

De volta ao shell – mas com janelas

# Papo de botequim 2.0

## Parte III

Janelas de listas de opções com o Zenity.

por Julio Cezar Neves

– Bem, meu amigo “shelleiro”. Você fez o exercício que te passei?

– Fiz, mas não sei se está certo. Primeiro fiz de uma forma e depois achei que você tinha posto uma casca de banana para eu escorregar e refiz. O **exemplo 1** mostra como ficou a minha versão final.

Primeiramente, fiz um *loop* de *while* com duas saídas: ou ele sai no *break* caso tenha informado um diretório correto ou encerra o programa no *exit 1* após ter escolhido abandonar na opção *--question* que coloquei.

Em seguida, troco o *IFS* (*Inter Field Separator*), que é o separador do Bash, para que o *for* pegue cada um dos arquivos selecionados. Me lembrei que você disse que o separador padrão era uma barra vertical (|) e usei esse macete do Shell. Dentro deste *loop* é que encontrei a pegadinha que você deixou: primeiro pensei que você quisesse que usasse a opção *--confirm-overwrite*, mas isso só seria válido se estivesse escolhendo o caminho completo do arquivo destino. Como estávamos escolhendo os arquivos a serem

movidos, tive de fazer uma ginástica (*ls \$DirDest | grep \$(basename \$Arq) > /dev/null*) para ver se cada um dos arquivos selecionados não existia no diretório destino. Era isso mesmo?

– Era, e o exercício ficou muito legal. Vamos pedir os chopes de praxe e aprender a usar a opção *--list* na sua plenitude.

Chico, dois chopes, um sem colarinho!

Hoje falaremos da opção *--list* (**tabela 1**) que, como o nome diz, gera no *zenity* listas de todas as formas, como:

- ◆ Listas propriamente ditas;
- ◆ Listas de *radio buttons*;
- ◆ Listas em *Check Lists*.

Esta opção lhe oferece uma lista para a escolha de um ou mais de seus componentes. Divide-se em três tipos: listas propriamente ditas, *radio list* e *check list*.

## Listas

Vejamos primeiramente o **exemplo 2** de utilização de lista. O resultado é mostrado na **figura 1**.

Mas isso ainda está muito simples. No **exemplo 3** vamos complicar um pouco mais.

**Tabela 1: Opção --list**

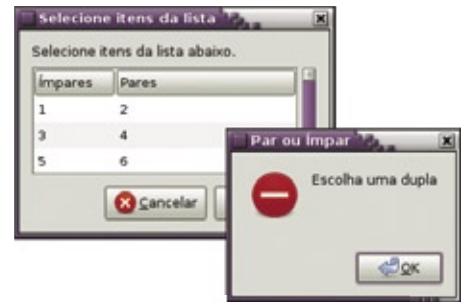
Opção	Efeito
<i>--text=TEXTO</i>	Define o texto da caixa.
<i>--column=TEXTO</i>	Define os cabeçalhos das colunas (um para cada coluna).
<i>--checklist</i>	Coloca <i>check boxes</i> na 1ª coluna.
<i>--radiolist</i>	Coloca <i>radio boxes</i> na 1ª coluna.
<i>--multiple</i>	Permite a seleção de diversas linhas.
<i>--separator=SEPARADOR</i>	Usado com <i>--multiple</i> , especifica o separador quando retornar mais de uma linha.
<i>--editable</i>	Permite editar os itens exibidos.
<i>--print-column=TEXTO</i>	Especifica qual coluna mandar para a saída primária. O retorno <i>default</i> é a 1ª. <i>ALL</i> pode ser usado para imprimir todas.
<i>--hide-column=INT</i>	Omite a coluna definida por <i>INT</i> .

## Exemplo 1: Exercício da última aula

```
01 #!/bin/bash
02 # Movendo arquivos a partir do diretório corrente
03 # Escolha do diretório destino
04 while true
05 do
06     DirDest=$(zenity --file-selection \
07         --directory \
08         --title "Escolha o diretório de destino")
09     && break
10     zenity --question \
11         --title "Falta diretório de destino" \
12         --text "Diretório destino não informado:
13         - Clique OK para finalizar;
14         - Clique CANCELAR para escolher diretório
15 destino" && exit 1
16 done
17 # Escolha dos arquivos
18 Arqs=$(zenity --file-selection \
19     --title "Escolha arquivos para mover para
20 $DirDest" \
21     --multiple) || {
22     echo Não foram escolhidos arquivos
23     exit 1
24 }
25 # Verifica se os arquivos já existem no
26 #+ diretório de destino e move um por um
27 IFS='|' # A barra (|) é o separador default do
28 --file-selection
29 for Arq in $Arqs
30 do
31     ls $DirDest | grep $(basename $Arq) >
32 /dev/null || {
33     mv $Arq $DirDest
34     continue
35 }
36 zenity --question \
37     --title "Arquivo já existe" \
38     --text "Já existe $(basename $Arq)
39 em $DirDest
40 - Clique em OK para sobregravar ou;
41 - Clique em CANCELAR para desistir"
42 && mv $Arq $DirDest
43 done
```

