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

22 de Julho de 2004 – Número 4

Editorial

Mais uma edição do **slackwarezine**, desta vez com um atraso terrível. Depois de lançar praticamente três edições seguidas (#2.5, #3 e #3.5) gostamos do descanso... mas agora voltamos com 16 páginas e com a corda toda!

Depois do lançamento do **slackware** 10, não podia faltar um artigo mostrando algumas dicas de configuração. Por isso temos o "Post-Install do **slackware** 10". São configurações básicas, praticamente para deixar a máquina rodando.

Ainda na esteira do lançamento da nova versão, um artigo sobre como instalar nela o kernel 2.6 (que já vem no /testing) e um sobre como instalar o **slackware** via NFS ou FTP. O artigo foi baseado no 9.1, mas não devem haver problemas em utilizá-lo para o 10.

Um artigo sobre redes WiFi completa esse combo, mostrando como configurar uma placa de rede "wireless" tanto no modo Ad-Hoc como no modo Access Point (o que são estes termos? Leia o artigo!)

Usuários mais avançados vão gostar do artigo sobre sincronia e replicação de arquivos utilizando o **unison**. Nada como manter o seu /home sincronizado em casa e no trabalho, sem os vários problemas que o rsync pode trazer...

Programadores de plantão irão adorar o artigo do Reinaldo, com um pequeno tutorial de como programar na Qt3. Montar uma janela, formatar o texto, criar botões, tudo isso em um "Hello World" com explicações detalhadas de como as coisas vão acontecendo.

Piter PUNK

Índice

Errata

Nas edições #3 e #3.5 (especial para o FISL) está errada a URL do **slackwarezine**, onde está: **www.slackware.com.br** substitua por **www.slackwarezine.com.br**

Ainda no número #3, está errada a data da edição! Onde está **19 de Maio de 2003** substitua por **19 de Maio de 2004**.

E, se alguém achar mais erros, por favor nos informe, nada pior que erros se propagando por aí...

slack users

Construindo Aplicações Gráficas em Qt3

1. Introdução

Neste artigo pretendo mostrar como combinar o C++ e as funcionalidades do Qt para a construção de aplicativos gráficos (GUI). Também irei mostrar uma particularidade do Qt que são os "signals e slots", usados para responder as ações do usuario ou mudanças no estado de objetos.

2. Aplicação Simples

A seguir um programa bem simples:

```
 1 #include <qapplication.h>
 2 #include <qlabel.h>

 3 int main(int argc, char *argv[])
 4 {
 5     QApplication app (argc, argv);
 6     QLabel *label = new Qlabel("Hello \
             SlackZine!", 0);
 7     app.setMainWidget(label);
 8     label->show();

 9     return app.exec();
10 }
```

Agora vou explicar o porque de cada linha deste pequeno programa, como compilar e como rodar. Cuidado com a linha 6, o texto completo dela é: "`QLabel *label = new QLabel("Hello SlackZine!", 0);`". Neste artigo, Utilizaremos a barra invertida `\` no final da linha para indicar que ela ainda não acabou, continuando na próxima linha.

Na linha 5 criamos um objeto `QApplication` para gerenciar a aplicação. Este objeto requer os argumentos "argc" e "argv" porque o Qt suporta alguns argumentos via linha de comando (como o `-display` do X por exemplo). Em toda a aplicação deve haver somente um objeto deste tipo, pois é ele quem vai gerenciar os recursos de toda a aplicação, como a fonte padrão e o cursor.

Na linha 6 instanciamos um objeto do tipo `QLabel` que mostrará a string "Hello SlackZine!". O `0` (zero) como segundo argumento (null pointer) diz ao construtor do `QLabel` que este `widget` é uma janela por si só, não uma `widget` que pertence a uma outra janela.

Uma `widget` é um elemento visual que, agrupados, formam a interface da aplicação. Botão, menus, barra de rolagem são exemplos de `widget`. Os `widgets` do Qt não são divididos entre "controls" e "containers": podem ser tratados tanto como "control" quanto como "container". Todos os `widgets` são subclasse da classe `QWidget` ou de uma de suas subclasses, e podem ser criados novos `widgets` como subclasse da `QWidget` ou de suas subclasses. Um `widget` pode ter quantos `widgets` filhos quiser, não há limitações. Um `widget` que não tem pai, é um `widget` do tipo "top-level" (uma janela).

Por exemplo, se em nossa aplicação exemplo tivéssemos criado um `frame` para colocar o `label` e alguma outra coisa como um botão, teríamos no lugar do 0 no construtor do `QLabel` uma referencia a instância do `frame`:

```
QLabel *label = new \
       QLabel("Hello SlackZine!", myFrame);
```

Na linha 7 fazemos com que o nosso `label` seja a janela principal da aplicação para que o comando para fechar a janela termine a aplicação. Sem uma janela principal, quando a última janela for fechada, a aplicação vai continuar rodando em background.

Na linha 8 fazemos com que nosso `label` fique visível. Os `widgets` são sempre criados para ficarem invisiveis.

Na linha 9 passamos o controle da aplicação para o Qt. O `exec()` vai retornar quando a aplicação for fechada.

3. Compilando

Para compilarmos nossa aplicação, teremos que criar um `Makefile`, e o jeito mais fácil de se fazer isso é usando a ferramenta "`qmake`".

Dentro da pasta com o seu `.cpp`:

```
       qmake -project
```

Será criado um .pro com o nome da pasta. Por exemplo, se o nome da pasta é `helloQt`, o seu `.pro` vai se chamar `helloQt.pro`.

Depois criaremos o Makefile:

```
       qmake helloQt.pro
```

Agora com o `Makefile` em mãos, você pode compilar a sua aplicação.

Fig. 01 - Hello SlackZine! no KDE

Você também pode formatar o texto do `label` com um HTML simples, por exemplo, substitua o testo do `label` por algum texto com formatação HTML:

```
-QLabel *label = new QLabel("Hello SlackZine!", 0);
+QLabel    *label    =    new    QLabel("<b>Hello</b>
<i>SlackZine</i>!", 0);
```

Fig. 02 - Hello SlackZine! com formatação HTML

continua

4. Fazendo Conexões

Agora poderíamos fazer com que nossa aplicação respondesse a alguma ação do usuário. Vamos pegar o nosso exemplo anterior e colocar um botão para fechar a janela. Vamos acrescentar um `QPushButton` e uma conexão para que o click no botão execute algo na aplicação. Para colocarmos nosso botão e mantermos o `label`, já que são dois componentes distintos, teremos que decidir quem vai ser a nossa janela principal.

Anteriormente a nossa aplicação continha apenas um `QLabel`, e ele era a nossa janela, pois como foi explicado, todo o `widget` é um container. Mas agora nossa aplicação tem dois componentes. Qual dos dois será o nosso container?

Para resolver este problema, vamos adicionar mais um `widget` que vai ser o nosso container, o `QVBox`.

```
 1 #include <qapplication.h>
 2 #include <qlabel.h>
 3 #include <qpushbutton.h>
 4 #include <qfont.h>
 5 #include <qvbox.h>

 6 int main(int argc, char *argv[])
 7 {
 8     QApplication app (argc, argv);

 9     QVBox *box = new QVBox(0);
10     box.resize( 200, 80 );

11     QLabel *label = new Qlabel("<b>Hello</b> \
              <i>SlackZine</i>!", box);
12     label->setAlignment( Qt::AlignHCenter | \
              Qt::AlignVCenter );

13     QPushButton *quit = new \
              QpushButton("Adeus SlackZine", box);
14     quit->setFont( QFont( "Times", 16, \
              QFont::Bold ) );

15     QObject::connect(quit, SIGNAL(clicked()),\
              &app, SLOT(quit()));

16     app.setMainWidget(box);
17     box->show();

18     return app.exec();
19 }
```

Agora vamos explicar como funciona os nossos novos componentes. Além dos componentes, modificamos algumas propriedades de cada um para que possamos ver a variedade de estilos que podemos dar a nossa aplicação.

Na linha 9 criamos o nosso `widget` principal, o `QVBox`. Este `widget` simplesmente cria uma caixa vertical para agrupar os `widgets` contidos nele. O `QVBox` organiza os seus `widgets` filhos em uma linha vertical, uma abaixo da outra. Na linha 10 modificamos o tamanho do nosso `QVBox` para ter 200 pixels de largura e 80 pixels de altura.

