LONDON METROPOLITAN UNIVERSITY

ITAHARI INTERNATIONAL COLLEGE

Module Code & Module Title

CS5001NT Network and Operating System

Assessment Weightage & Type

20% Individual Coursework

Year and Semester

2020 Autumn/Spring

Student Name: Delish Khadka

London Met ID: 19031630

College ID: np05cp4a190051

Assignment Due Date: 11-04-2021

Assignment Submission Date: 11-04-2021

Word Count:

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

Part A

1. Introduction

Shell is the primary program that computers use to receive code or commands and return information by executing the commands. Bash, which is also known as ‘Bourne Again Shell” allows us to do many tasks such as performing tasks on many files quickly using command line and automate workflows across operating systems. We can also write and execute scripts in bash just like the python programming language, which can be used in many operating systems.

In this coursework bash shell will be used to create a user interactive program by using different type of functions. The user have to input their answers and the program will guide the users until the end of the program by providing hints and chances for user to rectify their mistakes. The program will be simply running from the command line by using command : filename.sh Username ID

2. Script

```
1. #!/bin/bash
2. #Declaring function
3. # this WelcomeUser function is used to display the logged in username
   and ID and the date the program is running
4. WelcomeUser(){
5.   echo ""
6.   echo "------------------------------------------------------------
   ----"
7.   echo "Welcome to the selection game!!!"
8.   echo "You are logged in as:" $1
9.   echo "Your logged in user ID is:" $2
10.          echo "Program is starting at "$( date)
11.          echo "------------------------------------------------------
   ------------"
12.          echo ""
13.  }
14.  #this SelectCountry function is used to display country names and
   codes for the user to select the country for moving to further round
15.  #this function calls SelectTeam function
16.  SelectCountry(){
17.          echo "------------------------------------------------------
   ------------"
18.          echo "WELCOME TO LEVEL ONE"
19.          echo "HINT: Here you have to guess the best football team in
   the world"
20.          echo "-------------------------GOOD LUCK-------------------
   ---------"
21.          echo
   "********************************************************************"
22.          echo "HERE ARE THE FIVE BEST INTERNATIONAL FOOTBALL TEAMS "
23.          echo ""
24.          echo "_________________________"
25.          echo "|S.N |Countries |Code |"
26.          echo "........................."
27.          echo "|1   |Brazil    |BRZ  |"
28.          echo "|2   |Argentina |ARG  |"
29.          echo "|3   |Nepal     |NEP  |"
30.          echo "|4   |China     |CHI  |"
31.          echo "|5   |England   |ENG  |"
32.          echo "........................."
33.          echo ""
34.          echo "------------------------------------------------------
   ------------"
35.          SelectTeam
36.  }
37.  #SelectTeam function is used to set the best country in the world
   i.e. Brazil and error messages are coded  if the user inputs any
   country codes except brazil
38.  #this function calls playerlist function
39.  SelectTeam(){
40.          nationSelect=""
41.          until [[ $nationSelect == BRZ ]]
42.          do
```

```
43.                 echo ""
44.                 echo -e "Please enter the country code of your guessed
   team: \c"
45.                 echo ""
46.         read nationSelect
47.         case $nationSelect in
48.         "BRZ") echo "Congratulations! You made the correct guess.
   Brazil have won world cup five times"
49.         ;;
50.         "ARG") echo "Sorry! You guessed wrong answer . Please try
   again."
51.         ;;
52.         "CHI") echo "Sorry! You guessed wrong answer . Please try
   again."
53.         ;;
54.         "NEP") echo "Sorry! You guessed wrong answer . Please try
   again."
55.         ;;
56.         "ENG") echo "Sorry! You guessed wrong answer . Please try
   again."
57.         ;;
58.         *) echo "It seems you did not enter the correct country code,
   please input the correct country code of your guessed team"
59.         echo ""
60.         esac
61.         done
62.         echo ""
63.         PlayerList
64.  }
65.  # PlayerList function is used to display player name and player code
   so that the user can select three player codes and move on next round
66.  #Error messages are displayed from this function if the user enters
   wrong player code or enters less or  more than three player code and
   duplicate entries are also checked
67.  #this function calls PlayerChoosen function by passing 3 parameters
68.  PlayerList(){
69.         echo "------------------------------------------------------
   ------------"
70.         echo "WELCOME TO LEVEL TWO"
71.         echo "Here are the list of five best football players of the
   world"
72.         echo "_____________________________"
73.         echo "|S.N |Players       |Code|"
74.         echo "............................."
75.         echo "|1   |Lionel Messi  |LM  |"
76.         echo "|2   |Neymar Junior |NJ  |"
77.         echo "|3   |Kiran Chemjog |KC  |"
78.         echo "|4   |Zheng Zhi     |ZZ  |"
79.         echo "|5   |Harry Kane    |HK  |"
80.         echo "............................."
81.         echo ""
82.         echo "HINT: HERE YOU HAVE TO CHOOSE THREE OF THE BEST PLAYERS"
83.         echo ""
84.         echo -e "Please enter the players code(BY SEPARATING THEM WITH
   SPACE): \c"
85.         read FP1 FP2 FP3
86.         choose=($FP1 $FP2 $FP3)
```

```
87.         numOfArg=${#choose[@]}
88.         if [[ ${#choose[@]} -eq 3 ]]
89.         then
90.                 if [ $FP1 = $FP2 ] || [ $FP1 = $FP3 ] || [ $FP2 = $FP3
    ]
91.                 then
92.                         echo -e "ERROR INPUT!"
93.                         echo -e "YOU ENTERED THE SAME PLAYER CODE
    TWICE! PLEASE TRY AGAIN"
94.                         echo -e ""
95.                         PlayerList
96.                 else
97.                         for n in $choose
98.                         do
99.                                 if [[ $n == "LM" || $n == "NJ" || $n ==
    "KC" || $n == "ZZ" || $n == "HK" ]]
100.                                then
101.                                        continue
102.                                else
103.                                        echo ""
104.                                        echo "INVALID PLAYER CODE!
    PLEASE ENTER VALID PLAYER CODE."
105.                                        PlayerList
106.                                fi
107.                        done
108.                fi
109.                PlayerChoosen 3
110.         else
111.                echo ""
112.                echo -e "ERROR"
113.                echo -e "PLEASE ENTER THREE PLAYER CODE SEPARATED BY
    SPACE"
114.        PlayerList
115.        fi
116.  }
117. #this function is used to select one out of three players that are
    passed from PlayerList function
118. #the three players LM, NJ and KC are linked to their respective files
119. #the player ZZ and HK dont have their files so they dont display their
    details
120. PlayerChoosen(){
121.        if [ $1 == 3 ]
122.        then
123.                echo ""
124.                echo "CONGRATULATIONS! YOU HAVE MADE TO LEVEL 3"
125.                echo "Among the three best players. Choose your one
    favorite player"
126.                PS3="Select the player: "
127.                select player in $FP1 $FP2 $FP3
128.                do
129.                        case $player in
130.                        "LM")
131.                             cat LM
132.                             break
133.                          ;;
134.                         "NJ")
135.                             cat NJ
```

```
                                break
                            ;;
                            "KC")
                                cat KC
                                break
                            ;;
                            "ZZ")
                                echo "Sorry! there is currently no details
   of the player"
                                break
                            ;;
                            "HK")
                                echo "Sorry! there is currently no details
   of the player"
                                break
                            ;;
                            *)
                                    echo "Invalid input"
                                    PlayerChoosen 3
                            ;;
                        esac
                    done
                Contd
            else
                  echo ""
                  echo "Warning!! Please enter valid input"
                  PlayerList
            fi
}
#this function is called after the user successfully completes all the
   levels
#yes input causes the program to repeat whereas any other keys results
   in termination of the program
Contd(){
        echo ""
        echo -e "Do you want to continue this again? If you want than
   enter 'yes' and 'no or any key'  to exit: \c"
        echo ""
        read answer
        if [ $answer == yes ]
        then
                SelectCountry
        else
                exit
        fi
}
#the secret code is also made  and incase of 4 consecutive failed
   attempts the program terminates immediately
#it asks for the user name and ID while starting the program
#It calls the function SelectCountry and WelcomeUser
Secretcode=1234
if [ $# == 2 ]
then
        a=0
        while [[ $a -lt 4 ]]
        do
                echo -e "Enter the secret key:  \c"
```

```
187.                read  -s secretKey
188.                if [[ $secretKey == $Secretcode ]]
189.                then
190.                        WelcomeUser $1 $2
191.                        SelectCountry
192.                break
193.
194.                else
195.                        echo "Wrong! Please enter the valid secret Key
   $SecretKey."
196.                        ((a++))
197.                fi
198.        done
199. else
200.         echo "Please enter First Name and ID number for starting the
   program"
201. fi
202.
```

3. Testing

Test No.	1
Input	Run without putting name and id
Expected Output	To give an error message to enter name and id number
Actual Output	To give an error message to enter name and id number
Test Result	Test successful

Table 1: Testing no 1, run without name

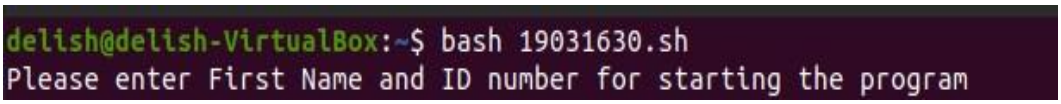

figure 1

Test No.	2
Input	Run with more than 2 arguments
Expected Output	To give an error message to enter name and id number
Actual Output	To give an error message to enter name and id number
Test Result	Test Pass

