LONDON
METROPOLITAN
UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS5001NA Network and Operating Systems

Assessment Weightage & Type

20% Individual Coursework

Year and Semester 2020-21

Spring

Student Name: Aashman Uprety

London Met ID: 19031231

College ID: NP01NT4A190066

Assignment Due Date: 2021/04/11

Assignment Submission Date: 2021/04/11

Title (TASK A): Unix Script Programming

Title (TASK B): Memory Management

Word Count (TASK B): 2100

Table of Contents

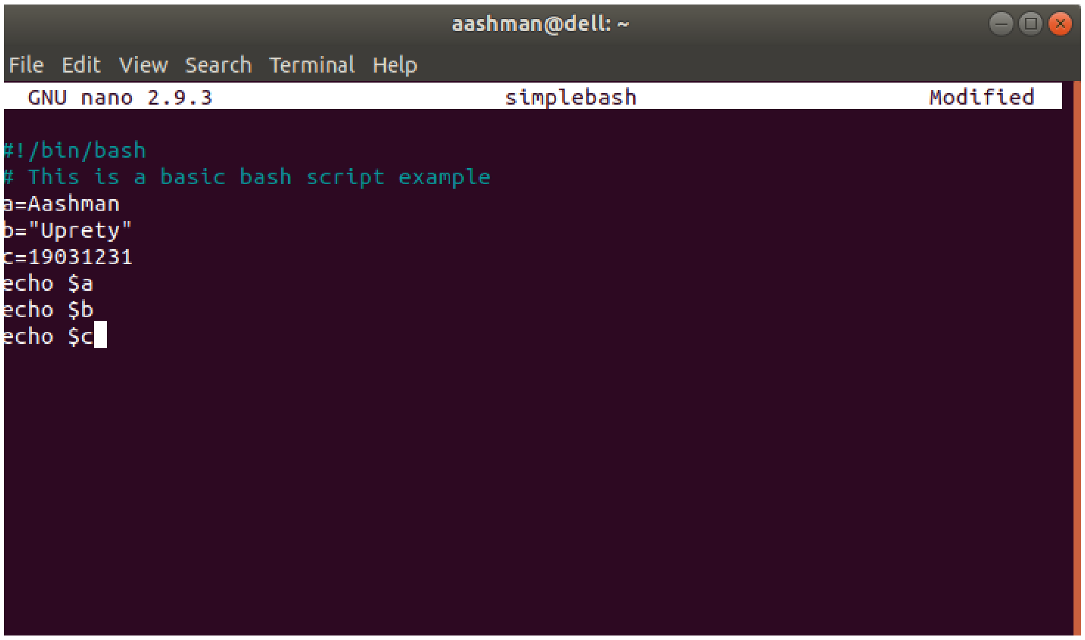
Table of Figures

Table of Tables

1. Task A

1.1 Introduction

Bash is also known as shell and is an interpreter that takes the command in the form of plain text and it calls the services in the operating system to do the specified tasks. For example, cd command changes the directory. The more advanced and improved version of Bourne Shell is called Bash. A shell scripting is writing a program for the shell to execute and a shell script is a file or program that shell will execute (Hiwarale, 2019). The default shell in the most Linux system is Bash because of which it is key to understand how to use it. A bash script file normally ends in .sh extension but it can also be declared by putting a shebang or hashbang at the top to make the interpreter understand it (Hiwarale, 2019). The hash and exclamation mark i.e. #! Is referred as Shebang. The thing to bear in mind is that it is not compulsory to write the shebang as it will run but its not practical and considered to be unwise. If you run a script without a shebang when using the Bash shell at a console, Bash will presume it is a Bash script so this can only work if the person running the code is running it in a Bash shell, which might not be the case for a number of reasons, which is risky (Hiwarale, 2019).



```
  GNU nano 2.9.3                  simplebash                  Modified

#!/bin/bash
# This is a basic bash script example
a=Aashman
b="Uprety"
c=19031231
echo $a
echo $b
echo $c
```

Figure 1: Simple Bash Script Example

1.2 Script

The program file or the script file has been presented below:

```
#!/bin/bash

name=$1

id=$2

key='aashman123'

count=1

date=$(date +%F)

time=$(date +%r)


if [ $# -ne 2 ];then

        echo "Two parameters are required to proceed into the program."

        exit

elif [[ $1 =~ ^[0-9]+$ ]];then

        echo "Name must be alphabets only."

        echo "//TRY AGAIN//"

        exit

elif ! [[ $2 =~ ^[0-9]+$ ]];then

        echo "Id must not be aplhabets."

        echo "//TRY AGAIN//"

        exit

fi


function_repeat(){

        echo -e "*************************************"
        echo -e "What do you want? To repeat or not to repeat the program ?
(Y/N): \c"

        read answer

        answer=${answer^^}

        echo -e "**************************************"

        if [[ $answer = "YES" || $answer = "Y"  ]];then

                function_country

        else

                echo -e "Preparing to exit the program.\c"

                echo -e "-+-+-+-+-+-+-+-+-+-+HAVE-A-NICE-DAY-+-+-+-+-+-+-+-+-
+"

                exit

        fi

}



function_player_select(){

        PS3="Choose only one option: "

        select code in $player1 $player2 $player3

        do

                case $code in

                $player1)

                        if [ -e $player1 ];then

                                echo "**************************************"

                                echo "You selected the player named: $player1"

                                cat $player1

                                function_repeat

                        else
                                echo "***********************************"
                                echo "Error: The program has no file named it!
TRY AGAIN"
                                function_country
                        fi
                        ;;
                        $player2)
                        if [ -e $player2 ];then
                                echo "***********************************"
                                echo "You selected the player named: $player2"
                                cat $player2
                                function_repeat
                        else
                                echo "***********************************"
                                echo "Error: The program has no file named it!
TRY AGAIN"
                                function_country
                        fi
                        ;;
                        $player3)
                        if [ -e $player3 ];then
                                echo "***********************************"
                                echo "You selected the player named: $player3"
                                cat $player3
                                function_repeat
                        else
                                echo "***********************************"
```

```
                                echo "Error: The program has no file named it! TRY AGAIN"

                                function_country

                        fi

                        ;;

                esac

        done

}



function_player(){

        player1='reset'

        player2='reset'

        player3='reset'

        echo -e "+****************+**********+"

        echo -e "| Players Name    |  Codes   |"

        echo -e "+****************+**********+"

        echo -e "| Lionel Messi    |   LM     |"

        echo -e "| Neymar Junior   |   NJ     |"

        echo -e "| Kiran Chemjong  |   KC     |"

        echo -e "| Zheng Zhi       |   ZZ     |"

        echo -e "| Harry Kane      |   HK     |"

        echo -e "+****************+*********+"

        until [[ ( "$player1" = "LM" || "$player1" = "NJ" || "$player1" = "HK" || "$player1" = "ZZ" || "$player1" = "KC" ) && ( "$player2" = "LM" || "$player2" = "NJ" || "$player2" = "HK" || "$player2" = "ZZ" || "$player2" = "KC" ) && ( "$player3" = "LM" || "$player3" = "NJ" || "$player3" = "HK" || "$player3" = "ZZ" || "$player3" = "KC" ) && ( "$player1" != "$player2" ) && ( "$player2" != "$player3" ) && ( "$player1" != "$player3" ) ]];do

                echo -e "*************************************"
                echo -e "Enter code of three player: \c"

                read player1 player2 player3

                player1=${player1^^}

                player2=${player2^^}

                player3=${player3^^}

                if [[ ${#player1} != '2' || ${#player2} != '2' || ${#player3}
!= '2' ]];then

                        echo -e "**************************************"

                        echo -e "Enter the codes of three players by separating
with spaces. \n PLEASE TRY AGAIN."

                elif [[ "$player1" = "$player2" || "$player2" = "$player3" ||
"$player1" = "$player3" ]];then

                        echo -e "**************************************"

                        echo -e "Repeated player names is not acceptable at any
cost. \n PLEASE TRY AGAIN."

                elif [[ "$player1" != "LM" && "$player1" != "NJ" && "$player1"
!= "HK" && "$player1" != "ZZ" && "$player1" != "KC" ]];then

                        echo -e "**************************************"

                        echo -e "See the codes above and enter the code
accordingly. Codes other then listed above are not allowed. \n PLEASE TRY
AGAIN."

                elif [[ "$player2" != "LM" && "$player2" != "NJ" && "$player2"
!= "HK" && "$player2" != "ZZ" && "$player2" != "KC" ]];then

                        echo -e "**************************************"

                        echo -e "See the codes above and enter the code
accordingly. Codes other then listed above are not allowed. \n PLEASE TRY
AGAIN."

                elif [[ "$player3" != "LM" && "$player3" != "NJ" && "$player3"
!= "HK" && "$player3" != "ZZ" && "$player3" != "KC" ]];then

                        echo -e "**************************************"

                        echo -e "See the codes above and enter the code
accordingly. Codes other then listed above are not allowed. \n PLEASE TRY
AGAIN."
```

```
                fi

        done

        echo -e "***************************************"

        function_player_select

}


function_country(){

        guess='reset'

        echo -e "+-------------+--------+"

        echo -e "|   Country   |  Codes |"

        echo -e "+-------------+--------+"

        echo -e "|   Brazil    |  BRZ   |"

        echo -e "|   Argentina |  ARG   |"

        echo -e "|   Nepal     |  NEP   |"

        echo -e "|   China     |  CHI   |"

        echo -e "|   England   |  ENG   |"

        echo -e "+-------------+--------+"

        while [[ "$guess" != "NEP" ]];do

                echo -e "***************************************"

                echo -e "Please Enter the Country code: \c"

                read guess

                guess=${guess^^}

                if [[ "$guess" != "NEP" ]];then

                        echo -e "***************************************"

                        echo -e "Sorry, code didn't matched. \n PLEASE TRY
AGAIN."
```

```
                fi


                if [[ "$guess" = "NEP" ]];then

                        echo -e "**************************************"

                        echo -e "You Did It."

                        echo -e "**************************************"

                        echo -e "Nepal is the best team in football"

                        echo -e "They have been doing very well these days."

                        echo -e "Kiran Chemjong is considered as the best
football player."

                        echo -e "**************************************"

                        function_player

                fi

        done


}


function_correct(){

        echo -e "**************************************"

        echo -e "WOW! You entered the correct key. You may proceed now!!"

        echo -e "**************************************"

        echo -e "Welcome, Your ID is $id\nName: $name\nDate: $date\nTime:
$time"

        echo -e "**************************************"

        function_country

}
```

```
secretkey(){

        while [[ "$key" != "$keyguess" ]];do

                echo -e "Enter the key to proceed into the program: \c"

                read keyguess

                if [[ "$key" != "$keyguess" ]];then

                        echo -e "**************************************"

                        echo -e "Sorry, it is not the right key"

                        (( attempt=4-$count ))

                        echo -e "You have $attempt chances left."

                        echo -e "Caution: Continuous failed attempt could lead
the program to terminate."

                        echo -e "**************************************"

                        (( count = count + 1))

                fi



                if [[ $count == 5 ]];then

                        echo -e "Caution: Too many failed attempts. Program is
exitting."

                        echo -e  "-+-+-+-+-+-+-+-+-+-+HAVE-A-NICE-DAY-+-+-+-+-
+-+-+-+-+-+"

                        exit

                fi



                if [[ "$key" == "$keyguess" ]];then

                        function_correct

                fi

        done
}

clear

echo ""

echo -e "You have 4 chances left."

secretkey
```