Na linha 11 temos o nosso `QLabel` criado em nossa primeira aplicação, mas com uma diferença, desta vez ele não é o nosso container principal. Então, no segundo argumento, ao invés de colocarmos o `0` (null pointer), colocamos o widget que será o seu pai, o `QVBox`. E assim temos o nosso primeiro exemplo de uma `widget` pai e `widget` filho, que neste caso é o `QVBox` e o `QLabel` respectivamente.

Na linha 12, usamos o método `setAlignment` do `QLabel` para pedir a este widget que alinhe o texto no centro na horizontal e na vertical, respectivamente. Este método aceita `Qt::AlignmentFlags` e `Qt::TextFlags`, que são do tipo `enum`. Você pode encontrar mais detalhes na documentação do Qt.

Na linha 13 criamos o nosso `QPushButton` para recebermos uma ação do usuário. Este botão vai conter a frase "Adeus SlackZine", e também vai ser um `widget` filho do `QVBox`.

Na linha 14 usamos o `QFont`, que é um widget que provê um suporte otimizado e melhorado para o uso de fontes, para dar um estilo diferente no texto de nosso `QPushButton`. Modificamos o texto padrão do `QPushButton` para ter uma fonte do tipo "Times" com o tamanho de 16 pixels e em negrito (`QFont::Bold`).

Na linha 15 usamos o `connect()`, uma das principais características do Qt. A `connect()` é uma função *`static`* da `QObject`, a `widget` "mãe". A `connect()` é quem vai fazer a conexão entre o sinal do `widget` e o `slot`. Neste caso vai ligar o sinal `clicked()` com o `slot quit()`. Daremos mais detalhes de `signals` e `slots` na próxima seção deste artigo.

Nas linhas 16 e 17 apenas mudamos de `label` para `box`, porque agora o `QVBox` é o nosso container e janela principal.

Agora compilamos nossa reformulada aplicação com os mesmo passos aprendidos anteriormente.

```
qmake -project
qmake helloQt.pro
```

Fig. 03 - `QPushButton` e um `QLabel` em um `QVBox` no KDE

Click o botão "Adeus SlackZine" para fechar a janela.

5. Signals e Slots

O mecanismo de `signal/slot` é uma das principais caracteristicas do Qt, e sem dúvida, a parte que mais a difere das outras toolkits. Os `signals` e `slots` são usados para fazer a comunicação entre os objetos. Podemos notificar um objeto em nossa aplicação quando outro teve o seu estado alterado. Este mecanismo permite que os objetos comuniquem-se entre si sem que um saiba nada sobre o outro.

Geralmente, as `toolkits` fazem este tipo de comunicação usando *`callbacks`*. Um *`callback`* é um ponteiro para a função, se você quer que uma função seja executada para avisar-lhe de algum evento, você passa um ponteiro para a outra função (a `callback`) para executar a função.

continua

Pode parecer confuso, mas resumindo, callbacks tem dois problemas. Primeiro, ela não uma "proteção" para tipo (type safe), sendo que nunca teremos certeza se a chamada da função irá chamar a callback com os argumentos corretos. Segundo, a callback é fortemente acoplada com a função que a chama, desde que a a função saiba qual callback chamar. Isto é apenas para dar uma idéia da diferença entre o mecanismo de signal/slot com o modelo tradicional.

Um sinal (signal) é emitido quando algum evento ocorre. O slot é uma função que é chamada em resposta a um sinal em particular. No caso da nossa aplicação, o connect() funciona da seguinte maneira:

```
connect(quit, SIGNAL(clicked()), &app, SLOT(quit()))
```

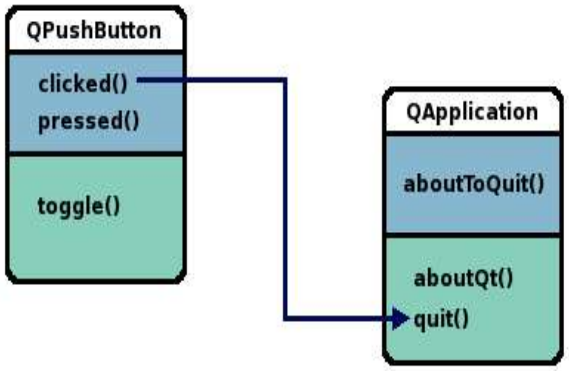

Fig 04 - Conexão do sinal clicked() do QPushButton com o slot quit() do QApplication

A chamada do connect() funciona assim:

```
connect (sender, SIGNAL(signal), receiver, \
       SLOT(slot));
```

Onde o sender e o receiver são apontamentos para os Qobjects, e onde o signal e o slot são funções sem o nome dos parâmetros. As macros SIGNAL() e SLOT() convertem estes argumentos em string.

Quando você conecta um signal a um slot ou a outro signal, em ambos os parâmetros devem ter a mesma quantidade, os mesmos tipos e a mesma ordem. Você pode conectar um signal com varios slots ou vice-versa.

Por exemplo, vários signals podem ser conectados com o mesmo slot:

```
connect(lcd, SIGNAL(overflow()), this, SLOT \
             (handleMathError()));
connect(calculator, SIGNAL(divisionByZero()), \
             this, SLOT(handleMathError()));
```

Quando um dos sinais forem emitidos o slot é chamado. Da mesma maneira, um signal pode ser conectado com vários slots:

```
connect(slider, SIGNAL(valueChanged(int)), \
       spinBox, SLOT(setValue(int)));
connect(slider, SIGNAL(valueChanged(int)),\
       this, SLOT(updateStatusBarIndicator(int)));
```

Quando o sinal é emitido, os slots são chamados um após o outro, em uma ordem arbitrária.

Por fim, um signal pode ser conectado em outro signal:

```
connect(lineEdit, SIGNAL(textChanged(const \
       QString &)),  this, \
       SIGNAL(updateRecord(const QString &)));
```

Você também pode apagar uma conexão. Mas isso é difícil de ser usado porque as conexões são automaticamente apagadas quando um dos objetos relacionados a ela for apagado.

```
disconnect(button, SIGNAL(quit()), this, SLOT \
       (closeAllWindows()));
```

Um detalhe interessante também é de que, se o *signal* tiver um número maior de parâmetros do que o do slot, os parâmetros a mais simplesmente serão ignorados:

```
connect(ftp, SIGNAL(rawCommandReply(int, \
       const QString &)), this, \
       SLOT(checkErrorCode(int)));
```

6. Considerações finais

Aqui deixo minha contribuição àqueles que tem interesse em conhecer está maravilhosa toolkit ou tem necessidade de um empurrão inicial para começar com o Qt. Não expliquei tudo relativo a signals e slots, mas tentei explicar o suficiente para que as pessoas consigam começar a desenvolver aplicações com C++/Qt. Para entender melhor esta parte, ainda falta explicar mais a fundo os signals e os slots, e o sistema de Mata-Object do Qt. Mas isso pode ficar para uma eventual segunda parte deste artigo. :)

Reinaldo Nolasco Sanches(r_linux) <r_linux@yahoo.com>
Camila A. Coelho (misfit) <misfit@linuxmail.org>

Instalando o slackware via NFS ou FTP

Introdução

Esta documentação tem o objetivo descrever os passos relativo da instalação do **slackware** via NFS ou FTP. É indicado que você saiba fazer a instalação padrão do **slackware** diretamente do CD.

Quando escrevi esta documentação o **slackware** estava na versão 9.1.0. O procedimento para versões anteriores e o current é o mesmo e provavelmente será idêntico para versões posteriores.

Na realidade desta documentação poderia ser escrito duas outras (NFS e FTP), porém como as mesmas possuem muitas coisas em comum resolvi colocar tudo em uma única documentação.

A instalação do **slackware** via NFS ou FTP não tem muita dificuldade, o que existe é uma "pegadinha" que até hoje não vi escrito em nenhuma documentação.

Curioso para saber sobre a "pegadinha"? Leia a documentação por completo para não cair nela :^D

Pré-requisitos

O disco de boot os discos 1 e 2 de instalação e o disco de rede podem ser baixados em qualquer mirror do slackware. O disco de boot se encontra dentro do diretório /bootdisks (e o mais comum é o bare.i) e os demais no /rootdisks. Para fazer a instalação você precisa:

- Disco do boot, utilize o mais adequado para o seu hardware.
- Disco 1 de instalação.
- Quatro disquetes de 3 1/2".
- CD 1 de instalação do **slackware** ou um mirror da árvore oficial do **slackware**.

