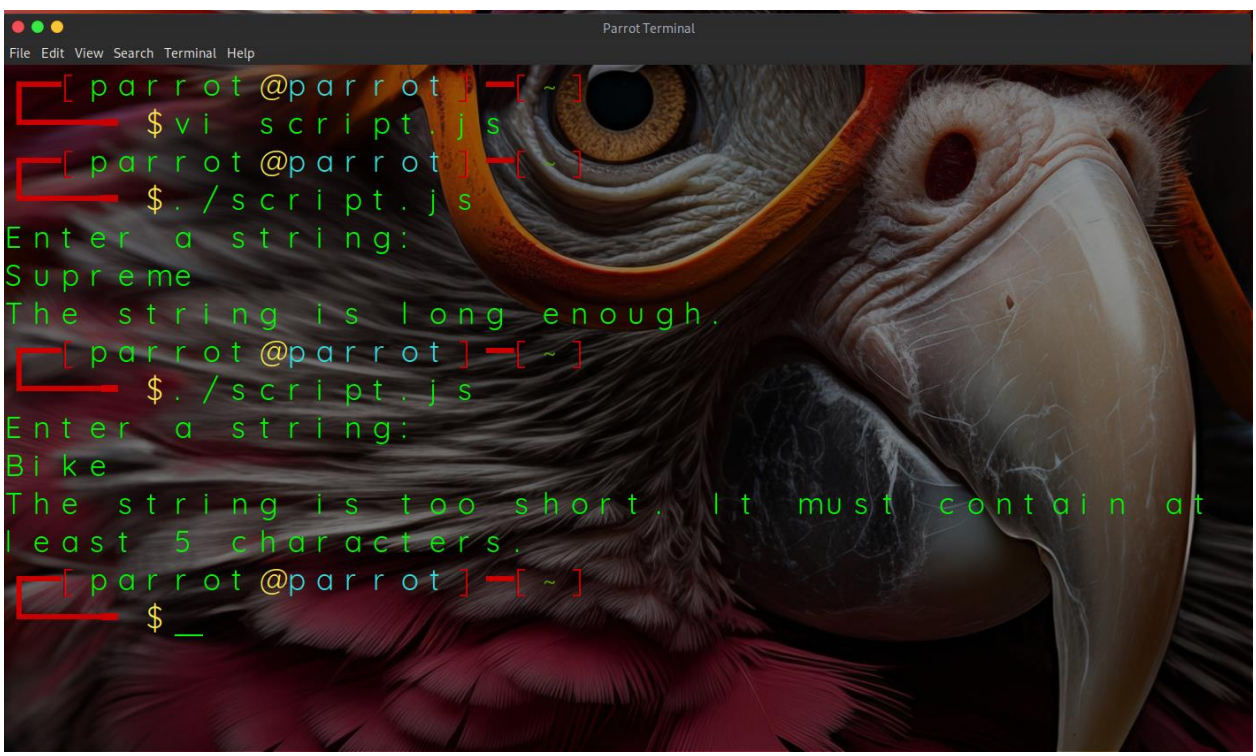
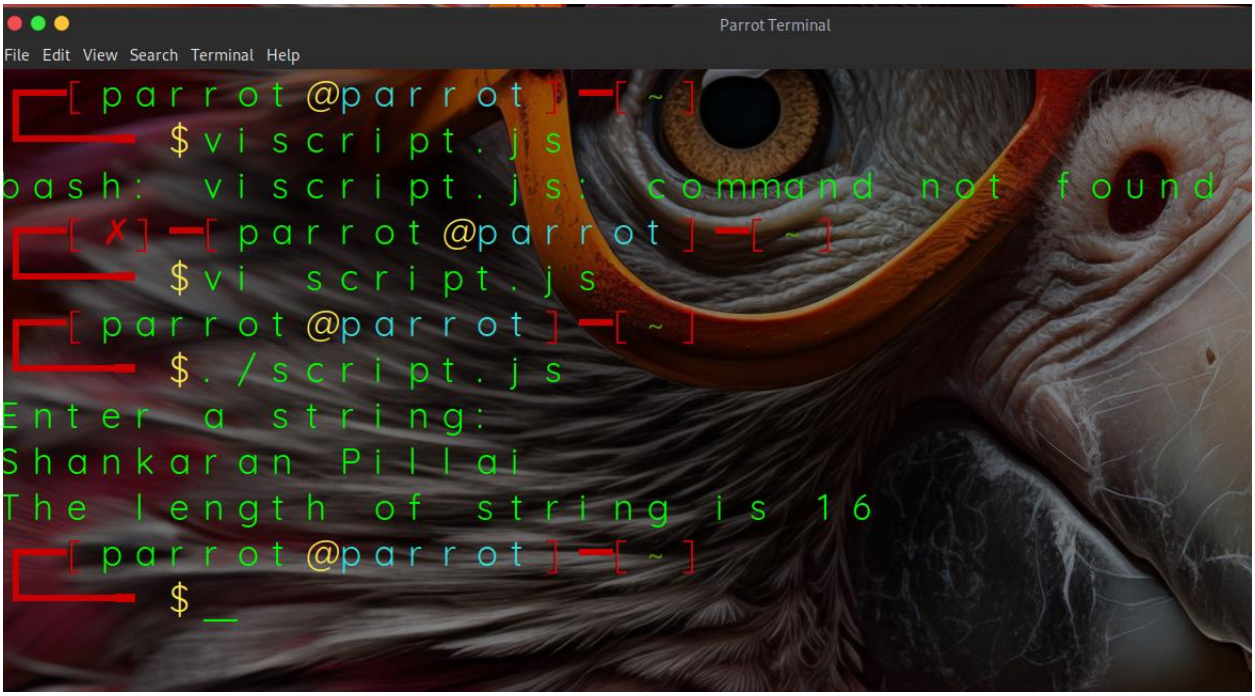
1. Write a shell script that accepts a string from the terminal and echo a suitable message if it doesn't have at least 5 characters including the other symbols.

```
