Programação Multiplataforma Utilizando GTK+

Pistori, H.

Universidade Católica Dom Bosco, Campo Grande, Brasil
pistori@ec.ucdb.br

Resumo

Este artigo aborda o problema da construção de software para múltiplas plataformas e apresenta algumas ferramentas que podem auxiliar neste tipo de desenvolvimento. Uma destas ferramentas, o GTK+, tem se tornado um padrão para ambientes Linux, e já possui implementações também para MS-Windows e Solaris. Apresentaremos o GTK+, junto com os motivos que nos levaram a sua adoção, e um tutorial, em português, que foi desenvolvido para facilitar a introdução desta ferramenta para alunos de cursos de Computação e profissionais da área.

Palavras Chave: Software Livre, Interfaces Gráficas, Toolkits, Portabilidade.

Introdução

Nos últimos anos, o desenvolvimento de software aplicativo para uso doméstico tem sido muito influenciado pela hegemonia dos Sistemas Operacionais da Microsoft: MS-DOS/Windows 3.11, Windows95 e Windows98. A grande maioria dos aplicativos desenvolvidos para utilização em computadores pessoais são executados e desenvolvidos nesta plataforma. As poderosas estratégias de Marketing da Microsoft, como a "venda embutida" de seu software em micro-computadores, fizeram com que a hegemonia se transformasse em Monopólio. Mas como em qualquer outro ramo de negócio, o monopólio fragiliza brutalmente o mercado consumidor, que torna-se refém de uma única empresa. No mercado de Sistemas Operacionais o monopólio é ainda mais preocupante, uma vez que todos os outros programas dependem do Sistema Operacional, que dá suporte a sua execução e portabilidade entre diferentes máquinas.

O surgimento da Internet, somado ao renascimento do software de fontes abertos[2, 13, 15] e a ascensão do Sistema Operacional GNU/Linux, tem motivado diversos programadores, distribuídos pelo mundo, a se unirem pela bandeira da "liberdade de escolha" no mundo da computação. O GNU/Linux, junto com aplicativos desenvolvidos "para ele", já ocupa espaço importante no mercado de servidores e tem mostrado crescimento significativo no mercado corporativo. No mercado "doméstico", no entanto, o MS-Windows ainda não teve seu monopólio abalado, principalmente pelos seguintes motivos: a Instalação do Linux ainda é complicada para o usuário comum e a quantidade de aplicativos que "rodam" em Linux ainda é muito pequena se comparada ao MS-Windows.

O primeiro problema vem sendo atacado principalmente pelas empresas que distribuem o Linux, empacotado com outros aplicativos. Estas empresas vem melhorando cada vez mais seus programas Instaladores, através do uso de Interfaces Gráficas, auto-detecção de dispositivos e utilização do idioma do país aonde o software é distribuído. O segundo problema é mais amplo e depende de muitos fatores, principalmente, a disponibilidade de pessoas e empresas em investir no desenvolvimento de programas para esta plataforma.

Para evitar que aconteça apenas uma troca de monopólio no mercado de Sistemas Operacionais, torna-se importante que os desenvolvedores busquem a produção multiplataforma. Além disto, a produção de programas que "rodam" tanto em Linux quanto em MS-Windows, facilitará a introdução de programa de código aberto e/ou gratuitos para usuários atuais do MS-Windows, que com o tempo, poderão migrar, sem traumas, para o Linux.

Nosso trabalho visa o estudo, divulgação e construção de material de apoio a aprendizagem de ferramentas de suporte à programação de aplicativos multiplataforma. Priorizamos, no estudo das soluções disponíveis, a disponibilidade de fontes, o baixo custo, a performance e a facilidade de uso e aprendizagem. Considerando que a programação multiplataforma deve ser inserida no inicio da formação dos futuros programadores, e que nesta fase, a grande maioria dos alunos tende a priorizar a literatura e recursos escritos no idioma nacional, optamos pelo desenvolvimento de material em língua Portuguesa. Este material também será disponibilizado gratuitamente pela Internet.

