
SOMMAIRE

Introduction.....	3
Objectif.....	3
Manipulations.....	3
Les interfaces graphiques.....	4
Présentation.....	4
La commande dialog.....	5
La commande Xdialog.....	6
Identifier le terminal.....	7
Récupérer des données.....	8
Checklist et Textbox.....	11
Logbox.....	12
Un script graphique.....	12
Un kill graphique.....	13
Annexe 1.....	14

© Copyright 2010 tv <thierry.vaira@orange.fr>

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License,

Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover.

You can obtain a copy of the GNU General Public License : write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

INTRODUCTION

Objectif

Être capable d'écrire des *shell* scripts possédant une interface graphique.

Manipulations

On va travailler dans l'arborescence suivante :

```
$HOME
  |
  - tv
    |
    - TPOS11
```

Créer l'arborescence de répertoires :

```
$ mkdir -p $HOME/tv/TPOS11
```

Se déplacer dans l'arborescence de travail :

```
$ cd $HOME/tv/TPOS11
```

LES INTERFACES GRAPHIQUES

Présentation

Il est possible d'ajouter une IHM graphique (à base de boîtes de dialogue) pour les shell scripts en utilisant un "*dialog program*" comme :

- le package `gtkdialog` fournit `gchooser`, `filechooser`, `gmessage` et `xtest`

```
$ gmessage -buttons
"Continuer:1,Arreter:0"
-center -print "Test"
```



- la commande `gdialog` (souvent remplacé par `zenity`)
- la commande `zenity`

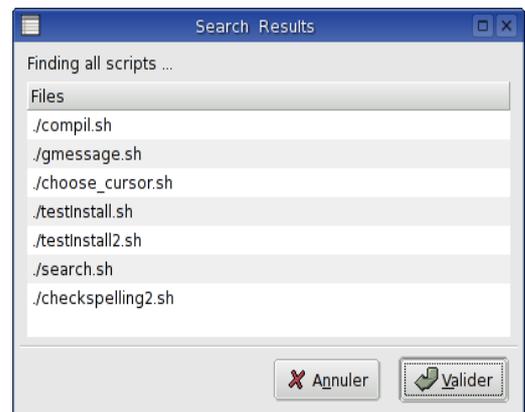
```
$ zenity --question --title "Alert"
--text "Windows a été détecté. Voulez-vous le supprimer ?"
```



```
$ find . -name '*.sh' | zenity --title "Search Results" --list
--text="Finding all scripts ..." --column="Files"
```

- la commande `kdialo`

```
$ kdialo --combobox "Choisir l'OS" Linux
Windows
```



- les commandes `dialog` (mode console) et `Xdialog` (mode Xwindow).

On utilisera `dialog` et `Xdialog`.

La commande dialog

L'historique de cette commande se révèle assez compliqué puisque son auteur ayant cessé de la supporter, plusieurs versions ont vu le jour.

Plus de détails sur : www.hightek.org/dialog/

Avec la distribution Mandriva, vous trouverez le paquetage **cdialog** qui correspond à une version de dialog. Pour vérifier sa présence, il suffit de taper dialog pour voir apparaître une aide succincte. Sinon, installer le paquetage :

```
# urpmi cdialog
```

Pour obtenir l'aide complète :

```
$ man dialog
```

Des exemples et la documentation sont disponibles dans :

```
$ ls /usr/share/doc/cdialog-*
```

Quelques boîtes de dialogues disponibles :

```
--calendar      <text> <height> <width> <day> <month> <year>
--checklist     <text> <height> <width> <list height> <tag1> <item1>
<status1>...
--fselect       <filepath> <directory> <height> <width>
--gauge         <text> <height> <width> [<percent>]
--infobox       <text> <height> <width>
--inputbox      <text> <height> <width> [<init>]
--menu          <text> <height> <width> <menu height> <tag1> <item1>...
--msgbox        <text> <height> <width>
--passwordbox   <text> <height> <width> [<init>]
--radiolist     <text> <height> <width> <list height> <tag1> <item1>
<status1>...
--tailbox       <file> <height> <width>
--tailboxbg     <file> <height> <width>
--textbox       <file> <height> <width>
--timebox       <text> <height> <width> <hour> <minute> <second>
--yesno         <text> <height> <width>
```

