MODULE 5

Create a Bash script 'file_analyzer.sh', to demonstrate the following concepts:

1. Recursive functions- Write a recursive function to search for files in a directory and its subdirectories containing a specific keyword.

2. Redirection and error handling- Log errors (e.g., invalid arguments, missing files) to 'errors.log' and display them in the terminal.

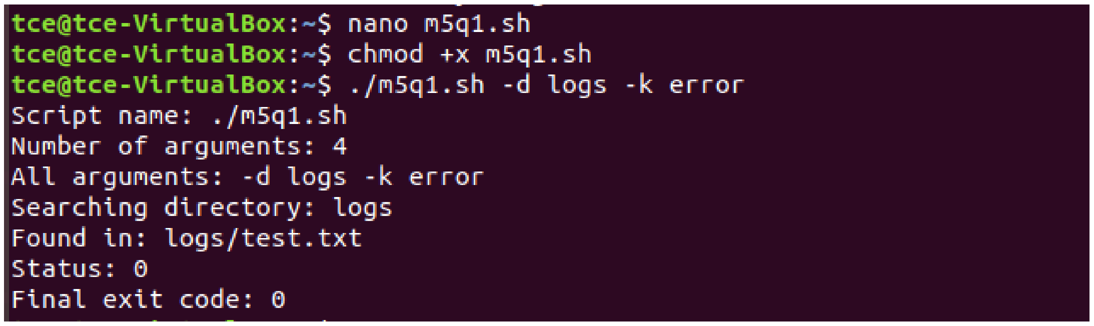
3. Here document and here string- Use a here document to display a help menu when the '--help' option is passed.- Search for a keyword in a specified file using a here string

4. Special parameters- Use parameters like '$0', '$#', '$?' and '$@' to provide meaningful feedback.

5. Regular expressions- Validate inputs with regular expressions (Check if the file exists and the keyword is not empty and valid)

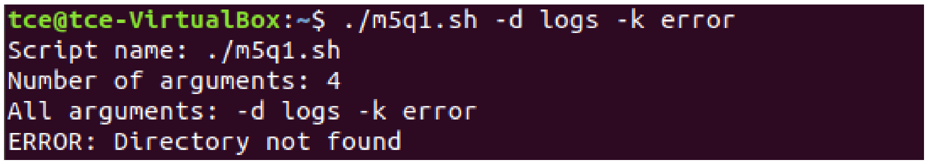
6. Command-line arguments using getopts- Use 'getopts' to handle: '-d ': Directory to search. '-k ': Keyword to search. '-f ': File to search directly. '--help': Display the help menu.

Example usage: # Recursively search a directory for a keyword ./file_analyzer.sh -d logs -k error # Search for a keyword in a file ./file_analyzer.sh -f script.sh -k TODO # Display the help menu ./file_analyzer.sh –help

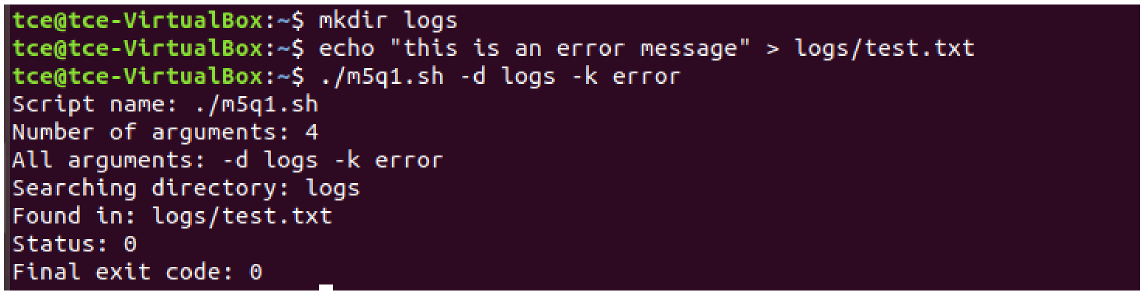


```
tce@tce-VirtualBox:~$ nano m5q1.sh
tce@tce-VirtualBox:~$ chmod +x m5q1.sh
tce@tce-VirtualBox:~$ ./m5q1.sh -d logs -k error
Script name: ./m5q1.sh
Number of arguments: 4
All arguments: -d logs -k error
Searching directory: logs
Found in: logs/test.txt
Status: 0
Final exit code: 0
```

Error handling working:

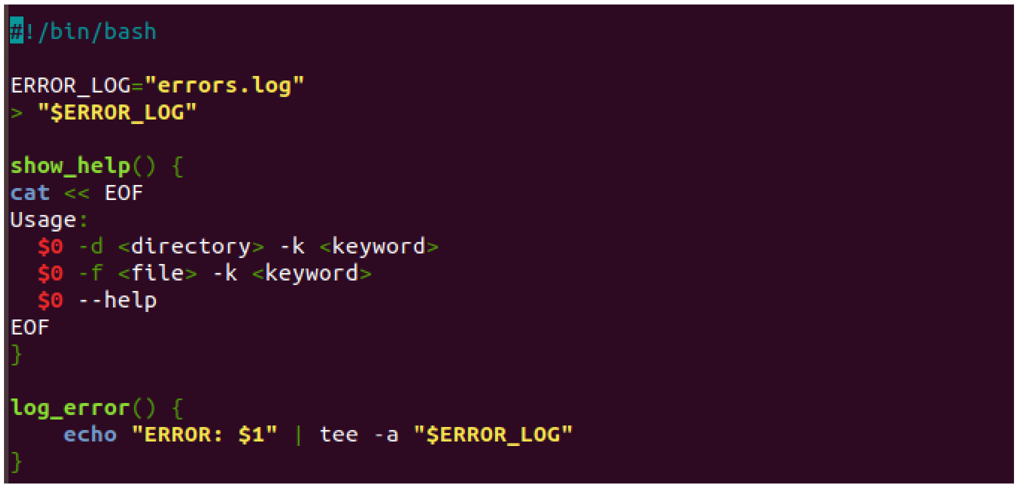


```
tce@tce-VirtualBox:~$ ./m5q1.sh -d logs -k error
Script name: ./m5q1.sh
Number of arguments: 4
All arguments: -d logs -k error
ERROR: Directory not found
```

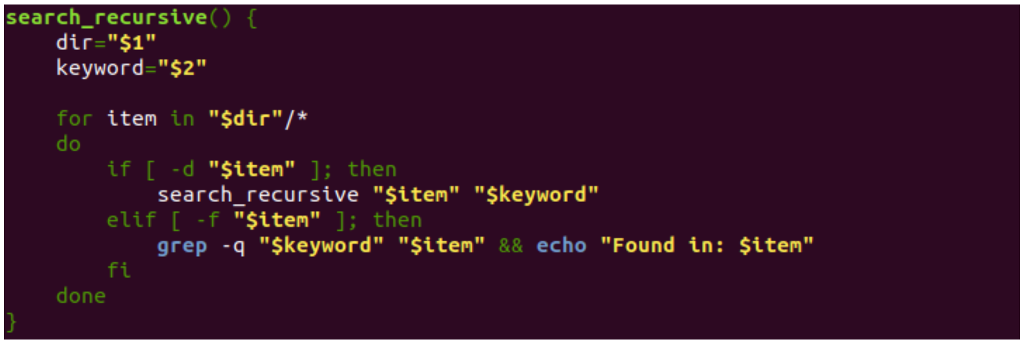


```
tce@tce-VirtualBox:~$ mkdir logs
tce@tce-VirtualBox:~$ echo "this is an error message" > logs/test.txt
tce@tce-VirtualBox:~$ ./m5q1.sh -d logs -k error
Script name: ./m5q1.sh
Number of arguments: 4
All arguments: -d logs -k error
Searching directory: logs
Found in: logs/test.txt
Status: 0
Final exit code: 0
```

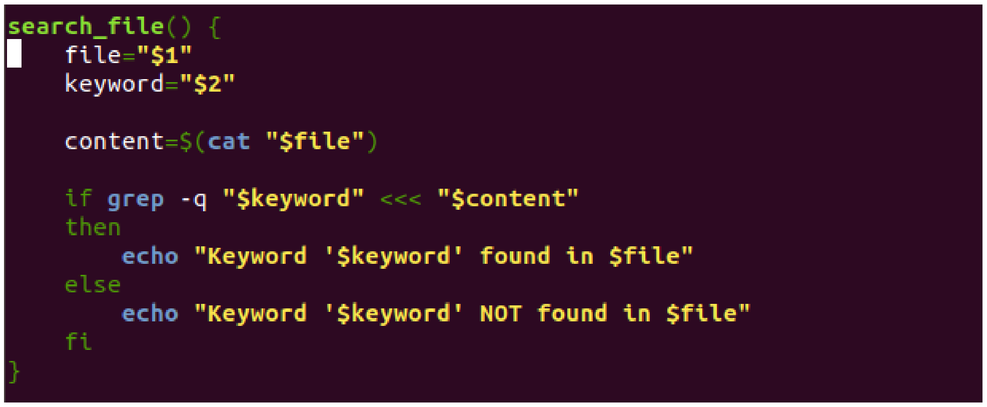
Scripts:



