

**Lab Report: Number Adventure & Sudoku Game**

Course Code: CIS232L

Course Title: Operating Systems Lab

**Submitted to**

Ms. Sonia Nasrin

Lecturer

Department of CIS

Daffodil International University

**Submitted by**

Motiullah Sajit

222-16-663

Department of CIS

Daffodil International University

Date of Submission 04.11.2023

**Table of Contents**

[**Introduction** 3](#_Toc149751764)

[**Motivation** 3](#_Toc149751765)

[**Objective** 4](#_Toc149751766)

[**Tools** 5](#_Toc149751767)

[**Methodology** 6](#_Toc149751768)

[**Implementation** 7](#_Toc149751769)

[Source Code: 7](#_Toc149751770)

[Output 18](#_Toc149751771)

[Main Menu: 18](#_Toc149751772)

[Creating User: 18](#_Toc149751773)

[Number Guessing: 19](#_Toc149751774)

[Leaderboard: 19](#_Toc149751775)

[Sudoku: 20](#_Toc149751776)

[NumberJack: 21](#_Toc149751777)

[**Conclusion** 22](#_Toc149751778)

# **Introduction**

The Number Adventure & Sudoku Game is a command-line-based project that combines two distinct games: a number adventure game and a Sudoku puzzle. This project aims to provide an engaging, interactive, and educational experience for users while exploring Linux shell scripting and various game development concepts.

# **Motivation**

The motivation behind this project stems from the desire to create an entertaining and educational command-line game that combines classic number-based puzzles with the interactivity of shell scripting. By merging the Number Adventure and Sudoku games, we aimed to offer users an engaging experience, test their number skills, and challenge them with the classic rules of Sudoku.

# **Objective**

The primary objective of this project is to create a multifaceted command-line game with the following key goals:

1. Develop a single-player number guessing game with multiple difficulty levels to challenge and entertain players.

2. Implement a multiplayer number guessing game that allows users to compete or collaborate with their peers.

3. Create a NumberJack game, which is a fun and educational number-based puzzle for players to enjoy.

4. Incorporate a classic Sudoku puzzle with well-defined rules to test users' logical thinking and problem-solving skills.

5. Design a user profile system to enable players to track their progress and scores.

6. Implement a leaderboard feature to showcase the top-performing players in the game.

7. Offer clear instructions for each game mode to ensure an enjoyable and frustration-free experience for all players.

8. Use shell scripting and Linux-based tools to create a seamless and intuitive user interface.

# **Tools**

The project utilized the following tools and technologies:

- Bash shell scripting: For creating the core logic and gameplay of the command-line games.

- `nc` (netcat) tool: Used for enabling the multiplayer mode, allowing players to communicate and compete in real-time.

- Git: For version control and collaborative development.

- GitHub: To host the project repository and facilitate collaborative contributions.

# **Methodology**

The development process followed these key steps:

1. Game Design: Initial planning and design of the games, including the user interface and game rules.

2. Bash Scripting: Writing Bash scripts for the number adventure game, NumberJack game, Sudoku game, user profiles, and leaderboards.

3. Integration: Merging the individual game scripts into a unified project with a menu-based interface.

4. Testing and Debugging: Extensive testing and debugging to ensure smooth gameplay and eliminate errors.

5. Documentation: Writing code comments and creating a README file to guide users on how to play the games.

6. Version Control: Using Git and GitHub for version control and collaboration with contributors.

# 

# **Implementation**

## Source Code:

#!/bin/bash

generate\_random\_number() {

  local min=$1

  local max=$2

  echo $((RANDOM % (max - min + 1) + min))

}

send\_message() {

  local message="$1"

  echo "$message" > /dev/tcp/$opponent\_ip/$opponent\_port

}

receive\_message() {

  nc -l -p $player\_port

}

declare -A user\_profiles

create\_user\_profile() {

  read -p "Enter a new user ID: " user\_id

  read -s -p "Enter a passcode: " passcode

  echo

  echo "$user\_id:$passcode:0" >> user\_profiles.txt

  echo "User profile created for user ID: $user\_id"

}

authenticate\_user() {

  read -p "Enter your user ID: " user\_id

  read -s -p "Enter your passcode: " passcode

  echo

  if grep -q "^$user\_id:$passcode:" user\_profiles.txt; then

    echo "Authentication successful. Welcome, $user\_id!"

    return 0

  else

    echo "Authentication failed. Invalid user ID or passcode."

    return 1

  fi

}

update\_leaderboard() {

  local user\_id="$1"

  local score="$2"

  sed -i "s/^$user\_id:[0-9]\*$/$user\_id:$score/" leaderboard.txt

}

# Function to display the leaderboard

display\_leaderboard() {

  clear

  echo "Local Leaderboard:"

  sort -t: -k2,2nr leaderboard.txt | while IFS=: read -r user\_id score; do

    echo "Player $user\_id Score: $score"

  done

  read -p "Press Enter to continue..."

