

# **ADMINISTRAÇÃO DE SISTEMAS LINUX**

Linux – Darlan Segalin - [darlanse@gmail.com](mailto:darlanse@gmail.com)

**UNIVERSIDADE DO OESTE DE SANTA CATARINA - UNOESC**

**CURSO: ADMINISTRAÇÃO DE SISTEMAS LINUX**

**INSTRUTOR: DARLAN SEGALIN**

[darlan.segalin@unoescxxe.edu.br](mailto:darlan.segalin@unoescxxe.edu.br)

[darlan@cbainfo.com.br](mailto:darlan@cbainfo.com.br)

[darlanse@gmail.com](mailto:darlanse@gmail.com)



## Conteúdo do Curso:

<b>-Objetivos</b>	
<b>-Módulo Básico</b> .....	8
Man	
Logout	
Shutdown -r now	
Su	
Ls (-la)	
Cd (deslocamento relativo x absoluto)	
Cp	
Mv	
Alias	
Date	
Clear	
Mkdir (-p)	
Rmdir (-p)	
Rm (-rf)	
Who (w, who is god, whoami)	
Df	
Free	
Cat /proc/cpuinfo	
Setterm	
Tput	
Uptime	
Arquivos da Inicialização	
Variáveis	
Variável PS1	
Arquivo /etc/motd	
Arquivo /etc/issue e /etc/issue.net	
Ps (aux)	
Kill	
Killall	
Sinais (1, 9, 15)	
Touch	
Find	
Locate	
Top	
Vi (:w :x :q :q! /)	
Background (&)	
Jobs	
Fg	
Ln (-s)	
Chmod (ugo +rwx) ( Entendendo as permissões básica no linux )	
Umask	
Chown (user:group)	
Chgrp	
Caracteres Especiais (~, [], *, ?)	
Home do Usuário	
Adicionando um Grupo	

Adicionando um Usuário  
Senhas SHADOW  
Arquivo /etc/passwd  
Arquivo /etc/group  
Porque saber editar manualmente usuários e grupos?

**Programação Shell Script .....19**

Características da Shell BASH  
#!/bin/bash  
If [ condicao ] then fi  
For variavel in lista do done  
Aprofundando em Shell Script  
Script Adiciona 1000 usuários  
Script Abrir Interface que Escolher  
Case  
Reescrevendo o script de abrir interface usando o case  
While  
O comando CUT  
Escrevendo um script para acertar as permissões do HOME dos usuários  
Para aprofundar: Script LogRotate.sh

**Utilizando o VI ..... 29**

Editando um texto  
Indo para a última linha  
Indo para a primeira linha  
Indo para a enésima linha  
Removendo uma linha  
Removendo a partir do cursor  
Colocando número nas linhas  
Substituindo linhas  
Salvando  
Saindo e salvando  
Saindo sem salvar  
Salvando para outro arquivo  
Saindo e salvando em outro arquivo  
Forçar salvação  
Inserir linha abaixo do cursor  
Inserir linha acima do cursor  
Copiar e colar linhas

**Entendendo o sistema de arquivos do Linux ..... 29**

Estrutura de Diretórios  
MOUNT  
FSTAB  
FDISK  
FSCK  
FDFORMAT  
MKFS  
MKSWAP

<b>Configuração e Instalação do Kernel</b> .....	33
Entendendo o conceito de módulos	
Gerenciando Módulos	
Adicionando Suporte a um Novo Hardware	
Personalizando o kernel	
<b>Ferramentas Básicas de Rede</b> .....	35
Ifconfig	
Route	
Netstat	
Ping	
Traceroute	
Nslookup	
Editando os arquivos de configuração manualmente	
<b>Compactadores e Empacotadores de Arquivos</b> .....	37
Gzip	
Zip	
Bzip	
Compress	
Tar	
<b>Agendamento de Tarefas</b> .....	39
Cron	
At	
<b>Entendendo o Super Daemon Inetd</b> .....	40
/etc/inetd.conf	
/etc/hosts.allow	
/etc/hosts.deny	
<b>Configurando o boot da máquina: Grub e LILO</b> .....	41
<b>Instalação e compilação de aplicativos</b> .....	45
O gcc	
O comando ./configure	
Módulos do perl	
O comando make	
O comando make install	
Quando ocorrem os problemas...	
<b>Gerenciamento de pacotes RPM</b> .....	47
Instalando	
Removendo	
Atualizando	
Pesquisando	
Erros Comuns	
<b>Gerenciamento de pacotes APT-GET</b> .....	48

<b>Configurando o NFS</b> .....	51
Escolhendo os diretórios a compartilhar	
Atualizando tabela de arquivos compartilhados	
Portmap	
Acionando o Portmap	
Acionando o NFS	
Verificando arquivos compartilhados pela máquina	
Verificando clientes que estão acessando os arquivos compartilhados	
<b>SAMBA</b> .....	53
Entendendo o Funcionamento do NetBIOS	
Entendendo o Funcionamento do Samba	
Configurando um servidor SAMBA	
Compartilhando Diretórios através do SAMBA	
<b>Configurando o SSHD</b> .....	55
Retirando a compatibilidade com a Versão 1	
Retirando o uso da diretiva UseLogin	
Aumentando a chave criptográfica	
Permitindo ao root se logar remotamente	
<b>Configurando o ProFTPd</b> .....	56
Habilitando usuário anonymous	
Mudando informações do servidor	
Diretivas de configuração	
<b>Configurando o Apache</b> .....	57
Sobre	
Porque utiliza-lo	
Exemplo de configuração	
<b>Segurança</b> .....	58
Quesitos para um sistema seguro	
<b>Montando um firewall linux</b>	
Entendendo o iptables	
Regras de entrada, saída e passagem de pacotes	
Tabelas do iptables	
Proibindo protocolos	
Proibindo portas e tipos de mensagens	
Proibindo flags TCP	
Proibindo estados de pacotes	
Limitando as conexões	

## OBJETIVOS

Os objetivos deste treinamento não são os de tornar seus adeptos especialistas avançados no sistema operacional, e sim usuários e administradores avançados.

Isto quer dizer que eles dominaram algumas características do sistema, mas devem estudar e se aprofundar muito para se tornarem especialistas e profissionais no mesmo.

Espero converter os usuários que leiam e façam o treinamento e mostrar-lhes quão fácil é entrar no mundo LINUX.

Percebam que a apostila é técnica, não sendo altamente instrutiva isoladamente, por isso, façam os exemplos desta e vejam os manuais para maiores entendimentos. Ela é apenas um complemento do treinamento completo, mas pode facilmente se tornar um curso através de um documento para as pessoas mais esforçadas e interessadas.

Não ficarei preso a uma única distribuição, mas gostaria de lembrar que pessoalmente eu utilizo o Ubuntu, Debian, CentOS, Suse, RedHAT, todos os exemplos mostrados serão em Mandriva Conectiva Linux (no qual sou certificado) e Debian Linux.

Também não mencionarei o uso de Configuradores Gráficos (LinuxConf e WebMin).

Os exemplos aqui apresentados são EXEMPLOS, embora as configurações funcionem podem não ser as ideais para um servidor e devem ser revistas e acrescentadas.

Esta apostila PODE e DEVE ser distribuída livremente, desde que não seja alterada.

## O Linux

O Linux é um sistema operacional criado em 1991 por *Linus Torvalds* na universidade de Helsinki na Finlândia. É um sistema Operacional de código aberto distribuído gratuitamente pela Internet. Seu código fonte é liberado como *Free Software* (software livre) o aviso de copyright do kernel feito por Linus descreve detalhadamente isto e mesmo ele não pode fechar o sistema para que seja usado apenas comercialmente. O sistema segue o padrão *POSIX* que é o mesmo usado por sistemas *UNIX* e suas variantes. Assim, aprendendo o Linux você não encontrará muita dificuldade em operar um sistema do tipo *UNIX*, *FreeBSD*, *HPUX*, *SunOS*, etc., bastando apenas aprender alguns detalhes encontrados em cada sistema. O LINUX NÃO É VULNERÁVEL A VÍRUS! Devido a separação de privilégios entre processos e respeitadas as recomendações padrão de política de segurança e uso de contas privilegiadas (como a de root, como veremos adiante), programas como vírus tornam-se inúteis pois tem sua ação limitada pelas restrições de acesso do sistema de arquivos e execução. Rede TCP/IP mais rápida que no Windows e tem sua pilha constantemente melhorada. O GNU/Linux tem suporte nativo a redes TCP/IP e não depende de uma camada intermediária como o WinSock. Em acessos via modem a Internet, a velocidade de transmissão é 10% maior. Só o kernel GNU/Linux não é suficiente para se ter um sistema funcional, mas é o principal. Existem grupos de pessoas, empresas e organizações que decidem "distribuir" o Linux junto com outros programas essenciais (como por exemplo editores gráficos, planilhas, bancos de dados, ambientes de programação, formatação de documentos, firewalls, etc). Algumas distribuições bastante conhecidas são: *Slackware*, *Debian*, *Red Hat*, *Conectiva*, *Suse*, *Monkey*, todas usando o SO Linux como kernel principal (a Debian é uma distribuição independente de kernel e pode ser executada sob outros kernels, como o GNU hurd).

## Módulo Básico

### - man

Comando que nos mostra o manual de um outro comando.

Uso: man [seção] comando

Exemplo de uso:

```
man 1 ls
```

Utilize /palavra para procurar e q para sair.

### - logout / exit

Fecha a shell do usuário. Este comando é utilizado quando se termina sua sessão ou para se trocar de usuário.

### - shutdown -r now

Comando utilizado para reiniciar a máquina. Possui alguns similares:

```
reboot
```

```
init 6
```

```
Ctrl + Alt + Del
```

Existe também o shutdown -h now que desliga a máquina. Seus similares:

```
poweroff
```

```
halt
```

```
Init 0
```

### - Su

Utilizado para se trocar de usuário sem efetuar logout. Muito comum em acessos via rede, já que via rede por default o root não pode se logar.

Usa-se também su -c "comando a executar" para se executar um comando com poderes de root e depois retornar. Obviamente será pedida uma senha.

Uso do su:

```
su darlan
```

### - Ls

Comando que serve para listar arquivos. Suas opções mais utilizadas são **-lFha**, onde o **-l** significa para listar as permissões (inclusive), o **F** para classificar, o **h** para imprimir o tamanho em linguagem humana e o **a** para listar todos os arquivos (lembrando que para o linux arquivos começados com **.** são ocultos).

OBS: O Mandriva Linux possui um alias chamado **l** para o comando **ls -lFha**, use-o e caso a sua distribuição não contenha tal alias, crie-o. Veja mais adiante como fazê-lo.

### - Cd (deslocamento relativo x absoluto)

Comando para mudar-se de diretório. O deslocamento absoluto se tem quando utilizamos a raiz (/) para indicarmos para onde queremos ir. Por exemplo, imaginemos que estamos no diretório **/usr/src/unoesc** e desejamos ir para o diretório **/usr/src/darlan**. Temos duas opções, a seguir:

**cd /usr/src/darlan** □ Deslocamento absoluto, observe o uso do / no início

do diretório para o qual queremos ir

**cd ../darlan** □ Deslocamento relativo, perceba que se estivéssemos em um outro diretório (/usr) por exemplo, não iríamos cair onde queremos. Daí a convenção de "relativo".

### - Cp

Copia arquivos. Use: **cp arquivoASerCopiado novoArquivo**

**Ex: cp -rp /teste/darlan /tmp**

opcao -p Copia todo diretorio e arquivos de /teste/darlan para o diretorio /tmp/ com a para manter as permissoes originais dos arquivos

Opções interessantes:

-i □ Pedir confirmação antes de substituir um arquivo existente

-R □ Cópia recursiva. Serve para copiar diretórios e seu conteúdo.

### - Mv

Mover arquivos. Use-o também para renomear.

Uso: **mv arquivo novaLocalizacao/**

**mv arquivo novoNome**

Recomendado:

-i □ Confirma antes de substituir um arquivo existente.

OBS: No Mandriva Linux existe um alias tanto para o comando cp como para o mv com a opção -i.

### - Alias

Cria um apelido para um comando. Tem precedência sobre o comando, ou seja, pode-se criar um alias do tipo: **alias l="ls -lFha"**. Toda vez que digitarmos l na verdade ele executará ls -lFha. Para gravar o alias no sistema linux colocar o comando na última linha do arquivo /etc/bashrc ou /etc/bash.bashrc dependendo da sua distribuição.

### - date

Ajustar a hora do sistema:

```
# date 051909522004
```

Onde o 05 corresponde ao mês, o 19 ao dia, o 09 as horas, o 52 aos minutos, o 20 às duas primeiras casas do ano e o 04 às duas segundas casas do ano, logo estou selecionando para o meu sistema a data 19/05/2004 e a hora 09h52min.

### - clear

Limpa a tela. Recomenda-se a criação de um alias chamado c para este comando.

### - mkdir (-p)

Comando para a criação de diretórios. Usa-se o -p caso se queira criar uma "árvore" de diretórios.

- **rmdir (-p) ou tambem pode se usar rm -rf /diretorio/**

Complemento do comando mkdir. Serve para remover um diretório vazio. A opção -p serve para remover uma árvore de diretórios vazia (sem arquivos).

- **Rm (-rf)**

Comando utilizado para apagar arquivos. Observe que o rm simplesmente não apaga diretórios. Sua opção -r indica para apagar recursivamente, ou seja, ir apagando todos os arquivos em subdiretórios e inclusive os próprios diretórios. A opção -f força apagar, e não emite mensagens de erro caso não exista um arquivo.

Ex: **rm -rf arquivoQueNaoExiste**

Não acontecerá NADA. Nenhuma mensagem de erro será informada.

- **Who (w, who is god, whoami)**

O comando **who** e **w** listam os usuários que estão logados na máquina. O **w** tem uma saída um pouco mais complexa, mostrando mais informações.

O comando **who is god** é uma sátira e retorna o nome de seu usuário.

O comando **whoami** (pode ser escrito **who am i**) também retorna o nome de seu usuário e é utilizado para saber com qual usuário você está logado, muito usado quando se utiliza o **su** e acaba se confundindo quem é você.

- **df -h**

Mostra informações de sistemas de arquivos montados (mesmo CDRom e Disquete).

- **Free**

Mostra informações de memória (swap inclusive).

- **Cat /proc/cpuinfo**

Informações muito completas de seu processador.

- **Setterm**

Este comando serve para modificar configurações do terminal do Linux, tais como cor de fundo e cor da letra.

Ex:

**setterm -background green** --> Fundo Verde

**setterm -foreground yellow** --> Letra "amarela". O

amarela está entre aspas devido ao fato de que a cor não parece ser amarelo não.

OBS: Este comando mudará a cor a partir do momento em que ele for dado, ou seja, você precisa imprimir algo na tela ou dar um clear para realmente mudar a cor.

- **Tput**

Utilizaremos este comando para posicionar o cursor na tela, onde quisermos. Ele será muito útil quando estivermos construindo shell scripts.

Ex:

**tput cup 5 10** □ Posiciona o cursor na linha 5 coluna 10.

## - Uptime

Mostra a quanto tempo o sistema está ligado. Os maiores uptimes da internet são com máquinas UNIX.

## - Arquivos da Inicialização

Alguns arquivos são executados quando o sistema reinicializa. O que nos será conveniente falar por agora será o arquivo /etc/rc.d/rc.local (todas as distribuições devem tê-lo implementado).

