Distribuição GoboLinux

André Detsch, Guilherme B. Bedin

[1] GoboLinux.org

{detsch,gbedin}@gobolinux.org

Resumo: Este artigo descreve a evolução de uma proposta de uma hieraquia de diretórios inovadora, diferente da tradicionalmente utilizada em sistemas UNIX, para uma distribuição GNU/Linux completa, denominada GoboLinux. São apresentadas as características de uso da distribuição, bem como as facilidades proporcionadas pelo GoboLinux.

Abstract: This paper describes the evolution of a proposal for a innovative directory hierarchy, different than that traditionally used in UNIX systems, into a complete GNU/Linux distribution, called GoboLinux. The main peculiar characteristics of the distribution are presented, as well as the facilities provided by GoboLinux.

1 Introdução

O GoboLinux ([10]) é uma distribuição Linux cuja principal característica é o uso da hierarquia de diretórios de mesmo nome apresentada em [2]. A idéia inicial do projeto era apenas estabelecer uma nova hierarquia de diretórios que facilitasse o gerenciamento e o compartilhamento de programas compilados entre alguns usuários. Entretanto a aceitação dos conceitos apresentados foi tão boa que levou ao aperfeiçoamento das idéias e criação de facilidades de instalação, para que mais pessoas pudessem utilizá-las.

Ao longo de pouco mais de um ano, a distribuição GoboLinux evoluiu significativamente. Entretanto, tomou-se o cuidado quanto a divulgação da distribuição: esperou-se que a distribuição e a hierarquia atingissem um grau de maturidade que possibilitasse seu uso por um grande número de pessoas. Este anúncio em escala mundial foi cautelosamente postergado, mas o início de discussões mais intensas sobre uma estrutura de diretório para o Linux que fosse mais adaptada os uso em *desktop* ([6]), nos levou a trabalhar mais intensamente de forma a enfim divulgar em grande escala o projeto ([5, 7]).

Este artigo, mais do que um relato de um ano de projeto GoboLinux, estabelece um marco entre uma idéia divulgada regionalmente e uma iniciativa capaz de influenciar mundialmente a utilização do sistema Linux. A Seção 2 descreve a hierarquia GoboLinux, lembrando os conceitos básicos de sua estrutura. Em seguida, são discutidos aspectos da distribuição GoboLinux (Seção 3). Por fim, a Seção 4 conclui o artigo, apresentado algumas metas e idéias ainda não implementadas.

2 Hierarquia GoboLinux

O objetivo da hierarquia GoboLinux é reestruturar a estrutura de diretórios Unix, para facilitar o gerenciamento de programas e entendimento do sistema, sem quebrar a compatibilidade com a árvore tradicional. Para isso, são utilizados três componentes básicos: o diretório `/Programs`, o diretório `/System/Links` e um conjunto de links para compatibilidade com a árvore UNIX tradicional.

O diretório `/Programs` contém um subdiretório para cada programa instalado (*Gimp*, *FileUtils*, *XFree86*, *Qt*, etc.). Cada um destes subdiretórios possui, por sua vez, um subdiretório para cada versão do programa em questão assim como um link `Current` que aponta para o subdiretório contendo a versão em uso. Os programas ficam contidos, na sua totalidade, dentro do subdiretório da versão correspondente, onde tipicamente são criados subdiretórios `bin`, `lib`, `sbin`, `share`, `man` e `info` que contém os arquivos que, na estrutura tradicional de diretórios UNIX, seriam copiados para `/usr/bin` (ou `/bin`), `/usr/lib` (ou `/lib`) e assim por diante. Cada programa possui também um diretório Settings que armazena os arquivos de configuração referentes ao programa (arquivos que estariam tipicamente localizados no `/etc`). Note que este diretório é único para todas as versões do programa, de forma que sejam mantidas as configurações personalizadas do aplicativo em caso de upgrade ou downgrade.

O diretório `/System/Links` concentra links referentes à cada categoria de arquivo instalado dentro dos diretórios das versões dos programas. O `/System/Links/Executables`, por exemplo, contém links para todos os arquivos de todos os subdiretórios bin e sbin (executáveis) dentro de cada verão de cada programa instalado. Processo similar ocorre com as bibliotecas, headers manuais e arquivos compartilhados. Conforme será explicado na Seção 3, a gerência destes links é realizada de maneira totalmente automática.

A compatibilidade com o legado UNIX é obtida através da criação de links extras, como por exemplo, `/etc -> /System/Settings`, `/bin->System/Links/Executables` e `/lib -> /System/Links/Libraries`, que espelham a árvore GoboLinux na árvore UNIX. Estes links são necessários devido à existência de caminhos hardcoded em scripts, bibliotecas, executáveis e arquivos de configuração em geral ([3]). Entretanto, estes caminhos podem ser facilmente ocultados utilizando a ferramenta *GoboHide* ([9]).

