



CELEPAR
INFORMÁTICA
do PARANÁ

Tecnologia para a Democracia




GOVERNO DO
PARANÁ

Gerência de Gestão de Ambientes – GGA
Coordenação de Planejamento, Pesquisa e Capacitação – CPPC

APOSTILA GNU/DEBIAN AVANÇADO

Direitos autorais:

Essa Apostila está licenciada sob uma Licença Creative Commons Atribuição-**Uso Não-Comercial-Compartilhamento** pela mesma licença 2.0 Brazil. Pode ser copiada, distribuída e modificada. Para ver uma cópia desta licença, visite <http://creativecommons.org/licenses/by-nc-sa/2.0/br/> ou envie uma carta para Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.




Atribuição-Uso Não-Comercial-Compartilhamento pela mesma licença 2.0 Brasil


Você pode:

- copiar, distribuir, exibir e executar a obra
- criar obras derivadas


Sob as seguintes condições:



Atribuição. Você deve dar crédito ao autor original, da forma especificada pelo autor ou licenciante.



Uso Não-Comercial. Você não pode utilizar esta obra com finalidades comerciais.




Compartilhamento pela mesma Licença. Se você alterar, transformar, ou criar outra obra com base nesta, você somente poderá distribuir a obra resultante sob uma licença idêntica a esta.

- Para cada novo uso ou distribuição, você deve deixar claro para outros os termos da licença desta obra.
- Qualquer uma destas condições podem ser renunciadas, desde que Você obtenha permissão do autor.

Qualquer direito de uso legítimo (ou "fair use") concedido por lei, ou qualquer outro direito protegido pela legislação local, não são em hipótese alguma afetados pelo disposto acima.

Este é um sumário para leigos da [Licença Jurídica \(na íntegra\)](#).

[Termo de exoneração de responsabilidade](#) 

Documento	Apostila de Debian Avançado
Versão	2.2-1
Data da Revisão	01/09/2008
Equipe Técnica	Jonsue Trapp Martins André Luiz de Souza Paula David Alves França Paulo César de Oliveira Peter Andreas Entschew
Páginas	124

Índice

1. Introdução.....	10
2. Instalando o GNU/Debian.....	11
3. Como Instalar Programas.....	17
3.1. APT.....	17
3.1.1. Reconfigurando a lista de pacotes.....	19
3.1.2. Após reconfigurar a lista de pacotes.....	19
3.1.3. O arquivo de configuração “apt.conf”.....	20
3.1.4. O arquivo de configuração “preferences”.....	21
3.2. dpkg.....	22
3.3. Compilando Programas a Partir do Código Fonte.....	22
3.3.1. Arquivos tarball.....	22
4. Gerenciando Discos e Partições.....	24
4.1. Obtendo Informações sobre os Discos Rígidos.....	24
4.2. Particionando o Disco com o cfdisk.....	24
4.3. Criando uma Partição Primária.....	25
4.4. Criando uma Partição Lógica.....	26
4.5. Excluindo uma Partição.....	26
4.6. Salvando as Alterações no Disco.....	26
4.7. Criando o Sistema de Arquivos.....	26
5. Gerenciador de Partida - GRUB.....	28
5.1. Como o GRUB Trabalha com Discos e Partições.....	28
5.2. Instalando o GRUB.....	28
5.3. No Disco Flexível (com interface de menu).....	29
5.4. Opções do Arquivo de Configuração.....	29
5.4.1. Parâmetros globais.....	29
5.4.2. Parâmetros que afetam apenas as imagens.....	30
5.4.3. Parâmetros enviados diretamente ao kernel.....	31
5.5. Usando a Linha de Comandos do GRUB.....	31
5.6. Removendo o GRUB do MBR.....	32
6. Sistema de Boot e Runlevels.....	34
6.1. Processo de Carregamento do Kernel.....	34
6.2. Entendendo o Funcionamento dos Níveis de Execução do Sistema (Runlevels).....	34
7. Sistema de Logs.....	36
7.1. Principais Arquivos de Log.....	36
7.2. Formato dos Arquivos de Log.....	36

7.3. Daemons de Log do Sistema.....	37
7.3.1. O daemon syslogd.....	37
7.3.2. O daemon klogd.....	38
7.3.3. Configurando o syslog.conf.....	39
7.4. Ferramentas para Arquivos de Log.....	41
7.4.1. logger.....	41
7.4.2. logcheck.....	42
7.4.3. logrotate.....	42
8. Comandos Avançados.....	46
8.1. Comandos de Uso Geral.....	46
8.1.1. cron.....	46
8.1.2. dd.....	48
8.1.3. diff.....	48
8.1.4. dpkg.....	50
8.1.5. export.....	52
8.1.6. find.....	53
8.1.7. fsck.....	53
8.1.8. last.....	55
8.1.9. lastlog.....	56
8.1.10. lsof.....	56
8.1.11. lspci.....	57
8.1.12. lsusb.....	58
8.1.13. mount.....	59
8.1.14. nice.....	59
8.1.15. renice.....	60
8.1.16. nohup.....	61
8.1.17. pidof.....	61
8.1.18. pstree.....	61
8.1.19. shred.....	62
8.1.20. stat.....	63
8.1.21. sync.....	64
8.1.22. tar.....	64
8.1.23. watch.....	66
8.1.24. which.....	67
8.2. Comandos de Rede.....	68
8.2.1. arp.....	68
8.2.2. dig.....	69
8.2.3. ifconfig.....	70
8.2.4. ifup e ifdown.....	71

8.2.5. mii-tool.....	72
8.2.6. netstat.....	73
8.2.7. ntpdate.....	74
8.2.8. route.....	75
8.2.9. tcpdmatch.....	77
8.2.10. traceroute.....	78
8.2.11. wget.....	79
9. Arquivos de Configuração.....	81
9.1. Interfaces de Rede.....	81
9.2. Interface de “loopback”.....	81
9.3. Serviços e Portas TCP/IP.....	82
9.4. Arquivos de Configuração.....	82
9.4.1. /etc/network/interfaces.....	82
9.4.2. /etc/hostname.....	84
9.4.3. /etc/hosts.....	84
9.4.4. /etc/host.conf.....	84
9.4.5. /etc/networks.....	85
9.4.6. /etc/resolv.conf.....	85
9.4.7. /etc/nsswitch.conf.....	86
9.4.8. /etc/services.....	86
9.4.9. /etc/protocols.....	86
9.4.10. /etc/alternatives/.....	86
9.4.11. /etc/default/rcS.....	87
9.4.12. /etc/pam.d/.....	87
9.4.13. /etc/security/.....	87
9.4.14. /etc/security/access.conf.....	87
9.4.15. /etc/security/limits.conf.....	88
9.4.16. /etc/fstab.....	88
9.4.17. /etc/hosts.allow.....	89
9.4.18. /etc/hosts.deny.....	90
9.4.19. /etc/inetd.conf.....	91
9.4.20. /etc/inittab.....	91
9.4.21. /etc/mtab.....	92
9.4.22. /etc/issue.....	92
9.4.23. /etc/issue.net.....	92
9.4.24. /etc/motd.....	93
10. Firewall iptables – Conceitos Básicos.....	94
10.1. Introdução.....	94
10.2. Tipos de Firewall.....	94

10.3. Planejando a Implementação.....	95
10.4. Conhecendo os conceitos do iptables.....	95
10.4.1. O que são regras?.....	95
10.4.2. O que são chains?.....	95
10.4.3. O que são tabelas?.....	95
10.5. Manipulando Chains.....	96
10.5.1. Adicionando regras.....	96
10.5.2. Listando regras.....	97
10.5.3. Apagando uma regra.....	98
10.5.4. Inserindo uma regra.....	99
10.5.5. Substituindo uma regra.....	99
10.5.6. Criando um novo chain.....	99
10.5.7. Renomeando um chain criado pelo usuário.....	100
10.5.8. Limpando as regras de um chain.....	100
10.5.9. Apagando um chain criado pelo usuário.....	101
10.5.10. Zerando o contador de bytes dos chains.....	101
10.5.11. Especificando o policiamento padrão de um chain.....	101
10.5.12. Especificando um endereço de origem/destino.....	102
10.5.13. Especificando a interface de origem/destino.....	102
10.5.14. Especificando um protocolo.....	103
10.5.15. Especificando portas de origem/destino.....	103
11. Compilação do Kernel.....	105
11.1. Preparação para Compilação.....	105
11.2. Compilando o Kernel.....	106
11.3. Utilitários do Kernel.....	108
11.3.1. depmod.....	108
11.3.2. insmod.....	109
11.3.3. lsmod.....	109
11.3.4. modinfo.....	110
11.3.5. modprobe.....	110
11.3.6. rmmod.....	112
11.3.7. sysctl.....	113
12. Anexos.....	115
12.1. NFS - Network File System.....	115
12.2. Shell Scripts.....	116
12.2.1. O primeiro script.....	116
12.2.2. Os primeiros erros.....	116
12.2.3. Melhorando seu primeiro script.....	117
12.2.4. Expressões aritméticas.....	117

12.2.5. Recebendo argumentos do shell.....	117
12.2.6. Estruturas de controle.....	118
12.2.7. Funções.....	121
12.2.8. sed.....	121

Índice de Figuras

Figura 1 - Definindo as opções de instalação do GNU/Debian.....	11
Figura 2 - Tela de configuração das partições do disco.....	13
Figura 3 - Finalizando a configuração das partições.....	14
Figura 4 - Menu principal do utilitário cfdisk.....	25

1. Introdução

Este documento, foi elaborado especialmente para o curso de Debian avançado ministrado pela Coordenação de Planejamento, Pesquisa e Capacitação – CPPC, aos técnicos da área de informática da Celepar e seus clientes.

O objetivo que desejamos alcançar com este material, é aprofundar os conhecimentos sobre conceitos e funcionalidades da distribuição GNU/Debian, como forma de capacitar o pessoal de técnico, para realização de atividades geralmente vistas como mais complexas, como por exemplo, a compilação do *kernel* do sistema.

Esta apostila foi gerada tendo como base o “Guia Foca GNU/Linux Avançado”, de *Gleydson Mazioli da Silva*, que pode ser acessado em “<http://focalinux.cipsga.org.br/guia/avancado/index.htm>”.

A Gerência de Gestão de Ambientes – GGA, por meio da Coordenação de Planejamento, Pesquisa e Capacitação – CPPC, espera que você possa aproveitar o conteúdo deste material para ampliar mais o seu conhecimento do Sistema Operacional GNU/Linux, e em especial, da distribuição Debian.

2. Instalando o GNU/Debian

Para ter acesso a todas as opções de instalação do GNU/Debian, devemos utilizar o modo “expert” da instalação. Esta é também, a opção recomendada para a instalação de servidores, visto que somente ela, dá acesso a funcionalidades que são mascaradas pela instalação padrão.

Siga os passos abaixo para realizar a instalação do Sistema Operacional:

1. Verifique as configurações de *setup* (BIOS) do equipamento, deixando o CD-ROM como primeiro dispositivo de *boot*.
2. Dê um *boot* com o CD do “Debian Business Card”, faça o *download* no seguinte endereço: <http://www.repositorios.eparana.parana/images/debian/etch/i386/debian-40r1-i386-businesscard.iso>
3. Escreva “**expert vga=788**” (sem as aspas) na tela inicial do CD do Debian (tela abaixo) e em seguida pressione <ENTER>. A figura a seguir, representa esta parte da instalação:



Figura 1 - Definindo as opções de instalação do GNU/Debian.

4. No menu principal do instalador do GNU/Debian, escolha a opção “Choose Language” e pressione <ENTER>. Na tela seguinte, escolha a opção “**Portuguese (Brazil) – Português do Brasil**” e pressione <ENTER>.
5. A próxima tela, pedirá que você escolha um país, território ou área, escolha a opção “**Brasil**” e pressione <ENTER>.
6. A seguir a tela “Escolha um locale” aparecerá. Escolha a opção “**pt_BR**” e pressione <ENTER> para gerar a codificação de caracteres necessária apenas para o nosso idioma.
7. Prosseguindo a instalação, será apresentada mais uma tela permitindo que o usuário escolha outros *locales*, isto não é necessário, basta então pressionar a tecla <TAB> e em seguida <ENTER>, para continuar.
8. Após a configuração do idioma do sistema, é necessário configurar o teclado, através da opção “Selecione um layout de teclado”. Pressione <ENTER> sobre esta opção, que consta do menu principal do instalador.
9. Na tela que solicitará o “Tipo de teclado”, selecione a opção “**Teclado estilo PC (AT ou PS/2)**” e pressione <ENTER>.
10. A seguir, na tela “Mapa de teclado a ser usado”, escolha o *layout* de teclado adequado para seu computador. Lembre-se, teclados que possuem “ç” deverão escolher a opção “**Português Brasileiro (br-abnt2)**”.

11. A instalação voltará para o menu principal. Escolha agora a opção **"Detectar e montar CDROM"** e pressione **<ENTER>**.
12. No questionamento sobre os "Módulos a serem carregados", apenas escolha a opção **<Continuar>** e pressione **<ENTER>**.
13. A próxima pergunta é sobre os módulos para cartões PCMCIA, geralmente utilizados em *notebooks*, responda **"Não"** a esta pergunta para prosseguir a instalação.
14. Num processo normal, a próxima tela informará que o CDROM foi detectado com sucesso. Basta escolher a opção **<Continuar>** e pressionar **<ENTER>**.
15. No menu principal, escolha a opção **"Carregar componentes do instalador do CD"** e pressione a tecla **<ENTER>**.
16. A tela seguinte, permitirá que você escolha que componentes deseja carregar. Não é necessário marcar nenhum componente, basta ir até a opção **<Continuar>** e pressionar **<ENTER>**.
17. Novamente no menu principal, acesse o item **"Detectar hardware de rede"**, pressionando a tecla **<ENTER>** sobre ele.
18. Após o processo automático de detecção do hardware, pressione **<ENTER>** sobre o item **"Configurar a rede"** do menu principal do instalador.
19. Na sequência uma série de perguntas serão feitas pelo instalador para realizar a configuração da rede da máquina. A primeira dessa série de perguntas é **"Configurar a rede automaticamente por DHCP ?"**, responda **"Não"** para que possamos configurar opções do protocolo TCP/IP manualmente.
20. Após o 19º passo, você será questionado sobre o endereço IP da máquina, informe o endereço no formato padrão, por exemplo: **10.15.22.15**
21. Em seguida, é necessário informar a mascara de rede. Informe a mascara a ser usada, conforme o exemplo: **255.255.255.0**
22. A próxima informação necessária, é o endereço do *gateway* (roteador) padrão da rede. Informe o endereço no mesmo formato já citado no passo 20 desta seção.
23. Continuando a instalação, agora é hora de informar os servidores de nomes (DNS) da rede. Informe os endereços IP dos servidores separados por espaço, caso deseje informar mais de um servidor, conforme o exemplo ao lado: **10.15.22.4 10.15.21.5**
24. Será mostrada então uma tela, que exibirá os dados fornecidos para as questões a partir do passo 20, confira os dados e se estiver tudo correto, acione a opção **"Sim"** e pressione **<ENTER>**. Caso você tenha fornecido alguma informação incorreta, acione a opção **"Não"** e pressione **<ENTER>** para que você possa digitar as informações novamente.
25. Confirmando positivamente o questionamento do passo 24, será apresentada uma tela que solicitará ao usuário que forneça um nome para a máquina (*hostname*). Informe um nome para a máquina de acordo com os padrões da organização a que a máquina pertence. Lembre-se, manter padrões é uma atitude importante para a administração de redes.
26. A próxima questão, é sobre o domínio (DNS) ao qual a máquina pertence. Informe o domínio em questão, no mesmo formato do exemplo a seguir: **celepar.parana**
27. Nesse momento começa a configuração dos repositórios para instalação remota de pacotes. Pressione **<ENTER>** sobre a opção **"Selecione um espelho para o repositório Debian"** do menu principal de instalação.
28. Na pergunta **"Protocolo para download de arquivos"**, selecione **"http"** e pressione **<ENTER>**.
29. Na questão **"País do espelho do repositório Debian"**, selecione a opção **"Fornecer informação manualmente"** e pressione **<ENTER>**.
30. A seguir em **"Nome da máquina do espelho do repositório Debian"**, informe o espelho do repositório da Celepar, fornecendo o parâmetro: www.repositorios.eaparana.parana

31. A próxima pergunta, é sobre o diretório do Debian nesta maquina espelho. Deixe conforme o padrão sugerido **`/debian/`** e pressione **<ENTER>**.
32. Agora a questão é sobre o *proxy* http. No caso da rede interna do Estado, presente na maioria do clientes da Celepar, não é necessário fornecer endereço de um servidor *proxy* para endereços da intranet. Por essa razão, basta pressionar **<ENTER>** para prosseguir a instalação.
33. Na pergunta **“Versão do Debian a ser instalada”**, opte sempre pela opção **“Stable”** e pressione **<ENTER>**.
34. Voltando ao menu principal de instalação, pressione **<ENTER>** sobre a opção **“Detectar discos”**.
35. Após a detecção automática do hardware, pressione **<ENTER>** sobre a opção **“Particionar discos”**.
36. O método de particionamento a ser escolhido, deve ser a opção **“Manual”**. Caso a tela mostrada, não apresente um indicativo de **“ESPAÇO LIVRE”** no disco ao qual se deseja particionar, conforme a figura abaixo exemplifica (listra vermelha), é necessário criar uma tabela de partições no disco em questão.

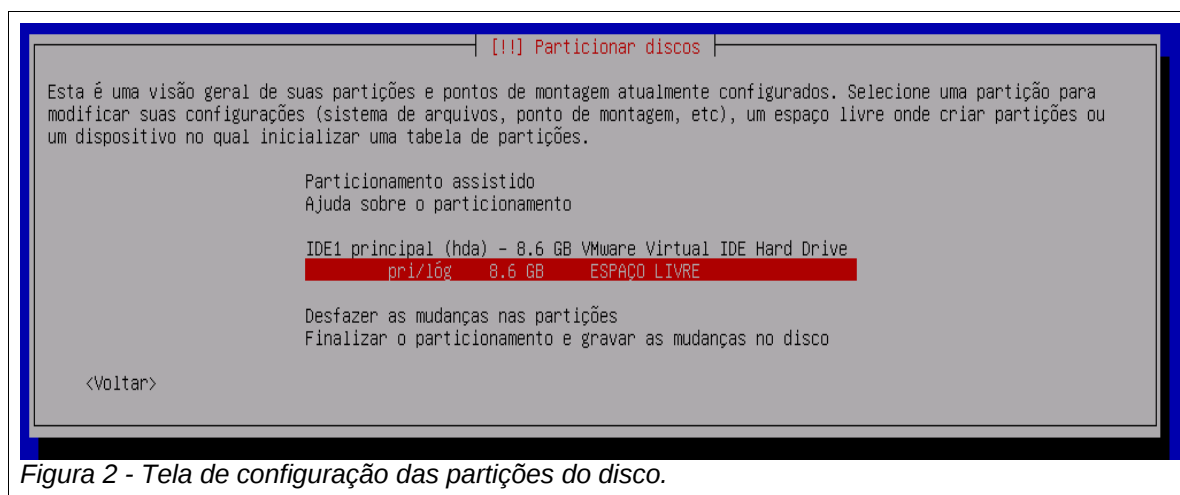


Figura 2 - Tela de configuração das partições do disco.

- Para criar uma nova tabela de partições, pressione **<ENTER>** sobre o nome que faz referência ao disco em questão, VMware Virtual IDE Hard Drive no exemplo da figura acima, e responda **“Sim”** a pergunta sobre se deseja criar uma nova tabela de partições. A seguir escolha o tipo de tabela de partições do **“msdos”** e pressione **<ENTER>** para encerrar a configuração.
37. Selecione a indicação de **“ESPAÇO LIVRE”** no disco ao qual se deseja particionar e pressione a tecla **<ENTER>**, para iniciar a criação de uma nova partição. Lembre-se, a criação de partições deve sempre respeitar o projeto de instalação definido para a maquina que está sendo configurada.
 38. No diálogo seguinte **“Como usar este espaço livre”**, selecione a opção **“Criar nova partição”** e pressione **<ENTER>**.
 39. A seguir, especifique o tamanho em Megabytes ou Gigabytes que será designado para a partição que está sendo configurada. Valores válidos, são por exemplo: **128 Mb** ou **1 GB**
 40. O **“Tipo da nova partição”** deverá ser prioritariamente **“Primária”** para partições do sistema como **“/boot”**, **“/”** ou áreas de troca. Outras partições como **“/home”**, **“/var”**, **“/srv”**, entre outras, poderão usar a opção **“Lógica”**.
 41. Na pergunta **“Localização para nova partição”**, responda sempre **“Inicio”** e pressione **<ENTER>**.

42. A tela de configuração da partição será exibida, nela você poderá ajustar o ponto de montagem, sistema de arquivos e outras opções. Para realizar a configuração de cada um destes itens, basta pressionar a tecla **<ENTER>** sobre cada uma das opções e ajustar as opções de acordo com a necessidade. Feitas as configurações necessárias, escolha a opção **“Finalizar a configuração da partição”**. Repita os passos do 37 ao 42, para criação das demais partições necessárias de acordo com o projeto de instalação da máquina. Abaixo, citamos um exemplo de particionamento genérico que pode ser usado para instalação:

Ponto de montagem	Tamanho	Usar como	Flag inicializável	Função
/boot	128 MB	Sistema de arquivos EXT3 com journalling	ligado	Partição para arquivos de <i>boot</i>
/	10 GB	Sistema de arquivos XFS com journalling	desligado	Partição raiz
swap	1 GB	área de troca	desligado	Área para swap de memória
/home	Estimar 100 Mb por usuário	Sistema de arquivos XFS com journalling	desligado	Partição para arquivos do usuário
/var	Restante do disco	Sistema de arquivos XFS com journalling	desligado	Partição para arquivos de dados dos serviços de rede

43. Após criar todas as partições necessárias escolha a opção **“Finalizar o particionamento e gravar as mudanças no disco”**, conforme a figura abaixo:

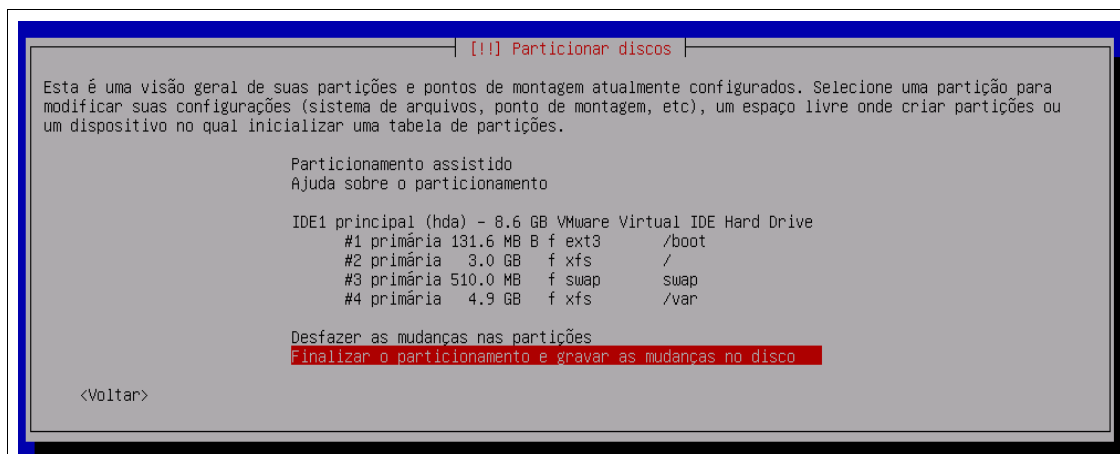


Figura 3 - Finalizando a configuração das partições.

44. Um diálogo de confirmação irá ser exibido **“Gravar estas mudanças nos discos ?”**, responda **“Sim”**.
45. Novamente no menu principal do instalador, pressione **<ENTER>** sobre o item **“Configurar fuso horário”**.
46. Na pergunta **“Selecione uma cidade em seu fuso horário”**, escolha **“São Paulo”** e

- pressione a tecla **<ENTER>**.
47. Você será enviado novamente ao menu principal do instalador, pressione **<ENTER>** sobre o item **"Configurar relógio"** agora.
 48. Quando o instalador perguntar se o relógio do sistema está configurado para UTC selecione a opção **"Sim"** e pressione **<ENTER>**.
 49. O próximo passo, é a configuração de usuários. Selecione o item **"Configurar usuários e senhas"** e pressione **<ENTER>**.
 50. Responda **"Sim"** à pergunta **"Habilitar senhas sombra (shadow) ?"**.
 51. Em **"Permitir login como root ?"**, responda **"Sim"** novamente.
 52. O próximo passo exige que você defina uma senha para a conta do *root*.
 53. Informe novamente, a mesma senha já fornecida para o usuário *root* para confirmação.
 54. Agora é necessário criar uma conta de usuário comum. Responda **"Sim"** a questão **"Criar uma conta de usuário normal agora ?"**.
 55. Em **"Nome completo para o novo usuário"**, digite **"celepar"** e pressione a tecla **<ENTER>**.
 56. Em **"Nome de usuário para sua conta"**, digite novamente **"celepar"** e pressione a tecla **<ENTER>**.
 57. No passo seguinte, escolha uma senha para o usuário que está sendo criado, digite-a no campo apropriado e
 58. Informe a senha novamente para verificação, a mesma digitada no passo acima, e pressione a tecla **<ENTER>**.
 59. De volta ao menu principal do instalador, selecione a opção **"Instalar o sistema básico"** e pressione a tecla **<ENTER>**.
 60. Aguarde o término da instalação dos pacotes base e quando for questionado sobre o *kernel* que deve ser instalado escolha a opção **"linux-image-2.6-"** juntamente com a arquitetura mais adequada a seu processador. Por exemplo, para computadores com CPU AMD Athlon, você deve escolher **"linux-image-2.6-k7"**, já para processador Intel Pentium IV, o interessante é escolher **"linux-image-2.6-P4"**, e assim por diante.
 61. O próximo passo, perguntará **"Ferramenta a ser utilizada para gerar o initrd de inicialização"**, escolha a opção **"initramfs-tools"** e pressione a tecla **<ENTER>**.
 62. No menu principal do instalador escolha **"Configurar o gerenciador de pacotes"** e pressione a tecla **<ENTER>**.
 63. Na questão **"Usar programas não-livres"**, responda **"Sim"**.
 64. O APT fará uma atualização da lista de pacotes nesse momento. Caso apareça um erro com relação ao repositório **"security"**, com a mensagem "Não foi possível acessar as atualizações de segurança", escolha **<Continuar>**, e pressione a tecla **<ENTER>**.
 65. Novamente no menu principal, escolha a opção **"Selecionar e instalar o software"** e pressione a tecla **<ENTER>**.
 66. Na pergunta **"Participar do concurso de utilização de pacotes ?"**, responda **"Não"**.
 67. Na tela de seleção de pacotes para instalação, desmarque todas as opções, escolha **<Continuar>**, e pressione a tecla **<ENTER>**.
 68. O próximo passo, é instalar o gerenciador de *boot*. No menu principal escolha a opção **"Instalar o GRUB em um disco rígido"** e pressione a tecla **<ENTER>**.
 69. Em seguida, quando questionado se deseja instalar o gerenciador GRUB no registro principal de inicialização, responda **"Sim"**. Este passo pode demorar um certo tempo, tenha paciência.
 70. A próxima tela pergunta se você deseja colocar uma senha no GRUB. Deixe o campo em branco (sem digitar nada) e pressione a tecla **<ENTER>**.

71. Bom chegamos ao final da instalação. De volta ao menu principal do instalador, escolha a última opção ainda não visitada "**Finalizar a instalação**" e pressione a tecla **<ENTER>**. Quando a mensagem informando que a instalação acabou aparecer, pressione **<Continuar>**. Seu computador será reiniciado.

Alguns procedimentos logo após o computador ter reiniciado:

- 1) Faça *logon* como *root* no sistema.
- 2) Edite as configurações do APT através do comando "**vim /etc/apt/sources.list**" e deixe como abaixo:
deb www.repositorios.eparana.parana/debian/ etch main contrib non-free
deb www.repositorios.eparana.parana/celepar/ etch main contrib non-free
deb www.repositorios.eparana.parana/security/ etch/updates main contrib non-free
- 3) Execute o comando "**apt-get update**" para atualizar a lista de pacotes.
- 4) Execute o comando "**apt-get install basico-servidor**".
- 5) Execute o comando "**apt-get clean**".
- 7) Execute o comando "**apt-get --purge remove \$(deborphan)**".

3. Como Instalar Programas

Este capítulo tem o objetivo de explicar as possibilidades encontradas no GNU/Debian para instalação e manutenção de programas. Você poderá entender melhor o funcionamento dos gerenciadores de pacotes como o “APT” e “dpkg”, e também, como instalar programas a partir do código fonte.

3.1. APT

O principal instalador de pacotes do GNU/Debian é uma ferramenta do APT, o “*apt-get*”. Na maioria das vezes em que precisarmos instalar ou remover um pacote, é este utilitário que utilizaremos.

É freqüente falarmos em pacotes em vez de programas, quando o assunto é instalação. Isso vem da idéia de “empacotamento” de programas, que é a forma como cada distribuição organiza os programas que a constituem. No Debian, cada pacote termina com a extensão “.deb”, nas distribuições derivadas do RedHat a extensão é “.rpm”.

Alguns pontos sobre como usar o *apt-get* devem ser ressaltados:

- Apenas o usuário “root” pode instalar ou remover pacotes.
- Somente uma instância pode ser executada. A segunda tentativa de iniciar, simultaneamente, a instalação de um pacote irá gerar uma mensagem de erro.

