

4 de diciembre de 2020

Juan Carlos Collado Custodio

2º ASIR – Curso 2019 - 2020

IES Virgen de la Paz

Shell Scripting con Interfaz gráfica para linux

Contenido

[Justificación del proyecto y Objetivos 2](#_Toc57454869)

[Introducción 3](#_Toc57454870)

[Metodología y desarrollo del proyecto 4](#_Toc57454871)

[Elección de interfaz gráfica 4](#_Toc57454872)

[Zenity 4](#_Toc57454873)

[Xmessage 4](#_Toc57454874)

[Gxmessage 5](#_Toc57454875)

[YAD 5](#_Toc57454876)

[Dialog 6](#_Toc57454877)

[Whiptail 7](#_Toc57454878)

[Elección final: Whiptail 7](#_Toc57454879)

[Desarrollo de la interfaz: Estructura 9](#_Toc57454880)

[Desarrollo de la interfaz: Uso de Whiptail 10](#_Toc57454881)

[Desarrollo de la interfaz: Manejo de stdout y stderr 14](#_Toc57454882)

[Desarrollo de la interfaz: Cambio de colores 15](#_Toc57454883)

[Resultados y discusión 16](#_Toc57454884)

[Menú inicial 16](#_Toc57454885)

[Gestión de usuarios 17](#_Toc57454886)

[Gestión de grupos 20](#_Toc57454887)

[Gestión de procesos 24](#_Toc57454888)

[Información del sistema 25](#_Toc57454889)

[Selección de combinación de colores/tema 27](#_Toc57454890)

[Conclusiones 28](#_Toc57454891)

[Bibliografía y referencias 29](#_Toc57454892)

[Anexos / Otros 30](#_Toc57454893)

# Justificación del proyecto y Objetivos

La motivación de este proyecto viene derivada de la flexibilidad que da el Shell scripting a la hora de ajustarse a cualquier situación o necesidad. Al mismo tiempo, siempre he considerado que ante el usuario medio es preferible tener un entorno controlado y limitado en lugar de darle acceso a todas las opciones disponibles del sistema, dado el riesgo que conlleva que cometa un error o sea víctima de un ataque informático.

El objetivo principal de este proyecto es hacer un menú de gestión del Sistema Operativo Linux, pensado para administradores, pero que no son expertos en el uso de la terminal de comandos, o que no tienen responsabilidad para realizar tareas más complejas: un ejemplo puede ser un becario de nueva incorporación en una empresa o un usuario temporal para un proyecto, al que no le quieren dar acceso completo a los sistemas.

Este menú no tiene como finalidad manejar todos los sistemas y servicios del equipo, ya que son muchas las posibilidades y opciones disponibles.

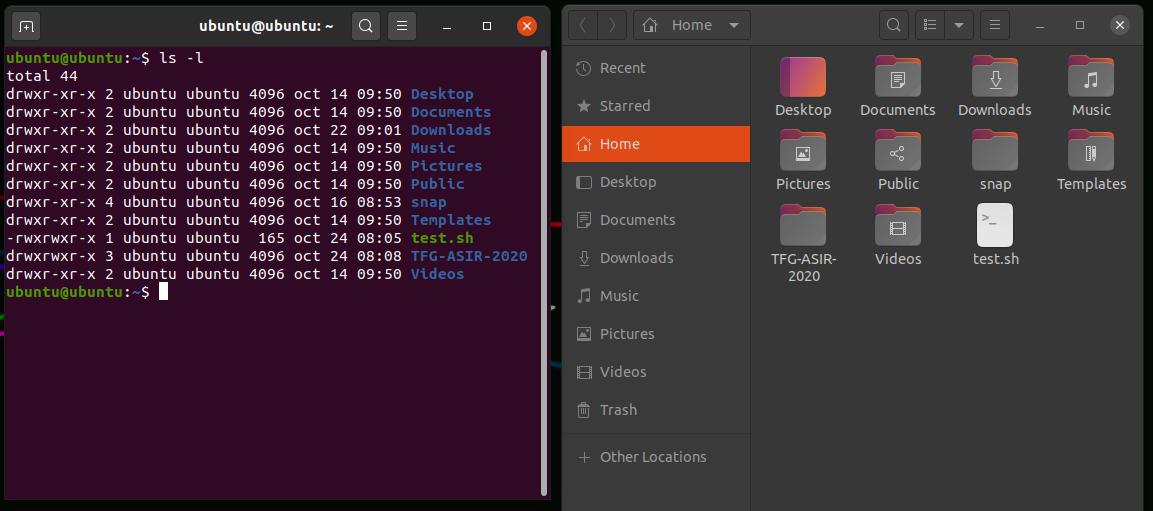
Para simplificar, el menú contendrá dos opciones para la gestión de usuarios y grupos, y otros dos menús de información de procesos y sistema, acabando con un menú adicional que permite cambiar el esquema de colores del propio menú.

# Introducción

La automatización de procesos es una necesidad imperante ante el aumento de la carga de trabajo constante en cualquier departamento de sistemas. Ya sea por ampliación de personal, material o tareas, siempre es necesario liberar parte de esa carga de trabajo en forma de automatización.

Para esto resulta muy útil la herramienta del “Shell scripting”: mediante código se pueden ejecutar tareas simples o complejas, según la necesidad, para atender a diferentes situaciones. Por ejemplo, automatizar la inclusión de nuevos usuarios en un sistema o el control del estado de los servidores.

Si bien el Shell scripting como herramienta es increíblemente útil, no todo el mundo se encuentra cómodo ante la falta de una interfaz gráfica y ante el usuario medio, el aspecto de una terminal de comandos produce una sensación difusa entre la confusión y el miedo a “romper algo”.

La terminal de comandos es una herramienta muy potente que permite realizar tareas de forma mucho más rápida y eficiente que una interfaz gráfica. Sin embargo, la mayoría de usuarios están acostumbrados a tener una interfaz intuitiva, sencilla y en general, a prueba de errores. 

Izquierda: Terminal de comandos. Derecha: Interfaz gráfica.