Pré-requisitos para instalação via NFS

- Disco 2 de instalação.
- Disco de rede.

Pré-requisitos para instalação via FTP

- Disco 2 de instalação.
- Disco de rede.

Os discos para a instalação via FTP podem ser baixados no seguinte endereço:

`http://prdownloads.sourceforge.net/slackftp`

O primeiro disco chama-se `install-ftp.2` e o segundo `network-ftp.dsk`

Preparando o Terreno

Primeiro vamos gerar os disquetes, utilize o comando abaixo para gerar o disco de boot:

```
$ cat bare.i > /dev/fd0
```

Repita o comando acima para os demais disquetes conforme o seu método de instalação (NFS ou FTP). Nesta documentação a máquina chamada `smith` será o servidor de onde as estações poderão instalar o **slackware** via NFS ou FTP.

Para a origem da instalação temos três opções: diretamente do CD, mirror local ou mirror remoto (somente FTP). Se for usar diretamente do CD é necessário primeiro montá-lo, insira o cd no driver de CD-ROM e digite o seguinte comando como root:

```
$ mount /mnt/cdrom
```

Você pode copiar o conteúdo do CD para o HD local gerando um mirror local ou baixar de qualquer mirror oficial do **slackware** como o `http://slackware.at`. Deste mirror você pode baixar via `http`, `ftp` ou `rsync`.

Preparando o Terreno para instalação via NFS

Execute o comando abaixo para inserir uma linha no arquivo `/etc/exports`:

```
$ echo "/mnt/cdrom/slackware   * \
       (ro,insecure,all_squash)" >> /etc/exports
```

Caso esteja usando um mirror local substitua no comando acima o diretório "`/mnt/cdrom/slackware`" pelo diretório "`/mirrordir/slackware-9.1/slackware`" do mirror local.

ATENÇÃO

Esta é a famosa "pegadinha", se o diretório exportado pelo NFS não contiver os diretório de `softwares series` (A, AP, D, E, ... X, XAP e Y) a instalação não irá funcionar e o pior, não emitirá nenhum erro.

Com esta linha no `exports` qualquer usuário da rede poderá montar este diretório como somente leitura. Reinicie o NFS para atualizar com a nova configuração:

```
$ /etc/rc.d/rc.nfsd restart
```

Preparando o Terreno para instalação via FTP

Remova a linha "`ftp`" do arquivo `/etc/ftpusers` para ativar o servidor anonymous do `proftpd`. Agora edite o arquivo `/etc/proftpd.conf` alterando a seguinte linha de "`<Anonymous ~ftp>`" para "`<Anonymous /mnt/cdrom>`". Caso esteja usando um mirror local substitua o diretório "`/mnt/cdrom`" pelo diretório "`/mirrordir/slackware-9.1`" do mirror local.

No caso do FTP não tem "pegadinha", porque no momento da instalação é necessário digitar o diretório que contém os diretórios de softwares series. Reinicie o servidor FTP para valer as novas configurações com o seguinte comando

```
$ /etc/rc.d/rc.inetd restart
```

continua

Instalando o slackware via NFS ou FTP

Instalando

Inicie o computador onde deseja instalar com o disco de boot (`bare.i`). Aguarde até que apareça o prompt "`boot:` ", esse prompt serve para passar parâmetros para o kernel do linux na inicialização. Se você não sabe que parâmetro passar ou não precise passar nenhum tecle ENTER.

Aguarde enquanto o kernel do linux é carregado. Quando aparecer "`VFS: Insert root floppy disk to be loaded to RAM disk and press ENTER`" remova o disco de boot e insira o primeiro disco de instalação (`install.1`) e tecle ENTER. Mais uma vez, aguarde enquanto o primeiro disco de instalação do **slackware** é carregado em memória RAM.

Quando aparecer "`Insert install.2 floppy disk to be loaded into RAM disk and press ENTER`" remova o primeiro disco de instalação e insira o segundo (**NFS:** `install.2` ou **FTP:** `install-ftp.2`) teclando ENTER em seguida.

Ao aparecer "`Enter 1 to select a keyboard map:`" se estiver utilizando um teclado US International (sem cedilha) tecle ENTER caso contrário tecle 1 e em seguida ENTER.

Escolha o mapa conforme o seu teclado, se não souber qual escolher e seu teclado possuir cedilha escolha a opção "`qwerty/br-abnt2.map`" e tecle ENTER. Na janela de título "`KEYBOARD TEST`" tecle 1 em ENTER em seguida.

Provavelmente irá aparecer a tela de login com "`slackware login:`" no final. Tecle "`root`" sem as aspas é claro e tecle ENTER. Apesar de simples este passo é MUITO importante, se você teclar algo diferente de "`root`" (minúsculo) terá graves problemas mais para frente.

```
root@slackware:/#
```

Esse é o prompt de instalação do **slackware**. Agora você já pode particionar o seu disco da forma que achar melhor usando `cfdisk` ou `fdisk`. Não vou explicar isso aqui pois não é a intenção desta documentação.

Após ter particionado o seu disco remova o segundo disco de instalação e insira o disco de rede (**NFS:** `network.dsk` ou **FTP:** `network-ftp.dsk`). No prompt digite "`network`" e telcle ENTER. Ele irá pedir para inserir o disco de rede, como você já inseriu simplesmente tecle ENTER novamente. Agora estamos no prompt "`network>`" do disco de rede.

Este é um procedimento específico para a instalação via FTP, caso esteja instalando via NFS pule para o próximo parágrafo. Para futuramente montar a partição via ftpfs precisamos subir agora o módulo do ftpfs teclando F e ENTER.

Em muitos dos casos (provavelmente o seu também) um simples ENTER irá detectar a sua placa de rede e subir o seu respectivo módulo, caso contrário mude para outro console carregue o módulo manualmente com o comando `modprobe`. Se o módulo da placa de rede for carregado com sucesso tecle ENTER para desmountar o disco de rede e voltar ao prompt de instalação. Você pode usar o comando "`lsmod`" para ver se o módulo realmente foi carregado.

Este é um procedimento específico para a instalação via FTP, caso esteja instalando via NFS pule para o próximo parágrafo. Perceba que ao executar "`lsmod`" o módulo `ftpfs` também é listado, caso não apareça você deve carregar o disco de rede novamente para carregar conforme descrito acima.

No prompt digite o comando `setup` e tecle ENTER.

```
root@slackware:/# setup
```

Realmente a brincadeira só começa agora, mas como a intenção desta documentação não é explicar passo a passo toda a instalação do **slackware** vamos pular direto para o passo "`SOURCE Select source media`" supondo que você já tenha efetuado os passos obrigatórios anteriores.

Se estiver instalando via NFS selecione a opção "`3 Install from NFS (Network File System)`" e tecle ENTER. Caso contrário selecione a opção "`4 Install from an FTP server`" e tecle ENTER.

Na próxima tela entre com um IP (ex. 192.168.1.21) para configurar a maquina na qual o **slackware** esta sendo instalado.

Entre com a máscara de rede (netmask), o `setup` assume por padrão a máscara "255.255.255.0" a qual iremos adotar nesta documentação como exemplo.

Agora é a vez do gateway (ex. 192.168.1.254), se o seu servidor onde se encontra as `softwares series` estiver em outra rede você deve configurar um gateway que consiga rotear para a rede dele. Isso vale tanto para a instalação FTP como para a NFS.

No caso de instalação via NFS este é o passo mais importante, o IP do NFS Server (ex. 192.168.1.2). Na próxima tela indique o diretório do NFS Server que contém os diretórios de `softwares series` conforme comentado anteriormente (ex. se estiver usando CD indique `/mnt/cdrom/slackware`, caso contrário o diretório correspondente no mirror.).

Este é um procedimento específico para a instalação via FTP, caso esteja instalando via NFS pule para o próximo parágrafo. Agora você precisa configurar o IP (ex. 192.168.1.2) do FTP Server e o diretório onde estão os `softwares series`. Se você configurou o diretório `root` do usuário `anonymous` como sendo /mnt/cdrom como indicado anteriormente, você deve entrar "`anonymous:senha@192.168.1.2/slackware`".