Este arquivo será o último a ser executado quando da inicialização do sistema.

Diversos arquivos são executados no processo de entrada de um usuário no sistema.

São eles:

- .bashrc
- .profile
- .bash\_login

Os 3 se localizam no HOME do usuário

E:

- /etc/bashrc
- /etc/profile

Observe que enquanto os 3 primeiros são exclusivos dos usuários (cada usuário pode ter suas configurações), os últimos 2 são globais a todos os usuários que entrarem no sistema.

Não recomenda-se alterar o arquivo /etc/profile já que este é de configurações e variáveis.

## - Variáveis

Variáveis nada mais são do que espaços na memória que armazenam valores. O linux possui variáveis do próprio sistema, que armazenam valores de configurações ou informações da máquina.

Para vê-las utilize o comando set.

Para darmos um valor a uma variável e torna-la global ao sistema, fazemos  
export variável=valor.

Para retirarmos uma variável fazemos unset variável.

## - Variável PS1

Esta variável guarda os valores para o PROMPT do Linux. Observe que estes valores podem ser variáveis interpretadas pela SHELL, e por default o são, ou seja, se você utilizar uma shell que não a shell BASH, eles podem ficar sem sentido.

A variável PS1 é uma variável normal do sistema, qualquer valor que for dado a ela irá ficar no prompt.

No entanto ela possui alguns valores especiais interessantes, eis alguns:

- \h      Host da máquina
- \W      Diretório Corrente
- \w      Caminha completo do diretório corrente

- \u     Nome do usuário
- \t     Hora
- \\\$    Fica \$ se for usuário normal e # se for root

Exemplo:

```
export PS1="[h@w]\\$ "
```

### - Arquivo /etc/motd

Este arquivo é lido pelo sistema quando um usuário loga na máquina e seu conteúdo é enviado para a tela do usuário, como uma mensagem de boas vindas ou algo do tipo. Preste atenção que comandos NÃO serão interpretados.

### - Arquivo /etc/issue

Tela vista ANTES do usuário se logar. Seria a própria mensagem antes do login. Observe que o arquivo /etc/issue.net é o mesmo que o issue mas é válido para conexões via rede (telnet). Em algumas distribuições este arquivo é apenas um link para o /etc/issue.

### - Ps (aux)

Comando que lista os processos em execução no sistema. Recomenda-se sempre utiliza-lo com as opções AUX, para que liste TODOS os processos ativos no sistema.

### - Kill

Serve para matar um processo em execução. Deve-se utilizar um dos sinais existentes para esta tarefa. O sinal padrão é o sinal 15. Após o sinal, deve-se informar o PID (identificador único de processos) do processo que se deseja matar (encerrar).

### - Killall

Implementação do linux muito interessante. Permite-se que se mate diversos processos com o mesmo nome de uma única vez. Observe que pode utilizá-lo para matar um único processo pelo nome, desde que se tenha o cuidado de perceber se não existem outros processos com este nome.

```
Ex: killall httpd  
Killall -9 vi
```

### - Sinais (1, 9, 15)

É importante se lembrar destes 3 sinais principais:

- 1-) SigHUP    Manda a aplicação reiniciar
- 9-) SigKILL    Manda o kernel tirar a aplicação da lista de processos ativos (mata mesmo!)
- 15-) SigTERM    Manda um sinal para que a aplicação termine

normalmente

Os sinais são utilizados para comunicações INTER-PROCESSOS, ou seja, quando um processo deseja indicar algo para outro processo.

Neste caso, o processo kill (killall é a mesma coisa), envia o sinal que pedimos para a aplicação.

Existem outros sinais (como SigINT) que são utilizados pelo sistema. Para visualizá-los utilize o comando kill -l.

## - Touch

Cria um arquivo texto vazio. Muito interessante na hora de se testar alguma coisa.

Uso: touch nomeDeArquivoaCriar nomeDeArquivoaCriar2 ...

Pode-se criar diversos arquivos de uma única vez.

## - Find

Busca arquivos. Muito avançado.

Uso: find DirAProcurar opções

Exemplos de uso:

find / -name Rodrigo.tar □ Procura a partir da raiz (no sistema todo) o arquivo chamado Rodrigo.tar

find /home -exec grep "teste" {} \; -exec ls -la {} \: □ Procura a partir do diretório /home arquivos com o conteúdo teste (grep teste) e lista este arquivo (ls -la).

find /usr -type l -ok rm -rf {} \; □ Procura no diretório /usr links (-type l) e caso encontre, confirma se deve ou não apagar (-ok rm -rf).

Consulte o manual para informações mais interessantes.

## - Locate

Busca arquivos, mas utiliza uma base de dados como padrão, o que o torna muito rápido. Cuidado!! Atualize sempre sua base de dados, ou irão aparecer arquivos que já foram removidos em suas buscas.

Outro problema do locate é o fato de que ele busca qualquer ocorrência da palavra a buscar, ou seja, se você fizer locate a, ele irá listar TUDO no sistema que contém a palavra a.

Para atualizar sua base de dados utilize: updatedb

Para buscar utilize: locate oqbuscar

## - Top

Método interessante de se visualizar os processos ativos na máquina.

Use: M --> Ordenar por consumo de memória

P --> Ordenar por consumo de CPU

### - Vi (:w :x :q :q! /)

Ótimo editor de textos que recomenda-se e muito saber. As opções vistas foram:

- Modo Comando, Fim de Linha e de Edição
- :w  Salva arquivo
- :x  Salva e sai
- :q  Sai quando você não alterou nada
- :q!  Sai sem salvar
- /palavra  Procura palavra
- n  Procura pela próxima ocorrência de palavra

### - Background (&)

O linux possui uma opção interessante que é a de mandar processos para o segundo plano, liberando assim o ambiente do usuário.

Pode-se fazer isso através do sinal & após qualquer comando.

### - Jobs

Lista os processos que estão em segundo plano, retornando o número do processo de segundo plano, que deverá ser utilizado para trazê-lo de volta.

### - Fg

Comando que trás de volta um processo do segundo plano.

Uso: fg numeroProcessodeSegundoPlanoRetornadoPeloJobs

### - ln (-s)

Este comando cria um link (atalho) entre diretórios e arquivos. Um link simbólico (opção -s) nada mais é do que um arquivo no HD que aponta para a área onde está o arquivo original. Se o original é apagado, o link fica "quebrado". Já um link direto (apenas ln) dá um outro nome para a mesma área do HD. Como um backup contra remoção indevida, no entanto usa-se o mesmo espaço do HD, referenciando-no de duas maneiras diferentes. Crie e compare. Um link direto não pode ser feito entre diretórios.

Uso: ln -s Original Link

ln Original Link

## - Entendendo as permissões no Linux ( CHMOD )

As permissões são usadas para definir quem pode acessar determinados arquivos ou diretórios, assim mantendo segurança e organização em seu sistema e sua rede.

Cada arquivo ou pasta tem 3 permissões.

(Usuário Dono) (Grupo Dono) (outros)

- Usuário dono: é o proprietário do arquivo;

- Grupo Dono: é um grupo, que pode conter vários usuários;
- Outros: se encaixam os outros usuários em geral.

Para ver a permissão de um arquivo digite no prompt:

## **\$ ls -lFha**

O comando "ls -lFha" faz uma listagem longa e detalhada no diretório atual.

As permissões vão aparecer assim:

- (r) Leitura
- (w) Escrita
- (x) Execução

Como as permissões são divididas em 3, irá aparecer assim:

(\_\_ \_\_) (\_\_ \_\_) (\_\_ \_\_), ou seja, (rwx)(rwx)(rwx)

Caso não haja todas as permissões, poderá aparecer incompleto:

rwxrw\_\_ \_\_x, ou seja, neste exemplo:

- Dono do arquivo tem permissão de Ler, Escrever e executar (rwx);
- Grupo tem permissão de Ler e Escrever (rw\_);
- Outros tem permissão apenas de executar. (\_\_ x);

Existem dois modos de definir uma permissão, através do modo Octal e modo Textual.

## **Textual**

Usamos letras antes das permissões (chmod é o comando para modificar permissões de arquivos):

**\$ chmod u+rw, g+w, o-rwx teste.txt**

Onde:

- U - representa usuário;
- G - representa grupo;
- O - representa outros.

Agora vejam o modo Octal.

## **Octal**

O modo Octal tem a mesma função de definir permissões, só que em números. Exemplo:

**\$ chmod 620 teste.txt**

(comando) (permissão) (arquivo)

Tipo de permissão Octal:

- 4 - Indica permissão de leitura;
- 2 - Permissão de escrita;
- 1 - Indica permissão de execução;
- 0 - Indica sem permissões.

Agora é simples, é só somar e ditar as permissões, exemplo:

$4 + 2 + 1 = 7$  (permissão de rwx)

$4 + 2 = 6$  (permissão rw)

$4 =$  (permissão r)

Exemplo: A permissão 610 indica que o arquivo tem permissão:

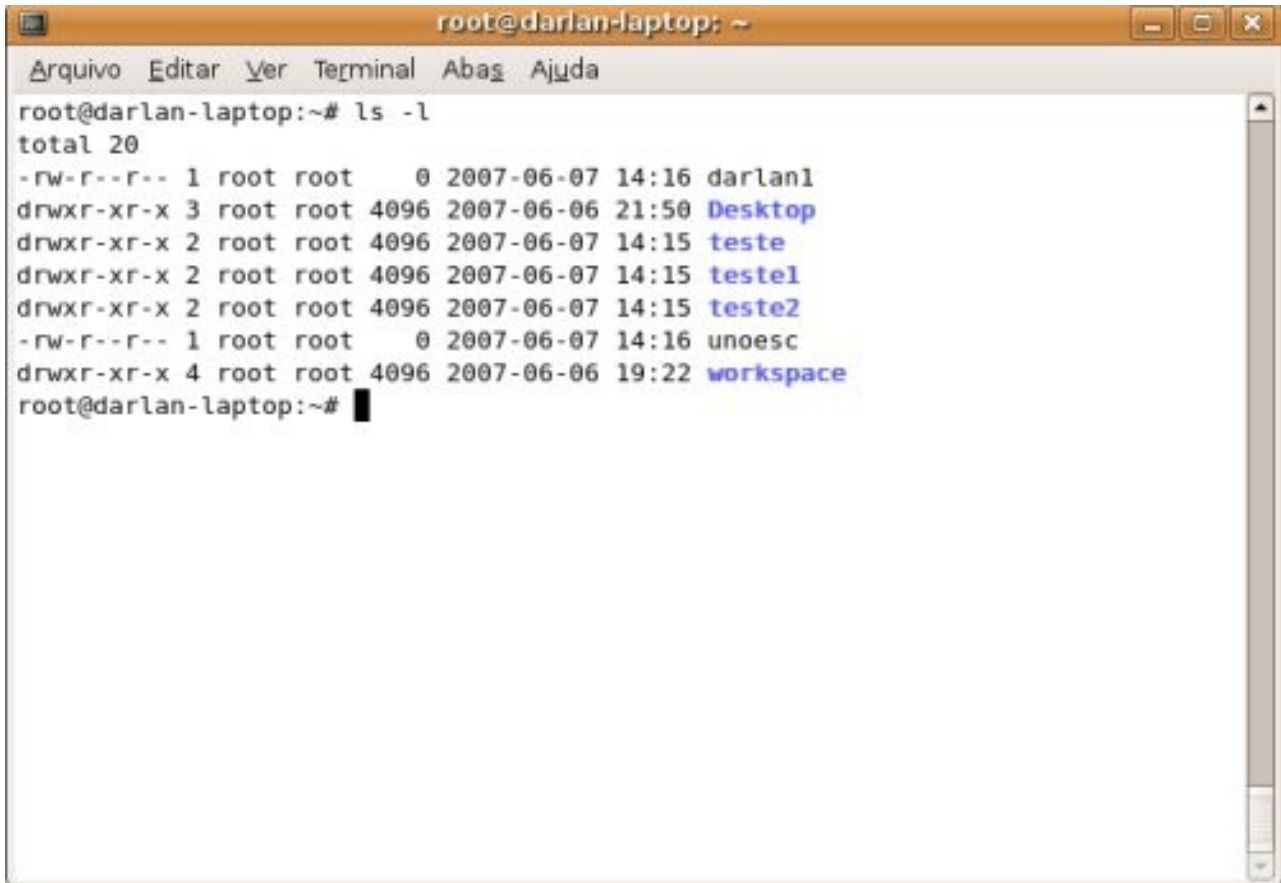
6 para dono do arquivo

1 para grupo e

0 para outros ou seja

dono= (rw\_) Grupo=(\_ \_ x) outros=(\_ \_ \_)

Percebam que quando é feita a listagem longa "ls -l", o primeiro caractere não é uma permissão. vejam a figura abaixo:



```
root@darlan-laptop: ~
Arquivo Editar Ver Terminal Abas Ajuda
root@darlan-laptop:~# ls -l
total 20
-rw-r--r-- 1 root root  0 2007-06-07 14:16 darlan1
drwxr-xr-x 3 root root 4096 2007-06-06 21:50 Desktop
drwxr-xr-x 2 root root 4096 2007-06-07 14:15 teste
drwxr-xr-x 2 root root 4096 2007-06-07 14:15 testel
drwxr-xr-x 2 root root 4096 2007-06-07 14:15 teste2
-rw-r--r-- 1 root root  0 2007-06-07 14:16 unoesc
drwxr-xr-x 4 root root 4096 2007-06-06 19:22 workspace
root@darlan-laptop:~#
```

O primeiro caracter "d" indica o tipo do arquivo, neste caso "d" indica que é um diretório.

Existem também outras permissões especiais, como você pode vê-las em:

- <http://focalinux.cipsga.org.br/guia/iniciante/ch-perm.htm#s-perm-especiais>

### - Umask

Comando que muda a máscara de permissões padrão para a criação de arquivos e diretórios.

Seu uso será explicado mais adiante, apenas em modo OCTAL e não CARACTERE. O modo CARACTERE foi explicado no curso básico, mas não o será nesta apostila.

### - Chown (user:group)

Utilizado para mudar o DONO e o GRUPO dono de um arquivo ou diretório.

Uso: chown novodono:novogrupo arquivoOudiretorio

Observe que a opção :novogrupo pode ser omitida ou trocada por

.novogrupo.

Também aqui existe a opção -R.

### - Chgrp

Utilizado para mudar apenas o grupo dono de um arquivo.

Uso: chgrp novogrupo ArquivoOuDiretório

### - Caracteres Especiais (~, [], \*, ?)

São também conhecidos como METACARACTERES.

Os mais comuns e utilizados são:

\* □ Simboliza TUDO

? □ Simboliza QUALQUER CARACTERE

~ □ Simboliza o HOME do usuário Corrente

[AB]\* □ Qualquer arquivo (\*) começado com A ou com B.

### - Home do usuário

Diretório que pertence ao usuário, onde ele pode tudo. O comando cd isolado leva o usuário até este diretório. Ao logar no sistema, o usuário cai também em seu home.

### - Adicionando um Grupo

Um grupo nada mais é do que a união de diversos usuários com as mesmas características. Por exemplo, poderíamos ter um grupo estudantes ou alunos.

Para adicionarmos este grupo, devemos utilizar o comando:

```
groupadd alunos
```

No arquivo /etc/group será adicionada uma entrada alunos, e será dado um GID (identificador de grupo) a este grupo.

### - Adicionando um Usuário

Qualquer pessoa que for utilizar o linux deve necessariamente possuir um usuário válido na máquina. Lembrando que NÃO devemos utilizar o root a menos que necessário, esta tarefa é importantíssima mesmo para usuários caseiros.

Adicionando:

```
useradd rodrigo -g alunos
```

Adicionamos o usuário rodrigo no grupo alunos. Observe que a opção -g nomegrupo não se faz necessária, e caso seja omitida, teremos comportamentos diferentes em algumas distribuições:

RedHat e familiares (incluindo Conectiva):

Será criado um grupo com o mesmo nome do usuário e este será adicionado neste grupo

Slackware:

O usuário será adicionado em um grupo chamado users

Isto ocorre devido a não padronização deste ato e ao fato de um usuário NECESSARIAMENTE pertencer a algum grupo

## - Senhas SHADOW

O esquema de senhas chamado SHADOW foi criado devido ao fato de o Linux (e os Unix-Like da vida) utilizarem em suas senhas um método de criptografia chamado DES (Data Encryption Standard). Este método é fraco (utiliza chaves de apenas 64 bits) e pode ser facilmente quebrado (Veja o livro Cracking DES para entender melhor sobre este assunto). Como o arquivo /etc/passwd necessita ter permissão de leitura para todos, qualquer usuário facilmente conseguiria obter a senha de ROOT do sistema.

Com isso criaram o SHADOW, onde as senhas criptografadas com o DES ficam no arquivo /etc/shadow que só pode ser visto pelo root.

Sobra então no arquivo /etc/passwd apenas um \* ou ! no lugar da senha criptografada do usuário.

Todas as distribuições linux trazem o SHADOW por padrão.

## - Arquivo /etc/passwd

Este arquivo contém os usuários cadastrados na máquina e informações sobre eles. Sua sintaxe é:

login:UID:GID:Descrição:Home:Shell

Onde:

- login  Nome do usuário na máquina
- UID  Identificador do usuário. O Linux utiliza este número para dar ou tirar permissões. Pode ser repetido entre usuários.
- GID  Identificador do grupo principal do usuário.
- Descrição  Qualquer coisa, se for omitido, deve-se deixar ::. Geralmente coloca-se nome e cargo do usuário
- Home  Diretório pessoal do usuário. Não necessariamente, mas recomenda-se que ele fique no /home e tenha o mesmo nome do usuário.  
Ex: Usuário: Rodrigo  
Home: /home/Rodrigo
- Shell  Shell que o usuário irá utilizar para se logar no sistema. Use /bin/false caso o usuário não deva logar. E /bin/bash caso deva.

OBS:

Deve ter ficado na cabeça do leitor atento o fato de CASO O USUÁRIO NÃO DEVA LOGAR.

Mas quando isto acontece?

Digamos que temos uma aplicação que deve ser executada com as permissões de um usuário. Criamos um para ela, mas este não é um usuário válido, como minha aplicação iria entrar na máquina? Este é um caso.

## - Arquivo /etc/group

Similarmente ao /etc/passwd este arquivo possui as configurações dos grupos (o /etc/passwd possui dos usuários)

Sua sintaxe geral é:

<grupo>:x:<gid>:<usuario1>,<usuario2>

Os usuários que pertencerem a este grupo estarão listados neste arquivo, a menos que o grupo seja primário do usuário, neste caso apenas estaria referenciado em

/etc/passwd no campo gid.

Preste atenção que é neste arquivo que o GID dos grupos está especificado, sendo que o /etc/passwd apenas consulta ele.

### - Porque saber editar manualmente usuários e grupos?

Esta é uma pergunta bem simples, já que teremos de editar manualmente em diversas situações onde desejarmos modificar opções de usuários e desejarmos fazer isto de uma forma rápida e segura.

## Características da Shell BASH

O TAB completa, tanto comandos como nomes de arquivos ou diretórios, use-o.

Setas para cima e para baixo movimentam-no entre os comandos que já foram digitados.

Shift+PageUP sobe a tela.

Shift+PageDown desce a tela

Ctrl+D efetua LOGOFF.

Cuidado!! O Linux FAZ DIFERENÇA ENTRE MAIÚSCULAS E MINÚSCULAS.

Cuidado!! Não existe UNDELETE no Linux porque seu sistema de arquivos se auto-defragmenta durante seu uso.

## Shell Script

Um shell script nada mais é do que se utilizar diversos comandos encadeados em um arquivo. Estes comandos serão executados na ordem em que forem vistos.

Use: sh arquivo ou ./arquivo □ para executar.

### #!/bin/bash

Deve ser utilizado no início dos shell script

Indica qual shell deverá ser utilizada para a execução

Não é necessário

Caso omitido, o script será executado usando-se a shell que o usuário estiver utilizando no momento.

Vejam os exemplos de script:

```
#!/bin/bash
```

```
echo "Tenha um bom dia!"
```

Basta salvarmos estas linhas em um arquivo de texto simples e darmos permissões de execução a ele

```
# chmod 766 script.sh
```

## Comentários

Um recurso muito importante são os comentários, através deles podemos explicar as funções e a finalidade dos comandos existentes no nosso script. Para criarmos um comentário, basta que iniciemos a linha com o sinal de #.

```
#!/bin/bash
# A função deste script é simplesmente mostrar uma mensagem
# na tela
echo "Tenha um bom dia!"
```

Outra possibilidade é fazer com que o próprio script solicite que se de entrada no parâmetro. Isso pode ser feito usando o comando read. Para melhor ilustrar essa situação usaremos um exemplo:

```
#!/bin/bash
# Script que implementa passagem de parâmetros solicitada pelo
# script
echo -n "Digite uma palavra:"
read PALAVRA
echo "Você digitou $PALAVRA"
```

A execução deste script seria:

```
# ./script3
Digite uma palavra: Linux
Você digitou Linux
```

Outra possibilidade interessante é de suprimirmos o echo, durante a utilização do comando read, assim não aparecerá na tela o que for digitado. Isso é aconselhado quando o script solicitar uma senha.

```
echo -n "Digite sua senha:"
read -s senha
```

### Expressões para comparação de valores ou variáveis inteiras

Expressão	Significado	Será verdadeiro se...
-lt	Less than	O primeiro valor for menor que o segundo.
-le	Less than or equal to	O primeiro valor for menor ou igual ao segundo
-eq	Equal to	O primeiro valor igual ao segundo
-ge	Greater than or equal to	O primeiro valor for maior ou igual ao segundo
-gt	Greater than	O primeiro valor for maior que o segundo
-ne	Not equal to	O primeiro valor for diferente do segundo

**Expressões de comparação de strings (caracteres)**

<b>Expressão</b>	<b>Significado</b>	<b>Será verdadeiro se...</b>
string1 == string2 ou string1 = string2	String1 igual a string2	As strings forem iguais
string1 != string2	String1 diferente de string2	Tiverem valores diferentes
string1 < string2	String 1 ordenada antes de string2	Se a string1 for ordenada em uma posição superiora a string2
string1 > string2	String1 ordenada depois de string2	Se a string1 for ordenada em uma posicao inferior a string2
-z	Zero lenght	A string tem tamanho zero
-n	No zero lenght	A string tem tamanho diferente de zero

**Expressões de comparação de arquivos**

<b>Expressão</b>	<b>Será verdadeiro se...</b>
-a arquivo	Se o arquivo existir
-b arquivo	Se o arquivo existir e for um dispositivo de bloco
-c arquivo	Se o arquivo existir e for um dispoitivo de caracteres.
-d arquivo	Se o arquivo existir e for um diretorio
-e	Se o arquivo existir
-f arquivo	Se o arquivo existir e não for um arquivo especial
-g arquivo	O arquivo existe e o SGID esta ativado
-h arquivo	O arquivo existe e é um link simbólico
-k arquivo	O arquivo existe e o stick bit esta ativado
-p arquivo	O arquivo existir e é do tipo pipe
-r arquivo	O arquivo existe e pode ser lido por qualque usuario
-s arquivo	O arquivo existe e seu tamanho é maior que zero
-t arquivo	O descritor existe e refere-se a um terminal
-u arquivo	O arquivo existe e possui o UID configurado
-w arquivo	O arquivo existe e é modificavel
-x arquivo	O arquivo existe e é executavel
-O arquivo	O arquivo existe e foi apropriado por um usuario em especifico
-G arquivo	O arquivo existe e for apropriado por um grupo especifico
-L arquivo	O arquivo existe e é um link simbolico
-S arquivo	O arquivo existe e é um socket

-N arquivo	O arquivo existe e foi modificado desde seu ultimo acesso
arquivo1 -nt arquivo2	Se o arquivo1 for mais recente que o arquivo2
arquivo1 -ot arquivo2	Se o arquivo1 for mais antigo que o arquivo2
arquivo1 -ef arquivo2	Se o arquivo 1 e o arquivo 2 tem o mesmo dispositivo e numero de inode
-o option	Se a opcao de Shell estiver habilitada

## Controle de fluxo

Para controlarmos o fluxo e a tomada de decisoes de nosso script, possuimos uma serie de estruturas, tais como:

- if
- case
- for
- while

### A instrução if

```
if condicao then
    comandos
fi
```

Neste caso, quando a condicao resultar em um valor true o bloco comando sera executado, caso contrario nada ocorrerá. Outra maneira de aplicarmos esta mesma estrutura de controle é:

```
if condicao then
    comandos
else
    comandos
fi
```

Neste caso se a condicao retornar verdadeira o primeiro bloco sera executado, caso contrario o segundo bloco sera executado. Outra possibilidade é o uso da seguinte estrutura:

```
if condicao then
    comandos
elif condicao then
    comandos
else
    comandos
fi
```

Neste caso será testada a primeira condicao, caso ela retorne false a segunda condicao sera testada, caso ambas sejam false os comandos do else serao executados.

```
if [-a "/etc/passwd"]
```

```
then
echo "The password file is still there!"
else
echo "The password file is gone!" |mail root
fi
```

### Instrução case

Outra maneira de controlarmos o fluxo em nosso script é pelo uso da estrutura case. A grande vantagem desta estrutura sobre a condicao if, é notada quando existe um grande numero de testes a serem realizados. O formato de uma instrução case é:

```
case string/variavel in
padrao1)
    comandos;;
padrao2)
    comandos;;
padrao3)
    comandos;;
*)
    comandos;;
esac
```

Na primeira linha indicamos uma variavel ou string, que desejamos testar. Cada padrao é um teste a ser feito com essa string ou variavel, note que cada padrao de comparacao é finalizado com um parentese. O ultimo padrao de teste é um asterisco, ele equivale ao else em uma clausula if, ou seja, caso nenhum teste retorne true, esta sera a condicao executada. O asterisco pode tambem ser usado para substituir parte do nome, como podemos ver no exemplo abaixo:

```
case $nomearquivo in
*.swc )
echo "Este é um documento do OpenWrite";;
*.gif\ *.jpg\ *.png )
echo "Este é um arquivo de imagem";;
*.txt
echo "Este é um arquivo de texto puro";;
esac
```

A clausula acima testa a variavel nomearquivo. O primeiro teste verifica se o nome dele possui uma sequencia qualquer e um .doc ao final. O segundo teste especialmente interessante, pois testa uma lista de possibilidades, e esta lista esta separada por uma barra vertical.

### As intruções while

Uma categoria de estrutura de controle especial são os lacos, ou loops. Esta estruturas possuem um codigo que sera repetido ate que uma determinada condicao seja satisfeita. Abaixo podemos ver a estrutura de um laço while e logo após um exemplo.

```
while condicao
do
comandos
done
```

Um exemplo de aplicação para o laço while:

```
while [ $valor -lt 50 ]
do
$valor = $(( $valor + 1 ))
done
```

O exemplo acima repete o comando “\$valor = \$(( \$valor + 1 ))” até que a condição (while [ \$valor -lt 50 ]) seja satisfeita. Ou seja, poderíamos traduzir a cláusula acima como: Enquanto o valor da variável valor for menor que cinquenta execute  
O valor da variável valor é igual ao valor da variável valor mais um  
feito

A instrução until, tem praticamente o mesmo funcionamento, com a diferença que a instrução while é executada enquanto uma determinada condição for verdadeira e a instrução until é executada enquanto um condição for falsa.

Estrutura da instrução until:

```
until condicao
do
    comandos
done
```

Exemplo de aplicação da instrução until:

```
until [ $valor -eq 50 ]
do
    valor=$(( $valor + 1 ))
    echo $valor
done
```

## A instrução for

Apesar da instrução for, também estar incluída na categoria dos loops ou laços, ela possui uma característica especial. Este tipo de laço executa seus comandos um determinado número de vezes. Abaixo temos a estrutura do laço for:

```
for variavel in lista;
do
comandos;
done
```

Vejamos como aplicar este laço:

```
for i in 'ls /usr/bin';  
do  
    echo "O nome do arquivo é $i";  
done
```

## Funções

Outra possibilidade é de criarmos funções afim de re-aproveitarmos o nosso código. Para criarmos uma função usamos a seguinte estrutura:

```
function nomedafuncao ()  
{  
    comandos  
}
```

## O comando echo

O comando echo é usado para dar saída de um texto na saída padrão. Se o comando echo for informado sem nenhum parâmetro ele irá abrir uma linha em branco.

### Opções do comando echo

Opção	Descrição
-e	Habilita a interpretação de caracteres de escape
-n	Suprime a nova linha depois da saída de comando
\a	Emite um sinal sonoro de alerta
\b	Insere um backspace
\c	O mesmo que -n
\f	Adiciona um linha em branco na saída

## Laboratório Shell Scripts

1. Crie um script que informa o nome de usuário atual e diretório home. (ver dica 1)
2. Crie um script que identifique se no momento é dia, tarde ou noite. (ver dica 2)
3. Crie um script que baseado em uma lista de nomes, crie uma conta de usuário para cada nome.
4. Crie um script que solicite os seguintes dados ao usuário:
  - Partição dos arquivos lidos na inicialização;
  - Nome do arquivo do kernel;
  - Nome do arquivo do initrd;
  - Tempo de espera;

Baseado nessas informações o script deve criar um arquivo de menu de inicialização do GRUB.

DICA 1) Comandos utilizados: pwd, whoami 2) Comando: date

## Mais instruções e exemplos SHELL

### If [ condicao ] then fi

oComando condicional, caso a opção for verdadeira ele executa o que estiver entre o then e o fi

oOpcionalmente pode-se utilizar o else.

oEx:

```
if [ $valor = "1" ]
```

```
then
```

```
    echo "Valor = 1 "
```

```
    echo " legal!"
```

□ Ocorrerá um erro, porque as " permitem que valores dentro dela sejam interpretados.

Neste caso o ! será interpretado como o operador NOT e causará um erro. Portanto deve-se utilizar `` que impede que qualquer coisa seja interpretada.

```
else
```

```
    echo ` Valor != 1 `
```

□ Agora esta certo

```
fi
```

### For variavel in lista do done

oComando de loop

oA variável irá assumir a cada itereção do loop um valor da lista

oEx:

```
for nome in Rodrigo Alberto Gilberto
```

```
do
```

```
    echo $nome
```

```
done
```

olrá ter a saída:

```
Rodrigo
```

```
Alberto
```

```
Gilberto
```

### Script Adiciona 1000 usuários

□ Cria-se um arquivo chamado /tmp/nomes.txt

□ Neste arquivo coloca-se o nome dos 1000 usuários.