## Exemplo 2: Lista simples

```
01 until Escolha=$(zenity --list \
02     --column Ímpares \
03     --column Pares \
04     1 2 \
05     3 4 \
06     5 6 \
07     7 8)
08 do
09     zenity --error \
10         --title "Par ou Ímpar" \
11         --text "Escolha uma dupla"
12 done
```



**Figura 1** Nesse exemplo, se o cara clicar em CANCELAR ou simplesmente fechar a janela da lista, será dada uma mensagem de erro.

zenity (também poderia ter usado <TAB>).

O exemplo 4 mostra como deixar essa lista mais bonita:

Como mostra a figura 2, agora a lista tem cabeçalho (--title) e texto explicativo (--text). Tá ficando bom! Mas, peraí, e se eu quiser escolher mais de um? Assim como na seleção de arquivos (--file-selection), as listas também permitem mais de uma escolha, e é para isso que serve a opção --multiple (exemplo 5). O resultado é mostrado na figura 3.

Quando usamos esse recurso, é legal que ele venha acompanhado de --separator (exemplo 6) para especificar qual será o caractere que servirá como separador entre as respostas recebidas. Se não usarmos essa opção, as nossas escolhas virão separadas por barras verticais (|).

Um último exemplo bastante interessante. Veja esse arquivo incompleto de UFs:

```
$ cat UFs
MG
PA
RJ
SP
```

O exemplo 7 mostra uma forma de capturar uma UF da tela e, ao mesmo tempo, incrementar esse arquivo:

Nesse momento, se você der um duplo clique em Nova UF (que será

Este é um exemplo ainda bem simples, sem cabeçalho (--title) e sem texto explicativo (--text). Especifiquei duas colunas (LoginName e UID) e peguei estes dois campos do /etc/passwd, trocando os dois-pontos (:) por espaço em branco, para servir como separador de campos do

o primeiro item da lista, como mostra a **figura 4**, e escrever, por exemplo, `SC`, a variável `$UF` receberá esse valor. O que permite isso é a opção `--`

`editable`. O mesmo pode ser feito (e acho até que de forma mais amigável e óbvia) usando a opção `--entry`, sob a forma de *ComboBox*.

O **exemplo 8** mostra como testar se o valor informado já existia no arquivo `UFs` e em seguida atualizá-lo.

A opção `-q` (*quiet*) do `grep` serve para que ele não mande a saída para a tela, caso a UF informada tenha sido encontrada no arquivo `UFs`.

### Exemplo 3: Opções extraídas de arquivo

```
01 zenity --list \
02     --column LoginName \
03     --column UID \
04     $(cut -f1, 3 -d: /etc/passwd | tr : ' ')
```

### Exemplo 4: Usando as opções --title e --text

```
01 zenity --list \
02     --title "Usuários Cadastrados" \
03     --text "Selecione o usuário desejado" \
04     --height 300 \
05     --column LoginName \
06     --column UID \
07     $(cut -f1,3 -d: /etc/passwd | tr : ' ')
```

### Exemplo 5: Usando a opção --multiple

```
01 SO=$(zenity --list \
02     --title "Sistemas Operacionais" \
03     --text "Quais podemos chamar de sistemas \
04     operacionais?" \
05     --height 230 \
06     --multiple \
07     --column "Sistema Operacional" \
08         "M$ Vista" \
09         "M$ XP" \
10         Linux \
11         Unix)
12 echo $SO
13 Linux|Unix
```

