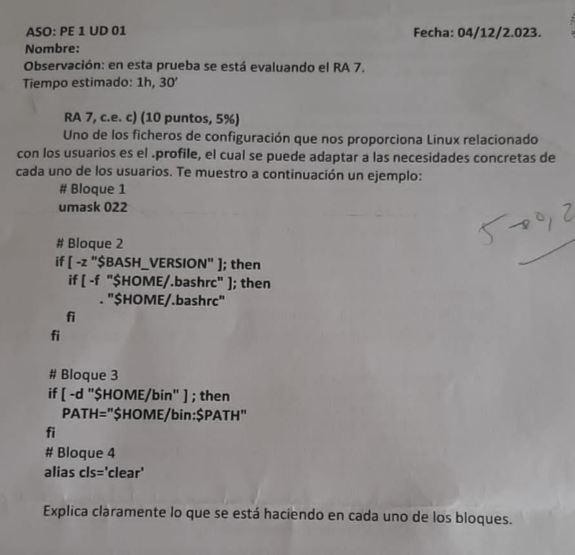
# EXAMEN UD01 ASO

**RA 7, c.e. c)**



**# Bloque 1**

Con el comando umask, se establecen como máscara, los permisos predeterminados para directorios 775 y para ficheros 644.

777

022

755

666

022

644

**# Bloque 2**

Se comprueba si está vacía la variable de entorno $BASH\_VERSION, después comprueba si existe el fichero .bashrc, expandiendo el contenido de la variable de entorno $HOME siendo la ruta de conexión en el home del usuario. Si existe, ejecuta el fichero .bashrc con .

**# Bloque 3**

Comprueba en primer lugar si existe el directorio /home/”nombre\_usuario”/bin, si existe se agrega al PATH la ruta $HOME/bin:$PATH. Por ejemplo:

If [ -d “$HOME/bin” ]; then

PATH=”/home/javier/bin:$PATH”

Fi

**# Bloque 4**

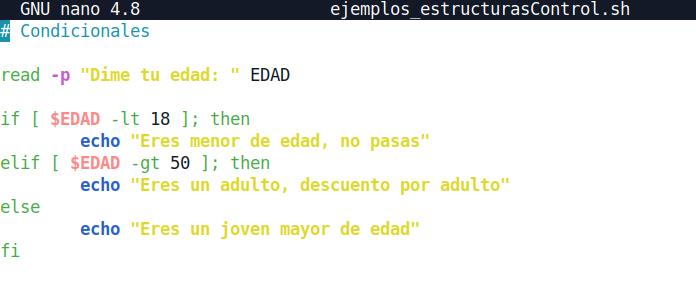
En este bloque, se especifica en el fichero .bashrc un alias al comando clear llamado cls, con el que al ejecutar en la terminal el comando cls, borrará pantalla, por que internamente llamará al comando clear.

**RA7, c.e.a)**

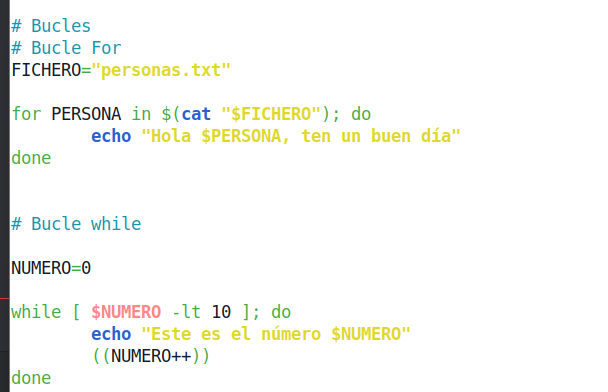


La función de las estructuras de control en los scripts de bash son dirigir el flujo de ejecución del programa, según los condicionales y bucles para ejecutar código repetidamente. Principales estructuras de control en Bash