□ Faz-se o seguinte script:

```
for user in `cat /tmp/nomes.txt`
```

```
do
```

```
    useradd $user
```

```
done
```

□ Observe o `cat /tmp/nomes.txt` no lugar da lista. A crase faz com que o comando seja executado.

□ O comando CAT lista o conteúdo de um arquivo, no caso nomes.txt que fica no lugar da LISTA.

□ A variável user assume valor por valor desta lista a cada iteração e é utilizado o comando useradd para adicionar o usuário.

□ Também pode ser escrito assim:

```
for user in `cat /tmp/nomes.txt` ; do useradd $user; done
```

## Script Abrir Interface que Escolher

- Lembre-se que o arquivo .xinitrc localizado no home do usuário é executado assim que a interface gráfica abre.

```
#!/bin/bash
echo "Escolha a interface"
echo " 1- KDE"
echo " 2- Wmaker"
echo " 3- Gnome"
read iface
if [ $iface = "1" ]
then
    kde
    exit
fi
if [ $iface = "2" ]
then
    echo "wmaker" > ~/.xinitrc
fi
if [ $iface = "3" ]
then
    echo "gnome" > ~/.xinitrc
fi
startx
```

OBS: Observe o echo "gnome" > ~/.xinitrc, utilizei o metacractere ~ para indicar o home do usuário, assim independente do usuário que executar, o script funcionará.

O comando exit serve para fechar o script, porque eu não desejo que o startx seja executado caso kde tenha sido.

Coloquei um único startx ao final do script, ao invés de um por if.

Utilizei o comando read para acessar um valor digitado pelo usuário e armazenei este valor na variável iface, que não precisa ser previamente declarada.

## Aprofundando-se em SHELL Script

### - Case

Usado quando se necessita de muitos If's.

Funciona assim:

```
case $Variável in
    valor) comando
        comando
        comando
;;
    valor2) comando
        comando
;;
```

```
        *) comando
           comando
           ;;
    esac
```

Exercício: Reescrever o script de abrir interface usando o case

## - While

Faz um loop que só sairá quando encontrar o comando break ou a condição for FALSA.

Sintaxe:

```
while Condição
do
    comandos
done
```

Ex:

```
a=0
while [ $a -le 10 ]
do
    echo $a
    expr $a + 1
done
```

Primeiro indiquei que a variável a tem valor 0.

O While testa se a variável a é menor ou igual a 10 (-le é um operador)

Se for, e é, imprime seu valor, primeiramente 0.

O comando expr soma 1 ao valor de a.

O loop é refeito, novamente se testa se a é menor que 10 e assim por diante.

Perceba o operador -le, temos outros:

```
-eq   Igual
-ne   Diferente
-gt   Maior que
-ge   Maior ou igual
-lt   Menor que
-le   Menor ou igual
```

## - O comando CUT

Este comando embora pouco conhecido é realmente MUITO útil. Sua função é a de capturar apenas uma parte de uma linha, ou expressão.

Quando iríamos querer isto?

Vamos pensar, lembrem-se do arquivo /etc/passwd, nele temos diversas informações.

Porque não simplesmente capturar o nome do usuário e nada mais?

Como faríamos isto? Listar apenas o nome dos usuários do sistema?

Veja:

```
cat /etc/passwd |cut -f1 -d":"
```

Mas o que fizemos?

Primeiramente listei o conteúdo do arquivo com o cat.

Canalizei ele para o comando cut.

**-f1**  Indica que quero o 1º campo. Nossa que primeiro campo?? Vou especificar com o -d

**-d":"**  Indiquei que o que separa cada campo é o caracterer dois pontos (:).

Exercício: Escrever um script para acertar as permissões do HOME dos usuários.

**DESAFIO:** Construir um script que faz o seguinte:

Lê um arquivo chamado LOGS.txt que possui a seguinte sintaxe e será criado a parte:

ArquivoDelog TAMANHO

Ex:

/log/tudo.log 10000

Após ler este arquivo, ele irá ser rodado via CRON (ver mais adiante) e de tempos em tempos ele irá checar os arquivos que estiverem no arquivo LOGS.txt e seus respectivos tamanhos.

Se o tamanho for atingido ou for passado, ele irá compactar o arquivo de log, gerando um chamado nomedoarquivo.log.DIAMESANO.tar.gz.

No nosso exemplo, o arquivo /log/tudo.log ao atingir o tamanho 10000 irá ficar:

/log/tudo.log.19012002.tar.gz  Dia 19 de janeiro de 2002

## Utilizando o VI

**Editando um texto:** vi nomedoarquivo

vi +linha nomedoarquivo  Abre direto na linha

**Indo para a última linha:** :\$

**Indo para a primeira linha:** gg ou :1

**Indo para a ultima linha:** GG

**Indo para a enésima linha:** :n

**Removendo uma linha:** dd

**Removendo a partir da linha n até a última:** :n,\$ d

**Colocando número nas linhas:** :set number

**Tirando número das linhas:** :set nonumber

**Substituindo palavras:** :% s/palavraSerSubstituida/palavraVaiSubstituir/

**Salvando:** :w

**Saindo e salvando:** :x ou :wq

**Saindo sem salvar:** :q ou :q! caso tenha modificado

**Salvando para outro arquivo:** :w outroarquivo

**Saindo e salvando em outro arquivo:** :x outroarquivo

**Forçar salvção:** :x! ou :wq!  Útil quando o arquivo está como RO

**Inserir linha abaixo do cursor:** o

**Inserir linha acima do cursor:** O  
**Inserir:** [INSERT] ou i  
**Substituir:** [INSERT 2x] ou r  
**Inserir no fim da linha:** A  
**Inserir após o cursor:** a  
**Copiar e colar linhas:** y  copia linha  
P  cola linha

## Entendendo o sistema de arquivos do Linux

**Nome:** ext2

**Características:** Pesquisa Binária  
Não fragmentação  
Permissões de Arquivo (podem ser extendidas)  
Arquivos com 255 caracteres no nome  
Qualquer caractere especial no nome  
Por default, não síncrono

**Nome:** ext3

**Características:** Novo sistema  
Journaling FS (assim como o do Aix e o ReiserFS)  
Pesquisa Binária

Grava o que foi feito, não necessita FSCK mesmo caso caia a energia  
Pode ser aumentado em tempo real, sem perda de dados  
Idem a ext2 no restante

### Estrutura de Diretórios

É importantíssimo a qualquer administrador de sistemas entender a estrutura de diretórios do sistema. Isso porque manter a padronização definida, o ajudará a saber onde as coisas estão e a futuros administradores ou auxiliares acharem tudo no sistema.

O Linux possui uma estrutura muito bem organizada e realmente a devemos seguir.

Irei colocar cada um dos diretórios principais e o que neles devem conter:

/etc  Configurações do sistema  
/lib  Bibliotecas compartilhadas necessárias ao sistema  
/mnt  Montagem de discos e periféricos  
/opt  Pacotes adicionais NÃO fornecidos com o sistema (não utilizado

quase)

/sbin  Diretório usado na inicialização do sistema, pacotes essenciais para manutenção. Demais pacotes para a administração do sistema devem ficar em /usr/sbin ou /usr/local/sbin.

/usr  Segundo maior diretório (logo após o /), recomenda-se monta-lo como RO para evitar danos. A grande maioria dos aplicativos do sistema e instalados com ele ficam a partir deste diretório.

/var  Diretório que contém arquivos variáveis, tais como spool (filas de email, crontab, impressão) e logs. Este diretório existe para que os arquivos que necessitem ser modificados fiquem nele e não no /usr, liberando assim sua montagem

como RO.

`/root` □ Alguns Unix-Like não o utilizam (utilizam `/home/root`). É o diretório que contém os arquivos do administrador (seu home).

`/proc` □ Diretório VIRTUAL onde o kernel armazena suas informações. Alguns dados deste podem ser modificados, como os abaixo de `/proc/sys` que contém informações muito pertinentes a performance tuning do sistema.

`/tmp` □ Diretório que contém arquivos temporários. Todos os usuários necessitam poder escrever neste diretório (para gravar seus temporários), no entanto um não pode apagar o temporário do outro (se não teríamos um problema), devido a isto este diretório possui uma permissão especial, que possibilita escrever mas só apagar aquilo que for seu. Esta permissão chama-se STICK BIT.

`/home` □ Os diretórios pessoais dos usuários devem ficar a partir daqui.

`/bin` □ Aplicativos e utilitários usados durante a inicialização do sistema, antes de qualquer sistema de arquivos ser montado. As shells dos usuários costumam ficar aqui também. Binários de usuários devem ficar em `/usr/bin`.

`/boot` □ Contém a imagem do kernel e tudo o que for necessário ao processo de boot, menos configurações.

`/dev` □ Dispositivos do sistema.

## MOUNT

Comando utilizado para se acessar qualquer dispositivo no Linux.

Como dispositivo entenda disquete, cdrom, e o próprio HD.

Sua sintaxe é:

```
mount -t tipofs /dev/dispositivo /ponto/de/montagem
```

Ponto de montagem: Diretório que será utilizado para acessar o dispositivo.

TipoFS: Tipo do sistema de arquivos do dispositivo, geralmente:

Vfat □ Fat32

Msdos □ Fat16

NTFS □ NTFS

Ext2 □ Linux

Iso9660 □ Cdrom

Umsdos □ FS especial, sistema Linux sobre FAT.

Monta FAT32 e FAT16, além de ext2.

OBS: Nossas partições são montadas durante o boot do sistema, por isso `/` é um ponto de montagem que indica uma das partições de nosso HD.

## FSTAB

Como foi dito, alguns sistemas de arquivos devem ser montados durante o boot, para que todo o resto funcione. É o caso de nossas partições ativas do linux e a própria swap.

No arquivo `/etc/fstab` estão as informações do que deverá ser montado automaticamente no boot da máquina e algumas opções de dispositivos que são muito acessados e seu FS não muda (ex: cdrom).

Isto faz com que utilizemos apenas mount /ponto/de/montagem para acessar tais dispositivos, já que o resto das informações o linux busca no fstab.

Sua sintaxe geral:

<i>Dispositivo</i>	<i>ponto de montagem</i>	<i>FS</i>	<i>opções</i>	<i>OrdemBackup</i>	<i>OrdemFSCK</i>
--------------------	--------------------------	-----------	---------------	--------------------	------------------

Onde opções podem ser:

- Auto**
- Async**
- Atime**
- Dev**
- Exec**
- Noatime**
- Noauto**
- Nodev**
- Nosuid**
- Nouser**
- Remount**
- Ro**
- Rw**
- Suid**
- Sync**
- User**
- Defaults** --> rw,suid,dev,exec,auto,nouser,async

#### **Ordem Backup:**

Usado pelo comando dump para fazer backup do FS.

Caso seja 0 não fará backup.

Números 1 acima serão feito backups na ordem (1 primeiro e assim por diante)

Números repetidos será feito backup do que estiver primeiro no fstab.

#### **Ordem FSCK:**

Usado pelo comando FSCK para checar o FS.

Caso seja 0 não será checado.

Números 1 acima serão checados na ordem (1 primeiro e assim por diante)

Em caso de números repetidos será checado primeiro, o que aparecer antes no FSTAB.

#### **FDISK**

Utilizado para se criar ou destruir partições. Observe que seu particionamento é SEMPRE destrutivo.

Uso: fdisk /dev/hdx  X indica o HD e não a partição

Será visto em aula suas opções:

---

---

---

---

---

---

---

---

---

---

## FCK

Não se necessita passá-lo, já que o sistema o faz automaticamente quando precisar, durante o boot.

Observe que caso algum pane aconteça na passagem automaticamente, será solicitada a senha do administrador e deverá ser passado o fsck manualmente.

Digite a senha e faça:

`fsck -c -v /dev/hdxy`  X seu HD, Y partição que deu problema.

`-c`  Manda checar

`-v`  Informa tudo o que está fazendo e pede confirmação

## FDFORMAT

Utilizado para formatar ***DISQUETES***.

Use: `fdformat /dev/fd0`.

Ele não irá criar um sistema de arquivos no disquete. Use o `mkfs`.

## MKFS

Utilizado para criar um sistema de arquivos.

Perceba a necessidade de se especificar o sistema a ser criado e o fato de que o kernel DEVE suportar este sistema.

Uso:

`mkfs -t tipofs /dev/dispositivo`

Onde `tipofs` é um dos tipos já mencionado e `dispositivo` pode ser qualquer dispositivo de armazenamento. Ex: `fd0`, `hda1`.

## MKSWAP

Utilizado para se "formatar" uma swap. Gerar seu sistema de arquivos. Utilizamos ele assim:

`mkswap -c /dev/hdxy`

A opção `-c` é opcional mas recomendada, já que checa o HD antes de criar a swap.

Após isto feito, deve-se necessariamente ativar a swap para que o sistema a reconheça.

Faça isso com o comando:

`swapon /dev/hdxy`

Alguns alunos mais "fuçados" já terão visto:

`swapon -a`

Que indica para se ativar a swap de todos os dispositivos de swap encontrados em `/etc/fstab`.

Lembre-se de adicionar a nova swap lá para que no próximo boot seu sistema a reconheça automaticamente.

Bem, como eu já disse o Linux é poderoso, Linux é Linux,

portanto temos uma opção de caso necessitemos de mais swap, mas não tenhamos uma partição a parte para isto.

É o caso da swap em arquivo. Sua desvantagem é o fato de ser mais lenta, já que estará dentro do sistema de arquivos ext2 além do seu próprio, mas é uma maneira de se ganhar memória SEM TER QUE REINICIAR ou qualquer coisa do tipo.

Para criar este "arquivo de swap", faça os seguintes passos:

- 1-) `dd if=/dev/zero of=/swap bs=1024 count=8139`
- 2-) `mkswap -c /swap`
- 3-) `swapon /swap`

Passo1: O comando `dd` "copia transformando". Ele foi utilizado para se criar o arquivo de swap sem conteúdo (`/dev/zero`) e com um tamanho equivalente a 8139 blocos (8MB). Obviamente você pode adequar este tamanho para o que necessitar.

Passo2: Cria-se o sistema de swap neste arquivo

Passo3: Ativa-se a swap deste arquivo.

Use o comando `free` e você verá que o espaço de swap foi incrementado.

## Configuração e Instalação do Kernel

A grande característica do Linux é o fonte aberto, que facilita o desenvolvimento. Ele realmente está muito desenvolvido e constantemente são lançados muitas versões novas de seu núcleo.

Algumas destas novidades são importantes para nós e por isso desejamos atualizar nosso sistema. Mesmo que não queiramos atualizar, devemos no mínimo recompilar nosso kernel que vem no sistema para que se adapte apenas a nossa máquina, fazendo assim uma grande economia de memória e desempenho.

O primeiro passo é baixar o novo kernel.

Faça isso acessando: [www.kernel.org](http://www.kernel.org)

### Entendendo o conceito de módulos

O kernel do linux é muito bem elaborado e projetado.

Obviamente foi pensado em diversas possibilidades do uso de um sistema operacional, como por exemplo o fato de não utilizarmos o cdrom 100% dos momentos em que estamos no computador.

Se não é assim, porque o suporte ao CD deve estar o tempo todo na memória, ocupando espaço?

Com isto inventaram o conceito de módulos, ou seja, quando o sistema operacional precisa, ele carrega o código para memória, usa e descarrega.

