

## Table of Contents

<b><u>Conceitos Básicos de Kylix</u></b> .....	<b>1</b>
<u>Apresentando e Executando</u> .....	1
<u>Modificando as</u> .....	2
<u>características de um objeto</u> .....	2
<u>Trabalhando com Eventos</u> .....	3
<u>Vantagens da Interface do Kylix</u> .....	5

# Conceitos Básicos de Kylix

Neste artigo voltado para todas as categorias de usuários, são apresentados os conceitos básicos de programação do Kylix e o desenvolvimento de um pequeno programa.

Baseado no Delphi, o Kylix traz um ambiente completo de desenvolvimento rápido de aplicações (RAD – Rapid Application Development). Apesar de sua facilidade de utilização, é uma ferramenta muito poderosa, sendo útil tanto ao desenvolvedor iniciante, quanto aos desenvolvedores experientes.

## Apresentando e Executando

Ao iniciar a execução do Kylix, você verá uma tela semelhante à da *Figura 1*. Esta tela é muito semelhante à do Delphi. Isto não é coincidência.

A janela central é a tela principal de uma aplicação Kylix. Para montar a interface com o usuário, o desenvolvedor escolhe o componente desejado na paleta de componentes (*Figura 2*), clicando sobre ele, para selecioná-lo e, em seguida, na janela, para colocá-lo lá. Pode-se então movimentar o componente e ajustar seu tamanho, dando o formato desejado.

As janelas do Kylix são chamadas de Forms. Associado a cada Form, há um arquivo de código correspondente, chamado de Unit. O código fonte dos programas em Kylix é escrito em Pascal orientado a objetos. Para visualizar a janela de código (*Figura 3*), basta teclar F12 ou selecionar a opção de menu View/Toggle Form/Unit.

Como você pode ver na *Figura 3*, mesmo um novo projeto já tem algum código. Uma Unit está intimamente ligada à Form: quando se cria uma nova Form, automaticamente é criada uma nova Unit e o código nela reflete o que está incluído na Form. Ao se inserir um novo componente na Form, o Kylix gera o código correspondente a ele na Unit. A Borland chama isto de *two-way tool* (ferramenta de duas vias).

Um aplicativo Kylix é chamado de Projeto e pode consistir de uma ou mais Forms. Excepcionalmente, um aplicativo pode não conter nenhuma Form, como por exemplo, um aplicativo do tipo console, para ser usado na linha de comando.

Para dar funcionalidade a um programa, deve-se inserir componentes na Form. O programa básico não mostra nada, a não ser uma janela vazia. Para adicionar componentes, você deve usar a paleta de

componentes, mostrada na *Figura 2*. Ela contém os diversos componentes que podem ser inseridos numa Form. Os componentes são agrupados em guias, para facilitar sua escolha. Esta separação é feita por funcionalidade:

- **Standard** – Contém os componentes de uso mais comum.
- **Additional** – Nesta guia são encontrados componentes mais elaborados, com funcionamento interno mais complexo.
- **System** – Aqui são colocados os componentes voltados à interação com o sistema operacional.
- **DBExpress** – Componentes de acesso a bancos de dados.
- **Data Controls** – Componentes data aware, isto é, os componentes são semelhantes aos da guia Standard ou Additional, porém têm ligação com as tabelas dos bancos de dados.

Além destas guias, podem ser encontradas diversas outras. Como novos componentes são criados usando-se o próprio Kylix, é muito comum adicionar componentes elaborados por terceiros, que acrescentam novas guias à paleta.

## Modificando as características de um objeto

Quando um componente é colocado numa Form, ele possui características padrões, que nem sempre são desejadas no programa final: Por exemplo, ao se colocar um TLabel na Form, o texto que é apresentado é “LabelXX”, onde XX é um número que varia conforme os labels que estão na Form.

Além disso, a posição ou tamanho podem não corresponder ao que se deseja. O Kylix permite alterar as propriedades de um componente e visualizar o efeito final antes de executar o programa. Para alterar a posição de um componente, basta selecioná-lo clicando sobre ele e arrastá-lo para a posição desejada. Para alterar seu tamanho, basta clicar sobre a borda, arrastando-a para o tamanho desejado.

Quando se quer alterar qualquer outra característica, utiliza-se o Object Inspector mostrado na Figura 4. Ele tem duas guias, Properties e Events. A primeira mostra as propriedades do componente e a segunda, seus eventos, que serão explicados mais adiante.

Quando um componente é selecionado, suas propriedades e respectivos valores também são mostrados. Para alterar o valor de uma propriedade, basta clicar na segunda coluna do Object Inspector, na frente da

propriedade desejada. Se, por exemplo, você colocar um label na Form e selecioná-lo, poderá alterar o texto que ele apresenta clicando em frente à propriedade Caption e digitando o novo texto. À medida que vai se digitando, o label reflete o texto na Form. Desta maneira, pode-se visualizar as modificações à medida que são processadas.

Com estas informações, podemos criar um primeiro programa em Kylix, o já famoso “Alô, Mundo”. Vemos aqui a verdadeira facilidade do desenvolvimento RAD: para criar uma aplicação como esta para X, em linguagem C, deveremos usar aproximadamente 100 linhas de código; para criar nosso programa não precisamos digitar nem uma linha de código!

