BASH

Interview Questions

Q: What does BASH stand for?

A: BASH stands for 'Bourne Again SHell', which is an improved version of the original Bourne Shell (sh).

Q: What is the primary purpose of BASH?

A: The primary purpose of BASH is to provide a command-line `interface` for interacting with the operating system, allowing users to execute commands, run scripts, manage files, and automate tasks.

Q: How do you create a `new` BASH script?

A: To create a `new` BASH script, you can use a text editor to create a file with a .sh extension. Start the script with the shebang line '#!/bin/bash' to indicate that it should be run with BASH.

Q: What is the significance of the shebang (#!) in a BASH script?

A: The shebang (#!) indicates the script's interpreter. When a script is executed, the operating system uses the path following the shebang to determine which interpreter to use to run the script.

Q: How would you make a BASH script executable?

A: You can make a BASH script executable by running the command 'chmod +x script.sh', where 'script.sh' is the name of your script.

Q: What is a variable in BASH and how do you declare one?

A: A variable in BASH is a named storage location that holds data. You can declare a variable by simply writing 'variable_name=value' without spaces. For example, 'my_var=10'.

Q: How do you access the value of a variable in BASH?

A: To access the value of a variable in BASH, prepend the variable name with a dollar sign ($). For example, to access 'my_var', you would use '$my_var'.

Q: What is the purpose of the 'echo' command in BASH?

A: The 'echo' command is used to display text or the value of variables to the terminal. It can also be used to print the output of commands.

Q: Explain the difference between '==' and '=' in BASH.

A: '==' is used for `string` comparison within double square brackets [[ ]], `while` '=' is used for assignment and can be used for `string` comparison in single brackets [ ]. In conditional expressions, '==' is preferred for clarity.

Q: What does the 'if' statement do in a BASH script?

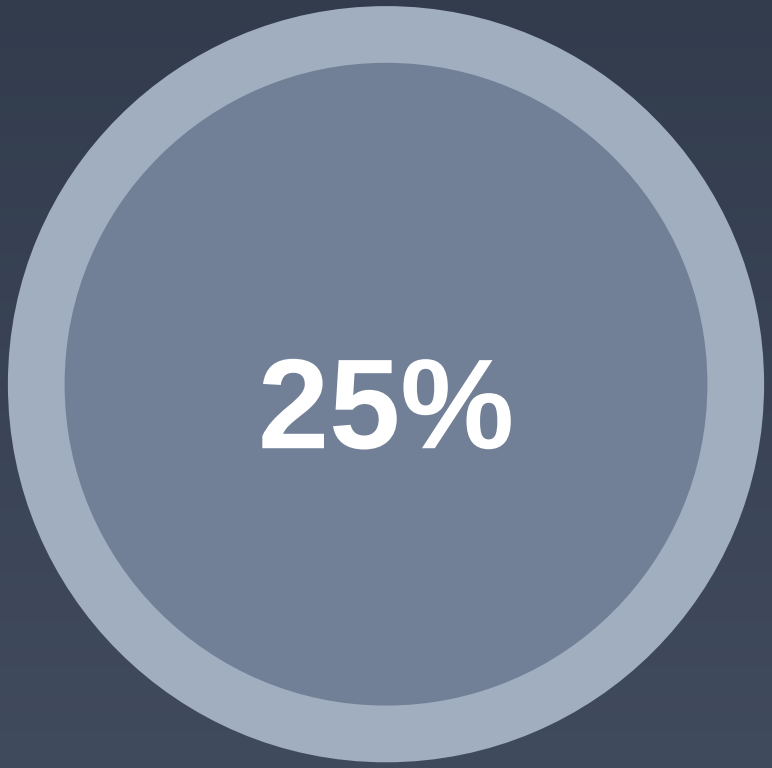
A: The 'if' statement allows for conditional execution of commands based on the evaluation of a condition. If the condition is true, the commands within the 'if' block are executed.

Q: How do you loop through a list of items in BASH?

A: You can loop through a list of items using a 'for' loop. For example: 'for item in item1 item2 item3; do echo $item; done'.

Q: What is the purpose of the '`case`' statement in BASH?

A: The '`case`' statement is used for multi-way branching, allowing the execution of different blocks of code based on the value of a variable or expression.



You're 25% through! Keep going! Success is built one step at a time.

Q: How do you read user input in a BASH script?

A: You can read user input using the 'read' command. For example: 'read -p "Enter your name: " name' will prompt the user and store the input in the variable 'name'.

Q: What are positional parameters in BASH?

A: Positional parameters are variables that represent the arguments passed to a script. They are accessed using '$1', '$2', etc., where '$1' is the first argument, '$2' is the second, and so on.

Q: What does the '&&' operator do in BASH?

A: The '&&' operator allows you to run a second command only if the first command succeeds (returns an exit status of 0). For example: 'command1 && command2'.

Q: How do you redirect output to a file in BASH?

