

Windows x Linux: Desenvolvimento de Aplicações

Fernando Lozano

<http://www.lozano.eti.br>

fernando@lozano.eti.br

Sobre o Autor

- ❑ Consultor independente com mais de 10 anos de atuação na área de TI, em projetos de desenvolvimento de sistemas, integração de redes e tuning de bancos de dados
- ❑ Detentor de diversas certificações Microsoft, Linux, IBM e Sun
- ❑ Colaborador das revistas Java Magazine, PC Master e Revista do Linux e do site DeveloperWorks da IBM
- ❑ Palestrante nas quatro últimas edições da COMDEX-SP e outros eventos nacionais e internacionais
- ❑ Autor do livro *Java em GNU/Linux* pela ed. Alta Books
- ❑ Conselheiro do Linux Professional Institute - Brasil

Objetivos

- ❑ Apresentar ao profissional ou estudante acostumado a programar em ambiente Windows as ferramentas e possibilidades disponíveis para programação em ambiente Linux
- ❑ Indicar caminhos para o desenvolvimento de aplicações multiplataforma, como preparação para uma futura migração ao Linux ou como forma de viabilizar uma rede heterogênea

Agenda

- ❑ Evolução dos IDEs para DOS e Windows
- ❑ A Cultura Windows
- ❑ Evolução dos IDEs para Linux
- ❑ A Cultura Linux
- ❑ Desenvolvimento Web
- ❑ Desenvolvimento Multiplataforma

A Evolução dos IDEs para DOS e Windows

Desenvolvimento em DOS

- ❑ O MS-DOS original fornecia uma única ferramenta de desenvolvimento: o interpretador **BASICA**
- ❑ Desenvolvimento “profissional” era realizado com o Macro Assembler da Microsoft (MASM), que deveria ser adquirido em separado
- ❑ Versões seguintes do MS-DOS incorporaram versões melhoradas do interpretador BASIC, como o **Quick Basic**
- ❑ Com o tempo, surgiram compiladores e interpretadores para outras linguagens, como dBase, Clipper, COBOL (Microfocus), FORTRAN, Pascal e C
- ❑ Todas essas ferramentas eram baseadas na linha de comando e no ciclo editar/compilar/linkeditar/executar

Surge o IDE

- ❑ Uma empresa até então desconhecida, a **Borland**, criou o conceito de IDE (Integrated Development Environment) ao lançar o **Turbo Pascal**
- ❑ Um IDE integra as várias ferramentas de desenvolvimento (editor de programas, compilador e depurador) em uma única interface orientada a menus
- ❑ O “turbo” vinha do fato de toda a compilação e linkedição ser realizada em memória, enquanto que ferramentas anteriores (de linha de comando) utilizavam arquivos temporários em disco
- ❑ Ainda não haviam ferramentas específicas para o IBM PC – a maioria das ferramentas era desenvolvida também para o CP/M

Dificuldades do Desenvolvimento DOS

- ❑ Limite de 640Kb de RAM
- ❑ Modelo de memória segmentado do 8086
- ❑ Aritmética de 16-bits
- ❑ Ausência de mecanismos de proteção de memória
- ❑ System calls primitivas do MS-DOS
- ❑ Falta de bibliotecas padronizadas
- ❑ Resultado: as ferramentas de desenvolvimento foram sendo expandidas para fornecer bibliotecas e componentes necessários à criação de aplicações “reais”, de modo que programas escritos para o Turbo Pascal (por exemplo) não poderiam ser compilados com o Microsoft Pascal

O Windows 3.x

- ❑ O Windows existia, como um “shell” para o DOS, há muitos anos, mas nunca havia conquistado espaço no mercado
- ❑ Somente com a versão 3.0 o Windows incorporou recursos em quantidade suficiente para atrair usuários e desenvolvedores
- ❑ Foi eliminado o limite de 640Kb de RAM, mas não o modelo de memória segmentada, atualizado para o 80286
- ❑ O Windows passou a incorporar um conjunto mais rico de system calls e algumas bibliotecas padronizadas para gráficos, multimídia e impressão, facilitando o desenvolvimento de aplicações para o usuário final

Nasce o IDE Visual

- ❑ Antes do surgimento do **Visual Basic**, o desenvolvimento para Windows utilizava as mesmas ferramentas do MS-DOS, frequentemente associadas a DOS-Extenders para utilizar mais memória RAM
- ❑ Entretanto, as bibliotecas padrão do Windows eram de baixo nível, de modo que um programa simples estilo Notepad necessitava de centenas de linhas de código
- ❑ O VB incorporou bibliotecas de nível mais alto, seja por extensões da linguagem ou pelos componentes VBX
- ❑ Mais do que isso, o VB incorporou ao IDE um desenhador visual e interativo de formulários, automatizando a geração de código para a interface gráfica