#!/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.

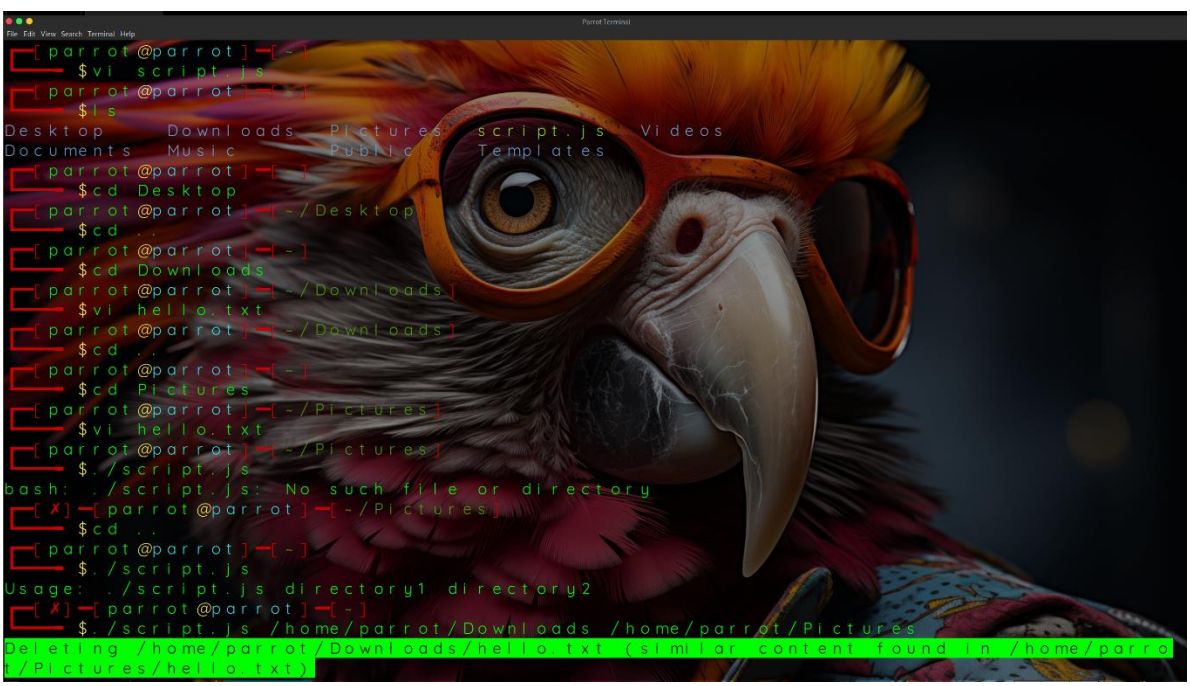
```
#!/bin/bash
echo "Enter a string:"
read user_input
input_length=${#user_input}
echo “The length of string is $input_length”
```



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.

