De volta ao shell – mas com janelas

Papo de botequim 2.0 Parte IV

No último fascículo da série, aprenda a usar os diversos mecanismos de entrada de dados nas janelas.

por Julio Cezar Neves

— Vamulá, vamulá! Cadê o exercício que te passei?

— Taqui, chefe. Acho que ficou legal. Veja:

```
$ cat festa.sh
#!/bin/bash
#  Resolução do exercício do
#+ Papo de Botequim 2.0.3

emails=$(zenity --list           \
      --title "Convites para a
➥festa"      \
      --text "Quem será
➥convidado?" \
      --multiple                 \
      --separator ' '            \
      --column Amigos            \
      --column ""                \
      --hide-column 2            \
      --print-column 2           \
      $(cut -f2-3 -d: cadastro |
➥tr : ' ' | sort))
echo $emails
```

— Me explique.

— Fácil! Usei a opção `--list` para gerar uma lista simples, mas poderia também ter utilizado a opção `--checklist` que teria o mesmo efeito. A opção `--multiple` permite a escolha de diversas pessoas simultaneamente. Como você pediu para não exibir o endereço de email para não poluir, tive de usar o `--hide-column 2`, e como você queria como resultado justamente os endereços de email, usei a opção `--print-column 2`. Os valores das colunas consegui cortando os campos *nome* e *email* do arquivo `cadastro`. Só que o separador desse arquivo é dois-pontos (`:`), então tive de trocá-los por branco. O `sort` foi só para facilitar a pesquisa.

— Ficou muito bom, mas por que você usou a opção `--separator ' '`?

Tabela 1: Opção --notification

Opção	Efeito
`--text=TEXTO`	Define o texto da notificação
`--listen`	Escuta comandos na entrada primária

Figura 1 Resultado do uso da opção `--notification`.

Exemplo 1: Argumento --listen com descritor de arquivo

```
01 exec 5> >(zenity --notification \
02      --window-icon alert.ico    \
 03     --listen)
```

— Pensei assim: como endereço de email não pode ter espaços em branco e ainda como todas as instruções para enviar email por linha de comando aceitam os destinatários separados por espaço em branco...

— Falou, é isso aí. Ficou muito bom, mas vamos começar o nosso bate-papo porque quero acabar com este assunto sobre *Zenity* ainda hoje. Vejamos a **tabela 1**.

A opção `--notification` exibe um ícone na barra de tarefas, normalmente ao lado do calendário. Então, se você fizer:

```
$ zenity --notification      \
     --window-icon alert.ico \
     --text "Fim do script   \
             em background"
```

você terá um resultado como o da **figura 1**. Aparecerá o ícone `alert.ico` na barra de tarefas e, se você puser o mouse sobre ele, aparecerá o texto definido pela opção `--text` no formato de dica. O ícone permanecerá lá até receber um clique.

Para que o ícone permaneça na barra após o clique, é necessário que você use o argumento `--listen`, e a forma mais fácil de fazê-lo é associando-o a um descritor de arquivo (*file descriptor*) via comando `exec`. Veja o **exemplo 1**.

Nesse ponto, graças ao comando `exec` dando a saída para uma substituição de processos gerada pela construção `>(...)`, o comando `zenity` está atrelado ao descritor de arquivos `5`, e se você colocar o mouse sobre o ícone aparecerá a dica "Notificação do Zenity". Para alterar isso, fazemos:

```
$ echo "tooltip: Inicie novo script
➥em background" >&5
```

onde *tooltip* é um comando para trocar a dica. Os outros comandos disponíveis são: `icon` e `visible`.

Tabela 2: Opção --progress

Opção	Efeito
`--text=TEXTO`	Define o texto do diálogo
`--percentage=PORCENTAGEM`	Define a porcentagem inicial
`-pulsate`	Pulsa a barra de progresso
`--auto-close`	Fecha o diálogo ao atingir 100%
`--auto-kill`	Mata o processo principal se a tecla *CANCELAR* for pressionada

Exemplo 2: Script bp1

```
01 $ cat bp1
02 #!/bin/bash
03 for ((int=1; int<=100; int++))
04 do
05     echo $int# Atualizou percentagem
06    sleep 0.03
07 done | zenity --progress                        \
08     --auto-close                                \
09     --text="Contando vagarosamente de 1 a 100" \
10     --title="Exemplo de Progresso"
```

Quando você terminar e quiser apagar o ícone da bandeja, basta fechar o descritor de arquivo `5`. Para isso, faça:

```
exec 5>&-
```

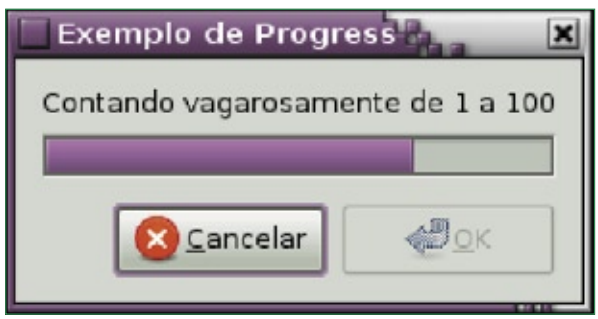

Figura 2 Barra de progresso do exemplo 2.