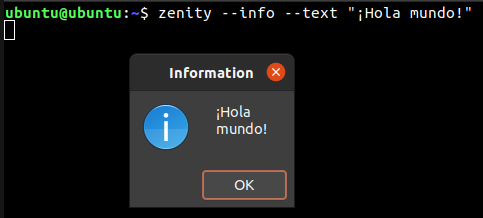
Es por ello que, para juntar ambos elementos, existen herramientas que permiten realizar tareas mediante Shell scripting con una interfaz gráfica que resulte amigable al usuario y permita a los usuarios avanzados personalizar la automatización de forma rápida y sencilla.

# Metodología y desarrollo del proyecto

## Elección de interfaz gráfica

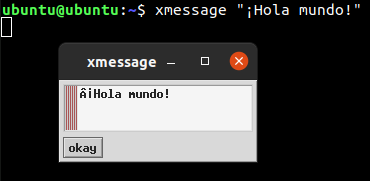
Ante la pregunta de “¿Cómo creo una interfaz para Shell scripting en Linux?” surgen varias herramientas para proporcionar una respuesta. Algunos ejemplos son los siguientes:

### Zenity



Zenity es un programa que usa las bibliotecas GTK+, integrado en GNOME, que despliega ventanas de diálogo o información procesable con scripts en bash. Es muy popular, ya que contiene funciones como calendario, selección de archivos, mensajes de error, información, advertencia, etc. Podemos ver el menú de ayuda con $ zenity –help-all.

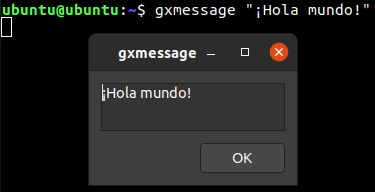
### Xmessage



Al igual que Zenity, xmessage despliega mensajes desde la terminal de comandos. Está disponible en todas las distribuciones que usan X para mostrar el entorno de escritorio, por lo que está incluso más difundido que Zenity. También es bastante antiguo, por lo que (como se ve en la captura) uno se puede encontrar con problemas de codificación del lenguaje.

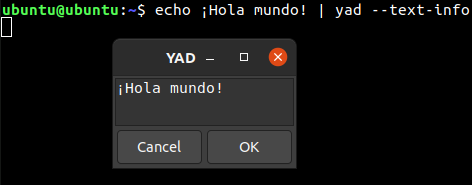
En el caso de xmessage, hay menos opciones para mostrar, ya que por ejemplo no se incluye el calendario.

### Gxmessage

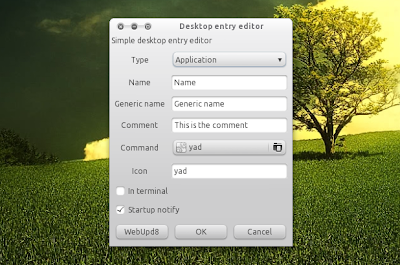


Gxmessage es un clon de xmessage que utiliza la biblioteca GTK+, de modo que combina la funcionalidad de xmessage y el aspecto de Zenity. Permite cambiar colores, tamaño de las ventanas, etc., del mismo modo que con xmessage, con el aspecto de la interfaz GNOME.

### YAD

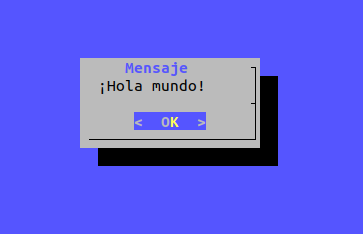


YAD (*Yet Another Dialog*) es una derivación (*fork*) de Zenity con muchas mejoras implementadas.

El proyecto Zenity recibe actualizaciones constantemente, sin embargo, éstas se limitan a mejoras de idioma y corrección de errores. Aquí es donde YAD brilla: casi todas las cosas que limitan a Zenity están disponibles en YAD. Éste es un ejemplo de una ventana que no se podría crear en Zenity:

Para instalarlo basta con lanzar $ sudo apt install yad en Ubuntu, o añadir el repositorio en otras distribuciones. Se pueden ver todas las opciones disponibles con $ yad –help-all.

### Dialog



Dialog es un programa dependiente de la biblioteca *ncurses* (*new curses*), que al contrario que la mayoría de los programas anteriores, no necesita ni siquiera un entorno de escritorio para mostrarse. Es muy sencillo de utilizar y conveniente debido a que los scripts con dialog funcionan incluso a través de una terminal SSH.

Mediante dialog podemos mostrar ventanas de tipo menú, mensajes/información, tipo checklist, radiolist, consultas si/no o pedir un input al usuario, etc. Toda esta información es utilizable mediante scripting para simplificar las tareas a los usuarios o limitarles las opciones disponibles.

### Whiptail



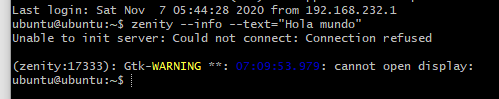
Whiptail es parte de la librería Newt, que a su vez depende de la biblioteca *curses*, escrita en C, y se encuentra disponible en la mayoría de las distribuciones de Linux. Aunque Newt todavía se encuentra en desarrollo, Whiptail está completado como programa, es decir, todas sus características están ya incluidas en lo que se refiere a ventanas y elementos de interfaz.

Para instalar Whiptail, basta con instalar el paquete newt mediante nuestro gestor de paquetes.

Whiptail comparte muchas características con Dialog. De hecho, casi todos los parámetros son similares, por lo que a primera vista sería indiferente usar una opción u otra. Sin embargo, Whiptail se encuentra instalado por defecto en casi todas las distribuciones basadas en Debian, y en el aspecto estético está más elaborado.