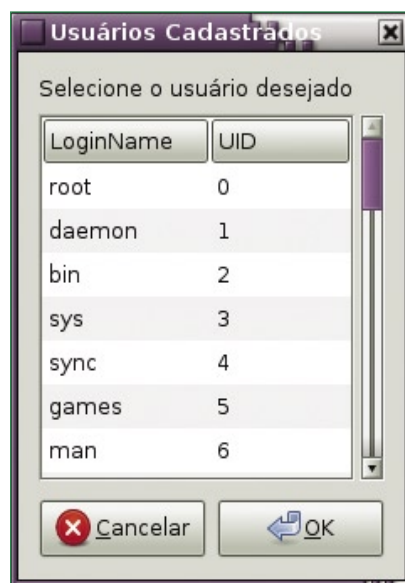
### Exemplo 6: Podemos determinar o separador para as respostas dadas com a opção --separator

```
01 SO=$(zenity --list \
02     --title "Sistemas Operacionais" \
03     --text "Quais podemos chamar de sistemas \
04     operacionais?" \
05     --height 230 \
06     --multiple \
07     --separator ^ \
08     --column "Sistema Operacional" \
09         "M$ Vista" \
10         "M$ XP" \
11         Linux \
12         Unix)
13 echo $SO
14 Linux^Unix
```

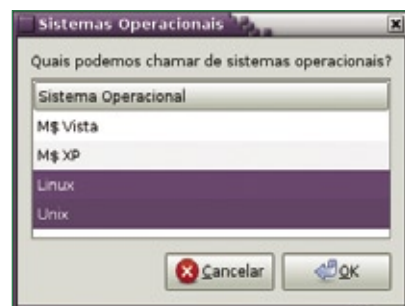
## Radio Lists

A múltipla escolha não é possível se a lista for uma *radio list*. A **tabela 2** mostra algumas opções para esse tipo de diálogo.

Algumas vezes não exibimos uma coluna, mas queremos seu valor como retorno. O **exemplo 9** mostra uma escolha de resoluções de telas (**figura 5**).



**Figura 2** Janela de opções com título e informação personalizados.



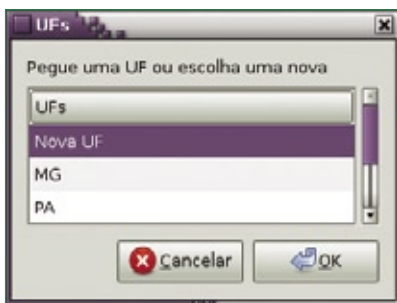
**Figura 3** Escolha de múltiplas opções.

**Exemplo 7: A opção --editable**

```
01 UF=$(zenity --list \
02     --editable \
03     --title="UFs" \
04     --text "Pegue uma UF ou escolha uma nova" \
05     --column="UFs" "Nova UF" $(cat UFs))
```

**Tabela 2: Opções do tipo radio**

A opção	Define
--radiolist	Uma lista com botões de rádio
--column	Os cabeçalhos das colunas
TRUE ou true	Um botão inicialmente marcado
FALSE ou false	Um botão inicialmente desmarcado

**Figura 4** Inclusão de opção na lista.

Nesse exemplo, você deve ter notado que existem três colunas, mas a segunda foi colocada somente para facilitar a programação posterior. Assim sendo, ela não deveria ser exibida e para isso usamos a opção `--hide-column`.

Vimos ainda que a segunda coluna é o valor retornado por padrão. Isso pode ser alterado com a opção `--print-column`, como mostrado no **exemplo 10**.

Quando escolhi o melhor sistema operacional (**figura 6**), se não tivesse usado a opção `--print-column=3`, a variável `$S0` teria recebido `Comunidade`, que por ser a segunda coluna é a padrão. Graças a essa opção, a variável recebeu "Linux".

Caso tivesse especificado `--print-column 2,3`, a resposta obtida seria `Comunidade|Linux`.

**Check Lists**

Nas *check lists* é comum termos mais de uma opção e por isso é desnecessário usarmos `--multiple`. Nesse caso, o uso de `--separator` é encorajado (**exemplo 11**).

A **figura 7** mostra o diálogo mais básico obtido com a opção `--checklist`. O **exemplo 12** é um pouco mais completo e complexo e mostra a janela da **figura 8**.

Como o `-o` é a condição *ou* do comando `find`, a saída já está prontinha para usar como `find . -type f $Usus`.

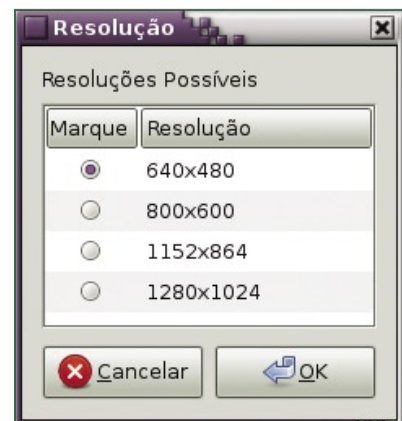
Após a substituição de variáveis feita pelo *Shell*, o comando executado será `find . -type f -user root -o -user julio`.