Caso esteja utilizando um mirror altere o diretório após o IP do FTP Server apontando para o diretório onde se encontra os `softwares series`.

Em seguida o `setup` irá configurar sua placa de rede com os dados fornecidos anteriormente e configurar o gateway se você possuir algum.

Se estiver instalando via NFS o `setup` irá rodar também o `rpc.portmap` para poder montar o NFS e em seguida montar o NFS. E por último o `setup` irá listar a tabela de partições montadas para você verificar se o NFS foi montado corretamente. Caso o NFS tenha montado corretamente tecle `n` e ENTER para continuar com a instalação ou `y` para revisar suas configurações de rede.

continua

Instalando o slackware via NFS ou FTP

Já na instalação via FTP, o setup irá listar a tabela de partições montadas para você verificar se a partição foi montada corretamente utilizando o ftpfs. Caso a partição tenha sido montado corretamente utilizando o ftpfs tecle n e ENTER para continuar com a instalação ou y para revisar suas configurações de rede.

A próxima tela deve aparecer os softwares series para que você selecione os que deseja instalar, se não aparecer provavelmente você errou em algum passo acima.

No caso de ter errado algo você pode selecionar "Cancel" teclando TAB e ENTER para cancelar a instalação e em seguida voltar a selecionar a opção "SOURCE Select source media" para reiniciar as configurações.

Se tudo estiver correto até aqui, agora é só seguir com o procedimento padrão de instalação do **slackware** como se fosse diretamente de um CD.

Conclusão

O mito que a instalação via rede do **slackware** é complicada foi completamente desvendada.
Em testes práticos tive menos problemas com a instalação via FTP do que via NFS. O interessante do NFS é fornecer acesso ao diretório patches para futuras atualizações do sistema em rede otimizando o espaço em disco em diversas máquinas.

Referências

- http://www.slackware.com/getslack/
- http://slackware.at
- http://slackftp.sourceforge.net
- http://www.piterpunk.hpg.ig.com.br
- http://www.google.com.br

Ricardo I. dos Santos <agent.smith@globo.com>

Redes Wi-fi e o slackware

1o. Introdução

Creio que a maioria das pessoas no mundo inteiro tenha wireless hoje em dia, até mesmo você (hein? eu? ..tenho nao), tem sim! Você não tem um celular? Ou até mesmo o controle de uma televisão?

O que muita gente vem confundindo é a palavra 'wireless'. Muitas dizem: "Meu notebook não tem wireless, só infra-red". O infravermelho é um wireless, a palavra wireless quer dizer 'sem fio', ou seja, tudo que é sem fio pode ser chamado de wireless.

Esse artigo pretende abordar alguns aspectos de redes wi-fi usando o Linux, mais especificamente, o **slackware**. Tenha certeza que tenha instalado o pacote wireless-tools (este pacote é padrão da distribuição).

2o. O que é Wi-Fi?

Wi-Fi significa: "wireless fidelity", um apelido para 802.11g como a IEEE (Instituto dos Engenheiros Elétricos e Eletrônicos) a classifica. Usa frequências de 2,4GHz, com uma velocidade até de 54Mbps trimodo. Antes desse tivemos o 802.11a(5Ghz) e 802.11b(2,4Ghz bimodo).

3o. Tipo de redes Wi-Fi

Veremos os dois tipos mais usados de redes wi-fi:

- **Access Point** -> onde você tem uma espécie de um 'hub com antena', que permite acessá-lo de qualquer lugar, conforme sua capacidade.

- **Ad-Hoc** -> para conexões sem um Access Point, podendo interligar duas placas de redes wi-fi (point-2-point).

4o. Placas Wi-Fi no Slackware

O **slackware** 10.0 veio com muitas novidades em se tratandode wireless:

- wireless-tools -> Pacote que contêm programas para você configurar sua rede wireless. Comandos 'iw*'
- linux-wlan-ng -> Módulos para placas wireless Prism (encontra-se no extra/)
- rc.wireless.conf -> Arquivo que contêm configurações pré-definidas para algumas placas
- rc.wireless -> Script de inicialização wireless

No caso de você ter uma placa Prism, basta instalar o pacote linux-wlan-ng-0.2.1pre21_2.6.7-i486-1.tgz que se encontra no extra/ do **slackware** 10.0 e carregar o módulo apropriado para a sua placa:

- prism2_cs Prism2.x & Prism3 PCMCIA
- prism2_pci Prism2.5 (ISL3874) PCI
- prism2_plx Prism2.x PCMCIA com PCI/PCMCIA
- prism2_usb Prism2.x USB

```
# modprobe prism2_pci
```

continua

Agora copiaremos o arquivo de exemplo de configuração da rede wireless:

```
# cp /etc/wlan/wlancfg-DEFAULT \
       /etc/wlan/wlancfg-Wireless
```

Editamos o arquivo de configuração para podermos chamá-lo, `/etc/wlan/wlan.conf`, e trocaremos a opção `SSID_wlan0` para a que criamos acima, no caso '`Wireless`', ficando assim:

```
SSID_wlan0="Wireless"
```

Agora basta executar os comandos:

```
# wlanctl-ng wlan0 lnxreq_ifstate ifstate=enable
# wlanctl-ng wlan0 lnxreq_autojoin ssid=Wireless \
        authtype=opensystem
```

O primeiro comando irá inicializar o driver e as funções MAC, o segundo seta um nome para nossa rede.

E por último, definiremos um IP:

```
# ifconfig wlan0 192.168.0.1 netmask 255.255.255.0
```

> **LEMBRE-SE:**
>
> Essas configurações devem ser feitas apenas para as placas Prism. O artigo cobriu a configuração dela devido o **slackware** vir com os drivers para a mesma.

Existem muitos drivers para Linux, basta uma busca rápida no google e você encontrará o seu. No meu caso, é uma: `00:09.0 Network controller: Broadcom Corporation BCM94306 802.11g (rev 02)`, quando testei o driver do site oficial me descepcionei. Ele era trial, com validade de 30 dias. Até que consegui achar outro driver free, mas não oficial, porém, funciona perfeitamente. Um fato interessante é que alguns drivers, necessitam dos arquivos .inf e .sys para instalação. Para conseguir esses arquivos você terá que fazer o download do driver para versão Windows.

5o. Conectando-se a um Access Point

Para se conectar a uma rede Access Point não tem muito segredo, basta seguir esses comandos como root:

```
# iwlist wlan0 scanning
```

Este comando irá mostrar os possíveis Access Point's que você pode se conectar e seus possíveis ESSID, que são os nomes de redes. Vamos supor que este comando nos retornou algumas redes e uma delas com o nome de 'net business'. Para acessá-la basta escolher o nome da rede com o comando:

```
# iwconfig wlan0 essid "net business"
```

O '`iwconfig`' é semelhante ao '`ifconfig`' só que feito especialmente para ser usado em redes wireless. Geralmente um Access Point é um servidor de DHCP, então o comando:

```
# dhcpcd -d wlan0
```

...basta para conseguirmos um IP e fazer parte da rede :) Ou então, basta definir um IP para a interface wlan0, como fizemos para placa Prism.

6o. Fazendo uma rede Ad-Hoc (point-2-point)

Imagine que você e um amigo foram viajar a negócios e no hotel descobriram que tinham esquecido o cabo de rede cross-over e que precisavam trocar alguns arquivos entre os notebooks, pois uma reunião importante estava para acontecer.

Os dois tinham placas de redes wi-fi e a solução foi deixar uma das placas em modo 'Ad-Hoc':

```
# iwconfig wlan0 mode Ad-Hoc
```

e definir um nome para a rede:

```
# iwconfig wlan0 essid "wireless"
```

Os comandos dois comandos podem ser digitados na mesma linha, por exemplo:

```
# iwconfig wlan0 mode Ad-Hoc essid "wireless"
```

Definimos um IP para nossa interface wlan0:

```
# ifconfig wlan0 192.168.0.1 netmask 255.255.255.0
```

Pronto! Temos um lado da nossa rede mandando sinais para quem quiser capitar.

