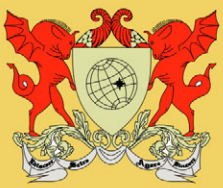
Material ampliado e adaptado do Guia Foca Linux

Curso Básico de LINUX

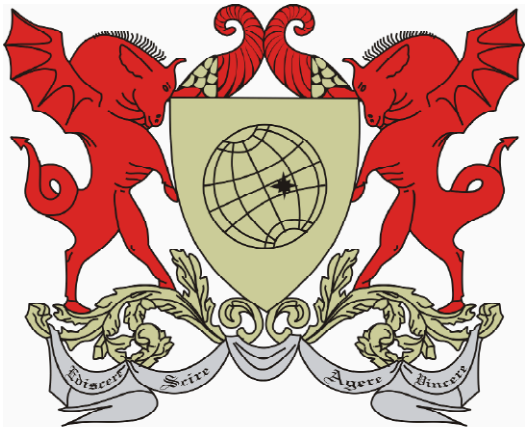
Módulo III



Universidade Federal de Viçosa

cead

Coordenadoria de Educação Aberta e à Distância



Universidade Federal de Viçosa

Nilda de Fátima Ferreira Soares - Reitora

Demétrius David da Silva - Vice-Reitor

CEAD - Coodenadoria de Educação Aberta e a Distância

Frederico Vieira Passos - Diretor

VAREJÃO-JR, C.G; COSTA,M.H. - Curso de Linux - Módulo I. Viçosa, 2010.

Layout: José Timóteo Júnior

Edição de imagens e capa: José Timóteo Júnior, Sávio Augusto Lopes

Editoração Eletrônica: Diogo Rodrigues, Hamilton Henrique Teixeira Reis; Rômulo Siqueira Santos.

Revisão: Diogo Rodrigues

Revisão Final: João Batista Mota

CEAD - Prédio CEE, Avenida PH Rolfs s/n
Campus Universitário, 36570-000, Viçosa/MG
Telefone: (31) 3899 2858 | Fax: (31) 3899 3352

Sumário

Curso Básico de Linux
Módulo III

Curso Básico de Linux
Módulo III

Grupo de Pesquisas Prof. Marcos Heil Costa

Capítulo 1 - Comandos para manipulação de contas

Este capítulo traz comandos usados para manipulação de contas de usuários e grupos em sistemas GNU/Linux. Entre os assuntos descritos aqui, estão: adicionar usuários ao sistema, adicionar grupos e incluir usuários existentes em novos grupos.

1.1 adduser

Adiciona um usuário ou grupo no sistema. Por padrão, quando um novo usuário é adicionado, é criado um grupo com o mesmo nome do usuário. Opcionalmente, o adduser também pode ser usado para adicionar um usuário a um grupo, como será visto adiante. Será criado um diretório *home* com o nome do usuário (a não ser que o novo usuário criado seja um usuário do sistema) e este receberá uma identificação. A identificação do usuário (UID) escolhida será a primeira disponível no sistema, especificada de acordo com a faixa de UIDs de usuários permitidas no arquivo de configuração /etc/adduser.conf. Esse é o arquivo que contém os padrões para a criação de novos usuários no sistema.

adduser [opções] [usuário/grupo]

Onde:

usuário/grupo

Nome do novo usuário que será adicionado ao sistema.

opções

-disable-passwd

Não executa o programa passwd para escolher a senha e somente permite o uso da conta após o usuário escolher uma senha.

--force-badname

Desativa a checagem de senhas ruins durante a adição do novo usuário. Por padrão, o adduser checa se a senha pode ser facilmente adivinhada.

--group

Cria um novo grupo ao invés de um novo usuário. A criação de grupos também pode ser feita pelo comando addgroup.

-uid [num]

Cria um novo usuário com a identificação [num] ao invés de procurar o próximo UID disponível.

-gid [num]

Faz com que o usuário seja parte do grupo [gid] ao invés de pertencer a um novo grupo que será criado com seu nome. Isso é útil caso deseje permitir que grupos de usuários possam ter acesso a arquivos comuns.

Caso estiver criando um novo grupo com adduser, a identificação do novo grupo será [num].

--home [dir]

Usa o diretório [dir] para a criação do diretório home do usuário, ao invés de usar o especificado no arquivo de configuração /etc/adduser.conf.

--ingroup [nome]

Quando adicionar um novo usuário no sistema, coloca o usuário no grupo [nome] ao invés de criar um novo grupo.

--quiet

Não mostra mensagens durante a operação.

--system

Cria um usuário de sistema ao invés de um usuário normal.

Os dados do usuário são colocados no arquivo /etc/passwd após sua criação. Os dados do grupo são colocados no arquivo /etc/group.

OBSERVAÇÃO: Caso esteja usando senhas ocultas (shadow passwords), as senhas dos usuários serão colocadas no arquivo /etc/shadow e as senhas dos grupos no arquivo /etc/gshadow. Isso aumenta a segurança do sistema, pois somente o usuário root pode ter acesso a esses arquivos, ao contrário do arquivo /etc/passwd, que possui os dados de usuários e devem ser lido por todos.

1.2 addgroup

Adiciona um novo grupo de usuários no sistema. As opções usadas são as mesmas do adduser.

addgroup [usuário/grupo] [opções]

1.3 passwd

Muda a senha do usuário ou grupo. Um usuário somente pode alterar a senha de sua conta, mas o superusuário (root) pode alterar a senha de qualquer conta de usuário, inclusive a data de validade da conta, etc. Os donos de grupos também podem alterar a senha do grupo com esse comando.

Os dados da conta do usuário, como nome, endereço e telefone, também podem ser alterados com este comando.

passwd [usuário/grupo] [opções]

Onde:

usuário

Nome do usuário/grupo que terá sua senha alterada.

opções

-g

Se especificada, a senha do grupo será alterada. Somente o root ou o administrador do grupo pode alterar sua senha. A opção -r pode ser usada com esta para remover a senha do grupo. A opção -R pode ser usada para restringir o acesso do grupo para outros usuários.

-x [dias]

Especifica o número máximo de dias que a senha poderá ser usada. Após o término do prazo, a senha deverá ser modificada.

-i

Desativa a conta caso o usuário não tenha alterado sua senha após o tempo especificado por -x.

-n [dias]

Especifica o número mínimo de dias para a senha ser alterada. O usuário não poderá mudar sua senha até que [dias] sejam atingidos desde a última alteração de senha.

-w [num]

Número de dias antecedentes que o usuário receberá o alerta para mudar sua senha. O alerta ocorre [num] dias antes do limite da opção -x, avisando ao usuário quantos dias restam para a troca de sua senha.

-l [nome]

Bloqueia a conta do usuário [nome]. Deve ser usada pelo root. A conta é bloqueada acrescentando um caractere à senha, para que não confira com a senha original.

-u [nome]

Desbloqueia a conta de um usuário bloqueada com a opção -l.

-S [nome]

Mostra o status da conta do usuário [nome]. A primeira parte é o nome do usuário seguido de L (conta bloqueada), NP (sem senha), ou P (com senha). A terceira parte é a data da última modificação da senha. A quarta parte é a período mínimo, máximo, alerta e o período de inatividade para a senha.

Procure sempre combinar letras maiúsculas, minúsculas e números ao escolher suas senhas. Não é recomendado escolher palavras normais para sevir de senha, pois elas são vulneráveis a ataques de dicionários cracker. Outra recomendação é utilizar senhas ocultas em seu sistema (shadow password).

Você deve ser o dono da conta para poder modificar a senha. O usuário root pode modificar/apagar a senha de qualquer usuário.

EXEMPLO:

Tente mudar sua senha! Digite passwd [Seu Nome].

1.4 newgrp

Altera a identificação de grupo do usuário. Para retornar a identificação anterior, digite exit e tecle <ENTER>. Para executar um comando com outra identificação de grupo de usuário, use o comando sg.

newgrp - [grupo]

Onde:

-

Se usado, inicia um novo ambiente após o uso do comando newgrp (semelhante a um novo login no sistema). Caso contrário, o ambiente atual do usuário é mantido.

grupo

Nome do grupo ou número do grupo que será incluído.

Quando este comando é usado, é pedida a senha do grupo que se deseja acessar. Caso a senha do grupo esteja incorreta ou não exista senha definida, a execução do comando é negada. A listagem dos grupos que pertence atualmente pode ser feita usando o comando id.

1.5 userdel

Apaga um usuário do sistema. Quando este comando é usado, ele apaga todos os dados da conta especificada e dos arquivos de contas do sistema.

userdel [-r] [usuário]

Onde:

-r

Apaga também o diretório HOME do usuário.

OBS: Note que uma conta de usuário não poderá ser removida caso ele estiver no sistema, pois os programas podem precisar ter acesso aos dados dele (como UID, GID) no /etc/passwd.

1.6 groupdel

Apaga um grupo do sistema. Quando usado, este comando apaga todos os dados do grupo especificado dos arquivos de contas do sistema.

groupdel [grupo]

Através do comando find, você pode ter certeza se não existem arquivos/diretórios criados com o grupo apagado.

OBS: Você não pode remover o grupo primário de um usuário. Remova o usuário primeiro.

1.7 lastlog

Mostra o último login dos usuários cadastrados no sistema.
É mostrado o nome usado no login, o terminal onde ocorreu a conexão e a hora da última conexão. Esses dados são obtidos através da pesquisa e formatação do arquivo /var/log/lastlog. Caso o usuário não tenha feito login, é mostrada a mensagem ** Never logged in **

lastlog [opções]

Onde:

opções

-t [dias]

Mostra somente os usuários que se conectaram ao sistema nos últimos [dias].

-u [nome]

Mostra somente detalhes sobre o usuário [nome].

A opção -t substitui a opção -u caso sejam usadas simultaneamente.

1.8 last

- ***Mostra uma listagem de entrada e saída de usuários no sistema.*** São mostrados os seguintes campos na lisNome do usuário
- Terminal onde ocorreu a conexão/desconexão
- O hostname (caso a conexão tenha ocorrido remotamente) ou console (caso tenha ocorrido localmente).
- A data e a hora do login/logout se estiver fora do sistema/“still logged in” se ainda estiver usando o sistema
- Tempo (em Horas:Minutos) que esteve conectado ao sistema.

A listagem é mostrada em ordem inversa, ou seja, da data mais atual para a mais antiga. A listagem feita pelo last é obtida de /var/log/wtmp.

last [opções]

Onde:

opções

-n [num]

Mostra [num] linhas. Caso não seja usada, todas as linhas são mostradas.

-R

Não mostra o campo HostName.

-a

Mostra o hostname na última coluna. Será muito útil se combinada com a opção -d.

-d

Usa o DNS para resolver o IP de sistemas remotos para nomes DNS.

-x

Mostra as entradas de desligamento do sistema e alterações do nível de execução do sistema.

O comando last pode ser seguido de um argumento, que será pesquisado como uma expressão regular durante a listagem.

O comando last usa o arquivo /var/log/wtmp para gerar sua listagem, mas alguns sistemas podem não possuir esse arquivo. O arquivo /var/log/wtmp somente é usado caso ele exista. Você pode criá-lo com o comando echo -n >/var/log/wtmp ou touch /var/log/wtmp.

- last - Mostra a listagem geral
- last -a - Mostra a listagem geral, incluindo o nome da máquina
- last gleydson - Mostra somente atividades do usuário gleydson
- last reboot - Mostra as reinicializações do sistema
- last tty1 - Mostra todas as atividades no tty1

1.9 sg

Executa um comando com outra identificação de grupo. A identificação do grupo de usuário é modificada somente durante a execução do comando. Para alterar a identificação de grupo durante sua seção shell, use o comando newgrp.

sg [-] [grupo] [comando]

Onde:

-

Se usado, inicia um novo ambiente durante o uso do comando (semelhante a um novo login e execução do comando). Caso contrário, o ambiente atual do usuário é mantido.

grupo

Nome do grupo com o qual o comando será executado.

comando

Comando que será executado. O comando será executado pelo bash.

Quando esse comando é usado, é pedida a senha do grupo que deseja acessar. Caso a senha do grupo esteja incorreta ou não exista, a execução do comando é negada.