Para criar o programa, tudo o que temos de fazer é inserir um componente TLabel, da guia Standard na Form, e mudar a propriedade Caption para “Linux, Cheguei!!!”. Para deixar o programa um pouco mais bonito, podemos escolher a propriedade Font e alterá-la para Lucida, tamanho 36, Bold Italic. Em seguida, podemos centralizar o texto na Form, usando a paleta de alinhamento, mostrada na *figura 5*. Se ela não estiver visível, você pode apresentá-la com a opção do menu View/Alignment Palette. Os botões da terceira coluna centralizam o Label na janela.

Nosso programa está pronto. Podemos executá-lo, usando a opção Run/Run do menu ou teclando F9. A compilação é praticamente instantânea e o programa já está rodando (*Figura 6*). Note que aqui não há código interpretado – tudo é compilado e um executável foi criado. Esta foi minha maior surpresa, quando, após anos de C++ e intermináveis compilações, eu executei meu primeiro programa em Delphi 1 (em 1995) – não pude acreditar na velocidade de compilação e, sem dúvida, no Kylix isto não é diferente.

## Trabalhando com Eventos

Mas nem tudo é colocar componentes na Form e configurar propriedades: devemos dar um pouco de funcionalidade ao programa, criar um pouco de código. A programação tradicional, usada comumente em modo texto (programas DOS ou Linux) é linear, onde a sequência do programa é conhecida, um passo após o outro. Eventualmente, chamamos uma sub-rotina e, após a execução desta, o programa retorna ao local onde a rotina foi chamada.

Na programação em ambiente gráfico, o programa espera algo acontecer, isto é, ele apenas reage à ação do usuário: quando ele clica um botão, a ação “A” é executada, ao teclar algo, a ação “B” é executada. Este tipo de programação é chamado de Programação Orientada a Eventos.

Em Kylix, estes acontecimentos são chamados de “eventos”. Quando um evento acontece, por exemplo, quando um botão do mouse é pressionado, o programa é avisado disto, indicando onde isso ocorreu. O programa, por sua vez, pode executar alguma rotina para processar este clique ou mesmo desprezá-lo.

O tratamento destes eventos em Kylix é muito fácil: a segunda guia do Object Inspector, Events, mostra os eventos disponíveis para o componente. Em geral, os nomes destes eventos são iniciados por On, seguido do tipo de evento. Assim, OnMouseDown é o evento que é disparado quando o usuário clica o botão do mouse e OnCreate é ativado quando um objeto é criado.

Pode-se associar uma rotina que será executada quando o evento for ativado. A esta rotina damos o nome de manipulador de eventos. Ao dar um clique duplo no espaço em branco em frente ao nome do evento, na guia Events do Object Inspector, abre-se o editor de código com o esqueleto de um manipulador de eventos como o mostrado a seguir:

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
  
end;
```

Digita-se aí a rotina que será executada quando o evento for acionado. Não é necessário criar código para todos os eventos de um objeto: quando não houver um manipulador definido para um evento, a ação padrão será executada. Para mostrar este conceito, colocaremos um botão na janela, modificando sua propriedade Caption para Sair. Ao dar um clique duplo sobre ele, abre-se o editor de código, com o manipulador do evento OnClick. O clique duplo sobre um componente ativa uma ação pré-determinada, que pode variar de componente a componente. Por exemplo, em alguns, é aberto o editor de código num evento, em outros, abre-se uma tela de configuração do componente. Devemos apenas colocar neste manipulador a linha mostrada em negrito:

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
Close;  
end;
```

Assim, quando o botão é clicado, o comando Close é executado, fechando a janela e o programa.

Como você pôde ver aqui, a programação com o Kylix é muito facilitada pelas ferramentas que oferece. Por outro lado, esta simplicidade não se traduz em programas simples, que têm pouca funcionalidade – apenas leva o desenvolvedor a se preocupar com aquilo que realmente deve se preocupar: os algoritmos de funcionamento dos seus programas. Facilitando a criação da interface com o usuário, sobra mais tempo para desenvolver programas melhores, mais otimizados e robustos.

### Vantagens da Interface do Kylix

Não é perdida a experiência adquirida pelos programadores Delphi no Windows, diminuindo a curva de aprendizado para a programação Linux

Mantém um ambiente de desenvolvimento semelhante para as duas plataformas.

O Kylix é um ambiente de trabalho completo; não precisa de nenhuma outra ferramenta para desenvolvimento de programas:

A edição de código é feita com seu próprio editor, com destaque de sintaxe (syntax highlighting), complementação de código e ferramentas para facilitar a criação de código e a navegação pelo programa.

A interface com o usuário é criada de maneira visual: o que se cria em tempo de projeto é o que o usuário verá em tempo de execução.

A compilação do código é feita usando-se o menu ou a barra de ferramentas; as mensagens de erro de compilação são mostradas em uma janela, podendo ser sincronizadas com o código fonte.

A depuração dos programas é feita com o depurador integrado, que permite parar em determinado ponto do código, executar linha a linha, ou analisar o valor das variáveis. O Kylix permite ainda mostrar o código Assembler gerado, depurando instrução a instrução.

#### **Nota:**

**O Kylix gera executáveis nativos: não são necessários programas interpretadores ou distribuição de parte do Kylix com os executáveis.**

**O Kylix permite desenvolver aplicações Linux, e não device drivers – para isto, você terá que usar o compilador gcc.**

**O Kylix também permite a criação de aplicativos não gráficos, usados em linha de comando**

**Bruno Sonnino**

[sonnino@netmogi.com.br](mailto:sonnino@netmogi.com.br)