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

02 de Junho de 2006 – Edição #13

Editorial

Finalmente uma nova edição do slackwarezine! Estávamos sem uma nova edição completa desde novembro do ano passado (quando a edição de novembro saiu em dezembro). Ou seja, justo no nosso aniversário de dois anos (janeiro de 2006) não saiu edição nenhuma -:(

Para compensar, pensamos em fazer uma edição dupla para março... que virou uma edição tripla para maio (e nesse meio tempo lançamos a #12.5 para o FISL). E, a medida que os planos ficavam mais ambiciosos, menos a zine ficava pronta.

Bom, de volta a sanidade; temos uma zine cheia do mesmo de sempre, do jeito que os slackers gostam: a revista de artigos técnicos, escrita por técnicos e para técnicos.

Instalação de softwares (mldonkey, trac), dicas de configuração (google talk), instalação de hardware (D-Link DWL-G510, SiI 3112A), administração da máquina (reordenar eths)... o mesmo cardápio que sempre tivemos e agora com um gostinho de "ressureição"

Nossos agradecimentos para todos os colaboradores que continuaram mandando artigos para a zine e para os leitore que ficavam perguntando "Quando sai a próxima?" no ICQ, no e-mail e no IRC. Essa edição é para vocês! -;)

Boa Leitura!

Piter PUNK

Índice

slackware to the real nerds

slack users

Instalando a placa PCI Wireless D-Link DWL-G510 (Rev. B)

Esta placa é uma boa opção para quem quer comprar uma que suporte o protocolo como WPA-PSK que não necessita de um servidor de autenticação (RADIUS) configurado em casa, e uma taxa máxima de transferência de 54 Mbps sem gastar muito.

Apesar da D-Link não disponibilizar um driver para GNU/Linux, é possível instalar este hardware usando madwifi que é um projeto que provê drivers para placas Wireless que usam chipsets da Atheros, e wpa_supplicant que adiciona o suporte ao protocolo WPA-PSK (WPA-Pre Shared Key) ao driver do madwifi.

Este roteiro ensina como configurar um **slackware** 10.2 com uma placa dessa como uma estação cliente de uma rede Wireless. Este roteiro deve funcionar também com outras placas que possuem chipset da Atheros. Consulte toda a documentação dos sites no final do documento para mais informações.

Arquivos necessários:

- `madwifi-ng-current.tar.gz` que pode ser baixado do seguinte endereço: `http://snapshots.madwifi.org/`
- `wpa_supplicant-0.4.7.tar.gz` que pode ser baixado do seguinte endereço: `http://hostap.epitest.fi/releases/`

Requerimentos para instalar o madwifi:

- `uudecode` (presente no pacote `bin-10.2-i486-1.tgz`);
- `kernel-headers-2.4.31-i386-1.tgz` e `kernel-source-2.4.31-noarch-1.tgz`;
- Suporte ao Wireless Extensions (versão recomendada: v17) - kernel compilado com a opção `CONFIG_NET_RADIO=y` (Por padrão essa opção já vem configurada. Examine seu `/boot/config` para verificar se há suporte ou não, e recompile se necessário);
- Suporte ao Crypto API - kernel compilado com a opção `CONFIG_CRYPTO=y` (mais uma vez, verifique seu `/boot/config`. Use o comando "`grep CRYPTO /boot/config`", por exemplo);
- `gcc` (mesma versão usada que compilou o kernel do Linux);
- Kernel recomendado: kernel versão 2.4.20 ou superior;

Extraia o arquivo `madwifi-ng-current.tar.gz` em "`/usr/src/`", por exemplo. Entre no dir. criado e dê um "`make && make install`". Os módulos serão criados e instalados. Agora vamos instalar o `wpa_supplicant`.

Requerimentos para instalar o wpa_supplicant

- kernel versão 2.4.x ou 2.6.x com suporte ao Wireless Extensions;
- arquivos do madwifi;

Extraia o arquivo `wpa_supplicant-0.4.7.tar.gz` e entre no dir. criado. Crie um arquivo chamado "`.config`" (sem aspas) e adicione as seguintes linhas ao arquivo:

```
CONFIG_DRIVER_MADWIFI=y
CFLAGS += -I/usr/src/madwifi-ng
CONFIG_CTRL_IFACE=y
CONFIG_DRIVER_WEXT=y
```

Substitua `/usr/src/madwifi-ng` pelo local onde você extraiu os arquivos do `madwifi`. No meu exemplo, eu extrai em `/usr/src/madwifi-ng`. Dê um "`make`" para compilar os programas. E quando terminar, copie os arquivos `wpa_supplicant` e `wpa_cli` para "`/usr/local/bin/`".

Agora falta criar e configurar o arquivo de configuração do `wpa_supplicant`. Para isso usamos o comando o `wpa_passphrase` como root. A sintaxe é a seguinte.

```
wpa_passphrase SSID_da_Lan \
            CHAVE_PRE_COMPARTILHADA
```

No meu caso:

```
# wpa_passphrase homelan \
      xxxxxxxxxxxxx > \
      /etc/wpa_supplicant.conf
```

Observe que a saída do comando foi redirecionada para o arquivo `/etc/wpa_supplicant.conf`. Substitua o `xxxx`... pela sua chave compartilhada. Visualize o arquivo. Deve ser parecer como este:

```
network={
        ssid="homelan"
        psk="xxxxxxxxxxxxxx"
}
```

O tamanho da chave depende de quem configurou o ponto de acesso ou roteador wireless. Use a mesma chave que você usou no campo PreSharedKey quando configurou seu roteador para o protoclo WPA-PSK. Quanto maior a chave melhor. Você não precisará digitar outra vez essa chave.

