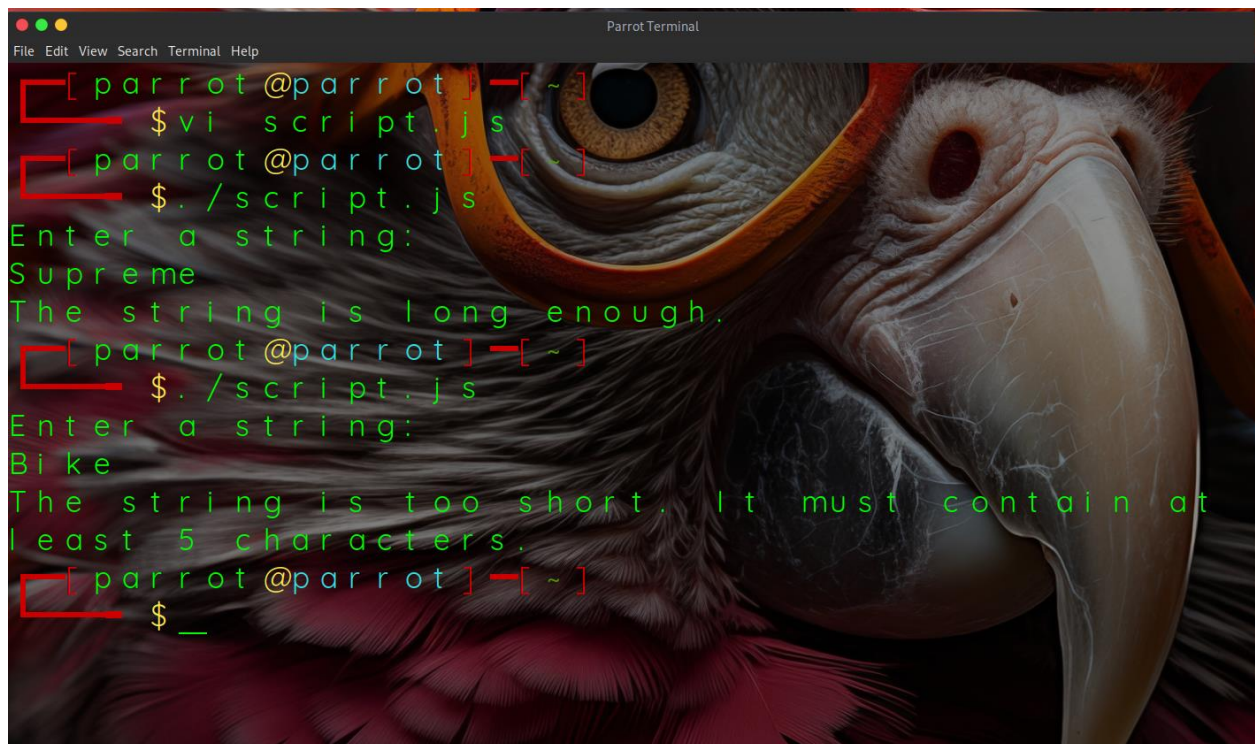


1. Write a shell script that accepts a string from the terminal and echo a suitable message if it doesn't have at least 5 characters including the other symbols.

```
#!/bin/bash
echo "Enter a string:"
read user_input
input_length=${#user_input}
if [ $input_length -lt 5 ]; then
    echo "The string is too short. It must contain at least 5 characters."
else
    echo "The string is long enough."
fi
```