}

play\_game() {

  local min=$1

  local max=$2

  local secret\_number=$(generate\_random\_number $min $max)

  local attempts=0

  local score=0

  local time\_limit=60

  echo "You have $time\_limit seconds to guess the number between $min and $max."

  local start\_time=$(date +%s)

  while true; do

    read -t $time\_limit -p "Guess the number between $min and $max: " guess

    local current\_time=$(date +%s)

    local elapsed\_time=$((current\_time - start\_time))

    if [ $? -ne 0 ]; then

      echo "Time's up! You didn't guess in time."

      break

    fi

    ((attempts++))

    if [[ $guess -lt $secret\_number ]]; then

      echo "Try higher!"

    elif [[ $guess -gt $secret\_number ]]; then

      echo "Try lower!"

    else

      echo "Congratulations! You guessed the number $secret\_number in $attempts attempts."

      score=$((100 - attempts))

      echo "Your score: $score"

      read -p "Press Enter to continue..."

      break

    fi

    if [ $elapsed\_time -ge $time\_limit ]; then

      echo "Time's up! You didn't guess in time."

      break

    fi

  done

}

play\_multiplayer\_game() {

  clear

  echo "Multiplayer Mode"

  echo "Waiting for another player to join..."

  echo "Share the following information with the other player:"

  echo "Your IP: $my\_ip"

  echo "Your Port: $my\_port"

  echo "1. Host a game"

  echo "2. Join a game"

  read -p "Select an option: " mp\_option

  case $mp\_option in

    1)

      player\_number=1

      player\_port=$my\_port

      opponent\_ip=""

      opponent\_port=""

      opponent\_ip=$(receive\_message)

      opponent\_port=$(receive\_message)

      echo "Player 2 has joined the game."

      ;;

    2)

      player\_number=2

      player\_port=$my\_port

      opponent\_ip=$opponent\_ip

      opponent\_port=$opponent\_port

      send\_message "$my\_ip"

      send\_message "$my\_port"

      echo "Connected to Player 1."

      ;;

    \*)

      echo "Invalid option. Please choose 1 or 2."

      return

      ;;

  esac

  local min=1

  local max=100

  local secret\_number=$(generate\_random\_number $min $max)

  local attempts=0

  local score=0

  while true; do

    read -p "Player $player\_number, guess the number between $min and $max: " guess

    ((attempts++))

    if [[ $guess -lt $secret\_number ]]; then

      echo "Try higher!"

    elif [[ $guess -gt $secret\_number ]]; then

      echo "Try lower!"

    else

      echo "Player $player\_number, you guessed the number $secret\_number in $attempts attempts."

      score=$((100 - attempts))

      echo "Your score: $score"

      send\_message "Player $player\_number has guessed the number in $attempts attempts with a score of $score."

      break

    fi

  done

}

# Function to play the NumberJack game

play\_numberjack\_game() {

  clear

  echo "Welcome to NumberJack Game!"

  ch=0

  while [ $ch -ne 3 ]; do

    echo "NumberJack Menu:"

    echo "1. Play NumberJack"

    echo "2. Instructions"

    echo "3. Return to Main Menu"

    read -p "Enter your choice: " ch

    case $ch in

      1)

        x=0

        c=0

        p=0

        read -p "Enter any number between 0 and 9: " n

        while [ $c -eq 0 ]; do

          x=11

          r=($(shuf -i 0-9 -n 10))

          echo "${r[@]} "

          for i in {1..10}; do

            a[$i]=$i

          done

          echo "${a[@]} "

          read -t 5 -p "Enter the index of your number: " x

          if [[ $? -gt 128 ]]; then

            c=1

            break

          fi

          if [ ${r[$(($x))-1]} -eq $n ]; then

            echo "Great"