Table 2: Testing 2, running with more than 2 arguments

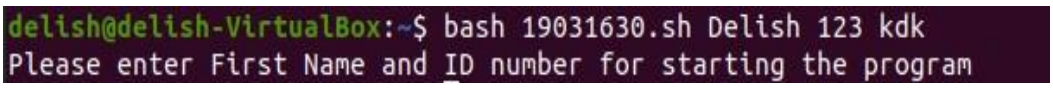

figure 2

Test No.	3
Input	Run with ID and username
Expected Output	Ask to enter the secret code
Actual Output	Asked to enter the secret code
Test Result	Test Pass

Table 3: Test 3 Running with ID and username

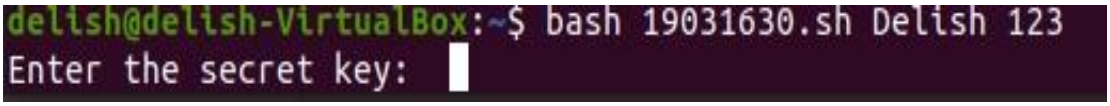

figure 3

Test No.	4
Input	Wrong password 4 times
Expected Output	The program gets terminated
Actual Output	Program terminated
Test Result	Test Pass

Table 4: Test 4, running with incorrect password for 4 times

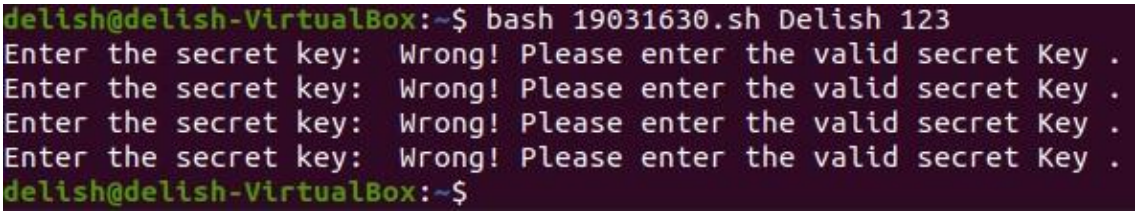

figure 4

Test No.	5
Input	Correct password
Expected Output	To ask to enter country code of team
Actual Output	Asked to enter the country code of team
Test Result	Test Pass

Table 5: Testing 5, running with correct password

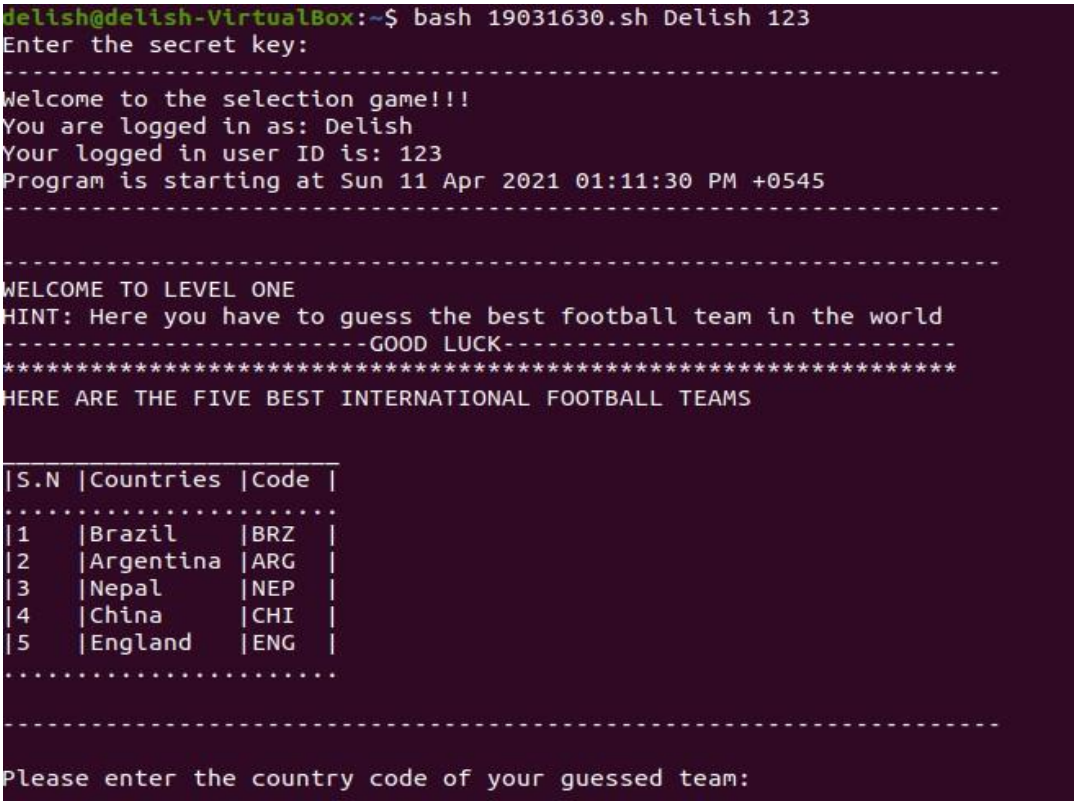

figure 5

Test No.	6

Input	Incorrect country name
Expected Output	To give an error message and ask to enter correct country code
Actual Output	An error message is given and asked to enter the correct country code
Test Result	Test Pass

Table 6: Testing 6, country name

figure 6

Test No.	7
Input	Wrong country code
Expected Output	To give an error message and ask to guess another country name

Actual Output	An error message is given and asked to guess another country name
Test Result	Test Pass

Table 7: Testing 7, Incorrect country code

figure 7

Test No.	8
Input	Entering only username while logging in

Expected Output	To give error message to enter both username and ID
Actual Output	An error message was given to enter both name and ID
Test Result	Test Pass

Table 8: Testing 8, correct country cod but wrong selection

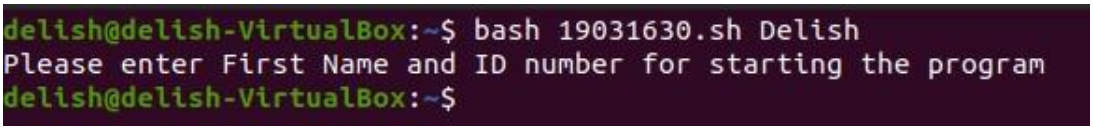

figure 8

Test No.	9
Input	Correct country code
Expected Output	To give congratulation message and ask user to select 3 best players
Actual Output	Congratulation message was given and asked user to select 3 best players
Test Result	Test Pass

Table 9: Testing 9, correct country code

figure 9

Test No.	10
Input	Code of 3 wrong player codes

Expected Output	To give an error message and ask to enter valid code
Actual Output	An error message is given and asked to enter valid player code
Test Result	Pass

Table 10: Testing 10, wrong player codes

```
HINT: HERE YOU HAVE TO CHOOSE THREE OF THE BEST PLAYERS

Please enter the players code(BY SEPARATING THEM WITH SPACE): lm nj kc

INVALID PLAYER CODE! PLEASE ENTER VALID PLAYER CODE.
------------------------------------------------------------------------
WELCOME TO LEVEL TWO
Here are the list of five best football players of the world
_________________________
|S.N |Players       |Code|
.........................
|1   |Lionel Messi  |LM  |
|2   |Neymar Junior |NJ  |
|3   |Kiran Chemjog |KC  |
|4   |Zheng Zhi     |ZZ  |
|5   |Harry Kane    |HK  |
.........................

HINT: HERE YOU HAVE TO CHOOSE THREE OF THE BEST PLAYERS

Please enter the players code(BY SEPARATING THEM WITH SPACE):
```

figure 10

Test No.	11
Input	Entered same player name twice
Expected Output	To give an error message and ask to choose three different players

Actual Output	An error message is given and asked to choose three different players
Test Result	Test Pass

Table 11: Testing 11, same player name twice

figure 11

Test No.	12
Input	Three different player code
Expected Output	To ask user to select one player amond the three
Actual Output	The user was asked to select one player from any three players
Test Result	Pass

Table 12: Testing 12, different player name

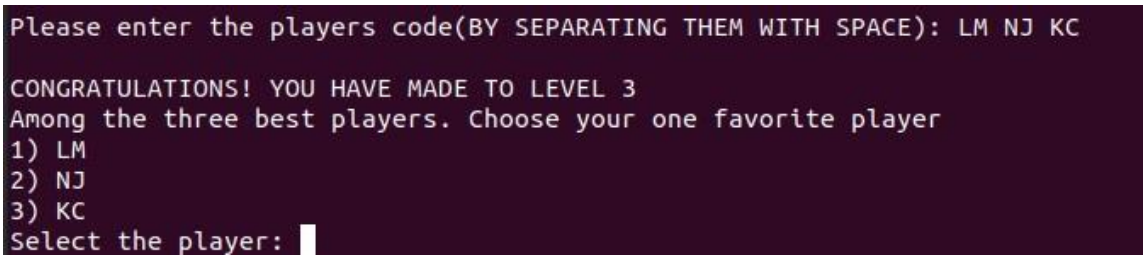

figure 12

Test No.	13
Input	Wrong id number of player
Expected Output	To give an error message and ask to select your favourite player from the list
Actual Output	An error message is given and asked to select your favourite player from the list
Test Result	Test Pass