2. Write a shell script to echo the string length of the given string as argument.

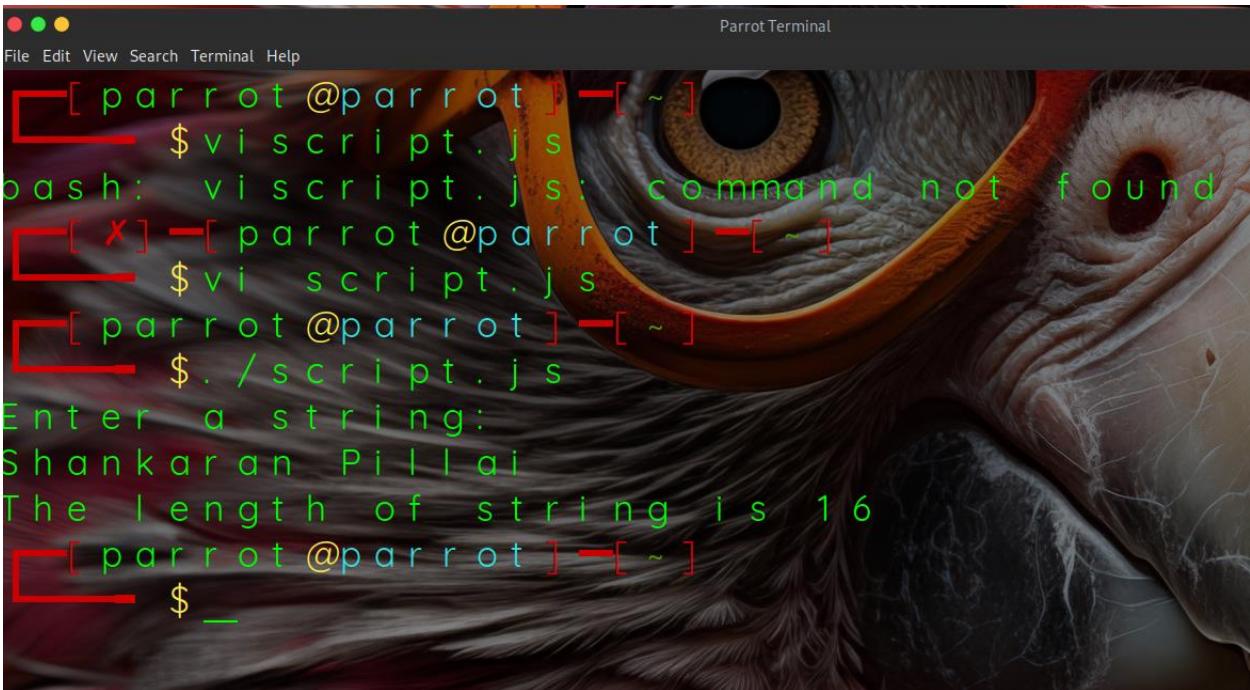
```
#!/bin/bash

echo "Enter a string:"

read user_input

input_length=${#user_input}

echo "The length of string is $input_length"
```



```
File Edit View Search Terminal Help

[ parrot@parrot ] ~
$ vi script.js
bash: vi script.js: command not found
[ X ] [ parrot@parrot ] ~
$ vi script.js
[ parrot@parrot ] ~
$ ./script.js
Enter a string:
Shankaran Pillai
The length of string is 16
[ parrot@parrot ] ~
$ _
```

3. Write a shell script that accepts two directory names as arguments and deletes those files in the first directory which are similarly named in the second directly. Note: Contents should also match inside the files.

```
#!/bin/bash

dir1="$1"

dir2="$2"

# Loop through files in directory1
for file1 in "$dir1"/*; do
    if [ -f "$file1" ]; then
```

```

# Extract filename from path

filename=$(basename "$file1")

# Check if a corresponding file exists in directory2 with the same name

file2="$dir2/$filename"

if [ -f "$file2" ]; then

    # Compare contents of files

    if cmp -s "$file1" "$file2"; then

        # Delete file1 if contents match file2

        echo "Deleting $file1 (similar content found in $file2)"

        rm "$file1"

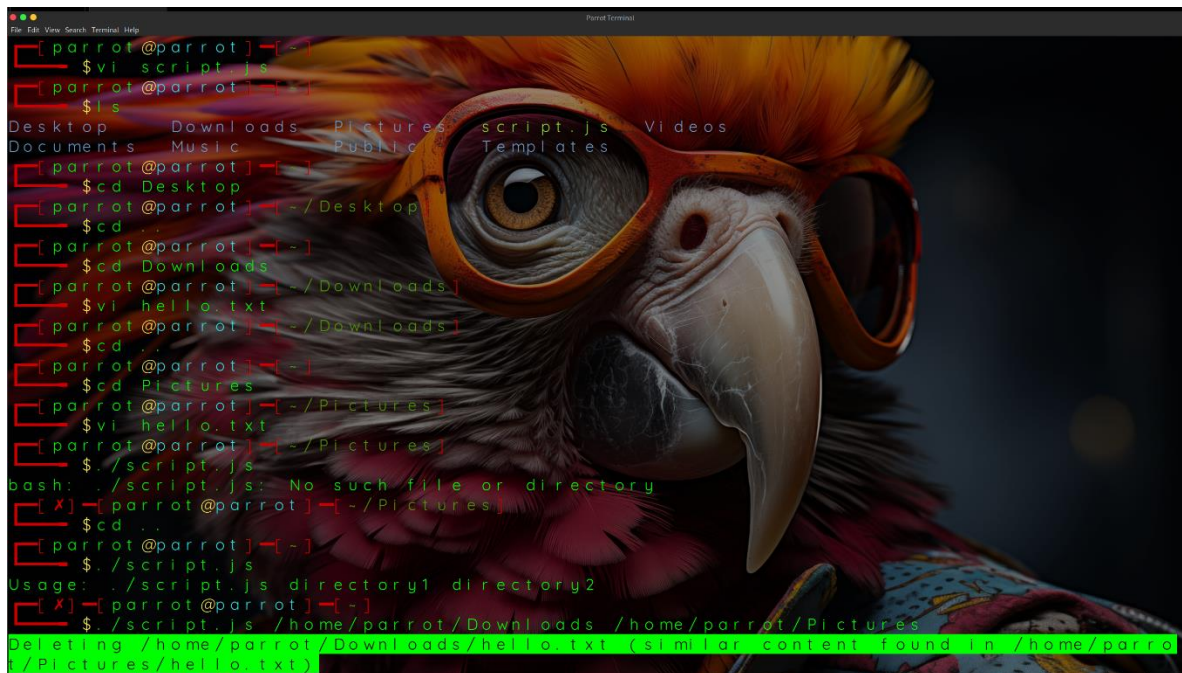
    fi

fi

done

echo "Operation complete."