### Gerenciando Módulos

Existem alguns comandos para o gerenciamento de módulos do sistema operacional. São eles:

- `modprobe -l` □ Lista os módulos disponíveis
- `modprobe módulo` □ Carrega o módulo na memória
- `insmod módulo` □ Carrega o módulo na memória mas não checa
- `lsmod` □ Lista os módulos carregados

## Adicionando Suporte a um Novo Hardware

Para isto, devemos recompilar nosso kernel

### Personalizando o kernel (recompilando)

Após baixar o arquivo do kernel, vamos supor chamado:  
linux-2.4.17.tar.bz2

Copie-o para /usr/src e entre neste diretório

Mova o kernel antigo para outro nome (mv linux linux.old)

Descompacte-o (veja compactadores nesta apostila).

Move ele para um nome mais descritivo (mv linux linux-2.4.17)

Crie um link chamado linux (alguns aplicativos precisam)

ln -s linux-2.4.17 linux

Entre no diretório dos fontes

cd linux

Inicie a configuração:

make menuconfig  Recomendado, pode ser make xconfig ou  
make config também

Escolha o que deseja inserir (m para módulo) e saia.

make dep

make clean

make bzImage

make install

make modules

make modules\_install

Vá para a raiz (cd /)

Mova o arquivo vmlinuz (imagem do kernel) para o diretório /boot

Padrão do FS do linux

Mova o arquivo System.map para o diretório /boot também

Edite o lilo.conf ou /boot/grub/menu.lst se for GRUB, para que o sistema boote pelo novo kernel

Digite lilo se for lilo

PRONTO!! Reinicie sua máquina.

OBS: TODOS OS DETALHES SERÃO VISTOS EM AULA. ATENTE-SE PARA CASO NO SEU SISTEMA JÁ EXISTA O ARQUIVO /BOOT/VMLINUZ (FAÇA UM BACKUP DE SEGURANÇA E PREVINA ISTO NO LILO.CONF TAMBÉM).

O LILO.CONF E O GRUB SERÁ EXPLICADO MAIS ADIANTE NESTA APOSTILA.

## Ferramentas Básicas de Rede

Ifconfig  Este comando retorna informações sobre as placas de rede (mesmo virtuais) e a interface lo.

Ele pode ser utilizado para mudar informações de rede assim:

ifconfig interface ip netmask mascaraREDE

Ex:

```
ifconfig eth0 192.168.0.1 netmask 255.255.255.0
```

```
ifconfig eth0:0 192.168.0.2 netmask 255.255.255.0
```

Primeiro, dizemos que o ip de nossa placa será 192.168.0.1 classe C. Depois, configuramos um ip virtual para esta placa também classe C.

**route** □ Comando que gerencia a tabela de roteamento estática do Linux. Apenas iremos utilizar este comando (em casos básicos, claro, roteamentos mais avançados podem ser feitos, lembrando que será um roteamento estático e limitado. Veja: Advanced Routing HOWTO para entender melhor o uso do iproute) para definir o gateway de nossa máquina:

□ **route add default gw IPdoGATEWAY**

□ Use: **route -n** para listar a tabela de roteamento.

**netstat** □ Lista conexões ativas. Use netstat -na para listar todas as conexões ativas e que estão escutando.

**ping** □ Famoso comando que utiliza o protocolo ICMP para verificar o delay de uma máquina a outra na rede.

**traceroute** □ Comando muito utilizado para se verificar o caminho efetuado por um pacote de uma máquina a outra na rede.

**Nslookup** □ Retorna o nome da máquina a partir de seu número IP.

Use: nslookup IP

Uma opção interessante seria:

nslookup -type=mx ip □ Retorna o servidor de email deste

domínio.

## Editando os arquivos de configuração manualmente

Esta parte do treinamento se torna um pouco mais complicada, devido ao fato de as muitas distribuições existentes trabalharem diferentemente com os arquivos de configuração.

No Slackware esta tarefa é muito mais simples, bastando editar-se o arquivo /etc/rc.d/rc.inet1. Neste arquivo, existem variáveis com nomes bem demonstrativos que bastam ser colocados seus valores corretos para trabalhar perfeitamente. Preste atenção que estamos ensinando apenas com uma única interface de rede, não cabendo a este treinamento aprofundar-se no uso de duas. Caso seja da sua necessidade, por favor mande um email para [darlan.segalin@unoescxxe.edu.br](mailto:darlan.segalin@unoescxxe.edu.br) que não exitarei em auxiliar-lhe.

No Conectiva Linux:

Edite o arquivo /etc/sysconfig/network □ Variáveis bem demonstrativas

Edite o arquivo /etc/sysconfig/network-scripts/ifcfg-eth0 □ Variáveis bem demonstrativas.

Estes dois scripts servem para setarmos o IP, GATEWAY e NOME da máquina. Observe que ifcfg-eth0 se trata da primeira interface, se tivéssemos um ifcfg-eth1 seria a segunda e assim por diante.

Podemos também setar uma virtual com ifcfg-eth0:0.

Caso queira setar um ip virtual, apenas copie o ifcfg-eth0 para ifcfg-eth0:0 e edite suas opções.

Após as alterações, faça ifdown eth0 e ifup eth0 para que elas sejam validadas.



Recomendo SEMPRE utilizar o comando route.

### **Configurando sua placa de REDE**

Obviamente este seria o primeiro passo, mas como ele é genérico a todas as distribuições quis coloca-lo aqui primeiro, onde o aluno já estará familiarizado com as configurações de rede do linux.

1-) Adicione suporte a sua placa de rede no kernel (recompilar), geralmente adicionamos como módulo, mas não necessariamente.

2-) Edite o arquivo /etc/modules.conf. Este arquivo contém configurações lidas antes de se carregar qualquer módulo no kernel.

3-) Coloque a seguinte linha:

```
alias eth0 nomemódulo
```

Para a segunda placa, coloque alias eth1 nomemódulo e assim por diante para outras placas de rede.

Perceba que o nome do módulo é apenas seu nome, sem extensão .o ou seu caminho no sistema.

Um exemplo seria:

```
alias eth0 dmfe  
alias eth1 rtl8139
```

Ok, tudo pronto, agora basta testar.

Primeiramente de um ping em seu GATEWAY (definido com o comando route).

Caso não responda cheque novamente as rotas, digite ifconfig para ver se a placa de rede está no ar.

Se tudo parecer correto mas continuar sem responder, verifique o cabo de rede.

## **Compactadores e Empacotadores de Arquivos**

Tópico importantíssimo para a utilização de qualquer sistema operacional, compactadores e empacotadores são SEMPRE utilizados na internet para diminuir o tamanho de arquivos (transferências mais rápidas) e unir diversos arquivos em um único.

Compactador □ Diminui o tamanho de um arquivo. Arquivos texto são mais facilmente compactados e tendem a diminuir entre 60 e 70% de seu tamanho.

Empacotador □ Une diversos arquivos ou diretórios em um único. Os usuários não lidam muito com este termo, já que no sistema operacional concorrente os compactadores são também empacotadores.

### **Gzip**

Ótimo compactador.

Para utilizar:

```
gzip arquivoAcompactar
```

O arquivo será substituído pelo mesmo nome mas com extensão .gz, que indica que ele foi compactado com o gzip.

Para descompactar:

```
gzip -d arquivoAdescompactar.gz
```

```
gunzip arquivoAdescompactar.gz
```

As duas opções acima são idênticas.

### **Zip**

Pouco utilizado no mundo Linux, é um compactador/empacotador.

Utilize zip arquivo.zip Arquivo1 Arquivo2.  
Cuidado ao zipar diretórios!!  
Para deszipar, use:  
unzip arquivo.zip

## Bzip

Compactador mais atual e mais eficiente que o GZIP. Não está sendo muito utilizado e atualmente está na versão 2.

Use:

```
bzip2 arquivoAcompactar
```

Este arquivo será substituído por um com o mesmo nome e extensão

.bz2.

Para descompactar:

```
bunzip2 arquivoAdescompactar.bz2
```

```
bzip2 -d arquivoAdescompactar.bz2
```

As 2 opções são iguais.

## Tar

Praticamente um MITO do mundo UNIX. Este excelente empacotador serve para a criação de BACKUPS, tendo muitas e muitas opções. Veremos aqui as mais importantes, tais como as opções de criação de um arquivo empacotado e já compactado (utilizando gzip, bzip2 ou compress).

Criando um arquivo .tar:

```
tar cvf nome.tar ArquivosouDiretoriosATARGIAR
```

Voltando um arquivo .tar:

```
tar xvf nome.tar
```

Criando um arquivo .tar.gz:

```
tar zcvf nome.tar.gz ArquivosouDiretoriosATARGZIPAR
```

Voltando um arquivo .tar.gz:

```
tar zxvf nome.tar.gz
```

Criando um arquivo .tar.bz2:

```
tar jcvf nome.tar.bz2 ArquivosouDiretoriosATARBZIPAR  Conectiva 10
```

Voltando um arquivo .tar.bz2:

```
tar jxvf nome.tar.bz2 ArquivosouDiretoriosATARBZIPAR  Conectiva 10
```

Existem muitas outras opções do TAR. Veja o manual para maiores

instruções.

## Agendamento de Tarefas

O agendamento de tarefas faz parte do dia-a-dia dos administradores de sistemas. Sempre é necessário automatizar as tarefas. Isto nós aprendemos através da criação de scripts. Mas nossos scripts muitas vezes necessitam ser executados de tempos em tempos. Como fazer isso? Programando sua execução. Utilize o cron e o at para isto.

### Cron

Agenda tarefas a serem executadas sempre. Isto quer dizer, por exemplo, tarefas a serem executadas todo dia a meia noite por exemplo. Ou todo dia 10 as 10:00

horas e assim por diante.

O cron permite o agendamento de tarefas por usuário. Cada um pode ter suas tarefas e estas serão executadas com as permissões deste usuário.

Comumente agendaremos tarefas como root.

Agendando uma tarefa:

### **editando o arquivo /etc/crontab**

Será aberto o editor vi, editando o arquivo correto. Ali dentro, siga a seguinte sintaxe (uma por linha):

**MinutoHoraDiadasemanaMesAno      usuarioparaexecutar      comando**

Ex:

```
0012***      root      mkdir -p /root/darlan/unoesc
```

No nossa linha acima será executado o comando `mkdir -p /root/darlan/unoesc`, as 12:00h, todos dias da semana, todos meses, e todos anos. O \* indica “todos”.

Após adicionar a linha, reiniciar o serviço Cron com o comando:

**/etc/init.d/cron restart**

dependendo sua distribuicao pode alterar o nome para `cron`

## **At**

Agendamento de tarefas a serem executadas uma única vez. Por exemplo, uma tarefa que necessite ser executada hoje a meia noite.

Uso:

```
at hora -f arquivo
```

Ex:

```
at midnight -f /root/Instalados/logrotate.sh
```

Utilize o `atq` para visualizar as tarefas agendadas e `atrm` numeroDatarefa para desagendar uma tarefa.

## **Entendendo o Super Daemon Inetd**

A função do `inetd` é a de economizar memória. Como? Imaginemos um servidor ftp que não recebe muitas requisições, digamos umas 5 por dia, por exemplo.

Este servidor ftp fica o tempo todo na memória da máquina, ocupando um espaço que não é necessário. No entanto, poderia haver alguma maneira de fazer com que este serviço não ficasse no ar o tempo todo, apenas quando necessário.

Daí surgiu o `inetd`. Ele fica no ar, esperando conexões para os mais diversos serviços (configurados em `/etc/inetd.conf`). Quando a conexão chega, ele apenas repassa para o serviço específico.

Infelizmente, como inconveniente temos o fato de que será necessário carregar o serviço para a memória, o que torna o processo mais lento.

### **/etc/inetd.conf**

Sua sintaxe é:

```
ftp      stream      tcp      nowait      root /usr/sbin/tcpd      in.ftpd -l
```

-a

Explicando os campos:

ftp      □ Serviço que escuta na porta 21 (ver `/etc/services`)

stream      □ Modo para conexões TCP (dgram para udp)

tcp  Protocolo de conexão  
nowait  Não ficará esperando  
root  Rodará como usuário root  
/usr/bin/tcpd in.ftpd -l -a  Programa a ser chamado e seus parâmetros

### **/etc/hosts.allow**

Observe no exemplo acima, que o programa a ser chamado é TCPD. Este é o programa que chamamos de TCP Wrappers. Como parâmetro é passado o serviço a ser protegido, no caso in.ftpd.

TCP Wrappers implementa um controle de acesso aos serviços que rodam sob o inetd através do número ip ou nome da máquina.

O arquivo /etc/hosts.allow contém os serviços que serão permitidos. Caso esteja sendo permitido, ele nem sequer irá checar o que está sendo negado.

Sua sintaxe é:

Serviço:IP

Ex:

in.ftpd:localhost  Libera localhost a conectar no serviço FTP

All:200.158.118.169  Libera tudo para a máquina 200.158.118.169

### **/etc/hosts.deny**

Neste arquivo proibimos os serviços. Sua sintaxe é igual a do arquivo /etc/hosts.allow.

## **Configurando o boot da maquina (GRUB)**

O **GRUB** (*GRand Unified Bootloader*), assim como o **LILO** é um gerenciador de boot. Seu trabalho é carregar o kernel do Linux ou iniciar outros sistemas operacionais. Ele surgiu da necessidade de um gerenciador de boot livre e por isso foi criado sob o guarda-chuva do projeto GNU. Com o **GRUB**, você consegue iniciar vários sistemas diferentes, seja Linux, BSDs, Mac, entre outros. Como neste manual gostamos do Linux, vamos puxar mais o saco dele :)

O **GRUB** tem algumas diversas vantagens sobre o **LILO** e é por isso que a maioria das distribuições estão substituindo o **LILO** pelo uso do **GRUB**. Além do menu bonitinho para a escolha do sistema operacional, tecnicamente o **GRUB** tem algumas boas vantagens. Então com essas características legais e divertidas, eu aconselho utilizar o **GRUB** para gerenciar as suas versões de kernel do Linux :)

O arquivo de configuração do **GRUB** geralmente está fixo em um lugar, que é o /boot/grub/menu.lst. Muitas distribuições fazem um link para /etc/grub.conf, então editar os dois arquivos funcionam. A seguir, um exemplo de configuração para dois sistemas operacionais: Linux e Windows:

**# Exemplo para dois sistemas Operacionais: Linux e Windows!**

**#**

**default=0**

**timeout=5**

```
splashimage=(hd0,1)/boot/grub/splash.xpm.gz
```

#### # Partição Linux

```
title Fedora Core
```

```
    root (hd0,1)
```

```
    kernel /boot/vmlinuz-2.6.13-1.1532_FC4 ro root=/dev/hda1 vga=791
```

```
    initrd /boot/initrd-2.6.13-1.1532_FC4.img
```

#### # Partição Windows

```
title Windows
```

```
    rootnoverify (hd1,0)
```

```
    chainloader +1
```

### Exemplo de Arquivo de Configuração do GRUB

No exemplo acima, assim como no **LILO** temos duas partes: uma global com as configurações do funcionamento geral do **GRUB** e uma parte com as definições dos sistemas. Na seção global, podemos definir muitas coisas como senhas, imagem do menu, cores, entre outros.

