Practical 'Historial'

Your team identifier is: SO_SA55
Practical's name: Historial

¡Atención!:

- Las modificaciones implementadas **no deben afectar a las especificaciones básicas del shell**, que deben seguir funcionando correctamente.
- Si una especificación requiere la impresión de mensajes, no se dará por correcta si no se ajustan exactamente al formato especificado.
- Salvo que se diga otra cosa, los mensajes a imprimir se imprimirán por salida estándar de consola y deben estar libres de códigos de color u otros de control adicionales para el terminal.

Historial de comandos

Se trata de manejar la entrada de comandos en el *prompt*, con el `tty` configurado en modo no canónico, y sin echo, de forma que el shell leerá de teclado (stdin) carácter por carácter (`getchar()`) en lugar de una línea entera tal como se procede en el prototipo proporcionado.

En este modo el shell es el responsable de acciones sobre la línea tecleada, que incluye por ejemplo borrar un carácter con la línea de retroceso, mover el cursor a derecha e izquierda por la línea tecleada hasta el momento, etc.

Al leer el carácter de nueva línea, `'\n'`, se considera completa la línea de comando y se procederá a su *parsing*, tal como lo realiza la función proporcionada `get_command()`.

Este modo nos va a permitir reutilizar rápidamente a los comandos tecleados, manteniendo un historial de comandos ya tecleados con anterioridad.

Se pide que las teclas de cursor arriba `↑` y abajo `↓` del teclado, nos permitan desplazarnos por el historial de manera similar a cómo ocurre en *bash*: la flecha arriba irá a una posición anterior a la actual del histórico de comando, y la flecha abajo a la posición del histórico siguiente a la actualmente mostrada.

Observa que al igual que ocurre en *bash*, se deberá poder editar cualquier comando recuperado de esta forma con los cursores ya que se convierte en el comando actual (desplazamiento del cursor, inserción y borrado de caracteres, …)

Se diseñará así mismo un comando interno `hist` que mostrará la lista de comandos tecleados junto con un índice para cada uno.

Así mismo, siguiendo una sintaxis similar a la del *shell de Bourne*, si se teclea el signo de admiración (`!`) seguido inmediatamente sin espacio de un número entero positivo (por ejemplo, `!2`), se volverá a ejecutar el comando que ocupe la posición indicada por dicho número en el historial (en este ejemplo el número 2). Nótese que dicho comando puede corresponder con un comando en *background* con el carácter `'&'` al final, o también incluir las redirecciones básicas.

También se pide incluir un comando interno `histclean` que limpie el historial dejando una lista de comandos vacía, tal como estaba al principio de lanzar el shell.

Ejemplo:

```
COMMAND-> hist
1 ps
2 ls -al
3 cd
…
7 echo -e Hola mundillo! > /tmp/echo-e
8 xclock -bg yellow -d -update 1 &
9 hist

COMMAND-> rm -f /tmp/echo-e # Borramos este fichero

COMMAND-> !7                # Se repetirá el comando nº 7 del historial
echo -e Hola mundillo! > /tmp/echo-e

COMMAND-> cat /tmp/echo-e   # El comando nº 4 tenía una redirección a este fichero
                            # Veamos su contenido
Hola mundillo!

[...]

COMMAND-> histclean         # Limpiaremos el historial
COMMAND-> hist              # El historial empieza de nuevo
1 hist
```

Este ejercicio se puede enfocar de dos maneras:

- Una es haciendo uso del terminal a bajo nivel.
- Otra es usando la librería *readline*, que a más alto nivel facilita el uso del terminal.

En los dos siguientes apéndices se describen las dos posibilidades.

Apéndice I: Información sobre el manejo del terminal a bajo nivel

Esta información es útil para el posicionamiento del cursor, navegación por el historial y cambio de colores sin la utilización de librerías,

sólo usando secuencias de escape ANSI.

- Configuración de *tty* con lectura de teclado pulsación a pulsación (carácter a carácter, modo no canónico, sin esperar a retorno de carro, "\n"), ni imprimir tampoco ECHO.

```
/* lee un caracter sin esperar a '\n' y sin eco */
#include <stdio.h>
#include <termios.h>
#include <unistd.h>
char getch()
{
   int shell_terminal = STDIN_FILENO;
   struct termios conf;
   struct termios conf_new;
   char c;
   tcgetattr(shell_terminal,&conf); /* leemos la configuracion actual */
   conf_new = conf;
   conf_new.c_lflag &= (~(ICANON|ECHO)); /* configuramos sin buffer ni eco */
   conf_new.c_cc[VTIME] = 0;
   conf_new.c_cc[VMIN] = 1;
   tcsetattr(shell_terminal,TCSANOW,&conf_new); /* establecer configuracion */
   c = getc(stdin); /* leemos el caracter */
   tcsetattr(shell_terminal,TCSANOW,&conf); /* restauramos la configuracion */
   return c;
}
```

- Interpretación de las teclas de cursor, borrar, etc...

```
/* Las teclas de cursor devuelven una secuencia de 3 caracteres,
   27 --- 91 --- (65, 66, 67 ó 68) */
char sec[3];
sec[0] = getch();
switch (sec[0])
{
  case 27:
    sec[1] = getch();
    if (sec[1] == 91) // 27,91,...
    {
     sec[2] = getch();
     switch (sec[2])
     {
       case 65: /* ARRIBA */
           break;
       case 66: /* ABAJO */
           break;
       case 67: /* DERECHA */
           break;
       case 68: /* IZQUIERDA */
           break;
       default:
     }
    }
    break;
  case 127: /* BORRAR */
      break
  default:
            ...
}
```

- Movimiento del cursor en el terminal para poder elegir línea y columna donde escribir. Estas son las secuencias de escape ANSI para el movimiento y posicionamiento del cursor:

```
- Posicionar el cursor en la línea L, columna C: \033[*/L/*;*/C/*H
- Mover el cursor arriba N líneas:               \033[*/N/*A
- Mover el cursor abajo N líneas:                \033[*/N/*B
- Mover el cursor hacia adelante N columnas:     \033[*/N/*C
- Mover el cursor hacia atrás N columnas:        \033[*/N/*D
- Guardar la posición del cursor:                \033[s
- Restaurar la posición del cursor:              \033[u
```

  Ejemplo para escribir el carácter "A" en la posición (10,10):

```
printf("\033[*/10/*;*/10/*H A");
```

- Aquí se presentan las secuencias de escape ANSI para cambiar de color el texto de salida en la consola (funcionará solo con terminales con soporte ANSI):

```
#define ROJO "\x1b[31;1;1m"
#define NEGRO "\x1b[0m"
#define VERDE "\x1b[32;1;1m"
#define AZUL "\x1b[34;1;1m"
#define CIAN "\x1b[36;1;1m"
#define MARRON "\x1b[33;1;1m"
#define PURPURA "\x1b[35;1;1m"
```

  Ejemplo de uso: