AbsTK

uma biblioteca para desenvolvimento unificado de aplicações em modo texto e gráfico

Hisham H. Muhammad
André Detsch

Resumo

- Introdução
- AbsTK
  - Classe `Screen`
  - Classe `Wizard`
- Implementações
- Exemplo de uso
  - Screenshots
- Considerações finais

Introdução

- Desenvolvimento de interfaces para instalação e configuração de sistemas
- Interfaces amigáveis “implicam” o uso de *widgets* gráficos
  - Lidar com aspectos de exibição
  - Qt, GTK+, ...
  - Inviabilidade de uso no console
- AbsTK: prototipação de interfaces abstraindo aspectos visuais específicos
  - Implementado em Python

AbsTK - Abstract Tool Kit

- Aplicação alvo: instalador do GoboLinux
  - Modo gráfico e console
- Objetivos
  - Simplicidade de prototipação
  - Independência de toolkit de exibição
  - Permitir o uso de campos arbitrários
  - Definição simples de interações
- Componentes: campos, telas, container (`Wizard`)
- Suporte a internacionalização (Qt Linguist)

Classe Screen

- Representa uma tela de interface
- Diversos métodos para adição de campos
  - `addButton`, `addBoolean`, `addList`, ...
- Na criação de campos podem ser definidos
  - Nome do campo – para acesso dos valores
  - Label – identifica o campo visualmente
  - Valor *default* – inicializa o campo
  - Mensagem de ajuda – *tooltip*
  - Callback – acionado quando campo é alterado

Classe `Wizard`

- Implementa um container de telas
- Acesso e alteração dos valores dos campos
  - `getValue(campo)`, `setValue(campo, valor)`
- Ativação e desativação de campos
  - `setEnabled(campo, true/false)`
- Quando possui apenas uma tela, apresenta uma caixa de diálogo

Implementações

- Implementação PyQt
  - Wrappers para widgets Qt
  - Layout automático - `QGridLayout`
  - *Sinais / slots* para implementação dos callbacks
- Implementação NCurses
  - NCurses disponível por padrão no Python
  - Widgets e callback implementados internamente
  - Oferece scroll vertical

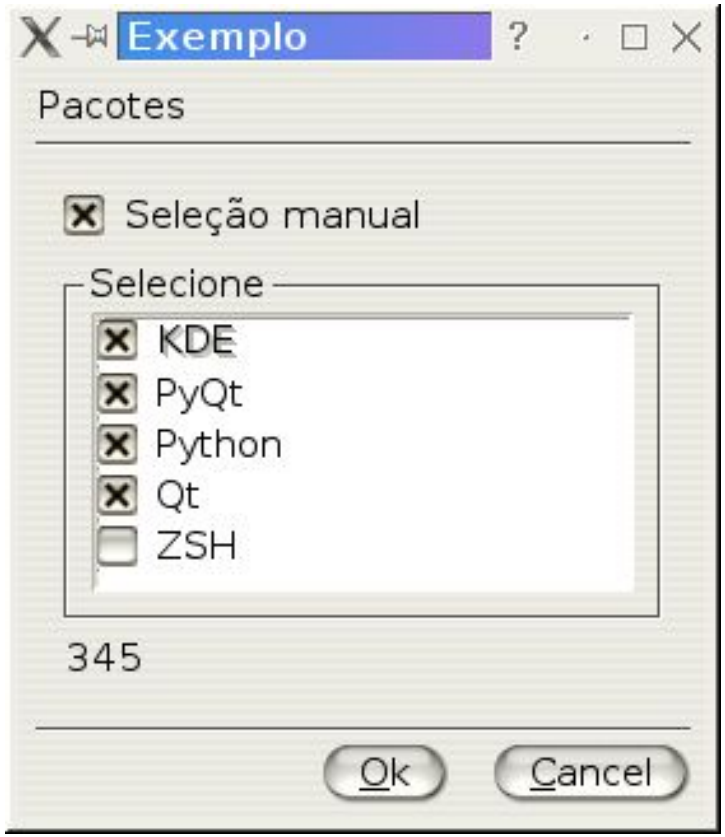
Exemplo de uso

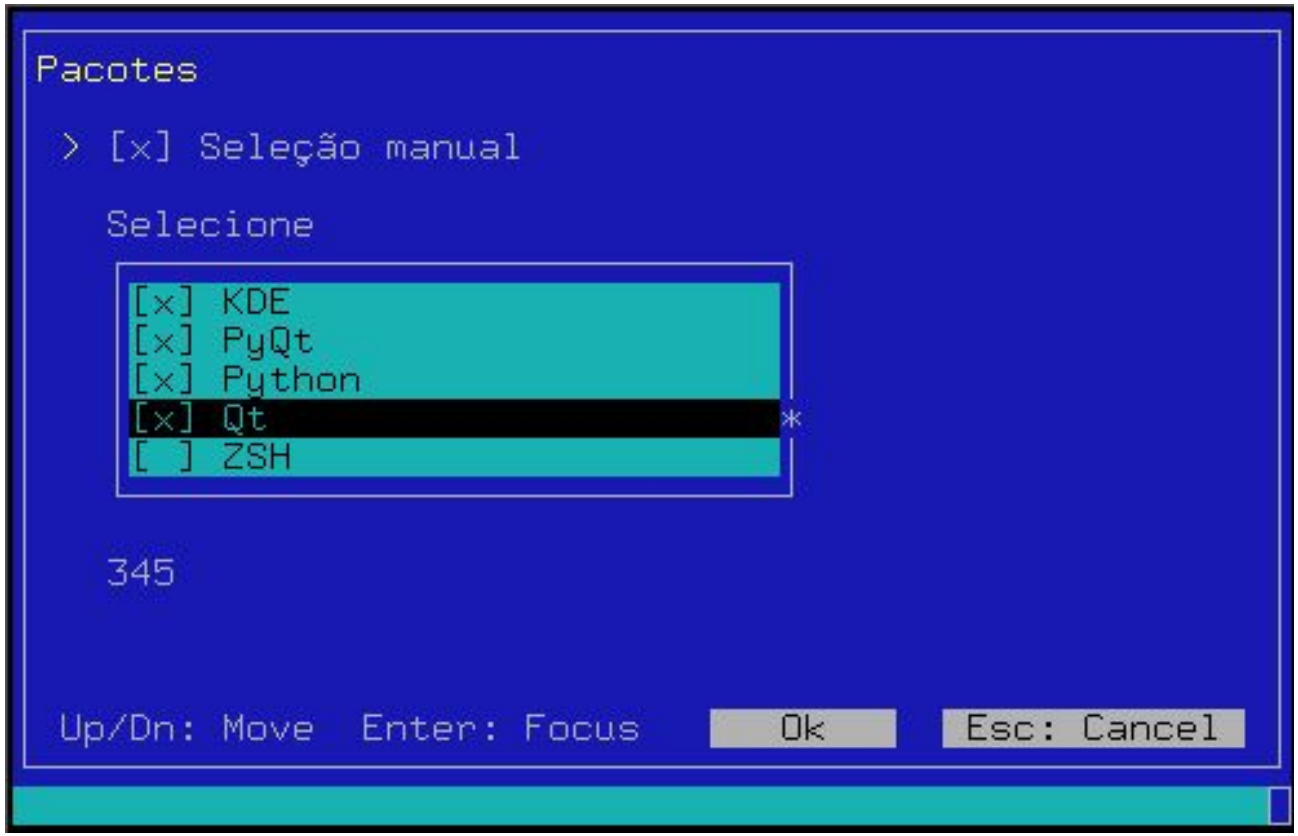
```
tamanhos={'KDE':250, 'Python':30, 'Qt':50, 'PyQt':15 }
nomes_pacotes = tamanhos.keys()

wizard = Wizard('Exemplo')
tela = Screen('Pacotes')

tela.addBoolean('habilita_selecao', 'Seleção manual',
   True, '', habilita_desabilita_lista)
tela.addCheckList('pacotes', 'Selecione',
   (nomes_pacotes,[]), '', soma_lista_e_atualiza_campo)
tela.addLabel('soma', '0')
wizard.addScreen(tela)

wizard.start()
```

Exemplo de uso

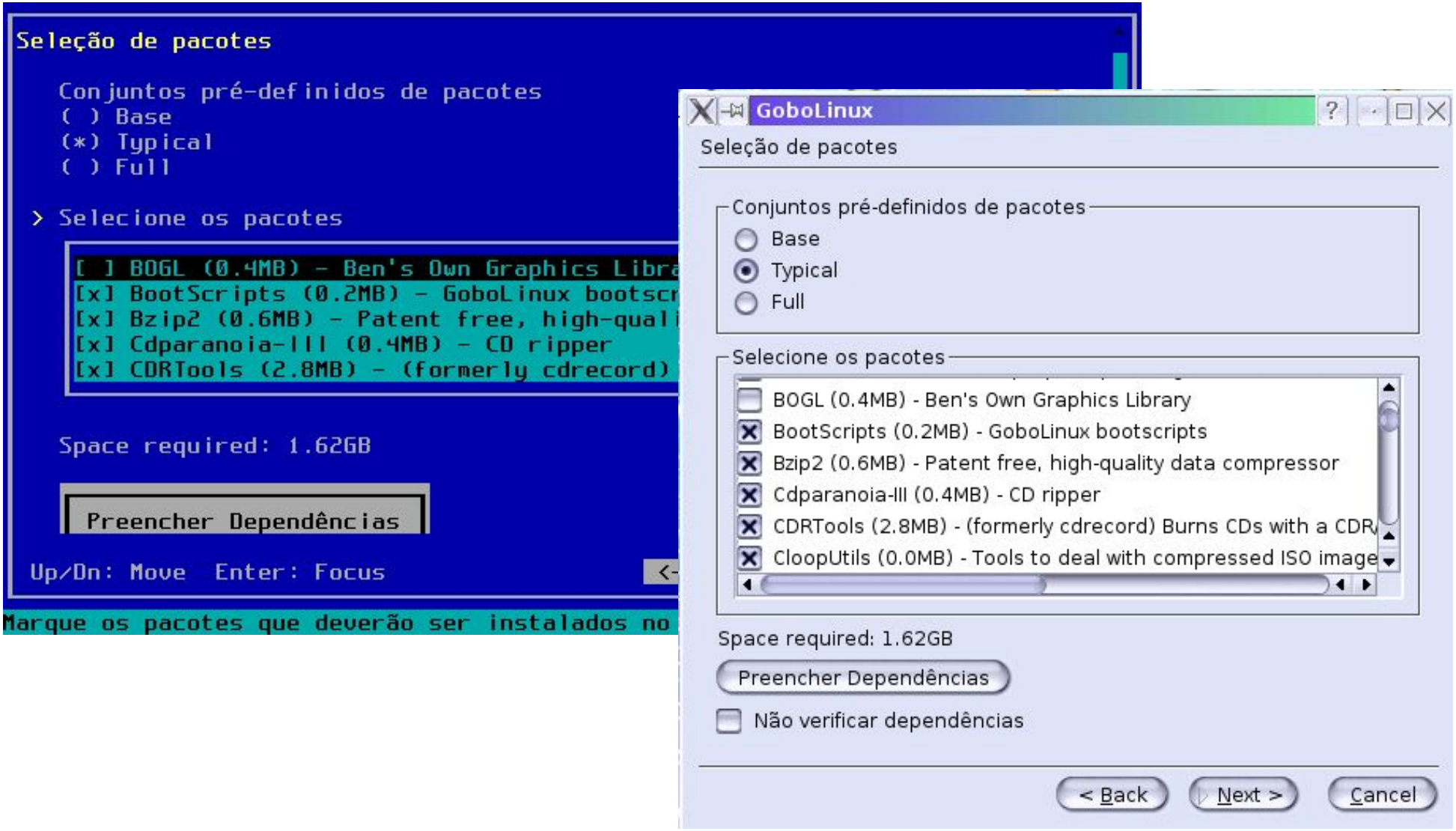
```
def habilita_desabilita_lista() :
    wizard.setEnabled('pacotes', wizard.getValue
     ('habilita_selecao'))

def soma_lista_e_atualiza_campo() :
    selecionados = wizard.getValue('pacotes')[1]
    s = 0
    for cada_um in selecionados :
        s = s + tamanhos[cada_um]
    wizard.setValue('soma', str(s))
```

