Will Watt – www.sxc.hu

PROGRAMAÇÃO

De volta ao shell – mas com janelas

Papo de botequim 2.0 Parte I

Na volta da série sobre programação shell, modernize seus scripts com o uso do Zenity.

por Julio Cezar Neves

Há alguns dias, eu estava andando no calçadão e eis que encontro o Chico, meu garçom preferido. Conversamos um pouco, quando ele me contou que estava trabalhando, vejam só, no "Pingüim Carioca". Como já sou mesmo chegado a um boteco e muito mais a um pingüim, no primeiro sábado que passaria em casa convidei um amigo para tomar um chope.

Lá chegando, logo após a primeira rodada de canecas, o amigo me fala na maior cara de pau:

– Pô, mermão (com sotaque carioca), você tem de se modernizar. Por que insistir em desenvolver programas com interfaces arcaicas orientadas a caractere? Por que você não parte para algo mais atual?

Fiquei indignado com aquilo e respondi:

Tabela 1: Opções de comando para diálogos

Opção	Efeito
`--calendar`	Mostra um calendário
`--entry`	Mostra uma caixa para inserir textos
`--error`	Mostra uma janela de erros
`--file-selection`	Mostra uma janela para selecionar arquivos
`--info`	Mostra uma janela com informações
`--list`	Mostra uma caixa com lista
`--notification`	Mostra uma janela de notificação
`--progress`	Mostra uma barra de progresso
`--question`	Mostra uma janela para fazer uma pergunta
`--text-info`	Mostra uma janela de texto
`--warning`	Mostra uma janela de advertência
`--scale`	Mostra um escala para opções

– Cara, é você que tem de se modernizar e largar essa porcaria que você chama de sistema operacional e que te deixa na mão a toda hora, sem falar nos custos, vírus e nas telas azuis que você ganha pelo menos uma vez ao dia.

A partir de hoje e nas próximas vezes que viermos aqui, vou lhe mostrar as interfaces gráfica que foram desenvolvidas para serem usadas diretamente do Shell, e você verá que são muito mais concisas e fáceis de usar do que as que você conhece.

É necessário, no entanto, ter um bom conhecimento de Shell, o que você consegue nos *Papos de Botequim* que foram publicados na Linux Magazine desde o fascículo número 1 até o número 11[1].

– Chico, traz mais dois chopes! Você não esqueceu que o meu é sem colarinho, né?

Agora, e nos nossos próximos encontros, vou lhe dar um curso sobre os programas que fazem interface gráfica para o Shell, começando pelo *Zenity*, que é completo e extenso o suficiente para que esse nosso papo se estique por mais uns três encontros.

Tabela 2: Parâmetros genéricos

Opção	Efeito
`--title`=TITULO	Define um título para as caixas
`--window-icon`=ICONE	Define um ícone para as caixas
`--width`=LARGURA	Define a largura das caixas
`--height`=ALTURA	Define a altura das caixas

Tabela 3: Parâmetros de calendário

Opção	Efeito
`--day`=INT	Define o dia padrão
`--month`=INT	Define o mês inicial
`--year`=INT	Define o ano inicial
`--date-format`=PATTERN	Define formato da data (mesmo formato do `date`)

Como tudo começou

Um dia, o Tujal me pediu para dar um acabamento legal em uns scripts que ele desenvolveu para o *CVS* (Concurrent Version System – ou seja, um gerenciador de código-fonte) e fui procurar na Internet como fazê-lo usando o Zenity, do qual já tinha ouvido falar muito bem.

Considero esse software muito importante, pois, dominando Shell e Zenity, você poderá desenvolver seus scripts em Shell – que, como você já viu ou verá nos 11 primeiros números da Linux Magazine, é uma linguagem simples e concisa – e dar-lhes um excelente acabamento gráfico com Zenity.

O Zenity é a cara do Shell: facílimo de usar e super conciso. Essas duas ferramentas se complementam de forma a facilitar sua vida em programas curtíssimos e poderosos.

Afinal de contas, quem é esse cara?

Zenity é um programa que se utiliza de ferramentas da GTK+ para produzir interfaces gráficas muito bem acabadas que atuarão entre scripts em Shell (e outras linguagem orientadas a caractere) e os usuários, provendo entre ambos uma correlação amigável e bonita.

O Zenity é um executável que recebe todos os parâmetros via linha de comando e retorna – no código de retorno (`$?`) ou na saída primária (*stdout*) – a escolha do usuário. Isso permite apresentar, pedir e trocar informações com o operador.