A próxima seção deste artigo trata do problema da portabilidade de aplicativos. A seção 3 analisa

algumas das atuais soluções disponíveis para minimizar o problema, enquanto que a seção 4, apresenta a solução que acreditamos ser uma das mais interessantes. Terminamos com as conclusões parciais do nosso trabalho, que está ainda em desenvolvimento.

Portabilidade de Aplicativos com Interfaces Gráficas

A grande aceitação da linguagem de programação C entre os desenvolvedores de Software, deve-se principalmente a sua alta portabilidade. Compiladores C estão disponíveis em praticamente todas as plataformas, desde PCs com MS-Windows, Macs e até computadores de Grande Porte rodando as mais diversas variações do Unix. Infelizmente, esta portabilidade fica completamente comprometida quando começamos a desenvolver programas com Interfaces Gráficas.

A grande maioria dos programas com Interfaces Gráficas da atualidade utilizam o modelo WIMP[4], Windows, Ícones, Menus e Ponteiro. No entanto, a linguagem C não oferece recursos padronizados para manipulação destas Interfaces. Muitos compiladores C são distribuídos com bibliotecas de funções (e cabeçalhos) que permitem o desenvolvimento destas Interfaces, como é o caso dos compiladores C da Borland e da Microsoft, com suas bibliotecas OWL[12] e MFC[10], no entanto, estas empresas não oferecem (pelo menos até o momento) implementações destas bibliotecas para outros Sistemas Operacionais além do MS-Windows. Desta forma, seu código acaba ficando "preso" a uma implementação do Compilador C e das plataformas para os quais o fabricante do Compilador tem interesse em comercializar seu produto.

Estratégias Disponíveis

Existem diversas estratégias que podem auxiliar no desenvolvimento de programas com Interfaces Gráficas portáveis. Uma das mais difundidas é a utilização da tecnologia Java, que permite a portabilidade de "arquivos executáveis" ao preço do acréscimo de um programa Interpretador (Máquina Virtual) entre o "executável" e o Sistema Operacional. Hoje já existem implementações da máquina virtual Java para todas as principais plataformas, e se o problema principal é a portabilidade, então Java é uma excelente escolha. No entanto, para muitas aplicações, a queda no desempenho decorrente do alto nível de abstração desta tecnologia inviabiliza completamente a sua utilização.

Uma outra maneira de minimizar o problema da portabilidade é fazer com que o seu código fonte não utilize diretamente as bibliotecas específicas de cada plataforma, e sim um conjunto de funções "intermediárias" , ou uma biblioteca "intermediária", responsável por "traduzir" os pedidos do seu aplicativo para uma plataforma específica. Esta "camada de software" irá separar o seu aplicativo dos detalhes da plataforma, e poderá ser utilizado no desenvolvimento de outros aplicativos também. Algumas destas "camadas de software", criadas muitas vezes para auxiliar no desenvolvimento de um determinado produto, acabaram se difundindo, e tornando-se um produto. Este produto é chamado normalmente de *GUI ToolKit* (Ferramentas para Construção de Interfaces Gráficas para Usuário), e podem ser encontrados facilmente pela Internet, tanto na forma de Software Livre [7] quanto Proprietários, a preços que variam de 0 a 3000 reais. Uma lista bastante completa destes Toolkits, incluindo links para a página principal de cada produto, pode ser encontrada em [14]

Dentre todos estes Toolkits, concluímos que o GTK+ [3, 8, 11] é o mais apropriado para o nosso objetivo de difundir a programação multiplataforma. Classificamos abaixo alguns Toolkits, pelo principal motivo que nos levou a não utiliza-los neste projeto:

Indisponível para Linux e/ou Windows: Borland OWL, Microsoft MFC, JX, LessTif

Software Comercial: Motif, Qt

Não é completamente C ou C++: Tcl/Tk, Lua

Utilização Complicada: Athena Widget

Software não Livre: XForms

A linguagem C foi utilizada como critério de avaliação do Toolkits por ser extremamente portável (os componentes não gráficos) e muito difundida. Também acreditamos que é fundamental que o Toolkit seja distribuído na forma de Software Livre, o que facilita a continuidade do produto e a ampliação de sua abrangência (novas plataformas) pelas pessoas que venham a adota-lo como ferramenta de apoio a portabilidade.

