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

31 de Dezembro de 2005 – Edição #12

Editorial

Última edição do ano! E, até agora, o nosso recorde de atraso, se levarmos em consideração que essa é a edição de Novembro -;)

Para a virada do ano montamos um zime com artigos bem interessantes. Um sobre como tornar o processo de criação de pacotes mais automático utilizando scripts, agora não existem desculpas para não fazer seus próprios pacotes.

Na área de hardware, um pequeno tutorial de como instalar e utilizar no amsn uma webcam da D-Link.

Para os administradores, a instalação do rdesktop (para dar manutenção em máquinas Windows), um passo-a-passo simples e rápido da instalação do PostFix usando pacotes já existentes e uma bom artigo sobre como configurar uma máquina rodando slackware como cliente de uma rede com autenticação LDAP.

Para os iniciantes, um artigo de como calcular os "números" de permissão. Particularmente, acho muito mais fácil aprender a contar até sete em binário do que fazer as contas do artigo -;), mas, cada um tem a sua maneira de fazer as coisas, e essa é uma das vantagens do Software Livre, justamente a Liberdade!

Boa Leitura e Feliz 2006!

Piter PUNK

Índice

Automatizando a Construção de Pacotes
Herbert Faleiros — 2

Brincando com Bits
Nycholas Oliveira e Oliveira — 6

Instalando a D-Link DSB C110
Yukatan "Kenjiro" Costa — 8

Instalando e Usando o rdesktop
Clayton Eduardo dos Santos — 9

Autenticando o Slackware SEM PAM em uma base LDAP
Flávio do Carmo Júnior — 10

Instalando o PostFix
Fabiano Silva de Carvalho — 12

slackware to the real nerds

slack users

Automatizando a Construção de Pacotes

Arquivos `SlackBuild` são `shell scripts` cuja finalidade é a automatização de todo processo de compilação de pacotes para o **`slackware`**.

Estes `scripts` também podem ser utilizados para otimização, recompilação ou personalização de pacotes oficiais, ou ainda quando quisermos seguir os padrões do Slackware na construção de pacotes não-oficiais, bastando para isto editarmos poucos parâmetros e deixá-los (os `SlackBuild`'s) responsáveis por todo o trabalho pesado (configurar, compilar, personalizar, etc).

Este artigo pressupõe que o leitor esteja familiarizado com todo o processo de construção de pacotes para o **`slackware`**.

Como exemplo descreverei o `script` que utilizo para construir o pacote do `amaroK` (um áudio player para o KDE).

Estes arquivos seguem regras bem simples e possuem padrões (alguns não oficiais) a serem seguidos, os principais deles são:

1. Nomenclatura do arquivo `.SlackBuild`

Basta seguirmos o padrão "`nome_do_programa.SlackBuild`" (ex. `amarok.SlackBuild`)

2. Arquivo de descrição do pacote (`slack-desc`)

Devemos incluir também um arquivo (necessário à construção do pacote) para descrevermos o que estamos empacotando, são os arquivos `slack-desc`.

3. Arquivo de configuração (`.options`)

Podemos adicionar (**opcionalmente**) outro arquivo contendo as configurações utilizadas durante a construção do pacote, nele incluiremos informações diversas como versão, revisão, flags de compilação, arquitetura do processador...

É interessante seguirmos este "padrão" para evitarmos editar o `SlackBuild` toda vez que quisermos mudar apenas uma variável, por exemplo, quando alterarmos a arquitetura do processador ou a versão/revisão do pacote.

Este arquivo deve seguir o padrão "`nome_do_programa.options`" (ex. `amarok.options`).

Um exemplo deste tipo de arquivo (seguindo os padrões do **`slackware`**) seria:

```
# Versão/nome do programa.
NAME=amarok
VERSION=1.3.6

# Revisão do pacote.
BUILD=1

# Arquitetura do processador.
ARCH=i486

# target arch, i486 para
# qualquer x86 de 32 bits.
TARGET=i486

# flags utilizadas pelo compilador.
CPUOPT="-O2 -march=$ARCH -mcpu=i686"

# Número de tarefas "paralelas".
NUMJOBS=-j4

# Arquivos temporários.
TMP=/tmp

# Código fonte.
SRC=$NAME-$VERSION.tar.bz2

# Pré-instalação do programa.
PKG=$TMP/package-$NAME

# Arquivo já empacotado.
TGZ=$NAME-$VERSION-$ARCH-$BUILD.tgz

# Documentação.
DOCS=$PKG/usr/doc/$NAME-$VERSION

# Onde o programa será instalado.
PREFIX=/opt/kde
```

```
# Configurações utilizadas.
CONFIGURE="--prefix=$PREFIX \
  --build=$TARGET-slackware-linux"

# doinst.sh e slack-desc.
INSTALL=$PKG/install
```

Quando não quisermos utilizar este tipo de arquivo (`.options`), as variáveis deverão ser instanciadas no próprio `SlackBuild`.

