LONDON METROPOLITAN UNIVERSITY

ITAHARI INTERNATIONAL COLLEGE

Module Code & Module Title

CS5001NT Network and Operating System

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2021 Spring

Student Name: Susan Shrestha

London Met ID: 19033540

College ID: NP05CP4S200039

Assignment Due Date: 11 April 2021

Assignment Submission Date: 11 April 2021

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

TASK A

INTRODUCTION

The main goal of this part of coursework is to writing a shell script that demonstrate the ability to organize interactive communication of its potential user with the UNIX environment in a friendly manner.

This particular report is prepared and structured after implementing the scenario and writing the script of the program where implementation follows usage of if then else condition, loops such as while, case and select condition. While commenting and naming convention is done for good code as well as providing the clear error messages for the user to understand. Next, testing is performed to validate and verify the working of script and screenshots are attached for the proof of testing and execution of the program.

Shell scripting is an open-source computer program that is designed to be run by the Linux or UNIX shell where a series of commands is written that are executed by the shell and hence reducing programming efforts. (guru99, n.d.)

In UNIX, a shell is an interface between user and an OS service that accepts human readable commands and executes those commands which runs automatically. There are two major components in an Operating system are **Kernel** and **Shell**. Where, a kernel lies at the nucleus of a computer system and makes the communication possible between hardware and software. Shell takes an input from user in the form of commands and process it and then produces result or output. A shell is accessed by the terminal which runs it. (guru99, n.d.)

There are two main shell in UNIX. They are,

1. **Bourne shell:** $ is the prompt for Bourne shell. The derivatives of Bourne shell are listed as,

- POSIX shell (sh)
- Korn Shell (sh)

- And the most popular BASH (Bourne Again Shell)

2. **C shell: %** is the prompt for C shell. Its categories are,
- C shell (csh)
- Tops C shell (tcsh)

As bash is the most popular and used shell, for this coursework bash shell is used for scripting that implements all the given scenarios.

SCRIPT

The scripting file for the program is written in bash shell in UNIX. The file is named as 19033540.sh included in the CW PART 2 folder along with the report documentation file.

```
1. #!/bin/bash
2. ##author: Susan Shrestha##
3. ##Date: April 1 2021##
4.
5.  ##displaying content about three selected players if data is
   found##
6. threePlayers() {
7.    PS3="Select a number:" ##using prompt string to make
   selection of player##
8.    select p in $p1 $p2 $p3 ##storing selection in variable
   p##
9.    do
10.               case $p in
11.               "LM")
12.               echo "#-----------------------------------
   ---------------------------------#"
13.               cat LM.txt ##displaying content for player
   using cat utility##
14.               echo "#-----------------------------------
   ---------------------------------#"
15.               echo -e "\n"
16.               break;;
17.               "NJ")
18.               echo "#-----------------------------------
   ---------------------------------#"
19.               cat NJ.txt
```

```
20.   "#-------------------------------------------------------
  -----------------#"
21.                 echo -e "\n"
22.                 break;;
23.                 "KC")
24.                 echo "#-------------------------------------
  ----------------------------------#"
25.                 cat KC.txt
26.                 echo "#-------------------------------------
  ----------------------------------#"
27.                 echo -e "\n"
28.                 break;;
29.                 "ZZ") echo "#-------------------------------
  ----------------------------------------#
30.         Error! Cannot fetch any data for Zheng Zhi!"
31.
32.                 echo "#-------------------------------------
  ----------------------------------#"
33.                 main
34.                 echo -e "\n"
35.                 break;;
36.                 "HK") echo "#-------------------------------
  ----------------------------------------#
37.         Error! Cannot fetch any data for Henry Kane!"
38.         echo "#-----------------------------------------------
  ----------------------------#"
39.                 main
40.                 echo -e "\n"
41.                 break;;
42.                 *) echo "#-----------------------------------
  ----------------------------------------#
43.         Error! Type the correct number."
44.                 echo "#-------------------------------------
  ----------------------------------#" ##printing error
  message if wrong selection is done##
45.                 echo -n ">> "
46.                 esac
47.         done
48.
49.         while true;
50.         do
51.                 echo "Continue? Yes [Y/y] or No [N/n]"
52.                 echo -n ">> " ##asking user to continue or
  exit the program##
53.                 read choice
54.                 case $choice in
```

```
55.                     [yY]* ) ##if y/Y is entered program
   continues##
56.                     main;; ##calls main function##
57.                     [nN]* ) ##if n/N is entered program
   exits##
58.                     exit;; ##exits program##
59.                esac
60.        done
61.   }
62.
63.   ##reading 3 player code as input and validating the player
   code input##
64.   bestPlayer (){
65.        flag=0
66.        while [ $flag = 0 ]
67.        do
68.                read p1 p2 p3
69.                if [ -z "$p1" ] || [ -z "$p2" ] || [ -z "$p3"
   ];
70.                then
71.                        echo "#-------------------------------
   ----------------------------------------#
72.        ERROR! Please enter 3 players code seperated by
   space"
73.                echo -e "#-----------------------------------
   ----------------------------------------#"
74.                echo -n ">> "
75.                elif [[ "LM" != "$p1" && "NJ" != "$p1" &&
   "KC" != "$p1" &&                    "ZZ" != "$p1" && "HK" !=
   "$p1" ]] || [[ "LM" != "$p2" &&                "NJ" != "$p2"
   && "KC" != "$p2" && "ZZ" != "$p2" && "HK" !=
      "$p2" ]] || [[ "LM" != "$p3" && "NJ" != "$p3" && "KC" !=
            "$p3" && "ZZ" != "$p3" && "HK" != "$p3" ]]
   ##validating the player codes if codes are not listed##
76.                then
77.                        echo "#-------------------------------
   ----------------------------------------#
78.        ERROR! Please enter correct players code!"
79.                echo "#-------------------------------------
   ----------------------------------#"
80.                echo -n ">> "
81.                elif [ $p1 == $p2 ] || [ $p3 == $p2 ] || [
   $p1 == $p3 ]; ##checking if player codes are redundant##
82.                then
83.                        echo "#-------------------------------
   ----------------------------------------#
84.        ERROR! Please enter 3 different players code!"
```

```
85.         echo "#-----------------------------------------
   ----------------------------#"
86.                 echo -n ">> "
87.                 else
88.                       threePlayers ##calls threePlayers
   funtion after validation##
89.                       flag=1 ##sets flag to 1 to end loop##
90.                 fi
91.         done
92.   }
93.
94.   bestTeam (){
95.         read team
96.         if [ "$team" == "NEP" ]
97.         then
98.                 echo -e "
99.   #-------------------------------------------------------
   -------------------#
100.        ~YES, YOUR GUESS IS CORRECT!
101.        ~Nepal is the best team in the world.
102.        ~Nepal has won Saff U-19 championship in 2015
103.        ~3 major tournaments in 2016 and gold medal in 2016
   south Asian games.
104. #-------------------------------------------------------
   -----------------#"
105.        else
106.                echo "#-------------------------------------
   -----------------------------------#
107.                Error! Wrong guess. Please guess again!"
108.                echo "#-------------------------------------
   -----------------------------------#"
109.                echo -n ">> "
110.                bestTeam
111.        fi
112. }
113.
114. main(){
115.        echo "
116.        **GUESS THE BEST FOOTBALL TEAM**
117.           **TYPE THE COUNTRY CODE**
118.             *------------------*
119.             | CODE | (COUNTRY)  |
120.             *------------------*
121.             | ARG    (Argentina)|
122.             | BRZ    (BRAZIL)   |
123.             | NEP    (Nepal)    |
124.             | CHI    (China)    |
```

```
125.             | ENG    (England)  |
126.             *------------------*" #asking user to guess the
  best football team#
127.        echo -n ">> "
128.        bestTeam
129.        echo "
130.        **THE BEST FIVE STAR PLAYERS ARE LISTED**
131.           **TYPE ANY THREE PLAYER'S CODE**
132.             *----------------------*
133.             | CODE | (PLAYER)       |
134.             *----------------------*
135.             | LM    (Lionel Messi)  |
136.             | NJ    (Neymar Junior) |
137.             | KC    (Kiran Chemjong)|
138.             | ZZ    (Zheng Zhi)     |
139.             | HK    (Henry Kane)    |
140.             *----------------------* "
141.        echo -n ">> "
142.        bestPlayer
143.        exit
144. }
145.
146. #program function: from where the user logs in to the
  program and plays on#
147. program (){
148.        echo "
149. #-------------------------------------------------------
  ----------------#
150.    ##       ## ######## ##         ######   #######  ##
  ## ########
151.    ##  ##  ## ##       ##        ##    ## ##     ## ###
  ### ##
152.    ##  ##  ## ##       ##        ##       ##     ## ####
  #### ##
153.    ##  ##  ## ######   ##        ##       ##     ## ## ###
  ## ######
154.    ##  ##  ## ##       ##        ##       ##     ## ##
  ## ##
155.    ##  ##  ## ##       ##        ##    ## ##     ## ##
  ## ##
156.     ###  ###  ######## ########  ######   #######  ##
  ## ########
157. #-------------------------------------------------------
  ----------------#"
158.        echo "
159. #-------------------------------------------------------
  ----------------#
```

```
160.        ~Name:$1            ~ID:$2
161.
162.        ~Date:$(date)
163. #-------------------------------------------------------
  ----------------#
164. " #printing the username ID and Date#
165.        main ##calls main function##
166. }
167.
168. #login function for getting into the program by typing the
  secret key#
169. login (){
170.        chances=5 ##initiating chances as 5##
171.        while [ $chances -gt 0  ] ##while loop for login
  function##
172.        do
173.              echo "Enter the secret key:"
174.              echo -n ">> "
175.              stty -echo ##hiding the secret key when user
  types##
176.              read key
177.              stty echo ##unhiding user keyboard input##
178.              if [ "$key" == "susz" ] ##checks if the key
  matches##
179.              then
180.                    program $1 $2 ##if key matches goto
  program function and pass 1st and 2nd argument as parameter##
181.              else
182.                    ((chances--)) ##if key doesnot match
  return error message and decrement chance and print##
183.                    echo -e ""
184.                    echo "#-------------------------------
  ---------------------------------------#
185.        Please enter the correct secret key! $chances chance
  left!"
186. echo "#-------------------------------------------------
  ----------------------#"
187.        echo -n ">> "
188.              fi
189.        done
190. }
191.
192. if [ $# == 2 ] #checks if the script file is initiated
  with two parameters as name and id
193. then
194.        login $1 $2  #calls login function passing the two
  parameter values#
```

```
195. elif [ "$1" = "" ] ##checks if first parameter is missing
  then prints error message##
196. then
197.        echo "#-----------------------------------------------
  ----------------------------#
198.        ERROR! Please enter your name!" #if the program is
  not initiated with two parameter value as name and id returns
  error to the user and exit$
199. echo "#-------------------------------------------------
  ----------------------#"
200. else
201.        echo "#-----------------------------------------------
  ----------------------------#
202.        ERROR! Try again with your Name and ID as
  parameters."
203.        echo "#-----------------------------------------------
  ----------------------------#"
204. fi
```

