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

05 de Abril de 2005 – Edição #8

Editorial

Chegamos ao slackwarezine número 8. E bem atrasados, pela primeira vez, uma edição normal não saiu em um mês ímpar. Para tentar minimizar esses atrasos, na página 11 temos algumas dicas sobre como escrever artigos para o zine, e sobre os prazos para a entrega deles.

Se uma instalação “full” com o sistema instalando automaticamente apenas os pacotes que você escolheu lhe empolga, dê uma lida no artigo do Fábio na página 7.

Os paranóicos de plantão vão se divertir com a integração entre GnuPG e VIm, um ícone da criptografia com a nata dos editores de texto; uma amálgama perfeita trazida por Deives Michellis

Os programadores não foram esquecidos, e desta vez contam com um artigo sobre programação 3D. Com ele pode-se aprender os fundamentos do OpenGL e, parece, que de onde veio esse vem ainda muitos mais -;)

E ainda tem vários outros artigos, para iniciantes e veteranos, tudo do bom e do melhor! Afinal, o leitor do slackwarezine merece o melhor.

Piter PUNK

Índice

slack users

Sem boot pelo CD? Sem disquete? Se tem linux, a instalação é garantida

Sei bem como é, você tem um computador em que dá tanto trabalho instalar o linux no infeliz que está desde 98 com a mesma instalação, pois em 98 foi a última vez em que se deu ao trabalho de fazer uma série de malabarismos exóticos que só de pensar já te desanima tentar de novo.

Notebooks sem boot pelo CD-ROM e sem drives de disquete (é, ele até vinha com um drive de disquete, mas o tempo cuidou de desaparecer com ele), velhos 486 All-In-One cujo drive de CD apenas lê CDs originais, etc, etc e etc... Estava olhando uma máquina problemática um dia desses e pensando no quê fazer. Queria fazer um fresh-install mas não queria ter muito trabalho... Até que me veio uma iluminação!

Para iniciar o procedimento de instalação do **slackware**, é necessário primeiramente ler o `kernel` e, em seguida ler o `initrd.img`, onde se localizam todos os programas de instalação. Se o meu problema for só "entrar" no sistema de instalação (o que seria o caso do notebook sem boot pelo CD e sem drive de disquete), basta conseguir ler o `kernel` e o `initrd` e tudo se resolve.

Como fazer isso? Fácil, primeiro, copie um dos `kernels` do **slackware** para o seu `/boot`. Os arquivos que precisamos se encontram em `/kernels/nomedokernel`. Assim, para copiar o `bare.i` devemos fazer:

```
# cp /mnt/kernels/bare.i/bzImage \
      /boot/install10.1
```

Em seguida, vamos copiar o `initrd`, que também será necessário:

```
# cp /mnt/isolinux/initrd.img \
      /boot/initrd-install10.1
```

Agora que os arquivos estão no lugar, basta editar o `lilo.conf`, criando uma entrada no LILO que irá ler o `kernel` da instalação e em seguida o `initrd.img`. A partir daí, podemos instalar o **slackware** normalmente, como se houvéssemos bootado pelo CD.

Edite então o `/etc/lilo.conf` e inclua as linhas:

```
image = /boot/install10.1
  initrd = /boot/initrd-install10.1
  label = install10.1
  read-only
```

Depois é só rodar o comando `lilo`:

```
# lilo
```

Reboote a sua máquina e escolha a opção "`install10.1`" no menu de inicialização.

Caso o seu problema envolva uma máquina sem drive de CD, é bom que ela tenha pelo menos uma placa de rede, assim podemos fazer a instalação via rede. Claro, continuamos sem poder usar os disquetes, que já era o nosso problema inicial...

Nesse caso, junto com o `initrd` e o `kernel`, você deve copiar o `network.dsk`. Aí tem um pouco de "truque" envolvido.

A instalação do **slackware** está preparada para ler o `network.dsk` de um disquete ou do CDROM. Não de uma partição no HD... a enganação é simples:

```
# mkdir /isolinux
# cp /mnt/rootdisks/network.dsk \
      /isolinux
```

Agora, reboote a sua máquina e escolha a opção "`install10.1`". Logo após entrar como usuário `root`, já no prompt, monte a sua partição `/` e rode o script `network`, como se estivesse efetuando uma instalação totalmente normal:

```
# mount /dev/hdxn /cdrom
# network
# umount /cdrom
```

Pronto! Você está no sistema de instalação e com sua placa de rede configurada, pronto para fazer uma instalação novinha em folha do **slackware** nessa máquina! Esse pequeno truque de instalação ajuda bastante é fácil de fazer!

