***UNIX MINI PROJECT***

**TITLE:**

**TIC TAC TOE GAME USING**

**BASH SCRIPTING.**

**GROUP MEMBER:**

**1.RAHULKUMAR DAS**

**ROLL NO: 16IT1015**

**SE/IT/A1**

**TITLE** :

**TIC TAC TOE GAME USING BASH SCRIPT**

Our project name is Tic-Tac-Toe game. But the question that comes to our mind is what is exactly Tic Tac Toe?

WHAT IS TIC TAC TOE?

This game is very popular and is fairly simple by itself. It is actually a two player game. In this game, there is a board with n x n squares. In our game, it is 3 x 3 squares. The goal of Tic-Tac-Toe is to be one of the players to get three same symbols in a row - horizontally, vertically or diagonally - on a 3 x 3 grid.

 Tic-tac-toe is not a very challenging game for human beings.

 If you’re an enthusiast ,you’ve probably moved from the basic  game to some variant like three dimensional tic-tac-toe on a larger grid. If you sit downright now to play ordinary three-by-three tic-tac-toe with a friend, what will probably happen is that every game will come out a tie. Both you and your friend can probably play perfectly, never making a mistake that would allow your opponent to win. But can you describehow you know where to move each turn? Most of thetime,you probably aren’t even aware of alternativepossibilities; you just look at the board and instantly know where you want to move. That kind of instant knowledgeis great for human beings, because it makes you a fastplayer. But it isn’t much help in writing a computer program. For that, you have to know very explicitly what your strategy is.

**THEORY:**

If any player is able to draw three Xs or three Os in the following combinations then that player wins. The combinations are:

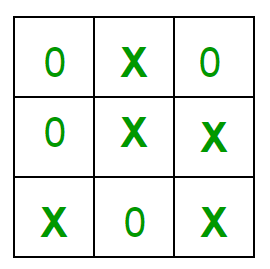
1. 0,1,2
2. 3,4,5
3. 6,7,8
4. 0,4,8
5. 2,4,6
6. 0,3,6
7. 1,4,7
8. 2,5,8

A player can play a [perfect game](https://en.wikipedia.org/wiki/Solved_game" \l "Perfect_play" \o "Solved game) of tic-tac-toe (to win or, at least, draw) if each time it is his turn to play he chooses the first available move from the following list, as used in Newell and Simon's 1972 tic-tac-toe program.[[15]](https://en.wikipedia.org/wiki/Tic-tac-toe" \l "cite_note-15)

1. **Win**: If the player has two in a row, they can place a third to get three in a row.
2. **Block**: If the opponent has two in a row, the player must play the third themselves to block the opponent.
3. **Fork**: Create an opportunity where the player has two threats to win (two non-blocked lines of 2).
4. **Blocking an opponent's fork**: If there is only one possible fork for the opponent, the player should block it. Otherwise, the player should block any forks in any way that simultaneously allows them to create two in a row. Otherwise, the player should create a two in a row to force the opponent into defending, as long as it doesn't result in them creating a fork. For example, if "X" has two opposite corners and "O" has the center, "O" must not play a corner in order to win. (Playing a corner in this scenario creates a fork for "X" to win.)
5. **Center**: A player marks the center. (If it is the first move of the game, playing on a corner gives the second player more opportunities to make a mistake and may therefore be the better choice; however, it makes no difference between perfect players.)
6. **Opposite corner**: If the opponent is in the corner, the player plays the opposite corner.
7. **Empty corner**: The player plays in a corner square.
8. **Empty side**: The player plays in a middle square on any of the 4 sides.

**Rules of the Game**

* The game is to be played between two people
* One of the player chooses ‘O’ and the other ‘X’ to mark their respective cells.
* The game starts with one of the players and the game ends when one of the players has onewhole row/ column/ diagonal filled with his/her respective character (‘O’ or ‘X’).
* If no one wins, then the game is said to be draw.



# Introduction to Linux Shell and Shell Scripting

If you are using any major operating system you are indirectly interacting to **shell**. If you are running Ubuntu, Linux Mint or any other Linux distribution, you are interacting to shell every time you use terminal. In this article I will discuss about linux shells and shell scripting so before understanding shell scripting we have to get familiar with following terminologies –

* Kernel
* Shell
* Terminal

**What is Kernel ?**

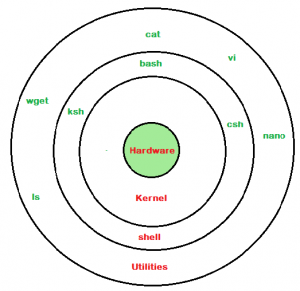
The kernel is a computer program that is the core of a computer’s operating system, with complete control over everything in the system. It manages following resources of the Linux system –

* + - File management
    - Process management
    - I/O management
    - Memory management
    - Device management etc.