TESTING

Test 1: run without name

Test No.	**1**
Input	bash coursework.sh
Expected Output	Prints error message and exits.
Actual Output	ERROR! Please enter your name!
Test Result	Test OK

Table 1: Testing no. 1

Screenshot:

```
>> susz@susz-VirtualBox:~$ bash coursework.sh
#--------------------------------------------------------------------------#
        ERROR! Please enter your name!
#--------------------------------------------------------------------------#
>> susz@susz-VirtualBox:~$
```

Figure 1: Testing no. 1

Test 2: run with more than 2 arguments

Test No.	**2**
Input	bash coursework.sh Susan 19 itahari
Expected Output	Prints error message and exits.
Actual Output	ERROR! Try again with your Name and ID as parameters.
Test Result	Test OK

Table 2: Testing no. 2

Screenshot:

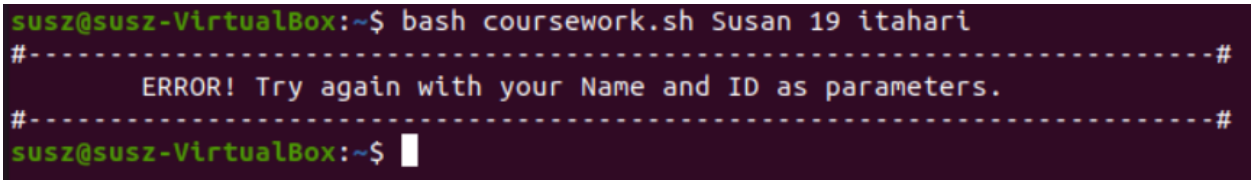

Figure 2: Testing no. 2

Test 3: run with username and ID

Test No.	**3**
Input	bash coursework.sh Susan 19
Expected Output	Asks user to enter the secret key
Actual Output	Enter the secret key
Test Result	Test OK

Table 3: Testing no. 3

Screenshot:

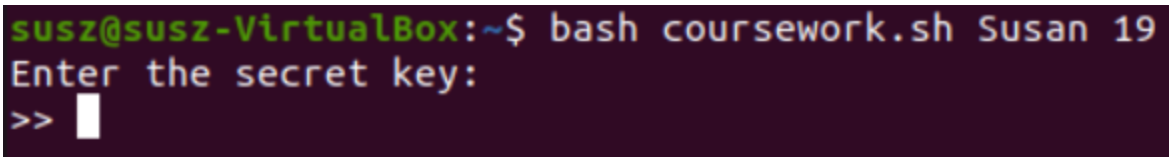

Figure 3: Testing no. 3

Test 4: run incorrect password 5 times

Test No.	**4**
Input	Entering incorrect password each time until the program exits
Expected Output	Prints error message until user enters the correct password for 4 next time after the first incorrect password. In total 5 chances.
Actual Output	Prints error message until user enters the correct password for 4 next time.
Test Result	Test OK

