Bash Snake

Proyecto Final
Sistemas Operativos

About

A simple snake game written in Bash.

How to Play

Direction control: vim-style keys h, j, k and l;
Quit: q
Accept both upper- and lower-case.

Some Implementation Details
---
 * Bash-snake has two processes, the foreground one responds to
   user's control commands, the background one draws the board.
   `kill` and `trap` are used to enable communication between the two processes.

   The foreground process `getchar()` ignores `SIGINT` and `SIGQUIT`, and
   replies to the signal of death `SIG_HEAD` by returning from the function `getchar()`.

   The background process `game_loop()` traps direction control signals from the keyboard,
   and self-defined signal `SIG_QUIT` which indicates the press of Q button.

 * Use `$!` to grasp the PID of the latest created background process.

 * When setting up several traps, it's necessary to guarantee that
   the normal execution will not be interrupted by the signal handling
   if the handlers envolve modification of some variables it depends on.

 * The snake is represented by the coordinate `$head_r` and `$head_c` indicating

the row and column on which the snake head is, and a string `$body` which
stores the directions from the head to tail.
e.g. a snake with this shape (`@` is the head):

```
@
oooo
  o
```

will have a `$body` with value 21112, meaning 'down, right, right, right, down'.

With the above scheme and the coordinates of the snake head, we can figure out
the position the the entire snake body, without storing every part of its body.

* The board is represented by a "2D array", which is actually implemented by
a bunch of bash arrays and the `eval` command. For example, assigning 123 to the
array entry `arr[5][6]` is done with `eval "arr$i[$j]=123"`.

* The board is re-drawn in each iteration, which happens every 0.03 seconds.

* Coloring is implemented with the escaped sequences of the terminal.

Integrantes:
- Miguel A. Richter Flores (0212627)
- Sheila Nahomi Barrera Pérez (0214901)
- Jorge Luis Nava Soriano (0219741)

Modificaciones realizadas por el alumno:

* Originalmente, cuando se corría el script, el juego iniciaba de manera inmediata por lo que se implementaron ciertas mejoras para hacerlo más interactivo.

* Lo primero que se realizó fue añadir un menú con un título personalizado del juego.

* Posteriormente, se separó el juego en funciones (además de las ya existentes) para poder implementarlo de manera sencilla en el CASE del menú.

* La idea es implementar un seguimiento de puntuación del top cinco jugadores, para ello, se modificó el script agregando una sección previa al inicio del juego, la cual solicita el nombre del jugador para poder realizar un seguimiento de su puntuación, este nombre, es mandado a segundo plano junto con el funcionamiento del juego.

* Al terminar el juego, es decir, perder, se agrega el nombre del jugador junto con su puntuación al archivo en el cual se guarda el top 5, sin importar que sean 6 momentaneamente.

* Luego se llama a la función ordenarScore, la cual filtra y organiza el top 5. El primer paso que lleva a cabo la función es crear dos arreglos temporales en los cuales se guardarán todos los valores del archivo, es decir, los 6 nombres y su puntuación correspondiente. Para realizar esto, se utiliza IFS, la cual sirve para la detección de delimitadores, en este caso, son comas.

* Posteriormente, se ordenan ambos arreglos de mayor a menor a través del ordenamiento burbuja, con base en la puntuación del jugador.

* Finalmente, se reescribe el documento el cual guarda el registro del top, pero únicamente los primeros 5 valores de los arreglos, de esta manera, no es necesario implementar otros algoritmos más complejos.

* La segunda opción del menú consiste en mostrar el top 5 de jugadores, el cual filtra la información de la misma manera en la cual se filtró con los delimitadores previamente explicada.

* La tercera opción es para salir del juego.