Antes de mais nada, é preciso notar que o **GRUB** trata as partições de um modo diferente que o Linux. Por exemplo, ao invés de `/dev/hda1` ele usa `(hd0,0)`. Veja a tabela abaixo:

Dispositivo	Equivalente no GRUB
<code>/dev/hda1</code>	<code>(hd0,0)</code> - Partição 1 da IDE Primária Master
<code>/dev/hda2</code>	<code>(hd0,1)</code> - Partição 2 da IDE Primária Master
<code>/dev/hdb1</code>	<code>(hd1,0)</code> - Partição 1 da IDE Primária Slave
<code>/dev/hdc3</code>	<code>(hd2,2)</code> - Partição 3 da IDE Secundária Master
<code>/dev/hdd2</code>	<code>(hd3,2)</code> - Partição 2 da IDE Secundária Slave

Agora vamos ver com detalhes das partes de configuração do **GRUB**!

### Seção de configuração global

```
default=0
```

Qual o sistema será iniciado por padrão caso o usuário não escolha nenhum outro. Serve junto com a opção *timeout* para quando o usuário não fizer nada, ele inicie no sistema padrão. Caso nada seja especificado, o primeiro sistema (*número 0*) será o padrão.

```
timeout=5
```

A quantidade de segundos que o **GRUB** vai esperar o usuário apertar alguma tecla antes de iniciar no sistema padrão definido pela opção *default*.

```
password --md5 $1$QWqsC1$Gtat14yn8l2fy6wUogC080
```

Configura a senha do **GRUB** criptografada em *MD5*. Faz com que o usuário tenha que digitar uma senha antes de poder editar algum dos itens do menu. Combinada com a opção *lock* na configuração das partições, faz com que o usuário só consiga iniciar o sistema digitando a senha.



### Importante

Para gerar esses caracteres malucos criptografados é simples, basta executar o comando grub como root e na shell dele digitar:

```
grub> md5crypt
```

```
Password: *****
```

```
Encrypted: $1$QWqsC1$Gtat14yn8l2fy6wUogC080
```

Então digitando a senha, o programa vai gerar os caracteres malucos e você pode copiar e colar no seu arquivo de configuração como explicado :)

```
splashimage=(hd0,1)/boot/grub/splash.xpm.gz
```

A imagem de fundo para o menu de escolha do sistema. No exemplo está indicando uma imagem no diretório /boot/grub/ em formato .xpm.gz (XPM GZipado). Você pode usar uma imagem de fundo, ou cores.

```
hiddenmenu
```

Se colocado, o **GRUB** não mostrará um menu interativo e vai iniciar o sistema padrão depois de *X segundos* (especificado na opção *timeout*). Se você quiser acessar o menu mesmo assim, terá que apertar **Esc** para mostrar o menu.

```
color frente/fundo [frente/fundo]
```

Com essa opção você especifica cores para os planos de frente e fundo do menu do **GRUB**. As cores são especificadas pelos seus nomes em inglês: black, blue, green, cyan, red, magenta, brown and light-gray; dark-gray, light-blue, light-green, light-cyan, light-cyan, light-red, light-magenta, yellow e white.

O segundo argumento funciona da mesma maneira que o primeiro, mas apenas para os itens que estiverem selecionados no menu (que você seleciona com as setas do teclado).

## Seção de configuração de partições

Aqui você vai colocar as partições/sistemas que vai querer bootar. Cada conjunto de linhas corresponde à um sistema diferente (veja no exemplo anterior :)). Vejamos aqui os parâmetros para a partição Linux que definimos:

```
title Fedora Core
```

Aqui é o título que vai aparecer no menu. No nosso exemplo usamos o "Fedora Core", mas pode ser qualquer outra coisa!

```
root (hd0,1)
```

A partição em que seu root se encontra. Lembrando que essa partição tem que ser onde está o /boot, que é onde fica a imagem do kernel.

```
kernel /boot/vmlinuz-2.6.13-1.1532_FC4 ro root=/dev/hda1 vga=791
```

O caminho do kernel para fazer o boot, junto com as opções do kernel. Aqui você vai usar a mesma coisa que no **LILO** por exemplo :)

```
initrd /boot/initrd-2.6.13-1.1532_FC4.img
```

Onde se encontra o arquivo `initrd` do kernel atual. O `initrd` contém módulos e informações iniciais para o sistema poder iniciar. Ele é carregado um pouco depois do kernel.

`lock` (não-especificado no exemplo)

Com a opção `lock`, você faz com que o usuário tenha que digitar uma senha antes de iniciar o sistema. Combine esta opção com o parâmetro `global password`, explicado anteriormente.



### Importante

Se for usar o parâmetro `lock`, use-o logo após a linha do parâmetro `title`, pois só assim todas as opções de início são bloqueadas.

Estas foram as opções para uma partição Linux. Se você quiser colocar uma Windows (como no exemplo que usei anteriormente), você só terá que substituir algumas coisinhas, como veremos a seguir:

`title Windows`

Mesma coisa que anteriormente, é o título, que no caso agora vai ser Windows :(

`rootnoverify (hd1,0)`

Especifica a partição `root` do sistema, mas ao contrário da opção `root`, não monta a partição. Como o boot do Windows é direto e não precisa ser carregado um kernel pelo **GRUB** (ele tem seu próprio carregador), então é necessário não montar a partição do Windows.

`chainloader +1`

Carrega a partição chamando um outro gerenciador de boot, que é o caso do próprio do Windows.

E assim com essas opções, você pode ir montando o seu menu de inicialização do **GRUB** como quiser! :)

## Mega-exemplo

Assim como na seção do **LILO**, colocarei aqui o mesmo exemplo, só que utilizando um arquivo de configuração do **GRUB**. Vou apresentar um arquivo de configuração para 4 sistemas diferentes, com senha e tudo mais :)

### # Arquivo de configuração do GRUB

#### # Seção de parâmetros globais do GRUB

`default=0`

`timeout=10`

`# password boboalegre`

`password --md5 $1$H35tC1$I45i5oNTY0UaNxZ8TjyIF.`

#### # Seção de partições do GRUB

`# Slackware Linux usando kernel 2.2.16`

`title Slackware Linux (Kernel 2.2.16)`

```
root (hd0,0)
kernel /boot/vmlinuz-2.2.16 ro root=/dev/hda1
```

```
# Conectiva Linux 5.1 usando kernel 2.2.17
title Conectiva Linux 5.1 (Kernel 2.2.17)
root (hd0,1)
kernel /boot/vmlinuz-2.2.17 ro root=/dev/hda2
initrd /boot/initrd-2.2.17.img
```

```
# Conectiva Linux 5.1 usando kernel 2.2.16-cl6
title Conectiva Linux 5.1 (Kernel 2.2.16-cl6)
root (hd0,1)
kernel /boot/vmlinuz-2.2.16-cl6 ro root=/dev/hda2
initrd /boot/initrd-2.2.16-cl6.img
```

```
# Debian 2.2 usando kernel 2.2.17
title Debian 2.2 (Kernel 2.2.17)
root (hd1,0)
kernel /boot/vmlinuz-2.2.17 ro root=/dev/hdb1
initrd /boot/initrd.img-2.2.17
```

```
# Debian 2.2 usando kernel 2.2.18
title Debian 2.2 (Kernel 2.2.18)
root (hd1,0)
kernel /boot/vmlinuz-2.2.18 ro root=/dev/hdb1
initrd /boot/initrd.img-2.2.18
```

```
# Ruindows e Nojenta e Oitxo
title Ruindows e Nojenta e Oitxo
rootnoverify (hd2,0)
chainloader +1
```

```
# Fim da configuração do GRUB
```



#### Importante

Para desativar um sistema operacional da iniciação do GRUB deve-se comentar ou deletar as linhas do título do sistema operacional (title) e sua opções abaixo.

## Instalando o GRUB

Na maioria das vezes a distribuição quando é instalada, instala o **GRUB** como gerenciador de boot. Com o grub instalado na *MBR* do HD, não será mais preciso ficar re-instalando, como é o caso do **LILO**. Mas se é a primeira vez que você configurou o **GRUB** e está instalando, aí tudo bem. Para instalar na MBR do HD Master Primário:

```
# grub-install /dev/hda
```

Se por acaso você quiser instalar em um disquete:

```
# grub-install /dev/fd0
```

Assim você terá um **GRUB** no disquete para quaisquer eventualidades :) Para desinstalar o **GRUB** da MBR, basta você instalar outra coisa por cima, **LILO** ou utilizando o comando `fdisk \mbr` no DOS/Windows.

## Configurando o boot da máquina: (LILO) /etc/lilo.conf

Lilo (LIinux LOader) é o gerenciador de boot mais utilizado e antigo no sistema operacional Linux.

Atualmente temos também ganhando muito prestígio o chamado GRUB. Um exemplo de sua configuração seria:

<code>boot=/dev/hda</code>	<input type="checkbox"/>	HD em que o LILO será instalado
<code>map=/boot/map</code>	<input type="checkbox"/>	Mapa do HD
<code>install=/boot/boot.b</code>	<input type="checkbox"/>	Geometria do HD
<code>prompt</code>	<input type="checkbox"/>	Exibir o prompt do lilo
<code>timeout=50</code>	<input type="checkbox"/>	5 segundos de espera
<code>image=/boot/vmlinuz-FwSec-1.0</code>	<input type="checkbox"/>	Imagem do kernel a ser carregada
<code>label=linux</code>	<input type="checkbox"/>	Quando for escolhido o nome linux
<code>root=/dev/hdb1</code>	<input type="checkbox"/>	Partição raiz = /dev/hdb1
<code>read-only</code>	<input type="checkbox"/>	Monta-se como RO
<code>password=teste</code>	<input type="checkbox"/>	Necessita-se digitar a senha TESTE
<code>other=/dev/hda3</code>	<input type="checkbox"/>	Na partição /dev/hda3
<code>label=outroSO</code>	<input type="checkbox"/>	Será carregado caso outroSO seja escolhido

## Instalação e compilação de aplicativos

No Linux a grande maioria dos aplicativos serão encontrados em seu estado natural, ou seja, em código fonte, geralmente na linguagem C.

Nesta parte do treinamento aprenderemos como compilar tais aplicativos e torná-los executáveis.

Sempre leia o arquivo README e/ou INSTALL.

Geralmente fazemos apenas:

```
./configure  
make  
make install
```

### **gcc**

Em minha opinião o melhor compilador C existente. O kernel do Linux foi feito para ser compilado no compilador C da GNU (o gcc).

Ele possui diversas características importantíssimas:

Total compatibilidade com o ANSI C

Recursos de otimização de código automáticos

Para compilar um programa em C, use:

```
gcc arquivo.c -o arquivoBinárioAserGerado
```

## O comando ./configure

A grande maioria dos aplicativos possui uma opção de configuração que se trata do comando ./configure. Este comando faz com que a aplicação verifique todas as necessidades para a compilação de um programa, tais como bibliotecas e outros aplicativos e o sistema operacional em uso. Geralmente podemos utilizar ./configure -prefix=/dir/a/ser/instalado para especificar um diretório onde a aplicação deverá ser instalada.

Observe atentamente possíveis erros aqui, como a falta de bibliotecas instaladas no sistema.

Quando falta uma biblioteca, devemos instalá-la antes de prosseguir.

Para isto, basta acessar [www.freshmeat.net](http://www.freshmeat.net) <<http://www.freshmeat.net/>> e procurar pela biblioteca faltante.

Ao baixá-la, descompactá-la e entrar no diretório, dando ./configure.

Após isto, make e make install.

Verificar se o diretório em que a biblioteca foi instalada está em /etc/ld.so.config, caso não esteja, inserir.

Após isto, digite ldconfig.

## Módulos do perl

Perl é uma grande linguagem de programação e muitas aplicações necessitam dele. Ele funciona modularizado (assim como o kernel do Linux). Para instalar um novo módulo, entre no diretório do mesmo e digite:

```
perl Makefile.pl
make
make install
```

## comando make

O comando make lê as instruções de um arquivo Makefile no diretório corrente e através destas, compila um programa com códigos separados e com as opções do compilador que se fazem necessárias. Após o ./configure a grande maioria das aplicações pedem o make.

## comando make install

Este comando apenas copia os binários gerados pelo comando make para seus respectivos locais dentro do sistema.

## Quando ocorrem os problemas...

Problemas ocorrem quando estamos fazendo algo errado. Tem certeza que leu o arquivo README e INSTALL e lá não fala nada de diferente para a instalação??

Sim?

Ok, então verifique as dependências, ou seja, verifique se não faltam aplicativos ou bibliotecas, lembre-se de sempre ver as mensagens de erro que aparecem, tanto no configure quanto no make.

Dificilmente uma aplicação conterà erros e estará disponível para download, mas muitas vezes será necessário modificar alguma opção não compatível com o seu sistema. Verifique os erros de compilação e baseado em conhecimento de programação (caso você não tenha, ou peça para quem tem ou tente baixar outra versão ou já binário) tente arrumar.

## Gerenciamento de pacotes RPM

Os pacotes RPM foram inventados pela RedHat Linux e fazem parte do projeto de desmistificação do Linux, tornando-o um sistema operacional mais facilmente gerenciável e acessível a usuários caseiros.

Os pacotes possuem extensão .RPM e são pré-compilados (não estamos falando dos SRPMs).

### Instalando

Para instalar um pacote RPM faça:

```
rpm -ivh pacote.rpm
```

### Removendo

Para remover um pacote RPM faça:

```
rpm -enomedopacote
```

### Atualizando

Para atualizar um pacote RPM faça:

```
rpm -uvh pacote.rpm
```

### Pesquisando

Para pesquisar se determinado pacote RPM está instalado faça:

```
rpm -qa |grep NOME
```

A opção -qa lista todos os instalados e o |grep procura pelo nome, já que geralmente não sabemos o nome, senão poderíamos utilizar diretamente rpm -q nome.

A saída do rpm -qa é o que deve ser utilizado pelo rpm -e quando se deseja excluir um pacote

### Erros Comuns

*Falha nas Dependências:*

Falta algum aplicativo que este precisa para instalar. Caso você tenha instalado tal aplicativo, mas o sistema não esteja reconhecendo (você não instalou como RPM), use a opção -nodeps.

*Pacote mais atualizado já instalado:*

Quando você tenta instalar um pacote mais antigo do que um já instalado, utilize a opção -oldpackage para forçar a instalação.

*Pacote já instalado:*

Quando você deseja instalar um pacote já instalado (talvez o que esteja instalado esteja com problemas), utilize a opção -replacepks.

*Arquivos conflitam:*

Isto acontece quando um pacote que está sendo instalado irá substituir um arquivo existente. Nestas ocasiões use a opção -replacefiles.

*Opção -force:*

```
Une -oldpackage + --replacepks + --replacefiles.
```

## Utilizando APT-GET

## Instalando programas com o apt-get

O apt-get é uma ferramenta extremamente poderosa e prática depois que você aprende os conceitos básicos. Ele pode ser encontrado não apenas no Debian, Ubuntu e no Kurumin, mas em outras distribuições baseadas no Debian, como o Xandros, Memphis e até mesmo no Linspire. Ferramentas como o urpmi, do Mandrake, o synaptic, do Conectiva e o yum, do Fedora também são baseados nele.

Em primeiro lugar, o apt-get utiliza um conceito de fontes de atualização. Ele pode obter pacotes de praticamente qualquer lugar, incluindo CD-ROMs do Debian, unidades de rede, etc. Mas o meio mais usado é justamente baixar os pacotes via internet, o que permite obter sempre as versões mais recentes dos programas.