Table 4: Testing no. 4

Screenshot:

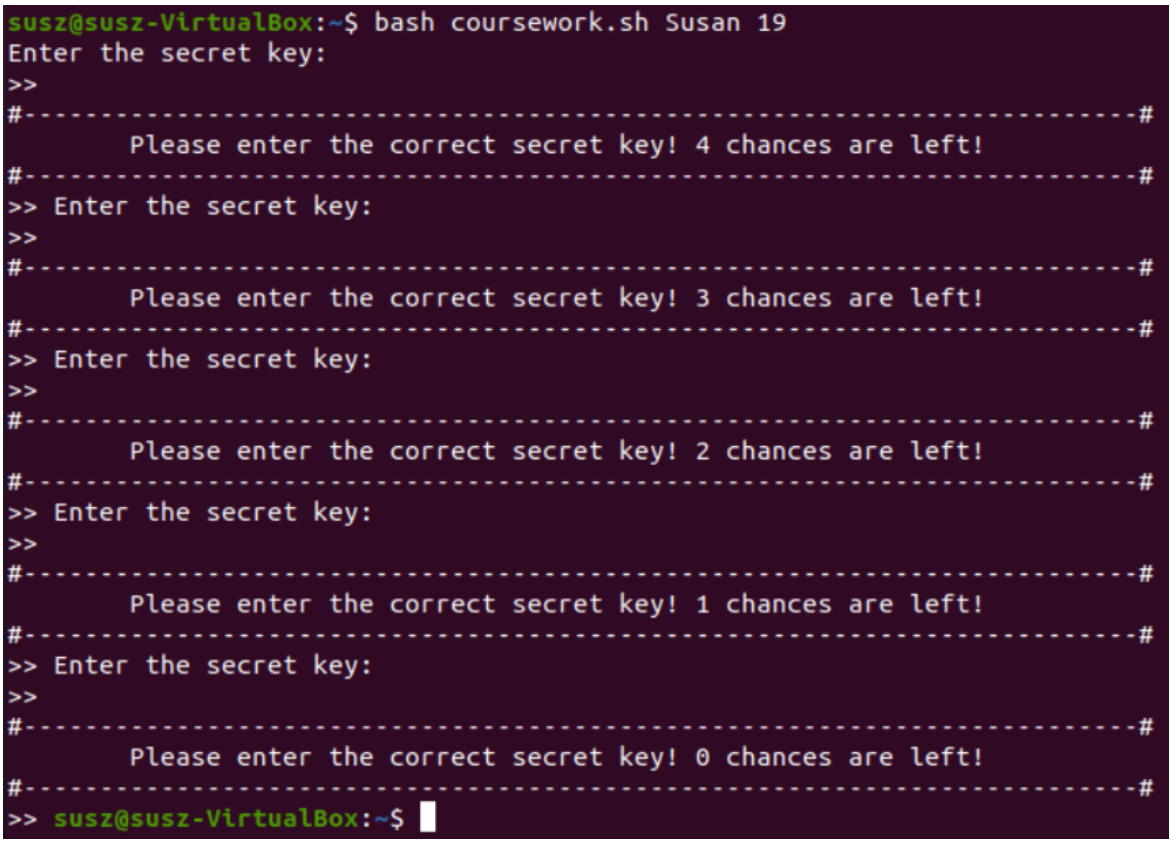

Figure 4: Testing no. 4

Test 5: run correct password

Test No.	**5**
Input	**susz** (password/secret key of program)
Expected Output	Program welcomes the user with user name, ID and date
Actual Output	Welcome screen with user Name, ID and Date
Test Result	Test OK

Table 5: Testing no. 5

Screenshot:

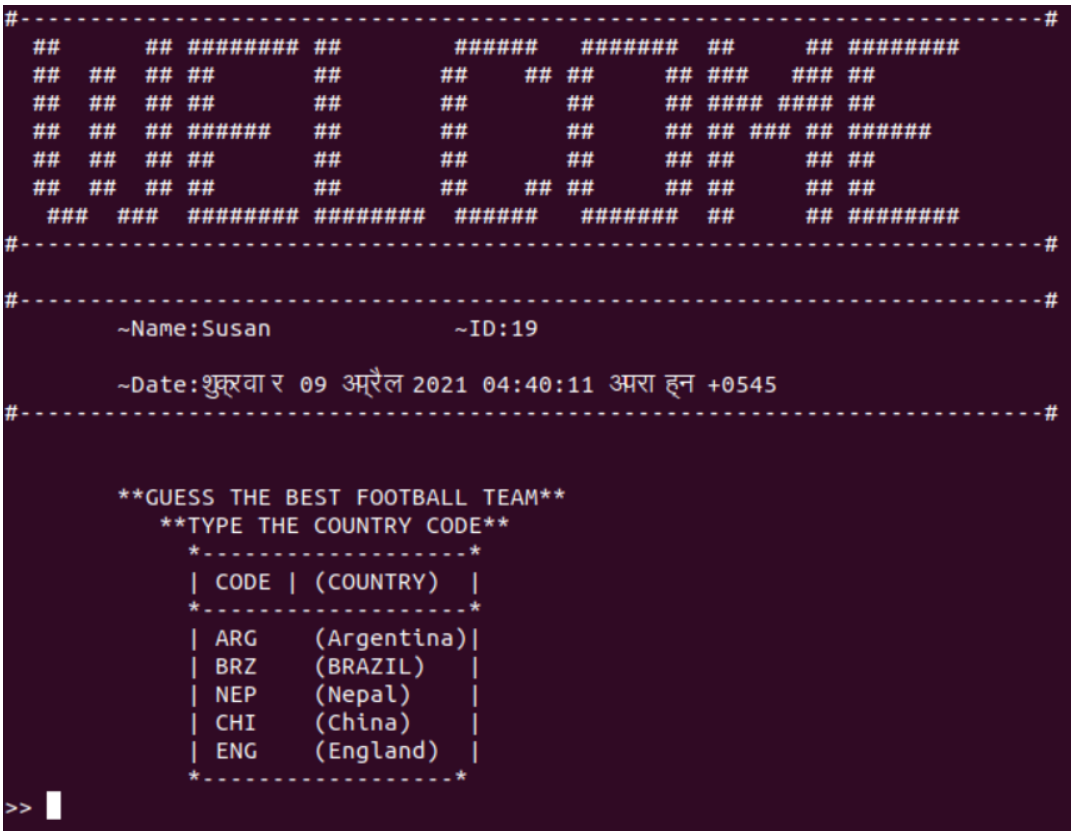

Figure 5: Testing no. 5

Test 6: enter country name

Test No.	**6**
Input	**Nepal**
Expected Output	Prints error message and ask user to guess again
Actual Output	Error! Wrong guess. Please guess again!
Test Result	Test OK

Table 6:Testing no. 6

Screenshot:

Figure 6: Testing no. 6

Test 7: incorrect country CODE

Test No.	**7**
Input	**IND**
Expected Output	Prints error message and ask user to guess again
Actual Output	Error! Wrong guess. Please guess again!
Test Result	Test OK

Table 7: Testing no. 7

Screenshot:

Figure 7: Testing no. 7

Test 8: correct country CODE but wrong selection

Test No.	**8**
Input	**ARG**
Expected Output	Prints error message and ask user to guess again
Actual Output	Error! Wrong guess. Please guess again!
Test Result	Test OK

Table 8: Testing no. 8

Screenshot:

Figure 8: Testing no. 8

Test 9: correct country CODE

Test No.	**9**
Input	**NEP**
Expected Output	Prints Guess Correct message and ask user to enter any 3 players code from listed five players
Actual Output	Guess correct message box and some information about Team Nepal. Next asks user to enter any 3 players code from listed five players
Test Result	Test OK

Table 9: Testing no. 9

Screenshot:

Figure 9: Testing no. 9

Test 10: pick 4 player code

Test No.	**10**
Input	**LM NJ KC HK**
Expected Output	Prints error message and ask user to enter again
Actual Output	Error! Please enter correct players code!
Test Result	Test OK

Table 10: Testing no. 10

Screenshot:

Figure 10: Testing no. 10

Test 11: pick same player code

Test No.	**11**
Input	**LM LM LM / LM LM NJ**
Expected Output	Prints error message and ask user to enter 3 different players code
Actual Output	Error! Please enter 3 different players code!
Test Result	Test OK

Table 11: Testing no. 11

Screenshot:

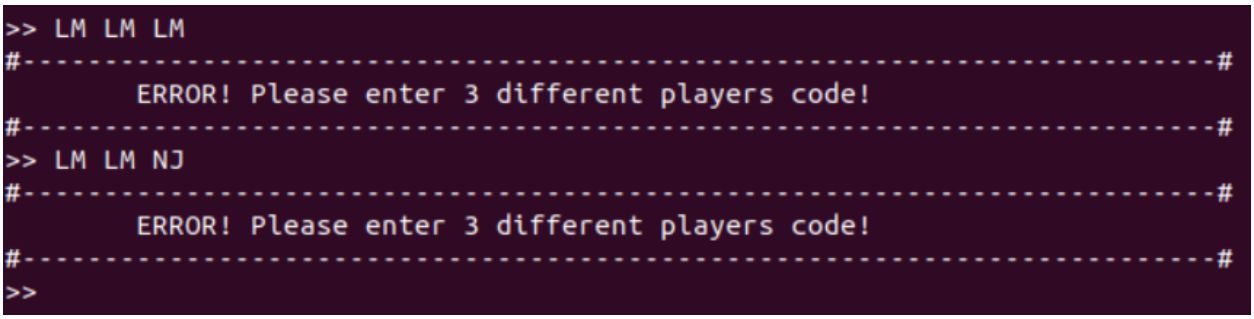

Figure 11: Testing no. 11

Test 12: wrong player code

Test No.	**12**
Input	**LL NH CR**
Expected Output	Prints error message and ask user to enter players code
Actual Output	Error! Please enter correct players code!
Test Result	Test OK

Table 12: Testing no. 12

Screenshot:

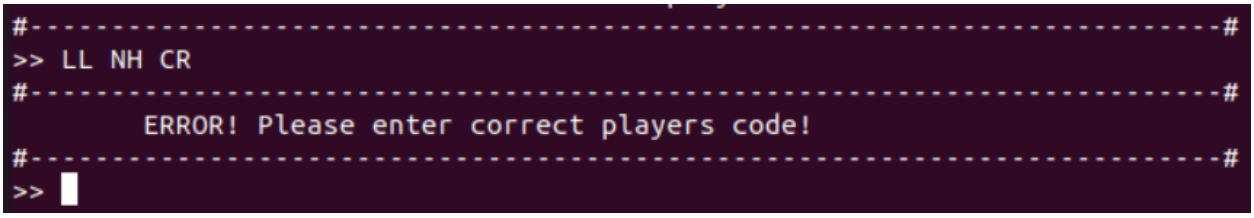

Figure 12: Testing no. 12

Test 13: different player code

Test No.	**13**
Input	**LM NJ ZZ**
Expected Output	Shows 3 selected player codes list and ask user to select corresponding number for listed codes
Actual Output	1) LM 2) NJ 3) ZZ Select a number:
Test Result	Test OK

Table 13: Testing no. 13

Screenshot:

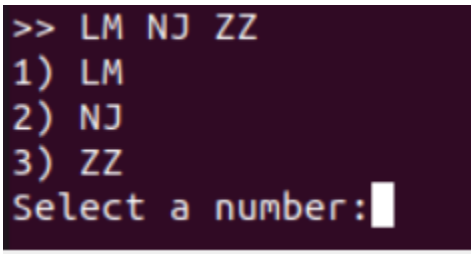

Figure 13: Testing no. 13

Test 14: wrong user selection

Test No.	**14**
Input	**4 / asdf**

Expected Output	Prints error message and ask user to select a number again
Actual Output	Error! Type the correct number.
Test Result	Test OK

Table 14: Testing no. 14

Screenshot:

```
>> LM NJ ZZ
1) LM
2) NJ
3) ZZ
Select a number:4
#------------------------------------------------------------------------#
        Error! Type the correct number.
#------------------------------------------------------------------------#
>> Select a number:asdf
#------------------------------------------------------------------------#
        Error! Type the correct number.
#------------------------------------------------------------------------#
>> Select a number:
```

Figure 14: Testing no. 14

Test 15: selection of player with no data

Test No.	**15**
Input	**3**
Expected Output	Prints error message box and start the program from guessing the best football team country
Actual Output	Error! Cannot fetch any data for Zheng Zhi!