Piter PUNK <piterpk@terra.com.br>

Usando o GnuPG dentro do VIm

Para aqueles que gostam de usar o VIM pra editar textos, e tem, digamos, uma certa queda fanático-religiosa por segurança, vão achar esses scripts de `vim` muito a seu gosto :)

Vamos ver como integrar o GnuPG (GNU Privacy Guard --> versão "open" do PGP - Pretty Good Privacy) direto no `vim` sem precisar de arquivos intermediários para isso, de uma forma bem simples e rápida.

Constitui-se basicamente em 2 partes - um conjunto de macros para assinar/verificar/"desassinar" um texto e um script do `vim` para gerenciar arquivos criptografados (`arquivo.gpg`).

Assinando textos

Adicione as seguintes macros no seu `~/.vimrc`:

```
# ~/.vimrc
... seus outros comandos ....

command! GS :%! gpg --clearsign
command! GU :%! gpg –no-verbose \
                  2>/dev/null
command! GV :w    !gpg --verify \
                  --no-verbose
# Note o espaço em
# :w <ESPAÇO> !gpg ..... ; é
# importante para o funcionamento
# correto do comando :)
```

O comando `:%!` no `vim` faz com que a mensagem passe por um filtro externo e volte para o texto. Assim, o `gnupg` vai assinar o texto e devolver pro `vim` (você vai ter que digitar sua pass-phrase para que o `gnupg` assine a mensagem). O `:w !comando` faz um "pipe" do texto atual para o comando especificado.

ATENÇÃO: se você não colocar o espaço, o vim vai salvar o arquivo com o nome do comando. Lendo a documentação do `vim`, é alguma compatibilidade obscura com o `vi` antigo e/ou o modo `ex`, ou algum Hocus Pocus similar.

Basta apertar `ESC` no `vim` e digitar `:GS` para assinar o texto, `:GV` para verificar uma assinatura e `:GU` para limpar a assinatura.

Criptografando textos

Vamos precisar de um plugin do `vim` para fazer isso. Plugins nada mais são que coleções de macros e scripts pro `vim` que serão automagicamente acionados em algum evento.

O script/plugin pode ser obtido direto do site do `vim`:

```
http://www.vim.org/scripts/\
      script.php?script_id=661
```

Basta salvá-lo no diretório `~/.vim/plugin/` com o nome `gnupg.vim`.

Agora abra um arquivo com extensão `.gpg`. O plugin automaticamente entra em ação na hora de salvar este arquivo, perguntando para quais usuários/chaves se quer criptografar o email. Pode-se criptografar de maneira que n chaves privadas consigam abrir o arquivo, basta ir adicionando nomes/emails/ids de chaves na hora de salvar (o plugin pede essas informações).

Então quando for me enviar algo, aproveite seu novo `vim` e manda criptografado para a chave `0x9BD77249` :)

Deives Michellis <thefallen@unitednerds.org>

slackwarezine

mais de um ano de tradição!

Programação 3D com OpenGL

Visão Geral

O OpenGL API(Application Programming Interface) foi desenvolvido pela Silicon Graphics Inc.(SGI) visando a criação de uma especificação para o desenvolvimento de aplicações gráficas 3D multiplataforma. Para poder utilizá-la em um programa, C por exemplo, basta referenciar a biblioteca GLUT com o comando `#include`.

Esta biblioteca foi projetada para ser uma interface bem transparente para o programador, desse modo ele programa as transformações 3D sem se preocupar com o controle do hardware.

OPenGL utiliza um padrão de controle de ações baseado em uma máquina de estados, este padrão trabalha com a habilitação dos comandos utilizados nas aplicações, uma vez ativados, estes comandos, estarão com o estado definido até o fim da aplicação ou até serem modificados ou desativados em um trecho mais a frente do código da aplicação.

A totalidade das opções, em OPenGl respeitam este princípio, podendo ser de qualquer tipo, tais como cores, posições, iluminação, transformações. Existem implementações da mesma para diversas linguagens e plataformas. As suas 200 funções dividem-se em cinco categorias:

- **Funções de primitivas:** definição de objetos 2D, 3D e imagens;
- **Funções de atributos:** definição de cores, linhas, propriedades de material, iluminação e texturas;
- **Funções de visão:** determina a posição, orientação e modo de projeção da câmera (visão do usuário);
- **Funções de entrada:** embora não seja a finalidade original da OpenGL, a biblioteca GLUT(GL Utility Tool Kit) implementa os eventos de teclado e mouse e busca tornar os programas multiplataforma.
- **Funções de controle:** todos os comandos que não sejam de exibição e nem de interação.

No mundo virtual de três dimensões o eixo x é horizontal e aponta para a direita. O eixo y é vertical e aponta para cima e o eixo z é a profundidade e aponta para fora da tela.

