

Uma mão para o Pascal

O Lazarus é um ambiente de desenvolvimento para o Free Pascal compatível com diversas plataformas, e que permite aos desenvolvedores criar interfaces gráficas com alguns cliques e um pouco de lógica de programação.

por **Tim Schürmann**

Adicionar uma interface de usuário simples aos seus programas em Object Pascal é entediante e leva muito tempo. Para simplificar a tarefa repetitiva de derivar os resultados de inúmeras classes, a Borland desenvolveu o Delphi em meados dos anos 1990. No ambiente de desenvolvimento Delphi, os programadores podem, com cliques, criar uma interface rapidamente. Infelizmente, o Delphi é um programa caro e só está disponível para Windows.

No entanto, um ambiente totalmente gratuito que responde pelo nome de Lazarus [1] é uma alter-

nativa para seu análogo comercial: sua base é o compilador Free Pascal, incluído na maioria das distribuições e cujo grande escopo de funções é comparável ao bom e velho Delphi em muitas áreas. Além disso, funciona em Windows e em Mac Os X.

Início

Ao ser iniciado pela primeira vez, o Lazarus bombardeia o desenvolvedor com diversas janelas, mas os programadores de Delphi se sentirão imediatamente em casa com elas (figura 1). Para iniciar, selecione o elemento gráfico necessário na paleta do topo da tela e coloque-o na caixa de diálogo *Form* (nome dado pelo Lazarus a todas as caixas de diálogos e janelas no programa resultante). Após soltar o elemento, você pode usar as bordas para redimensioná-lo, como se fosse uma ferramenta de desenho, e arrastá-lo para a posição correta. Os ajustes mais finos e as modificações de propriedades, como os rótulos de botões, são o domínio do *object inspector*.

No plano de fundo, o Lazarus gera automaticamente o código fonte, ao qual é necessário adicionar lógica de programação em um editor de janelas. O *Object Inspector* o ajudará com isso:

ele não somente lista todos os possíveis eventos que a interface gráfica entende, mas deixa que você aponte e clique para atribuir-lhes métodos existentes ou criar um método em branco.

Fundamentos

O Lazarus oferece o conjunto de recursos típico de qualquer ambiente de desenvolvimento. O gerenciador de projetos ajuda a manter o controle de projetos complexos, e a janela *Project Inspector* lhe permite controlar os arquivos e pacotes envolvidos. Para compilar um aplicativo, tudo o que se precisa fazer é apertar o botão verde de *play*. Caso necessário, pode-se atribuir configurações individuais de compilação para cada projeto.

Os erros são listados em uma janela separada. Clicar em um alerta o leva diretamente até a parte correspondente do código (figura 2). O debugger integrado o ajuda a investigar problemas e interrupções no programa nos pontos de quebra previamente definidos na janela do editor. Também é possível ter uma visão passo-a-passo de seus programas, visualizando valores e variáveis.

O editor oferece o tipo de escopo que se espera de um IDE profissional,

Quadro 1: Comércio livre

O Lazarus liga estaticamente a LCL ao aplicativo Object Pascal. Para suportar o desenvolvimento de programas comerciais, apesar dessa característica, a biblioteca de classes está disponível sob uma licença LGPL modificada. Infelizmente isso não se aplica a todas as unidades que ele contém. Antes de usar o componente em um aplicativo comercial, é uma boa ideia conferir a licença, que está embarcada no código fonte para o componente. O Lazarus, em si, não é licenciado sob GPL.

incluindo destaques de sintaxe e empacotamento do código. O recurso de autocompletar sugere possíveis métodos e nomes de classe, ajudando-o ao preencher os parâmetros. Inclusive, adiciona automaticamente o `begin` e o `end` obrigatórios ao código (figura 3).

Um atalho de teclado `lhe` permite adicionar componentes para as linhas selecionadas ou indentá-las. Você pode armazenar os blocos de código em templates. Para usar o código, pressione um atalho seguido por `[Ctrl]+[J]`, para dizer ao Lazarus que ele deve substituir todo o template. O ambiente de desenvolvimento inclui templates de códigos para construções comuns, como loops `for` e blocos `begin/end`.