Em `scripts` mais elaborados podemos adicionar rotinas que detectam automaticamente a revisão do pacote, a última versão estável, efetuam o download do código-fonte ou permitem o acesso ao código atual (não-estável) de desenvolvimento (via CVS).

Para a versão, poderíamos fazer o seguinte (opcionalmente implementando o que foi descrito acima):

```
# Onde o projeto está hospedado
# (lista os códigos fonte).
HOST="http://\
      prdownloads.sourceforge.net/\
      $NAME/?sort_by=date&sort=desc"

# Tenta detectar a última versão
# estável do programa.
VERSION=`lynx -dump -source "$HOST" |\
      grep $NAME- | \
      head -1 | \
      sed -r 's/(.+'$NAME'-)\
      (.+)(\.tar.+)/\2/'`
```

Este algoritmo funciona bem (sem alterações) para a maioria dos projetos hospedados no `sf.net`.

E para atualizar automaticamente a revisão do pacote:

```
# Última revisão (caso exista).
test -f $NAME.build || \
      echo 0 > $NAME.build

# Incrementa a revisão.
BUILD=`echo "$(cat $NAME.build) + 1"|\
      bc`

# Armazena a atual.
echo $BUILD > $NAME.build
```

Com isto, na próxima revisão/versão do pacote não será mais necessário editarmos nem mesmo o arquivo de opções. Poderiamos até mesmo adicionar um `script` destes (com detecção automática) ao cron e mantermos o programa sempre atualizado sem mesmo ficarmos sabendo.

4. Script (arquivo) .SlackBuild

Inicialmente adicionamos informações sobre o que o `script` faz (e informações que forem pertinentes, como licença, autor, etc).

```
#!/bin/sh
#
# amarok.SlackBuild
# (amaroK - Slackware build script)
# author: Herbert Alexander Faleiros
#  <herbert@faleiros.eti.br>
```

Feito isto armazenamos o diretório corrente e em seguida carregamos o nosso arquivo de configurações:

```
CWD=`pwd`
. $CWD/amarok.options
```

O próximo passo é efetuarmos uma "limpeza" pré-construção (e criarmos alguns dos diretórios que serão utilizados mais adiante):

```
rm -rf $PKG $NAME-$VERSION
mkdir -p $PKG $DOCS $INSTALL
```

O procedimento acima certifica-se de que não contaminaremos o pacote com versões/revisões antigas.

Se quisermos (opcionalmente) efetuar o donwload do código-fonte (automatização do arquivo de opções já citado):

```
test -f $SRC || wget $HOST/$SRC
```

Neste caso é necessário sobreescrever a variável `HOST` no arquivo de opções (faça isto no final do arquivo):

```
HOST=http://ufpr.dl.sourceforge.net/\
      sourceforge/$NAME
```

No exemplo acima efetuaremos o download de um dos mirrors do `sf.net`. Em seguida extraímos/acessamos o código-fonte:

```
tar -xjvf $SRC
cd $NAME-$VERSION
```

Depois configuramos e compilamos o programa:

```
CFLAGS=$CPUOPT CXXFLAGS=$CFLAGS
./configure $CONFIGURE
make $NUMJOBS || exit 1
```

Terminado o processo de compilação de todos os arquivos necessários ao funcionamento do programa instalamos o mesmo em um local temporário, isto é necessário para que possamos manipular os arquivos antes de instalá-los definitivamente:

```
make install DESTDIR=$PKG
```

Alguns programas não implementam o DESTDIR nos Makefile's, nestes casos devemos editar manualmente o arquivo com as configurações para que os mesmos sejam instalados corretamente no diretório temporário $PKG, como exemplo descrevo o que costumo fazer com o Makefile dos drivers do MPlayer (Matrox/Radeon):

```
cat Makefile | \
      sed -r 's/^(MDIR.+ )\
      (\/lib.+)/\1$(DESTDIR)\2/' > \
      Makefile.tmp
mv Makefile makefile.old
mv Makefile.tmp Makefile
```

Isto corrige a ausência do DESTDIR, mas cada programa deve ser analisado individualmente. Para quem tem menos paciência (ou não quer analisar os Makefile's) uma opção seria utilizar algo como o checkinstall (eu acho um pecado construir pacotes com este tipo de programa, mas enfim...).

Outro padrão a ser seguido é adicionarmos a documentação do programa, licenças, autores, ao $PKG/usr/doc/$NAME-$VERSION:

```
cp [A-Z]* $DOCS
cd $DOCS
rm -f Makefile*
```

Devido à padronização no processo de documentação da maioria dos programas as instruções genéricas acima funcionam na grande maioria dos pacotes, pois arquivos como AUTHORS, COPYING, LICENSE, TODO, INSTALL, ChangeLog, enquadram-se bem ao padrão que passamos ao cp.