O uso de matrizes é bastante comum quando se trabalha com computação gráfica. Em OpenGL as matrizes não precisam ser conhecidas e nem acessadas diretamente pelo programador. Já são implementadas as funções matemáticas necessárias para rotação, translação (deslocamento), escala (mudança na proporção) e projeção perspectiva (exibição do mundo 3D na tela 2D).

Compilando programas que utilizam a Glut com o gcc.

Normalmente, nas distribuições Linux mais populares, tudo o que se precisa fazer para compilar um programa que use OpenGL é instalar o pacote `glut-3.7-i486-1`. Para compilar programas que utilizam a glut em C, basta utilizar a seguinte linha de comando:

```
# gcc nomedoprograma.c \
      -o nomedoprograma \
      -L/usr/X11R6/lib -lglut
```

Animação 3D

O programa que veremos logo a seguir, desenha um cubo e o rotaciona continuadamente de grau em grau em torno dos eixos y e z até o usuário pressionar `ESC` e encerrar o programa. Seguem no texto as linhas do programa exemplo comentadas.

Este comando, inclui todas as funções OpenGL e GLUT necessárias

```
#include <GL/glut.h>
```

`Teclas()` é a função que será chamada sempre que uma tecla for pressionada. O condicional `IF` verifica se a tecla é `ESC` e encerra o programa se for o caso. A espera de 10 milisegundos é para evitarmos a perda do evento.

```
void Teclas( unsigned char key,
             int x, int y) {
   if (key == 27) {
      usleep(10000);
      glutDestroyWindow(glutGetWindow());
      exit(0);
   }
}
```

continua...

Declaração da variável que guarda o ângulo da rotação em graus.

```
short int rot=0;
```

Função de desenho da cena.

```
void DesenhaCena() {
```

O `glLoadIdentity()` serve para esquecer todas as transformações feitas até o momento. Como a função `DesenhaCena()` será chamada constantemente, é importante reiniciar a matriz de transformações.

```
glClear(GL_COLOR_BUFFER_BIT);
glLoadIdentity();
```

Desloca o cubo para dentro da tela (para mais longe do usuário), caso contrário o observador ficaria dentro do cubo. O primeiro parâmetro é o deslocamento em x, o segundo em y e o terceiro em z.

```
glTranslatef(0.0f,0.0f,-15.0f);
```

Operação para rotacionar, fica sempre entre 0 e 359.

```
rot = rot % 360;
```

Rotação de "rot" graus em torno dos eixos y e z. O primeiro parâmetro é o ângulo em graus. Os outros três indicam em que eixo rotacionar.

```
glRotatef(rot++,0.0f,1.0f,0.0f);
glRotatef(rot++,0.0f,0.0f,1.0f);
```

Início do desenho do cubo:

`glBegin()` inicia a descrição de um objeto. O parâmetro indica o tipo dele. Para fazer um tetraedro usaríamos `GL_TRIANGLES`. O `glVertex3f()` define um vértice no espaço. Para `GL_QUADS`, precisamos ter entre o `glBegin()` e o `glEnd()` pelo menos 4 vértices (um quadrado). Para `GL_TRIANGLES`, precisamos ter entre o `glBegin()` e o `glEnd()` pelo menos 3 vértices (um triângulo).

O `glColor3f()` indica a cor dos próximos vértices. Enquanto não for alterado o estado do `glColor3f()` os objetos desenhados terão todos a mesma cor.

```
glBegin(GL_QUADS);
```

Agora vamos iniciar o desenho da face frontal do cubo, determinando a sua cor e a posição de cada um dos vértices.

continua...

Vértice do topo do quadrado do topo em verde.

```
glColor3f(0.0f, 1.0f, 0.0f);
glVertex3f( 1.0f, 1.0f,-1.0f);
```

