Automatizando e Padronizando Instalações Linux Usando Anaconda e Kickstart

FRANCISCO KEM ITI SAITO[1]

[1]4Linux - R. Teixeira da Silva, 660 6° andar Paraíso - CEP 04002-033 São Paulo (SP)
saito@saito.com

Resumo: *O presente estudo objetiva uma metodologia de instalação mais automatizada para estações Fedora Core envolvendo: o uso do Kickstart; criação de CD personalizado e criação de um* script *PHP para personalização da instalação e* scripts *de pós-instalação.*

Palavras-Chave: *Instalação, Kickstart,* `ks.cfg`*, Anaconda, Automação de Instalações, Personalização de Instalações*

1 Comentários Iniciais

A criação de um ambiente de produção para estações de trabalho requer esforços consideráveis para instalação e configuração. Laboratórios de informática e redes corporativas são carentes de uma forma de instalação padronizada e automatizada. É necessário minimizar a intervenção do agente instalador ao mínimo possível e padronizar o máximo possível para que intervenções futuras de reinstalação e manutenção sejam facilitadas. As vantagens almejadas pela automação e padronização podem ser resumidas como se segue:

- menor dependência do nível técnico do profissional;
- maior controle sobre a homogeneidade das estações;
- melhor controle de versões de *software*;
- rápida recuperação de estações;
- rápida instalação inicial;
- diminuição dos custo iniciais e de manutenção.

Instalar uma estação de trabalho é uma tarefa razoavelmente simples. Instalar muitas estações de trabalho é um processo repetitivo e demorado. O processo exige muita concentração e diretivas rígidas para a instalação, junto a pessoal capacitado e disciplinado. Manter a homogeneidade das instalações pode consumir muito tempo e é sujeita a erros.

A escolha cuidadosa de pacotes a serem instalados é uma das etapas mais meticulosas de uma instalação. E quanto mais complexa a escolha e maior a quantidade de detalhes, maior é a chance de erros. Esta escolha de *software* pode ser inicialmente padronizada através da escolha de um ou mais grupos padrões de aplicativos disponíveis na instalação da distribuição utilizada. Contudo, caso a instalação necessite remover muitas opções e adicionar outras tantas, a tarefa de instalação em muitas estações é penosa e sujeita a erros. Além disso o administrador fica preso aos pacotes disponíveis no repositório da distribuição escolhida.

Essa necessidade de automação no processo de instalação levou a RedHat a criar o método de instalação Kickstart, descrito em (HAMILTON, 1999), (RED HAT, INC., 2003a) e (RAU *et al.*, 2005), O Kickstart torna possível instalações com o mínimo de intervenção do agente instalador. Além disso, o projeto Anaconda (RAU *et al.*, 2005), provê uma ferramenta em modo texto e gráfico que facilita o processo de instalação. O objetivo deste artigo, portanto, é propor uma estratégia baseada no Anaconda e no Kickstart para personalização e padronização de instalações automáticas de Linux em laboratórios.

2 Kickstart

Como comentado na seção anterior, o Kickstart torna possível instalações com o mínimo de intervenção. Para isso, o Kickstart usa um arquivo de configuração, descrito em (MCCALLUM, 2004), que contém basicamente as respostas que o instalador Anaconda precisa saber. Trata-se de um arquivo texto simples, sendo que um exemplo comum é o arquivo `anaconda-ks.cfg`, disponível na pasta `/root` logo após uma instalação. Entre outras ações, o Kickstart possibilita:

- seleção de idioma para instalação;
- seleção de idioma para estação;
- seleção de teclado;
- seleção de fuso horário;
- definição da senha do superusuário;
- definições sobre particionamento;
- o gerenciador de inicialização, *bootloader*;
- configuração do servidor gráfico;
- execução de *scripts* para ajustes e configurações iniciais;
- configuração básica de rede;
- níveis de segurança no *firewall*.
- níveis de segurança no Security Enhanced Linux (SELinux)[1];
- escolher qual o método de autenticação;
- escolha dos grupos de pacotes, com possibilidade adição e remoção de pacotes individuais;