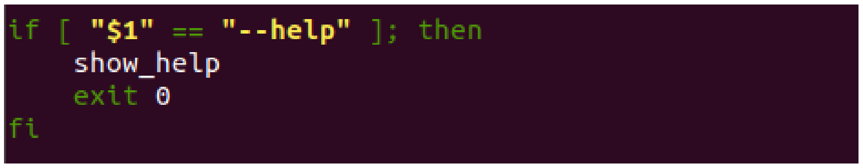
```
#!/bin/bash

ERROR_LOG="errors.log"
> "$ERROR_LOG"

show_help() {
cat << EOF
Usage:
  $0 -d <directory> -k <keyword>
  $0 -f <file> -k <keyword>
  $0 --help
EOF
}

log_error() {
    echo "ERROR: $1" | tee -a "$ERROR_LOG"
}
```



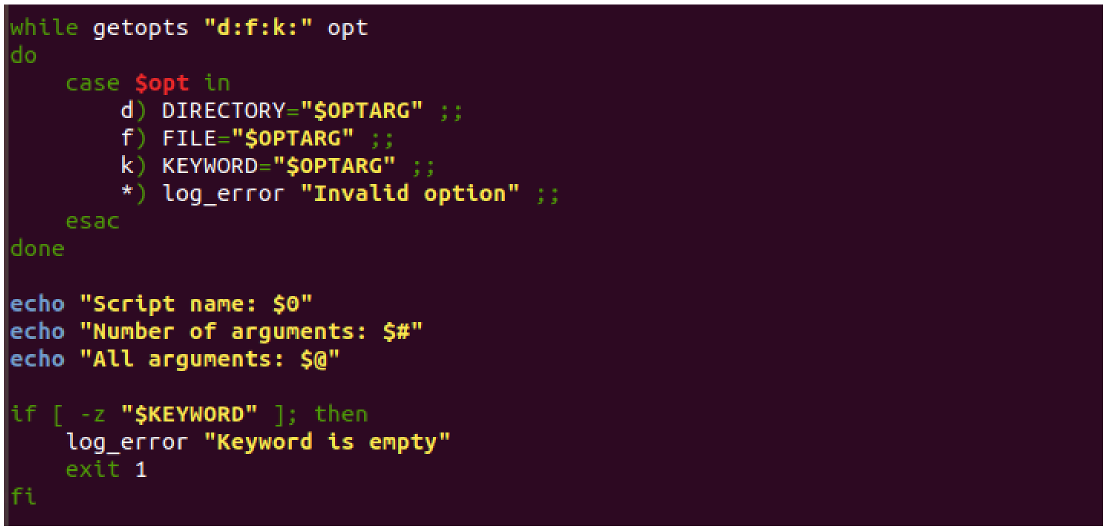
```
search_recursive() {
    dir="$1"
    keyword="$2"

    for item in "$dir"/*
    do
        if [ -d "$item" ]; then
            search_recursive "$item" "$keyword"
        elif [ -f "$item" ]; then
            grep -q "$keyword" "$item" && echo "Found in: $item"
        fi
    done
}
```



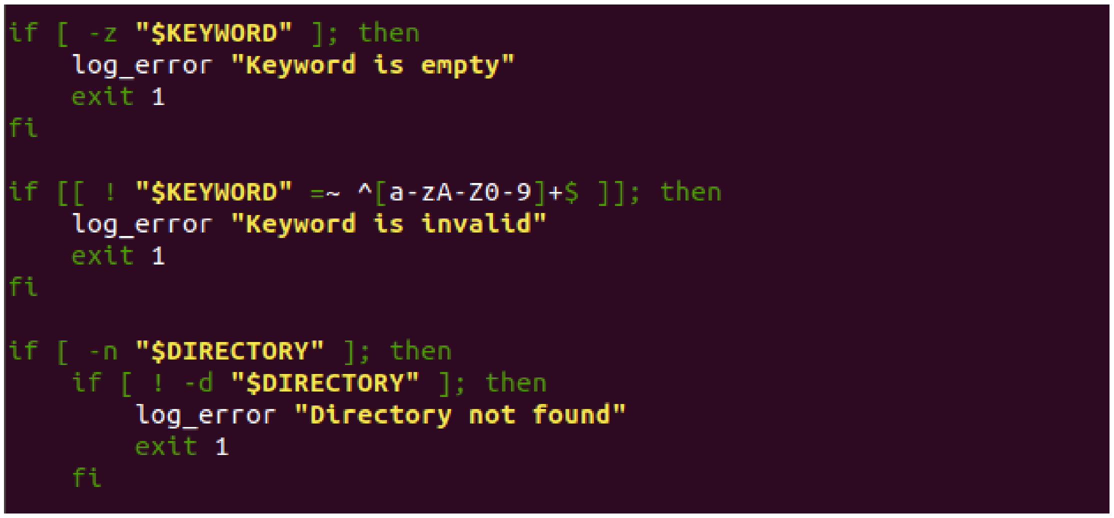
```
search_file() {
    file="$1"
    keyword="$2"

    content=$(cat "$file")

    if grep -q "$keyword" <<< "$content"
    then
        echo "Keyword '$keyword' found in $file"
    else
        echo "Keyword '$keyword' NOT found in $file"
    fi
}
```



