Module Code & Module Title

CS5001NA Networks and Operating System

Assessment Weightage & Type

20% Individual Coursework

Year and Semester

2019-20 Autumn

Student Name: Siddhartha Ghimire

London Met ID: 19031691

College ID: NP01CP4A190148

Assignment Due Date: 11th April 2021

Assignment Submission Date: 11th April 2021

Word Count (TASK B): 2163

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline 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.

Table of Contents

Table of Figures

Table of Tables

Task A

Introduction

Bash is known as the ‘Bourne Again Shell’ (educative, 2019). It is descended from the ‘Thompson Egg’ and then from the Bourne ‘sh’ shell (educative, 2019). Bash has other ‘siblings’ (for example, ksh), ‘cousins’ (for example, tcsh), and ‘children’ (for examle, zsh) (educative, 2019). Bash is the most well-known, commonly used, and widely available shell. A shell program is one that requires the user to instruct the machine about what to do. Bash is a command-line shell application.

A shell script is a computer program that is intended to be executed by the Unix/Linux shell. Shell scripts must have several constructs that instruct the shell environment about what to do and when to do it.

Script

```
#!/bin/bash
if [ $# = 2 ]
then
if [[ $1 =~ [Aa-Zz] && $2 =~ ^[0-9]+$ ]] #passing two parameters $1 which contain
letters and $2 which conatins number
then
echo "----------------------------------------------------------------------------------------------------
--"
echo -e "\t\tYou are required to enter the program name here ==> \c"
read program
else
echo -e "Alert!\nInvalid Name or ID. Please enter a valid First Name or ID."
exit
fi
else
echo -e "Enter your first name as the first parameter and ID as the second parameter"
exit
fi
Total=4 #providing four attempts to the secret key
SecretKeyVerify(){
echo -e "Enter a secret key ==> \c"
read -s secretkey
until [ $secretkey = "siddhartha" ]
```

```
do
if [ $Total -lt 1  ]
then
exit
else
echo "Secret key did not matched.Try Again.You have got $Count attempts left."
((Total--)) #reducing every attempt by -1
SecretKeyVerify
fi
done
}
SecretKeyVerify
echo -e "**********\n\t\t\t\tWELCOME ==> $program" #Greeting with the program name
echo -e "\t\t\t\t\tID ==> $2"
echo -e "\t\t\t\t\tFirst Name ==> $1"
Today=$(date +"Year: %Y, Month: %m, Day: %d")
echo -e "\t\t\t\t$Today"
ChooseCountry(){
echo -e "\t-------------------------------------------------------------------"
echo -e "\t-------------------------------------------------------------------"
echo -e "\t\tHere are five countries team that play football"
echo -e "\n\t\t\tName:Brazil\tCode:BRZ"
echo -e "\t\t\tName:Argentina\tCode:ARG"
echo -e "\t\t\tName:Nepal\tCode:NEP"
echo -e "\t\t\tName:China\tCode:CHI"
echo -e "\t\t\tName:England\tCode:ENG"
echo -e "\nGuess and type the code for one of the top five football teams ==> \c"
read Code
CountryCode=${Code,,}
until [[ $CountryCode == "arg" || $CountryCode == "brz" || $CountryCode == "nep" ||
$CountryCode == "chi" || $CountryCode == "eng" ]]
do
echo "Invalid country code, please try again."
ChooseCountry
done
case "$CountryCode" in
"arg")
echo -e "\nCorrect Answer!\n==> Argentina has 14 Copa America titles to its credit, as
well as\n    the FIFA Confederations Cup in 1992. The country's Olympic\n
representative has two gold medals."
;;
"brz")
echo "Incorrect answer, please try again."
ChooseCountry;;
"nep")
echo "Incorrect answer, please try again."
```

```
ChooseCountry;;
"chi")
echo "Incorrect answer, please try again."
ChooseCountry;;
"ENG")
echo "Incorrect answer, please try again."
ChooseCountry;;
esac
}
ChooseCountry
PlayerSelect(){
echo -e "\t--------------------------------------------------------------------"
echo -e "\t--------------------------------------------------------------------"
echo -e "\t\tHere are five outstanding football players from the unit."
echo -e "\n\t\t\tName:Lionel Messi\tCode:LM"
echo -e "\t\t\tName:Neymar Junior\tCode:NJ"
echo -e "\t\t\tName:Kiran Chemjong\tCode:KC"
echo -e "\t\t\tName:Zheng Zhi\t\tCode:ZZ"
echo -e "\t\t\tName:Harry Kane\t\tCode:HK"
echo -e "\nEnter three of the five codes given ==> \c"
read player1 player2 player3
players=($player1 $player2 $player3)
if [[ ${#players[@]} -eq 3 && $player1 =~ [Aa-Zz] && $player2 =~ [Aa-Zz] && $player3
=~ [Aa-Zz] ]]
then
if [[ $player1 = $player2 || $player2 = $player3 || $player1 = $player3 ]]
then
echo "You cannot enter the same player multiple times."
PlayerSelect
else
echo -e "\t--------------------------------------------------------------------"
fi
else
echo "Error! Please enter code for respective three players."
PlayerSelect
fi
}
PlayerSelect
PlayerChoose(){
PS3='Please enter a number from the list above ==> '
select Player in ${player1^^} ${player2^^} ${player3^^}
do
if [ -z $Player ]
then
echo "You did not choose the number from the above list"
PlayerChoose
```

```
else
echo "The chosen player is $Player"
fi
break
done
ReProgram(){
echo "There is no external file of the player that you choose. Try Again"
PlayerSelect
PlayerChoose
}
case "$Player" in
"LM")
if [ -f lm.txt ] # -f finds the file lm.txt within the same directory
then
cat lm.txt #if found then run the file using cat
else
ReProgram #else call the method ReProgram again
fi;;
"NJ")
if [ -f nj.txt ]
then
cat nj.txt
else
ReProgram
fi;;
"KC")
if [ -f kc.txt ]
then
cat kc.txt
else
ReProgram
fi;;
"ZZ")
if [ -f zz.txt ]
then
cat zz.txt
else
ReProgram
fi;;
"HK")
if [ -f hk.txt ]
then
cat hk.txt
else
ReProgram
fi;;
```

```
esac
}
PlayerChoose
echo -e "\n-------------------------------------------------------------------"
close(){
echo -e "Do you want to repeat the program again yes/no ? ==> \c"
read Answer
if [ $Answer = "yes" ]
then
PlayerSelect
PlayerChoose
close
elif [ $Answer = "no" ]
then
exit
else
echo "Please answer either yes or no"
close
fi
}
close
```

Testing

Test 1

Objective	When a user runs the program without entering username, an error message is shown.
Action	Executing the program with bash shell, program_name and id.
Expected Result	A message should appear prompting to enter the username.
Actual Result	A message appeared asking for the username.
Conclusion	Test Successful

Table 1: Test 1

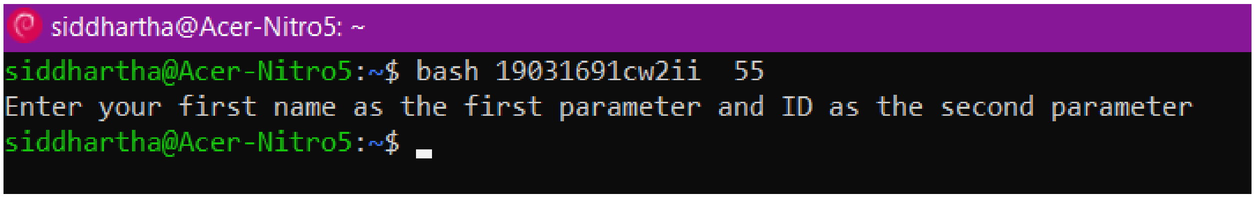


Figure 1: Test 1

Test 2

Objective	The program executes while the user runs it with the username and id.
Action	Executing the program with bash shell, program_name, username and id.
Expected Result	The program should run and prompt user for the program name.
Actual Result	The program ran successfully and asked for the program's name.
Conclusion	Test Successful

Table 2: Test 2

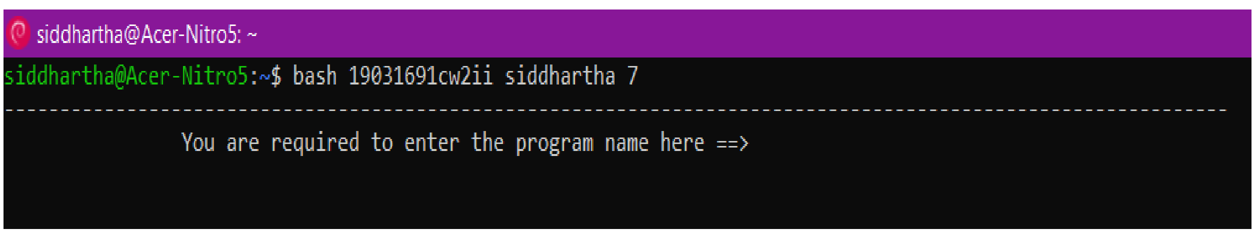


Figure 2: Test 2

Test 3

Objective	The program terminates while the user runs it with incorrect passwords for five times.
Action	Executing the program with incorrect password for five times.
Expected Result	The program should terminate while the user executes it with incorrect password for five times.

Actual Result	The program was successfully terminated.
Conclusion	Test Successful

Table 3: Test 3

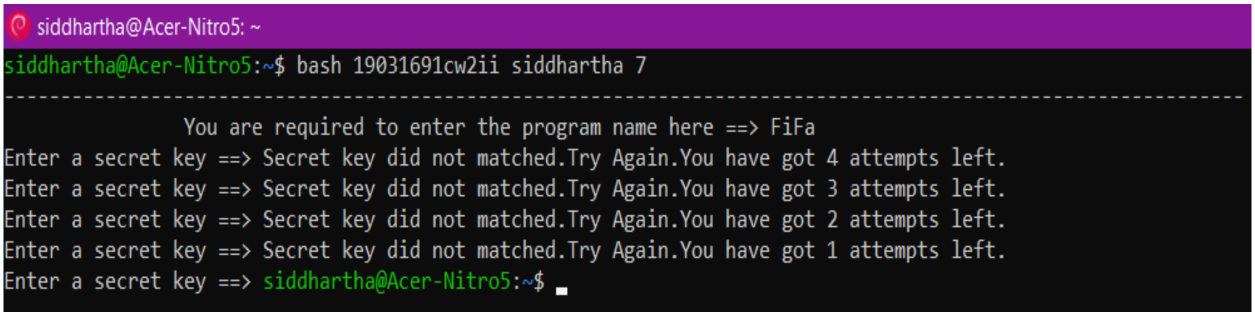
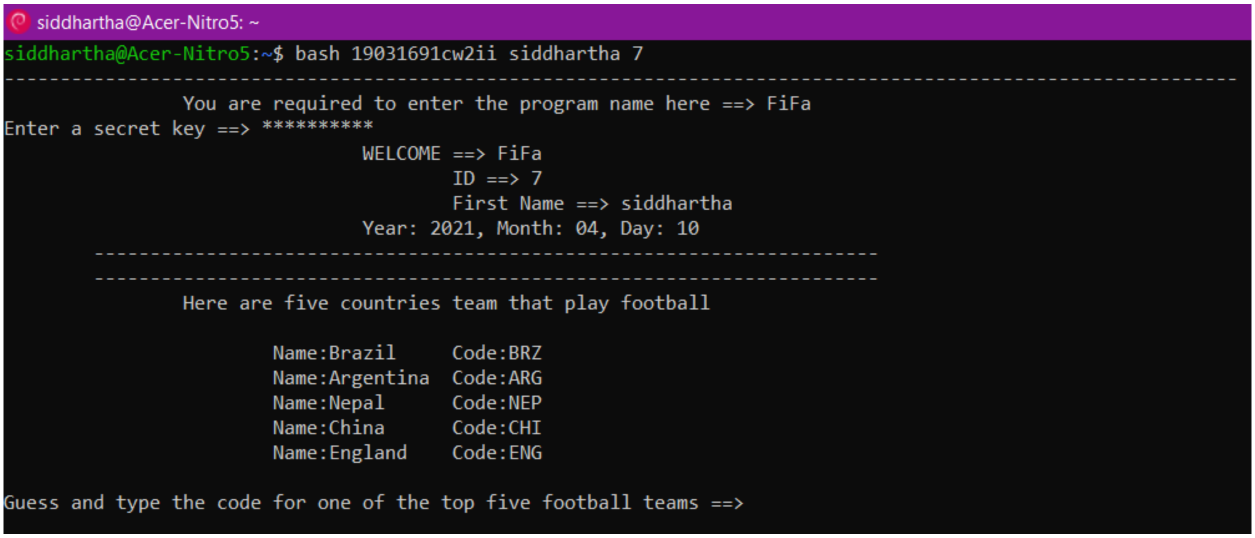