Para declarar uma classe, é possível simplesmente definir os métodos e propriedades e pressionar um atalho de teclado para criar, automaticamente, um método básico vazio com os métodos `get` e `set` correspondentes. O recurso *Quick Syntax Check* descobre erros de digitação antes que você inicie a construção do bloco, e outros assistentes apontam blocos abertos de código fonte ou `IFDEF/ENDIFs`.

Orientação

O Lazarus tem uma boa seleção de ferramentas que ajudam os desenvolvedores a navegar pelo código fonte. Você pode, por exemplo, realizar buscas e substituições de forma flexível com o uso de expressões regulares. Além de buscar em todo o texto, o navegador do código também permite buscar somente pacotes, nomes de unidade e designadores para um termo específico.

A janela *CodeExplorer* ajuda a manter o controle sobre programas de códigos mais longos. Ela cria um delineador de todos os tipos, variáveis, interfaces, implementações e unidades usadas para pular para a posição correspondente no código com um único clique. O *Code Observer*, incluído no *CodeExplorer*, aponta para estilos pobres de programação, como indentação incorreta ou procedimentos excessiva-

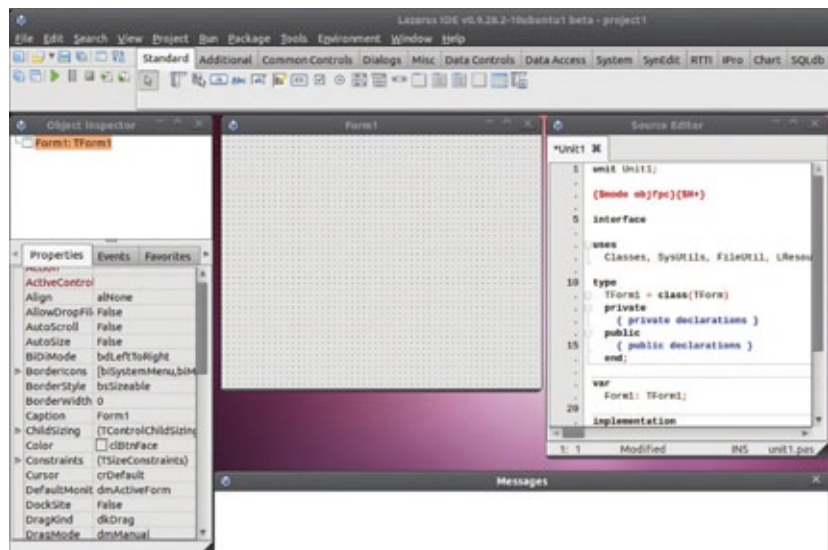


Figura 1 O Lazarus dá as boas vindas ao desenvolvedor com diversas janelas, que parecem desordenadas. O formulário vazio no centro é o aplicativo resultante.

mente longos ou vazios. Outras janelas mostram as dependências entre as unidades e os formulários criados até o momento. Um editor FPDoc ajuda a documentar o código fonte de acordo com o padrão FPDoc.

O Lazarus também suporta refatoração por meio da renomeação de designadores no projeto, extraindo o código fonte selecionado em um novo procedimento ou invertendo atribuições

(convertendo `A := B` em `B := A`). Além disso, o editor rastreia automaticamente os métodos abstratos ainda não implementados e cria um método vazio. A única coisa que falta é um link para um sistema de atribuição de versões.

Clássico moderno

Se você arrastar um botão em um formulário no Lazarus, o ambiente de desenvolvimento gera automaticamente

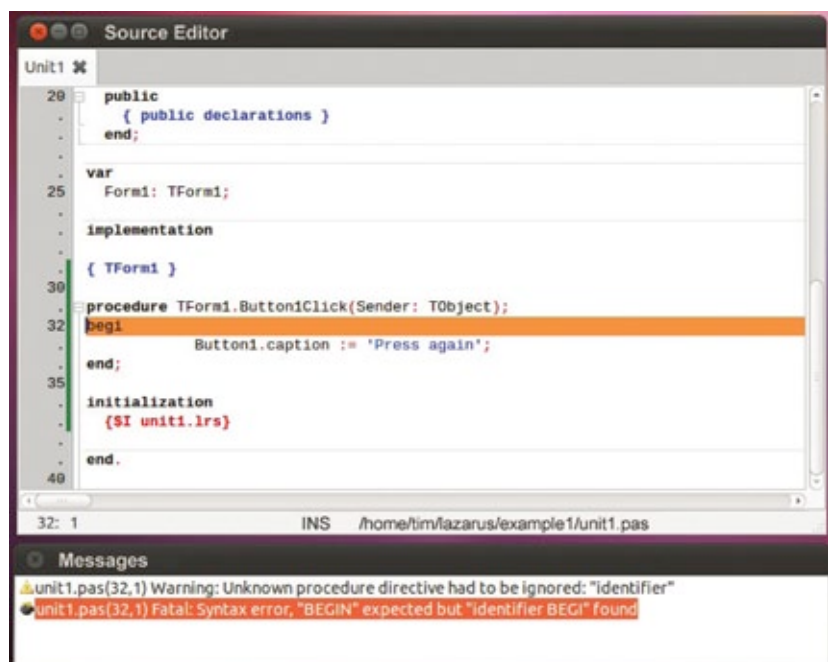


Figura 2 O debugger integrado permite que os desenvolvedores investiguem erros no código, como problemas de digitação.

um objeto de tipo `Tbutton` no plano de fundo. A classe correspondente vem como cortesia da Biblioteca de Componentes Lazarus (LCL, na abreviação em inglês). Esta consiste de uma biblioteca de classes normal que pode ser usada independentemente do Lazarus, com unidades e classes que criam a interface para seu programa em Object Pascal.

A LCL não desenha sozinha os elementos gráficos individuais na tela, mas depende de uma biblioteca legada do sistemas para isso. Os desenvolvedores podem escolher essa biblioteca no momento de construir o código. Os usuários Linux podem escolher entre Qt, Gtk+ ou Gtkz. O suporte para a o Pascal Gui Toolkit [2] está atualmente em desenvolvimento.

Além disso, a LCL está disponível para outros sistemas operacionais, suportando funções de sistemas diretamente no Windows e no Windows CE, e em Cocoa e Carbon no Mac OS X. A figura 4 mostra a estrutura da LCL e como ela interage com os componentes individuais.

Transferência de conhecimento

Graças à LCL, você pode escrever aplicativos compatíveis com várias plataformas facilmente, tendo somente que recompilá-las no sistema operacional alvo em um momento posterior (escreva uma vez, compile em todos os lugares). O Lazarus mostra como isso funciona em termos práticos: o ambiente de desenvolvimento foi programado inteiramente em LCL e pode ser reconstruído rapidamente. Também é possível usar uma biblioteca de interface gráfica diferente ao selecionar o item do menu em tempo real.

Os desenvolvedores precisam tomar cuidado com duas armadilhas: uma delas é que algumas interfaces (como o Qt) estão oficialmente em beta, embora sejam estáveis e os desenvolvedores já as utilizem para

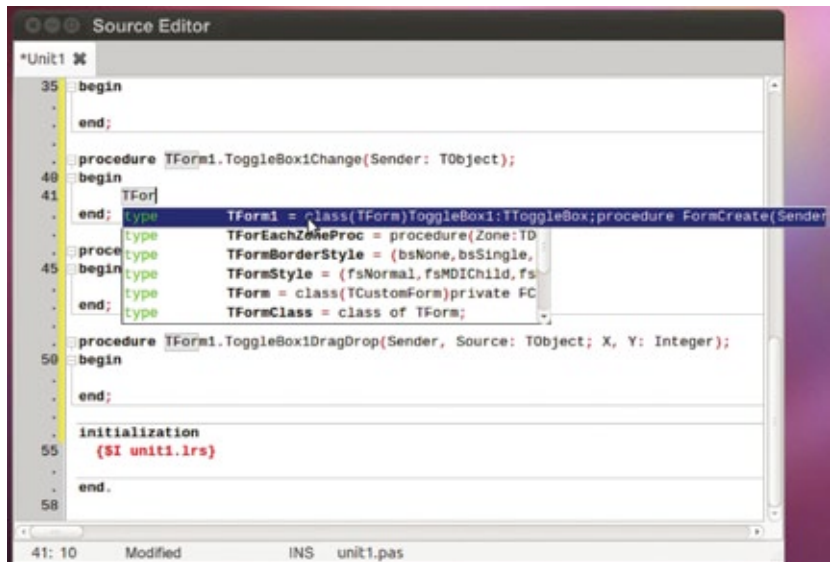


Figura 3 O editor faz sugestões enquanto você escreve.

produção. Outra diz respeito à LCL tentar seguir as linhas guias do sistema operacional alvo. Isso pode levar os componentes a agir de maneira diversa em várias plataformas. Por exemplo, com o Windows não será possível mudar a escala de caixas de diálogo com o mouse, embora se possa fazê-lo no X11. Sem contar que os sistemas operacionais lidam com atalhos de várias maneiras. O wiki do Lazarus [3] inclui dicas sobre programação para plataformas cruzadas.

A LCL faz mais do que desenhar janelas coloridas na tela. Ela inclui, por exemplo, classes para acesso simples e rápido a banco de dados, incluindo

PostgreSQL, dBase e MySQL. No Lazarus, é possível colocar componentes de conexão em banco de dados dentro de um formulário como se fosse qualquer elemento de interface, onde aparecerão como ícones. No aplicativo finalizado, eles ficam no plano de fundo e executam suas tarefas de modo transparente (figura 5).

Dois oráculos

A biblioteca de componentes visuais do Delphi (VCL) foi o modelo para a LCL. O `Tbutton` é um lembrete intencional para os desenvolvedores que tinham familiaridade com o Delphi. Apesar disso, a LCL não

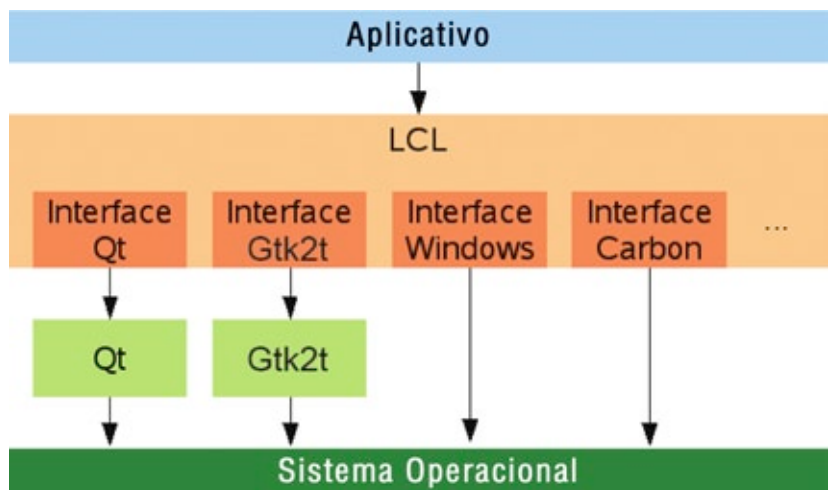


Figura 4 A LCL atua como uma interface entre o programa Object Pascal e as bibliotecas de interface para o sistema operacional suportado.

é totalmente compatível com VCL. Algumas vezes, isso ocorre por conta da independência de plataforma e é intencional. Em outras, os componentes simplesmente estão faltando.

Isso é particularmente verdadeiro no caso de classes de aplicativos multimídia, como `Tanimate`; componentes específicos do Windows, como `TmediaPlayer`; e acesso a rede, como o ADO. Assim, é impossível transferir programas Delphi e Kylix para o Lazarus sem alguma modificação. Embora o Lazarus ofereça uma série de assistentes para ajudar nessa conversão, o processo fatalmente envolverá algum trabalho manual. A wiki do Lazarus [3] oferece uma lista das diferenças e várias páginas com dicas e guias para conversões.

Conclusão

Uma vez que você se familiariza com a aparência do programa – que inicialmente parece um ambiente desordenado – o Lazarus o ajudará a criar um programa simples, independente de plataforma, em apenas alguns minu-