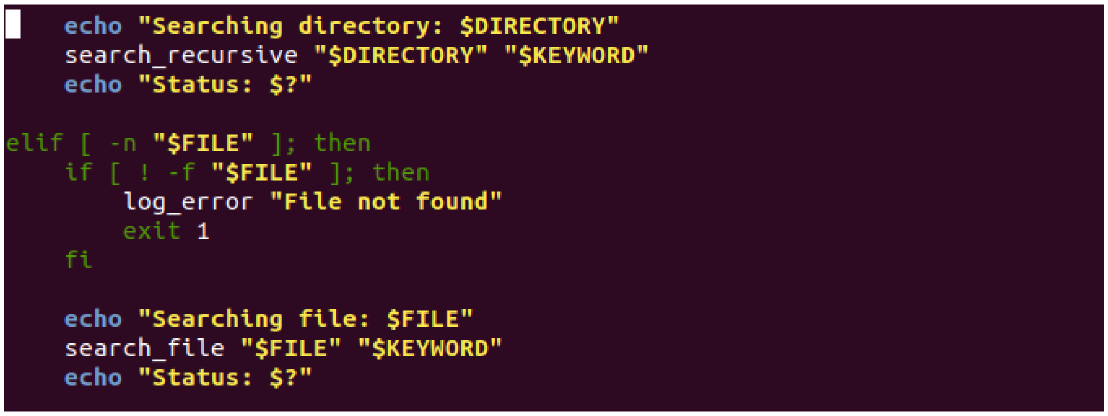
```
if [ "$1" == "--help" ]; then
    show_help
    exit 0
fi
```



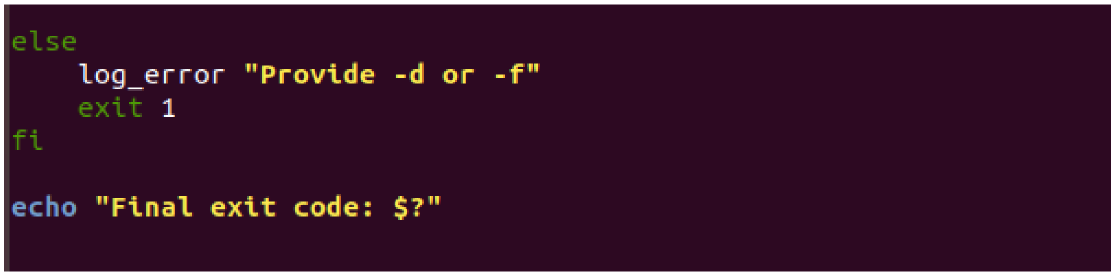
```
while getopts "d:f:k:" opt
do
    case $opt in
        d) DIRECTORY="$OPTARG" ;;
        f) FILE="$OPTARG" ;;
        k) KEYWORD="$OPTARG" ;;
        *) log_error "Invalid option" ;;
    esac
done

echo "Script name: $0"
echo "Number of arguments: $#"
echo "All arguments: $@"

if [ -z "$KEYWORD" ]; then
    log_error "Keyword is empty"
    exit 1
fi
```



```
if [ -z "$KEYWORD" ]; then
    log_error "Keyword is empty"
    exit 1
fi

if [[ ! "$KEYWORD" =~ ^[a-zA-Z0-9]+$ ]]; then
    log_error "Keyword is invalid"
    exit 1
fi

if [ -n "$DIRECTORY" ]; then
    if [ ! -d "$DIRECTORY" ]; then
        log_error "Directory not found"
        exit 1
    fi
```



```
    echo "Searching directory: $DIRECTORY"
    search_recursive "$DIRECTORY" "$KEYWORD"
    echo "Status: $?"

elif [ -n "$FILE" ]; then
    if [ ! -f "$FILE" ]; then
        log_error "File not found"
        exit 1
    fi

    echo "Searching file: $FILE"
    search_file "$FILE" "$KEYWORD"
    echo "Status: $?"
```



```
else
    log_error "Provide -d or -f"
    exit 1
fi

echo "Final exit code: $?"
```