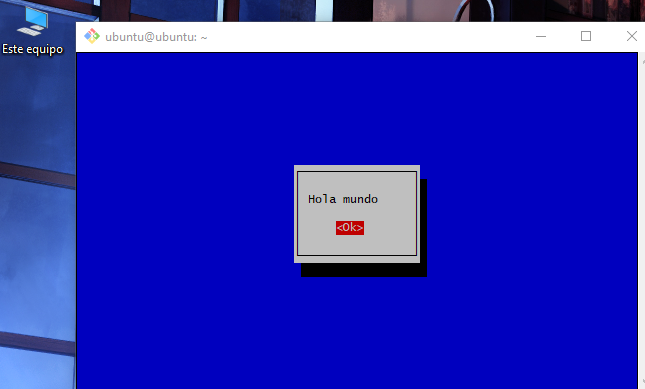
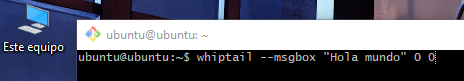
## Elección final: Whiptail

Si bien las opciones basadas en GTK+ son visualmente muy agradables, hay que tener en cuenta que no pueden funcionar de forma remota o en sistemas sin entorno de escritorio instalado (como es habitual en servidores).



Ejemplo: Lanzar Zenity por terminal SSH no es posible

Sin embargo, tanto Dialog como Whiptail, al no necesitar interfaz gráfica, funcionan sin ningún problema en la terminal SSH:



Ejemplo de Whiptail vía SSH en una terminal en Windows

Por este motivo, y por la sencillez de uso, me decanto por Whiptail como herramienta para desarrollar la interfaz del proyecto.

## Desarrollo de la interfaz: Estructura

La interfaz contiene varios menús, de los cuales en algunos casos derivan otros submenús. Se centran en 5 apartados concretos, con la siguiente estructura:

1. Gestión de usuarios
   1. Añadir o crear nuevos usuarios
   2. Modificar información de usuarios
      1. Nombre completo (Nombre y Apellidos)
      2. Departamento del usuario
      3. Teléfono de empresa
      4. Teléfono personal
      5. Observaciones
   3. Eliminar usuarios
   4. Mostrar información del usuario actual
2. Gestión de grupos
   1. Añadir o crear nuevos grupos
   2. Modificar información de grupos
   3. Eliminar grupos
   4. Consultar a qué grupos pertenece el usuario
3. Información de Procesos del Sistema
   1. Ver procesos activos del sistema
   2. Ver procesos activos del usuario actual
   3. Ver los 10 procesos que más consumen en este momento
4. Información de Servicios del Sistema
   1. Información general del equipo
   2. Cantidad de memoria RAM en uso y disponible
   3. Tiempo que lleva el equipo en marcha
   4. Capacidad disponible de discos duros / particiones
5. Menú de cambio de colores del programa

Antes de abrirse el menú inicial, el script comprueba si el usuario que lanza el script tiene permiso de administrador. Este requisito se debe a que las gestiones de algunos menús requieren permiso de administrador para funcionar, y en caso de no tenerlo, el script no ejecuta esas funciones correctamente.

Existe la posibilidad de solicitar permisos de administrador específicamente en las opciones que lo requieren, pero este detalle expulsa momentáneamente al usuario a la terminal de Linux y puede resultar confuso.

## Desarrollo de la interfaz: Uso de Whiptail

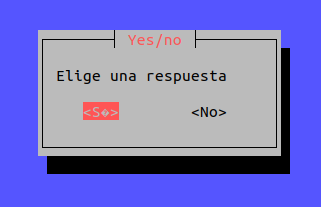
El uso de Whiptail como comando se reduce a unas pocas opciones después del comando. En general, la sintaxis es la siguiente:

$ whiptail –title “Título” –msgbox “Texto del mensaje” *altura anchura*

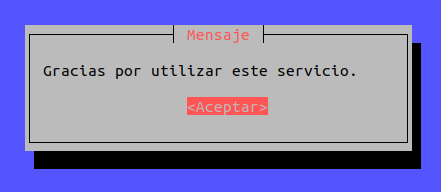
La altura y anchura se indican en columnas. Se pueden poner ambas a 0 para que se ajuste la caja al contenido, pero no siempre funciona correctamente (depende del contenido). El título tampoco es imprescindible, pero es recomendable para saber el contexto del mensaje. Sí es necesario poner estos argumentos, o de lo contrario Whiptail nos muestra un mensaje de error.

El argumento –msgbox puede ser sustituido por el tipo de caja de diálogo que queramos, que pueden ser los siguientes:

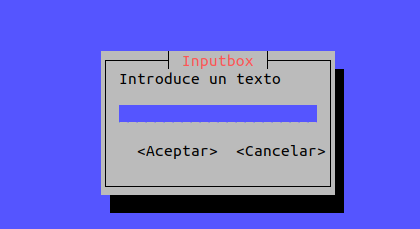
* --yesno *texto altura anchura* : Esta caja de diálogo muestra una pregunta tipo booleana (sí/no) definida por el campo texto, de una altura y anchura especificadas. Si elegimos la opción Si, Whiptail nos devuelve el valor 0. Si elegimos No, nos devuelve el valor 1, y si salimos pulsando Esc o se produce un error, nos devuelve el valor -1.



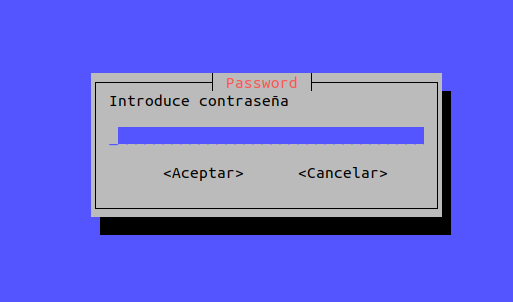
* --msgbox *texto altura anchura* : Este mensaje es similar al anterior, con la excepción de que se muestra solamente un botón de Ok, cuyo texto también se puede cambiar. Este mensaje se muestra hasta que el usuario pulsa el botón.



