LONDON METROPOLITAN UNIVERSITY

islington college

(इस्लिङ्टन कलेज)

Module Code & Module Title

CS5001NA Networks and Operating System

Assessment Weightage & Type

10% Individual Coursework

Year and Semester

2020-21 Autumn

Student Name: Bijay Bharati

Group: L2C4

London Met ID: 19030824

College ID: NP01CP4A190041

Assignment Due Date: 11th April 2021

Assignment Submission Date: 11th April 2021

Word Count (Task B): 2183

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.

Contents

Figures

Tables

Task A

Introduction

This is the task A of the coursework that deals with creation of a UNIX program. This is the first of the two tasks.

Here, we are creating a small UNIX program to interact with the user taking basic inputs and displaying the results. The program is made user friendly with clean interface and proper input validations that help the users understand what to do and what is happening in the program.

The program is created as per the instructions provided; the script is presented below along with various test cases to demonstrate the functionality of the program.

Script

Below is the code of the program.

```
#!/bin/bash

# checking if 2 arguments are passed, if the 2 arguments are passed, the
program continues
# it is assumed that name and id can be string
# in echo command \n is for new line, \t is for tab and  \e[numberm is used
for text color


if [ $# = 2 ]; then


# validation for program name, program name can only contain alphabets

    while ! [[ $programName =~ ^[Aa-Zz]+$ ]]; do
        echo -e "\nEnter the name you would like to give this program: \c"
        read programName
        if ! [[ $programName =~ ^[Aa-Zz]+$ ]]; then
            echo -e "\e[93mMake sure the name you entered does not contain
spaces, numbers or special characters\e[39m"
        fi
    done

# assigning values for different variables
# secret is the password
# attempts is to keep track of how many times wrong secret has been entered
# entry is for allowing the program to continue

    secret="a"
    attempts=0
    entry=0


# user will be asked to enter secret until it is correct, user has 4 tries

    while [ $attempts -lt 4 ]; do
        echo -e "\nWhat is the secret?"
        read input

        if [ "$input" = "$secret" ]; then

            entry=1
            break

        else

            attempts=$(expr $attempts + 1)
            echo -e "failure, attempts remaining: \e[91m$(expr 4 -
$attempts)\e[39m"

        fi
    done
```

```
# when user enters correct secret, entry=1 so that program continues

    if [ $entry == 1 ]; then
        echo -e
"\n______________________________________________________________________"
        echo -e "\n Welcome to \e[96m$programName\e[39m"
        echo -e "\n You are \e[92m$1\e[39m"
        echo -e "\n Your ID is \e[92m$2\e[39m"
        echo -e "\n This message was displayed on: \e[92m$(date)\e[39m"
        echo -e
"\n_______________________________________________________________________"

# this is to keep the program in loop

        while true; do
            echo -e "\n Now, you must guess which team is the best football
team."

            echo -e
"\n_______________________________________________________________________\n"
            echo -e "\t\t\tBRZ   Brazil"
            echo -e "\t\t\tARG   Argentina"
            echo -e "\t\t\tNEP   Nepal"
            echo -e "\t\t\tCHI   China"
            echo -e "\t\t\tENG   England"
            echo -e
"\n_______________________________________________________________________\n"

# it is assumed Brazil is best team, so it will ask user for best tem until
code for Brazil BRZ is entered

            until [[ ${countryCode^^} = "BRZ" ]]; do
                echo -e "Give country code: \c"
                read countryCode

# converting entered countru code to upper case for comparision

                countryC=${countryCode^^}

# checking if country code is among the valid codes

                if ! [[ $countryC =~ ^(BRZ|ARG|NEP|CHI|ENG)$ ]]; then
                    echo -e "\n\e[91mInvalid country code typed\e[39m\n"

# checking if country code is BRZ of not

                elif [[ $countryC =~ ^(ARG|NEP|CHI|ENG)$ ]]; then
                    echo -e "\n\e[93mGuess Again, this is not the best
football team\e[39m\n"

# when country code is BRZ

                else
```

```
                    echo -e "\n\e[92mCongratulations you selected the best
football team\e[39m\n"
                    echo -e
"\n______________________________________________________________________"
                    echo -e "\nBrazil is the top football team in the
world.\nThe top players in the world have been from Brazil.\nBrazil has won
the football world cup 5 times."
                    echo -e
"\n______________________________________________________________________"
                fi
            done

# selection for players

            echo -e "\nNow, select any 3 players"

            echo -e
"\n______________________________________________________________________\n"
            echo -e "\t\t\tLM  Lionel Messi"
            echo -e "\t\t\tNJ  Neymar Junior"
            echo -e "\t\t\tKC  Kiran Chemjong"
            echo -e "\t\t\tZZ  Zeng Zhi"
            echo -e "\t\t\tHK  Harry Kane"
            echo -e
"\n______________________________________________________________________\n"

            echo -e "Enter the player codes seperated by spaces"


# 3 inputs are taken for player

            while [ playerCount != 3 ]; do
                echo -e "\nEnter 3 player codes: \c"
                read p01 p02 p03
                p1=${p01^^}
                p2=${p02^^}
                p3=${p03^^}
                players=($p1 $p2 $p3)
                playerCount=${#players[@]}

# validations for proper player code and repeating codes

                if ! [[ $playerCount == 3 ]]; then
            echo -e "\n\e[93mPlease enter 3 player codes\e[39m\n"
        elif ! [[ $p1 =~ ^(LM|NJ|KC|ZZ|HK) ]]; then
                    echo -e "\n\e[91mOne or more invalid player codes
entered\e[39m\n"
                elif ! [[ $p2 =~ ^(LM|NJ|KC|ZZ|HK) ]]; then
                    echo -e "\n\e[91mOne or more invalid player codes
entered\e[39m\n"
                elif ! [[ $p3 =~ ^(LM|NJ|KC|ZZ|HK) ]]; then
                    echo -e "\n\e[91mOne or more invalid player codes
entered\e[39m\n"
                elif [[ "$p1" = "$p2" ]] || [[ "$p1" = "$p3" ]] || [[ "$p2"
== "$p3" ]]; then
                    echo -e "\n\e[93mPlayer codes repeated\e[39m\n"
                else
```

```
                    echo -e "\n\e[92mThank you for selecting three 
players\e[39m\n"
                    break
                fi
            done
            echo -e 
"\n_______________________________________________________________________\n"

            echo -e "\nChoose the player you would like to know more about\n"

# asking to choose 1 player to know more about, the players p1, p2 and p3 are 
taken from above

            PS3="Enter number (1-3): "
            select player in $p1 $p2 $p3; do
                if [[ -z "$player" ]]; then
                    echo -e "\n\e[93mInvalid choice, try again\e[39m\n"
                else

# the file is checked if it exists and is readable
# the file name is same as player code and the value of player code is used 
to find the file

                    echo -e "\n\e[92myou choose $player\n\e[39m"
                    echo -e "\nThe player details from the player's file is 
printed below\n"
                    if ! [ -f $player ]; then
                        echo -e "\n\e[91mError: File not found\e[39m\n"
                    elif ! [ -r $player ]; then
                        echo -e "\n\e[91mError: File not readable\e[39m\n"
                    else
                        echo -e 
"\n______________________________________________________________________\n"
                        echo "$(cat $player)"
                        echo -e 
"\n______________________________________________________________________\n"
                    fi
                    break
                fi
            done

# loop to repeat program, if y is typed, program repeats, else it exits

            echo -e "\nPress Y if you want to continue: \c"
            read option
            if [[ $option = "Y" ]] || [[ $option = "y" ]]; then
                echo -e "\n\e[96mThe Program Repeats\e[39m"

                countryCode=''
            else
                echo -e "\n\e[96mThe program has been stopped\e[39m"
                break
            fi
        done

# message for failed attempts
```

```
    else
        echo -e "\n\e[91mToo many failed attempts, you may not enter\e[39m"

    fi

# message for improper name, id


else

    echo -e "\n\e[93mMake sure that you enter name and ID properly\e[39m \n"
    echo -e "Here is an example: bash cw2 \e[92m\"Jon Doe\" 1290312\e[39m\n"

fi
```

Testing

Test: run without username and id

Test No	1
Input	The program was run without providing username and id as parameters.
Expected Output	The user should be alerted about the error and proper method should be mentioned.
Actual Output	The user was alerted and shown how to add username and id.
Test Result	Test is successful.

Table 1: Test 1

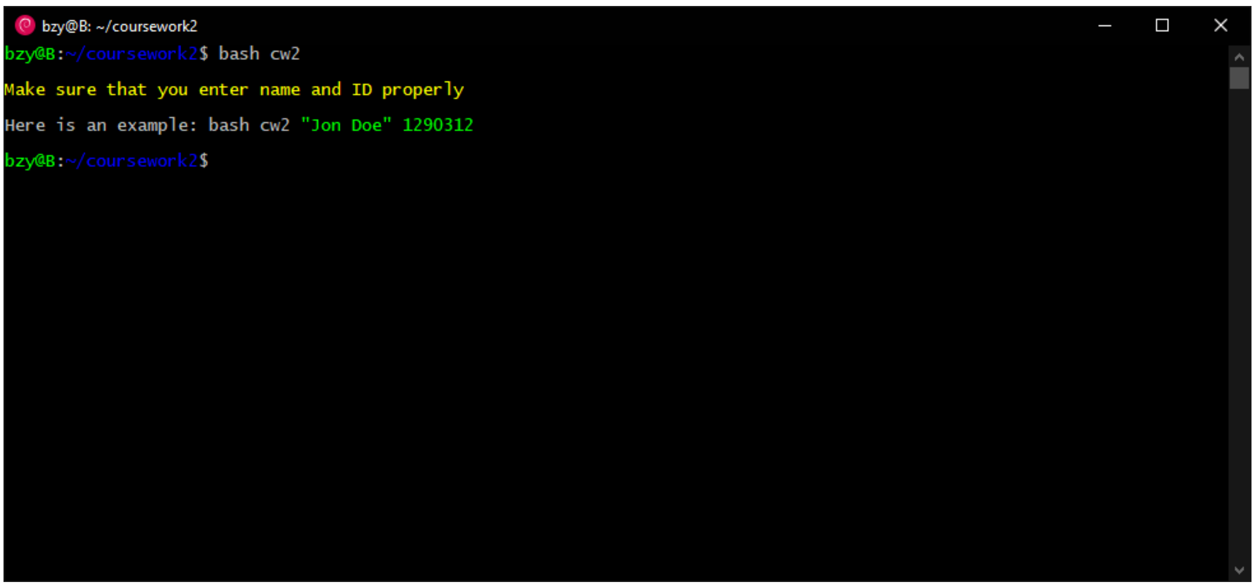


Figure 1: Running the program without username and id

Test: run with correct username and id

Test No	2
Input	The program was run with proper username and id in the parameters.
Expected Output	The program should work properly and ask the user to enter the name for the program.
Actual Output	The user was prompted to enter program name.
Test Result	Test is successful.

Table 2: Test 2

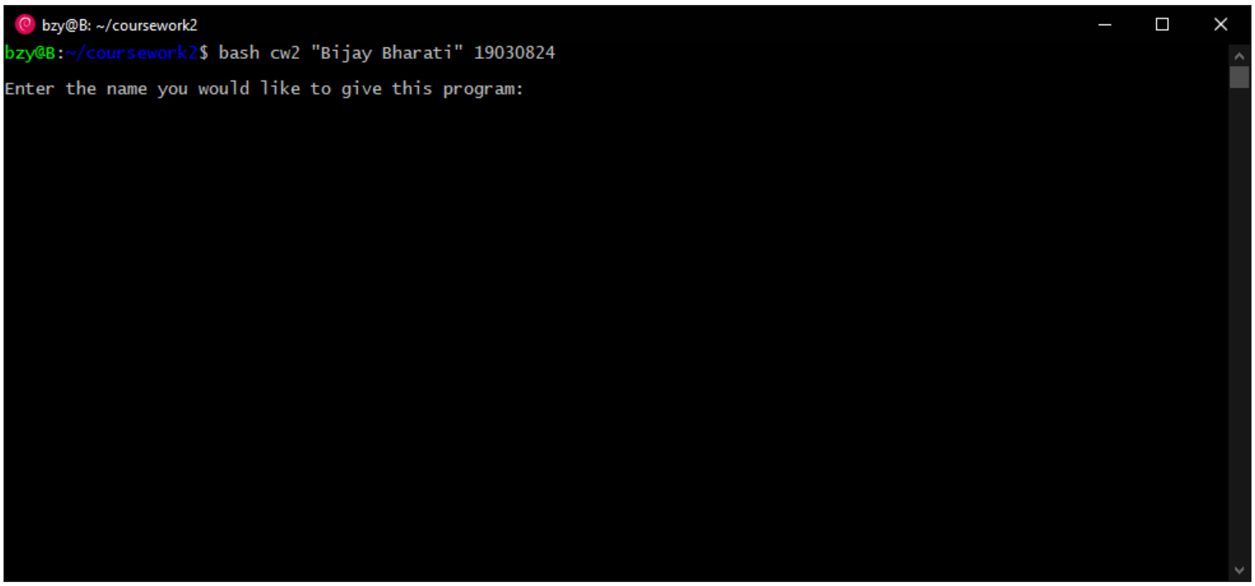


Figure 2: Running the program with username and id

Test: run with incorrect password/ secret

Test No	3
Input	Incorrect password/ secret text was typed when running.
Expected Output	The user should not be allowed to enter after 4 failed attempts.
Actual Output	Access was denied after 4 failed attempts.
Test Result	Test is successful.

Table 3: Test 3

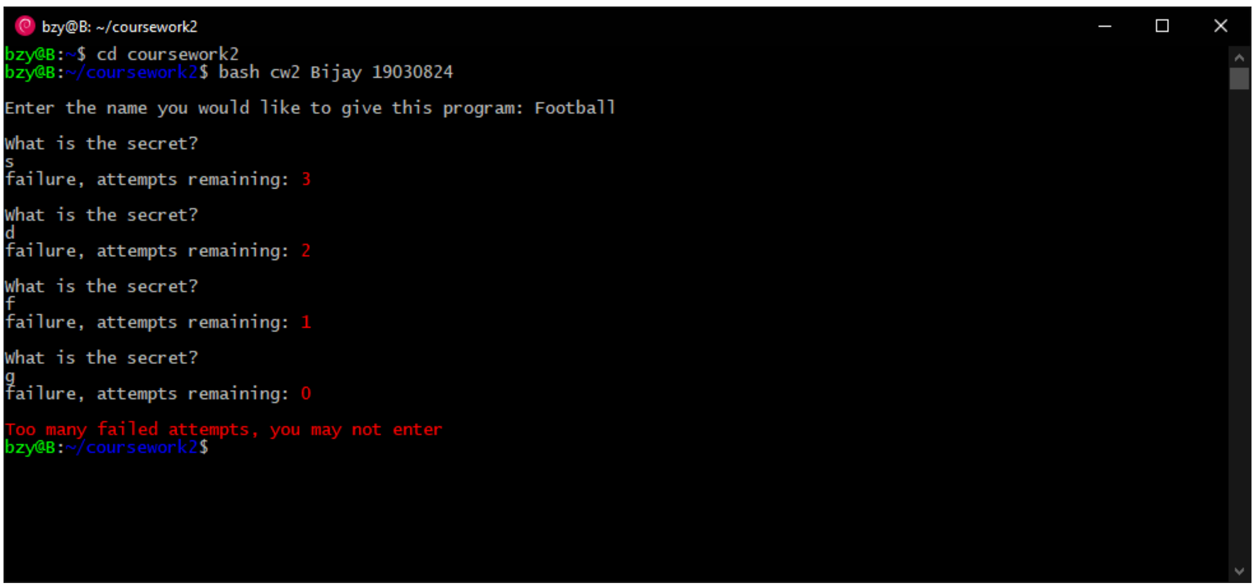


Figure 3: Running with incorrect password

Test: run with correct password

Test No	4
Input	Correct password/ secret text was typed when running.
Expected Output	The user should be welcomed to the program and the next phase should be executed
Actual Output	The user was welcomed to the program and the program continues.
Test Result	Test is successful.