Table 13: Testing 13, wrong id number

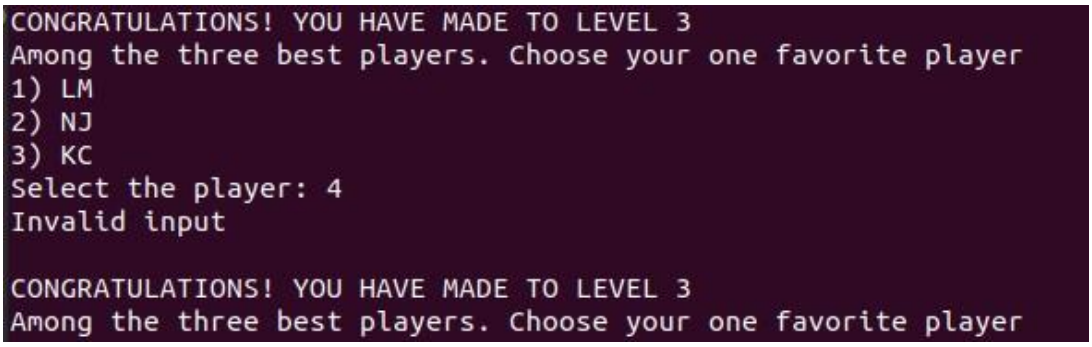

figure 13

Test No.	14
Input	ID of player who don’t have details

Expected Output	To inform user that there is no details of the player
Actual Output	User was notified there was no player details
Test Result	Pass

Table 14: Testing 14, player without details

```
CONGRATULATIONS! YOU HAVE MADE TO LEVEL 3
Among the three best players. Choose your one favorite player
1) LM
2) NJ
3) ZZ
Select the player: 3
Sorry! there is currently no details of the player
```

figure 14

Test No.	15
Input	Player who are assigned with details
Expected Output	To give the details of selected player and prompt if the user wants to restart the program again
Actual Output	Details of selected player was given and prompt if the user wants to restart the program again was also given
Test Result	Test Pass

Table 15: Testing 15, player detail

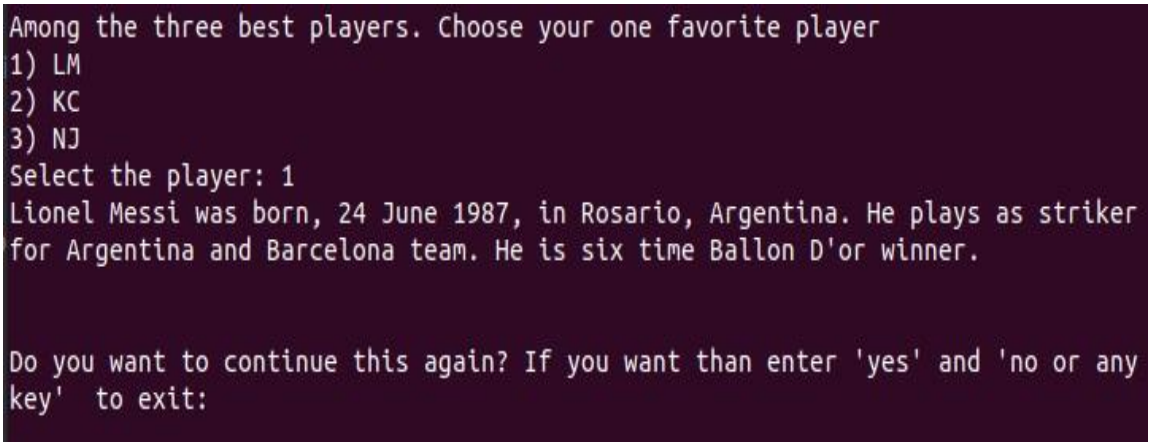

figure 15

Test No.	16
Input	yes
Expected Output	To re-start the whole program
Actual Output	The whole program is re-executed
Test Result	Test Pass

Table 16: Testing 16, exit

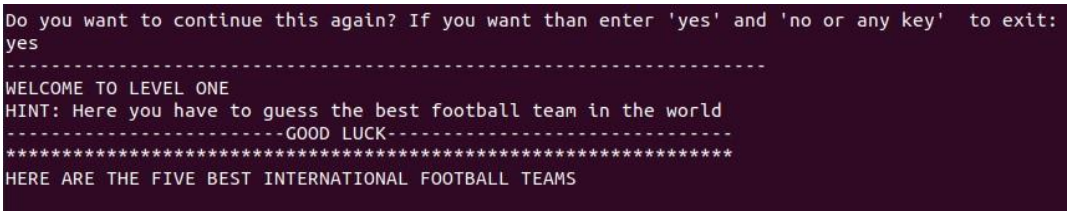

figure 16

Test No.	17
Input	no
Expected Output	The program will be closed

Actual Output	The program is exited
Test Result	Test Pass

Table 17: Testing 17, exit ii

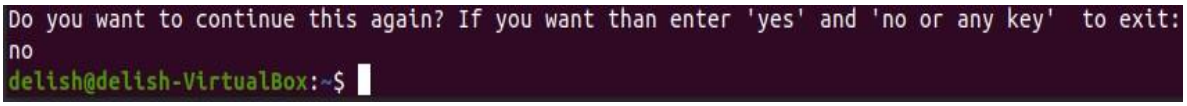

figure 17

4.Content of three files:

LM: Lionel Messi was born, 24 June 1987, in Rosario, Argentina. He plays as striker for Argentina and Barcelona team. He is six time Ballon D'or winner.

NJ: Neymar da Silva Santos Jr. was born on February 5, 1992, in Mogi das Cruzes, São Paulo, Brazil. He plays as striker for Brazil and PSG team. He has won world soccer young player of the year.

KC: Kiran Chemjong was born in March 20 1990 in dhankuta district of Nepal. He plays as goalkeeper for Nepali national team. He has won many national awards for best goalkeeper of the year

5. Conclusion

The program was created using bash shell and using various functions. The coursework was important to demonstrate how bash shell can be used to create programs by coding.

The program was successfully tested by checking all types of error that can appear while the program is running. Three external files LM, NJ and KC was created with the details of corresponding players. The design was mostly done by using simple box design using the echo command. The coursework made us apply problem solving logics in coding part and a lot of research was needed for the successful completion of the coursework.

Part B

1. Introduction:

A network operating system (NOS) is software that runs on a server and enables the server to manage data, users, groups, security, applications, and other network functions. Network operating systems are designed and optimized to support concurrent, multi-user environments. This means that with the network operating system, many users can use resources on the server at the same time. Network operating systems provide services for remote network clients.

The term memory management generally refer to various operations such as memory allocation/deallocation, virtualization, protection, allocation, sharing, etc. The part of the operating system that manages the storage hierarchy is called the memory manager. Its job is to manage memory efficiently: keep track of which parts of memory are in use, allocate memory to processes when they are needed, and free it up when finished. (Corin, et al., 2016)

Backing up processes requires enough memory and cannot be done if there is not enough memory. Applications are loaded through the NOS and interact with it to cause the configuration of network elements. If an application is stopped and downloaded, the rules installed on the devices will remain in their memory because the current NOS does not provide a mechanism to remove it automatically. Such behavior can be detrimental and affect the stability of the network. This causes unwanted network actions and prevents newly installed rules from working properly. The memory management system frees up the TCAM (Ternary Content Address Memory) space occupied by stream entries installed by terminated applications and makes it available for an application that is already running or for a new application. For this, the memory management system interacts with the NOS to keep track of all the rules installed on the devices by current applications, to identify when new applications need to be stopped or started, and the rules are applied automatically. (Joshy & Ramadas, 2016)

2. Aims and Objectives

The aims and objectives for this coursework is:

I. To write about memory issues occuring in Operating systems and NOS.

II. To address different errors the users might face during the use of operating systems.

III. To learn how security and protection works on the operating systems.

IV. To gather resources from various sources for in-depth research.

V. To understand how operating systems functions and transmission of data through network.

VI. To gain knowledge about high end workstations which programmers use for developing systems.

3. Background:

3.1. Physical Memory

Physical memory also known as Random Access Memory (RAM) is a very fast but volatile form of data storage. RAM modules are usually measured in nanoseconds, while physical drives are usually measured in milliseconds. When a system runs out of physical memory, delays often occur throughout the system or in severe cases, completely lock the system. (Huffman, 2015)

Switching to network operating systems raises two issues: how to provide high performance to users and how to enable users to share data easily. Performance is a problem in network environments because all data access may require network and disk access. Network access will be required if the data being accessed is stored on another workstation disk; diskless workstations aand data sharing workstations may need to have adequate network access. The performance problem can be solved by using large memory. Memory can be used to cache recently accessed file data and thus remove a large amount of disk and network access. By efficiently using large physical memories such as file data repositories, workstations can achieve high performance even without using a local disk; This high performance can be achieved by giving each workstation a consistent view of file system data without putting pressure on networks or servers. (SILBERSCHATZ, et al., 2004)

When memory is dynamically allocated, it must be controlled by the operating system. Generally, there are two ways to track memory usage: free lists and bitmaps. With a bitmap, memory is divided into units allocated in as few words and as large as several kilobytes. Each unit allocated in the map has a corresponding bitmap, that is 0 if the unit is free and 1 if it is occupied. Another way to keep track of memory is to keep a linked list of allocated and free memory segments, where one process is in a segment or a blank space between two processes. Each entry on the hole (H) or process (P) list indicates the address in which it starts, its duration, and a pointer for the next item. When processes and holes are kept in a list by address, various algorithms can be used to allocate memory to a created process. (Nelson, 1988)

