UMA NOVA PROPOSTA PARA A ÁRVORE DE DIRETÓRIOS UNIX

Hisham Muhammad André Detsch

Programa de Pós-Graduação em Computação Aplicada - PIPCA
Centro de Ciências Exatas e Tecnológicas
UNISINOS - Universidade do Vale do Rio dos Sinos
São Leopoldo - RS - Brasil
{hisham,detsch}@exatas.unisinos.br

Resumo. Este artigo propõe a hierarquia de diretórios GOBOLINUX como nova árvore para sistemas operacionais baseados no UNIX, em especial o Linux. Esta nova abordagem tem como principais vantagens a melhor organização funcional e facilidade de gerência e instalação de software a partir do código fonte, uma vez que a estrutura de pacotes é explicitada na própria árvore de diretórios.

Abstract. This work proposes the GOBOLINUX directory hierarchy, as a new tree for UNIX-based operating systems, such as Linux. Highlights of this new approach are a greater functional organization and takes advantage of installation from source code to provide a improved software management, making the structure of packages explicit in the directory tree.

1 Introdução

O sistema operacional UNIX surgiu em ambientes onde os usuários acessavam um servidor central de aplicações através de estações com capacidade de armazenamento reduzido (ou mesmo inexistente). Muitas das características deste sistema, entre elas a estrutura de armazenamento de dados em forma de árvore de diretórios, refletem isto. O modelo de armazenamento em árvore mostra-se adequado até hoje; entretanto, a lógica por trás da hierarquia de diretórios do UNIX baseia-se em premissas que hoje em dia não correspondem mais à realidade da grande maioria das instalações Linux existentes. O sistema Linux cada vez mais é instalado em estações pessoais que armazenam e executam as aplicações do usuário. Neste contexto, não há um "servidor de aplicações", e ainda assim persistem convenções como a existência de diferentes repositórios para bibliotecas (`/lib`, `/usr/lib` e `/usr/local/lib`).

Em virtude do ritmo rápido de desenvolvimento do software livre (*"release early, release often"* [3]), o processo de instalação e remoção de programas tornou-se comum e freqüente, diferentemente do que ocorria quando do estabelecimento da hierarquia de diretórios do UNIX, na qual as distribuições do Linux usualmente se baseiam. Os critérios de organização da árvore de diretórios tradicional não levam em conta estas necessidades; torna-se interessante realizar uma revisão destes critérios e buscar uma alternativa.

Este artigo apresenta uma árvore de diretórios idealizada a partir das necessidades e características de uso dos sistemas Linux atuais e compatível com o legado UNIX. Inicialmente são discutidos aspectos da hierarquia clássica (Seção 2). A seguir, é apresentada uma visão geral das abordagens empregadas nas árvores de diretórios de outros sistemas (Seção 3). Na Seção 4 é descrita a hierarquia concebida, enquanto na Seção 5 relatam-se experiências relativas ao uso deste modelo. Finalmente, a Seção 6 conclui o artigo.

2 Características da hierarquia atual

Na árvore UNIX os diretórios servem a dois propósitos: diferenciar categorias de arquivos e diferenciar a sua localização na rede. Os arquivos da categoria "executável" de todas as aplicações são armazenados em seis diretórios: `/bin`, `/usr/bin`, `/usr/local/bin`, `/sbin/`, `/usr/sbin/` e `/usr/local/sbin`[1], onde o critério de escolha

[1] Arquivos do X Window System são, historicamente uma exceção a esta regra, possuindo toda uma hierarquia *Unix-like* sob `/usr/X11R6/`

é a localização física (local ou remota). Alguns programas, todavia, acabam sendo instalados em localizações diferentes dos critérios acima listados por questões de compatibilidade, fazendo com que a especificação da árvore UNIX possua uma série de detalhes e exceções. Por exemplo, existe uma lista arbitrária de quais executáveis devem estar presentes no diretório `/bin`, independentemente dos critérios apontados.

