**Shell Scripting Fundamentals**

* **Review Shell Syntax:**
  + Study variable assignment and usage (e.g., C=$A+$B, D=$(($A+$B))).
  + Practice using if, elif, and else conditional statements.
  + Understand how to use for and while loops.
  + Pay attention to command-line arguments and how to access them (e.g., $1, $#, $@).

|  |
| --- |
| * #!/bin/bash * if [ -f names.txt ]; then * echo "regular file" # Checks if 'names.txt' is a regular file. * fi * if [ -d ./ ]; then * echo "it's directory" # Checks if the current directory './' is a directory. * fi * if [ -e names.txt ]; then * echo "File exists" # Checks if 'names.txt' exists (file or directory). * fi * if [ -r names.txt ]; then * echo "u can read" # Checks if 'names.txt' is readable. * fi * if [ -w names.txt ]; then * echo "u can write" # Checks if 'names.txt' is writable. * fi * if [ -e names.txt ]; then * echo "u can execute" # This condition should be -x to check execute permission, not -e. -e just checks existence * fi * if [ logs.txt -ot names.txt ]; then * echo "logs" # Checks if 'logs.txt' is older than 'names.txt'. * else * echo names # If logs.txt is not older than names.txt, it prints "names", not the content of names.txt * fi * if [ $# -ge 1 ]; then * echo "I am happy" # Checks if the number of command-line arguments ($#) is greater than or equal to 1. * echo $1 # Prints the first command-line argument ($1). * elif [ -n "$2" ] && [ -z "$2" ]; then * echo "I am NOT happy" # This condition will \*never\* be true. A string cannot be both non-empty (-n) AND empty (-z) at the same time. * fi * if [ -z names.txt ]; then * echo "Nothing inside" # This checks if the \*string\* "names.txt" is empty, not if the file is empty. It will always be false. * fi * file=$(cat name.txt 2>> error.txt) # Capture standard output of cat name.txt, redirect standard error to error.txt * if [ -z "$file" -a 2 -ge 1 ]; then * echo "Nothing inside" # Checks if $file is empty AND if 2 is greater than or equal to 1 (which is always true). * else * echo "Where's the fuck did u find it?" # This part will almost always execute, unless cat name.txt produces no output AND an error * fi |

|  |  |
| --- | --- |
| * 10 * 1+1 * 2 * 3 * hi, there * 5 * 10 * 76 syntax.sh * 76 * names.txt * You gave me 1 this much of variables * h * e * l * l * o * w * hello * 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 | * 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 * John Smith * Alice Wonderland * Bob The Builder * Carol Danvers * Peter Parker * Bruce Wayne * John * Smith * Alice * Wonderland * Bob * The * Builder * Carol * Danvers * Peter * Parker * Bruce * Wayne |

* **Mastering Conditional Expressions:**
  + Familiarize yourself with the syntax for file tests (e.g., -f, -z).
  + Practice using string comparisons (e.g., =, !=) and integer comparisons (e.g., -gt, -lt, -eq, -ne).
  + Understand how to combine conditions using -a (and) and -o (or).

|  |
| --- |
| * #!/bin/bash * if [ -f names.txt ]; then # If names.txt is a regular file * echo "regular file" * fi * if [ -d ./ ]; then # If current dir is a directory * echo "it's directory" * fi * if [ -e names.txt ]; then # If names.txt exists * echo "File exists" * fi * if [ -r names.txt ]; then # If names.txt is readable * echo "u can read" * fi * if [ -w names.txt ]; then # If names.txt is writable * echo "u can write" * fi * if [ -x names.txt ]; then # If names.txt is executable (was -e, should be -x) [cite: 6] * echo "u can execute" * fi * if [ logs.txt -ot names.txt ]; then # If logs.txt is older than names.txt * echo "logs" * else * echo names * fi * if [ $# -ge 1 ]; then # If at least 1 argument * echo "I am happy" * echo $1 * elif [ -n "$2" ] && [ -z "$2" ]; then # This is always false * echo " I am NOT happy" * fi * if [ -z names.txt ]; then # If "names.txt" is empty (wrong check) * echo "Nothing inside" * fi * file=$(cat name.txt 2>> error.txt) # Get file content, redirect errors * if [ -z "$file" -a 2 -ge 1 ]; then # If file is empty and 2 >= 1 * echo "Nothing inside" * else * echo "Where's the fuck did u find it?" * fi |
| regular file  it's directory  File exists  u can read  u can write  names  I am happy  hi  Nothing inside |