It is often mistaken that **[Linus Torvalds](https://en.wikipedia.org/wiki/Linus_Torvalds" \t "_blank)** has developed Linux OS, but actually he is only responsible for development of Linux kernel.  
Complete Linux system = Kernel + [GNU](https://en.wikipedia.org/wiki/GNU" \t "_blank) system utilities and libraries + other management scripts + installation scripts.

**What is Shell ?**

A shell is special user program which provide an interface to user to use operating system services. Shell accept human readable commands from user and convert them into something which kernel can understand. It is a command language interpreter that execute commands read from input devices such as keyboards or from files. The shell gets started when the user logs in or start the terminal.



Shell is broadly classified into two categories –

* + - * + Command Line Shell
        + Graphical shell

**Command Line Shell**

Shell can be accessed by user using a command line interface. A special program called **Terminal** in linux/macOS or **Command Prompt** in Windows OS is provided to type in the human readable commands such as “cat”, “ls” etc. and then it is being execute. The result is then displayed on the terminal to the user.

It will list all the files in current working directory in long listing format.  
Working with command line shell is bit difficult for the beginners because it’s hard to memorize so many commands. It is very powerful, it allows user to store commands in a file and execute them together. This way any repetitive task can be easily automated. These files are usually called **batch files** in Windows and **Shell** Scripts in Linux/macOS systems.

**Graphical Shells**

Graphical shells provide means for manipulating programs based on graphical user interface (GUI), by allowing for operations such as opening, closing, moving and resizing windows, as well as switching focus between windows. Window OS or Ubuntu OS can be considered as good example which provide GUI to user for interacting with program. User do not need to type in command for every actions.

There are several shells are available for Linux systems like –

* + [BASH (Bourne Again SHell)](https://en.wikipedia.org/wiki/Bash_(Unix_shell)" \t "_blank) – It is most widely used shell in Linux systems. It is used as default login shell in Linux systems and in macOS. It can also be installed on Windows OS.
  + [CSH (C SHell)](https://en.wikipedia.org/wiki/C_shell" \t "_blank) – The C shell’s syntax and usage are very similar to the C programming language.
  + [KSH (Korn SHell)](https://en.wikipedia.org/wiki/Korn_shell" \t "_blank) – The Korn Shell also was the base for the POSIX Shell standard specifications etc.

Each shell does the same job but understand different commands and provide different built in functions.

**Shell Scripting**

Usually shells are interactive that mean, they accept command as input from users and execute them. However some time we want to execute a bunch of commands routinely, so we have type in all commands each time in terminal.  
Shell scripts are similar to the **[batchfile](https://en.wikipedia.org/wiki/Batch_file" \t "_blank)** in MS-DOS. Each shell script is saved with **.sh**file extension eg. **myscript.sh**

A shell script have syntax just like any other programming language. If you have any prior experience with any programming language like Python, C/C++ etc. it would be very easy to get started with it.  
A shell script comprises following elements –

* + Shell Keywords – if, else, break etc.
  + Shell commands – cd, ls, echo, pwd, touch etc.
  + Functions
  + Control flow – if..then..else, case and shell loops etc.

**Why do we need shell scripts ?**

There are many reasons to write shell scripts –

* + To avoid repetitive work and automation
  + System admins use shell scripting for routine backups
  + System monitoring
  + Adding new functionality to the shell etc.

**Advantages of shell scripts**

* + The command and syntax are exactly the same as those directly entered in command line, so programmer do not need to switch to entirely different syntax
  + Writing shell scripts are much quicker
  + Quick start

**Disadvantages of shell scripts**

* + Prone to costly errors, a single mistake can change the command which might be harmful
  + Slow execution speed
  + Design flaws within the language syntax or implementation
  + Not well suited for large and complex task

**DESCRIPTION OF CODE:**

**1.RESET FUNCTION:**

In this function we initialize the contents of the array with “ .“ which denotes that the array is full and that location can be used by the player for move in the game.

