2023/2024 Trimester 2

Ibrahim Khalil Uthman

Student ID: 2315852

MOD003218 Operating Systems

Contents

[Part 1A: Linux Shell Script 2](#_Toc164686239)

[Source Code: 2](#_Toc164686240)

[Part 2: Research Questions 19](#_Toc164686241)

[Question 1 19](#_Toc164686242)

[Question 2 21](#_Toc164686243)

[Question 3 21](#_Toc164686244)

[Question 4 23](#_Toc164686245)

[Question 5 24](#_Toc164686246)

[References 34](#_Toc164686247)

# Part 1A: Linux Shell Script

# Source Code:

#!/bin/bash

# Function to lookup IP for a domain

lookup() {

# Print the domain name

echo "Domain: $1"

# Use the host command to get the IP address of the domain

# grep is used to filter the output for lines containing "has address"

# cut is used to extract the IP address from the line

ip=$(host $1 | grep "has address" | cut -d " " -f 4)

# Check if the IP variable is empty

if [ -z "$ip" ]

then

# If it is, print an error message and return 1 to indicate an error

echo "Invalid domain name"

return 1

else

# If it's not, print the IP address

echo "IP Address:"

# Print each IP on a separate line

for address in $ip; do

echo $address

done

echo "-----------------------------"

return 0

fi

# No need to unset variables as they are local to the function

}

# Function to process file

process\_file() {

# Check if the file exists

if [ ! -f "$1" ]; then

# If the file doesn't exist, print a message and return

echo "File does not exist :( "

return 1

fi

# Print a welcome message and the name of the file being processed

echo "------------------------------------------------"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo " Welcome to IP WIZARD "

echo "------------------------------------------------"

echo "IP Address(s) found from domains in $1"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "------------------------------------------------"

# Initialize a flag to check if a valid domain was found

valid\_domain\_found=0

# Read the file line by line

while IFS= read -r line

do

# Lookup the IP for each domain in the file

if lookup $line; then

valid\_domain\_found=1

fi

done < "$1"

# If no valid domain was found in the file, print a message

if [ $valid\_domain\_found -eq 0 ]; then

echo "File doesn't contain a valid domain name or is empty :("

else

echo "Thank you for using IP WIZARD :)"

fi

# No need to unset variables as they are local to the function

}

# Function to display menu

menu() {

# Print a welcome message

echo "------------------------------------------------"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo " Welcome to IP WIZARD "

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "------------------------------------------------"

# Loop to keep the menu running until the user chooses to quit

while true; do

echo "Please select the option you want to proceed:"

echo "1) Input a domain name"

echo "2) Quit from the program"

# Read the user's choice

read -p "Enter your choice: " choice

# Process the user's choice

case $choice in

1)

# Loop to keep asking for a domain until a valid one is entered

while true; do

read -p "Enter domain name: " domain

echo "-----------------------------"

lookup $domain && break

done

;;

2)

# If the user chooses to quit, print a goodbye message and exit the script

echo "Thank you for using IP WIZARD :)"

exit 0

;;

\*)

# If the user enters an invalid choice, print an error message

echo "Invalid choice. Please enter 1 or 2."

;;

esac

done

# No need to unset variables as they are local to the function

}

# Main script

if [ $# -eq 0 ]

then

# If no arguments are provided, display the menu

menu

elif [ $# -eq 1 ]

then

# If one argument is provided, assume it's a file name and process the file

process\_file $1

else

# If more than one argument is provided, print an error message

echo "Too many parameters entered"

echo "Please enter only one parameter"

echo "Thank you :)"

fi

# No need to unset variables as they are local to the function

**PROGRAM PSEUDOCODE:**

1. Start

2. Check if there are command line arguments.

3. If no arguments, go to the menu function:

- Display menu options.

- If option 1 is selected, ask for a domain name and go to the lookup function.

- If option 2 is selected, print a goodbye message and end.

- If an invalid option is selected, print an error message and go back to displaying menu options.

4. If one argument is provided, go to the process\_file function:

- Check if the file exists.

- If it doesn't, print an error message and end.

- If it does, read the file line by line and for each line, go to the lookup function.

- After all lines have been read, check if a valid domain was found. If not, print an error message. If yes, print a thank you message.

5. If more than one argument is provided, print an error message and end.

6. End

In the lookup function:

1. Start

2. Print the domain name.

3. Use the host command to get the IP address of the domain.

4. Check if the IP variable is empty.

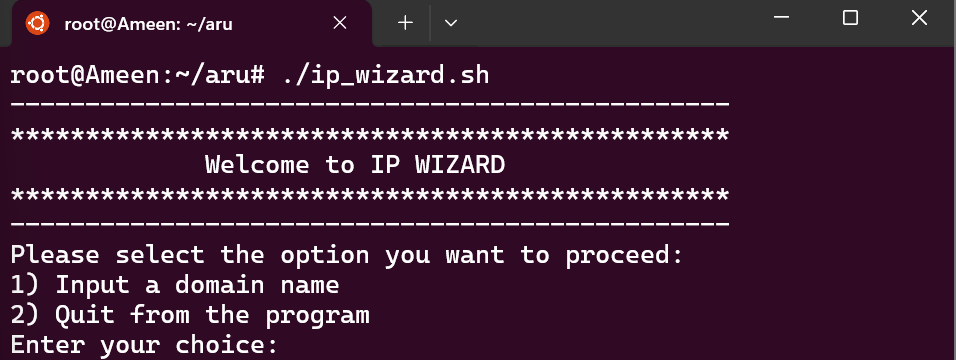
5. If it is, print an error message and return 1.