Exemple :

```
$ dialog --title "Titre" --msgbox "Hello world" 0 0
```



La commande Xdialog

Plus de détails sur : xdialog.dyns.net. Pour vérifier sa présence, il suffit de taper `Xdialog` pour voir apparaître une aide succincte. Sinon, installer le paquetage :

```
# urpmi Xdialog
```

Pour obtenir l'aide complète :

```
$ man Xdialog
```

Des exemples et la documentation sont disponibles dans :

```
$ ls/usr/share/doc/Xdialog-*
```

Quelques boîtes de dialogues disponibles :

```
--yesno <text> <height> <width>
--msgbox <text> <height> <width>
--infobox <text> <height> <width> [<timeout>]
--gauge <text> <height> <width> [<percent>]
--progress <text> <height> <width> [<maxdots> [[-]<msglen>]]
--inputbox <text> <height> <width> [<init>]
--2inputsbox <text> <height> <width> <label1> <init1> <label2> <init2>
--3inputsbox <text> <height> <width> <label1> <init1> <label2> <init2>
<label3> <init3>
--combobox <text> <height> <width> <item1> ... <itemN>
--rangebox <text> <height> <width> <min value> <max value> [<default
value>]
--2rangesbox <text> <height> <width> <label1> <min1> <max1> <def1>
<label2> <min2> <max2> <def2>
--3rangesbox <text> <height> <width> <label1> <min1> <max1> <def1> ..
<label3> <min3> <max3> <def3>
--spinbox <text> <height> <width> <min value> <max value> <default
value> <label>
--2spinsbox <text> <height> <width> <min1> <max1> <def1> <label1>
<min2> <max2> <def2> <label2>
--3spinsbox <text> <height> <width> <min1> <max1> <def1> <label1> ...
<min3> <max3> <def3> <label3>
--textbox <file> <height> <width>
--editbox <file> <height> <width>
--tailbox <file> <height> <width>
--logbox <file> <height> <width>
--menubox <text> <height> <width> <menu height> <tag1> <item1>
{<help1>}...
--checklist <text> <height> <width> <list height> <tag1> <item1>
<status1> {<help1>}...
--radiolist <text> <height> <width> <list height> <tag1> <item1>
<status1> {<help1>}...
```

```
--buildlist <text> <height> <width> <list height> <tag1> <item1>
<status1> {<help1>}...
--treeview <text> <height> <width> <list height> <tag1> <item1>
<status1> <item_depth1> {<help1>} .
--fselect <file> <height> <width>
--dselect <directory> <height> <width>
--calendar <text> <height> <width> <day> <month> <year>
--timebox <text> <height> <width>
```

Exemple :

```
$ Xdialog --title "Titre" --msgbox "Hello world" 0 0
```



Identifier le terminal

Technique n°1 :

On peut tester la variable d'environnement \$DISPLAY :

```
#!/bin/bash
if [ -n "$DISPLAY" ]
then
  # en mode graphique, on utilise Xdialog !!!
  DIALOG="Xdialog"
else
  # en mode console, on dialog !!!
  DIALOG="dialog"
fi
$ DIALOG --title "Titre" --msgbox "Hello world" 0 0
```

et vérifier que le programme de dialogue est présent :

```
#!/bin/bash
# Quelle IHM utilisée ? (version complète)

if [ -n "$DISPLAY" ]
then
  # en mode graphique, on utilise Xdialog !!!
  [ -z "$DIALOG" -a -x "`which Xdialog 2>&-`" ] && DIALOG="Xdialog"
else
  # en mode console, on dialog !!!
  [ -z "$DIALOG" -a -x "`which dialog 2>&-`" ] && DIALOG="dialog"
fi
```

```
# on n'en a pas trouvé ! aie !
if [ -z "$DIALOG" ]
then
    ERROR="Ce script a besoin d'un \"dialog\" program comme Xdialog ou
cdialog ..."
    if [ -x "`which gmessage 2>&-`" -a -n "$DISPLAY" ]
    then
        gmessage "$ERROR"
    else
        echo "$ERROR"
    fi
    exit 2
fi
```