Vamos analisar o **exemplo 12**:

O primeiro comando `cut -f1,3 -d: /etc/passwd` retira o primeiro e o terceiro campos `login` e `UID`, separados por dois-pontos `:` de todos os usuários registrados em `/etc/passwd`. Após esse comando, a primeira linha recebida de `/etc/passwd` ficaria `root:0`.

**Exemplo 8: Incrementar o arquivo com a opção fornecida**

```
01 grep -q "^$UF" UFs ||
02 {
03     echo $UF >> UFs
04     sort UFs -o UFs
05 }
```

**Figura 5** Lista de opções usando o tipo *radio*.**Exemplo 9: Escolha de resolução de tela**

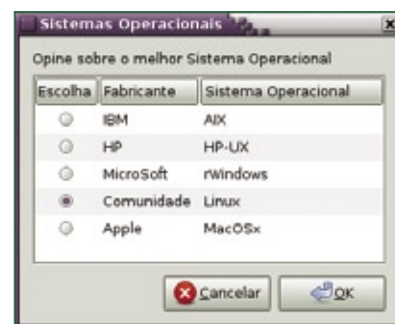
```
01 Resolucao=$(zenity --list \
02     --height 230 \
03     --title "Resolução" \
04     --text "Resoluções Possíveis" \
05     --radiolist \
06     --hide-column 2 \
07     --column "Marque" \
08     --column " " \
09     --column "Resolução" \
10     TRUE 0 "640x480" \
11     FALSE 1 "800x600" \
12     FALSE 2 "1152x864" \
13     FALSE 3 "1280x1024")
```

**Exemplo 10: Utilização da opção --print-column**

```

01 SO=$(zenity --list \
02     --title "Sistemas Operacionais"
03     --text "Opine sobre o melhor
➔ Sistema Operacional" \
04     --radiolist \
05     --width 350 \
06     --height 265 \
07     --print-column 3 \
08     --column "Escolha" \
09     --column "Fabricante" \
10     --column "Sistema Operacional" \
11     TRUE IBM AIX \
12     FALSE HP HP-UX \
13     FALSE MicroSoft rWindows \
14     FALSE Comunidade Linux \
15     FALSE Apple MacOSx )
16 echo $SO
17 Linux

```



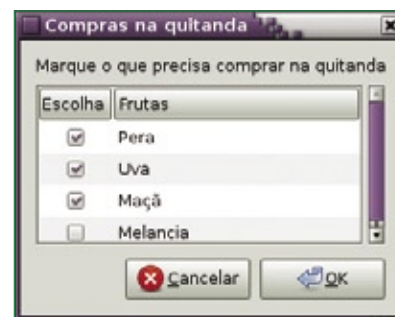
**Figura 6** Janela de opções em três colunas.

**Exemplo 11: Exemplo simples de Check lists**

```

01 zenity --list \
02     --title "Compras na quitanda" \
03     --text "Marque o que precisa comprar na
➔ quitanda" \
04     --checklist \
05     --column Escolha \
06     --column Frutas \
07     true Pera \
08     true Uva \
09     true Maça \
10     false Melancia \
11 Pera|Uva|Maça

```



**Figura 7** Diálogo básico usando Check lists.

**Exemplo 12: Exemplo simples de Check lists**

```

01 Usus='-user'$(zenity --list \
02     --checklist \
03     --title "Pesquisa de arquivos por usuário" \
04     --text "Selecione usuários para listar
➔ arquivos" \
05     --height 400 \
06     --width 260 \
07     --separator ' -o -user ' \
08     --column Marque \
09     --column "Login Name" \
10     --column UID \
11     $(cut -f1,3 -d: /etc/passwd | sort | \
12         tr : ' ' | xargs -L1 echo FALSE))
13 echo $Usus
14 -user root -o -user julio

```

➤ A saída é então redirecionada para um `tr`, que troca os dois-pontos por um espaço em branco. Após esse comando,

a primeira linha recebida de `/etc/passwd` ficaria `root 0`.

➤ Esses dois campos separados por um espaço em branco vão para

o comando `xargs` que, com a opção `-L1`, joga uma linha de cada vez para o final do `echo`.