Screenshots

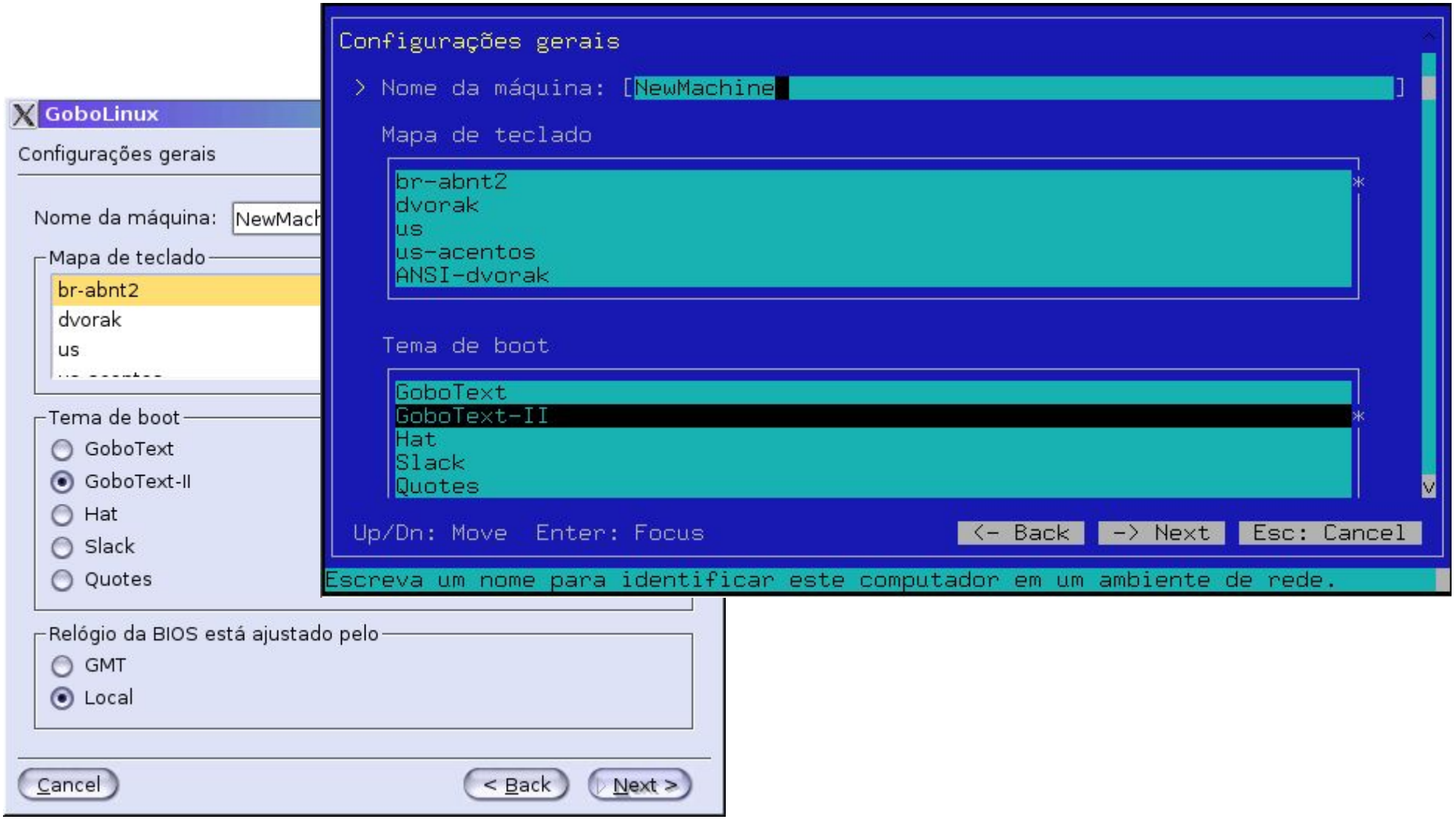




Screenshots



Screenshots



Considerações finais

- Bastante adequado para seu objetivo inicial, mas indicado também para uso geral
- Código estável, licença GPL
  - http://www.gobolinux.org/abstk
- Para o futuro...
  - Implementação de novos *back-ends*
  - Suporte a novos tipos de campos