GTK+, GLADE e Material de Apoio a Aprendizagem

O GTK+ (The Gimp ToolKit) é um software livre, distribuído gratuitamente pela Internet [3] e que tem sido usado em grande projetos, como o GIMP [5] e o GNOME [6], principalmente pela facilidade de uso e alto desempenho que pode ser atingido pelos executáveis (principalmente em ambientes Linux). Para facilitar o seu porte para outras plataformas, o GTK se divide em 3 camadas de bibliotecas.

As duas primeiras camadas, GLib e GDK, são dependentes da plataforma e oferecem tanto funções gerais (GLib), como gráficas (GDK), que são usadas pela terceira camada: GTK. A terceira camada é independente de plataforma.

Os nomes das funções e dos tipos definidos nestas bibliotecas são bastante esclarecedores, e possuem sempre um prefixo para identificar a biblioteca, como por exemplo: *gdk_draw_rectangle* (função da GDK usada para desenhar um retângulo), *gtk_window_set_title* (função da GTK usada para alterar o título de uma janela) e *g_print* (função da GLib que deve ser usada no lugar de printf para garantir portabilidade). O GTK oferece suporte para todos os principais tipos de objetos gráficos, como por exemplo, botões, menus e barras de rolagem. Objetos do tipo *caixas de posicionamento* são utilizadas para posicionar os objetos na tela, de maneira relativa, de forma que a janela pode ser livremente redimensionada pelo usuário (como em JAVA).

O GLADE[1] é uma ferramenta que permite a criação de Interfaces sem necessidade de programação direta (no estilo Delphi e Visual C), facilitando drasticamente a manutenção dos objetos gráficos (Eg: Alterar Texto do Botão ou Acrescentar novo Ítem ao Menu Principal). A descrição da interface é armazenada em um arquivo XML, o que facilita que outros programas utilizem esta descrição. Este é o caso da biblioteca *libGlade*[9], que permite que a interface de um programa seja criada em tempo de execução, a partir da descrição em XML, possibilitando assim a alteração sem necessidade de recompilação (bastando apenas editar o arquivo XML, utilizando uma ferramenta da sua escolha).

Em http://www.ec.ucdb.br/~pistori/gtk nós disponibilizamos um Tutorial, em português, para programadores com alguma experiência em C, que queiram começar a utilizar o GTK. O tutorial apresenta os conceitos essenciais da programação usando GTK, através de códigos completos (passos) simples e documentados. O primeiro passo é um programa mínimo (ver figura 1), que apenas mostra uma janela na tela, o nono passo utiliza botões e arquivos gráficos (xpm), o passo10 ensina a utilizar o GLADE para gerar a interface, o último passo (passo13), até o momento, implementa um "Balde de Tinta". Os passos foram criados e testados em ambientes Linux, mas nesta mesma página podem ser encontradas instruções para produção de executáveis, destes mesmos passos (sem alteração), para ambientes MS-Windows e Solaris. A página mostra também um Jogo de Damas, que foi portado para o GTK+ a partir de fontes escritos para a biblioteca OWL do Turbo C3.11. Este trabalho foi executado em cerca de 2 dias, e o programa que antes era executado apenas em Windows95, agora já possui versões para Linux e Solaris.