**Commands and Utilities**

* **File Manipulation:**
  + Practice using head, tail, sort, uniq, cut, find, mv, rm.
  + Understand how to redirect input and output (e.g., <, >, >>, 2>>).
  + Review file permissions and how to change them with chmod.
* **Text Processing:**
  + Practice using grep and egrep for pattern matching.

|  |
| --- |
| * egrep "cat|dog" file.txt # Find lines containing either "cat" or "dog" * grep "cat\|dog" file.txt # Equivalent grep command (requires escaping) * egrep "(hello)+" file.txt # Find lines with one or more occurrences of "hello" * grep "\(hello\)\+" file.txt # Equivalent grep command (requires escaping) * egrep "colou?r" file.txt # Find lines with "color" or "colour" * grep "colou\?r" file.txt # Equivalent grep command (requires escaping) |

* Understand how to use wc to count lines, words, and characters.
* **General Commands:**
  + Memorize the basic functions of commands like ls, mkdir, touch, cat, cd, nano, man, clear, nl, file, tr, let, rmdir, and cp.

**Specific Tasks**

* **Write scripts to:**
  + Process command-line arguments.
  + Read files line by line using while loops.
  + Find files based on name or other criteria.
  + Manipulate text within files (e.g., extract specific lines, sort, remove duplicates).
  + Perform calculations using let.
  + Rename or move files.
  + Use conditional statements to make decisions.
* **Given a code snippet, explain what it does.**
* **Identify errors in a given script.**
* **Write one-line commands to perform specific tasks.**

**Text Example 1: data.txt**

Plaintext

apple,10,red,sweet

banana,5,yellow,sweet

orange,12,orange,tart

grape,25,purple,tart

apple,8,green,sour

kiwi,15,brown,sour

banana,7,yellow,sweet

**Text Example 2: logs.txt**

Plaintext

2024-07-20 10:00:01 - User logged in: user123

2024-07-20 10:05:22 - File accessed: /home/user123/document.txt

2024-07-20 10:12:48 - Error: Connection timeout

2024-07-20 11:01:15 - User logged out: user123

2024-07-21 09:22:30 - User logged in: guest

2024-07-21 09:28:45 - File accessed: /var/log/system.log

2024-07-21 10:15:00 - Warning: Disk space low

2024-07-21 11:59:59 - User logged out: guest

**Text Example 3: names.txt**

Plaintext

John Smith

Alice Wonderland

Bob The Builder

Carol Danvers

Peter Parker

Bruce Wayne

**Text Example 4: numbers.txt**

Plaintext

1

10

100

1000

10000

2

20

200

2000

**Example Exercises Using These Texts**

Here are some exercises that utilize these text examples, combining the commands and concepts from your files:

**Using data.txt**

* **Extract fruit names:** Write a command to extract only the fruit names from data.txt.
* **Filter by color:** Write a script to ask the user for a color and then print all lines from data.txt that match that color.
* **Calculate total quantity:** Write a script to calculate the total quantity of all fruits.
* **Find the sweetest fruit:** Write a script to determine which fruit is listed as "sweet" and has the highest quantity. (Similar to the "biggest line" example in codeInShell.txt but applied to different criteria)
* **Count unique fruits:** Write a command to list the unique fruit names.