Para usar o apt-get, o primeiro passo é rodar o comando "apt-get update", que faz com que o apt-get verifique todos os repositórios disponíveis e baixe a lista com os pacotes disponíveis em cada um. Isso permite que ele crie uma espécie de banco de dados, com os pacotes disponíveis, onde cada um pode ser encontrado e qual endereço contém a versão mais recente. Este comando deve ser executado periodicamente. O ideal é que você o use uma vez por semana, ou sempre que for fazer alguma instalação importante:

### # apt-get update

Lembre-se de que para virar root, basta digitar "su" no terminal e fornecer a senha configurada durante a instalação. Ao rodar o Kurumin a partir do CD, use o comando "sudo su" para definir a senha.

No arquivo `/etc/apt/sources.list` é o arquivo onde fica armazenado as configurações de quais repositórios de pacotes o apt deve procurar, da Internet, rede ou cd-rom.

Para adicionar repositório do Cd de sua distribuição utiliza-se o comando:

### # apt-cdrom add

Para procurar se existe um determinado pacote nos repositórios do apt usa-se o comando abaixo:

**# apt-cache search music\*** (Exemplo onde ira procurar todos programas relacionadas a palavra "music" com o \* no final ou antes também pode ser procurado contendo apenas parte do nome do programa.)

Terminado, você pode começar a instalar os programas, usando o comando:

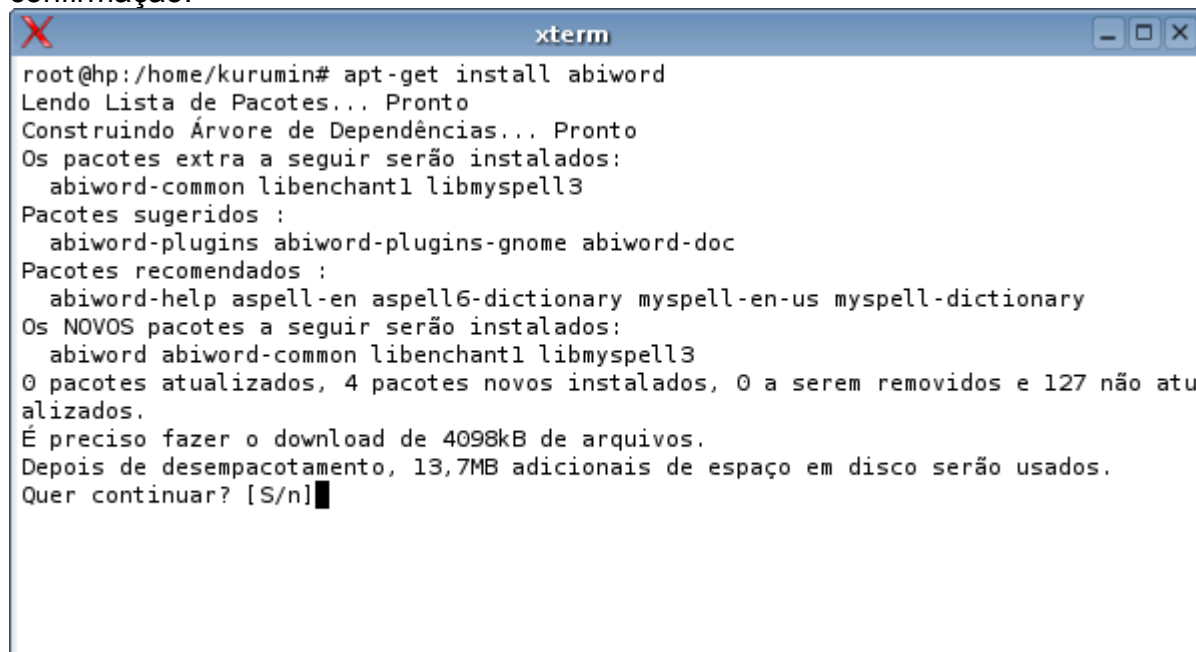
### # apt-get install "seguido do pacote desejado".

Para instalar o Abiword (o processador de textos), por exemplo, use o comando:

### # apt-get install abiword

Veja que o apt-get cuida de toda a parte chata. No meu caso, por exemplo, é preciso instalar também os pacotes "abiword-common", "libenchant1" e "libmyspell3", que o apt-

get instala automaticamente junto com o pacote principal, depois de pedir uma confirmação.



```
root@hp:/home/kurumin# apt-get install abiword
Lendo Lista de Pacotes... Pronto
Construindo Árvore de Dependências... Pronto
Os pacotes extra a seguir serão instalados:
  abiword-common libenchant1 libmyspell3
Pacotes sugeridos :
  abiword-plugins abiword-plugins-gnome abiword-doc
Pacotes recomendados :
  abiword-help aspell-en aspell6-dictionary myspell-en-us myspell-dictionary
Os NOVOS pacotes a seguir serão instalados:
  abiword abiword-common libenchant1 libmyspell3
0 pacotes atualizados, 4 pacotes novos instalados, 0 a serem removidos e 127 não atu
alizados.
É preciso fazer o download de 4098kB de arquivos.
Depois de desempacotamento, 13,7MB adicionais de espaço em disco serão usados.
Quer continuar? [S/n]
```

Terminada a instalação, o Abiword já está pronto para usar. Você vai encontrar o ícone dentro do menu "Escritório e utilitários", no iniciar. Alguns programas podem não criar corretamente os ícones no iniciar, mas você sempre pode chamá-los via terminal ou criar o ícone manualmente, usando o kmenuedit.

Além de instalar, é possível usar o apt-get para atualizar qualquer pacote do sistema. Para isso, repita o comando de instalação, como em:

```
# apt-get install abiword
# apt-get install k3b
# apt-get install mplayer
```

... e assim por diante.

Quando o programa solicitado já está instalado, o apt-get verifica se existe uma versão atualizada e, em caso afirmativo, já a baixa e instalada automaticamente. Caso contrário, ele simplesmente avisa que a versão mais recente já está instalada e não faz nada. Quando você não se lembrar do nome completo do programa, digite apenas as primeiras letras e pressione a tecla **TAB** duas vezes, assim você verá uma lista com as alternativas possíveis.

Uma vez instalado o programa, o comando para chamá-lo pelo terminal (em 99% dos casos) é o próprio nome do pacote. Por exemplo, para usar o k3b, você instala o pacote "k3b" e para chamá-lo pelo terminal, usa o comando "k3b". Na maioria dos casos, é criado um ícone no iniciar, mas, caso necessário, você pode fazer isso manualmente usando o kmenuedit, que você acessa ao clicar com o botão direito sobre o "K" do iniciar.

Em casos onde o programa realmente "desapareça" depois de instalado, sem rastros aparentes, você pode procurá-lo usando o comando "whereis" (onde está), como em:

```
$ sudo updatedb
```

```
$ whereis realplay  
realplayer: /usr/bin/realplay
```

Em muitos casos, o programa pode ser instalado em uma pasta fora do PATH (as pastas /bin, /usr/bin, /usr/local/bin, etc.), fazendo com que o sistema não encontre o comando. Nestes casos, chame-o indicando o caminho completo ou crie um link para ele (usando o comando "ln -s") dentro da pasta "/usr/bin", ou "/usr/local/bin", onde o sistema consiga localizá-lo diretamente, como em:

```
# ln -s /usr/lib/realplayer10/realplay /usr/bin/realplay
```

Lembre-se de que em muitos casos o nome do executável do programa pode ser diferente do nome do programa. Por exemplo, o executável do Realplayer é "realplay" e o do VMware Player é "vmplayer".

Um detalhe interessante é que, mesmo ao atualizar um programa, as suas configurações são mantidas. Ao atualizar o Firefox ou o Konqueror, por exemplo, você não perde seus bookmarks. Isso acontece porque as configurações e arquivos referentes aos programas são armazenados em pastas ocultas dentro do seu diretório de usuário. Os bookmarks, cache, cookies e outros arquivos do Firefox, por exemplo, vão para a pasta ".mozilla/firefox", dentro do seu home. O apt-get nunca altera estes arquivos, de forma que suas preferências sempre são preservadas durante os upgrades.

Um segundo tipo são os arquivos de configuração do sistema, que também fazem parte dos pacotes. Quando um pacote traz uma nova versão de um determinado arquivo de configuração, mas o apt-get percebe que o arquivo anterior foi alterado por você, ele pergunta se você quer manter o arquivo atual ou se você quer substituí-lo pela nova versão. O conselho geral nestes casos é responder não à substituição (que é o default). Isso mantém o arquivo atual, que, afinal, está funcionando. Autorize a substituição apenas quando você souber do que se trata.

Lembre-se de rodar o "**apt-get update**" periodicamente, de preferência uma vez por semana ou antes de instalar qualquer programa ou atualização importante. Assim você terá certeza de que o apt instalará sempre as versões mais recentes dos programas.

O apt não apenas torna a instalação de novos programas bem mais simples, mas diminui também a necessidade de estar sempre instalando versões mais recentes da distribuição, já que você pode ir atualizando os programas mais usados sempre que souber de uma versão mais nova. É possível também remover pacotes instalados, neste caso usando o parâmetro "remove", como em:

```
# apt-get remove abiword
```

Caso os arquivos referentes ao programa tenham se corrompido de alguma maneira (desligamentos incorretos, problemas de BIOS, etc. ;), você pode forçar sua reinstalação, usando o parâmetro "--reinstall".

Normalmente o apt-get avisa que o programa já está em sua versão mais recente e não faz nada:

```
# apt-get install bluefish
```

Lendo Lista de Pacotes... Pronto  
Construindo Árvore de Dependências... Pronto  
bluefish já é a versão mais nova.

Adicionando o "--reinstall" ele prossegue com a instalação, mesmo que o pacote já seja a versão mais recente, substituindo todos os arquivos referentes a ele e resolvendo o problema.

### # apt-get install --reinstall abiword

Finalmente, existe a opção de atualizar todo o sistema, o que é feito usando os comandos:

```
# apt-get update
```

O "apt-get update" é o comando que baixa a lista dos pacotes disponíveis, que já vimos. O "apt-get upgrade", por sua vez, age de forma bem diferente: ele verifica todos os pacotes do sistema e tenta atualizar todos de uma vez, o que geralmente resulta em uma longa lista de atualizações:

Além da linha de comando existem, naturalmente, programas gráficos que tentam facilitar a configuração, como o **Synaptic**, que abordo a seguir. A questão de facilitar ou não é, na verdade, polêmica, pois muita gente acha mais fácil trabalhar com o Synaptic, enquanto outros acham sua interface muito complicada e preferem continuar usando os comandos. Mas, de qualquer forma, ele não deixa de ser uma opção interessante.

### DICAS ( Alguns programas úteis que pode ser instalados com APT )

```
# apt-get install mplayer (programa similar ao media player do Windows)
```

```
# apt-get install xmms (Winamp Linux)
```

```
# apt-get install beryl beryl-manager (Programas para Animações 3D no linux, melhor que o vista)
```

```
# apt-get install wine (Emulador de Programas Windows, Pode ser instalado photoshop, corel draw, office e varios outros programas do Windows)
```

## Configurando o NFS

Network File System (NFS) consiste em um serviço muito utilizado no mundo UNIX e obviamente portado para o Linux para o compartilhamento de arquivos em rede.

Ele permite se acessar uma máquina remota, montando-a em sua própria como um dispositivo qualquer, permitindo assim um acesso simples, como a um diretório local.

### oEscolhendo os diretórios a compartilhar

Os diretórios que desejarmos compartilhar devem ser mencionados no

arquivo /etc/exports.

A sintaxe deste arquivo segue o seguinte exemplo:

```
/home/focker          192.168.0.*(ro)
```

Primeiramente temos o diretório a ser compartilhado: /home/focker

Máquinas que podem acessar o compartilhamento: 192.168.0.\* -->

ClasseC

Permissões de acesso: ro --> Somente Leitura

Observe que por mais que você compartilhe com permissões de leitura e escrita (rw) é necessário que no sistema de arquivos local, tal permissão também seja dada a todos no sistema, isto devido ao fato de que as permissões do sistema local sobressaem-se sobre as do NFS.

oAtualizando tabela de arquivos compartilhados

Para atualizar a tabela de arquivos compartilhados devemos utilizar o comando `exportfs -a`.

oPortmap

Serviço necessário para o bom funcionamento do NFS e de qualquer serviço baseado em RPC (Remote Procedure Call).

RPC foi inventado pela Sun para que serviços não utilizassem uma porta fixa para rodar.

Devido ao fato de não rodar em uma porta fixa, faz-se necessário que algum serviço indique aos clientes que tentem acessar em qual porta o servidor está rodando, esta é a função do Portmap.

oAccionando o Portmap

Para ativar o portmap basta chamá-lo na linha de comando e no caso do Conectiva ainda tem-se a opção:

```
cd /etc/rc.d/init.d
```

```
./portmap start
```

oAccionando o NFS

Após todas estas configurações feitas, deve-se acionar o NFS. Chame-o na linha de comandos (`nfsd`).

No Conectiva:

```
cd /etc/rc.d/init.d
```

```
./portmap start
```

oVerificando arquivos compartilhados pela máquina

Para verificar quais arquivos estão sendo compartilhados por uma máquina, faça:

```
showmount -e IP
```

No caso de você estar na máquina que você quer ver, apenas faça `showmount -e`.

oVerificando clientes que estão acessando os arquivos compartilhados

**Para se verificar quais clientes estão acessando a máquina, faça:**

```
showmount IP
```

Caso você esteja na máquina, faça apenas `showmount`.

## SAMBA



*guest account = nobody* --> Permitirá acesso a usuários não autenticados

## Compartilhando Diretórios através do SAMBA

Como pudemos perceber na seção global de nosso arquivo *smb.conf*, não está sendo especificado nenhum diretório que será compartilhado.

Iremos fazer isso agora:

<i>[public]</i>	--> Nome do compartilhamento
<i>comment = Espaco Publico</i>	--> Comentário para este compartilhamento
<i>browseable = yes</i>	--> Poderá ser visto por todos
<i>create mode = 0777</i>	--> Ao ser criado, a permissão será 0777
<i>directory mode = 0777</i>	--> Mesmo para as permissões de diretório
<i>path = /home/Backups</i>	--> Localização do diretório compartilhado no sistema
<i>public = yes</i>	--> Este diretório tem acesso público, todos podem acessá-lo
<i>only guest = yes</i>	--> Usuários não autenticados irão acessar diretório
<i>read only = no</i>	--> Não está somente para leitura

Esta configuração acima é um exemplo sutil de um compartilhamento aberto para todos os indivíduos que desejarem acessar.

Agora, caso se deseje algo mais restrito, poderíamos ter compartilhamentos baseados em usuários (usuário + senha):

<i>[rodrigo]</i>	--> Nome do compartilhamento
<i>comment = Directorio Rodrigo</i>	--> Comentário
<i>browseable = yes</i>	--> Poderá ser visto
<i>create mode = 0770</i>	--> Máscara da criação
<i>directory mode = 0770</i>	--> Máscara de diretórios
<i>path = /home/rodrigo</i>	--> Localização na máquina
<i>valid users = @rodrigo</i>	--> Usuários que podem acessar
<i>read only = no</i>	--> Pode-se escrever

Existem diversas configurações avançadas que se podem fazer através do samba, que serão vistas mais adiantes neste treinamento.

Obviamente devido a escassez do tempo, muitas delas serão deixadas para que o aluno aprenda sozinho. O principal do caminho lhes será fornecido.

## Acessando arquivos compartilhados através do Linux

Obviamente os usuários Linux também irão querer acessar dados compartilhados pelas máquinas Windows (ou servidores SAMBA).