Para instalar um pacote, o *apt-get* obedece a seguinte ordem:

- **Resolve dependências:** Verifica se o pacote requerido está disponível para instalação e quais as suas dependências. Caso não haja nenhum conflito com os pacotes instalados e suas dependências sejam sanadas é mostrado um relatório para o usuário solicitando que confirme as alterações que serão realizadas.
- **Baixa:** Ele acessa o *mirror* que você configurou e faz o *download* do pacote e suas dependências para o seu computador.
- O diretório “/var/cache/apt/archives” recebe os pacotes baixados.
- **Descompacta:** O pacote é descompactado para iniciar a instalação.
- **Configura:** Dependendo do pacote, antes de instalar, é necessário perguntar sobre algum tipo de preferência ao usuário.
- **Instala:** Todos os componentes do pacote são copiados para os diretórios devidos.

As principais ações que executaremos com este comando são:

- ***apt-get update*** - Existe uma lista, que contém uma descrição de todos os pacotes disponíveis para o Debian, disponível dentro de cada repositório de pacotes. O *apt-get* consulta esta lista, para saber o que pode ser instalado. Este comando faz uma comparação entre a lista que você tem armazenada e a lista que está no servidor de pacotes. Se a sua lista for mais velha, ele faz o *download* da mais recente. A atualização de um único pacote provoca o lançamento de uma nova lista.
- ***apt-get upgrade*** - Esta é a opção responsável pela atualização de pacotes. Quando você executa este comando, uma lista de pacotes é copiada do repositório para a máquina local do usuário, onde é feita uma comparação entre as versões de *software* existentes no repositório e na máquina local, caso existam versões mais recentes para os programas que o usuário possui em seu sistema, então, o APT automaticamente atualizará estes programas no computador do usuário. Não é preciso dizer qual pacote você quer atualizar, independente de ser apenas um, ou serem vários, ele fará toda atualização necessária.
- ***apt-get clean*** - Após a instalação de um pacote, não precisamos necessariamente, manter o arquivo “.deb” em nosso sistema (em “/var/cache/apt/archives”). O processo de instalação não remove os pacotes baixados! Se você não remover os pacotes baixados, começará a acumulá-los no disco rígido. Com o passar do tempo isso pode causar um problema de falta de espaço. O “*apt-get clean*” faz a remoção de todos os pacotes no diretório de cache do APT.

- **apt-get install PACOTE** - Executamos este comando para realizar a instalação de pacotes no sistema. Se quisermos instalar mais de um programa, basta escrever todos os nomes, separados por espaço na linha de execução do comando. Se um pacote precisa de outros para ser instalado, isto é, se ele tem pré-requisitos, eles também serão selecionados para instalação automaticamente. Quando você requisita a instalação de um pacote que não tem dependências, o *download* começa imediatamente. Caso existam dependências, elas são mostradas a você e o programa aguarda a sua confirmação ("Y/n") para continuar. Existem vários motivos para ele esperar por uma confirmação: a lista de dependências pode ser muito grande e você pode não querer instalar todos os pacotes, pode não haver espaço em disco suficiente para instalar o programa e/ou suas dependências, o pacote ser incompatível com outro já instalado e exigir que este seja removido, ou ainda, o repositório não é uma fonte confiável para o Debian, que acontece quando não existe na máquina a chave pública do repositório de onde vem a atualização. Existe ainda, um caso em que há mais de uma versão disponível para o mesmo pacote, se um pacote chamado "*nome*" possui uma versão "1.0-0" e "2.0-0", por exemplo, a versão mais nova será instalada por padrão, entretanto, podemos querer instalar a versão mais antiga (1.0-0), tendo que executar o comando da seguinte forma: `apt-get install nome=1.0-0`
- **apt-get dist-upgrade** - Possibilita a atualização completa da distribuição utilizada pelo usuário. É importante que usuário tenha em mente, que a execução deste comando num sistema que utilize meta-pacotes customizados, como é o caso do "Debian Desktop Paraná" usado na Celepar, ou versões de *software* instaladas manualmente (através de compilação do código fonte, por exemplo), poderão gerar problemas graves, impossibilitando inclusive o uso posterior do sistema. Para a devida utilização deste comando, é necessário a edição do arquivo de configuração do APT, o "`/etc/apt/sources.list`", configurando o mesmo para utilizar os repositórios da nova distribuição que será instalada no sistema.
- **apt-get remove PACOTE** - Remove um ou mais pacotes. Esta operação faz a desinstalação de um pacote do sistema e não a remoção do ".deb" do diretório de cache do APT. Em alguns casos, os arquivos de configuração e/ou dados do pacote são mantidos. Isso pode ser bom ou ruim. Se você removeu um pacote por acidente, todo o seu trabalho de configuração dele ainda estará preservado. Preste atenção às mensagens mostradas durante a remoção de um pacote para saber o que está acontecendo. Observe também que se algum pacote depende do pacote a ser removido, esse pacote também será removido.
- **apt-get --purge remove PACOTE** - Remove um ou mais pacotes e seus respectivos arquivos de configuração.
- **apt-get source PACOTE** - Possibilita que o código fonte de um programa seja copiado para máquina local em vez do binário do programa e seus *scripts* de configuração, que compõem o pacote do programa propriamente dito. Para baixar arquivos contendo o fonte de programas para sua máquina, é necessário configurar adequadamente o arquivo "`/etc/apt/sources.list`", para isso veja o tópico "Reconfigurando a lista de pacotes", na sequência.
- **apt-get build-dep PACOTE** - Este comando, tenta satisfazer as dependências, que um determinado pacote contendo código fonte de um programa possui, para ser compilado. Por exemplo, suponha que você deseja recompilar o código fonte do "tuxracer" (um joguinho), se você executar o comando "`apt-get build-dep tuxracer`", o APT tentará instalar todos os pacotes que sejam necessários para compilação do pacote "tuxracer". Em algumas situações, é necessário instalar pacotes além daqueles que são trazidos automaticamente pelo comando "`apt-get build-dep`". Isto é algo natural e você deverá se acostumar a estas situações.
- **apt-get check** - É uma ferramenta de diagnóstico. Ele atualiza o *cache* de pacotes e verifica se há alguma dependência quebrada que necessita ser resolvida (utilizando-se "`apt-get -f install`") no sistema.
- **apt-get install --reinstall PACOTE** - Em algumas situações, precisaremos reinstalar um pacote já presente no sistema. Este é o comando que permite fazer isso. Tome o cuidado de fazer uma cópia dos seus arquivos de configuração para não ter surpresas desagradáveis.

Você deve estar se perguntando, como fazer para saber o que instalar, ou qual o nome de um determinado programa (pacote) no Debian. Para responder a essas dúvidas, vamos apresentá-lo aos dois programas criados para resolver este problema, e seus parâmetros:

- ***apt-cache search NOME*** - Para procurar por um pacote qualquer, executamos este comando. Ele faz uma pesquisa na lista de pacotes disponíveis, procurando pacotes que possuam a expressão fornecida no argumento "NOME". A saída deste comando, apresenta todos os pacotes que apresentam a palavra fornecida como argumento em parte do nome ou na descrição do pacote. Existem formas mais complexas de busca que podem ser mais claras ou mais específicas, mas geralmente, a utilização do comando "grep" para filtrar a saída já é suficiente para grande maioria dos casos.
- ***apt-cache show PACOTE*** - Uma vez descoberto o nome correto do pacote Debian, você pode visualizar todas as informações sobre o pacote, bem como seu mantenedor, sua versão, sua descrição, dentre outras informações relevantes.
- ***apt-cache showsrc PACOTE*** - Tem a mesma função que "apt-cache show", porém, atua mostrando informações sobre pacotes fontes.
- ***synaptic*** - É uma interface gráfica para a instalação de pacotes. Ela também é utilizada em outras distribuições, o Ubuntu é uma delas.

3.1.1. Reconfigurando a lista de pacotes

Os pacotes que instalaremos em nosso sistema estão disponíveis em servidores ao redor do mundo. Estes servidores contêm todos os pacotes disponíveis para o Debian. Eles são criados com uma técnica de espelhamento e, graças a isto, podemos baixar os pacotes do servidor mais próximo de nós, diminuindo o tempo de instalação.

A qualquer momento podemos reconfigurar a lista de servidores que utilizaremos através da edição do arquivo de configuração "/etc/apt/sources.list", usando o seu editor de texto preferido. A configuração padrão para os computadores GNU/Debian versão etch da rede do Estado do Paraná é apresentada abaixo:

```
deb http://www.repositorios.eparana.parana/debian/ etch main contrib non-free
deb http://www.repositorios.eparana.parana/celepar/ etch main contrib non-free
deb http://www.repositorios.eparana.parana/marillat/ etch main
deb http://www.repositorios.eparana.parana/security/ etch/updates main contrib non-free
```

Na configuração descrita acima, estamos apontando o APT para os repositórios internos da Celepar.

A ordem das linhas faz diferença, o primeiro servidor a ser consultado é o da primeira linha e assim por diante. Assim, devemos colocar o que está mais próximo no início do arquivo. Também não pode haver espaço entre o início da linha e a palavra "deb".

Cada linha representa uma lista que deverá ser copiada para o seu sistema.

Se você encontrar alguma linha que comece com "deb-src", poderá comentá-la acrescentando o símbolo "#" no início da linha. Este tipo de linha, serve para baixar o código fonte de um programa.

Lembre-se que os pacotes com extensão ".deb", já foram compilados para uma arquitetura específica, por isso, é só baixar e instalar. Quando você necessitar compilar um programa, e neste caso precisar do seu código fonte, uma linha como a que é mostrada abaixo deve ser inserida no arquivo de configuração do APT:

```
deb-src http://www.repositorios.eparana.parana/debian/ etch main
```

3.1.2. Após reconfigurar a lista de pacotes

Todas as vezes que você alterar o arquivo "/etc/apt/sources.list" deverá atualizar a(s) lista(s) existentes no seu sistema, fazendo:

```
apt-get update
```

Caso este procedimento não seja obedecido, e você tente instalar um pacote disponibilizado recentemente no repositório, receberá uma mensagem de erro.

3.1.3. O arquivo de configuração “apt.conf”

O arquivo “/etc/apt/apt.conf”, contém a configuração principal a ser utilizada pelas ferramentas do APT. Sempre que uma ferramenta do grupo APT é iniciada, ela irá ler a variável de ambiente “APT_CONFIG”, e posteriormente, a configuração deste arquivo para se ajustar as preferências do usuário.

Todas as opções que são passadas pelo usuário através da linha de comandos, sobrescrevem as configurações contidas no arquivo “apt.conf”.

O arquivo “apt.conf” é organizado sob a forma de árvore em divisões funcionais, como por exemplo: o grupo “APT”, o grupo “Acquire”, o grupo “DPKG” e a seção “Dir” (para diretórios de configuração). Sintaticamente, o arquivo pode ser configurado sob as seguintes formas:

```
APT::Get::Assume-Yes "true";
```

ou

```
APT {  
  Get {  
    Assume-Yes "true";  
  };  
};
```

Nos exemplos acima, podemos perceber que tanto os caracteres “::”, quanto “{“ e “}”, são utilizados para definir qual o grupo funcional, seção e opções estamos tratando naquele espaço do arquivo de configuração. Em ambos exemplos que citamos (acima), estamos tratando da opção “Assume-Yes”, da seção “Get”, do grupo funcional “APT”, contudo, de formas sintaticamente diferentes, mas produzindo o mesmo resultado.

Linhas que se iniciam com os caracteres “/”, ou um bloco de texto, que se inicia com “/*” e termina com “*/”, são considerados comentários e são ignorados.

A seguir explicaremos a finalidade de cada grupo funcional, que pode ser configurado através do “apt.conf”:

- O grupo “APT”
É utilizado para controlar as opções gerais para uso de todas as ferramentas do APT.
- O grupo “Acquire”
É divisão funcional usada para configurar as rotinas de *download* de pacotes e manipuladores de URI's utilizados pelas ferramentas do APT.
- O grupo “Dir”
Definição de *layout* de diretórios. Usado para a configuração dos diretórios que serão utilizados pelo APT.
- O grupo “Dselect”
Configurações que afetam o uso da ferramenta “dselect”, quando ela é utilizada em conjunto com o APT.
- O grupo “DPKG”
Configurações que afetam o uso do “dpkg”. As opções que fazem parte desta seção, alteram a forma normal como o dpkg atua, quando invocado pelo APT.
- O grupo “Debug”
Utilizado para configurar opções para depuração no uso das ferramentas do APT.

Cada um destes grupos funcionais, possui uma gama própria de opções, que não discutiremos aqui, por ser muito vasta. Se você precisar saber qualquer detalhe sobre as opções que podem ser utilizadas neste arquivo de configuração, estude a documentação oficial do arquivo, através do comando “**man apt.conf**”.

Exemplos:

Exemplo 01: Configurando um serviço de proxy para ser utilizado pelo APT. Esta configuração, possibilita buscar pacotes de programas na Internet através de um servidor proxy existente na sua rede.

```
Acquire
{
    http
    {
        Proxy "http://usuario:senha@proxy.organizacao:8080/";
    };
};
```

Exemplo 02: Configurando a arquitetura e *release* (versão) padrão dos pacotes a serem utilizados pelo sistema.

```
APT
{
    Architecture "i386";
    Default-Release "stable";
};
```

Exemplo 03: Configurando o diretório de cache do APT.

```
Dir "/"
{
    Cache "var/cache/apt/"
    {
        Archives "archives/";
        srcpkgcache "srcpkgcache.bin";
        pkgcache "pkgcache.bin";
    };
};
```

Um ótimo arquivo de exemplos de configuração do "apt.conf" pode ser encontrado em "/usr/share/doc/apt/examples/configure-index.gz".

3.1.4. O arquivo de configuração "preferences"

O arquivo /etc/apt/preferences controla a prioridade de versão de pacotes. É possível manter fontes stable, testing e unstable ao mesmo tempo. Caso tenha inserido no arquivo /etc/apt/sources.list repositórios de versões diferentes (stable e testing por exemplo) é necessário criar o arquivo preferences para que possa na instalação selecionar qual versão terá prioridade e de qual versão você deseja instalar os pacotes.

obs.: Muito cuidado quando estiver trabalhando com versões diferentes, uma má instalação de pacote pode gerar quebras e até mesmo a paralisação do sistema.

Segue um exemplo para o arquivo preferences:

```
Package: *
Pin: release a=stable
Pin-Priority: 900

Package: *
Pin: release a=testing
Pin-Priority: 400
```

Neste exemplo para todos os repositórios stable, a prioridade será 900 e para testing 400, ou seja, vale a prioridade mais alta, então o stable será a opção default, sempre que utilizarmos o comando apt-get com as opções install, update, upgrade, source, dist-

upgrade, o pacote será do stable.

Para instalar pacotes do testing devemos utilizar o comando:

```
# apt-get -t testing install [pacote]
```

O mesmo procedimento se aplica as opções update, upgrade, dist-upgrade ou source.

Podemos definir que um pacote sempre venha da release testing, para isso dentro do arquivo `/etc/apt/preferences`, por exemplo:

```
Package: amule  
Pin: release a=testing  
Pin-Priority: 905
```

O pacote desejado vai ter prioridade maior que o de qualquer outro release, então o pacote desejado será da release testing.

3.2. dpkg

É importante lembrar do comando “dpkg”. Ele também instala, remove e reconfigura pacotes. O dpkg vem aos poucos perdendo usuários para o APT, uma vez que este último, proporciona maiores facilidades de utilização, sendo a mais significativa dessas facilidades, o fato de instalar o *software* e suas dependências automaticamente, algo que ainda não é possível com o dpkg.

Mesmo com o avanço do APT, é muito importante estudar e entender o funcionamento e as opções que o dpkg possui. Isto porque, o dpkg continua sendo a principal ferramenta de obtenção de informações sobre os pacotes instalados no sistema, e em vários casos, de resolução de problemas com pacotes. Por essa razão, explicamos o uso desta ferramenta e de suas principais opções na seção “Comandos de Uso Geral”, do capítulo “Comandos Avançados”, deste documento.

3.3. Compilando Programas a Partir do Código Fonte

Embora atualmente, a maioria dos programas para o GNU/Debian estejam disponíveis para instalação via pacotes, ainda assim, em alguns casos é preciso compilar programas a partir de seu código fonte. Este procedimento, será descrito no tópico a seguir.

3.3.1. Arquivos tarball

De modo geral, códigos fonte são distribuídos através de arquivos compactados. A grande maioria destes arquivos, possui uma estrutura de diretórios, os arquivos fonte em si, um arquivo de configuração chamado “*Makefile*”, documentação e outros arquivos encapsulados em um arquivo no formato “tar”, com compressão de dados, usando gzip (“*.tar.gz*” ou “*.tgz*”), ou bzip2 (“*.bz2*” ou “*.tbz2*”).

Para iniciar a compilação de um código fonte, a primeira coisa a se fazer é obter o código para ser compilado a partir da Internet, pacote ou outro meio qualquer. Após o pacote estar disponível localmente, você deverá extrair o conteúdo para um diretório adequado a compilação (ex.: `/install`). É possível extrair um pacote utilizando o “tar” em conjunto com utilitário de compressão como gzip ou bzip2, da seguinte maneira:

```
tar -xvzf nome_do_arquivo.tar.gz  
ou  
tar -xvjf nome_do_arquivo.tar.bz2
```

Após a extração do conteúdo do arquivo compactado, é necessário entrar no diretório que foi criado durante a extração dos arquivos utilizando o comando “cd”. O programa mais utilizado, para compilar códigos fonte no GNU/Linux é o “gcc”, um compilador C/C++ livre. De maneira geral, alguns pacotes GNU são necessários para compilar um programa de código livre. Os mais comuns na distribuição Debian são os

pacotes: “make”, “autoconf”, “gcc”, “g++” e “libc6-dev”.

Abaixo listamos os procedimentos gerais para configuração/compilação de códigos fontes no GNU/Debian. Lembre-se que este procedimento pode variar um pouco dependendo do código que se está compilando.

- **O arquivo “configure”** – Este arquivo é um *script shell* que examina o sistema para verificar se diversas dependências necessárias para compilar o projeto serão satisfeitas. Ele deve ser executado digitando-se “./configure” dentro do diretório que contém o código-fonte do programa a ser compilado. Esse *script*, também pode conter parâmetros que podem ser passados na linha de execução para configuração de opções específicas, para consultar a lista de parâmetros disponíveis, tente executar o *script* “./configure --help”. Se o *script* “configure” não encontrar alguma das dependências necessárias para compilação do programa, um erro é gerado e a execução é finalizada automaticamente, gerando uma mensagem semelhante à abaixo:

```
checking for SDL - version >= 1.2.0... no
configure: error: *** SDL version 1.2.0 not found!
```

Nesse caso, nos falta a biblioteca de desenvolvimento (cabeçalhos) “SDL” versão 1.2.0 ou mais recente. Caso quiséssemos resolver este problema, precisaríamos instalar o pacote “libsdl1.2-dev” para resolver essa dependência. A maioria das bibliotecas necessárias para a compilação de programas no GNU/Debian têm o prefixo “lib” e o sufixo “-dev”, como por exemplo, “libc6-dev”, “libsdl1.2-dev”, etc.

- **make** – O comando “make” utiliza as configurações que foram criadas pelo *script* “configure” para compilar múltiplos arquivos de código fonte de um projeto. Utiliza também um arquivo de descrição (chamado “makefile” ou “Makefile”) que está presente no diretório onde o código fonte foi extraído. Seu conteúdo é composto de regras que definem as dependências entre arquivos fonte e os comandos necessários para a compilação. A partir deste arquivo, ele executa seqüências de comandos que são interpretados pelo *shell* para realizar a compilação de maneira correta.
- **make install** – É o comando utilizado para compilar e em seguida executar o instalador do programa que acabamos de compilar. Alguns programas não possuem essa facilidade, nestes casos, o administrador terá também o trabalho manual de instalar o programa compilado corretamente, em geral, realizando atividades como copiar arquivos executáveis para diretórios como “/usr”, e os arquivos de configuração para “/etc”, etc.

4. Gerenciando Discos e Partições

Neste capítulo, mostraremos como listar e obter informações a respeito dos discos rígidos que o sistema possui, criar, remover e formatar partições.

4.1. Obtendo Informações sobre os Discos Rígidos

A forma mais rápida e fácil de se obter informações relacionadas aos discos rígidos, que um determinado sistema GNU/Linux possui, é executando o utilitário “fdisk” em conjunto com o parâmetro “-l”, conforme é mostrado abaixo:

```
fdisk -l
```

A execução do comando exibirá informações a respeito de todos os discos rígidos que o sistema possui. A seguir, um exemplo da saída do comando, em um sistema com dois discos rígidos IDE:

```
# fdisk -l
```

```
Disk /dev/hda: 20.0 GB, 20020396032 bytes
255 heads, 63 sectors/track, 2434 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/hda1		1	62	497983+	82	Linux swap / Solaris
/dev/hda2	*	63	93	249007+	83	Linux
/dev/hda3		94	1066	7815622+	83	Linux
/dev/hda4		1067	2434	10988460	83	Linux

```
Disk /dev/hdb: 80.0 GB, 80020396032 bytes
255 heads, 63 sectors/track, 8420 cylinders
Units = cylinders of 16065 * 512 = 16025280 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/hdb1		1	62	497983+	82	Linux swap / Solaris
/dev/hdb2		63	8420	729007+	83	Linux

Como podemos ver através do exemplo acima, várias informações podem ser obtidas facilmente, como: números de discos rígidos que o sistema possui, tamanho dos discos, nome de cada dispositivo, partições que cada disco tem e seus respectivos atributos, enfim, uma série de dados que ajudam o administrador a administrar seus dispositivos de armazenamento.

Vale ressaltar que caso seu sistema possua discos utilizando técnicas de RAID via *hardware*, quando o comando for executado, os discos serão mostrados de acordo com as configurações RAID que estiverem em vigor no seu sistema. Por exemplo, numa configuração em RAID 1 (*mirroring*), apesar de existirem fisicamente dois discos na máquina, apenas um será mostrado com “fdisk -l”.

Nos próximos tópicos, explicaremos como utilizar outro utilitário de particionamento de discos. Caso você tenha despertado interesse pelo uso do fdisk, dê uma olhada em sua documentação com o comando “man fdisk”.

4.2. Particionando o Disco com o cfdisk

Para iniciar o particionador, como root, digite “cfdisk” seguido do nome do dispositivo (HD) a ser particionado, por exemplo “**cfdisk /dev/hda**”. Lembre-se, “/dev/hdX” é utilizado para HD's IDE, já “/dev/sdX” é usado para discos SCSI, SATA e Pen-drives.


```

cfdisk 2.12r

Disco: /dev/hda
Size: 80026361856 bytes, 80.0 GB
Heads: 255 Sectors per Track: 63 Cylinders: 9729

Nome      Opções      Tipo Part. Tipo SA      [Rótulo]      Size (MB)
-----
hda1      Inicializar Primária Linux ext3      131,61
hda2      Primária Linux XFS      10001,95
hda3      Primária Linux swap / Solaris 1003,49
hda4      Primária Linux XFS      68886,72

[Iniciali.] [Excluir ] [ Ajuda ] [Maximize] [ Mostre ]
[ Sair ] [ Tipo ] [Unidades] [ Gravar ]

Alterna a opção da partição atual como inicializável

```

Figura 4 - Menu principal do utilitário cfdisk.

As opções que utilizaremos no cfdisk estão descritas abaixo:

- **Nova** – Cria uma nova partição, primária ou lógica, definida por um tamanho menor ou igual ao tamanho disponível no HD. A partição primária, é o tipo de partição básica, definida por apenas um tipo de sistema de arquivos em que nenhuma “sub-partição” pode ser criada (por padrões internacionais, apenas 4 partições primárias podem ser criadas por HD). Uma partição lógica, é uma partição primária, em que várias “sub-partições” podem ser criadas (extendendo-se o limite de 4 partições por HD).
- **Excluir** – Remove uma partição já existente no disco, deixando a área do HD que continha a partição com espaço livre, para que uma ou mais partições possam ser criadas em seu lugar.
- **Tipo** – Exibe uma lista com os tipos de partição que podem ser criados e ao final define o tipo de partição a ser usado.
- **Inicializável** – Define uma partição inicializável (bootável).
- **Gravar** – Salva as alterações feitas na tabela de partições do disco.
- **Ajuda** – Exibe a ajuda do cfdisk.

4.3. Criando uma Partição Primária

Para criar uma nova partição, devemos, primeiramente, escolher através das setas para cima e para baixo do teclado uma área do HD que esteja livre, deixando-a marcada. Feito isso, com as setas direita e esquerda do teclado, deve-se escolher a opção “Nova”.

Agora devemos escolher a opção “Primária”.

A próxima opção é definir o tamanho da partição, sendo que o tamanho máximo que pode ser usado já está pré-definido, mas podemos mudá-lo apenas digitando o novo valor em MB (não precisamos apagar o valor que já está no campo, basta apenas digitar o novo valor). Por exemplo, se temos 2GB livres e quisermos criar duas partições de 1GB, devemos digitar o valor “1024” neste campo e, então, repetir o processo de criação de partição.

Quando todas as alterações estiverem feitas, devemos selecionar a opção “Gravar”, para que as alterações sejam efetivamente salvas no HD. Se não selecionarmos a opção de gravação e sairmos do cfdisk, nenhuma alteração será feita.

4.4. Criando uma Partição Lógica

Partições lógicas, são utilizadas quando necessitamos particionar o disco em mais de 4 partes, que poderiam ser feitas apenas com partições primárias, isto porque, as partições lógicas possibilitam expandir o número total de partições do disco bem além do total permitido nas primárias. Devido à isso, normalmente deixamos as partições lógicas no final do HD, onde possam ser criadas de acordo com a quantidade necessária que precisarmos.

OBS.: Se 4 partições primárias forem criadas, e a soma das 4 não for igual ao tamanho total do HD, não poderemos mais particionar o espaço restante, onde aparecerá a mensagem “Inutilizável” no lugar de “Espaço livre”.

Selecionando o bloco do disco onde ainda existe a mensagem “Espaço livre”, escolheremos a opção “Nova”.

Quando formos perguntados se a partição é primária ou lógica, escolheremos a opção “Lógica”.

Definimos o tamanho da nova partição, digitando o valor do tamanho da partição em MB.

Você poderá repetir este procedimento para criar quantas partições forem necessárias no seu caso, e quando todas as alterações estiverem sido feitas, devemos selecionar a opção “Gravar”, para que as alterações sejam efetivamente salvas no HD. Caso você não faça isso, nenhuma alteração será feita no particionamento do disco.

4.5. Excluindo uma Partição

Para remover uma partição, basta selecioná-la e utilizar a opção “Excluir”. Este procedimento não pergunta se você realmente deseja remover a partição, como na maioria dos outros programas faz, mas não há motivo para se preocupar, se alguma partição for excluída por engano, basta sair do utilitário sem selecionar a opção “Gravar”, que as alterações não serão salvas na tabela de particionamento do disco.

4.6. Salvando as Alterações no Disco

Quando todas as alterações forem concluídas, devemos selecionar a opção “Gravar”, para que elas sejam efetivamente salvas no HD. Se não selecionarmos a opção de gravação e sairmos do cfdisk, nenhuma alteração será salva, e todo o trabalho terá que ser refeito.

CUIDADO: Ao selecionar a opção “Gravar” as alterações não poderão ser desfeitas, portanto, verifique cuidadosamente se as alterações estão mesmo corretas antes de gravá-las no disco.

4.7. Criando o Sistema de Arquivos

Ensinaresmos duas formas para se criar o sistema de arquivos em uma partição, uma delas é utilizando o próprio cfdisk e a outra, utilizando uma ferramenta chamada “mkfs”.

Para definirmos o tipo de uma partição, e conseqüentemente, criá-la automaticamente, devemos selecionar a opção “Tipo” do cfdisk, e em seguida, escolher uma das opções de sistemas de arquivo disponíveis. Depois disso, resta gravar as mudanças através da opção “Gravar” e a partição estará criada e já com o sistema de arquivos selecionado disponível para o uso.

Podemos também criar uma partição sem definir o tipo dela, entretanto, para que uma partição fique funcional, temos obrigatoriamente que criar um sistema de arquivos em seu espaço. Quando criamos apenas o espaço da partição sem definir um sistema de arquivos a ser utilizado naquele espaço, podemos empregar um outro método para criar o sistema de arquivos naquela partição. Para isso, basta utilizar o utilitário mkfs especificando o tipo de sistema de arquivos que desejamos utilizar na partição em questão. Abaixo um exemplo de uso deste utilitário:

```
mkfs -t xfs /dev/hda1
```

No exemplo acima, criamos um sistema de arquivos do tipo “xfs” na partição “/dev/hda1”. Essa é a maneira mais simples de se criar um sistema de arquivos em uma partição, após ela ter sido criada sem um tipo definido. Também podemos usar as extensões do utilitário mkfs, para realizar a criação de sistemas de arquivos específicos. Abaixo um exemplo de utilização de extensões que faz exatamente a mesma operação que o exemplo citado anteriormente:

```
mkfs.xfs /dev/hda1
```

5. Gerenciador de Partida - GRUB

O Gerenciador de Partida é um programa que carrega um Sistema Operacional e/ou permite escolher qual será iniciado. Normalmente, este programa é gravado no setor de *boot* de uma partição ativa ou no *Master Boot Record (MBR)* do disco rígido.

O **GRUB (Grand Unified Boot Loader)**, é mais uma alternativa de gerenciador de inicialização, e apresenta alguns recursos extras com relação as outras opções disponíveis. Ele é flexível, funcional e poderoso, podendo inicializar sistemas operacionais como o Windows (9x, ME, NT, 2000 e XP), Dos, Linux, GNU Hurd, *BSD, OS/2 etc. Podemos destacar também o suporte aos sistemas de arquivos EXT2, EXT3 e ReiserFS, FAT16 e FAT32 (Win 9x/ME), FFS (*Fast File System* usado no *BSD), minix (MINIX OS) etc.