```



```

[parrot@parrot] ~$ vi script.js
[parrot@parrot] ~$ ls
Desktop  Downloads  Pictures  script.js  Videos
Documents Music      Public   Templates
[parrot@parrot] ~$ cd Desktop
[parrot@parrot] ~/Desktop$ cd ..
[parrot@parrot] ~$ cd Downloads
[parrot@parrot] ~/Downloads$ vi hello.txt
[parrot@parrot] ~/Downloads$ cd ..
[parrot@parrot] ~$ cd Pictures
[parrot@parrot] ~/Pictures$ vi hello.txt
[parrot@parrot] ~/Pictures$ ./script.js
bash: ./script.js: No such file or directory
[X] [parrot@parrot] ~/Pictures$ cd ..
[parrot@parrot] ~$ ./script.js
Usage: ./script.js directory1 directory2
[X] [parrot@parrot] ~$ ./script.js /home/parrot/Downloads /home/parrot/Pictures
Deleting /home/parrot/Downloads/hello.txt (similar content found in /home/parrot/Pictures/hello.txt)

```

4. Write a shell script to display the processes running on the system for every 30 seconds, but only for 3 times.


```
#!/bin/bash

count=0

while [ $count -lt 3 ]; do    # for 3 times only

    echo "Processes running on the system:"

    ps aux

    count=$((count + 1))      # COUNT EVERY TIME U DO

    if [ $count -lt 3 ]; then

        echo "Waiting for 30 seconds before next check..."

        sleep 30              # Wait for 30 seconds.

    fi

done
```

```
File Edit View Search Terminal Help
parrot 1823 0.0 0.2 390580 12800 ? Ssl 09:29 0:00 /usr/libexec
parrot 1841 0.0 0.1 535156 7424 ? Ssl 09:29 0:00 /usr/libexec
parrot 1857 0.0 0.1 236752 6784 ? Ssl 09:29 0:00 /usr/libexec
root 1873 0.0 0.0 2480 1664 ? Ss 09:29 0:00 fusermount3
parrot 1887 0.0 0.3 335900 19176 ? Ssl 09:29 0:00 /usr/libexec
parrot 1942 0.0 0.8 418312 34104 ? Sl 09:29 0:00 /usr/lib/mat
parrot 1944 0.0 0.6 491764 34100 ? Sl 09:29 0:00 /usr/lib/mat