Um efeito colateral é a cópia de arquivos como Makefile's, portanto a instrução final remove os mesmos.

Outro procedimento útil no caso do amaroK seria criar um link simbólico para a documentação em HTML:

```
ln -sf $PKG$PREFIX/share/doc/HTML
```

Feito isto acessamos o diretório onde pré-instalamos o programa e efetuamos as tarefas pós-compilação ainda pendentes:

```
cd $PKG
```

Copiamos o arquivo de descrição (slack-desc) para seu local final:

```
cp $CWD/slack-desc $INSTALL
```

Por segurança verificamos/(re)setamos as permissões/donos dos arquivos:

```
chown 0.0 . -R

# Consertamos o que eventualmente o
# chown acima estragou.
for i in /bin /sbin /usr/bin \
      /usr/sbin; do
  chown 0.bin $i -R
done
# Teste de sanidade.
for i in a-st go-w; do
  chmod $i . -R
done
```

Em seguida "estripamos" os ELFs (binários e bibliotecas executáveis):

```
find . | xargs file | grep ELF | \
      cut -d: -f1 | \
      xargs strip –strip-unneeded \
      2>/dev/null
```

Renomearmos todos os arquivos de configuração existentes seguindo o padrão "nome_do_arquivo.new", por exemplo:

```
mv $PKG/etc/wgetrc $PKG/etc/wgetrc.new
```

Isso garante que, durante um upgradepkg, tais arquivos (já existentes e/ou personalizados) não sejam sobrescritos, caso contrário teríamos de reconfigurar novamente todo o programa.

Outro padrão a seguirmos é compactarmos os manuais (no caso do KDE os manuais são armazenados em PKGPREFIX/man/man?/*):

```
if [ -d $PKG/usr/man ]; then
  gzip -9 $PKG/usr/man/man?/*
fi
```

Finalmente criamos o pacote:

```
makepkg -l y -c n $TGZ
```

Os parâmetros instruem o makepkg a remover todos os links simbólicos (e adiciona-los ao $INSTALL/doinst.sh) e a não alterar nenhuma permissão/dono no pacote a ser criado.

Opcionalmente podemos gerar os hashes do pacote (útil para verificações de integridade):

```
for HASH in md5 sha1; do
  ${HASH}sum $TGZ > $TGZ.$HASH
done
```

Ou ainda assinarmos o mesmo (não é muito viável/seguro num SlackBuild):

```
gpg -sda $TGZ
```

Para quem quiser instalar/atualizar pelo próprio SlackBuild, basta adicionar:

```
upgradepkg $TGZ || installpkg $TGZ
```

5. Arquivo doinst.sh

Neste arquivo o makepkg armazena as instruções para que sejam recriados os links simbólicos removidos, é usado também para tarefas pós-instalação.

Caso haja algum arquivo renomeado para a extensão .new é necessário adicionarmos instruções para que os mesmos retornem ao seu estado inicial caso não seja necessário manter arquivos pré-existentes (geralmente o programa está sendo instalado pela primeira vez), no caso citado anteriormente (wget) seria algo como:

```
if ! [ -e /etc/wgetrc ]; then
  mv /etc/wgetrc.new /etc/wgetrc
fi
```

Antes de executar o SlackBuild é necessário ativar seu bit de execução do mesmo e que todos os arquivos necessários à construção do pacote encontrem-se disponíveis:

```
$ chmod +x amarok.SlackBuild

$ ls -l
-rw-r--r--  1 herbert users 8649808 \
      2005-11-08 03:40 \
      amarok-1.3.6.tar.bz2
-rwxr--r--  1 herbert users     944 \
      2005-09-13 02:44 \
      amarok.SlackBuild
-rw-r--r--  1 herbert users     642 \
      2005-09-13 02:40 \
      amarok.options
-rw-r--r--  1 herbert users     250 \
      2005-09-13 02:46 \
      slack-desc
```

Agora é só executar (como root) o SlackBuild em questão:

```
# ./amarok.SlackBuild
```

O básico é isso, quem estiver interessado em maiores detalhes (ou scripts mais complexos/elaborados) poderá consultar os sources da própria distro ou os SlackBuild's que se encontram no endereço http://www.faleiros.eti.br/SlackBuild.

Herbert Faleiros
<herbert@faleiros.eti.br>

Autores

Clayton Eduardo dos Santos, Trabalha com Linux desde 2003 e com Slackware desde 2004. 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.

Fabiano Silva de Carvalho é Analista de Sistemas pela Universidade Salgado de Oliveira (2000) e mestrado em Engenharia Elétrica e de Computação pela UFG (2003). Atualmente é Professor Assistente da Universidade Salgado de Oliveira, Analista de Sistemas da Companhia Energética de Goiás e Professor Universitário da Faculdade de Tecnologia Senai de Desenvolvimento Gerencial. Tem experiência na área de Ciência da Computação, atuando principalmente nos seguintes temas: Sistemas Multiagentes, Aprendizado por Reforço, Data Warehouse, Implementação e Integração. Publicou 3 trabalhos em anais de eventos.Trabalha com Linux desde 1996.