|  |
| --- |
| * #!/bin/bash * echo "In your basket you have these fruits: " * basket=$(cut -d ',' -f1 text.txt) * for i in $basket;do * echo $i * done * echo "Gimme color to give you your fruit match: " * color="" * read color * while IFS=',' read -r name num ocolor type;do * if [ $ocolor = $color ];then * echo "$name, $num, $ocolor, $type" * fi * done < text.txt * total=$(wc -l < text.txt) * echo "Total quantity of fruits in your basket: $total fruits" * sweet=$(grep "sweet" text.txt | cut -d ',' -f2) * rm quantity.txt * for i in $sweet;do * echo $i >> quantity.txt * done * max=$(sort quantity.txt | head -1) * max=$(grep $max text.txt) * echo "$max" * #fruit\_list=$(cut -d ',' -f1 text.txt | sort | uniq) * #count=$(echo $fruit\_list | wc -w) * #echo "There are $count unique fruits: $fruit\_list" * count=0 * names="" * fruit=$(cut -d ',' -f1 text.txt) * rm names.t * for i in $fruit; do * echo "$i" >> names.t * done * exist=0 * while read -r name; do * for j in $names; do * if [ "$j" = "$name" ]; then * exist=1 * fi * done * if [ "$exist" -eq 0 ]; then * names="$names $name" * ((count++)) * fi * exist=0 * done < names.t * echo "There're $count of fruits: $names" |
| * In your basket you have these fruits: * apple * banana * orange * grape * apple * kiwi * banana * Gimme color to give you your fruit match: * yellow * banana, 5, yellow, sweet * banana, 7, yellow, sweet * Total quantity of fruits in your basket: 7 fruits * apple,10,red,sweet * There're 5 of fruits: apple banana orange grape kiwi |

**Using logs.txt**

1. **Find login/logout times for a user:** Write a script that takes a username as input and prints the login and logout times for that user.
2. **Count error/warning messages:** Write a command to count the number of lines containing "Error" or "Warning".
3. **Extract file paths:** Write a command to extract the file paths accessed (e.g., /home/user123/document.txt).
4. **Find the latest event:** Write a command to display the last line of the log file.
5. **Analyze user activity:** Write a script to determine how many unique users are in the log file.

|  |
| --- |
| * #!/bin/bash * grep "logged" logs.txt | while read -r lines;do * echo $lines | grep "in" | while read -r line;do * echo "$line ---1" * done * echo $lines | grep "out" | while read -r line;do * echo "$line ---2" * done * done * count=$(grep "logged in" logs.txt | wc -l) * echo "Before starting, there --$count-- of user in the system" * logedin=$(grep "logged in" logs.txt | grep $1 | cut -d ' ' -f2) * logedout=$(grep "logged out" logs.txt | grep $1 | cut -d ' ' -f2) * echo "User log time: " * echo " Logged in: $logedin" * echo " Logged out: $logedout" * count=0 * while read -r line;do * for i in $line;do * if [ $i = $1 ];then * echo "$line" >> "$1.txt" * ((count++)) * break * elif [ $count -eq 1 ];then * echo "$line" >> "$1.txt" * fi * done * done < logs.txt * #lines="" * rm "n$1.txt" * cat "$1.txt" | uniq | while read -r line;do * echo $line >> n$1.txt * #lines="$lines\n$line" * done * rm "$1.txt" * count=$(egrep 'Error|Warning' n$1.txt | wc -l) * echo "$1 user got $count error/warning messages" * path=$(cut -d ' ' -f6 n$1.txt | egrep '/\*/') * echo "$1 user accessed this pathes: $path" * echo "$1 user had done before logging out: $(tail -2 n$1.txt | head -1 | cut -d ":" -f4)" |
| * 2024-07-20 10:00:01 - User logged in: user123 ---1 * 2024-07-20 11:01:15 - User logged out: user123 ---2 * 2024-07-21 09:22:30 - User logged in: guest ---1 * 2024-07-21 11:59:59 - User logged out: guest ---2 * Before starting, there --2-- of user in the system * User log time: * Logged in: 10:00:01 * Logged out: 11:01:15 * user123 user got 1 error/warning messages * user123 user accessed this pathes: /home/user123/document.txt * user123 user had done before logging out: Connection timeout |

**Using names.txt**

1. **Extract first names:** Write a command to extract the first names.
2. **Reverse the names:** Write a script to print the names in the format "Smith, John".
3. **Count names with a specific letter:** Write a script to count how many names contain the letter "a" (case-insensitive).
4. **Print names in uppercase:** Write a command to print all names in uppercase.
5. **Sort names alphabetically:** Write a command to sort the names alphabetically.