Aplicações Pessoais x Sistemas de Informação

- ❑ O Windows só se tornou popular o suficiente para justificar o a criação de IDEs visuais depois que houve uma massa crítica de aplicações de produtividade pessoal (editores de texto, desenho, planilhas, etc)
- ❑ Então estas aplicações foram desenvolvidas sem IDEs visuais!
- ❑ Na verdade, como estas aplicações tem muito código (em relação as telas) e exigem componentes visuais customizados (como print previews), IDEs a la VB fazem pouca diferença
- ❑ É no desenvolvimento de sistemas de informação que os IDEs são importantes, para gerar em pouco tempo grande quantidade de telas de entrada de dados e de consulta

O Windows de 32-bits

- ❑ O Windows NT (incluindo seus sucessores 2000 e XP) é um sistema operacional completamente novo (em vez de ser uma extensão do DOS como o Windows 3.x, 95, 98 e ME) focado no mercado corporativo
- ❑ Tornou o desenvolvimento ainda mais fácil ao incorporar novos system calls para comunicação inter-processos e redes, além de abandonar o modelo de memória segmentado do 8086 e 80286
- ❑ Entretanto, não foi aceito como estação de trabalho até o lançamento do XP, o que levou à migração de suas bibliotecas para o Windows 95 e seus sucessores

OLE Controls e ActiveX

- ❑ Com a expansão da quantidade de systemcalls do sistema operacional Windows e o surgimento de outros IDEs visuais (como Delphi), fez-se necessário um mecanismo melhor para a construção de bibliotecas de componentes
- ❑ A Microsoft definiu os OLE Controls (OCX) para substituir o VBX e suportar vários IDEs da empresa (VB, MSVC, VFP), modelo adotado também pelo Delphi, PowerBuilder e outros
- ❑ Outra finalidade dos OLE Controls era permitir inserir documentos de um aplicativo (ex: gráfico) dentro de documentos de outro aplicativo (ex: editor de textos)
- ❑ Entretanto, a definição de componentes OLE foi feita em baixo nível, gerando o **DLL Hell**

A Cultura Windows

- ❑ Como o SO fornecia apenas componentes de baixo nível (e nenhuma ferramenta de apoio ao desenvolvimento), os IDEs tornaram-se “inchados” e incompatíveis entre si
- ❑ As dificuldades do padrão OCX fizeram com que cada IDE fornecesse sua própria biblioteca de componentes, de modo que o programador não aproveita quase nada do seu know-how ao mudar de IDE
- ❑ Pior, frequentemente uma nova versão do mesmo IDE é incompatível com aplicações desenvolvidas com versões anteriores do mesmo IDE (ex: VB 4/5/6, Delphi 6/7)
- ❑ O foco nos designers de formulários leva a aplicações de difícil manutenção

Evolução dos IDEs para Linux

Origens do Unix

- ❑ O sistema operacional Unix foi desenvolvido para uso interno da AT&T e fornecido gratuitamente para várias universidades e instituições de pesquisa conveniadas com a empresa
- ❑ Seu objetivo era suportar o trabalho de programadores, de modo que o Unix desde cedo fornecia várias system calls, bibliotecas padronizadas de alto nível e vários utilitários de apoio ao desenvolvimento
- ❑ Sua disponibilidade para universidades fez com que a maior parte das novas tecnologia desenvolvidas desde meados dos anos 70 TI fosse incorporada ao Unix antes de outros sistemas, em especial o BSD Unix, origem da Internet

Desenvolvimento C em Unix

- ❑ Baseado em ferramentas de linha de comando
 - ❑ Editor vi
 - ❑ Compilador C
 - ❑ Linkeditor
 - ❑ Depurador
 - ❑ Bibliotecas
 - ❑ Makefiles (gerenciador de configurações)
 - ❑ RCS (sistema de controle de versões)
- ❑ Como o desenvolvimento do próprio SO era baseado na linguagem C (e não assembler, como no DOS e Windows), naturalmente suas bibliotecas eram de utilização mais fácil pelos programadores

Aplicações Gráficas em Unix

- ❑ As bibliotecas gráficas do Unix, baseadas no X Window System, não eram de baixo nível como as bibliotecas do Windows, de modo que era viável criar aplicações sem construtores visuais de formulários
- ❑ Entretanto, não havia um único padrão de **toolkit** (biblioteca de componentes visuais) devido às disputas entre os grandes fornecedores de sistemas Unix: IBM, Sun, HP, Digital e SGI
- ❑ Aliado ao custo das máquinas RISC, o Unix não atingiu massa crítica como SO em estação de trabalho e não demandou aplicações de automação de escritório, o forte do Windows
- ❑ O desenvolvimento “rápido” era feito em scripts **TCL**