Flavio do Carmo Junior a.k.a. drkn, 22 anos, cursando Sistemas de Informações. Teve seu primeiro computador, um 386, aos 10 anos que o satisfez ate enjoar de "Prince of Persia" depois de uns 2 anos, voltou a mexer em computadores em 95 quando ganhou um Celeron 400Mhz, quakemaniaco de carteirinha se viu em desvantagem quando surgiu o ADSL e então iniciou-se no mundo Linux em 99 pulando por várias distribuições até conhecer o slackware. (ponto-final :)

Herbert Alexander Faleiros (aka ratmannn), programador, graduando em Física pela UFSCar. Seu primeiro contato com o Slackware ocorreu em 2000. Atualmente trabalha desenvolvendo soluções em Java para servidores de aplicações, nas horas vagas escreve artigos sobre criptografia e build scripts para o Slackware.

Nycholas de Oliveira e Oliveira, 20 anos, se interessou por linux em 2003 mais profissionalmente em 2004 felizmente trabalhando com slackware logo de cara :D. Trabalha como administrador de sistemas e redes, programador em uma fundação de uma universidade pública e luta pesado todos os dias para passar no vestibular.

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).

Brincando com Bits

Todo mundo sabe que permissões de arquivos e diretórios e importante, principalmente para garantir a segurança do sistema contra pessoas não autorizadas e/ou maliciosas. E sabemos também como atribuir/retirar permissões de um arquivo ou diretório. Para quem ainda não sabe disso vai alguns links logo abaixo sobre o assunto:

http://focalinux.cipsga.org.br/guia/iniciante/ch-bas.htm#s-basico-arquivo
http://focalinux.cipsga.org.br/guia/intermediario/ch-disc.htm#s-disc-sistarq
http://focalinux.cipsga.org.br/guia/intermediario/ch-perm.htm

Por outro lado poucas pessoas sabem como contar os bits de permissão de arquivos/diretórios.

Por exemplo quando se faz o comando:

```
slack@zine:~$ chmod 0754 file
slack@zine:~$ ls -l file
-rwxr-xr--  1 slack zine file
```

Que seria a mesma coisa que:

```
slack@zine:~$ chmod u=rwx,g=rw,o=r \
      file
slack@zine:~$ ls -l file
-rwxrw-r--  1 slack zine file
```

Agora você sabe como transformar as permissões do tipo caractere (`-rwxrw-r--`) para numeral (`0754`)? (Não :P) É isso que vou apresentar agora em um artigo simples e direto. Vamos ao que interessa.

Temos nove variáveis de permissão: `-rwxrwxrwx`. Só que vamos trabalhar de três em três (`rwx`).

Vamos ao um exemplo de permissão, `r-x`. Pegamos quais bits queremos ativar e colocamos baixo deles os números nessa ordem, `210` e logo abaixo coloca-se os números, `222`:

```
r - x
2 1 0
2 2 2
```

Aí multiplicamos o primeiro número pelo segundo. Mais só vamos multiplicar os números de estão abaixo dos bits que vamos ativar:

```
r - x
2 1 0
2 2 2
-----
4 - 1
```

Agora os matemáticos de plantão vão me crucificar vivo! `0*2=1` ?!? Isso mesmo quando for multiplicação por `0` atribui-se `1` ao resultado e não `0`. Agora tudo explicado ninguém vai mais me destroçar :D.

Depois disso tudo é só somar o resultado das multiplicações:

```
4 + 1 = 5
```

`chmod u=rx` e a mesma coisa que `chmod 0500`:

```
slack@zine:~$ chmod u=rx file
slack@zine:~$ ls -l file
-r-x------  1 slack zine file

slack@zine:~$ chmod 0500 file
slack@zine:~$ ls -l file
-r-x------  1 slack zine file
```

Faça alguns teste e divirta-se :)).

```
slack@zine:~$ chmod u=rwx,g=rw,o=r \
      file
slack@zine:~$ ls -l file
-rwxrw-r--  1 slack zine file

r w x - r w - - r - -
2 1 0 - 2 1 0 - 2 1 0
2 2 2 - 2 2 2 - 2 2 2
----- - ----- - -----
4 2 1 - 4 2 - - 4 - -
+++++ - +++++ - +++++
  7   -   6   -   4

slack@zine:~$ chmod 0764 file
slack@zine:~$ ls -l file
-rwxrw-r--  1 slack zine file
```

