O slackware é a distribuição linux mais antiga ainda em atividade. Tendo sido criada por Patrick Volkerding em 1993, a partir da SLS.

Em todos esses anos, a distro conquistou ardorosos utilizadores, principalmente graças à sua filosofia de simplicidade e estabilidade.

Um produto de extrema qualidade para usuários com esta mesma característica. E este zine é de slacker para slacker.

slackware zine

29 de Novembro de 2004 – Edição #6

Editorial

Este zine sai, além de atrasado, como portador de más notícias. Não sabemos quantos sabem, mas o criador do **slackware**, Patrick Volkerding está doente e beeem doente.

Como a distância não permite uma ajuda maior, vamos concentrar as nossas orações (ou rezas, ou pensamento positivo, ou seja-lá-o-que-as-suas-crenças-permitam-que-você-faça) e cruzar os dedos, torcendo pela melhora dele.

Na ausência do Patrick, os updates de segurança estão sob responsabilidade de alguns membros do GUS-Br, e podem ser encontrados no seguinte endereço:

http://www.slackware.org.br/~patrick/WORKGUS

Bom, deixando de lado as más notícias, essa edição do zine trás uma série de artigos interessantes, e dois novos colaboradores.

O primeiro artigo apresentado mostra como utilizar o SVG para imcrementar as suas páginas na internet, logo em seguida, Clayton Eduardo nos apresenta como utilizar o CUPS para imprimir em máquinas Windows. Algumas dicas de como usar o CVS são apresentadas pelo novato Formiga. Utilizar o VNC para trabalhar remotamente, configurar um DNS+DHCP em um único programa e instalar **slackware** em um Macintosh completam essa edição. Fechamos com chave de ouro com o artigo sobre sistemas de arquivos criptografados, do Wolvie.

Espero que se divirtam

Piter PUNK

Índice

simplicity is divine

slack users

Gráficos Vetoriais para Web

Para cativar diretamente a percepção dos nossos usuários, usamos e abusamos de componentes gráficas em nossas páginas. A maioria das vezes utilizamos gráficos rasterizados, ou seja, criados de maneira estática, como os formatos de imagem GIF e JPG, mas sempre pensamos como seria interessante selecionar apenas partes de interesse em uma imagem, ou ainda, editar via linha de código seu conteúdo.

Foi a partir desse princípio, presentes em formatos como CAD, que a W3C (`http://www.w3c.org`), organização responsável pela especificação dos padrões utilizados na internet, criou um novo formato para gráficos chamado de SVG (Scalable Vector Graphics). O SVG é um formato para descrever gráficos vetoriais e animações, utilizando arquivos XML(eXtensible Markup Language) que contenham um conjunto padronizado de tags para os elementos básicos de desenho, além de permitir a alteração de estilo de seus gráficos, por CSS e interação por JavaScipt.

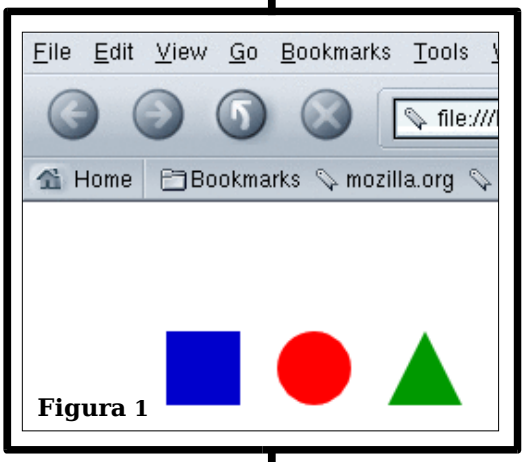


Figura 1

Um gráfico vetorial corresponde a um conjunto de entidades geométricas, como retas, quadrados, círculos, curvas simples e complexas. Para se criar um gráfico nesse formato, necessita-se manipular essas tags elementares, por meio de um editor de texto, como o VI, do mesmo modo como é criado uma página simples em HTML.

Para abrir um SVG por um navegador, é necessário que se possua um plug-in apropriado para renderizá-lo. Por tratar-se de um gráfico vetorial, podemos obter um perfeito zoom, pois o plug-in renderiza novamente as primitivas gráficas no tamanho requisitado sem perder a definição da figura.

Existem vários plug-ins na web, mas infelizmente, ainda não obtiveram resultados satisfatórios para utilização em navegadores como Mozilla e FireFox. O plug-in da Adobe renderiza perfeitamente gráficos simples para Mozilla, como o exemplo abaixo, mas deixa a desejar em aplicações mais complexas que utilizam eventos em JavaScript. Segundo os desenvolvedores da Adobe, está prevista o lançamento de uma nova versão do plug-in, com a correção desses erros para novembro. Enquanto isso, podemos estudar o código padrão SVG...

Na **figura 1** é apresentado o resultado do código descrito acima, exemplificando alguns dos elementos básicos. Quando aberto no navegador Mozilla, com instalação do plug-in Adobe Viewer, encontrado em `http://www.adobe.com/svg`.