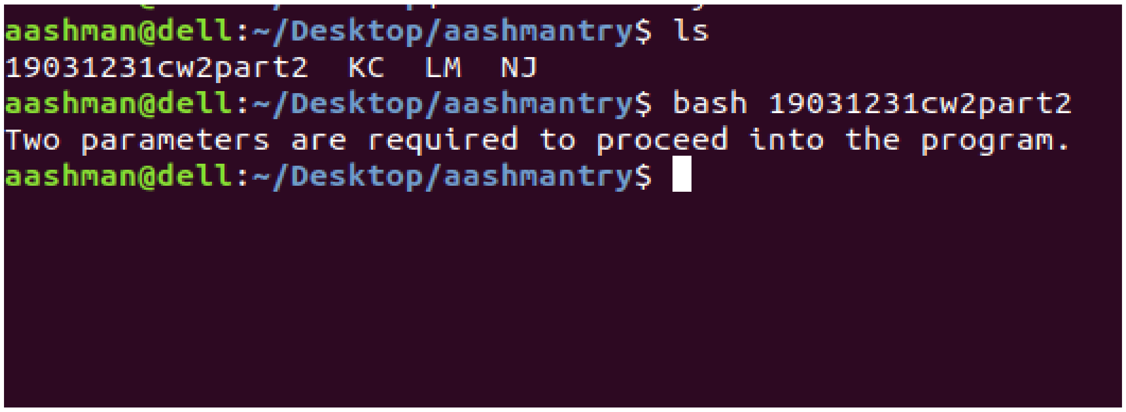
1.3 Testing

The testing session is very important as it helps to know whether our program meets the requirement or not. Number of testing were done to know whether the needed criteria was met or not in our program. The screenshots along with the task carried has been presented as a screenshot and in a tabulated form.

1.3.1 TEST 1: RUNNING WITHOUT USERNAME AND ID

Objective	To run the program without giving username and ID.
Action	Run the program without two parameters.
Expected Result	The message to input two parameters should be shown.
Actual Result	The message was displayed.
Conclusion	Test was successful.

Table 1: Table for test 1



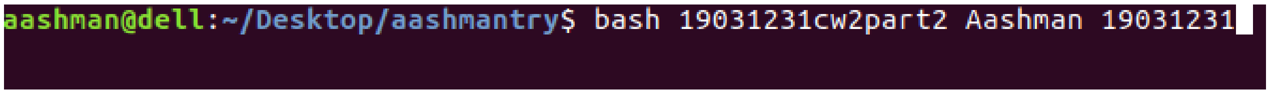
```
aashman@dell:~/Desktop/aashmantry$ ls
19031231cw2part2  KC  LM  NJ
aashman@dell:~/Desktop/aashmantry$ bash 19031231cw2part2
Two parameters are required to proceed into the program.
aashman@dell:~/Desktop/aashmantry$ 
```

Figure 2: Screenshot for test 1

1.3.2 TEST 2: RUNNING WITH CORRECT USERNAME AND PASSWORD

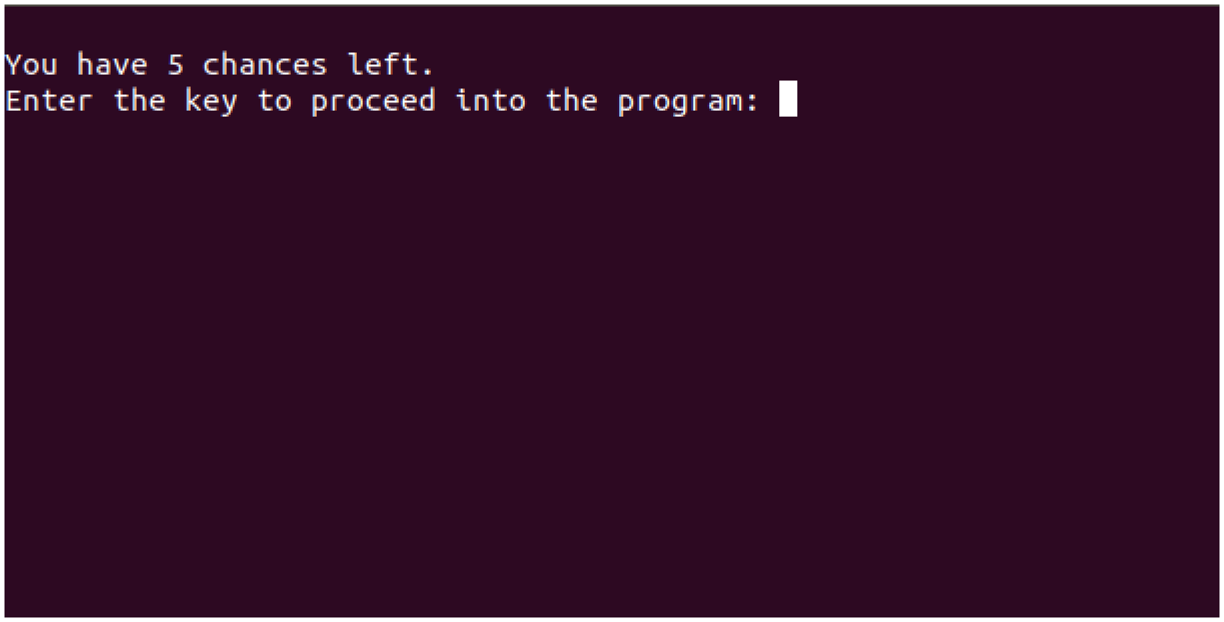
Objective	To run the program with correct username and password.
Action	Run the program without two parameters.
Expected Result	The program should run.
Actual Result	The program was started.
Conclusion	Test was successful.

Table 2: Table for test 2



```
aashman@dell:~/Desktop/aashmantry$ bash 19031231cw2part2 Aashman 19031231
```

Figure 3: Screenshot for test 2 (Command)



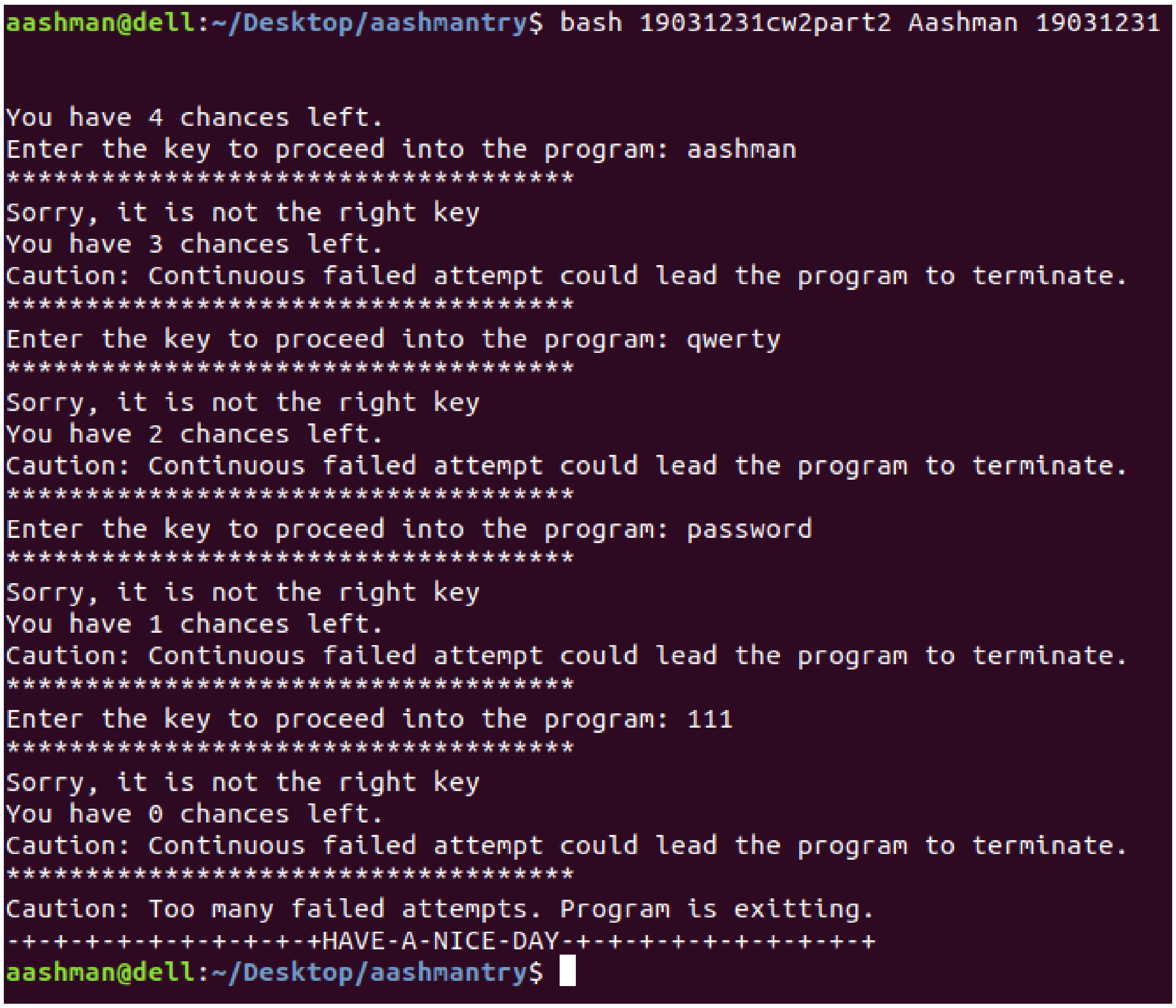
```
You have 5 chances left.
Enter the key to proceed into the program:
```

Figure 4: Screenshot for test 2 (Output)

1.3.3 TEST 3: RUNNING WITH INCORRECT PASSWORD FOUR TIMES

Objective	To run the program by giving incorrect password.
Action	Incorrect password was given four times.
Expected Result	The program gives an error message and terminate itself.
Actual Result	The program was terminated automatically.
Conclusion	Test was successful.