Por utilizar o padrão *Multiboot*, ele é capaz de carregar diversas imagens de inicialização (uma por vez) e módulos. Por esse motivo, ele é o único gerenciador de inicialização capaz de carregar o conjunto de servidores do GNU Hurd. O *GRUB* também permite buscar imagens do *Kernel* pela rede, por cabo seriais, suporta discos rígidos IDE, SATA e SCSI, informar a quantidade total de memória RAM ao sistema, tem interface voltada para linha de comandos ou menus de escolha, além de suportar sistemas sem discos e terminais remotos.

Como possui inúmeros recursos, será apresentada sua utilização básica, ficando como sugestão ao leitor procurar se aprofundar mais em suas possibilidades de uso e configuração.

5.1. Como o GRUB Trabalha com Discos e Partições

O *GRUB* trabalha com uma notação diferente para apontar discos e partições sendo necessário algumas explicações antes de prosseguir. Veja a tabela comparativa:

Disco IDE		Disco SCSI ou SATA		Disquete	
Dispositivo no Linux	Dispositivo no GRUB	Dispositivo no Linux	Dispositivo no GRUB	Dispositivo no Linux	Dispositivo no GRUB
		/dev/sda	(hd0) # Disco SCSI ID 0		
/dev/hda	(hd0)	/dev/sda1	(hd0,0) # Disco SCSI ID 0, partição 1		
/dev/hda1	(hd0,0)	/dev/sda2	(hd0,1) # Disco SCSI ID 0, partição 2		
/dev/hda2	(hd0,1)	/dev/sdb	(hd1) # Disco SCSI ID 1		
/dev/hdb	(hd1)	/dev/sdb1	(hd1,0) # Disco SCSI ID 1, partição 1		
/dev/hdb1	(hd1,0)	/dev/sdb2	(hd1,1) # Disco SCSI ID 1, partição 2		
/dev/hdb2	(hd1,1)			/dev/fd0	(fd0)

OBS: Os discos IDE, SCSI e SATA são referenciados ambos como (hd?) pelo *GRUB*.

5.2. Instalando o GRUB

A instalação do *GRUB* só precisa ser executada uma única vez. Caso seja necessária alguma mudança como por exemplo adicionar uma nova imagem, esta pode ser feita apenas editando o arquivo de configuração "*menu.lst*", com o comando "vim /boot/grub/menu.lst".

Normalmente, durante a instalação do GNU/Debian você será questionado se deseja instalar o *GRUB*, o que torna o processo mais prático, uma vez que ao ser instalado ele reconhecerá automaticamente os demais Sistemas Operacionais instalados na sua estação e os incluirá na lista de sistemas disponíveis.

Um método simples para adicionar o *GRUB* para gerenciar seu *MBR*, é executar o comando:

```
/sbin/grub-install /dev/hda
```

Este comando grava o *GRUB* no *MBR* do primeiro disco rígido IDE do sistema e cria o diretório */boot/grub*, onde estarão os arquivos necessários para o seu funcionamento. É interessante executar o comando *“update-grub”*, que atualiza dinamicamente a lista de sistemas que podem ser carregados durante a inicialização da máquina, e os adiciona ao arquivo *“menu.lst”* automaticamente, conforme a instalação corrente.

Também é conveniente, ler o arquivo de configuração de exemplo do *GRUB* e otimizá-lo às suas necessidades. Para checar este arquivo, utilize o seguinte comando:

```
vim /usr/share/doc/grub/examples/menu.lst
```

5.3. No Disco Flexível (com interface de menu)

Pode haver um momento, em que você deseje ter um disquete, que funcione com vários sistemas e não dependa de um disco fixo. Neste caso, você poderá armazenar suas configurações do *GRUB* em um disco flexível (disquete). Para criar um disquete de inicialização, siga os passos abaixo:

1. Com a privilégios de administrador do sistema (root), crie um sistema arquivos em um disquete com o comando: `mke2fs /dev/fd0`
2. Monte o disquete que você acabou de formatar: `mount /dev/fd0 /floppy -t ext2`
3. Crie um diretório para o *GRUB*: `mkdir /floppy/grub`
4. Copie os arquivos necessários para inicialização, com os seguintes comandos:

```
cp /usr/lib/grub/i386-pc/stage[12] /floppy/grub
cp /usr/share/doc/grub/examples/menu.lst /floppy/grub
```
5. Desmonte o disquete: `umount /floppy`
6. Execute o *GRUB*: `/sbin/grub`
7. Este último comando (passo 6), disponibiliza a linha de comando do *GRUB*. Nela, digite os seguintes comandos:

```
grub> install (fd0)/grub/stage1 d (fd0) (fd0)/grub/stage2 p (fd0)/
grub/menu.lst
grub> quit
```

Neste momento, o disquete estará pronto. Note que o arquivo *“menu.lst”* que foi copiado para ele, é um arquivo de exemplo, sendo necessário que você o configure de acordo com suas necessidades.

5.4. Opções do Arquivo de Configuração

Esta seção descreve o arquivo *“menu.lst”* com explicações sobre as opções mais usadas. Este arquivo é dividido em parâmetros Globais, que afetam o arquivo todo e parâmetros que só tem efeito para as imagens do sistema que será carregado. Algumas opções podem ser passadas para o *kernel* do Linux no momento do boot, algumas delas também serão detalhadas.

5.4.1. Parâmetros globais

- **timeout** = Define um tempo (em segundos) de espera. Se nenhuma tecla for pressionada, carrega a imagem padrão.
- **default** = Define qual será a opção padrão que deve ser automaticamente selecionada quando nenhuma outra for especificada em um tempo definido pelo parâmetro *“timeout”*.
- **fallback** = Caso ocorra algum erro inesperado e a opção padrão não possa ser carregada, este parâmetro define qual a outra opção deve ser utilizada.
- **color** = Permite que você escolha as cores usadas no menu de *boot*.
- **password** = Permite que você especifique uma senha. Está será solicitada sempre que houver necessidade de realizar uma função que não seja carregar as imagens disponíveis, como por exemplo acessar a linha de comandos do *GRUB*. Você pode

utilizar também o parâmetro “*password*” para esconder um arquivo que contenha outras configurações, como um arquivo “*menu.lst*” secreto. O arquivo pode ter um nome qualquer.

Ex.: *password = senha (hd0,0)/boot/grub/secret.conf*

É possível ter várias entradas do parâmetro *password* em um mesmo arquivo de configuração, sendo que uma delas é usada para bloquear o acesso as imagens/linha de comandos, e as outras, usadas para carregar arquivos de opções do *GRUB*. Durante a inicialização do sistema, você poderá digitar a tecla “p” para entrar com a senha que protege as imagens ou linha de comandos do *GRUB*, conforme a sua configuração.

- ***hiddenmenu*** = Está opção faz com que o menu de opções não seja mostrado, e a inicialização seja feita pela imagem definida pelo parâmetro “*default*”, depois de expirado o tempo no parâmetro “*timeout*”. O usuário pode requisitar o menu com as opções pressionando a tecla <ESC> antes que o tempo definido em *timeout* expire.

5.4.2. Parâmetros que afetam apenas as imagens

- ***title*** = Define um texto que será apresentado no menu de *boot* para identificar o sistema a ser inicializado.
- ***root*** = Determina qual é a partição raiz do sistema a ser inicializado.
- ***rootnoverify*** = Idêntico ao parâmetro anterior (*root*), mas não tenta montar a partição-alvo, o que é necessário para alguns sistemas como o DOS e o MS Windows.
- ***kernel*** = Nesta opção, você informa qual o *kernel* que será inicializado. Também é possível passar parâmetros diretamente para o *kernel* que será carregado.
Ex.: *kernel (hd0,0)/boot/vmlinuz-2.4.16 vga=791*
- ***module*** = Faz com que algum módulo necessário para o *boot* seja carregado. Lembre-se que estes não são módulos do *kernel* (módulos de som, rede, etc.) e sim módulos necessários ao *boot* de alguns sistemas, como por exemplo os utilizados pelo GNU Hurd.
- ***lock*** = Quando você deseja controlar se uma pessoa pode ou não iniciar um sistema que esteja listado nas opções do menu de *boot*, você pode utilizar esta opção que faz com que a senha especificada com o parâmetro “*password*” seja solicitada no momento em que o usuário tentar carregar a imagem em questão.
- ***pause*** = Emite uma mensagem na tela e espera uma tecla ser pressionada.
- ***makeactive*** = Torna a partição ativa. Este comando está limitado as partições primárias dos discos.
- ***chainloader*** = Alguns sistemas como o Windows ou o Dos, armazenam seu próprio gerenciador de *boot* no início da partição em que ele está instalado. Para efetuar a carga destes sistemas através do *GRUB*, você precisa pedir para que o gerenciador de inicialização de tal sistema seja carregado e faça seu trabalho, iniciando o sistema em questão.
- ***hide e unhide*** = Esconde/mostra uma partição respectivamente. Estas duas opções, são necessárias quando houver mais de uma versão do DOS ou do Windows na máquina em partições diferentes, já que estes sistemas detectam automaticamente a partição. Vamos a um simples exemplo para ilustrar uma situação bem comum: Suponha que o Windows esteja instalado na primeira partição primária do primeiro disco rígido (*hd0,0*) e o DOS na segunda partição primária (*hd0,1*). Quando quisermos carregar estes sistemas, devemos ajustar o arquivo “*/boot/grub/menu.lst*”, adicionando as seguintes configurações:

Porção do arquivo “*/boot/grub/menu.lst*”

```
title Windows
hide (hd0,1)
unhide (hd0,0)
rootnoverify (hd0,0)
chainloader +1
makeactive
```

```
title Dos
```

```
hide (hd0,0)
unhide (hd0,1)
rootnoverify (hd0,1)
chainloader +1
makeactive
```

- **map** = Alguns sistemas não permitem ser iniciados quando não estão no primeiro disco (DOS, Windows 9x etc). Para resolver esta e outras situações deste tipo, o *GRUB* tem um comando que permite enganar tal sistema mapeando as unidades de disco do modo como lhe for mais conveniente.

Imagine que você tenha o primeiro disco (*hd0*) com o GNU/Linux instalado e em um outro disco (*hd1*) com o Windows/DOS instalado. O Windows/DOS não permitem serem inicializados desta forma, e como solução, você poderia usar a seguinte entrada no arquivo de configurações “/boot/grub/menu.lst” do *GRUB*:

```
title Windows
unhide (hd1,0)
rootnoverify (hd1,0)
chainloader +1
map (hd1) (hd0)
makeactive
```

Isso faz com que o disco (*hd1*), onde Windows/DOS está instalado, seja apresentado a este sistema como (*hd0*), um artifício que permitirá que estes sistemas sejam carregados normalmente.

5.4.3. Parâmetros enviados diretamente ao kernel

Pode ser necessário passar alguns parâmetros para o *kernel* no momento do carregamento do SO. Você poderá fazer isso, editando as linhas do arquivo “/boot/grub/menu.lst”, conforme o exemplo a seguir:

```
# Exemplo de entrada no “menu.lst”
title Linux 2.6.18-5
root (hd0,0)
kernel (hd0,0)/boot/vmlinuz-2.4.16 vga=791 mem=512M ramdisk=0
```

Neste exemplo, a linha com o comando “kernel” é usada para indicar qual imagem deve ser carregada. As opções que seguem (“vga”, “mem” e “ramdisk”), são parâmetros que devem ser passados diretamente ao *kernel* do sistema a ser carregado, no momento de sua inicialização.

5.5. Usando a Linha de Comandos do *GRUB*

O *GRUB* possui inúmeros recursos, mas com certeza um dos mais importantes e que merece maior destaque, é sua linha de comandos. A maioria dos comandos usados no arquivo de configuração “/boot/grub/menu.lst”, são válidos aqui e muitos outros estão disponíveis. Uma breve apresentação da linha de comandos será dada, ficando por conta do leitor se aprofundar o quanto achar necessário.

Quando o *GRUB* é carregado, você pode se deparar com sua linha de comandos, ou se possuir um arquivo de configuração, um menu de escolha. Mesmo usando os menus de escolha, podemos utilizar todo o poder da linha de comandos do gerenciador de partida, bastando para isso, seguir as instruções que são mostradas no rodapé da tela do menu de escolha do *GRUB*.

Caso o *GRUB* seja carregado usando o menu de escolha, podemos digitar “e” para editar as entradas que estejam selecionadas no menu ou “c” para ter acesso a linha de comandos (lembre-se que pressionar <ESC> faz com que você volte aos menus de escolha).

Caso o parâmetro “password” tenha sido especificado no arquivo “/boot/grub/menu.lst”, será necessário antes de acessar as outras opções (que estarão desabilitadas), pressionar “p” e entrar com a senha válida para habilitar a execução de outras opções.

Quando o acesso a linha de comandos for obtido, você pode verificar os comandos disponíveis, pressionando duas vezes a tecla <TAB>. Note, que você também pode utilizar esta

tecla para completar nomes de comandos, bem como, parâmetros de alguns comandos. Alguns comandos disponíveis:

- **cat** - Este comando permite verificar o conteúdo de um arquivo qualquer, o qual deve estar gravado em um dispositivo ligado a sua máquina. Embora seja um recurso útil, nenhuma permissão de acesso é verificada e qualquer pessoa que tenha acesso a linha de comandos do *GRUB*, poderá listar o conteúdo de arquivos importantes do seu sistema. Para contornar este problema, configure adequadamente o parâmetro “password” no arquivo “/boot/grub/menu.lst”. Não esqueça que ainda é possível utilizar um disquete com o *GRUB* para iniciar a máquina, o que permite usar a linha de comandos pelo disquete.
Ex: `grub> cat (hd0,0)/etc/passwd`
- **cmp** - Este comando é utilizado para comparar dois arquivos.
Ex: `grub> cmp (hd0,0)/arquivo1 (hd0,0)/arquivo2`
- **configfile** - Carrega um arquivo de configuração do *GRUB*.
Ex: `grub> configfile (hd0,0)/boot/grub/menu.lst`
- **displayapm** - Mostra informações sobre APM.
- **displaymem** - Mostra informações sobre a memória RAM.
- **find** - Permite encontrar um arquivo. A saída deste comando disponibiliza o nome completo do caminho para o arquivo e a partição onde o mesmo está localizado.
Ex: `grub> find stage1`
- **geometry** - Mostra informações sobre a geometria reconhecida para seu dispositivo de armazenamento principal, e permite que você defina uma geometria personalizada, caso esta esteja sendo reconhecida de forma errada.
- **help** - É um comando para ver a ajuda sobre a utilização de outros comandos.
Ex: `help cmp`
- **install** - Instala o *GRUB*, embora não seja recomendado o uso deste comando diretamente, já que é possível cometer erros facilmente e sobrescrever a tabela de partições de seu disco.
Ex: `install (fd0)/grub/stage1 d (fd0) (fd0)/grub/stage2 p (fd0)/grub/menu.lst`
- **setup** - Você pode usar este comando para instalar o *GRUB*. Note que sua sintaxe é menos complexa do que a usada pelo comando “install”.
Ex:
`grub> root = (hd0,0)`
`grub> setup = (hd0)`
- **quit** - Abandona a linha de comandos do *GRUB*.
- **reboot** - Reinicia o computador.
- **boot** - Efetua o carregamento através das opções definidas via linha de comando. Suponha um sistema Linux instalado em (hd0,0), podemos passar os seguintes comandos na linha de comandos para efetuar o *boot* de uma imagem do GNU/Linux:
Ex:
`grub> root (hd0,0)`
`grub> kernel (hd0,0)/boot/vmlinuz-2.4.16 vga=6`
`grub> boot`

Muitos outros comandos estão disponíveis tanto na linha de comandos do *GRUB* quanto no arquivo de configuração “/boot/grub/menu.lst”. Um estudo mais aprofundado pode ser feito, lendo a documentação oficial do *GRUB* com o comando “man grub”.

5.6. Removendo o *GRUB* do *MBR*

Não existe a necessidade de se remover o *GRUB* do *MBR*, pois não há utilização para o mesmo vazio. Mas caso você queira fazer isso de qualquer forma, há várias formas, uma delas é usar o utilitário “fdisk” do MS DOS, juntamente com o parâmetro “/mbr” (“fdisk /mbr”), outra forma mais arriscada inclusive, é usando o utilitário “dd” do GNU/Linux da seguinte forma, supondo que você esteja utilizando um disco conectado a controladora principal IDE:

```
dd if=/dev/zero of=/dev/hda bs=446 count=1
```

Ao instalar sistemas como o MS Windows XP, 2000, 2003, entre outros que não Microsoft, o *GRUB* será automaticamente substituído pelos gerenciadores de partida

relacionados ao SO em questão.

(Os detalhes contidos na seção, foram desenvolvidos por Alexandre Costa alebyte@bol.com.br como contribuição ao guia FOCA GNU/Linux, e também, retirados das páginas do manual oficial do *GRUB*, e editados pela equipe da CELEPAR.)

6. Sistema de Boot e Runlevels

Neste tópico, explanaremos sobre o processo de carregamento do SO (Sistema Operacional) e dos níveis de execução que ele pode assumir.

6.1. Processo de Carregamento do Kernel

Enquanto o núcleo do SO é carregado na inicialização do sistema, ele exibe uma série de informações sobre os dispositivos de hardware, serviços e recursos que estão sendo disponibilizados pelo sistema. Todo este processo, é conhecido como “processo de *boot*” ou “processo de inicialização” do sistema. As mensagens que aparecem durante o processo de inicialização, podem ser conferidas posteriormente pelo usuário através do comando “**dmesg**”.

Durante o processo de carregamento do SO, o “**init**”, que é o processo principal de controle de inicialização, tem a função de invocar outros processos que estão organizados e divididos sob a forma de níveis, níveis de execução como propriamente dito. Cada um destes níveis de execução, possui um conjunto de processos próprio que podem ser executados separadamente uns dos outros, em cada um dos níveis existentes.

Tradicionalmente, são utilizados 7 níveis de execução diferentes, enumerados de 0 a 6. Por padrão, o nível 0 é utilizado para realizar o desligamento (*shutdown*) normal da máquina. O nível 6, é usado para invocar o processo de reinicialização (*reboot*) do sistema. O nível 1, também conhecido como modo monousuário, é o modo no qual o *Kernel* do Linux só carrega os recursos necessários para o funcionamento básico do SO e disponibiliza apenas acesso ao superusuário do sistema (*root*), de modo geral, para que este possa realizar a manutenção de algum ponto falho do sistema. Os *runlevels* 2 a 5, são utilizados para carga dos diferentes serviços providos pelo sistema, e possuem uma característica comum entre si, atuarem no modo multiusuário.

O comando “**init**”, pode ser usado seguido de um número representando um *runlevel* desejado (de 0 a 6), para alterar o nível de execução atual de um sistema Linux. Abaixo um exemplo do uso do comando para alterar o nível de execução atual do sistema:

#init 3

Você poderá verificar o *runlevel* atual do sistema, através do comando:

#ps aux | grep init

A saída deste comando será algo como:

```
root    1    0.0 0.2 844   72 ? S   Sep 6  0:16 init [2]
user    9121 0.0 0.9 884   296 p3 D  13:25  0:00 grep init
```

Neste exemplo, o nível de execução atual do SO é o 2, representado pela indicação “init [2]” presente na primeira linha da saída do comando. Se preferir, você também pode utilizar o comando “**runlevel**”, como no exemplo abaixo:

#runlevel

A saída deste comando será algo como:

N 2

O “N 2” da saída do comando “**runlevel**”, indica que o nível de execução atual do sistema é o 2, se você mudar o nível para outro qualquer, o nível anterior será mostrado no lugar do “N” da saída acima. No Linux existem muitas formas de se fazer a mesma coisa, escolha a que mais lhe agrada ;-)

6.2. Entendendo o Funcionamento dos Níveis de Execução do Sistema (Runlevels)

Como já comentamos na seção anterior (“Processo de Carregamento do Kernel”), o “**init**”

é o responsável pelo controle do processo de inicialização do sistema. O “init” utiliza o arquivo “/etc/inittab” durante sua execução, para configurar cada *runlevel* do sistema. É também neste arquivo que se encontra a configuração do nível de execução padrão do sistema, que pode ser verificado através da linha que contém a designação “:initdefault:” deste arquivo (/etc/inittab). No GNU/Debian o arquivo “/etc/inittab” possuirá uma linha com a seguinte definição:

id:2:initdefault:

A linha acima, indica que o nível de execução padrão do sistema é o 2. Podemos alterar este parâmetro, para ajustar o nível padrão de execução de um sistema Linux qualquer.

Na distribuição GNU/Debian, os diretórios “/etc/rc[0-6].d” contém as ligações simbólicas para arquivos em “/etc/init.d”, que são acionados pelo “init” no nível de execução correspondente. Por exemplo, o arquivo “S10sysklogd” em “/etc/rc2.d”, é um *link* simbólico para “/etc/init.d/sysklogd”. Então, o que aconteceria se você removesse o arquivo “/etc/rc2.d/S10sysklogd” ? Simplesmente, o serviço “sysklogd” deixaria de ser executado no nível de execução 2 do seu sistema.

O GNU/Debian, possui o seguinte padrão para definir se uma ligação simbólica em “/etc/rc[0-6].d” iniciará ou interromperá a execução de um serviço em “/etc/init.d”:

- Se um *link* é iniciado com a letra K (*kill*), quer dizer que o serviço será interrompido naquele nível de execução. O que ele faz na verdade, é executar o *script* do serviço em questão em “/etc/init.d” seguido da opção “stop”.
- Se um *link* é iniciado com a letra S (*start*), quer dizer que o serviço será iniciado naquele nível de execução. Isto fará com que o *script* do serviço em questão em “/etc/init.d” seja invocado seguido da opção “start”.

Por ordem, os *links* com a letra “K” são executado primeiro seguido pelos que iniciam pela letra “S”. A ordem com que são executados, depende também do valor numérico que acompanha o *link*, por exemplo, os seguintes arquivos são executados em sequência:

```
S10sysklogd
S12kernel
S20inetd
S20linuxlogo
S20logoutd
S20lprng
S89cron
S99xdm
```

Note, que os arquivos que iniciam com o mesmo número (S20*), são executados por ordem alfabética.

Para inserir ou remover scripts nos runlevels é interessante utilizar a ferramenta update-rc.d, que é a maneira correta (eu até diria: essa é a forma mais elegante) para controlar o uso dos scripts de inicialização:

Exemplos de uso:

```
# update-rc.d ssh defaults
```

Inserir o serviço ssh (/etc/init.d/ssh) em todos os runlevels para iniciar e parar de forma correta dependendo de cada runlevel.

```
# update-rc.d ssh start 20 2 .
```

Inserir o serviço ssh (/etc/init.d/ssh) apenas no runlevel 2 para iniciar.

```
# update-rc.d -f ssh remove
```

Remover o serviço ssh (/etc/init.d/ssh) de todos os runlevels em que está registrado.

7. Sistema de Logs

Arquivo de *Log*, é uma designação dada aos arquivos que são utilizados pelos programas do sistema para registrar suas atividades. Estes registros, em geral, são compostos por mensagens informativas, de alerta e de erro, geradas pelos programas, durante sua execução. Estes arquivos possuem informações muito úteis para o administrador do sistema. Através deles, é possível verificar o funcionamento do sistema, o comportamento dos programas, prevenir e corrigir erros e auditar o ambiente operacional.

No Linux, há uma estrutura central para armazenamento e funcionamento do sistema de *logs*. Neste SO, os arquivos de *log*, são guardados comumente no diretório `/var/log`, e um serviço especial denominado **“syslog”**, que pode fazer de forma unificada, todo o registro de atividades do sistema (*kernel* e outros programas).

7.1. Principais Arquivos de Log

Dentro do diretório `/var/log` podemos encontrar uma variedade de arquivos de *log*. Dentre eles, alguns merecem destaque:

Nome do Arquivo	Descrição
messages	É um arquivo de log geral do sistema. Guarda desde algumas mensagens do kernel até registros enviados pelos programas do usuário.
syslog	Usado principalmente para registrar atividades do kernel e de alguns serviços do sistema. É o mais importante arquivo de log de um ambiente Linux, juntamente com <code>/var/log/messages</code> .
auth.log	Usado para guardar informações sobre a autenticação de usuários no sistema.
wtmp	É um arquivo binário, que guarda a contabilidade de tempo de acesso ao sistema por parte dos usuários. Para verificar o conteúdo deste arquivo, é necessário o uso do comando <code>“last”</code> .
Xorg.[número].log	Guarda informações relacionadas ao servidor X.
dmesg	Mensagens do Kernel no momento de inicialização do sistema. Pode ser lido, através do comando de nome homônimo.
debug	Possui informações com maior nível de detalhes do que em <code>/var/log/syslog</code> e <code>/var/log/messages</code> . Interessante para verificação de problemas no sistema.

7.2. Formato dos Arquivos de Log

Um arquivo de *log*, é normalmente composto pelos seguintes campos:

Data|Hora|Maquina|daemon|mensagem

O campo `“maquina”` é o nome do computador que registrou a mensagem (a maquina pode atuar como um servidor de *logs* registrando mensagens de diversos computadores em sua rede). O campo `“daemon”`, indica qual programa gravou a mensagem.

O uso dos utilitários da console, pode ajudar muito na pesquisa e monitoração de *logs* do sistema. Por exemplo, para obter todas as mensagens do *kernel* da estação de trabalho `“wrk1”`, eliminando os campos `“wrk1”` e `“kernel”`, poderíamos utilizar o seguinte conjunto de comandos:

```
cat -A /var/log/*|grep 'wrk1'|grep 'kernel'|awk '{print $1 $2 $3 $6 $7 $8 $9 $10 $11 $12}'
```

Os parâmetros iniciados por `“$”` do comando `“awk”`, indicam que campos serão listados, (omitimos `“$4”` e `“$5”` que são respectivamente `“wrk1”` e `“kernel”`). Outro utilitário bastante usado

para monitoração de *logs* é o “logcheck”, veremos mais detalhes sobre ele nas próximas seções.

7.3. Daemons de Log do Sistema

Os principais *daemons* (serviços) de *log* do sistema, registram as mensagens de saída (atividades) do *kernel* (klogd) e sistema (syslogd) nos arquivos que são armazenados em “/var/log”.

A classificação de qual arquivo em “/var/log” receberá um determinado tipo de mensagem, é controlada pelo arquivo de configuração “/etc/syslog.conf”, através de parâmetros de configuração denominados “facilidades” e “níveis”. Para maiores detalhes sobre estes conceitos, consulte a seção “Configurando o syslog.conf” deste documento.

7.3.1. O daemon syslogd

Este *daemon* controla o registro de *logs* do sistema.

syslogd [opções]

opções

-f

Especifica um arquivo de configuração alternativo ao “/etc/syslog.conf”.

-h

Permite redirecionar mensagens recebidas a outros servidores de *logs* especificados.

-l [computadores]

Especifica um ou mais computadores (separados por “:”) que deverão ser registrados somente com o nome de máquina ao invés do FQDN (nome completo, incluindo domínio).

-m [minutos]

Intervalo em minutos que o syslog mostrará a mensagem --MARK--. O valor padrão é 20 minutos, 0 desativa.

-n

Evita que o processo caia automaticamente em *background*. Necessário principalmente se o “syslogd” for controlado pelo “init”.

-p [soquete]

Especifica um soquete UNIX alternativo ao invés de usar o padrão “/dev/log”.

-r

Permite o recebimento de mensagens através da rede através da porta UDP 514. Esta opção é útil para criar um servidor de *logs* centralizado na rede. Por padrão, o servidor syslog rejeitará conexões externas.

-s [domínios]

Especifica a lista de domínios (separados por “:”) que deverão ser retirados antes de enviados ao *log*.

-a [soquetes]

Especifica soquetes adicionais que serão monitorados. Esta opção será necessária se estiver usando um ambiente “chroot”. É possível usar até 19 soquetes adicionais.

-d

Ativa o modo de depuração do syslog. O syslog permanecerá operando em primeiro plano e mostrará as mensagens no terminal atual.

Na distribuição GNU/Debian, o *daemon* “syslogd” é iniciado através do *script* “/etc/init.d/sysklogd”.

7.3.2. O daemon klogd

Este *daemon* controla o registro de mensagens do *kernel*. Ele atua monitorando as mensagens enviadas pelo *kernel* e as desvia para o *daemon* de monitoramento “syslogd”, por padrão.

klogd [opções]

opções

-d

Ativa o modo de depuração do *daemon*.

-f [arquivo]

Envia as mensagens do *kernel* para o arquivo especificado ao invés de enviar ao *daemon* do syslog (syslogd).

-i

Envia um sinal para o *daemon* recarregar os símbolos de módulos do *kernel*.

-l

Envia um sinal para o *daemon* recarregar os símbolos estáticos e de módulos do *kernel*.

-n

Evita a operação em *background* (segundo plano). Útil se iniciado pelo “init”.

-k [arquivo]

Especifica o arquivo que contém os símbolos do *kernel*. Exemplos deste arquivo estão localizados em “/boot/System.map-xx.xx.xx”. O registro dos símbolos em um arquivo de *log* ao invés dos números que são informados pelo *kernel* por padrão, tornam a depuração de um problema mais simples, já que os símbolos são mais descritivos do que os números.

-o

Faz com que o *daemon* leia e registre todas as mensagens encontradas nos *buffers* do *kernel*, e após esta operação, o *daemon* é encerrado.

-p

Ativa o modo paranóia. Isto fará com que o klogd somente carregue detalhes sobre os módulos, quando os caracteres “Oops” forem detectados nas mensagens do *kernel*. É recomendável ter sempre a última versão do klogd e evitar a utilização desta opção em ambientes críticos.

-s

Força a utilização da interface de chamadas do sistema para comunicação com o *kernel*.

-x

Esconde tradução EIP, assim ele não lê o arquivo “/boot/System.map-xx-xx-xx”.