Test Result	Test OK

Table 15: Testing no. 15

Screenshot:

Figure 15: Testing no 15

Test 16: right user selection

Test No.	16
Input	**1**
Expected Output	Reads the player data and displays as a message box containing information about selected player
Actual Output	Message box with information of Lionel Messi
Test Result	Test OK

Table 16: Testing no. 16

Screenshot:

```
>> LM NJ ZZ
1) LM
2) NJ
3) ZZ
Select a number:1
#-------------------------------------------------------------------------#
Luis Lionel Andres (“Leo”) Messi is an Argentinian soccer player who plays forw
ard for the FC Barcelona club and the Argentina national team. At the age of 13
, Messi moved from Argentina to Spain after FC Barcelona agreed to pay for his
medical treatments.

There he earned renown as one of the greatest players in history, helping his c
lub win more than two dozen league titles and tournaments. In 2012, he set a re
cord for most goals in a calendar year, and in 2019, he was named Europe's Ball
on d'Or winner for the sixth time.
#-------------------------------------------------------------------------#


Continue? Yes [Y/y] or No [N/n]
>>
```

Figure 16: Testing no. 16

Test 17: Continue (Yes)

Test No.	17
Input	**Y / y / yes**
Expected Output	Continues the program from Guessing the best football team country
Actual Output	Program continues by asking user to guess the best football team
Test Result	Test OK

Table 17: Testing no. 17

Screenshot:

Figure 17: Testing no. 17

Test 18: Continue (No)

Test No.	18
Input	**N / n / No**
Expected Output	Exits the program
Actual Output	Program exits.
Test Result	Test OK

Table 18: Testing no. 18

Screenshot:

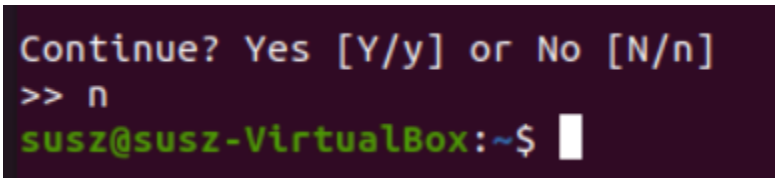

Figure 18: Testing no. 18

CONTENT OF THREE FILES (TEXT)

- **LM**

  Luis Lionel Andres (“Leo”) Messi is an Argentinian soccer player who plays forward for the FC Barcelona club and the Argentina national team. At the age of 13, Messi moved from Argentina to Spain after FC Barcelona agreed to pay for his medical treatments.

  There he earned renown as one of the greatest players in history, helping his club win more than two dozen league titles and tournaments. In 2012, he set a record for most goals in a calendar year, and in 2019, he was named Europe's Ballon d'Or winner for the sixth time.

  (*source:https://www.biographyonline.net/sport/football/lionel-messi.html#:~:text=Lionel%20Messi%20was%20born%2C%2024,growth%20hormone%20deficiency%20(GHD).)*

- **NJ**

  Neymar drew attention for his impressive soccer abilities at an early age. He emerged as a star for Santos FC as a teenager, winning four straight Player of the Year awards while becoming one of Brazil's most popular public figures. Neymar made the leap to Europe to join FC Barcelona for the start of the 2013-14 season and became a fixture for a club that claimed numerous domestic and international titles. After leading the Brazilian men to their first Olympic gold medal in 2016, the star forward transferred to France's Paris Saint-Germain the following year.

  (*source: https://www.biography.com/athlete/neymar)*

- **KC**

Kiran Chemjong (Nepali: किरण चेम्जोङ) (born 1990-03-20) is currently enrolled in goalkeeper positions from the Three Star Club and Nepali National Team. After graduating from ANFA Academy Chemjong joined Machhindra Football Club where he spent a year. After showing impressive performance between the sticks for Red Lions, Mega Three Star Club offered him a job in 2065 B.S. He has won many tournaments with Mega Three Star Club including British Gurkha Cup and Aaha Gold Cup Football Tournament.

(*source: https://peoplepill.com/people/kiran-chemjong)*

CONCLUSION

To conclude, this part of coursework has been knowledgeful while doing and equally useful in terms of understanding and writing shell script program. Bash (Bourne Again Shell) is used for writing the program script. While writing the program script, various websites and internet resources have been accessed to solve the program requirement. After the completion of scripting, testing for various test cases were performed. While doing the coursework, the guidance and help of lecturer and tutor was appreciable.

After the completion of this part of coursework, the knowledge required to complete the scripting and the thrust of curiosity was satisfied. I am glad and proud to learn these concepts of UNIX and shell operations while improving my personal skills on UNIX OS.

PART B

INTRODUCTION

For any computer Operating System, memory management is a critical part. Memory management is the process which controls the entire computer memory by assigning the chunks called blocks to various programs to boost overall system performance. Memory management includes various techniques and methods such as memory placement, memory allocation/deallocation, mapping, swapping, virtualization, page coloring, paging, segmentation and so on. These methods are used for countering problems such as internal fragmentation, external fragmentation, false sharing, issues in data locality, process security, process latency, CPU time wastage, and so on.

Here, concepts such as internal and external fragmentation is to be understood before moving forward. Internal fragmentation is a memory issue where there is a gap between required memory space and allotted memory space and generally occurs when a process requires extra memory space than the size of allotted memory block. External fragmentation is another memory issue when there are small and distant memory blocks that cannot be allocated to any process. It usually occurs when a process is removed from main memory.

AIMS AND OBJECTIVES

The main aim is to write a technical report that focuses on memory issues that consist of well searched content from various sources including book, journals, reports, web resources and text book. Next is to understand the working of physical memory including registers, cache and primary memory in contrast.

To carry out this report major task to be done is research about memory issues, memory management techniques and physical memory. In this research task, to understand and write the contents from reports, book and web sources is another focus.

While some of the researches are to be done by reading journals and PhD. Thesis presentation. To find out major memory issues and its solution is must.

BACKGROUND

PHYSICAL MEMORY

Physical memory is one of the major components in any computer or computer system. No process can be executed without loading in RAM (main memory). Generally, a memory is required to be tremendously fast so that the CPU must not held by memory when an instruction is executed and it must have large space as to store large amount of data. These criteria are not fully satisfied by any technology till now so physical memory is constructed as a hierarchy of layers which is arranged as memory with high speed and less capacity are at top layers and memory with lower speeds and higher storage capacity are arranged at bottom layers consisting Registers at the topmost layer followed by Cache memory, Main memory, magnetic disk and Magnetic tape at bottom layer. (Jain, 2019)

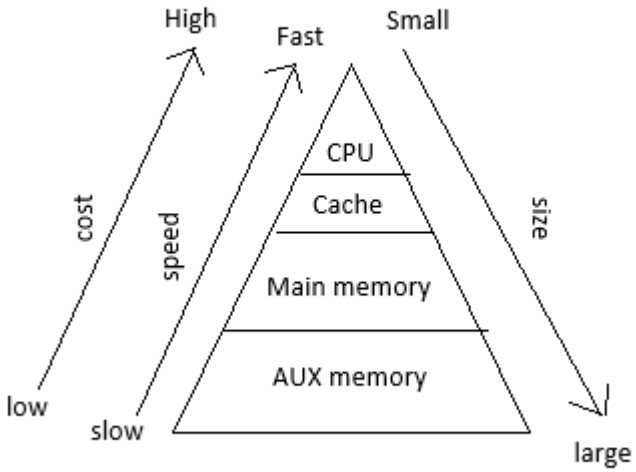

Figure 19: Memory Hierarchy

Registers are nearly as fast as CPU as they are made of same materials as CPU due to which registers can be accessed without any delay. The storage capacity of registers depends on the architecture of CPU as 32 x 32-bits on 32-bit CPU and 64 x 64-bits on 64-bit CPU. The software using the register should manage register itself by deciding what to keep in it so the processing becomes faster. (Tanenbaum, 2008)

New CPUs are provided with two cache memory namely; L1 and L2 where L1 provides decoded data and instructions to execution engine of CPU whereas L2 cache memory can hold recently used words at size of megabytes.