Figure 3: Test 3

Test 4

Objective	The program executes while the user runs it with correct password.
Action	Executing the program with correct password.
Expected Result	The program should greet the user with their program name, id, first name, date, and country code when the user executes it with the correct password.
Actual Result	The program was successfully executed, greeting the user with their program name, id, first name, date and country code when the user executes it with the correct password.
Conclusion	Test Successful

Table 4: Test 4



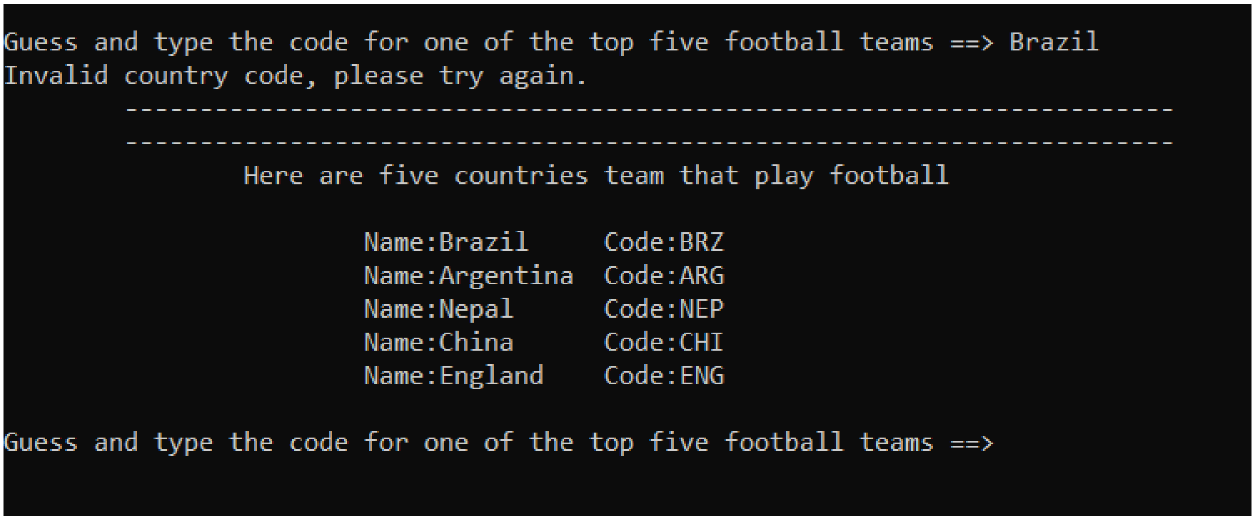
```
siddhartha@Acer-Nitro5: ~
siddhartha@Acer-Nitro5:~$ bash 19031691cw2ii siddhartha 7
--------------------------------------------------------------------------------------------------------------
                You are required to enter the program name here ==> FiFa
Enter a secret key ==> **********
                                WELCOME ==> FiFa
                                        ID ==> 7
                                        First Name ==> siddhartha
                                Year: 2021, Month: 04, Day: 10
        ----------------------------------------------------------------------
        ----------------------------------------------------------------------
                Here are five countries team that play football

                        Name:Brazil     Code:BRZ
                        Name:Argentina  Code:ARG
                        Name:Nepal      Code:NEP
                        Name:China      Code:CHI
                        Name:England    Code:ENG

Guess and type the code for one of the top five football teams ==>
```

Figure 4: Test 4

Test 5

Objective	The program repeats again while the user input with country name.
Action	Executing the program with country name.
Expected Result	The program should display the message to enter country code and repeat the program again.
Actual Result	The program showed a message instructing the user to enter the country code and then repeated the procedure.
Conclusion	Test Successful

Table 5: Test 5



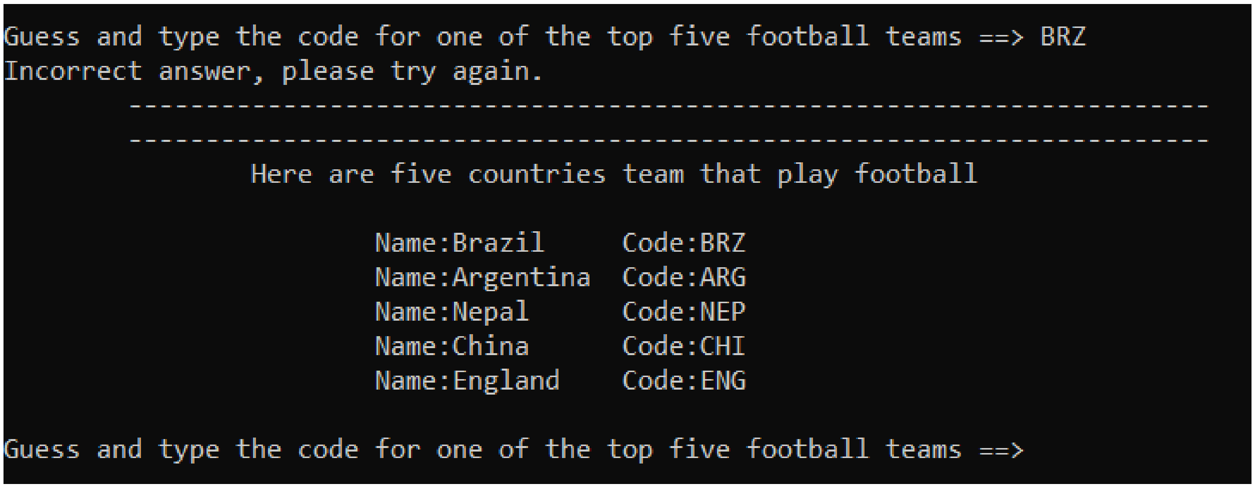
```
Guess and type the code for one of the top five football teams ==> Brazil
Invalid country code, please try again.
        ----------------------------------------------------------------------
        ----------------------------------------------------------------------
                Here are five countries team that play football

                        Name:Brazil      Code:BRZ
                        Name:Argentina   Code:ARG
                        Name:Nepal       Code:NEP
                        Name:China       Code:CHI
                        Name:England     Code:ENG

Guess and type the code for one of the top five football teams ==>
```