6. If it's not, print the IP address and return 0.

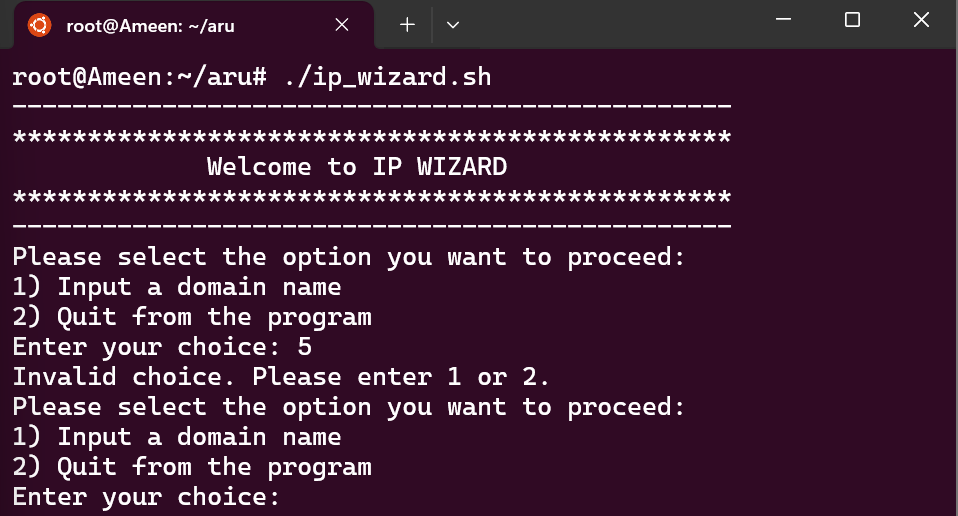
7. End

**PROGRAM SCREENSHOTS:**

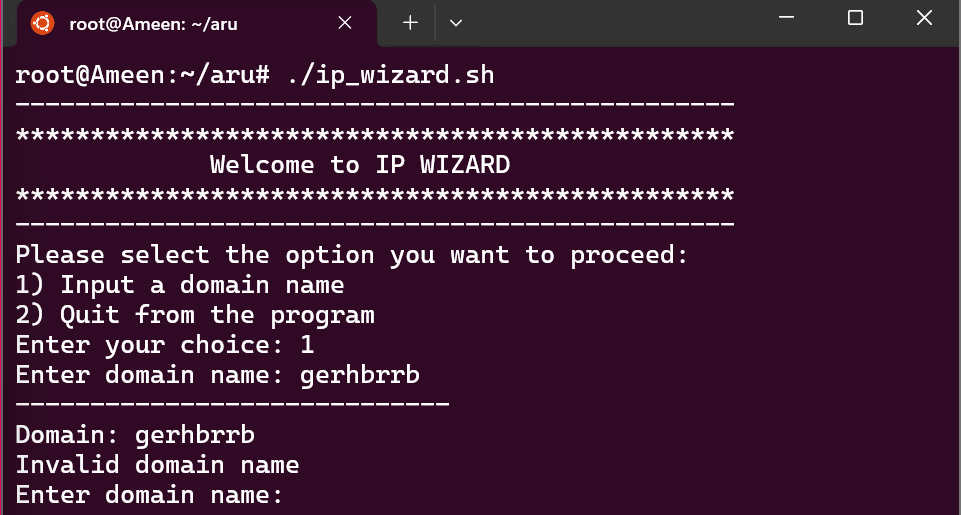
When the user adds no argument, he gets to select an option from the menu, option 1 “Input a domain name” or option 2 “Quit from the program”.



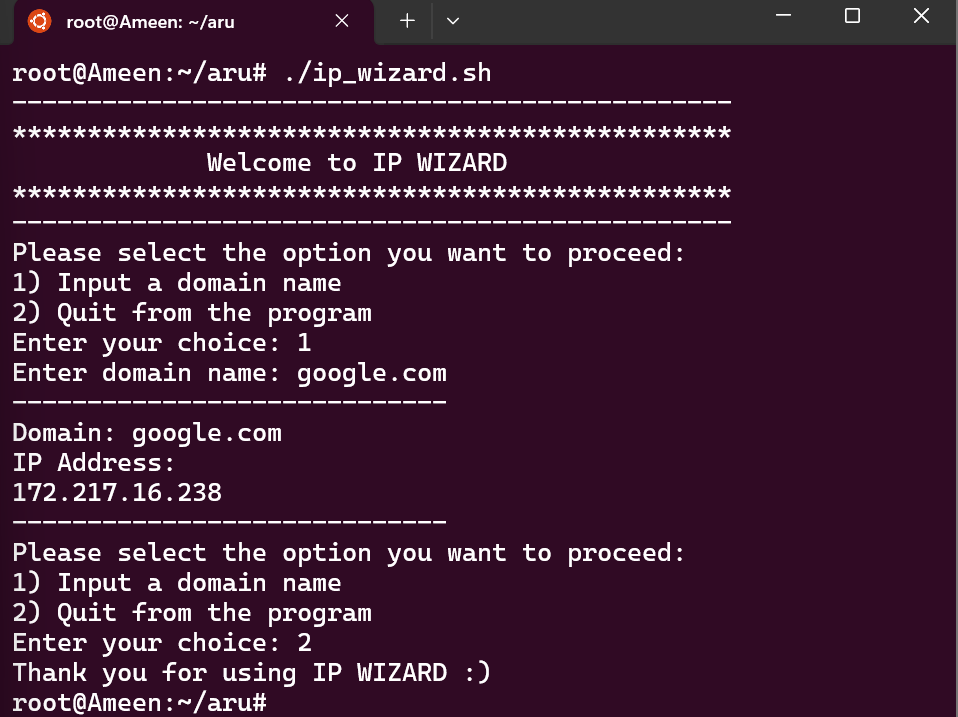
If the user enters an invalid menu choice (i.e any other input asides from “1” & “2”), the program will display an “Invalid choice. Please enter 1 or 2” prompt. This will be the response until an appropriate option is picked.



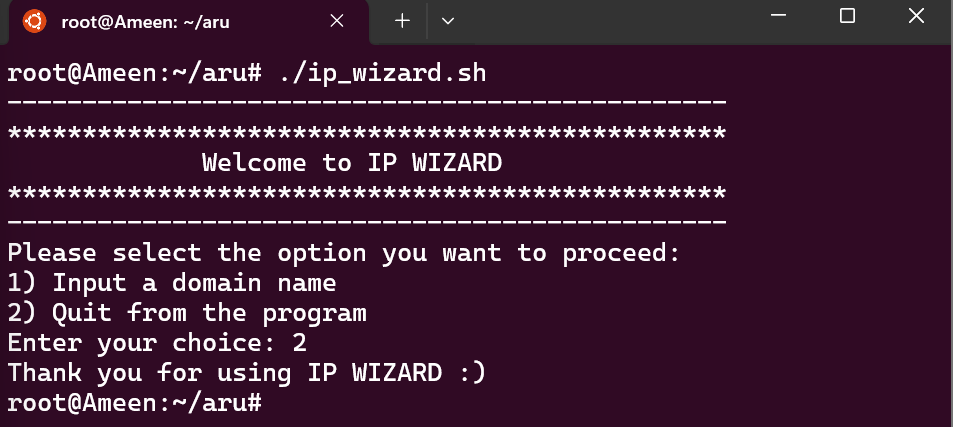
When the user selects option “1” to input domain name and inputs an invalid domain name a prompt “Invalid domain name” will appear and the user won’t be able to proceed until a valid domain name is entere.



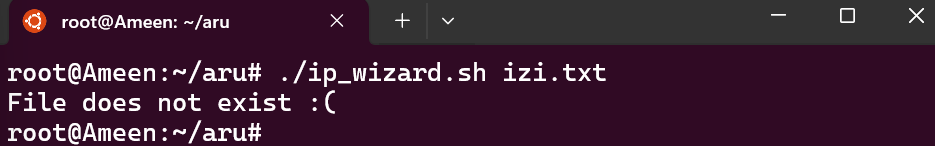
When the user enters a valid domain name the program provides the IP Address for the domain, and the user is shown the menu again to either input a domain name or quit the program.



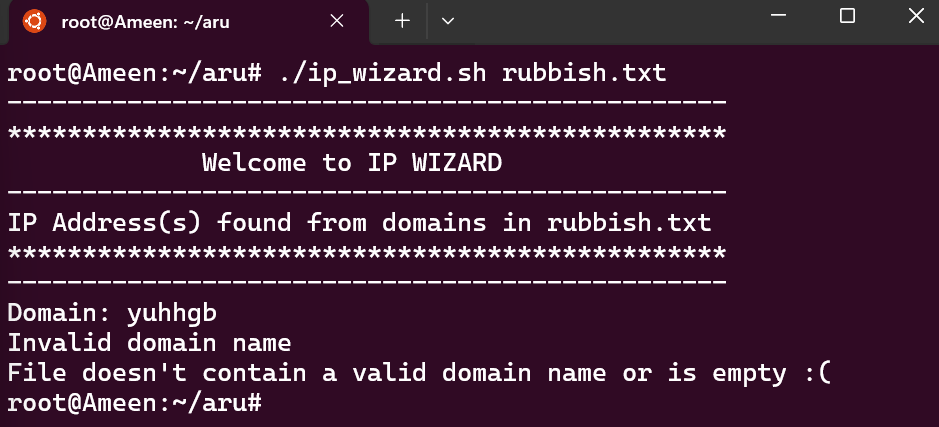
In the event the user wishes to quit the program, the input of “2” will bring an end to the program and a “Thank you for using IP WIZARD ☺“ prompt will be displayed.