7.3.3. Configurando o syslog.conf

O arquivo de configuração “/etc/syslog.conf”, define os arquivos que irão armazenar os registros de *log* feitos pelo sistema, no seguinte formato:

facilidade.nível destino

Os sinalizadores “facilidade” e “nível” são definidos separados por um “.”, e possuem parâmetros que definem o que será registrado nos arquivos de *log* do sistema. O sinalizador “facilidade”, é usado para especificar que tipo de programa está enviando a mensagem. Os seguintes níveis são permitidos (em ordem alfabética):

- auth - Mensagens de segurança/autorização (é recomendável usar “authpriv” ao invés deste).
- authpriv - Mensagens de segurança/autorização (privativas).
- cron - *Daemons* de agendamento (cron e at).
- daemon - Outros *daemons* do sistema que não possuem facilidades específicas.
- ftp - *Daemon* de FTP do sistema.
- kern - Mensagens do *kernel*.
- lpr - Sub-sistema de impressão.
- local0 a local7 - Reservados para uso local.
- mail - Sub-sistema de e-mail.
- news - Sub-sistema de notícias da USENET.
- security - Sinônimo para a facilidade “auth” (evite usá-la).
- syslog - Mensagens internas geradas pelo syslogd.
- user - Mensagens genéricas de nível do usuário.
- uucp - Sub-sistema de UUCP.
- O caractere “*” (asterisco) - Confere com todas as facilidades.

Mais de uma facilidade pode ser especificada na mesma linha do syslog.conf separando-as com o caractere “,”.

O sinalizador “nível”, especifica a importância da mensagem. Os seguintes níveis são permitidos (em ordem de importância invertida; da mais para a menos importante):

- emerg - O sistema está inutilizável.
- alert - Uma ação deve ser tomada imediatamente para resolver o problema.
- crit - Condições críticas.
- err - Condições de erro.
- warning - Condições de alerta.
- notice - Condição normal, mas significativa.
- info - Mensagens informativas.
- debug - Mensagens de depuração.
- none - Nenhuma prioridade.
- O caractere “*” (asterisco) - Confere com todos os níveis.

Além destes níveis, os seguintes sinônimos estão disponíveis:

- error - Sinônimo para o nível “err”.
- panic - Sinônimo para o nível “emerg”.
- warn - Sinônimo para o nível “warning”.

O sinalizador “destino”, define o destino das mensagens, pode ser um arquivo, um pipe (se iniciado por um “|”), um computador remoto (se iniciado por uma “@”), ou para determinados usuários do sistema (especificando os *logins* separados por vírgula) ou para todos os usuários que estão usando o SO via o comando “wall” (usando o caractere “*”).

Conjuntos de sinalizadores de “facilidades” e “níveis” podem ser agrupadas separando-as por pelo caractere “;”.

Observações:

- Sempre utilize a tecla <TAB> ao invés da barra de espaços para separar os parâmetros do arquivo “/etc/syslog.conf”.
- Alguns sinalizadores de “facilidades” como o “security”, emitem um alerta

sonoro no sistema e enviam uma mensagem para a console, como forma de alerta ao administrador e usuários que estão utilizando o sistema.

Existem ainda 4 caracteres que garantem funções especiais: "*", "=", "!" e "-". A seguir uma breve descrição da função que cada um deles desempenha:

- O caractere "*" - Todas as mensagens da facilidade especificada serão redirecionadas.
- O caractere "=" - Somente o nível especificado será registrado.
- O caractere "!" - Todos os níveis especificados e de maior importância NÃO serão registrados.
- O caractere "-" - Pode ser usado para desativar a sincronização de escrita imediata do arquivo de *log*. Tem a vantagem de obter maior desempenho, porém, como a gravação não é síncrona, caso o sistema seja desligado abruptamente, você perderá a informação das atividades que ainda não foram salvas no arquivo.

Os caracteres especiais "=" e "!" podem ser combinados em uma mesma regra.

Veja abaixo um exemplo de um arquivo `/etc/syslog.conf` padrão de um sistema GNU/Debian:

```
#
# Primeiro alguns arquivos de log padrões. Registrados por facilidade:
#

auth,authpriv.*          /var/log/auth.log
*.*;auth,authpriv.none   -/var/log/syslog
cron.*                   /var/log/cron.log
daemon.*                 -/var/log/daemon.log
kern.*                   -/var/log/kern.log
lpr.*                    -/var/log/lpr.log
mail.*                   /var/log/mail.log
user.*                   -/var/log/user.log
uucp.*                   -/var/log/uucp.log

#
# Registro de logs do sistema de mensagens. Divididos para facilitar
# a criação de scripts para manipular estes arquivos.
#
mail.info                -/var/log/mail.info
mail.warn                -/var/log/mail.warn
mail.err                 /var/log/mail.err

# Registro para o sistema de news INN
#
news.crit                 /var/log/news/news.crit
news.err                 /var/log/news/news.err
news.notice              -/var/log/news/news.notice

#
# Alguns arquivos de registro "pega-tudo".
# São usadas "," para especificar mais de uma prioridade (por
# exemplo, "auth,authpriv.none") e ";" para especificar mais de uma
# facilidade.nível que será gravada naquele arquivo.
# Isto permite deixar as regras consideravelmente menores e mais legíveis
#
*.=debug;\
    auth,authpriv.none;\
    news.none;mail.none   -/var/log/debug
```



```

*.=info;*.=notice;*.=warn;\
    auth,authpriv.none;\
    cron,daemon.none;\
    mail,news.none          -/var/log/messages

#
# Emergências são enviadas para qualquer um que estiver logado no sistema.
# Isto é feito através da especificação do "*" como destino das mensagens e são
# enviadas através do comando wall.
#
*.emerg                      *

#
# É interessante ter as mensagens mostradas no console,
# mas somente em consoles que não são utilizadas. Caso você queira esta
# funcionalidade, descomente as 4 linhas abaixo:

#daemon,mail.*;\
#   news.=crit;news.=err;news.=notice;\
#   *.=debug;*.=info;\
#   *.=notice;*.=warn       /dev/tty8

# O pipe "/dev/xconsole" é usado pelo utilitário "xconsole". Para usá-lo,
# você deve executar o "xconsole" com a opção "-file":
#
# $ xconsole -file          /dev/xconsole [...]

#
# NOTA: Ajuste as regras abaixo, ou ficará maluco se tiver um site
# muito acessado.
#
daemon.*;mail.*;\
    news.crit;news.err;news.notice;\
    *.=debug;*.=info;\
    *.=notice;*.=warn       /dev/xconsole

# A linha baixo envia mensagens importantes para a console em que
# estamos trabalhando (principalmente para quem gosta de ter
# controle total sobre o que está acontecendo com seu sistema).
*.err;kern.debug;auth.notice;mail.crit    /dev/console

```

7.4. Ferramentas para Arquivos de Log

Nesta seção, descreveremos algumas ferramentas úteis para o monitoramento, gerenciamento e utilização dos arquivos de log.

7.4.1. logger

Este comando permite enviar uma mensagem para os arquivos de *log* do sistema. A mensagem é enviada aos arquivos, através do *daemon* "syslogd", ou via soquete do sistema, é possível especificar dados como prioridade, nível, nome de identificação do processo, etc. Seu uso é muito útil em *scripts* ou em outros eventos do sistema.

```
logger [opções] [mensagem]
```

Onde:

mensagem

É a mensagem que será enviada ao *daemon* “syslogd” para ser gravada num dos respectivos arquivos de *log* presentes no sistema.

opções

-i

Registra o PID (número identificador) do processo.

-s

Envia a mensagem para a saída padrão (STDOUT) e para o serviço “syslogd”.

-f [arquivo]

Envia o conteúdo do arquivo especificado como mensagem ao “syslogd”.

-t [nome]

Especifica o nome do processo responsável pelo *log* que será exibido antes do PID na mensagem do “syslogd”.

-p [prioridade]

Especifica a prioridade da mensagem. Deve-se utilizar a notação “facilidade.nível”. Verifique os tipos de prioridade/níveis existentes no Linux no arquivo de configuração “/etc/syslog.conf”. O valor padrão que é utilizado caso este parâmetro não seja especificado é “user.notice”.

-u [soquete]

Envia a mensagem para o soquete especificado ao invés do “syslogd”. Útil para enviar mensagens através da rede para um servidor de *log*.

Exemplo:

logger -i -t GGA “Teste teste teste”

logger -i -s -f /tmp/mensagem -p user.info

7.4.2. logcheck

É um programa usado para enviar um e-mail periodicamente ao administrador do sistema (através do “cron” ou outro serviço com a mesma função) alertando sobre os eventos que ocorreram desde a última execução do programa. As mensagens do **logcheck**, são tratadas por arquivos em “/etc/logcheck” e organizadas em categorias, antes de serem enviadas por e-mail, isto garante muita praticidade na interpretação dos eventos ocorridos no sistema. É necessário um serviço SMTP configurado (exim, sendmail, etc) para que o e-mail elaborado por esta ferramenta possa ser enviado ao administrador.

As categorias são organizadas por ordem de importância (da mais para menos importante), e vão desde avisos sobre atividades ilícitas sendo executadas no sistema (providências devem ser tomadas imediatamente para resolver a situação) até eventos anormais do sistema (mensagens de inicialização, mensagens dos *daemons* do sistema, etc.).

O tipo de mensagem que será incluída/ignorada nos *logs* enviados pela ferramenta, podem ser ajustadas pelo administrador do sistema através dos arquivos/diretórios presentes em “/etc/logcheck”. Nomes de arquivos/diretórios contendo a palavra “ignore” são usados para armazenar expressões regulares que NÃO serão enviadas pelo **logcheck**. É permitido o uso de expressões regulares perl/sed para especificar as mensagens nos arquivos de *log*.

Para maiores detalhes sobre esta ferramenta, consulte o manual do comando:

man logcheck

7.4.3. logrotate

Usado para fazer a rotação dos arquivos de *log* do sistema. A rotação de *log* é

processo que consiste em guardar o arquivo de *log* que está sendo utilizado atualmente pelo sistema e fazer com que os *daemons* passem a utilizar um novo arquivo para armazenar os novos registros das atividades do sistema. Este processo também é conhecido como arquivamento de *log*.

As tarefas do **logrotate** são programadas via “cron” ou outro serviço de agendamento disponível no sistema. Opcionalmente, os arquivos de *log* antigos poderão ser compactados para diminuir a utilização de espaço em disco ou enviados por e-mail. A rotação dos arquivos de *log* proporciona maior agilidade quando precisamos encontrar algum detalhe útil, o que seria mais difícil e demorado usando um arquivo de *log* muito extenso.

A rotação de *log*, é feita de acordo com o tamanho máximo que um determinado arquivo de *log* pode atingir, mas a opção “-f” da ferramenta pode ser usada para “forçar” uma rotação no sistema de *log*.

O arquivo principal de configuração do **logrotate** é o “/etc/logrotate.conf”. Abaixo um modelo de configuração deste arquivo:

As configurações abaixo, afetam globalmente o funcionamento do logrotate:

```
# Faz a rotação do arquivos de log semanalmente
weekly
```

```
# Mantém as últimas 4 cópias de logs anteriores
rotate 4
```

```
# Erros de não existência dos logs são enviados por e-mail ao usuário “root”
mail root
```

```
# Cria novos arquivos de log (vazios) após rotacionar os antigos
create
```

```
# O parâmetro “compress” é utilizado para fazer a compressão dos arquivos
# antigos. O parâmetro “delaycompress” é usado para que o primeiro log
# seja mantido
compress
delaycompress
```

```
# Executam os scripts em “prerotate” e “postrotate” a cada vez que os logs
# forem arquivados
nosharedscripts
```

```
# Definimos um diretório que poderá conter definições individuais de rotação
# de log para cada serviço do sistema. Contudo, alertamos que diversas
# configurações individuais podem deixar a interpretação deste arquivo confusa
include /etc/logrotate.d
```

opções # Define opções específicas para a rotação mensal de “/var/log/wtmp”. As

```
# definidas individualmente como neste exemplo, sobrepõem as opções globais
# definidas anteriormente.
```

```
# As definições para “/var/log/wtmp”, implicam que caso o arquivo atinja 5MB
# (size 5M) ele será arquivado, será criado um novo arquivo com permissão
# 0664 e pertencerá ao usuário “root” e o grupo “utmp” (create 0664 root utmp)
# e será mantida somente uma cópia do log anterior (rotate 1).
```

```
/var/log/wtmp {
    monthly
    create 0664 root utmp
    size 5M
    rotate 1
}
```

```
}

# Define opções específicas para a rotação mensal de "/var/log/btmp".
# Neste caso, o parâmetro "missingok" fará com que sejam gerados alertas
# informando ao administrador que o arquivo não existe.
/var/log/btmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

# Define opções específicas para a rotação semanal de "/var/log/lastlog".
/var/log/lastlog {
    missingok
    weekly
    create 0664 root utmp
    rotate 1
}

# Define opções específicas para a rotação diária de "/var/log/messages".
# Neste caso, o será arquivo ao atingir o tamanho de 1Mb (size 1M), então o
# novo arquivo será criado com as mesmas permissões do arquivo anterior.
# O comando "killall -1 syslogd" será executado após a rotação (postrotate)
# para que o daemon syslogd funcione corretamente, mas somente uma vez
# durante a rotação de vários arquivos de logs (sharedscripts).
# Serão mantidas as 10 últimas cópias (rotate 10) do arquivo /var/log/messages
# compactadas (o parâmetro "compress" foi especificado globalmente neste
# arquivo de configuração).
/var/log/messages {
    daily
    size 1M
    sharedscripts
    postrotate
        /sbin/killall -1 syslogd
    endsript
    rotate 10
}

# Define opções específicas para a rotação mensal dos arquivos em
# "/var/log/mirror/*". A falta desses arquivos não precisa ser notificada
# ao administrador (missingok), mesmo assim o parâmetro "nomail" evitará
# isto de qualquer forma. Os logs rotacionados não serão compactados
# (nocompress) e serão mantidas as últimas 7 cópias (rotate 7) dos arquivos.
/var/log/mirror/* {
    montly
    nomail
    missingok
    nocompress
    rotate 7
}

# Fim do arquivo de configuração
```

Qualquer definição de parâmetro especificado nos arquivos individuais de configuração, substituirá as definições anteriores. Quando o número máximo de arquivos de *log* mantidos pela opção "rotate" é atingida, os arquivos eliminados serão enviados para o usuário especificado na opção "mail". A utilização da diretiva "nomail" evita isso.

Quando for utilizar coringas para se referir a determinados arquivos dentro de um diretório, não utilize a sintaxe "log-xxx-*", porque isto forçaria a compressão de arquivos

".gz" já comprimidos, gerando arquivos do tipo ".gz.gz..." e derrubando o processamento da sua máquina gerada por um *loop* de compactação e enchendo as entradas de diretório. Ao invés disso, utilize a sintaxe "log-xxx-*.log", ou outra, modificando a configuração do programa que gera os *logs*.

Observação:

- É importante enviar um sinal "HUP" ao programa que grava um determinado arquivo de log crítico para que não ocorram problemas após a rotação, isto pode ser feito usando o parâmetro "postrotate".

8. Comandos Avançados

Pretendemos abordar neste tópico, a utilização de alguns comandos do ambiente GNU/Linux que são de grande importância no dia-a-dia do administrador de sistemas. Tome o cuidado necessário com relação a execução destes comandos em ambientes de produção.

8.1. Comandos de Uso Geral

8.1.1. cron

O “cron” na realidade não é um simples comando e sim um serviço do sistema. O cron é um serviço de agendamento, que permite que as atividades cadastradas em suas tabelas, sejam executadas numa data e hora pré-determinadas.

As tarefas são definidas no arquivo `“/etc/crontab”` e por arquivos individuais de usuários em `“/var/spool/cron/crontabs/[usuário]”` (criados através do programa `crontab`). Adicionalmente, a distribuição Debian utiliza os arquivos no diretório `“/etc/cron.d”` como uma extensão para o `“/etc/crontab”`.

Para agendar uma nova tarefa, basta editar o arquivo “/etc/crontab” com qualquer editor de texto (como o “pico” ou “vi”) e definir o mês/dia/hora que a tarefa será executada. Não é necessário reiniciar o *daemon* do cron, porque ele verifica seus arquivos a cada minuto.

O arquivo “/etc/crontab” tem o seguinte formato:

```

52      18      1      *      *      root      run-parts      --report
/etc/cron.monthly
|
|_ Comando que será executado
|_ UID que executará o comando
|_ Dia da semana (0-7)
|_ Mês (1-12)
|_ Dia do Mês (1-31)
|_ Hora
|_ Minuto

```

Onde:

Minuto

Valor entre 0 e 59.

Hora

Valor entre 0 e 23.

Dia do Mês

Valor entre 1 e 31.

Mês

Valor entre 1 e 12 (identificando os meses de Janeiro a Dezembro).

Dia da Semana

Valor entre 0 e 7 (identificando os dias de Domingo a Sábado). Note que tanto 0 quanto o 7 equivalem ao Domingo.

usuário

O usuário especificado será usado para executar o comando (o usuário deverá existir).

comando

Comando que será executado. Podem ser usados parâmetros normais usados na linha de comando.

Os campos do arquivo, são separados por um ou mais espaços ou tabulações. Um caractere “*” (asterisco), pode ser usado nos campos de data e hora para especificar todo o intervalo disponível. O caractere “-” (hífen), serve para especificar períodos de execução (incluindo o número inicial/final). O caractere “,” (vírgula), serve para especificar uma listagem de números. Passos podem ser especificados através do caractere “/” (barra). Veja os exemplos no final desta seção.

O arquivo gerado em “/var/spool/cron/crontabs/[usuário]” pelo comando “**crontab**”, tem o mesmo formato do “/etc/crontab” exceto por não possuir o campo usuário (UID), pois o nome do arquivo já identifica o usuário no sistema.

Para editar um arquivo de usuário em “/var/spool/cron/crontabs” ao invés de editar o arquivo “/etc/crontab” use o comando “crontab -e”, para listar as tarefas daquele usuário use “crontab -l” e para apagar o arquivo de tarefas do usuário “crontab -r” (adicionalmente, você pode remover somente uma tarefa através do “crontab -e”, apagando a linha correspondente).

Observação:

Não esqueça de incluir uma linha em branco no final do arquivo, caso contrário o último comando não será executado.

O **cron** define o valor de algumas variáveis automaticamente durante sua execução; a variável “SHELL” é definida como “/bin/sh”, “PATH” como “/usr/bin:/bin”, “LOGNAME”, “MAILTO” e “HOME” são definidas através do arquivo “/etc/passwd”. Os valores padrões destas variáveis, podem ser substituídos especificando um novo valor nos arquivos do cron.

Exemplos de um arquivo “/etc/crontab”:

SHELL=/bin/sh

PATH=/sbin:/bin:/usr/sbin:/usr/bin

00 10 * * * root sync

Executa o comando “sync”, como usuário “root”, todo dia as 10:00 hs.

00 06 * * 1 root updatedb

Executa o comando “updatedb”, como usuário “root”, toda segunda-feira as 06:00.

10,20,40 * * * marcius banner

Executa o comando “banner”, como usuário “marcius”, todos os dias e a toda a hora em 10, 20 e 40 minutos.

*** /10 * * * root fetchmail**

Executa o comando “fetchmail”, como “root”, de 10 em 10 minutos todos os dias.

15 0 25 12 * root echo "Feliz Natal ;-)" | mail john

Envia um e-mail as 0:15 todo o dia 25/12 para john, utilizando o usuário “root” desejando feliz natal ;-0.

30 5 ** 1-6 root tar -czf /backup/backup.tar.gz /home

Executa o comando "tar -czf /backup/backup.tar.gz /home" automaticamente as 5:30 de segunda-feira a sábado, usando para isto o usuário "root".

8.1.2. dd

O comando "dd" é utilizado para copiar blocos de dados de um dispositivo de armazenamento. Com o **dd**, todos os dados são copiados a nível de bloco, não interessando o tipo do sistema de arquivos utilizado no dispositivo. É um utilitário muito útil para criar cópias fiéis (imagens) a partir de um dispositivo de armazenamento para outro ou para arquivos. Exemplos de uso:

```
dd if=/dev/zero of=/var/arquivo_paginacao bs=1M count=256
```

Onde:

- O parâmetro "if" seguido pelo argumento "/dev/zero", informa a origem dos dados, neste caso, o dispositivo gerador de *bits* com valor 0 (zero) do sistema.
- O parâmetro "of" seguido pelo argumento "/var/arquivo_paginacao", informa o destino dos dados copiados, aqui, o arquivo "arquivo_paginacao", que receberá todos os *bits* com valor 0 (zero) gerados pela origem (parâmetro "if").
- O parâmetro "bs" que permite definir o tamanho do bloco a ser copiado. Neste exemplo, o bloco tem 1 *Megabyte* (1024 Kb) de tamanho. É necessário atenção na definição deste parâmetro, para maiores detalhes consulte a documentação do **dd** (**man dd**).
- O parâmetro "count" informa o número de blocos que serão copiados. No exemplo acima, serão copiados 256 blocos de 1 *Megabyte* cada, criando um arquivo de 256 *Megabytes* de tamanho total e com todos os *bits* de dados contendo o valor 0 (zero).

Vamos a um outro exemplo:

```
dd if=/dev/hdb of=/dev/hdc bs=1G count=4
```

Onde:

- O parâmetro "if", define a origem dos dados a serem copiados, neste caso, a partir do dispositivo "/dev/hdb".
- Por sua vez o parâmetro "of", define o destino dos dados, "/dev/hdc", outro dispositivo de armazenamento.
- Serão copiados blocos de dados com 1 *Gigabyte* de tamanho, definido através do parâmetro "bs".
- O número de blocos a serem copiados, 4 no exemplo acima, foi definido através do parâmetro "count".

8.1.3. diff

Compara dois arquivos e mostra as diferenças entre eles. O comando "diff" é usado somente para a comparação de arquivos em formato texto. As diferenças encontradas, podem ser redirecionadas para um arquivo, que posteriormente poderá ser usado pelo comando "**patch**", para alterar um arquivo que não contém as diferenças. Isto é útil para grandes arquivos de texto, já que é possível copiar somente as modificações (geradas através do diff, que são muito pequenas) e aplicar no arquivo para atualizá-lo (através do patch) evitando um grande trabalho manual. Este é um sistema de atualização muito usado na atualização dos códigos fonte do *kernel* do GNU/Linux.

```
diff [diretório1/arquivo1] [diretório2/arquivo2] [opções]
```

Onde:

diretório1/arquivo1 **diretório2/arquivo2**

Arquivos e/ou diretórios que serão comparados.

opções

-lines [num]

Gera a diferença em um certo número de linhas de contexto definido pelo argumento "num". Por padrão o diff gera um arquivo com 2 linhas que é o mínimo necessário para o correto funcionamento do comando patch.

-a
Compara os dois arquivos como arquivos texto.

-b
Ignora espaços em branco como diferenças.

-B
Ignora linhas em branco inseridas ou apagadas nos arquivos.

-i
Ignora diferenças entre maiúsculas e minúsculas nos arquivos.

-H
Usa análise heurística para verificar os arquivos.

-N
Em uma comparação de diretórios, se o arquivo apenas existe em um diretório, trata-o como presente mas vazio no outro diretório.

-P
Em uma comparação de diretórios, se o arquivos apenas existe no segundo diretório, trata-o como presente mas vazio no primeiro diretório.

-q
Informa somente se os arquivos possuem diferenças entre si, não mostra as diferenças entre eles.

-r
Compara diretórios e sub-diretórios existentes recursivamente.

-S [arquivo]
Inicia a comparação de diretórios pelo arquivo definido em "[nome]".

-t
Aumenta a tabulação das diferenças encontradas.

Use o comando "zdiff" para comparar diretamente arquivos compactados pelo utilitário **gzip**. Use o comando "sdiff" para visualizar as linhas diferentes entre os dois arquivos em formato texto simples.

Exemplos:

diff texto.txt texto1.txt

Compara o arquivo "texto.txt" com "texto1.txt" e exibe suas diferenças na tela.

diff -Bt texto.txt texto1.txt

Compara o arquivo "texto.txt" com "texto1.txt", ignorando linhas em branco e aumentando o espaço de tabulação na apresentação das diferenças encontradas.

diff texto.txt texto1.txt > texto.diff

Compara o arquivo "texto.txt" com "texto1.txt" e gera um arquivo chamado "texto.diff" contendo a diferença entre eles. Este arquivo poderá ser usado pelo comando patch para aplicar as diferenças existente entre os dois no arquivo "texto.txt".

diff -r /usr/src/linux-2.2.13 /usr/src/linux-2.2.14 > patch-2.2.14.diff

Compara recursivamente os diretórios e sub-diretórios de “linux-2.2.13” e “linux-2.2.14” e grava as diferenças entre eles no arquivo “patch-2.2.14.diff”.

8.1.4. dpkg

O “dpkg” é um programa de computador que é a base do Sistema de Gerenciamento de Pacotes da distribuição GNU/Debian. Foi criado por *Ian Jackson* em 1993. O dpkg, é similar ao RPM, que é utilizada em outras distribuições como o Red Hat, é usado para construir, instalar, testar, remover e fornecer informações sobre os pacotes Debian.

O dpkg é composto por uma série de ferramentas, que desempenham funções específicas em relação aos pacotes Debian. A nossa intenção aqui, não é explicar as opções que cada uma delas possui, e sim, apenas fornecer uma breve descrição do que cada uma delas faz. As ferramentas são:

- **dpkg-source** - Empacota e desempacota os arquivos fontes de um pacote Debian.
- **dpkg-deb** - Empacota e desempacota pacotes binários.
- **dpkg-reconfigure** - Permite que o usuário execute ajustes nos pacotes já instalados no sistema, também é conhecido como “reconfigurador de pacotes”. É uma das ferramentas mais utilizadas do conjunto de aplicações do dpkg.

Vamos nos concentrar agora, no utilitário dpkg em si. Vamos explorar melhor sua sintaxe de uso e suas principais ações e opções. Na linha abaixo, é mostrado a sintaxe padrão de uso deste utilitário:

```
dpkg [ações] [opções] [pacotes]
```

Onde:

pacotes

Juntamente com algumas opções do dpkg, é necessário informar o nome de um (ou mais) pacote para que a ação definida por aquela opção possa ser desempenhada. Por exemplo, no uso da opção “-i” que faz a instalação de um pacote, é preciso informar qual o pacote que deverá ser instalado.

ações:

-i pacote

Instala um pacote no sistema.

--unpack pacote

Desempacota um pacote, extraindo-o do formato “.deb” e montando uma estrutura de diretórios e arquivos de acordo com o esqueleto que foi usado na sua construção.

--configure pacote

Realiza a reconfiguração de um pacote que já está instalado no sistema. A reconfiguração consiste no ato de desempacotar os arquivos do pacote original, fazer a cópia de segurança dos arquivos de configuração do pacote instalado atualmente e executar o *script* “postinst” provido pelo pacote original. Tem o mesmo efeito que o comando “dpkg-reconfigure”.

-r pacote

Desinstala um pacote do sistema. Dependendo da configuração dos scripts de remoção do pacote, arquivos de dados e configuração poderão permanecer no sistema.

--update-avail

Atualiza a lista que contém a descrição dos pacotes disponíveis no sistema (mostrada com o comando “dpkg -s pacote”). A lista é armazenada no arquivo “/var/lib/dpkg/available”.

-A pacote

Atualiza a lista que contém a descrição dos pacotes disponíveis no sistema,

adicionando a informações sobre o pacote informado através do argumento homônimo.

--clear-avail

Limpa o arquivo que contém a lista com a descrição dos pacotes instalados no sistema. Evite usar esta ação.

-C

Procura por pacotes que foram instalados somente parcialmente em seu sistema. Muito interessante para resolução de problemas.

--get-selections padrão

Procura na lista de pacotes instalados no sistema pela expressão fornecida através do argumento "padrão".

--print-architecture

Imprime a arquitetura padrão utilizada pelos pacotes instalados no sistema.

-P pacote

Faz a remoção completa de uma pacote do sistema, incluindo arquivos de dados, de configuração, etc. Use esta opção com cautela.

-l [pacotes]

Lista todos os pacotes instalados no sistema com suas respectivas informações. É uma opção das mais utilizadas pelos administradores. Opcionalmente, você poderá especificar o nome de um ou mais pacotes (separados por espaços) para verificar suas informações.

-s pacote

Exibe as informações sobre um determinado pacote. Estas informações são retiradas do arquivo "/var/lib/dpkg/available".

-S padrão

Pesquisa pelos arquivos que um determinado pacote possui no sistema, de acordo com a expressão fornecida pelo argumento "padrão".

-L pacote

Permite visualizar todos arquivos relacionados ao pacote especificado.

-b diretório [nome]

Permite construir um pacote Debian, por meio de um diretório que esteja organizado sob a forma padrão do Debian para construção de pacotes, e opcionalmente, especificar um nome para o pacote gerado.

-c pacote

Lista o conteúdo de um pacote.

-l pacote

Exibe informações sobre um determinado pacote.

-X pacote diretório

Extraí e exibe o conteúdo de um pacote especificado através do argumento homônimo, para dentro do diretório fornecido pelo usuário.

opções:

As opções do comando dpkg tanto podem ser especificadas através da linha de comandos, como através do arquivo de configuração do dpkg, o "/etc/dpkg/dpkg.cfg".

--abort-after=número

Ajusta o número de erros que poderão ocorrer antes da execução do dpkg ser

encerrada.

-B

Quando um pacote é desinstalado do sistema, pode ocorrer que outros pacotes ainda instalados dependam do que foi removido. Esta opção faz com que estes pacotes também sejam removidos do sistema automaticamente.

--ignore-depends=pacote

Faz com que as dependências para a instalação de um pacote sejam ignoradas. Atualmente, o dpkg emite alertas sobre as dependências de instalação não resolvidas de um pacote. Esta opção faz com esses alertas sejam sumariamente ignorados e a instalação do pacote prossiga. Evite utilizar esta opção, ela gera problemas de “pacotes quebrados” no sistema, ou seja, pacotes que não tem todas suas dependências resolvidas.

