Dialog e Derivados

João Marcelo Uchôa de Alencar

Universidade Federal do Ceará - Quixadá

25 de Outubro de 2019

Introdução

- *Dialog* é um programa usado para desenhar interfaces amigáveis para o usuário.
- Ele é um comando como qualquer outro, podendo ser invocado diretamente no terminal ou inserido em *scripts*.
- Existem alguns derivados que criam telas em servidores X e ambientes como Gnome e KDE.
- *sudo apt install dialog*
- Elementos
  - Botões.
  - Janelas.
  - Menu.
  - Entradas para texto.

Meu Primeiro Exemplo em Dialog

```
$ dialog --msgbox "Programação de Scripts" 5 40
```

- Desenhamos uma caixa de mensagens (*msgbox*).
- O tamanho dela é 5 por 40.
- São linhas e colunas, nada de *pixels*.

As 15 Caixas Mágicas

Tipo	**Ação**
calendar	Calendário
checklist	Lista de opções
fselect	Escolha de arquivo
gauge	Barra de progresso
infobox	Mensagem
inputbox	Entrada de Texto
menu	Menu de opções
msgbox	Mensagem com botão
passwordbox	Digita uma senha
radiolist	Lista de opções
tailbox	Saída do comando *tail -f*
tailboxbg	Saída no *background*
textbox	Conteúdo de um arquivo
timebox	Escolhe um horário
yesno	Pergunta Sim/Não

Exemplos de Comandos

```
$ dialog --title "Escolha a data" --calendar "" 0 0 21 12 1999

$ dialog --title "Escolha onde instalar" --fselect /usr/share/vim 0 0

$ dialog --title "Instalação dos Pacotes" \
--gauge "\nInstalando vim-6.0a.tgz..." 8 40 60

$ dialog --title "Aguarde" --infobox "\nFinalizando em 5 segundos..." 0 0

$ dialog --title "Confirmação" --passwordbox "Confirme a senha:" 0 0

$ dialog --title "Perfil" --menu "Escolha o perfil de instalação:" \
0 0 0 mínima "Instala o mínimo" completa "Instala tudo" \
customizada "Você escolhe"
```

Exemplos de Comandos

```
$ dialog --title "Parabéns" --msgbox \
"Instalação finalizada com sucesso." 6 40

$ dialog --title "Pergunta" --radiolist "Há quanto tempo você usa o Vi?" \
0 0 0 iniciante "até um ano" on experiente \
"mais de um ano" off guru "mais de três anos" off

$ tail -f /var/log/syslog > out & dialog \
--title "Monitorando mensagens do Sistema" --tailbox out 0 0

$ dialog --title "Visualizando Arquivo" \
--textbox "/usr/share/vim/vim74/indent.vim" 0 0

$ dialog --title "Ajuste o Relógio" \
--timebox "\nDica: use as setas e o TAB." 0 0 23 59 30

$ dialog --title "AVISO" --yesno \
"\nO Vi foi instalado e configurado. Executá-lo?" 0 0
```

Agora, como recuperar a entrada?

- A linha de comando é sempre longa, com várias opções.
- O texto é redimensionado na caixa de maneira automática.
- O **código de retorno** permite saber se o usuário escolheu **Sim/Não, Ok/Cancelar**.
- A **saída de erro, STDERR,** permite recuperar os textos e itens escolhidos.

Parâmetros Obrigatórios da Linha de Comando

```
$ dialog --tipo-da-caixa '<texto>' <altura> <largura>
```

- texto: palavra ou frase que aparece no ínicio da caixa.
- altura: número de linhas da caixa.
- largura: número de colunas da caixa.

Como reconhecer respostas SIM ou NÃO

```
$ dialog --yesno "sim ou não?" 0 0
```

Como saber o que foi escolhido?

```
$ dialog --yesno "sim ou não?" 0 0; echo Retorno: $?;
```

Memorizando

SIM=0, NÃO=1

```
$ dialog --yesno "Quer ver as horas?" 0 0 \
&& echo "Agora são: $(date)";
```