Conclusões

Ainda não existe um consenso sobre a melhor forma de abordar o problema da portabilidade de interfaces gráficas, mas o crescimento na utilização do GTK+, sua facilidade de uso e forma de licença (livre) e distribuição, apontam esta ferramenta como uma das melhores escolhas da atualidade. Muito trabalho ainda precisa ser feito em relação ao suporte para o desenvolvimento, compilação, linkedição e execução em ambientes que não utilizam X ("ambiente gráfico" padrão para ambientes UNIX), como MS-Windows, OS/2 e Mac, mas a forma com que o projeto GTK+ foi criado facilita este porte.

Mais importante do que a ferramenta, no entanto, é a cultura de produção de software portável. O usuário deve ter o direito de escolher a plataforma em que irá executar o seu aplicativo. Neste sentido, nosso tutorial para o GTK+, é um instrumento importante para divulgação destes conceitos entre alunos de cursos de Computação, principalmente dos primeiros anos, quando a maioria ainda tem bastante dificuldade com o Inglês. O formato de "passo a passo", com os fontes sendo incrementados "vagarosamente" para incluir conceitos mais complexos, é bastante apropriado para aqueles que estudam sem instrutor, e tem gerado bons resultados entre os alunos da nossa Universidade. Pretendemos continuar a aprimorar esta tutorial e mantê-lo sempre disponível, gratuitamente, pela Internet.

Referências

[1] Damon Chaplin. *http://glade.pn.org/*. Acessado em Agosto 2000.

[2] Chris Dibona (ed.) et al. *Open Sources: Voices from the Open Source Revolution*. O'Reilly, 1999.

[3] Peter Mattis et al. *http://www.gtk.org/*. Acessado em Agosto 2000.

[4] Susan Weinschenk et al. *GUI Design Essentials*. John Wiley and Sons, 1997.

[5] Free Software Foundation. *http://www. gimp. org/*. Acessado em Agosto 2000.

[6] Free Software Foundation. *http://www. gnome. org/*. Acessado em Agosto 2000.

[7] Free Software Foundation. *http://www.gnu. org/~philosophy/free-sw.html*. Acessado em Agosto 2000.

[8] Eric Harlow. *Developing Linux Application with GTK+ and GDK*. Indiana: New Riders, 1999.

[9] James Henstridge. *http://www.daa.com.au/ ~james/gnome/*. Acessado em Agosto 2000.

```
/* Programa: passo1.c

   . Programa minimo utilizando GTK+.
   . Para compila-lo execute o script chamado gera-passo1
*/

#include <gtk/gtk.h>

int main(int argc, char *argv[])
{
   // Ponteiro para um objeto que devera' ser utilizado para manipular
   // a janela principal do seu programa.
   GtkWidget * janela_principal;

   // Inicializa GTK+. Deve vir antes de qualquer chamada para outras funcoes do GTK+.
   gtk_init(&argc,&argv);

   // Cria a janela principal (ainda invisivel). O parametro GTK_WINDOW_TOPLEVEL
   // Indica que a janela que nao esta 'dentro' de nenhuma outra.
   janela_principal = gtk_window_new(GTK_WINDOW_TOPLEVEL);

   // Torna a janela visivel
   gtk_widget_show(janela_principal);

   // Chama funcao do GTK que coloca o programa em um loop de
   // espera por eventos (clique do mouse, pressionamento de teclas, ...)
   gtk_main();

   return 0;
}
```

Figura 1: Programa passo1.c

[10] Ivor Horton. *Beginning MFS Programming.* Wrox Press, 1997.

[11] Marc Watson Kurt Wall and Mark Whitis. *Linux Programming.* Indiana: New Riders, 1999.

[12] Ted Neward. *Core Owl 5.0 : Owl Internals for Advanced Programmers.* Manning Publications Company, 1997.

[13] Eric S. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary.* O'Reilly, 1999.

[14] Li-Cheng Tai. *http://www.geocities.com/SiliconValley/Vista/7184/guitool.html.* Acessado em Agosto 2000.

[15] Peter Wayner. *Free for All: How LINUX and the Free Software Movement Undercut the High-Tech Titans.* Harper Business, 2000.