Technique n°2 :

Il peut s'avérer utile d'identifier le terminal sur lequel s'exécute le script. Pour cela, il faut traiter la variable \$TERM qui contiendra le plus souvent :

- "linux" pour un mode console
- "xterm" pour un mode Xwindow

```
#!/bin/bash
# Quelle IHM utilisée ?

case $TERM in
xterm) DIALOG="Xdialog --stdout --title Test";;
linux*) DIALOG="dialog --stdout --title Test";;
*)      echo "ERREUR: terminal inconnu"; exit 1;;
esac

$DIALOG --msgbox "Hello world !" 0 0
exit 0
```

Récupérer des données

Pour envisager une interaction avec le script, il faut distinguer les deux situations suivantes :

- validation (OK) ou annulation (CANCEL) d'une boîte de dialogue : il faut alors traiter le code retour \$? (OK: \$? égal à 0 ou CANCEL: \$? égal à 1) soit avec un if (ou un test) soit en créant un groupement de commandes avec || ou &&
- récupérer les données manipulées par la boîte de dialogue : il faut alors utiliser la substitution de commande (`commande` ou \$(commande)) pour récupérer le résultat dans une variable ou rediriger le flux de sortie vers un fichier et traiter ce fichier par la suite

Exemple 1 : groupement || et substitution \$(commande)

```
#!/bin/bash
# Réaliser une saisie (INPUTBOX) puis un affichage (MSGBOX)
# 1. Saisie d'un nom : un nom par défaut est fourni
# on récupère le nom saisi dans la variable nom (ne pas oublier --stdout
# )
# si Annuler alors on sort ...
nom=$(Xdialog --backtitle "TP1" --stdout --title "Test INPUTBOX"
--inputbox "Entrez votre nom :" 0 0 "VAIRA Thierry") || exit 1

# 2. On affiche le nom saisi
Xdialog --backtitle "TP1" --title "Test MSGBOX" --msgbox "Hello $nom" 0
0
exit 0
```

Exemple 2 : (même chose mais différemment) code retour \$? et redirection dans un fichier

```
#!/bin/bash
# Réaliser une saisie (INPUTBOX) puis un affichage (MSGBOX)
# on a besoin d'un fichier temporaire
fichier_temp=$(mktemp /tmp/${basename $0}.XXXXXX)
# on a besoin de le supprimer automatiquement quand le script se termine
trap "rm -f $fichier_temp" EXIT

# 1. Saisie d'un nom : un nom par défaut est fourni
# on redirige le nom saisi dans un fichier temporaire
Xdialog --backtitle "TP1" --title "Test INPUTBOX" --stdout --inputbox
"Entrez votre nom :" 0 0 "VAIRA Thierry" > $fichier_temp

# 2. Test $? ?
case $? in
0)
echo "Ok ! ";; # on continue ...
1)
echo "Cancel ! " # si Annuler alors on sort ...
exit 1;;
255)
echo "Closed ! " # si Fermer alors on sort ...
exit 255 ;;
esac

# 3. On récupère la réponse donc le nom saisi
nom=$(cat $fichier_temp)

# 4. On affiche le nom saisi
Xdialog --backtitle "TP1" --title "Test MSGBOX" --msgbox "Hello $nom" 0
0

exit 0
```