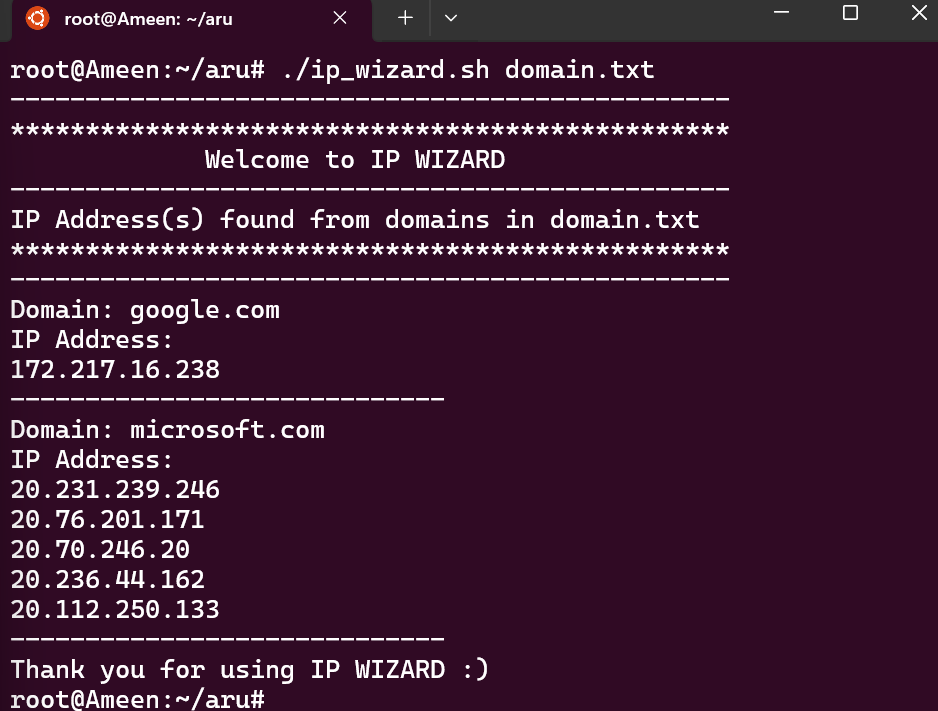
If the user enters a file that does not exist, a “File dose not exist” prompt appears.



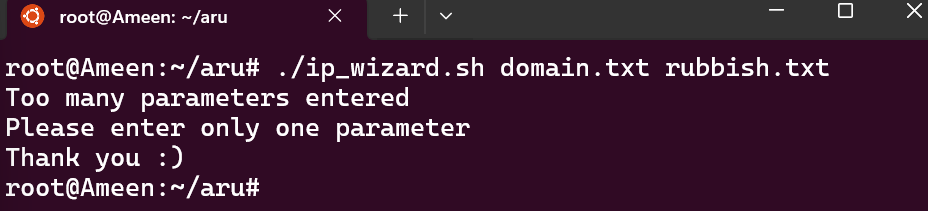
If the file entered as the parameter is empty or it does not contain a valid domain name an “Invalid domain name” prompt appears.



If the user enters a valid file that contains domain names a list of domain names and their corresponding IP address(es) appear.



If the users entered more than one parameter a “Too many parameters entered” prompt appears.



PART 1B

MS-DOS BASH SCRIPT

**SOURCE CODE:**

@echo off

setlocal enabledelayedexpansion

REM Display a welcome message

echo ------------------------------------------------

echo \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

echo Welcome to IP WIZARD

echo \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

echo ------------------------------------------------

REM Check if a command line parameter (domain name) is provided

if "%~1"=="" (

REM If no parameter is provided, call the menu function

call :menu

) else if not "%~2"=="" (

REM If more than one parameter is provided, display an error message and exit

echo Too many parameters entered

exit /b

) else (

REM If exactly one parameter is provided, proceed with the script

REM Start a loop to read the file line by line

for /F "tokens=\*" %%A in (%~1) do (

REM Display the current domain name

echo Domain: %%A

REM Set a flag to determine whether to print the IP address

set printFlag=no

REM Start another loop to run the nslookup command on the current domain name and parse its output

for /F "delims=" %%B in ('nslookup %%A 2^>nul') do (

REM Check if the current line of the nslookup output contains the string "Addresses: "

REM If it does, set the printFlag to "yes"

echo %%B | findstr /R /C:"Addresses: " >nul && set printFlag=yes

REM If the printFlag is "yes", print the current line of the nslookup output (the IP address)

if "!printFlag!"=="yes" (

echo %%B

)

)

REM If the printFlag is still "no", no IP address was found, so display an error message

if "!printFlag!"=="no" (

echo Invalid domain name or network error

)

REM Print a separator line

echo -----------------------------

)

REM Thank the user for using the script and then exit

echo Thank you for using IP WIZARD

exit /b

)

REM Define the menu function

:menu

REM Display a menu to the user and prompt them to make a choice

echo Please select the option you want to proceed:

echo 1) input a domain name

echo 2) quit from the program

set /p choice=Enter your choice:

REM If the user's choice was "1", prompt the user to enter a domain name and then perform the same nslookup process as before

if "!choice!"=="1" (

set /p domain=Enter a domain name:

echo -----------------------------

echo Domain: !domain!

set printFlag=no

for /F "delims=" %%B in ('nslookup !domain! 2^>nul') do (

echo %%B | findstr /R /C:"Addresses: " >nul && set printFlag=yes

if "!printFlag!"=="yes" (

echo %%B

)

)

echo -----------------------------

if "!printFlag!"=="no" (

echo Invalid domain name or network error

)

REM Check if the user has entered 2 to quit

if "!choice!"=="2" (

echo Thank you for using IP WIZARD

exit /b

)

goto :menu

) else if "!choice!"=="2" (

REM If the user's choice was "2", thank the user and exit the script

echo Thank you for using IP WIZARD

exit /b

) else (

REM If the user's choice was neither "1" nor "2", display an error message and go back to the menu

echo Invalid choice. Please enter 1 or 2.

goto :menu

)

**PROGRAM PSEUDOCODE:**

1. Start

2. Display "Welcome to IP Wizard"

3. Check if a command line parameter (domain name) is provided

- If no parameter is provided, go to step 4 (Menu)

- If more than one parameter is provided, display "Too many parameters entered" and End

- If exactly one parameter is provided, go to step 5

4. Menu:

- Display options: "1) input a domain name" and "2) quit from the program"

- If choice is "1", ask for a domain name and go to step 5

- If choice is "2", display "Thank you for using IP WIZARD :)" and End

- If choice is neither "1" nor "2", display "Invalid choice. Please enter 1 or 2." and go back to Menu

5. Perform nslookup on the domain name

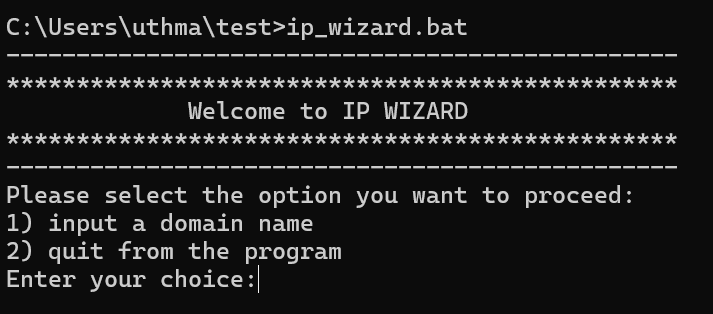
- If IP address is found, display it

- If no IP address is found, display "Invalid domain name or network error"