Table 4: Test 4

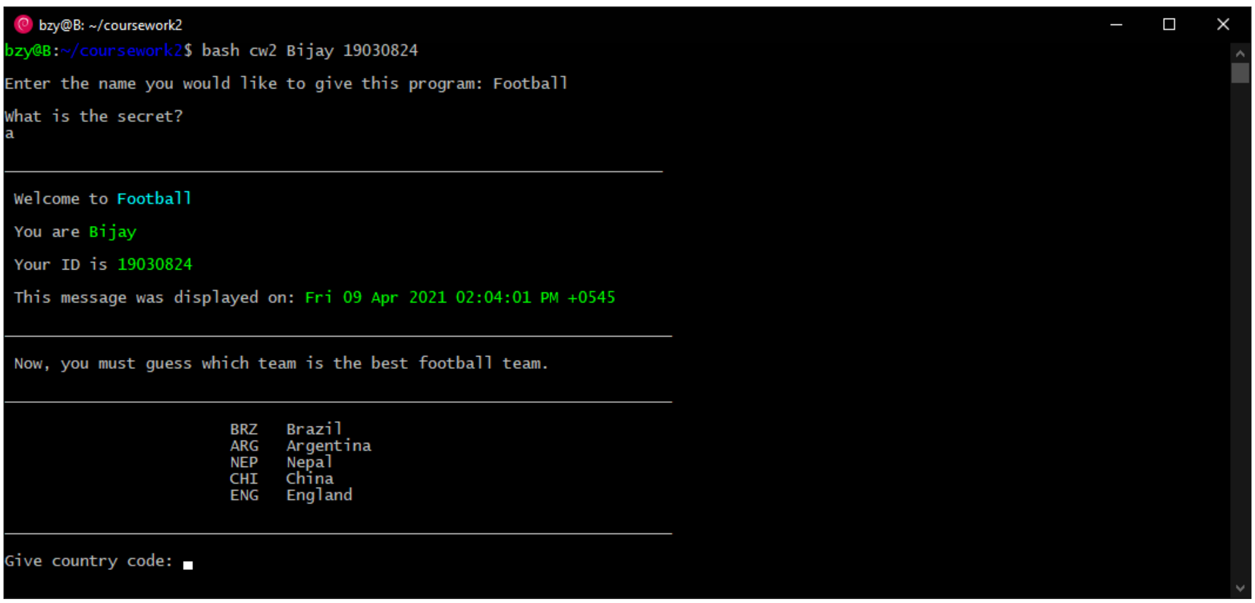


Figure 4: Running with correct password

Test: run with invalid country code

Test No	5
Input	Invalid country code was typed when prompted to guess country with best football team.
Expected Output	The user should be informed about the invalid code and prompted to enter the code again.
Actual Output	The user was informed that the country code is invalid and asked to type again.
Test Result	Test is successful.

Table 5: Test 5

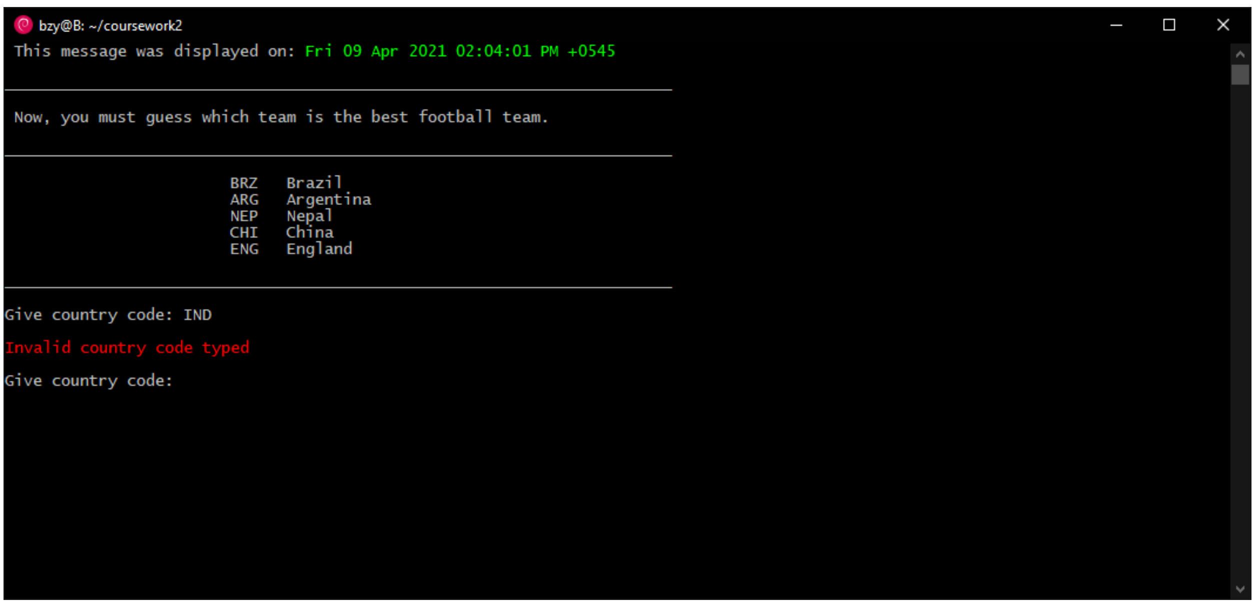


Figure 5: Typing invalid country code

Test: run with valid country code, but not the right guess

Test No	6
Input	Valid country code was typed but it is not the best team.
Expected Output	The user should be prompted to guess again which is the best team.
Actual Output	The user was prompted to guess which is the best team again.
Test Result	Test is successful.

Table 6: Test 6

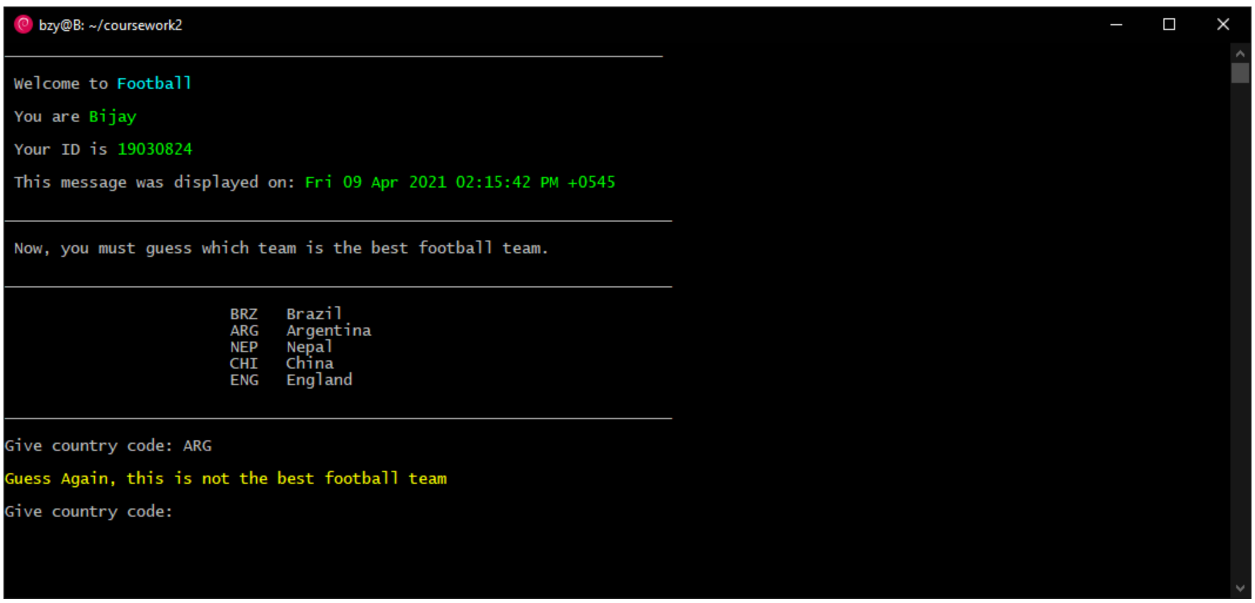


Figure 6: Incorrect guess for the best football team

Test: run with valid country code with the right guess

Test No	7
Input	Valid country code was typed which was the code for the best team.
Expected Output	Short description of the team should be given, and the user should be prompted to give the code for 3 players.
Actual Output	The user was given short description of the team and asked to enter 3 player codes.
Test Result	Test is successful.

Table 7: Test 7

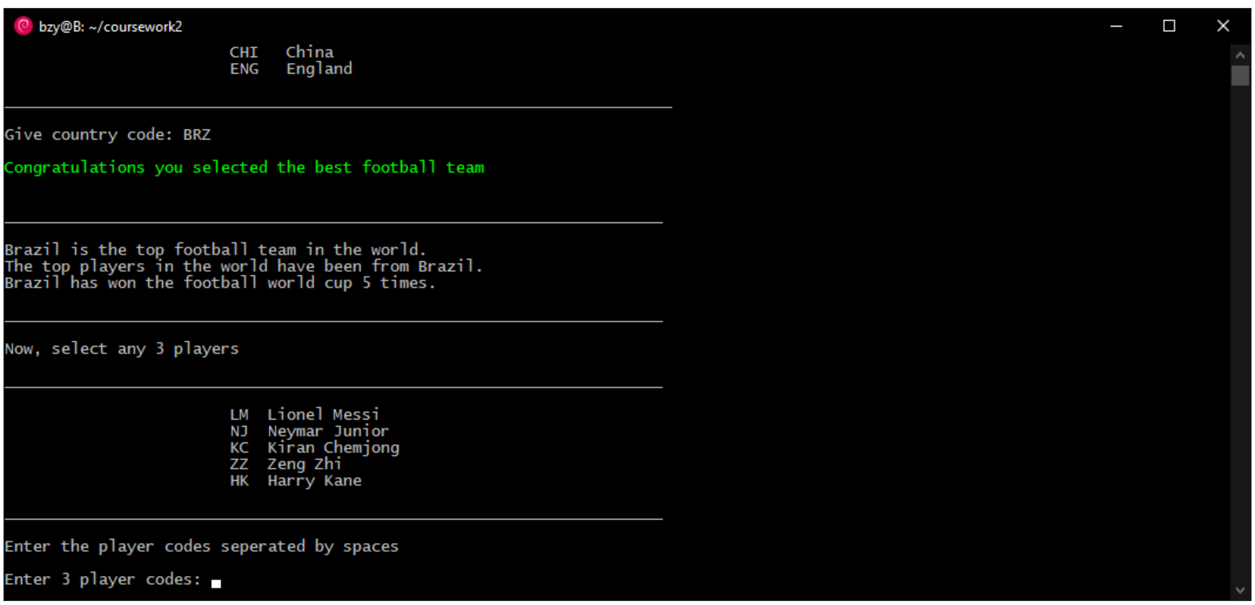


Figure 7: Correct guess for the best football team

Test: invalid player codes were typed

Test No	8
Input	Invalid player codes were typed.
Expected Output	User should be notified about the error and asked to enter player codes again.
Actual Output	The user was notified about the error and asked to enter player codes again
Test Result	Test is successful.