3.2. Memory Placement

Memory consists of a large set of bytes, each with its own address. The CPU receives instructions from memory based on the value of the program counter. Memory space is protected by comparing the CPU hardware of each address generated in user mode with the programs. Only operating systems can load base and limit registers, which use a special privilege guide. Because privileged instructions can only be executed in kernel mode and only the operating system runs in kernel mode, only the operating system can load basic programs and limit registers. (SILBERSCHATZ, et al., 2004) However, a process can be temporarily swapped out of memory to backup storage and then returned to memory for later interruption. The operating system can be added with little or a lot of memory. The main factor affecting this determination is the location of the interrupt vector. Because the interrupt vector is often low in memory, programmers usually place the operating system in low memory as well. In contiguous memory allocation, each process resides in a portion of memory adjacent to the portion where the next process resides. (TANENBAUM & BOS, 1998) The reinstallation schedule provides an efficient way to dynamically change the size of the operating system. One of the simplest methods of allocating memory is to divide the memory into multiple partitions of fixed size. Each partition can have a process. Therefore, the degree of multiprogramming is bound by the number of partitions. In the multi-partition mode, when the partition is free, a process is selected from the input queue and loaded on the free partition. When the process is complete, the partition is available for another process. When processes enter the system, they are placed in an ingress queue. The operating system takes into account the memory requirements of each process and the amount of available memory space in deciding which processes to allocate memory to. When space is allocated to a process, it is loaded into memory and can compete for CPU time. When a process is finished, it frees its memory, which the operating system can then fill with another process from the input queue.

3.3. Page coloring

Page Coloring is a performance optimizer designed to ensure that access to nearby virtual memory pages takes full advantage of the processor cache. In past years,

processor caches tend to map virtual memory rather than physical memory. This leads to a number of problems, including the need to flush the cache at each context switch in some cases, and problems aligning the cached data. To work around this problem, tactile pages are given different colors in physical memory, the maximum number of colors depending on the size and the combination of the cache with the page size. The list of free pages is organized to distinguish between these colors, and neighboring virtual pages are guaranteed to be assigned different colors. Different use of page colors allows the operating system to restrict access to a process using only a subset of the cache. (Lynch & Flynn, 1991)

Due to page color limitations for an application, only some of the memory can be allocated to that application. If the system runs out of pages of a certain color, the application is under memory pressure, which can lead to significant delays due to page swapping and negative effects on the performance of other applications due to cache conflicts. And also because of the high overhead of online color change in a dynamic, multi-programmed rhythm environment. (Zhang, et al., 2008)Color change can often lead to excessive overhead on a large number of app pages which is largely detrimental to the benefit of coloring pages.

By controlling the color of the page assigned to a request, the operating system can manipulate cache blocks according to the grain size of the page multiplied by the cache associate. The maximum number of colors a platform can support is the cache line size multiplied by the number of sets and divided by page size.

3.4. Description of Paging and Segmentation

Most virtual memory systems use a technique called paging, which is used to access data more quickly. The correlated group units in physical memory are called page frames. Pages and page frames are generally the same sizes. When a process starts, the operating system loads the programs in the process page table, taken from a copy stored in main memory. At the end of the process, the page table no longer needs memory addresses. The advantages of this method are that it is simple and does not require memory addresses for mapping (Corin, et al., 2016). And the downside is that loading the full page table on any context switch would kill performance completely. The page table

can also reside entirely in the main memory. All the hardware needs then may be a single register that points to the beginning of the page table.

Pieces into which a program is divided that are not necessarily all the same size are called segments. Segmentation gives the user a picture of the process that paging does not. The big advantage of segmentation is to provide protection between components. Segmentation also makes it easy to share procedures or data between multiple processes by taking little support from hardware, in the form of protection bits. (OSTEP.ORG, 2008) A well-known example is a shared library. In a segmented system, the graphics library can be placed in a segment and shared with multiple processes, eliminating the need to have it in the address space of each process. Sometimes segmentation and paging are combined to supply a two-dimensional virtual storage . (TANENBAUM & BOS, 1998)