Table 3: Table for test 3



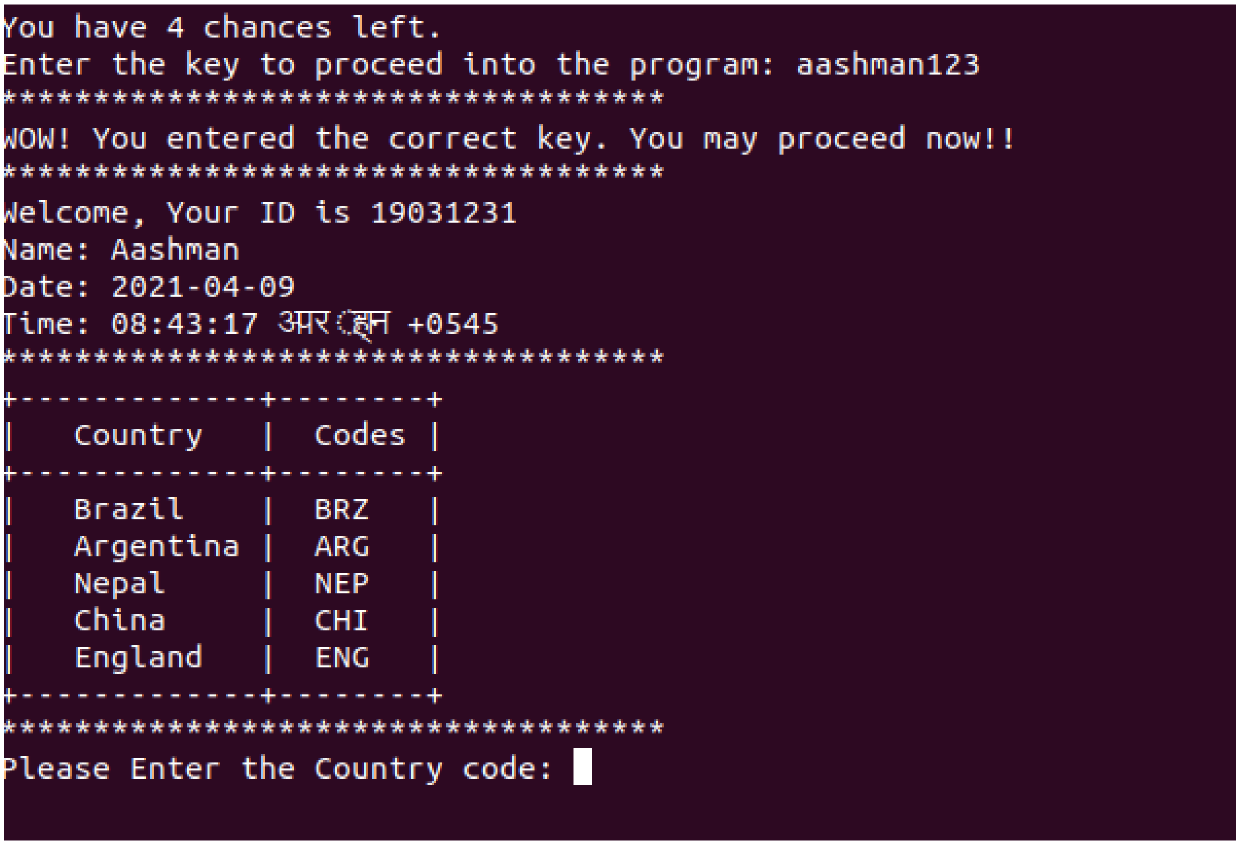
```
aashman@dell:~/Desktop/aashmantry$ bash 19031231cw2part2 Aashman 19031231


You have 4 chances left.
Enter the key to proceed into the program: aashman
************************************
Sorry, it is not the right key
You have 3 chances left.
Caution: Continuous failed attempt could lead the program to terminate.
************************************
Enter the key to proceed into the program: qwerty
************************************
Sorry, it is not the right key
You have 2 chances left.
Caution: Continuous failed attempt could lead the program to terminate.
************************************
Enter the key to proceed into the program: password
************************************
Sorry, it is not the right key
You have 1 chances left.
Caution: Continuous failed attempt could lead the program to terminate.
************************************
Enter the key to proceed into the program: 111
************************************
Sorry, it is not the right key
You have 0 chances left.
Caution: Continuous failed attempt could lead the program to terminate.
************************************
Caution: Too many failed attempts. Program is exitting.
-+-+-+-+-+-+-+-+-+-+HAVE-A-NICE-DAY-+-+-+-+-+-+-+-+-+-+
aashman@dell:~/Desktop/aashmantry$ 
```

Figure 5: Screenshot for test 3

1.3.4 TEST 4: RUN WITH CORRECT PASSWORD

Objective	To run the program with correct password.
Action	Run the program by giving correct secret key.
Expected Result	The user should be able to access the program.
Actual Result	The user was able to access the program.
Conclusion	Test was successful.

Table 4: Table for test 4



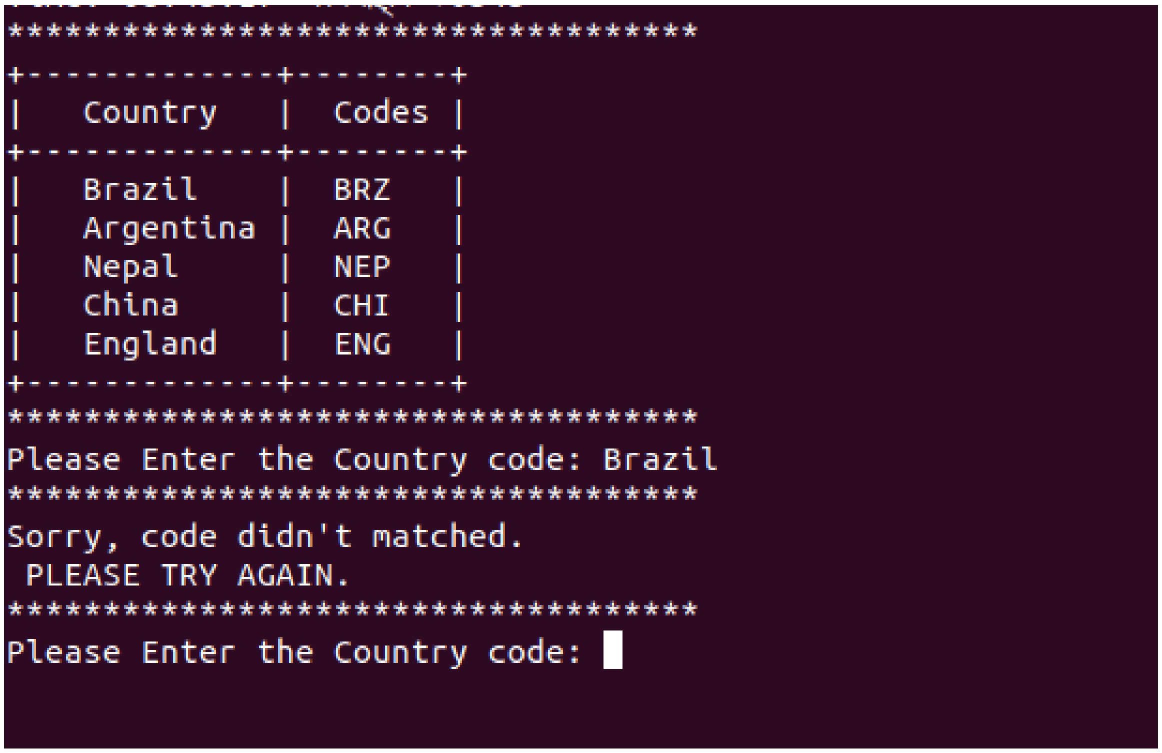
```
You have 4 chances left.
Enter the key to proceed into the program: aashman123
************************************
WOW! You entered the correct key. You may proceed now!!
************************************
Welcome, Your ID is 19031231
Name: Aashman
Date: 2021-04-09
Time: 08:43:17 अपराह्न +0545
************************************
+-------------+--------+
|   Country   |  Codes |
+-------------+--------+
|   Brazil    |  BRZ   |
|   Argentina |  ARG   |
|   Nepal     |  NEP   |
|   China     |  CHI   |
|   England   |  ENG   |
+-------------+--------+
************************************
Please Enter the Country code: 
```

Figure 6: Screenshot for test 4

1.3.5 TEST 5: RUN WITH FULL COUNTRY NAME

Objective	To give full country name.
Action	Run the program by entering the country name instead of code.
Expected Result	The program should not take it as an input and prompt the user to try again.
Actual Result	The program asked to try again.
Conclusion	Test was successful.