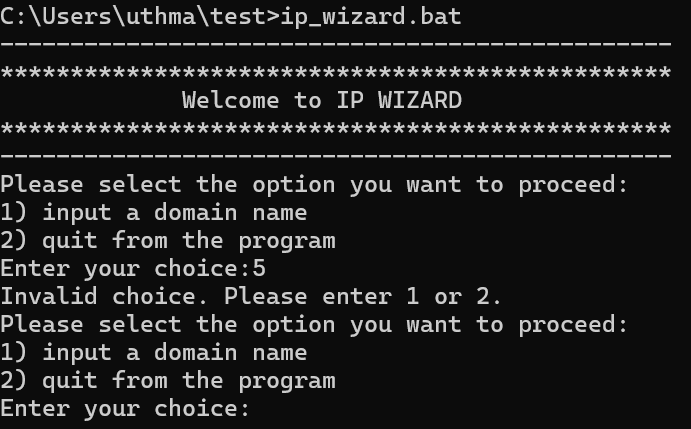
6. Display "Thank you for using IP WIZARD :)" and End

**PROGRAM SCREENSHOTS:**

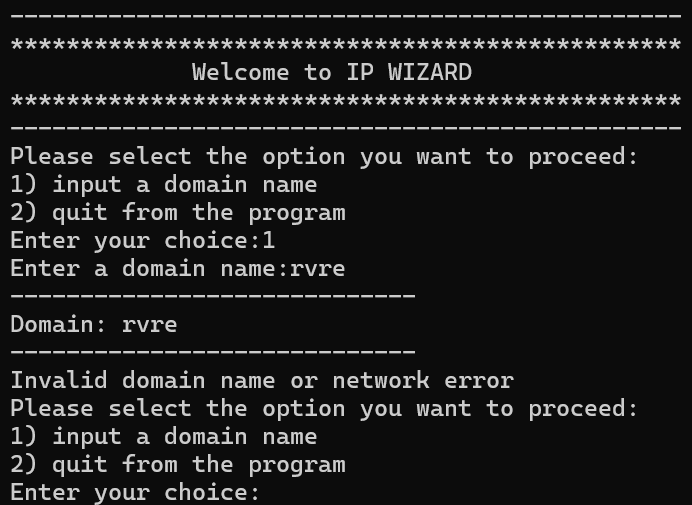
When the user adds no argument, he gets to select an option from the menu, option 1 “Input a domain name” or option 2 “Quit from the program”.



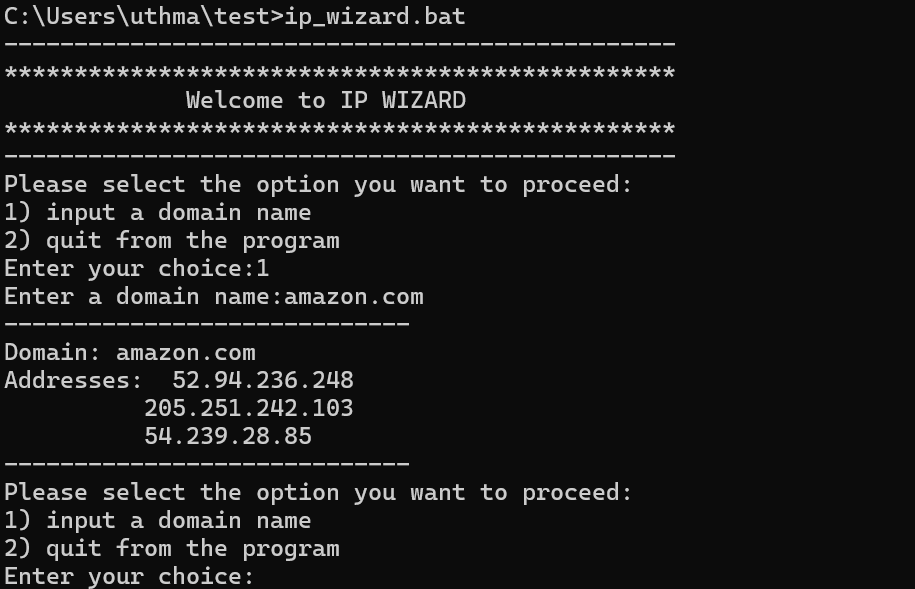
If the user enters an invalid menu choice (i.e any other input asides from “1” & “2”), the program will display an “Invalid choice. Please enter 1 or 2” prompt. This will be the response until an appropriate option is picked.



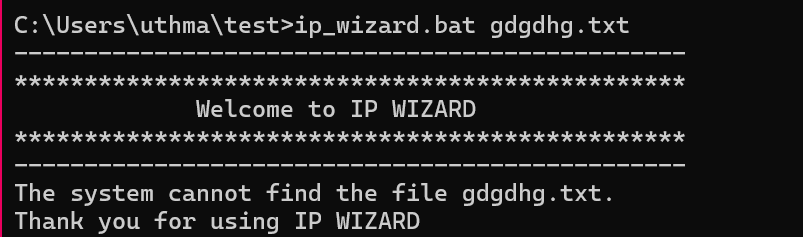
When the user selects option “1” to input domain name and inputs an invalid domain name a prompt “Invalid domain name” will appear and the user won’t be able to proceed until a valid domain name is entered.



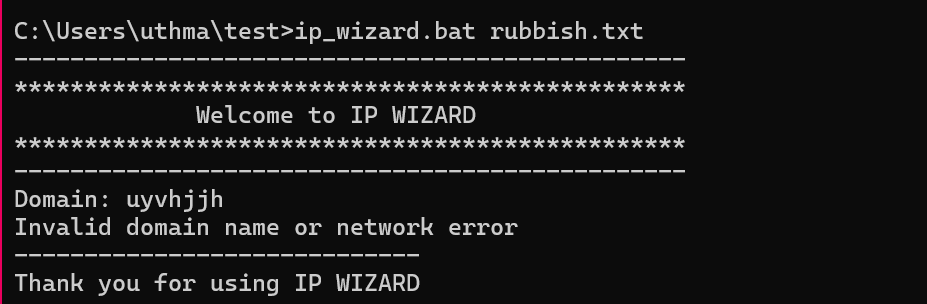
When the user enters a valid domain name the program provides the IP Address for the domain, and the user is shown the menu again to either input a domain name or quit the program.



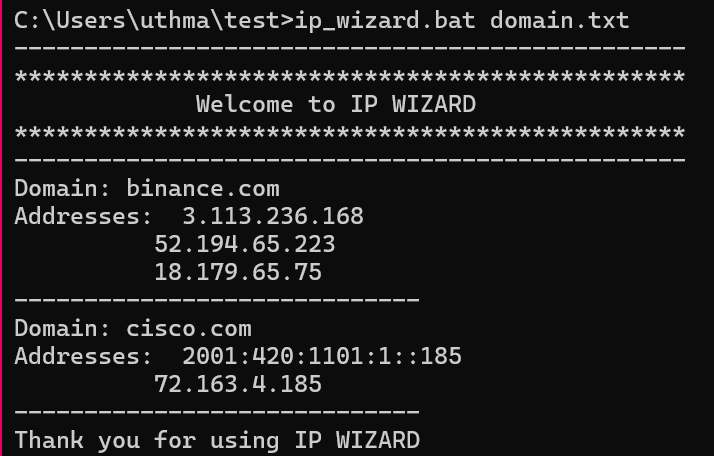
If the user enters a file that does not exist, “The system cannot find the file” prompt appears.



If the file entered as the parameter does not contain a valid domain name an “Invalid domain name or network error” prompt appears.



If the user enters a valid file that contains domain names a list of domain names and their corresponding IP address(es) appear.



If the users entered more than one parameter a “Too many parameters entered” prompt appears.