Figure 5: Test 5

Test 6

Objective	The program repeats again when the user input incorrect country code.
Action	Executing the program with incorrect country code.
Expected Result	The program should display the message to enter country code and repeat the program again.
Actual Result	The program showed a message instructing the user to enter the country code again and then repeated the procedure.
Conclusion	Test Successful

Table 6: Test 6



```
Guess and type the code for one of the top five football teams ==> BRZ
Incorrect answer, please try again.
        ----------------------------------------------------------------------
        ----------------------------------------------------------------------
                Here are five countries team that play football

                        Name:Brazil     Code:BRZ
                        Name:Argentina  Code:ARG
                        Name:Nepal      Code:NEP
                        Name:China      Code:CHI
                        Name:England    Code:ENG

Guess and type the code for one of the top five football teams ==>
```

Figure 6: Test 6

Test 7

Objective	The program executes with the short description and players with their code.
Action	Executing the program with correct country code.
Expected Result	The program should execute with short description of country and players with their code.
Actual Result	The program executed successfully with short description of its country and players with their code.
Conclusion	Test Successful

Table 7: Test 7

```
Guess and type the code for one of the top five football teams ==> ARG

Correct Answer!
==> Argentina has 14 Copa America titles to its credit, as well as
    the FIFA Confederations Cup in 1992. The country's Olympic
    representative has two gold medals.
        ----------------------------------------------------------------------
        ----------------------------------------------------------------------
                Here are five outstanding football players from the unit.

                        Name:Lionel Messi       Code:LM
                        Name:Neymar Junior      Code:NJ
                        Name:Kiran Chemjong     Code:KC
                        Name:Zheng Zhi          Code:ZZ
                        Name:Harry Kane         Code:HK

Enter three of the five codes given ==>
```

Figure 7: Test 7

Test 8

Objective	The program repeats again when the user input four players separated by space.
Action	Executing the program with four player code.
Expected Result	The program should repeat again with the error to input only three player's code.
Actual Result	The program repeated successfully with the error to input only three player's code respectively.
Conclusion	Test Successful

Table 8: Test 8

```
Enter three of the five codes given ==> LM NJ KC ZZ
Error! Please enter code for respective three players.
        ----------------------------------------------------------------------
        ----------------------------------------------------------------------
                Here are five outstanding football players from the unit.

                        Name:Lionel Messi       Code:LM
                        Name:Neymar Junior      Code:NJ
                        Name:Kiran Chemjong     Code:KC
                        Name:Zheng Zhi          Code:ZZ
                        Name:Harry Kane         Code:HK

Enter three of the five codes given ==>
```

Figure 8: Test 8

Test 9

Objective	The program repeats again when the user input the same players separated by space.
Action	Executing the program with same player code.
Expected Result	The program should repeat again with the error not to input the same player's code multiple times.
Actual Result	The program repeated successfully with the error not to input the same player's code multiple times.
Conclusion	Test Successful

Table 9: Test 9

```
Enter three of the five codes given ==> LM LM LM
You cannot enter the same player multiple times.
        ----------------------------------------------------------------------
        ----------------------------------------------------------------------
                Here are five outstanding football players from the unit.

                        Name:Lionel Messi       Code:LM
                        Name:Neymar Junior      Code:NJ
                        Name:Kiran Chemjong     Code:KC
                        Name:Zheng Zhi          Code:ZZ
                        Name:Harry Kane         Code:HK

Enter three of the five codes given ==>
```

Figure 9: Test 9

Test 10

Objective	The program terminates when the user input wrong user id.
Action	Executing the program with wrong user id.
Expected Result	The program terminates with the error not to input invalid user id.
Actual Result	The program terminated successfully with the error not to input invalid name or user id.
Conclusion	Test Successful

Table 10: Test 10

```
siddhartha@Acer-Nitro5: ~
siddhartha@Acer-Nitro5:~$ bash 19031691cw2ii siddhartha 3c
Alert!
Invalid Name or ID. Please enter a valid First Name or ID.
siddhartha@Acer-Nitro5:~$ bash 19031691cw2ii siddhartha c3
Alert!
Invalid Name or ID. Please enter a valid First Name or ID.
siddhartha@Acer-Nitro5:~$
```

Figure 10: Test 10

Test 11

Objective	The program executes when the user input right user id.
Action	Executing the program with right user id.
Expected Result	The program executes with no error when the user input right user id.
Actual Result	The program executed successfully with no error when the user input right user id.
Conclusion	Test Successful

Table 11: Test 11

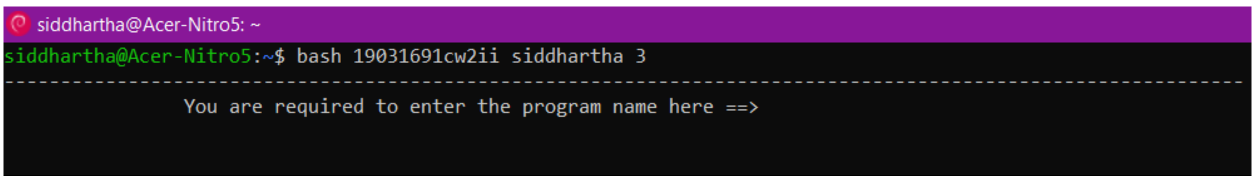


Figure 11: Test 11

Test 12

Objective	The program will rerun when the user enters a player id that does not exist in the file
Action	Executing the program with wrong player id.
Expected Result	The program should execute with no external file of player when user input the player id that does not exist in the file.
Actual Result	The program executed with error when user input the player id that does not exist in the file.
Conclusion	Test Successful