|  |
| --- |
| * #!/bin/bash * first\_name=$(cut -d ' ' -f1 names.txt) * if [ -e surnames.txt ];then * rm surnames.txt * fi * for a in $first\_name;do * echo "$(grep $a names.txt | cut -d ' ' -f2), $a" >> surnames.txt * echo "$(echo $a | tr [:lower:] [:upper:])" * done * let count=$(cut -d ' ' -f1 names.txt | egrep 'a|A' | wc -w)+$(cut -d ' ' -f2 names.txt | egrep 'a|A' | wc -w) * echo "There are --$count-- of names that contain lettar \"a\"" * #echo "$(cut -d ' ' -f1 names.txt | cut -c1 | tr [:lower:] [:upper:])$(cut -d ' ' -f1 names.txt | cut -c2-) " * cut -d ',' -f2 surnames.txt | sort | while read -r line;do * echo "$(grep $line surnames.txt)" * done |
| * JOHN * ALICE * BOB * CAROL * PETER * BRUCE * There are --6-- of names that contain lettar "a" * Wonderland, Alice * The, Bob * Wayne, Bruce * Danvers, Carol * Smith, John * Parker, Peter |

**Using numbers.txt**

1. **Calculate the sum:** Write a script to calculate the sum of all numbers.
2. **Find numbers greater than a value:** Write a script that takes a number as input and prints all numbers from the file greater than that number.
3. **Find the average:** Write a script to calculate the average of the numbers.
4. **Find the largest and smallest number:** Write a script to determine the largest and smallest numbers.
5. **Filter even numbers:** Write a script to print only the even numbers from the file.

|  |
| --- |
| * #!/bin/bash * let total=0 * read=$(cat numbers.txt) * for i in $read;do * let total=$total+$i * if [ $i -gt $1 ];then * echo $i * fi * #if (( $i%2 == 0 ));then * if [ $(($i%2)) -eq 0 ];then * echo "Even: $i" * fi * done * echo "The sum of numbers.txt is: $total" * echo "The avarage of numbers.txt is: $(($total/$(wc -l < numbers.txt)))" |
| * Even: 10 * 100 * Even: 100 * 1000 * Even: 1000 * 10000 * Even: 10000 * Even: 2 * 20 * Even: 20 * 200 * Even: 200 * 2000 * Even: 2000 * The sum of numbers.txt is: 13333 * The avarage of numbers.txt is: 1481 |

**1. ls (List directory contents)**

* **Basic usage:**

Bash

ls

This will list the files and directories in your current working directory.

* **Long listing:**

Bash

ls -l

This provides more detailed information, including permissions, owner, size, and modification date. (As potentially shown in the image prompt).

**2. mkdir (Create directories)**

* **Basic usage:**

Bash

mkdir new\_directory

This creates a directory named new\_directory in your current location.

**3. touch (Create empty files or update timestamps)**

* **Create a file:**

Bash

touch new\_file.txt

This creates an empty file named new\_file.txt.

**4. cat (Concatenate and display files)**

* **Display a file's content:**

Bash

cat file.txt

This will print the contents of file.txt to your terminal.

**5. cd (Change directory)**

* **Go to a specific directory:**

Bash

cd Documents

This changes your current directory to the Documents directory.

* **Go back to the previous directory:**

Bash

cd -

* **Go to your home directory:**

Bash

cd ~

or just:

Bash

cd

**6. man (Display manual pages)**

* **Get help for a command:**

Bash

man ls

This will display the manual page for the ls command, explaining its options and usage.

**7. chmod (Change file permissions)**

* **Grant read permission to the user for myfile.txt (from codeInShell.txt):**

Bash

chmod u+r myfile.txt

**8. wc (Word count)**

* **Count lines in a file (from codeInShell.txt):**

Bash

n=$(wc -l < file.txt)

echo "$n"

This counts the number of lines in file.txt and stores the result in the variable n.

* **Count characters in a variable (from codeInShell.txt):**

Bash

lines="some text here"

chars=$(echo "$lines" | wc -m)

echo "$chars"

**9. head (Output the first part of files)**

* **Display the first 4 lines of file.txt (from codeInShell.txt):**

Bash

head -n 4 file.txt

or

Bash

T=$(head -4 file.txt)

echo "$T"

**10. tail (Output the last part of files)**

* **Display the last 3 lines of the output of head -n 5 file.txt (from codeInShell.txt):**

Bash

head -n 5 file.txt | tail -n 3

**11. sort (Sort lines of text files)**

* **Sort the contents of file.txt and output to result.txt (from codeInShell.txt):**

Bash

sort "file.txt" > result.txt

**12. uniq (Remove duplicate lines)**

* **Sort file.txt and remove duplicate lines, saving to result.txt (from codeInShell.txt):**

Bash

sort "file.txt" | uniq > result.txt

**13. cut (Remove sections from each line of files)**

* **Example (not directly in your snippets, but common):**

Bash

cut -d':' -f1 /etc/passwd

This would cut the /etc/passwd file, using : as a delimiter, and display only the first field (usernames).