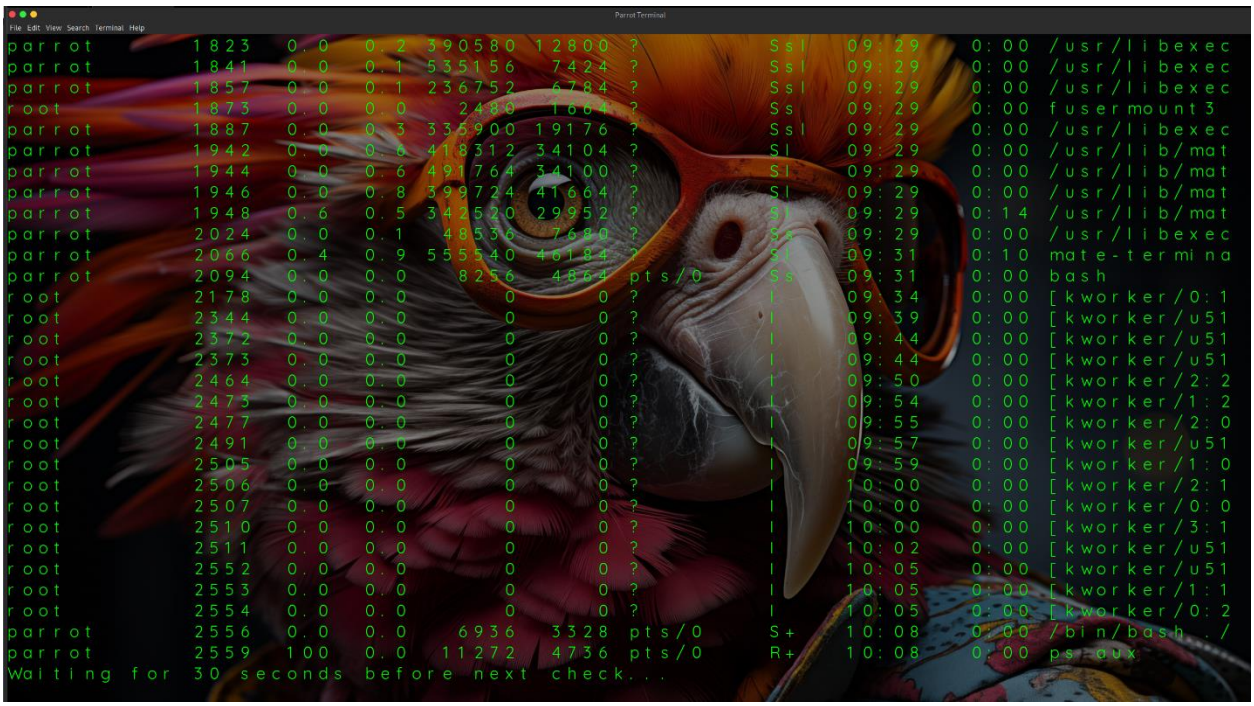
```
#!/bin/bash
dir1="$1"
dir2="$2"
# Loop through files in directory1
for file1 in "$dir1"/*; do
   if [ -f "$file1" ]; then
```

```
        # 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
      fi
    done
    echo "Operation complete."