Exemple 3 : groupement de plusieurs boîtes de dialogue avec &&

```
#!/bin/bash
dlg_cmd="Xdialog --stdout --title Test"

# 1. Récupération des données
# Si OK sur la première boîte de dialogue
# Alors on affiche la deuxième boîte puis les données récupérées
# Sinon on sort
date=$(($dlg_cmd --calendar "Entrez une date" 0 0 0 0 0) &&
heure=$(($dlg_cmd --timebox "Entrez une heure" 0 0) || exit 1

# 2. Affichage des données récupérées
echo "Date: $date - Heure: $heure"
exit 0
```

Exemple 4 : la commande Xdialog permet aussi d'enchaîner plusieurs boîtes de dialogue

```
#!/bin/bash
NAME=`Xdialog --stdout --title "Boîtes chaînées" \
    --msgbox "Un exemple de boîtes chaînées" 0 0 \
    --buttons-style text --yesno "On continue ?" 0 0 \
    --buttons-style icon --inputbox "Donner un nom ?" 0 0 "" \
    --msgbox "Terminé !" 0 0`

case $? in
  0)
    echo "Ok ! ";;
  1)
    echo "Cancel ! "
    exit 1;;
  255)
    echo "Closed ! "
    exit 255 ;;
esac

if [ "$NAME" != "" ] ; then
  echo "Votre nom est \"$NAME\"."
fi
```

Remarque : Xdialog permet aussi de faire des boîtes de dialogue de type wizard avec l'option --wizard (boîte de dialogue avec les boutons "Précédent" et "Suivant") ...

CHECKLIST ET TEXTBOX

1) Transformer le script suivant en version Xdialog.

```
#!/bin/bash
# version cdialog
# test d'un choix utilisateur avec
# CHECKLIST et affichage d'un texte avec TEXTBOX

# on a besoin d'un fichier temporaire
TMP=$(mktemp /tmp/reponse.XXXXXX)

# on efface le fichier temporaire lorsque le script se termine
trap "rm -f $TMP" 0 2 5 15

# affiche la boite de dialogue de choix
dialog --backtitle "Séquence 1" --stdout --title "Test CHECKLIST"
--separate-output --checklist "Selectionner les jours :" 25 30 7 LUN
Lundi 0 MAR Mardi 0 MER Mercredi 0 JEU Jeudi 0 VEN Vendredi 0 SAM Samedi
0 DIM Dimanche 0 > $TMP

# Test si OK ou CANCEL ?
case $? in
1)
    echo "Cancel ! "
    exit 1;;
255)
    echo "Closed ! "
    exit 255 ;;
esac

# Affichage avec TEXTBOX du choix utilisateur
dialog --backtitle "Séquence 1" --title "Test TEXTBOX" --textbox $TMP 0
0
```

LOGBOX

2) Tester, puis commenter (#) chaque ligne de ce script.

```
#!/bin/bash

TITLE="$(basename $0)"

repertoire=${1:-$(pwd)}

DIALOG="Xdialog --title $TITLE -stdout"

choixRepertoire=$(($DIALOG --dselect $repertoire 0 0) || exit 1

# on met des "" à $choixRepertoire pour se protéger des espaces
# dans le nom du répertoire
du -sch "$choixRepertoire" | tr '\t' ' ' | $DIALOG --logbox - 0 0
```

UN SCRIPT GRAPHIQUE

Soit le script `fr2en.sh` fourni en annexe 1 :

```
$ ./fr2en.sh Il pleut beaucoup
It rains a lot
$ ./fr2en.sh
Il pleut beaucoup
Ctrl-d
It rains a lot
$ echo "Il pleut beaucoup" | ./fr2en.sh
It rains a lot
```

3) Intégrer un interface graphique à ce script. Pour terminer, en utilisant exactement le même fichier source, fournir en une seule commande la version, que vous nommerez `en2fr.sh`, qui traduit par défaut de l'anglais vers le français.

UN KILL GRAPHIQUE

On veut réaliser une commande kill graphique en utilisant une interface graphique.