Exemplo: sg root ls /root

1.10 Adicionando o usuário a um grupo extra

Para adicionar um usuário em um novo grupo e assim permitir que ele acesse os arquivos/diretórios que pertencem àquele grupo, você deve estar como root e editar o arquivo /etc/group com o comando vigr. Esse arquivo possui o seguinte formato:

NomedoGrupo:senha:GID:usuários

Onde:

NomedoGrupo

É o nome daquele grupo de usuários.

Senha

Senha para ter acesso ao grupo. Caso esteja utilizando senhas ocultas para grupos, as senhas estarão em /etc/gshadow.

GID

Identificação numérica do grupo de usuário.

usuarios

Lista de usuários que também fazem parte daquele grupo. Caso exista mais de um nome de usuário, eles devem estar separados por vírgula.

Desse modo, para acrescentar o usuário *joao* ao grupo audio, a fim de ter acesso aos dispositivos de som do Linux, acrescente o nome no final da linha: *audio:x:100:joao*. Pronto, basta digitar *logout* e entrar novamente com seu nome e senha. Você estará fazendo parte do grupo audio (confira digitando groups ou id).

Outros nomes de usuários podem ser acrescentados ao grupo audio, bastando separar os nomes com vírgula. Para adicionar automaticamente um usuário a um grupo, você também pode usar o comando adduser da seguinte forma:

adduser joao audio

Isso adicionará o usuário joao ao grupo audio, da mesma forma que fazendo-se a edição manualmente.

1.11 chfn

Muda os dados usados pelo comando finger, que será visto mais adiante.

chfn [usuário] [opções]

Onde:

usuário

Nome do usuário.

opções

-f [nome]

Muda o nome completo do usuário.

-r [nome]

Muda o número da sala do usuário.

-w [tel]

Muda o telefone de trabalho do usuário.

-h [tel]

Muda o telefone residencial do usuário.

-o [outros]

Muda outros dados do usuário.

Caso o nome que acompanha as opções (como o nome completo) contenha espaços, use “” para identificá-lo.

Exemplo: chfn -f “Nome do Usuário root” root

1.12 id

Mostra a identificação atual do usuário, grupo primário e outros grupos aos quais pertence.

id [opções] [usuário]

Onde:

usuário

É o usuário do qual desejamos ver a identificação, grupos primários e complementares.

opções

-g, --group

Mostra somente a identificação do grupo primário.

-G, --groups

Mostra a identificação de outros grupos aos quais pertence.

-n, --name

Mostra o nome do usuário e grupo ao invés da identificação numérica.

-u, --user

Mostra somente a identificação do usuário (user id).

-r, --real

Mostra a identificação real de usuário e grupo, ao invés da efetiva. Esta opção deve ser usada junto com uma das opções: -u, -g, ou -G.

Caso não sejam especificadas opções, id mostrará todos os dados do usuário.

EXEMPLO:

Digite no terminal *id*

id –user

id -r -u.

1.13 logname

Mostra seu login (username).

logname

EXEMPLO:

Digite no seu terminal longname.

1.14 users

Mostra os nomes de usuários usando atualmente o sistema.
Os nomes de usuários são mostrados através de espaços sem detalhes adicionais. Para ver maiores detalhes sobre os usuários, utilize os comandos id e who.

users

Os nomes de usuários atualmente conectados ao sistema são obtidos do arquivo /var/log/wtmp.

1.15 groups

Mostra os grupos a que o usuário pertence.

groups [*usuário*]
Exemplo: groups, groups root tagem:

Capítulo 2 - Redirecionamentos e Pipe

Este capítulo explica o funcionamento dos recursos de direcionamento de entrada e saída do sistema GNU/Linux.

2.1 >

Redireciona a saída de um programa/comando/script para algum dispositivo ou arquivo, ao invés do dispositivo de saída padrão (tela). Quando é usado com arquivos, esse redirecionamento cria ou substitui o conteúdo do arquivo.

Por exemplo, você pode usar o comando ls para listar arquivos e usar ls >listagem para enviar a saída do comando para o arquivo listagem. Use o comando *cat* para visualizar o conteúdo do arquivo listagem.

O mesmo comando pode ser redirecionado para o segundo console / dev/tty2 usando: ls >/dev/tty2. O resultado do comando ls será mostrado no segundo console (pressione <ALT+F2> para mudar para o segundo console e <ALT+F1> para retornar ao primeiro).

2.2 >>

Redireciona a saída de um programa/comando/script para algum dispositivo ou final de arquivo ao invés do dispositivo de saída padrão (tela). A diferença entre esse redirecionamento duplo e o simples, é se caso for usado com arquivos, adiciona a saída do comando ao final do arquivo existente ao invés de substituir seu conteúdo.

Por exemplo, você pode acrescentar a saída do comando ls ao arquivo

listagem do capítulo anterior usando ls / >>listagem. Use o comando cat para visualizar o conteúdo do arquivo listagem.

2.3 <

Direciona a entrada padrão de arquivo/dispositivo para um comando. Esse comando faz o contrário do anterior: envia dados ao comando.

Você pode usar o comando cat <teste.txt para enviar o conteúdo do arquivo teste.txt ao comando cat, que mostrará seu conteúdo (é claro que o mesmo resultado pode ser obtido com cat teste.txt, mas esse exemplo serviu para mostrar a funcionalidade do <).

2.4 <<

Este redirecionamento serve principalmente para marcar o fim de exibição de um bloco. Ele é especialmente usado em conjunto com o comando cat, mas também tem outras aplicações. Por exemplo:

```
cat << final
este arquivo
será mostrado
até que a palavra final seja
localizada no inicio da linha
final
```

2.5 | (pipe)

Envia a saída de um comando para a entrada do próximo comando para continuidade do processamento. Os dados enviados são processados pelo próximo comando que mostrará o resultado do processamento.

EXEMPLO:

ls -la|more - este comando faz a listagem longa de arquivos que depois é enviado ao comando more (ele tem a função de efetuar uma pausa a cada 25 linhas do arquivo).

Outro exemplo é o comando locate find|grep bin/. Nele, todos os caminhos/arquivos que contém *find* na listagem serão mostrados (inclusive man pages, bibliotecas, etc.), então enviamos a saída desse comando para grep bin/ a fim de mostrar somente os diretórios que contém binários. Mesmo que a listagem ocupe mais de uma tela, podemos acrescentar o more: locate find|grep bin/|more.

Podem ser usados mais de um comando de redirecionamento (<, >, |) em um mesmo comando.

2.6 Diferença entre o "|" e o ">"

A principal diferença entre o | e o >, é que o Pipe envolve processamento entre comandos, ou seja, a saída de um comando é enviado a entrada do próximo e o > redireciona a saída de um comando para um arquivo/ dispositivo.

Você pode notar pelo exemplo acima (ls -la|more) que *ls* e *more* são comandos porque estão separados por um |! Se um deles não existir ou estiver digitado incorretamente, será mostrada uma mensagem de erro.

Um resultado diferente seria obtido usando um > no lugar do |; A saída do comando ls -la seria gravada num arquivo chamado more.

2.7 tee

Envia o resultado do programa para a saída padrão (tela) e para um arquivo ao mesmo tempo. Este comando deve ser usado com o pipe.

comando|tee [arquivo]

Exemplo: ls -la|tee listagem.txt, a saída do comando será mostrada normalmente na tela e ao mesmo tempo gravada no arquivo listagem.txt.

Capítulo 3 - Comandos Básicos de Rede

Este capítulo traz alguns comandos úteis para uso em rede e ambientes multiusuário.

3.1 who

Mostra quem está atualmente conectado ao computador. Esse comando lista os nomes de usuários que estão conectados ao seu computador, o terminal e data da conexão.

who [opções]

Onde:

opções

-H, --heading

Mostra o cabeçalho das colunas.

-i, -u, --idle

Mostra a quanto tempo o usuário está parado em Horas:Minutos.

-m, i am

Mostra o nome do computador e usuário associado ao nome. É equivalente a digitar who i am ou who am i.

-q, --count

Mostra o total de usuários conectados aos terminais.

-T, -w, --mesg

Mostra se o usuário pode receber mensagens via talk (conversação).

+ O usuário recebe mensagens via talk

- O usuário não recebe mensagens via talk.

? Não foi possível determinar o dispositivo de terminal onde o usuário está conectado.

3.2 w

Mostra quem está conectado no sistema e o que cada um está fazendo.

w [*opções*][*usuário*]

Onde:

usuário

Nome do usuário do qual se deseja ver os detalhes. Se o usuário não for digitado, o comando w mostra detalhes de todos os usuários conectados no sistema.

opções

-h

Não mostra o cabeçalho

-u

Ignora os nomes de usuários enquanto verifica os processo atuais e tempos de CPU.

-f

Mostra ou oculta o campo FROM na listagem.

3.3 whoami

Mostra o nome que usou para se conectar ao sistema. É útil quando você usa várias contas e não sabe com qual nome entrou no sistema :-)

whoami

3.4 hostname

Mostra ou muda o nome de seu computador na rede.

3.5 ftp

Permite a transferência de arquivos de um computador remoto para o local e vice-versa. O File Transfer Protocol é o sistema de transmissão de arquivos mais usado na Internet. É requerida a autenticação do usuário para que seja permitida a conexão. Muitos servidores ftp disponibilizam acesso anônimo aos usuários (restrito).

Uma vez conectado a um servidor ftp, você pode usar a maioria dos comandos do GNU/Linux para operá-lo.

ftp [ip/dns]

Segue uma lista com alguns dos comandos mais usados no FTP:

ls

Lista arquivos do diretório atual.

cd [diretório]

Entra em um diretório.

get [arquivo]

Copia um arquivo do servidor ftp para o computador local. O arquivo é gravado, por padrão, no diretório onde o programa ftp foi executado.

hash [on/off]

Por padrão esta opção está desligada. Quando ligada, faz com que o caractere # seja impresso na tela, indicando o progresso do download.

mget [arquivos]

Semelhante ao get, mas pode copiar diversos arquivos e permite o uso de coringas.

send [arquivo]

Envia um arquivo para o diretório atual do servidor FTP (você precisa de uma conta com acesso a gravação para fazê-lo).

prompt [on/off]

Ativa ou desativa a pergunta para a cópia de arquivo. Se estiver como off assume sim para qualquer pergunta.

Exemplo: ftp ftp.br.debian.org.

3.6 ping

Verifica se um computador está disponível na rede. Este comando é muito utilizado por alguns programas de conexão e administradores para verificar se uma determinada máquina está conectada à rede e também para verificar o tempo de resposta de cada máquina da rede. O ping envia pacotes ICMS ECHO_REQUEST para um computador. Este, quando recebe o pacote, envia uma resposta ao endereço de origem, avisando que está disponível na rede.

ping [opções][IP/DNS]

Onde:

IP/dns

Endereço IP ou nome DNS do endereço.

opções

-c [num]

Envia [num] pacotes ao computador de destino.

-f

Flood ping. Envia novos pacotes antes de receber a resposta do pacote anterior. Para cada requisição enviada, um . é mostrado na tela. Para cada resposta recebida, um backspace é mostrado. Somente o usuário root pode utilizar essa opção, que auxilia bastante na detecção de erros de transmissão de pacotes em interfaces das máquinas em sua rede.

-i [seg]

Aguarda [seg] segundos antes de enviar cada pacote.

-q

Não mostra as requisições enquanto são enviadas, somente mostra as linhas de sumário no início e término do programa.

-s [tamanho]

Especifica o tamanho do pacote que será enviado.

-v, --verbose

Saída detalhada: tanto os pacotes enviados como recebidos são listados.

Exemplo: ping 192.168.1.1, ping www.br.debian.org.

3.7 ssh

Esta é a ferramenta usada para seções de console remotos, isto é, acessar computadores a distância de forma segura. Para conectar a um servidor ssh remoto:

ssh usuario@endereço

Se o nome do usuário for omitido, seu login atual do sistema será usado. O uso da opção *-C* é recomendado para ativar o modo de compactação dos dados (útil em conexões lentas). A opção *-l usuário* pode ser usada para alterar a identificação de usuário (quando não é usada, o login local é usado como nome de usuário remoto). Uma porta alternativa pode ser especificada usando a opção *-p porta* (a 22 é usada por padrão).