Por exemplo, o comando `zenity --question` apresentará uma janela com uma pergunta e dois botões (um de *OK* e outro de *CANCEL*). O código de retorno (`$?`) será `0` ou `1`, dependendo se você clicou no *OK* ou *CANCEL*. O comando `zenity --entry` pedirá uma entrada de dados e mostrará os mesmos dois botões. Você obterá, da mesma forma, um código de retorno (`$?`) igual a `0` ou `1` e a saída primária receberá o que foi digitado. Experimente!

Opções de diálogo

A **tabela 1** mostra uma síntese das opções da linha de comando para exibição do diálogo. A partir de agora, esmiuçaremos todos os parâmetros aceitos em cada uma destas opções descritas. No entanto, alguns argumentos podem ser utilizados por qualquer uma das opções e é por eles que começaremos nossa aprendizagem.

Parâmetros genéricos

A **tabela 2** contém os parâmetros genéricos do comando `zenity`.

O parâmetro `--title` especifica o título que fica na borda superior da caixa. `--window-icon` permite alterar os ícones padrão das caixas de diálogo, enquanto `--width` e `--height` especificam, respectivamente, a largura e a altura das caixas de diálogo, em pixels.

Calendário

O diálogo `calendar` serve para você escolher uma data. As opções `--day`, `--month` e `--year` permitem a especificação, respectivamente, do dia

Figura 1 Diálogo de calendário.

Exemplo 1: Comando para exibição de calendário

```
01 until data=$(zenity --calendar    \
02     --title="Datas dos vôos"      \
03     --text="Escolha uma data para o
➥ vôo de ida" \
04     --day=$(date +%-d)            \
05     --month=$(date +%-m)          \
06     --year=$(date +%Y))
07 do
08     :
09 done
10 echo $data
11 12-07-2008
```

Tabela 4: Opções de diálogos de advertência

Opção	Efeito
`--text=`TEXTO	Define o texto da advertência
`--no-wrap`	Não permite quebra automática do texto

padrão, o mês e o ano do calendário que será exibido (veja a **tabela 3**).

Veja o fragmento de script do **exemplo 1**. Nele, me aproveitei da facilidade do botão *CANCEL* devolver um código de retorno (`$?`) diferente de zero para permanecer no *loop* do `until`, que nada faz (comando `:`), e só terminar quando uma data for escolhida. O resultado da escolha (**figura 1**) foi para a variável `$data`.

Neste exemplo, não foi necessário informar os parâmetros –day, –month e –year, pois seus valores padrão (default) são os da data de hoje, e foram justamente estes os valores informados. Mas note o uso das máscaras `+%-d` e `+%-m`, necessárias para evitar a inserção de um zero antes dos números de um algarismo. O zenity precisa receber argumentos inteiros decimais nas opções `--day`, `--month` e `--year`, e o comando `date` `+%m` retornaria, por exemplo `09`, o que pode ser interpretado como um inteiro octal.

Como você viu, a saída padrão é no formato *dd-mm-aaaa*. Para alterar esse formato, podemos usar os mesmos caracteres de formatação do comando `date`. Experimente o seguinte:

```
zenity --calendar \
    --date-format="%Y/%m/%d"
```

O diálogo da **figura 1**, após você selecionar uma data, vai retornar algo como `2008/07/28`.

Advertência

A opção de diálogo `--warning` serve para você fazer uma advertência ao operador. O **exemplo 2** contém um fragmento de programa que precisa ser executado pelo usuário.

As opções específicas desse tipo de diálogo são `--text` e `--no-wrap` (**tabela 4**).

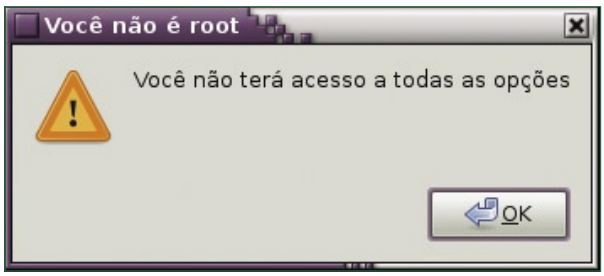

Figura 2 Diálogo de advertência.

Exemplo 2: Diálogo de advertência

```
01 [ $UID -eq 0 ] ||
02     zenity --warning              \
03         --title="Você não é root" \
04         --text="Você não terá acesso
➥ a todas as opções"
```

Na **linha 1**, comparamos a variável do interna `$UID`, do operador, e comparamos com zero (qualquer usuário root tem `UID=0`). Caso seja diferente, um diálogo de atenção será disparado (**figura 2**).

Pergunta

O diálogo de pergunta (`--question`) possui as mesmas opções que o de advertência (**tabela 4**) e lhe fará uma pergunta e oferecerá como resposta dois botões: *OK* e *CANCEL*. Caso seja pressionado o *OK*, o código de retorno (`$?`) será igual a zero; caso contrário, será diferente de zero.

Aproveitando isso, após o diálogo de advertência mostrado na **figura 2**, poderíamos perguntar ao operador se ele deseja mudar para root (**figura 3**), como faz o **exemplo 3**.

Ao fim da execução, a variável `$Resp` teria obrigatoriamente o valor `s` ou `n`.

As páginas de manual, por incrível que pareça, têm um exemplo bastante engraçado do uso desta opção. Confira.

Temos ainda o parâmetro `--no-wrap`, que pode ser usado caso o texto definido por `--text` seja grande e você não queira dividi-lo em mais de uma linha.

Exemplo 3: Diálogo de pergunta

```
01 Resp=n
02 zenity --question      \
03     --title "Responda" \
04     --text "Deseja mudar para
➥ root?" && Resp=s
```

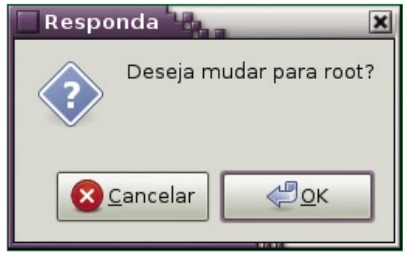

Figura 3 Diálogo de pergunta.

Com entrada

Os diálogos do tipo `--entry` (entrada) abrem uma janela de diálogo requisitando uma entrada de dados ao operador, sendo suas opções descritas na **tabela 5**. Vamos continuar nos exemplos em que o cara deveria ser root para aproveitar todas as facilidades oferecidas por um deter-

Tabela 5: Opções de diálogos de entrada

Opção	Efeito
`--text=TEXTO`	Define o texto da caixa
`--entry-text=TEXTO`	Valor padrão
`--hide-text`	Esconde o valor digitado (senhas)

Exemplo 4: Diálogo de entrada

```
01 if  zenity --question  \
02     --title "Responda" \
03     --text "Deseja mudar para root?"
04 then
05     Senha=$(zenity --entry               \
06         --title "Captura de Senha"       \
07         --text "Informe a senha de root:" \
08         --hide-text)
09 fi
10 echo $Senha
11 123456
```

Exemplo 5: Valor padrão na entrada

```
01 Usuario=$(zenity --entry                                  \
02     --title "Captura nome de usuário"                     \
03     --text "Informe o nome do usuário na máquina remota:" \
04     --entry-text $LOGNAME)
```

Tabela 6: Formato do arquivo de aniversariantes

Campo	Data de Nascimento	Nome	Endereço de e-mail	Telefone
Formato	DD/MM/AAAA	Xxxx Xxxx	Usuário@dominio	(DD) NNNN-NNNN

minado aplicativo, mas, primeiramente, vamos pegar nosso exemplo anterior e inseri-lo nesse contexto (**exemplo 4**).

O comando `if` testou o diálogo de pergunta e, caso o botão clicado seja `OK`, o programa lançará a caixa de entrada de dados da **figura 4** para capturar a senha. Repare que a opção `--hide-text` não deixa o texto que está sendo digitado aparecer na tela.

Também poderíamos oferecer um valor padrão com a opção `--entry-text`, como mostram o **exemplo 5** e a **figura 5**.

– Pois é, amigo, está gostando?

– É, por enquanto o negócio parece ser bom e conciso como você falou...

Figura 4 Diálogo de entrada de dados.

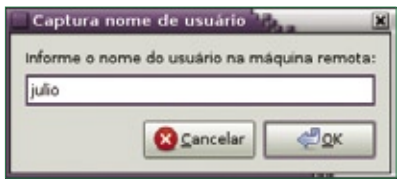

Figura 5 Diálogo com valor padrão para o texto.

– Então, de brincadeira, vamos desenvolver um "sistemeco" para nos lembrar dos aniversários dos amigos. Dessa vez, vou te deixar a incumbência de fazer um script para montar o arquivo de aniversariantes, que tem o layout descrito na **tabela 6**.

– E o separador entre os campos é o dois-pontos (:). O script será feito em shell, porém, todas as interações com o usuário, isso é, leitura dos dados, avisos de erros e advertências, serão gráficas.

– Chico, fecha a conta que eu vou nessa, mas não demoro a voltar... ■

Mais informações

[1] Baixe todas as edições da série Papo de Botequim: http://www.linuxmagazine.com.br/noticia/1729

Sobre o autor

Julio C. Neves é um dos mais antigos colaboradores do SL no Brasil e trabalha atualmente no SERPRO ao lado de Marcos Mazoni, contribuindo para que a principal empresa de TI do Brasil mantenha-se cada vez mais na liderança latino americana no uso e difusão de SL.