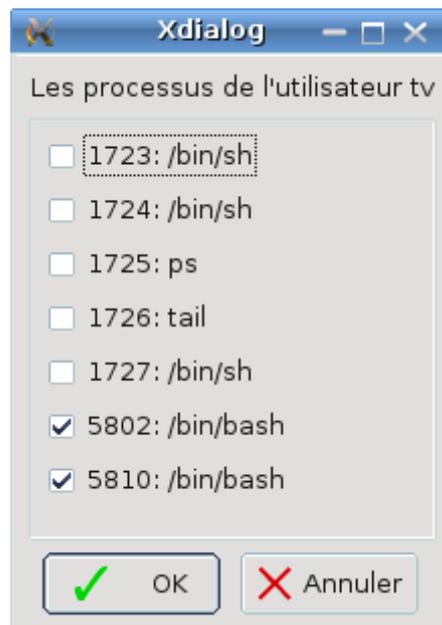
Lorsque le script sera appelé sans paramètre, il affichera tous les processus de l'utilisateur et proposera de cocher ceux qu'il désire « tuer ». Lorsqu'il est appelé avec un paramètre, il "pré-cochera" en plus les processus qui contiennent ce paramètre comme sous chaîne.

Par défaut, tous les processus seront tués avec le signal 15 (ou 9). Pour les tests, utiliser un echo "kill ..." pour éviter de tuer tout ce qui bouge !

Modifier le script pour qu'il accepte un autre numéro de signal.

Un exemple est donnée ci-dessous :

```
$ ./tuer.sh -2 bash
kill -2 5802
kill -2 5810
```



4) Fournir le script tuer.sh.

ANNEXE 1

```
#!/bin/sh
#
# Traduction avec Google
#
#           Author: Erick Gallesio [eg@essi.fr]
#   Creation date: 11-Nov-2007 21:03 (eg)
# Last file update: 12-Oct-2008 23:40 (eg)
#
# modification du filtrage suite au changement
# de l'outil de traduction Google (voir remarques)
#
#           Author: Thierry Vaira [thierry.vaira@orange.fr]
#   Creation date: 02-Jan-2009
# Last file update: 02-Jan-2009

# Version finale (avec awk)

# Remarques :
# avant, la traduction se trouvait entre ces deux balises :
# balise1='<div id=result_box dir="ltr">'
# balise2='</div>'

# le script initial utilisait donc les commandes suivantes
# pour extraire la traduction :
# wget ... | grep "$balise1" | sed -e "s/.*$balise1//" | sed -e
"s=$balise2.*=="

# maintenant, la traduction est formatée comme suit dans la page HTML
renvoyée :
# exemple : pour 'jaune ' -> 'yellow'
# <span title="jaune" onmouseover="this.style.backgroundColor='#ebff9'"
onmouseout="this.style.backgroundColor='#fff'">yellow</span>

# -----
scriptname=$(basename $0)

case $scriptname in
    fr2en.sh) from="fr"; to="en";;
    *)       from="en"; to="fr";;
esac

phrase=''
```

```

while [ $# -gt 0 ]
do
  case $1 in
    --from) from=$2; shift 2;;
    --to) to=$2; shift 2;;
    *) phrase="$phrase $1"; shift;;
  esac
done

if [ "$phrase" = "" ]
then
  # Pas de phrase sur la ligne de commande => lire depuis stdin
  phrase=$(cat)
fi

phrase=$(echo "$phrase" |sed -e "s/ /%20/g")

# -----

traduire()
{
  url="http://translate.google.com/translate_t?
text=$phrase&sl=$from&tl=$to&ie=8859-1"
  wget -q -U "" -O - $url |
    awk '{match($0, /<span[^>]*title=\"[^\"]+\"[>]*>([<]*)
</span>/, a); print a[1]}' |
    sed '/^$/d' # suppression de toutes les lignes vides
}

# -----

traduire $from $to "$phrase"

```