Table 5: Table for test 5



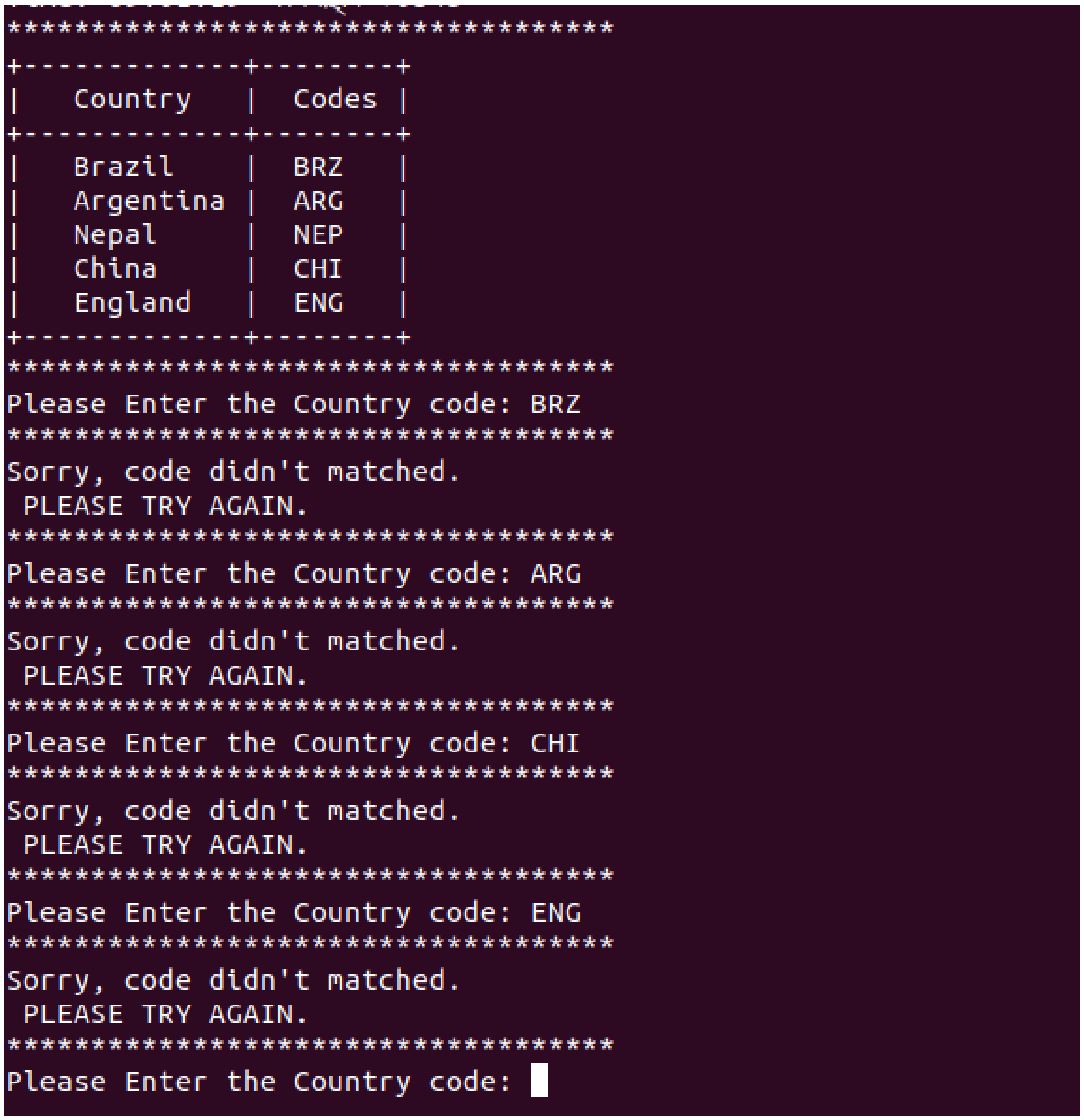
```
************************************
+-------------+--------+
|   Country   |  Codes |
+-------------+--------+
|   Brazil    |  BRZ   |
|   Argentina |  ARG   |
|   Nepal     |  NEP   |
|   China     |  CHI   |
|   England   |  ENG   |
+-------------+--------+
************************************
Please Enter the Country code: Brazil
************************************
Sorry, code didn't matched.
 PLEASE TRY AGAIN.
************************************
Please Enter the Country code: 
```

Figure 7: Screenshot for test 5

1.3.6 TEST 6: GIVING INCORRECT COUNTRY CODE

Objective	To give incorrect country code.
Action	Run the program by giving incorrect country code.
Expected Result	The program should give a prompt to try again and again until the correct code is guessed.
Actual Result	The program asked to try again until correct country code was guessed by the user.
Conclusion	Test was successful.

Table 6: Table for test 6



```
************************************
+-------------+--------+
|   Country   |  Codes |
+-------------+--------+
|   Brazil    |  BRZ   |
|   Argentina |  ARG   |
|   Nepal     |  NEP   |
|   China     |  CHI   |
|   England   |  ENG   |
+-------------+--------+
************************************
Please Enter the Country code: BRZ
************************************
Sorry, code didn't matched.
 PLEASE TRY AGAIN.
************************************
Please Enter the Country code: ARG
************************************
Sorry, code didn't matched.
 PLEASE TRY AGAIN.
************************************
Please Enter the Country code: CHI
************************************
Sorry, code didn't matched.
 PLEASE TRY AGAIN.
************************************
Please Enter the Country code: ENG
************************************
Sorry, code didn't matched.
 PLEASE TRY AGAIN.
************************************
Please Enter the Country code: 
```

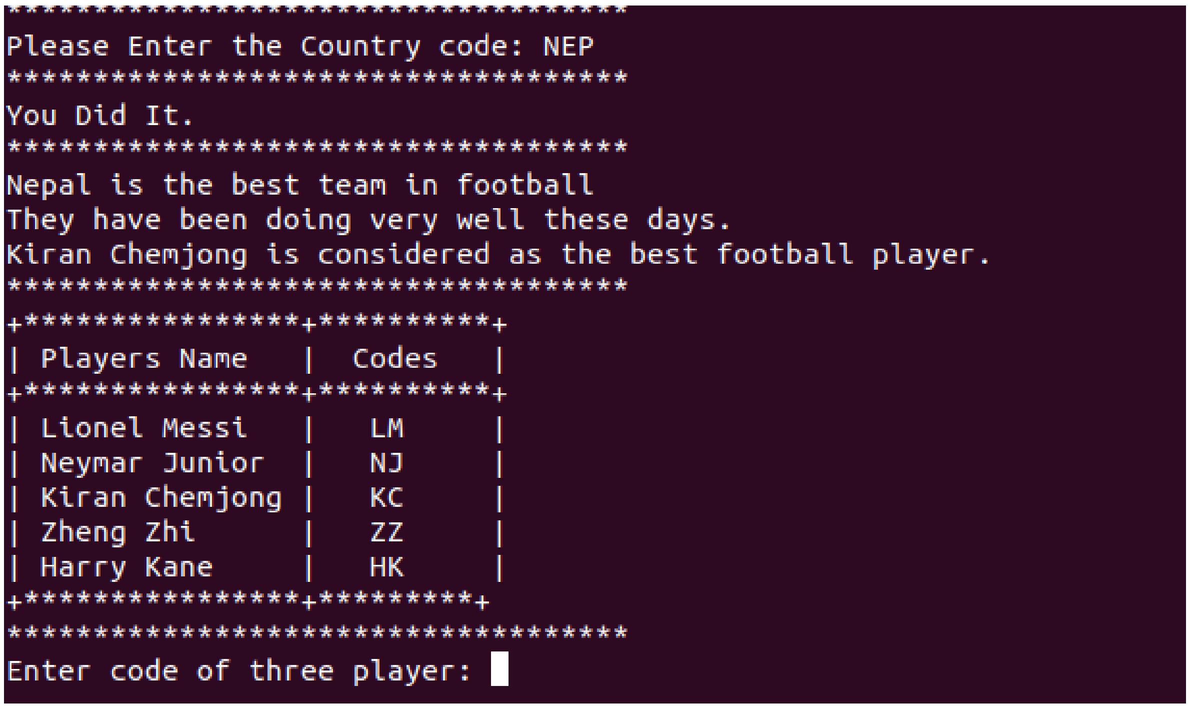
Figure 8: Screenshot for test 6

1.3.7 TEST 7: GIVING CORRECT COUNTRY CODE

Objective	To give correct country code.
Action	Run the program by giving correct country code.
Expected Result	The program should display some lines about the football team of that country. And it should ask to input code of three players.

Actual Result	The program displayed that the country code guessed was correct and displayed some lines. And it asked to input code of three players.
Conclusion	Test was successful.

Table 7: Table for test 7



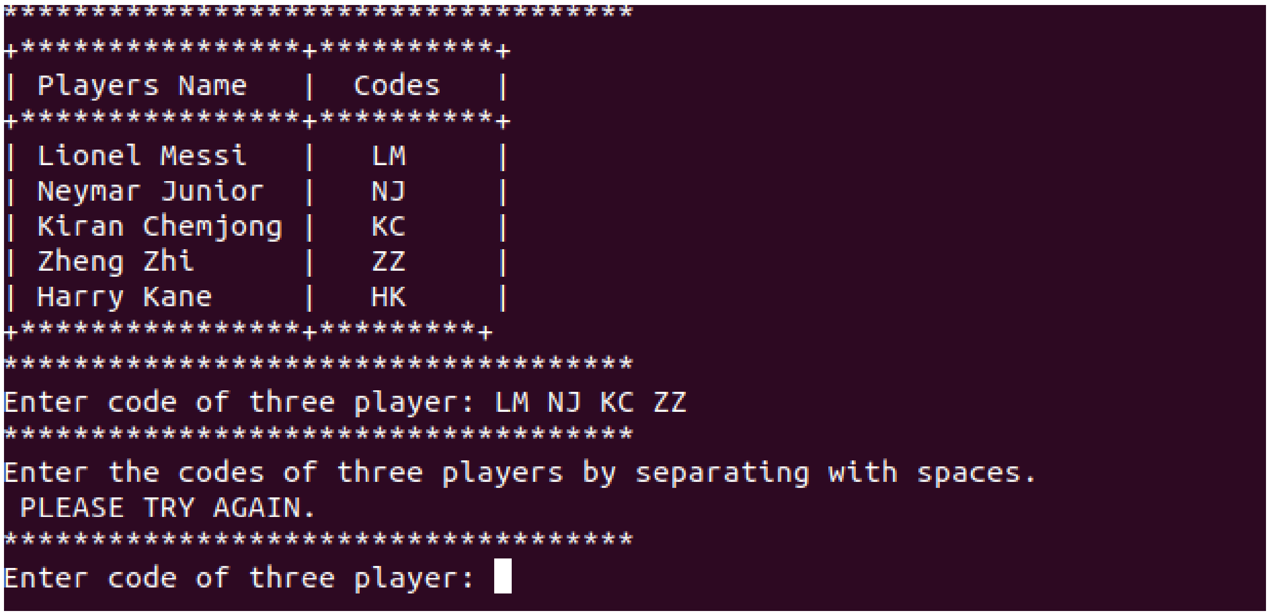
```
*************************************
Please Enter the Country code: NEP
*************************************
You Did It.
*************************************
Nepal is the best team in football
They have been doing very well these days.
Kiran Chemjong is considered as the best football player.
*************************************
+***************+**********+
| Players Name   |  Codes   |
+***************+**********+
| Lionel Messi   |   LM     |
| Neymar Junior  |   NJ     |
| Kiran Chemjong |   KC     |
| Zheng Zhi      |   ZZ     |
| Harry Kane     |   HK     |
+***************+*********+
*************************************
Enter code of three player:
```

Figure 9: Screenshot for test 7

1.3.8 TEST 8: PICKING FOUR PLAYER CODES

Objective	To give four player codes instead of three players.
Action	Four player codes were given.
Expected Result	The program prompts the user to type the codes of three players only by separating with spaces.
Actual Result	The program asked the user to type the three codes again by separating with spaces.
Conclusion	Test was successful.

Table 8: Table for test 8



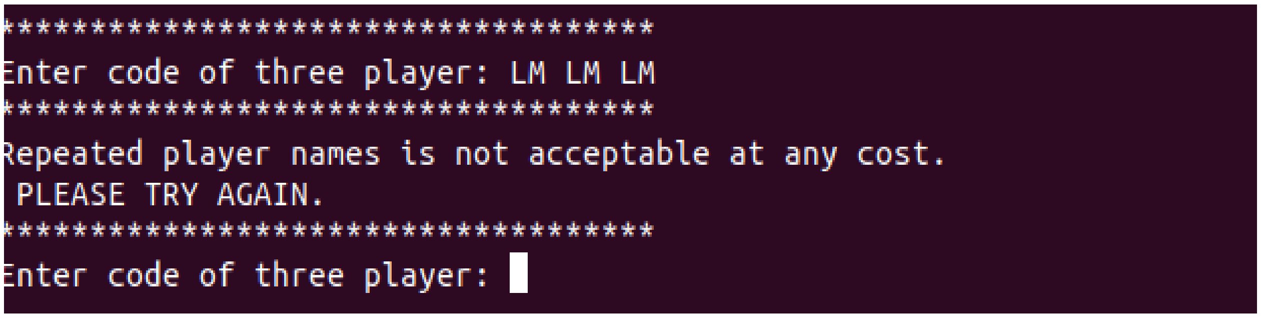
```
************************************
+***************+**********+
| Players Name   |  Codes   |
+***************+**********+
| Lionel Messi   |   LM     |
| Neymar Junior  |   NJ     |
| Kiran Chemjong |   KC     |
| Zheng Zhi      |   ZZ     |
| Harry Kane     |   HK     |
+***************+*********+
************************************
Enter code of three player: LM NJ KC ZZ
************************************
Enter the codes of three players by separating with spaces.
 PLEASE TRY AGAIN.
************************************
Enter code of three player: 
```

Figure 10: Screenshot for test 8

1.3.9 TEST 9: GIVING SAME PLAYER CODE

Objective	To give same player codes.
Action	Same player codes were given.
Expected Result	The program prompts the user by saying same player codes is not acceptable and to try again.
Actual Result	The program displayed that same player codes are not allowed at any cost and to try again.
Conclusion	Test was successful.

Table 9: Table for test 9



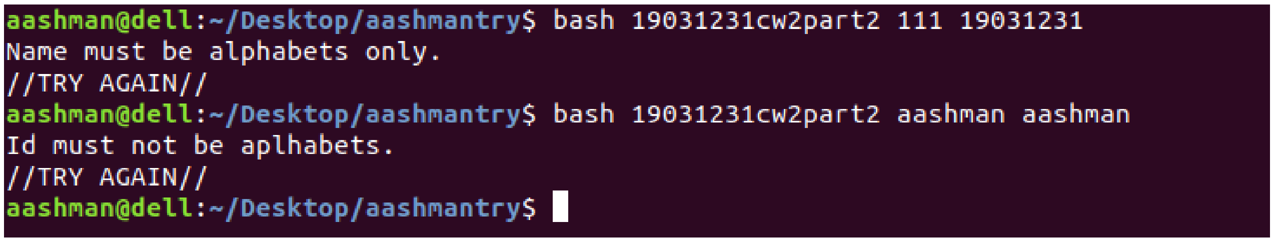
```
************************************
Enter code of three player: LM LM LM
************************************
Repeated player names is not acceptable at any cost.
 PLEASE TRY AGAIN.
************************************
Enter code of three player: 
```

Figure 11: Screenshot for test 9

1.3.10 TEST 10: RUNNING WITH WRONG USERNAME AND USER ID

Objective	To run the program by giving wrong username and user id.
Action	Wrong username was given.
Expected Result	The program prompts the user by saying name should be alphabets only and id should not be alphabets.
Actual Result	The program displayed that name should be alphabets only and id should not be alphabets.
Conclusion	Test was successful.

Table 10: Table for test 10



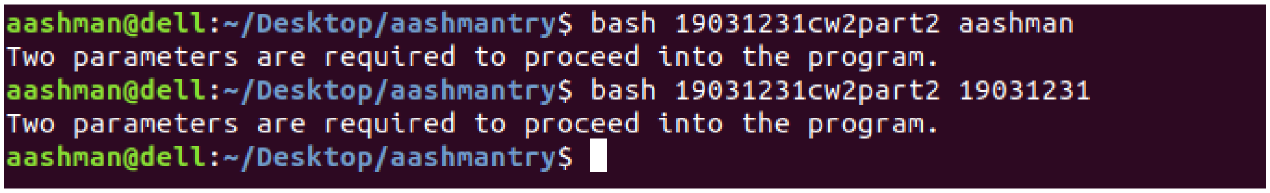
```
aashman@dell:~/Desktop/aashmantry$ bash 19031231cw2part2 111 19031231
Name must be alphabets only.
//TRY AGAIN//
aashman@dell:~/Desktop/aashmantry$ bash 19031231cw2part2 aashman aashman
Id must not be aplhabets.
//TRY AGAIN//
aashman@dell:~/Desktop/aashmantry$ 
```

Figure 12: Screenshot for test 10

1.3.11 TEST 11: RUNNING THE PROGRAM WITH ONLY ONE PARAMETER

Objective	To run the program by giving only one parameter.
Action	Only one parameter was given.
Expected Result	The program prompts the user that two parameters are required for the program to run.
Actual Result	The program displayed that two parameters are required for the program to run.
Conclusion	Test was successful.

Table 11: Table for test 11



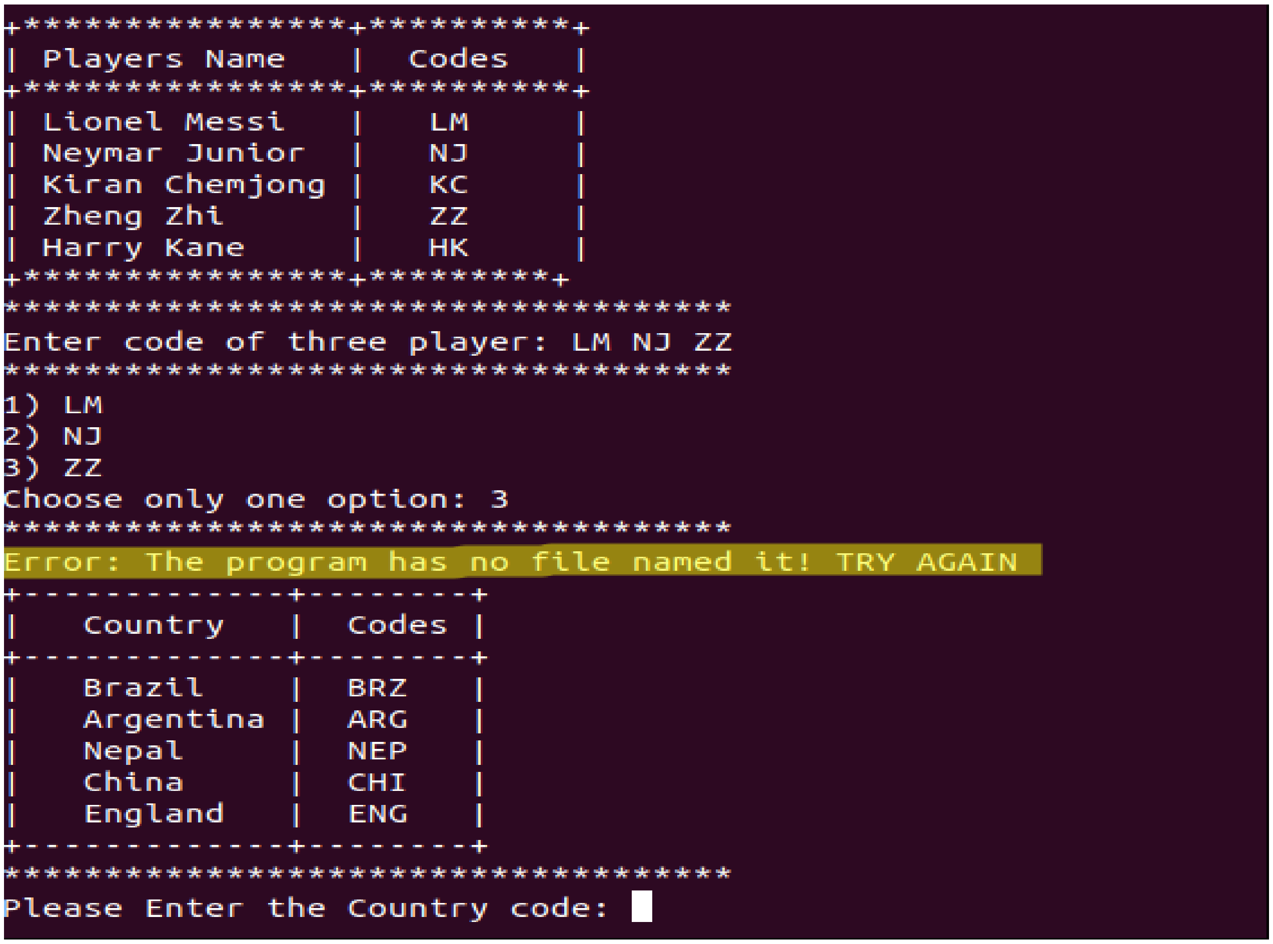
```
aashman@dell:~/Desktop/aashmantry$ bash 19031231cw2part2 aashman
Two parameters are required to proceed into the program.
aashman@dell:~/Desktop/aashmantry$ bash 19031231cw2part2 19031231
Two parameters are required to proceed into the program.
aashman@dell:~/Desktop/aashmantry$ 
```

Figure 13: Screenshot for test 11

1.3.12 TEST 12: NO EXTERNAL FILE OF PLAYER

Objective	To run the program and select the player that has no external file.
Action	The program was run and, in the step, to select one out of three players, the player having no external file was selected.
Expected Result	The program should throw an error saying the program has no file named it and the program should go to country selection automatically.
Actual Result	The program throwed an error saying the program has no file named it and the program went to country selection automatically.
Conclusion	Test was successful.