Diego Fiori de Carvalho
<dfiori@grad.icmc.usp.br>

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd" [
  <!ATTLIST svg xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink">
]>

<!-- Inserção de código svg com padrão xml. -->
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">

<!-- Título é inserido do mesmo modo que HTML. -->
  <title>SVG - Slackwarezine</title>

<!-- abre espaço para texto, parecido com H1(HTML). -->
<desc>Scalable Vector Graphics - Meu primeiro SVG</desc>

<!-- retângulo, posição x e y, largura e comprimento e estilo determinando a cor azul.-->
<rect x="90" y="70" width="40" height="40" style="fill: #00C"/>

<!-- círculo, posição x e y, raio e estilo determinando a cor vermelha. -->
<circle cx="170" cy="90" r="20" style="fill: #F00"/>

<!--- polígono, posição x e y para cada vértice, no caso um triângulo e estilo determinando a cor
verde. -->
  <polygon points="210,110 230,70 250,110" style="fill: #090"/>

</svg>
```

Imprimindo em impressoras Windows pelo CUPS

Em edições anteriores do **slackwarezine**, tivemos a oportunidade de ler artigos realmente bons relacionados à impressão em ambiente Linux.
Bem, você deve estar se perguntando agora: "Se os artigos foram escritos em outras edições, porque um novo artigo abordando o mesmo assunto?".

Essa pergunta é natural e irei respondê-la com prazer. Acredito que grande parte dos usuários Linux, em especial os newbies, seja por necessidade ou mesmo por curiosidade, em ambiente local ou corporativo, dependem de uma impressora instalada em uma máquina com "aquele" sistema operacional. Como esse é, em muitos casos, um "mal necessário", resolvi escrever esse artigo... :)
Vamos a ele!!!

Como o processo de instalação do CUPS já foi detalhado na edição número 02 do **slackzine**, não irei repeti-lo aqui, somente darei algumas dicas que julgo importantes durante o processo de instalação/configuração de uma impressora remota instalada em uma máquina Windows.

Uma primeira dica, que deve ser observada antes mesmo de instalar o CUPS, é verificar se o SAMBA está corretamente instalado no sistema, pois caso contrário, o tipo de compartilhamento baseado no protocolo smb não será listado no gerenciador de impressoras do CUPS durante a configuração da impressora, uma vez que o samba não está presente no sistema, ou ainda, não foi iniciado.
Para verificar se o samba já foi instalado junto com o seu bom e velho **slackware**, basta verificar se existe um registro do mesmo em `/var/log/packages`, bastando para tanto, digitar:

```
ls /var/log/packages/sam*
```

ou ainda,

```
pkgtool
```

Caso o pacote samba tenha sido instalado, pode ser que ele não tenha sido iniciado durante o processo de boot, para tanto, faça o seguinte:

```
cd /etc/rc.d
chmod +x rc.samba
```

Verifique também se o seu sistema possui o arquivo de configuração `smb.conf`, normalmente instalado em `/etc/samba`, não confundir com o arquivo `smb.conf-sample`, que oferece exemplos de configuração comentados (e pode ser usado como base para você criar o seu `smb.conf`)

Estando o samba instalado, configurado corretamente no sistema e iniciado, vamos partir para a configuração das impressoras.

Pois bem, em sistemas baseados em Win9x/ME ou mesmo Linux, o processo de configuração das impressoras remotas é idêntico e bastante simples, basta que se acrescente uma entrada do tipo:

```
smb://máquina_que_a_impressora_está_instalada/nome_do_compartilhamento_da_impressora
```

> **DICA:**
> Para que o acesso possa ser realizado corretamente através do `hostname` da máquina, lembre-se de incluir uma entrada referente a ela no arquivo `/etc/hosts` ou no servidor DNS de sua rede local.

Em sistemas baseados em Win2k/XP, a sintaxe é um pouco diferente e às vezes isso pode fazer com que você perca alguns minutos ou mesmo algumas horas pesquisando a solução. A sintaxe é a seguinte:

```
smb://grupo_de_trabalho_windows/usuario@senha/máquina_que_a_impressora_está_instalada/nome_do_compartilhamento_da_impressora
```

Exemplo prático:

```
smb://administrativo/clayton@senha/maquina04/hp840c
```

Feito isso sua impressora Windows remota deverá funcionar normalmente, no entanto, em alguns casos é possível que você receba a fatídica mensagem: `NT_STATUS_UNSUCCESSFUL`...

Bem, muito provavelmente sua máquina linux ou mesmo a máquina windows onde a impressora está instalada está com as portas bloqueadas, para corrigir esse problema, você precisa desabilitar as seguintes portas em seu firewall: `137`, `138`, `139` e `445`.

Se o problema for na máquina Windows, provavelmente você terá que jogar seu "free firewall" no lixo, pois ele não possui suporte a gerenciamento de portas, portanto, você terá de optar uma versão comercial com suporte a esse recurso.

Acredito que, com essas dicas, sua impressora remota windows funcionará perfeitamente.

Clayton Eduardo <claytones@terra.com.br>

Guia Rápido para o CVS

1. Criando o ambiente de trabalho

Crie um diretório raiz onde trabalhará com os módulos CVS.

Se quiser criar um módulo, crie o diretório, os arquivos e depois use o comando `import`, na raiz do diretório do módulo:

```
cvs import -m"Nome Completo do Módulo"\
      NomeModulo NomeVendedor NomeVersão
```

No diretório raiz criado para trabalhar com os módulos CVS, para fazer download via `cvs` de um módulo, faça:

```
cvs checkout NomeModulo
```

O `checkout` irá fazer o download do conteúdo do módulo para o diretório atual.

2. Trabalhando com os arquivos locais e atualizando

O trabalho local é feito normalmente. Entretanto, deve-se prestar atenção à remoção e criação de arquivos, pois os mesmos devem ser informados ao CVS. Após fazer as alterações, pode-se verificá-las com:

```
cvs diff caminho/relativo/do/arquivo
```

ou

```
cvs diff
```

para verificar todas as mudanças

NOTA:

os subcomandos do comando `cvs` têm parâmetros: `cvs --help subcomando`

Ao término das alterações locais, deve-se atualizar a árvore CVS. Para isto, usa-se o subcomando `commit`:

```
cvs commit caminho/relativo/do/arquivo
```

Será aberto o editor de textos especificado em `$EDITOR` para que seja digitado o comentário relacionado à atualização. Para que a atualização seja feita diretamente, faz-se:

```
cvs commit -m"Digite aqui a alteração\
      feita nesta versão" \
      caminho/relativo/do/arquivo
```

Pode-se fazer uma atualização global com:

```
cvs commit -m"Digite aqui cada mudança\
      feita em cada um dos arquivo"
```

Antes de começar...

Para acessar localmente:

```
export CVSROOT=/var/lib/cvs
```

Para acessar remotamente:

```
export \

CVSROOT=":pserver:user@host:\
      /var/lib/cvs"
```

Autenticação:

```
cvs login
```

Para que a autenticação não seja necessária, basta copiar a chave pública usuário local para o `$HOME/.ssh/authorized_keys` do usuário remoto autenticado.

Free is Good...
Open Source is Good...

O comportamento para "cvs commit" é o mesmo de "cvs commit arquivo"; portanto, a mensagem do "commit" global deve especificar as mudanças feitas em cada arquivo.

O comentário de mudanças feitas no arquivo, requerido pelo subcomando commit, tem função de relatório de mudanças da versão atual em relação à versão anterior dos arquivos do projeto. Caso não seja especificado o parâmetro "-m", o comando "cvs commit" mostra um cabeçalho pré-montado no editor de textos que é chamado, especificando os arquivos alterados, como no exemplo:

```
CVS: Modified Files
CVS: arquivo1 arquivo2 arquivo3
CVS: ----------------------------------
```

Qualquer linha que comece com "CVS" será removida automaticamente. Módulos e subdiretórios de módulos podem ser marcados como inativos. Para isto, usa-se o comando

```
cvs release NomeModulo
```

ou

```
cvs release NomeModulo/diretorio
```

Nota: todos os caminhos são relativos ao diretório corrente ($PWD) e não à árvore principal do módulo utilizado

Para remover um arquivo, subdiretório ou módulo do repositório, o subcomando utilizado é o "remove":

```
cvs remove \
      nomedo{arquivo|subdiretório|módulo}
```

Para adicionar um arquivo ou subdiretório ao módulo em uso, o subcomando utilizado é o "add":

```
cvs add nomedo{arquivo|subdiretório}
```

Para que as alterações dos subcomandos remove e add façam efeito, o subcomando "commit" deve ser utilizado posteriormente.

Formiga <eduardo.c.lisboa@gmail.com>

Compartilhando o X com o VNC 4

Aqui vai uma dica rapidinha pra quem quer compartilhar a sessão do X sem ter q rodar o X inteiro dentro do VNC (Xvnc, na verdade).

Com o Real VNC 4 - http://www.realvnc.com/ - veio um programa chamado x0vncserver. Esse programa fica consultando os eventos do X local e os reflete na sua conexão VNC.

O único "truque" é que o x0vncserver não procura pelo arquivo de password padrão (o ~/.vnc/passwd), sendo necessário passar parâmetros para o programa.

Pra criar a senha do VNC, rode o comando vncpasswd. ele criara o ~/.vnc/passwd pra você. Agora, basta passar esse parâmetro para o x0vncserver (se você estiver no próprio X a ser compartilhado):

```
x0vncserver PasswordFile=~/.vnc/passwd
```

ou

```
DISPLAY=:0.0 x0vncserver \
      PasswordFile=~/.vnc/passwd
```

A variável DISPLAY é útil se você estiver conectado via SSH e precisa acessar o X da máquina remota ; basta fazer um "su - usuário" e rodar o x0vncserver com a variável DISPLAY setada.

Deives Michellis “the fallen”
<thefallen@unitednerds.org>

Macintosh+slackware?

Instalando o slackintosh

1. Introdução

Bom, depois de muito tempo pensando em como conseguir um Macintosh, finalmente coloquei a mão em um deles para a minha coleção: um PowerMac 5500. Máquina bem interessante, com monitor integrado e já veio com modem e placa de rede.

Só havia um problema, e esse problema se chamava MacOS. Mais precisamente MacOS 8.6. Instantes depois da compra, lá estava eu pesquisando qual UNIX-like poderia rodar nessa máquina. As respostas dessa pesquisa foram bem promissoras.

Tanto o Linux como o NetBSD rodam nesse Mac, e rodam muito bem! O primeiro que instalei foi o NetBSD, pois a documentação era bem mais farta, e o procedimento de instalação estava indicado passo-a-passo. Depois de instalado e de ter aprendido direito como esse Mac funciona, resolvi passar para o próximo estágio: instalar Linux nele.

Nessa hora a minha decepção, o **slackintosh** havia sido descontinuado -:(. Outra pesquisa no google e achei um mirror do projeto, com os arquivos para a instalação do **slackintosh** `8.1`, `9.0` e `9.1`. Como o `8.1` estava mais completo, resolvi baixar e instalar e, de quebra, arranjar assunto para um artigo -;). Nada lhe impede de montar um CD com o esqueleto do `8.1` e os pacotes do `9.1`, por exemplo.

2. Considerações iniciais

Esse método que estou empregando funciona apenas para Macs classificados como OldWorld, ou seja PowerMacs que tenham sido fabricados antes do iMac. Para Macs NewWorld é possível gerar um CD bootável e o procedimento é muuuuuito mais simples.

PowerMacs da série x100 também não vão se beneficiar da técnica usada (mas nada impede de eu descobrir outra assim que colocar o 6100 que tem aqui para funcionar).

Por último, como REALMENTE não gostei do MacOS, eu eliminei-o da máquina. Da mesma maneira que os meus PCs, o Mac roda apenas o Linux.

3. A instalação

3.1. Discos de Boot

Vá na URL abaixo e pegue os discos de boot e root:

```
http://mike.quintero.staff.noctrl.edu/\
      slackintosh/
```

Os discos de boot e de root chamam-se, respectivamente: `slackintosh-bootdisk-1.bin` e `slackintosh-rootdisk-1.bin`

Crie os dois discos utilizando o `dd` e coloque primeiro o bootdisk e, quando solicitado, o rootdisk. No final do carregamento do segundo disco, o sistema vai tentar detectar o seu CD-ROM. Como a maior parte dos Macintoshs pré-iMac usam CD-ROMs SCSI, o detector do disquete que procura apenas por dispositivos IDE não vai detectar nada. Você vai precisar inserir o "endereço" do seu CD-ROM. Se estiver com preguiça de procurar, o endereço é `/dev/scd0`

3.2. Particionamento

Depois de carregar o CD, você já deve ter percebido que o sistema de instalação é igual ao do **slackware** que todos nós conhecemos. Não adianta selecionar agora o tipo de teclado, aperte simplesmente <enter>, entre como root e vamos efetuar o particionamento com o `fdisk`:

```
# fdisk /dev/hda
```

Na realidade, este não é o `fdisk` que estamos habituados, é o `pmac-fdisk`, mas foi alterado para que a saída fique parecida com a do `fdisk` e o instalador do **slackware** possa reconhecer as partições pelos seus nomes.

Mais uma diferença: ao contrário do `fdisk` com o qual estamos acostumados, este não detecta automaticamente a próxima posição livre. Cada partição começa exatamente onde termina a partição anterior, exemplo:

```
            Início      Fim
      hda1: 1           64
      hda2: 64          5532
```

Graças a esta característica, é extremamente útil usar o comando 'p' para apresentar a tabela de partições sempre que for incluir uma nova. Você vai precisar de no mínimo 4 partições:

- Uma com a tabela de partições (é do Mac essa, não podemos dar pitaco)
- Uma para a partição `/boot` (aproximadamente 60MB)
- Uma para a partição `/`
- Uma para `swap`

A seqüência para criar essas partições é bem simples, o HD que usei tem 2.1GB:

- Reinicia a tabela de partições (`i`)
  - Cria partição `/boot` (`c`)
  - Escolhe cilindro inicial (`64`)
  - Seleciona o tamanho desejado (`60M`)
- Cria partição `/` (`c`)
  - Escolhe cilindro inicial (o cilindro final da partição `/boot`)
  - Seleciona o tamanho desejado (`1850M`)
- Cria partição de `swap` (`c`)
  - Escolhe cilindro inicial (idem item anterior)
  - Seleciona o tamanho desejado (RAM < tamanho < 2*RAM)
  - Dê para essa partição o nome de "`swap`" (sem as aspas)
- Escrever tabela (`w`)
- Confirmar (`y`)
- Sair (`q`)

3.3. Instalação Propriamente Dita

É executar o `setup` e ir seguindo os passos recomendados. É idêntico ao do **`slackware`** para x86. Alguns detalhes são importantes:

1. É necessário instalar a `glibc` da série `L`, sem ela seu sistema não vai funcionar
2. Instale a série `SLACKINTOSH` ela é totalmente obrigatória.
3. A partição `/boot` deve ser formatada como `ext2`

No final da instalação, apenas lembre-se de NÃO rebootar.

3.4. OpenFirmware

Essa é uma das partes mais divertidas da instalação, trabalhar com o OpenFirmware. Para falar a verdade, é um parto. O OpenFirmware faz o papel da BIOS nos Macintoshs, mas, antes da versão 3 praticamente todos tem bugs ou configurações exóticas. Agora, tentem advinhar qual a versão que vem nos PowerMacs pré-iMac? Obviamente que não a versão 3, teremos que trabalhar com as buguentas versões 1.x e 2.x.

Para entrar no OpenFirmware, você deve pressionar simultaneamente as teclas "`Option + Command + O + F`". E, para melhorar a situação, a entrada e saída default do OpenFirmware é pela porta serial!!! Graças a Deus é possível manipular os parâmetros do OpenFirmware diretamente pelo Linux (comando `nvsetenv`)

A primeira alteração que vamos fazer é para o sistema entrar automaticamente no OpenFirmware e não tentar carregar o Linux/MacOs/Whatever direto. Para isso basta:

```
# nvsetenv auto-boot? false
```

Logo em seguida, vamos configurar as entradas e saídas para os locais corretos

```
# nvsetenv input-device kbd
# nvsetenv output-device video
```

Geralmente isso funciona, no 5500, o `output-device` precisa ser `"/bandit/ATY,264GT-B"`. Colocar o `boot-device` para a sua partição /boot é uma boa idéia também:

```
# nvsetenv boot-device ata/ata-disk@0:0
```

De novo, no caso do 5500 precisei setar algumas coisinhas extras:

```
# nvsetenv use-nvramrc? true
# nvsetenv nvramrc \
      "cpoke 0a7 0f3000032 \
      cpoke 093 0f3000033 \
      cpoke 03e 0f300003a"
```

O primeiro serve para carregar uma `nvramrc`. A segunda linha seta o conteúdo da `nvramrc` e deve ser toda digitada em uma linha só, sem as `\` (esses comandos servem para ligar a tela, não contentes em fazer a saída padrão ser o console serial, os engenheiros da Apple deixaram o brilho do monitor no mínimo!!!)

Dê uma olhada em:

```
http://penguinppc.org/bootloaders/\
            quik/quirks.php
http://www.netbsd.org/Ports/macppc/\
            models.html
```

Se quiser ver exatamente o que é necessário setar na nvram para a sua máquina. Ah! Se você fizer alguma besteira horrível, ”`Option + Command + P + R`“restauram a configuração original do OpenFirmware, se mesmo assim não der, dentro do seu Mac tem um botãozinho redondo, ele chama CUDA e se você apertá-lo, seu Mac volta para configuração de fábrica.

3.5. Instalando o QUIK

Não, não é um achocolatado (ou seria "amorangado"?), `quik` é um bootloader para ser utilizado em máquina PPC. No universo do PC seria uma espécie de LILO ou GRUB. É o conjunto `quik` e OpenFirmware que vai bootar a sua máquina.

Como o **`slackintosh`** era um hobby e, pior ainda (para a gente), o desenvolvedor tinha um Mac NewWorld, a instalação do `quik` deve ser feita de maneira totalmente manual.

O primeiro passo é criar um `quik.conf`, ele é bem parecido com um `lilo.conf` da vida, mas tem menos parâmetros. Crie-o diretamente na partição `/boot` que, no momento, está montada sob o `/mnt`:

```
# vi /mnt/boot/quik.conf
```

Edite o arquivo deixando-o mais ou menos assim:

```
timeout = 100
partition = 2
read-only
default = linux
image = /vmlinux
        label = linux
        root=/dev/hda3
```

Isso tudo quer dizer: espere 10 segundos (`timeout = 100`), depois procure no hda2 (`partition = 2`) por um arquivo chamado vmlinux (`image = /vmlinux`), se achar, por favor carregue-o. Bem simples não é?

Ainda não é a hora de rebootar. Antes é necessário copiar outro kernel para o diretório `/mnt/boot`. Infelizmente, o que está lá o `quik` não consegue carregar, ele tem problemas com kernels muito grandes. Felizmente, dentro do CD, no diretório `boot` existe um outro kernel -:). Copie-o para o `/mnt/boot`:

```
# cp /var/log/mount/boot/vmlinux \
                        /mnt/boot/vmlinux
```

Agora é hora de um pouco de black magic:

```
# cd /mnt
# pivot_root . mnt/floppy
# exec chroot . sh
# umount /mnt/floppy
# quik -v -C /boot/quik.conf
```

E pronto! Se tudo correu bem, agora você pode rebootar a sua máquina com o bom e velho "`shutdown -rf now`". Lembre de desmontar todas as partições e rodar o `sync`.

4. Finalizando

Se tudo deu certo, é para você agora estar cara-a-cara com o prompt do OpenFirmware, um:

```
0 >
```

Extremamente intimidador. Simplesmente escreva "`boot`" e aperte `<enter>`. É para aparecer agora um novo prompt, escrito "`boot:`". Neste prompt, ou você aguarda 10 segundos ou simplesmente escreve "`linux`" (que é o label que a gente deu para o kernel) e aperte `<enter>`.

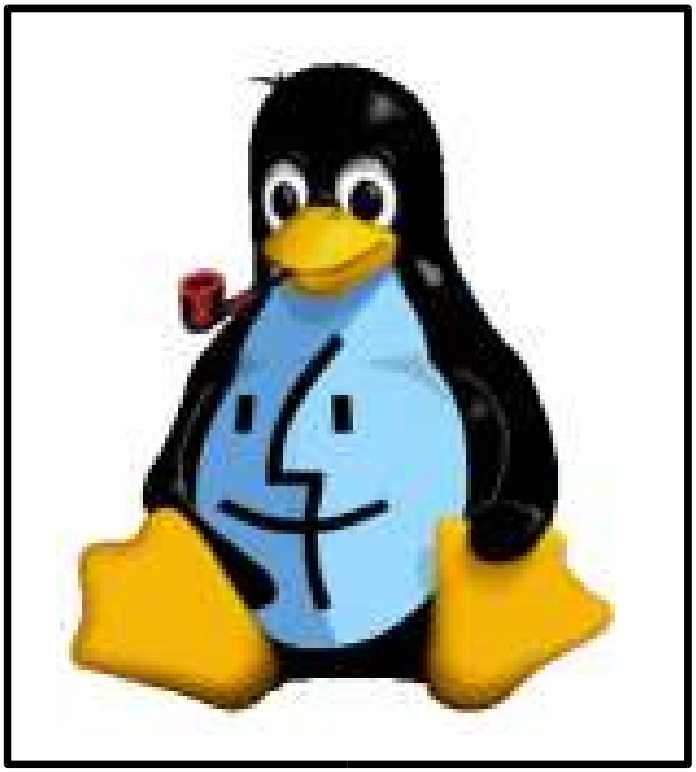
Dentro de instantes é para aparecer as mensagens do kernel correndo pela tela, e o simpático pingüim no canto superior esquerdo da tela. Logue como `root` e reinicie a máquina, vamos avisar ao OpenFirmware que não precisamos mais que o prompt dele fique aparecendo antes de todos os nossos boots, para isso basta digitar no `0 >`