Table 12: Test 12

```
Please enter a number from the list above ==> 2
The chosen player is ZZ
There is no external file of the player that you choose. Try Again
        ----------------------------------------------------------------------
        ----------------------------------------------------------------------
                Here are five outstanding football players from the unit.

                        Name:Lionel Messi       Code:LM
                        Name:Neymar Junior      Code:NJ
                        Name:Kiran Chemjong     Code:KC
                        Name:Zheng Zhi          Code:ZZ
                        Name:Harry Kane         Code:HK

Enter three of the five codes given ==>
```

Figure 12: Test 12

Test 13

Objective	The program will repeat again when the user input yes.
Action	Entering yes to repeat the program.
Expected Result	The program should repeat with no errors.
Actual Result	The program executed successfully with no error.
Conclusion	Test Successful

Table 13: Test 13

```
---------------------------------------------------------------------
Do you want to repeat the program again yes/no ? ==> yes
        ----------------------------------------------------------------------
        ----------------------------------------------------------------------
                Here are five outstanding football players from the unit.

                        Name:Lionel Messi       Code:LM
                        Name:Neymar Junior      Code:NJ
                        Name:Kiran Chemjong     Code:KC
                        Name:Zheng Zhi          Code:ZZ
                        Name:Harry Kane         Code:HK

Enter three of the five codes given ==>
```

Figure 13: Test 13

Test 14

Objective	The program will exit when the user input no.
Action	Entering no to exit the program.
Expected Result	The program should exit with no errors.
Actual Result	The program terminated successfully with no error.
Conclusion	Test Successful

Table 14: Test 14

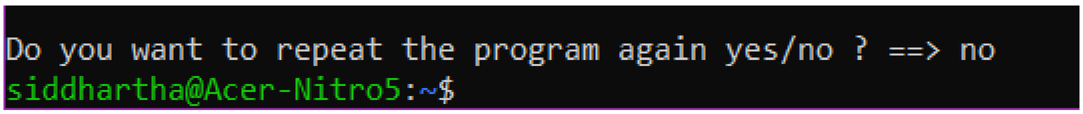


Figure 14: Test 14

Contents of three files: (TEXTS)

LM

Lionel Messi

Lionel Messi is an Argentine professional footballer who currently plays as a striker for FC Barcelona and the Argentina national team. He has won the FIFA Ballon d'Or six times (four of them consecutively) and an Olympic gold medal for Argentina's Olympic Football Team in 2008. Messi, who was born and raised in central Argentina, was diagnosed as an infant with a growth hormone deficiency. He moved to Spain at the age of 13 to join Barcelona, which promised to pay for his medical attention. Messi made his professional debut in October 2004 at the age of 17.

NJ

Neymar Junior

Neymar is a Brazilian professional footballer. He was born in Mogi das Cruzes, Brazil, on February 5, 1992. He is also an actress and producer, with credits including xXx: Return of Xander Cage (2017), Money Heist (2017), and SFR TV Commercial

(2019). With the Brazil Olympic Football Team, he won a silver medal in 2012 and a gold medal in 2016. He has a son called Davi Lucca born in 2011 with his ex-girlfriend Carol Dantas.

KC

Kiran Chemjong

He was born in the Nepalese town of Dhankuta. Chemjong joined the Machhindra Football Club after graduating from the ANFA Academy. Chemjong's outstanding results secured him a promotion to the Three Star Club in 2007 after just a year at the club. With Three Star Club, he has won several competitions, including the British Gurkha Cup and the Aaha Gold Cup Football Competition.

Conclusion

The program required by course work 2 has been successfully designed and evaluated. The bash shell (bash) is used to develop a program.

Task B

Introduction

Computer memory is one of the most critical elements in evaluating device output because it provides a space for programs to store and retrieve data in the short term. It preserves the information that the machine is currently using so that it can be recovered easily. General-purpose computer must hold many programs in memory to maximize CPU efficiency and the speed at which the computer reacts to its users, requiring the use of memory management (Silberschatz, et al., 2012)

Memory management is an integral component of any computer Operating System (OS), since it is the mechanism that manages and coordinates computer memory, assigning parts known as blocks to different running programs in order to maximize overall system performance (Roberto Doriguzzi Corin, 2016). Memory management generally refers to a number of operations such as memory allocation/deallocation, virtualization, security, mapping, switching, and some others (Roberto Doriguzzi Corin, 2016). Buffering, caching, and spooling are some memory-management components.

Memory-management algorithms range from a bare-machine approach to paging and segmentation techniques. Each method has advantages and drawbacks. The choice of a memory-management approach for a given device is influenced by a variety of variables, most notably the system's hardware architecture. Since many algorithms necessitate hardware support, many devices have tightly coupled hardware and operating-system memory management.

Built-in CPU registers are usually available within one loop of the CPU clock. Most CPUs are capable of translating instructions and executing basic operations on register contents at a rate of one or more operations per clock tick. The base register stores the smallest legal physical memory address, while the limit register determines the range dimension. For example, in the below figure the base register contains 300040 and the limit register contains 120900, the software has legal access to all addresses from 300040 to 420939 (inclusive) (Silberschatz, et al., 2012).

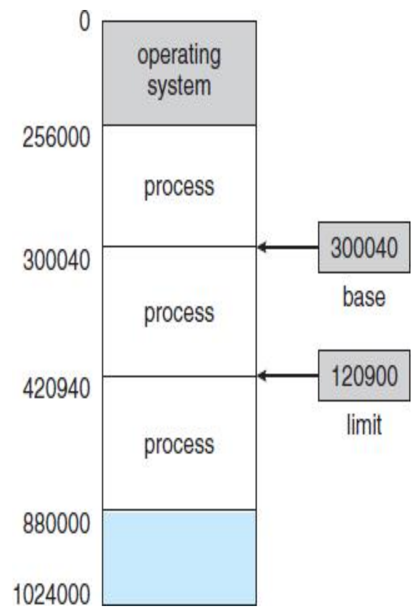


Figure 15: A logical address space is defined by a base and a limit register (Silberschatz, et al., 2012).

Every attempt by a program running in user mode to reach operating-system memory or other users' memory results in a trap to the operating system, which considers the attempt as a fatal mistake, as seen in the figure below (Silberschatz, et al., 2012).