# Part 2: Research Questions

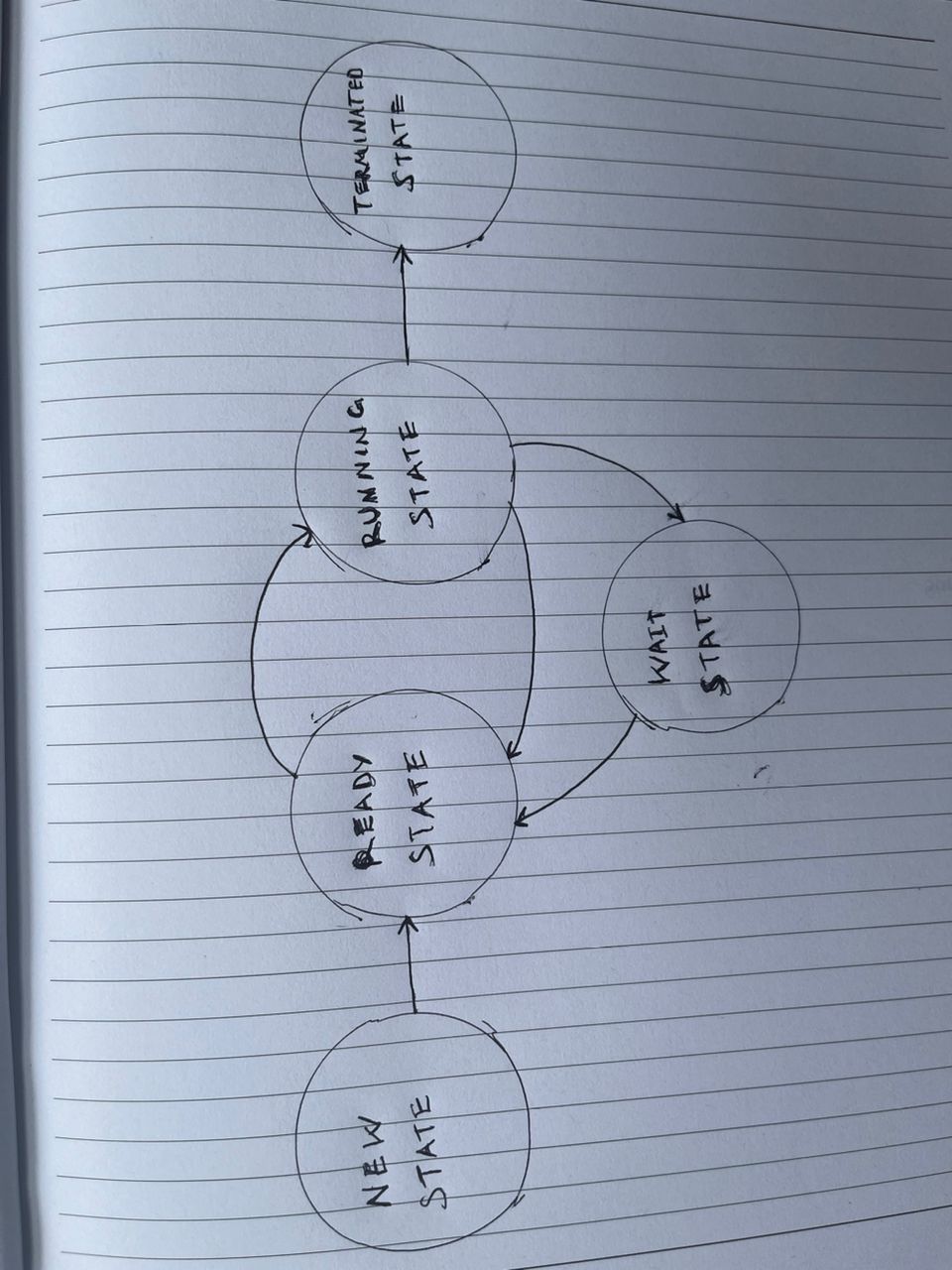
# Question 1

Describe the life cycle of a process, using diagrams, with details about each state. Write down the conditions that a process is switched:

(i) from Ready state to Running state

(ii) from Running state to Ready state

**ANSWER**



A process's life cycle usually consists of multiple states, each of which denotes a distinct phase of the process's execution. The life cycle states of a process are described here, along with information about each state and the prerequisites for changing between them:

**New State:** A process goes into the new state when it is first created.

In this phase, memory and CPU time are allocated by the operating system, which also initialises the process.

Until it is prepared to be carried out, the process stays in its new condition.

The process is loaded into main memory and is waiting to be allocated to a CPU for execution when it is in the ready stage.

**Ready State:** The process is loaded into main memory and is waiting to be allocated to a CPU for execution when it is in the ready stage.

The CPU is now working on another process, but the procedure is ready to begin.

Requirements to enter the running state:

Using scheduling techniques like round-robin or priority scheduling, the CPU scheduler determines which process will be executed first.

**Running State:** The process is actively being carried out by the CPU when it is in the operating state.

The process is using CPU resources and carrying out its instructions.

Requirements for the changeover to the prepared state:

When a higher-priority process becomes available or the current process's time slice expires, the CPU scheduler pre-empts the running process to make room for another.

**Blocked (Wait) State:** While waiting for an outside event or resource, a process that is in the blocked state is momentarily unable to move forward.

Waiting for user input, a semaphore or mutex, or the completion of I/O operations are common causes for the blocked state to occur.

Requirements for the changeover to the prepared state:

The process can now continue because the external event or resource has become available. After that, the process enters the ready state and waits for a CPU to be assigned.

**Terminated (Exit) State:** In this state, the user or the operating system have either ended the process or it has finished executing.

Process-related data structures are deallocated, and resources assigned to the process are released.

1. from Ready state to Running state

when a task is chosen for execution from the ready queue by the CPU scheduler.

The scheduling techniques used for this selection include round-robin, shortest job next, and priority scheduling.

After that, the CPU is tasked with carrying out the chosen process.

1. from Running state to Ready state

When a process with a higher priority becomes available for execution or the current process's time slice ends, the CPU scheduler will pre-empt it.

The operating process is stopped by the CPU scheduler, which then returns it to the ready queue.

When the CPU scheduler chooses the interrupted process for future execution, it is ready to resume.

# Question 2

Shortest Job First (SJF) is a relatively simple CPU scheduling algorithm for process management. Consider that there are two busy multi-programming systems A and B. The new processes created on System A are mostly long CPU burst processes with only occasional short CPU burst processes. The new processes created on System B are mostly very short CPU burst processes.

(i) State whether process starvation is likely to occur on either or both System A and System B.

(ii) If process starvation occurs on System A, state which type of processes is likely to be starved. Similarly, if process starvation occurs on System B, state which type of processes is likely to be starved. Your answer should refer to relevant characteristics of SJF algorithm.

**ANSWER**

i. Starvation process is more likely to occur in system A because of its lengthy CPU burst processes, which might cause shorter processes to wait for extended periods of time.

While it is less likely to occur in system B because it handles very short CPU burst tasks fast, there is reduced chance of process hunger because fewer processes have to wait for other processes to finish.

ii. In both cases, the SJF Characteristic which prioritise shorter CPU burst processes, can lead to process starvation for the opposite type of processes, for system A short burst processes will likely starve, while long burst processes will likely starve in System B. This emphasises the importance of understanding the workload characteristics and system environment when selecting an appropriate scheduling algorithm.

# Question 3

Consider the kitchen example and you are now the manager of the kitchen. There are 3 bowls, 2 stirrer, and 1 measuring cups in the kitchen. There are 3 chefs working in the kitchen. Chef C1 is holding 2 bowls, 1 cup, and needing 1 stirrer. Chef C2 is holding one bowl, one stirrer, and needing one cup. Chef C3 is holding one stirrer and needing one stirrer.

(i) There will be bad consequences when a deadlock occurs in the kitchen. Your boss is very concerned and ask you in theory the conditions for deadlock to occur. Answer your boss.

(ii) Draw a Resource Allocation Graph (RAG) to describe the resource allocation situation outlined above.

(iii) Based on your RAG and discuss if a deadlock has occurred or the situation concerning deadlock.