```
0 > setenv auto-boot? true
0 > boot
```

Antes que eu me esqueça, o kernel que acabamos escolhendo não tem várias coisas importantes, como o suporte a disquetes e à rede. Para deixar a máquina realmente usável, recompile o kernel e "abuse" dos módulos, para tentar reduzir ao máximo o tamanho do kernel que será carregado.

Agora sim, boa sorte!!! -:)

Piter PUNK <piterpk@terra.com.br>



dnsmasq

Pague um e leve dois

Estes tempos me pediram para transformar um 486 em um servidor DHCP/DNS. Como as requisições destes protocolos são pequenos e não tão frequentes numa rede local, não exigiria muito da máquina. Porém ao instalar o **slackware** na máquina, vejo passar a descrição de um pacote: dnsmasq (small DNS and DHCP server).

O dnsmasq é um servidor simples de configurar e leve. Ele é implementado para fornecer resolução de nomes, mas também faz o papel de DHCP. Isto era perfeito para a rede em questão: local, com poucas máquinas, que não deveriam ser divulgadas no DNS oficial, e amarradas por MAC.

Está disponível na série N do **slackware** 10.0.
O dnsmasq pode usar o servidor de DNS oficial ou o arquivo /etc/hosts. No meu caso, eu preferi criar as entradas para as máquinas no arquivo hosts. Quando o dnsmasq inicia, lê estas entradas e armazena.

O arquivo de configuração é dnsmasq.conf, e o principal serão as linhas onde você pode configurar os hosts. Por exemplo a linha:

```
dhcp-host=\
       00:0c:6e:fd:6b:5e,172.22.74.1,sulamita
```

especifica que o host com MAC 00:0c:6e:fd:6b:5e deverá receber o ip 172.22.74.1 e o nome sulamita. Isto é o que a documetação diz, porém nos testes não consegui, tive que especificar tudo no hosts.

Com isto pronto, você já tem um servidor DHCP e DNS básico. Porém claro que queremos incrementar. Por exemplo, você pode querer informar para as máquinas qual o servidor de e-mail, atraves do campo "mx-host". Existem também algumas opções que podem fazer alguma diferença no desempenho da rede. A opção "filterwin2k", que irá ignorar as requisições geradas periodicamente e sem necessidade do W2k. Mas ainda tem mais, se você quiser que aquele servidor resolva a sua rede e nada mais, utilize o campo "no-resolv". Assim, se ele não souber responder alguma coisa, ele não irá buscar o servidor de DNS principal e tentar resolver. Para restringir ainda mais, use a opção "local=/redelocal/", que irá responder apenas a queries originadas desta rede.

A interface em que o dnsmasq irá ouvir também é importante quando este servidor pode estar ligado a mais de uma rede e você não quer que ele sirva a uma das redes, especificando as interfaces atraves da opção "interface=ethX".

Se a linha com esta opção estiver comentada, significa que ele irá ouvir em todas as interfaces. Você pode repetir esta linha para as interfaces que quer atender, ou especificar a(s) que não quer com o "except-interface".

Talvez você não queira misturar seu próprio arquivo hosts com os hosts para o dnsmasq. Você pode criar então estas entradas em um outro arquivo e usar a opção “addn-hosts=/onde/está/seu/arquivo". O domínio pode ser especificado com "domain=seudominio", e a faixa de IPs a serem oferecidos através do "dhcp-range". No dhcp-range você pode especificar apenas a faixa:

```
dhcp-range=172.22.1.1,172.22.255.253
```

ou além disto a máscara de rede padrão para as máquinas, e qual o prazo de validade da requisição:

```
dhcp-range=\
      172.22.1.1,172.22.255.253,255.255.0.0,12h
```

Aqui especificamos a faixa de ip de 172.22.1.1 a 172.22.255.253, máscara de rede 255.255.0.0, e a validade disto para 12 horas.

O dnsmasq utiliza opções da RFC 2132 para uma resposta DHCP. Alguns comuns são rota padrão(3), endereço de broadcast(28), servidor NTP(42). Então você pode enviar para as máquinas a rota padrão atraves de uma linha assim:

```
dhcp-option=3,172.22.255.254
```

O dnsmasq pode servir também apenas como um "proxy" dns, onde você pode querer apenas a parte do dhcp e usar o DNS de um servidor já existente. Você pode enviar o endereço deste servidor para as máquinas cliente através da linha