parrot 1946 0.0 0.8 399724 41664 ? Sl 09:29 0:00 /usr/lib/mat
parrot 1948 0.6 0.5 342520 29952 ? S 09:29 0:14 /usr/lib/mat
parrot 2024 0.0 0.1 48536 7680 ? Ss 09:29 0:00 /usr/libexec
parrot 2066 0.4 0.9 555540 46184 ? S 09:31 0:10 mate-termi na
parrot 2094 0.0 0.0 8256 4864 pts/0 Ss 09:31 0:00 bash
root 2178 0.0 0.0 0 0 ? I 09:34 0:00 [kworker/0:1
root 2344 0.0 0.0 0 0 ? I 09:39 0:00 [kworker/u51
root 2372 0.0 0.0 0 0 ? I 09:44 0:00 [kworker/u51
root 2373 0.0 0.0 0 0 ? I 09:44 0:00 [kworker/u51
root 2464 0.0 0.0 0 0 ? I 09:50 0:00 [kworker/2:2
root 2473 0.0 0.0 0 0 ? I 09:54 0:00 [kworker/1:2
root 2477 0.0 0.0 0 0 ? I 09:55 0:00 [kworker/2:0
root 2491 0.0 0.0 0 0 ? I 09:57 0:00 [kworker/u51
root 2505 0.0 0.0 0 0 ? I 09:59 0:00 [kworker/1:0
root 2506 0.0 0.0 0 0 ? I 10:00 0:00 [kworker/2:1
root 2507 0.0 0.0 0 0 ? I 10:00 0:00 [kworker/0:0
root 2510 0.0 0.0 0 0 ? I 10:00 0:00 [kworker/3:1
root 2511 0.0 0.0 0 0 ? I 10:02 0:00 [kworker/u51
root 2552 0.0 0.0 0 0 ? I 10:05 0:00 [kworker/u51
root 2553 0.0 0.0 0 0 ? I 10:05 0:00 [kworker/1:1
root 2554 0.0 0.0 0 0 ? I 10:05 0:00 [kworker/0:2
parrot 2556 0.0 0.0 6936 3328 pts/0 S+ 10:08 0:00 /bin/bash . /
parrot 2559 100 0.0 11272 4736 pts/0 R+ 10:08 0:00 ps aux
Waiting for 30 seconds before next check...
```

5. Write a shell script that displays the last modification time of any file.

```
#!/bin/bash

if [ -z "$1" ]; then

    echo "Usage: $0 <filename>"
```

```

exit 1

fi

filename="$1"

if [ ! -e "$filename" ]; then

    echo "File not found: $filename"

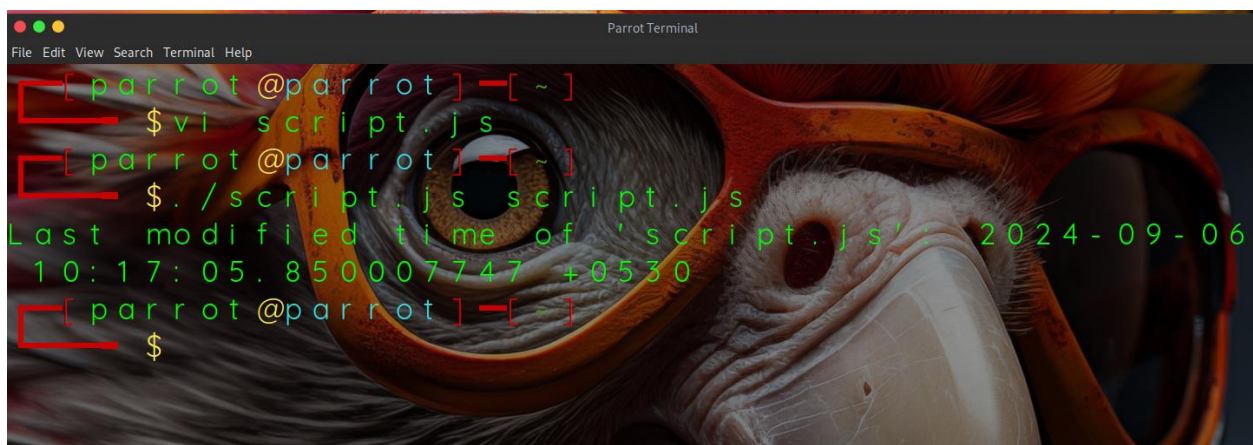
    exit 1

fi

last_modified=$(stat -c %y "$filename")

echo "Last modified time of '$filename': $last_modified"