--no-act

Faz uma simulação de acordo com as ações envolvidas. Por exemplo, você pode simular a instalação ou remoção de um pacote para verificar se algum erro será gerado pelo sistema de gerenciamento de pacotes. Tenha certeza que você especificou esta opção antes da citação de qualquer ação na linha de comando, caso contrário, esta opção não terá efeito e a ação será desempenhada de fato.

-R

Faz com que o dpkg opere no modo recursivo.

-G

Informa ao dpkg para não instalar um pacote caso uma versão mais recente do pacote em questão já esteja instalada no sistema.

-E

Não realiza a instalação de um pacote cuja a mesma versão já se encontra instalada no sistema.

--log=arquivo

Permite especificar um arquivo onde o dpkg vai realizar os registros das suas operações. Por padrão isto é feito em “/var/log/dpkg.log”.

Exemplos:

dpkg -i tuxracer-1.20.deb

Instala o pacote “tuxracer”.

dpkg -r tuxracer

Remove o pacote “tuxracer” do sistema.

dpkg -P mysql-server-4.1-5

Remove completamente o pacote “mysql-server” do sistema, inclusive seus arquivos de dados e de configuração.

dpkg -l

Lista todos os pacotes instalados no sistema.

8.1.5. export

Exibe, configura e remove uma variável de ambiente. Variáveis de ambiente são diferentes das variáveis de *shell*, que podem ser configuradas/removidas com os comandos “set” e “unset” respectivamente.

Variáveis de ambiente, são utilizadas para guardar dados voláteis relacionados a sessão do usuário no sistema. Por exemplo, a variável “USERNAME” possui o nome do usuário que iniciou a sessão, já a variável “EDITOR”, guarda o caminho e o nome para o

editor de textos padrão do *shell*.

Variáveis de ambiente também podem ser definidas através do arquivo de configuração `/etc/environment`.

```
export [opções] [variável=valor]
```

Onde:

variável=valor

Nome da variável de ambiente que se deseja configurar e seu respectivo valor. Caso a variável não exista, ela será criada e receberá o valor discriminado após o sinal de igualdade, caso já exista, somente receberá o valor definido após o sinal de igual.

opções:

-f [variável/função]

Caso a opção `-f` seja informada, significa que o usuário está informando uma função *shell* e não uma variável.

-n [variável/função]

Remove uma variável/função de ambiente.

Exemplos:

export

Exibe as variáveis de ambiente do usuário.

export teste=Linux

Cria a variável de ambiente `teste` caso ela não exista, e ajusta seu valor para `Linux`.

export -n teste

Remove a variável de ambiente `teste`.

8.1.6. find

O comando `find` pode ser utilizado não só para procurar arquivos e diretórios, mas também, para executar outros comandos em conjunto com o resultado da pesquisa realizada. Veja o exemplo abaixo:

```
find . -name "*.txt" -type f -exec wc -l {} \;
```

Onde:

- O caractere `."` é o diretório atual em que o usuário se encontra.
- O parâmetro `-name` seguido do argumento `"*.txt"`, informa ao comando `find` para procurar por qualquer arquivo/diretório que contenha qualquer nome (uso do caractere coringa `"*"`) seguido pela expressão `".txt"`.
- O parâmetro `-type f`, faz com que o comando `find` procure somente por arquivos.
- O parâmetro `-exec` seguido pelo argumento `"wc -l {} \;"`, informa ao comando `find` que ele deverá executar o que foi fornecido como argumento (o comando `"wc -l"`, neste caso) para cada resultado da busca realizada (os caracteres `"{ }"`). Os caracteres `"\;"` no final do argumento, sinalizam o término do comando.

8.1.7. fsck

O utilitário de sistema `fsck` (de *"File System Check"* ou *"File System Consistency Check"*), é uma ferramenta usada para checar a consistência de um sistema de arquivos no GNU/Linux.

Em geral, o `fsck` é executado na inicialização do sistema, quando é detectado que um sistema de arquivos está num estado inconsistente, indicando um desligamento anormal, como um travamento ou desligamento de energia. Tipicamente, o `fsck` disponibiliza opções para reparar sistemas de arquivos danificados: interativamente (o usuário tem que decidir como consertar problemas específicos), permitir ao `fsck` resolver

problemas específicos (assim o usuário não precisa responder nenhuma questão), ou rever os problemas que precisam ser resolvidos num sistema de arquivos sem realmente resolvê-los.

O utilitário também pode ser executado manualmente pelo administrador do sistema se este acreditar que há um problema com um determinado sistema de arquivos. Para esta atividade, é sempre muito importante desmontar o sistema de arquivos em questão, antes de iniciar o procedimento de checagem.

O `fsck` possui extensões para cada tipo específico de sistema de arquivos suportados pelo SO. Assim, o executável `"fsck.reiserfs"` é apropriado para checagens do sistema de arquivos `"ReiserFS"`, já o `"fsck.ext3"`, é adequado ao `"EXT3"`, e assim por diante. Alguns sistemas de arquivos como o `"XFS"` por exemplo, possuem extensões do `fsck` virtuais (não realizam a checagem de fato), devido as suas particularidades técnicas (o `XFS` por exemplo, faz isso durante a montagem do sistema de arquivos, caso seja necessário). É importante que o administrador do sistema, procure a extensão correta para o sistema de arquivos ao qual deseja verificar ou os utilitários adequados para desempenhar essa função em cada um deles.

```
fsck[.extensão] [opções] [sistema arquivos]
```

Onde:
extensão

É a extensão adequada ao sistema de arquivos que será checado. Normalmente, as extensões possuem o mesmo nome que o sistema de arquivos em questão.

sistema de arquivos

É o sistema de arquivos que deverá ser checado. Por exemplo, `"/dev/hda2"`, `"/dev/sda5"`, etc. Mais de um sistema de arquivos pode ser informado com este argumento, separados por espaços.

opções:

Explicaremos aqui alguns opções gerais do `fsck`, já que cada extensão poderá conter opções de modo peculiar.

-s

Permite visualizar a checagem de vários sistemas de arquivos por vez no modo interativo.

-t tipo

Permite especificar o tipo do sistema de arquivos a ser checado, por exemplo, `"ext3"` para o sistema de arquivos `"EXT3"`.

-A

Tenta realizar a checagem de todos os sistemas de arquivos que estão marcados para verificação no arquivo `"/etc/fstab"`. Para maiores detalhes sobre o arquivo `"/etc/fstab"`, consulte a seção com o nome do arquivo no capítulo `"Arquivos de configuração"`.

-C

Exibe o progresso da checagem.

-N

Não executada, apenas mostra o que seria feito.

-P

Quando a opção `"-A"` é especificada, faz a checagem do sistema de arquivos raiz (`"/"`) em paralelo com outros sistemas de arquivos. Este tipo de checagem não é muito seguro de ser praticado, evite-o.

-R

Quando se está checando todos os sistemas de arquivos com a opção `"-A"`, esta opção faz com que a checagem do sistema de arquivos raiz (`"/"`) seja evitada.

-T
Não exibe o título do programa durante o início da checagem.

-V
Exibe mais detalhes sobre as atividades que estão sendo realizadas pelo fsck.

Exemplos:

fsck.reiserfs /dev/hda2

Realiza a checagem do sistema de arquivos “/dev/hda2”, utilizando a extensão para o tipo de sistema de arquivos “ReiserFS”.

fsck -t ext3 -C /dev/sda4

Realiza a checagem do sistema de arquivos “/dev/sda4”, utilizando a extensão para o tipo de sistema de arquivos “EXT3” e exibindo o progresso da verificação.

8.1.8. last

Mostra uma listagem de entrada e saída de usuários no sistema. São mostrados os seguintes campos na listagem:

- Nome do usuário;
- Terminal onde ocorreu a conexão/desconexão;
- O nome da máquina (caso a conexão tenha ocorrido remotamente) ou console (caso tenha ocorrido localmente);
- A data em que o usuário entrou e saiu do sistema (*login/logout*), a hora (*login/down*) e caso ainda esteja utilizando o sistema, a mensagem “Still logged in”;
- O Tempo no formato “Horas:Minutos” em 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 comando “last” é obtida do arquivo de log “/var/log/wtmp”.

last [opções]

opções

-n [num]

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

-R

Não mostra o campo HostName (nome da máquina).

-a

Mostra o hostname (nome da máquina) 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, contudo, este argumento deve só ser válido se representar um nome de usuário, uma console ou se for a palavra “reboot”.

Exemplos:

last

Mostra a listagem geral de acessos ao sistema.

last -a

Mostra a listagem geral incluindo o nome da máquina.

last marcius

Mostra somente atividades do usuário “marcius”.

last reboot

Mostra os registros das reinicializações do sistema.

last tty1

Mostra todas as atividades no console “tty1”.

8.1.9. lastlog

Mostra qual foi a última vez que os usuários cadastrados no sistema fizeram *login* (entraram no sistema). É mostrado o nome usado no login, o terminal onde ocorreu a conexão e a hora da última conexão. Estes dados são obtidos através da pesquisa e formatação do arquivo “/var/log/lastlog”. Caso o usuário nunca tenha feito *login* no sistema consultado, é mostrada a mensagem “*** Never logged in ***”.

Este utilitário é muito importante para vigiarmos os acessos dos usuários ao sistema, mas para essa atividade ser efetiva, temos que observar que os registros de *log* presentes em “/var/log/lastlog” podem não estar completos devido as atividades de arquivamento do “**logrotate**”, e isso, deve ser levado em consideração quando os registros de acessos forem consultados.

lastlog [opções]

opções

-t [dias]

Mostra somente os usuários que se conectaram ao sistema nos últimos dias. Utilize o argumento “dias” para definir a quantidade de dias antes da data atual que deverão ser exibidos.

-u [usuário]

Mostra somente detalhes de acesso com relação ao usuário definido pelo argumento “usuário”.

Exemplos:

lastlog

Exibe a listagem completa de acessos ao sistema por usuário.

lastlog -t 3

Exibe a lista de acesso dos últimos 3 dias.

lastlog -u marcius

Exibe os registros de acesso do usuário “marcius”.

8.1.10. lsof

O utilitário “lsof” lista os arquivos, diretórios, *pipes*, bibliotecas, entre outros recursos do sistema, que estão sendo utilizados por um determinado processo. É um comando muito interessante de se utilizar, quando somos impedidos de desmontar um determinado volume no sistema, pois permite averiguar qual processo está utilizando o recurso em questão.

lsof [opções] [nome]

Onde:**nome**

Define o nome do arquivo, diretório, *pipe*, biblioteca, etc, que se deseja verificar se está sendo ou não utilizado por um determinado processo.

opções:**+D [diretório]**

Faz com que lsof procure por todas as instâncias do diretório especificado no argumento homônimo que estão sendo utilizadas pelos processos do sistema.

-N

Utilizado para verificar recursos (arquivos, diretórios, etc) que estão sendo utilizados em uma estrutura de rede baseada em NFS.

-p [PID]

Permite especificar o PID de um determinado processo para listar todos os recursos (arquivos, diretórios, etc) utilizados por ele.

-u [usuário/UID, ...]

Verificar os recursos (arquivos, diretórios, etc) em uso por usuários. Pode-se especificar múltiplos usuários pelo nome de *login* no sistema ou UID, separados por vírgula. É preciso ser administrador do sistema para verificar os recursos de outros usuários.

-U

Permite verificar todos os soquetes Unix que estão em uso no momento.

-T[sinalizadores]

Esta opção verifica o *status* das conexões TCP/IP, assim como o comando **"netstat"**. O parâmetro **"-T"** possui alguns sinalizadores que permitem filtrar os resultados do comando lsof, para mais detalhes consulte a documentação com o comando **"man"**.

-S [segundos]

Permite especificar o tempo limite em segundos para funções (lstat, readlink e stat) do *kernel*. O menor valor que pode ser especificado é 2, o padrão é 15, caso não seja definido um valor, o padrão é utilizado. Muito cuidado com este parâmetro, pois ele afeta o comportamento de funções importantes do sistema.

Exemplos:

lsof /var

Lista todos os recursos (arquivos, diretórios, *pipes*, etc) que estão em uso no diretório **"/var"**.

lsof +D /var/log

Lista todos os processos e os respectivos recursos (arquivos, diretórios, *pipes*, etc) que estão sendo utilizados no diretório **"/var/log"**.

lsof -p 2042

Lista todos os recursos (arquivos, diretórios, *pipes*, etc) utilizados pelo processo com PID **"2042"**.

lsof -u marcius,ze

Lista todos os recursos (arquivos, diretórios, *pipes*, etc) utilizados pelos usuários **"marcius"** e **"ze"** no sistema.

8.1.11. lspci

Permite verificar informações detalhadas sobre o barramento PCI e AGP e os dispositivos que estão plugados nestes barramentos. Na maioria das vezes, este utilitário consegue reconhecer e exibir o nome correto do controlador de cada dispositivo conectado ao barramento PCI/AGP do computador, em alguns casos contudo, os dispositivos podem ser mostrados como **"Unknow Device"** denotando o desconhecimento dispositivo. Para tentar contornar este tipo de situação, você pode usar o *script* **"update-**

pciids", que fará o *download* de um novo arquivo de definições para ser utilizado pelo utilitário "lspci". Em casos onde o acesso a Internet é feito via serviço de Proxy, é necessário ajustar a variável de ambiente "http_proxy" com o uso do comando "**export**" para que update-pciids funcione corretamente.

lspci [opções]

opções:

-v

Exibe informações detalhadas sobre cada componente presente no barramento PCI do computador.

-vv

Exibe informações ainda mais detalhadas do que se usado o parâmetro "-v".

-vvv

Exibe todas as informações disponíveis a respeito do *hardware* conectado ao barramento PCI da máquina.

-n

Exibe o código do fabricante e do dispositivo ao invés dos respectivos nomes.

-nn

Exibe o código do fabricante e do dispositivo, além dos respectivos nomes.

-b

Exibe a informação sobre o *hardware* e os respectivos números de IRQ utilizados por cada um dos componentes. Interessante para descobrir conflitos.

-t

Exibe informação no formato de árvore. Interessante utilizar em conjunto com o parâmetro "-v".

-d [cód. fabricante]:[cód. dispositivo]

Exibe somente informações dos dispositivos de um determinado fabricante (argumento "cód. fabricante"), ou ainda de forma mais restritiva, de um determinado dispositivo de um determinado fabricante (definindo os argumentos "cód. fabricante" e "cód. dispositivo").

Exemplos:

lspci

Exibe informações sobre os barramentos PCI/AGP e os dispositivos que estão conectados a estes barramentos.

lspci -v

Exibe mais detalhes sobre os barramentos PCI/AGP e os dispositivos conectados a eles.

lspci -n

Exibe o código do fabricante e dos dispositivos conectados aos barramentos PCI/AGP do computador.

lspci -d 10ec:8139

Exibe informações a respeito do dispositivo com código "8139" do fabricante com código "10ec".

8.1.12. lsusb

Permite listar os dispositivos USB conectados ao computador. Útil para exibir informação sobre os dispositivos removíveis do sistema.

`lsusb [opções]`

opções:

`-v`

Exibe informações detalhadas dos dispositivos USB conectados ao sistema.

`-s [barramento]:[dispositivo]`

Permite visualizar informação específica sobre o dispositivo conectado a um determinado barramento USB do computador ou sobre todos os dispositivos plugados a um determinado barramento (especificando somente o argumento "barramento").

`-D [dispositivo]`

Faz com que, ao invés do comando `lsusb` verificar o diretório `/proc/bus/usb` para obter informações sobre os dispositivos USB conectados ao computador, ele exiba informações específicas de acordo com o argumento "dispositivo" fornecido pelo usuário. O argumento "dispositivo" deve ser definido como, por exemplo, `/proc/bus/usb/001/001`.

Exemplos:

`lsusb`

Exibe informações a respeito de todos os dispositivos USB conectados ao sistema.

`lsusb -v`

Exibe informação detalhada, sobre todos os dispositivos conectados aos barramentos USB disponíveis na máquina.

`lsusb -s 002:001`

Mostra informações sobre o primeiro dispositivo USB (argumento "001") conectado ao segundo barramento (argumento "002") USB da máquina.

`lsusb -D /proc/bus/usb/001/002`

Mostra informações específicas sobre o segundo dispositivo conectado ao primeiro barramento USB da máquina.

8.1.13. `mount`

O "mount" é um utilitário usado para disponibilizar, algo também conhecido como montar, um determinado sistema de arquivos. Algumas vezes, temos a necessidade de alterar as opções da montagem de um sistema de arquivos que já se encontra disponível (montado) no sistema, sendo esta uma operação muito útil para manutenção do sistema de arquivos, nessas ocasiões, a execução do comando com a opção "remount" pode ser necessária. Veja o exemplo abaixo:

```
mount -t ext3 /dev/hda2 /boot -o remount,ro
```

Onde:

- O parâmetro "-t" seguido do argumento "ext3", informa o tipo do sistema de arquivos a ser disponibilizado, neste caso o "ext3".
- O argumento "/dev/hda2" é o sistema de arquivos que se deseja montar.
- O argumento "/boot" é o ponto de montagem deste sistema de arquivos.
- O parâmetro "-o" permite que as opções de montagem sejam informadas, neste caso, os argumentos "remount,ro" que fazem com que o sistema de arquivos seja "remontado" (remount) em modo somente leitura (ro).

8.1.14. `nice`

Configura a prioridade da execução de um programa. Para reajustar a prioridade de um programa já em execução, utilize o comando "renice", para maiores detalhes, consulte as informações sobre o renice na sequência desta seção.

`nice [opções] [comando]`

**Onde:
comando**

Comando/programa que terá sua prioridade ajustada.

**opções:
-n [numero]**

Configura a prioridade que o programa será executado. Se um programa for executado com maior prioridade, ele usará mais recursos do sistema para seu processamento, caso tenha uma prioridade baixa, ele permitirá que outros programas tenham preferência no uso dos recursos. A prioridade de execução de um programa pode ser ajustada de -20 (a mais alta) até 19 (a mais baixa). Por padrão, os programas são executados com prioridade igual a 10. Para ajustar prioridades altas, abaixo de 0 (zero), é necessário ter poderes de administrador do sistema (usuário "root").

Exemplos:

nice -n -19 find / -name apropos

Configura a prioridade de execução do comando "find" para -19 (muito alta).

nice -n -1 top

Configura a prioridade de execução do comando "top" para -1 (alta).

8.1.15. renice

Este utilitário altera a prioridade de um ou mais programas em execução. É necessário ter poderes de administrador do sistema (usuário "root") para conceder prioridade mais alta do que a atual a um determinado processo. O comando renice, necessita do PID (*Process Identification Number*) do processo que se deseja alterar a prioridade. Você também pode informar o UID (*User Identification Number*) do usuário ao renice, dessa forma, você poderá reajustar a prioridade de todos os programas daquele usuário.

renice [prioridade] [opções]

**Onde:
prioridade**

É o número que define a nova prioridade do processo, pode variar de -20 (prioridade mais alta) à 19 (prioridade mais baixa).

opções

-g

Permite especificar um ou mais grupos, para alterar a prioridade de todos os processos, que estejam sendo executados sob a permissão destes grupos.

-u

Permite especificar um ou mais usuários, para alterar a prioridade de todos os processos, que estejam sendo executados sob a permissão destes usuários.

-p

Permite especificar um ou mais PID para que a prioridade dos processos com estes números de identificação sejam alterados.

Exemplos:

renice 0 1335

Altera a prioridade do processo com PID 1335 para 0 (zero).

renice -5 987 -u root marcius -p 900 800

Faz com que os processos com PID 987, 900 e 800 e todos os processos dos usuários "root" e "marcius" tenham sua prioridade alterada para -5.

renice -1 -g cppc

Altera a prioridade de todos os processos que estejam sendo executados sob permissão do grupo "cppc" para -1.

8.1.16. nohup

Executa um comando ignorando os sinais de interrupção. O comando poderá ser executado até mesmo em segundo plano, caso o usuário que tenha iniciado a execução do comando tenha saído do sistema.

nohup [comando]

As mensagens de saída do comando nohup são direcionadas para o arquivo "~/nohup.out".

Exemplos:

nohup find / -uid 0 >/tmp/rootfiles.txt &

nohup updatedb &

8.1.17. pidof

Retorna o PID do processo especificado.

pidof [opções] [nome]

Onde:

nome

Nome do processo que se deseja obter o número PID.

opções

-s

Retorna somente o primeiro PID encontrado.

-x

Retorna o PID do *shell* que está executando o *script*.

-o [PID]

Omite um determinado PID da listagem de saída do comando.

Exemplos:

pidof nmbd

Retorna o PID do processo chamado "nmbd".

pidof smbd nmbd sshd bash -o 2355

Retorna uma lista de PID com exceção do PID 2355.

8.1.18. pstree

Mostra a estrutura de processos em execução no sistema em forma de árvore.

pstree [opções] [pid]

Onde:

pid

Número do processo que terá sua árvore listada. Se omitido, lista todos os processos.

opções

- a
Mostra opções passadas na linha de comando.
- c
Mostra toda a estrutura (inclusive sub-processos do processo pai).
- G
Usa caracteres gráficos no desenho da árvore de processos.
- h
Destaca o processo atual e seus antecessores.
- H [PID]
Destaca o processo especificado.
- l
Não faz quebra de linha.
- n
Classifica pelo número PID ao invés do nome.
- p
Mostra o número PID entre parênteses após o nome do processo.
- u
Mostra também o dono do processo.
- U
Usa o conjunto de caracteres Unicode para o desenho da árvore.

Exemplos:

pstree

Mostra a árvore de processos.

pstree -Ap

Mostra a árvore de processos indentada por caracteres ASCII e juntamente com cada processo o número de seu PID.

8.1.19. shred

É um utilitário para ocultação de conteúdo. Com ele é possível sobrescrever o conteúdo de um arquivo de modo que sua recuperação seja extremamente complexa e na maioria das vezes impossível. Opcionalmente, também é possível excluir o arquivo.

Quando se trata de dados confidenciais, é necessário tomar as devidas precauções no momento de apagar uma informação, e em muitos casos, os administradores recorrem a *scripts* que envolvem o dispositivo gerador de *bits* aleatórios “/dev/random”, o que torna a operação mais trabalhosa. Nestes casos, o uso do comando “shred” pode ser bastante prático e muito útil.

shred [opções] arquivo

Onde:

arquivo

É o arquivo alvo da operação de shred.

opções:

-f

É o modo “força bruta”. Muda inclusive a permissão do arquivo, se necessário, para permitir a escrita de dados no arquivo.

-n número

Escreve aleatoriamente o arquivo o número de vezes definido pelo argumento "número". O padrão é 25 vezes.

-s tamanho

Permite especificar o tamanho que o arquivo deverá ter após sobrescrito. O tamanho pode ser definido usando os sinalizadores "K" para kilobytes, "M" megabytes e "G" gigabytes.

-u

Sobrescreve a informação contida no arquivo e o remove após o término dessa operação.

-x

Não arredonda o tamanho do arquivo até o próximo bloco de dados cheio. Está é uma ação padrão para arquivos não-regulares.

-z

Após sobrescrever o arquivo, preenche o mesmo com bits de valor zero para evitar a detecção do uso do comando shred.

Exemplos:

shred teste.txt

Sobrescreve os dados contidos no arquivo "teste.txt".

shred -s 1M teste.txt

Sobrescreve os dados contidos no arquivo "teste.txt", sendo que ao final da operação o arquivo terá 1 megabyte de tamanho.

shred -u -z teste.txt

Sobrescreve, ajusta todos os bits do arquivo "teste.txt" com valor zero e depois o apaga.

8.1.20. stat

Mostra informações detalhadas sobre um *inode* (arquivo do ponto de vista do SO). As informações exibidas pelo utilitário são: número do dispositivo, numero *inode*, direitos de acesso, números de *links* absolutos, ID's de usuário e grupo, tipo de dispositivo para o *inode*, tamanho total em *bytes*, número de blocos alocados, tamanho do bloco, data e hora do último acesso, última modificação e alteração e o contexto de segurança.

stat [opções] arquivos

Onde:

arquivos

São os arquivos aos quais desejamos obter as informações.

opções:

-c formato

Exibe a saída conforme especificado no argumento "formato".

-f

Exibe informações sobre o sistema de arquivos onde o arquivo está localizado e não sobre o arquivo em si.

-L

Esta informação serve para obter informações de um arquivo que é referenciado por um link simbólico. Você deve utilizar esta opção, apontando para um link simbólico em vez de um arquivo comum.

-t
Imprime a saída na forma concisa. Útil para análise por outros programas.

Exemplos:

stat teste.txt

Exibe informações detalhadas sobre o *inode* do arquivo “teste.txt”.

stat -L meulinksimbolico

Exibe informação sobre o arquivo referenciado pelo *link* simbólico “meulinksimbolico”.

stat -f teste.txt

Mostra informações a respeito do sistema de arquivo sob o qual o arquivo “teste.txt” é armazenado.

8.1.21. sync

Força a gravação dos dados que estão no *cache* de disco (em memória) fisicamente nos dispositivos de armazenamento. O *cache* é um mecanismo de aceleração que permite que os dados sejam armazenados temporariamente na memória ao invés de serem imediatamente gravados no dispositivo de armazenamento, quando o sistema estiver ocioso, é realizada a persistência dos dados no dispositivo de armazenamento. O GNU/Linux procura utilizar toda memória RAM disponível para o *cache* de programas acelerando seu desempenho de leitura/gravação.

O sync é bastante útil quando temos que desligar o sistema emergencialmente e não podemos executar o desligamento no modo normal (com o comando “halt” ou “shutdown”), a execução do sync nestes casos, evita a perda de dados.

sync

8.1.22. tar

Muitas pessoas imaginam que o “tar” seja um poderoso compactador, isto não é verdade. Na realidade, este utilitário é um arquivador, ou seja, possui a capacidade de reunir vários arquivos/diretórios num mesmo arquivo, sendo a compactação deste arquivo uma opção para o usuário. O tar pode atuar juntamente com os compactadores: gzip (.gz), bzip2 (.bz2) e compress (.Z). Este é um comando extremamente útil na geração de cópias de segurança.

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.

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 [arquivo]
Usa o argumento “[arquivo]” especificado para geração do arquivo “.tar”.
- 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 uma única mídia como um disquete, por exemplo, seja dividida em mais de uma mídia.
- O
Grava o arquivo no formato VT7 ao invés do ANSI.
- O, --to-stdout
Descompacta arquivos para a saída padrão (monitor) 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]
Define o nome do volume em que arquivo tar será guardado. Útil para utilização com fitas de backup.
- W, --verify
Tenta verificar o arquivo gerado pelo tar após gravá-lo.
- x
Extraí arquivos gerados pelo tar.
- X [arquivo]
Tenta apagar o arquivo informado como argumento em “[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 de compressão definido no argumento “[programa]” durante o processamento dos arquivos.

A extensão precisa ser especificada no arquivo de destino para sua correta identificação:

- 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. Uma dica importante é a de usar o comando “file” no arquivo tar para descobrir com que utilitário ele foi compactado.

Exemplos:

tar -cf index.txt.tar index.txt

Cria um arquivo chamado “index.txt.tar” que armazenará o arquivo “index.txt”. Você pode notar usando o comando “ls -lh”, que o arquivo foi somente arquivado sem compactação, isto é útil para juntar diversos arquivos em um só.

tar -xf index.txt.tar

Extraí o arquivo “index.txt” do arquivo tar “index.txt.tar”.

tar -czf index.txt.tar.gz index.txt

O mesmo que o exemplo de arquivamento anterior, só que agora é usado compactação a opção “-z” (compactação através do programa gzip). Você agora pode notar digitando “ls -lh” que o arquivo “index.txt” foi compactado e depois arquivado no arquivo “index.txt.tar.gz”.

tar -cvf /dev/st0 /home

Faz o arquivamento dos diretórios que se encontram em “/home” (diretórios pessoais dos usuários) no dispositivo de fita “/dev/st0” (drive de fita SCSI).

tar -xzf index.txt.tar.gz

Descompacta e extraí o arquivo “index.txt” a partir do arquivo “index.txt.tar.gz”.

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”.

8.1.23. watch

Executa um comando de forma repetida, por padrão a cada 2 segundos, e exibe a saída do comando em tela cheia. O “watch” também permite monitorar a diferença entre os resultados obtidos com os comandos executados, algo realmente útil para avaliação de resultados.

`watch [opções] comando`

Onde:

comando

É o comando que se deseja utilizar e que o watch irá executar repetidamente. É interessante especificar o comando utilizando aspas, como por exemplo, “ls -lh /tmp”.

opções:**-d, --differences[=cumulative]**

Realça as diferenças encontradas na saída do comando executado. Se o parâmetro "cumulative" for especificado, o watch irá mostrar de forma realçada tudo o que foi alterado desde a primeira execução do comando.

-n segundos

Define o intervalo de tempo em segundos, através do argumento homônimo, em que o watch deverá executar o comando especificado pelo usuário.

-t

Não exibir o cabeçalho do comando watch.

Exemplos:

watch -n 5 "cat teste.txt"

Executa o comando "cat teste.txt" a cada 5 segundos e exibe a saída na tela.

watch -t -n 3 --differences=cumulative "cat -n teste.txt"

Executa o comando "cat -n teste.txt" a cada 3 segundos, não exibindo o cabeçalho do watch e marcando as alterações sofridas pelo arquivo desde a primeira execução do comando.

8.1.24. which

Lista o caminho completo do comando que está sendo executado. Utilizado principalmente por *scripts shell*. Para realizar a busca pelo comando a ser executado, o "which" usará a variável de ambiente "PATH".

which [opções] comando

Onde:**comando**

É o comando que desejamos saber qual é o caminho completo para o executável. Deve ser informado sem o uso de qualquer parâmetro.

opções:**-a**

Exibe todos os caminhos em que o comando especificado foi encontrado e não só o primeiro (a partir de onde ele é prioritariamente executado).