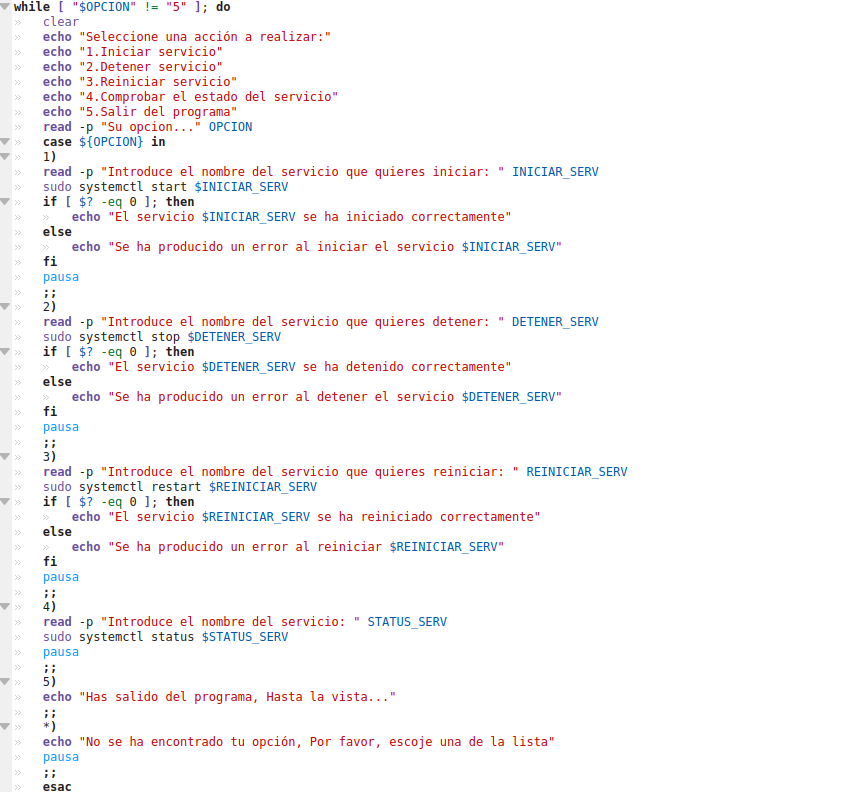
* Condicionales: if, else, elif



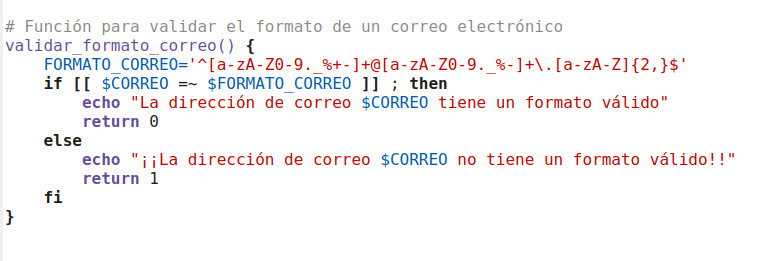
* Bucles: For, While

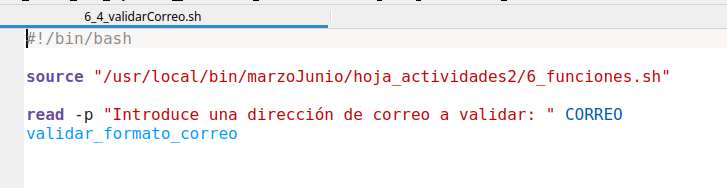


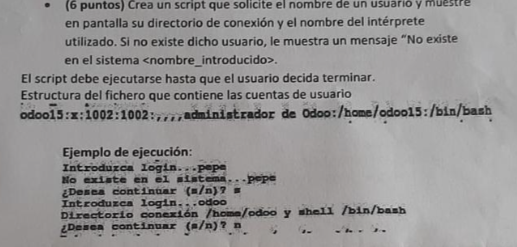
* Case



* Funciones









**RA 7, c.e.b)**



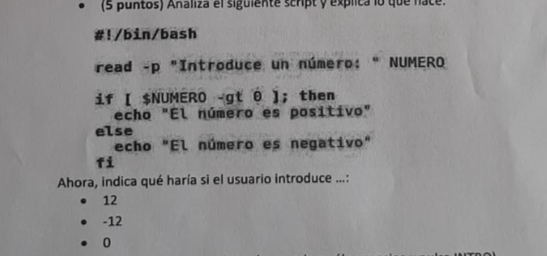
La depuración en los scripts en bash se refiere a las técnicas para analizar la traza de un script analizando cada uno de los caminos que toma el script en función a las condiciones, permitiendo así la depuración de errores.



Podemos depurar errores en la sintaxis, errores de lógica, errores de tiempo de ejecución y errores en variables no definidas.

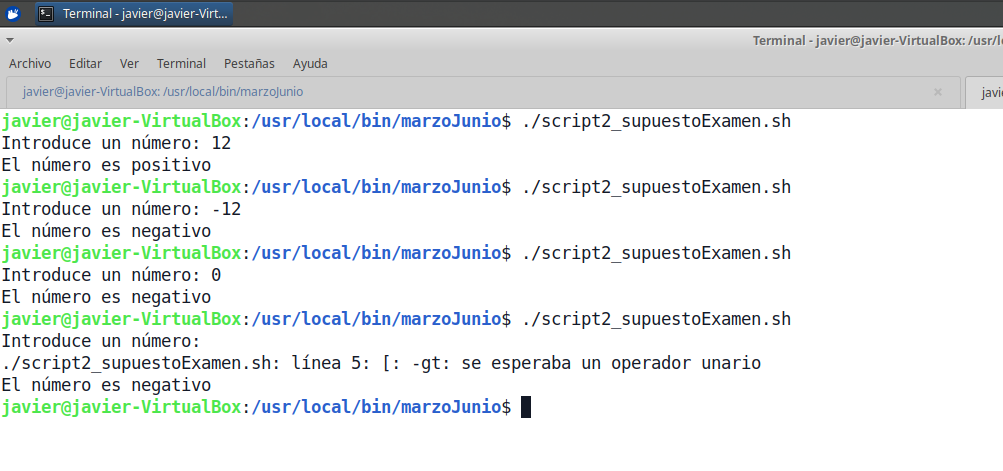


El comando bash -x que permite ir viendo la traza del programa mientras se va ejecutando, o en su defecto, añadir en las líneas que queramos set -x para empezar a depurar por esa línea y set+x para terminar la depuración.



Este script le solicita a el usuario un número por teclado, depués con el condicional if se evalúa la condición si el número introducido por el usuario es mayor que 0, entonces se muestra el mensaje que el número es positivo, en caso contrario si el número introducido es negativo por que es menor que 0, se mostraría el mensaje de que el número es negativo.

* Cuando el usuario mete un 12, se ejecuta el bloque de código del if. Es decir, el número es positivo.
* Cuando el usuario mete un -12, se ejecuta el bloque de código del else, sacando un mensaje por pantalla de que el número es negativo.
* Si el usuario introduce un 0, dirá que el número es negativo, por lo que está mal y este tipo de error es un error lógico.
* Si el usuario mete espacios en blanco dará errores, de tiempo de ejecución, por que se está comparando una cadena vacía con un número.



Esto lo podemos arreglar, cambiando el tipo de expresión númerica a expresiones de cadenas de texto, es decir en vez de -eq ==.