Table 12: Table for test 12



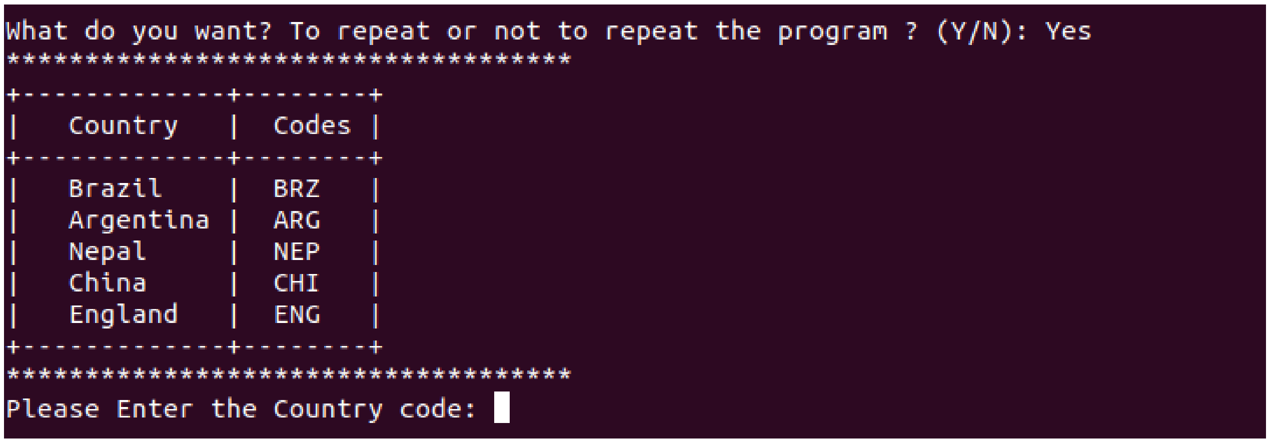
```
+****************+**********+
| Players Name    |  Codes   |
+****************+**********+
| Lionel Messi    |   LM     |
| Neymar Junior   |   NJ     |
| Kiran Chemjong  |   KC     |
| Zheng Zhi       |   ZZ     |
| Harry Kane      |   HK     |
+****************+*********+
************************************
Enter code of three player: LM NJ ZZ
************************************
1) LM
2) NJ
3) ZZ
Choose only one option: 3
************************************
Error: The program has no file named it! TRY AGAIN
+-------------+--------+
|   Country   |  Codes |
+-------------+--------+
|   Brazil    |  BRZ   |
|   Argentina |  ARG   |
|   Nepal     |  NEP   |
|   China     |  CHI   |
|   England   |  ENG   |
+-------------+--------+
************************************
Please Enter the Country code:
```

Figure 14:Screenshot for test 12

1.3.13 TEST 13: TO REPEAT THE PROGRAM (YES)

Objective	To type yes while program asks to exit the program or not.
Action	'Yes' was given when program asked to quit or not quit the program.
Expected Result	The program should repeat.
Actual Result	The program was repeated.
Conclusion	Test was successful.

Table 13: Table for test 13



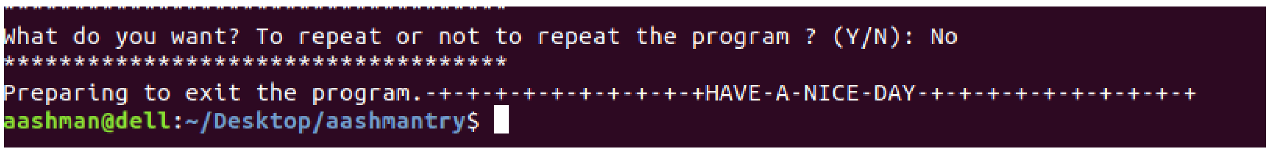
```
What do you want? To repeat or not to repeat the program ? (Y/N): Yes
************************************
+------------+-------+
|   Country    |  Codes |
+------------+-------+
|   Brazil     |  BRZ    |
|   Argentina |  ARG    |
|   Nepal      |  NEP    |
|   China      |  CHI    |
|   England    |  ENG    |
+------------+-------+
************************************
Please Enter the Country code:
```

Figure 15: Screenshot for test 13

1.3.14 TEST 14: TO REPEAT THE PROGRAM (NO)

Objective	To type yes while program asks to exit the program or not.
Action	'No' was given when program asked to quit or not quit the program.
Expected Result	The program should exit.
Actual Result	The program was exited.
Conclusion	Test was successful.

Table 14: Table for test 14



```
What do you want? To repeat or not to repeat the program ? (Y/N): No
************************************
Preparing to exit the program.-+-+-+-+-+-+-+-+-+-+HAVE-A-NICE-DAY-+-+-+-+-+-+-+-+-+-+
aashman@dell:~/Desktop/aashmantry$
```

Figure 16: Screenshot for test 14

1.4 Contents of three files

The contents of the three external files made for the script has been presented below.

1.4.1 KC

This is Kiran Chemjong.

From Nepal Football Team.

He is a goalkeeper for Nepal National Football Team.

1.4.2 LM

This is Lionel Messi.

From Argentina Football Player.

He plays as forward for Barcelona football club.

1.4.3 NJ

This is Neymar Junior.

From Brazilian Football Player.

He plays for Brazil National Team.

1.5 Conclusion

It was not an easy task to complete this coursework as it required lots of research on internet to clear out many confusions I had. The tutorials and lab session were so fruitful, and it has helped me a lot to make this script file. The completion of script file was fun along and I even became more familiar with bash shell scripting and UNIX. Researching on various articles about how certain commands work has made me understand better about the working mechanism of how program or scripts works.

I took help from different tutorials on Udemy, coursera, YouTube and with tutors too when I encountered with certain problems. Continuous hard work and managing the time in a well-mannered way helped me to complete this coursework in time. All in all, the completion of this coursework has given me better idea on how to write scripts in an interactive way.

2. Task B

2.1 Introduction

Memory management also known as memory allocation is the part of an Operating System that does the major tasks like controlling and coordinating the computer memory (Arne, et al., 2016). It also allocates different blocks to different programs that are running in order to optimize the overall system performance. In other words, memory allocation is referred as memory virtualization, protection, etc. (Arne, et al., 2016). We need a memory in our computer system that is fast, large and nonvolatile i.e. the contents of the memory do not get erased even if the computer system gets turned off. There are three different types of memory that can be used inside a computer system which are cache memory, RAM and HDD/SSD. The operating system has the ability or a feature called memory manager in it whose primary task is to manage the different tasks like allocating memory to certain processes, free up memory, keeping the trace of memory that is currently being used, etc. (Pal, 2020) Moreover, it is in the hand of an Operating System to not let the system go into deadlock and even if it goes on a deadlock, the Operating System has a feature to deal with that case too (Pal, 2020). Managing the program memory incorporates two related functions, called allocation and recycling. Memory management is all about allocation and optimization of finite physical resources. The vital task of memory management is to reallocate the memory spaces like the memory manager controls and supervises the processes that are being executed and needs to be executed and the memory manager gives and frees the memory accordingly (Pal, 2020). Likewise, protection, sharing, logical organization, physical organization are also the major tasks of memory manager.

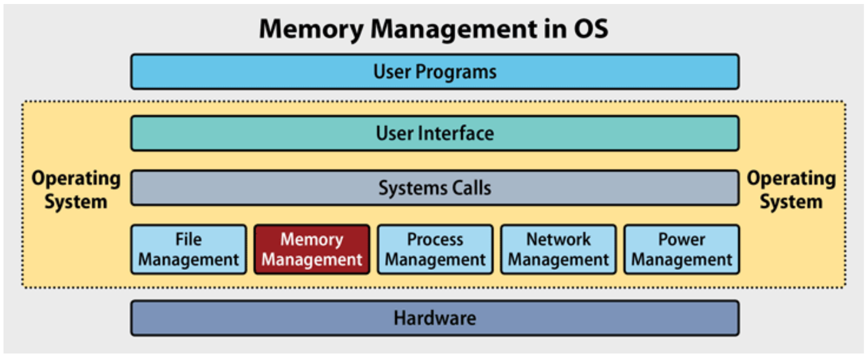


Figure 17: Memory Management

2.2 Aims and Objectives

- To be familiar about memory management.
- To know about physical memory.
- To know about storage allocation
- To know about paging and segmentation
- To know about page coloring.
- To research different articles about memory management in network and operating system.
- To get a general idea about how memory management in OS helps to make the tasks smooth and increase the performance of the system

2.3 Background

2.3.1 Physical Memory

Physical memory is also known as Random Access Memory which is in the motherboard of the computer system and it is directly accessible to the CPU. Physical memory has the information needed to execute a program like opening a game. It is called primary memory of the computer system because any program that needs to run should first be stored in it. The primary memory is very smart in nature as a process can be temporarily swapped out when the process is not in use and it can be brought back in order to continue the execution process. The main memory is divided into two sections i.e. one for the operating system and the other for the user processes. (includehelp, 2019)

2.3.2 Virtual Memory

Virtual memory refers to a layer of abstraction that sits on top of physical memory. Virtual memory is created by using secondary memory as physical memory. The definition of virtual memory is focused on memory allocation that is not contiguous. It has several advantages, including the option to use physical memory as a huge cache and the ability to keep different address spaces. From the perspective of the programmer, it lets physical memory look infinite. The virtual memory comes into use when the space of primary memory is full. (includehelp, 2019)

2.3.3 Order of Memory Management (Locality of reference)

A phenomenon known as locality of reference occurs when a computer program tends to access the same collection of memory locations over a period. In other words, it refers to the ability of a computer program to access commands with similar addresses. The locality of reference property is primarily seen in a program by loops and subroutine calls. The locality of reference is implemented to utilize the full benefit of cache memory in computer organization. It indicates that all the instructions referred by the processor are localized in nature. If the processor searches an instruction 'j', the probability is very high that after execution of instruction 'j', it will search for 'j+1'. That is a processor request instruction from memory which are residing in memory in nearby locations. (Singh, 2019)