(iv) In a real computer system, what will be the chefs, bowls, cups, stirrers represented for? Justify your answers with reasons.

**ANSWER**

(i) In theory, there are four conditions that must hold simultaneously for a deadlock to occur:

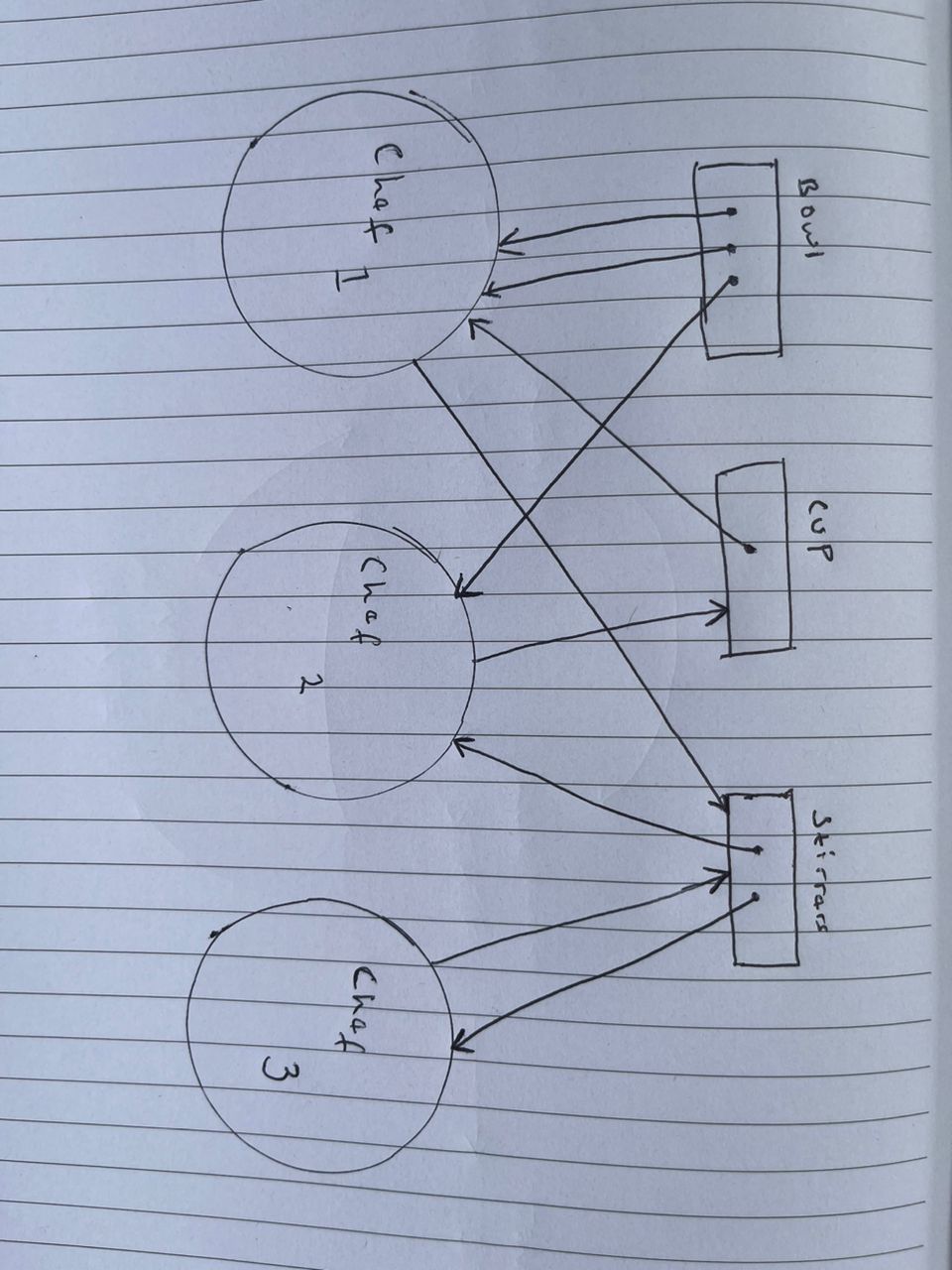
1. Mutual Exclusion: This condition states that at least one resource must be held in a non-sharable mode. In other words, only one process at a time can use the resource. If another process requests that resource, the requesting process must be delayed until the resource has been released.

2. Hold and Wait: This condition states that a process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.

3. No Pre-emption: This condition states that resources cannot be pre-empted. The resources can only be released voluntarily by the process holding them, after that process has completed its task.

4. Circular Wait: This condition states that there must be a circular chain of two or more processes, each of which is waiting for a resource that the next process in the chain holds.

(ii) Resource Allocation Graph (RAG):



- The chefs (C1, C2, C3) would be represented as processes.

- The kitchen items (bowls, stirrers, cups) would be represented as resources.

- If a chef is holding an item, there would be an arrow from the item to the chef.

- If a chef is waiting for an item, there would be an arrow from the chef to the item.

(iii) Based on the description, it doesn't seem like a deadlock has occurred. While some chefs are waiting for resources, there's no circular wait condition present. For example, C1 is waiting for a stirrer, but no other chef is waiting for a resource that C1 holds.

(iv) In a real computer system:

- The chefs would represent processes or threads.

- The bowls, cups, stirrers would represent resources, such as memory, files, or I/O devices.

- The act of a chef holding an item would represent a process that has been allocated a resource.

- The act of a chef needing an item would represent a process that is waiting for a resource.

The reasons for this representation are that processes in a computer system often need to acquire resources to perform tasks, similar to how chefs need kitchen items to prepare meals. Deadlocks in a computer system occur when processes are unable to proceed because the resources they need are being held by other processes, similar to chefs being unable to prepare meals because other chefs are using the items they need.

# Question 4

Consider an online computer system booking system for air tickets. It is expected that thousands of customers are using the system at the same time. A customer typically spends around 10 minutes to complete the querying, booking, and payment processes. Normally customers access the booking systems using a smartphone application that communicates with a web server for almost every operation.

(i) One design decision is about selecting the most suitable CPU scheduling algorithm for the server. In the evaluation, suggest the performance metric that is most relevant? Justify your answer.

(ii) To support more and more customers, the server needs an upgrade. Suggest whether it is more important to increase the size of the main memory or using a more powerful CPU. Justify your answer.

(iii) Study the following article.

<https://www.catchpoint.com/blog/web-caching>

Based on the information given in the article, discuss if web caching is a suitable solution to achieve the performance metric you answered in part (i).

**ANSWER**

i. The average response time or latency that clients experience is an important performance statistic to take into consideration when determining which CPU scheduling method is best for the server in the online booking system. This metric calculates how long it takes the system to react to queries from users and to complete booking and payment procedures. Better performance and a more responsive system are indicated by a reduced average response time, which has a direct impact on user experience and customer satisfaction.

ii. Increasing the main memory size is more crucial in the case of updating the server to accommodate more users than utilising a more potent CPU. This is due to the fact that managing concurrent user requests and data processing—tasks that would greatly profit from a bigger main memory—are the core functions of the online booking system. By enabling the server to keep more data in memory, the main memory size can be increased. This will decrease the frequency of disc I/O operations and enhance system responsiveness and performance.

iii. By caching frequently accessed booking-related data, such as flight schedules, ticket availability, and payment processing pages, closer to the users, the system can significantly reduce the time taken to fetch and deliver this information, leading to faster query, booking, and payment processes for customers. Therefore, web caching is a suitable solution to achieve the performance metric of reducing average response time in the online booking system.

# Question 5

In this question, you are asked to complete the two following questions given in the practical exercises. Program code with comments and explanation should be given in the Word report.