Na outra ponta, ou no outro notebook - no nosso caso - por desencargo de conciência faremos uma busca por um Access Point:

```
# iwlist wlan0 scanning
wlan0   Scan completed :
        Cell 01 - Address: 8A:3D:1B:4A:9F:CD
              ESSID:"wireless"
              Protocol:IEEE 802.11b
              Mode:Ad-Hoc
              Frequency:2.462GHz
              Quality:0 Signal level:0 Noise level:0
                    Encryption key:off
                    Bit Rate:1Mb/s
                    Bit Rate:2Mb/s
                    Bit Rate:5.5Mb/s
                    Bit Rate:11Mb/s
```

Opa! Achou nossa rede, nos deu o nome da rede, MAC address, protocolo, tipo de rede, frequência, tipo de criptografia, entre outras coisas.

Basta definir o ESSID exatamente como nos retornou:

```
# iwconfig wlan0 essid "wireless"
```

Um IP:

```
# ifconfig wlan0 192.168.0.2 netmask 255.255.255.0
```

E pronto! Os dois notebooks estão interligados, puderam fazer a troca de arquivos e obtiveram sucesso na reunião :) Uma solução simples, rápida e feita com poucos comandos.

Leandro Toledo <toledo@coredumped.org>

Kernel 2.6 e o slackware 10

Para decepção de muitos o **slackware** 10 não veio com a opção para o kernel 2.6 logo na instalação. Mesmo assim, ele vem com todos os utilitários e configurações para se utilizar o 2.6 e, não é todo mundo que sabe, inclusive vem os pacotes com o 2.6.7 pré-compilado no diretório `/testing`.

Ou seja, sem muito trabalho é possível ter o seu **slackware** 10 utilizando um kernel na série 2.6, ao invés do "default", o 2.4.26. A primeira coisa a fazer é baixar os pacotes necessários:

- `kernel-generic-2.6.7-i486-1.tgz`
- `kernel-modules-2.6.7-i486-2.tgz`
- `alsa-driver-1.0.5a_2.6.7-i486-1.tgz`

O primeiro deles é o kernel propriamente dito, o segundo são os módulos e o terceiro são os módulos de som do alsa. Se você é utiliza uma placa Wireless Prism, deve instalar também o pacote: `linux-wlan-ng-0.2.1pre21_2.6.7-i486-1.tgz`, localizado no `/extra/linux-wlan-ng`.

Instalação para quem usa ext2

Se a sua partição `/` for do tipo `ext2` e o seu disco for IDE, a única coisa que precisa fazer é:

```
# removepkg kernel-ide
# installpkg kernel-generic-2.6.7-i486-1.tgz
# upgradepkg kernel-modules-2.6.7-i486-2.tgz
# upgradepkg alsa-driver-1.0.5a_2.6.7-i486-1.tgz
# lilo
```

E agora é só rebootar a máquina.

Instalação para quem usa ext3, reiser, etc...

Mas, se a sua partição `/` não for do tipo `ext2`, ou os seus discos são SCSI, aí a coisa complica um pouco. O kernel que vem compilado no pacote não possui suporte a nenhum outro sistema de arquivos e, muito menos, a controladoras SCSI.

E agora? O que fazer? Bom, o suporte a sistemas de arquivo e a diferentes controladoras de disco está compilado como módulo, basta arranjar alguma maneira de ler esses módulos *ANTES* de montar o `/`. Situação difícil, já que os módulos costumam estar no `/lib/modules` que está no `/`.

Para resolver esse tipo de problemas e possibilitar carregar módulos antes de ter a partição `/` montada existe o `initrd`, que nada mais é que uma ramdisk de inicialização. Assim, você pode guardar os módulos que você precisa dentro dessa ramdisk e não precisa ter todos eles compilados no kernel.

Assim sendo, vamos criar a nossa ramdisk para podermos utilizar o kernel 2.6.7 com o `/` sendo `ext3`, `reiser` ou o que mais acharmos interessante -:). O primeiro passo para isso é instalar o pacote do `mkinitrd`, que está na série A: `mkinitrd-1.0.1-i486-2.tgz`

Depois do pacote instalado, vamos criar esta o initrd. É bem fácil, por exemplo, se você usa reiser na sua partição `/`, deve carregar o módulo do `reiserfs`:

```
# mkinitrd -c -m reiserfs
```

Com isso será criado um initrd com o módulo do reiser dentro dele. O mesmo você deve fazer se precisar carregar mais que um módulo, vamos supor que o seu sistema de arquivos seja `ext3`, o `ext3` depende do `jbd`, então o `jbd` deve ser carregado antes do `ext3`, como no exemplo:

```
# mkinitrd -c -m jbd:ext3
```

E, para não ficar apenas nos exemplos que estão no `README.initrd` (é, existe uma ótima documentação lá), a criação de um `initrd` com suporte a várias placas SCSI da Adaptec (todas as que usem o módulo aic7xxx, como a 2940):

```
# mkinitrd -c -m aic7xxx:jbd:ext3
```

Depois de criado o initrd, altere o `lilo.conf` avisando que o seu kernel vai utilizá-lo, basta inserir a linha:

```
initrd = /boot/initrd.gz
```

Se não souber onde colocar esta linha, coloque logo abaixo de onde é definida a imagem do seu kernel. E é isso, depois destes passos, execute o `lilo` e seja feliz!

Uma última dica antes de terminar o artigo, para saber qual módulo depende de qual, você tem pelo menos duas possibilidades, a primeira é olhar a saída do lsmod
e dar uma olhadinha, por exemplo, para o ext3:

```
# lsmod | grep ext3
ext3                  106088  3
jbd                    47256  1 ext3
```

Com isso nós vemos que o `jbd` é usado pelo `ext3` (veja que ele aparece na última coluna da linha com as informações do `jbd`. Depois você deve fazer a mesma pesquisa para o `jbd`, e assim por diante, até a pesquisa retornar apenas uma linha.

```
# lsmod | grep jbd
jbd                    47256  1 ext3
```

No segundo método, o processo é o mesmo, só que utilizamos o modules.dep, um arquivo localizado no /lib/modules/versão-do-kernel:

```
# grep ext3 modules.dep
/lib/modules/2.6.7/kernel/fs/ext3/ext3.ko: \
       /lib/modules/2.6.7/kernel/fs/jbd/jbd.ko
```

Assim vemos que o `ext3` precisa do `jbd` e de novo fazemos a mesma busca para o `jbd` e por aí vai... Acho que já deu para entender o mecanismo..

Piter PUNK <piterpk@terra.com.br>

Sincronia/Réplica de Arquivos com o Unison

Você provavelmente já ouviu muita gente falar de réplica disso, réplica daquilo, sincronização, etc. Talvez até já tenha brincado com CodaFS, ou InterMezzo, ou mesmo EVMS ou DRBD. Todos esses são implementações no kernel, e requerem uma configuração um tanto quanto complexa, e muitas vezes não são flexíveis o suficiente ou não tem alguma caracteristica especifica pra comportar a sua aplicação. Ou podem ser simplesmente um "overkill", como dizem os gringos.

Aqui vai uma dica de um sistema de réplica simplissimo de montar e configurar, trabalha sobre SSH, e pode mesmo sincronizar um diretório entre o Windows e um Unix.

1. O que é

O Unison trata-se de uma melhoria ou aprimoramento no algorítmo RSYNC. O `rsync` trabalha com sincronização de um lado apenas, ou seja, um "mestre" e todos os outros são "escravos", que simplesmente refletem as alterações do nó mestre. O Unison tenta fazer a sincronização de ambos os lados, conciliando as alterações conflitantes.

O Unison pode usar como transporte uma conexão TCP/IP normal (com o unison rodando em modo daemon ou via inetd), um compartilhamento NFS (ou SAMBA, ou qualquér diretório montado do sistema) ou conexões RSH ou SSH.

Toda a sincronia é feita em ambiente de usuários, não sendo necessário nenhuma alteração ou configuração do kernel.

O Unison pode ser obtido do site

`http://www.cis.upenn.edu/~bcpierce/unison/`

2. Como funciona

O Unison primeiramente levanta uma lista de sincronia de arquivos do diretório especificado e solicita que o Unison do "outro lado" faça o mesmo.

Se houverem novos arquivos/diretorios de qualquér dos lados, o Unison réplica essas diferenças. Quando há 2 arquivos conflitantes, o Unison permite que se escolham diversos cursos de ação: pode-se forçar um servidor a ser o mestre (quando conflitar, use a cópia do servidor X), ou pode-se escolher o mais recente como mestre, ou pode-se fazer uma mistura ("merge") entre os 2 arquivos conflitantes (com o `diff`, por exemplo).