```



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

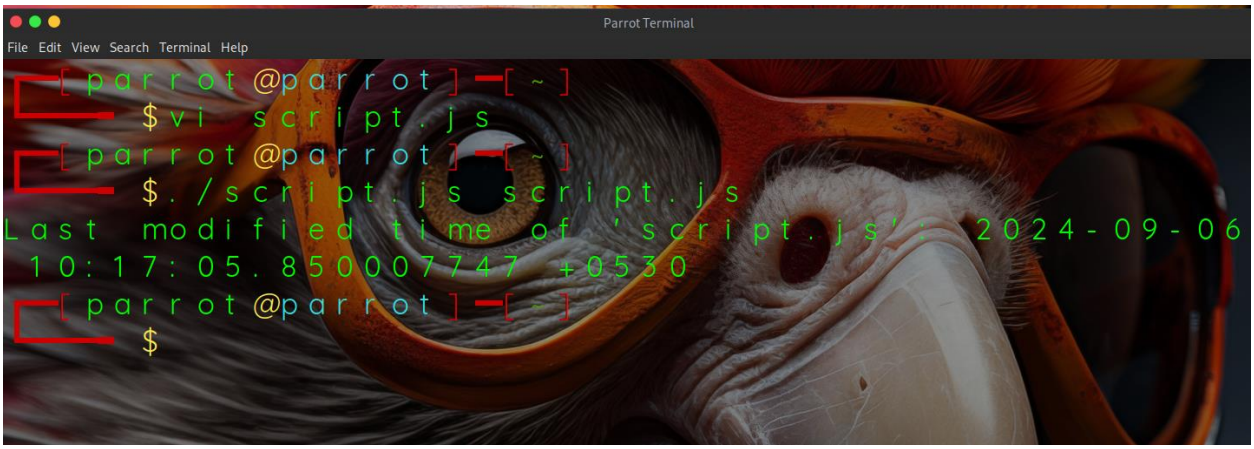
```
#!/bin/bash
count=0
while [ $count -lt 3 ]; do     # for 3 times only
        echo "Processes running on the system:"
        ps aux
        count=$((count + 1))          # COUNT EVERY TIME U DO
        if [ $count -lt 3 ]; then
                echo "Waiting for 30 seconds before next check..."
                sleep 30              # Wait for 30 seconds.
        fi
done
```