Nestes casos, temos três opções:

*Utilizar o SMBClient*

*Utilizar o SMBMount*

*Utilizar o comando mount -t smbfs*

A terceira opção chama a segunda quando executada no sistema, portanto não iremos explicá-la.

Nenhuma das duas opções necessitam configurações da máquina Linux que acessará os compartilhamentos.

Basta fazer:

```
smbclient \\\servidor\compartilhamento -U <usuario>
```

ou

```
smbmount \\\servidor\compartilhamento <ponto de montagem>
```

```
username=<usuario>
```

A grande diferença entre as duas opções anteriores está no fato do SMBMOUNT criar um ponto de montagem para o compartilhamento remoto, permitindo assim que o acessemos como um diretório local, enquanto o smbclient possui um cliente específico.

## Configurando o SSHD

O SSH veio para substituir as fraquezas encontradas com o TELNET. Ele fornece recursos para administração remota criptografada.

Seu arquivo de configuração comumente fica em /etc/ssh/sshd\_config e para ativá-lo basta digitarmos sshd, já que ele vem pré-configurado ou iniciando seu serviço com no init.d com o comando **/etc/init.d/sshd restart**.

O servidor SSH lê os arquivos de configuração /etc/hosts.allow e /etc/hosts.deny assim como o TCP Wrappers.

### Retirando a compatibilidade com a Versão 1

É importante atentarmos para isto, já que tal compatibilidade possuía uma vulnerabilidade em algumas versões do OpenSSH.

Mude a linha:

```
Protocol 2,1 do arquivo de configuração para Protocol 2
```

### Retirando o uso da diretiva UseLogin

A diretiva UseLogin também possuía vulnerabilidade em algumas versões do OpenSSH, portanto convém desabilitá-la (atente que esta vem desabilitada por padrão).

Para isto, basta alterar:

```
UseLogin yes para UseLogin no
```

### Aumentando a chave criptográfica

Você pode querer aumentar o tamanho da chave criptográfica a ser negociada com o servidor. Para isto basta alterar a diretiva:

```
ServerKeyBits 624
```

Para:

```
ServerKeyBits 2048 por exemplo, como eu costumo recomendar.
```

Deixe pelo menos 1024 bits de chave criptográfica.

## Permitindo ao root se logar remotamente

Isto não é nem um pouco recomendado, mas não sei por qual motivo insano você pode necessitar disto, portanto ensinarei aqui.

Lembre-se você pode logar-se remotamente com um usuário e utilizar o su para se tornar root.

Caso deseje se logar diretamente como root, use a diretiva:

```
PermitRootLogin yes
```

## Configurando o ProFTPD

Eu recomendo fortemente o uso do ProFTPD sobre o WU FTPD (padrão na maioria das distribuições, não no Slackware 8.0).

Esta recomendação é devida ao fato do servidor Wu FTPD possuir muitas vulnerabilidades e um histórico de falhas muito grande e o proftpd ter sido feito com grande ênfase em segurança.

Abaixo coloco um exemplo de um arquivo de configuração do proftpd e explico suas diretivas mais importantes.

Preste atenção que o proftpd possui MUITOS recursos. Para conhecer todos eles, acesse o site [www.proftpd.org](http://www.proftpd.org) <<http://www.proftpd.org/>> e veja o manual.

O proftpd lê os arquivos /etc/hosts.allow e /etc/hosts.deny assim como o TCP Wrappers.

### Arquivo: /etc/proftpd.conf

```
ServerName      "Firewalls Security FTP Server"    Nome do servidor
ServerType standalone  Tipo: pode ser standalone ou inetd para rodar sob o inetd
DefaultServer   on
Port            21                                Escutará na porta 21
Umask          022
MaxInstances    30
User            nobody                             Rodará com permissões do usuário nobody
Group           nobody
RootLogin       no                                Não permitirá o root se logar via ftp
DisplayConnect  /etc/welcome.FTP
ServerIdent on  "Bem Vindo"
<Directory /*>
    AllowOverwrite on  Permite sobrescrever arquivos em qualquer diretório
</Directory>
<Anonymous ~ftp>  Usuário anonymous cairá no home do usuário ftp
    User ftp  Logará como usuário ftp
    Group ftp
    UserAlias anonymous ftp  Permite entrar com user=anonymous que virará ftp
    MaxClients 10  Máximo de conexões anônimas permitidas
<Limit WRITE>
    Deny All  Não permite nenhum usuário anonymous escrever no sistema
</Limit>
</Anonymous>
```

## Configurando o Apache

### Sobre

O apache pode ser encontrado em [www.apache.org](http://www.apache.org) e é um servidor WEB muito poderoso e gratuito.

Atualmente ele vem sendo utilizado por cerca de 60% dos servidores WEB do mundo e foi inclusive recomendado pelo GartnerGroup.

Com o atual crescimento e onda de vírus e ataques a servidores WEB no mundo, o apache ganhou muita confiabilidade devido a sua imunidade a estes ataques e a comprovação de sua extrema segurança.

### Porque utilizá-lo

- Robusto
- Seguro
- Rápido
- Flexível
- Administrável
- Cheio de recursos e opções
- Fácil configuração

### Exemplo de configuração

Aqui forneço um exemplo de configuração do Apache e explico os pontos mais importantes.

#### # Configurações Principais

Port 80	<input type="checkbox"/> Servidor Roda na Porta 80
User nobody	<input type="checkbox"/> Roda com permissões do user nobody
Group nobody	
ServerType standalone	<input type="checkbox"/> Roda standalone e não sob inetd
ServerRoot /etc/httpd	<input type="checkbox"/> Arquivos do servidor estão em /etc/httpd
MinSpareServers 2	<input type="checkbox"/> Mínimo de 2 servidores ativos (e sem conexão)
MaxSpareServers 4	<input type="checkbox"/> Máximo de 4 servidores ativos (e sem conexão)
StartServers 2	<input type="checkbox"/> Inicia 2 servidores no começo
MaxClients 150	<input type="checkbox"/> Máximo de 150 conexões

# Módulos (apenas um exemplo, o servidor não funcionará se não forem declarados outros módulos necessários)

LoadModule php4_module	modules/libphp4.so	<input type="checkbox"/> Identifica módulo php4
<IfDefine SSL>	<input type="checkbox"/> Se for definido SSL (httpd -DSSL)	
LoadModule ssl_module	modules/libssl.so	<input type="checkbox"/> Identifica módulo ssl
</IfDefine>		
ClearModuleList	<input type="checkbox"/> Limpa lista de módulos carregados	
AddModule mod_php4.c	<input type="checkbox"/> Carrega (insere) módulo no apache	

```
<IfDefine SSL>
    AddModule mod_ssl.c
</IfDefine SSL>

# Hosts Virtuais
NameVirtualHost 200.180.126.26

# Diretórios
<Directory "/var/www/html/unoesc" >
    Options Indexes FollowSymLinks Includes ExecCGI
    AllowOverride None
    Order allow, deny
    Allow from all
</Directory>
<Directory "/home/httpd/tcpdump" >
    Options Indexes FollowSymLinks Includes ExecCGI
    AllowOverride None
    Order allow, deny
    Allow from all
</Directory>

# Arquivos
<Files .htaccess>
    Order allow, deny
    Deny from all
</Files>

# Logs
ErrorLog /log/apache/ApacheError.log
LogLevel warn
AccessLog /log/apache/ApacheAccess.log

# Opções Diversas
UserDir web
DirectoryIndex index.html index.htm index.php
AccessFileName htaccess
HostnameLookups Off
Alias /focker

# Locations
<Location /focker>
    Order deny,allow
    Deny from all
    Allow from localhost
</Location>
```

Diretório "/var/www/html/unoesc" Opções de acesso ao diretório

Não permite ser sobreposto pelo arquivo

Ordem de permissão

Permite todos

Arquivos nomeados .htaccess

Log de Erros do Apache

Logs no nível de Alerta (normal)

Logs de Acesso do Apache

Diretório nos homes dos usuários que armazenam suas páginas, será acessado assim: `ww.servidor.com.br/~usuario`

Estes arquivos são os índices das páginas

Opções de acesso também são encontradas no arquivo `.htaccess`

Não transforma Ips em Nomes

Apelido para `/focker`

Caso acessem `www.servidor.com.br/focker`

Ordem de permissão

Proíbe todos

Exceto da máquina local

## Segurança

### Quesitos para um sistema seguro

Muitas pessoas nos perguntam constantemente o que é necessário para se montar um sistema seguro.

Possivelmente elas se decepcionam com as respostas:

Basta se manter atualizado e ter uma boa política de segurança

Isto quer dizer que devemos agir com segurança, sempre pensar na segurança.

Não existem regras fixas. Não existe segurança 100%.

O básico que podemos fazer será visto aqui, obviamente técnicas e implementações mais avançadas de segurança existem, mas não caberiam no escopo deste treinamento.

E vale sempre lembrar:

DESABILITE tudo que não for necessário!!!

## Montando um firewall linux

### Entendendo o iptables

Netfilter e as iptables são a mais nova inovação em Firewall para Linux. Finalmente diversos conceitos avançados de Firewall foram implementados para um sistema Linux.

Máquinas Linux podem agora substituir caríssimas “caixas pretas”, ou seja, hardwares comerciais feitos exclusivamente para atuarem como Firewall de uma rede.

#### □ Regras de entrada, saída e passagem de pacotes

Existem 3 possibilidades para um pacote transitando em uma com um Firewall e que envolvam este Firewall.

1-) O pacote é para o Firewall

Regra de Entrada: INPUT

2-) O pacote passará pelo Firewall

Regra de Repasse: FORWARD

3-) O pacote sairá do Firewall

Regra de Saída: OUTPUT

#### □ Tabelas do iptables

O Iptables como o próprio nome indica trabalha com tabelas. Existem 3 tabelas no Iptables, a saber:

1-) Tabela Filter

Tabela default que trata os pacotes normalmente

2-) Tabela NAT

Utilizada em mascaramentos e NAT

3-) Tabela Mangle

Utilizada para reescrever informações dos pacotes

Obviamente devido ao tempo, iremos tratar apenas da tabela FILTER

#### □ Proibindo protocolos

Antes de lidarmos com o iptables em si, é muito importante salientarmos que um bom Firewall consiste de um conjunto de regras bem elaboradas, com uma política de segurança consistente.

Alguns de vocês já escutaram e todos com certeza escutarão falar sobre a CHAIN default de um firewall.

Mas o que é isto?

Chain DEFAULT nada mais é do que o que o Firewall deverá fazer ao se deparar com uma condição que não está prevista, ou seja, não existe uma regra específica para esta condição.

O ideal é SEMPRE utilizarmos CHAIN DEFAULT DROP, mais ou menos aquela política:

“ TUDO que não estiver explicitamente liberado está proibido”

Quando fazemos isto, apenas devemos ir colocando nossas regras para liberar o que desejamos fazer, todo o resto já estará proibido.

Vamos para a parte prática:

# Colocando as CHAIN Default

```
iptables -P INPUT DROP          --> Chain default de entrada
iptables -P OUTPUT ACCEPT       --> Chain default de saída
iptables -P FORWARD DROP       --> Chain default de repasse
```

# Proibiremos todos os pacotes TCP:

```
iptables -A INPUT -j DROP -p tcp
```

-A INPUT --> Significa que estamos colocando esta regra ao final de nossas regras de entrada

-j DROP --> Significa que esta regra é uma regra de PROIBIÇÃO

-p tcp --> Esta regra se refere ao protocolo TCP

□ *Proibindo portas e tipos de mensagens*

# Proibindo darem ping em nós

```
iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

Novamente, agora o que mudamos?

-p icmp --> Protocolo ICMP (usado pelo aplicativo PING)

--icmp-type --> Esta regra nos permite especificar que tipo de mensagem icmp estaremos lidando, neste caso a solicitação de ping (echo-request)

# Permitindo para nós darmos ping

```
iptables -A INPUT -p icmp --icmp-type echo-replay -j ACCEPT
```

Observando que agora mudamos o tipo de mensagem icmp e aceitamos este tipo.

□ *Proibindo flags TCP*

O que são FLAGS TCP?

Obviamente que um profissional de segurança precisa dominar perfeitamente os conceitos envolvidos em rede.

Não é o objetivo deste treinamento desenvolvermos toda esta habilidade nos alunos, mas sim, colocarmos eles à par do que existe no mundo Linux.

Explicando de um modo simples, flags TCP são campos de um pacote TCP que permitem identificar qual o objetivo de uma TRANSMISSÃO.

Ou seja, podemos desejar a abertura de uma conexão (Flag SYN ativada), ou o

fechamento de uma conexão (Flag FIN ativada).

Podemos também especificar a aceitação de uma conexão (Flag Syn e ACK) ativadas.

Um Firewall pode ser chamado de Stateful Firewall quando possui armazenada os estados das conexões que por ele se estabelecem.

Ou seja, ele sabe quem abriu ou quem está fechando uma conexão através dele, criando tabelas de controle (connection track).

Vamos imaginar um Firewall Linux de Antigamente (antes da série 2.4 do kernel do sistema, quando ainda utilizávamos ipchains).

Caso você tenha bloqueado um pedido de conexão na porta telnet de uma máquina de sua rede:

```
ipchains -A forward -j deny -p tcp -d 192.168.0.1 23 --syn
```

Observe que a opção final --syn especifica que a flag SYN deste pacote tem de estar ativada.

Os atacantes aproveitavam da “falta de memória” do Firewall para fazer o que chamamos de Stealth Scan, ou seja, pesquisar as portas abertas de uma máquina de um modo que o Firewall não detectava.

Como eles faziam isso?

Forjando pacotes com a FLAG ACK ativa, como se fosse uma aceitação de conexão. Devido a não ter este controle, os scans passavam despercebidos pelo Firewall.

Com o iptables podemos controlar as flags e os estados das conexões.

Controlando Flags:

```
iptables -A INPUT -p tcp -j DROP --tcp-flags SYN,FIN SYN,FIN
```

O que isto quer dizer?

Quer dizer que caso as FLAGS SYN e FIN (--tcp-flags SYN,FIN SYN,FIN) estejam setadas no nosso sistema irá bloquear a conexão. Isto faz sentido, já que ninguém iria querer ABRIR e FECHAR a conexão em um mesmo pacote.

#### □ *Proibindo estados de pacotes*

Estado de um pacote podem ser 4:

- Novo (NEW)
- Estabelecido (ESTABLISHED)
- Relacionado (RELATED)
- Inválido (INVALID)

Um pacote se diz novo quando está sendo estabelecida uma conexão

Estabelecido quando faz parte de uma conexão já estabelecida

Relacionado quando é um novo pacote, mas faz parte de uma conexão já estabelecida

O FTP por exemplo utiliza 2 portas de comunicação, caindo nesta regra

Inválido

Quando possui algo estranho com o mesmo

Uma regra interessante seria a seguir:

```
iptables -A INPUT -j DROP -p tcp -m state --state INVALID
```

Onde estamos proibindo todos os pacotes inválidos de virem para o Firewall.

□ *Limitando as conexões*

Muitos ataques realizados (a grande maioria) visam apenas causar algum tipo de negação de serviço, ou seja, impedir que o servidor funcione adequadamente.

Para evitar este ataque, podemos utilizar algumas regras simples, porém práticas de controle:

```
iptables -A INPUT -j ACCEPT -p icmp -m limit --limit 3/m
```

Com esta regra limitamos o número de pacotes icmp aceitos em nosso firewall para 3 por minuto.