Na primeira conexão, a chave pública do servidor remoto será gravada em ~/.ssh/know_hosts ou ~/.ssh/know_hosts2 (dependendo da versão do servidor ssh remoto), e verificada a cada conexão como checagem de segurança para se certificar que o servidor não foi alvo de qualquer ataque ou modificação não autorizada das chaves. Por padrão, o cliente utilizará o protocolo ssh versão 1. A opção *-2* permite usar o protocolo versão 2.

EXEMPLOS:

 # Conecta-se ao servidor remoto usando o login do usuário atual. Por exemplo, suponhamos que você esteja num computador identificado como jose@computador.org e deseje acessar o computador jose@computador2.org

ssh computador2.org

 # Conecta-se ao servidor remoto usando outro login. Por exemplo, suponhamos que você esteja num computador identificado como jose@computador.org e deseje acessar o computador maria@computador.org

ssh maria@computador.org

 # Conecta-se ao servidor remoto usando compactação e o login john
ssh computador.org -C -l john

3.1 Redirecionamento de conexões do X

O redirecionamento de conexões do X Window poderá ser habilitado em ~/.ssh/config ou /etc/ssh/ssh_config ou usando as opções *-A -X* na linha de comando do ssh (as opções *-a* e *-x* desativam as opções acima respectivamente). Uma variável *$DISPLAY* é criada automaticamente, para fazer o redirecionamento ao servidor X local.

Ao executar um aplicativo remoto, a conexão é redirecionada a um DISPLAY proxy criado pelo ssh (a partir de :10, por padrão) que faz a conexão com o display real do X, ou seja, ele pulará os métodos de autenticação xhost e cookies. Por medidas de segurança é recomendável habilitar o redirecionamento individualmente somente se você confia no administrador do sistema remoto.

3.8 scp

Permite a cópia de arquivos entre o cliente/servidor ssh. A sintaxe usada por esse comando é a seguinte:

scp [*origem*] [*destino*]

- Os parâmetros de *origem* e *destino* são semelhantes ao do comando cp, mas possuem um formato especial quando é especificada uma máquina rem. Um caminho padrão - Quando for especificado um arquivo local. Por exemplo: /usuario/teste/arquivo.tar.gz.
- usuario@host_remoto:/diretório/arquivo - Quando desejar copiar o arquivo de/para um servidor remoto usando sua conta de usuário. Por exemplo: gleydson@ftp.debian.org:~/arqs.

A opção *-C* é recomendável para aumentar a taxa de transferência de dados usando compactação. Caso a porta remota do servidor sshd seja diferente de 22, a opção *-P porta* deverá ser especificada (é “P” maiúscula mesmo, pois a *-p* é usada para preservar permissões/data/horas dos arquivos transferidos).

EXEMPLOS:

 # Para copiar um arquivo local chamado home/jose/teste/teste4.ncl para o diretório pessoal de maria@computador.org
scp -C home/jose/teste/teste4.ncl maria@computador.org:./

 # Para fazer a operação inversa a acima, ou seja, copiando do servidor remoto para o local onde estou é só inverter os parâmetros origem/destino:

scp -C maria@computador.org:/home/maria/teste4.ncl /home/jose/teste/

 # O exemplo abaixo faz a transferência de arquivos entre 2 computadores remotos:
O arquivo teste4.ncl é lido do servidor computador@org e copiado para computador2.org ambos usando o login jose
scp -C jose@computador.org:/home/jose/teste.sh jose@computador2.org:/home/jose/

Capítulo 4 - Compactadores

Esta seção explica o que são e como usar programas compactadores no GNU/Linux, as características de cada um, como identificar um arquivo compactado e como descompactar um arquivo compactado usando o programa correspondente.

A utilização de arquivos compactados é método útil principalmente para reduzir o consumo de espaço em disco ou permitir grandes quantidades de texto serem transferidas para outro computador através de disquetes.

4.1 O que fazem os compactadores/descompactadores?

Compactadores são programas que diminuem o tamanho de um arquivo (ou arquivos) através da substituição de caracteres repetidos. Para entender melhor como eles funcionam, veja o próximo exemplo:

compactadores compactam e deixam arquivos compactados

Após a compactação da frase:

%dores %m e deixam arquivos %dos

O que aconteceu realmente foi que a palavra compacta se encontrava 3 vezes na frase acima, e foi substituída por um sinal de %. Para descompactar o processo seria o contrário: Ele substituiria % por compacta e nós teríamos a frase novamente restaurada.

Você deve ter notado que o tamanho da frase compactada caiu quase pela metade. A quantidade de compactação de um arquivo é chamada de taxa de compactação. Assim, se o tamanho do arquivo for diminuído à metade após a compactação, dizemos que conseguiu uma taxa de compactação de 2:1 (lê-se dois para um), se o arquivo diminuiu em quatro vezes, dizemos que conseguiu uma compactação de 4:1 (quatro para um), e assim sucessivamente.

Para controle dos caracteres que são usados nas substituições, os programas de compactação mantém cabeçalhos com todas as substituições usadas durante a compactação. O tamanho do cabeçalho pode ser fixo ou definido pelo usuário - depende do programa usado na compactação.

Este é um exemplo bem simples para entender o que acontece durante a compactação: os compactadores executam instruções muito avançadas e códigos complexos para atingir uma alta taxa de compactação.

- Observações: não é possível trabalhar diretamente com arquivos compactados! É necessário descompactar o arquivo para usá-lo. Note que alguns programas atualmente suportam a abertura de arquivos compactados, mas na realidade eles apenas simplificam a tarefa, descompactando o arquivo, abrindo e o recompactando assim que o trabalho estiver concluído.

- Arquivos de texto têm uma taxa de compactação muito melhor que arquivos binários, porque possuem mais caracteres repetidos. É normal atingir taxas de compactação de 10/1 ou superiores quando se compacta um arquivo de texto. Arquivos binários, como programas, possuem uma taxa de compactação média de 2:1.

- Note que também existem programas compactadores especialmente desenvolvidos para compactação de músicas, arquivos binários, imagens, textos, etc.

4.1.1 Tipos de compactação

Existem basicamente dois tipos de compactação: a compactação *sem perdas* e a compactação *com perdas*.

Os exemplos a seguir tentam explicar de forma simples os conceitos envolvidos.

A compactação sem perdas, como o próprio nome diz, não causa nenhuma perda nas informações contidas no arquivo. Quando você compacta e descompacta um arquivo, o conteúdo é o mesmo do original.

A compactação com perdas é um tipo específico de compactação desenvolvido para atingir altas taxas, porém com perdas parciais dos dados. É aplicada a tipos de arquivos especiais, como músicas e imagens ou arquivos que envolvam a percepção humana.

Sabe-se que o ouvido humano não é tão sensível a determinados sons e frequências. Portanto, a compactação de um arquivo de música poderia deixar de gravar os sons que seriam pouco percebidos, resultando num arquivo menor. Uma compactação do tipo ogg ou mp3 utiliza-se destes recursos. O arquivo resultante é muito menor que o original, porém alguns dados sonoros são perdidos. Você só os nota se estiver reproduzindo a música num equipamento de alta qualidade e se tiver um ouvido bem aguçado. Para efeitos práticos, você está ouvindo a mesma música e economizando muito espaço em disco.

Outro exemplo de compactação com perdas são as imagens jpg. Imagine que você tem uma imagem com 60000 tons de cor diferentes, mas alguns tons são muito próximos de outros. Então, o compactador resume para 20000 tons de cor e a imagem terá 1/3 do tamanho original. O nosso olho conseguirá entender a imagem sem problemas e quase não perceberá a diferença. Exemplos de extensões utilizadas em imagens compactadas são jpg, png, gif.

Apesar das vantagens da grande taxa de compactação conseguida nos processos com perdas, nem sempre podemos utilizá-lo. Quando compactamos um texto ou um programa, não podemos ter perdas, senão o nosso texto sofre alterações ou o programa não consegue ser executado. Nem mesmo podemos tem perdas quando compactamos imagens ou músicas que serão utilizadas em processos posteriores de masterização, mixagem ou impressão em alta qualidade.

4.2 Extensões de arquivos compactados

- As extensões identificam o tipo de um arquivo e assim o programa necessário para trabalhar com ele. Existem dezenas de extensões que identificam arquivos compactados. Quando um arquivo (s) é compactado, uma extensão correspondente ao programa usado é adicionada ao nome do arquivo (caso o arquivo seja compactado pelo gzip receberá a extensão .gz, por exemplo). Ao descompactar acontece o contrário: a extensão é retirada do arquivo. Abaixo segue uma listagem de extensões mais usadas e os programas correspondentes:.gz - Arquivo compactado pelo gzip. Use o programa gzip para descompactá-lo.
- .bz2 - Arquivo compactado pelo bzip2. Use o programa bzip2 para descompactá-lo.
- .Z - Arquivo compactado pelo programa compress. Use o programa uncompress para descompactá-lo.
- .zip - Arquivo compactado pelo programa zip. Use o programa unzip para descompactá-lo.
- .rar - Arquivo compactado pelo programa rar. Use o programa unrar para descompactá-lo.
- .tar.gz - Arquivo compactado pelo programa gzip no utilitário de arquivamento tar. Para descompactá-lo, você pode usar o gzip e depois o tar ou somente o programa tar usando a opção -z.
- .tgz - Abreviação de .tar.gz.
- .tar.bz2 - Arquivo compactado pelo programa bzip2 no utilitário de arquivamento tar. Para descompactá-lo, você pode usar o bzip2 e depois o tar ou somente o programa tar usando a opção -j.
- .tar.Z - Arquivo compactado pelo programa compress no utilitário de arquivamento tar. Para descompactá-lo, você pode usar o uncompress e depois o tar ou somente o programa tar usando a opção -Z.

4.3 gzip

É praticamente o compactador padrão do GNU/Linux, possui uma ótima taxa de compactação e velocidade. A extensão dos arquivos compactados pelo gzip é a .gz, na versão para DOS, Windows NT é usada a extensão .z.

gzip [opções] [arquivos]

Onde:

arquivos

Especifica quais arquivos serão compactados pelo gzip. Caso seja usado um -, será assumida a entrada padrão. Coringas podem ser usados para especificar vários arquivos de uma só vez.

opções

-d, --decompress [arquivo]

Descompacta um arquivo.

-f

Força a compactação, compactando até mesmo links.

-l [arquivo]

Lista o conteúdo de um arquivo compactado pelo gzip.

-r

Compacta diretórios e sub-diretórios.

-c [arquivo]

Descompacta o arquivo para a saída padrão.

-t [arquivo]

Testa o arquivo compactado pelo gzip.

-[num], --fast, --best

Ajustam a taxa de compactação/velocidade da compactação. Quanto melhor a taxa, menor é a velocidade de compactação e vice-versa. A opção --fast permite uma compactação rápida e tamanho do arquivo maior. A opção --best permite uma melhor compactação a uma velocidade menor.

O uso da opção -[número] permite especificar uma compactação individualmente, usando números entre 1 (menor compactação) e 9 (melhor compactação). É útil para buscar um bom equilíbrio entre taxa de compactação/velocidade (especialmente em computadores muito lentos).

Quando um arquivo é compactado pelo gzip, é automaticamente acrescentada a extensão .gz ao seu nome.

O gzip também reconhece arquivos compactados pelos programas zip, compress, compress -H e pack. As permissões de acesso dos arquivos são também armazenadas no arquivo compactado.

- Exemplos: gzip -9 texto.txt - Compacta o arquivo texto.txt usando a compactação máxima (compare o tamanho do arquivo compactado usando o comando ls -la).
- gzip -d texto.txt.gz - Descompacta o arquivo texto.txt
- gzip -c texto.txt.gz - Descompacta o arquivo texto.txt para a tela
- gzip -9 *.txt - Compacta todos os arquivos que terminam com .txt
- gzip -t texto.txt.gz - Verifica o arquivo texto.txt.gz.

4.4 zip

Utilitário de compactação compatível com pkzip (do DOS) e trabalha com arquivos de extensão .zip. Possui uma ótima taxa de compactação e velocidade no processamento dos arquivos compactados (comparando-se ao gzip).

zip [*opções*] [*arquivo-destino*] [*arquivos-origem*]

Onde:

arquivo-destino

Nome do arquivo compactado que será gerado.

arquivos-origem

Arquivos/diretórios que serão compactados. Podem ser usados coringas para especificar mais de um arquivo de uma só vez.

opções

-r

Compacta arquivos e sub-diretórios.