O esquema de "merge" para arquivos conflitantes requere que você faça scripts para juntar os arquivos. Se arquivos/diretorios forem removidos, o Unison réplica essas alterações também.

3. Aplicações Práticas

Aqui vão algumas idéias de como usar o mecanismo de réplica:

- **Backups** - Mantenha um servidor de reserva com os dados do servidor principal espelhados nele.
- **Sincronizar Home Directories** - Manter seu Home Directory em casa sincronizado com o seu Home Directory no Serviço e vice-versa. Nada mais de ter que fazer SSH pra ver aquele texto que você esqueceu em casa :)

Mas espere um pouco. Se eu posso fazer a réplica de alterações para ambos os lados, por que manter um servidor de reserva? Por que não deixa-lo rodando em paralelo com outro server, já que eles vão sincronizar alterações/novos arquivos?

É aqui que o Unison fica realmente interessante. Com ele podemos montar aplicações de alta disponibilidade sem precisar mexer no kernel com replicações que nem sempre funcionam. Se eu fizer isso distribuído, ainda posso aumentar o desempenho da aplicação por distribuir a carga.

Esse tipo de sincronia funciona maravilhosamente bem com caixas de correio Maildir. No ambiente de email não há necessidade de sincronia imediata de mensagens (podemos sincronizar de 1 em 1 minuto ou mais, por exemplo, que não haverá problemas com as aplicações)

Outras aplicações que trabalhem com arquivos independentes (que não sejam alterados frequentemente e nem simultaneamente) também poderao tirar muito benefício dessa sincronização sem se preocupar com merges de arquivos.

4. Mãos a obra!

Chega de bla bla bla e vamos partir pra parte prática.

4.1. Instalando

No site do Unison já existem versões pré-compiladas estáticas do Unison. Pegue a versão em modo texto (`unison.linux-static-textui`) e copie para o `/usr/bin/unison` . Se quiser compila-lo você mesmo, vai necessitar do compilador Objective Caml (versão 3.04 ou superior).

4.2. Configurando

Vamos primeiro configurar o Unison pra replicar o Home Directory e depois para replicar Maildirs de emails.

4.2.1. Replicando o Home Directory

Crie no seu Home Directory o diretório `~/.unison` . Agora vamos criar uma configuração chamada "home". O arquivo deverá chamar `home.prf`. E o seu conteúdo está listado na próxima página:

continua

```
#~/.unison/home.prf
# Unison preferences file
root = /home/thefallen/
# Dica: SEMPRE coloque // entre o nome da máquina
# e o diretório a sincronizar
root = ssh://outra.máquina.com//home/thefallen/
# Caso seu ssh esteja em outra porta...
#rshargs = -oPort=2222
# Diretorios que quero sincronizar
path = Docs
path = MyDocs
path = c
# Arquivos que não devem ser replicados
ignore = Name *.log*
ignore = Name .*
# diretório que não deve ser replicado
ignore = Path tmp
# Não replique diretorios/arquivos que tenham "tmp"
# no nome
ignore = Regex .*tmp.*
# batch == roda sem fazer perguntas; indicado para
# tarefas cron e similares
#batch = true
# encare a cópia mais recente como master; não
# tente fazer "merge"
prefer = newer
# Piloto automatico :)
auto = true
# Sincronize a hora de modificação.
times = true
log = true
logfile = /home/thefallen/.unison/réplica.log
```

Agora digite "`unison home`" e veja o Unison sincronizando seu Home Directory :). Você pode usar ssh com autenticação por chave pública, e nem a senha vai precisar digitar :)

4.2.2. Replicando diretorios de email Maildir

Supondo que seu diretório de emails no formato Maildir seja `/var/vmail`, e que o outro servidor seja `10.10.0.2`, e que você esteja usando o Courier IMAPd. Crie o arquivo de configuração maildir.prf dentro do `~/.unison/`

```
#~/.unison/maildir.prf
root = /var/vmail/
# lembre-se: SEMPRE com 2 / depois do nome
# da máquina
root = ssh://10.10.0.2//var/vmail/
# medidas de seguranca pra não copiar arquivos
# temporarios de entrega de mensagem ou do IMAP
ignore = Name maildirsize
ignore = Name courier*
ignore = Path */tmp/*
# AutoPilot is on :)
auto = true
batch = true
times = true
# Prefiro a cópia mais recente como master
prefer = newer
# se vc quiser que o unison replique donos/owners
# de arquivo ; precisa do poder do root
#owner = true
```

4.2.3. Replicando XYZ

No site do Unison existe uma documentação muito boa (em inglês). Você pode inventar muitas maneiras de aproveitar o sistema de replicação de arquivos do Unison (lembra do "Neston"? Invente uma!)

4.3. Múltiplos Servidores

O Unison, por natureza, trabalha apenas com replicação de dados entre 2 máquinas. Para colocar múltiplos servidores na replicação, você precisar colocar múltiplas replicas entre os servidores (ou manter um dos servidores como um nó "central", e todos vão replicar com ele). O script `syncmgr` ajuda nessa hora; ele só vai permitir que um servidor sincronize por vez.

4.4. Uma palavra de cautela

Embora o Unison seja preparado para lidar com replicações simultâneas (você replicaria daqui pra lá e eu replicaria de lá pra cá os mesmos arquivos), esse recurso não foi exaustivamente testado e convém evitar essa situação. Com alguns scripts, conseguimos evitar essas situações. Veja no próximo item.

4.5. Automatizando as replicas

Você pode por a replicação pra rodar no `cron`, se quiser. No entanto, uma coisa me ocorreu. E se uma replicação demorar mais do que o intervalo do `cron` e eu iniciar uma replicação quando a outra ainda estiver rodando (se eu colocar pra sincronizar de 1 em 1 minuto por exemplo)? Já imaginou a bagunça (e uso de CPU e Disco) que isso poderia causar?

Para escaparmos dessa "race-condition", basta mantermos um único processo que faz a rotina de sincroniza-espera-sincroniza-espera... Podemos melhorar ainda mais essa idéia colocando alguns mecanismos básicos de travamento que impedem (ou tentam impedir) que ocorram 2 sincronizações (tanto remotas como locais) de um mesmo perfil de configuração.

4.5.1. cron job

Simplesmente coloque uma tarefa no cron com a seguinte sintaxe:

```
*/15 * * * * /usr/bin/unison maildir \
       -silent -batch -auto
```

Se algo acontecer que o unison não consiga resolver, o `cron` vai lhe enviar a saída do comando por email.
A flag `-silent` faz com que sejam apresentados apenas erros e a flag `-batch` diz ao unison pra não perguntar nada ao usuário.

4.5.2. Init service

Você pode também criar uma tarefa no INIT que mantem o serviço de replicação rodando. Dessa maneira, apenas ocorre 1 sincronização por vez (não há o perigo de "encavalar" como poderia acontecer com o cron) e caso o script caia ou você mate inadvertidamente o script, o INIT o inicia de novo.

continua

De permissão de execução (com chmod 755 remotesync syncmgr) e adicione uma linha parecida com essa no /etc/inittab:

```
# /etc/inittab:
s001:2345:respawn:/bin/su - \
       USUARIO_QUE_DEVERA_REPLICAR -c \
       "/opt/filesync/bin/syncmgr maildir"
```

Faça um reload no init com o comando "telinit q" ou "init q" e acompanhe os logs do que acontece no /opt/filesync/var/log/.

4.5.3. Processo em background

Se você não gosta da idéia de colocar servicos no INIT, pode simplesmente coloca-lo em background com o auxílio do comando nohup. Bastaria rodar o comando:

```
nohup /opt/filesync/bin/syncmgr  maildir \
       1>/dev/null 2>/dev/null 0</dev/null &
```

Para para-lo, basta mandar um kill -9 no proceso (ou killall -9 syncmgr).