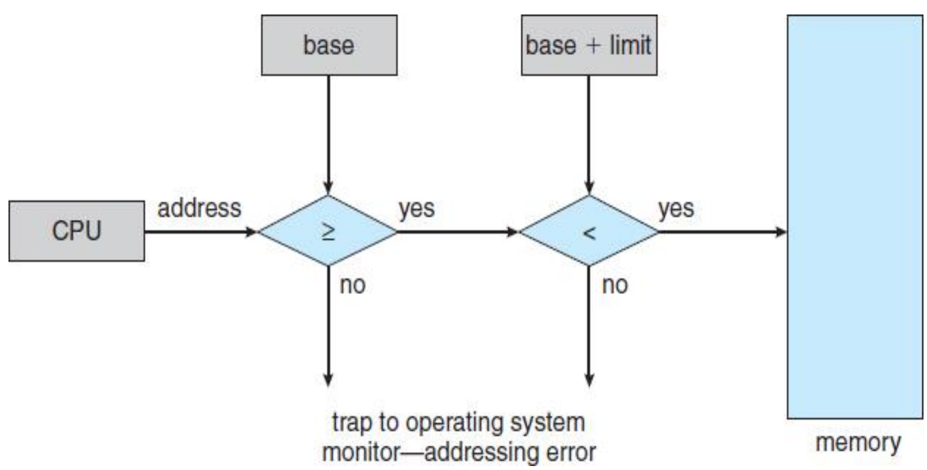


Figure 16 Base and limit registers have hardware address security (Silberschatz, et al., 2012).

This scheme forbids a user program from (inadvertently or intentionally) altering the operating system's or other users' code or data structures. Only the operating system can load the base and limit registers using a special privileged instruction. Since privileged instructions can only be executed in kernel mode, and only the operating system can execute in kernel mode. The operating system will change the value of the registers, but user applications cannot change the contents of the registers (Silberschatz, et al., 2012).

Aims and Objectives

The goal and purpose of this task is to comprehend the fundamental principle of memory control in an operating system. Furthermore, it goes into depth about physical memory, memory positioning, page coloring, paging and segmentation, page size difference, and so on. This task would also include a brief overview of different methods of organizing memory hardware, an exploration of various strategies for allocating memory to processes, and a thorough discussion of how paging operates in modern computer systems.

Background

Physical Memory

The RAM (Random Access Memory) of the device, which is connected to the motherboard, is referred to as physical memory. Primary memory is the only data type that can be addressed explicitly by the CPU (Prerana Jain, 2019). The instructions for the program to run are stored in primary memory. When a machine needs memory to load an operating program or open a document, RAM is the first memory that is used. Physical memory is linear addressable memory, with memory addresses increasing linearly and each byte being directly addressable (Prerana Jain, 2019). Physical memory is used to assign processor information in a first-in-last-out (FILO) method. Physical memory is the system's primary memory. Any user application that will be run should be stored here.

A method can always be executed in main memory. When a method is not in operation, it can be temporarily switched out of main memory and taken back in for continued execution (Prerana Jain, 2019). The operating system and the user program are also allocated memory in the main memory. As a result, we distribute main memory in the most effective manner possible. The main memory is split into two sections: one for the operating system and one for user operations. The user region of memory can be either contiguous or non-contiguous (Prerana Jain, 2019). Every process in the contiguous memory allocation should remain in a single section of main memory. The most effective solution in main memory is to split it into fixed-size blocks, with each block having just one memory (Prerana Jain, 2019).

RAM modules are normally counted in nanoseconds (1000^{-3}), and physical disks in milliseconds (1000^{-1}). This means that physical memory is roughly 100,000 times faster than a regular physical disk. As a result, wherever possible, Windows and Windows Server keep the most commonly used memory pages in physical memory and rely on a disk only when necessary.

Memory Placement

When there are a number of holes wide enough to handle the newly generated operation, the operating system must also decide where to position incoming programs. Three memory placement techniques have been widely used for this purpose:

- First-fit - The process is inserted into the first available hole wide enough to fit it, enabling the operating system to make a swift decision (Wells, 2009).
- Best-fit - The process is inserted in the hole that best suits it, leaving less space unused but adding management overhead (searching for the best fit) and leaving small gaps that cannot be used by other systems (Wells, 2009).
- Worst-fit - The process is inserted in the largest hole possible, regardless of complexity, potentially leaving enough room for another program while generating unnecessary management overhead (searching for the worst fit) and possibly leaving smaller gaps that cannot be used by other systems (Wells, 2009).

Regardless of the memory placement technique used, a number of small gaps in memory can form over time that cannot be used by new processes because they are just too small (Wells, 2009). This is known as external fragmentation. To combat this, operating systems developed the ability to relocate partitions, essentially moving the gaps to the top of memory and merging them into a much wider empty memory area (Wells, 2009). Although this method, known as compaction, helps memory to be used more effectively, it also adds to the complexity of the memory management function (Wells, 2009). The extra device overhead needed to execute a compaction scheme is important (Wells, 2009).

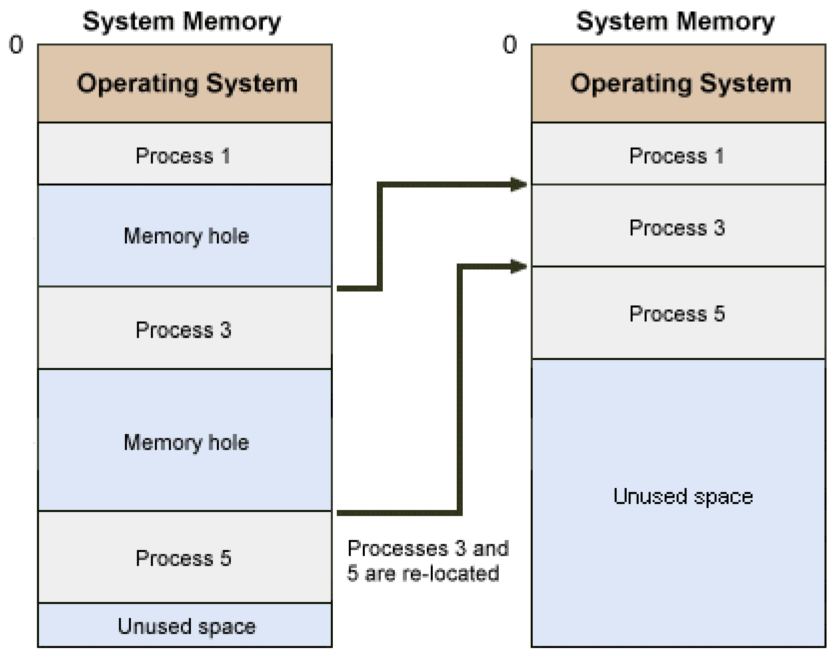


Figure 17: Memory compaction (Wells, 2009).