Surge o Linux

- ❑ O Linux nasceu da necessidade de um clone do Unix compatível com PCs padrão, e da tentativa da AT&T em bloquear o acesso (gratuito) das universidades ao Unix original
- ❑ Assim como o Unix, o Linux era focado em ambientes multiusuários e servidores de rede, para usuários de elevado nível técnico
- ❑ Dentro do alto grau de especialização e demandas mais rígidas de confiabilidade de servidores, bibliotecas e ferramentas de apoio eram mais importantes do que IDEs visuais
- ❑ Além disso, a necessidade de suportar múltiplas plataformas de hardware dificultava a construção de ferramentas visuais

Ferramentas GNU

- ❑ O Linux inclui o conjunto de ferramentas de desenvolvimento do GNU
 - ❑ Editores vim e Emacs
 - ❑ Compilador GCC
 - ❑ Linkeditor dinâmico
 - ❑ Depurador GDB (cli), xgdb, DDD e GVD (visuais)
 - ❑ Bibliotecas
 - ❑ Makefiles
 - ❑ CVS
 - ❑ Shell scripts
 - ❑ Perl

A Popularização do Linux

- ❑ O Linux, por rodar em computadores baratos (PCs), aliado à crescente insatisfação do mercado com a (baixa) qualidade e segurança dos softwares Windows, começou a se tornar popular também como estação pessoal, gerando a demanda por aplicações para o usuário final
- ❑ Dois ambientes gráficos se firmaram como padrão: KDE e Gnome
- ❑ Assim como no Windows, as aplicações de produtividade foram desenvolvidas em C/C++, com pouco apoio de recursos visuais
- ❑ Mas faltavam IDEs visuais para desenvolvimento rápido de sistemas de informação

IDEs Visuais e Não-Visuais

- ❑ Editores de programas tradicionais do Unix como o Emacs forneciam todos os recursos de um IDE, exceto pelo construtor visual de formulários
- ❑ Depuradores visuais já eram comuns no Unix, e com a vantagem de suportarem várias linguagens de programação diferentes!
- ❑ A popularização do Gnome e KDE levou ao surgimento de versões gráficas dos editores de programas, como o **SciTE**, **CoolEdit**, **nEdit**, **Moleskine** e vários outros
- ❑ Recursos extras, como assistentes para geração de aplicações foram incorporados a editores de programas mais sofisticados como o **KDevelop** e o **Anjuta**

A Cultura Linux/Unix

- ❑ Desenvolvedores Unix estão acostumados a encontrar um bom suporte no próprio SO para o desenvolvimento de aplicações, e valorizam a liberdade de mudar de IDE
- ❑ Desenvolvedores Unix se preocupam mais com a confiabilidade e segurança de suas aplicações
- ❑ Programas Unix são criados para realizar uma única tarefa da melhor maneira possível, e para serem facilmente integrados a outros programas, especializados em outras tarefas, para resolver um problema maior
- ❑ O Linux está intimamente ligado à Internet, que mudou o panorama das ferramentas de desenvolvimento, como veremos adiante...

Desenvolvimento Windows x Linux

- ❑ IDEs monolíticos, que tem que realizar todas as tarefas
- ❑ Aplicações estão amarradas ao IDE
- ❑ Foco em assistentes e em “não precisa ser um programador”
- ❑ Pacotes integrados
- ❑ Dependência do fornecedor
- ❑ IDEs leves, que atuam como ponto de integração para softwares de terceiros
- ❑ Aplicações amarradas apenas ao Linux (em geral, nem isso)
- ❑ Foco em automação e em “saber o que se está fazendo”
- ❑ Integrar programas de origens diferentes
- ❑ Escolha (e mudança) do fornecedor

IDEs Visuais para Linux

- ❑ **QtDesigner**, que pode ser integrado ao **KDevelop**, ambos suportando apenas C/C++ e o toolkit Qt (KDE)
- ❑ **Glade**, que pode ser integrado ao **Anjuta**, ambos suportando o toolkit GTK+ (Gnome) para desenvolvimento em C/C++, ou utilizado de forma independente para suportar Ada, Pascal, PHP, Perl, Java, Python, ...
- ❑ **PerlComposer** (Perl com GTK+)
- ❑ **Kylix** (compatível com Delphi e baseado no Qt)
- ❑ **Lazarus** (Free Pascal, baseado no GTK+)
- ❑ **NetBeans, JBuilder, JDevelop, Websphere Studio** (Java com Swing)

O Glade

- ❑ Proposta de desenvolvimento visual diferente dos IDEs do mundo Windows
- ❑ O GTK+ é independente de linguagem e de SO
- ❑ O Glade também tenta ser, apesar de oferecer recursos (opcionais) específicos para o Gnome
- ❑ Pode gerar o código das janelas em C e outras linguagens
- ❑ Mas o preferido é salvar apenas a *descrição* das janelas em formato XML, processado em tempo de execução pela biblioteca libGlade
- ❑ Permite modificar o layout das janelas sem recompilar
- ❑ Melhor suporte a idiomas orientais