Apesar de a opção `--notification` ainda não aceitar animações, o meu dicotômico amigo Aurélio A. Heckert (aka Aurium) – dicotômico porque é extremamente competente em shell e SVG, unindo dessa forma o mundo gráfico ao orientado a caractere – fez um script sensacional usando *Bash*, Zenity e SVG que monta um boneco andando na área de notificação. Veja o fonte em [1].

Progresso

Use a opção `--progress` (**tabela 2**) para criar barras de progresso. Usando-se a opção `--progress`, o Zenity lê os dados da entrada primária linha a linha. Caso a linha contenha somente um número, o percentual da barra de progresso é atualizado com esse número.

Vejamos o script `bp1` no **exemplo 2**. Nele, o parâmetro `--auto-close` foi usado para a janela de diálogo fechar-se automaticamente quando a percentagem atingir 100% (**figura 2**).

Se uma linha for prefixada com um jogo-da-velha (`#`), o texto introduzido pela opção `--text` é atualizado automaticamente (**exemplo 3**).

Exemplo 3: Atualização automática

```
01 $ cat bp2
02 #!/bin/bash
03 (
04 echo 10# Atualizou percentagem
05 sleep 1
06 echo "# Atualizando logs do mail"# Substituiu o texto
07 sleep 1
08 echo 20# Atualizou percentagem
09 sleep 1
10 echo "# Resetando os jobs do cron"# Substituiu o texto
11 sleep 1
12 echo 50# Atualizou percentagem
13 sleep 1
14 echo Esta linha será ignorada# Não substituiu o texto
➥porque faltou #
15 sleep 1
16 echo 75# Atualizou percentagem
17 sleep 1
18 echo "# Rebutando o sistema"# Substituiu o texto
19 sleep 1
20 echo 100# Atualizou percentagem
21 sleep 1
22 ) | zenity --progress                          \
23       --title="Atualização dos logs"           \
24       --text="Pesquisando os logs de mail..." \
25       --percentage=0 ||
26          zenity --error                        \
27            --text="Atualização cancelada."
```

Exemplo 4: Previsão do número de arquivos

```
01 $ cat bp4
02 #!/bin/bash
03 sleep $1 &
04 Seg=$(date +%S)
05 i=0
06 exec 8> >(zenity 2>/dev/null --progress --width 300 --auto-kill /
07-title 'Contagem de Segundos')
08 while pidof -s sleep > /dev/null 2>&1
09 do
10      Seg1=$(date +%S)
11    [ $Seg -ne $Seg1 ] && {
12        Seg=$Seg1
13        echo $((++i*100/$1)) >&8
14        echo "# Passaram $i segundos" >&8
15        }
16 done
17 exec 8>&-
```

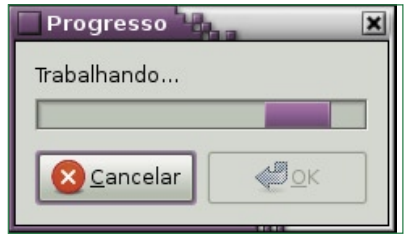

Figura 3 Barra de progresso pulsante.

No **exemplo 3**, o parâmetro `--percentage` foi usado para definir a porcentagem inicial da barra de progresso. Repare que em caso de cancelamento pelo botão *CANCELAR* ou clicando no botão no canto superior direito da janela, o fluxo do programa é redirecionado (`||`) para um diálogo de erro do Zenity.

O **exemplo 4** já conta com duas novas opções. A opção `--pulsate` é usada quando não temos meios para atualizar as porcentagens, e portanto cria a barra pulsante (**figura 3**). Já a opção `--auto-kill` deve ser usada se desejarmos encerrar o comando que iniciou a barra de progresso, no caso o `find`, se o botão *CANCELAR* for pressionado.

No **exemplo 5**, substituir o `sleep` por outro comando com suas particularidades pode praticamente tornar-se genérico para instruções mensuráveis. Por exemplo, podemos saber quantos arquivos serão expandidos em um `tar` (com `tar -tvf arq.tar | wc -l`) e quantos já foram a cada momento (basta ver quantos arquivos existem no diretório e subtrair da quantidade existente antes do `tar`).

Repare que a barra de progresso (**figura 4**) é associada ao descritor de arquivos `8` e a cada incremento de segundos passamos a esse descritor:

- um número para alterar a escala de progresso (`echo $((++i*100/$1))`) que representa a fração do tempo decorrido. Supondo que `$1` seja igual a 20 e já tenham decor-

rido 5 segundos (`$i`), `5*100/20` seria igual a 25, isto é, 25% do tempo (de 20 segundos) já haviam passado;

- uma linha precedida por jogo-da-velha (`echo "# Passaram $i segundos"`) para preencher a área que seria da opção `--text`.