-e

Permite encriptar o conteúdo de um arquivo .zip através de senha. A senha será pedida no momento da compactação.

-f

Somente substitui um arquivo compactado existente dentro do arquivo .zip se a versão é mais nova que a atual. Não acrescenta arquivos ao arquivo compactado. Deve ser executado no mesmo diretório em que o programa zip foi executado anteriormente.

-F

Repara um arquivo .zip danificado.

-[NUM]

Ajusta a qualidade/velocidade da compactação. Pode ser especificado um número de 1 a 9. O 1 permite mínima compactação e máxima velocidade, 9 permite uma melhor compactação e menor velocidade.

-i [arquivos]

Compacta somente os [arquivos] especificados.

-j

Se especificado, não armazena caminhos de diretórios.

-m

Apaga os arquivos originais após a compactação.

-T [arquivo]

Procura por erros em um arquivo .zip. Caso sejam detectados problemas, utilize a opção -F para corrigi-los.

-y

Armazena links simbólicos no arquivo .zip. Por padrão, os links simbólicos são ignorados durante a compactação.

-k [arquivo]

Modifica o [arquivo] para ter compatibilidade total com o pkzip do DOS.

-l

Converte saltos de linha UNIX (<LF>) para o formato <CR+LF> (usados pelo DOS). Use esta opção com arquivos Texto.

-ll

Converte saltos de linha DOS (<CR+LF>) para o formato UNIX (<LF>). Use essa opção com arquivos texto.

-n [extensão]

Não compacta arquivos identificados por [extensão]. Ele é armazenado sem compactação no arquivo .zip, muito útil para uso com arquivos já compactados.

Caso sejam especificadas diversas extensões de arquivos, elas devem ser separadas por:

Por exemplo, zip -n .zip:.tgz arquivo.zip *.txt.

-q

Não mostra mensagens durante a compactação do arquivo.

-u

Atualiza/adiciona arquivos ao arquivo .zip

-X

Não armazena detalhes de permissões, UID, GID e datas dos arquivos.

-z

Permite incluir um comentário no arquivo .zip.

- Caso o nome de arquivo de destino não termine com .zip, esta extensão será automaticamente adicionada. Para a descompactação de arquivos .zip no GNU/Linux, é necessário o uso do utilitário unzip. Exemplos: zip textos.zip *.txt - Compacta todos os arquivos com a extensão .txt para o arquivo textos.zip (compare o tamanho do arquivo compactado digitando ls -la).
- zip -r textos.zip /usr/*.txt - Compacta todos os arquivos com a extensão .txt do diretório /usr e sub-diretórios para o arquivo textos.zip.
- zip -9 textos.zip * - Compacta todos os arquivos do diretório atual usando a compactação máxima para o arquivo textos.zip.
- zip -T textos.zip - Verifica se o arquivo textos.zip contém erros.

4.5 unzip

Descompacta arquivos .zip criados com o programa zip. Este programa também é compatível com arquivos compactados pelo pkzip do DOS.

unzip [opções] [arquivo.zip] [arquivos-extrair] [-d diretório]

Onde:

arquivo.zip

Nome do arquivo que se deseja descompactar. Podem ser usados coringas para especificar mais de um arquivo para ser descompactado.

arquivos-extrair

Nome dos arquivos (separados por espaço) que serão descompactados do arquivo .zip. Caso não seja especificado, é assumido * (todos os arquivos serão descompactados).

Se for usado -x arquivos, os arquivos especificados não serão descompactados. O uso de coringas é permitido.

-d diretório

Diretório onde os arquivos serão descompactados. Caso não for especificado, os arquivos serão descompactados no diretório atual.

opções

-c

Descompacta os arquivos para stdout (saída padrão) ao invés de criar arquivos. Os nomes dos arquivos também são mostrados (veja a opção -p).

-f

Descompacta somente arquivos que existam no disco e mais novos que os atuais.

-l

Lista os arquivos existentes dentro do arquivo .zip.

-M

Efetua uma pausa a cada tela de dados durante o processamento (a mesma função do comando more).

-n

Nunca substitui arquivos já existentes. Se um arquivo existe, ele é pulado.

-o

Substitui arquivos existentes sem perguntar. Tem função contrária à opção -n.

-P [SENHA]

Permite descompactar arquivos .zip usando a [SENHA]. CUIDADO! qualquer usuário conectado em seu sistema pode ver a senha digitada na linha de comando digitada.

-p

Descompacta os arquivos para stdout (saída padrão) ao invés de criar arquivos. Os nomes dos arquivos não são mostrados (veja a opção -c).

-q

Não mostra mensagens.

-t

Verifica o arquivo .zip em busca de erros.

-u

Idêntica à opção -f, só que também cria arquivos que não existem no diretório.

-v

Mostra mais detalhes sobre o processamento do unzip.

-z

Mostra somente o comentário existente no arquivo.

Por padrão, o unzip também descompacta sub-diretórios caso o arquivo .zip tenha sido gerado com zip -r.

- Exemplos: unzip texto.zip - Descompacta o conteúdo do arquivo texto.zip no diretório atual.
- unzip texto.zip carta.txt - Descompacta somente o arquivo carta.txt do arquivo texto.zip.
- unzip texto.zip -d /tmp/texto - Descompacta o conteúdo do arquivo texto.zip para o diretório /tmp/texto.
- unzip -l texto.zip - Lista o conteúdo do arquivo texto.zip.
- unzip -t texto.zip - Verifica o arquivo texto.zip.

4.6 tar

Na verdade o tar não é um compactador, e sim um "arquivador" (ele junta vários arquivos em um só), mas pode ser usado em conjunto com um compactador (como o gzip ou zip) para armazená-los compactados. O tar também é muito usado para cópias de arquivos especiais ou dispositivos do sistema. É comum encontrar arquivos com a extensão .tar, .tar.gz, .tgz, .tar.bz2, .tar.Z, .tgZ. O primeiro é um arquivo normal gerado pelo tar e todos os outros são arquivos gerados através tar junto com um programa de compactação – gzip (.gz), bzip2 (.bz2) e compress (.Z).

tar [opções] [arquivo-destino] [arquivos-origem]

Onde:

arquivo-destino

É o nome do arquivo de destino. Normalmente especificado com a extensão .tar, caso seja usado somente o arquivamento ou .tar.gz/.tgz caso seja usada a compactação (usando a opção -z).

arquivos-origem

Especifica quais arquivos/diretórios serão compactados.

opções

-c, --create

Cria um novo arquivo .tar

-t, --list

Lista o conteúdo de um arquivo .tar

-u, --update

Atualiza arquivos compactados no arquivo .tar

-f, --file [HOST:]F

Usa o arquivo especificado para gravação ou o dispositivo /dev/rmt0.

-j, --bzip2

Usa o programa bzip2 para processar os arquivos do tar

-l, --one-file-system

Não processa arquivos em um sistema de arquivos diferentes de onde o tar foi executado.

-M, --multi-volume

Cria/lista/descompacta arquivos em múltiplos volumes. O uso

de arquivos em múltiplos volumes permite que uma grande cópia de arquivos que não cabe em um disquete, por exemplo, seja feita em mais de um disquete.

-o

Grava o arquivo no formato VT7 ao invés do ANSI.

-O, --to-stdout

Descompacta arquivos para a saída padrão ao invés de gravar em um arquivo.

--remove-files

Apaga os arquivos de origem após serem processados pelo tar.

-R, --record-number

Mostra o número de registros dentro de um arquivo tar em cada mensagem.

--totals

Mostra o total de bytes gravados com a opção --create.

-v

Mostra os nomes dos arquivos enquanto são processados.

-V [NOME]

Inclui um [NOME] no arquivo tar.

-W, --verify

Tenta verificar o arquivo gerado pelo tar após gravá-lo.

x

Extrai arquivos gerados pelo tar

-X [ARQUIVO]

Tenta apagar o [ARQUIVO] dentro de um arquivo compactado .tar.

-Z

Usa o programa compress durante o processamento dos arquivos.

-z

Usa o programa gzip durante o processamento dos arquivos.

--use-compress-program [PROGRAMA]

Usa o [PROGRAMA] durante o processamento dos arquivos. Ele deve aceitar a opção -d.

-[0-7][lmh]

Especifica a unidade e sua densidade.

- A extensão precisa ser especificada no arquivo de destino para a identificação correta: Arquivos gerados pelo tar precisam ter a extensão .tar
- Caso seja usada a opção -j para compactação, a extensão deverá ser .tar.bz2
- Caso seja usada a opção -z para compactação, a extensão deverá ser .tar.gz ou .tgz
- Caso seja usada a opção -Z para a compactação, a extensão deverá ser .tar.Z ou .tgZ

É importante saber qual o tipo de compactador usado durante a geração do arquivo .tar, pois será necessário especificar a opção apropriada para descompactá-lo.

Exemplos:

- tar -cf index.txt.tar index.txt - Cria um arquivo chamado index.txt.tar, que armazenará o arquivo index.txt. Você pode notar, digitando ls -la, que o arquivo index.txt foi somente arquivado (sem compactação). Isso é útil para juntar diversos arquivos num só.
- tar -xf index.txt.tar - Desarquiva o arquivo index.txt criado pelo comando acima.
- tar -czf index.txt.tar.gz index.txt - Idêntico ao exemplo de arquivamento anterior, só que agora é usada a opção -z (compactação através do programa gzip). Agora é possível notar, digitando ls -la, que o arquivo index.txt foi compactado e depois arquivado no arquivo index.txt.tar.gz (você também pode chamá-lo de index.txt.tgz, que também identifica um arquivo .tar compactado pelo gzip)
- tar -xzf index.txt.tar.gz - Descompacta e desarquiva o arquivo index.txt.tar.gz criado com o comando acima.
- gzip -dc index.tar.gz | tar -xf - - Faz o mesmo que o comando acima, só que de uma forma diferente: primeiro descompacta o arquivo index.txt.tar.gz e envia a saída do arquivo descompactado para o tar que desarquivará o arquivo index.txt.
- tar -cjf index.txt.tar.bz2 index.txt - Arquiva o arquivo index.txt em index.txt.tar.bz2 compactando através do bzip2 (opção -j).
- tar -xjf index.txt.tar.bz2 - Descompacta e desarquiva o arquivo index.txt.tar.bz2 criado com o comando acima.
- bzip2 -dc index.txt.tar.bz2 | tar -xf - - Faz o mesmo que o comando acima, só que de uma maneira diferente: primeiro descompacta o arquivo index.txt.tar.bz2 e envia a saída do arquivo descompactado para o tar, que desarquivará o arquivo index.txt.
- tar -t index.txt.tar - Lista o conteúdo de um arquivo .tar.
- tar -tz index.txt.tar.gz - Lista o conteúdo de um arquivo .tar.gz.

4.7 bzip2

É um novo compactador, que vem sendo cada vez mais usado, pois consegue atingir a melhor compactação em arquivos texto se comparado aos já existentes (em consequência, sua velocidade de compactação também é menor: quase duas vezes mais lento que o gzip). Suas opções são praticamente as mesmas usadas no gzip e você também pode usá-lo da mesma forma. A extensão dos arquivos compactados pelo bzip2 é a .bz2

bzip2 [opções] [arquivos]

Onde:

arquivos

Especifica quais arquivos serão compactados pelo bzip2. Caso seja usado um -, será assumida a entrada padrão. Coringas podem ser usados para especificar vários arquivos de uma só vez.

opções

-d, --decompress [arquivo]

Descompacta um arquivo.

-f

Força a compactação, compactando até mesmo links.

-l [arquivo]

Lista o conteúdo de um arquivo compactado pelo bzip2.

-r

Compacta diretórios e sub-diretórios.

-c [arquivo]

Descompacta o arquivo para a saída padrão.

-t [arquivo]

Testa o arquivo compactado pelo bzip2.

-[num], --fast, --best

Ajustam a taxa de compactação (velocidade). Quanto melhor a taxa, menor é a velocidade de compactação e vice-versa. A opção --fast permite uma compactação rápida e tamanho do arquivo maior. A opção --best permite uma melhor compactação a uma velocidade menor.