Exemplos:

which -a ls

Imprime na tela, todos os caminhos encontrados para o executável "ls".

8.2. Comandos de Rede

8.2.1. arp

Todos pacotes IP que circulam numa rede, devem conter o endereço IP e o MAC (*Media Access Control*) do remetente e do destinatário. Para se obter o endereço MAC do computador de destino, o protocolo ARP (*Address Resolution Protocol*) envia um *broadcast* com o IP do destinatário para a rede, requisitando o endereço MAC do mesmo. Para racionalizar o uso do recurso de *broadcast* na rede, o SO mantém uma tabela, chamada "tabela de *cache* ARP", que contém os endereços IP e os respectivos MAC das máquinas com as quais o computador local se comunica com maior frequência.

O comando "arp", permite exibir, adicionar e remover entradas da tabela de *cache* ARP do sistema. As entradas de uma tabela ARP podem ser marcadas pelos seguintes sinalizadores: "C" (completo) que representa um registro comum, "M" (permanente) que representa um registro estático e "P" (publicação) que representa um proxy ARP. Estes sinalizadores, podem ser conferidos através da coluna chamada "Flags", durante a visualização da tabela cache ARP.

arp [opções] [nome/IP] [MAC]

Onde:

endereço/IP

É o endereço IP ou o nome da máquina que desejamos adicionar a tabela de cache ARP do sistema.

MAC

É o endereço físico (MAC) correspondente a máquina que desejamos adicionar a tabela cache ARP do sistema.

opções:

-a [nome/IP]

Exibe as entradas da tabela em relação ao endereço IP / nome da máquina fornecido como argumento. Caso não seja fornecido nenhum argumento, exibe todas as entradas da tabela.

-d nome/IP [pub]

Remove o registro de uma máquina da tabela. Você deve informar o nome da máquina ou endereço IP correspondente ao registro que deseja excluir da tabela. Informe o parâmetro "pub", caso queira excluir uma entrada de um proxy ARP.

-D

Permite especificar uma interface de rede associada a um determinado endereço físico (MAC). Esta opção pode ser utilizada em conjunto com o parâmetro "-s" para criação de uma entrada de proxy ARP.

-f arquivo

Lê as entradas a partir de um arquivo fornecido como argumento e as adiciona a tabela.

-H tipo

Permite especificar o tipo de protocolo a ser utilizado na camada de rede. Exemplos de protocolos que podem ser utilizados como argumento são: "ether" para redes ethernet (este é o padrão), "ax25" protocolo de rede via rádio, "arcnet", "pronet" e "netrom".

-i interface

Se você estiver exibindo dados da tabela, esta opção fará com que os dados a serem mostrados sejam filtrados, imprimindo na tela somente as entradas correlatas a

interface informada através do argumento fornecido. No caso de adicionar uma entrada a tabela, esta opção fará com que aquela interface, seja associada ao endereço que está sendo configurado (como na opção "-D").

-n
Exibe os endereços IP, em vez dos nomes, de cada entrada da tabela de cache ARP.

-s endereço/IP MAC [pub]
Adiciona uma entrada permanente na tabela cache ARP para a máquina a qual foi fornecido o argumento "endereço/IP". O parâmetro "pub" pode ser utilizado para criar uma entrada de proxy ARP.

Exemplos:

arp

Exibe o conteúdo completo da tabela cache ARP.

arp -a 10.15.15.15

Exibe a entrada correspondente ao endereço "10.15.15.15". Caso não haja nenhuma entrada na tabela que corresponda ao endereço fornecido, um erro será reportado ao usuário.

arp -n

Mostra as entradas da tabela cache ARP usando endereços IP em vez de nomes no formato DNS.

arp -s 10.15.15.15 00:01:a0:5f:55:ab -i eth0

Cria uma entrada permanente na tabela cache ARP, para a máquina com endereço "10.15.15.15", associando esta máquina ao endereço físico "00:01:a0:5f:55:ab" e a interface de rede "eth0".

8.2.2. dig

É usado para realizar pesquisas em servidores DNS. É mais flexível e exibe informações mais detalhadas que o obsoleto "nslookup". É muito utilizado para descobrir problemas relacionados a estrutura de resolução de nomes DNS numa rede. O comando "dig", utiliza todos servidores DNS cadastrados no arquivo "/etc/resolv.conf" para realizar as pesquisas efetuadas pelo usuário.

dig [servidor] [opções] [alvo]

Onde:

servidor

É o nome ou endereço IP do servidor DNS que deve ser consultado para realizar a pesquisa pelo domínio que se deseja pesquisar.

alvo

É o domínio ou endereço IP que se deseja pesquisar (ex.: "www.debian.org").

opções:

-b endereço

Especifica o IP da máquina de origem sob o qual a pesquisa deve ser realizada. Útil para máquinas com múltiplos endereços IP.

-f arquivo

Faz com o dig opere no modo batch. Dessa forma, é possível especificar um arquivo (pelo argumento "arquivo") que contenha as pesquisas que devem ser feitas pelo comando dig. O arquivo deve ser organizado da mesma forma como uma pesquisa dig feita através da linha de comandos.

-p porta

Especifica uma porta para acesso ao servidor DNS para realizar a consulta. Por padrão, a porta 53 é utilizada.

-x endereço

Realiza a pesquisa reversa, ou seja, permite que o usuário especifique um endereço TCP/IP para que este seja resolvido, um nome correspondente para aquele endereço será retornado como resultado.

Exemplos:

dig 10.15.61.248 -p 53 www.debian.org

Pesquisa quais são os servidores que respondem pelo domínio "www.debian.org", através do servidor DNS "10.15.61.248" usando a porta 53.

dig -x 10.15.61.248

Faz a pesquisa reversa para o endereço "10.15.61.248".

8.2.3. ifconfig

O utilitário "ifconfig", é utilizado para configurar e exibir informações sobre as interfaces de rede do computador. Diferentemente do "mii-tool" (veja este comando mais adiante), ele permite verificar e ajustar os protocolos de rede que o sistema irá utilizar para se comunicar em rede. Durante o processo de boot da máquina, ifconfig é invocado para configurar as interfaces de rede do computador de acordo com as configurações gravadas no arquivo "/etc/network/interfaces".

ifconfig [-v] [-a] [-s] [interface] [opções]

Onde:

interface

É a interface de rede que desejamos consultar ou modificar. Exemplos de interface de rede são a eth0, etho:0, eth1, etc.

opções:

-a

Disponibiliza informação sobre todas as interfaces atualmente disponíveis.

-s

Exibe uma tabela resumida sobre envio/recebimento de dados nas interfaces, funciona analogamente ao comando "**netstat -i**" (veja mais detalhes sobre este comando mais adiante).

-v

Mostra informações mais detalhadas, principalmente em algumas situações de erro específicas.

up

É um sinalizador que faz com que a interface alvo seja ativada, ou seja, faz com que ela fique disponível para que o sistema possa utilizá-la.

down

Assim como "up" é um sinalizador, porém, tem função inversa a deste, sendo "down", responsável pela desativação de uma interface de rede.

[-]arp

Ativa ou desativa (usando o sinal de "-") o uso do protocolo ARP na interface.

[-]promisc

Ativa ou desativa (usando o sinal de "-") o modo de transmissão de pacotes de rede promíscuo.

[_]allmulti
Ativa ou desativa (usando o sinal de "-") o modo de transmissão de pacotes de rede multicast.

metric número
Este parâmetro ajusta a métrica de rede, definida pelo argumento "número", para a interface alvo.

mtu número
Este parâmetro ajusta a "Unidade Máxima de Transferência" (MTU) da interface.

netmask endereço
Ajusta o endereço da máscara da sub-rede, definido pelo argumento "endereço", do protocolo TCP/IP.

irq endereço
Ajusta o endereço de IRQ do dispositivo em questão. Opção utilizada para placas de rede antigas que não suportam mudança dinâmica de IRQ.

io_addr endereço
Configura o endereço de IO do dispositivo.

mem_start endereço
Configura uma área de memória compartilhada para ser usada pelo dispositivo.

[_]broadcast [endereço]
Permite ajustar o endereço de broadcast da interface ou ativar e desativar (usando o sinal de "-") o sinalizador IFF_BROADCAST.

multicast
Configura a interface para o uso do modo de transmissão de pacotes de rede multicast.

address endereço
Configura o endereço TCP/IP da interface, através do argumento "endereço".

Exemplos:

ifconfig

Exibe informações de todas as interfaces de rede ativas do computador.

ifconfig -v eth0

Exibe informações sobre a interface "eth0".

ifconfig eth0 192.168.1.1 netmask 255.255.255.0

Configura e ativa a interface de rede "eth0", com endereço de rede "192.168.1.1" e máscara de sub-rede "255.255.255.0".

8.2.4. ifup e ifdown

O comando "ifup" é usado para ativar interfaces de rede. Ao contrário, "ifdown" é utilizado para desativá-las. A ativação de uma determinada interface de rede, é feita com base no arquivo de configuração "/etc/network/interfaces".

ifup/ifdown [opções] [interface]

Onde:

interface

É o dispositivo de rede que desejamos ativar, por exemplo, "eth0".

opções:**-a**

Ativa/desativa todas as interfaces de rede definidas com a cláusula "auto" no arquivo "/etc/network/interfaces".

--force

Força a ativação/desativação da interface de acordo com as opções definidas pelo usuário.

-i arquivo

Permite que a configuração de ativação/desativação das interfaces sejam feitas com base num arquivo diferente de "/etc/network/interfaces". Utilize o argumento "arquivo" para definir qual será o arquivo de configuração alternativo a ser usado.

Exemplos:**ifup -a**

Ativa todas as interfaces de rede da máquina, que contém a cláusula "auto" em sua configuração no arquivo "/etc/network/interfaces".

ifdown -i arquivo_configuracao.conf

Desativa as interfaces de rede de acordo com as configurações do arquivo alternativo "arquivo_configuracao.conf".

ifup eth0

Ativa a interface de rede "eth0".

ifdown eth1

Desativa a interface de rede "eth1".

8.2.5. mii-tool

O "mii-tool" é uma ferramenta muito versátil, que possibilita ao administrador do sistema obter várias informações a respeito das interfaces de rede que o computador possui, bem como, modificar o estado da conexão física de rede (*link*) de cada uma delas.

É necessário ter privilégios de administrador do sistema para poder usar esta ferramenta.

mii-tool [opções] [interface]

Onde:**interface**

É a interface de rede que desejamos consultar ou modificar. Exemplos de interface de rede são a eth0, eth1, etc.

opções:**-v**

Exibe informações com maior nível de detalhamento.

-R

Reconfigura a interface com as opções padrão para aquele dispositivo.

-r

Reinicia a negociação pela taxa de transferência de dados da interface.

-w

Monitora mudança de status em um link numa determinada interface.

-l

Utilizado em conjunto com "-w", grava as mudanças de status do link de uma

interface no arquivo de log do sistema “/var/log/syslog”.

-F

Desabilita o modo de auto-negociação e possibilita que o administrador defina a velocidade de transferência de dados da interface. Os modos a serem definidos manualmente podem ser: 1000baseTx-FD, 1000baseTx-HD, 100baseTx-FD, 100baseTx-HD, 10baseT-FD, ou 10baseT-HD.

Exemplos:

mii-tool -v eth0

Exibe informações detalhadas sobre a interface de rede “eth0”.

mii-tool -w eth0

Monitora a mudança da taxa de transferência de dados na interface de rede “eth0”.

mii-tool eth0 -F 100baseTx-FD

Define a velocidade de transferência de dados da interface de rede “eth0” para “100baseTx-FD”.

8.2.6. netstat

Mostra o *status* das conexões de rede, tabelas de roteamento, estatísticas das interfaces, conexões mascaradas e mensagens.

`netstat [opções]`

opções:

-i [interface]

Mostra estatísticas da interface [interface].

-M, --masquerade

Se especificado, também lista conexões mascaradas (*IP Masquerading*).

-n, --numeric

Usa endereços numéricos ao invés de tentar resolver nomes das máquinas, usuários e portas.

-p

Exibe o PID, nome do programa e o respectivo soquete que está sendo utilizado para comunicação.

-c, --continuos

Mostra a listagem atualizada a cada segundo até que as teclas **<CTRL> + <C>** sejam pressionadas.

Se não for especificada nenhuma opção, os detalhes das conexões atuais serão mostradas.

Exemplos:

netstat

Exibe o status de todas as conexões de rede da máquina.

netstat -p

Exibe as informações de conexão do sistema, exibindo o PID, o nome de cada executável e o soquete específico utilizado para comunicação.

8.2.7. ntpdate

Ajusta a data e a hora do computador local via NTP (Network Time Protocol). O utilitário “ntpddate”, possibilita consultar um servidor NTP para sincronizar a data e hora local do computador. É de grande valia para manter a data e a hora sempre corretas, evitando problemas em registros de *log*, bancos de dados e outros serviços que dependem dos dados do calendário e relógio do sistema.

O ntpdate pode ser executado manualmente, via uma tarefa agendada pelo **cron**, via *scripts*, principalmente durante o *boot* do sistema, ou ainda, através de um *daemon* dedicado quando se necessita do máximo de precisão e otimização. Em geral, para estações e servidores de rede que não necessitem de muita precisão (décimos de segundos, por exemplo), utiliza-se *scripts* de *boot* e tarefas agendadas no cron para manter a data e a hora do computador ajustadas corretamente.

ntpdate [opções] servidor

Onde:

servidor

É a maquina que provê o serviço NTP para a rede, de modo que outros computadores possam sincronizar seu calendário e relógio local com o desse servidor.

opções:

-a chave

Permite especificar uma chave criptográfica para realizar a autenticação no servidor NTP. Tanto cliente como o servidor devem possuir chaves correspondentes entre si. Por padrão, a função de autenticação é desabilitada.

-k arquivo

Possibilita que o administrador especifique um arquivo contendo a chave criptográfica para realizar autenticação no servidor NTP. Por padrão, o arquivo “/etc/ntp.keys” é utilizado para esta função.

-p número

Define o número de amostras de tempo que devem ser adquiridas de cada servidor NTP para realizar o calculo do tempo. Pode ser utilizado um número de amostras de 1 à 8, o padrão é 4.

-q

Faz somente a pesquisa pela data e hora no servidor NTP sem alterar o calendário e o relógio do computador local.

-s

Faz com que a saída do comando ntpdate seja redirecionada para o arquivo de log do sistema (“/var/log/syslog”) ao invés da saída padrão. Especialmente interessante, quando o ajuste da hora e data é feito via agendamento com o cron.

-t segundos

Especifica o tempo em que o cliente NTP aguardará por uma resposta de um servidor. O tempo é especificado em múltiplos de 0.2 segundos, sendo o tempo de espera padrão, 1 segundo.

-u

Força a utilização de portas não-privilegiadas para comunicação com um servidor NTP. Útil em situações onde o servidor NTP encontra-se protegido por um firewall, por exemplo. Portas de comunicação TCP não-privilegiadas são as portas de 1024 à 65535.

Exemplos:

ntpdate -s ntp.pr.gov.br

Sincroniza a data e a hora do computador local usando o servidor NTP “ntp.pr.gov.br” e direciona a saída do comando ntpdate para o arquivo de log do sistema.

ntpdate -q ntp.pr.gov.br

Faz uma pesquisa para saber a data e a hora corrente no servidor NTP “ntp.pr.gov.br”.

Exemplo de agendamento do ntpdate via cron

0 ** root ntpdate -s ntp.pr.gov.br**

Fim do exemplo

Agenda a execução do ntpdate para sincronização com o servidor NTP “ntp.pr.gov.br”, a cada hora do dia, em todos os dias da semana, direcionando a saída do comando para o arquivo de log do sistema.

Para realizar a sincronização do calendário e da hora durante a inicialização do sistema, é possível acrescentar a seguinte linha ao final do arquivo “/etc/init.d/bootmisc.sh”, por exemplo:

ntpdate -s ntp.pr.gov.br

Não esqueça de ajustar o endereço do servidor NTP conforme sua necessidade. Para clientes da CELEPAR, é recomendável o uso do servidor que citamos nos exemplos (ntp.pr.gov.br).

8.2.8. route

O comando “route” permite adicionar e remover rotas TCP/IP na tabela de roteamento do Linux. Para modificar a tabela de roteamento, é necessário privilégios de administrador do sistema. O roteamento de pacotes, é técnica que permite guiar os dados transmitidos pela rede através de outras interfaces de rede ou roteadores (equipamento dedicado ao roteamento) com o objetivo de atingir um destino numa rede diferente daquela em que o computador de origem faz parte.

route [add/del] [opção] [endereço]

Onde:**add/del**

Permite adicionar (parâmetro “add”) ou remover (“del”) uma rota estática TCP/IP.

endereço

Para alguns parâmetros que utilizamos com o comando route, é necessário fornecer um endereço TCP/IP relativo a função da opção a ser usada. Por exemplo, a opção “net”, exige que seja informado um endereço de rede como “192.168.1.0”.

opções:**-A**

Permite verificar ou modificar a tabela de roteamento especificando a família do protocolo a ser alterado. Exemplos de família para serem utilizadas com este parâmetro são: inet (IPv4, é o padrão), inet6 (IPv6), ax25 (AX.25), netrom (NET/ROM AMPR), ipx (Novell IPX), ddp (Appletalk) e x25 (X.25).

-F

Altera informações contidas na “Base de Informações de Repasse” do Kernel (Kernel FIB), que é na verdade, a tabela de roteamento padrão do sistema. Este é o padrão para as operações do comando route.

-C

Modifica o cache de roteamento do Kernel (mantido em memória).

-n

Exibe os endereços no formato numérico ao invés de tentar resolver nomes.

-e

Mostra a tabela de roteamento no formato do comando “**netstat**”.

target endereço

Permite especificar tanto um host (maquina ou equipamento de rede) ou uma rede de destino. Pode ser especificado numericamente ou através de nomes.

-net endereço

Permite especificar a rede de destino através do uso do argumento “endereço”.

-host endereço

Permite especificar um host (maquina ou equipamento de rede) de destino através do uso do argumento “endereço”.

netmask mascara

Permite especificar uma mascara de sub-rede, usando o argumento “mascara”, para o host ou rede de destino que está sendo configurado.

gw roteador

Permite definir o endereço do roteador padrão da interface, através do uso da cláusula “default” antes do parâmetro em questão, ou do roteador que dá acesso à rede que está se estabelecendo a rota, através do uso do argumento “roteador”.

metric número

Define a métrica de rede a ser utilizada para uma determinada rota.

mss número

Permite especificar o “Tamanho Máximo do Segmento” (MSS) para a rota. Este parâmetro pode ser utilizado quando se deseja trafegar pacotes de rede do menor tamanho possível e quando a operação de negociação do tamanho MTU (Maximum Transfer Unit) não funciona corretamente para aquela rede.

reject

Configura um bloqueio para uma determinada rota. Não é o mesmo funcionamento que um firewall possui. Esta cláusula quando informada, simplesmente faz com que a procura por uma determinada rota falhe, ou seja, é como se a rota para uma determinada rede ou host não existisse realmente.

dev interface

Permite especificar a interface de rede para onde os pacotes deverão ser direcionados para atingirem a rede ou host de destino.

Exemplos:

route

Exibe a tabela de roteamento do Kernel. A coluna chamada “Opções”, mostrada quando executamos o comando dessa forma, possui os seguintes símbolos e os respectivos significados:

U = A rota está ativa.

H = O destino é um host e não uma rede.

G = Usar roteador.

R = Reintegrar rota para roteamento dinâmico.

D = Rota instalada dinamicamente por um daemon.

M = Rota modificada ou redirecionada por um daemon.

A = Instalada por “addrconf”.

C = Entrada em cache.

! = Rota rejeitada.

route add -net 192.56.76.0 netmask 255.255.255.0 dev eth1

Adiciona uma rota para a rede “192.56.76.0” com mascara classe C, através do dispositivo de rede (interface) “eth1”.

route add default gw 192.168.0.1 dev eth0

Adiciona o roteador padrão para a rede a qual o computador faz parte, utilizando a interface “eth0”.

route add -net 224.0.0.0 netmask 240.0.0.0 gw 192.168.1.1 metric 0 dev eth1

Adiciona uma rota para a rede “224.0.0.0” com mascara “240.0.0.0”, através do roteador “192.168.1.1”, estabelecendo a métrica mínima (valor zero) para se alcançar esta rede usando a interface “eth1”.

route del -net 10.0.0.0 netmask 255.0.0.0

Exclui a rota para a rede “10.0.0.0” com mascara “255.0.0.0” da tabela de roteamento.

route add -net 10.0.0.0 netmask 255.0.0.0 reject

Faz com que pesquisa por rotas para a rede “10.0.0.0” com mascara “255.0.0.0”, sejam rejeitadas, como se a rota não existisse de fato, porém, não exclui a rota da tabela de roteamento do Kernel.

route del -net 10.0.0.0 netmask 255.0.0.0 reject

Apaga a rota rejeitada para a rede “10.0.0.0” com mascara “255.0.0.0”. Quando uma rota esta rejeitada, para apagá-la é necessário usar a cláusula “reject”.

8.2.9. tcpdmatch

Simula uma tentativa de conexão via TCP/IP ao sistema. Esta ferramenta é bastante útil para verificação de segurança da maquina local. O “tcpdmatch” checa as regras existentes nos arquivos de configuração “/etc/hosts.allow” e “/etc/hosts.deny” para determinar se uma tentativa de conexão originada de um determinado endereço IP, maquina ou domínio obterá sucesso no acesso a determinado *daemon* que esteja sendo executado na maquina.

As checagens do tcpdmatch, não levam em consideração as proteções proporcionadas por *firewalls* como o “iptables”.

tcpdmatch [opções] daemon endereço

Onde:

daemon

É o executável principal do serviço que se deseja verificar a possibilidade de acesso. Exemplos de *daemons* são: sshd, ftpd, mysqld, etc.

endereço

É o endereço IP, nome da maquina ou domínio de origem da conexão. Serve para simular uma conexão de entrada direcionada para a maquina local.

opções:

-d

Faz com que tcpdmatch considere as configurações dos arquivos “hosts.allow” e “hosts.deny” do diretório local em vez dos que estão presentes sob o diretório “/etc”.

-i arquivo

Permite especificar um arquivo de configuração alternativo nos moldes do “inetd.conf”, em vez do arquivo padrão localizado sob o diretório “/etc”.

Exemplos:

tcpdmatch sshd 192.168.1.4

Testa se a tentativa de conexão ao daemon SSH “sshd” vinda do endereço

"192.168.1.4" será ou não aceita pelo sistema.

tcpdmatch -d ftpd 10.15.61.5

Verifica se a tentativa de conexão ao daemon FTP "ftpd" vinda do endereço "10.15.61.5" será ou não aceita pelo sistema, levando em consideração os arquivos "hosts.allow" e "hosts.deny" presentes no diretório atual.

8.2.10. traceroute

Mostra o caminho percorrido por um pacote de rede até chegar ao seu destino. Este comando, mostra na tela o caminho percorrido entre os roteadores de rede e o tempo gasto de retransmissão. É útil para detecção de problemas de rede.

traceroute [opções] [hostname/IP de destino]

Onde:

hostname/IP destino

É o endereço para onde o pacote será enviado, por exemplo, "www.debian.org". Caso o tamanho do pacote não seja especificado, o utilitário assumirá o tamanho padrão de pacote (40 bytes).

opções:

-l

Mostra o tempo de vida do pacote (ttl).

-m [num]

Ajusta a quantidade máximas de ttl dos pacotes. O padrão é 30.

-n

Mostra os endereços numericamente ao invés de usar resolução DNS.

-p [porta]

Ajusta a porta que será usada para o teste. A porta padrão é 33434.

-r

Pula as tabelas de roteamento e envia o pacote diretamente ao computador conectado a rede.

-s [endereço]

Usa o endereço IP/DNS definido pelo argumento "endereço", como origem para o envio de pacotes em computadores com múltiplos endereços IP ou nomes de máquina.

-v

Mostra mais detalhes sobre o resultado do traceroute.

-w [num]

Configura o tempo máximo de aguardo por uma resposta. O padrão é 3 segundos.

Exemplos:

traceroute www.debian.org

Mostra o caminho percorrido pelos pacotes de rede desde a máquina de origem (computador local) até a máquina de destino (www.debian.org).

traceroute -v -w 5 www.uol.com.br

Exibe mais informações sobre a rota seguida pelos pacotes enviados pela máquina de origem até a máquina de destino e configura o tempo limite de espera por resposta para 5 segundos.

8.2.11. wget

Realiza o *download* de forma não-interativa de arquivos da *Web*. É um utilitário muito utilizado em *scripts shell* para baixar arquivos de um servidor HTTP remoto. Os arquivos copiados via “wget” serão disponibilizados no diretório corrente em que usuário estava quando invocou o utilitário.

O wget trabalha em *background* (segundo plano) e tem a capacidade de executar o *download* de arquivos via HTTP/HTTPS (inclusive via servidores *Proxy*) e FTP, mesmo que o usuário que iniciou o *download* não esteja mais usando o sistema.

Para sua configuração, wget utiliza globalmente o arquivo “/etc/wgetrc”, e alternativamente, os usuários podem manter suas opções particulares no arquivo “.wgetrc” contido em seu diretório pessoal (caso ele não exista, pode ser a partir da cópia de “/etc/wgetrc”).

wget [opções] [url]

Onde:

url - É o endereço do arquivo que se deseja fazer o download. Deve ser informado no formato Web (ex.: “<http://www.servidor.com.br/arquivo.txt>”).

opções:

-a arquivo

Faz com que toda a saída de wget seja salva de forma incremental no arquivo de log determinado pelo argumento “arquivo”. Caso o arquivo não exista, ele será criado.

-b

Faz com que o utilitário seja executado em segundo plano.

-c

Continua o download de um arquivo parcialmente baixado para o computador. Para que esta opção possa ser utilizada, é necessário que o servidor FTP ou HTTP suporte a técnica de “Range Header”.

-d

Ativa a depuração do comando. Útil para solucionar problemas.

--delete-after

Exclui cada arquivo baixado no computador local logo após o término do download dos mesmos. Útil para fazer o carregamento do cache de objetos de um servidor Proxy, por exemplo.

--follow-ftp

Baixar arquivos presentes nos links FTP dos documentos HTML processados por wget. Por padrão este tipo de operação é desabilitada.

-i arquivo

Permite que o usuário especifique um arquivo contendo as URL's que devem ser baixadas por wget. Os endereços Web citados em conjunto com a invocação do comando, tem precedência sobre as URL's contidas no arquivo. O arquivo deve conter uma URL por linha.

-l nível

Define até em que nível de diretório uma cópia recursiva deve ser feita.

-m

Permite fazer o espelhamento de um site inteiro para o computador local.

-nd

Não recriar a estrutura de diretórios quando estiver fazendo uma cópia recursiva.

-o arquivo

Direciona as saídas de wget para o arquivo determinado no argumento homônimo. O arquivo não é usado de forma incremental como na opção “-a”.

--proxy-user=usuário

Permite que um usuário do serviço de Proxy seja informado, para ser usado para baixar os arquivos. Esta opção, depende do ajuste da variável de ambiente “http_proxy” ou do parâmetro homônimo nos arquivos de configuração do wget (“/etc/wgetrc” ou “~/.wgetrc”).

--proxy-passwd=senha

Especifica uma senha para o usuário definido através da opção “--proxy-user”.

-r

Ativa a cópia recursiva dos dados.

-w segundos

Especifica o tempo em segundos entre as tentativas que são feitas para baixar os arquivos. Você pode usar o sinalizador “m” para informar o tempo em minutos, “h” para horas ou “d” para dias.

--waitretry=segundos

Especifica o tempo em segundos em que wget tentará novamente baixar um arquivo após uma falha no download. O padrão é não esperar e prosseguir com o processamento da próxima URL.

Exemplos:

wget <http://ocs.eparana.parana/index.html>

Faz a cópia do arquivo “index.html” a partir do site “http://ocs.eparana.parana”.

wget -m <http://ocs.eparana.parana>

Faz a cópia recursiva de todo o conteúdo do site “<http://ocs.eparana.parana>”.

wget -proxy-user=marcius -proxy-passwd=velhinha <http://zig.com/linux.tar.gz>

Faz a cópia do arquivo “linux.tar.gz” a partir do site “<http://zig.com>” através do serviço de Proxy HTTP, especificando usuário e senha para acesso a Internet através deste serviço.

wget --waitretry=15 <ftp://10.15.22.205/tnsnames.ora>

Realiza o download do arquivo “tnsnames.ora” a partir do site FTP “10.15.22.205”, usando o tempo de 15 segundos entre as tentativas falhas de baixar o arquivo.

9. Arquivos de Configuração

Este tópico, tem o objetivo de fundamentar alguns dos principais conceitos de rede e serviços que utilizam este recurso. Além disso, comentaremos sobre os arquivos de configuração mais importantes do sistema.

9.1. Interfaces de Rede

Os arquivos especiais que representam os dispositivos de rede (interfaces de rede) no GNU/Linux, estão localizados sob o diretório `/dev`. A maioria destes arquivos, é criada dinamicamente pelo sistema de detecção de hardware do GNU/Debian no momento da inicialização do sistema. Este é o caso, por exemplo, das interfaces `eth`, quando o módulo do dispositivo em questão está disponível e pode ser carregado pelo *Kernel* durante o *boot*.

Em alguns casos, os dispositivos são criados dinamicamente em tempo de execução por programas que necessitam utilizar a interface. Por exemplo, uma interface `ppp` será criada no momento em que o usuário desejar se conectar a um provedor Internet via linha discada.