➤ Após este comando, a primeira linha recebida de `/etc/passwd` ficaria `FALSE root 0`, criando dessa forma as três colunas para serem listadas.

Repare ainda no **exemplo 12** que o separador (`--separator`) não precisa ter somente um caractere.

O **exemplo 13** mostra como gerar a listagem dos arquivos, que produz a janela da **figura 9**.

Aqui tem uma pegadinha: o `find` sempre retorna `true` mesmo quando não acha arquivo algum que atenda aos seus critérios. Então tive de guardar a sua saída na variável `$Arqs` para testar se ela possui conteúdo (`[ $Arqs ]`) quando a defino como uma coluna do zenity. Caso contrário, abro uma caixa de atenção para dizer que o `find` não deu nenhum retorno. Poderia nesse ponto colocar também um `exit 1` para não aparecer a lista de arquivos vazia.

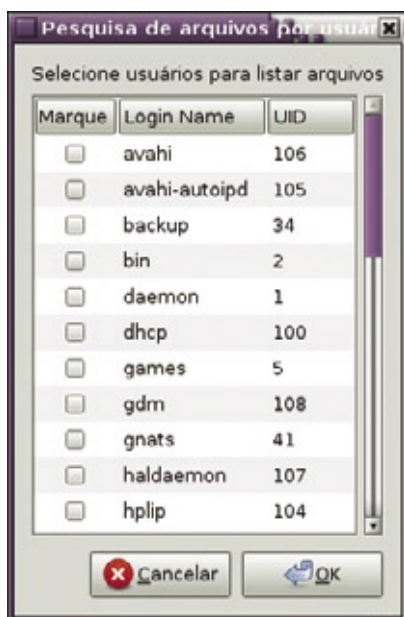


### Exemplo 13: Listagem de arquivos

```
01 zenity --list \
02     --column Arquivos \
03     $(Arqs=$(find . -type f $Usus); [ "$Arqs" ]
04     && echo "$Arqs" ||
05     zenity --warning \
06     --text "Usuário sem arquivos")
```

### Tabela 3: Formato de arquivo usado na próxima aula

Campo	Data de Nascimento	Nome	Endereço de e-mail	Telefone
Formato	AAAAMMDD	Xxxx Xxxx	usuário@dominio	(DD) NNNN-NNNN



**Figura 8** Janela Check list em três colunas.



**Figura 9** Janela Check list em três colunas.

– Cara, quanto tempo perdi fazendo interface a caractere. Esse tal de zenity é legal mesmo!

– É, ele é muito legal, mas não é a panacéia universal. Para muitas aplicações, a interface a caractere ainda é a mais indicada.

Por hoje chega, mas antes vou te deixar um exercício para você consolidar o que aprendemos dessa vez. No primeiro dia que conversamos sobre o zenity, te pedi para fazer um programa para ler um arquivo com o formato mostrado na **tabela 3**.

Pois é, pegue esse arquivo e monte uma lista para que você escolha as pessoas que você quer convidar para o seu aniversário. Detalhe: você deverá poder escolher várias pessoas ao mesmo tempo. A lista exibirá os nomes para seleção, mas retornará os emails das pessoas selecionadas, para que você possa enviar-lhes o convite. ■

### Sobre o autor

**Julio C. Neves** trabalha no SERPRO, que ele descreve como empresa exemplo do uso de Software Livre no Governo Federal. Ele também dá cursos de Shell e Zenity em São Paulo, Brasília ou turmas fechadas em qualquer localidade.

# Uma empresa tão livre quanto a sua imaginação.

Pensando na sua liberdade de pensamento, a F13 Tecnologia oferece produtos, soluções e serviços em Linux e Softwares livres, como suporte técnico presencial ou remoto e cursos de formação com certificação, tais como:

- Formação Linux com ênfase na LPI (4 módulos totalizando 160 horas)
- Formação PHP (3 módulos totalizando 120 horas)
- Firewall Avançado (40 horas)
- Controle de versões com CVS, SVN e Trac (8 horas)
- Virtualização com Xen (40 horas)
- Serviço de diretórios com OpenLdap (40 horas)
- Correio Eletrônico Avançado (40 horas)
- Voip & Asterisk com ênfase em DialPlan (40 horas – Curso ministrado por instrutor com certificação DCAP)
- Administração de Bancos de Dados Livres (PostgreSQL e MySQL – 40 horas)



(85) 3252.3836  
www.f13.com.br