A: You can redirect output to a file using the '>' operator. For example: 'command > output.txt' will write the output of 'command' to 'output.txt'. Use '>>' to append to a file.

Q: What is a subshell in BASH?

A: A subshell is a child process created by the shell to execute commands in a separate environment. It is created by enclosing commands in parentheses, e.g., '(command1; command2)'.

Q: How do you check the exit status of the last command executed in BASH?

A: You can check the exit status of the last command executed by examining the special variable '$?'. A value of 0 indicates success, `while` `any` non-zero value indicates an error.

Q: What is the purpose of the 'trap' command in BASH?

A: The 'trap' command allows you to specify commands to be executed when the shell receives certain signals or events, such as SIGINT (Ctrl+C). It is useful for cleanup operations.

Q: Explain how to create a `function` in BASH.

A: You can create a `function` in BASH using the following syntax: 'function_name() { commands; }'. For example: 'my_function() { echo "Hello"; }'. Call the `function` by its name.

Q: What is 'wildcard' in BASH?

A: Wildcards are special characters used to represent one or more characters in filenames. For example, '*' represents zero or more characters, `while` '?' represents a single character.

Q: How can you find the current working directory in BASH?

A: You can find the current working directory by using the 'pwd' command, which stands for 'print working directory'.

Q: What is 'grep' and how is it used in BASH?

A: 'grep' is a command-line utility used to search for specific patterns within files or output. It can be used like `this` : 'grep pattern filename' to find lines matching 'pattern' in 'filename'.

Q: How do you pass options to a BASH script?

A: You can pass options to a BASH script by including them as arguments when executing the script. For example : './script.sh -a -b'. Inside the script, you can access these options using positional parameters.

Halfway there! Every expert was once a beginner.

Q: What is the difference between 'source' and '.' in BASH?

A: Both 'source' and '.' are used to execute commands from a file in the current shell environment. They are interchangeable, but 'source' is more readable.

Q: How would you check if a file exists in BASH?

A: You can check if a file exists using the '-e' flag in a conditional statement. For example: 'if [-e filename]; then echo "File exists"; fi'.

Q: What does BASH stand for?

A: BASH stands for 'Bourne Again SHell'. It is an enhanced version of the original Bourne shell (sh) and includes features from the Korn shell (ksh) and the C shell (csh).

Q: How do you declare a variable in BASH?

A: To declare a variable in BASH, simply use the syntax: variable_name=value. For example, '`myVar` =10'. Note that there should be no spaces around the '=' sign.

Q: What is the purpose of the '`export`' command?

A: The '`export`' command is used to set environment variables in BASH. When you `export` a variable, it becomes available to `any` subprocesses or child shells that are spawned from the current shell session.

Q: Explain the difference between '==' and '=' in BASH.

A: '==' is typically used for `string` comparison in conditional expressions, `while` '=' is used for assignment. In BASH scripts, when comparing strings, it is common to use '=' within '[ ]' or '[[ ]]'. For example: '["$a" = "$b"]'.

Q: What is a shebang, and why is it important?

A: A shebang is a character sequence at the beginning of a script file that indicates which interpreter should be used to execute the script. It usually looks like '#!/bin/bash'. It is important because it ensures that the script runs in the correct shell environment.

Q: How can you pass arguments to a BASH script?

A: Arguments can be passed to a BASH script by including them after the script name in the command line. Inside the script, you can access these arguments using positional parameters like '$1', '$2', and so on, where '$1' refers to the first argument.

Q: What is the purpose of the 'set -e' command?

A: 'set -e' is used in BASH scripts to instruct the shell to immediately exit if `any` command within the script returns a non-zero exit status. `This` is useful for error handling, ensuring that the script stops execution upon encountering an error.

Q: How do you create a `function` in BASH?

A: Functions in BASH can be defined using the following syntax: 'function_name() { commands; }'. For example: '`myFunction` () { echo "Hello, World!"; }'. You can then call the `function` by simply using its name.

Q: What is the difference between '>' and '>>' in BASH?