O uso da opção -[número] permite especificar uma compactação individualmente usando números entre 1 (menor compactação) e 9 (melhor compactação). É útil para buscar um bom equilíbrio entre taxa de compactação/velocidade (especialmente em computadores muito lentos).

Quando um arquivo é compactado pelo bzip2, é automaticamente acrescentada a extensão .bz2 ao seu nome. As permissões de acesso dos arquivos são também armazenadas no arquivo compactado.

- Exemplos: bzip2 -9 texto.txt - Compacta o arquivo texto.txt usando a compactação máxima (compare o tamanho do arquivo compactado usando o comando ls -la).
- bzip2 -d texto.txt.bz2 - Descompacta o arquivo texto.txt
- bzip2 -c texto.txt.bz2 - Descompacta o arquivo texto.txt para a saída padrão (tela)
- bzip2 -9 *.txt - Compacta todos os arquivos que terminam com .txt
- bzip2 -t texto.txt.bz2 - Verifica o arquivo texto.txt.bz2.

4.8 rar

rar é um compactador desenvolvido por Eugene Roshal e possui versões para GNU/Linux, DOS, Windows, OS/2 e Macintosh. Trabalha com arquivos de extensão .rar e permite armazenar arquivos compactados em vários disquetes (múltiplos volumes). Se trata de um produto comercial, mas possui boas versões shareware e pode ser muito útil em algumas situações.

rar [ações] [opções] [arquivo-destino.rar] [arquivos-origem]

Onde:

arquivo-destino.rar

É o nome do arquivo de destino

arquivos-origem

Arquivos que serão compactados. Podem ser usados coringas para especificar mais de um arquivo.

ações

a

Compacta arquivos

x

Descompacta arquivos

d

Apaga arquivos especificados

t

Verifica o arquivo compactado em busca de erros.

c

Inclui comentário no arquivo compactado

r

Repara um arquivo .rar danificado

l

Lista arquivos armazenados no arquivo compactado

u

Atualiza arquivos existentes no arquivo compactado.

m

Compacta e apaga os arquivos de origem (move).

e

Descompacta arquivos para o diretório atual

p

Mostra o conteúdo do arquivo na saída padrão

rr

Adiciona um registro de verificação no arquivo

s

Converte um arquivo .rar normal em arquivo auto-extraível. Arquivos auto-extraíveis são úteis para enviar arquivos a pessoas que não possuem o programa rar. Basta executar o arquivo e ele será automaticamente descompactado (usando o sistema operacional no qual foi criado). Note que essa opção requer que o arquivo default.sfx esteja presente no diretório home do usuário. Use o comando find para localizá-lo em seu sistema.

opções

o+

Substitui arquivos já existentes sem perguntar

o-

Não substitui arquivos existentes

sfx

Cria arquivos auto-extraíveis. Arquivos auto-extraíveis são úteis

para enviar arquivos a pessoas que não têm o programa rar. Basta executar o arquivo e ele será automaticamente descompactado. Note que o processo requer que o arquivo default.sfx esteja presente no diretório home do usuário. Use o comando find para encontrá-lo em seu sistema.

y

Assume sim para todas as perguntas

r

Inclui sub-diretórios no arquivo compactado

x [ARQUIVO]

Processa tudo menos o [ARQUIVO]. Podem ser usados coringas

v[TAMANHO]

Cria arquivos com um limite de tamanho. Por padrão, o tamanho é especificado em bytes, mas o número pode ser seguido de k (kilobytes) ou m (megabytes).

Exemplo: rar a -v1440k ... ou rar a -v10m ...

p [SENHA]

Inclui senha no arquivo. CUIDADO, pessoas conectadas ao seu sistema podem capturar a linha de comando facilmente e descobrir sua senha.

m [0-5]

Ajusta a taxa de compactação/velocidade de compactação. 0 não faz compactação alguma (mais rápido), somente armazena os arquivos; 5 é o nível que usa mais compactação (mais lento).

ed

Não inclui diretórios vazios no arquivo

isnd

Ativa emissão de sons de alerta pelo programa

ierr

Envia mensagens de erro para stderr

inul

Desativa todas as mensagens

ow

Salva o dono e grupo dos arquivos.

ol

Salva links simbólicos no arquivo ao invés do arquivo físico ao qual o link faz referência.

mm[f]

Usa um método especial de compactação para arquivos multimídia (sons, vídeos, etc). Caso for usado mmf, força o uso do método multimídia mesmo que o arquivo compactado não seja desse tipo.

Os arquivos gerados pelo rar do GNU/Linux podem ser usados em outros sistemas operacionais. Para isso, basta ter o rar instalado. Quando é usada a opção -v para a criação de múltiplos volumes, a numeração dos arquivos é feita na forma: arquivo.rar, arquivo.r00, arquivo.r01, etc, durante a descompactação os arquivos serão pedidos em ordem. Se você receber

a mensagem cannot modify volume durante a criação de um arquivo .rar, provavelmente o arquivo já existe. Apague o arquivo existente e tente novamente.

Exemplos:

- rar a texto.rar texto.txt - Compacta o arquivo texto.txt em um arquivo com o nome texto.rar
- rar x texto.rar - Descompacta o arquivo texto.rar
- rar a -m5 -v1400k textos.rar * - Compacta todos os arquivos do diretório atual, usando a compactação máxima no arquivo textos.rar. Note que o tamanho máximo de cada arquivo é 1440 para ser possível gravá-lo em partes para disquetes.
- rar x -v -y textos.rar - Restaura os arquivos em múltiplos volumes criados com o processo anterior. Todos os arquivos devem ter sido copiados dos disquetes para o diretório atual antes de prosseguir. A opção -y é útil para não precisarmos responder yes a toda pergunta que o rar fizer.
- rar t textos.rar - Verifica se o arquivo textos.rar possui erros.
- rar r textos.rar - Repara um arquivo .rar danificado.

Capítulo 5 - Editor de Texto

Não se esqueça: Pratique cada comando e exemplo deste capítulo assim que eles são apresentados, para entender o seu funcionamento. Este é um capítulo extremamente prático, com muitos comandos para se decorar. Para fixar o aprendizado é necessário praticar.

O sistema operacional GNU/Linux, assim como qualquer outro Unix, possui todos os arquivos de configuração em formato texto, diferentemente de outros sistemas operacionais, que armazenam suas configurações em arquivos binários. Isso possui várias vantagens, entre elas a facilidade de modificação das configurações e a dificuldade de corrompimento dos arquivos, pois eles são simplesmente textos.

Ao se deparar com a necessidade de alterar algum desses arquivos, você pode utilizar editores bem completos, como o jed e XEmacs, ou um mais simples, como o gedit e até o joe. Mas nenhum desses editores se compara ao Vi, em termos de disponibilidade. O Vi existe em todas as distribuições do Linux e está disponível em qualquer emergência.

Levando em consideração esse fato, torna-se indispensável conhecer pelo menos o uso básico do Vi, apesar desse editor possuir variantes mais

completas, como o Vim.

O vim é o sucessor mais difundido do vi. vi é a sigla de Visual Interface No início dos anos 80, não era tão comum ter um editor de textos visual, ou seja, você ver na tela o texto que está sendo editado.

Muito mais tarde, em 1992, apareceu no mundo Unix um concorrente do vi, o vim (de "Vi IMitator", o imitador do vi) e logo se tornou popular, pois, além de ser um clone muito bem feito do vi, possuía muitas outras funcionalidades, como uso da tecla <TAB> para completar nomes de arquivos, vários níveis de "undo" (desfazer comando), reconhecimento de sintaxe e histórico de linha de comando.

Então, de "imitator" ele passou a ser chamado de "Vi IMproved", o vi melhorado. O vim se tornou padrão em sistemas Linux.

5.1 Um pouquinho sobre a interface do vim

O vim possui vários modos de operação, que são como estados do editor:

Modo	*Tecla*	*Rodapé*	*Descrição*
Inserção	*i*	*-- INSERT --*	*Inserção de texto*
Comandos de texto	*<ESC>*		*Comandos de manipulação*
Linha de comando arquivo	*:*	*:*	*Comandos de manipulação*
Visual	*v*	*-- VISUAL --*	*Seleção visual de texto*
Busca	*/*	*/*	*Busca de padrões no texto*
Reposição	*R*	*-- REPLACE --*	*Inserção sobrescrevendo*

Para acessar qualquer um desses modos, na dúvida aperte <ESC> antes da sua tecla de acesso (2ª coluna da tabela). Por exemplo, para entrar no modo de inserção, basta pressionar a tecla i.

Para identificar em qual modo você está, olhe no rodapé da tela (3ª coluna da tabela). O modo de inserção, por exemplo, é identificado pela mensagem - INSERT - na parte inferior esquerda da tela.

5.2 Entrando no vim

- Abre o arquivo "arquivo". Se ele não existir, o cria:

vim arquivo

- Abre com cursor no fim do arquivo:

vim arquivo +

- Abre com cursor na linha 13:

vim arquivo +13

- Abre com cursor na 1ª ocorrência da palavra "unix":

vim arquivo +/unix

5.3 Saindo do vim

:wq :x ZZ	Salva o arquivo e sai do editor
:w	Salva o arquivo corrente com o nome especificado
:w!	Salva o arquivo corrente no arquivo especificado
:q	Sai do editor
:q!	Sai do editor sem salvar as alterações realizadas
:w novo_arquivo.txt	Salva o arquivo atual com novo nome

5.4 Editando Texto

Ao entrar no vim, ele está no modo de comandos. Como saber? Olhe no rodapé da tela. Nada na última linha.

Para começar a Inserir um texto, aperte "i". Você verá um "--INSERT --" no rodapé. Agora você pode digitar seu texto normalmente.

Quer parar para salvar o arquivo? Tecle <ESC> e em seguida digite :w

Chega de editar e quer salvar e sair? Tecle <ESC> e em seguida digite :wq

5.5 Alterando Caracteres

Para alterar somente um caractere, posicione o cursor sobre este e pressione r, seguido do novo caractere.

Para alterar um grupo de caracteres, posicione o cursor sobre o primeiro destes e pressione R, seguido dos novos caracteres.

Para apagar até o fim da palavra (à direita do cursor) e entrar no modo de inserção: cw

Para apagar até o fim da linha e entrar no modo de inserção: c$

Para apagar até o começo da linha e entrar no modo de inserção: c^

Também são comandos do vim:

A	Insere texto depois do cursor
R	Insere texto no início da linha onde se encontra o cursor
A	Insere texto no final da linha onde se encontra o cursor
O	Adiciona linha abaixo da linha atual
O	Adiciona linha acima da linha atual
Ctrl + h	Apaga o último caractere

5.6 Copiando e colando

5.6.1 Usando o mouse:

No modo de Inserção, note que o cursor do teclado está numa posição, e o cursor do mouse em outra.

Selecione uma parte do texto com o mouse, segurando o botão esquerdo.

Ao colar, o texto selecionado com o MOUSE será inserido a partir do cursor do TECLADO, enquanto o texto ainda está selecionado.

Experimente!

Dependendo do mouse, a colagem pode ser feita dos seguintes modos:

- Apertando o botão direito

- Segurando o botão esquerdo e apertando o direito ao mesmo tempo

- Apertando o botão do meio (em mouses com três botões)

Tente! O seu será umas das três alternativas acima.

5.6.2 Usando o modo visual:

Entre no modo visual (<ESC>v) e aperte as setas do teclado, movendo o cursor e selecionando o texto desejado. O comando para copiar o texto é y.

Volte para o modo de comandos (<ESC>) e posicione o cursor do teclado no lugar onde você quer colar o texto selecionado. O comando de colagem é o p, de Paste.

5.6.3 Usando o modo comando:

- Para copiar a linha atual:

yy

- Para copiar as próximas cinco linhas:

5yy

- Para colar na próxima linha, abaixo do cursor

p

- Para colar acima do cursor:

P

5.7 Apagando texto

Como no tópico anterior, use o modo visual (<ESC>v) para selecionar o texto desejado. Para apagá-lo, digite d, de delete.