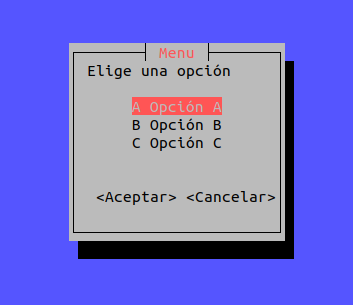
* --infobox *texto* *altura anchura* : Este mensaje de información es similar al anterior, sin embargo desaparece tras mostrarse al usuario sin esperar su interacción, por lo que es recomendable usarlo durante un proceso para informar al usuario de que debe esperar a que termine de ejecutarse.
* --inputbox *texto* *altura anchura* : Esta caja de diálogo es simplemente una petición al usuario de información, cuya salida se envía al stderr y que hay que recuperar para poder usar la cadena en otras partes de nuestro script. Esto se explica más adelante.



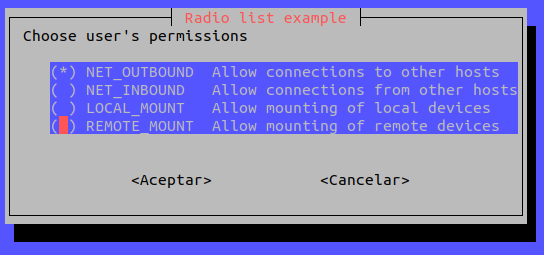
* --passwordbox *texto* *altura anchura* : De forma similar al inputbox, esta caja de diálogo acepta una entrada por parte del usuario, pero sin mostrar lo que se introduce. No se realiza ningún tipo de encriptación durante el proceso, por lo que el manejo de la cadena de texto debe realizarse con cautela para evitar lecturas no deseadas.



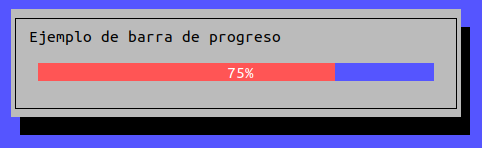
* --textbox *texto* *altura anchura* : Este mensaje simplemente muestra el contenido de un archivo de texto en la caja de diálogo, como un visor de textos típico. Se pueden usar las teclas RePág y AvPág, al igual que Inicio y Fin para moverse por el documento. Si las líneas son demasiado largas, las teclas de dirección también sirven para desplazarse.
* --menu *texto altura anchura altura\_del\_menú* [*etiqueta item*] : Tal y como sugiere el nombre, esta caja de diálogo es un menú que sirve para presentar varias opciones al usuario, de las que debe elegir una. Cada línea del menú está compuesta por una etiqueta y un ítem. La etiqueta distingue cada entrada de las demás, mientras que el ítem es una breve descripción de lo que representa esa opción. El usuario puede desplazarse entre las opciones con las flechas de arriba/abajo. Si bien la documentación encontrada indica que se puede usar la primera letra de la etiqueta como atajo de teclado, este aspecto no funciona correctamente con números.



* --checklist *texto altura anchura altura\_de\_la\_lista* : De forma similar a la opción menu, checklist nos muestra una lista de opciones. Sin embargo, éstas aparecen con un cuadro tipo check, que podemos dejar marcado o no por defecto con la opción ON u OFF junto a la línea correspondiente. Se pueden elegir una, varias opciones, o ninguna, según como hayamos configurado el script. Para seleccionar hay que utilizar la barra espaciadora.
* --radiolist *texto altura anchura altura\_de\_la\_lista* : Al igual que con los dos tipos de elementos anteriores, radiolist nos muestra una lista de opciones de las cuales debemos elegir una exclusivamente. Para desplazarnos entre las opciones usamos las flechas de arriba/abajo y seleccionamos con la barra espaciadora.



* --gauge *texto altura anchura porcentaje* : Esta ventana muestra una barra de progreso, que conviene utilizar junto a un bucle for para mostrar su avance hasta que termine dicho bucle. Es conveniente para indicar al usuario que debe esperar a que termine un proceso que sabemos que tiene un tiempo determinado.



## Desarrollo de la interfaz: Manejo de stdout y stderr

Al observar el uso de Whiptail en el script, en especial en las opciones que tienen que devolver un valor, se puede ver que al final del comando hay una serie de indicaciones que a primera vista no tienen explicación aparente.

3>&1 1>&2 2>&3

Esto se debe a la forma que tiene Whiptail de manejar las salidas stdout y stderr, es decir, la salida estándar (stdout) y el error estándar (stderr). Por defecto, Whiptail envía un resultado 0 o 1 si la ejecución es correcta o incorrecta respectivamente. El resultado de un inputbox por ejemplo, devuelve un 0 por el stdout si el usuario ha introducido un dato y un 1 si no introduce nada. Por otro lado, Whiptail utiliza el stderr como destino del input del usuario.

Esto significa que, si utilizamos Whiptail dentro de una variable, el resultado que se guarda en esa variable sería 0 o 1 en lugar del input del usuario.

Es por esto que realizamos un intercambio de los valores stdout y stderr al final del comando, que se lee de derecha a izquierda:

* 2>&3 envía la salida de stderr a una salida Nº 3 que creamos para este caso
* 1>&2 envía la salida del stdout al stderr
* 3>&1 redirige la salida que guardamos del stderr al stdout

La salida Nº 3 se puede ignorar ya que solo nos sirve para este intercambio.

## Desarrollo de la interfaz: Cambio de colores

Debido a la muy limitada documentación del programa Whiptail, no ha sido fácil encontrar los parámetros que permiten el cambio de color de la interfaz. Siguiendo las indicaciones de [esta consulta de StackOverflow](https://stackoverflow.com/questions/46619539/changing-colours-in-whiptail) y [esta otra de AskUbuntu](https://askubuntu.com/questions/776831/whiptail-change-background-color-dynamically-from-magenta/781062#781062), he determinado que los valores que definen los colores de la interfaz se encuentran en la variable de entorno $NEWT\_COLORS, y que son modificables mediante el comando export y definiendo los valores según una lista concreta.

case $colormenuoption in

            A)

*#Oscuro*

                export NEWT\_COLORS='

                root=,gray

                window=white,black

                title=magenta,black

                border=white,black

                textbox=white,black

                button=magenta,black

                actbutton=black,magenta

                listbox=white,black

                '

                whiptail --msgbox "Muestra de colores" 0 0

                ;;

La paleta de colores por defecto es intercambiable con el comando  
sudo update-alternatives --config newt-palette.