E para os bits especiais? Para os bits especiais e o mesmo processo, só que em vez de separar de três em três os caracteres de permissão vamos fazer o somatório com todos eles:

```
slack@zine:~$ chmod u=rx,g=wxs,o=rwxt\
       file
slack@zine:~$ ls -l file
-r-x-wsrwt  1 slack zine file

r - x - - w s - r w t
  2       1       0
  2       2       2
+++++ - +++++ - +++++
  -   -   2   -   1    => 2 + 1 = 3

r - x - - w s - r w t
2 1 0 - 2 1 0 - 2 1 0
2 2 2 - 2 2 2 - 2 2 2
----- - ----- - -----
4 - 1 - - 2 1 - 4 2 1
+++++ - +++++ - +++++
  5   -   3   -   7

slack@zine:~$ chmod 3537 file
slack@zine:~$ ls -l file
-r-x-wsrwt  1 slack zine file
```

Quando algum bit especial aparecer em caixa-alta (maiúsculo) significa que o bit normal não foi setado:

```
slack@zine:~$ chmod u=rs,g=rxs,o=rt \
      file
slack@zine:~$ ls -l file
-r-Sr-sr-T  1 slack zine file
```

Nesse caso quando você for fazer os cálculos dos bits normais eles vão ser ignorados onde um bit especial estiver de caixa-alta, por exemplo:

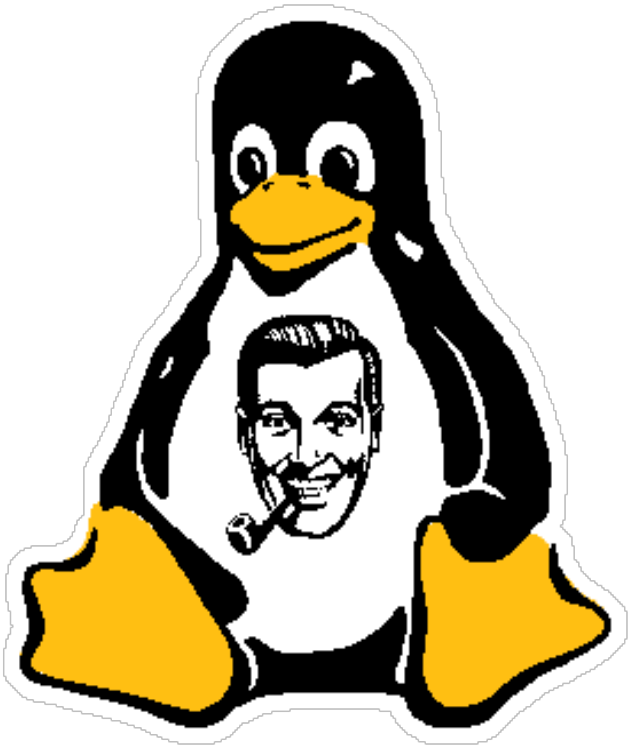
```
r - S - r - s - r - T
  2       1       0
  2       2       2
+++++ - +++++ - +++++
  4   -   2   -   1   => 4 + 2 + 1 = 7

r - - - r - x - r - -
2 1 0 - 2 1 0 - 2 1 0
2 2 2 - 2 2 2 - 2 2 2
----- - ----- - -----
4 - - - 4 - 1 - 4 - -
+++++ - +++++ - +++++
  4   -   5   -   4

slack@zine:~$ chmod 7454 file
slack@zine:~$ ls -l file
-r-Sr-sr-T  1 slack zine file
```

Até a próxima :P.

Nycholas de Oliveira e Oliveira
<nycholas@gmail.com>

Patrick,

all of the Brazilian Slack users congratulate you on the arrival of Briah, and wish you a Happy New Year!



E um feliz 2006 para todos os nossos leitores.

Keep slacking!

Slackwarezine team

Instalando a D-Link DSB C110

S.O. e softwares utilizados

Kernel linux 2.6.14
amsn 0.95b (que só se acha via CVS):
http://amsn.sourceforge.net/
spca50xx:
http://sourceforge.net/\
projects/spca50x/
videodev:
módulo que já deve estar presente após a instalação/compilação do seu kernel.

Mãos à obra

Pelo que pude ver o kernel do Linux não tem suporte nativo a esta minha webcam. Então o jeito foi pesquisar na Internet por alguma alternativa. Falando com um aqui, outro ali, ouvi falar do tal spca50xx. Mesmo que sua webcam não seja igual à minha, dê uma olhada no site desse driver. Lá tem uma lista bem grande de webcams suportadas por esse driver.