Alguns programas assumem que determinados arquivos estejam em diretórios específicos (por exemplo, `/lib/cpp`, `/usr/bin/python`). Isto é uma fonte de incompatibilidades mesmo entre diferentes distribuições do Linux que seguem o modelo tradicional de diretórios. Mas o maior problema causado por esta abordagem é a dificuldade na remoção de programas, uma vez que arquivos de diferentes aplicações ficam "misturados" no mesmo diretório e diferentes arquivos de uma mesma aplicação ficam espalhados por diferentes diretório.

A maneira encontrada pelas empresas e grupos que produzem distribuições do Linux para manter uma correspondência entre os arquivos e as aplicações das quais são parte é através do conceito de *"gerenciamento de pacotes"*, que consiste em realizar as instalações e remoções de software através de um programa que mantém uma base de dados que relaciona os arquivos existentes no sistema aos aplicativos de que fazem parte. A principal limitação deste método é o fato de que a instalação de aplicativos a partir da compilação do código-fonte provoca inconsistência na base de dados.

Uma prática comum atualmente é procurar manter a hierarquia `/usr` mantida pelo gerenciador de pacotes e instalar os programas compilados a partir do código-fonte na hierarquia `/usr/local`. Isto mantém a base de dados consistente mas não resolve a questão de como remover programas compilados a partir do fonte.

O padrão de hierarquia UNIX define ainda um diretório extra, `/opt`, para a instalação de grupos de aplicativos de forma separada do resto do sistema, o que pode ser considerado um reconhecimento da existência dos problemas enumerados acima. Ainda assim, isto causa um conflito de critérios no próprio padrão. Ao instalar localmente um aplicativo qualquer, um conjunto de critérios ditaria a sua localização no `/opt/`*`aplicativo`*`/bin`, e segundo outros critérios, no `/usr/local/bin`, dependendo de como é interpretado o padrão.

3 Abordagens alternativas

Praticamente todos os sistemas operacionais desenvolvidos após o UNIX utilizam o modelo de árvore de diretórios. As diferentes organizações aplicadas para a hierarquia de diretórios dos sistemas refletem as mudanças tanto na forma da qual os computadores são utilizados como na capacidade de armazenamento destes. Abaixo são descritas as árvores de diretórios presentes no MAC OS X e AtheOS, sistemas desktop que possuem certo grau de herança UNIX.

A adoção de um kernel e ferramentas baseadas, respectivamente, no Mach 3.0 e FreeBSD trouxe à Apple Computer o desafio de combinar uma hierarquia UNIX à interface e lógica de funcionamento familiar aos usuários do Mac OS. O Mac OS X ([1]) utiliza uma estratégia incomum para atingir este resultado. Na interface gráfica é exibida uma árvore de diretórios Macintosh, contendo diretórios como `/System/Library` e `/Network/`*`Estacao_Remota`*. Na verdade, estes diretórios formam um subconjunto da verdadeira árvore de diretórios. Além disso, a interface exibe alguns diretórios em localizações diferentes da real. Por exemplo, o diretório `/Mac OS X` é um link para o diretório raiz, e alguns diretórios, como `/Applications`, aparecem na interface apenas como `/Mac OS X/Applications`. Acessando-se o sistema de arquivos através de um shell, tornam-se acessíveis diretórios UNIX como `/usr` e `/etc`. Note que esta abordagem é somente possível em um ambiente proprietário, onde toda a interface primária do sistema é desenvolvida por apenas uma empresa. Em um sistema como o Linux, tal abordagem seria impossível, devido à heterogeneidade das interfaces gráficas existentes.

O sistema operacional AtheOS ([5]) utiliza uma hierarquia parcialmente baseada na árvore UNIX. No AtheOS, por exemplo, o diretório `/usr` é usado com a finalidade que, no UNIX, é a do o `/opt`. Um aspecto de design interessante aplicado por este sistema é a possibilidade de re-localização de um aplicativo após instalado. Isto é possibilitado pelo diretório ^, reconhecido pela API do sistema como sendo "o diretório onde se localiza o executável", analogamente ao ~, que representa "o diretório *home* do usuário". Infelizmente, o Linux, por ser um clone do UNIX, não pode utilizar soluções como a apresentada acima, que trariam problemas grandes de portabilidade (uma aplicação Linux sem interface gráfica pode ser facilmente portada para o AtheOS, mas a recíproca não é verdadeira).

Existem programas que tentam prover alternativas, ainda que incompletas, para gerar um certo grau de reorganização da árvore de diretórios. Dois dos mais utilizados são o *GNU Stow* ([4]) e o *Encap* ([2]). Ambos seguem a idéia básica apresentada pelo software *Depot* ([7]), desenvolvido em Carnegie Mellon. O princípio é manter duas árvores de diretórios: uma real, onde os arquivos são organizados por aplicativos; e outra organizada da forma tradicional, contendo links para os arquivos localizados na primeira árvore[2]. O *GNU Stow* pretende ser uma alternativa simplificada ao *Depot*, uma vez que, ao contrário do *Depot*, não requer uma base de dados. Ao utilizar o *Stow*, deve-se compilar a aplicação com os caminhos baseados em `/usr/local` e instalá-la em relação a `/usr/local/stow/aplicativo`. O Encap utiliza uma sistemática bastante parecida, adicionando alguns recursos, como um suporte rudimentar a controle de versões (o programa gerenciador de pacotes, *epkg*, tenta detectar versões através do nome do aplicativo criado em `/usr/local/encap`, por exemplo, `sed-2.0` e `sed-3.0.2`).

4 Hierarquia GOBOLINUX

A idéia básica por trás da hierarquia GOBOLINUX é combinar idéias dos outros sistemas apresentados com o sistema de links introduzido pelo *Depot*, criando uma nova hierarquia que mantém compatibilidade total com a árvore UNIX. Assim como no *Depot* as versões dos programas[3] (*Gimp*, *Fileutils*, *GLibC*, *Qt*, etc.) são instaladas, na sua totalidade, dentro de um diretório próprio. Dentro deste diretório tipicamente são criados subdiretórios `bin`, `sbin`, `lib`, `man` e `info` que contém os arquivos que, na estrutura tradicional de diretórios UNIX, seriam copiados para `/usr/bin` (ou `/bin`), `/usr/sbin` (ou `/sbin`), `/usr/lib` (ou `/lib`) e assim por diante.

Para tanto, foi estabelecido um diretório `/Programs` que contém um subdiretório para cada programa instalado. 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. 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. Esta característica facilita o controle de versão, uma vez que são mantidas as configurações personalizadas do aplicativo em caso de upgrade ou downgrade.

A instalação de programas a partir dos fontes é feita através de scripts. 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* 2.3.2, será executado automaticamente o comando `configure --prefix=/Programs/Qt/2.3.2/`. 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 instalada uma (nova) versão de algum programa, são criados (automaticamente, via scripts) links para seus arquivos em diretórios que centralizam estes links de acordo com o tipo. Por exemplo, o diretório `/System/Links/Executables` armazena links para os arquivos executáveis de todos os programas (contidos nos subdiretórios `bin` e `sbin`). Desta forma, todos os executáveis podem ser chamados (ou acessados) a partir de um único diretório (o mesmo ocorrendo com as bibliotecas, headers e info / man pages). Esta abordagem se difere das adotadas nos sistemas de gerenciamento de links simbólicos discutidos na Seção 3 a partir do momento em que a estrutura de links gerada não reflete a hierarquia UNIX, mas sim uma divisão funcional dos arquivos.

A figura 1 descreve com maiores detalhes a estrutura geral da hierarquia de diretórios proposta. Uma característica importante desta estrutura é a inexistência de um diretório `share` global (apesar dos programas poderem ter seus próprios diretórios `share`). Esta decisão se explica pelo fato de que, apesar deste diretório possibilitar o compartilhamento de dados entre diferentes programas, na prática este diretório acaba servindo apenas como repositório de arquivos específicos da aplicação que não têm um lugar próprio na hierarquia UNIX (por exemplo, ícones e fontes). Desta forma optou-se por não criar um diretório `/System/Links/Shared` pois os arquivos dos diferentes `share` não têm relação entre si.