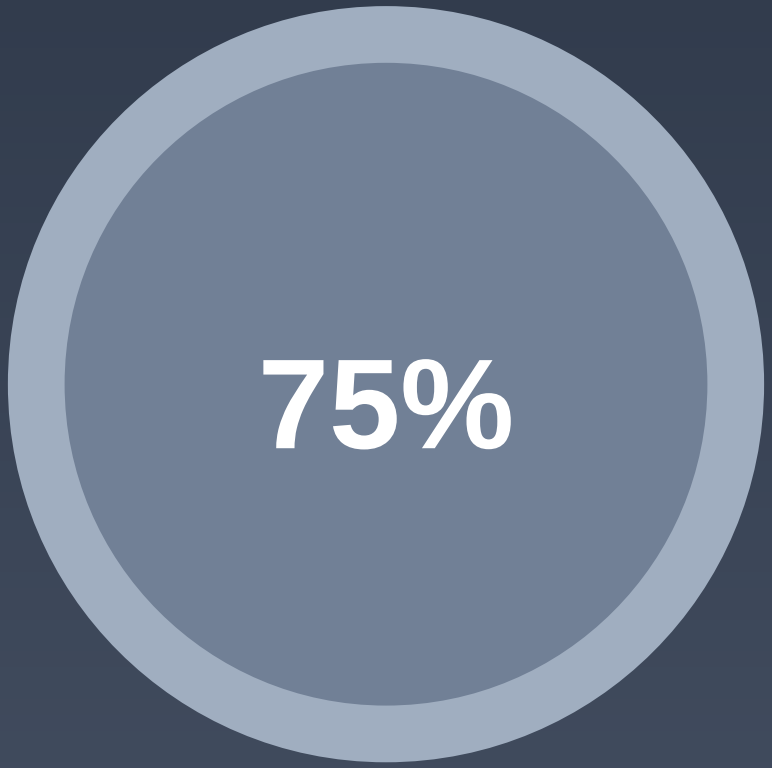
A: '>' is used for redirecting output to a file, overwriting the file if it exists. '>>' is used for appending output to a file without overwriting it. For example, 'echo "Hello" > file.txt' will overwrite file.txt, `while` 'echo "World" >> file.txt' will append 'World' to the end of the file.

Q: What is a subshell in BASH?

A: A subshell is a child shell that is created by the current shell to execute a command. It is often created by enclosing commands in parentheses, e.g., '(command1; command2)'. Variables changed in a subshell do not affect the parent shell.

Q: Explain the use of '&&' and '||' operators in BASH.

A: '&&' is a logical AND operator that executes the second command only if the first command succeeds (returns a zero exit status). '||' is a logical OR operator that executes the second command only if the first command fails (returns a non-zero exit status).



You're at 75%! Almost done, push through and finish strong!

Q: How do you read user input in a BASH script?

A: You can use the 'read' command to read user input. For example, 'read -p "Enter your name: " name' prompts the user for input and stores it in the variable 'name'.

Q: What are arrays in BASH, and how do you declare them?

A: Arrays in BASH are variables that can store multiple values. You can declare an `array` using the syntax: 'array_name=(value1 value2 value3)'. You can access elements of the `array` using 'array_name[index]'.

Q: How do you check if a file exists in BASH?

A: You can check if a file exists using the '-e' test operator within an '[ ]' or '[[ ]]'. For example: 'if [-e filename]; then echo "File exists"; fi'.

Q: What is the significance of the '$?' variable in BASH?

A: '$?' is a special variable that holds the exit status of the last command executed. A value of '0' typically indicates success, `while` `any` other value indicates an error or failure.

Q: Explain the use of wildcards in BASH.

A: Wildcards are special characters used in BASH to match filenames or patterns. For example, '*' matches `any number` of characters, '?' matches a single character, and '[]' can be used to specify a range of characters.

Q: What does the 'trap' command do in BASH?

A: The 'trap' command is used to specify commands to execute when the shell receives certain signals or when the script exits. It is useful for cleanup tasks or handling interruptions gracefully.

Q: How can you find out which shell you are currently using?

A: You can find out which shell you are currently using by executing the command 'echo $SHELL' or 'ps -p $$'. These commands will display the path of the current shell.

Q: What is the difference between 'if' and '`case` ' statements in BASH?

A: 'if' statements are used for conditional execution based on `boolean` expressions, `while` '`case` ' statements are used for pattern matching against a variable's value. '`case` ' is often more readable for multiple conditions.

Q: How do you make a script executable?

A: To make a script executable, you need to change its permissions using the 'chmod' command. For example, 'chmod +x script.sh' makes 'script.sh' executable.

Q: What is a here document in BASH?

A: A here document (heredoc) allows you to redirect multi-line text input to a command. It starts with '<<' followed by a delimiter. For example: 'cat << EOF Hello, World! EOF' will print 'Hello, World!'.

Q: How can you schedule a script to run at a specific time in BASH?

A: You can schedule a script to run at a specific time using the 'cron' service in Linux. You can edit the crontab file using 'crontab -e' and add a line specifying the timing and the command to execute.

Q: What is the difference between 'local' and 'global' variables in BASH?

A: Local variables are those that are declared within a `function` and are only accessible within that function. Global variables can be accessed from anywhere within the script. You can declare a local variable using the 'local ' keyword.

Q: What is the difference between a shell variable and an environment variable in BASH?

A: In BASH, a shell variable is a variable that is defined and used within the current shell session. It is local to the shell and cannot be accessed by child processes. On the other hand, an environment variable is a `type` of variable that is exported to child processes, allowing them to access its value. Environment variables are typically defined using the '`export` ' command. For example, to create a shell variable, you would use 'myvar=10', and to create an environment variable, you would use '`export` myvar=10'. `This` distinction is important for controlling the scope and `accessibility` of variables in scripts and shell sessions.

Thank You!

You've completed all the questions.

""The expert in anything was once a beginner." — Helen Hayes"