No modo comandos:

- Para apagar dez caracteres a partir do cursor:

10x

- Para apagar quatro palavras a partir do cursor:

4dw

- Para apagar a linha atual:

dd

- Para apagar sete linhas a partir do cursor:

7dd

- Para apagar do cursor ao fim da linha:

d$

- Para apagar do cursor ao inicio da linha:

d^

Também podem ser usados para apagar textos:

X	Deleta o caractere que está sob o cursor
Dw	Deleta a palavra, da posição atual do cursor até o final
Dd	Deleta a linha atual
D	Deleta a linha a partir da posição atual do cursor até o final
J	Une a linha corrente a próxima

5.8 Pedindo ajuda

Para chamar o help do vim:

:help

A ajuda do vim é construída com hiperlinks (texto destacado entre |barras|). Para navegar através do hiperlink, posicione o cursor sobre ele e pressione <Ctrl+]>.

Navegação	Comando
Seguir hiperlink	Ctrl-]
Voltar	Ctrl-T

5.9 Desfazendo e refazendo alterações

Para desfazer última ação:

- Para refazer a última ação:

.

- Para refazer uma ação desfeita:

<CTRL>+r

5.10 Mostrando informações sobre o arquivo

Para exibir informações sobre o arquivo atual:

<CTRL>+g

As informações terão a seguinte ordem:

“NomeDoArquivo.txt” | Modificado | número de linhas | posição relativa

5.11 Procurando textos no arquivo

- Para buscar a palavra “futuro” do cursor para frente:

/futuro

- Para buscar a palavra “passado” do cursor para trás:

?passado

- Repetindo a última busca para frente:

n

- Repetindo a última busca para trás:

N

- Para ir três linhas abaixo da próxima palavra teste:

/teste/+3

12. Busca múltipla

Para localizar a palavra primeira e logo depois desta localizar a palavra segunda:

/primeira/;segunda

- Para realizar uma busca, ao mesmo tempo, unix, linux e aix:

/unix\|linux\|aix

13. Navegando no documento

- Para ir para a linha 123

123 <SHIFT>+g

- Para ir para a última linha

<SHIFT>+g

Também são comandos do vim:

Ctrl+f	Move o cursor para a próxima tela
Ctrl+b	Move o cursor para a tela anterior
H	Move o cursor para a primeira linha da tela
M	Move o cursor para o meio da tela
L	Move o cursor para a última linha da tela
H	Move o cursor um caractere à esquerda
J	Move o cursor para a próxima linha
K	Move o cursor para linha anterior
L	Move o cursor um caractere à direita
W	Move o cursor para o início da próxima palavra (Ignora a pontuação)
W	Move o cursor para o início da próxima palavra (Não ignora a pontuação)
B	Move o cursor para o início da palavra anterior (Ignora a pontuação)
B	Move o cursor para o início da palavra anterior (Não ignora a pontuação)
0	Move o cursor para o início da linha atual
^	Move o cursor para o primeiro caractere não branco da linha atual
$	Move o cursor para o final da linha atual
Ng	Move o cursor para a linha n
G	Move o cursor para a última linha do arquivo

5.14 Substituindo texto

- Para trocar a primeira ocorrência da palavra antiga por nova na linha atual:

:s/antiga/nova

- Para trocar todas as ocorrências da palavra antiga por nova na linha atual:

:s/antiga/nova/g

- Para trocar todas as ocorrências da palavra antiga por nova nas linhas 5 a 18:

:5,18s/antiga/nova/g

- Para trocar todas as ocorrências da palavra antiga por nova no arquivo todo: :%s/antiga/nova/g

5.15 Editando vários arquivos ao mesmo tempo

- Todos na mesma tela com divisão na horizontal

vim -o arq1 arq2 arq3

- Todos na mesma tela com divisão na vertical

vim -O arq1 arq2 arq3

- Uma única tela com arquivos sobrepostos

vim arq1 arq2 arq3

Movimentação entre as janelas

Ctrl+w h <-

Ctrl+w j abaixo

Ctrl+w k acima

Ctrl+w l ->

Será aberta uma janela com o conteúdo do arq1, porém eu posso navegar entre os diversos arquivos da lista, segundo os comandos abaixo:

:e nome ; abre outro arquivo

:ls ; lista os arquivos abertos

:bn ; próximo arquivo

:bp ; arquivo anterior

:b1 ; vai para o primeiro arquivo, ou :b2, :b3,

5.16 Diversos

- Para inserir o conteúdo do arquivo ‘parte1’ dentro do arquivo atual:

:r parte1

- Vendo diferenças entre arquivos

vim -d arquivo1 arquivo2

- Para encontrar o parêntese correspondente digite % sobre um deles e o outro será destacado.

Capítulo 6 - Como obter ajuda no sistema

Dúvidas são comuns durante o uso do GNU/Linux e existem várias maneiras de se obter ajuda e encontrar a solução para algum problema. O GNU/Linux é um sistema bem documentado: provavelmente tudo o que imaginar fazer ou aprender já está disponível para leitura e aprendizado, seja na internet ou no man (páginas de manual).

As páginas de manual acompanham quase todos os programas GNU/Linux. Elas trazem uma descrição básica do comando/programa e detalhes sobre o funcionamento de opção. Uma página de manual é visualizada na forma de texto único, com rolagem vertical. Também documenta parâmetros usados em alguns arquivos de configuração.

A utilização da página de manual é simples. Digite:

man [seção] [comando/arquivo]

Onde:

seção

É a seção de manual que será aberta. Se omitida, mostra a primeira seção sobre o comando encontrada (em ordem crescente).

comando/arquivo

Comando/arquivo pra o qual deseja exibir informações.

A navegação dentro das páginas de manual é feita com o uso das teclas:

- q - Sai da página de manual
- PageDown ou f - Rola 25 linhas abaixo
- PageUP ou w - Rola 25 linhas acima
- SetaAcima ou k - Rola 1 linha acima
- SetaAbaixo ou e - Rola 1 linha abaixo
- r - Redesenha a tela (refresh)
- p ou g - Início da página
- h - Ajuda sobre as opções da página de manual
- s - Salva a página de manual em formato texto no arquivo especificado (por exemplo: /tmp/ls).

Capítulo 7 - Personalização do Sistema

Este capítulo ensina como personalizar algumas características de seu sistema GNU/Linux para distribuição Debian.

7.1 Variáveis de Ambiente

A utilização das variáveis de ambiente é um método simples e prático, que permite a especificação de opções de configuração de programas sem precisar mexer com arquivos no disco ou opções. Algumas variáveis do GNU/Linux afetam o comportamento de todo o Sistema Operacional, como o idioma utilizado e o path. Variáveis de ambiente são nomes que contém algum valor e têm a forma Nome=Valor. As variáveis de ambiente são individuais para cada usuário do sistema ou consoles virtuais e permanecem residentes na memória RAM até que o usuário saia do sistema (log-off) ou até que o sistema seja desligado.

As variáveis de ambiente são visualizadas/criadas através do comando set ou echo $NOME (apenas visualiza) e exportadas para o sistemas com o comando export NOME=VALOR.

Nos sistemas Debian, o local usado para especificar variáveis de ambiente é o /etc/environment (descrito mais adiante). Todas as variáveis especificadas nesse arquivo serão inicializadas e automaticamente exportadas na inicialização do sistema.

Exemplo: Para criar uma variável chamada TESTE que contenha o valor 123456 digite: export TESTE=123456. Agora, para ver o resultado, digite: echo $TESTE ou set|grep TESTE. Note que o $ que antecede o nome TESTE serve para identificar que se trata de uma variável e não de um arquivo comum.

7.2 Modificando o Idioma usado em seu sistema

- O idioma usado no seu sistema pode ser modificado facilmente através das variáveis de ambiente. Atualmente, a maioria dos programas estão sendo localizados. A localização é um recurso que especifica arquivos que contém as mensagens dos programas em outros idiomas. Você pode usar o comando locale para listar as variáveis de localização do sistema e seus respectivos valores. As principais variáveis usadas para determinar qual idioma os programas localizados utilizarão são: LANG - Especifica o idioma_PAIS local. Podem ser especificados mais de um idioma na mesma variável, separando-os com :. Dessa forma, caso o primeiro não esteja disponível para o programa, o segundo será verificado, e assim por diante. A língua inglesa é identificada pelo código C e usada como padrão caso nenhum locale seja especificado.

  Por exemplo: export LANG=pt_BR, export LANG=pt_BR:pt_

PT:C

- LC_MESSAGES - Especifica o idioma em que serão mostradas as mensagens dos programas. Seu formato é o mesmo de LANG.

- LC_ALL - Configura todas as variáveis de localização de uma só vez. Seu formato é o mesmo de LANG.

As mensagens de localização estão situadas em arquivos individuais de cada programa em /usr/share/locale/[Idioma]/LC_MESSAGES. Elas são geradas através de arquivos potfiles (arquivos com a extensão .po ou .pot e são gerados catálogos de mensagens .mo. As variáveis de ambiente podem ser especificadas no arquivo /etc/environment. Dessa forma, as variáveis serão carregadas toda vez que seu sistema for iniciado. Você também pode especificar as variáveis de localização em seus arquivos de inicialização .bash_profile, .bashrc ou .profile. Assim, toda vez que entrar no sistema, as variáveis de localização personalizadas serão carregadas.

Siga as instruções a seguir de acordo com a versão de sua distribuição Debian:

Debian 3.0

Acrescente a linha pt_BR ISO-8859-1 no arquivo /etc/locale.gen, rode o utilitário locale-gen para gerar os locales e acrescente as variáveis de localização no arquivo /etc/locales.def seguindo a forma:

export LANG=pt_BR

export LC_ALL=pt_BR

export LC_MESSAGES=pt_BR

Note que o arquivo /etc/environment também pode ser usado para tal tarefa, mas o locales.def foi criado especialmente para lidar com variáveis de localização na Debian 3.0.

Debian 2.2

Coloque essas variáveis no arquivo /etc/environment. Assim, toda a vez que seu sistema for iniciado, as variáveis de localização serão carregadas e exportadas para o sistema, estando disponíveis para todos os usuários.

Para as mensagens e programas do X-Window serem exibidas em seu idioma local, é preciso colocar as variáveis no arquivo ~/.xserverrc do diretório home de cada usuário e dar a permissão de execução neste arquivo (chmod 755 .xserverrc). Lembre-se de incluir o caminho completo do arquivo executável do seu gerenciador de janelas na última linha dele (sem o & no final). Caso contrário, o Xserver será finalizado logo após ler esse arquivo.

Seguem abaixo alguns exemplos de localização com as explicações:

- export LANG=pt_BR - Usa o idioma pt_BR como língua padrão do sistema. Caso o idioma Português do Brasil não esteja disponível, C é usado (Inglês).

- export LANG=C - Usa o idioma Inglês como padrão (é a mesma coisa de não especificar LANG, pois o idioma Inglês é usado como padrão).

- export LANG=pt_BR:pt_PT:es_ES:C - Usa o idioma Português do Brasil como padrão. Caso não esteja disponível, usa o Português de Portugal, se não estiver disponível usa o Espanhol e por fim o Inglês.

É recomendável usar a variável LC_ALL para especificar o idioma. Dessa forma, todas os outras variáveis (LANG, MESSAGES, LC_MONETARY, LC_NUMERIC, LC_COLLATE, LC_CTYPE e LC_TIME) serão configuradas automaticamente.

7.3 alias

Permite criar um apelido a um comando ou programa. Por exemplo, se você gosta de digitar (como eu) o comando ls --color=auto para ver uma listagem longa e colorida, você pode usar o comando alias para facilitar as coisas digitando: alias ls='ls --color=auto' (não se esqueça da meia aspa 'para identificar o comando'). Agora, quando você digitar ls, a listagem será mostrada com cores.

Se você digitar ls -la, a opção -la será adicionada no final da linha de comando do alias: ls --color=auto -la, e a listagem também será mostrada em cores.

Se quiser utilizar isso toda vez que entrar no sistema, adicione a informação do alias a um dos arquivos: .bash_profile ou .bashrc, como veremos logo adiante.

7.4 Arquivo /etc/profile

Este arquivo contém comandos que são executados para **todos** os usuários do sistema no momento do login. Somente o usuário root pode ter permissão para modificar este arquivo.

Este arquivo é lido antes do arquivo de configuração pessoal de cada usuário (.profile (root) e .bash_profile).