Table 8: Test 8

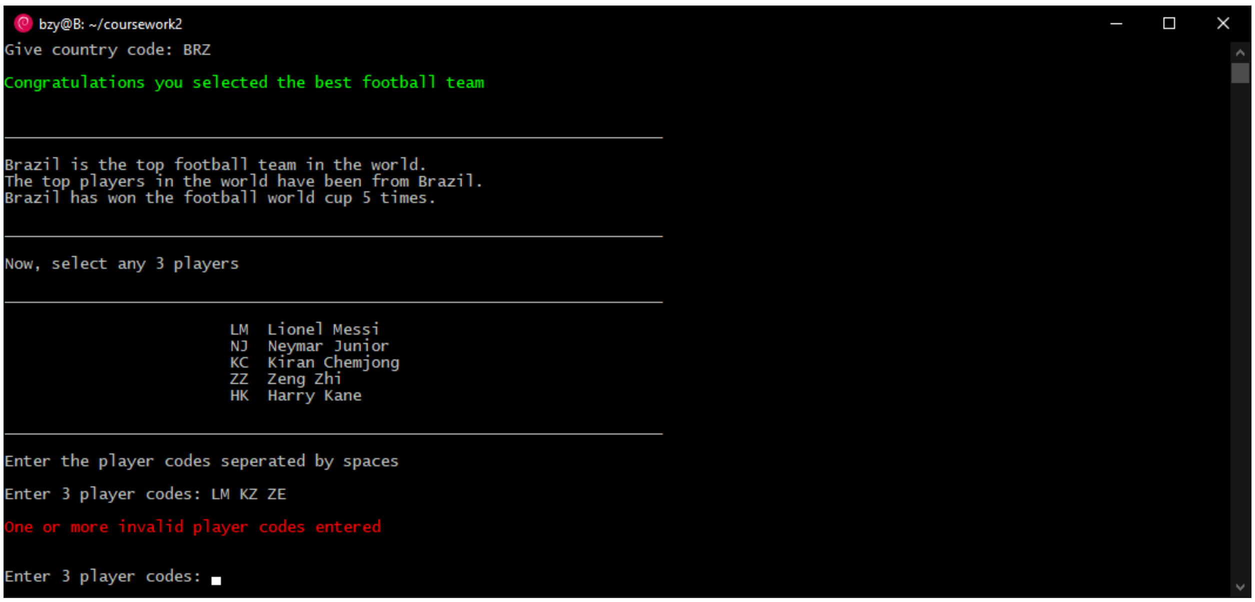


Figure 8: Entering invalid player codes

Test: duplicate player codes were typed

Test No	9
Input	Same player code was typed multiple times.
Expected Output	User should be notified about the error and asked to enter player codes again.
Actual Output	The user was notified about the error and asked to enter player codes again
Test Result	Test is successful.

Table 9: Test 9

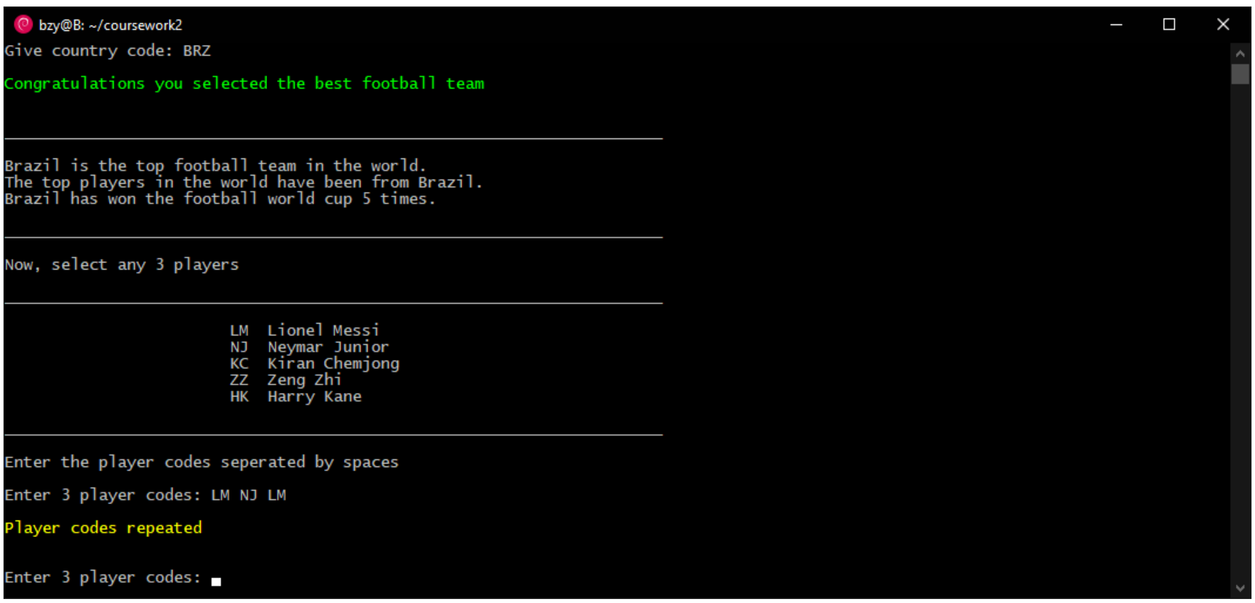


Figure 9: Repeating player codes

Test: unique player codes were typed

Test No	10
Input	3 unique and valid player codes were typed
Expected Output	The user should be prompted to select any one of the players to know more about.
Actual Output	The user was prompted to select any one of the players to know more about.
Test Result	Test is successful.

Table 10: Test 10

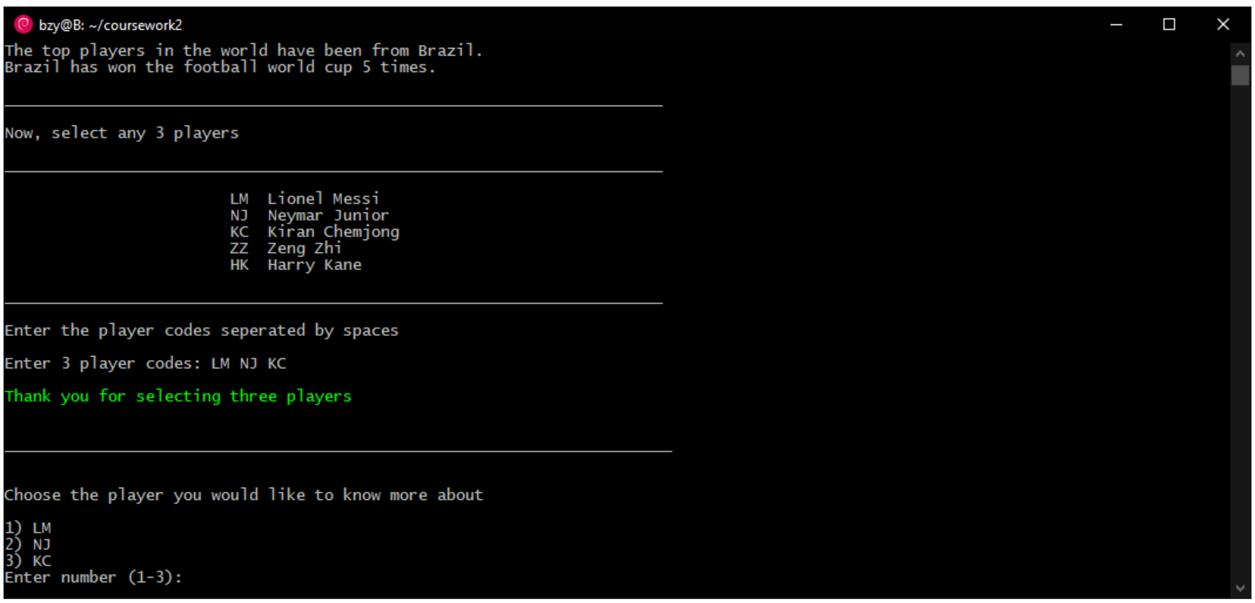


Figure 10: 3 unique and valid player codes entered

Test: invalid player selection made

Test No	11
Input	Invalid choice was made when asked to choose a player.
Expected Output	The user should be notified about the error and prompted to choose again.
Actual Output	The user was notified about the error and prompted to choose again.
Test Result	Test is successful.