Initialize the variable player=1

Initialize the variable to gamestatus=1, which will be a check condition for the game to terminate.

Now printing a statement to denote that the game has been started.

**2.SET FUNCTION:**

This function takes the value of the row and column from the user and after doing a simple arithmetic calculation the symbol “X” or “0” is marked at the respective index.

Then it will check whether the index chosen by the player is available for the move or not. If the index contains “ .“ that means the location is available for the move by the player. If it’s empty then that index location get’s replaced by the symbol of the player. If it’s not empty the a message will be displayed “You can’t place there” because it’s already filled.

**3.PRINT FUNCTION:**

This function prints the 1D array in the required format and this helps us to diiferentiate the place which is available for the player and which is not.

**4.CHECKMATE FUNCTION:**

This is an important function as it contains the winning condition. IF condition is used here and is called to check the condition everytime the player plays whether the game is over or not.

**5.CHECKGAME FUNCTION:**

This function contains all the possible ways in which any one of the player can win.

**6. WHILE LOOP:**

It works on the condition whether the gamestatus is equal to 1 which means that no player has still won the game and a draw condition.

This loop takes input from the user in which row and column the symbol should be inserted.

It also contains the condition where we can restart the game.

Also if the user enters the input in wrong format it says “wrong input”

If the gamestatus changes to 0 that means any one of the player has won the game.

If all the locations are occupied by “X” or “0” a message will be displayed saying “ The Game is draw”