# Resultados y discusión

## Menú inicial

El script se inicia con una comprobación de si el usuario es administrador del equipo. Este hecho es necesario para la gestión de usuarios y grupos, y para algunas opciones de otros menús que requieren herramientas con permiso de administrador.

if [ $UID != 0 ]; then

    whiptail --title "Error" --msgbox "Este script se debe lanzar como Administrador (sudo).\nSe cerrará el programa tras este aviso." 0 0

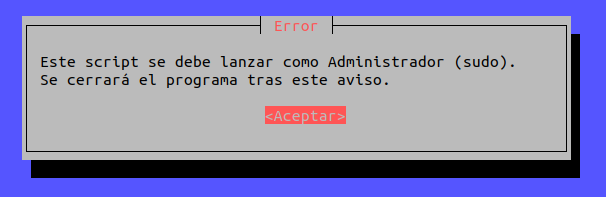
    exit

else

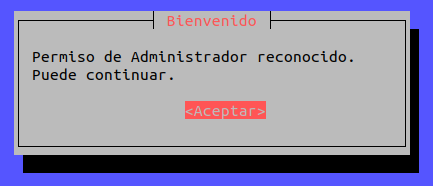
    whiptail --title "Bienvenido" --msgbox "Permiso de Administrador reconocido.\nPuede continuar." 0 0

fi

Si lanzamos el script sin permiso de administrador, se nos mostrará el siguiente aviso:



En caso de lanzar el script correctamente como usuario Administrador, el aviso será el siguiente:



Y se nos mostrará el menú principal, consistente en un menú en Whiptail del que deriva un comando case, en el cual se encuentran anidados el resto de menús:

mainmenu() { *#De este menú derivan el resto de submenús.*

    mainmenu\_option=$(

        whiptail --title "Administración del Sistema" --nocancel --menu "Elige una opción" 0 0 6 \

        "A" "Gestión de usuarios." \

        "B" "Gestión de grupos" \

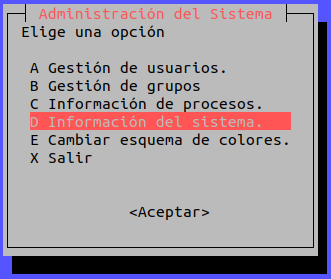
        "C" "Información de procesos." \

        "D" "Información del sistema." \

        "E" "Cambiar esquema de colores." \

        "X" "Salir" 3>&1 1>&2 2>&3)

}



Las opciones son seleccionables con las teclas de dirección o con las letras A-E y X para la opción de salir.

## Gestión de usuarios

El menú de gestión de usuarios incluye las opciones de crear, modificar o eliminar usuarios, y ver información los usuarios conectados al sistema. Ésta última opción requiere el paquete “finger”, disponible en casi todas las distribuciones de Linux. En caso de no tenerla instalada, la opción mostrará un aviso solicitando su instalación para funcionar correctamente.

user\_mgt\_menu() {

    usermenu\_option=$(whiptail --title "Gestión de usuarios" --nocancel --menu "Seleccione una opción" 0 0 5 \

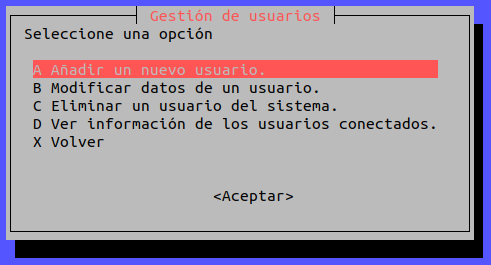
        "A" "Añadir un nuevo usuario." \

        "B" "Modificar datos de un usuario." \

        "C" "Eliminar un usuario del sistema." \

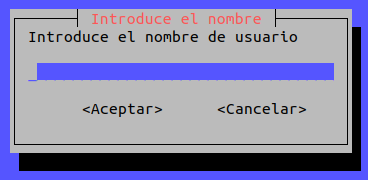
        "D" "Ver información de los usuarios conectados." \

        "X" "Volver" 3>&1 1>&2 2>&3)}



La creación de usuarios es bastante simple: Se solicita un nombre de usuario, una contraseña y su confirmación, y se pregunta si el usuario se debe unir a un grupo especificado.

nombre\_add=$(whiptail --title "Introduce el nombre" --inputbox "Introduce el nombre de usuario" 0 0 3>&1 1>&2 2>&3)

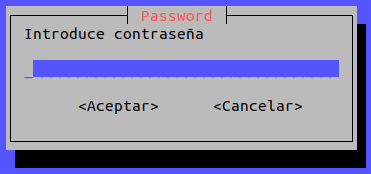


password\_ask() {

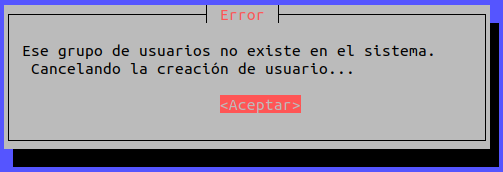
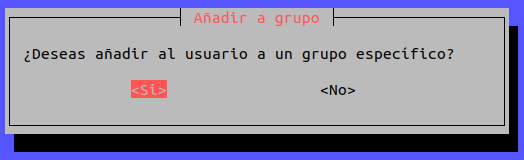
    password=$(whiptail --title "Password" --passwordbox "Introduce contraseña" 8 39 3>&1 1>&2 2>&3)

    passwordcheck=$(whiptail --title "Password" --passwordbox "Introduce confirmación de contraseña" 8 39 3>&1 1>&2 2>&3)

}



La unión del usuario a un grupo solo se realiza si el grupo de usuarios existe previamente en el equipo, de lo contrario se mostrará un mensaje de error y no se realizará la creación del usuario.