Exemplo 5: Janela de entrada de texto

```
01 zenity --text-info                              \
02      --width=600                                \
03      --height 300                               \
04      --title Pingando...                        \
05      --filename <(ping -c4 `zenity --title "Ping" \
06          --entry                                \
07          --text "Enter the IP or URL to ping..."`)
```

Tabela 3: Opção --scale

Opção	Efeito
`--text=TEXTO`	Define o texto da escala
`--value-INT`	Define o valor inicial (padrão)
`--min-value=INT`	Define `INT` como o menor valor da escala
`--max-value=INT`	Define `INT` como o maior valor da escala
`--step=INT`	Define a unidade da escala
`--print-partial`	Exibe valores parciais
`--hide-value`	Esconde os valores da escala

Exemplo 6: Escala gráfica

```
01 Aluno=Zezinho
02 Nota=$(zenity --scale                  \
03      --title "Nota dos Alunos"         \
04      --text "Informe a nota de $Aluno" \
05      --min-value 0                     \
06      --max-value 5                     \
07      --value 2)
08 echo $Nota
09 2
```

Exemplo 7: Mudando a cor de fundo

```
01 zenity --scale                          \
02     --title "Cores do Fundo de Tela"    \
03     --text "Escolha a cor desejada"     \
04     --print-partial                     \
05     --max-value 9 |
06    xargs -i bash -c "tput setab {}; clear"
```

Exibir texto

A opção `--text-info` deve ser usada para abrir uma janela de exibição de textos. Vejamos no **exemplo 5** um fragmento de programa para entender como funciona.

O problema nesse tipo de construção é que a janela de texto com as estatísticas do `ping` aparecerão somente após a conclusão do comando (**figura 4**), pois repare que o arquivo definido pela opção `--filename` é o próprio comando `ping`.

Outro problema que surge é a perda de formatação em saídas de comandos colunados como o `ls -l`. Isso se dá porque não podemos optar por uma fonte monoespaçada, e então a largura das letras é variável. Teste isso para confirmar:

```
$ zenity --text-info           \
    --title "Lista de arquivos" \
    --width=600                \
    --filename <(ls -l)
```

Escala

Use a opção `--scale` (**tabela 3**) para exibir uma escala numérica. No **exemplo 6** (**figura 5**), o valor mínimo (`--min-value`) da nota seria `0`, o máximo (`--max-value`) seria `5` e a escala apontaria inicialmente para `2` (`--value`).

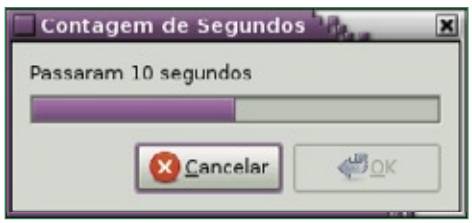

Figura 4 Ferramenta gráfica de `ping`.

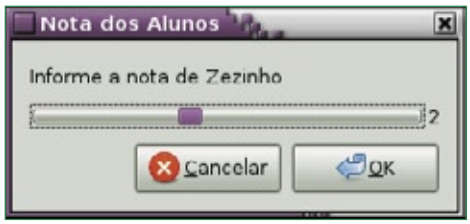

Figura 5 Escala gráfica.

Usando a opção `--print-partial`, a cada ponto que parasse o cursor da escala, o valor atual seria mandado para a saída padrão. A opção `--hide-value` não mostra na janela o valor atual do cursor, o que dificulta muito a escolha.

Veja no **exemplo 7** uma forma de mudar a cor de fundo de uma tela orientada a caractere, aproveitando que a opção `--print-partial` manda para a saída primária o valor da escala a cada vez que se solta o botão do mouse.

Uma das formas de mudar a cor de fundo é com o comando `tput setab n`, em que `n` varia de zero a nove, sendo que as cores variam somente entre os valores `0` (preto) e `7` (branco). O `8` nada faz e o zero restaura a cor inicial de fundo.

Como a cada parada do mouse podemos dar somente um comando e era necessário usarmos dois (o `setab` para mudar a cor e o `clear` para propagar a cor por toda a tela), foi necessário o comando `bash -c`, que, como sabemos, executa em um subshell os comandos entre aspas. O `xargs` foi usado para passar a cor como um parâmetro para o `setab`.

Câmbio final

Nos últimos quatro meses, tentei passar para vocês no Papo de Botequim tudo que sei sobre Zenity. Como o nosso assunto termina por aqui, só tem uma coisa a dizer:

Fim de Papo :-(
Câmbio final e desligo. ■

Mais informações

[1] Boneco caminhando: http://www.colivre.coop.br/Aurium/BonecoCorredorBash

Sobre o autor

Julio C. Neves agradece ao SERPRO, que descreve como empresa exemplo do uso de Software Livre no Governo Federal.