```
dhcp-option=6,172.22.1.1
```

e utilizar a opção "no-hosts".

Existem muitas mais opções, estas são as que achei mais interessantes. É claro que por ser pequeno o dnsmasq tem suas limitações, não possuindo todas as opções de um servidor DNS ou DHCP padrão, porém dependendo da situação é ótimo para organizar a rede e reutilizar aquele velho micro encostado. Coisas toscas também são legais e muito úteis.

Sulamita Garcia <toskinha@gmail.com>

FileSystems Criptografados

Buenas povo, o objetivo desde breve documento é descrever o processo para criação e utilização de um volume cryptografado utilizando a `cryptoapi` do kernel o mesmo pode ser feito em um arquivo no disco ou em uma partição normal.

Para os felizes usuários da release 2.6 do kernel não existe muito o que mexer, só configuração do kernel e patchear e recompilar o `util-linux`, já no caso do 2.4 é preciso também patchear o kernel. Então mãos à obra.

1. Kernel patch

Se você usa kernel da série 2.6.x, pode pular direto para próxima seção, caso contrário, baixe o arquivo `patch-cryptoloop-jari-2.4.22.0` na URL abaixo:

```
http://www.kernel.org/pub/\
      linux/kernel/crypto/v2.4/testing/
```

e aplique, o patch acima foi testado no kernel 2.4.25 e 2.4.26 aplicando em ambos com um pequeno hunk de algumas linhas no `config.in`. Lembre sempre que as linhas terminadas com \ no final devem ser todas digitadas como sendo uma única linha (e sem os \ ).

Para aplicar o patch e só seguir a velha receita.. entre no diretório onde estão os fontes do kernel e então execute:

```
# patch -p1 < \
      /caminho/completo/\
      patch-cryptoloop-jari-2.4.22.0
```

2. Configuração de kernel

Selecione as opções `CONFIG_BLK_DEV_LOOP` e `CONFIG_BLK_DEV_CRYPTOLOOP`, ambas se encontram na seção de `Block Devices` da configuração do kernel e também a opção `CONFIG_CRYPTO` e mais todas opções dos algoritmos de criptografia a serem usados, (recomendo `AES` ou `TWOFISH`), estas opções se encontram em `Cryptographic options`. Fica a gosto do freguês escolher entre compilar essas opções como módulos ou então builtin no kernel. Compile o kernel com as novas opções e instale-o assim como os módulos necessários e tudo mais.

3. Util-linux

Mais uma vez temos diferenças aqui na implementação para kernel 2.4 e 2.6, é necessário um patch extra para que compile num sistema com os kernel headers do 2.6, atente que o patch é somente necessário em sistemas com os headers do kernel 2.6 instalados! Se este for o seu caso, o patch pode ser encontrado em:

```
http://www.ece.cmu.edu/~rholzer/\
      cryptoloop/\
      util-linux-2.12-kernel-2.6.patch
```

e além desse tenho o patch para o `cryptoloop` via `cryptoapi` em si, este arquivo é o `util-linux-2.12-cryptoapi-losetup.patch.bz2` e é encontrado em:

```
http://gentoo.seren.com/gentoo/distfiles/
```

Os dois patches são aplicados nos mesmo moldes do patch para o kernel que aplicamos anteriormente. Baixe os fontes do `util-linux` em:

```
http://www.kernel.org/pub/linux/utils/\
      util-linux/util-linux-2.12a.tar.gz
```

descompacte os fontes com o comando:

```
# tar zxvf util-linux-2.12a.tar.gz
```

Aplique o patches necessários (considerando que os patches estão no diretório logo abaixo do diretório dos fontes do `util-linux`):

```
# cd util-linux-2.12a
# patch -p1 < \
      ../util-linux-2.12-kernel-2.6.patch
```

(relembrando, este patch somente é necessário em sistemas com kernel headers da versão 2.6)

```
# bunzip -c \
      ../util-linux-2.12-cryptoapi-*.bz2 \
      | patch -p1
```

Com os patches aplicados basta compilar e instalar os mesmos o que é feito com:

```
# make
# make install
```

E voilá, seu sistema esta pronto para utilização de volumes criptografados.

4. Criação dos volumes criptografados

Com tudo compilado e rodando é só criar os arquivos ou partições para armazenar os dados criptografados. Vou abordar a criação de um arquivo em disco pois o mesmo engloba a mesma configuração que seria utilizada para uma partição criptografada.

Inicialmente precisamos criar um arquivo com o tamanho que desejamos para o volume criptografado.

Isso pode ser facilmente criado com o comando `dd`, usaremos como fonte de dados o `/dev/urandom` por ser mais rápido (apesar de menos randômico) que o `/dev/random`. Usamos essas fontes ao invés do tradicional `/dev/zero` por que o arquivo cheio de dados randômicos dificulta muito a criptoanálise dos dados (sim, estamos sendo paranóicos), o comando para criação do arquivo é:

```
# dd if=/dev/urandom of=volume \
      bs=1M count=50
```

O comando acima cria um arquivo chamado `volume` com 50M, os parâmetros são:

- `if` - arquivo/device que será utilizado como origem de dados
- `of` - arquivo/device destino dos dados
- `bs` - block size, quantidade de dados acumulada antes de ser enviado um bloco de dados para o arquivo/device, aceita valores numéricos em bytes e operadores K para kilobytes e M para megabytes G para gigabytes e assim por diante
- `count` - é a contagem de blocos a serem escritos/lidos, no caso do comando acima 50, como o tamanho do bloco eh de 1M teremos um arquivo de 50M.

Agora devemos vincular o arquivo/device a um device `loopback` com o algoritmo de criptografia desejado, antes porém no caso de termos compilado os algoritmos e loopdevices como módulos iremos carregá-los:

```
# modprobe cryptoloop
# modprobe aes
```

Agora vincularemos um loop device e o algoritmo de criptografia ao arquivo/device que acabamos de criar (o arquivo `volume`). Utilizaremos como algoritmo de criptografia o AES. Se você não usou módulos no seu kernel, não se preocupe, não é o carregamento do módulo que indica o tipo de criptografia e sim a linha de comando que revelaremos a seguir...

Patrick,

all slack-users send our best wishes and hope you get better soon!!

Temos também que escolher o tamanho do bloco a ser usado com o algoritmo, blocos maiores são mais "fortes" mas são mais lentos e requerem mais memória para a decodificação, blocos de 128, 256, ou 512 são os mais recomendados oferecendo grande segurança e uma ótima relação segurança/velocidade, irei criar o volume com bloco de 128 bits o comando para isso é:

```
# losetup -e aes-128 /dev/loop0 volume
```

O comando pedirá para ser entrada uma senha que será a senha para acesso ao arquivo/device, essa senha deve ser segura para garantir o sigilo dos dados e deve-se lembrar que essa senha não poderá ser alterada ou recuperada, em caso de necessidade de nova senha deverá se criar um novo volume e se copiar todos os dados para o novo volume. Os parâmetros do comando acima são:

`-e aes` - algoritmo de criptografia a ser usado mais o tamanho do bloco
`/dev/loop0` - device de loopback a ser usada, existem 8 disponíveis do 0 ao 7
`volume` - arquivo ou device que será vinculado ao loopback

Após vinculada o device loopback poderá ser manipulado como um block device comum, por exemplo vamos criar um sistema de arquivos nele para podermos transferir arquivos para o mesmo:

```
# mke2fs /dev/loop0
```

Isso criará o sistema de arquivos padrão `ext2` no arquivo/device criptografado, agora é só montar ele no local de sua preferência e usá-lo a vontade, algo como:

```
# mount -t ext2 /dev/loop0 /mnt/crypt
```

Copie arquivos normalmente para dentro do volume, edite, enfim trate como uma partição normal.
Para desmontar e "esconder" o conteúdo do mesmo deve-se primero desmontar o device:

```
# umount /mnt/crypt
```

E depois desvinculado do loopback device:

```
# losetup -d /dev/loop0
```

Pronto, tudo que se poderá ver agora é um monte de bytes perdidos sem sentido ;) para voltar a montar o device deve-se proceder novamente como no começo:

```
# losetup -e aes-128 /dev/loop0 volume
```

E entrar a mesma senha e então voltar a montá-lo e tudo mais. Nestes tempos de Hds removíveis, chaveiros USB e outras mídias “móveis”, ter como proteger seus dados sempre é importante. Espero que o artigo tenha sido útil.

Boa sorte e have fun!!!

Thomas a.k.a. Wolvie
<wolvie@unitednerds.org>

Autores

Piter PUNK, é mantenedor e principal desenvolvedor do slackpkg. Possui experiência com UNIX e Linux desde '96 tendo escrito diversos artigos em revistas da área, atualmente, trabalha como desenvolvedor de jogos na 3WT Corporation.

Wolvie a.k.a Thomas, usa linux desde 98 com uma (BEM) rapida passagem por algumas distribuições foo até encontrar a luz, sobreviveu a 2 anos do curso de m.p.c.e na FATEC-SP e hoje trabalha (ou finge que) com linux e infra-estrutura de redes. dizem as más linguas também que dá pitacos em C, python, shell script e asm-x86. toca guitarra mal e porcamente e tenta tocar um banda de som não definido.

Clayton Eduardo dos Santos, trabalha com Linux a cerca de um ano e meio e com Slackware a cerca de um ano. É matemático, mestre em Engenharia Elétrica pela USP de São Carlos e atualmente desenvolve seu projeto de pesquisa de Doutorado no Departamento de Engenharia Elétrica na USP de São Carlos , utilizando ferramentas 100% baseadas em software livre.

Sulamita Garcia a.k.a. Toskinha, trabalha com linux desde 1999 e com Slackware desde 2001, como Analista de Suporte, É mantenedora do site de Alta Disponibilidade na UnderLinux. Atualmente participa do projeto LinuxChix-Br, e trabalha como Projetista de Software da Cyclades Corporation.

Diego Fiori Carvalho, é aluno do Bacharelado em Informática do ICMC-USP, São Carlos S.P.,utiliza Linux desde 2002, desenvolveu um sistema de multimídia interativa para Treinamento em Linux, tem grande interesse por desenvolvimento de aplicações gráficas e atualmente é desenvolvedor da empresa 3WT de São Carlos.

Deives Michellis "thefallen", Tecnólogo em Processamento de Dados pela FATEC/SP e Gerente de Desenvolvimento de Soluções Linux do Grupo GEO. Também nerds de carteirinha e ativista linux nas horas vagas.

Eduardo Costa Lisboa a.k.a. Formiga, é um sujeito discreto e avesso à publicidade. Ou talvez não tenha lido e-mails do dia 28/11 para 29/11 para mandar o mini-currículo