1. Quando é carregado através de um shell que requer login (nome e senha), o bash procura estes arquivos em seqüência e executa os comandos contidos, caso existam:/etc/profile
2. ~/.bash_profile
3. ~/.bash_login
4. ~/.profile

Ele interrompe a pesquisa assim que localiza o primeiro arquivo no diretório do usuário (usando a sequência acima). Por exemplo, se você tem o arquivo ~/.bash_login e ~/.bash_profile em seu diretório de usuário, ele processará o /etc/profile e após isto o ~/.bash_profile, mas nunca processará o ~/.bash_login (a menos que o ~/.bash_profile seja apagado ou renomeado).

Caso o bash seja carregado através de um shell que não requer login (um terminal no X, por exemplo), o seguinte arquivo é executado: ~/.bashrc.

Observação: Nos sistemas Debian, o profile do usuário root está configurado no arquivo /root/.profile. Isso ocorre porque, se o bash for carregado através do comando sh, ele fará a inicialização clássica deste shell lendo primeiro o arquivo /etc/profile e após o ~/.profile e ignorando *.bash_profile* e *.bashrc,* que são arquivos de configuração usados somente pelo bash. Por exemplo, inserir a linha mesg *y* no arquivo /etc/profile permite que todos os usuários do sistema recebam pedidos de talk de outros usuários. Se um usuário não quiser receber pedidos de talk, basta somente adicionar a linha mesg n no arquivo pessoal .bash_profile.

7.5 Arquivo .bash_profile

Este arquivo reside no diretório pessoal de cada usuário. É executado por shells que usam autenticação (nome e senha). *.bash_profile* contém comandos que são executados para o usuário no momento do login no sistema após o /etc/profile. Note que esse é um arquivo oculto, pois tem um "." no início do nome.

Por exemplo, colocando a linha: alias ls='ls --colors=auto' no .bash_ profile, é criado um apelido para o comando ls --colors=auto usando ls. Assim, toda vez que você digitar ls, será mostrada a listagem colorida.

7.6 Arquivo .bashrc

Possui as mesmas características do .bash_profile, mas é executado por shells que não requerem autenticação (como uma seção de terminal no X).

Os comandos desse arquivo são executados no momento que o usuário inicia um shell com as características acima. Note que esse é um arquivo oculto, pois tem um "." no início do nome.

7.7 Arquivo .hushlogin

Deve ser colocado no diretório pessoal do usuário. Este arquivo faz o bash pular as mensagens do /etc/motd, número de e-mails, etc. Ele exibe imediatamente o aviso de comando após a digitação da senha.

7.8 Arquivo /etc/environment

Armazena as variáveis de ambiente que são exportadas para todo o sistema. Uma variável de ambiente controla o comportamento de um programa, registra detalhes úteis durante a seção do usuário no sistema, especifica o idioma das mensagens do sistema, etc.

Exemplo do conteúdo de um arquivo /etc/environment:

```
LANG=pt_BR
LC_ALL=pt_BR
LC_MESSAGES=pt_BR
```

7.9 Diretório /etc/skel

Este diretório contém os modelos de arquivos .bash_profile e .bashrc, que serão copiados para o diretório pessoal dos usuários no momento que for criada uma conta no sistema. Dessa forma, você não precisará configurar esses arquivos separadamente para cada usuário.

Capítulo 8 - Programando em Shell Script

Este capítulo apresenta uma introdução ao shell script. Conhecimento básico de programação é requerido para o entendimento de alguns tópicos apresentados neste capítulo. O Shell utilizado neste curso é o Bourne Again Shell (bash).

8.1 O Ambiente Linux

Para entender o que é e como funciona o Shell, primeiro devemos saber como funciona o ambiente em camadas do Linux. Dê uma olhada no gráfico abaixo:

Neste gráfico é possível notar que a camada de hardware é a mais profunda, além de ser formada pelos componentes físicos do seu computador. Envolvendo-a, há a camada do kernel, que é o cerne do Linux, seu núcleo, e é quem bota o hardware para funcionar, fazendo seu gerenciamento e controle. Os programas e comandos que envolvem o kernel dele se utilizam para realizar as tarefas aplicativas para que foram desenvolvidos. Fechando tudo isso existe o Shell, que leva esse nome porque, em inglês, Shell significa concha, carapaça, isto é, fica entre o usuário e o sistema operacional, de forma que tudo o que interage com o

sistema operacional precisa passar pelo seu crivo.

8.2 O Ambiente Shell

Bom, já que para chegar ao núcleo do Linux, no seu kernel, que é o que interessa a todo aplicativo, é necessária a filtragem do Shell, vamos entender como ele funciona, de forma a tirar o máximo de proveito das inúmeras facilidades que ele nos oferece.

Por definição, o *Linux* é um sistema multiusuário – não podemos nunca esquecer disso – e para permitir o acesso de determinados usuários e barrar a entrada de outros, existe um arquivo chamado /etc/passwd, que, além de fornecer dados para a função de "leão-de-chácara" do Linux, também provê informações para o login daqueles que passaram pela primeira barreira. O último campo de seus registros informa ao sistema qual Shell a pessoa vai receber ao se "logar".

O Shell, que se vale da imagem de uma concha envolvendo o sistema operacional propriamente dito, é o nome genérico usado para tratar os filhos dessa idéia, que foi aparecendo ao longo dos anos de existência do sistema operacional Unix. Hoje existem diversos tipos de Shell e, dentre eles, destacamos o sh (Bourne Shell), o ksh (Korn Shell), o bash (Bourne Again Shell) e o csh (C Shell).

8.2.1 Principais Shells

*Bourne Shell (*sh*)*

Desenvolvido por Stephen Bourne, da Bell Labs (da AT&T, onde também foi desenvolvido o Unix), este foi durante muitos anos o Shell default do sistema operacional Unix. É também chamado de Standard Shell, por ter sido durante vários anos o único (até hoje é o mais utilizado, pois foi portado para todos os ambientes Unix e distros Linux).

*Korn Shell (*ksh*)*

Desenvolvido por David Korn, também da Bell Labs, é um superset do sh, isto é, possui todas as facilidades do sh e muitas outras. A compatibilidade total com o sh vem trazendo muitos usuários e programadores de Shell para esse ambiente.

*Bourne Again Shell (*bash*)*

Este é o Shell mais moderno e cujo número de adeptos mais cresce em todo o mundo, seja por ser o Shell default do Linux, seu sistema operacional hospedeiro, seja por sua grande diversidade de comandos, que incorpora inclusive diversas instruções características do C Shell.

*C Shell (*csh*)*

Desenvolvido por Bill Joy, da Berkley University, é o Shell mais utilizado em ambientes BSD e Xenix. A estruturação de seus comandos é bem similar à da linguagem C. Seu grande pecado foi ignorar a compatibilidade com o sh, partindo por um caminho próprio.

Também existem outros Shells. Neste material usaremos apenas o Bourne Again Shell (bash).

8.3 O que são os Shell Scripts?

Na linha de comandos de um shell, podemos utilizar diversos comandos um após o outro, ou mesmo combiná-los numa mesma linha. Se colocarmos diversas linhas de comandos em um arquivo texto simples, teremos em mãos um Shell Script, ou um script em shell, já que Script é uma descrição geral de qualquer programa escrito em linguagem interpretada (ou seja, não compilada). Outros exemplos de linguagens para scripts são o PHP, Perl, Python, JavaScript, NCL e muitos outros. Podemos então ter um script em PHP, um script Perl e assim em diante.

Uma vez criado, um ShellScript pode ser reutilizado quantas vezes for necessário. Seu uso, portanto, é indicado na automação de tarefas que serão realizadas mais de uma vez. Todo sistema Unix e similares é repleto de scripts em shell para a realização das mais diversas atividades administrativas e de manutenção do sistema.

Os arquivos de lote (batch - arquivos *.bat) do Windows são também exemplos de ShellScripts, já que são escritos em linguagem interpretada e executados por um Shell do Windows, em geral o command.com ou hoje em dia o cmd.exe. Os Shells do Unix, porém, são inúmeras vezes mais poderosos que o interpretador de comandos do Windows, podendo executar tarefas muito mais complexas e elaboradas.

Os scripts shell podem ser agendados para execução através da tabela crontab, entre outras coisas. Eles são ferramentas indispensáveis aos administradores de sistemas Unix.

O Shell mais comum e provavelmente o que possui mais scripts escritos para ele é também um dos mais antigos e simples, o sh. Esse shell está presente em todo o sistema tipo Unix, incluindo Linux, FreeBSD, AIX, HP-UX, OpenBSD, Solaris, NetBSD, Irix, etc. Por ser o shell nativo mais comum, é natural que se prefira escrever scripts para ele, tornando o script mais facilmente portável para outro sistema.

Os Shells não estão diretamente associados a um ou outro tipo de Unix, embora várias empresas comerciais tenham suas próprias versões de Shell. No software livre, o Shell utilizado num sistema é, em geral, exatamente o mesmo utilizado em outro. Por exemplo, o bash encontrado no Linux é o mesmo shell bash encontrado no FreeBSD e pode também facilmente ser instalado no Solaris ou em outros sistemas Unix comerciais, a fim de ser utilizado como interface direta de comandos ou como interpretador de scripts. O mesmo acontece com o tcsh e vários outros shells desenvolvidos no modelo de Software Livre.

- Os shell scripts podem conter estruturas de programação e estruturas de decisão (if)
- estruturas de repetição (for, while)
- funções etc.

8.4 Programação em Shell-Script

Nota: neste curso não são abordados conceitos básicos de programação. É necessário ter uma noção básica de programação para entender bem o conteúdo apresentado aqui.

8.4.1 Primeiros passos

Uma das vantagens do shell-script é que ele não precisa ser compilado, ou seja, basta apenas criar um arquivo texto qualquer e inserir comandos

nele. Para dar a esse arquivo a definição de shell-script, teremos que incluir uma linha no começo do arquivo:

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

Essa linha diz ao interpretador de comandos para que, quando o usuário executar o arquivo, ele execute através do programa /bin/bash, significando então que é um script shell!

Depois de colocada a linha, basta apenas tornar o arquivo executável, utilizando o comando chmod. Vamos seguir com um pequeno exemplo de um script shell que mostre na tela: Hello cruel world!:

```
#!/bin/bash

echo 'Hello cruel world!'
```

Fácil, hein? A primeira linha a gente já sabe para que serve. A segunda linha mostrará na tela a frase "Hello cruel world!", utilizando o comando echo, que serve justamente para isso (imprimir coisas na tela). Como você pôde ver, todos os comandos digitados diretamente na linha de comando poderão ser incluídos no seu script shell, criando uma série de comandos. E é essa combinação de comandos que forma o chamado shell script. Tente também dar o comando file nome_do_arquivo. Veja que a definição dele é de Bourne-Again Shell Script (Bash Script).

Para o arquivo poder se executável, você tem de atribuir o modo de executável para ele. E como já vimos, o comando chmod se encarrega disso:

```
chmod +x nome_do_arquivo
```

Pronto, o arquivo poderá ser executado com um simples ./nome_do_arquivo.

8.4.2 Conceito de variáveis em shell-script

Variáveis são caracteres que armazenam dados, são uma espécie de atalho. O bash reconhece uma variável quando ela começa com $, ou seja, a diferença entre *palavra* e *$palavra* é que a primeira é uma palavra qualquer e a outra uma variável. Para definir uma variável, utilizamos a seguinte sintaxe:

variavel="valor"

O "valor" será atribuído à "variável". "Valor" pode ser uma frase, números e até outras variáveis e comandos. O valor pode ser expressado entre as aspas (""), apóstrofos ('') ou crases (``). As aspas vão interpretar as variáveis que estiverem dentro do valor, os apóstrofos lerão o valor literalmente, sem interpretar nada e as crases vão interpretar um comando e retornar a sua saída para a variável. Vejamos exemplos para entender melhor:

(Digitando no terminal)

```
$ variavel="Eu estou logado como usuário $user"
$ echo $variavel
Eu estou logado como usuário cla


$ variavel='Eu estou logado como usuário $user'
$ echo $variavel
Eu estou logado como usuário $user
```

$ variavel=”Meu diretório atual é o `pwd`”

$ echo $variavel

Meu diretório atual é o /home/cla

Se você quiser criar um script em que o usuário deve interagir com ele, é possível que você queira que o próprio usuário defina uma variável e para isso usamos o comando read, que dará uma pausa no script e ficará esperando o usuário digitar algum valor e teclar <ENTER>. Por exemplo:

$ echo “Entre com o valor para a variável: “ ; read minhavariavel

O usuário digita e tecla Enter. Vamos supor que ele digitou a frase “Seja livre, use Linux”:

$ echo $variavel

Seja livre, use Linux

8.4.3 Outros comandos divertidos para se usar

Existem inúmeros comandos no Linux e, para explicar todos, teríamos que publicar um verdadeiro livro. Você pode usar livremente qualquer comando texto disponível no seu Linux. Também, se quiser, há muitas descrições na página de manual do bash, que pode ser acessada com o comando man bash.