Bom, na verdade a versão do spca50xx que eu peguei (spca5xx-20051001) não compila com o kernel 2.6.14. Então eu fiz uma pequena gambiarra. Na hora da compilação do driver deverá surgir mensagens de erro como abaixo:

```
spca5xx-20051001/drivers/usb/\
      spca5xx.h:23:40 : missing \
      binary operator before token "("
spca5xx-20051001/drivers/usb/\
      spca5xx.h:44:40 : missing \
      binary operator before token "("
spca5xx-20051001/drivers/usb/\
      spca5xx.h:54:41 : missing \
      binary operator before token "("
```

Bom, minha gambiarra consiste em editar esse arquivo e modificar essas linhas, inseririndo aspas simples (') antes e depois de cada constante, ficando assim, respectivamente:

```
#if 'LINUX_VERSION_CODE' < \
'KERNEL_VERSION(2,5,0)'

#if 'LINUX_VERSION_CODE' > \
'KERNEL_VERSION(2,5,41)'

#if 'LINUX_VERSION_CODE' >= \
'KERNEL_VERSION(2,4,20)' && \
'LINUX_VERSION_CODE' < \
'KERNEL_VERSION(2,5,0)'
```

Pronto, agora o driver deve compilar. Então...

```
# make && make install
```

Se tudo correu bem o driver já está compilado e instalado. Carregue os módulos 'videodev' e 'spca5xx':

```
# modprobe videodev
# modprobe spca5xx
```

Agora basta instalar o amsn. Baixe o arquivo do CVS deles, descompacte com 'tar -zxvf'. Entre no diretório recém criado e execute:

```
# ./configure
# make
```

Agora entre no diretório utils/linux/capture (que está dentro do diretório criado na descompactação do amsn). Rode o script 'test.tcl' para testar sua webcam:

```
# ./test.tcl
```

Primeiro vá em 'Choose device'. Depois em 'Camera settings'. Não precisa perder muito tempo fuçando aí. O importante é que o 'test.tcl' ache sua webcam e tu consigas ver a imagem que a câmera captura.

Supondo que tudo correu bem no 'test.tcl', agora é só usar o amsn para transmitir imagens através da sua webcam.

Mas 'peraí'! Tem um segredinho faltando. Em muitos casos ainda ficará faltando liberar no firewall e/ou no modem ADSL as portas que o amsn utiliza para a transmissão de vídeo. Então... basta liberar as portas 6891 e 6892 (protocolo tcp) e tudo funcionará maravilhosamente bem.

Espero que outros não passem pelo mesmo trabalho que eu passei para descobrir que o furo da bala eram aquelas portas ;)

Yucatan “Kenjiro” Costa
<yucatan@globo.com>

Instalando e Usando o rdesktop

Bem, como vocês já devem ter percebido, costumo escrever artigos baseados em situações reais, e normalmente inusitadas, que surgem no meu cotidiano, mais especificamente, em meu ambiente de trabalho.

Como todos nós sabemos, infelizmente, é praticamente impossível trabalharmos em um ambiente que roda 100% *nix. A algumas semanas atrás, tive de realizar uma manutenção remota em um máquina com "aquele" sistema operacional. Como de costume, saí em busca de alguma alternativa livre que pudesse ser utilizada no bom e velho **slackware** e tive uma grata surpresa.

A manutenção, obrigatoriamente, deveria ser realizada utilizando o Terminal Service do Window$ o que num primeiro momento me levou a pensar que não teria êxito em minha busca, mas estava enganado, existe uma excelente alternativa livre para *nix chamada `rdesktop`.

O `rdesktop`, atualmente na versão 1.4.1, é um cliente gráfico *nix compatível com o Terminal Service. O processo de instalação é trivial, primeiro faça o download usando o `lynx` ou o `wget`:

```
$ lynx \
      http://rdesktop.sourceforge.net/
$ wget http://\
      optusnet.dl.sourceforge.net/\
      sourceforge/rdesktop/\
      rdesktop-1.4.1.tar.gz
```

Em seguida, o procedimento “de sempre”:

```
$ tar -xvzf rdesktop-1.4.1.tar.gz
$ cd rdesktop-1.4.1
$ ./configure
$ make
$ su
# checkinstall
```

Vamos a utilização do software. Suponha que você tenha uma máquina Window$ XP/2000/2003 com acesso remoto habilitado cujo ip é 192.168.0.13. A partir de nossa máquina **slackware**, basta digitarmos:

```
$ rdesktop 192.168.0.13
```

O X irá abrir uma janela já com a tela de autenticação, pedindo um usuário/senha válidos. Funciona muito bem, no entanto, como vocês devem ter percebido a resolução é de apenas 256 cores, para mudar isso, basta acrescentar:

```
$ rdesktop -a 16 192.168.0.13
```

O `rdesktop` será iniciado com 16 bits de cor. As opções "`-a 8`", "`-a 15`" e "`-a 24`" também são válidas. O usuário senha também pode ser especificado via linha de comando:

```
$ rdesktop -u usuario -p senha \
      -a 16 192.168.0.13
```

Várias outras

- `-z` Habilita compressão dos dados trafegados (só funciona com 8 bits de cor);
- `-f` Habilita modo tela cheia, para reverter, basta pressionar as teclas `Ctrl+Alt+Enter`;
- `-T` "mensagem" : Muda o título da janela da sessão;
- `-k` nome_do_mapa : Habilita mapa de teclado específico para a sessão iniciada.