Vértice da esquerda do quadrado do topo.

```
glVertex3f(-1.0f, 1.0f,-1.0f);
```

Vértice da direita do quadrado do topo.

```
glVertex3f(-1.0f, 1.0f, 1.0f);
```

Vértice de baixo do quadrado do topo.

```
glVertex3f( 1.0f, 1.0f, 1.0f);
```

As outras faces foram feitas de forma análoga.

```
// Parte de baixo do cubo em laranja
glColor3f(1.0f,0.5f,0.0f);
glVertex3f( 1.0f,-1.0f, 1.0f);
glVertex3f(-1.0f,-1.0f, 1.0f);
glVertex3f(-1.0f,-1.0f,-1.0f);
glVertex3f( 1.0f,-1.0f,-1.0f);

// Parte de frente do cubo em vermelho
glColor3f(1.0f,0.0f,0.0f);
glVertex3f( 1.0f, 1.0f, 1.0f);
glVertex3f(-1.0f, 1.0f, 1.0f);
glVertex3f(-1.0f,-1.0f, 1.0f);
glVertex3f( 1.0f,-1.0f, 1.0f);

// Parte de trás do cubo em amarelo
glColor3f(1.0f,1.0f,0.0f);
glVertex3f( 1.0f,-1.0f,-1.0f);
glVertex3f(-1.0f,-1.0f,-1.0f);
glVertex3f(-1.0f, 1.0f,-1.0f);
glVertex3f( 1.0f, 1.0f,-1.0f);

// Parte da Esqueda do cubo em Azul
glColor3f(0.0f,0.0f,1.0f);
glVertex3f(-1.0f, 1.0f, 1.0f);
glVertex3f(-1.0f, 1.0f,-1.0f);
glVertex3f(-1.0f,-1.0f,-1.0f);
glVertex3f(-1.0f,-1.0f, 1.0f);

   glEnd();
}
```

Com o cubo desenhado, podemos nos preocupar agora com a execução do programa e ela começa na função `main()`.

```
int main(int argc, char **argv) {
```

Põe no estado GLUT, habilita a biblioteca do OPenGl.

```
glutInit(&argc, argv);
```

Programação 3D com OpenGL

Cria uma janela e o parâmetro entre aspas descreve o título, caso esta janela não possa ser criada o programa é encerrado com a chamada exit().

```
if (glutCreateWindow\
      ("Cubo 3D em OPENGL") <= 0) {
            exit(0);
}
```

Passa para o GLUT qual a função que desenha a cena.

```
glutDisplayFunc(&DesenhaCena);
```

Avisa o GLUT que função chamar quando o programa estiver ocioso.

```
glutIdleFunc(&DesenhaCena);
```

Passa para o GLUT qual função trata eventos do teclado.

```
glutKeyboardFunc(&Teclas);
```

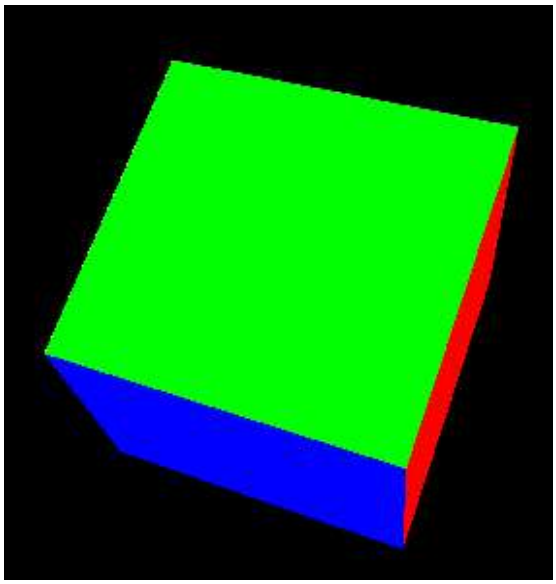
Define o tipo de projeção(perspectiva) e o tamanho da janela(300x300 pixels)

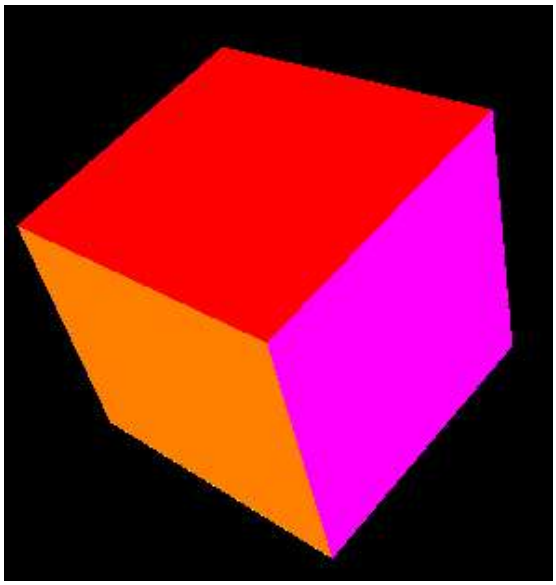
```
glMatrixMode(GL_PROJECTION);
gluPerspective(45.0f,(GLfloat)300/
(GLfloat)300,0.1f,100.0f);
glMatrixMode(GL_MODELVIEW);
```

Inicia Loop principal do OpenGL.

```
glutMainLoop();
return 1;
}
```

E pronto! Temos um maravilhoso cubo giratório. E isso criado praticamente a partir do “nada”.





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

Automatizando a instalação do **slackware linux**

Automatizar o processo de instalação de qualquer sistema linux hoje em dia para um administrador de uma empresa significa ganho de tempo e dinheiro. Às vezes estamos tão atarefados que nem tempo para nos alimentar direito temos. Pensando neste aspecto resolvi escrever este artigo para ajudar aos vários administradores (ou mesmo usuários iniciantes) a tornar a instalação do **slackware 10.1** em várias máquinas mais rápida, automatizando boa parte do processo após haver configurado uma máquima "modelo".

Este processo pode ser feito graças aos `tagfiles` do **slackware** que são arquivos onde você diz qual pacote deve ou não ser adicionado ao seu sistema durante a instalação. Estes arquivos são encontrados dentro de cada uma das séries nos CDs e mirrors do **slackware** com o nome `tagfile`.

Veja um fragmento de um desses arquivos localizado em `/mnt/cdrom/slackware/n/tagfile` (ou seja, é o arquivo responsável pela instalação da série `N`):

```
apache: OPT
autofs: OPT
bind: REC
bitchx: OPT
...
htdig: OPT
imapd: OPT
inetd: REC
iproute2: OPT
iptables: ADD
```

Observando o arquivo veremos que ele possui um formato:

```
nome do pacote : status
```

Onde `status` possui 4 opções:

- `ADD`: Pacote requerido para o bom funcionamento requerido, nem sempre é valido
- `SKP`: Pacote que não será instalado
- `REC`: Pacote não requerido, mas recomendado
- `OPT`: Também não requerido, é opcional

Conhecendo estes arquivos e suas funções podemos dar início a customização da instalação. Parto do pressuposto que você já passou pelas primeiras etapas de instalação do **slackware** e chegou até o ponto da seleção de quais das séries (`A`, `AP`, `K`, `L`, etc.), marque quais delas serão necessárias a sua instalação e prossiga. No perfil de instalação (`full`, `expert`, etc...) selecione o modo `expert`, a personalização dos tagfiles iniciará daqui em diante. Marque todos os pacotes a serem instalados.

No final desta etapa o sistema dará início a instalação do **slackware**, neste momento os `tagfiles` já foram gerados no diretório `/var/log/setup/tmp/tagfiles`. Pressione as teclas `CTRL+ALT+F2` para abrir um console, monte sua unidade de diskete (ou mesmo um HD) e copie este diretório para o diskete/HD montado:

```
# mkdir /tags
# mount -t tipo_de_partição /dev/fd0 /tags
# cp -r /var/log/setup/tagfiles /tags
# umount /tags
```

> **Obs:** Para aqueles que desejam o conteúdo do segundo CD, aguarde a primeira instalação até que seja pedido o CD2 do **slackware**. Pois apenas nesse momento os `tagfiles` das séries que estão neste CD são gerados.

Seus `tagfiles` foram salvos. Se quiser continuar com a instalação nesta máquina, aguarde. Se não, reinicie o sistema e quando for pedido o perfil de instalação selecione `tagfiles`, neste ponto abra um console e monte seu diskete/HD:

```
# mkdir /floppy
# mount /dev/fd0 /floppy
```

Volte para a tela de instação pressionando as teclas `CTRL+ALT+F1`, indique o caminho completo de onde se encontra os `tagfiles`, exemplo:

```
# /mnt/floppy/tagfiles
```

Pressione ENTER e pronto, aguarde o final da instalação :)

Fábio Becamp <fabiobecamp@slacklife.com.br>

Mudando para um HD maior

Ok, o tempo passou e o seu HD velho de guerra de 2GB não serve mais para nada. Acabou de comprar um de 80GB, mas... e o seu sistema? Deu tanto trabalho para arrumar, deixar redondinho... será que vai precisar instalar tudo de novo? Não, existe uma maneira de transferir o sistema de um HD para o outro, e uma maneira bem segura, basta seguir os passos deste artigo.

Coloque o HD novo na sua máquina. Pode colocá-lo como primário ou secundário, tanto faz. No exemplo estamos utilizando o HD com o sistema original como sendo o `hda` e o HD para o qual vamos transferir os sistema é o `hdc`. Boote o seu sistema no runlevel 1. Para isso, basta acrescentar "`1`" no final do prompt do `LILO`:

```
LILO: Linux 1
```

Abra o `/etc/fstab` e veja quais particões são utilizadas atualmente. É razoavelmente mais tranqüilo fazer uma migração mantendo a mesma estrutura de partições, embora isso não seja obrigatório. Por exemplo, observando as partições locais temos:

```
root@(none):~# grep "^/dev" /etc/fstab
/dev/hda1   swap   swap  defaults   0 0
/dev/hda3   /      ext3  defaults   1 1
/dev/hda7   /usr   ext3  defaults   1 2
/dev/hda5   /opt   ext3  defaults   1 2
/dev/hda6   /home  ext3  defaults   1 2
/dev/cdrom  /mnt/cdrom  iso9660 \
                noauto,users,ro  0 0
/dev/fd0    /mnt/floppy vfat \
                noauto,users     0 0
```

Com essas informações em mãos, é hora criar essas partições no /dev/hdc usamos para isso o fdisk.

```
root@(none):~# fdisk /dev/hdc
```

A primeira coisa a fazer no HD é apagar as partições existentes. Em seguida, vamos criar as partições que usamos no `fstab`. Se você pretende alterar a sua estrutura de partições, essa é a hora. Crie as partições de acordo com o seu novo plano. No exemplo, resolvemos manter a mesma estrutura, para não precisar editar o `fstab` depois.

Com esse objetivo, criamos um `hdc1` para `swap` (partição do tipo `82`) , uma partição `hdc2` extendida contendo o `/usr` o `/opt` e o `/home` e uma partição `hdc3` para ser a nossa partição `/`. Como o novo HD é muito maior e pretendemos utilizar como repositorio de dados, foi criada uma `hdc4` que sera montada sob o `/data`. Com todas as partições criadas, deve-se agora formatá-las. Novamente, se pretende mudar o sistema de arquivos que utiliza, essa é a hora. No exemplo, iremos manter ainda a `ext3`:

```
root@(none):~# for i in \
                  `fdisk -l /dev/hdc | \
                  grep "^/dev.*Linux$" | \
                  cut -f1 -d\ `; do
                        mke2fs -j $i
               done
```

Esse comando irá formatar todas as partições do tipo Linux como `ext3`. Se quisesse formatar com ReiserFS, trocaria onde está `mke2fs -j` por `mkreiserfs`. Fica faltando apenas formatar a partição `swap`:

```
root@(none):~# mkswap -c /dev/hdc1
```

Estando tudo OK, monte a nova partição `/` (`/dev/hdc3`) no `/mnt/hd` e copie para lá os diretórios presentes na raiz do sistema, em seguida faça o mesmo com a estrutura do `/mnt`:

```
root@(none):~# mount /dev/hdc3 /mnt/hd
root@(none):~# (cd /mnt/hd ; mkdir `ls /`)
root@(none):~# (cd /mnt/hd/mnt ; \
                        mkdir `ls /mnt`)
```

Se uma das suas partições estiver "dentro" da árvore, por exemplo: `/usr/local` que estah dentro do `/usr`, você deve criar todo o caminho necessário para alcancá-la. No caso do `/usr/local`, como já criamos o `usr`, deveria apenas ser criado o `local` (`mkdir /mnt/hd/usr/local`), se fosse o `/var/spool` (`mkdir /mnt/hd/var/spool`) e por aí vai. No caso do exemplo, precisamos criar um `/data`, que apesar de não estar no "interior" da árvore, não existe no `/` original.

continua...

Com tudo isso pronto, monte todas as partições que serão utilizadas:

```
root@(none):/mnt/hd# mount /dev/hdc7 \
                        /mnt/hd/usr
root@(none):/mnt/hd# mount /dev/hdc5 \
                        /mnt/hd/opt
root@(none):/mnt/hd# mount /dev/hdc6 \
                        /mnt/hd/home
```

E agora é a hora de iniciar a cópia, lembre-se de não copiar nem o conteúdo do `/proc` nem o do `/mnt`. O primeiro é um sistema de arquivos "virtual" e com vários arquivos bem grandinhos se forem copiados (por exemplo, o `/proc/kcore`) e o segundo é onde está o nosso HD, o que seria um caso horrível de cópia cíclica.

```
root@(none):/mnt/hd# cd / && \
      cp -a  `/bin/ls -1Ab | \
            grep -v "^mnt\|^proc"` /mnt/hd
```

Depois de digitar esse comando, será necessário esperar uma quantidade razoável de tempo, enquanto todos os seus dados são copiados de um HD para outro. Ao final da espera, deve-se editar o `/etc/lilo.conf` para que seja possível dar o `boot` pelo novo HD.

A alteração é bem simples, embora estranha, devemos dizer ao LILO que o que está como `hdc` na realidade seria o `hda` (`0x80` na BIOS). No início do arquivo inclua:

```
disk=/dev/hdc bios=0x80
map=/mnt/hd/boot/System.map
```

E troque onde está:

```
boot = /dev/hda
```

Para

```
boot = /dev/hdc
```

Lembre-se, no nosso EXEMPLO o HD novo é o `/dev/hdc`. Se no seu caso for outro HD, coloque o dispositivo correto, sob o risco de ficar com um computador que não liga. É bem fácil restaurar esse tipo de problema com um disco de `boot` ou com o CD do **`slackware`**, mas o melhor a fazer é sempre evitar os problemas.

A título de curiosidade, basta bootar com o CD do **`slackware`** e, seguir as instruções indicadas na tela de instrução sobre como montar um sistema já existente. Depois de montado o sistema apenas execute o `lilo`.