            ((p=p+1))

          else

            c=1

            break

          fi

        done

        ;;

      2)

        echo "HELP: INSTRUCTIONS TO PLAY THE NUMBERJACK GAME. "

        echo "1. Select any number between 0 and 9 (inclusive) i.e 0 and 9 are accepted."

        echo "2. Two lists will appear in front of you."

        echo "3. The upper list will have a list of randomly shuffled numbers between 0 and 9."

        echo "4. See if you can find your chosen number in that list within 5 seconds."

        echo "5. Then you have to enter the index of that number indicated in the second list below."

        echo "6. The game will continue until you are correct and enter the number on time (before 5 sec)."

        ;;

      3)

        break

        ;;

      \*)

        echo "Invalid choice. Please select a valid option."

        ;;

    esac

    if [ $c -eq 1 ]; then

      echo -e "\nGAME OVER\n"

      echo "You scored $p points"

    fi

  done

}

my\_ip=$(hostname -I | awk '{print $1}')

my\_port=$(shuf -i 1024-49151 -n 1)

leaderboard\_file="leaderboard.txt"

if [ ! -f "$leaderboard\_file" ]; then

  touch "$leaderboard\_file"

  echo "1:0" >> "$leaderboard\_file"

  echo "2:0" >> "$leaderboard\_file"

fi

play\_sudoku() {

declare -a board=(

    5 3 0 0 7 0 0 0 0

    6 0 0 1 9 5 0 0 0

    0 9 8 0 0 0 0 6 0

    8 0 0 0 6 0 0 0 3

    4 0 0 8 0 3 0 0 1

    7 0 0 0 2 0 0 0 6

    0 6 0 0 0 0 2 8 0

    0 0 0 4 1 9 0 0 5

    0 0 0 0 8 0 0 7 9

)

print\_board() {

    echo -e "Sudoku Board:"

    for ((row=0; row<9; row++)); do

        for ((col=0; col<9; col++)); do

            echo -n "${board[row\*9 + col]} "

            if [ $(((col + 1) % 3)) -eq 0 ] && [ $col -lt 8 ]; then

                echo -n "| "

            fi

        done

        echo

        if [ $(((row + 1) % 3)) -eq 0 ] && [ $row -lt 8 ]; then

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

        fi

    done

}

is\_valid\_move() {

    local row=$1

    local col=$2

    local num=$3

    for ((i=0; i<9; i++)); do

        if [ "${board[row\*9 + i]}" -eq "$num" ] || [ "${board[i\*9 + col]}" -eq "$num" ]; then

            return 1

        fi

    done

    local box\_start\_row=$((row - row % 3))

    local box\_start\_col=$((col - col % 3))

    for ((i=box\_start\_row; i<box\_start\_row+3; i++)); do

        for ((j=box\_start\_col; j<box\_start\_col+3; j++)); do

            if [ "${board[i\*9 + j]}" -eq "$num" ]; then

                return 1

            fi

        done

    done

    return 0

}

is\_solved() {

    for ((row=0; row<9; row++)); do

        for ((col=0; col<9; col++)); do

            if [ "${board[row\*9 + col]}" -eq 0 ]; then

                return 1

            fi

        done

    done

    return 0

}

play\_sudoku() {

    while true; do

        print\_board

        echo -e "Enter row (1-9) and column (1-9) to place a number (0 to quit):"

        read -p "Row: " row

        read -p "Column: " col

        if [ "$row" -eq 0 ] || [ "$col" -eq 0 ]; then

            echo "Exiting Sudoku game."

            break

        fi

        if [ "$row" -ge 1 ] && [ "$row" -le 9 ] && [ "$col" -ge 1 ] && [ "$col" -le 9 ] && [ "${board[((row-1)\*9 + col-1)]}" -eq 0 ]; then

            read -p "Enter a number (1-9): " number

            if is\_valid\_move $((row-1)) $((col-1)) "$number"; then

                board[((row-1)\*9 + col-1)]="$number"

            else

                echo "Invalid move. Try again."

            fi

        else

            echo "Invalid row or column. Try again."

        fi

        if is\_solved; then

            print\_board

            echo "Congratulations! You've solved the Sudoku puzzle!"

            break

        fi

    done

}

print\_menu() {

    while true; do

        echo -e "\nSudoku Game Menu:"

        echo "1. Play Sudoku"

        echo "2. Return to Main Menu"

        read -p "Enter your choice: " choice

        case "$choice" in

            1)

                play\_sudoku

                ;;

            2)

                break

                ;;

            \*)

                echo "Invalid choice. Please select a valid option."