Page coloring

Page coloring is a program technique that rules the mapping of physical memory pages to cache blocks on a processor. Memory pages that map to the same cache blocks have the same color assigned to them (as illustrated by Figure 1) (liujunming, 2019).

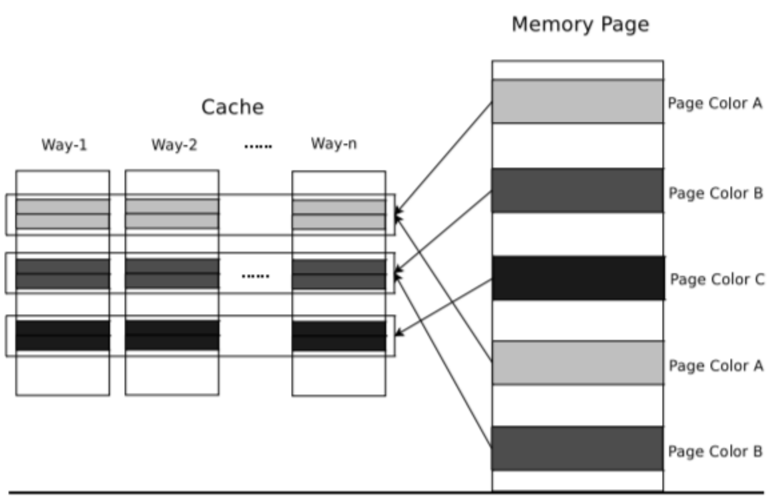


Figure 18: An illustration of the page coloring technique (liujunming, 2019).

The operating system's only approach in a physically handled cache without special hardware support to monitor cache sharing is to control the virtual to physical mappings used by individual processes (liujunming, 2019). Traditional page coloring [Taylor 1990] aims to delegate contiguous virtual memory pages to physical pages that would be distributed around the cache. To do this, contiguous physical memory pages are assigned separate colors, and contiguous virtual pages are guaranteed to be assigned distinct colors (liujunming, 2019).

A virtual memory subsystem that lacks cache coloring is less deterministic in terms of cache efficiency, since variations in page allocation from one program run to the next will result in significant differences in program performance (liujunming, 2019).

A different use of page coloring helps the operating system to limit a process's accesses such that it only uses a subset of the cache. On a multi-core platform, the shared cache space can thus be partitioned among multiple concurrently executing applications (liujunming, 2019).

Paging and Segmentation

Physical memory grew greater, but the memory space used for each process could vary greatly, and the key challenge for the operating system was attempting to find the most fitting memory space for a specific process. Some of the memory would inevitably be unused, and in many cases, a process would be too massive to fit into any of the available spaces (Wells, 2009).

One approach was to partition each process into a set of comparatively short fixed-size blocks known as pages, which could be loaded into memory as required. Memory was often organized into identically sized blocks known as page-frames (Wells, 2009). Each page could be loaded onto any available page frame, which ensured that user processes could use the entire available memory space, allowing for even more processes to be in memory at the same time. This was a much more effective use of computer memory, but it still placed a much greater workload on the operating system in terms of memory storage overhead, since it now had to keep track of several pages for each user operation (Wells, 2009).

One advantage of a paging scheme is that only a small subset of the pages dealing with a particular procedure must be loaded at any given time. In reality, only the page that is actually being executed needs to be kept in memory. When a process refers to a page that is not in main memory, a page fault is created, causing the operating system to load the missing page into memory from secondary storage (Wells, 2009).

The extent at which this happens is determined by the size of the page as well as the overall number of pages that form a procedure. In fact, there would be a minimal number of pages that must be kept in memory at all times for a given procedure. This is regarded as the minimum working range, and it is highly affected by the total scale of the user software (Wells, 2009).

Another method of utilizing memory, known as segmentation, is based on the concept of splitting a process into blocks, but the size of the blocks (or segments) varies. A scheme that enables the use of variable size segments can be helpful from the standpoint of a programmer because it allows for the development of modular programs; however, the operating system must also not only keep track of the beginning address of each segment, but must also measure the offset to the end of each segment because they are variable in size. Some programs merge paging and segmentation by making segments variable-size blocks made up of fixed-size pages (Wells, 2009).

Page size variation

The size of a page, which is a block of stored memory, is referred to as page size in computers. When a program starts, most operating systems decide the page size. This function allows it to determine the most effective usage of memory when the program is running (Silberschatz, et al., 2012).

The hardware support for paging is seen in the figure below. Per CPU-generated address is divided into two parts: a page number (p) and a page offset (d). The page number is used as an index into a table of page numbers (Silberschatz, et al., 2012). The base address of each file in physical memory is stored in the page table. The physical memory address that is sent to the memory device is described by combining this base address with the page offset. The memory paging model is depicted in Figure below (Silberschatz, et al., 2012).

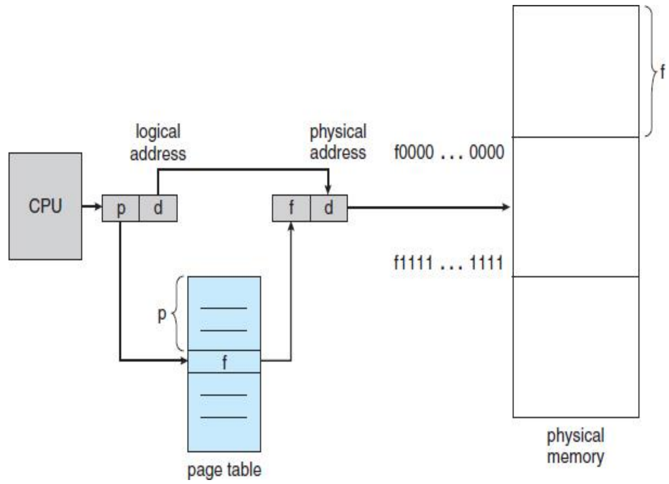


Figure 19 Paging Hardware (Silberschatz, et al., 2012).