Aproveite e, lembrando que devemos usar os arquivos que estão no novo HD, troque também a imagem do kernel a ser lida, de:

```
image = /boot/vmlinuz
```

Para:

```
image = /mnt/hd/boot/vmlinuz
```

E é só isso. Se a sua imagem não se chama `vmlinuz`, faça as alterações necessárias e, em seguida, execute o LILO, deve aparecer algo como:

```
root@(none):~# lilo
Added Linux *
Added New
```

Agora é só desligar o computador, trocar os dois HDs e ir para o abraço!

Ah! Caso você tenha alterado a estrutura das suas partições (por exemplo, antes era tudo em uma partição e agora foi organizado em várias) deve-se editar o `/etc/fstab` antes de trocar o HD e rebootar a máquina. Caso contrário é possível que o sistema não chegue a bootar.

Assim sendo, supondo que o seu `fstab` antes fosse assim:

```
/dev/hda1   /     ext3  defaults  1 1
```

E que agora além do `/` você criou um `/home` no `hdc4`, deve colocar:

```
/dev/hda4   /home ext3  defaults  1 2
```

No final do arquivo. Preste bastante atenção, colocamos `/dev/hda4` ao invés de `hdc4`, porque quando forem trocados os HDs, o `hda` será o nosso atual `hdc`.

Não será necessário "consertar" o `lilo.conf`, pois quando fomos editá-lo, editamos o que está no nosso, agora obsoleto, HD de 2GB, e a cópia que se encontrava no `hdc` (agora `hda`) continua intacta.

E agora está realmente terminado. Se, mesmo assim você estiver precisando de maiores informações, no `/usr/doc/Linux-HOWTOs` existe um `Hard-Disk-Upgrade` de onde foram retirados alguns truques usados neste artigo, é uma boa leitura. Boa Sorte com a sua migração! Ah! Se o HD de 2GB ainda estiver OK, mande para mim -;) Os meus computadores e eu vamos gostar bastante...

Piter PUNK <piterpk@terra.com.br>

Clientes de mensagens instantâneas

Programas de comunicação, hoje em dia é quase impossível ficar sem eles :D. E no mundo do software livre existem zilhões de programas para facilitar essa tarefa, neste artigo iremos analizar alguns desses programas de comunicação, mais especificamente, programas de mensagens instantâneas

O grande exemplo quando se fala de mensagens instantâneas é o ICQ. ICQ é um serviço de mensagem instantâneas muito bom (na minha opinião o 2 melhor, 1. é irc) e um pouco antigo, foi criado oficialmente em 1996 e mais de 150 Milhões de usuários registrados. E, você não vai ficar de for a dessa onda, já que existem diversos programas que conseguem conectar na rede do ICQ, como por exemplo:

mICQ

Na minha opinião é o melhor cliente de ICQ que existe. É leve, funciona só em modo texto, e diferente do que muitas pessoas imaginam o projeto não morreu :D

O que é favorável nele é a praticidade, leveza (pois não exige muito processamento) e as cores que não são cansativas e nem muito chamativas... mas quem não gosta de cor.. pode desabilitar :)

Excelente para quem tem uma maquina não tão nova e nem tão rapida (e para as novas e rápidas também) Só acho complicado até hoje receber arquivos por ele, nunca consegui (e também nunca precisei :D)

licq

O `licq` é um dos clientes de icq mais conhecidos hoje, é muito parecido com o `icq2000b` (pra M$ Windows). Feito em C++ e Qt tem uma "carinha bonita" e também funciona em modo texto.

Tem skins bem legais é bem facinho de usar, pega toda sua lista de contatos do servidor da Mirabilis mas é pesado e consome muito processamento.

GAIM

O GAIM é bem conhecido por ser Multi-Protocolo, suporta ICQ, MSN Messenger, Yahoo Messenger, IRC, Jabber e mais algumas coisinhas, ele é leve e facinho de usar também, você consegue conectar todos os seus protocolos ao mesmo tempo. Por exemplo: você usar ICQ e MSN Messenger, você não precisa conectar um ter que fechar para conectar o outro, você consegue conectar os 2 no mesmo gaim.

Tive alguns problema com o gaim, ele de vez em quanddo dáum segmentation fault do nada, mas não é um problema comum, possui muitos usuários e esses dizem funciona muito bem. Dos clientes listados, é o único que vem no próprio **slackware**.

centerICQ

Outro cliente que multi-protocolo, nao sei se é tão conhecido quanto o GAIM e o `licq`, mas conheço algumas pessoas que usam, funciona em modo texto, **EU** não gostei. Muito pesado e um pouco confuso de mexer. Gostei muito do Jabber em modo texto :) funciona legal, existem alguns apaixonados por `centericq` eu gostei mesmo só do Jabber.