Existem ainda opções mais específicas, como redirecionamento de periféricos e diretórios locais/remotos, ativação/desativação dos sons emitidos na sessão remota ou direcionamento para a máquina local, `cache` das imagens transferidas, entre outros recursos. Nada que um "`man rdesktop`" não resolva... :)

Vale lembrar que o `rdesktop` pode ser utilizado em conjunto com o `LTSP`, cuja instalação passo a passo foi publicada em uma das últimas edições do `slackwarezine`. Desse modo, é possível se utilizar estações `diskless` para acesso remoto a ambientes Windows a partir de estações rodando Linux.

Bem pessoal, o propósito desse artigo é fornecer uma "dica" de aplicação que pode facilitar a vida de um administrador em uma situação que exija a operação de um servidor/estação com outro S.O..

Clayton Eduado dos Santos
<clayton@slackpoint.com.br>

Autenticando o Slackware ***SEM PAM*** em uma base LDAP

Introdução

E um procedimento realmente simples, mas que as vezes nos assusta pelo fato do **slackware** ser diferente da maioria das distribuições por não adotar o PAM, mas veremos que isso não é problema algum.

De qualquer forma, se lermos um pouco sobre PAM e LDAP veremos que o LDAP nao faz requisicao nenhuma ao PAM e que ele nao interfere em nada. Pois e, e eu que coloquei o PAM no **slackware** pra isso, porém não foi inválido - eu gosto do PAM.

Não vou comentar o funcionamento do PAM e LDAP aqui, pois não é esse o escopo do texto. Considero que você já conhece o LDAP e tenha o mínimo de informações sobre a estrutura (árvore) dele na sua rede.

1. Pacotes necessários

- OpenLDAP:
  `http://www.openldap.org`
- nss_ldap:
  `http://www.padl.com/\`
  `OSS/nss_ldap.html`

Acesse os sites acima e baixe os softwares nas versões atualizadas. Eu estou utilizando o OpenLDAP 2.3.11 e nss_ldap 2.44 neste artigo. Para complementar esse artigo, consulte a edição #2 do `slackwarezine`, que também trata de LDAP sem PAM, mas aborda também a criação do servidor.

2. Descompactando, configurando, compilando e instalando os pacotes

2.1. OpenLDAP

Lembre que um `prompt` começando por `$` = procedimento possivel como usuário e `#` = procedimento necessariamente como `root`

Vamos até o diretório do download e seguir os passos:

```
$ tar -zxvf openldap-VERSAO.tgz
$ cd openldap-VERSAO/
$ ./configure --enable-slapd=no \
      --enable-backends=no \
      --enable-slurpd=no
$ make depend
$ make
# make install
```

Perceba que na linha `./configure` estamos desabilitando qualquer trabalho do LDAP como SERVER, visto que a intenção do artigo é um client.

2.2. nss_ldap

Da mesma forma, vamos até o diretório do download e seguir os passos:

```
$ tar -zxvf nss_ldap*.tgz
$ cd nss_ldap-VERSAO/
$ ./configure
$ make
# make install
```

Feito, bem simples.

3. Adaptando as configurações para nosso ambiente

3.1. Editando o arquivo /usr/local/etc/openldap/ldap.conf

```
# /usr/local/etc/openldap/ldap.conf #

BASE              dc=example,dc=com
HOST              192.168.0.3

rootbinddn cn=Manager,dc=example,\
      dc=com
nss_base_passwd ou=Users,ou=People,\
      dc=example,dc=com?sub
nss_base_shadow ou=Users,ou=People,\
      dc=example,dc=com?sub
nss_base_group ou=Groups,dc=example,\
      dc=com?one

# /usr/local/etc/openldap/ldap.conf #
```

Claro, substitua "`dc=example,dc=com`" para o seu suffix e "`cn=Manager`", "`ou=Users`", "`ou=Groups`" para o sua estrutura correspondente, e HOST pelo IP do servidor LDAP.

Atenção nas linhas `nss_*`, verifique e, se necessário, adapte para o seu modelo de base do LDAP. O conteúdo `?sub` ou `?one` define a forma de busca que sera feita a partir daquele ponto afim de localizar o objeto.

Depois de tudo acertado nesse arquivo, para checar se voce ja consegue uma identificação na base LDAP, execute o comando:

```
# ldapsearch -x
```

Ele deve retornar a estrutura da sua base com um final parecido com o abaixo, se houver algum erro verifique todos os passos e rode o comando novamente.

```
# search result
search: 2
result: 0 Success
```

O `status` "`0 Success`" significa tudo OK.