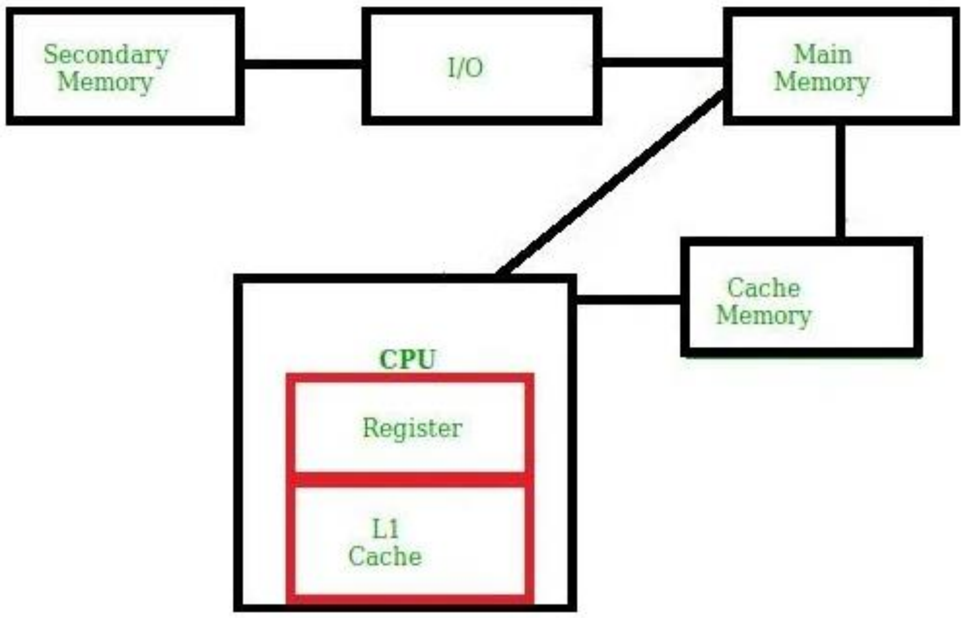


Figure 18: Locality of reference (Banger, 2020)

2.3.4 Memory Placement

The act of allocating physical or virtual memory address space to a process is known as memory allocation. Static and dynamic memory allocation are the two main types of memory allocation. Assigning memory space to a method is called memory allocation. Memory allocation is a general linking feature of the term. With the aid of an example let's understand binding. Suppose an object, with the collection of attributes, is in a system. Now, for certain set of attributes a component of this entity will have values. We must have memory allocated to these attributes to store those values. Thus, the act of allocating the memory address to the variable attribute is called memory allocation. And the act of binding / specifying the values to the variable attributes is called binding. This binding action must be performed before use of the variable during program execution. (T, 2019)

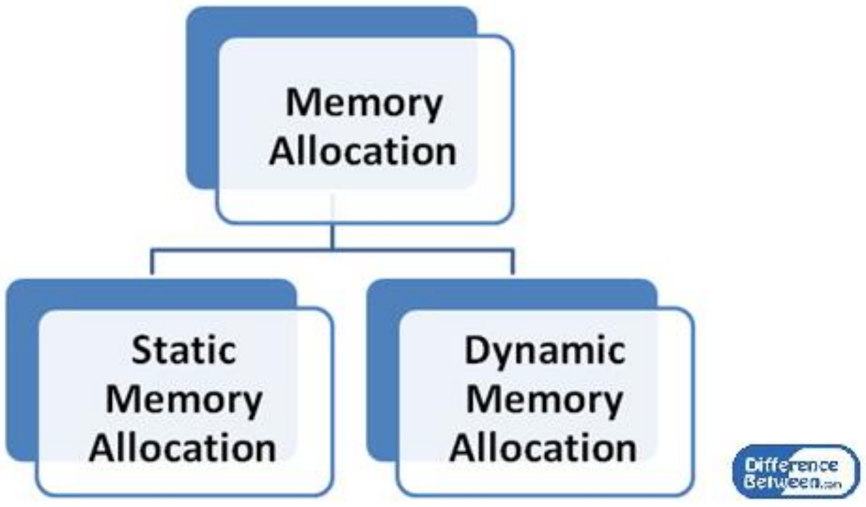


Figure 19: Types of memory allocation (Lithmee, 2018)

2.3.4.1 Static Memory Allocation

The reserved memory is fixed in static memory allocation. It is not possible to adjust the memory allocation after it has been made. There is no way to maximize or decrease memory. If a programmer writes int x in C, for example, this indicates that the variable will store an integer value. The number of bytes used is determined by the machine. (Lithmee, 2018)

2.3.4.2 Dynamic Memory Allocation

It is also important to adjust the memory capacity. As a result, dynamic memory allocation is possible. The memory can expand or shrink depending on the insertions and deletions of data items. Dynamic memory allocation is the term for this. The biggest benefit of dynamic memory allocation is that it helps you conserve space. The programmer can assign, and release memory as required. Memory can be reallocated during execution and released when no longer needed. Static memory allocation is inefficient sometimes so in that case, dynamic memory allocation is more effective. One downside is that dynamic memory allocation is difficult to enforce. (Lithmee, 2018)

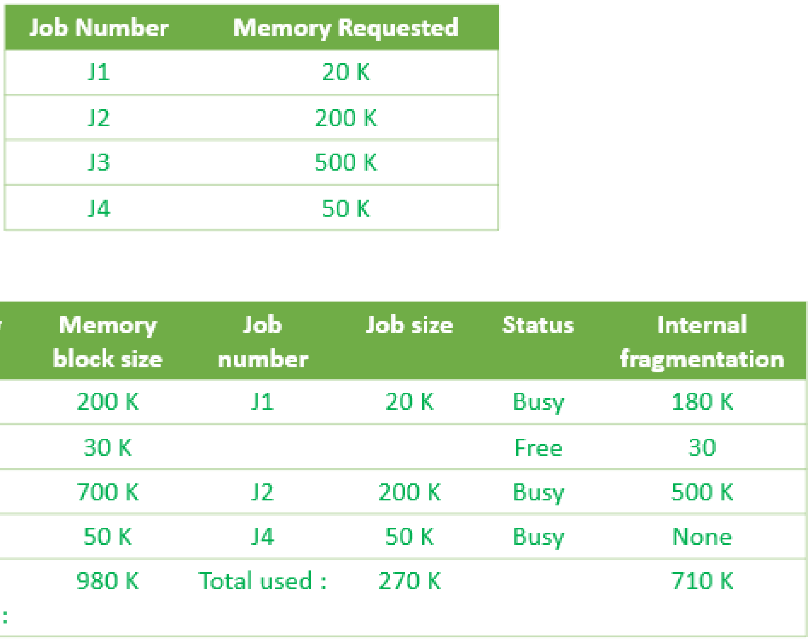
The memory is located above the data segment's static portion. Programs will both request and return memory that was previously dynamically reserved. When memory is no longer needed, it can be restored. Memory can be returned in any order, regardless of how it was initially allocated. Where previously allocated memory has been returned between blocks of memory still in use, the heap can grow "holes." A new dynamic memory request could result in a set of addresses being returned from one of the holes. Since the cumulative memory used by the holes can be high, but the holes cannot be used to accommodate dynamic demands, memory is lost if there are too many small holes. Memory fragmentation is the term for this case. It's difficult to keep track of memory that has been reserved and de-allocated. All of this is handled by a modern operating system.

When it's time to deploy a process into primary memory, the OS chooses which free block of memory to assign if there are several free blocks of appropriate size (GeeksforGeeks, 2020). There are many algorithms for the selection of memory namely first fit, best fit, worst fit, next fit, etc.

2.3.4.2.1 First Fit

The First Fit algorithm searches the related list, stopping when it detects the first large enough space to store a process and loading it into that hole and as a result of this, two partitions are created (javatpoint, 2018). In these two partitions, one becomes hole whereas the other stores any processes.

The connected list is maintained by the First Fit algorithm in increasing order of starting index. This algorithm is the easiest to apply of all the algorithms and creates larger holes than the others. (javatpoint, 2018)



Job Number	Memory Requested
J1	20 K
J2	200 K
J3	500 K
J4	50 K

Memory location	Memory block size	Job number	Job size	Status	Internal fragmentation
10567	200 K	J1	20 K	Busy	180 K
30457	30 K			Free	30
300875	700 K	J2	200 K	Busy	500 K
809567	50 K	J4	50 K	Busy	None
Total available :	980 K	Total used :	270 K		710 K

Figure 20: First Fit Memory Allocation (geeksforgeeks, 2020)

2.3.4.2.2 Best Fit

The operating system looks for an empty memory block by using the best suited memory allocation scheme (prepinsta, 2019). The operating system allocates memory to the process as it detects the memory block with the least amount of memory waste (prepinsta, 2019). This method is thought to be the safest because it leads in one of the most efficient memory allocation but its main drawback is its time consuming (prepinsta, 2019).

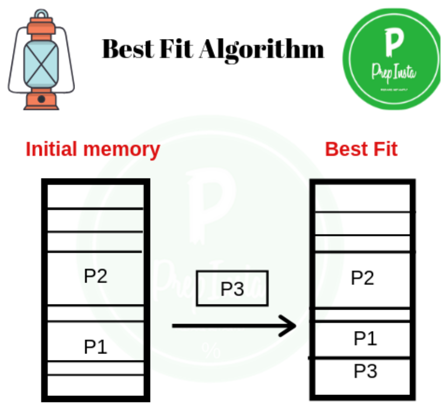


Figure 21: Best Fit Memory Allocation (prepinsta, 2019)

2.3.4.2.3 Worst Fit

Any time the worst fit algorithm runs, it searches the whole list and attempts to find the largest hole in the list that can satisfy the process's requirements (javatpoint, 2018). Despite the fact that this algorithm creates wider gaps for other processes to load, it is not the best solution because it is slower as it scans the whole list repeatedly (javatpoint, 2018).