Tiago Machado <garoto@uol.com.br>

slackwarezine precisa de você!

Sim, é exatamente o que você leu ali em cima. Nós precisamos, e muito, de você. Não só para ler o zine, ou enviar sugestões, mas precisamos dos seus artigos. Todos conhecem a nossa proposta: uma revista escrita por técnicos e para técnicos.

Aqui não tem espaço para politicagens e filosofices, afinal, mais importante que falar é colocar mão na massa, é configurar, é desenvolver software, é trocar informações para tornar o nosso trabalho e o trabalho dos outros mais fácil.

Por tudo isso o seu artigo é importante. Não precisa descobrir a pólvora, ou escrever um tratado de mecânica celeste; a nossa proposta inclui artigos para iniciantes e para usuários experientes. A idéia não é agradar a todos, mas pelo menos conseguir com que todos encontrem algo agradável -;).

Muita gente tem nos enviado artigos, nos mais diversos formatos e assuntos. E, muitas vezes, pelo formato com que esses artigos chegam, não são aproveitados. Por isso, quando mandar o seu artigo siga essas diretrizes:

- Mande o artigo em .txt, sem formatação.
- Os parágrafos não devem ter quebras de linha e devem ser separados por uma linha em branco
- Linhas de comando e código, preferencialmente quebradas com 40 caracteres. Como as vezes é impossível e ilegível um texto assim, use o bom senso
- Figuras em anexo
- Os artigos devem ser enviados até último dia do mês anterior. Se você quiser publicar na edição de maio, deve enviar o artigo até o último dia útil de abril
- Já mande com o artigo o seu mini-currículo
- Envie o e-mail para editor@slackwarezine.com.br e não diretamente para um dos editores.

Essas regras facilitam o seu trabalho e o nosso, aumentando a qualidade da publicação como um todo, mais tempo para diagramar e tempo para que os autores possam ver como cada artigo será publicado, diminuindo a ocorrência de erros.

Muito Obrigado a todos os nossos leitores!

Editores

Autores

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

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

Fábio Becamp, começou a usar unix desde 2000, inicieu com um freebsd 3.x depois no mesmo ano mudou pro Slackware. Atualmente faz parte do Slacklife, outro grupo de slackers brasileiro, cursa Ciências da Computação na UNAMA e trabalha como administrador de redes na VirtualLink Consultoria

Kaio Rafael de Souza Barbosa, 22 anos, é System Administrador, Trabalha em Manaus em uma Multinacional somente com Solaris e Linux. Começou a utilizar o GNU em 1998 e logo depois se apaixonou pelo Slackware em 1999 e usa atualmente em seu notebook diariamente.

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

Tiago Machado a.k.a. gar0t0, estudante de Ciência da Computação. Usuário Slackware desde junho de 2003 quando realmente tomou coragem e apagou o velho windows da máquina.

SSH sem senha

Apesar de comentado em várias listas de discussões, ainda assim, algumas pessoas têm dúvidas em relação de como conectar através do ssh sem senha, utilizando apenas as chaves pública e privada.

Um processo muito útil para a administração remota, podendo utilizar o ssh em scripts e no sincronismo via rsync, para ficar apenas em dois exemplos práticos. Este artigo ensina como gerar e configurar devidamente estas chaves para tal tipo de autenticação.

1 – Gere chave pública do tipo RSA em A

```
kaio@A~:$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key \
      (/home/kaio/.ssh/id_rsa): ENTER
Enter passphrase \
      (empty for no passphrase): ENTER
Enter same passphrase again: ENTER
Your identification has been saved \
      in /home/kaio/.ssh/id_rsa.
Your public key has been saved \
      in /home/kaio/.ssh/id_rsa.pub.
The key fingerprint is:
de:f9:be:1f:30:65:86:5b:12:1b:ff:59:53:9e:ca:a1
```

2 – Copie a chave pública de A para B.

```
kaio@A~:$ scp .ssh/id_rsa.pub \
                  kaio@B:~/chavepuba
```

3 – Logue em B e crie o arquivo responsável por autorizar suas chaves. Muita atenção para o >>, com isso a chave de A será adicionada no final do arquivo, não apagando as chaves já existentes.

```
kaio@B~:$ cat chavepuba >> \
                 .ssh/authorized_keys
```

Com todos os passos completos, resta a apenas o teste do host A para ver se tudo está funcionando.

```
kaio@A~:$ ssh B
```

Um outro interessante exemplo desta aplicação é quando você precisa que outros usem a sua conta sem a necessidade fornecer a eles a sua preciosa senha. Se você pretende controlar diversos servidores com esse método, lembre-se de proteger bem a máquina A, pois se alguém tiver acesso a ela, terá acesso a todas as outras máquinas...

Kaio Rafael <kaiorafael@click21.com.br>