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

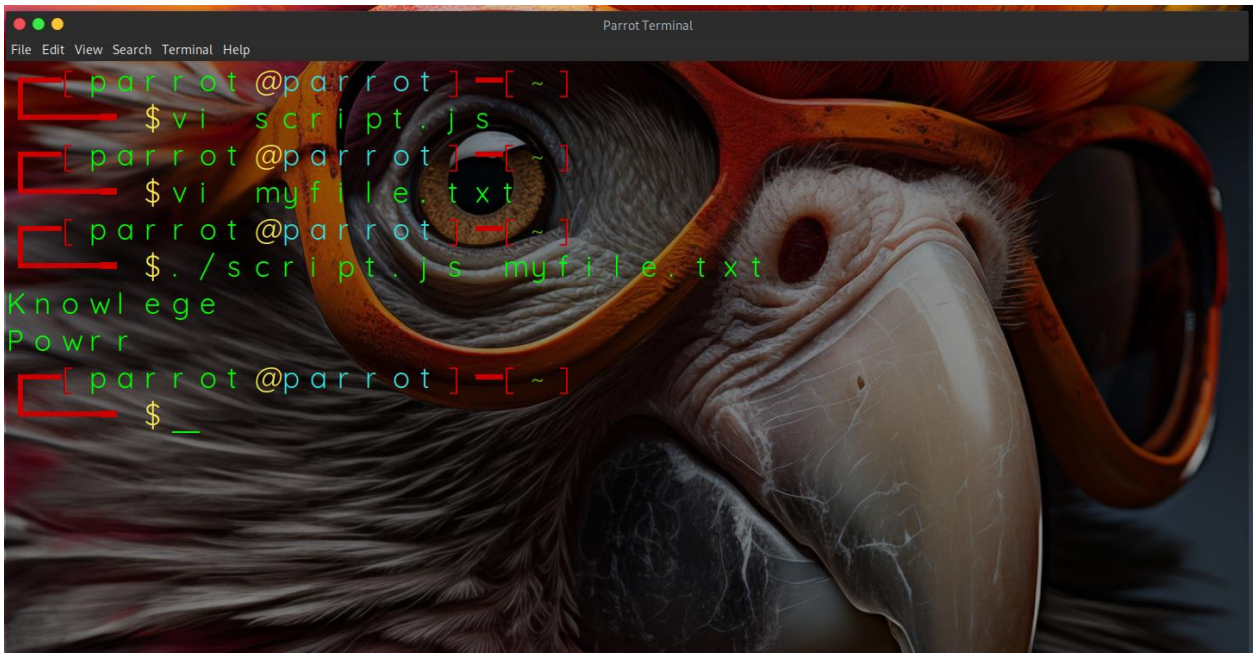
```
#!/bin/bash
if [ -z "$1" ]; then
   echo "Usage: $0 <filename>"
```

```
    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"
```



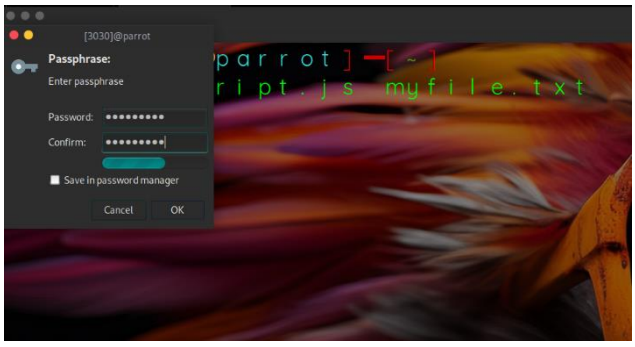
6. Write a shell script to check the spellings of any text document given as an argument.

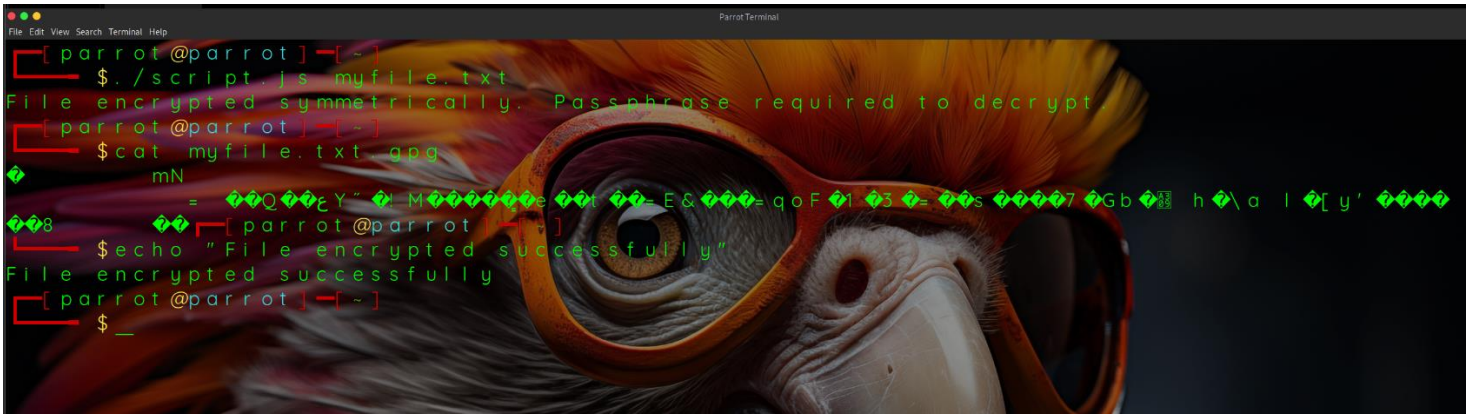
```
#!/bin/bash

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.