Porém, a gente so fez uma conexao a base LDAP do servidor pelo nosso LDAP local, agora vamos colocar o **`slackware`** para trabalhar com o LDAP. Primeiro, façamos ele reconhecer o LDAP local.

Copie o arquivo `/usr/local/etc/openldap/ldap.conf` para o diretório `/etc` e outra vez como `/etc/nss_ldap.conf`

```
# cp /usr/local/etc/\
      openldap/ldap.conf \
      /etc/ldap.conf
# cp /usr/local/etc/\
      openldap/ldap.conf \
      /etc/nss_ldap.conf
```

3.2. Editando o arquivo: /etc/nsswitch.conf

```
##### /etc/nsswitch.conf #####
passwd:           ldap compat
shadow:           ldap compat
group:            ldap compat
##### /etc/nsswitch.conf #####
```

Vá até o arquivo `/etc/nsswitch.conf` e deixe-o dessa forma. Provavelmente a linha "`shadow:`" não deve existir, adicione-a ou não conseguirá logar-se no sistema. Se no lugar de "`compat`" estiver "`files`" então deixe "`ldap files`" mesmo, o `compat` é um conjunto de arquivos que incluem o `files` e mais alguns, utilizado pelo **`slackware`**.

Salve as alterações, feche o arquivo e vamos testar com o comando:

```
# id usuario_da_base_ldap
```

eu fiz:

```
# id teste
uid=1005(teste) gid=513(Domain Users)
groups=513(Domain Users),
1001(slackwarezine)
```

O usuario "`teste`" não existe no sistema atual, mesmo assim ele pôde ser identificado na base LDAP e consultado.

4. Conclusão

Esse artigo descreve um procedimento muito simples, porem quando fui colocar meu **`slackware`** para autenticar na base LDAP tive muitas dúvidas e muito pouco material foi encontrado sobre **`slackware`** se autenticando em base LDAP remota.

Flávio do Carmo Jr <billpp@gmail.com>

Instalando o PostFix

1) Instale o mysql (Pode ser obtido no CD de instalação do **slackware**):

```
# installpkg mysql-4.0.23a-i486-1.tgz
```

2) Instale o cyrus-sasl (Pode ser obtido no CD de instalação do **slackware**):

```
# installpkg \
      cyrus-sasl-2.1.21-i486-3.tgz
```

3) Configure o Servidor de DNS com o Registro MX adequado (ip do servidor de e-mail). O Servidor de DNS pode ser outro servidor na rede. A instalação e configuração do servidor de dns não será tratada aqui.

4) Instale o postfix (Pode ser obtido em www.linuxpackages.net):

```
# installpkg \
      postfix-2.2.5-i486-1stb.tgz
```

5) Configure os parâmetros (myorigin, mydestination, mynetworks, myhostname, mydomain) no arquivo /etc/postfix/main.cf .

```
myorigin = $mydomain
mydestination = $myhostname,\
      localhost.$mydomain, \
      localhost, $mydomain
mynetworks = 192.168.0.0/16, \
            127.0.0.0/8
myhostname = mail.seudominio.com.br
mydomain = seudominio.edu.br
```

No main.cf podem ser configurados em torno de 300 parâmetros. Por hora esses cinco são suficientes. Para otimização e configuração extra do servidor de mensagens consulte a documentação do postfix.

6) Crie um link no diretório /etc para o arquivo /etc/postfix/aliases

```
# ln -sf /etc/postfix/aliases \
      /etc/aliases
```

7) Gere a base de dados de aliases

```
# newaliases
```

8) Inicie o Servidor de Mensagens

```
# sh /etc/rc.d/rc.postfix start
```

Se tudo foi feito corretamente, neste ponto o servidor de mensagens já está funcionando. Para testar:

```
telnet mail.seudominio.com.br 25
```

9) Instale o imapd (Responável pelos protocolos POP3 e IMAP. Pode ser obtido no CD de instalação do **slackware**).

```
# installpkg imapd-4.62-i486-1.tgz
```

10) O imapd trabalha associado ao inetd. Então devemos configurar o inetd. Para isso altere o arquivo /etc/inetd.conf descomentando as linhas referentes aos protocolos IMAP e POP3..

```
#pop3    stream  tcp     nowait \
      root    /usr/sbin/tcpd \
      /usr/sbin/popa3d
#imap2   stream  tcp     nowait \
      root    /usr/sbin/tcpd  \
      imapd
```

Retire os # que estão na frente de cada uma dessas linhas.

11) Inicie o inetd:

```
# sh /etc/rc.d/rc.inetd start
```

12) Os protcolos POP3 e IMAP já devem estar ativos. Para testar o POP3:

```
telnet mail.seudominio.com.br 110
```

13) Crie as contas dos usuários:

```
# adduser
```

14) Configure os clientes (thunderbird, mozilla-mail, outlook, ...)

Fabiano Silva de Carvalho
<fscarvalho@gmail.com>