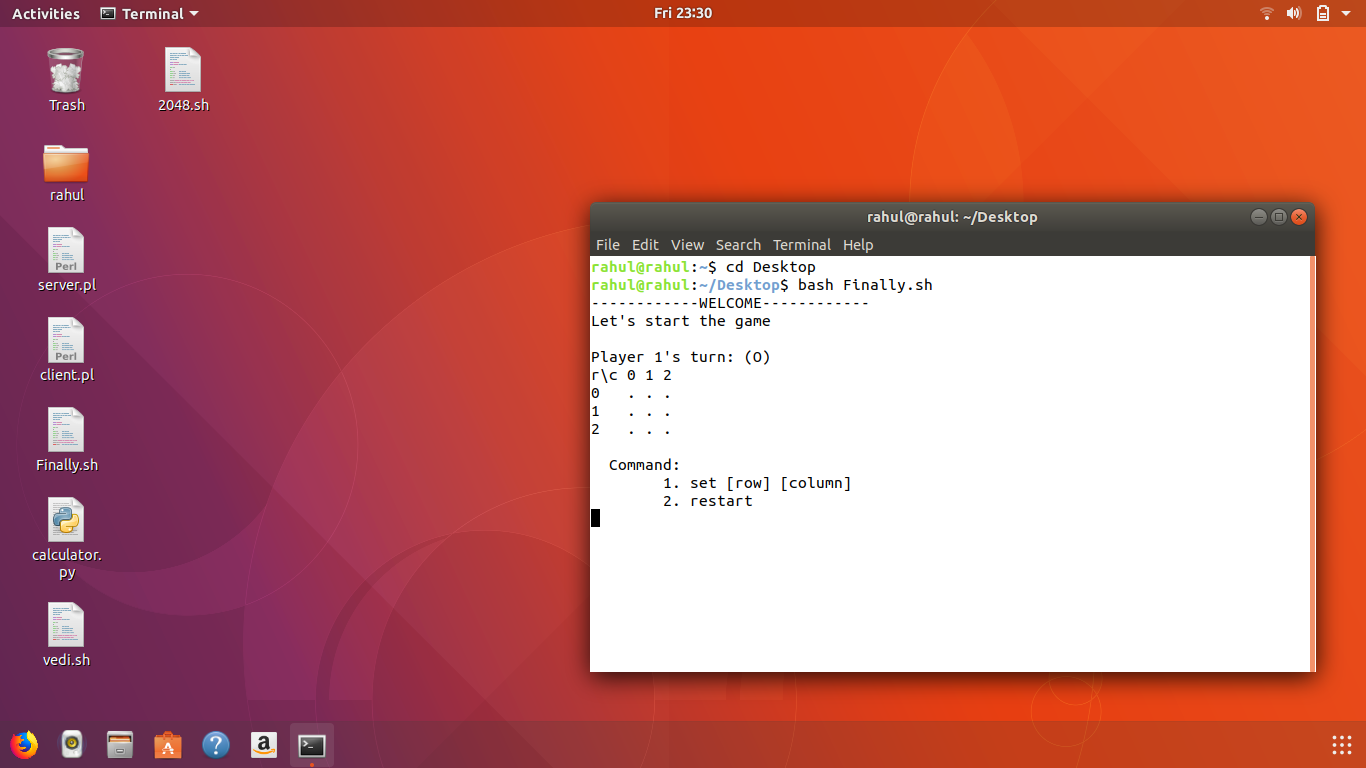
**REVIEW OF BASIC UNIX COMMANDS**

Familiar commands often useful in shell scripts

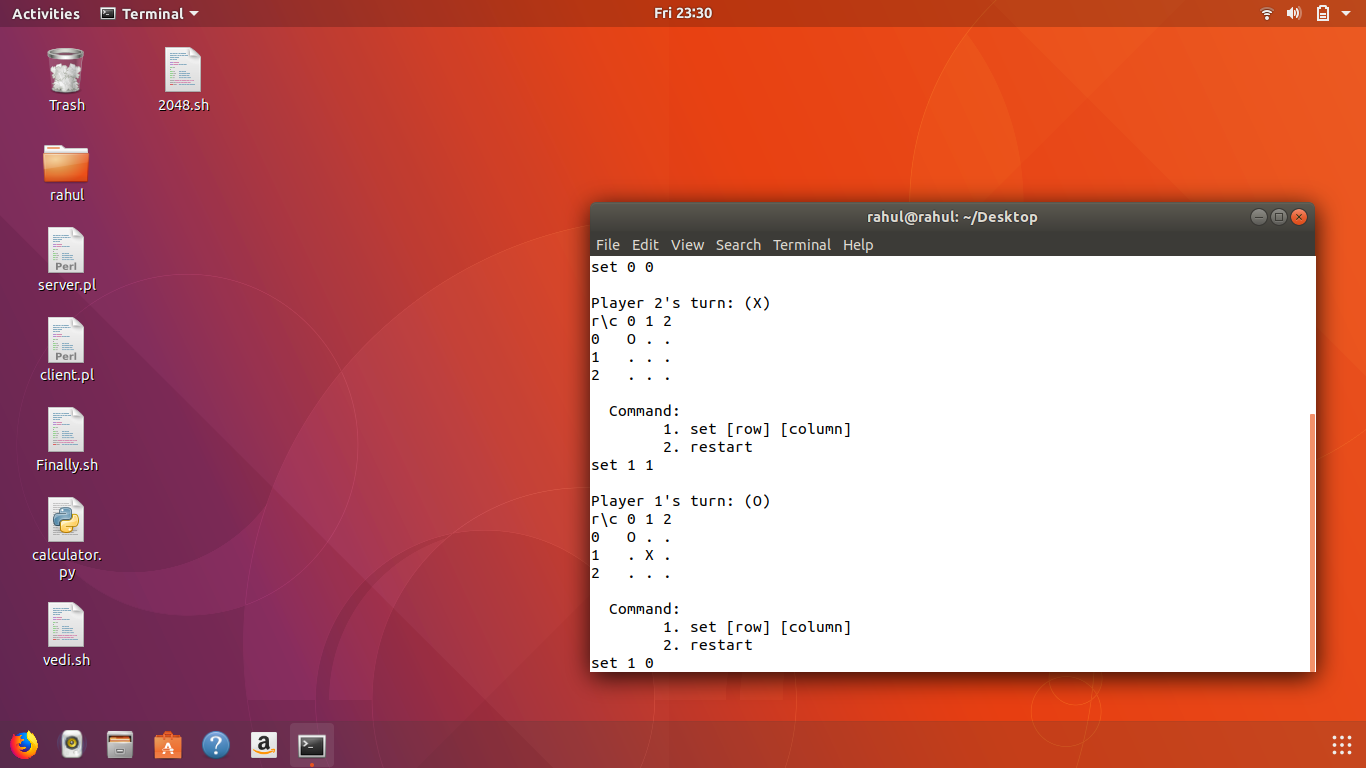
* cat -Displays the content of the file name.
* cp - copy a file
* date - print the date and time
* grep - scan for a string
* head - show first lines of a file
* tail - show last lines of a file
* mv - move or rename a file
* wc - count lines, words, characters