Following the memory hierarchy, next is main memory (RAM) which is a volatile memory. When CPUs requests or process are not fulfilled by the cache memory, they are stored in main memory to be executed. Beside RAM there is a non-volatile memory known as ROM which is pre-programmed during manufacture and cannot be modified later.

In memory hierarchy, magnetic tape which holds large amount of data. A tape reader is required to access the magnetic tape and then the requested block of tape can be accessed by the user at high level. (Tanenbaum, 2008)

MEMORY PLACEMENT

Sequential patterns, association rules and several data mining tasks use sophisticated data structures that are pointer based such as hash tree. These structures lack sub-optimal data locality due to which in case of multiprocessor, access to data structures leads to false sharing. So, recursive data structure is made to access multiple time during each iteration. In case of data locality and false sharing, well memory placement of data structures can boost performance. (Srinivasan Parthasarathy, n.d.)

To improve execution time, simple placement schemes and set of placement policies can be taken in use as well as it provides faster memory freeing option and efficient usage of pre-allocated memory. (Srinivasan Parthasarathy, n.d.) Some integral memory placement policies can be described as,

- **Improving Reference Locality**

  It is observed that memory latency for data related applications such as association mining must be reduced. Some specific policies for improving reference locality can be listed as,

  - Common Candidate Partitioned Database (CCPD)
  - Simple Placement Policy (SPP)
  - Localized Placement Policy (LPP)
  - Global Placement Policy (GPP)

  (Srinivasan Parthasarathy, n.d.)

- **Reducing False Sharing**

  Most different problem faced with shared-memory systems is false sharing, that generally exists when two different shared variables are located in same cache block. It leads the block exchange between processors even if the processors access different variables.

  A solution to this problem is Padding and Aligning, that can be done by placement of unrelated read and write data on different cache lines. While padding terminates false sharing, it creates unacceptable memory space overhead and loss in data locality.

  Thus, to eliminate false sharing completely a new scheme called Local Counter Array-Global Placement Policy (LCA-GPP) is formed by combining GPP with Privatize (and Reduce) scheme that produces a private copy of the data which is used locally, so that operations on that data do not cause false sharing. This allows to keep local array of counters per processor. This scheme completely terminates false sharing with acceptable memory wastage along with eliminating the need for lock synchronization among processors. (Srinivasan Parthasarathy, n.d.)

PAGE COLORING

Page coloring can be understood as a cache partitioning mechanism to optimize performance by controlling the mapping of physical memory pages to (processor's) cache lines that ensures the access to contiguous pages in virtual memory make the best use of processor cache. (FreeBSD, n.d.) Page coloring is widely used Operating System technique to improve cache and memory performance (Jiang Lin, 2008).

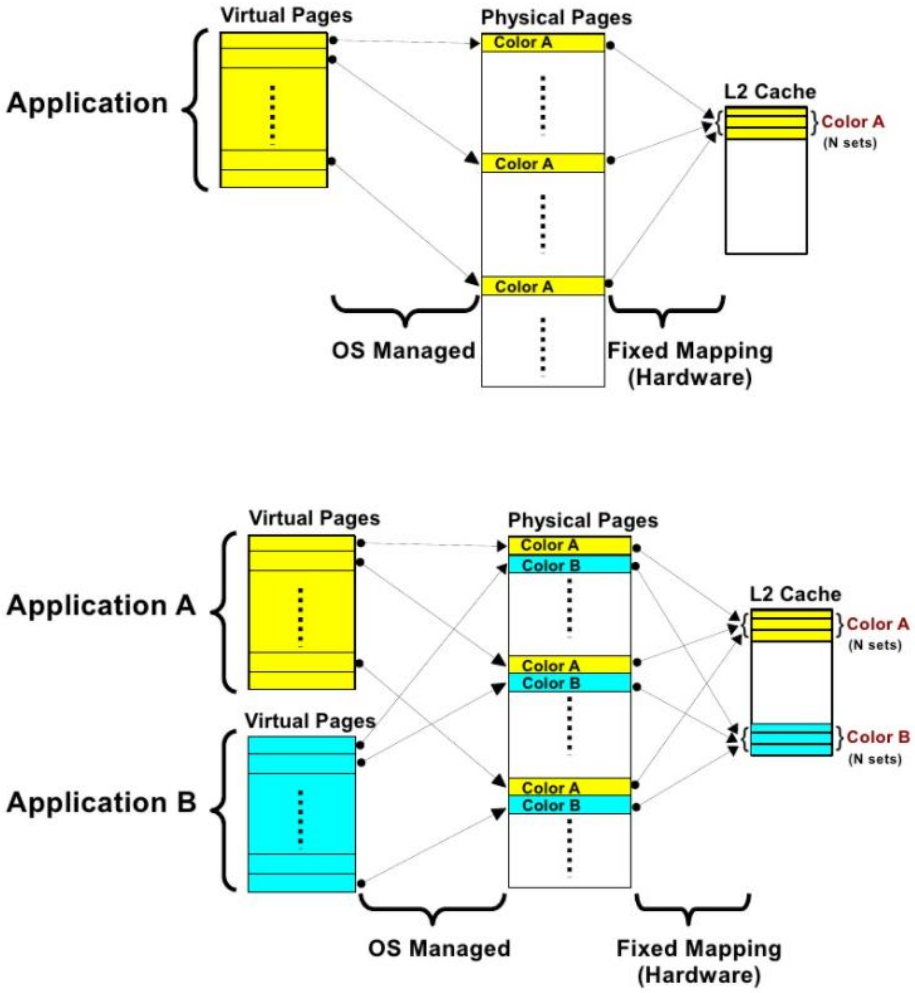

Figure 20: Page Coloring Technique

(Source:https://image.slidesharecdn.com/foe-12721365923473-phpapp02/95/phd-thesis-presentation-16-1024.jpg?cb=1272118675)

Above figure depicts the general concept of page coloring technique. In a physically indexed L2 cache, every physical page has a defined mapping to continuous

group of cache lines where all physical pages with same color are mapped to a group cache lines of same color. Thus, L2 cache can be managed efficiently by controlling the color of pages assigned to a process. (Junghoon Kim, 2011)

PAGING AND SEGMENTATION

A computer is able to specify extra memory than the amount physically installed on the system known as virtual memory and it is used to emulate the computer's RAM.