En caso de tener el paquete finger instalado, la opción D del menú de gestión de usuarios mostrará el siguiente mensaje, añadiendo más información conforme haya más usuarios conectados. Para comprobar si el paquete finger está instalado, se utiliza el comando which, que comprueba la ruta de instalación de un paquete indicado.

which finger >/dev/null

if [[ $? -eq 0 ]]; then

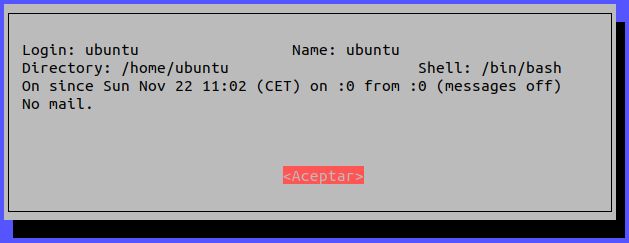
whiptail --textbox /dev/stdin 12 0 <<<"$(finger -lp)"

else

whiptail --title "Paquete necesario" --msgbox "Se requiere la instalación del paquete 'finger'

para el funcionamiento correcto de esta opción.\n\

Puedes instalarlo con el comando '$ sudo apt install finger' o el equivalente de tu distribución de Linux." 0 0



Desafortunadamente el paquete finger no se encuentra en otros idiomas, por lo que la información se muestra en inglés. Los datos que se muestran son:

* Nombre de usuario
* Nombre completo
* Directorio /home
* Tipo de Shell
* Tiempo que lleva el usuario conectado y en qué terminal
* Si está configurado en el sistema, correos recibidos sin leer.

## Gestión de grupos

El menú de gestión de grupos de usuarios funciona de forma similar al menú de gestión de usuarios mostrado anteriormente. Las opciones disponibles son las siguientes:

group\_mgt\_menu() {

    groupmenu\_option=$(

        whiptail --title "Gestión de grupos" --nocancel --menu "Seleccione una opción" 0 0 7 \

        "A" "Añadir un nuevo grupo." \

        "B" "Modificar datos de un grupo." \

        "C" "Eliminar un grupo del sistema." \

        "D" "Ver usuarios de un grupo concreto." \

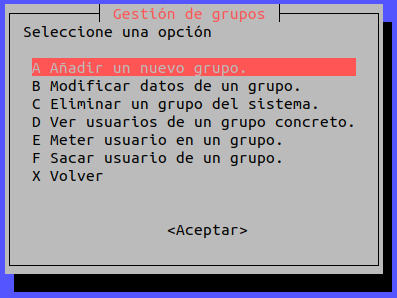
        "E" "Meter usuario en un grupo." \

        "F" "Sacar usuario de un grupo." \

        "X" "Volver" 3>&1 1>&2 2>&3

    )

}



La creación de un grupo es un proceso muy básico: se solicita un nombre de grupo al usuario mediante un cuadro de diálogo de Whiptail, y una vez introducido el nombre, se crea el grupo.

group\_add=$(whiptail --title "Crear grupo" --inputbox "Introduce el nombre de grupo" 0 0 3>&1 1>&2 2>&3)

if [[ -z "$group\_add" ]]; then

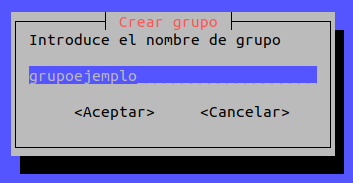
whiptail --title "Error" --msgbox "No has introducido un nombre para el grupo.\n Volviendo al menú anterior..." 0 0

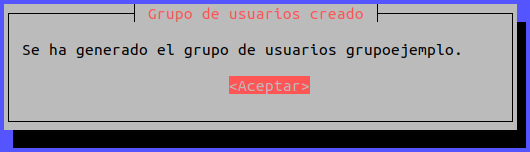
break

fi

groupadd "$group\_add"

whiptail --title "Grupo de usuarios creado" --msgbox "Se ha generado el grupo de usuarios $group\_add." 0 0





La eliminación de grupos de usuarios sigue el mismo proceso, con el detalle de que, si el grupo a eliminar es un grupo protegido del sistema, no se realiza la eliminación:

group\_del\_safetycheck=$(getent group $group\_del | cut -d: -f3)

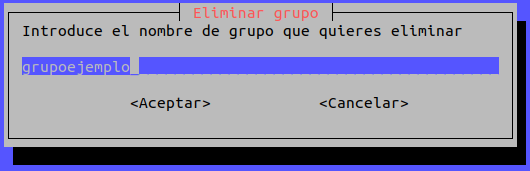
if [[ $group\_del\_safetycheck -lt 999 || $group\_del\_safetycheck -gt 65534 || -z "$group\_del\_safetycheck" ]]; then

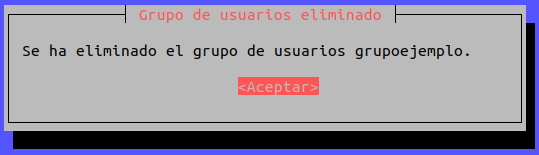
                    whiptail --title "Error" --msgbox "Ese grupo de usuarios no existe o no se puede eliminar.

\nVolviendo al menú anterior..." 0 0

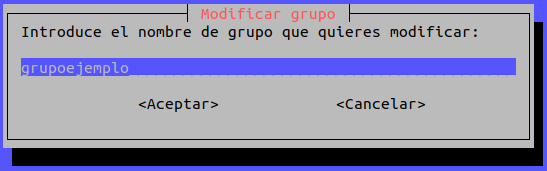
break

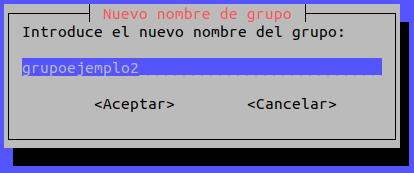
fi

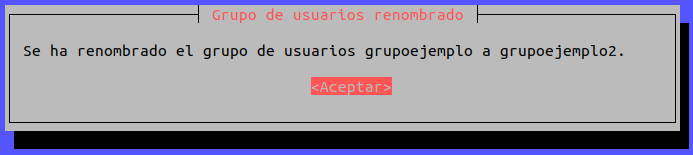




La modificación del nombre solicita el nombre del grupo a modificar y el nuevo nombre para el grupo.