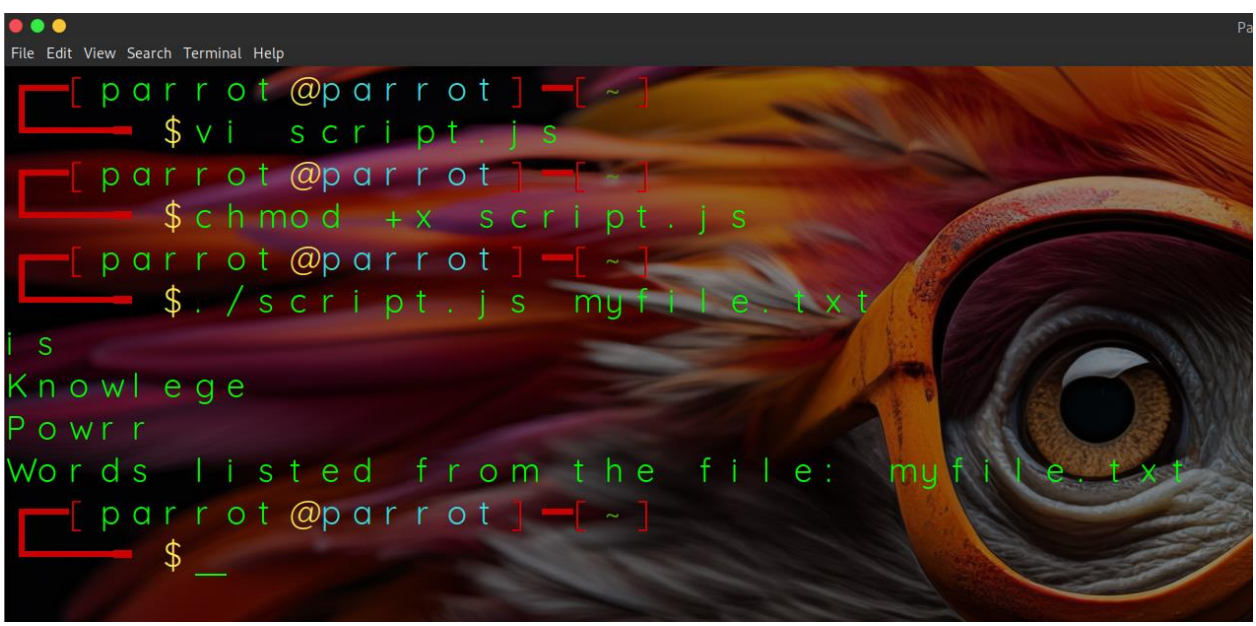
```
#!/bin/bash
if [ -z "$1" ]; then                    # if file not provided
   echo "Usage: $0 <text_file>"
   exit 1
fi
text_file="$1"
if [ ! -f "$text_file" ]; then          # if file provided but do not exist
   echo "File not found: $text_file"
   exit 1
fi
gpg --symmetric --output "$text_file.gpg" "$text_file"          # use gpg (GNU privacy guard for enc)
echo "File encrypted symmetrically. Passphrase required to decrypt."
```





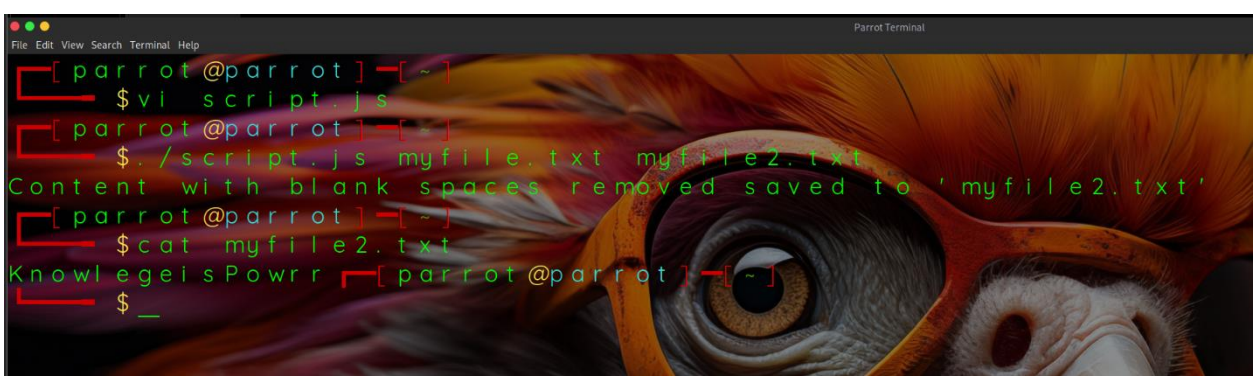
8. Combine the above commands in a shell script so that you have a small program for extracting a wordlist.

```
#!/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"
```



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
```