                ;;

        esac

    done

}

print\_menu

}

play\_number\_adventure\_game() {

    while true; do

        clear

        echo "Number Adventure Game"

        echo "1. Single Player"

        echo "2. Multiplayer"

        echo "3. Instructions"

        echo "4. Leaderboard"

        echo "5. Create User Profile"

        echo "6. Login"

        echo "7. Sudoku"

        echo "8. NumberJack Game"

        echo "9. Quit"

        read -p "Select an option: " choice

        case $choice in

            1)

                clear

                echo "Choose a Difficulty Level:"

                echo "1. Easy (1-50)"

                echo "2. Medium (1-100)"

                echo "3. Hard (1-200)"

                echo "4. Very Hard (1-300)"

                echo "5. Insane (1-500)"

                read -p "Select a difficulty level: " difficulty\_choice

                case $difficulty\_choice in

                    1)

                        play\_game 1 50

                        ;;

                    2)

                        play\_game 1 100

                        ;;

                    3)

                        play\_game 1 200

                        ;;

                    4)

                        play\_game 1 300

                        ;;

                    5)

                        play\_game 1 500

                        ;;

                    \*)

                        echo "Invalid difficulty level. Please choose 1, 2, 3, 4, or 5."

                        ;;

                esac

                ;;

            2)

                play\_multiplayer\_game

                ;;

            3)

                clear

                echo "Instructions:"

                echo "- This is a number guessing game combined with Linux shell commands."

                echo "- In single-player mode, guess the number to earn points."

                echo "- In multiplayer mode, compete or collaborate with others."

                echo "- Use Linux commands to solve puzzles and advance in the game."

                echo "- Check leaderboards for your ranking."

                echo "- Have fun and learn Linux commands!"

                read -p "Press Enter to go back to the menu..."

                ;;

            4)

                display\_leaderboard

                ;;

            5)

                create\_user\_profile

                ;;

            6)

                authenticate\_user

                if [ $? -eq 0 ]; then

                    echo "Press Enter to continue..."

                    read

                    play\_game 1 100

                    update\_leaderboard "$user\_id" "$score"

                else

                    echo "Press Enter to go back to the menu..."

                    read

                fi

                ;;

            7)

                play\_sudoku

                ;;

            8)

                play\_numberjack\_game

                ;;

            9)

                echo "Thanks for playing! Goodbye."

                exit

                ;;

            \*)

                echo "Invalid option. Please choose a valid option."

                ;;

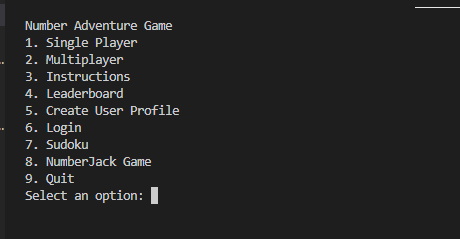
        esac

    done

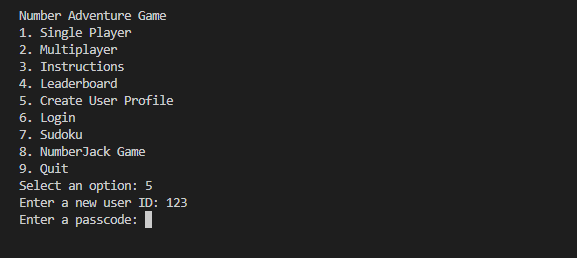
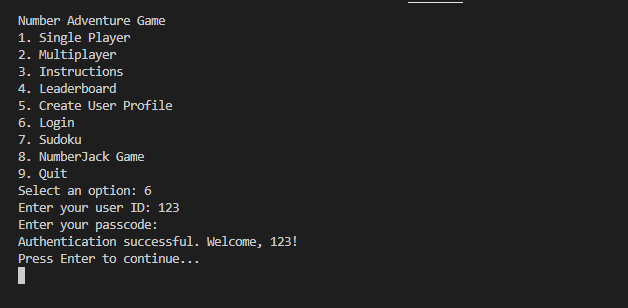
}