Quando a chave for alterada no roteador Wireless, repita o comando acima já substituindo pela nova chave. Adicione as seguintes duas linhas ao arquivo `/etc/wpa_supplicant.conf`:

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=wheel
network={
        ssid="homelan"
        psk="xxxxxxxxxxxxxxx"
}
```

ATENÇÃO! ALTERE AS PERMISSÕES DO ARQUIVO:

```
# chmod 640 /etc/wpa_supplicant.conf
```

Carregue os módulos necessários (`ath_pci`, `ath_hal` e `wlan`) com `modprobe`. Crie um script contendo as seguintes linhas:

```
#!/bin/bash
wlanconfig ath0 create wlandev \
      wifi0 wlanmode sta
wpa_supplicant -Bw -Dmadwifi -iath0 \
      -c/etc/wpa_supplicant.conf
```

Altere as permissões, e execute-o como root. Dentro de alguns segundos sua placa vai se conectar à rede cujo SSID você definiu no "`/etc/wpa_supplicant.conf`" se você digitou a chave corretamente. Agora falta pegar o IP. Como root:

```
# dhcpcd ath0
```

Confirme com `/sbin/ifconfig` mesmo. Para verificar as configurações da placa Wireless, experimente o `iwconfig`. Outros comandos interessantes são `wpa_cli` (para permitir que usuários não-root alterem configurações do `wpa_supplicant`). Por favor, não vá se conectar na rede do seu vizinho! Leia os manuais do `madwifi` e do `wpa_supplicant` para outros comandos interessantes e ajustes finos.

Toda a documentação acima foi retirada dos seguintes site/documentos:

- `http://madwifi.org/`
- `http://hostap.epitest.fi/\ wpa_supplicant/`
- `http://madwifi.org/wiki/UserDocs/\ 802.11i`

Eduardo Braga <z41d@click21.com.br>

GoogleTalk e **slackware**

O Google lançou um IM (instant messenger) próprio. A boa notícia é que seguiram um protocolo aberto, o do Jabber (`http://www.jabber.org`), portanto, para aqueles que querem utilizar mais este IM no **`slackware`** basta configurar qualquer software que tenha suporte ao Jabber, como o `kopete`.

Neste caso apenas certifique-se de ter um crypto-plugin do Qt instalado, que é o `qca-tls`. No **`slackware`** não há esta biblioteca, portanto é necessário que a instalemos antes de configurarmos o acesso ao IM em questão.

Faça o download, extraia os arquivos e depois compile:

```
$ wget http://delta.affinix.com/\
      qca/qca-tls-1.0.tar.bz2
$ tar xjf qca-tls-1.0.tar.bz2
$ cd qca-tls-1.0
$ ./configure --qtdir=/usr/lib/qt
# make && make install
```

Para quem quiser criar um pacote (tgz) ou não quiser compilar a biblioteca manualmente, disponibilizei um script ao estilo do Slackware, ou seja, um SlackBuild no seguinte local:

```
http://www.faleiros.eti.br/\
      SlackBuild/qca-tls
```

Neste caso, apenas execute o script:

```
$ chmod +x qca-tls.SlackBuild
# ./qca-tls.SlackBuild
```

Feito isto, podemos (agora) configurar o `kopete` para acessarmos o Google Talk:

Vá em "Settings", depois em "Configure Kopete" e "New Account". Selecione o "Jabber" (em Messaging Services) e adicione seu e-mail do Gmail e senha. Em seguida selecione a aba "Connection", marque a opção "Use SSL" e em "Override default server" altere/adicione "talk.google.com".

Agora é só conectar.

Herbert Faleiros aka ratmmmam
<herbert@faleiros.eti.br>

Desenvolvimento Organizado com Subversion e Trac com PostgreSQL no **slackware**

Introdução

Quem estuda ou trabalha com desenvolvimento sabe, pelo menos teoricamente, da importância da Gerência de Configuração de Software(SCM, do inglês Software Configuration Management). O problema é que na vida real das empresas, onde eficiência e qualidade são objetivos diários, controlar processos e as atividades relacionadas ao desenvolvimento de software é de suma importância para se alcançar um software de qualidade.

Num processo de desenvolvimento de software temos que lidar com quantidades de documentos e códigos muito grande e a organização, o versionamento e as mudanças destes devem ser registradas com precisão e estarem disponíveis (acesso rápido e fácil) para diversas pessoas envolvidas no projeto.

Com a popularização do software livre e dos processos bastante organizados de vários projetos open source, hoje podemos contar com diversos softwares de grande qualidade e acessíveis, utilizados em Gerência de configuração de software. As software houses não têm mais desculpas para não desenvolverem softwares com qualidade nos processos.

Neste artigo vamos implementar o Subversion para Controle de Versão e o Trac com PostgreSQL para o Controle de Mudanças. O Subversion é uma ferramenta de Controle de Versão que foi construída para substituir o CVS, mas com as devidas melhorias nos pontos fracos do CVS, como por exemplo controlar a versão dos diretórios, de cópias e de renomeações. O Trac é um ferramenta com interface web, escrita em Python, de Controle de Mudanças em projetos de software, que se integra ao Subversion e oferece apoio à documentação e ao acompanhamento do projeto.

Por quê Trac com PostgreSQL? Na implementação padrão o Trac utiliza SQLite, mas enfrentei alguns problemas para utilizar o Trac com SQLite, nas instalações que fiz o Trac só funcionava com uma determinada versão da SQLite. Achei o problema muito limitante então parti para o PostgreSQL, apesar da SQLite ser mais flexível para backups e restores, o PostgreSQL é bastante utilizado e acho que não deixa nada a dever ao propósito do artigo.

Este artigo é apenas uma breve introdução de como fazer o Subversion mais o Trac com PostgreSQL funcionar no **slackware**. Portanto para outros recursos, como controle de acesso ao repositório e ao Trac e toda a administração deste sistema deve ser buscado em outras fontes. Comece pela documentação oficial onde você encontrá todos os passos e informações necessárias para extrair o máximo destas excelentes ferramentas.

Utilizaremos como base de instalação o **slackware** 10.2.

Instalando Dependências

Indicarei aqui apenas as dependências básicas necessárias à implementação do proposto:

python-2.4.1
Interpretador da linguagem Python. Disponível no CD 2 do **slackware** 10.2.

postgresql-8.1.3
Banco de Dados que armazena as informações do Trac. Qualquer versão acima de 7.3.X deverá funcionar(testado com as versões 7.4.8 e 8.1.3). Pacote disponível:

```
http://www.linuxpackages.net/\
            pkg_details.php?id=8189
```

swig-1.3.28
SWIG (Simplified Wrapper and Interface Generator) é uma biblioteca que liga programas escritos em C e C++ com outras linguagens, no nosso caso o Subversion e o Python (Subversion SWIG bindings). Esta é a dependência mais crítica, pois para podermos habilitar o suporte ao Python no Subversion devemos ter um casamento perfeito entre o `swig` e o Subversion. Evite mudanças neste quesito. Pacote disponível:

```
http://www.slacky.it/download/\
      development/swig/1.3.28/\
      swig-1.3.28-i486-1sl.tgz
```

Todas as dependências acima já estão no formato do **slackware**, para instalar basta um:

```
# installpkg NOME_DO_PACOTE.tgz
```

psycopg2-2.0b8
Interface que liga o Python com o PostgreSQL. Uma observação importante é que para ter uma ligação perfeita entre o Python e o PostgreSQL esta biblioteca deve ser compilada com a versão do PostgreSQL a ser utilizada, de preferência na própria máquina onde se encontra o PostgreSQL. Download disponível:

```
http://initd.org/pub/software/\
      psycopg/psycopg2-2.0b8.tar.gz
```

Para instalar o `psycopg2` descompacte o arquivo e execute:

```
# cd psycopg2-2.0b8
# python setup.py build
# python setup.py install
```

clearsilver-0.10.2
É um template engine para embutir diversas linguagens no HTML. Download disponíveis em:

```
http://www.clearsilver.net/\
      downloads/\
      clearsilver-0.10.2.tar.gz
```

Para instalar o `clearsilver` descompacte o arquivo e execute:

```
# cd clearsilver-0.10.2
# ./configure --prefix=/usr \
      --with-python=/usr/bin/python
# make
# make install
```

Apesar de acabarmos de instalar o `clearsilver` o que nos interessa ficou de fora.

É o módulo `neo_cgi.so` que habilita o Python a trabalhar com o `clearsilver`. Neste caso copiamos o mesmo para o local correto. Ainda dentro da pasta `clearsilver-0.10.2`, executamos:

```
# cp python/neo_cgi.so \
      /usr/lib/python2.4/\
      site-packages/
```

Instalando o Subversion e o Trac

Dependências instaladas, passamos aos protagonistas deste artigo.

subversion-1.3.0
É um sistema de Controle de Versão muito poderoso e eficiente.

Muitos devem ter notado que no **slackware** 10.2 o `subversion` se tornou parte da distribuição, é uma excelente notícia. O problema é que o binário official do **slackware** não suporta a ligação com o `python` via `swig`, ou seja temos que recompilar. Então caso tenha o `subversion` oficial instalado, remova-o desta forma:

```
# removepkg subversion-1.2.3-i486-1
```

Agora baixe os fontes da nova versão do `subversion`:

```
http://subversion.tigris.org/\
      downloads/\
      subversion-1.3.0.tar.bz2
```

Para compilar manualmente execute, após descompactar o arquivo:

```
# cd  subversion-1.3.0
# ./configure \
      PYTHON=/usr/bin/python \
      --prefix=/usr \
      --enable-shared \
      --disable-static --with-pic \
      --without-berkeley-db \
      --with-ssl \
      --with-zlib --with-swig
# make
# make swig-py
# make install
# make install-swig-py
```

Caso ocorra algum problema, poderá estar relacionado ao `swig`, reveja as dependências. Se persistir o problema tente uma outra versão do `swig`.

Ao executar `make swig-py` e `make install-swig-py` será instalada a ligação do Subversion com o Python via SWIG.

Contudo os módulos do python ficarão em um local não padrão para o nosso **slackware**. Para indicarmos ao python onde encontrar os módulos de ligação com subversion executamos:

```
# echo /usr/lib/svn-python \
      /usr/lib/python2.4/\
      site-packages/subversion.pth
```

Existem outras formas de fazer esta indicação ao python como setar as variáveis de ambiente PYTHON_SITE e PYTHONPATH , criar links simbólicos ou até mesmo copiar ou mover os módulos para o local apropriado. A maneira utilizada acima é a mais limpa.

Pronto compilamos e instalamos o "novo" subversion com ligação ao python via swig.

trac-0.9.4

É um sistema de controle de mudanças escrito em python que se integra ao subversion, oferencendo apoio à documentação e ao acompanhamento do projeto. Download disponível em:

```
http://ftp.edgewall.com/pub/\
      trac/trac-0.9.4.tar.gz
```

Descompacte o arquivo e execute:

```
# cd trac-0.9.4
# python setup.py build
# python setup.py install
```

Configurando um ambiente básico

Para configurarmos um repositório básico devemos conhecer previamente como funciona um sistema de Controle de Versão, especialmente o subversion. A primeira coisa a fazer é definir um esquema para o repositório. O esquema pode contar com um repositório para cada projeto ou um repositório abrigando vários projetos. Vamos escolher 1:1 e criar o nosso repositório: (daqui para frente substitua os caminhos/diretórios de acordo com o seu sistema)

```
$ svnadmin create \
      /home/wasare/projeto1
```

O comando acima criará um repositório no formato fsfs, padrão atual do subversion. No desenvolvimento de projetos em equipe devemos organizar o nosso repositório para que possamos ter várias "linhas de desenvolvimento" em paralelo sem que uma interfira na outra.

ATENÇÃO: O Subversion é baseado na biblioteca APR (Apache Portable Runtime library). O fonte já traz todos os arquivos necessários. O problema é que a biblioteca APR é compilada e instalada com outros programas também, por exemplo, apache2 e a própria apr e apr-util, caso tenha algum destes instalado, tome cuidado ao compilar o Subversion pois o mesmo substituirá a APR instalada, causando conflitos e problemas de funcionamento dos programas que dependem da biblioteca. O mesmo ocorre caso compile e instale outros programas que utilizarão a APR, depois do Subversion, neste caso é o Subversion que terá problemas. Apesar dos possíveis problemas, tudo pode ser resolvido na hora da compilação, passe os parâmetros certos e nada de problemas. Na compilação que faremos quando houver qualquer biblioteca APR instalada no sistema, será substituída. Para evitar este comportamento basta informar, durante o processo de compilação, onde está a APR que gostaria de usar. Acrescente os parâmetros --with-apr=/usr e --with-apr-util=/usr ao ./configure, sendo /usr o caminho base onde a APR está localizada (arquivos apr-config e apu-config).

Para um bom esquema convenciona-se criar três diretórios que norteam a organização do repositório e do andamento do projeto. Os diretórios são:

trunk

raíz do projeto ou linha principal de desenvolvimento, sua principal característica é que tudo nele deve estar estável.

branches

são as ramificações ou linhas paralelas à linha principal, normalmente cada desenvolvedor deverá ter um "branch". Estas ramificações, quando se tornam estáveis, são mescladas com a raíz.

tags

este contém algumas versões específicas que recebem um nome ou rótulo mais amigável como release-1, release-2, etc. Representa um snapshot ou cópia de um projeto em um determinado momento. O que está neste diretório nunca deve ser alterado.

Lembre-se que num sistema de controle de versão nada é perdido, tudo que é adicionado ou retirado é versionado e os arquivos/diretórios/cópias e suas versões continuam lá através do tempo, podem ser recuperados a qualquer momento, referenciando-os pelo número da versão.

Então vamos completar nosso esquema de repositório:

```
$ mkdir esquema
$ cd esquema
$ mkdir trunk branches tags
$ svn import . \
      file:///home/wasare/projeto1 \
      --message 'Esquema Inicial'
Adding         trunk
Adding         branches
Adding         tags
Committed revision 1.
```

Pronto temos nosso repositório preparado para funcionar. Agora podemos passar ao Trac. Para inicializar um projeto no Trac com PostgreSQL é necessário que já exista um banco de dados previamente criado, como não faz parte do escopo deste artigo tratar de detalhes do PostgreSQL, vamos considerar que já exista o famigerado banco com nome de trac_projeto1.

A configuração básica do Trac é bem simples, o utilitário trac-admin implementa um assistente que facilita o nosso trabalho, veja os passos e a descrição logo em seguida:

```
$ trac-admin \
      /home/wasare/trac-projeto1 \
      initenv
Project Name [My Project]> Projeto 1
Database connection string \
      [sqlite:db/trac.db]> \
      postgres://\
      user:senha@localhost/\
      trac_projeto1
Path to repository [/var/svn/test]> \
      /home/wasare/projeto1
Templates directory \
      [/usr/share/trac/templates]>
```

No comando inicial passamos como parâmetro o caminho completo (com o diretório) onde será criado o ambiente Trac para o projeto 1, o diretório também é criado no processo. Na sequência o assistente solicita os outros parâmetros como o nome do projeto, a string de conexão com o banco onde deverá ser substituido por valores corretos: user, senha e trac_projeto1 (usuário do banco, a senha e o nome do banco, respectivamente), o caminho para o repositório (o qual criamos anteriormente) e o diretório de templates o qual deixamos padrão (ENTER).

Colocando o Trac On line

O Trac é acessado via navegador web e existem três formas de colocá-lo on line: como servidor standalone (embutido no próprio Trac), via CGI ou fastCGI ou via mod_python do apache. O melhor método é via mod_python, contudo para simplificar utilizaremos o método mais simples e que já está disponível no Trac. O método de servidor standalone é bastante flexível e tem como principal vantagem a indepêndencia de servidor web e a velocidade, tão rápida quanto pelo mod_python, segundo a documentação. As desvantagens ficam por conta dos poucos recursos e pela falta de suporte à HTTPS. Para levantar o servidor execute:

```
$ tracd --port 8080 \
      home/wasare/trac-projeto1
```

Para outros projetos adicione ao comando à frente do último parâmetro o caminho completo para o outro projeto e assim por diante (Consulte a documentação para mais recursos).

Se não houve nenhum erro, aponte o seu navegador predileto para http://localhost:8080, serão listados os projetos on line. Selecione o projeto listado e clique no menu "Browse Source" e confira se vai aparecer uma listas com o esquema do repositório criado (branches, tags e trunk).

Palavras finais

O Trac não possui internacionalização oficial, existem apenas alguns templates traduzidos, você mesmo pode traduzi-los desde que saiba o que esteja fazendo. Se for traduzi-los ou pegar algum já pronto tome cuidado pois estes tem relação direta com a funcionalidade(lógica) do sistema e não é apenas na apresentação, e as alterações entre as versões são muitas. Os templates se encontram em /usr/share/trac/templates para todos os projetos ou em /home/wasre/trac-projeto1/templates para o projeto em questão, pode-se ter templates global ou local, este último com maior prioridade.

Agora você pode personalizar o ambiente Trac do seu projeto ao seu gosto e ter tudo muito bem organizado. Desta forma vai sobrar mais tempo para se concentrar apenas no desenvolvimento em si, nada de ficar mais gastando o seu precioso tempo com documentação das mudanças e dos arquivos do projeto.

Wanderson Santiago dos Reis
<wandersonsreis@gmail.com>

Instalando o mldonkey no **slackware**

Introdução

mldonkey é um cliente para compartilhamento de arquivos (P2P) que é capaz de acessar praticamente todas as principais redes existentes. Dentre estas redes cito as principais: Edonkey (ed2k), FastTrack (Kazaa), Bittorrent, Gnutella/Gnutella2 (LimeWire), Overnet, SoulSeek, etc. É considerado também o mais avançado cliente da ed2k (eDonkey).

O código fonte do último release estável encontra-se em:

```
http://download.berlios.de/pub/\
                  mldonkey/spiralvoice
```

Instalando

Após efetuarmos o download extraímos os arquivos:

```
$ tar xjf mldonkey-2.7.1.tar.bz2
```

Entramos no diretório com o código fonte:

```
$ cd mldonkey-2.7.1
```

No diretório "packages" há um SlackBuild para a construção de um pacote para o **slackware**, neste artigo descreveremos em detalhes vários pontos específicos que ainda não foram inclusos neste SlackBuild, portanto não iremos utilizá-lo. Fica então como dica para quem não quiser fazer tudo manualmente.

Em seguida geramos os scripts de configuração:

```
$ (cd config && autoconf)
```

Preparamos os arquivos para serem compilados e compilamos o código fonte:

```
$ echo yes | CFLAGS="-O2 \
      -march=i486 -mcpu=i686" \
      ./configure --disable-gui \
      --build=i486-slackware-linux
```

Por segurança não criaremos um binário com vínculos à bibliotecas do sistema:

```
$ gmake mlnet.static
```

Não é necessário no caso do **slackware** o "gmake install", pois o único arquivo que iremos utilizar é o binário "mlnet.static", portanto basta copiá-lo para o local onde iremos armazená-lo, ou seja /usr/bin:

```
# chown 0.bin mlnet.static
# chmod 755 mlnet.static
# mv mlnet.static /usr/bin
```

Por questões de compatibilidade podemos (opcionalmente) adicionar os links simbólicos listados no Makefile:

```
# cd /usr/bin
# for i in \
      mlslsk mldonkey mlgnut \
      mldc mlbtl; do \
      ln -sf mlnet.static $i; done
```

Finalizado o processo de compilação preparamos o sistema para que a execução do binário em questão seja mais segura e compatível com os padrões do Slackware.

3. Configurando

Criamos um usuário com ID alta e sem acesso à shell no sistema (segurança):

```
# useradd -u 1500 -s \
      /usr/bin/mlnet.static \
      -d /home/mldonkey mldonkey
# mkdir /home/mldonkey
# chown mldonkey.users \
      /home/mldonkey -R
```

Feito isto criamos um arquivo de configurações contendo diversas variáveis necessárias às regras de firewall e execução do script de incialização:

```
#!/bin/sh
#
# mldonkey.conf (Global variables for
# MLDonkey scripts).
# Herbert Alexander Faleiros
# <herbert@faleiros.eti.br>
#

USER=mldonkey
HOME=`cat /etc/passwd | \
      grep $USER | cut -d: -f6`

HOST=`cat /etc/HOSTNAME | \
            cut -d" " -f1`
HTTP_PORT=`cat $HOME/downloads.ini |\
            grep http_port | \
            tr -d "a-z=_ "`
```

```
ED2K_TCP=`cat $HOME/donkey.ini |\
      grep "port =" |tr -d "a-z= " |\
      sed -n 1p`
ED2K_UDP=`expr $ED2K_TCP + 4`

FASTTRACK=`cat $HOME/fasttrack.ini |\
      grep "port =" | tr -d "a-z=_ "`

C_BITTORRENT=`cat \
      $HOME/bittorrent.ini | \
      grep "client_port =" | \
      tr -d "a-z =_"`
T_BITTORRENT=`cat \
      $HOME/bittorrent.ini | \
      grep "tracker_port =" | \
      tr -d "a-z =_"`

GNUTELLA=`cat $HOME/gnutella.ini |\
      grep "port =" | \
      tr -d "a-z =_" | sed -n 1p`

G2=`cat $HOME/gnutella2.ini | \
      grep "port =" | \
      tr -d "a-z =_" | sed -n 1p`
```

Em nosso script de firewall acrescentamos o seguinte (ex. liberar ed2k):

```
if [ -x /etc/rc.d/rc.mldonkey ]; then
  . /etc/mldonkey.conf
  $IPTABLES -A INPUT -p tcp -i $EXIF \
      --dport $ED2K_TCP -j ACCEPT
  $IPTABLES -A INPUT -p udp -i $EXIF \
      --dport $ED2K_UDP -j ACCEPT
fi
```

Neste caso `IPTABLES` é `/sbin/iptables` e `EXIF` a interface de rede externa. Para liberar o acesso às demais redes basta seguir o exemplos acima. Em seguida criamos nosso script de inicialização:

```
#!/bin/sh
#
# rc.mldonkey (MLDonkey - Slackware
# style init script).
# Herbert Alexander Faleiros
# <herbert@faleiros.eti.br>
#

. /etc/mldonkey.conf

function mlstart() {
  echo "Starting MLDonkey."
  cd $HOME && su $USER &>/dev/null &
}

function mlstop() {
  OPTS="--spider --quiet"
  for CMD in commit save kill; do
    wget $OPTS \
      "http://$HOST:$HTTP_PORT/\
      submit?q=$CMD"
  done
}

case $1 in
  start)
    mlstart
  ;;
  stop)
    echo "Stoping MLDonkey properly."
    mlstop &>/dev/null
  ;;
  restart)
    mlstop; sleep 1; mlstart
  ;;
  *)
  echo "Usage: $0 start|stop|restart"
  exit 1
  ;;
esac
```

Para ativarmos o script (simulando o comportamento de um "daemon"):

```
# chmod +x /etc/rc.d/rc.mldonkey
```

Acrescentamos ao `/etc/rc.d/rc.local` o seguinte (se quiser que o MLDonkey seja executado durante a inicialização do sistema, claro):

```
if [ -x /etc/rc.d/rc.mldonkey ]; then
  /etc/rc.d/rc.mldonkey start
fi
```

E ao `/etc/rc.d/rc.6` (antes de executarmos o `hwclock`, por ex.) o seguinte:

```
if [ -x /etc/rc.d/rc.mldonkey ]; then
  /etc/rc.d/rc.mldonkey stop
fi
```

Verificamos agora as permissões dos arquivos:

```
# chown 0.0 \
      /etc/rc.d/rc.firewall \
      /etc/rc.d/rc.mldonkey \
      /etc/mldonkey.conf
# chmod 744 \
      /etc/rc.d/rc.{firewall,mldonkey}
# chmod 644 /etc/mldonkey.conf
```

Caso o `mldonkey` nunca tenha sido executado, inicie o "daemon" e o finalize em seguida (necessário para que sejam criados todos os arquivos utilizados pelos scripts descritos acima).

Feito isto execute o `/etc/rc.d/rc.firewall` e em seguida o `/etc/rc.d/rc.mldonkey`.

Como frontend sugiro a utilização do `kmldonkey`. Um `SlackBuild` para o mesmo se encontra aqui:

```
http://www.faleiros.eti.br/\
            SlackBuild/kmldonkey
```

Herbert Faleiros aka ratmmmam
<herbert@faleiros.eti.br>

Automatizando a Reordenação de 'N' Interfaces de Rede

Na rede que administro, o acesso à internet é feito através de 2 links dedicados de 1 Mbps com balanceamento de carga. Tive, por muito tempo, problemas de organização das NICs, já que sempre usei kernel monolítico em meus firewalls, para evitar backdoors e rootkits baseados em módulos, como o LKM, e, devido a isso, de nada adiantava configurar o arquivo /etc/modules.conf, tentando ordenar as placas, afinal, não existem módulos nessa situação: tudo está em um único bloco.

Certo dia, acessei o site da Slackwarezine e fiz o download das revistas que eu ainda não tinha. Para minha sorte, havia na edição 8.5o e na 9o uma matéria exatamente sobre o assunto. Pois bem, pus a mão na massa, seguindo o artigo, mas, mesmo depois de ler a errata da edição 9o, a troca de interfaces dava 'pau'. E aí?

Novamente mãos à obra, verifiquei que o nameif em conjunto com o arquivo /etc/mactab somente funcionavam, se apenas houvesse 2 placas de rede (a máquina tem três, uma em cada link e mais uma para a rede interna).

Depois de realizar várias tentativas, descobri que manualmente, duas a duas, funcionava a troca de índices das placas. Para isso, usei uma eth3 (fictícia), lembrando que minha máquina tem a eth0, eth1 e eth2, ou seja, usei uma eth'X' que estivesse livre, para servir, como se fosse uma variável auxiliar de troca (como no conhecido algoritmo de ordenação Bubble Sort).

Mas então, por que o 'nameif -s' com o /etc/mactab não funcionavam no meu sistema? Explicação: supondo que as interfaces que trocariam seus índices fossem a eth0 e eth1, o 'nameif -s' usaria a eth2, como variável auxiliar, para realizar a ordenação, porém essa eth2 não poderia existir fisicamente no computador. Como no meu firewall, havia a terceira placa de rede (eth2), ligada ao outro link internet, o 'nameif -s', ao invés de usar uma eth3 (livre), usava a eth2. Devido a isto, aparecia na tela uma mensagem de erro, informando que a eth2 estava ocupada e a ordenação não era realizada. A partir daí, para resolver o problema, decidi criar o pior cenário possível de tal forma que me obrigasse a trocar os índices de 'N' placas, passando a montar o script abaixo, para realizar a ordenação automática dessas 'N' placas.

Autores

Eduardo Braga, é Técnico Judiciário / Operação de Computadores da SJRJ (Justiça Federal) e presta suporte técnico a técnicos e usuários em geral. Usa GNU/Linux há algum tempo. Fã do Slackware. Atualmente, estudando Redes de Computadores na Estácio de Sá.

Herbert Alexander Faleiros (aka ratmmmam), programador, graduando em Física pela UFSCar. Seu primeiro contato com o Slackware ocorreu em 2000. Atualmente trabalha desenvolvendo soluções emJava para servidores de aplicações.

Nivio Souza. trabalha com Linux, desde 1999, e com Slackware, desde 2004. É aluno do 5° período do Curso de Ciência da Computação da Universidade Católica de Petrópolis e nerd nas horas vagas.

Yucatan "Kenjiro" Costa, Bacharel em Ciência da Computação e Pós-Graduado (Especialista) em Programação Avançada e Redes. Trabalha como Administrador de Redes da Escola Técnica Alto Jacuí. Árduo defensor do Slackware Linux, botou as mãos em um computador pela primeira vez em 1985 (um CP-500), mas só teve contato com Linux em 1997 (Slackware Linux 3.0)

Wanderson Santiago dos Reis, Estuda e trabalha com Linux e Software Livre desde 2000. É Técnico em Informática e atualmente cursa o 6º período de Graduação Tecnológica em Informática no Agronegócio no CEFET - Bambuí (MG), onde também trabalha com programação e soluções em Software Livre. Acredita que a adoção e desenvolvimento de Sistemas Livres por governos e empresas é um caminho seguro para a evolução do conhecimento.

```
#!/bin/bash
#
#---------------------------------------
#           Ordenador Automático
#          para 'N' Placas de Rede
#---------------------------------------
#
# rc.cheth version 1.0 is a changer
# name (eth'N') to rearrange NICs.
#     Copyright (C) 2006 - Nivio Souza
#           <capnivio@gmail.com>
#
# This script is free software; you can
# redistribute it and/or modify it under
# the terms of the GNU General Public
# License as published by the Free
# Software Foundation; either version 2
# of the License.
#
# This script is distributed in the
# hope that it will be useful, but
# WITHOUT ANY WARRANTY; without even the
# implied warranty of MERCHANTABILITY or
# FITNESS FOR A PARTICULAR PURPOSE.  See
# the GNU General Public License for
# more details.
#
# You should have received a copy of the
# GNU General Public License along with
# this script; if not, write to the
#
# Free # Software Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston,
# MA  02110-1301, USA

#
# Configuração
#
# Este script necessita ser chamado
através
# do arquivo /etc/rc.d/rc.local
#
# Apesar de outras fontes recomendarem o
# uso da chamada do 'nameif' no
# /etc/rc.d/rc.inet1, ele somente
# funcionou para kernel monolítico e
# também, para modular, quando chamado
# do rc.local.
#
# Portanto coloque a seguinte entrada no
# arquivo /etc/rc.d/rc.local:
#
# if [ -x /etc/rc.d/rc.cheth ]; then
#      . /etc/rc.d/rc.cheth
# fi