**SCREENSHOTS OF WORKING:**

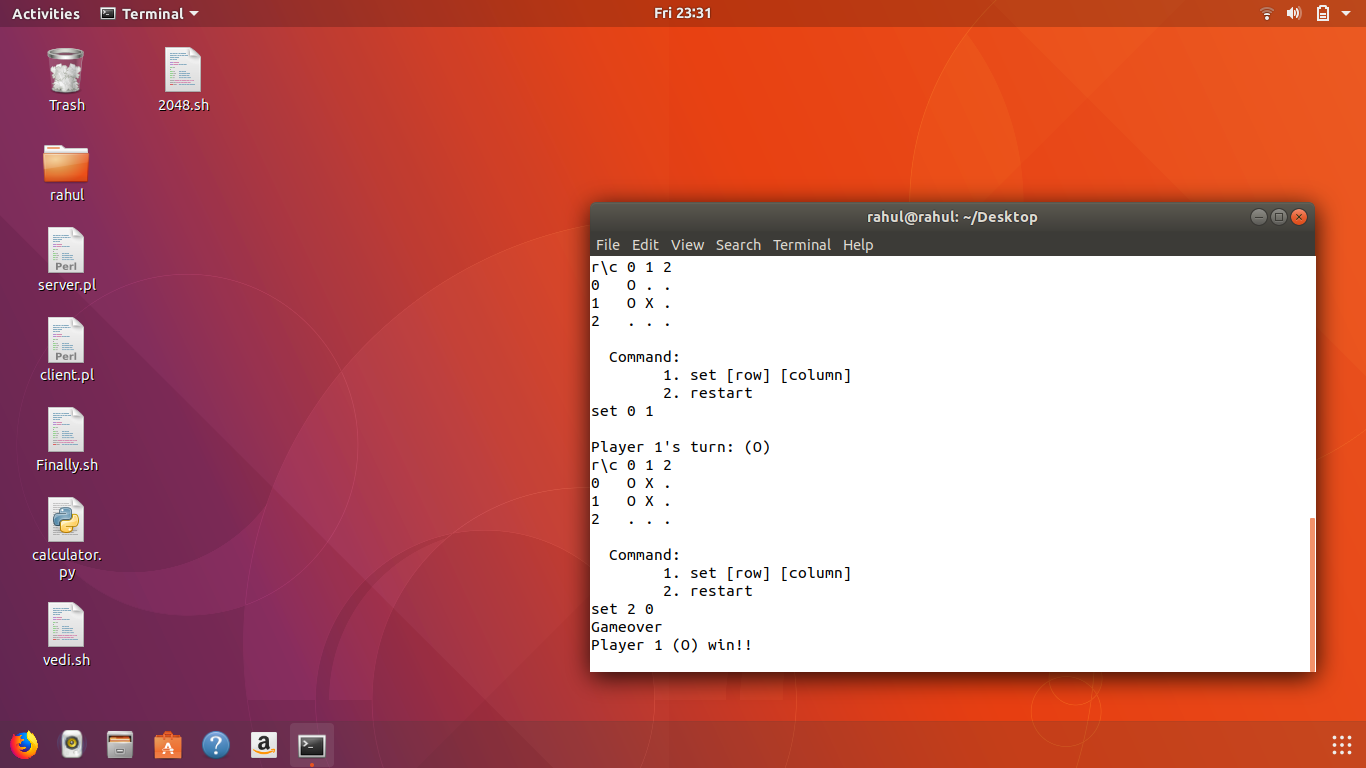
**1.Execution of game:**



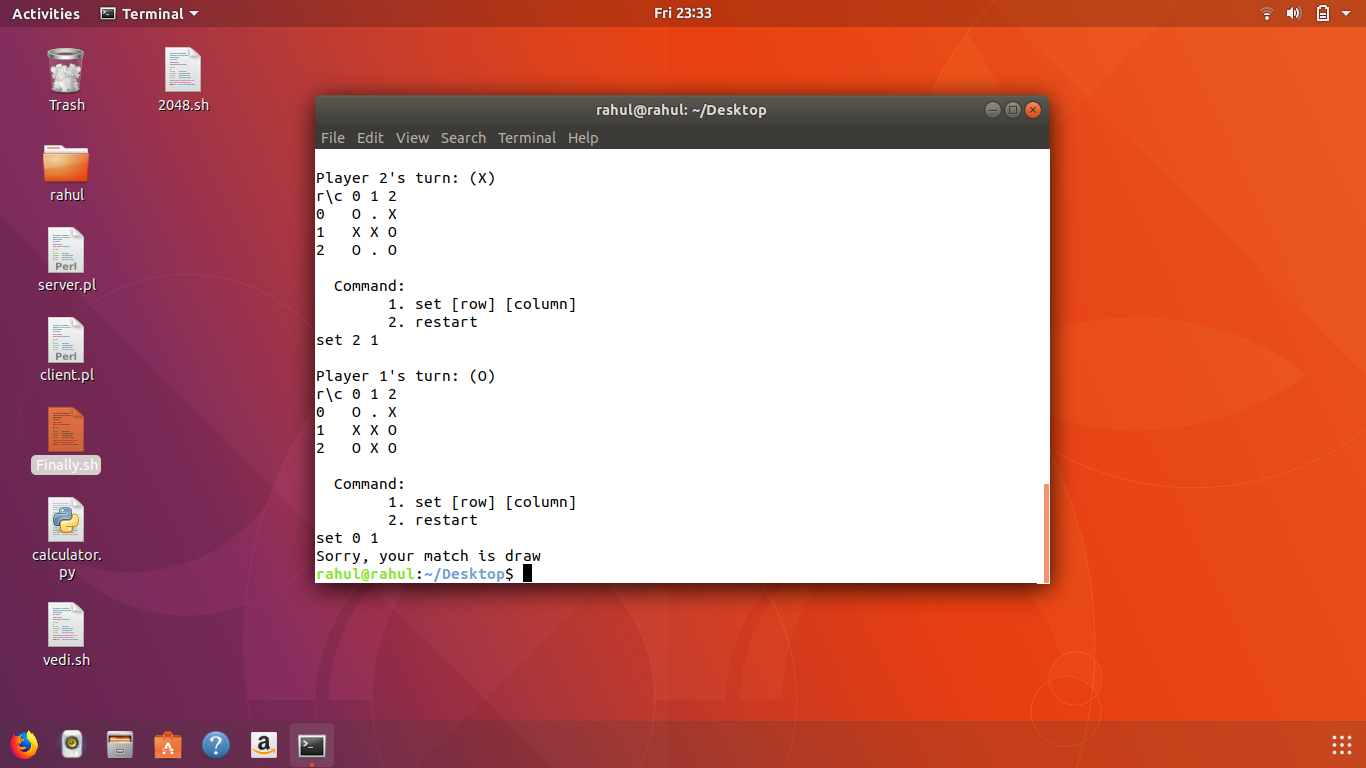
**2.User gives input:**