3 Distribuição GoboLinux

O projeto GoboLinux teve seu início quando, baseando-se nos passos descritos pelo Linux From Scratch ([1]), compilou-se um sistema base utilizando uma nova estrutura de diretórios. Esta estrutura já vinha sendo discutida e parcialmente utilizada a alguns meses, durante os quais, sobre uma distribuição tradicional, novos programas iam sendo gradativamente compilados e estruturados de acordo com a nova estrutura de diretórios ([4]). Uma vez compilado um sistema totalmente estruturado dentro das nossas idéias, publicou-se um artigo sobre o assunto ([2]) e várias pessoas demonstraram interesse. Com o passar do tempo e a cada novo usuário, o GoboLinux passou a ter, gradativamente, toda a funcionalidade que faz com que, hoje, possa ser de fato considerada uma distribuição Linux completa. Esta seção apresenta algumas de suas principais características.

Gerenciamento de Pacotes

Conforme descrito na Seção anterior, cada versão de cada programa possui o seu próprio diretório. Isto possibilita que a gerência de pacotes binários se dê de forma extremamente simples e sem depender de banco de dados. O script "`CreatePackage`", responsável pela criação do pacote a partir de um programa instalado, basicamente compacta o diretório da versão indicada, criando um arquivo cujo nome contém o nome do programa, a versão e a arquitetura para qual foi compilado [1] (por exemplo, `Qt--3.1--i686.tar.bz2`). Porém, antes de realizar a compactação, o script identifica, com o auxílio do `ldd`, as dependências do pacote, gerando um arquivo "`.dependencies`" que lista de quais programas (e respectivas versões) o novo pacote depende.

A instalação de um pacote se dá através do script "`InstallPackage`", que descompacta o pacote dentro do diretório `/Programs` e chama, internamente, o script "`LinkProgram`",

[1] Atualmente, sempre i686.

que cria os links referentes ao pacote dentro da estrutura `/System/Links`. Após, são identificadas as dependências pendentes, ou seja, quais programas do "`.dependencies`" não está instalado no sistema, informando o usuário sobre estas pendências. Caso exista um pacote referente a alguma dependência não satisfeita em algum diretório de uma lista pré-definida (contendo, entre outros, o diretório de pacotes do CD de instalação), o usuário é informado, e pode proceder com a instalação da dependência sem precisar de nenhum comando adicional.

A desintalação de pacotes é realizada pela simples remoção do diretório correspondente (por exemplo, `rm -rf /Programs/Qt/3.0.5`). Como resultado desta remoção, surgirão links quebrados dentro da estrutura `/System/Links`. Sua existência não compromete o funcionamento do sistema, mas são removidos automaticamente pelo script "`RemoveBrokenLinks`".

Compilação de Programas

A compilação de novos programas é automatizada através do script "`CompileProgram`". No caso de programas com arquivo de configuração de instalação (*configure*) gerado a partir do *GNU autoconf* (grande maioria dos softwares livres) o script utiliza o parâmetro `--prefix` para definir o destino dos arquivos sendo instalados. Por exemplo, no caso de se instalar o *Qt* 3.1, o script executará o comando `configure --prefix=/Programs/Qt/3.1/`. No caso de programas cuja instalação não permite a passagem deste parâmetro, o *configure* (ou o próprio *Makefile*) deve ser ajustado, automaticamente, através de scripts que buscam substituir os caminhos contidos no arquivo ou mesmo, em alguns casos específicos, manualmente. Uma vez criados (ou ajustados) os *Makefile*s, o CompileProgram prossegue a instalação rodando make (que compila os fontes) e make install (que copia/instala os arquivos compilados para o destino especificado). A seguir, é executado o script "`LinkProgram`", que cria os links na estrutura `/System/Links`.

Processo de Instalação

Inicialmente, a instalação do GoboLinux se dava através da copia do sistema base de uma máquina para outra. A partir do momento em que a base de usuários se tornou maior, foi criado um CD de instalação, que torna a instalação do GoboLinux uma tarefa bastante simples.

O CD de instalação é bootável, e o sistema executado é o próprio GoboLinux. O arquivo `/Depot/Docs/INSTALL` guia o processo de instalação, que é composto de algumas etapas bastante simples, citadas a seguir. Após iniciar o sistema, o usuário deve preparar uma partição para instalar o sistema e montá-la. A seguir, deve ser executado o script "`BaseInstall`" passando como parâmetro o ponto de montagem escolhido. Após esta execução, um sistema básico estará instalado, sendo necessário apenas ajustar os arquivos de configuração fstab, que descreve os drives e pontos de montagem utilizados pelo sistema, e o arquivo Options, arquivo de configuração dos bootscripts do GoboLinux (ajustando configurações de rede e teclado, por exemplo). Uma vez reinicializada a máquina, pode-se instalar pacotes adicionais disponíveis no próprio CD de instalação.

