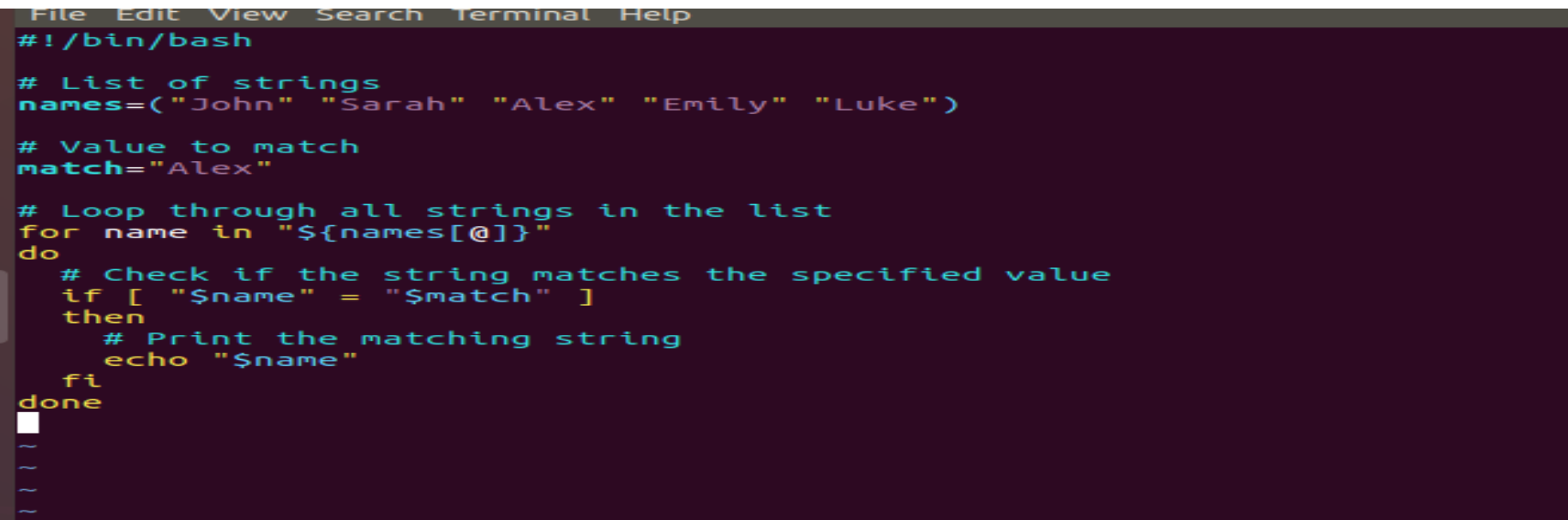


Assignment 11 [string equality]

-Snehal Deshmukh

Create a user case for string equality :-

Suppose you have a list of strings, and you want to print only the strings that match a specific value. You can use a loop in Bash scripting to do this, and string equality will be helpful in checking the values.

A screenshot of a terminal window with a dark purple background. The terminal shows a Bash script being executed. The script defines an array of names, sets a match value, and uses a for loop with an if statement to print only the names that match the specified value. The output shows the name 'Alex' being printed.

```
File Edit View Search Terminal Help
#!/bin/bash

# List of strings
names=("John" "Sarah" "Alex" "Emily" "Luke")

# Value to match
match="Alex"

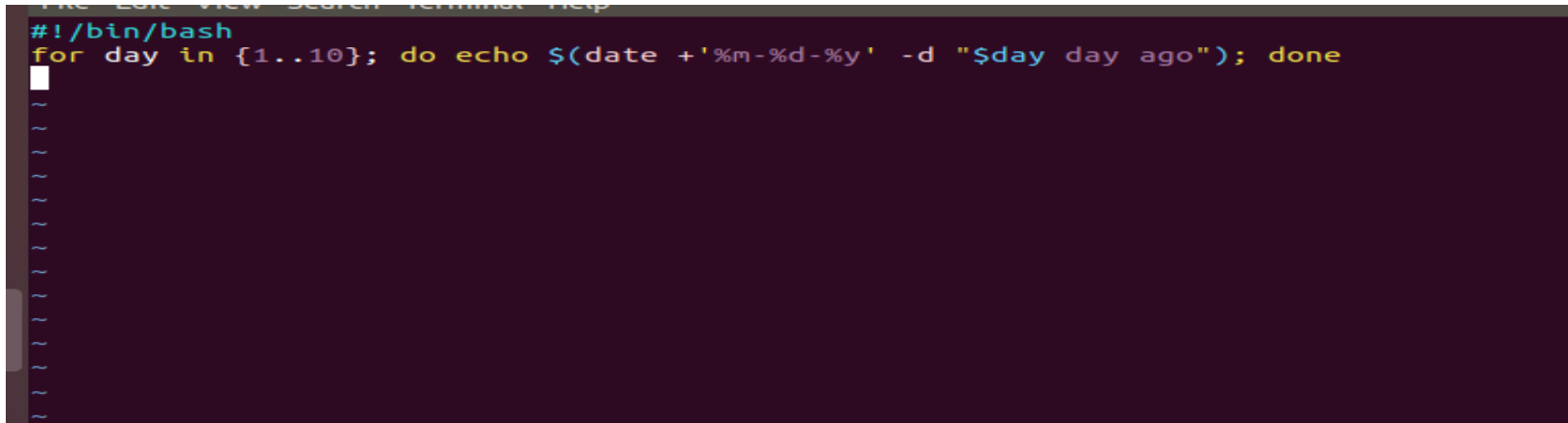
# Loop through all strings in the list
for name in "${names[@]}"
do
    # Check if the string matches the specified value
    if [ "$name" = "$match" ]
    then
        # Print the matching string
        echo "$name"
    fi
done
~
~
~
~
```

Names is an array containing the list of strings, and match is the value you want to match. The for loop iterates over all strings in the list using the "**`${names[@]}`**" syntax. The `[]` test checks if the string `$name` is equal to the specified value `$match` using the `=` operator. **If the string matches the specified value, the echo command prints the matching string.**

```
demo@ubuntu:/home/snehal/loop$ ./strings.sh  
Alex  
demo@ubuntu:/home/snehal/loop$
```

Overall, this use case demonstrates how string equality can be used in loops in Bash scripting to filter and process data based on specific values.

Create a use case for printing date 'mm-dd-yy' for 10 days in a month.

A terminal window with a dark purple background and a light gray sidebar on the left. The terminal shows a bash script being entered. The script is a for loop that iterates from 1 to 10, using the date command to format the date for that day of the month relative to the current date. The format string is '%m-%d-%y'. The output of the script is not visible in the screenshot.

```
#!/bin/bash
for day in {1..10}; do echo $(date +%m-%d-%y' -d "$day day ago"); done
```

In this command, we're using a for loop to iterate through the numbers 1 to 10. For each iteration, we're using the date command to format the date for that day of the month, relative to the current date. Specifically, we're using the -d flag to specify that we want to calculate the date a certain number of days from now (in this case, 1 to 10 days from now), and the + symbol followed by the desired format string to specify the desired date format.

The resulting formatted dates are then printed to the console using the echo command, with each date printed on a new line.

```
demo@ubuntu:/home/snehal/loop$ ./while.sh
```

```
03-06-23
```

```
03-05-23
```

```
03-04-23
```

```
03-03-23
```

```
03-02-23
```

```
03-01-23
```

```
02-28-23
```

```
02-27-23
```

```
02-26-23
```

```
02-25-23
```