Table 11: Test 11

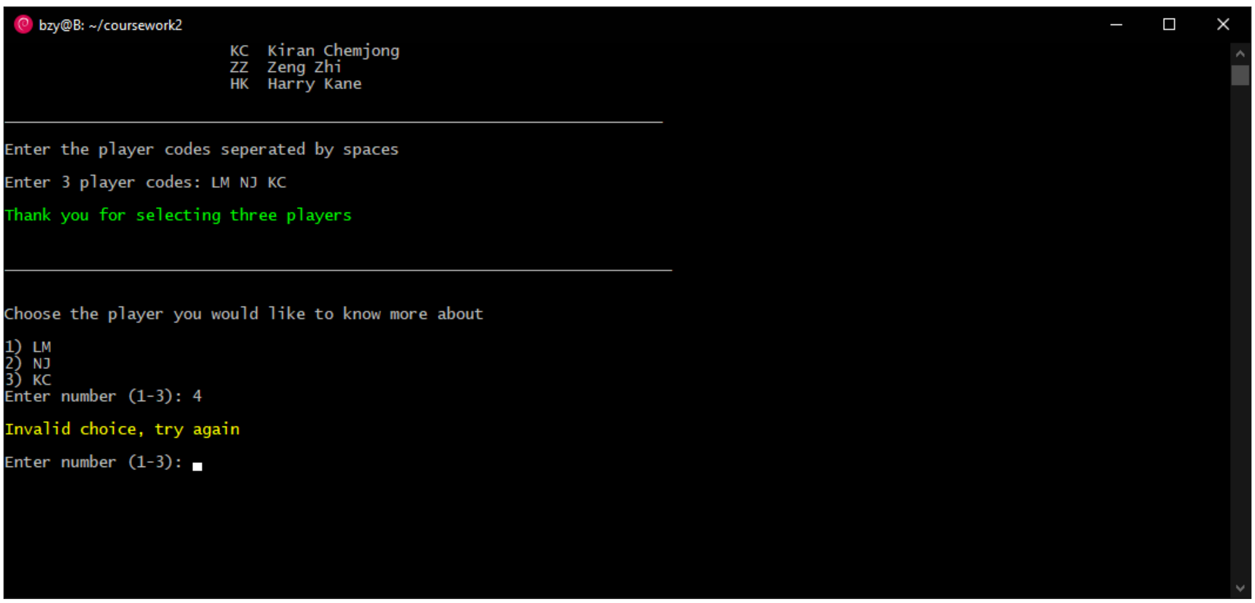


Figure 11: Invalid player choice:

Test: valid player selection made

Test No	12
Input	Valid choice was made when asked to choose a player.
Expected Output	Information about the player from the correct file should be displayed.
Actual Output	Information from correct file was displayed.
Test Result	Test is successful.

Table 12: Test 12

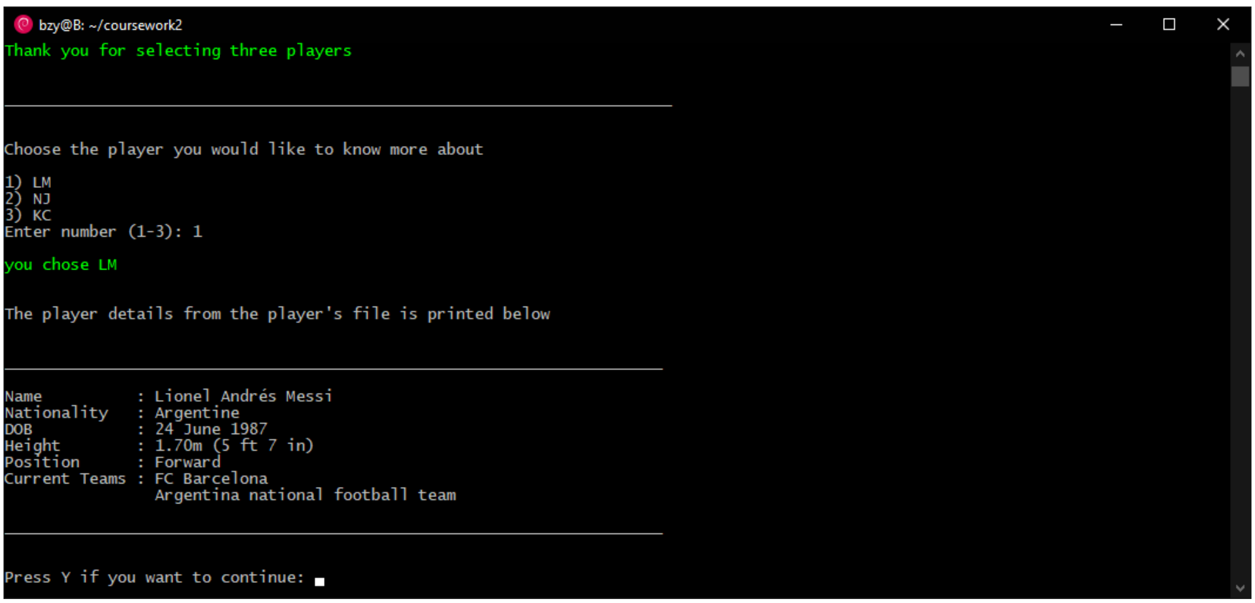


Figure 12: Correct player choice:

Test: file with player information is not found

Test No	13
Input	Valid choice was made when asked to choose a player but required file was not found.
Expected Output	Proper error message should be displayed.
Actual Output	Error message file not found was displayed.
Test Result	Test is successful.

Table 13: Test 13

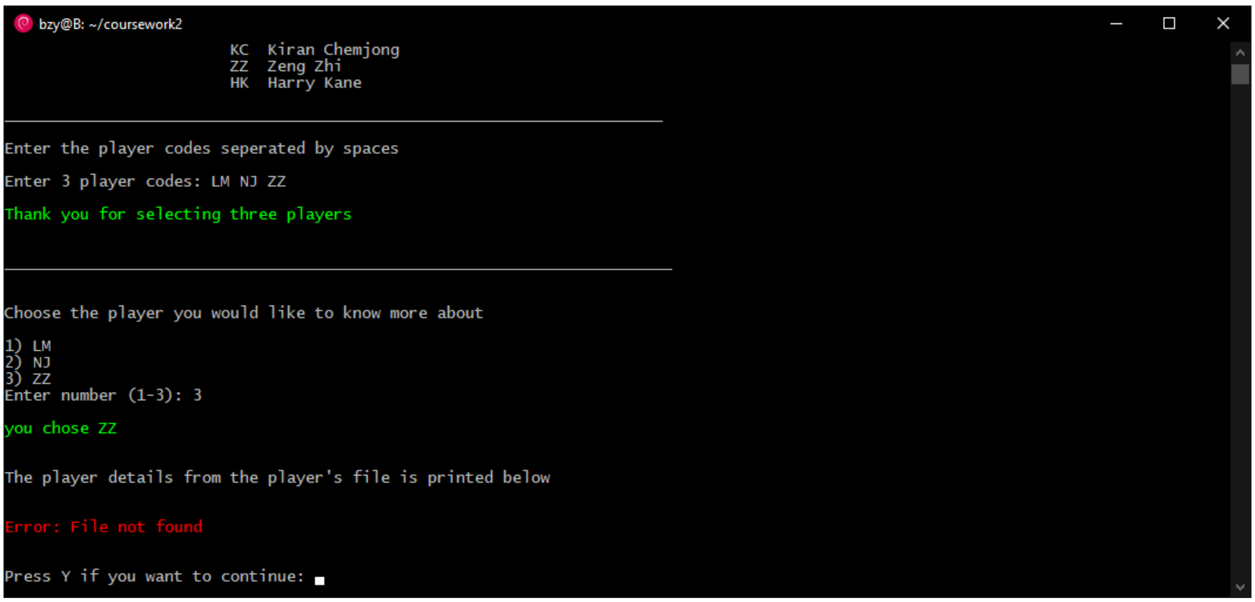
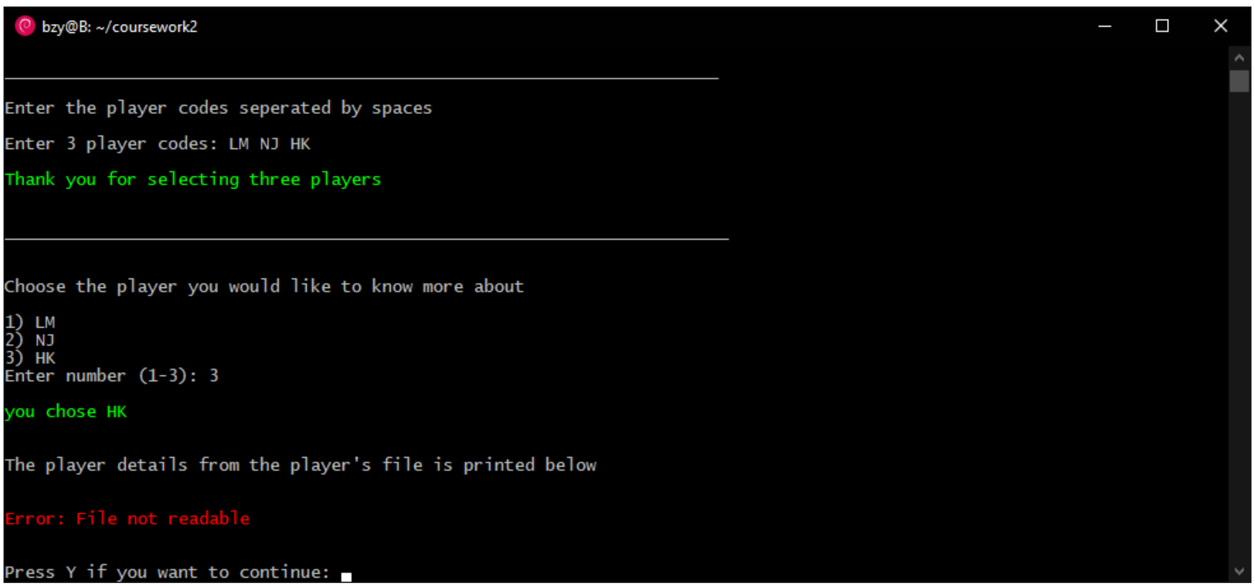