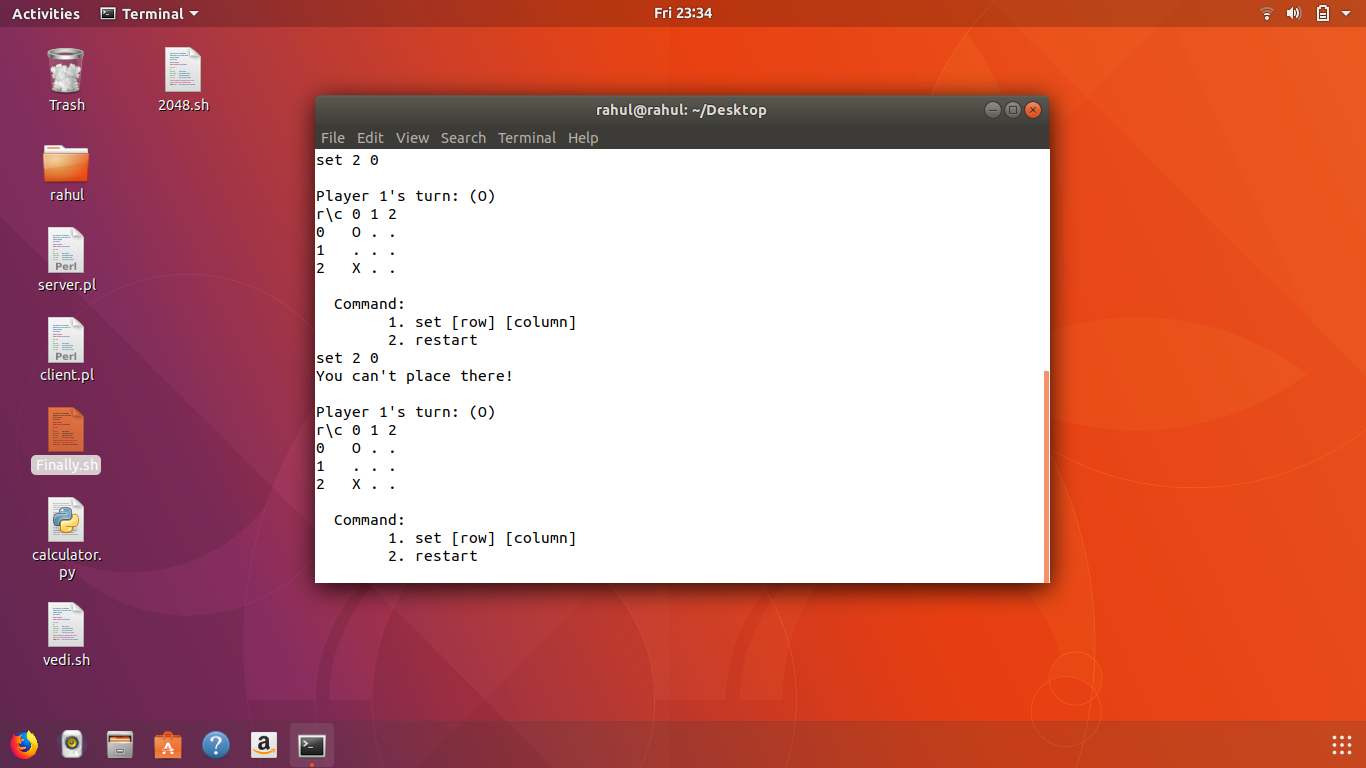
**3. Player 1 wins:**



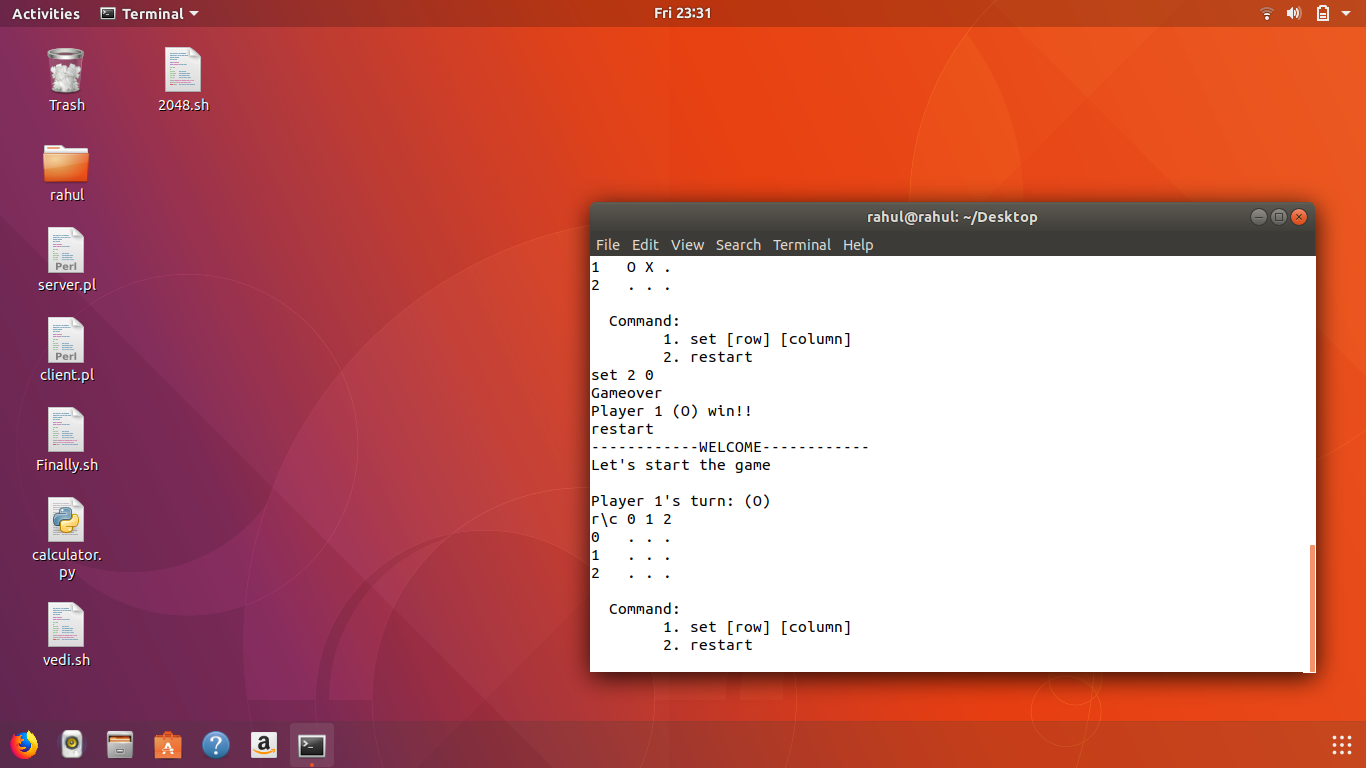
**4.When the match is draw:**



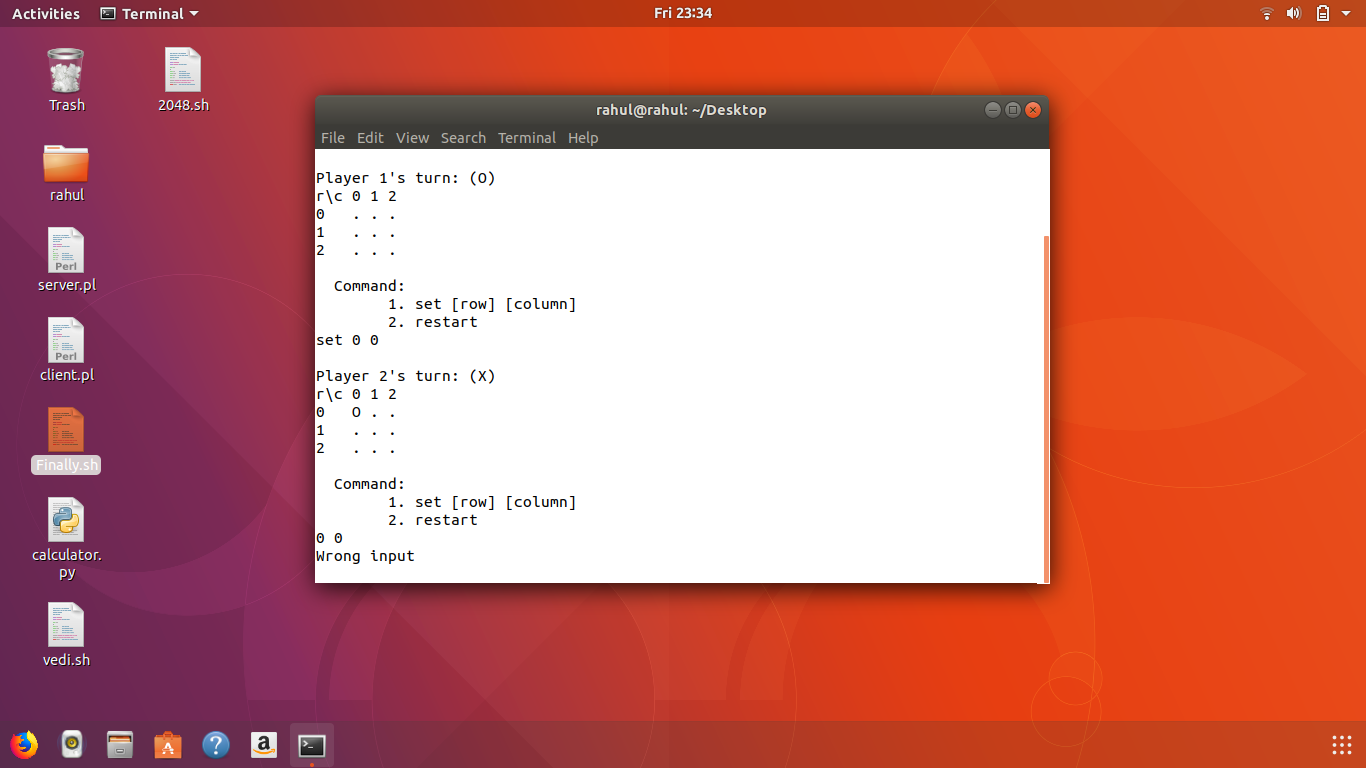
**5. Preoccupied space:**



**6.Restart command:**



**7.Wrong format**



**CONCLUSION:**

In the end, we would like to conclude that our aim was to make tic tac toe game using bash script. Our code is written for two player version i.e Human vs human. The final outcome (win, lose, draw) was the only information available regarding the quality of the play.