Na tabela a seguir, você pode encontrar uma listagem de comandos para usar em seu shell script:

echo

Imprime um certo texto na tela, ou onde você indicar através de um redirecionador.

read

Captura dados do usuário e coloca numa variável. O primeiro parâmetro passado é a variável. (Exemplo: read nome_da_variavel)

exit

Finaliza o script. Caso seja passado algum número depois, ele retorna esse número como a saída do comando. Esse método geralmente serve para indicar se o comando foi bem sucedido ou não. (Exemplo: exit 1)

sleep

Dá uma parada em segundos no script, sendo esses segundos o primeiro argumento. (Exemplo: sleep 15 faz o script parar por 15 segundos)

clear

Limpa a tela.

stty

Configura o terminal temporariamente. Útil, por exemplo, para não aparecer o que o usuário digita na hora de escrever uma senha. (Exemplo: stty -echo e stty echo)

tput

Altera o modo de exibição, como por exemplo, as quantidades padrões de colunas de caracteres do terminal.

8.4.4 Controles de Fluxo

Controles de fluxo são comandos que vão testando algumas alternativas, e de acordo com essas alternativas vão executando outros comandos. Vamos ver aqui cada um deles.

8.4.4.1 Controle de fluxo com o if

Um dos comandos de controle de fluxo mais usados é certamente o if, que é baseado na lógica “se p for verdade, faça q. Se não, faça r”. Vamos dar um exemplo:

```
#!/bin/bash
if [ -e $linux ]
then
echo ‘A variável $linux existe.’
else
echo ‘A variável $linux não existe.’
fi
```

O que este pedaço de código faz? O if testa a seguinte expressão: se a variável $linux existir, então (then) ele diz que existe com o echo; se não (else), ele diz que não existe. O comando fi (if invertido) finaliza o bloco do comando if. O operador -e usado é pré-definido. Você pode encontrar a listagem dos operadores na tabela a seguir:

Operador	Descrição
-eq	Igual
!=	Diferente
-gt	Maior
-lt	Menor
!	Negação
-o	“Ou”
-d	Se for um diretório
-e	Se existir
-z	Se estiver vazio
-f	Se contiver texto
-O	Se o usuário for dono
-r	Se o arquivo pode ser lido
-w	Se o arquivo pode ser alterado
-x	Se o arquivo pode ser executado

Tabela de Operadores de fluxo do bash

Vamos para alguns exemplos, assim entendemos melhor o uso:

```
#!/bin/bash
if [ -z $naoexisto ]
then
echo 'A variável $naoexisto realmente não existe! Incrível.'
echo -n 'Criando... '
naoexisto="agoraeuexisto"
echo 'feito.'
else
echo 'Oops! Não era pra você existir!'
fi
- - - - - - - - -
#!/bin/bash
a=1
b=2

if [ $a -lt $b ]
then
echo '$a é menor que $b!'
elif [ $a -gt $b ]
then
echo '$a é maior que $b!'
elif [ $a -eq $b ]
then
echo '$a é igual a $b!'
fi
 - - - - - - - -
#!/bin/bash
if [ -d "/etc" ]
then
echo '/etc existe e é um diretório!'
if [ -f "/etc/passwd" ]
then
echo '/etc/passwd existe e é um arquivo!'
fi
else
echo '/etc não existe :('
fi
```

Note nos exemplos acima que citamos um "comando" não visto antes: o elif, que é uma combinação de else e if. Ao invés de fechar o if para criar outro, usamos o elif para testar uma expressão no mesmo comando if.

8.4.4.2 Fluxo com o case

O case é para controle de fluxo, tal como é o if. Mas enquanto o if testa expressões não exatas, o case vai agir de acordo com os resultados exatos. Vejamos um exemplo:

```
case $1 in
parametro1) comando1 ; comando2 ;;
parametro2) comando3 ; comando4 ;;
*) echo "Você tem de entrar com um parâmetro válido" ;;
esac
```

Aqui aconteceu o seguinte: o case leu a variável $1 (que é o primeiro parâmetro passado para o programa) e comparou com valores exatos. Se a variável $1 for igual a "parametro1", então o programa executará o comando1 e o comando2; se for igual a "parametro2", executará o comando3

e o comando4 e assim por diante. A última opção (*), é uma opção padrão do case, ou seja, se o parâmetro passado não for igual a nenhuma das outras opções anteriores, esse comando será executado automaticamente.

Você pode ver que, com o case fica muito mais fácil criar uma espécie de “menu” para o shell script do que com o if. Vamos demonstrar a mesma função anterior, mas agora usando o if:

```
if [ -z $1 ]; then
echo “Você tem de entrar com um parâmetro válido”
exit
elif [ $1 = “parametro1” ]; then
comando1
comando2
elif [ $1 = “parametro2” ]; then
comando3
comando4
else
echo “Você tem de entrar com um parâmetro válido”
fi
```

Veja a diferença. É muito mais prático usar o case! A vantagem do if é que ele pode testar várias expressões que o case não pode. O case é mais prático, mas o if pode substituí-lo e ainda abrange mais funções.

8.4.4.3 Fluxo com o for

O laço for substitui, sucessivamente, uma variável por um valor, e executa os comandos especificados em cada interação (substituição do valor da variável). Veja o exemplo:

```
for i in *
do
cp $i $i.backup
mv $i.backup /usr/backup
done
```

Esse código executa o backup de todos os arquivos no diretório atual, onde o script for executado.

Primeiramente o laço for atribui um valor de retorno do comando * (que é equivalente a um ls sem nenhum parâmetro) para a variável $i. Depois, executa o bloco de comandos. Em seguida ele atribui outro valor do comando * para a variável $i e executa novamente os comandos. Isso se repete até que não sobrem valores de retorno do comando *.

Como o exemplo ilustra, o laço for pode ser bem útil no tratamento de múltiplos arquivos. Você pode organizar dados, fazer backup, entre outras coisas.

8.4.4.4 Fluxo com o while

O while testa continuamente uma expressão, até que ela se torne falsa.

Exemplo:

```
variavel=”valor”
while [ $variavel = “valor” ]; do
comando1
comando2
done
```

O que acontece aqui é o seguinte: enquanto a $variavel for igual a “valor”, o while ficará executando os comandos 1 e 2, até que a $variavel não seja mais igual a “valor”. Se no bloco dos comandos a $variavel mudasse, o while iria parar de executar os comandos quando chegasse em done, pois agora a expressão $variavel = “valor” não seria mais verdadeira.

8.4.4.5 Fluxo com o until

Tem as mesmas características do while, a única diferença é que ele faz o contrário. Veja o exemplo abaixo:

```
variavel=”naovalor”
until [ $variavel = “valor” ]; do
comando1
comando2
done
```

Ao invés de executar o bloco de comandos (comando1 e comando2) até que a expressão se torne falsa, o until testa a expressão e executa o bloco de comandos até que a expressão se torne verdadeira. No exemplo, o bloco de comandos será executado desde que a expressão $variavel = “valor” não seja verdadeira. Se no bloco de comandos a $variavel for definida como “valor”, o until pára de executar os comandos quando chega ao done.

Vejamos um exemplo para o until que, sintaticamente invertido, serve para o while também:

```
var=1
count=0
until [ $var = “0” ]; do
comando1
comando2
if [ $count = 9 ]; then
var=0
fi
count=`expr $count + 1`
done
```

Primeiro, atribuímos à variável *$var* o valor 1. A variável $count será uma contagem para quantas vezes quisermos executar o bloco de comandos. O until executa os comandos 1 e 2, enquanto a variável $var for igual a 0. Então usamos um if para atribuir o valor 0 para a variável $var, se a variável $count for igual a 9. Depois soma-se 1 à variável $count (o comando expr avalia expressões aritméticas. Digite man expr para detalhes). Isso cria um laço que executa o comando 10 vezes, porque a cada vez que o comando do bloco de comandos é executado, soma-se 1 à variável $count e quando chega em 9, a variável $var é igualada a zero, quebrando assim o laço until.

8.4.5 Usando vários scripts em um só

Pode-se precisar criar vários scripts shell que fazem funções diferentes, mas e se você precisar executar em um script shell um outro script externo para que este faça alguma função, e assim não precisar reescrever todo o código? É simples, você só precisa incluir o seguinte comando no seu script shell:

```
. bashscript2
```

Isso executará o script shell bashscript2 durante a execução do seu script shell. Nesse caso, ele será executado no mesmo shell em que está sendo usado o comando. Para utilizar outro shell, você simplesmente substitui o . pelo executável da shell, assim:

```
sh script2
tcsh script3
```

Nessas linhas o script2 será executado com o shell sh e o script3 com o shell tcsh.

8.4.6 Variáveis especiais

Variável	Função
$0	Nome do script que está sendo executado
$1-$9	Parâmetros passados à linha de comando (de 1 a 9)
$#	Número de parâmetros passados
$?	Valor de retorno do último comando ou de todo o shell script (o comando exit 1, retorna o valor 1).
$$	Retorno do PID (Process ID).

Tabela de variáveis especiais

Você também encontra muitas variáveis já predefinidas na página de manual do bash (comando man bash, seção Shell Variables).

8.4.7 Funções

Funções são blocos de comandos que podem ser definidos para uso posterior em qualquer parte do código. Praticamente todas as linguagens de programação usam funções que ajudam a organizar o código. Vejamos a sintaxe de uma função:

```
nome_da_funcao() {
comando1
comando2
...
comandoN
}
```

Fácil de entender, não? A função funcionará como um simples comando próprio. Você executa a função em qualquer lugar do script shell e os comandos 1, 2 e outros serão executados. A flexibilidade das funções permite facilitar a vida do programador, como no exemplo final deste capítulo.

8.4.8 Exemplo final de shell-script

Agora vamos ver um exemplo de um programa que utilize o que aprendemos anteriormente.

A funcionalidade desse script pode ser observada lendo-se o código.

```
#!/bin/bash
# Exemplo Final de Script Shell
Principal() {
echo “Exemplo Final sobre o uso de scripts shell”
echo “----------------------------------------”
echo “Opções:”
echo
echo “1. Trasformar nomes de arquivos”
echo “2. Adicionar um usuário no sistema”
echo “3. Deletar um usuário no sistema”
echo “4. Fazer backup dos arquivos do /etc”
echo “5. Sair do exemplo”
echo
echo -n “Qual a opção desejada? “
read opcao
case $opcao in
1) Transformar ;;
2) Adicionar ;;
3) Deletar ;;
4) Backup ;;
5) exit ;;
*) “Opção desconhecida.” ; echo ; Principal ;;
esac
}
Transformar() {
echo -n “Para Maiúsculo ou minúsculo? [M/m] “
read var
if [ $var = “M” ]; then
echo -n “Que diretório? “
read dir
for x in `/bin/ls` $dir; do
y=`echo $x | tr ‘[:lower:]’ ‘[:upper:]’`
if [ ! -e $y ]; then
mv $x $y
fi
done
elif [ $var = “m” ]; then
echo -n “Que diretório? “
read dir
for x in `/bin/ls` $dir; do
y=`echo $x | tr ‘[:upper:]’ ‘[:lower:]’`
if [ ! -e $y ]; then
mv $x $y
fi
done
fi
}
Adicionar() {
clear
echo -n “Qual o nome do usuário a se adicionar? “
read nome
adduser nome
Principal
}
Deletar() {
clear
```

```
echo -n “Qual o nome do usuário a deletar? “
read nome
userdel nome
Principal
}
Backup() {
for x in `/bin/ls` /etc; do
cp -R /etc/$x /etc/$x.bck
mv /etc/$x.bck /usr/backup
done
}
Principal
```