Abaixo, a identificação de algumas interfaces de rede utilizadas pelo GNU/Linux. O caractere `"?"`, significa um número que identifica as interfaces sequencialmente, iniciando pelo 0 (zero):

- `eth?` - Placa de rede Ethernet ou WaveLan.
- `ppp?` - Interface de rede PPP (protocolo ponto a ponto).
- `slip?` - Interface de rede serial.
- `eql` - Balanceador de tráfego para múltiplas linhas.
- `plip?` - Interface de porta paralela.
- `arc?e`, `arc?s` - Interfaces Arcnet.
- `sl?`, `ax?` - Interfaces de rede AX25 (respectivamente para *Kernels* 2.0.xx e 2.2.xx).
- `fddi?` - Interfaces de rede FDDI.
- `dli??`, `sdl?` - Interfaces Frame Relay, respectivamente para para dispositivos de encapsulamento DLCI e FRAD.
- `nr?` - Interface Net Rom.
- `rs?` - Interfaces Rose.
- `st?` - Interfaces Strip (Starmode Radio IP).
- `tr?` - Interfaces Token Ring.
- `wlan?` - Interfaces wirelles.

9.2. Interface de "loopback"

A interface *loopback*, é um tipo especial de interface que permite fazer conexões, com a própria máquina. Todos os computadores que usam o protocolo TCP/IP, utilizam esta interface, e existem várias razões para isto, por exemplo, você pode testar vários programas de rede sem interferir o funcionamento de outras máquinas de sua rede. Por convenção, o endereço IP `"127.0.0.1"`, foi escolhido especificamente para funcionar como *loopback*. Assim, toda conexão cujo alvo é o endereço `"127.0.0.1"`, será feita com o próprio computador local do usuário.

A configuração da interface *loopback* é simples e você deve ter certeza de que ela foi feita corretamente. Você pode conferir se a configuração está correta, utilizando o comando `"ifconfig"` sem nenhum parâmetro adicional e conferindo se a interface `"lo"` está configurada e apontando para o endereço `"127.0.0.1"`. Caso você precise configurar manualmente a interface de *loopback*, você poderá acrescentar as seguintes linhas a seu arquivo `"/etc/network/interfaces"`:

```
auto lo
iface lo inet loopback
```

Após adicionar as linhas acima ao arquivo, execute o seguinte comando:

```
ifup lo
```

Caso a interface *loopback* não esteja configurada de forma correta, você poderá ter

problemas quando tentar se conectar a qualquer serviço de rede que esteja sendo executado localmente.

9.3. Serviços e Portas TCP/IP

As aplicações fazem acesso à camada de transporte do protocolo de rede através de portas. As portas, podem ser vistas como canais de comunicação. Cada porta é referenciada como número inteiro, que a identifica, e a aplicação a qual ela dá suporte. Este número é um valor de 16 *bits*, que varia de 1 à 65535.

Cada serviço de rede oferecido por um computador, deve utilizar uma porta de comunicação exclusiva. As portas de número 1 à 1024, são conhecidas como “portas privilegiadas” porque os serviços oferecidos através delas, executam com autoridade de superusuário (*root*). Elas também são chamadas de “portas bem conhecidas”, porque são padronizadas para os mesmos serviços em vários sistemas.

O comando “*nmap*”, traz informações sobre as portas TCP/UDP de uma determinada máquina:

nmap -p1-65535 localhost

Realiza uma pesquisa completa em todas as portas da máquina e mostra as portas que estão abertas.

nmap 10.10.10.10

Realiza uma pesquisa nas portas privilegiadas da máquina 10.15.20.20 e mostra as que estão abertas.

9.4. Arquivos de Configuração

9.4.1. */etc/network/interfaces*

Este é o arquivo de configuração usado pelos programas “*ifup*” e “*ifdown*”, respectivamente para ativar e desativar as interfaces de rede. O que estes utilitários fazem na realidade é carregar os utilitários “*ifconfig*” e “*route*” através dos argumentos passados do arquivo “*/etc/network/interfaces*”, permitindo que o usuário iniciante configure uma interface de rede com mais facilidade.

Abaixo, um exemplo de configuração do arquivo *interfaces*:

Arquivo */etc/network/interfaces*

**iface eth0 inet static
address 192.168.1.50
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
gateway 192.168.1.1**

As interfaces e roteamentos são configurados na ordem que aparecem neste arquivo. Cada configuração de interface inicia com a palavra chave “*iface*”. A próxima palavra é o nome da interface que se deseja configurar (a mesma forma usada pelos comandos “*ifconfig*” e “*route*”). Você também pode usar “IP aliases” (técnica para adicionar mais de um endereço IP a mesma interface), mas tenha certeza que a interface real é inicializada antes que a virtual. Por exemplo, se você deseja usar a interface virtual “*eth0:0*”, você deverá garantir que “*eth0*”, a interface real, foi configurada antes.

A próxima palavra especifica a família de endereços da interface. Escolha a opção “*inet*” para as redes TCP/IP, “*ipx*” para o protocolo IPX e “*ipv6*” se deseja utilizar o protocolo IPV6.

A palavra “*static*”, especifica o método que a interface será configurada, em nosso exemplo (acima em **negrito**), uma interface com endereço estático (fixo). Outros métodos e seus parâmetros são especificados abaixo:

O método “***loopback***”

É usado para configurar a interface de *loopback* (*lo*) do protocolo IPv4.

O método “static”

É usado para configurar um endereço IPv4 fixo para a interface. As opções que podem ser usadas com este método, são as seguintes:

Parâmetros obrigatórios:**address** endereço

Endereço IP da Interface de rede (por exemplo, 192.168.1.50).

netmask máscara

Máscara de rede a ser utilizada pela interface (por exemplo, 255.255.255.0).

Parâmetros opcionais:**broadcast** endereço

Endereço de *broadcast* da rede (por exemplo, 192.168.1.255).

network endereço

Endereço da rede (por exemplo, 192.168.0.0).

gateway endereço

Endereço do roteador padrão (por exemplo, 192.168.1.1). O roteador é o equipamento de rede responsável por conectar o seu computador a outras redes.

O método “dhcp”

Este método é usado para obter automaticamente os parâmetros de configuração do protocolo de rede, através de um servidor DHCP da rede, usando as ferramentas: “dhclient”, “pump” ou “dpcpcp” (os dois últimos somente *kernel*s 2.0.x e 2.2.x). Abaixo, os parâmetros que podem ser utilizados:

hostname nome

Nome da estação de trabalho que será requisitado. (pump, dhcpd).

leasehours horas

Tempo de concessão da configuração a ser usada pelo computador local em horas (pump).

leasetime segundos

Tempo de concessão da configuração a ser usada pelo computador local em segundos (dhcpd).

vendor vendedor

Uma expressão (*string*) de identificação do vendedor (dhcpd).

client identificação

Uma expressão (*string*) de identificação do cliente (dhcpd).

O método “bootp”

Este método é usado para obter um endereço via *bootp*:

bootfile arquivo

Faz com que o servidor forneça o arquivo de inicialização especificado no argumento “arquivo” para as estações da rede que fizerem uma requisição *bootp* a ele.

server endereço

Especifica o endereço do servidor *bootp*.

hwaddr endereço

Ao invés de usar o endereço de hardware (MAC) real do servidor, utiliza o alternativo fornecido pelo argumento “endereço”.

Algumas opções se aplicam a todo tipo de interface:

noauto

Não configura automaticamente a interface quando o “ifup” ou “ifdown” são

executados com a opção “-a” (normalmente usada durante a inicialização ou desligamento do sistema).

pre-up comando

Executa um comando antes da inicialização da interface.

up comando

Executa um comando após a interface ser iniciada.

pre-down comando

Executa um comando antes de desativar a interface.

down comando

Executa um comando após desativar a interface.

Os comandos que são executados através das opções “up”, “pre-up”, “pre-down” e “down”, podem ser utilizados várias vezes na mesma interface, mas eles são executados na sequência em que são usados. Outra informação importante, é que se um dos comandos falhar, nenhum dos restantes é executado. Para forçar a execução dos demais, adicione o artifício “|| true”, ao final da linha de comando em questão.

9.4.2. /etc/hostname

Arquivo lido pelo utilitário “hostname” para definir o nome de sua estação de trabalho.

9.4.3. /etc/hosts

O arquivo “/etc/hosts”, faz a associação entre um nome de computador e um endereço IP. Basicamente tem a mesma função do serviço DNS, porém, atua somente localmente. Atualmente, este arquivo é utilizado para configurações de resolução de nomes muito específicas e de escopo local (ex.: máquinas virtuais), e ainda, para aqueles serviços e programas que não suportam a resolução de nomes via serviço DNS (ex.: algumas versões do Lotus Notes).

A inclusão de um computador neste arquivo dispensa a consulta a um servidor de nomes para obter um endereço IP, sendo muito útil para máquinas que são acessadas frequentemente e que não tem seu endereço IP constantemente modificado. A desvantagem de usar este recurso, é que você mesmo precisará manter este arquivo atualizado. Em um ambiente de rede bem planejado e que não possui particularidades com relação a resolução de nomes, o único endereço que aparecerá neste arquivo, referenciará o nome do computador local, como no exemplo a seguir:

Exemplo de arquivo /etc/hosts

127.0.0.1 localhost ecelepar00001.celepar.parana

Você pode especificar mais que um nome de computador por linha como demonstrado no exemplo acima.

OBS: Caso encontre problemas de lentidão para resolver nomes e até para executar alguns aplicativos (como o gdm, por exemplo), verifique se existem erros neste arquivo de configuração.

9.4.4. /etc/host.conf

O arquivo “/etc/host.conf” é o local onde é possível configurar alguns itens que alteram a forma como a resolução de nomes é feita no sistema. De modo geral, o exemplo abaixo serve para a maioria dos ambientes:

#Exemplo do arquivo “/etc/host.conf”

order hosts,bind

multi on

De acordo com nosso exemplo, a ordem de resolução de nomes (parâmetro “order hosts,bind”) será feita primeiro através da consulta do arquivo “/etc/hosts”, e em seguida, uma pesquisa no serviço DNS. O parâmetro “multi on”, faz com que todos os endereços IP correspondentes ao nome pesquisado sejam retornados ao programa que emitiu a solicitação. Há ainda outros parâmetros que podem ser configurados para as finalidades mais variadas. Para maiores detalhes sobre estas configurações, consulte o manual “**man host.conf**”.

9.4.5. /etc/networks

O arquivo “/etc/networks” tem uma função similar ao arquivo “/etc/hosts”. Ele contém um banco de dados simples de nomes de redes com seus respectivos endereços. O formato deste arquivo é:

```
Nome_da_Rede    Endereço_da_Rede
```

Abaixo, um exemplo de configuração deste arquivo:

#Exemplo de arquivo “/etc/networks”

```
loopnet 127.0.0.0
celepar 10.15.20.0
nasa    200.192.45.0
```

A configuração de redes nomeadas feita através do “/etc/networks”, é particularmente interessante, quando utilizamos comandos como o “**route**”. Isto porque, este comando poderá exibir o nome das redes em vez dos endereços correspondentes, o que torna a saída do comando muito mais amigável.

9.4.6. /etc/resolv.conf

O “/etc/resolv.conf” é o arquivo de configuração principal do mecanismo de resolução de nomes do GNU/Linux. É um arquivo texto simples, com um parâmetro por linha, e que contém o endereço de servidores DNS que serão usados pelo sistema. Existem três palavras chaves normalmente usadas na configuração deste arquivo:

domain

Especifica o domínio DNS ao qual a máquina pertence.

search

Especifica uma lista de nomes de domínio alternativos ao procurar por um computador, separados por espaços. A linha do parâmetro “search”, pode conter no máximo 6 domínios ou 256 caracteres.

nameserver

Especifica o endereço IP de um servidor de nomes de domínio para resolução de nomes (DNS). Pode-se especificar múltiplos servidores com esta opção.

Como exemplo, o /etc/resolv.conf se parece com isto:

```
domain celepar.parana
search celepar.parana pr.gov.br
nameserver 10.15.61.248
nameserver 10.15.61.246
```

Neste exemplo, o nome de domínio que irá complementar o nome da máquina é o “celepar.parana”. Quando uma pesquisa for realizada para localizar uma máquina por seu nome, primeiramente o sufixo “celepar.parana” será acrescentado ao nome a ser pesquisado, caso a máquina não seja encontrada, o sufixo “pr.gov.br” será acrescentado, e a pesquisa será reenviada aos servidores DNS especificados com a opção “nameserver”, na tentativa de resolver o nome e obter o endereço IP da máquina em questão.

9.4.7. /etc/nsswitch.conf

É a evolução do “/etc/hosts.conf”. Possibilita que a ordem de procura para resolução de nomes seja customizada para cada serviço. Ele é utilizado nas últimas distribuições por diversas bibliotecas em vez de “/etc/hosts.conf”.

9.4.8. /etc/services

O arquivo “/etc/services” é um banco de dados simples que associa um nome a uma porta de comunicação utilizada pelos protocolos TCP/UDP. É um arquivo texto de formato muito simples, cada linha representa um serviço. Cada item, é dividido em três campos separados por qualquer número de espaços em branco (tab ou espaços). Os campos são:

nome porta/protocolo apelido # comentário

nome

Uma palavra simples que representa o nome do serviço sendo descrito.

porta/protocolo

Este campo é dividido em dois sub-campos, veja detalhes abaixo:

porta

Um número que define o número da porta em que o serviço estará disponível. Muitos dos serviços comuns tem designados um número de serviço. Estes estão descritos no RFC-1340.

protocolo

Este sub-campo, pode ser ajustado para TCP ou UDP. É importante notar que o item “18/tcp” é diferente do item “18/udp”, e que não existe razão técnica para o mesmo serviço usar estas duas portas distintas. Normalmente, se um serviço está disponível em ambos os protocolos TCP e UDP, você precisará especificar ambos.

apelidos

Outros nomes podem ser usados para se referir a entrada deste serviço.

comentário

Qualquer texto aparecendo em uma linha após um caractere “#”, é ignorado e tratado como comentário.

9.4.9. /etc/protocols

O arquivo “/etc/protocols” mapeia números de identificação de protocolos em nomes de protocolos. Isto é usado por programadores para permiti-los especificar protocolos por nomes em seus programas, e também, por alguns programas tal como “tcpdump”, possibilitando estes a mostrar nomes em vez de números em sua saída. A sintaxe geral deste arquivo é:

nome_protocolo número_identificador apelidos #Comentários

9.4.10. /etc/alternatives/

É um diretório que contém *links* para diversos aplicativos padrões utilizados pelo sistema. Dentre eles são encontrados *links* para os editores do sistema, programas clientes de terminal padrão, gerenciadores gráficos, entre outros.

Por exemplo, se você desejar usar o editor “jed” ao invés do “vi”, remova o *link* “editor” com o comando “rm editor”, localize o arquivo executável do jed com o comando “which jed” e crie um *link* para ele neste diretório usando “ln -s /usr/bin/jed editor”. De

agora em diante o editor padrão usado pela maioria dos aplicativos será o jed.

9.4.11. `/etc/default/rcS`

Contém variáveis padrões que alteram o comportamento de inicialização dos *scripts* contidos em `/etc/rcS.d/`.

Por exemplo, se quiser menos mensagens na inicialização do sistema, ajuste o valor da variável `VERBOSE` para o valor `no`.

Para detalhes sobre as variáveis que podem ser configuradas neste arquivo, consulte a documentação oficial do rcS com o comando `man rcS 5`.

Recomendamos fortemente que você estude primeiro qual é função de cada variável antes de modificá-las, pois isto pode evitar transtornos graves no funcionamento do sistema.

9.4.12. `/etc/pam.d/`

Este diretório possui arquivos de configuração de diversos módulos PAM existentes em seu sistema. Os módulos PAM, desempenham os serviços de autenticação do sistema. Há vários arquivos neste diretório, cada um deles corresponde a um serviço que depende do esquema de autenticação dos módulos PAM.

9.4.13. `/etc/security/`

Este diretório contém arquivos para controle de segurança e limitadores de uso de recursos que serão aplicados aos usuários do sistema. O funcionamento de muitos dos arquivos deste diretório depende de modificações nos arquivos do diretório `/etc/pam.d/` para habilitar as funções de controle, acesso e restrições.

9.4.14. `/etc/security/access.conf`

É lido no momento do *login* do usuário e permite definir quem terá acesso ao sistema e a partir de onde (terminais, endereços de rede, etc). O formato deste arquivo são 3 campos separados pelo caractere `:",` cada linha contendo uma regra de acesso.

O primeiro campo deve conter o caractere `+` ou `-`, para definir se aquela regra permitirá (`+`) ou bloqueará (`-`) o acesso do usuário.

O segundo campo deve conter uma lista de nomes de usuários (inclusive no formato `usuário@computador`), grupos, a palavra `ALL` (todos usuários/grupos) ou `EXCEPT` (com exceção para usuários/grupos).

O terceiro campo deve conter uma lista de terminais `"tty"` (para acessos locais), nomes de computadores, nomes de domínios (iniciando com `."`), endereço IP de computadores ou endereço IP de redes (finalizando com `."`) de origem. Também podem ser usadas as palavras `ALL` (qualquer máquina), `LOCAL` (máquinas da rede local, cujo nome não possui o caractere `."`) e `EXCEPT` (com exceção para máquina).

Abaixo um exemplo do arquivo `"access.conf"`:

```
# Somente o root poderá usar tty1
```

```
#
```

```
:-ALL EXCEPT root:tty1
```

```
# bloqueia o acesso ao console a todos usuários exceto marcius e joselito.
```

```
#
```

```
:-ALL EXCEPT marcius joselito:console
```

```
# Bloqueia acessos remotos de contas privilegiadas (grupo gga-suporte), com  
#exceção das tentativas de conexões vindas de máquinas da rede local ou que  
#fazem parte do domínio "celepar.parana".
```

```
#
```

```
:-gga-suporte:ALL EXCEPT LOCAL .celepar.parana
```

```
# Algumas contas, não tem permissão de acessar o sistema de nenhum lugar:
#
-:playmobil ze karioka:ALL
```

Todas as outras contas que não se encaixam nas regras acima, podem acessar o # sistema de qualquer lugar.

9.4.15. /etc/security/limits.conf

Define limites de uso dos recursos do sistema para cada usuário ou grupos de usuários. Os recursos são descritos em linhas da seguinte forma:

```
#<domínio>      <tipo> <item> <valor>
```

O campo “domínio”, pode ser um nome de usuário, um grupo (especificado sob a forma “@grupo”) ou o caractere coringa “*” que significa “todos usuários e grupos”.

O campo “tipo”, pode assumir o valor “soft” para os limites mínimos, “hard” para os limites máximos ou “-” quando nenhum destes dois tipos é aplicável (como na utilização do item “maxlogins”, por exemplo).

O campo “item”, pode assumir os seguintes valores:

core - limita o tamanho do arquivo “core” em *Kilobytes*;

data - tamanho máximo de dados em *Kilobytes*;

fsize - Tamanho máximo de arquivos que podem ser criados/modificados em *Kilobytes*;

memlock - Espaço máximo de endereços bloqueados na memória em *Kilobytes*;

nofile - Número máximo de arquivos abertos;

rss - Tamanho máximo dos programas residentes na memória virtual em *Kilobytes*;

stack - Tamanho máximo de pilha em *Kilobytes*;

cpu - Tempo máximo usado na CPU em minutos;

nproc - Número máximo de processos;

as - Limite de espaço de endereços;

maxlogins - Número máximo de acessos realizados por este usuário;

priority - Prioridade que os programas deste usuário serão executados.

Abaixo, exemplificamos o arquivo “/etc/security/limits.conf”, para melhor entendimento:

```
#<domínio>      <tipo> <item>      <valor>
*               soft   core        0
*               hard   rss         10000
@estagiarios    hard   nproc       20
@usuarios       hard   fsize       4096
@estagiarios    hard   nofile      50
ftp             hard   nproc       0
@estagiarios    -      maxlogins   10
```

9.4.16. /etc/fstab

Contém detalhes para a montagem dos sistemas de arquivos do sistema. É um arquivo de configuração de extrema importância no sistema. Este arquivo é utilizado pelos comandos “mount” e “swapon”, para disponibilizar os dados e a área de paginação do sistema.

Abaixo, citamos um exemplo do arquivo “/etc/fstab”:

/etc/fstab: static file system information.

```
#
# <sistema>      <ponto montagem> <tipo> <opções>      <dump> <ordem>
proc            /proc           proc   defaults      0      0
/dev/hda3       /               xfs    defaults      0      1
/dev/hda2       /boot           ext3   defaults      0      2
```

/dev/hda4	/home	xfs	defaults	0	2
/dev/hda1	none	swap	sw	0	0
/dev/hdb	/media/cdrom0	udf,iso9660	user,noauto	0	0
/dev/fd0	/media/floppy0	auto	rw,user,noauto	0	0

No exemplo acima, podemos verificar que o arquivo “/etc/fstab”, é composto pelas seguintes colunas:

Sistema

Corresponde ao sistema de arquivos que se deseja montar. Esta coluna conterá partições de disco, dispositivos como CD-ROM, *drive* de disquetes, etc, e uma área especial da memória para abrigar o sistema de arquivos do *Kernel* o “/proc”.

Ponto de montagem

É o diretório sob o qual, o sistema de arquivos definido no parâmetro anterior, será disponibilizado para utilização tanto do sistema bem como dos usuários.

Tipo

É o tipo do sistema de arquivos usado pelos dispositivos para armazenar os dados. Sempre deve-se utilizar o tipo de sistema de arquivos correto para o dispositivo a ser montado, caso contrário, o sistema de arquivos não será montado. Exemplos de tipos de sistemas de arquivo são: ext2, ext3, reiserfs, fat, ntfs, udf, iso9660, xfs, swap, entre outros.

Opções

É a coluna onde podemos definir as opções de montagem do sistema de arquivos. As opções variam de acordo com o tipo de sistema de arquivos a ser montado, mas existem também, opções globais que podem ser usadas em qualquer tipo de sistema de arquivos (ex.: noauto, user, owner, comment, etc). As opções globais mais utilizadas são:

noauto - Para impedir que o sistema de arquivos seja montado automaticamente, durante o processo de inicialização ou através do comando “mount -a”.

user – Para possibilitar que usuários comuns possam montar os sistemas de arquivos marcados com esta opção. Lembre-se que somente o usuário “root” tem permissão para montar sistemas de arquivos.

Dump

Esta é uma opção utilizada pelo utilitário “dump”, para determinar quais os sistemas de arquivos devem ser processados por este utilitário. O valor “0” (zero), significa que aquele sistema de arquivos não necessita do dump. Caso você tenha interesse sobre o funcionamento do dump, acesse sua página do manual através do comando “man dump”.

Ordem

Especifica a ordem de checagem feita durante a inicialização do sistema pelo utilitário “**fsck**” nos sistemas de arquivos. O sistema de arquivos raiz (“/”) sempre deverá ter como ordem de checagem o valor “1” por sua importância, os demais sistemas de arquivos devem possuir um valor maior ou igual a “2”. O valor “0” (zero) significa que o sistema de arquivos não deverá ser checado.

9.4.17. /etc/hosts.allow

É um dos arquivos de controle de acesso a serviços de rede do GNU/Linux, assim como “/etc/hosts.deny”. As regras contidas neste arquivo são utilizadas pelo *daemon* “tcpd”, mais conhecido como “TCP Wrapper”, para fazer a liberação do acesso ao sistema orientada a serviços.

O arquivo “/etc/hosts.allow”, tem a função de permitir o acesso de um determinado endereço IP ou nome de máquina, aos *daemons* que controlam os serviços que são executados no computador local. Por exemplo, você poderá cadastrar a máquina “10.15.15.15”, no arquivo “/etc/hosts.allow”, para que ela possa acessar o *daemon*

"sshd".

Para testar as regras contidas neste arquivo, você poderá usar o utilitário "**tcpdmatch**".

O formato deste arquivo de configuração, é demonstrado a seguir:
daemons : máquinas [:comando]

Onde:
daemons

São os serviços que você deseja liberar o acesso a uma determinada máquina ou a um conjunto delas. Exemplos de *daemons* são: sshd, ftpd, indented, etc.

máquinas

É o endereço IP, nome da máquina, ou em alguns casos, o domínio (iniciado pelo caractere "."), que terá acesso aos *daemons* especificados pelo parâmetro explicado anteriormente.

comandos

É um parâmetro opcional que permite especificar um comando que será executado pelo *shell* "/bin/sh", quando uma conexão recebida corresponder a uma determinada regra existente. Particularmente interessante quando o computador está sofrendo um ataque.

Você poderá utilizar caracteres coringas como "*" e "?" para especificar nomes e endereços de máquinas, além das palavras chave "ALL", "LOCAL", "UNKNOWN", "KNOWN", "PARANOID" e "EXCEPT". As palavras chave "ALL" e "EXCEPT", também podem ser usadas no parâmetro "**daemons**". Para entender o significado de cada uma das palavras chave, consulte a documentação através do comando "man hosts.allow".

Exemplos:

ALL: ALL

Libera o acesso a todos os daemons do sistema a qualquer máquina/domínio.

sshd:10.15.15.15 ecelepar00001

Libera o acesso ao daemon SSH "sshd" para as máquinas "10.15.15.15" e "ecelepar00001".

sshd ftpd: 10.* EXCEPT 10.15.15.15

É necessário muito cuidado na observação desta regra. Ela libera o acesso aos daemons SSH "sshd" e FTP "ftpd" para todas as máquinas da rede "10.0.0.0/8", com exceção para "10.15.15.15", que ainda assim, pode ter seu acesso garantido, caso não exista uma regra explícita em "/etc/hosts.deny", impedindo seu acesso ao sistema, já que por padrão, o acesso aos serviços é liberado.

9.4.18. /etc/hosts.deny

É um dos arquivos de controle de acesso a serviços de rede do GNU/Linux, assim como "/etc/hosts.allow". Todas as regras contidas neste arquivo, tem o objetivo de proibir o acesso de uma ou mais máquinas a um determinado *daemon* da máquina local, função contrária daquela desempenhada por "/etc/hosts.allow".

O formato como o arquivo deve ser configurado e os parâmetros que podem ser usados na sua configuração, são exatamente os mesmos de "/etc/hosts.allow" (veja o tópico daquele arquivo para mais detalhes).

É importante frisar aos leitores deste documento que, por padrão, o acesso aos *daemons* da máquina local via rede são liberados, embora isso possa não ser possível por configurações particulares que o usuário fez em cada um deles. Portanto, caso se queira negar o acesso a determinado *daemon*, é necessário configurar apropriadamente este arquivo.

Exemplos:

ALL: ALL

Proíbe o acesso a qualquer daemon da maquina local a qualquer computador.

sshd : 10.* EXCEPT 10.15.15.15

Proíbe o acesso ao daemon SSH "sshd" a qualquer maquina da rede "10.0.0.0/8" com exceção da maquina "10.15.15.15".

mysqld : * EXCEPT 10.*

Proíbe o acesso ao daemon do MySQL "mysqld" a todas as maquinas, com exceção as maquinas da rede "10.0.0.0/8".

9.4.19. /etc/inetd.conf

O arquivo `/etc/inetd.conf` é o arquivo de configuração do *daemon* "inetd" (*Internet Daemon*), também conhecido como "super servidor".

Para saber qual é função do daemon "inetd", é necessário entender o processo de comunicação numa rede TCP/IP, mesmo que de modo muito simplificado. Assim, para que uma maquina possa disponibilizar qualquer serviço nesse tipo de rede, é necessário que um *daemon* permaneça continuamente escutando a rede, através de uma porta de comunicação, para que possa receber as solicitações destinadas à aquele serviço. Toda troca de informação entre duas maquinas, ocorre pela utilização de um endereço de rede e uma porta. Então, quanto mais serviços uma maquina tem que disponibilizar, maior é o número de portas que ela deverá usar, e maior também, será seu consumo de recursos (CPU, memória, etc), já que necessita manter estes *daemons* executando permanentemente no sistema.

A idéia trazida pelo inetd, é otimizar e simplificar todo este processo. Para isso, os serviços a serem disponibilizados pela maquina, ao invés de utilizarem *daemons* específicos, utilizaram apenas o inetd, que redirecionará as requisições de forma apropriada somente para o serviço correspondente, racionalizando o consumo de recursos da maquina e evitando que várias portas fiquem abertas, o que pode gerar problemas de segurança. Para usufruir de toda esta facilidade, é necessário simplesmente o ajuste adequado do arquivo de configuração do inetd, o `/etc/inetd.conf`.

O formato deste arquivo, é exposto abaixo:

<serviço> <tipo_soquete> <protocolo> <flags> <usuário> <daemon> <parâmetros>

A seguir, uma linha de exemplo de configuração do arquivo:

```
#<serviço> <tipo_soquete> <protocolo> <flags> <usuário> <daemon> <parâmetros>
netbios-ssn stream tcp nowait root /usr/sbin/tcpd /usr/sbin/smbd
```

Mesmo que você não tenha a necessidade de disponibilizar nenhum serviço na rede, é importante editar este arquivo para desabilitar os serviços providos via inetd, comentando (com o caractere "#") o inicio das linhas do arquivo que correspondem a configuração de um serviço. É importante entender também, que serviços com elevados números de acessos, devem ser configurados para trabalhar com *daemons* próprios e não com o inetd, devido ao ganho de desempenho que será obtido nesse modo de operação.

O arquivo `/etc/inetd.conf`, possui muitas opções de configuração, caso você necessite saber mais elas, consulte a documentação com o comando `"man inetd.conf"`.

9.4.20. /etc/inittab

Este é o arquivo de configuração utilizado pelo programa "init" para a inicialização do sistema, para maiores sobre este assunto, consulte a seção "Sistema de Boot e Runlevels" presente neste documento.

