TUI: Text User Interfaces en GNU/Bash y Python

*— Eugenia Bahit agradece a **Hugo** (@huguidugui) por la **revisión ortográfica** de este artículo*

La creación de interfeces de Texto parece ser algo olvidado en este Siglo. Sin embargo, en el mundo de los servidores donde los entornos gráficos no dejarán de ser una molestia innecesaria, las TUI no parecen tener fecha de caducidad.

Mucho se habla (y hablo) de las GUI (*Graphical User Interfaces*) pero de las **TUI** (*Text User Interfaces*) nos olvidamos de su existencia hasta que aparece alguien que nos lo recuerda y eso, fue lo que sucedió días atrás en *Twitter* cuando un usuario me preguntaba cómo crearlas.

A raíz de lo anterior y tras haberle comentado que antiguamente, empleaba `dialog` para GNU/Bash y `ncurses` en C y Python. *Dialog*, siempre me había resultado simple y sencillo de utilizar y jamás me había dejado con «sensación de insatisfacción» ya que tenía todo que un programador podría necesitar para la TUI de su *Script* o aplicación y entonces me surgió la pregunta **¿por qué no utilizaba `dialog` en Python?**

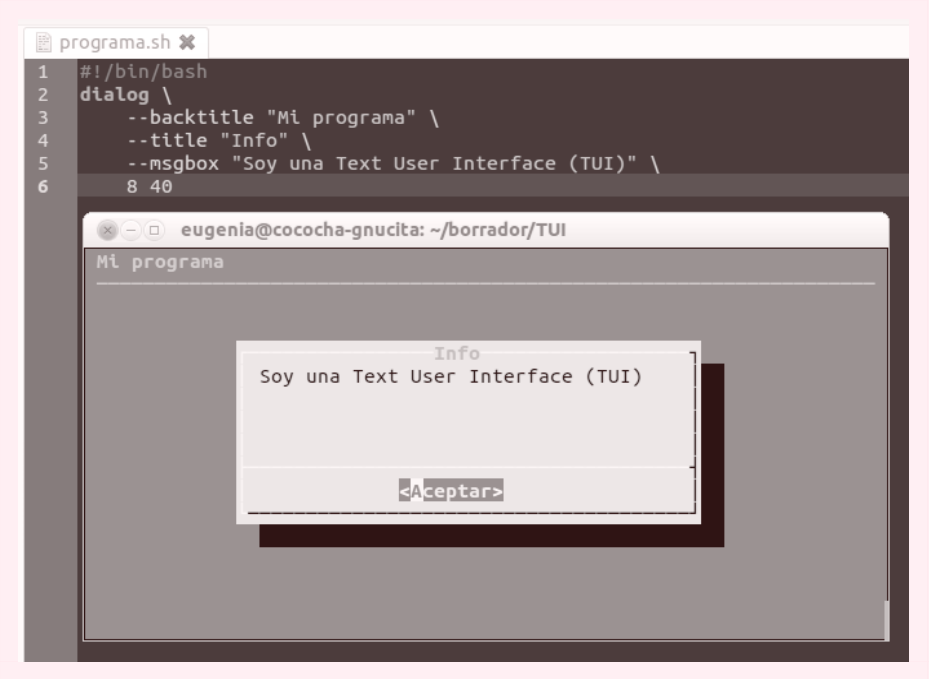
Y así fue que me surgieron unas ganas tremendas de escribir una guía sobre este tema del cual en más de 15 años no he escrito nunca. ¡Espero que la disfruten!

Quiero comenzar primero por introducirlos en el mundo `dialog` de GNU/Bash para que puedan ir familiarizándose con esta herramienta. El **sitio Web oficial** de `dialog` es http://invisible-island.net/dialog/

> *Dialog comenzó en 1994 de la mano del programador **Savio Lam** pero desde 1999 es escrito y mantenido por **Thomas E. Dickey**, creador de **Lynx**, **ncurses** y **xterm** -entre muchas otras herramientas- quien lamentablemente **decidió cambiar la original licencia GPL** del programa por **LGPL**, permitiendo así que cualquier persona pudiese crear mejoras privativas alejándolas así del alcance de la comunidad y de su consecuente estudio.*

¿Qué es dialog?

Dialog es una herramienta que provee de pseudo interfaces gráficas (o interfaces gráficas textuales) para nuestras aplicaciones de consola.



Entre los tipos de «artilugios» disponibles, se encuentran los cuadros de diálogo (si/no), calendarios, entradas de texto, listas desplegables, etc.

