echo $val=5000**
- **exec val**

Explanation

export val=5000 will create a global variable named val and set the value to 5000. The variable val will be available to any scripts or applications run in that terminal session.

declare val is used to declare a variable. In this case, no value would be assigned. The scope of the variable is not global.

echo $val=5000 will display the value of $val, which is empty, and then displays =5000 as the output.

exec val will display the error "bash: exec: val: not found."

References

- 2.5.1 Environment Variables
- 2.5.2 Manage Environment Variables
- 2.5.3 Environment Variable Facts
- 14.1.1 Bash Scripting Overview
- 14.1.2 Bash Script Execution
- 14.2.1 Bash Shell Environments and Shell Variables
- 14.2.2 Bash Shell Parameters, User Variables and Expansions
- 14.2.3 Bash Shell Variables and Parameters

14.2.4 User Variables and Shell Arithmetic

14.2.5 Arrays and Expansions

14.2.6 Shell Environments, Bash Variables and Parameters Facts

q_envvfct_lp5_export.question.fex

Question 12: ✔ Correct

Anna, a system administrator, wants to set a global variable that can be used each time she logs in. Currently, she has to set the variable each time a terminal is opened.

Which of the following files would need to be modified to make a global variable persistent? (Choose TWO.)

- [ ] ~./bash_logout
- [ ] ~./bash_history
- [x] ~./bash_profile
- [x] /etc/profile

Explanation

~./bash_profile and /etc/profile can be modified to add the global variable declaration. Each time Anna logs on, the variable will be set.

~./bash_history stores the command history.

~./bash_logout is used to perform cleanup when exiting a bash shell.

References

- 2.5.1 Environment Variables
- 2.5.2 Manage Environment Variables
- 2.5.3 Environment Variable Facts
- 14.1.1 Bash Scripting Overview
- 14.1.2 Bash Script Execution
- 14.2.1 Bash Shell Environments and Shell Variables
- 14.2.2 Bash Shell Parameters, User Variables and Expansions
- 14.2.3 Bash Shell Variables and Parameters
- 14.2.4 User Variables and Shell Arithmetic
- 14.2.5 Arrays and Expansions

14.2.6 Shell Environments, Bash Variables and Parameters Facts

q_envvfct_lp5_global.question.fex