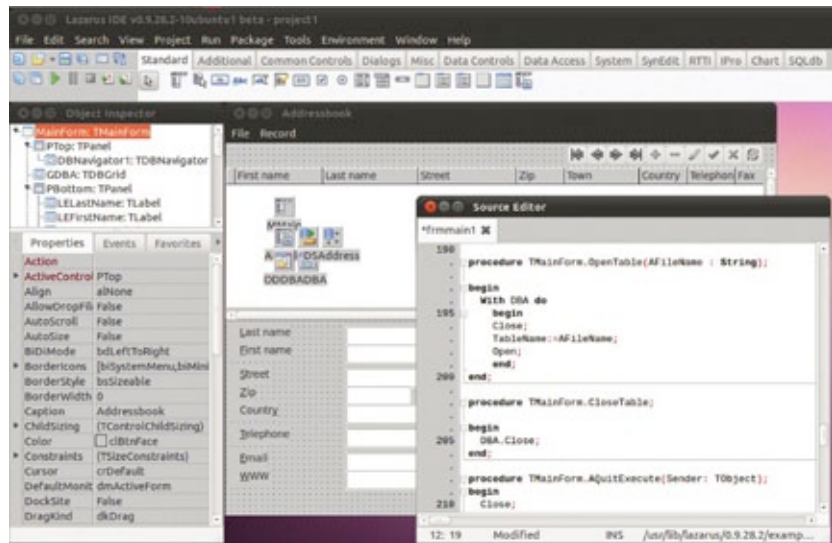


Figura 5 O Lazarus inclui diversos programas de amostra, como este gerenciador de endereços, que exemplifica a programação com banco de dados. Os ícones na área branca da janela principal são os objetos conectores ao banco de dados.

tos. Uma calculadora, por exemplo. Ao mesmo tempo, o desenvolvedor com um ambiente legado de desenvolvimento teria tempo apenas de trabalhar com a derivação de classes para os botões. O wiki abrangente do programa [3] ajuda os desenvolvedores na curva inicial de aprendizado.

As conversões do Delphi constituem um processo relativamente indolor, embora algum trabalho manual seja necessário. A recompensa é um aplicativo Object Pascal que poderá ser compilado em todos os grandes sistemas operacionais.

Apesar de mais de dez anos de desenvolvimento, o trabalho ainda está em andamento no Lazarus e na LCL, sendo que o processo se encontra no ponto zero (a versão atual é 0.9.30). No entanto, não deixe que isso o afaste do Lazarus: ele é estável e compete com o Delphi em igualdade de condições quando combinado ao Free Pascal. ■

Quadro 2: Uma jornada ao passado

Baseado no Algol 60, Niklaus Wirth desenvolveu a linguagem de programação estruturada Pascal no início dos anos 1970, principalmente com propósitos educacionais. O programa se tornou extremamente popular nos anos 1980 graças ao Turbo Pascal, da Borland. Com o sucesso, a Borland estendeu o escopo de recursos do pacote e a linguagem em si. No final dos anos 1980, a Borland adicionou o conceito de orientação a objetos e colocou os resultados no Object Pascal.

Em meados dos anos 1990, a Borland introduziu um ambiente totalmente novo com o nome de Delphi, que permitia aos programadores desenvolver de maneira conveniente e rápida ao mesmo tempo em que criavam a interface gráfica. A biblioteca de interface era a Visual Component Library (VCL), completamente desenhada para Windows. A Borland tentou entrar no mercado de Linux em 2001 com o Kylix. Tratava-se de uma interface Delphi emulada via Wine, capaz de criar programas gráficos para Linux com recursos nativos do sistema operacional, usando a biblioteca de interface CLX, uma variante do VCL. Hoje, o Delphi é desenvolvido e comercializado pela Embarcadero Technologies. O Kylix foi tirado do mercado devido à falta de sucesso.

Insatisfeito com a abordagem da Borland, o estudante Paul Klämpfl começou a trabalhar em um compilador gratuito nos anos 1990. A compatibilidade com o Turbo Pascal e Delphi, além da portabilidade fácil, rapidamente fizeram do Free Pascal um dos compiladores Pascal mais populares. O que lhe faltava era um ambiente de programação visual no estilo do Delphi. O projeto Megido tentou remediar essa situação, mas fracassou em 1999. O Lazarus surgiu dessa fraqueza – daí seu nome bíblico.

Mais informações

- [1] Lazarus: <http://www.lazarus.freepascal.org/>
- [2] fpGUI: <http://fpgui.sourceforge.net/>
- [3] Wiki do Lazarus: <http://wiki.lazarus.freepascal.org/>

Gostou do artigo?

Queremos ouvir sua opinião. Fale conosco em cartas@linuxmagazine.com.br
Este artigo no nosso site: <http://nm.com.br/article/6271>