play\_number\_adventure\_game

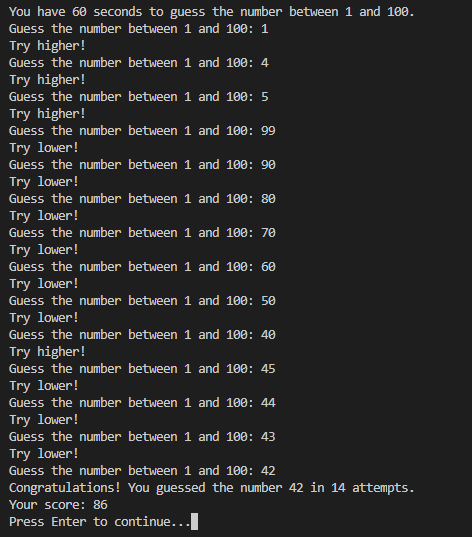
## Output

Main Menu: ****

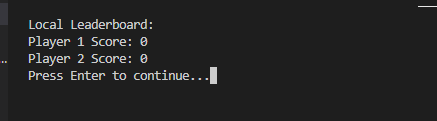
## Creating User:

****User Login: ****

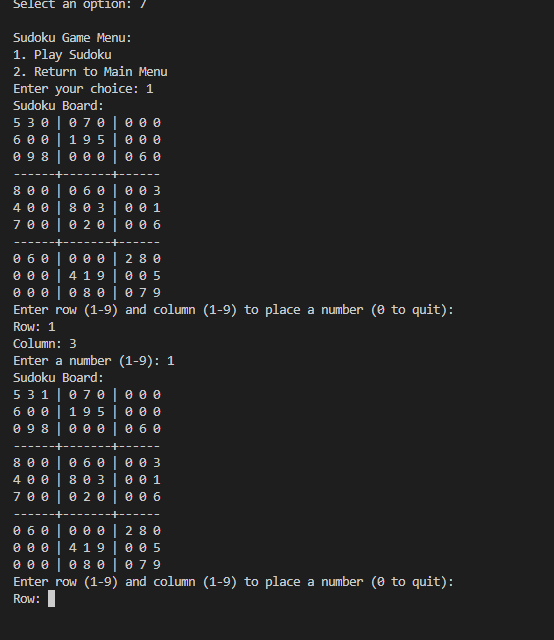
## Number Guessing:

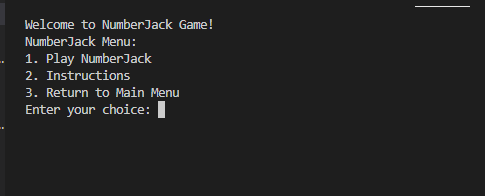
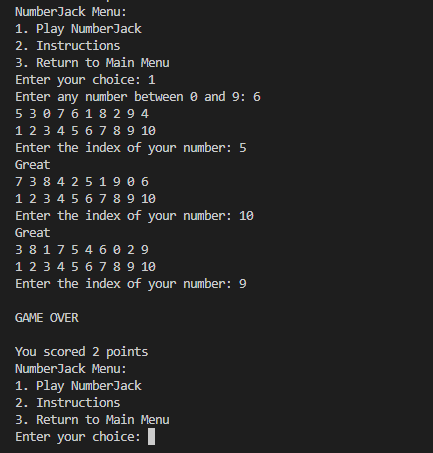
****

## Leaderboard:

****

## Sudoku:

****

NumberJack: **  
**

# **Conclusion**

The Number Adventure & Sudoku Game project successfully combines multiple entertaining and educational command-line games into one cohesive experience. By offering users a choice of single-player and multiplayer modes, classic number puzzles, and a Sudoku challenge, the project achieves its goal of providing an enjoyable and interactive environment.

Additionally, the user profile system and leaderboard allow players to track their progress and compete for top positions. The project demonstrates the capabilities of Bash scripting and Linux-based tools for creating a seamless and intuitive user interface.

In conclusion, the Number Adventure & Sudoku Game project represents a successful fusion of classic games with modern command-line scripting, offering users a fun and educational way to explore the world of numbers and puzzles while enhancing their Linux command-line skills.

**GitHub Link: https://github.com/motiullahsajit/Number-Adventure**