Figure 13: File not found

Test: file with player information is not readable

Test No	14
Input	Valid choice was made when asked to choose a player but required file was not readable.
Expected Output	Proper error message should be displayed.
Actual Output	Error message file not readable was displayed.
Test Result	Test is successful.

Table 14: Test 14


```
bzy@B: ~/coursework2
________________________________________________
Enter the player codes seperated by spaces
Enter 3 player codes: LM NJ HK
Thank you for selecting three players

________________________________________________

Choose the player you would like to know more about
1) LM
2) NJ
3) HK
Enter number (1-3): 3
you chose HK

The player details from the player's file is printed below

Error: File not readable

Press Y if you want to continue:
```

Figure 14: File not readable

Test: Y is typed when asked to continue

Test No	15
Input	Y is typed when asked to continue.
Expected Output	The program should repeat from the point when it asks to guess country code for best team.
Actual Output	The program repeats from the point when it asks to guess country code for best team.
Test Result	Test is successful.

Table 15: Test 15

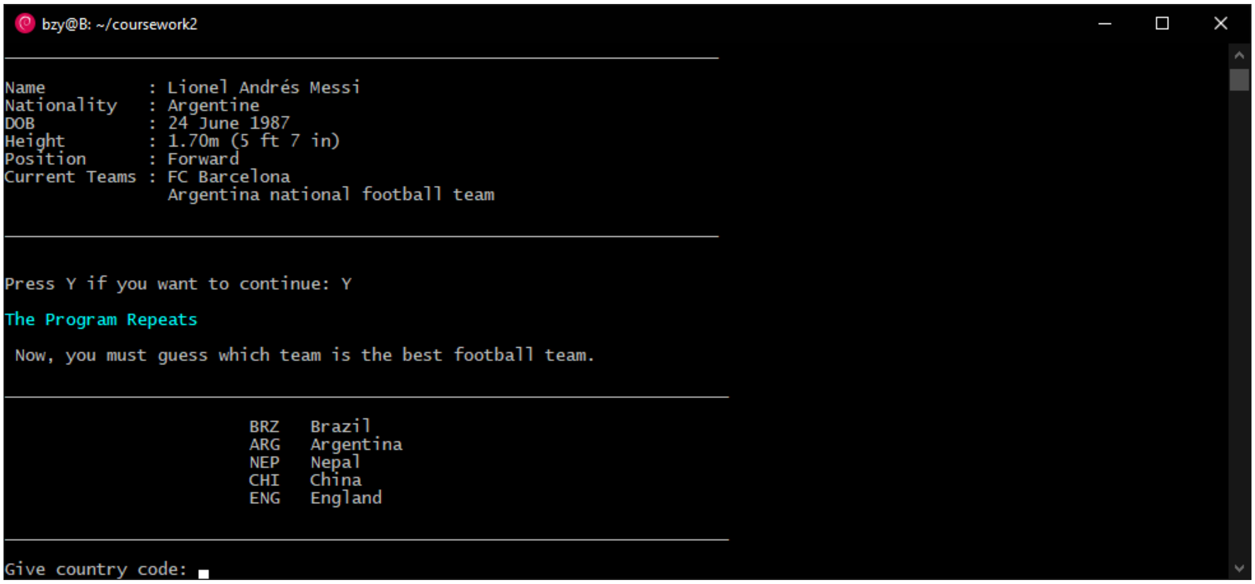


Figure 15: Y is typed

Test: Y is not typed when asked to continue

Test No	16
Input	Y is not typed when asked to continue.
Expected Output	The program should exit.
Actual Output	The program exits.
Test Result	Test is successful.

Table 16: Test 16

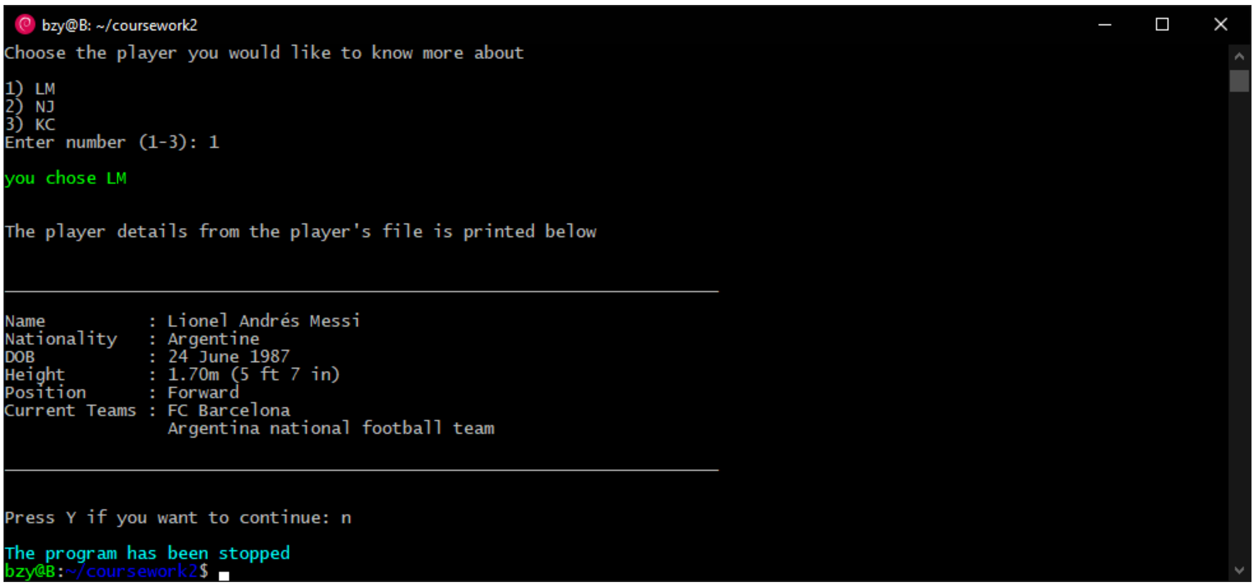


Figure 16: Y is not typed when asked to continue

Content of the files

LM file contents

Name : Lionel Andrés Messi
Nationality : Argentine
DOB : 24 June 1987
Height : 1.70m (5 ft 7 in)
Position : Forward
Current Teams : FC Barcelona
Argentina national football team

NJ file contents

Name : Neymar da Silva Santos Júnior
Nationality : Brazillian
DOB : February 5, 1992
Height : 1.75 m (5 ft 9 in)
Position : Forward
Current Teams : Paris Saint-Germain
Brazil

KC file contents

Name : Kiran Chemjong Kumar Limbu
Nationality : Nepalese
DOB : 24 March 1990
Height : 6 ft 2 in (1.88 m)
Position : Goalkeeper
Current Teams : RoundGlass Punjab FC
Nepal

Conclusion

The programming portion was successfully completed and tested to remove errors and provide proof of functionality. Debian for windows using windows sub system for Linux was used to develop the program in bash shell, bash was used to run the program.