Paging is a memory management technique that divides the process address space into pages of equivalent size. A size of process is measured in the number of pages. Likewise, main memory is also divided into small fixed-sized blocks of physical memory termed as frames and to have optimum usage of main memory and avoid external fragmentation, the size of the frames is specified same as the size of a page. (tutorialspoint, n.d.)

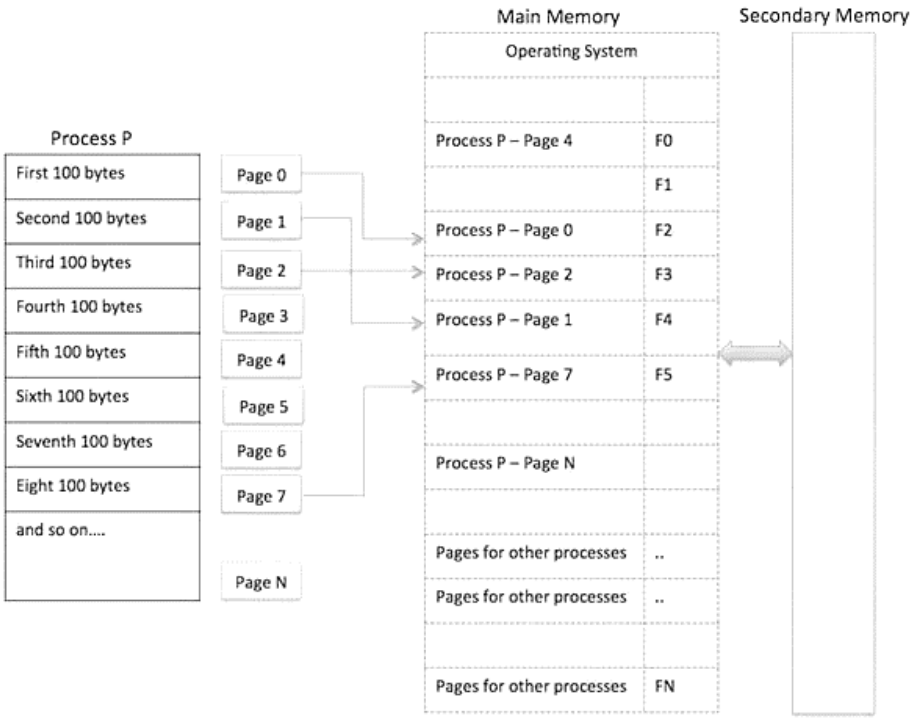

Figure 21: Paging Technique

(source: https://www.tutorialspoint.com/operating_system/images/paging.jpg)

Address translation is a significant concept in paging, which has logical address and physical address where logical address is simply the page address represented by page number and offset, and physical address is frame address generally represented by frame number and the offset.

Mathematically,

Logical address = page number + page offset

Physical address = frame number + page offset

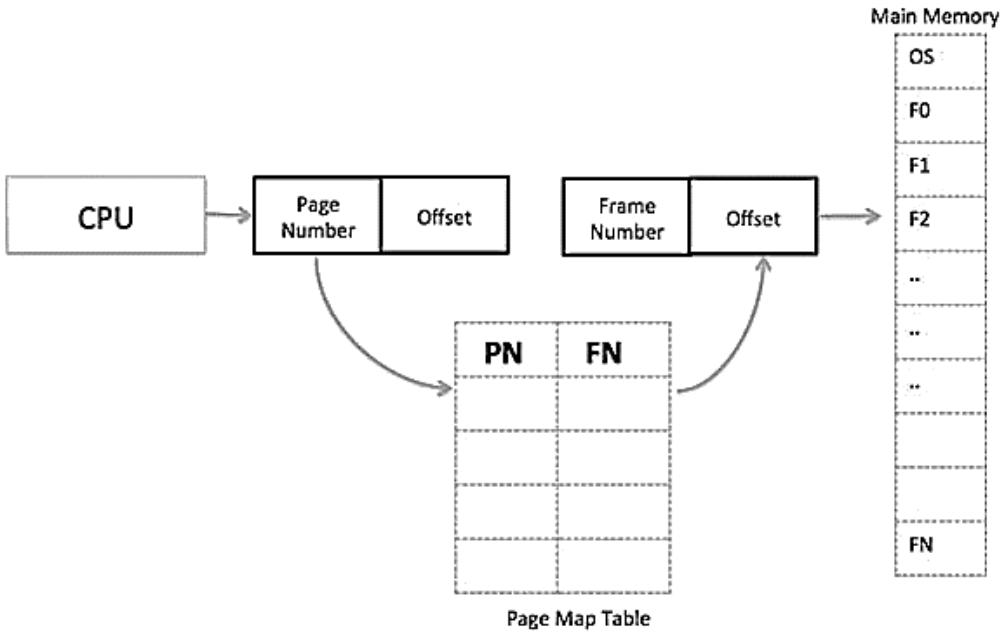

Figure 22: Page Map Table

(source: https://www.tutorialspoint.com/operating_system/images/paging.jpg)

To keep track of link between process page to frame in physical memory, a data structure known as page map table is used as represented in the above diagram. When a frame is allocated to any of the page, system convert the logical address into physical address and make entry in the page table that is used during the program execution. The concept of paging is implemented when a process is to be executed in the system, its related pages are stacked into available memory frames. For example, a program is of 8 KB but a system memory can handle only 5 KB at a time then OS will shift unused pages

of memory to secondary memory to make RAM free for other processes and take them back when required by program. Therefore, this process keeps continuing during the whole execution of program. (tutorialspoint, n.d.)

The major drawback of paging is page table needs extra memory space which is not suitable for system with small RAM size also paging suffers internal fragmentation.

Coming towards **Segmentation**, each task is divided into segments having different sizes one for each module having pieces which performs related functions. In segmentation, each of the segment is a logical address space of the program. Segmentation technique is very similar to paging but in segmentation the segments are of variable-length size and in paging pages are fix sized. In segmentation, virtual address space is divided into separate logical segments which are a part of physical memory. (Yang, n.d.)