**14. nl (Number lines of files)**

* **Basic usage:**

Bash

nl file.txt

This will number the lines of file.txt.

**15. file (Determine file type)**

* **Check the type of myfile.txt:**

Bash

file myfile.txt

This will tell you if it's a regular file, directory, etc.

**16. find (Find files and directories)**

* **Find files named like $1 in your home directory (from codeInShell.txt):**

Bash

find ~ -name "$1"

If you ran a script with ./my\_script report.txt, this would find files named report.txt in your home directory.

* **Find regular files named t\*.txt in the current directory (from codeInShell.txt):**

Bash

find ./ -type f -name 't\*.txt'

**17. tr (Translate or delete characters)**

* **Example (not directly in your snippets, but common):**

Bash

echo "lowercase" | tr '[:lower:]' '[:upper:]'

This would translate all lowercase characters to uppercase, outputting "LOWERCASE".

**18. let (Perform arithmetic operations)**

* **Add two variables (from codeInShell.txt):**

Bash

A=5

B=10

let C=$A+$B

echo "$C" # Output: 15

D=$(($A+$B)) # Another way to do arithmetic expansion

echo "$D" # Output: 15

* **Increment a variable (from codeInShell.txt):**

Bash

i=0

while [ $i -lt 5 ]; do

echo "$i"

((i+=1)) # Arithmetic increment

done

or

Bash

i=0

while [ $i -lt 5 ]; do

echo "$i"

let i=i+1 # Using let for increment

done

* **Integer division (from codeInShell.txt):**

Bash

result=100

i=0

while [ $i -lt 2 ]; do

((i+=1))

let result=$result/2

echo "$result" # Output: 50, 25

done

**19. rm (Remove files or directories)**

* **Remove a file (from codeInShell.txt):**

Bash

f="temp.txt"

# ... some operations ...

rm "$f"

**Use rm with caution!** It permanently deletes files.

**20. rmdir (Remove empty directories)**

* **Remove an empty directory:**

Bash

rmdir empty\_directory

This will only work if empty\_directory is indeed empty.

**21. cp (Copy files and directories)**

* **Copy a file:**

Bash

cp source.txt destination.txt

* **Copy a directory recursively (to include its contents):**

Bash

cp -r source\_directory destination\_directory

**22. for (Looping construct)**

* **Loop through words in a string (from codeInShell.txt):**

Bash

for l in $(cat file.txt); do

echo "Word: $l"

done

* **Loop through command-line arguments (from codeInShell.txt):**

Bash

for arg in "$@"; do

echo "Argument: $arg"

done

* **Loop through files matching a pattern (from codeInShell.txt):**

Bash

for i in \*.txt; do

egrep '^a.{9}' "$i"

done

* **Loop through a sequence of numbers (from codeInShell.txt):**

Bash

lines="some\ntext\nhere"

for i in $(seq 0 $(($(echo "$lines" | wc -m) - 1))); do

char="${lines:i:1}"

echo "Position $i: $char"

done

**23. while (Looping construct)**

* **Read a file line by line (from codeInShell.txt):**

Bash

while read -r line; do

echo "Line: $line"

done < input.txt

* **Loop based on a condition (from codeInShell.txt):**

Bash

i=0

while [ $i -lt 5 ]; do

echo "$i"

((i+=1))

done

* **Process lines from a command output (from codeInShell.txt):**

Bash

lines=$(egrep "^a" "file.txt")

echo "$lines" | while read -r line; do

echo "Line starting with 'a': $line"

done

**24. grep, egrep (Find patterns in text)**

* **Find lines starting with "a" in file.txt (from codeInShell.txt):**

Bash

lines=$(egrep "^a" "file.txt")

echo "$lines"

* **Find lines starting with "c" in $1 (from codeInShell.txt):**

Bash

lines=$(egrep "^c" "$1")

* **Find lines in \*.txt files starting with 'a' followed by exactly 9 characters (from codeInShell.txt):**

Bash

for i in \*.txt; do

egrep '^a.{9}' "$i"

done