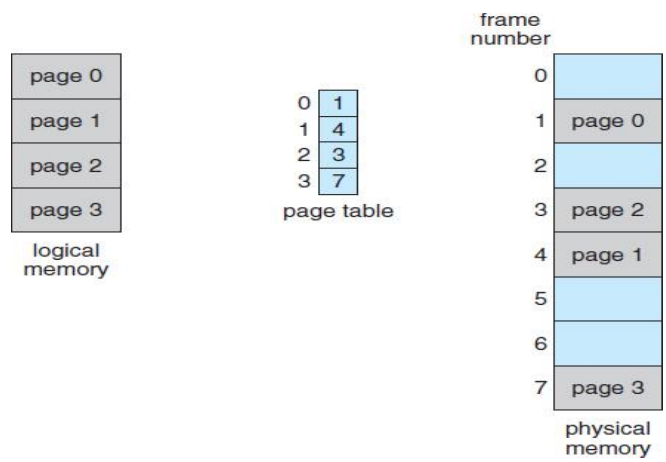


Figure 20 Paging model of logical and physical memory (Silberschatz, et al., 2012).

A page is a power of two in size, ranging from 512 bytes to 1 GB per page based on the device architecture. The use of a power of two as a page size facilitates the conversion of a logical address into a page number and page offset (Silberschatz, et al., 2012). If the logical address space is 2m bytes in size and a page is 2n bytes in size, then the high-order m n bits of a logical address designate the page number and the n loworder bits designate the page offset (Silberschatz, et al., 2012). As a result, the logical address is as follows:

Figure 21 Logical address (Silberschatz, et al., 2012).

where p is a web table index and d is the displacement inside the page.

Conclusion

Memory-management algorithms for multiprogrammed operating systems vary from basic single-user systems to segmentation and paging. Any memory address created by the CPU must be validated and potentially mapped to a physical address. The verification cannot be applied (effectively) in applications. Many features distinguish the different memory-management algorithms (contiguous allocation, paging, segmentation, and combinations of paging and segmentation).

As a result, this task was accomplished with much commitment and extensive analysis from various websites, journals, books, and so on.

References

educative, 2019. *What is Bash?.* [Online]
Available at: https://www.educative.io/courses/master-the-bash-shell/3j8399P3M6M
[Accessed 10 April 2021].

liujunming, 2019. *page coloring的介绍 - L.* [Online]

Available at: http://liujunming.top/2019/10/06/page-coloring%E7%9A%84%E4%BB%8B%E7%BB%8D/
[Accessed 07 April 2021].

Prerana Jain, 2019. Concept of physical and virtual memory in Operating System. *includehelp.com*, 03 January.

Roberto Doriguzzi Corin, D. S. E. S. A. S., 2016. *Empowering Network Operating Systems with Memory Management Techniques.* [Online]
Available at:
https://www.researchgate.net/publication/294522133_Empowering_Network_Operating_Systems_with_Memory_Management_Techniques
[Accessed 2 April 2021].

Silberschatz, A., Galvin, P. & Gagne, G., 2012. *Operating System Concepts.* 9th ed. 111 River Street, Hoboken: Wiley.

T.S.Adams, 2021. *EasyTechJunkie: Conjecture Corporation.* [Online]
Available at: https://www.easytechjunkie.com/what-is-the-memory-controller.htm
[Accessed 8 April 2021].

Wells, C. J., 2009. *Memory Management.* [Online]
Available at: https://www.technologyuk.net/computing/computer-software/operating-systems/memory-management.shtml
[Accessed 7 April 2021].

Appendix

Memory controller

Computers store active data in Random Access Memory (RAM) chips as they are running. RAM chips are inserted into a computer's motherboard and connected to the processor via the front side bus (T.S.Adams, 2021). They effectively serve as a direct route for the exchanging of variables and program data. The memory controller is a chip that is usually found on the motherboard's northbridge (T.S.Adams, 2021). It manages read and write operations with machine memory, as well as maintaining the RAM working by providing electric current to the memory.

RAM is typically faster than other forms of media, such as hard disks and optical disks. However, one of the drawbacks of RAM is that it needs a continuous flow of power to function (T.S.Adams, 2021). When the power supply is interrupted, the information contained in RAM chips is destroyed. This requirement is met by the memory controller, which "refreshes" the RAM at a steady rate while the cpu is turned on (T.S.Adams, 2021).

The memory controller sends a pulse of electronic current through the RAM chips during a "refresh" (T.S.Adams, 2021). The amount of current that is transmitted through RAM is determined by the computer's Binary Input Output System (BIOS). This happens at least once every 64 milliseconds, keeping the RAM working and the data contained inside safe from destruction due to power outages (T.S.Adams, 2021). Data would be destroyed in fractions of a second if the memory controller was not present.

The memory controller also handles RAM chip read and write operations. It determines which demultiplexer circuit to use for data storage and retrieval (T.S.Adams, 2021).

Virtual Memory

Virtual memory is a technique that enables the operation of operations that are not entirely contained within memory. The fact that applications can be bigger than physical memory is a significant benefit of this system (Silberschatz, et al., 2012). Furthermore, virtual memory abstracts main memory into an incredibly massive, uniform collection of storage, separating logical memory from real memory as used by the user (Silberschatz, et al., 2012). This approach relieves programmers of the burden of memory-storage constraints. Virtual memory also enables processes to quickly exchange files and incorporate shared memory. Furthermore, it provides an effective framework for process development. Virtual memory, on the other hand, is difficult to incorporate and, if used carelessly, will significantly reduce performance (Silberschatz, et al., 2012).

Cache Memory

Caching is a main premise in computing systems. Normally, materials are stored in a retrieval facility (such as main memory). When it is used, it is temporarily copied onto a faster retrieval unit. Internal programmable registers, such as index registers, also serve as a high-speed cache for main memory (Silberschatz, et al., 2012). The register-allocation and register-replacement algorithms are applied by the programmer (or compiler) to determine the details to hold in registers and which to keep in main memory (Silberschatz, et al., 2012). To control a cache built into the CPU, the hardware automatically accelerates memory access without the need for any operating-system interference (Silberschatz, et al., 2012).