(i) Write a MS-DOS batch program which prompts the user for the name of a batch file, checks it exists (outputting an error and stopping if it does not) then goes on to ask the user for the author of the batch file’s name. It then adds two comment (rem) lines to the start of the batch file the first says “Written by” and the author’s name and the second the date and time. When it finishes the batch file should still be runnable with no errors! <https://canvas.anglia.ac.uk/courses/33881/files/4029059?module_item_id=1877624>

**MS-DOS Batch Program:**

@echo off

setlocal

REM Prompt the user for the name of the batch file

echo Please enter the name of the batch file:

set /p batchfile=

REM Check if the file exists

if not exist %batchfile% (

REM If the file does not exist, output an error message and stop the program

echo Error: File does not exist.

exit /b

)

REM If the file exists, prompt the user for the author's name

echo Please enter the author's name:

set /p author=

REM Get the current date and time

set datetime=%date% %time%

REM Add two comment lines at the start of the batch file

REM The first line says "Written by" followed by the author's name

REM The second line contains the date and time

echo REM Written by %author% >> %batchfile%

echo REM %datetime% >> %batchfile%

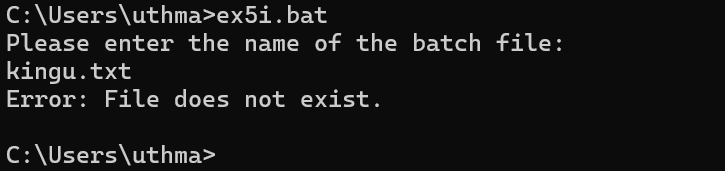
REM Delete the variables

set "batchfile="

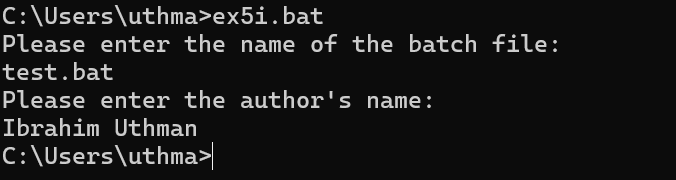
set "author="

**Program Screenshot:**

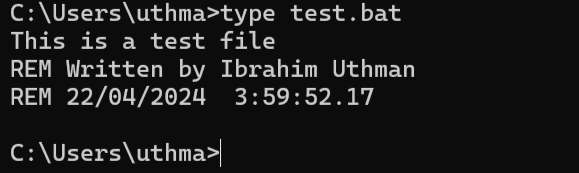
When the user enters batch file that does not exist



When he enters a batch file name that exists, he is prompted to Enter Author’s name



This is to show that the Name and Date were successfully appended to the end of the batch file



(ii) Write a bash script that uses a looping menu and case statements to perform a range of simple file operations. For each option the user is prompted for the correct number of parameters for the specific command and where necessary verifies the files/directories exist or not as appropriate. Option 1 is to copy one file to another, option 2 to rename one file to another, option 3 to create a directory and option 4 quit.

<https://canvas.anglia.ac.uk/courses/33881/files/4029061?module_item_id=1877627>

**Linux Bash Script:**

#!/bin/bash

# Start an infinite loop

while true; do

# Display the menu options

echo "1. Copy file"

echo "2. Rename file"

echo "3. Create directory"

echo "4. Quit"

# Prompt the user to choose an option

read -p "Choose an option: " option

# Perform an action based on the user's choice

case $option in

1) # Copy file

# Prompt the user to enter the source and destination files

read -p "Enter source file: " src

read -p "Enter destination file: " dest

# Check if the source file exists

if [ -f "$src" ]; then

# If it exists, copy it to the destination

cp "$src" "$dest"

echo "File copied successfully."

else

# If it doesn't exist, display an error message

echo "Source file does not exist."

fi

;;

2) # Rename file

# Prompt the user to enter the old and new file names

read -p "Enter old file name: " old

read -p "Enter new file name: " new

# Check if the old file exists

if [ -f "$old" ]; then

# If it exists, rename it to the new name

mv "$old" "$new"

echo "File renamed successfully."

else

# If it doesn't exist, display an error message

echo "File does not exist."

fi

;;

3) # Create directory

# Prompt the user to enter the directory name

read -p "Enter directory name: " dir

# Check if the directory exists

if [ ! -d "$dir" ]; then

# If it doesn't exist, create it

mkdir "$dir"

echo "Directory created successfully."

else

# If it exists, display an error message

echo "Directory already exists."

fi

;;

4) # Quit

# Display a message and break the loop to quit

echo "Exiting..."

break

;;

\*) # Invalid option

# Display an error message

echo "Invalid option."

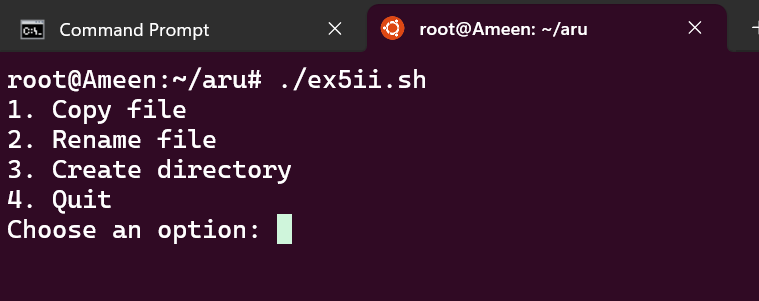
;;

esac

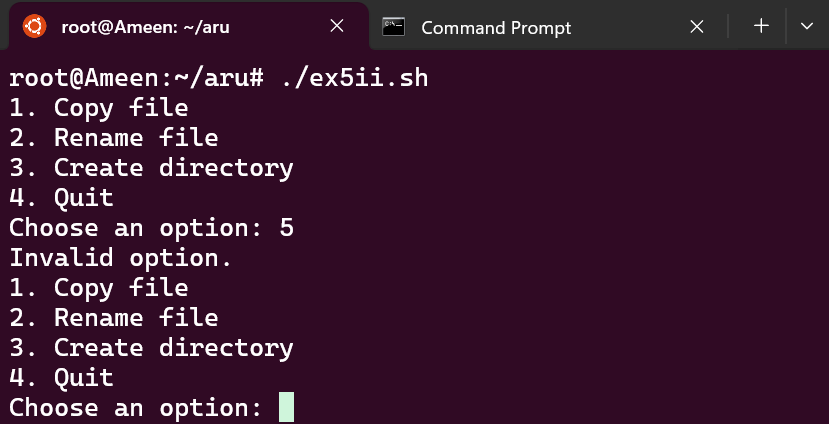
done

**Program Screenshot:**

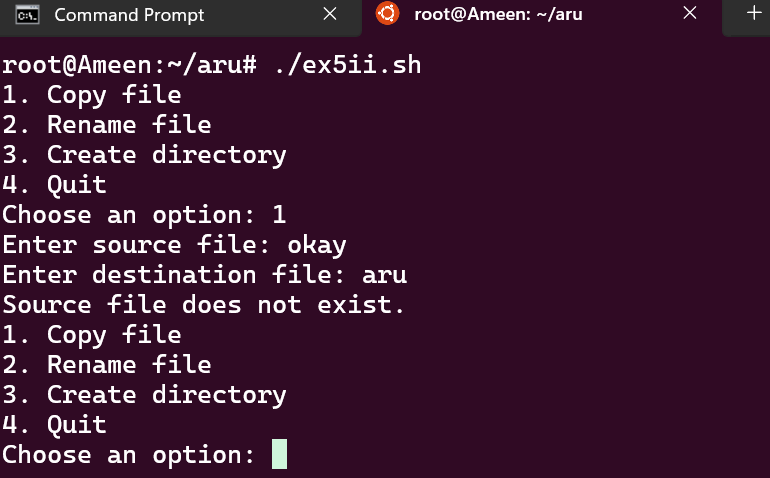
By running the code the user is displayed the menu to select from