Automação adicional na instalação pode ser obtida no processo através da inclusão de *scripts* de instalação no arquivo de configuração do Kickstart. Os *scripts* de pré-instalação devem ser feitos preferencialmente na linguagem Python ou em um conjunto reduzido de comandos de *Shell Script*, já que nesta etapa os recursos disponíveis são reduzidos. Já *scripts* de pós instalação podem ser realizados em qualquer linguagem de linha de comando que tenha sido instalada.

Os grupos de pacotes são classificados por funcionalidade. Dentro dos grupos existem pacotes opcionais, padrões e mandatórios. Os pacotes opcionais e padrões podem ser livremente removidos ou adicionados (RAU *et al.*, 2005). Os grupos mandatórios não são visíveis na interface de seleção e são instalados sempre. O bloco `%packages`, mostrado na Figura 1, contém um exemplo de seleção de pacotes. A representação utilizada é a seguinte:

[1] SELinux: `http://www.nsa.gov/selinux`

- um sinal '@' antes do nome representa um grupo, e.g. **@games**;
- um sinal de '–' indica que um pacote sugerido não será instalado, e.g. **-freeciv**;
- um pacote despido de sinais indica um item opcional, e.g. **mozilla-mail**.

```
        ...
%packages
@games
-freeciv
mozilla-mail
        ...
```

Figura 1: Seleção de pacotes

Desde a versão 8, a Red Hat usa um arquivo XML para definição de grupos de pacotes (RED HAT, INC., 2003a). Este arquivo, `comps.xml`, define quais são os pacotes disponíveis para instalação. O ajuste fino de pacotes deve ser realizado a partir dos grupos definidos nesse arquivo. A alteração do arquivo `comps.xml` exige cuidados, pois contém mais de cinco mil linhas e o instalador faz uma verificação rígida de sintaxe XML. A Figura 2 mostra um exemplo desse arquivo.

```
<group>
  <id>somegroup</id>
  <name>Sample Group</name>
  <default>true</default>
  <uservisible>false</uservisible>
  <description>This is a silly sample group</description>
  <packagelist>
    <packagereq type="mandatory">bash</packagereq>
  </packagelist>
</group>
```

Figura 2: Trecho do arquivo `comps.xml`

Uma das maiores dificuldades no ajuste fino de uma instalação é a existência de pacotes que não fazem parte do `comps.xml`, e que são de interesse da instalação. Como exemplo de adição de pacotes não disponíveis na distribuição padrão e de pacotes não disponíveis na seleção durante a instalação, o pacote `xmms-mp3`[2] foi inserido no CD de instalação do Fedora Core 2, junto com o XFCE[3] e o OpenOffice.org[4].

3 Metodologia

A primeira etapa para personalização e automatização é definir um modelo para cada perfil de instalação. Isso pode ser obtido por uma instalação planejada, com seleção cuidadosa de pacotes e parâmetros. Isto fornecerá um arquivo de configuração do Kickstart

[2] XMMS: `http://www.xmms.org/`.
[3] XFCE: `http://www.xfce.org/`.
[4] OpenOffice.org: `http://www.openoffice.org/`.

mais próximo do pretendido. Para estre trabalho, três perfis simples foram definidos para configuração do Kickstart:

Genérico: Instalação simples de uma estação baseada em GNOME para uso apenas com *software* de produtividade.

Personalizado: Instalação de uma estação com XFCE e OpenOffice.org.

Servidor: Instalação de uma estação para uso em treinamento de servidores.

As instalações foram realizadas em uma máquina de testes e os arquivos de relatório e configuração do instalador foram armazenadas para análise e uso posterior. Definido o perfil modelo, recria-se o conteúdo do futuro CD usando as ferramentas disponíveis no pacote `Anaconda-runtime` e *scripts* para coletar os arquivos RPMs necessários e testar dependências. Posteriormente o conteúdo do futuro CD recebe um arquivo de configuração Kickstart e a etapa de *boot* é modificada para instalação não assistida pela modificação do arquivo `isolinux.cfg`.

Um esforço grande foi feito para que a instalação coubesse em apenas um CD. Um dos motivos é que, na distribuição Fedora, os CDs são criados de tal forma que pacotes mais populares ficam em CDs de numeração menor. Contudo, qualquer inclusão de suporte ao Português solicita o terceiro CD. Além disso, a troca de CDs impossibilita uma instalação não assistida. Para instalações múltiplas e simultâneas, o processo necessita de diversos conjuntos de CDs, aumentando o custo e o tempo necessários. Sincronizar a criação destes CDs e garantir que os conjuntos são funcionais exige bastante trabalho. Aliado ao método Kickstart, a instalação não-assistida através de um único CD pode conseguir:

- homogeneidade das instalações para um grande parque;
- melhor controle sobre os aplicativos instalados;
- distribuição do CD para treinandos – a instalação em casa/escritório será a mesma usada no laboratório;
- possibilidade de atualização de versões durante o processo de criação do CD – isso diminue consideravelmente o consumo de banda, já que as novas estações já nascem atualizadas;
- diminuição no custo de instalação e manutenção;
- possibilidades novas para que tutores tenham alunos com instalações padronizadas no ensino a distância.

O conteúdo personalizado pode ser usado em instalações via HTTP, usando um servidor Apache e um *script* em PHP para geração de roteiros de instalação sob demanda. Com isso, obtém-se instalação não-assistida e padronizada mesmo em máquinas que não disponham de leitoras de CD. Entretanto: Mesmo com o uso do CD personalizado e uso do método Kickstart, um grande parque de máquinas implica na existência de características que devem ser únicas para cada estação, entre outras:

1. Endereçamento;
2. Identidade da estação,
3. Função da estação, implicando no conjunto de pacotes a serem instalados.

A criação de um repositório central permite diminuir ainda mais os custos de instalação e manutenção. Contudo, a existência de sítios distantes e sem acesso razoável a Internet pede a existência dos CDs. Para instalação via HTTP, um *script* provê o roteiro da instalação e ações pós-instalação para pré-configuração básica da instalação. O CD ou disquetes de *boot* devem ser modificados para instalação via HTTP e o operador pode então usar parâmetros na linha de comando inicial para personalizar a instalação, como por exemplo: "`linux ks=http://192.168.1.1/ks.php?tipo=ws1`".

4 Implementação

4.1 Obtenção da lista de pacotes instalados

A lista de pacotes instalados pode ser obtida através do comando `rpm -qa > instalados`. Não é possível saber a partir dessa lista qual a arquitetura foi usada pelo instalador (i686 ou i586 ou i386 ou noarch). Caso isso seja necessário, deve-se executar o comando com mais opções: `rpm -qa --queryformat "%{NAME}.%{ARCH}\n"`. Detalhes sobre o comando `rpm` podem ser obtidos consultando-se (BAILEY, 1998), (BARNES, 1999) ou sua respectiva página de manual[5].

O uso de grupos de pacotes como definidos no arquivo `comps.xml` é extremamente trabalhoso caso ocorram muitas inclusões e trocas de pacotes. Como a necessidade de personalizar o sistema pode-se estender a testes e inclusões durante o uso, optou-se por criar um mecanismo que:

1. Fosse independente dos pacotes efetivamente instalados;
2. Fosse independente do arquivo `comps.xml` original.

Este trabalho mostra a criação de um CD de instalação a partir de uma pasta com arquivos RPMs, estejam eles instalados efetivamente ou não e independe do momento da instalação dos pacotes, não usando o `anaconda-ks.cfg`.

4.2 Criação de uma cópia da árvore de diretórios do CD

Uma cópia do primeiro CD original foi copiada para a pasta `my.cd`. Os arquivos RPMs da pasta `/my.cd/Fedora/RPMS` foram em seguida apagados. Todos os arquivos `TRANS.TBL` também foram apagados. Esses arquivos são úteis apenas para mapeamento de nomes em eventuais problemas com comprimento de nomes de arquivos ou gravação com parâmetros errados.

O objetivo dessa etapa é criar um CD personalizado, mas manter a maioria dos arquivos disponíveis no CD original, incluindo os programas e *scripts* de instalação. Contudo, os arquivos mantidos somam cerca de 150MB, o que deixa um espaço disponível para os arquivos RPMs personalizados de cerca de 550MB.

4.3 Criação e cópia dos pacotes instalados

Os arquivos RPMs dos quatro CDs originais disponíveis foram copiados para uma pasta chamada `normal`. Em uma máquina de testes, pacotes foram adicionados e removi-

[5] A página de manual do comando `rpm` pode ser lida através do comando `man rpm`.

dos para personalização. O aplicativo `system-config-packages` (*Adicionar e Remover Aplicações*) foi usado juntamente com o uso direto do comando `rpm` na linha de comando.

De posse da lista de arquivos instalados, é necessário coletar os arquivos correspondentes. Um *script* em linguagem Perl foi desenvolvido para realizar a transferência dos arquivos da pasta `normal` para a pasta `/my.cd/Fedora/RPMS`. O *script* lê linha a linha os arquivos instalados; adiciona os finais de nome de arquivo possíveis e testa cada combinação para verificar a existência. Caso a equivalência seja encontrada, o *script* move o arquivo RPM.

Para garantir que não existirão problemas de quebra de dependências foi utilizado um *script* descrito em (WATSON, 2005) para testar o conjunto de arquivos RPMs selecionados. O *script* foi executado dentro da pasta de arquivos selecionados. A saída do ≪script≫ não deve mencionar nenhuma dependência. Como bônus o procedimento mostra uma estimativa do espaço em disco que será ocupado pelo conjunto.

Para continuar a criação dos CDs, pacotes de ferramentas adicionais do Anaconda foram instaladas. Essas ferramentas servirão para gerar arquivos necessários à criação do CD.

4.4 Geração das listas hdlist e hdlist2

O arquivo `comps.xml` contém os grupos e categorias de pacotes RPMs disponíveis para o instalador Anaconda. Porém não são listados os nomes e localizações dos arquivos propriamente ditos. Também não existe mapeamento para as versões e arquitetura definidas dos pacotes. Desta forma é necessário que exista um mecanismo para que ocorra o mapeamento entre o pacote listado no arquivo `comps.xml` e o arquivo em si.

O Anaconda usa os arquivos `hdlist` e `hdlist2` para essa função. Esses arquivos mapeiam os arquivos RPMs disponíveis. Para gerar esses arquivos, é utilizado o comando `genhdlist`, disponibilizado no pacote `Anaconda-runtime`. Antes de usá-lo algumas variáveis de ambiente necessitam ser criadas:

A pasta principal do futuro CD:

```
export BASE=/home/build/my.cd
```

A pasta das bibliotecas do Anaconda:

```
export PYTHONPATH=/usr/lib/anaconda
```

A pasta dos utilitários Anaconda:

```
export PATH=$PATH:/usr/lib/anaconda-runtime
```

Após a definição da variáveis de ambiente, o comando:

```
genhdlist --productpath=Fedora $BASE,
```

substitui os arquivos hdlist e hdlist2 na pasta `/my.cd/Fedora/base`. É necessário especificar o parâmetro `--productpath=Fedora`, pois o padrão é ≪RedHat≫. É possível até trocar o nome para qualquer outro desde que também se troque o nome na pasta `my.cd`.

4.5 Geração do arquivo de ordem de instalação

O Anaconda precisa saber em que ordem instalar os pacotes disponíveis. A ordem incorreta de instalação pode gerar problemas de dependência, mesmo com a disponibilidade dos arquivos necessários. A ordem de instalação pode ser obtida com o comando:

```
pkgorder $BASE/i386 i386 Fedora {\textgreater} $BASE/pkgorder.txt
```

4.6 Regeneração das imagens de instalação

O comando `buildinstall` é usado para gerar o arquivo .discinfo e a assinatura de data para o CD. O comando deve ser executado na pasta `/usr/lib/anaconda-runtime`, como superusuário:

```
buildinstall --pkgorder $BASE/pkgorder.txt --comp dist-1 \
--product Fedora --version 1 --release "GINUX" --prodpath \
Fedora $BASE/ | tee log.buildinstall
```

Sem essa etapa, o Anaconda não reconhece o CD como válido. Caso o volume de dados ultrapasse o CD, o `Anaconda-runtime` disponibiliza o programa `splittree.py` para dividir os arquivos em diversos CDs. Nesse caso, como observado em (RAU *et al.*, 2005), a ordem dos comandos é `genhdlist`, `pkgorder`, `buildinstall` e finalmente `splittree`.

4.7 Modificação do arquivo `isolinux.cfg`

Para que o CD inicie automaticamente uma instalação via Kickstart, é necessário editar o arquivo `isolinux.cfg` dentro da pasta `isolinux`. Para tanto as linhas a seguir foram inseridas nos locais correspondentes:

```
default=linux ks
```

e

```
linux ks=cdrom:/ks.cfg initrd=initrd.img ramdisk_size=8192
```

Um arquivo de exemplo está disponível no *site* deste projeto[6]. É possível também alterar as instruções iniciais do CD personalizando os arquivos de extensão `.msg` na pasta `isolinux`.

Depois é necessário copiar o arquivo `ks.cfg` de referência para a pasta `/my.cd`. É possível copiar mais de um arquivo de configuração do Kickstart no CD. Para usar um arquivo diferente, basta informar ao instalador:

```
linux ks=cdrom:/ks-alternativo.cfg
```

Isso permite que o mesmo CD seja usado para instalações automatizadas com diferentes parâmetros de segurança e perfis de *hardware* e usuário.

[6] *Site* do projeto: `http://www.ginux.ufla.br/projetos/kickstart`.

A imagem ISO do CD foi criada usando a pasta `my.cd` e o Mkisofs. A imagem gerada pode ser gravada usando qualquer programa de gravação de CDs como o Cdrecord[7].

4.8 A geração do `comps.xml`

Uma das estações de teste foi personalizada pela instalação do ambiente XFCE. A lista de pacotes necessários ao final do processo ultrapassou a capacidade de um CD convencional. Foi necessário remover uma série de pacotes relacionados ao GNOME e a eventuais programas não utilizados. Cada remoção era interrompida pela possível quebra de dependências. Ao final foi possível, com folga, ter uma lista de pacotes que contemplasse OpenOffice.org e XFCE em um único CD.

Contudo a tarefa de descobrir que pacote faz parte de qual grupo e se vai ser adicionado como um item novo ou simplesmente selecionado torna-se inviável para um esforço isolado. O uso do `comps.xml` diretamente é trabalhoso e sujeita o administrador à fadiga ocular extrema e dores insuportáveis na coluna. O arquivo é muito grande e a busca, mesmo com ferramentas, é trabalhosa e sujeita a erros.

A solução encontrada foi criar um arquivo `comps.xml` a partir do zero, tornando mandatória a instalação dos pacotes selecionados. Isso é bastante coerente com a metodologia em análise. Se a intenção final é uma instalação não assistida, então todos os pacotes selecionados são mandatórios.

Alguns pacotes foram adicionados à pasta de testes, mas não foram efetivamente instalados. Isto acarretou no nascimento de um novo problema: a metodologia de usar a consulta RPM já não funcionaria. A metodologia de criação de banco de dados RPM temporário descrito em (WATSON, 2005) foi essencial para essa etapa. Desta forma pôde-se descobrir o efeito da adição ou remoção de um pacote RPM sem afetar o sistema.

Esse conjunto de problemas levou à solução de criar um *script* em Perl que gerasse o arquivo `comps.xml` a partir da listagem de um diretório. Um arquivo de inclusão de topo é impresso. A listagem obtida pelo comando `ls -la` é tratada e posteriormente os elementos de versão e arquitetura de cada nome de arquivo são removidos.

O marcadores XML, já com status de mandatório, são adicionados. Finalmente um arquivo de inclusão de rodapé é impresso pelo *script*. Com isso, a área `%packages` do arquivo `ks.cfg` pode ser deixada em branco, uma vez que todos os pacotes e grupos são mandatórios neste método. Utilizando essa abordagem, é possível criar um arquivo `comps.xml` curto e com apenas os pacotes desejados. Contudo alguns pacotes necessários não são visíveis apenas pelos testes de dependências, e.g. a falta do `xfwm4`, gerenciador de janelas do XFCE, não gera problemas de dependência, mas deixa o sistema inutilizável. Uma vez tendo copiado o arquivo `comps.xml` gerado para a pasta `my.cd`, basta criar o CD e proceder às instalações.

[7] Mkisofs e Cdrecord fazem parte do pacote `cdrtools`, disponibilizado em `http://cdrecord.berlios.de/old/private/cdrecord.html`.

5 O servidor de instalações

Esta seção apresenta os procedimentos executados para a configuração de um servidor de instalações via *Web*. O objetivo com isso é obter uma forma de instalar várias estações de forma padronizada, a partir de um repositório comum, minimizando custos com mídias.

5.1 Preparação do servidor DHCP

O arquivo /etc/dhcpd.conf foi configurado para fornecer endereços na faixa 192.168.1.10 a 192.168.1.199. Ele não fornece nenhuma outra informação. O objetivo é apenas fornecer a estação a ser instalada uma rota com o servidor de instalação. Os endereços fornecidos pelo configurador não são os mesmos da faixa servida pelo DHCP.

5.2 Configuração para instalação via `ks.cfg` estático

A instalação precisa ser iniciada a partir de um disquete ou de um CD. Para automatizar o sistema, o arquivo `isolinux.cfg` na pasta isolinux do disquete/CD foi modificado para refletir uma instalação a partir de um arquivo `ks.cfg` proveniente do servidor HTTP:

```
linux ks=http://192.168.1.1/ks.cfg
```

Esse arquivo `ks.cfg`, por sua vez, reflete a instalação via HTTP. O trecho mostrado na Figura 3, mostra que a lista de pacotes é vazia, já que todos os pacotes são mandatórios. Além disso, mostra que os pacotes estão disponíveis na URL descrita.

```
...
install
url --url http://192.168.1.1/.
...
%packages
```

Figura 3: Trecho de arquivo `ks.cfg`

5.3 Arquivo `ks.cfg` dinâmico via *Web*

Aproveitando-se da distribuição de arquivo `ks.cfg`, este projeto utilizou-se da criação de arquivos `ks.cfg` dinâmicos, utilizando para isso de um *script* simples em PHP. O motivo da criação deste *script* é gerar perfis de instalação baseados em um pedido de arquivo via HTTP. Esses perfis são devolvidos na forma de arquivos de configuração do Kickstart. Desta maneira centraliza-se a criação dos arquivos de instalação. Mesmo que a instalação seja feita por CD, o *script* ainda é bastante útil:

1. Estudantes podem digitar o endereço de configuração com os parâmetros corretos e receber o arquivo ks.cfg correspondente e posteriormente gravá-lo em um disquete e iniciar a instalação local via CD. Isto permite que o objetivo da homogeneidade seja atingido mesmo com instalações via CD.
2. Testes de compatibilidade de novos perfis de instalação com os CD já disponíveis.

3. Atualizações de distribuição.

A listagem apresentada na Figura 4 mostra um *script* mínimo para o criador de arquivos `ks.cfg`. Basicamente os pedidos são analisados e então passados para a criação do arquivo correspondente.

```
<?php
include("ks-generico.inc");// configuracoes comuns
if (strtolower($_GET[tipo]=="ws1")) {
       print "#Estação Padrão Tipo WS1\n";
       include("ks-ws1.inc"); include("ks-ws1.scripts.inc");
       }
else {
       print "Estação Padrão Tipo WS1";
       include("ks-ws1.inc"); include("ks-ws1.scripts.inc");
       }
?>
```

Figura 4: *Script* simples para criação de arquivos `ks.cfg`

5.4 Configurações pós-instalação

Na etapa de pós-instalação, a estação instalada pode receber configurações adicionais, com uso de *scripts* externos. A Figura 5 ilustra um trecho de arquivo `ks.cfg` com configurações adicionais utilizando-se *scripts* em um repositório NFS.

```
%post
# Serviços
chkconfig sshd --level 35 on
#NFS
mkdir -p /mnt/REDE
echo '192.168.1.11:/home/REDE /mnt/REDE nfs exec,dev,suid,rw 1 1' \
  >> /etc/fstab
mount /mnt/REDE
#Instala últimas atualizações de disponíveis
rpm -Uvh /mnt/REDE/*rpm
#Executa script de finalização
bash /mnt/REDE/faz.o.que.quer.com.esta.maquina.sh
```

Figura 5: Seção de pós-instalação de arquivo `ks.cfg`

6 Outros métodos

A clonagem de discos é uma solução alternativa ao caminho proposto neste artigo. Exemplos de programas disponíveis para clonagem são o Partimage[8], o comando dd[9], UDP-

[8] Partimage: `http://www.partimage.org/`.

[9] O comando `dd` faz parte do pacote coreutils, disponível em `http://www.gnu.org/software/coreutils/`.

cast[10] e Norton Ghost[11]. As vantagens dos métodos de clonagem em relação ao método descrito neste artigo são:

1. A rápida implementação: o método apresentando neste artigo é razoavelmente complexo.
2. Facilidade de criação de imagens: comandos simples como `dd` são facilmente incorporados em programas *Shell Script*.

Como desvantagens do método de clonagem em relação ao método descrito neste artigo, podem ser citadas:

1. A dificuldade de gerenciar parques heterogêneos. No método descrito neste artigo, todos os métodos de detecção de uma instalação convencional são usados pelo Anaconda.
2. A necessidade de personalizar as instalações após a clonagem. Sem ações posteriores, as máquinas são cópias idênticas, com configurações a serem feitas após a instalação.
3. A criação de perfis de instalação pode ser feita pela armazenagem de imagens de disco. Isso pode significar em grande consumo de espaço em disco. No método HTTP descrito neste artigo, basta um repositório centralizado e espaço insignificante para os *scripts* de criação de roteiros de instalação.
4. Qualquer alteração mínima no modelo a ser replicado implica em regeneração das imagens. Isso é trabalhoso e demorado. No método descrito neste artigo basta editar os arquivos de configuração ou editar os *scripts* de geração em PHP.

Um método de automação de instalações existe para a distribuição Debian[12]. O método FAI[13] (Instalação Totalmente Automática de Debian) não foi testado neste trabalho. O método promete um alto grau de personalização e controle das instalações. Contudo, não é disponível para distribuições baseadas em pacotes RPMs.

7 Conclusões

O processo de personalização do instalador do Fedora Core é trabalhoso e necessita de bastante estudo e cuidados. A sua utilização em locais com um parque instalado grande ou em locais com pessoal de manutenção reduzido compensa este trabalho. O controle de versões, a homogeneidade das instalações representa um grande passo para a diminuição do custo total de propriedade. O mercado corporativo procura soluções que permitem diminuir custos.

O uso da metodologia descrita neste artigo em um laboratório de testes com doze máquinas com configurações de *hardware* distintas mostrou-se eficaz, com plena obtenção dos objetivos desejados.

[10] UDPcast: `http://udpcast.linux.lu/`.
[11] Norton Ghost: `http://www.symantec.com/home_homeoffice/products/backup_recovery/ghost10/`.
[12] Debian: `http://www.debian.org`.
[13] FAI: `http://www.informatik.uni-koeln.de/fai/`.

O Linux permite diminuição de custos com aquisição de licenças. As estratégias de sedução por facilidade de gerenciamento e manutenção do parque instalado são fundamentais para adoção em massa do Linux como alternativa viável no mercado de estações de trabalho.

Observe que o presente trabalho facilita a criação de uma distribuição baseada em RPMs. O caminho para isso inclui principalmente (MORRIS, 2005):

- Remoção de todas as referências (texto e imagens) a "Red Hat" e "Fedora" no código-fonte do instalador Anaconda e em várias ferramentas como `inittools`;
- Troca do arquivo `redhat-release` ou `fedora-release` por um arquivo correspondente à distribuição;
- Adequação das dependências a pacotes como `fedora-logos`, `fedora-release`, bem como recompilação dos mesmos;
- Edição dos arquivos de mensagem de *boot* e de escolha de métodos de instalação na pasta isolinux;
- Adequação dos *menus* à nova distribuição.

Dessa maneira, este trabalho sugere como trabalhos futuros a criação de mecanismos que automatizem o processo de criação de distribuições personalizadas.

Referências

BAILEY, E. C. *Maximum RPM: Taking the Red Hat Package Manager to the Limit*. Raileigh: Red Hat Software Inc., 1998. Disponível em: <http://www.rpm.org/>.

BARNES, D. *RPM HOWTO, Revision 3.0*. The Linux Documentation Project, 3 November 1999. Disponível em: <http://www.ibiblio.org/pub/Linux/docs/HOWTO/RPM-HOWTO>.

HAMILTON, M. *RedHat Linux KickStart HOWTO, v0.2*. The Linux Documentation Project, January 1999. Disponível em: <http://www.ibiblio.org/pub/Linux/docs/HOWTO/KickStart-HOWTO>.

MCCALLUM, E. *Hands-Off Fedora Installs with Kickstart*. August 2004. WWW (O'Reilly linuxdevcenter.com). Disponível em: <http://www.linuxdevcenter.com/pub/a/linux/2004/08/19-/kickstart.html>.

MORRIS, J. *How I did it, or "HOWTO Roll your own"*. 15 mar. 2005. WWW. Disponível em: <http://www.whiteboxlinux.org/howto.html>. Acesso em: 05/06/2005.

RAU, A.; AHMED, A.; LONG, B.; SCHWARZ, B.; TAYLOR, F.; MARTIN, J.; KATZ, J.; GLASS, J.; M., M. K.; DEVINE, P.; PIANTA, P. *Anaconda Documentation Project*. Anaconda Documentation Project, 18 Mar 2005. Disponível em: <http://rau.homedns.org/twiki/bin/view-/Anaconda/AnacondaDocumentationProject>.

RED HAT, INC. *Red Hat Linux 9: Red Hat Linux Customization Guide*. Raleigh NC: Red Hat, Inc., 2003. Disponível em: <http://www.redhat.com/docs/manuals/linux/>.

RED HAT, INC. *Red Hat Linux 9: Red Hat Linux x86 Installation Guide*. Raleigh NC: Red Hat, Inc., 2003. Disponível em: <http://www.redhat.com/docs/manuals/linux/>.

WATSON, G. *Customized RedHat Linux Kickstart Installation Cdrom*. 2005. WWW. Acesso em 18/05/2005. Disponível em: <http://256.com/gray/docs/redhat_boot/>.