Você pode fazer um script de rc para ele. Segue um exemplo:

```
#!/bin/sh
# /etc/rc.d/rc.unison
case "$1" in
  start)
    echo -n "Iniciando replicação UNISON: "
    echo -n " maildir"
    su nobody -c \
       "nohup /opt/filesync/bin/syncmgr maildir \
       1>/dev/null 2>/dev/null 0</dev/null &"
    echo -n " home"
    su thefallen -c \
       "nohup /opt/filesync/bin/syncmgr home \
       1>/dev/null 2>/dev/null 0</dev/null &"
    echo ""
    ;;
    stop)
      echo "Parando replicação UNISON"
      killall -9 syncmgr 2>/dev/null
    ;;
esac
```

Coloque o script dentro do diretório /etc/rc.d, com o nome de rc.unison. Dê a ele permissão de execução e acrescente estas linhas no seu /etc/rc.d/rc.M:

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

4.5.4. syncmgr e remotesync

Vamos agora criar os scripts. Crie o diretório /opt/filesync e os subdiretorios bin, var/log e var/lock. Vamos agora criar 2 scripts no /opt/filesync/bin/. O primeiro deles, syncmgr inicia as conexões, enquanto o remotesync garante que elas não ocorram ao mesmo tempo.

As listagens de ambos os scripts estão nas próximas páginas. Figura 1.1 e 1.2 no caso do syncmgr e Figura 2 para o remotesync.

Convém que você faça a primeira replicação manualmente, sem o modo batch. Caso você tenha configurado algo errado, vai perceber :) .

> **NOTA**
>
> se for utiliza-lo rodando pelo cron, edite o syncmgr e remova a linha que faz o loop

5. Conclusão

Como pudemos observar, o Unison é um software rico em recursos, sendo que apenas uma pequena parte deles foi abordada aqui.

Quando descobri esse software fiquei bastante impressionado, especialmente com o fato dele ser completamente em ambiente de usuário, inclusive com transporte SSH para as comunicações.

Já tenho o unison rodando aqui há algumas semanas, e até agora não houve nenhum problema, e a performance das replicações foi agradavelmente surpreendente.

Pra quem gostou da idéia, vale explorar suas opções; o site possui uma documentação muito boa e explicativa.

Deives Michellis <thefallen@unitednerds.org>

Figura 1.1 – Arquivo /opt/filesync/bin/syncmgr

```
#!/bin/sh
# Arquivo /opt/filesync/bin/syncmgr
#
# Copyright (c) 2004, Deives Michellis
#    All rights reserved.
#
#    Redistribution and use in source and binary forms, with or without
#    modification, are permitted provided that the following conditions are
#    met:
#      * Redistributions of source code must retain the above copyright
#        notice, this list of conditions and the following disclaimer.
#      * Redistributions in binary form must reproduce the above copyright
#        notice, this list of conditions and the following disclaimer in
#        the documentation and/or other materials provided with the
#        distribution.
#      * Neither the name of the Deives Michellis nor the names of its
#        contributors may be used to endorse or promote products derived
#        from this software without specific prior written permission.
#
#    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
#    "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
#    LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
#    A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
#    OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
#    SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
#    LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
#    DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
#    THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
#    (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
#    OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
export PATH=/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin

DATADIR=/opt/filesync/
# Qto tempo esperar se a sincronização já estiver acontecendo (remotamente)
# Numero de vezes q a verificação de sincronia remota é feita
MAXWAIT=12
# Quanto tempo espera em cada verificação
SLEEP=10
# De quanto em quanto tempo tentar sincronizar
INTERVAL=180
# quem informar caso "algo aconteca"
EMAILADDR=seu_email@aqui.com

# Verificação do nome do profile
SYNCNAME=`echo $1 | sed -e 's/[^a-z0-9]//gi'`
if [ "$SYNCNAME" = "" ]; then
  echo "Invalid Profile name"
  exit 1
fi

trap "rm -f $DATADIR/var/lock/$SYNCNAME/$$ \
       $DATADIR/var/log/réplica.$$.log" 0 1 9 15

if [ ! -d $DATADIR/var/lock/$SYNCNAME ]; then
  mkdir -p  $DATADIR/var/lock/$SYNCNAME/
fi
```

continua na Figura 1.2

continua

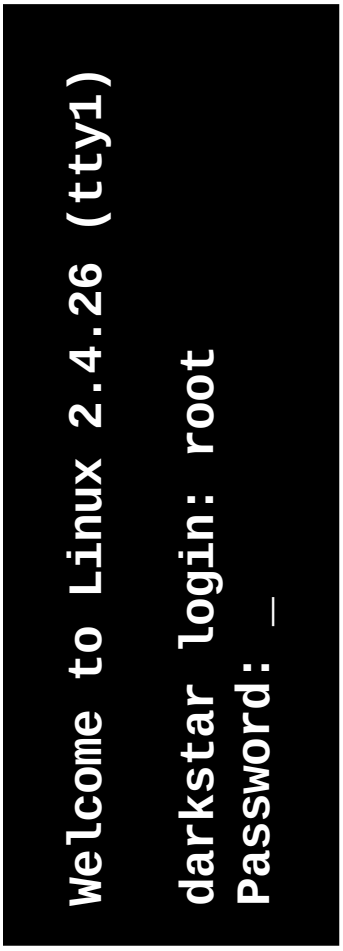

Figura 1.2 – Continuação do Arquico /opt/filesync/bin/syncmgr

```
# Main loop - remova a linha do "while" se for usa-lo via cron
while true; do
  COUNT=0
  while [ `ls -1 $DATADIR/var/lock/$SYNCNAME/ 2>/dev/null| wc -l ` -gt 0 ]; do
    if [ $COUNT -ge $MAXWAIT ]; then
      break
    fi
    COUNT=`expr $COUNT + 1`
    sleep $SLEEP
  done
  touch $DATADIR/var/lock/$SYNCNAME/$$
  echo "Subject: REPLICA LOG $SYNCNAME
Date: `date -R`
From: Unison Sync Manager <$EMAILADDR>
" > $DATADIR/var/log/réplica.$$.log
  unison $SYNCNAME -servercmd "$DATADIR/bin/remotesync $SYNCNAME" -dumbtty \
    -batch >>$DATADIR/var/log/réplica.$$.log 2>>$DATADIR/var/log/eplica.$$.log

RETVAL=$?
  rm -f $DATADIR/var/lock/$SYNCNAME/$$

  if [ $RETVAL -eq 2 ]; then
    /usr/sbin/sendmail -f MAILER-DAEMON -- $EMAILADDR \
      < $DATADIR/var/log/réplica.$$.log
  fi
  sleep $INTERVAL
# Fim do Main loop - remova o "done" se for usa-lo via cron
done
```

Figura 2 – Arquivo /opt/filesync/bin/remotesync

```
#!/bin/sh
# Arquivo /opt/filesync/bin/remotesync
#
export PATH=/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin

DATADIR=/opt/filesync/
MAXWAIT=12
SLEEP=10
SYNCNAME=`echo $1 | sed -e 's/[^a-z0-9]//gi'`

trap "rm -f $DATADIR/var/lock/$SYNCNAME/$$" 0 1 9 11 15

if [ ! -d $DATADIR/var/lock/$SYNCNAME ]; then
       mkdir -p  $DATADIR/var/lock/$SYNCNAME/
fi
COUNT=0
while [ `ls -1 $DATADIR/var/lock/$SYNCNAME/ 2>/dev/null| wc -l ` -gt 0 ];
do
  if [ $COUNT -ge $MAXWAIT ]; then
    break
  fi
  COUNT=`expr $COUNT + 1`
  sleep $SLEEP
done
touch $DATADIR/var/lock/$SYNCNAME/$$
unison -server
rm -f $DATADIR/var/lock/$SYNCNAME/$$
```

Slackware
Keep It Simple Stupid

Post-Install slackware 10

Saiu mais um release do **slackware**. Como sempre (e como em qualquer distribuição) são necessárias algumas alterações para deixar o sistema "redondo". Seguindo este documento voce pode fazê-las em menos de 5 minutos, que irão valer cada segundo.

1. Montando o CDROM e Floppy

O padrão do **slackware** é permitir apenas que os donos dos dispositivos de `cdrom` e `floppy` possam montá-los. Na prática, isso quer dizer que apenas o root vai conseguir.