A compatibilidade com o legado UNIX é obtida através da criação de links extras não presentes no diagrama acima, como por exemplo, `/etc -> /System/Settings`, `/bin->System/Links/Executables` e `/lib -> /System/Links/Libraries`, que espelham a árvore GOBOLINUX na árvore UNIX. Diferentemente das propostas anteriores que visaram organizar a hierarquia de diretórios mantendo a compatibilidade his-

[2] O sistema operacional AtheOS também utiliza uma técnica similar para manter compatibilidade com aplicações UNIX.

[3] define-se programa como sendo um grupo de aplicativos ou bibliotecas

```
/                          Diretório raiz
|--Programs                Programas
|--Mount                   Ponto de montagem de sistemas de arquivos locais ou remotos
|--Users                   Áreas dos usuários
|--System
|    |--Boot               Arquivos necessários para o boot (kernel e bootloader)
|    |--Links
|    |    |--Executables   Links para os arquivos dos subdiretórios bin e sbin dos programas
|    |    |--Headers       Links para os arquivos do subdiretório include dos programas
|    |    |--Libraries     Links para os arquivos do subdiretório lib dos programas
|    |    \--Manuals
|    |         |--info     Links para os arquivos do subdiretório info dos programas
|    |         \--man{1-9} Links para os arquivos dos subdiretórios man/man{1-9} dos programas
|    |--Settings           Arquivos de configuração e link para os arquivos dos subdir. Settings
|    \--Variable           Dados Variáveis
|         \--Temp          Arquivos Temporários
|--proc                    Arquivos de Status do Kernel (gerenciado pelo proc file system)
\--dev                     Arquivos de Dispositivos (gerenciado pelo dev file system)
```

Figura 1: Hierarquia de diretórios GOBOLINUX

tórica, no GOBOLINUX há apenas um ponto de instalação de programas, sem manter paralelamente uma árvore *legacy*.

5 Experiência

A experiência prática com a estrutura de diretórios apresentada neste artigo pode ser dividida em duas partes. Em um primeiro momento, partiu-se de uma distribuição baseada em pacotes e, gradativamente, as atualizações de programas eram feitas seguindo-se a nova estrutura, desinstalando o pacote referente à versão anterior do mesmo. Nesta etapa, amadureceu-se as idéias estruturais e verificou-se a validade e viabilidade das mesmas.

Já numa segunda etapa, optou-se por compilar todo o sistema seguindo-se a hierarquia GOBOLINUX, com o intuito de se ter um total controle de todos os arquivos instalados na máquina. Para isso, foi utilizada como base a documentação criada pelo projeto “Linux From Scratch” ([6]).

6 Conclusões

Conforme apresentado na Seção 3, a busca de alternativas para reorganizar a árvore de diretórios UNIX é asssunto de uma série de projetos. Este artigo apresentou uma nova proposta de estruturação de diretórios UNIX, que difere dos projetos anteriores principalmente no fato de não se manter, paralelo à arvore de diretórios principal, uma árvore *legacy*, possibilitando uma maior auto-consistência e elegância. A experiência prática mostrou a total viabilidade e compatilbilidade das idéias apresentadas, além de evidenciar as facilidades introduzidas pela nova estrutura.

Maiores informações sobre o GOBOLINUX podem ser obtidas em `http://cscience.org/~gobo/gobolinux.html`.

Referências

[1] Apple Computer Inc., *“Inside Mac OS X - System Overview”*, ISBN: 1400524806, 292p., Fev. 2001.

[2] *“Encap Archive”*, Computing and Communications Services Office at the University of Illinois at Urbana-Champaign, `http://www.encap.org`, 2002.

[3] Eric Raymond, *“The Cathedral & The Bazaar”*, O’Reilly and Associates, hardback edition, Jan. 2001.

[4] Guillaume Morin, Bob Glickstein, *“GNU Stow”*, `http://www.gnu.org/software/stow/stow.html`, 2001.

[5] Kurt Skauen, *“AtheOS”*, `http://www.atheos.cx`, 2002.

[6] Gerard Beekmans, *“Linux From Scratch”*, `http://www.linuxfromscratch.org/`, acessado em 17/03/2002

[7] Wallace Colyer, Walter Wong, *“Depot: A Tool for Managing Software Environments”*, Usenix LISA VI Conference, 1992.