```



6. Write a shell script to check the spellings of any text document given as an argument.

```

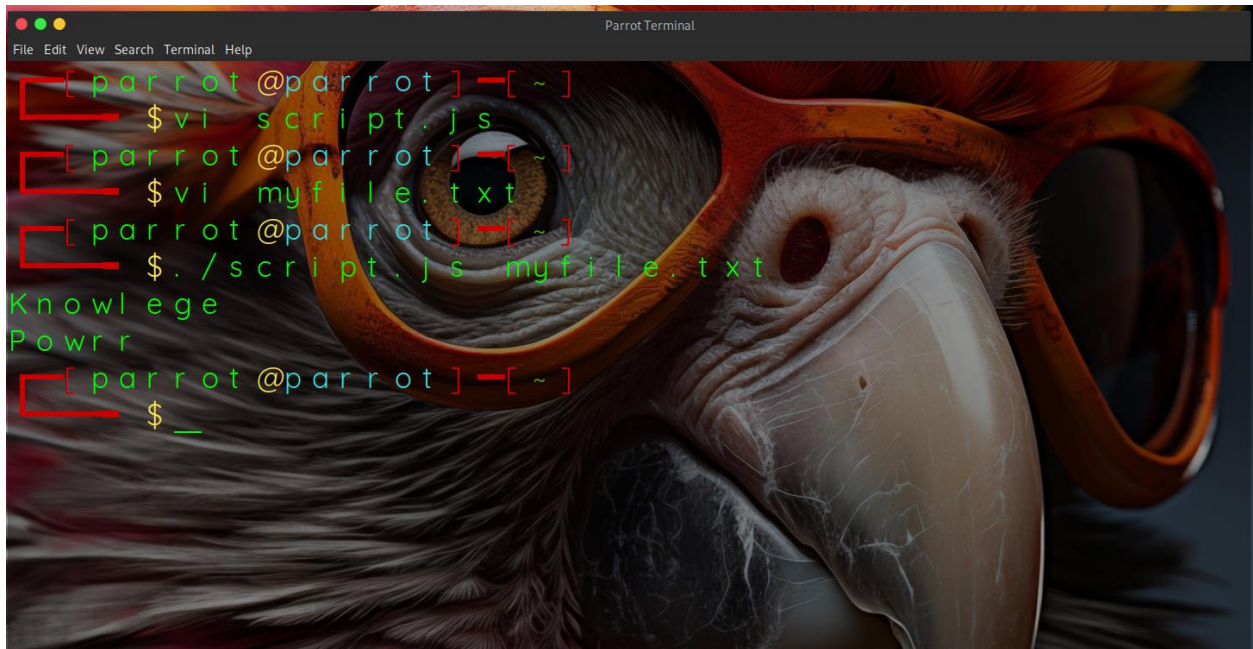
#!/bin/bash

if [ -z "$1" ]; then                                # if text file name not passed
    echo "Usage: $0 <text_file>"
    exit 1
fi

text_file="$1"                                       # if file name given but file not present
if [ ! -f "$text_file" ]; then
    echo "File not found: $text_file"
    exit 1
fi

cat "$text_file" | aspell list | sort -u             # using inbuilt lib. Do spell check.