¿Cómo se obtiene?

Dialog puede instalarse directamente desde repositorios. En **Debian** y derivados:

```
# apt-get install dialog
```

Casos de Uso

El uso más frecuente de las TUI es en el *Scripting* y programas de consola en general. Mediante el comando `man dialog` se puede acceder a toda la ayuda del programa y obtener así, detalles claros sobre su sintaxis y argumentos, lo que nos podría dar una idea más amplia de sus posibles usos.

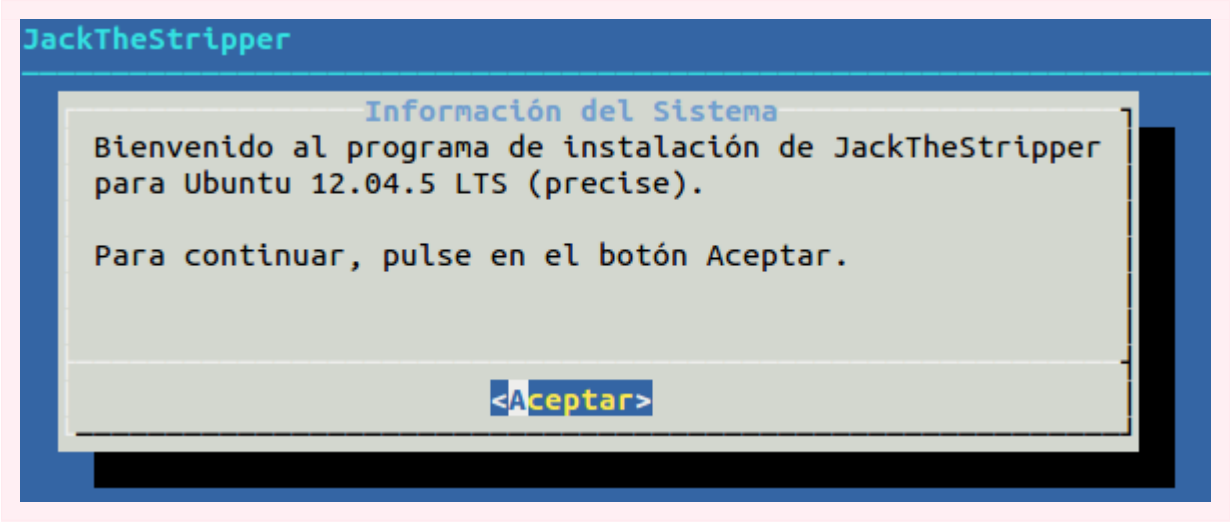
La sintaxis básica para utilizar `dialog` es:

```
dialog --opciones-comunes --tipo-de-diálogo "Contenido" alto ancho --opciones-del-diálogo
```

Caja de diálogo normal: Mostrando la información del sistema operativo

```
#!/bin/bash
# Archivo: programa.sh

system_info=`lsb_release -ds`
system_codename=`lsb_release -cs`

read -d '' msg_bienvenida << EOF
Bienvenido al programa de instalación de JackTheStripper para $system_info
($system_codename).

Para continuar, pulse en el botón Aceptar.
EOF

dialog \
    --backtitle "JackTheStripper" \
    --title "Información del Sistema" \
    --msgbox "$msg_bienvenida" \
    10 60
```

El código anterior producirá un diálogo como el siguiente:



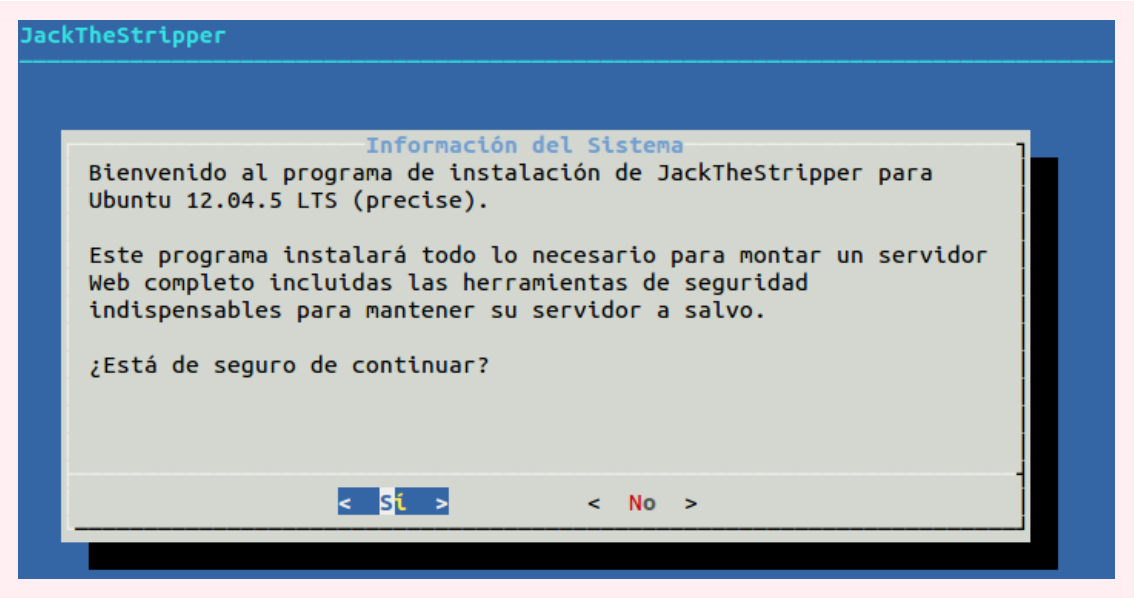
Si deseas probar el código anterior (y los siguientes) y aún no sabes cómo hacerlo, abre un terminal y procura seguir estos pasos:

1. Crear un archivo: `touch programa.sh`
2. Asígnale permisos de ejecución: `chmod +x programa.sh`
3. Abre el archivo anterior con cualquier editor de textos, copia el código y pégalo en el archivo programa.sh.
4. Tras guardar el archivo, ejecútalo desde la terminal escribiendo: `./programa.sh`

Diálogo de confirmación: Confirmando antes de actuar

```
#!/bin/bash
system_info=`lsb_release -ds`
system_codename=`lsb_release -cs`

read -d '' msg_bienvenida << EOF
Bienvenido al programa de instalación de JackTheStripper para $system_info
($system_codename).

Este programa instalará todo lo necesario para montar un servidor Web completo incluidas las
herramientas de seguridad indispensables para mantener su servidor a salvo.

¿Está de seguro de continuar?
EOF

dialog \
    --backtitle "JackTheStripper" \
    --title "Información del Sistema" \
    --yesno "$msg_bienvenida" \
    15 70
```

El código anterior producirá un diálogo como el siguiente:



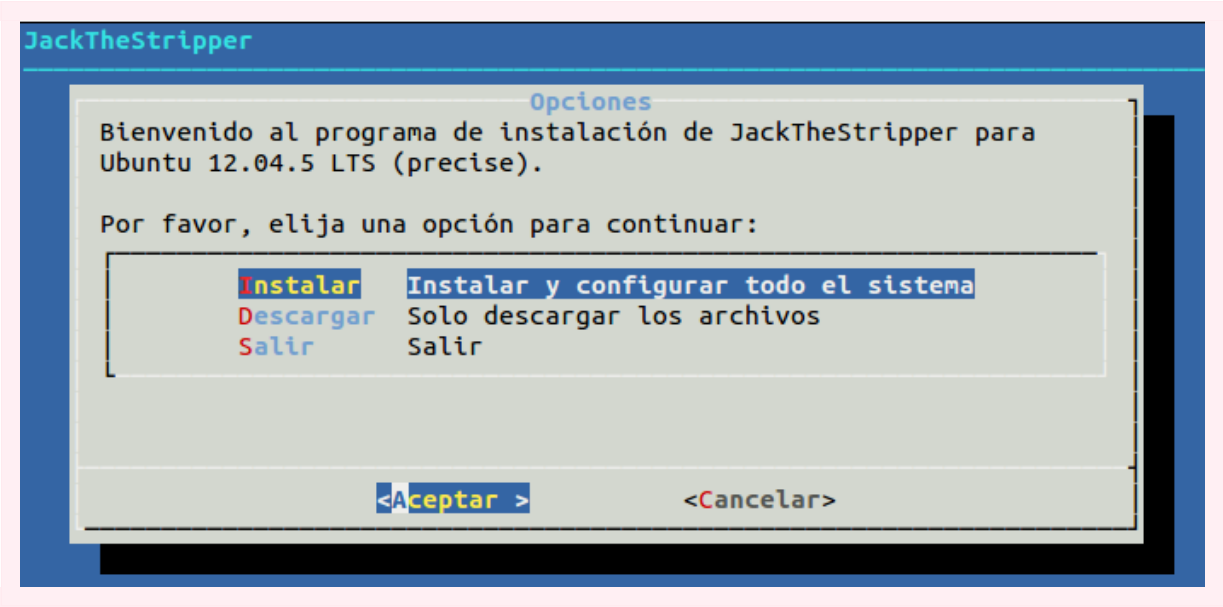
La opción elegida por el usuario, será capturada y representada como: 0=>Sí, 1=>No, 255=>cualquier tecla de escape que canceló la operación. Un ejemplo:

```
respuesta=$?

case $respuesta in
    0) cd cd deploymyserver; sudo ./dms.sh;;  # ejecuta el archivo de instalación
    1) rm -R deploymyserver;;  # Elimina los archivos descargados
    255) echo "Bye!";;   # No hace nada e imprime un mensaje en pantalla
esac
```

Menú: Eligiendo opciones de un menú

```
#!/bin/bash
system_info=`lsb_release -ds`
system_codename=`lsb_release -cs`

read -d '' msg_bienvenida << EOF
Bienvenido al programa de instalación de JackTheStripper para $system_info
($system_codename).

Por favor, elija una opción para continuar:
EOF

dialog \
    --backtitle "JackTheStripper" \
    --title "Opciones" \
    --menu "$msg_bienvenida" \
    15 70 3 \
    Instalar "Instalar y configurar todo el sistema" \
    Descargar "Solo descargar los archivos" \
    Salir "Salir"
```

El código anterior producirá un diálogo como el siguiente:



El número **3** corresponde a la cantidad de opciones del menú que serán visibles (resaltado en rojo en el código). Las siguientes líneas corresponden a cada uno de los ítem del menú, siendo la *string* de la izquierda, el valor que será almacenado por `dialog`. Puede capturarse dicho valor almacenándolo en un archivo temporal:

```
dialog \
    ...
    Instalar "Instalar y configurar todo el sistema" \
    Descargar "Solo descargar los archivos" \
    Salir "Salir" 2> /tmp/opcion-elegida

cat /tmp/opcion-elegida
```

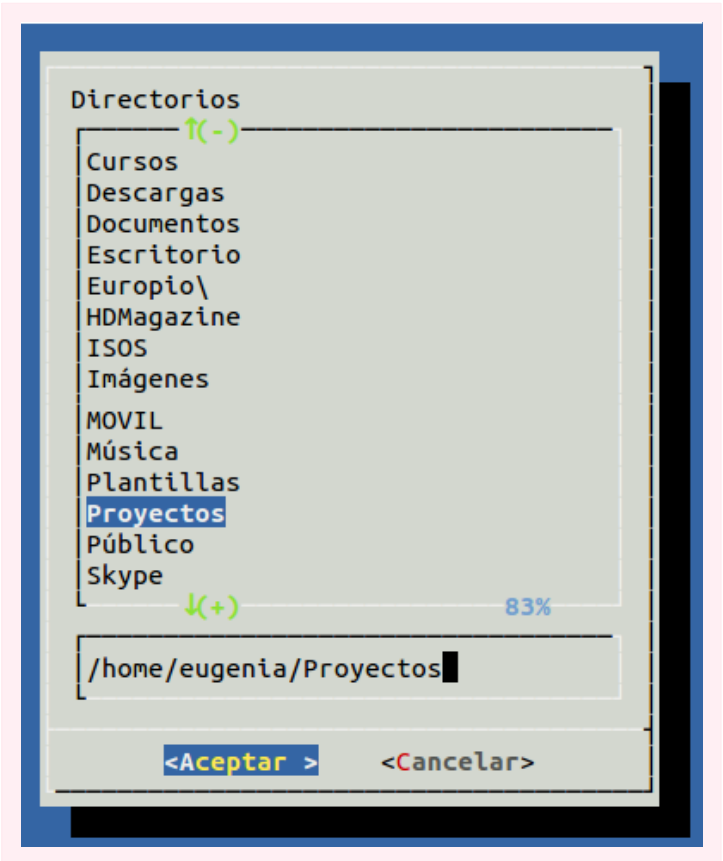
O también se podría capturar en una variable, agregando la opción común `--stdout`, aunque no es la forma más recomendada por su autor:

```
salida=$(dialog --stdout \
    --backtitle "JackTheStripper" \
    --title "Opciones" \
    --menu "$msg_bienvenida" \
    15 70 3 \
    Instalar "Instalar y configurar todo el sistema" \
    Descargar "Solo descargar los archivos" \
    Salir "Salir")

echo $salida
```

Selector de archivos y directorios: Eligiendo un directorio de instalación

```
#!/bin/bash
directorio=$(dialog \
    --stdout \
    --backtitle "Europio Engine Installer" \
    --dselect $HOME/ \  # utilizar --fselect en vez de –dselect para seleccionar archivos
    15 40)

cd $directorio
```

El código anterior producirá un diálogo como el siguiente:



Lo mismo pero desde Python

Lógicamente que los diálogos vistos anteriormente, son solo una mera introducción a la larga lista de artilugios provistos por `dialog`. **Vivek Gite** escribió una excelente **guía sobre *Shell Scripting*[2]** en la cual dedica un **capítulo sobre `dialog`[3]** que te puede resultar muy interesante. Pero teniendo ya una visión general sobre `dialog` y sus usos, hablar de la librería **`pythondialog`** será mucho más sencillo.

PythonDialog (pythondialog.sourceforge.net) puede instalarse mediante un simple «pip», disponiendo de las versiones para Python 2 y 3 de forma separada:

```
eugenia@cococha-gnucita:~$ pip search pythondialog
pythondialog              - A Python interface to the UNIX dialog utility ...
python2-pythondialog      - A Python interface to the UNIX dialog ... (Python 2 backport)
eugenia@cococha-gnucita:~$
eugenia@cococha-gnucita:~$ sudo pip install python2-pythondialog
```

Emplear esta librería es tan sencillo como:

1. importar la clase `Dialog` del módulo;
2. crear una instancia de la clase y comenzar a crear cuadros de diálogo.

Los cuadros de diálogo se crean mediante métodos del objeto `Dialog`, cuyos nombres equivalen a los diversos tipos de diálogos provistos por `dialog`. Los argumentos de opción que antes se pasaban con `--opcion` ahora se pasan como claves (*keyword args*). De esta forma, para crear un cuadro de diálogo normal, bastaría con las siguientes instrucciones:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from dialog import Dialog


dialog = Dialog()
dialog.msgbox(title=u"Título", text=u"Mensaje del cuadro de diálogo")
respuesta = dialog.yesno(title=u"Título", text=u"Está seguro de continuar?")

if respuesta == 'ok':
    print "Está seguro"
elif respuesta == 'cancel':
    print "No. No estaba seguro"
```

Claro que no todo es color de rosa y algunas dificultades se pueden presentar. La más notable de ellas es para los hispanoparlantes, el problema eterno con los caracteres no ASCII en Python. No basta con definir la codificación de caracteres como UTF-8 al inicio. Pero se pueden evitar todos los conflictos empleando `cadena.decode('utf-8')` como se muestra a continuación:

2 http://bash.cyberciti.biz/guide/Main_Page

3 http://bash.cyberciti.biz/guide/Bash_display_dialog_boxes

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from dialog import Dialog


dialog = Dialog()
dialog.msgbox(title=u"Título", text=u"Mensaje del cuadro de diálogo")
respuesta = dialog.yesno(title=u"Título",
    text="¿Está seguro de continuar?".decode('utf-8'))

if respuesta == 'ok':
    print "Está seguro"
else:
    print "No. No estaba seguro"
```

Luego, las opciones comunes pueden «setearse» mediante métodos, aunque en versiones recientes no son del todo intuitivos. Por ejemplo:

```
dialog = Dialog()
dialog.set_background_title("Título del programa")
```

Es la versión moderna de:

```
dialog = Dialog()
dialog.add_persistent_args(["--backtitle", "Título del programa"])
```

Ninguno de los dos es demasiado intuitivo pero tal vez, en la forma correspondiente a las versiones anteriores, con solo recordar un método, bastaba para definir la lista con el argumento original y su valor.

El sitio Web oficial es http://pythondialog.sourceforge.net/ pero no posee más que unos pocos ejemplos. Sin embargo, siempre podemos hacer un:

```
eugenia@cococha-gnucita:~$ python
...
>>> from dialog import Dialog
>>> help(Dialog)
```

Lo anterior, nos permitirá acceder a una documentación muy completa sobre esta librería. ¿Mi recomendación? Prueba primero «jugar» con `dialog` en GNU/Bash. Entiéndelo, disfrútalo y encuentra una utilidad. Recién allí, comienza a experimentar desde Python pero ya, con una idea más concreta y una necesidad de uso real. De esa forma, lograrás mejores resultados.