Photo credits: unsplash (Added by John Schnobrich)

How to modernize your Bash Scripts by adding GUI

Tired of displaying raw text on Console? Try Zenity & notify-send

Shalitha Suranga Follow
Sep 26 · 5 min read ★

Bash scripts contain a set of instructions written in Bash command language and those scripts can be executed in Unix shell interpreters. We use bash scripts to automate several tasks that are apparently time consuming if we follow the manual way of doing it. But if we compare with modern computing bash scripts are old fashioned stuff since

all the interactions with the user are done through the command line interface. We know that several developers are using eye catching signs and colors in order to highlight important things in the console interface. If a particular old fashioned bash script is used by a highly technical audience that is fine. But if it is used by a general audience it is obviously good to have some user friendly interactions.

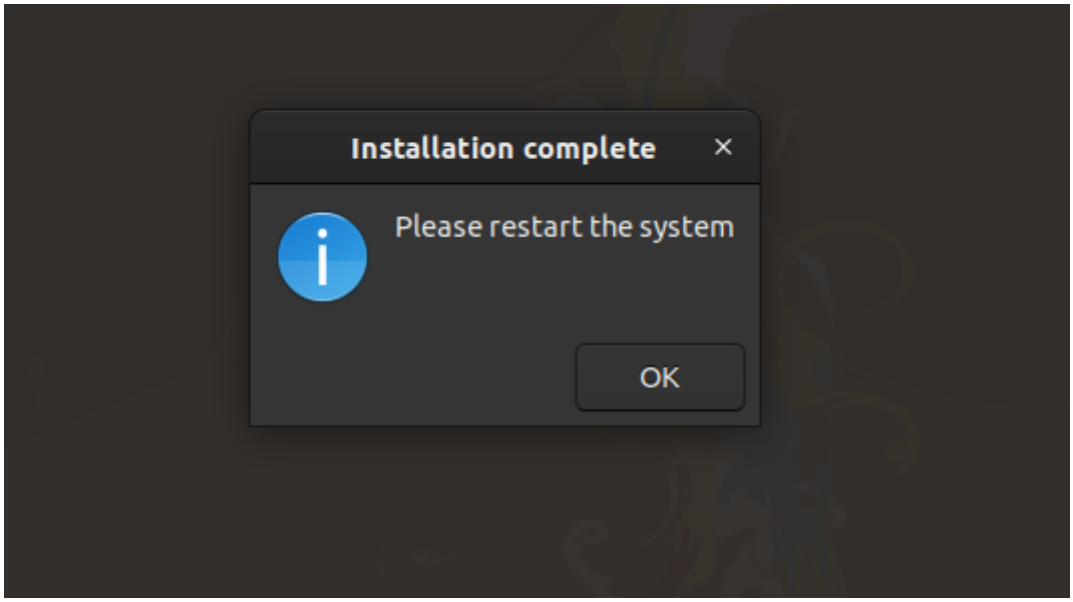
In fact, you are able to include GUI (Graphical User Interface) based input/output components into your next bash script using the *Zenity* command line tool which helps us to display GTK dialog boxes. Furthermore, Native GUI notifications can be displayed using the notify-send command line tool. These two tools are usually coming with popular Linux distributions, therefore there is no need for any kind of pre-installations.

Message boxes

It's obviously nice to display completion of a task with a native message box to the user rather than printing raw text in the console. Error, Info, Question and Warning type message boxes can be generated easily with *Zenity*.

Information message box `zenity --info`

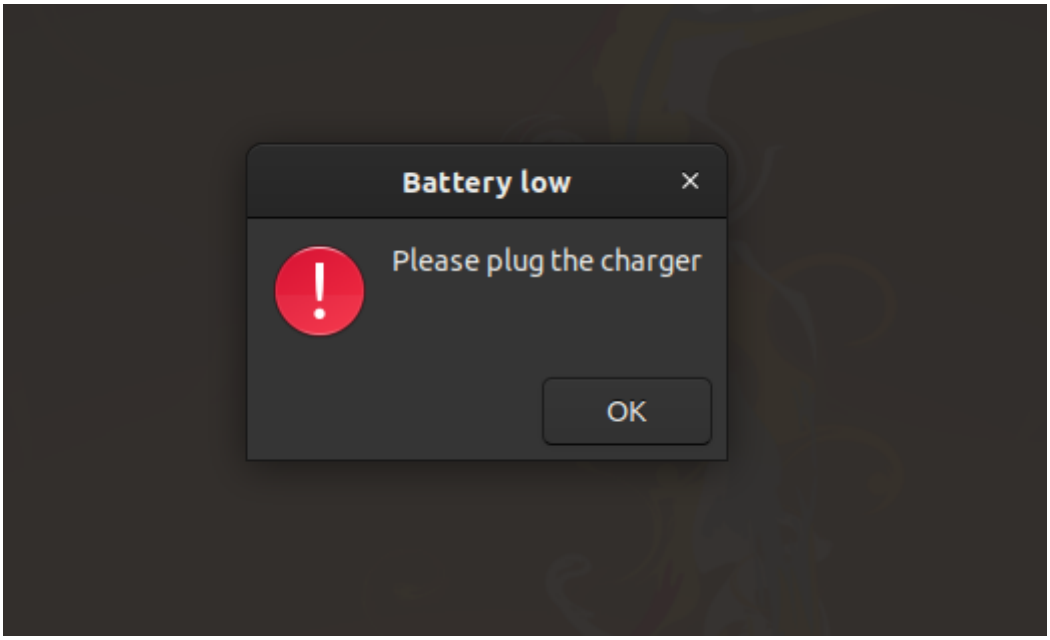
```
$ zenity --info --title="Installation complete" --text="Please
restart the system" --no-wrap
```



An example for "Information type message box"

Warning message box `zenity --info`

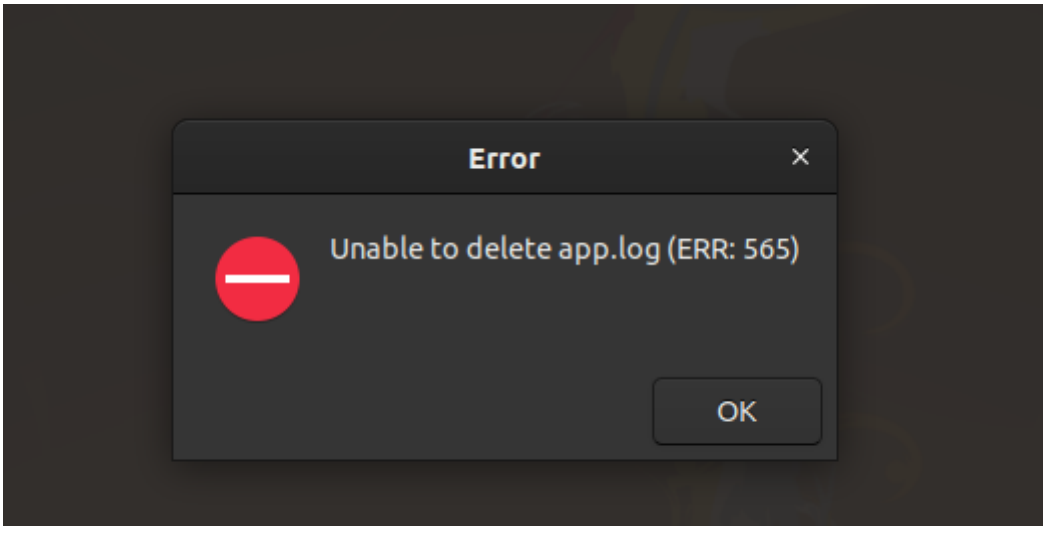
```
$ zenity --warning --title="Battery low" --text="Please plug your computer" --no-wrap
```



An example for ”Warning type message box”

Error message box `zenity --error`

```
$ zenity --error --title="Error" --text="Unable to delete app.log (ERR: 565)" --no-wrap
```

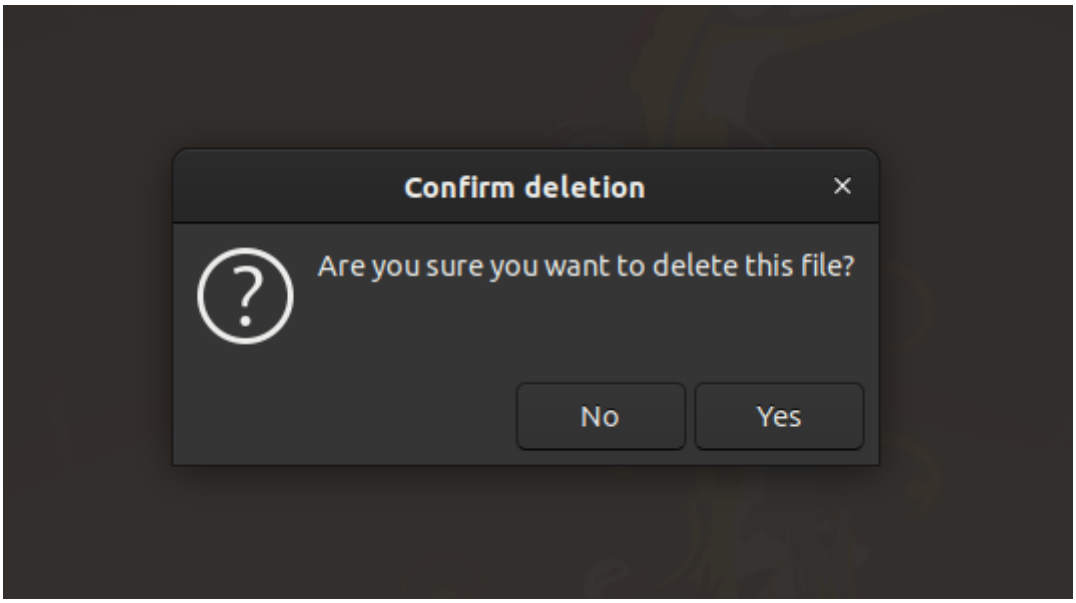


An example for "Error type message box"

Error message box `zenity --question`

Question type messages can be used if a set of instructions need to be executed if the user allows. For example, deleting a file from disk. This could be achieved using a simple *if condition* or `$?` special variable which stores the last return value.

```
if zenity --question --title="Confirm deletion" --text="Are you sure
you want to delete this file?" --no-wrap
    then
        zenity --info --title="Success" --text="app.log was removed"
--no-wrap
fi
```



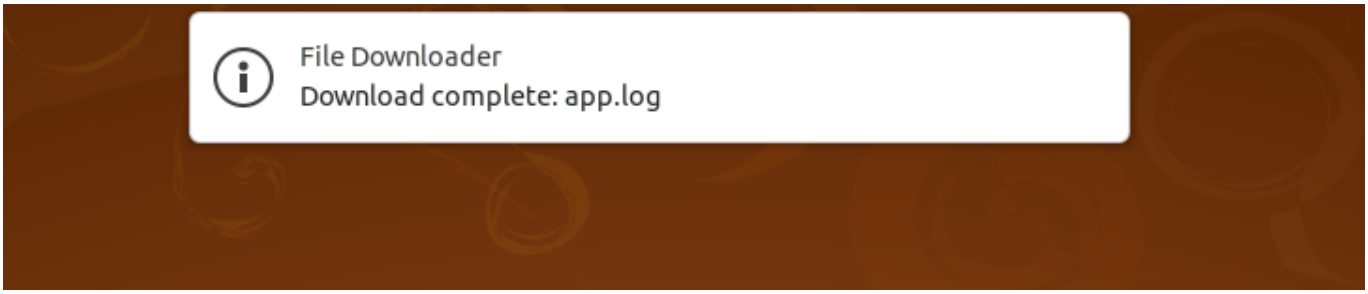
An example for "Question type message box"

Notifications

Notifications are great for displaying status of long running batch instructions. It is so important that users will get the notification even if they are doing some other work than looking into the console to see what is going on. Native notifications can be easily generated with the help of `notify-send` command line tool

Consider the following simple example…

```
$ notify-send "File Downloader" "Download complete: app.log"
```



An example for notify-send's usage

Zenity also has a feature to send notifications but *notify-send* gives us a lot more freedom for adjustments.

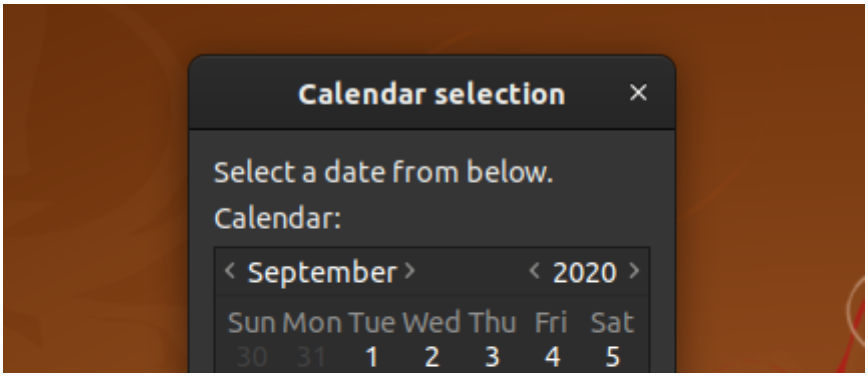
Input elements

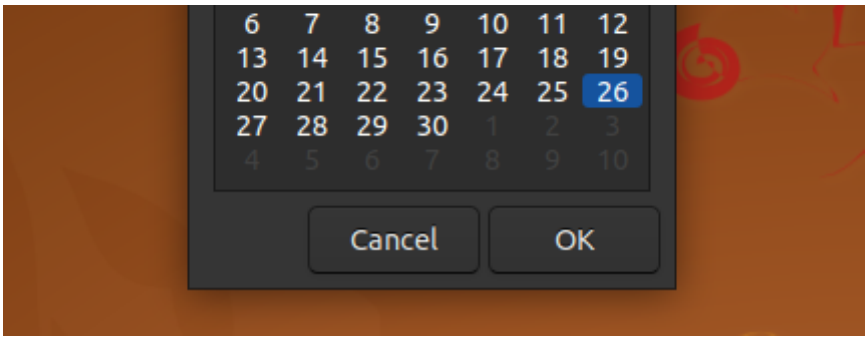
Zenity offers nice support for collecting user input by offering various sorts of input elements. It has the following types of input boxes.

Calendar input box `zenity --calendar`

This is a nicer way to capture date input from the user rather than asking the user to enter required date in *yyyy-mm-dd* format from the console.

```
I=$(zenity --calendar)
zenity --info --text="Selected date: $I" --no-wrap
```



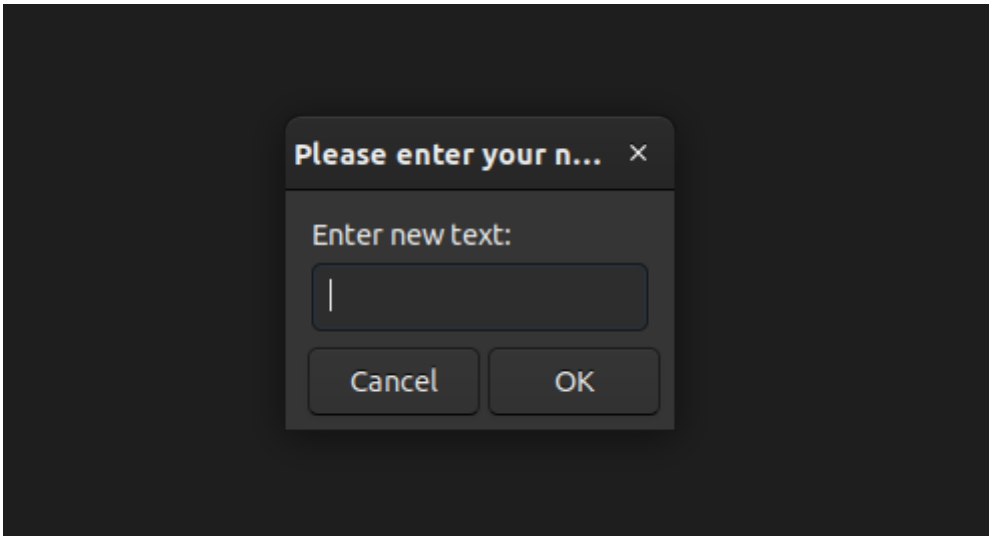


An example for calendar input box

String input box `zenity --calendar`

We usually use the *read* command to get some string input from the console. Usability of the bash script for non-tech people could be enhanced by offering a GUI text field which also accepts generic key strokes (Home/End keys etc.) and simple copy-paste like features.

```
NAME=$(zenity --entry --title="Please enter your name")
if [ -n "$NAME" ]
then
    zenity --info --text="Hello $NAME" --no-wrap
fi
```



An example for string input box

Very similarly `zenity --password` can be used to capture a secret string from the user such as a password or a PIN number. Moreover, password input allows you to enable the

username field as well. Then it will be returning username and the password separated with | character.

File selection dialog `zenity --file-selection`

Native save/open dialog could be displayed smoothly. I used this feature in Neutralinojs too.

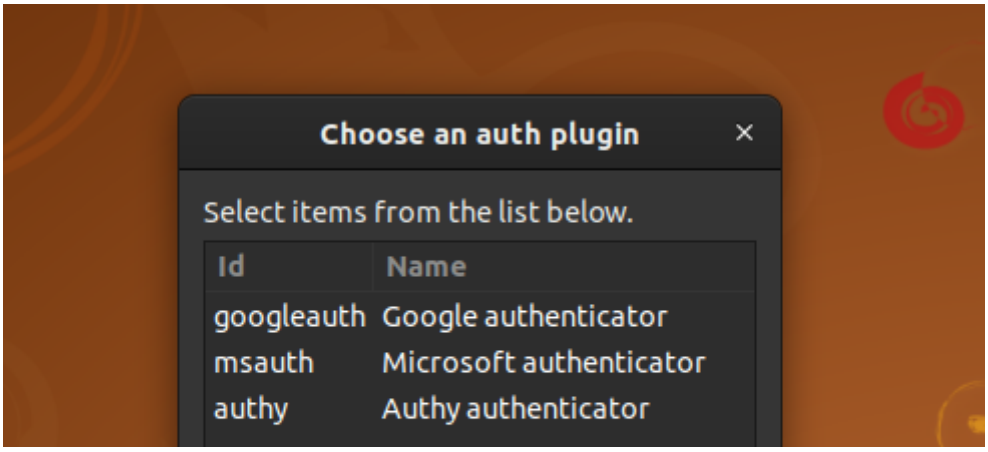
```
$ zenity --file-selection --title="Select a file"
```

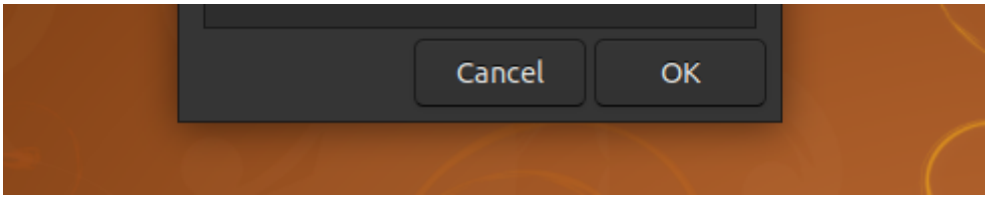
List selections

If we use normal console input for asking some choices from the user we may implement several shorten key inputs for desired choices of the user. For instance, asking users to enter letter *A* for accepting an option and on the other hand asking users to enter letter *B* for activating another option. This method could be improved a bit from a user's perspective by displaying GUI based list selection.

Let's assume that you are making an installation script and you need to ask which helper plugin needs to be installed for two factor authentication with the main software program.

```
$ zenity --list --title="Choose an auth plugin" --column="Id" --
column="Name" googleauth "Google authenticator" \
msauth "Microsoft authenticator" \
authy "Authy authenticator"
```





An example for list selection window

. . .

Advanced example

As mentioned above there are several GUI input elements that can be used with bash scripts instead of using raw text all the time. In addition, I will show you a bit more advanced example which was implemented using these native GUI elements.

```
#!/bin/bash

FILE=$(zenity --file-selection --title="Select a html file" --file-filter="*.html")
if [ -z  "$FILE" ]
then
    exit 1
fi

if zenity --question --title="HTML viewer" --text="Would you like to open this file with
then
    xdg-open "file://${FILE}"
else
    zenity --text-info --filename="$FILE" --html --title="HTML viewer"
fi
```

ZenityExample.sh hosted with ❤️ by GitHub view raw

Happy *bash*ing 😎

Programming Technology Software Engineering Bash Linux