```



7. Write a shell script to encrypt any text file.

```
#!/bin/bash

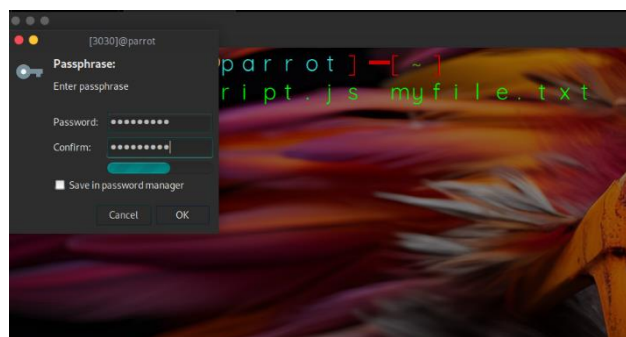
if [ -z "$1" ]; then          # if file not provided
    echo "Usage: $0 <text_file>"
    exit 1
fi

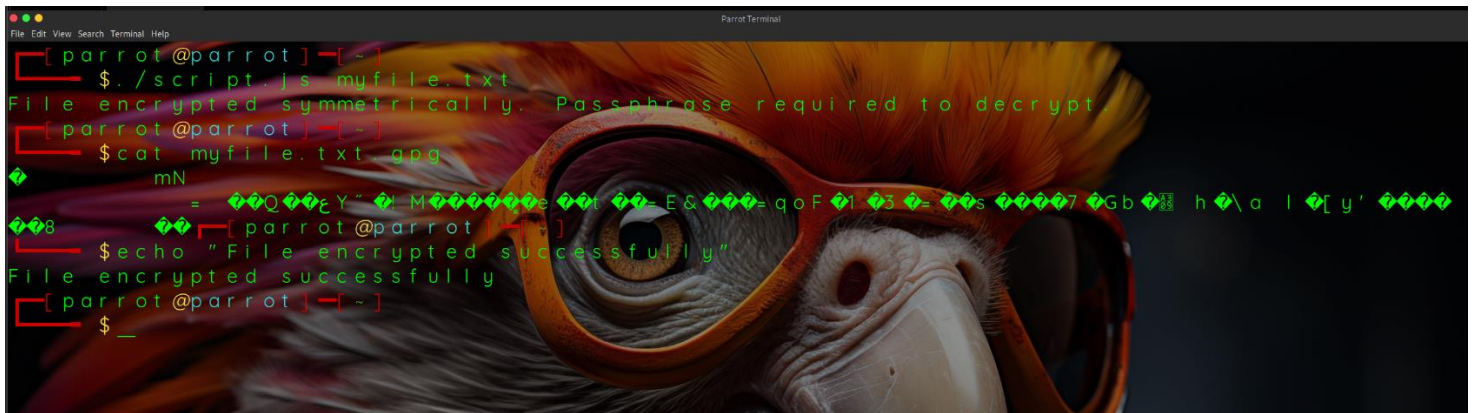
text_file="$1"

if [ ! -f "$text_file" ]; then # if file provided but not exist
    echo "File not found: $text_file"
    exit 1
fi

gpg --symmetric --output "$text_file.gpg" "$text_file" # use gpg (GNU privacy guard for enc)

echo "File encrypted symmetrically. Passphrase required to decrypt."
```





```

[parrot@parrot] ~$ ./script.js myfile.txt
File encrypted symmetrically. Passphrase required to decrypt.
[parrot@parrot] ~$ cat myfile.txt.gpg
mN
= QY" M-E&=qoF13s7Gb h\o l[y'
8 [parrot@parrot] ~$ echo "File encrypted successfully"
File encrypted successfully
[parrot@parrot] ~$ _

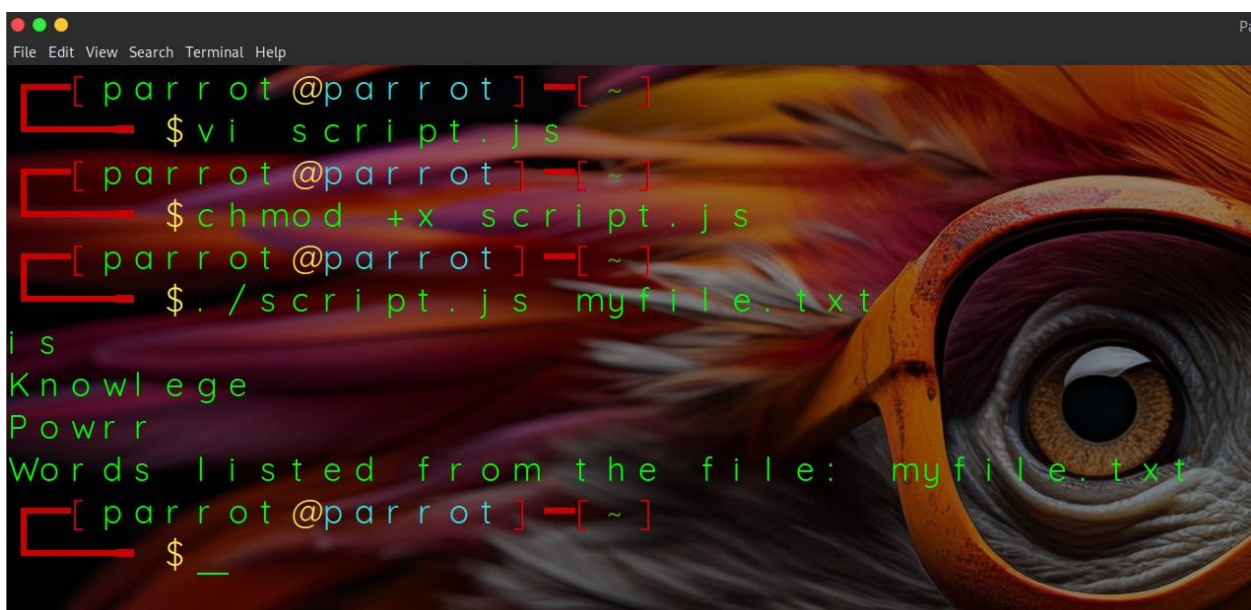
```