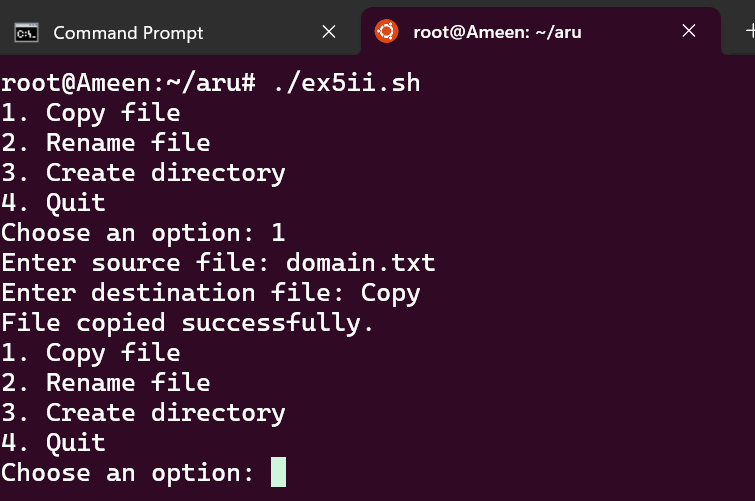
When the user selects an invalid option he is prompted an error message till he selects a valid option



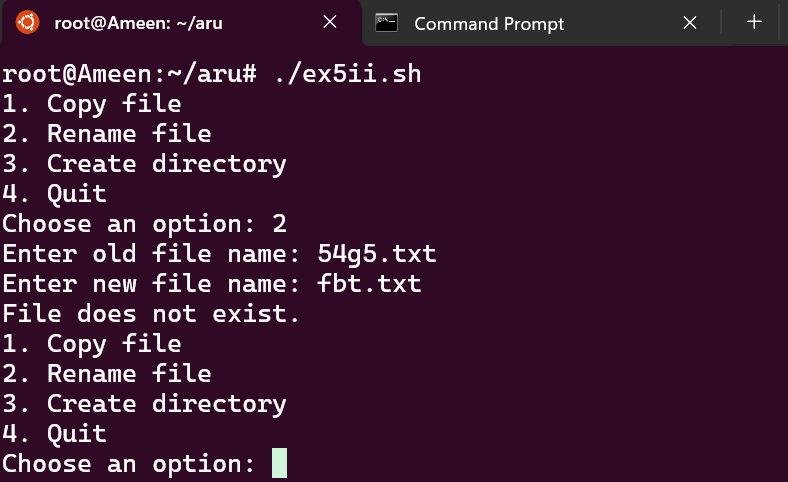
If the user enters a file to code that doesn’t exist he is shown an error message and shown the menu



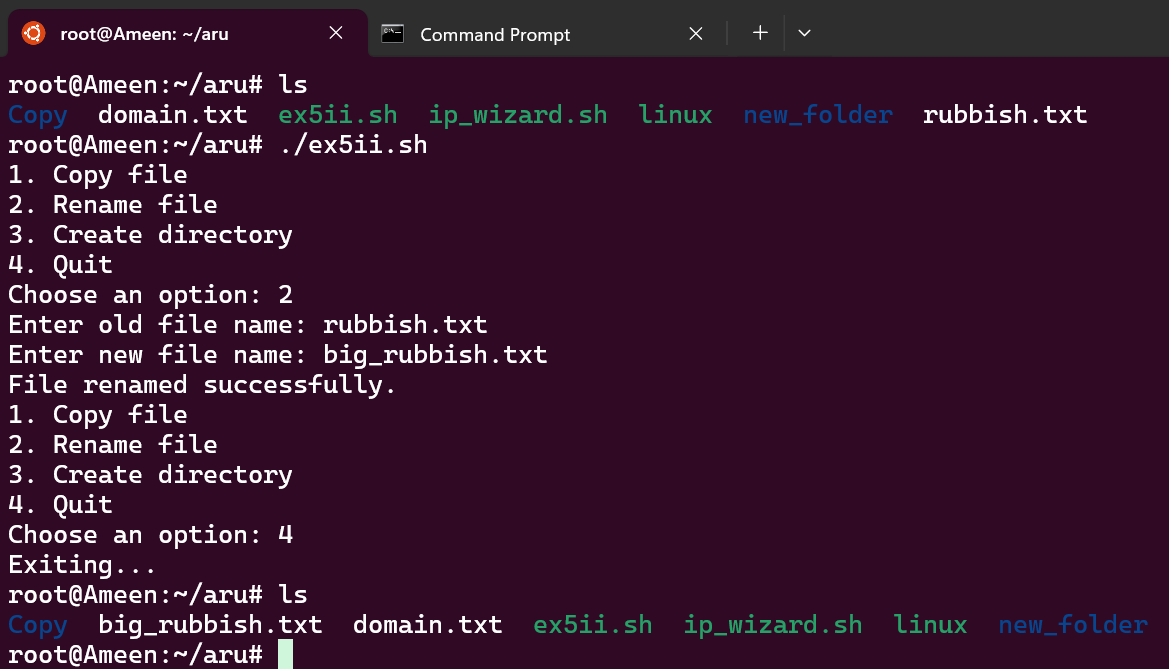
If the user enters a valid file to copy he receives an output to show it was successfully and he is shown the menu again



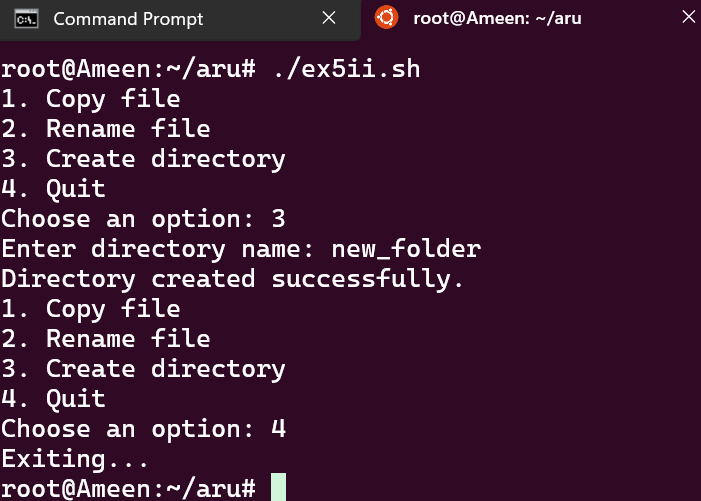
If a user enters a non-existing file to rename, he receives an error message and shown the menu again



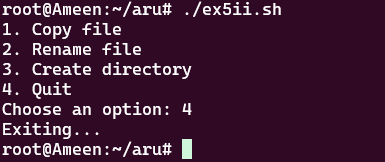
This is to show the program successfully renames a valid file entered by the user



When the user selects the option to create a directory, he receives a message to show it was successful and he is shown the menu again



In the case if the user selects option to exist



# References

Silberschatz, A., Galvin, P. B., & Gagne, G. (2019). Operating system concepts essentials (10th ed.). John Wiley & Sons.

Ali, S.M., Alshahrani, R.F., Hadadi, A.H., Alghamdi, T.A., Almuhsin, F.H. and El-Sharawy, E.E., 2021. A Review on the CPU Scheduling Algorithms: Comparative Study. *International Journal of Computer Science & Network Security*, *21*(1), pp.19-26.

Lawley, M. and Reveliotis, S., 2001. Deadlock avoidance for sequential resource allocation systems: Hard and easy cases. *International Journal of Flexible Manufacturing Systems*, *13*, pp.385-404.

Goel, N. and Garg, R.B., 2013. A comparative study of cpu scheduling algorithms.

Tonello, J.S., Practical Linux DevOps.