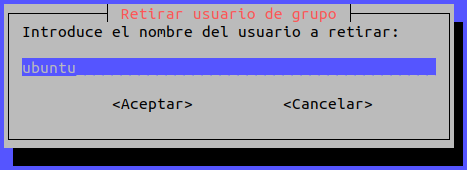
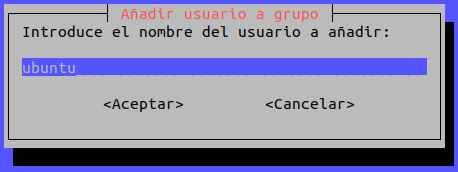


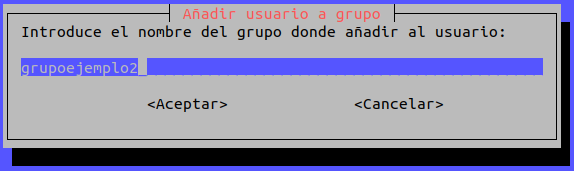


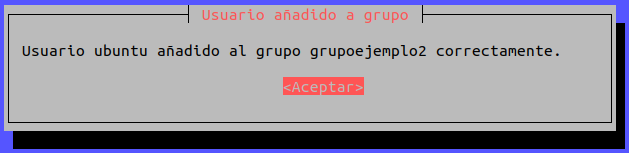


Las funciones de añadir y retirar usuario de grupo son también bastante simples: se solicita el nombre del usuario y del grupo en cada caso y se realiza la función correspondiente si los parámetros son aceptados.

En todos los campos se realizan verificaciones por si el dato que introduce el usuario está en blanco o no es válido.

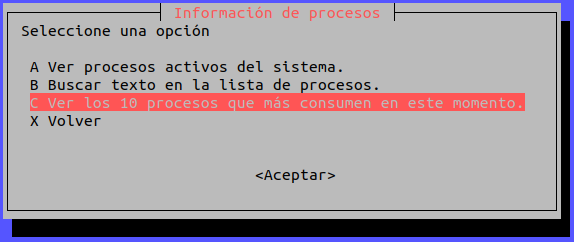






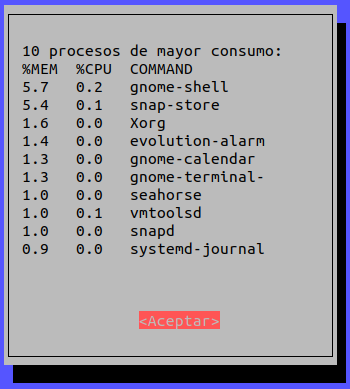
## Gestión de procesos

Al contrario que los menús anteriores, el menú de gestión de procesos no realiza una gestión *per se*, si no que muestra información en varias opciones. Esto se debe a que el control de procesos es un sistema muy concreto en lo que se refiere a consultar información, ya que los comandos para controlar procesos tienen demasiados argumentos como para hacer un menú con cada uno de ellos.



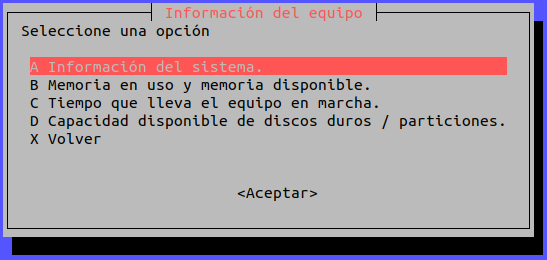
Es por esto que este menú se limita a mostrar información de los procesos del sistema, a modo de ejemplo de lo que se puede hacer con una interfaz gráfica como es Whiptail. Evidentemente, según el entorno y las necesidades del Administrador o de la empresa en cuestión, estos scripts se pueden modificar para mostrar la información deseada en lugar de los ejemplos que se muestran.

Uno de estos ejemplos es el siguiente mensaje, que lista el resultado de un comando ps, ordenando los resultados por consumo de memoria, y mostrando los diez primeros, es decir: mostrar los diez procesos que mayor consumo de memoria tienen en el sistema en el momento de la consulta.



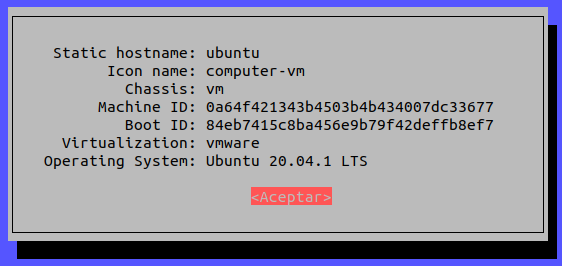
## Información del sistema

De forma similar a la gestión de procesos, el menú de Información del Sistema solo muestra información en varias opciones relacionadas con el estado del equipo. Entre ellas se muestran las siguientes:

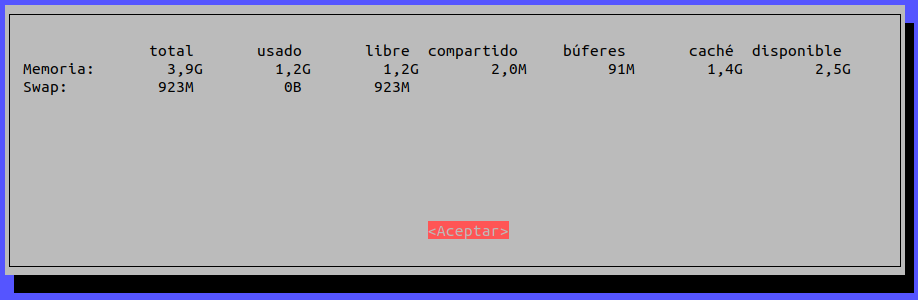


El menú A es el resultado del comando hostnamectl, que muestra información básica del sistema. Para ello, Whiptail lee el resultado del comando desde /dev/stdin, donde se aloja temporalmente el resultado de dicho comando, para no tener que utilizar variables o archivos de texto.