Ambiente para Desenvolvimento GTK+

- ❑ Glade
(desenhador de formulários)
- ❑ Emacs, SciTE, Moleskine, Ajuta
(editor de programas)
- ❑ GCC, Java, Perl, Python, PHP
(compilador ou interpretador)
- ❑ DDD, GVD
(depurador)
- ❑ CVS
(trabalho em equipe)
- ❑ MySQL, PostgreSQL, Firebird
(Banco de Dados)

Desenvolvimento Web

A Web Muda Tudo

- ❑ Uma aplicação web (intranet, extranet, etc...) é obrigatoriamente dividida em dois componentes:
 - ❑ Páginas HTML que rodam no navegador
 - ❑ Lógica de negócios que roda no servidor
- ❑ Esta divisão, junto com a maior atenção à qualidade das páginas de modo geral (e não apenas em grandes sites) fez com que a “programação visual” perdesse importância
- ❑ Além disso, a maior demanda por segurança, confiabilidade e performance de um site web em relação a uma aplicação em PC fez o mercado perceber que o IDE visual era de pouca valia na web

Desenvolvimento Web

- ❑ Foco em oferta de conteúdo e personalização, não mais em cadastro e consulta
- ❑ Baseado em ferramentas de script de origem Unix (PHP, ASP, ColdFusion, JSP, ..)
- ❑ Multi-linguagem (HTML, JavaScript, Flash, XML, mais o script de servidor)
- ❑ Multiplataforma (Em geral, servidores web são Unix, não Windows)
- ❑ Dificuldade de integração de depuradores e outras ferramentas de IDEs tradicionais, amarradas a uma única linguagem e SO

Ambiente para Desenvolvimento Web

- ❑ Quanta, BlueFish, Mozilla
(editores HTML)
- ❑ Emacs, SciTE, Moleskine, Ajuta
(editor de programas)
- ❑ Java, Perl, Python, PHP
(compilador ou interpretador)
- ❑ Apache
(servidor web)
- ❑ CVS
(trabalho em equipe)
- ❑ MySQL, PostgreSQL, Firebird
(Banco de Dados)

O Projeto Eclipse

- ❑ Surgiu quando a IBM abriu o código das suas ferramentas de desenvolvimento
- ❑ O objetivo era viabilizar uma nova geração de IDEs, mais focada em qualidade de código, metodologias ágeis (Extreme Programming) e menos em programação visual, que fosse integrável a ferramentas para a web
- ❑ Suporte a múltiplas plataformas e linguagens de programação graças à sua arquitetura de plug-ins
- ❑ Baseado em Java, mas utilizando componentes visuais nativos (SWT) em vez do Swing
- ❑ Tornou possíveis IDEs especializados em nichos como programação embarcada

Desenvolvimento Java com o Eclipse

- ❑ Java2 SDK
(compilador e jvm)
- ❑ Eclipse
(IDE)
- ❑ JDT
(plug-ins para desenvolvimento Java)
- ❑ CVS
(trabalho em equipe)
- ❑ Plug-ins Adicionais
(X-Men, JFaceDBC, Lombok, MyEclipse, JBossIDE, WebApp, SWTDesigner, Clarion, ...)

Eclipse e Outras Linguagens

- ❑ C/C++
(CDT)
- ❑ PHP
(PHPeclipse, WebStudio)
- ❑ Python
(Xored, Pycclipse)
- ❑ Pascal
(pasclipse)
- ❑ COBOL
(Eclipse Consortium)
- ❑ ... e ainda tem mais!

Desenvolvimento Multiplataforma

- ❑ Adote uma linguagem + bibliotecas com suporte a vários SOs
 - ❑ Java
 - ❑ PHP, Python, Perl
 - ❑ GCC / Mingw
 - ❑ Free Pascal
- ❑ Utilize uma biblioteca gráfica multiplataforma, ou dê preferência ao desenvolvimento web
 - ❑ SWT, Fltk, wxWindows
 - ❑ GTK+ (livre), Qt (proprietário em Windows e Mac)

Desenvolvimento Multiplataforma

- ❑ Adote um IDE ou editor de programas multiplataforma
- ❑ Eclipse, SciTE
- ❑ A maioria dos bancos de dados já é multiplataforma
 - ❑ Oracle, DB2, Sybase, Interbase
 - ❑ MySQL, PostgreSQL, Firebird
- ❑ Complemente com ferramentas de produtividade multiplataforma
 - ❑ OpenOffice, Mozilla, Gimp
- ❑ Faça uso de ferramentas de apoio ao desenvolvimento
 - ❑ ArgoUML, CVS, Bugzilla, Ant, Xplanner,

FIM