Para resolver esse problema e permitir aos seus usuários que possam ler CDs e ler e escrever em disquetes, basta editar o `/etc/fstab` e, nas linhas:

```
/dev/cdrom /mnt/cdrom  iso9660 noauto,owner,ro 0 0
/dev/fd0   /mnt/floppy auto    noauto,owner    0 0
```

Troque onde está "`owner`" por "`users`". Aproveitando a deixa, dê permissão ao seu usuário para que possa ler o drive de CD diretamente (para tocar músicas por exemplo):

```
# chmod 666 /dev/cdrom
```

Pronto! Agora os seus usuários podem ler e escrever o que quiserem e ainda escutar um CDzinho...

2. Curtindo um Som

A primeira coisa a fazer para "curtir um som" é aumentar o volume. O ALSA vem (como padrão) com todos os canais no mudo, não me perguntem o porquê. Para corrigir esse pequeno defeito, utilize o "`alsamixer`" para regular o volume do som (e tirar os canais do mudo), em seguida, armazene as novas configurações com:

```
# alsactl store
```

Com isso, o root pode escutar todos os sons que quiser. Agora temos uma questão polêmica: no slackware 10, todos os dispositivos de som tem como grupo o "`audio`" o que indica claramente que a maneira correta de fazer com que os seus usuários utilizem o som seja incluí-los nesse grupo.

Este seria o "jeito certo". Particularmente, ao invés de incluir todos os meus usuários um por um no grupo "`audio`", prefiro mudar o grupo dos dispositivos de som para o "`users`", assim eu tenho o mesmo efeito (e preciso fazer a alteração uma vez só):

```
# chgrp -R users /dev/dsp0 /dev/mixer0 \
       /dev/sequencer /dev/snd
```

Agora os seus usuários também podem escutar o som que quiserem.

3. Desligamento Automático

Quem possui um computador com fonte ATX (praticamente todo mundo hoje em dia), deve gostar quando usamos o comando "`shutdown`" e ele desliga automaticamente. Para fazer isso, edite o `/etc/rc.d/rc.modules` e descomente (retire o `#` da frente) a linha

```
#/sbin/modprobe apm
```

No seu próximo boot, a o módulo `apm` será carregado e o `apmd` também, automaticamente.

4. Configurando o X

No X existem duas configurações básicas a fazer, carregar o mapa correto de teclado e arrumar o protocolo do mouse. Claro, existem várias outras, mas não se encaixam em um documento sobre um "postinstall" rápido...

4.1. O mapa de teclado

Os dois teclados mais comuns no Brasil são o ABNT2 e o US-Internacional. Então iremos configurar os dois (já que são os mais comuns).

Edite o `/etc/X11/xorg.conf` e inclua as seguintes linhas na parte de configuração de teclado (se houver alguma dúvida sobre onde colocar essas linhas, coloque-as logo após a linha que diz `Driver "Keyboard"`)

Para teclados ABNT2:

```
Option "XkbRules"   "xorg"
Option "XkbModel"   "abnt2"
Option "XkbLayout"  "br"
```

Para teclados US-Internacional

```
Option "XkbRules"   "xorg"
Option "XkbModel"   "pc105"
Option "XkbLayout"  "us_intl"
```

Troque onde está `pc105` por `pc102` se o seu teclado for daqueles SEM as teclas "Windows" e "Menu".

4.3. Configurando a Rodinha do Mouse

Configurar a rodinha do mouse também é rapidamente resolvido editando o `/etc/X11/xorg.conf`, troque onde está `PS/2` por `IMPS/2` na linha em que está escrito:

```
Option "Protocol" "PS/2"
```

Depois inclua mais estas linhas com algumas configurações adicionais:

```
Option      "ZAxisMapping" "4 5"
Option      "Buttons" "5"
```

Pronto! Se a rodinha não funcionar, provavelmente você deve trocar o protocolo. Vários outros além do `IMPS/2` suportam a "rodinha", como o `ExplorerPS/2`, `NetScrollPS/2`, `Intellimouse` (serial), etc...

continua

Post-Install slackware 10

5. Impressora

A configuração da impressora já foi abordada com detalhes nos números #2 (CUPS) e #3 (LPRng) do slackwarezine. Independente de qual dos dois você resolver usar, lembre-se de descomentar o trecho referente a impressora no /etc/rc.d/rc.modules:

```
#if cat /proc/ksyms | grep "\[lp\]" ...
...
um monte de linhas
...
#    /sbin/modprobe lp
#  fi
#fi
```

Basta retirar o # da frente de todas essas linhas (inclusive das listadas como "`um monte de linhas`") que o módulo será carregado. Depois disso, utilize o seu sistema de impressão favorito (dica de amigo: `/usr/share/apsfilter/SETUP` e use o LPRng).

6. Por último...

Mas não menos importante, o famigerado, miserável e maldito (para ficar apenas nos adjetivos publicáveis) c acentuado! E morte ao infeliz que resolveu fazer a GTK ter um mapa de teclados próprio e não ler o do X, espero que seja condenado a usar WindowsXP em um P100!!

Bom, depois do desabafo, para resolver o problema da GTK basta editar o `/etc/profile.d/gtk+.sh` e colocar a seguinte linha:

```
export GTK_IM_MODULE=xim
```

E pronto. E... pronto mesmo! -:) Acabou. Claro que você pode configurar o X com o `xorgconfig` ou com o `xorgcfg`, pode manter o seu CDROM como para montagem só do `owner`, etc... Este artigo serve mais para passar umas dicas e uma lista de coisas para não serem esquecidas, nada pior que depois de estar usando a máquina descobrir que algo não está funcionando.

Uma última dica, lembre-se de, logo após a instalação, verificar se existem patches de segurança para serem instalados. Isso pode ser feito através de ferramentas como o `slackpkg` (que está no diretório `/extra` dos CDs de instalação do **slackware**) ou acessando o mirror mais próximo. Uma boa lista de mirrors pode ser encontrada no próprio site do **slackware**, mas não é muito atualizada...

Os pacotes com as correções de segurança ficam dentro do diretório "`/patches/packages`" nos mirrors. Baixe os pacotes encontrados e faça a atualização com o `upgradepkg`. Se preferir usar o `slackpkg`, instale o pacote, edite o `/etc/slackpkg/mirrors` e faça:

```
# slackpkg update
# slackpkg upgrade patches
```

E por hoje é só! -:)

Piter PUNK <piterpk@terra.com.br>

AUTORES

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

Leandro Toledo, 19 anos, iniciou com computadores em 1993 e Linux em 1998, usando Slackware 3.4 kernel 2.0.30. Desde entao, vem acompanhando a evolução desse maravilhoso sistema juntamente com toda a comunidade de softwares livres. Trabalha na área a 2 anos, atualmente, como sysadmin de redes linux numa agência de publicidade.

misfit, 23 anos, bacharel em SI pelo Mackenzie/SP, LPI-1 certified, linux user #215987. Participante dos grupos Linuxchix-BR e GUS-BR, contribui para o projeto linuxbr.org nas horas vagas e seus interesses incluem segurança, programação O.O., dipositivos embedded e punk rock.

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 no desenvolvimento de jogos para dispositivos móveis na 3WT Corporation.

Reinaldo Nolasco Sanches (r_linux), atualmente trabalhando na Mandic LTDA no desenvolvimento de aplicações para a Web. Graduando-se em Bacharel de Sistemas de Informações pela Universidade Presbiteriana Mackenzie. Meus interesses são C/C++, Qt, Linux, Sistemas Distribuidos, Programação de Games, Inteligência Artificial, Computação Evolutiva e Heavy Metal.

Ricardo Iramar dos Santos (Agent Smith), Engenheiro Elétrico no papel, Analista de Segurança na carteira de trabalho, Peão na empresa e Programador/Curioso na essência. Trabalha atualmente na Etek porém locado fisicamente em empresa parceira chamada AT&T Latin America recentemente adquirida pela Telmex. Conheceu o Linux através do Conectiva 4.2, passou por várias distribuições até conhecer a versão 8.0 do Slackware o qual adotou como distribuição predileta.

A equipe do **slackwarezine** agradece todas as colaborações, críticas e sugestões que tem recebido.

Se você também quer participar envie o seu artigo, sugestão ou crítica para:

editor@slackwarezine.com.br

Seu artigo será avaliado e, dependendo de uma série de fatores (inclusive espaço disponível) será publicado. O artigo não precisa ser específico sobre **slackware**, mas quando escrevê-lo, lembre-se que é o **slackwarezine** -;)

Dúvidas sobre artigos publicados devem ser enviadas diretamente a seus autores.