Sistema de Boot

O GoboLinux possui seu próprio sistema de configuração de *boot* (pacote `BootScripts`), onde os dois principais scripts fazem a maior parte de trabalho, `Init` e `Done`. Para lidar com os diferentes runlevels existentes, ainda existem os scripts `Multi` e `Single`. Estes arquivos são simplemente seqüências de comandos, prefixados pela palavra Exec e uma mesagem de texto. Por exemplo, uma parte do `Init`:

```
Exec "Setting clock..." SetClock
Exec "Loading keymap..." loadkeys "$KeymapLayout"
```

Tarefas mais sofisticadas, são definidas como funcões do shell no arquivo `Tasks`. Estas funções também podem ser chamadas pela linha-de-comando usando o script "`RunTask`". As configurações são definidas através de variáveis de ambiente no arquivo `Options`.

Documentação

Por se tratar de uma distribuição não-convencional, a documentação do GoboLinux é uma parte importante do projeto. A página do GoboLinux ([10]) possui, além de uma lista de discussão [2], FAQs e alguns HOW-TOs. Está em fase de desenvolvimento um manual completo sobre as funcionalidades do GoboLinux, com ênfase na utilização de seus scripts, que pode ser encontrado em [8].

4 Conclusões e Projetos Futuros

O GoboLinux é um projeto em franca expansão. Após a divulgação em sites mundialmente conhecidos, as discussões sobre o tema e a procura por informações tem aumentado bastante [3]. Ao falar de projetos futuros, pode-se dividir o projeto em duas partes: O GoboLinux enquanto estrutura de diretórios alternativa e o GoboLinux enquanto distribuição.

A estrutura de ditetórios GoboLinux pode ser utilizada em conjunto com distribuições tradicionais, aproveitando o suporte e a disseminação das mesmas. Está em fase de implementação um *wrapper* de chamadas para o *kernel* que captura, durante a instalação de algum programa, tentativas de gravação em diretórios como `/usr/lib` e as adapta para o caminho adequado na estrutura GoboLinux. Este mecanismo, potencialmente, irá facilitar a adaptação de distribuições para a nova hierarquia.

Já a distribuição GoboLinux, além de servir de referência para a nova hierarquia, também introduz outras características peculiares, como o sistema de boot. Para o futuro, tais inovações irão continuar, de forma a adaptar, cada vez mais, o Linux à realidade de uso em computadores *desktop*.

Referências

[1] Gerard Beekmans. "*Linux From Scratch*", `http://www.linuxfromscratch.org`

[2] Hisham H. Muhammad, André Detsch, "*Uma nova proposta para a árvore de diretório Unix*", Workshop de Software Livre 2002, Porto Alegre, Maio 2002.

[3] Hisham H. Muhammad, Rafael Guterres Jeffman, "*Portabilidade e flexibilidade em software livre: a experiência do GoboLinux*", `http://gobolinux.org/documentation/wsl2003/`

[4] Hisham. H. Muhammad, *LodeLinux*, `http://www.cscience.org/~lode/computer/lodelinux.php`

[5] Hisham H. Muhammad, "*The Unix tree rethought: an introduction to GoboLinux*", `http://www.kuro5hin.org/story/2003/5/9/05015/62649`, `http://www.osnews.com/comment.php?news_id=3511`

[6] "*If I had my own Distro...*", `http://slashdot.org/article.pl?sid=03/04/29/1958212&mode=thread`

[7] "*GoboLinux Rethinks The Linux Filesystems*", `http://developers.slashdot.org/article.pl?sid=03/05/10/1636245`

[8] Leandro Motta Barros, "*The GoboLinux Magnificent Book*", `http://lmbarros.tripod.com/GoboLinux/GoboLMB-020030202.pdf`

[9] Lucas Correia Villa Real, Felipe W. Damasio, "*GoboHide*", `http://gobolinux.org/documentation/wsl2003/`

[10] Projeto GoboLinux, `http://www.gobolinux.org`

[2] `http://gobolinux.org/mailman/listinfo/gobo-l`

[3] Aos revisores: a publicação dos artigos em escala mundial foi iniciada em 09/05. Em apenas seis horas de anúncio no www.slashdot.org, foram registrados mais de 10.000 acessos (de 82 países diferentes) ao nosso site. A versão revisada do artigo incluirá uma breve análise da repercussão.