Exemplo Sim/Não

```
#!/bin/sh
# lsj.sh -- o script do "ls joiado"
# Este script faz parte do http://aurelio.net/shell/dialog

# Zerando as opções
cor= ; ocultos= ; subdir= ; detalhes=

# Obtendo as configurações que o usuário deseja
dialog --yesno 'Usar cores?'                0 0 && cor='--color=yes'
dialog --yesno 'Mostrar arquivos ocultos?' 0 0 &&  ocultos='-a'
dialog --yesno 'Incluir sub-diretórios?'   0 0 &&   subdir='-R'
dialog --yesno 'Mostrar visão detalhada?'  0 0 && detalhes='-l'

# Mostrando os arquivos
ls $cor $ocultos $subdir $detalhes
```

Como obter o texto que o usuário digitou?

```
$ dialog --inputbox "Digite seu nome:" 0 0
```

Como guardar o que foi usado em uma variável?

- Redirecionar a saída de erros (stderr) para um arquivo e depois ler o conteúdo do mesmo.
- Redirecionar a saída de erros (stderr) para a saída padrão (stdout).
- Usar a opção $--$ *stdout* do dialog.

```
$ nome=$(dialog --stdout --inputbox \
"Digite seu nome:" 0 0)
$ echo $nome
```

Como obter o item de um menu?

```
$ dialog --menu "Opções:" 0 0 0 1 um 2 dois 3 três
```

Como guardar a opção em uma variável?

```
$ opcao=$(dialog --stdout --menu "Opções:" \
0 0 0 1 um 2 dois 3 três)
$ echo $opcao
$ opcao=$(dialog --radiolist "As cores: " 0 0 0 \
   amarelo "a cor do sol" OFF \
   verde "a cor da grama" ON \
   azul "a cor do céu" OFF)
$ echo $opcao
```

Como obter múltiplos itens de uma lista?

```
estilos=$( dialog --stdout \
        --checklist 'Você gosta de:' 0 0 0 \
        rock  '' ON  \
        samba '' OFF \
        metal '' ON  \
        jazz  '' OFF \
        pop   '' ON  \
        mpb   '' OFF )
echo "Você escolheu: $estilos"
```

Como obter múltiplos itens de uma lista?

```
estilos=$( dialog --stdout \
        --separate-output \
        --checklist 'Você gosta de:' 0 0 0 \
        rock  '' ON  \
        samba '' OFF \
        metal '' ON  \
        jazz  '' OFF \
        pop   '' ON  \
        mpb   '' OFF )

echo "$estilos" | while read LINHA
do
         echo "--- $LINHA"
done
```

Botão Cancelar/Esc/Help

```
case $? in
          0) echo O usuário apertou o botão OK (ou o Yes) ;;
          1) echo O usuário apertou o botão CANCELAR (ou o No) ;;
          2) echo O usuário apertou o botão HELP ;;
        255) echo O usuário apertou a tecla ESC ;;
          *) echo Retorno desconhecido;;
esac
```

Exemplo

Podemos fazer uma agenda de nomes/e-mails com *dialog*?

1. Exibir menu com opções: Adicionar, Remover, Exibir Usuário, Exibir Todos e Sair.
2. Adicionar
   - 2.1 Exibir diálogo para receber nome.
   - 2.2 Exibir diálogo para receber e-mail.
   - 2.3 Adicionar no arquivo.
3. Remover
   - 3.1 Exibir diálogo para receber e-mail.
   - 3.2 Remover todos os nomes com o mesmo e-mail.
4. Exibir Usuário
   - 4.1 Exibir diálogo para receber e-mail.
   - 4.2 Exibir o nome e o e-mail, caso contrário afirmar que usuário não existe.
5. Exibir Todos
   - 5.1 Exibir todos os e-mails cadastrados.
6. Sair

Após executar as opções 2, 3, 4 ou 5, deve retornar ao menu principal.

Atividades

Coloque o arquivo *compactador.sh* na pasta *atividades/atividade11*. O objetivo do *script* é definir telas para as seguintes ações:

1. Apresentar uma tela requisitando o **caminho de um diretório**.
2. Formar uma **lista com os nomes dos arquivos** (sem subdiretórios) do diretório citado. O usuário deve escolher um ou mais arquivos.
3. Exibir duas **opções de compactação**: gzip ou b2zip.
4. Questionar o **nome do arquivo compactado** a ser criado.
5. Criar o arquivo compactado com os arquivos selecionados do diretório e exibir uma tela de sucesso com o **nome final do arquivo**.