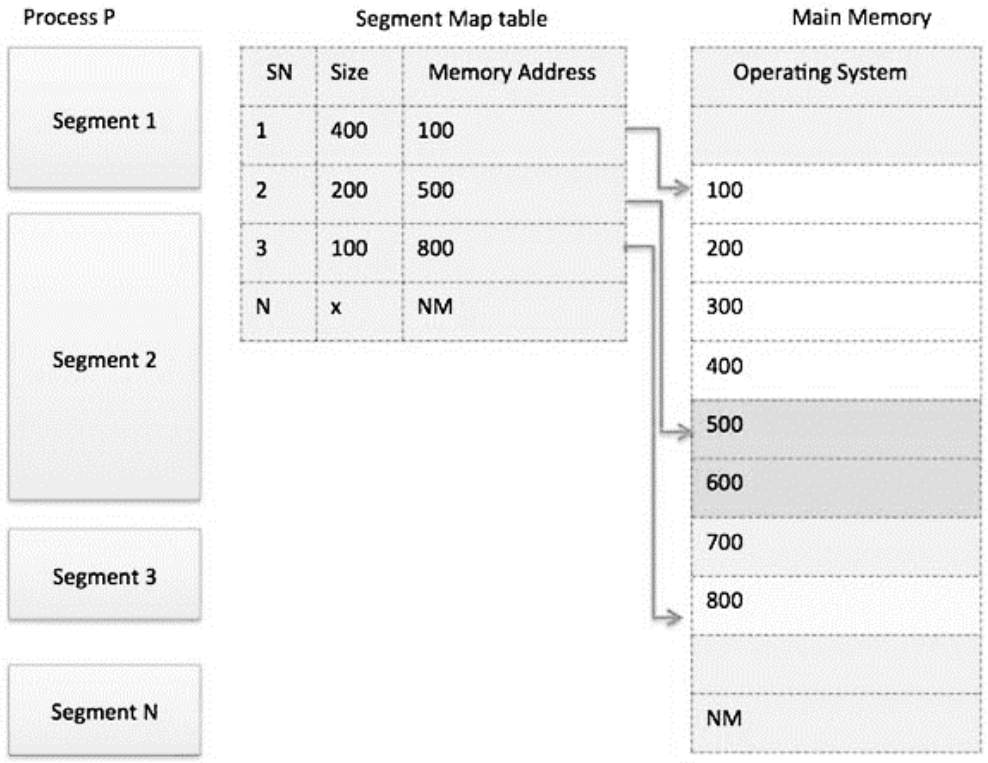

Figure 23: Segmentation Technique

When a program is executed, in segmentation, segments consist of the program's main function, data structures, utility functions. Here, an OS maintains a

segment map table for each process, memory blocks, segment number, size and memory location in the main memory. for each and every segment, the segment map table records the starting address and length of the segment also, a reference to memory location contains a value that recognizes a segment and offset. (tutorialspoint, n.d.)

Segmentation has several advantages in memory management such as it is efficient and easy to relocate segment rather than entire program, it avoids allocating unused memory, it provides flexible protection and most importantly it offers efficient translation that is segmentation has small segmentation map table that fits in Memory Management Unit.

Although, paging and segmentation may be similar but these memory management techniques are different in several factor such as,

- In paging, page size is declared by the computer hardware whereas in segmentation the segment size is decided by the user as it stores program's functionalities and data structure. (Studytonight, n.d.)
- Paging major drawback is internal fragmentation while segmentation may bring external fragmentation.

PAGE SIZE VARIATION

The page size is a parameter chosen by the OS. For hardware designed with 512-byte pages, the OS can consider or hold the page pairs 0 and 1, or 2 and 3 and so on as 1-KB pages by allocating two consecutive 512-byte page frames for them. (Tanenbaum, 2008)

The page size variation can be divided into small page size and large page size. As determining best page size comes with balanced competing factor resulting worst at most cases. There are two main factors that contend for a small page size. When text, data or stack segment is chosen randomly it will not fill a basic number of pages that will cause half of the page to be empty. This wastage of extra space in the page is called internal fragmentation. Next argument for small page size is about a program consisting

of eight phases of 4-KB each. So, with the 32-KB size, the program must be allocated with 32 KB all the time. Likewise, a 16-KB page size requires only 16 KB and with 4-KB page size or smaller it requires 4 KB. That points a large page size will cause more idle program to be in memory rather than small page size. (Tanenbaum, 2008)

Hence, small pages lead to large page table as program will need many pages. For example, a 32-KB program requires four 8-KB pages each whereas for 64 512-byte page is required. Some devices are required to load page table into registers each and every time when CPU switch to another process. On these devices, the space occupied by page table increases as the page size decreases.

CONCLUSION

In this report, the rationale behind the Memory Management techniques and some major memory issues for Operating System is discussed and concepts about Physical memory, memory placement, page coloring, page size variation, paging and segmentation are explained technically with diagrams.

In particular, it is argued that how physical memory is represented in memory hierarchy, how can CPU process faster with proper memory placement and allocation techniques, how to reduce false sharing and improve data locality, what are the major difference between paging and segmentation, how can paging technique overcome external fragmentation but leads to internal fragmentation, page size variation with example, how system translates logical address to physical address and data structures such as page map table and segment map table.

After the depth research and understanding about physical memory and memory issues, appropriate memory placement policy and technique most be used depending upon the system environment and compatibility.

REFERENCES

FreeBSD. (n.d.) *Page Coloring* [Online]. Available at: https://docs.freebsd.org/en_US.ISO8859-1/articles/vm-design/page-coloring-optimizations.html.

guru99. (n.d.) *Shell Scripting Tutorial* [Online]. Available from: https://www.guru99.com/introduction-to-shell-scripting.html.

Jain, P. (2019) *Introduction to Memory Management* [Online]. Available from: https://www.includehelp.com/operating-systems/concept-of-physical-and-virtual-memory.aspx [Accessed 06 April 2021].

Jiang Lin, Q.L.X.D.Z.Z.X.a.P.S. (2008) Gaining Insights into Multi-Core Cache Partitioning: Bridging the Gap between Simulation and Real Systems. In *14th International Conference on High-Performance Computer Architecture*. Salt Lake City, UT, USA: IEEE. pp.367-78.

Junghoon Kim, J.K.D.A.Y.I.E. (2011) *Computational Science and Its Applications - ICCSA 2011*. 2011th ed. Heidelberg, Berlin: Springer.

Srinivasan Parthasarathy, M.J.Z.a.W.L. (n.d.) *Memory Placement Techniques for Parallel Association Mining*. Rochester NY 14627: KDD-98 Proceedings University of Rochester.

Studytonight. (n.d.) *Difference between Paging and Segmentation* [Online]. Available from: https://www.studytonight.com/operating-system/difference-between-paging-and-segmentation [Accessed 8 April 2021].

Tanenbaum, A.S. (2008) *Modern Operating System*. 3rd ed. Amsterdam, The Netherlands: Pearson Education, Inc.

tutorialspoint. (n.d.) *Difference between internal and external fragmentation* [Online]. Available from: https://www.tutorialspoint.com/difference-between-internal-fragmentation-and-external-fragmentation#:~:text=Internal%20Fragmentation%20occurs%20when%20a,removed%20from%20the%20main%20memory.&text=Internal%20Fragmentation%20occurs%20when%20Paging,occurs%20 [Accessed 8 April 2021].

tutorialspoint. (n.d.) *OS - Memory Management* [Online]. Available from: https://www.tutorialspoint.com/operating_system/os_memory_management.htm [Accessed 7 April 2021].

Yang, J. (n.d.) *Segmentation and Paging* [Online]. Available at: https://www.cs.columbia.edu/~junfeng/13fa-w4118/lectures/l05-mem.pdf [Accessed 9 April 2021].