# Anunciando o servico
printf "Ordenando as placas de rede, \
conforme conveniência ..."

# Variáveis de ambiente
# Localização do aplicativo ifconfig
IFC="/sbin/ifconfig"

# MAC Address que você quer (DES -
# desejado)  na eth0
MAC_DES[0]="AA:AA:AA:AA:AA:AA"

# MAC Address que você quer na eth1
MAC_DES[1]="BB:BB:BB:BB:BB:BB"

# MAC Address que você quer  na eth2
MAC_DES[2]="CC:CC:CC:CC:CC:CC"

# MAC Address que você quer na eth'N'
MAC_DES[N]="FF:FF:FF:FF:FF:FF"

# MAC_ENC[X] = MAC Address encontrado
# na eth'X'

# Verifica os índices 'N' das interfaces
# de rede existentes (eth'N')
IETH=`$IFC | grep eth | /bin/cut -c 4`

# Verifica quais são os MAC Address
# reais que o sistema atribuiu as NICs
count=0
for i in $IETH
do
    MAC_ENC[$i]=`$IFC eth$i | \
              grep HWaddr | \
              tr -s " " | cut -f 5 -d " "`
    count=$((count+1))
done
#Ordena as NICs duas a duas
ordenar_ETHS()
{
    /sbin/nameif eth$count \
                              ${MAC_ENC[$i]}
    /sbin/nameif eth$i ${MAC_DES[$i]}
    /sbin/nameif eth$j ${MAC_ENC[$i]}
}
# Verifica quais trocas de MAC Address
# precisam ser realizadas (Bubble Sort)
for i in $IETH
do
  for j in $IETH
  do
    if [ "$i" != "$j" ]; then
       if [ "${MAC_DES[$i]}" == \
            "${MAC_ENC[$j]}" ]; then
                  $IFC eth$i down
                  $IFC eth$j down
                  ordenar_ETHS
                  AUX=${MAC_ENC[$i]}
                  MAC_ENC[$i]=\
                         ${MAC_DES[$i]}
                  MAC_ENC[$j]=$AUX
       fi
    fi
  done
done
# Se tudo deu certo, reinicia a rede
# com as placas reordenadas.
. /etc/rc.d/rc.inet1 restart

# Verificando, se as trocas foram
# efetuadas corretamente.
erro=0
for i in $IETH
do
    MAC_ENC[$i]=`$IFC eth$i | \
              grep HWaddr | \
              tr -s " " | cut -f 5 -d " "`
    if [ "${MAC_DES[$i]}" != \
         "${MAC_ENC[$i]}" ]; then
              erro=1
    fi
done

if [ $erro -eq 0 ]; then
    printf "  [  OK  ]\n"
else
    printf "  [FALHOU]\n"
fi
```

Configurando o Kernel para HD SATA e Controladora SiI 3112A (Asus A7N8X-DeLuxe)

Recentemente adquiri um HD SATA (um Samsung que não lembro o modelo) e queria colocá-lo pra funcionar em minha máquina, junto com os dois HDs IDE e dois drives (DVD e CDRW) que tenho. Bom, segundo me disseram funcionaria. Então resolvi botar a mão na massa.

Ah, favor notar que não estou falando em AID aqui. Estou apenas colocando um HD SATA para funcionar junto com outros dois HDs IDE.

Meu Hardware

- Mother Board -> ASUS A7N8X-Deluxe (controladora SATA Silicon Image SiI 3112A)
- HD Samsung (SATA)

Meu Software

- Slackware 10.2
- Kernel 2.6.14.3

Primeiro é necessário checar se o suporte a SATA está habilitado (fisicamente) em sua placa mãe. Verifique a posição do jumper SATA_EN1. Ele deve estar em '1 2' (qualquer dúvida verifique o manual da placa mãe).

Então agora vamos verificar a configuração do kernel.

```
cd /usr/src/linux
make menuconfig

-Device Drivers  --->
--ATA/ATAPI/MFM/RLL support  --->
--- <*> Silicon Image chipset support

--SCSI device support  --->
---SCSI low-level drivers  --->
----<*> Serial ATA (SATA) support
----<*>   Silicon Image SATA support
```

Compile o kernel com os comandos de sua preferência e vamos à configuração do lilo.

Por que mexer no `lilo`? Bom, no meu caso em particular tudo funcionou bem após a compilação do kernel e reinicialização do computador.

O problema é que meu **`slackware`** levava em torno de três (3) minutos para inicializar. Rídiculo não? Então resolvi prestar atenção nas mensagens da inicialização e vi que ele gastava mais de 2 minutos tentando detectar se havia um outro HD SATA na controladora secundária (o que não existia).

Por isso precisamos dar uma "fuçadinha" na configuração do `lilo` de forma a informar o kernel que NÃO EXISTE OUTRO HD SATA. Se na sua configuração de hardware houver outro HD SATA, desconsidere este próximo passo.

Edite o `lilo.conf` e insira a seguinte linha:

```
append="hdg=noprobe"
```

(se já houver um '`append`' no seu `lilo.conf`, apenas adicione aquele parâmetro no final da linha, deixando um espaço em branco entre os parâmetros)

Pronto, instale o `lilo` (com o comando '`lilo`'), reinicie a máquina e agora é só diversão ;)

Qualquer dúvida ou sugestão mande um email. Ah! Não esqueça de me avisar se conseguiste fazer alguma outro modelo ou marca de Mother Board funcionar.

May the Force be with you!

Yucatan "Kenjiro Tanaka" Costa
<yucatan@globo.com>