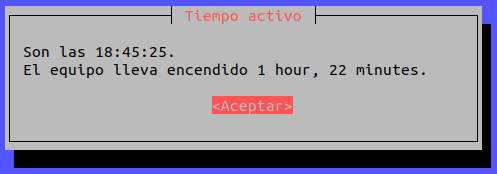
whiptail --textbox /dev/stdin 0 0 <<<"$(hostnamectl)"



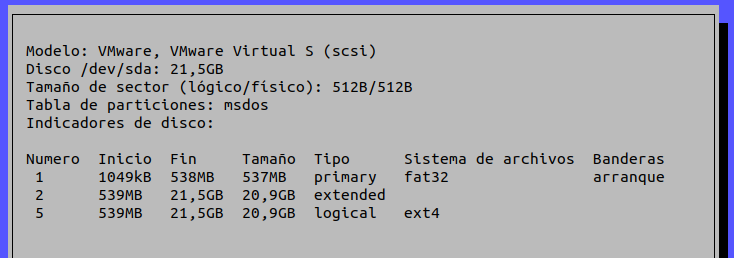
De forma similar, la opción B del menú muestra el resultado del comando free adaptado con varios argumentos para mejor visibilidad.



La opción C mostraba originalmente el resultado del comando uptime, pero para mayor comprensión añadí el resultado del comando a una variable para poder manipularla después. Desafortunadamente el comando uptime solo está disponible en inglés, por lo que parte del mensaje aparece en este idioma.



Por último, la opción D es el resultado del comando parted -l, que requiere permisos de administrador para funcionar, lo que motivó también el requerimiento de lanzar el script como administrador.

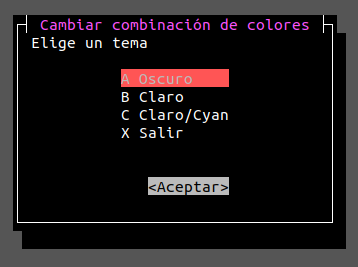


## Selección de combinación de colores/tema

Al no contar con una documentación al uso del programa Whiptail, originalmente no concebí la posibilidad de cambiar los colores de los elementos del menú. Estas opciones son el resultado de mucha investigación y pruebas de ensayo/error, hasta descubrir qué elementos cambiaban de color con cada parámetro.

Como resultado, pude añadir las opciones de cambio de color, con la posibilidad de que el cambio durara solamente durante la sesión en la que se lanza el script, es decir, no es un cambio permanente.

Las opciones de cambio de color contienen 3 opciones: tema oscuro, tema claro, y una variación de éste último con color cyan.



Muestra del menú con tema oscuro

# Conclusiones

Ante el usuario medio o inexperto, es sabido que el uso de la terminal para manejar el sistema operativo es confuso y poco intuitivo. Siempre es preferible mostrar una interfaz con opciones limitadas que impidan que el usuario “meta la pata” y pueda poner en peligro la integridad del sistema operativo.

De forma similar al desarrollo de aplicaciones web, el desarrollo de una un Shell script con interfaz gráfica necesita de un back-end donde se realizan las operaciones y un front-end para el usuario. Por mucho que sepamos desarrollar un back-end que funcione a la perfección, si el desarrollo de la interfaz no es adecuado, podemos confundir al usuario y llegar a un resultado no deseado.

Éste ha sido el caso, durante el desarrollo del script, al intentar utilizar un menú tipo radiolist en una de las opciones: la selección se debía realizar con la barra espaciadora, y no había ninguna indicación de este hecho, lo que provocaba confusión entre los usuarios que testeaban el script.

El desarrollo de una interfaz debe tener como objetivo principal la comprensión y la facilidad de uso, centrándose en un aspecto intuitivo que dirija la atención del usuario al campo que queremos, y que no utilice términos o expresiones que den lugar a error.

La experiencia del usuario es primordial a la hora de desarrollar la interfaz, por lo que siempre se debe tener en cuenta el punto de vista del usuario y ver nuestro trabajo como alguien que nunca ha utilizado este sistema. Si es posible, contar con conocidos o compañeros ajenos al proyecto y que puedan echarle un vistazo ayuda a encontrar problemas que de otra forma se nos pasarían por alto.

# Bibliografía y referencias

* Stack Overflow: [How to show a GUI message box from a bash script in linux?](https://stackoverflow.com/questions/7035/how-to-show-a-gui-message-box-from-a-bash-script-in-linux)
* Linux Man Page: [Zenity](https://linux.die.net/man/1/zenity)
* Linux Man Page: [xmessage](https://linux.die.net/man/1/xmessage)
* SysTutorials: [gxmessage](https://www.systutorials.com/docs/linux/man/1-gxmessage/)
* WebUpd8.org: [YAD: Zenity on steroids](http://www.webupd8.org/2010/12/yad-zenity-on-steroids-display.html)
* AplicacionesySistemas.com: [Cómo crear menús en tus scripts](https://aplicacionesysistemas.com/dialog-crear-menus-tus-scripts/)
* Usuario *wafsek* en gist.github.com: [An example of a Whiptail menú with functions](https://gist.github.com/wafsek/b78cb3214787a605a28b)
* Usuario *i2shar* en github.com: [Whiptail command line user interface demo](https://github.com/i2shar/whiptail-demo)
* Usuario *JazerBarclay* en github.com: [Whiptail examples](https://github.com/JazerBarclay/whiptail-examples)
* Tecmint.com: [Using Shell Scripting to Automate Linux System Maintenance Tasks – Part 4](https://www.tecmint.com/using-shell-script-to-automate-linux-system-maintenance-tasks/)
* Wikibooks.org: [Bash Shell Scripting/Whiptail](https://en.wikibooks.org/wiki/Bash_Shell_Scripting/Whiptail)
* Bradgillap.com: [Bash GUI Whiptail Menu Tutorial Part 1](https://www.bradgillap.com/guide/post/bash-gui-whiptail-menu-tutorial-series-1)

# Anexos / Otros

* Código ejecutable: “script.sh”
* Enlace al repositorio en Github: <https://github.com/SmokingChildren/TFG-ASIR-2020>