The task has been successful in helping to understand loops and other programming concepts like manipulating user input and handling errors.

Task B

Introduction

This part of the report focuses on research work. Computer memory is the core of the report. The report will be about general topics related to computer memory like physical memory, memory placement, memory issues etc. it is to be noted that when talking about memory, this report is not about storage media like hard disks or flash drives, the memory here relates to RAM and other memory used by the computers to function. This section is meant as an introduction to the vast world of computer memory.

Aims and Objective

The objective here is to help students understand the way computers work and how memory is used and to make familiar with researching and writing technical reports. This is a required skill, to be able to search for information and make sense of the information.

Background

When we talk about computer memory, it generally refers to the primary memory or the main memory. It is the memory that is directly accessible to the CPU. The CPU uses the main memory to execute instructions and run programs, any program that e use is first loaded in the main memory.

Memory used in our computer should be fast and large so that we do not face problems when using large applications. The memory system is constructed as a hierarchy of layers. The table below shows a typical hierarchy (the numbers are rough approximations.

Table 17: Memory hierarchy (Tanenbaum, 2015)

Typical access time		Typical capacity
1	Registers	< 1 KB
2	Cache	4 MB
10	Main memory	1– 8 GB
10	Magnetic disk	1 – 4 TB

The top layer consists of the registers internal to the CPU. They are made of the same material as the CPU and are thus just as fast as the CPU. Consequently, there is no delay in accessing them. The storage capacity available in them is typically 32 × 32 bits on a 32-bit CPU and 64 × 64 bits on a 64-bit CPU. Less than1 KB in both cases. Programs must manage the registers (i.e., decide what to keep in them) themselves, in software. (Tanenbaum, 2015).

The cache is the high-speed memory directly accessible to the CPU. There can be many levels of cache memory, each level is larger but slower. In a windows pc, the size of cache can be seen in the task manager. This memory holds the most executed tasks and information for the processor so that it is readily available.

Next is main memory. Main memory refers to RAM (Random Access Memory). The higher speed and capacity RAM a system has, the smother it runs. Before RAMs, tiny magnetic memory called core memory were used but they have been obsolete for decades. All the applications are loaded in the RAM before execution. RAM is volatile and the data is lost when power is switched off. There is also ROM (Read Only Memory) which holds the instructions to run when booting up. ROM is generally permanent and read only but can be modified by force and programmable ROMs are also available. There is also another memory called CMOS (Complementary Metal-Oxide-Semiconductor). The CMOS memory has its own battery and it stores date, BIOS settings and some hardware settings.

Magnetic disks are secondary memory and they hold data that we want to save. There are multiple types of secondary memory like tapes, sd cards, flash drives, CDs etc but the report is not concerned with secondary memory, so this topic is not discussed more.

Physical Memory

Physical memory in a computer refers to the actual RAM in the system. RAM can be considered the main work force in the computer as all tasks are loaded in RAM.

RAM has 2 main types:

- Dynamic RAM (DRAM) uses a transistor and a capacitor to represent and store a bit of data. The charge needs to be refreshed every few seconds. (BBC, 2021)
- Static RAM (SRAM) uses a group of transistors combined for each bit of data. They do not lose the charge while in use so SRAM is much faster than DRAM. SRAM technology can be non-volatile when used for flash memory in storage and ROM. (BBC, 2021)

DRAM stores data by “writing a charge to the capacitor by way of an access transistor” and was invented in 1966 by Robert Dennard at IBM and was patented in 1967. DRAM looks at the state of charge in a transistor-capacitor circuit a charged state is a 1 bit; the low charge is seen as a 0 bit. (Thrnton, 2017)

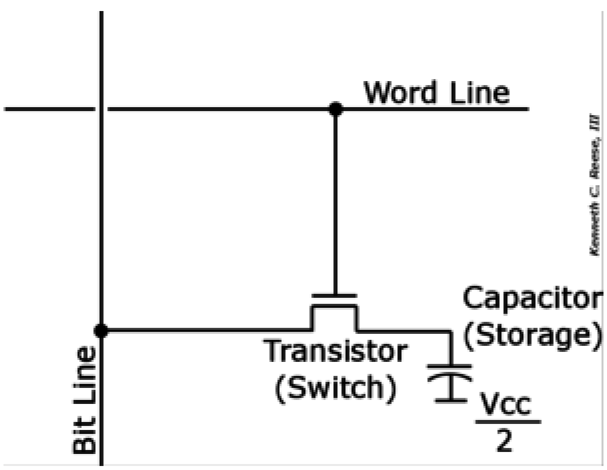


Figure 17:DRAM stores one bit as memory using a transistor and a capacitor. (credit: Kenneth C. Reese, III)

Several of these transistor-capacitor circuits together creates a “word” of memory.

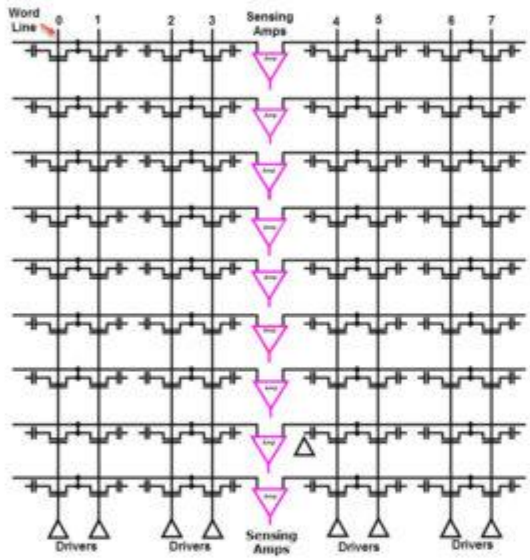


Figure 18: Array of DRAM cells

DRAM capacitors can loose the charge that they hold that is why they are volatile. DRAMs need constant supply of power and need to be refreshed frequently. (Thrnton, 2017)

SRAM does not use capacitors. SRAM uses several transistors in a cross-coupled flip-flop configuration and does not have the leakage issue and does not need to be refreshed. (Thrnton, 2017)

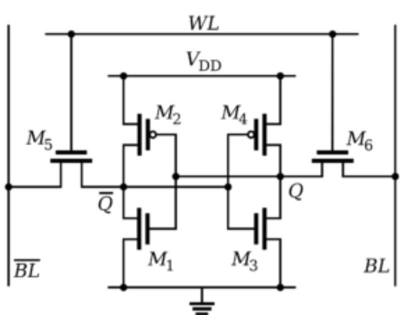


Figure 19: SRAM cell with six transistors. (Credit: Inductiveload [Public domain], via Wikimedia Commons).

SRAM is faster but more expensive and is used for cache. Our systems use DRAM for main memory because they are cheaper and larger in memory size compared to SRAM.

We can see the basic information about the RAM that our system uses through task manager if we are using a windows system.

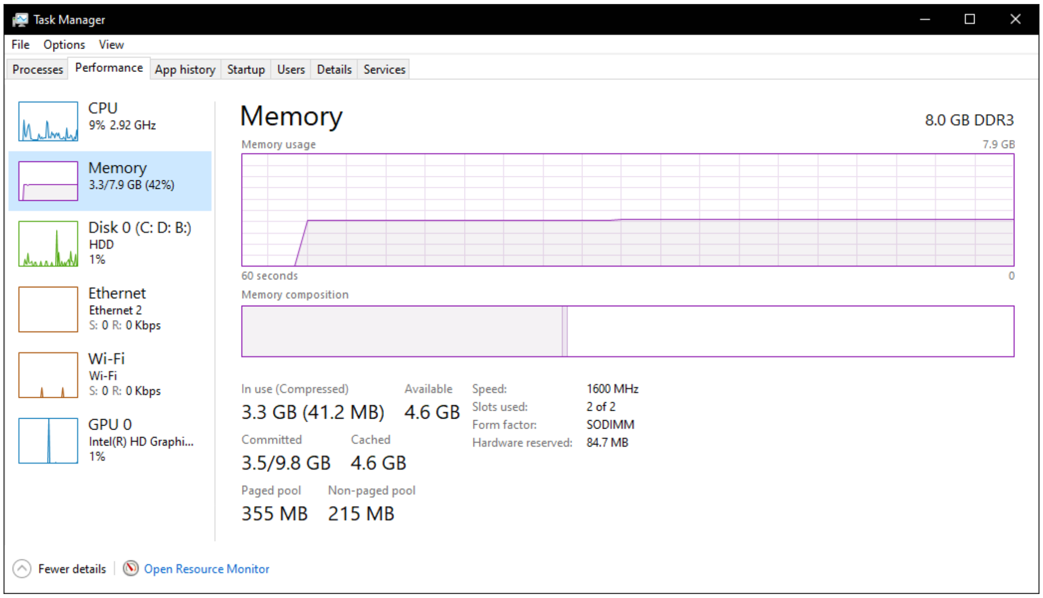


Figure 20: Using task manager to see system memory

Everything is loaded in the RAM when the computer starts including the OS. Even when we are not using any programs, RAM is being used by different services and the OS in the background to keep the system running. RAMs come in many types; this system uses DDR3 RAM in SODIMM (Small Outline Dual In-line Memory Module) form factor (small, generally used in laptops). There are tools like CPU-Z that can give more information on the RAM we are using. DDR stands for double data rate and which means, the RAM can send and receive signals twice per cycle. Most modern systems use DDR RAM the number after the DDR represent their class, higher number means newer generation with better performance and improvements. DDR RAMs are not backward or forward compatible.

Memory Management

Memory is a vital resource in any system. The amount of memory a system has is limited and although we have much better memory systems than we did in the past, the software have also gotten larger and more complex and growth in multimedia applications like games that require more memory have given a rise to the need to manage the available memory properly and efficiently.

Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution. Memory management keeps track of each memory location, regardless of either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status. (Tutorialspoint, 2021)

There are many methods of memory management some of them are:

- Fixed partitioning: this partitioning approach divided into a fixed number of partitions just one process can be loaded into one partition at the same time. This method is easy to implement but can cause internal fragmentation reducing performance, memory is wasted and multiprogramming is limited. (Alabdulaly, 2016)
- Dynamic Partitioning: Partitions are created dynamically, each process loaded into a partition is exactly have same size as the process. This process uses the memory more efficiently and there is no internal fragmentation but the weakness is e inefficient use of processor because of the need for compaction and external fragmentation. (Alabdulaly, 2016)
- Simple Paging: there is no external fragmentation in this method but there might be no correspondence between page protections. Settings and application data structures, requiring per process page tables, usually operating system need more storage for its internal data structures. (Alabdulaly, 2016)

- Simple Segmentation: in this approach is no internal fragmentation. Weaknesses of this approach are: reduce the overhead compared to dynamic partitioning approach and improved the memory utilization.
- Virtual-Memory Paging: there is no external fragmentation and higher capacity for multiprogramming is possible but memory management via this method is more complex.
- Virtual-Memory Segmentation: this method also supports high level of multiprogramming but is more complex.

Page Coloring

Page coloring is a performance optimization designed to ensure that accesses to contiguous pages in virtual memory make the best use of the processor cache. (Dillon, 2021)

Instead of assigning random physical pages to virtual addresses, which may result in non-optimal cache performance, page coloring assigns reasonably contiguous physical pages to virtual addresses. Thus, programs can be written under the assumption that the characteristics of the underlying hardware cache are the same for their virtual address space as they would be if the program had been run directly in a physical address space. (Dillon, 2021)

Paging and Segmentation

Paging is a memory management technique in the simplest terms paging is temporarily using the hard disk as RAM. Windows allows to manage paging files under advanced settings, but it is not recommended to temper with this setting. Paging files are automatically maintained by the operating system.

Paging is a storage mechanism that allows OS to retrieve processes from the secondary storage into the main memory in the form of pages. In the Paging method, the main memory is divided into small fixed-size blocks of physical memory, which is called frames. The size of a frame should be kept the same as that of a page to have maximum utilization of the main memory and to avoid external fragmentation. Paging is used for faster access to data, and it is a logical concept. (guru99, 2021)

Segmentation is also a memory management method. Segmentation is like paging, only difference being in paging, the memory is divided into fixed blocks but in segmentation, memory size may or may not be fixed.

The operating system maintains a segment map table for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset. (Tutorialspoint, 2021)

Segmentation uses less memory than paging but can cause external fragmentation when the segments are loaded and removed from the main memory repeatedly.

Page size variation

The page size can be determined manually for each drive in windows system under advanced performance systems and can even be disabled but neither option is suggested, as the OS can manage the page size itself and disabling paging can have negative consequences and some programs may not run if paging is disabled.

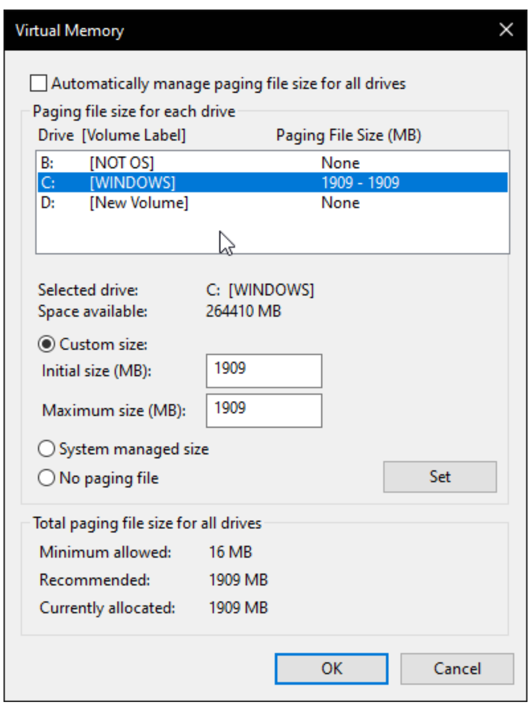


Figure 21: Modifying page size

Several factors are involved in determining correct page size. Two factors argue for a small page size a randomly selected data segment will not fill all the pages and on average half of the final page will be wasted, this waste is internal memory fragmentation. Example if a page size is 4kb but only 1kn is used, 3kb is wasted. With n segments in memory and a page size of p bytes, np/2 bytes will be wasted on internal fragmentation. This reasoning argues for a small page size. (Tanenbaum, 2015)

Larger page size will cause more wastage but on the other hand, smaller page size means the need for many pages, and thus a large page table. This can slow down the system as more time will be needed to look for files. Also, small pages use up much valuable space in the TLB. TLB stands for Translation Lookaside Buffer, it is a cache that is used to reduce time taken to access a user memory location. TLBs are essential for performance, so a balance between large and small page size is needed. For instance, large pages for the kernel and smaller ones for user processes. (Tanenbaum, 2015).

The optimal page size is given by the equation $p = \sqrt{2se}$ (Tanenbaum, 2015).

Where s is the process size and e is the entry page size.

Conclusion

Even when a system is idle, a lot is going on and memory is a vital resource that is carefully managed by the OS. The behind the scenes operation is complex and to keep the system running nothing is ever in a still state, there are many processes and programs running in the system and all this is with the help of memory, computer memory is the core of the system and it is constantly working until power is turned off.

The research part of the coursework was very insightful about the working mechanism of operating systems, I learnt many new things that I did not know before starting this coursework.

References

Alabdulaly, W. (2016) MemoryManagement techniques and. *International Journal of Scientific & Engineering Research*, 7(4), p.1182.

BBC. (2021) *Bitesize* [Online]. Available from: https://www.bbc.co.uk/bitesize/guides/zmb9mp3/revision/7 [Accessed 10 April 2021].

Dillon, M. (2021) *Design elements of the FreeBSD VM system* [Online]. Available from: https://docs.freebsd.org/en/articles/vm-design/index.html [Accessed 10 April 2021].

guru99. (2021) *guru99* [Online]. Available from: https://www.guru99.com/paging-in-operating-system.html#:~:text=Paging%20is%20a%20storage%20mechanism,memory%2C%20which%20is%20called%20frames. [Accessed 10 April 2021].

Tanenbaum, A.S. (2015) *Modern Operating Systems*. 4th ed. Amsterdam: Pearson Education Limited 2015.

Thrnton, S. (2017) *MICROCONTROLLERTIPS* [Online]. Available from: https://www.microcontrollertips.com/dram-vs-sram/#:~:text=DRAM%20is%20at%20least%20ten,use%20as%20main%20processor%20memory. [Accessed 10 April 2020].

Tutorialspoint. (2021) *Tutorialspoint* [Online]. Available from: https://www.tutorialspoint.com/operating_system/os_memory_management.htm [Accessed 10 April 2021].

Appendix

Memory Controller: the memory controller is a chip that is responsible for managing data in the memory. When information is in the memory, the processor sends a request to get that information. The memory controller takes the information from the RAM and passes it to the processor.

Virtual Memory: virtual memory is the combination of physical memory RAM and disk memory. When applications are loaded in the RAM and not enough space is available, the hard disk is also used as RAM temporarily. It is a memory management feature of the OS that simulates physical memory, but at the OS level, the term can be used for all the memory available to the system.

Cache Memory: the processor has its own high-speed memory, accessing this memory is the fastest. The cache holds the most frequently used data. There are two cache memories. L1 cache is built in the processor and is made of SRAM L1 cache is fixed as, it is already built into the processor. There is also L2 cache memory, it is located on the main system board and, as it is outside the CPU, it can also be considered external cache memory.