8. Combine the above commands in a shell script so that you have a small program for extracting a wordlist.

```

#!/bin/bash
if [ -z "$1" ]; then
    echo "Usage: $0 <text_file>"
    exit 1
fi
text_file="$1"
if [ ! -f "$text_file" ]; then
    echo "File not found: $text_file"
    exit 1
fi
cat "$text_file" | tr -sc '[:alpha:]' '\n' | sort -u
echo "Words listed from the file: $text_file"

```



```

[parrot@parrot] ~$ vi script.js
[parrot@parrot] ~$ chmod +x script.js
[parrot@parrot] ~$ ./script.js myfile.txt
is
Knowl ege
Powrr
Words listed from the file: myfile.txt
[parrot@parrot] ~$ _

```

9. Write a shell script which reads the contents in a text file and removes all the blank spaces in them and redirects the output to a file.

```
#!/bin/bash

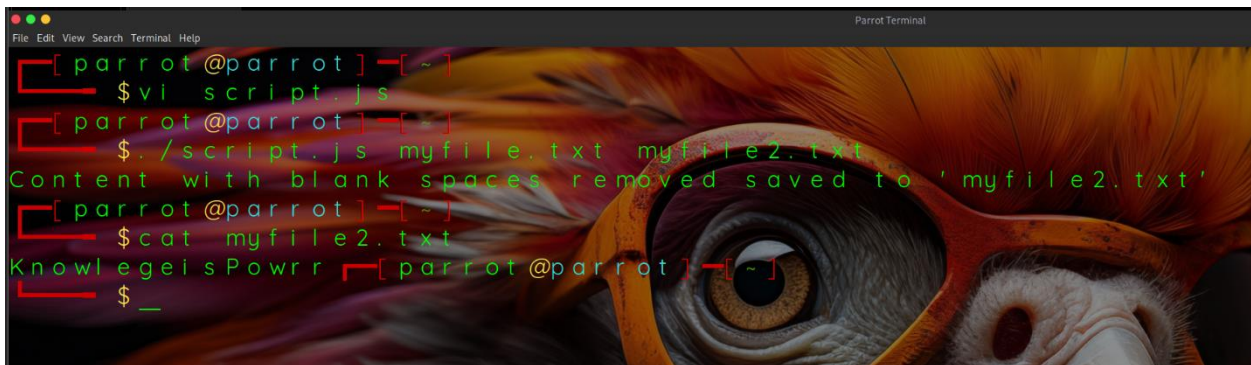
if [ -z "$1" ]; then
    echo "Usage: $0 <input_file> <output_file>"
    exit 1
fi

input_file="$1"
output_file="$2"

if [ ! -f "$input_file" ]; then
    echo "File not found: $input_file"
    exit 1
fi

tr -d '[:space:]' < "$input_file" > "$output_file"

echo "Content with blank spaces removed saved to '$output_file'"
```



10. Write a shell script that changes the name of the files passed as arguments to lowercase.

```
#!/bin/bash

if [ $# -eq 0 ]; then

    echo "Usage: $0 <file1> [<file2> ...]"

    exit 1

fi

for filename in "$@"; do

    if [ ! -e "$filename" ]; then

        echo "File not found: $filename"

    else

        lowercase=$(echo "$filename" | tr '[:upper:]' '[:lower:]')

        if [ "$filename" != "$lowercase" ]; then

            mv "$filename" "$lowercase"

            echo "Renamed '$filename' to '$lowercase'"

        else

            echo "File '$filename' is already lowercase"

        fi

    fi

done
```