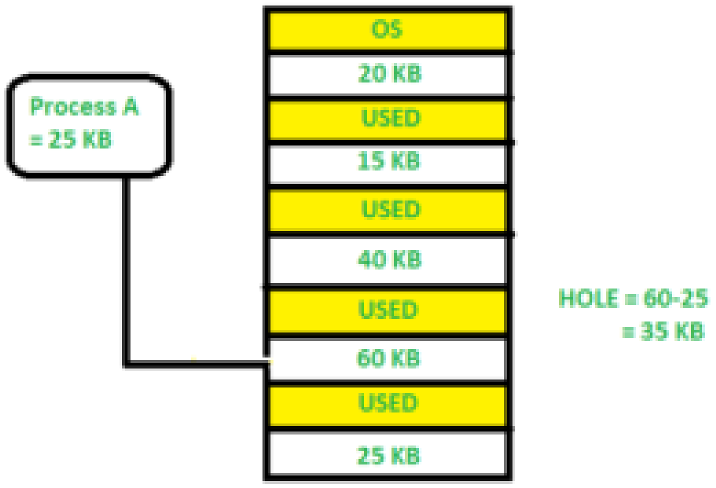


Figure 22: Worst Fit Memory Allocation (geeksforgeeks, 2020)

2.3.5 Page Coloring

Page coloring is a speed enhancement that makes the most of the processor cache when accessing contiguous pages in virtual memory (freebsd, 2018). CPU caches used to map virtual memory rather than physical memory in the past. This resulted in a slew of issues, including the need to clear the cache on any context swap in some situations and data aliasing issues in the cache (freebsd, 2018). Modern processor caches specifically map physical memory to overcome these issues which indicates that multiple adjacent pages in a process's address space might not equate to two adjacent pages in the cache (freebsd, 2018). The page coloring code would not allocate page 20 of physical memory to page 1 of a process's virtual memory if page 16 of physical memory is allocated to page 0 of a process's virtual memory and the cache will accommodate 4 pages (freebsd, 2018).

2.3.6 Paging and Segmentation

Paging is a technique to manage memory. Paging allows a procedure to be saved in a non-contiguous way in memory and the problem of external fragmentation is solved by storing processes in a non-contiguous way (TechDifferences, 2019). The physical and logical memory spaces are split into the same fixed-size blocks to enforce paging (TechDifferences, 2019). Frames are the fixed-sized blocks of physical memory, and pages are the fixed-sized blocks of logical memory (TechDifferences, 2019). Process pages from logical memory space are inserted into the frames of physical memory address space when a process has to be run (TechDifferences, 2019). The page number and page offset are now separated in the address provided by the CPU for accessing the frame (TechDifferences, 2019). Page number is used as an index in the page table; each method has its own page table that maps the logical address to the physical address (TechDifferences, 2019). The base address of the page contained in the frame of physical memory space is stored in the page table (TechDifferences, 2019). The frame number in physical memory where the page is stored is determined by combining the base address identified by the page table with the page offset (TechDifferences, 2019).

Like paging, segmentation is a non-contiguous memory allocation scheme. Segmentation, like paging, does not break processes indiscriminately into mounted(fixed) size pages (GeeksforGeeks, 2020). It's a theme of variable partitioning sizes. Segmentation, like paging, does not split secondary and primary memory into equal-sized partitions (GeeksforGeeks, 2020). Segments are the partitions of the secondary memory area network (GeeksforGeeks, 2020). The data for each section are held in a table called the segmentation table (GeeksforGeeks, 2020). The Segment table includes two key pieces of information about segments: Base, which is the segment's bottom address, and Limit, which is the segment's length (GeeksforGeeks, 2020). So, we can conclude that internal fragmentation is caused by paging whereas external fragmentation is caused by segmentation.

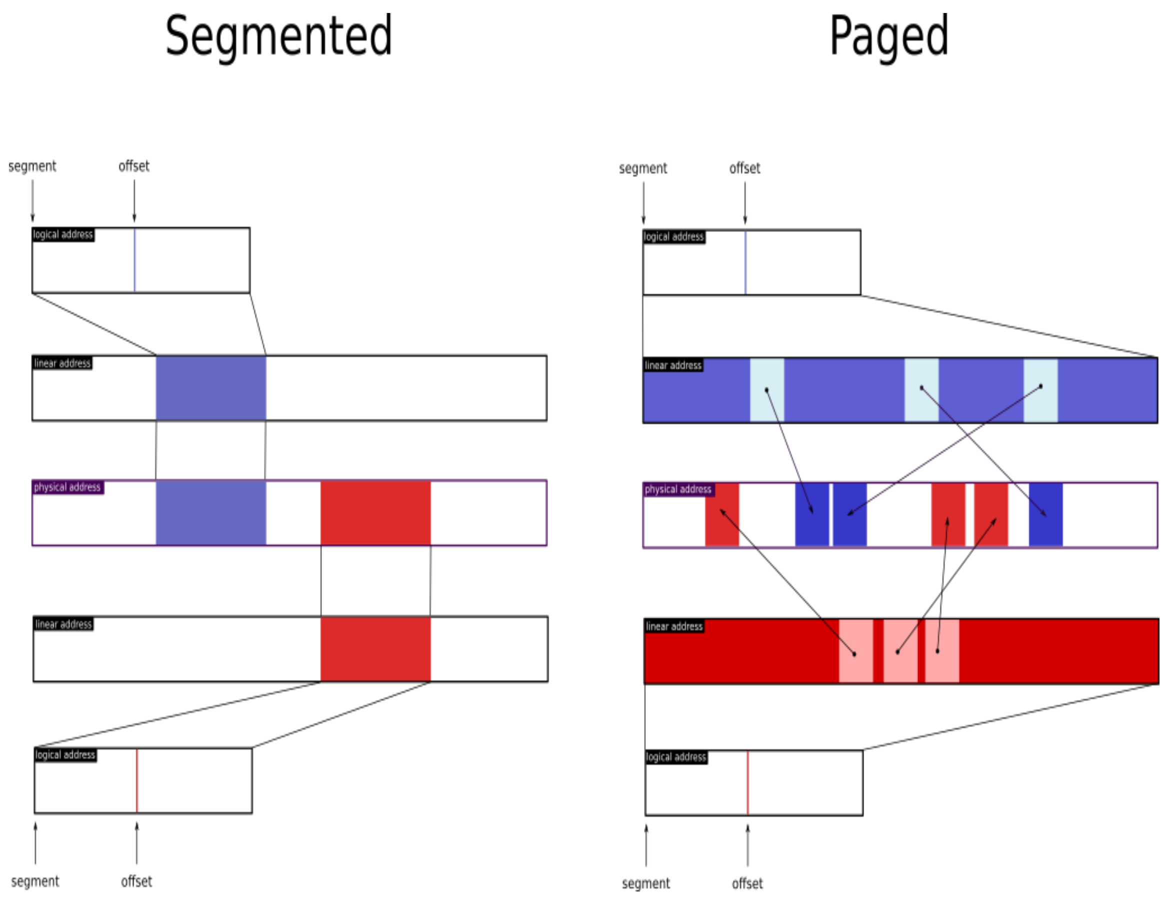


Figure 23: Paging and Segmentation (stackoverflow, 2017)

2.4 Conclusion

The task completed was the ultimate result of continuous hard work, effort, research, proper guidance from tutors, etc. In task B of the report, we discussed about memory management and writing this report has helped me to know how memory management takes place in any OS. The terminologies in memory management were a completely new topic so it took some time for me to know about the basics of it. The completion of both the tasks have developed my skills in researching and finding information from the web. All in all, in the combined report we discussed about memory management and bash shell scripting.

3. References

Arne, S., Domenico, S., Elio, S. & Roberto, D. C., 2016. *Empowering Network Operating Systems with Memory Management Techniques.* UK: inproceedings.

Banger, R. S., 2020. *Cache Memory* **|** *Locality of Reference in Computer Organization and OS.* [Online]
Available at: https://digitalthinkerhelp.com/cache-memory-and-locality-of-reference-in-computer-organization-os/
[Accessed 10 April 2021].

freebsd, 2018. *Page Coloring.* [Online]
Available at: https://docs.freebsd.org/en/articles/vm-design/page-coloring-optimizations.html
[Accessed 10 April 2021].

GeeksforGeeks, 2020. *Difference Between Paging and Segmentation.* [Online]
Available at: https://www.geeksforgeeks.org/difference-between-paging-and-segmentation/
[Accessed 11 April 2021].

geeksforgeeks, 2020. *First-Fit Allocation in Operating Systems.* [Online]
Available at: https://media.geeksforgeeks.org/wp-content/uploads/20191012233716/first.odg.png
[Accessed 10 April 2021].

geeksforgeeks, 2020. *Partition Allocation Methods in Memory Management.* [Online]
Available at: https://media.geeksforgeeks.org/wp-content/uploads/20200524132634/WORST-FIT-300x225.png
[Accessed 10 April 2021].

GeeksforGeeks, 2020. *Partition Allocation Methods in Memory Management.* [Online]
Available at: https://www.geeksforgeeks.org/partition-allocation-methods-in-memory-management/
[Accessed 10 April 2021].

Hiwarale, U., 2019. *Bash Scripting: Everything you need to know about Bash-shell programming.* [Online]
Available at: https://medium.com/sysf/bash-scripting-everything-you-need-to-know-about-bash-shell-programming-cd08595f2fba

includehelp, 2019. *Concept of physical and virtual memory in Operating System.* [Online]
Available at: https://www.includehelp.com/operating-systems/concept-of-physical-and-virtual-memory.aspx
[Accessed 10 04 2021].

javatpoint, 2018. *Partitioning Algorithms.* [Online]
Available at: https://www.javatpoint.com/os-partitioning-algorithms
[Accessed 10 April 2021].

Lithmee, 2018. *Difference Between Static and Dynamic Memory Allocation.* [Online]
Available at: https://www.differencebetween.com/difference-between-static-and-vs-

dynamic-memory-allocation/
[Accessed 10 April 2021].
Pal, T., 2020. *Memory Management in Operating Systems – Simple Explanation.* [Online] Available at: https://technobyte.org/memory-management-os-simple-explanation/
[Accessed 10 April 2021].
prepinsta, 2019. *Best Fit Algorithm in Operating System (OS)* **|** *Program in C.* [Online] Available at: https://prepinsta.com/operating-systems/page-replacement-algorithms/best-fit/
[Accessed 10 April 2021].
Singh, B., 2019. *Locality of Reference and Cache Operation in Cache Memory.* [Online] Available at: https://www.geeksforgeeks.org/locality-of-reference-and-cache-operation-in-cache-memory/
[Accessed 10 April 2021].
stackoverflow, 2017. *emory segmentation, heap and mmap.* [Online] Available at: https://i.stack.imgur.com/01Q9Z.png
[Accessed 10 April 2021].
TechDifferences, 2019. *Difference Between Paging and Segmentation in OS.* [Online] Available at: https://techdifferences.com/difference-between-paging-and-segmentation-in-os.html#:~:text=Paging%20and%20segmentation%20both%20are%20the%20memory%20management,internal%20fragmentation%20the%20segmentation%20leads%20to%20external%20fragmentation.
[Accessed 10 April 2021].
T, N., 2019. *Memory Allocation.* [Online] Available at: https://binaryterms.com/static-and-dynamic-memory-allocation.html
[Accessed 10 April 2021].