3.5. Page size variation

The page size is a parameter that can be chosen by the operating system. Each physical page frame in the system is assigned a struct page that is used to track the status. The distinction between the different page types is very fuzzy and the page types are identified by the flags or lists in which they appear rather than by the objects to which they belong. (Gorman, 2007)A large page size will cause more wasted space to be in memory than a small page size. The original approach of expanding the program to a fixed amount of space and then overlaying is altered, and there are a variety of options for replacing pages, depending in part on the allocation method applied. If each program in the main store is restricted to using a fixed amount of store, the replacement process is usually triggered by requesting new pages from that particular program. If the memory constraints are not so severe, the replacement of a page associated with a program can be imposed by any program that requests a new page. Without further refinement, both methods have disadvantages: in the first case, a program that tended to use one or more pages at the same time above its fixed allocation would work inefficiently under most replacement algorithms, and in the second case a visualization situation could arise, for example, when two programs in the main store alternately replace the pages that belonged together and which were needed and kept the machine almost indefinitely. Therefore, it has been

suggested that programs should be allowed to indicate which pages they have left out (or which pages they need). (TANENBAUM & BOS, 1998)

4. Conclusion

In this part of the coursework various search was done by using various type of resources. Different kind of books and online journals and research papers played a big role for the completion of this research. Searching for the research materials, I also got chance to study about many other things that was not needed for this report but useful for personal knowledge.

This assignment helped a lot to learn about the of memory issues that can be faced in the future personally or professionally. Also, the report writing ability by giving proper citation will always be needed for learners. In short, report writing part was a great opportunity to learn about new things in the tech world and familiarize ourselves with the vast world of technological advances.

5. References

Bibliography

Corin, R. D., Siracusa, D. & Salvadori, E., 2016. *Empowering Network Operating Systems with Memory Management.* s.l., ResearchGate, p. 6.

Gorman, M., 2007. *Understanding the linux virtual memory manager.* s.l.:Prentice Hall PTR.

Huffman, C., 2015. *Physical Memory,* s.l.: Science Direct.

Joshy, A. & Ramadas, S., 2016. A COMPARATIVE STUDY ON NETWORK OPERATING SYSTEMS. *IOSR Journal of Computer Engineering (IOSR-JCE),* p. 7.

Lynch, W. L. & Flynn, M. J., 1991. *PAGING PERFORMANCE WITH PAGE COLORING,* Stanford: Stanford University.

Nelson, M. N., 1988. *Physical Memory Management in a Network Operating System,* California: s.n.

OSTEP.ORG, 2008. *Segmentation.* [Online] Available at: https://pages.cs.wisc.edu/~remzi/OSTEP/vm-segmentation.pdf

SILBERSCHATZ, A., GALVIN, P. B. & GAGNE, G., 2004. *Operating system concepts: 9th edition.* California: s.n.

TANENBAUM, A. S. & BOS, H., 1998. *Modern Operating Systems 4th edition.* Amsterdam: Pearson.

Zhang, X., Dwarkadas, S. & Shen, K., 2008. *Towards Practical Page Coloring-based Multi-core Cache Management.* [Online] Available at: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.726.7659&rep=rep1&type=pdf

6. Appendix

6.1. Memory Controller

The memory controller has three types of interface generation: standard chip selection generation for SRAM, ROM, and basic devices that only require chip selection, dedicated SDRAM controller to provide correct memory control signals, connections and timing for SDRAM devices. User defined control signal is used by the programmable machine for SDRAM and other type of memory. The local and 60x buses use the memory controller, so there are separate commands for both. There are several parity options available.

During the "upgrade," the memory controller sends a pulse of electronic current through the RAM chips. The amount of power that RAM sends through the computer's binary input and output system (BIOS) is selected. This occurs at least every 64 milliseconds, keeping RAM active and protecting stored data from loss due to power outages. The data could be lost in a fraction of second Without the memory controller. The memory controller also

manages the read and write operations on the RAM chips. It is used to select the appropriate demultiplexer circuit for data storage and retrieval.

6.2. Virtual Memory

The simple concept of virtual memory is that each program has its own address space, which is divided into chunks called pages. Each page is a series of range of addresses. These pages are allocated physical memory, but not all pages require physical memory to run the program. When the program references a part of the address space that resides in physical memory, the hardware immediately performs the necessary mapping. When the program points to a portion of its address space that is not in physical memory, the operating system is alerted to find the missing part and reconnect the failed instruction. With virtual memory, instead of a single movement for text and data items, it can map the entire address space into relatively small units for physical memory.

6.3. Cache Memory

The concept of caches is similar to virtual memory in that an active part of slow memory is stored in duplicate in a high-speed cache. When a memory request is generated, the request is sent to cache first, and if the cache cannot respond, the request is sent to main memory. The difference between cache and virtual memory is a matter of implementation; both concepts are conceptually the same because both are based on the correlation properties observed in the address reference sequences. The cache implementation is completely different from the virtual memory implementation due to cache speed requirements.

Caches can also be used to store content on disk (disk cache) or storage (for example, tape cache). pages on a web server. The cache is usually much faster and much smaller than the memory whose content is cached. Like any part of the memory hierarchy, it is only useful if it can meet many of the larger memory references.