O formato deste arquivo, é mostrado na linha abaixo:

identificação:níveis_execução:ação:processo

Agora, vamos mostrar um exemplo funcional do arquivo “/etc/inittab”, e comentarmos sobre suas características principais. Os comentários iniciam-se com o caractere “#” e são ignoradas pelo programa init:

```
## Indica o nível de execução padrão do sistema
```

```
id:3:initdefault:
```

```
# System initialization.
```

```
## Indica o diretório de scripts a ser executado,
```

```
## dependendo do nível de execução configurado.
```

```
si::sysinit:/etc/rc.d/rc.sysinit
```

```
l0:0:wait:/etc/rc.d/rc 0
```

```
l1:1:wait:/etc/rc.d/rc 1
```

```
l2:2:wait:/etc/rc.d/rc 2
```

```
l3:3:wait:/etc/rc.d/rc 3
```

```
l4:4:wait:/etc/rc.d/rc 4
```

```
l5:5:wait:/etc/rc.d/rc 5
```

```
l6:6:wait:/etc/rc.d/rc 6
```

```
# Trap CTRL-ALT-DELETE
```

```
## A seqüência de teclas indica o comando descrito,
```

```
## ou seja, reinicializa o sistema.
```

```
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

```
## Se por algum motivo a energia cair, o sistema
```

```
## agenda o desligamento do sistema para 2 minutos
```

```
## a partir do momento da queda de energia.
```

```
pf::powerfail:/sbin/shutdown -f -h +2 "Falta de energia: Sistema desligando"
```

```
## Se a força for restaurada antes de ser executado
```

```
## o comando anterior a este será cancelado.
```

```
pr:12345:powerokwait:/sbin/shutdown -c "Cancelando o desligamento"
```

```
# Run gettys in standard runlevels
```

```
## Executa terminais em níveis de execução padrão,
```

```
## utilizados para conexões da console.
```

```
1:2345:respawn:/sbin/mingetty tty1
```

```
2:2345:respawn:/sbin/mingetty tty2
```

```
3:2345:respawn:/sbin/mingetty tty3
```

```
4:2345:respawn:/sbin/mingetty tty4
```

```
5:2345:respawn:/sbin/mingetty tty5
```

```
6:2345:respawn:/sbin/mingetty tty6
```

9.4.21. /etc/mtab

Contém uma lista dos sistemas de arquivos montados atualmente no sistema. Seu conteúdo é idêntico ao do arquivo virtual “/proc/mounts”. O comando “mount” executado sem parâmetros, exibe o conteúdo de “/etc/mtab”.

9.4.22. /etc/issue

Contém um texto ou mensagem que será mostrada antes do *login* do sistema.

9.4.23. /etc/issue.net

Mesma utilidade do arquivo “/etc/issue”, mas é mostrado antes do *login* numa sessão “telnet”. Outra diferença, é que este arquivo aceita os seguintes tipos de variáveis:

%t - Mostra o terminal (tty) atual.
%h - Mostra o nome de domínio qualificado (FQDN).
%D - Mostra o nome do domínio NIS.
%d - Mostra a data e hora atual.
%s - Mostra o nome do Sistema Operacional.
%m - Mostra o tipo de hardware do computador.
%r - Mostra a revisão do Sistema Operacional.
%v - Mostra a versão do Sistema Operacional.
%% - Mostra um simples sinal de porcentagem (%).

9.4.24. /etc/motd

Exibe um texto ou mensagem após usuário se conectar com sucesso ao sistema. Também é usado pelo telnet, ftp, e outros servidores que requerem autenticação do usuário (nome e senha).

10. Firewall iptables – Conceitos Básicos

A função principal de um *firewall*, é regular o tráfego de dados entre redes distintas e impedir a transmissão e/ou recepção de dados nocivos ou não autorizados de uma rede para outra. Este conceito inclui a aplicação de técnicas como a de filtragem de pacotes e de *proxy* de aplicações (análise de conteúdo dos pacotes), nas redes que empregam os protocolos TCP/IP.

A implementação e o gerenciamento de *firewalls*, se tornou uma das competências do administrador contemporâneo, isto porque, é cada vez maior a necessidade manter a informação segura. E é por essa razão, que decidimos incluir esta seção neste documento.

Este capítulo tem como objetivo fazer uma rápida introdução sobre os conceitos e o funcionamento do *firewall* “iptables” que acompanha a distribuição Debian e é uma das ferramentas para filtragem de pacotes mais utilizadas atualmente.

10.1. Introdução

O iptables foi introduzido a partir versão 2.4 do *Kernel* do GNU/Linux, com o objetivo de substituir o “ipchains” que era utilizado pelas versões 2.2. Este novo *firewall*, tem como vantagem ser muito estável (assim como o ipchains e ipfwadm), confiável, permitir muita flexibilidade na programação de suas regras, mais opções disponíveis para o controle de tráfego e melhor organização devido a organização aprimorada das etapas de roteamento.

O iptables é um *firewall* em nível de pacotes (veja mais detalhes sobre este conceito na seção “Tipos de Firewall”) e funciona baseado em endereços/portas de origem/destino. Ele desempenha suas funções através da comparação de regras, organizadas e armazenadas em tabelas internas, para saber se um pacote tem ou não permissão para adentrar a rede/máquina que está sendo protegida. Em configurações mais restritivas, o pacote é bloqueado e registrado para que o administrador do sistema tenha condições de avaliá-lo posteriormente.

O iptables também pode ser usado para modificar e monitorar o tráfego da rede, fazer NAT (“IP *masquerading*” e “*source/destination* NAT”), redirecionamento e marcação de pacotes, modificar a prioridade de pacotes que entram e saem do seu sistema, contagem de bytes, dividir tráfego entre máquinas e criar proteções contra várias técnicas de ataque (*anti-spoofing*, syn flood, DoS, etc).

As possibilidades oferecidas pelos recursos de filtragem iptables e a sua eficácia, dependem em grande parte dos conhecimentos do administrador do sistema em relação aos conceitos de funcionamento das redes TCP/IP e da manipulação precisa das regras que são utilizadas pela ferramenta para fazer a validação de cada pacote que trafega pela rede/máquina protegida por este *firewall*. É necessário portanto, que o administrador seja consciente e tenha claro em sua mente o que deseja quais serviços serão protegidos, quais são os endereços que serão aceitos/bloqueados, quais portas terão redirecionamento, etc.

10.2. Tipos de Firewall

De forma simplificada, existem dois tipos de *firewall* a saber:

- Nível de aplicação - Este tipo de *firewall*, analisa o conteúdo do pacote para tomar suas decisões de filtragem. *Firewalls* deste tipo, são mais intrusivos (pois analisam o conteúdo de tudo que passa por ele) e permitem um controle relacionado com o conteúdo do tráfego. Alguns *firewalls* em nível de aplicação combinam recursos básicos existentes em *firewalls* em nível de pacotes combinando as funcionalidade de controle de tráfego/controla de acesso em uma só ferramenta.
- Nível de pacotes - Este tipo de *firewall* toma as decisões baseadas nos parâmetros do pacote, como porta/endereço de origem/destino, estado da conexão, e outros parâmetros do pacote. O *firewall* então pode barrar o pacote (DROP) ou aceitá-lo (ACCEPT). O iptables é um excelente *firewall* que se encaixa nesta categoria.

10.3. Planejando a Implementação

Antes de iniciar a construção de um *firewall*, é necessário levar em consideração os seguintes aspectos:

- Quais serviços precisam ser protegidos. Serviços que devem ter acesso garantido a usuários externos e quais serão bloqueados a todas ou a determinadas máquinas. É recomendável implantar uma política austera de acesso a todas portas menores que a 1024 (portas privilegiadas) por disponibilizarem serviços muito importantes.
- Que tipo de conexões serão bloqueadas/permitidas. Serviços com autenticação em texto plano e potencialmente inseguros como rlogin, telnet, FTP, NFS, DNS, LDAP, SMTP RCP, X-Window, etc. são serviços que devem ter acesso garantido somente para máquinas/redes que você confia.

Outros fatores que devem ser levados em consideração são:

- Que máquinas terão acesso livre e quais serão restritas;
- Que serviços deverão ter prioridade no processamento;
- O volume de tráfego que o servidor manipulará. Através disso você pode balancear o tráfego entre outras máquinas, configurar proteções contra ataques do tipo DoS, syn flood, etc;
- Que pacotes terão permissão para trafegar de uma rede para outra.

A análise destes pontos pode determinar a complexidade do *firewall*, custos de implementação, prazo de desenvolvimento e tempo de maturidade para implementação. Existem muitos outros pontos que podem influenciar a implementação de um sistema de segurança, procuramos elencar aqui, o que consideramos ser os mais comuns para que você tenha uma base para seu projeto.

10.4. Conhecendo os conceitos do iptables

Neste tópico, explicaremos a estrutura de funcionamento no qual o iptables é baseado.

10.4.1. O que são regras?

As regras são como comandos passados ao iptables para que ele realize uma determinada ação (como bloquear ou permitir a passagem de um pacote), de acordo com o endereço/porta de origem/destino, interface de origem/destino, etc. As regras são armazenadas dentro dos chamados “*chains*” e processadas na ordem que são inseridas.

As regras são armazenadas em memória, o que significa que quando o computador for reiniciado todas serão perdidas. Por este motivo, elas deverão ser gravadas em um arquivo para serem aplicadas a cada inicialização do sistema.

Um exemplo de regra:

```
iptables -A INPUT -s 123.123.123.1 -j DROP
```

10.4.2. O que são chains?

Os “*chains*”, são locais onde as regras do *firewall* definidas pelo usuário são armazenadas para operação do *firewall*. Existem dois tipos de *chains*: os embutidos (como os *chains* INPUT, OUTPUT e FORWARD) e os criados pelo usuário. Os nomes dos *chains* embutidos devem ser especificados sempre em letras maiúsculas (note que os nomes dos *chains* devem ser escritos respeitando maiúsculas e minúsculas).

10.4.3. O que são tabelas?

Tabelas são os locais usados para armazenar os *chains* e um conjunto de regras com uma determinada característica em comum. As tabelas podem ser referenciadas com a opção “-t nome” do iptables. Por padrão, existem 3 tabelas disponíveis no iptables:

filter

Esta é a tabela padrão, contém 3 *chains* padrões:

INPUT - Consultado para os pacotes que chegam a máquina

OUTPUT - Consultado para os pacotes que saem da máquina
 FORWARD - Consultado para os pacotes que são redirecionados para outra interface de rede ou outra máquina.

Os *chains* INPUT e OUTPUT são verificados somente para conexões que se originam na maquina local.

nat

Usada para pacotes que necessitam criar outra conexão (IP *masquerading*, *source/destination* NAT, *Port Forwarding* e *Proxy* transparente são alguns exemplos). Possui 3 *chains* padrões:

PREROUTING - Consultado quando os pacotes precisam ser modificados logo que chegam. É o *chain* ideal para realização de DNAT e redirecionamento de portas.

OUTPUT - Consultado quando os pacotes gerados localmente precisam ser modificados antes de serem roteados. Este *chain* somente é consultado para conexões que se originam de endereços IP de interfaces locais.

POSTROUTING - Consultado quando os pacotes precisam ser modificados após o tratamento de roteamento. É o *chain* ideal para realização de SNAT e IP *Masquerading*.

mangle

Utilizada para alterações especiais de pacotes (como modificar o “Tipo de Serviço” (TOS) ou outros detalhes). Possui 5 *chains* padrões:

INPUT - Consultado quando os pacotes precisam ser modificados antes de serem enviados para o *chain* INPUT da tabela “filter”.

FORWARD - Consultado quando os pacotes precisam ser modificados antes de serem enviados para o *chain* FORWARD da tabela “filter”.

PREROUTING - Consultado quando os pacotes precisam ser modificados antes de ser enviados para o *chain* PREROUTING da tabela “nat”.

POSTROUTING - Consultado quando os pacotes precisam ser modificados antes de serem enviados para o *chain* POSTROUTING da tabela “nat”.

OUTPUT - Consultado quando os pacotes precisam ser modificados antes de serem enviados para o *chain* OUTPUT da tabela “nat”.

10.5. Manipulando Chains

Neste tópico, vamos aprender a gerenciar o iptables. Para isso, é necessário compreender como efetuar algumas operações básicas. Os procedimentos de gerenciamento envolvem a criação, modificação, exclusão e visualização do conjunto de regras que são organizadas nos *chains*, e esses por sua vez, em tabelas.

10.5.1. Adicionando regras

Como exemplo, vamos criar uma regra que bloqueia o acesso a nossa própria máquina (127.0.0.1). Primeiro, verificaremos com um “ping” a resposta dada pela máquina:

```
#ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=0.6 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=0.5 ms

--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.5/0.5/0.6 ms
```

Da forma prevista, a máquina deve ter respondido a todas as requisições. Agora, vamos incluir (opção “-A”) uma regra, no *chain* INPUT (-A INPUT), da tabela de filtragem (-t filter), que bloqueie (-j DROP) qualquer pacote cujo destino (opção “-d”) seja o endereço “127.0.0.1” (-d 127.0.0.1):

```
iptables -t filter -A INPUT -d 127.0.0.1 -j DROP
```


Agora verificamos novamente com o comando “ping” as respostas dadas pela máquina local:

```
#ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
```

```
--- 127.0.0.1 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
```

Desta vez a máquina não respondeu, pois todos os pacotes com o destino “127.0.0.1” são rejeitados. A opção “-A” é usada para adicionar novas regras ao final do conjunto de regras de cada *chain*. Além de “-j DROP” que serve para rejeitar os pacotes, podemos também usar “-j ACCEPT” para aceitar pacotes. A opção “-j” é chamada de “alvo da regra” ou somente “alvo”, pois define o destino do pacote que atravessa a regra.

OBS1: - O acesso a interface *loopback* (127.0.0.1) não deve ser de forma alguma bloqueado permanentemente, pois muitos aplicativos utilizam soquetes TCP para realizarem conexões localmente, caso do gerenciador de janelas X-Window System.

OBS2: - A tabela “filter” será usada por padrão caso nenhuma tabela seja especificada através da opção “-t”.

10.5.2. Listando regras

A seguinte sintaxe é usada para listar as regras criadas:

```
iptables [-t tabela] -L [chain] [opções]
```

Onde:
tabela

É uma das tabelas usadas pelo iptables. Se a tabela não for especificada, a tabela “filter” será usada como padrão.

chain

Um dos *chains* disponíveis na tabela citada pelo argumento acima. Caso o *chain* não seja especificado, todos os *chains* da tabela serão mostrados.

opções:

As seguintes opções podem ser usadas para listar o conteúdo de *chains*:

-v

Exibe mais detalhes sobre as regras criadas nos *chains*.

-n

Exibe endereços de máquinas/portas como números ao invés de tentar a resolução DNS e consulta ao /etc/services. A resolução de nomes pode tomar muito tempo dependendo da quantidade de regras que suas tabelas possuem e da velocidade de sua conexão.

-x

Exibe números exatos ao invés de números redondos. Também mostra a faixa de portas de uma regra de *firewall*.

--line-numbers

Exibe o número da posição da regra na primeira coluna da listagem.

Para listar a regra criada anteriormente usamos o comando:

```
#iptables -t filter -L INPUT
```

```
Chain INPUT (policy ACCEPT)
```

target	prot	opt	source	destination
DROP	all	--	anywhere	localhost

Os campos exibidos, possuem o seguinte significado:

Chain INPUT

Nome do *chain* listado.

(policy ACCEPT 78 packets, 5820 bytes)

Policiamento padrão do *chain*, neste caso, o padrão é aceitar os pacotes (policy ACCEPT). Esta linha também apresenta o número de pacotes, 78 no exemplo, e o número de *bytes* que atravessaram essa regra.

pkts

Quantidade de pacotes que atravessaram a regra.

bytes

Quantidade de *bytes* que atravessaram a regra. Pode ser referenciado com K (*Kilobytes*), M (*Megabytes*), G (*Gigabytes*).

target

O alvo da regra, o destino do pacote. Pode ser ACCEPT, DROP ou outro chain. Veja Especificando um alvo, Seção 10.3.6 para detalhes sobre a especificação de um alvo.

prot

Protocolo especificado pela regra. Pode assumir os valores “udp”, “tcp”, “icmp” ou “all”.

opt

Opções extras passadas a regra. Normalmente “!”.

in

Interface de entrada (de onde os dados chegam).

out

Interface de saída (para onde os dados vão).

source

Endereço de origem.

destination

Endereço de destino.

outras opções:

Estas opções normalmente aparecem quando são usadas a opção “-x”:

- **dpt** ou **dpts** - Especifica a porta ou faixa de portas de destino.
- **reject-with icmp-port-unreachable** - Significa que foi usado o alvo REJECT naquela regra.

10.5.3. Apagando uma regra

Para apagar uma regra, existem duas alternativas:

1. Quando sabemos qual é o número da regra no *chain* (listado com a opção -L) podemos referenciar a regra por seu número diretamente. Por exemplo, para apagar a primeira regra criada listada:

```
iptables -t filter -D INPUT 1
```

Esta opção não é a mais indicada quando temos um *firewall* complexo com um

grande número de regras por *chains*, neste caso a segunda opção é a mais apropriada, veja:

2. Usamos a mesma sintaxe para criar a regra no *chain*, mas trocamos a opção “-A” por “-D”:

```
iptables -t filter -D INPUT -d 127.0.0.1 -j DROP
```

Então a regra correspondentes no *chain* INPUT da tabela “filter” será automaticamente apagada (confira listando o *chain* com a opção “-L”). Caso o *chain* possua várias regras semelhantes, somente a primeira será apagada.

OBS: Não é possível apagar os *chains* padrões do iptables (INPUT, OUTPUT...).

10.5.4. Inserindo uma regra

Precisamos que o tráfego vindo (opção “-s”) de “192.168.1.15” não seja rejeitado pelo nosso *firewall*. Não podemos adicionar uma nova regra (com a opção “-A”) pois esta seria incluída no final do *chain* e o tráfego seria rejeitado pela primeira regra (nunca atingindo a segunda). A solução é inserir a nova regra antes da regra que bloqueia todo o tráfego para o endereço “127.0.0.1” na posição 1:

```
iptables -t filter -I INPUT 1 -s 192.168.1.15 -d 127.0.0.1 -j ACCEPT
```

Após este comando, temos a regra inserida na primeira posição do *chain* (repare no número 1 após INPUT) e a antiga regra número 1 passa a ser a número 2. Desta forma, a regra acima será consultada, se a máquina de origem possuir o endereço “192.168.1.15”, então os pacotes terão permissão de chegar ao destino (máquina local), caso contrário, o tráfego será bloqueado pela regra seguinte.

10.5.5. Substituindo uma regra

Após criarmos a regra exemplificada na seção “Adicionando uma regra”, percebemos que a nossa intenção, era somente bloquear os pacotes que são enviados pelo utilitário “ping” (pacotes ICMP) para o endereço “127.0.0.1”, e não havia necessidade portanto, de bloquear todo o tráfego que chegasse até a máquina local. Neste caso, existem duas alternativas: a primeira é apagar a regra e inserir uma nova no lugar; a segunda, é modificar diretamente a regra já criada sem afetar outras regras existentes e manter a sua ordem no *chain*. Caso você opte pela segunda alternativa, poderá utilizar o seguinte comando:

```
iptables -t filter -R INPUT 2 -d 127.0.0.1 -p icmp -j DROP
```

O número 2, representa a regra que será substituída no *chain* INPUT da tabela “filter”, e deve ser especificado obrigatoriamente. O comando acima, substituirá a regra 2 do *chain* INPUT (-R INPUT 2) bloqueando (-j DROP) qualquer pacote do protocolo ICMP (-p icmp) com o destino “127.0.0.1” (-d 127.0.0.1).

10.5.6. Criando um novo chain

Em *firewalls* organizados com um grande número de regras, é interessante criar *chains* personalizados para organizar regras de um mesmo tipo, ou que tenham por objetivo, analisar um tráfego de uma mesma categoria (interface, endereço de origem, destino, protocolo, etc), pois podem consumir muitas linhas de *chains* padrões e tornar o gerenciamento do *firewall* confuso (e conseqüentemente causar sérios riscos de segurança). O tamanho máximo para um nome de *chain* é de 31 caracteres e podem conter tanto letras maiúsculas quanto minúsculas. O comando para criar novos *chains* é:

```
iptables [-t tabela] [-N novochain]
```

Para criar o *chain* chamado “internet” (que pode ser usado para agrupar as regras de acesso a Internet), poderíamos usar o seguinte comando, por exemplo:

```
iptables -t filter -N internet
```

Para inserir regras no *chain* "internet", basta especifica-lo após a opção "-A":

```
iptables -t filter -A internet -s 200.200.200.200 -d 127.0.0.1 -j DROP
```

E então criamos um desvio (opção "-j") do *chain* INPUT para o *chain* internet:

```
iptables -t filter -A INPUT -j internet
```

Se uma máquina com o endereço "200.200.200.200" tentar acessar sua máquina (127.0.0.1), o iptables consultará as seguintes regras:

INPUT

```
-----
| Regra1: -s 192.168.1.15 -d 127.0.0.1 -j DROP |
|-----|
| Regra 2: -j internet                          |
|-----|
```

internet

```
-----
| Regra1: -s 200.200.200.200 -d 127.0.0.1 -j DROP |
|-----|
```

O pacote que possui o endereço de origem "200.200.200.200", passa pela primeira regra do *chain* INPUT, a segunda regra deste mesmo *chain*, direciona o fluxo de checagem para o *chain* internet. No *chain* internet, a primeira regra confere com o endereço de origem "200.200.200.200" e o pacote será bloqueado. Após o processamento da única regra existente neste *chain* (internet), o fluxo de checagem é redirecionado ao *chain* anterior (INPUT), e como não há mais nenhuma regra para ser verificada, o processamento é finalizado.

10.5.7. Renomeando um chain criado pelo usuário

Se por algum motivo, for necessário renomear um *chain* criado por você, nas tabelas "filter", "nat" ou "mangle", isto poderá ser feito, usando a opção "-E" do iptables:

```
iptables -t filter -E nome-antigo novo-nome
```

Note que não é possível renomear os *chains* padrões do iptables (INPUT, OUTPUT,...).

10.5.8. Limpando as regras de um chain

Para limpar todas as regras de um *chain*, use o seguinte comando:

```
iptables [-t tabela] [-F chain]
```

Onde:
tabela

É a tabela que contém o *chain* que desejamos excluir todas as regras.

chain

É o *chain* que desejamos limpar. Caso um *chain* não seja especificado, todos os *chains* da tabela terão suas regras excluídas.

Exemplos:

```
iptables -t filter -F INPUT
iptables -t filter -F
```

10.5.9. Apagando um chain criado pelo usuário

Para apagarmos um *chain* criado pelo usuário, usamos a seguinte sintaxe:

```
iptables [-t tabela] [-X chain]
```

Onde:

tabela

Nome da tabela que contém o *chain* que desejamos excluir.

chain

Nome do *chain* que desejamos apagar. Caso não seja especificado, todos os *chains* definidos pelo usuário na tabela especificada serão excluídos.

Os *chains* padrões do iptables não podem ser apagados.

Exemplos:

```
iptables -t filter -X internet  
iptables -X
```

10.5.10. Zerando o contador de bytes dos chains

Este comando zera os campos "pkts" e "bytes" de uma regra do iptables. Estes campos podem ser visualizados com o comando "iptables -L -v". Caso você deseje reiniciar a contagem de pacotes/bytes de um *chain*, use a seguinte sintaxe:

```
iptables [-t tabela] [-Z chain] [-L]
```

Onde:

tabela

Nome da tabela que contém o *chain* que queremos zerar os contadores de bytes e pacotes.

chain

Este argumento define o *chain* que deve ter os contadores zerados. Caso não seja especificado, todos os *chains* da tabela terão os contadores reiniciados. Note que as opções "-Z" e "-L" podem ser usadas juntas, assim o *chain* será listado e imediatamente zerado. Isto evita a passagem de pacotes durante a listagem de um *chain*.

Exemplos:

```
iptables -t filter -Z INPUT
```

10.5.11. Especificando o policiamento padrão de um chain

O policiamento padrão determina o que acontecerá com um pacote quando ele chegar ao final das regras contidas em um *chain* e não se enquadrar a nenhuma delas. O policiamento padrão do iptables é "ACCEPT", mas isto pode ser alterado com o comando:

```
iptables [-t tabela] [-P chain] [ACCEPT/DROP]
```

Onde:

tabela

Tabela que contém o *chain* que desejamos modificar o policiamento padrão.

chain

Define o *chain* que terá o policiamento modificado. O *chain* deve ser especificado obrigatoriamente com este argumento.

ACCEPT/DROP

ACCEPT faz com os pacotes sejam aceitos caso nenhuma regra do *chain* se

enquadre à aquele pacote (usado em regras permissivas). DROP rejeita os pacotes caso nenhuma regra do *chain* se enquadre (usado em regras restritivas).

O policiamento padrão de um *chain*, pode ser exibido através do uso do comando “iptables -L”, conforme o exemplo abaixo:

```
# iptables -L INPUT
```

Chain INPUT (policy ACCEPT)

target	prot	opt	source	destination
DROP	icmp	--	anywhere	localhost

No exemplo acima, o policiamento padrão de INPUT é ACCEPT (policy ACCEPT), o que significa que qualquer pacote que não seja rejeitado pelas regras contidas neste *chain*, serão por padrão aceitos pelo sistema. Para alterar o policiamento padrão deste *chain* usamos o comando:

```
iptables -t filter -P INPUT DROP
```

NOTA: Os policiamentos PERMISSIVOS (ACCEPT), normalmente são usados em conjunto com regras restritivas no *chain* correspondente (tudo é bloqueado e o que sobrar é liberado), já os policiamentos RESTRITIVOS (DROP), são usados em conjunto com regras permissivas no *chain* correlato (tudo é liberado e o que sobrar é bloqueado pelo policiamento padrão).

10.5.12. Especificando um endereço de origem/destino

As opções “-s” (ou “--src” / “--source”) e “-d” (ou “--dst” / “--destination”) servem para especificar endereços de origem e destino respectivamente. É permitido usar um endereço IP completo (como “192.168.1.1”), um *hostname* (nome da máquina), um endereço FQDN (scelepar00001.celepar.parana) ou um par rede/máscara (como “200.200.200.0/255.255.255.0” ou “200.200.200.0/24”).

Caso um endereço/máscara não sejam especificados, é assumido o valor “0/0” como padrão (todas as máquinas de todas as redes). A interpretação dos endereços de origem/destino dependem do *chain* que está sendo especificado (como INPUT ou OUTPUT, por exemplo).

OBS: Caso seja especificado um endereço FQDN e este resolver mais de um endereço IP, serão criadas várias regras, uma para cada endereço IP retornado. É recomendável sempre que possível, a especificação de endereços IP nas regras, pois além de serem muito rápidos (dispensam resolução DNS), são mais seguros para evitar que nosso *firewall* seja enganado por um ataque de “IP spoofing”.

```
# Bloqueia o tráfego vindo da rede “200.200.200.*”:
```

```
iptables -t filter -A INPUT -s 200.200.200.0/24 -j DROP
```

```
# Bloqueia conexões com o destino “10.1.2.3”:
```

```
iptables -t filter -A OUTPUT -d 10.1.2.3 -j DROP
```

```
# Bloqueia o tráfego da máquina “ecelepar00001.celepar.parana” com destino a rede # “210.21.1.3”. Neste exemplo, nossa máquina possui o endereço “210.21.1.3”:
```

```
iptables -t filter -A INPUT -s ecelepar00001.celepar.parana -d 210.21.1.3 -j DROP
```

10.5.13. Especificando a interface de origem/destino

As opções “-i” (ou “--in-interface”) e “-o” (ou “--out-interface”), especificam as interfaces de origem/destino dos pacotes. Nem todas as *chains* aceitam as interfaces de origem/destino simultaneamente, a interface de entrada (-i) nunca poderá ser especificada em um *chain* como o OUTPUT, e a interface de saída (-o) por sua vez,

nunca poderá ser especificada em um *chain* como o INPUT. Abaixo uma rápida referência:

Tabela	Chain	Interface	
		Entrada (-i)	Saída (-o)
filter	INPUT	SIM	NÃO
	OUTPUT	NÃO	SIM
	FORWARD	SIM	SIM
nat	PREROUTING	SIM	NÃO
	OUTPUT	NÃO	SIM
	POSTROUTING	NÃO	SIM
mangle	PREROUTING	SIM	NÃO
	OUTPUT	NÃO	SIM

O caminho do pacote na interface será determinado pelo tipo da interface e pela posição dos *chains* nas etapas de seu roteamento. O *chain* OUTPUT da tabela “filter” somente poderá conter a interface de saída (veja a tabela acima). No caso do *chain* FORWARD da mesma tabela, podemos perceber que este é o único que aceita a especificação de ambas as interfaces, este é um ótimo *chain* para controlar o tráfego que passa entre as diferentes interfaces de rede do *firewall*.

Por exemplo, para bloquear o acesso do tráfego de qualquer máquina com o endereço IP “200.123.123.10”, que venha da interface ppp0 (uma placa de fax-modem), podemos utilizar o seguinte comando:

```
iptables -t filter -A INPUT -s 200.123.123.10 -i ppp0 -j DROP
```

A mesma regra pode ser especificada de uma forma um pouco diferente:

```
iptables -t filter -A INPUT -s 200.123.123.10 -i ppp+ -j DROP
```

O sinal de “+” funciona como um coringa, assim a regra terá efeito em qualquer interface de ppp0 a ppp9. As interfaces ativas no momento podem ser listadas com o comando “**ifconfig**”. Também é permitido, especificar uma regra que faça referência a uma interface que ainda não existe, isto é interessante, para conexões intermitentes como o PPP (acesso a Internet via modem).

Vamos a outro exemplo, suponha agora que queremos bloquear a passagem de tráfego da interface ppp0 para a interface eth1 (uma de nossas redes internas). Podemos fazer isso usando o seguinte comando:

```
iptables -t filter -A FORWARD -i ppp0 -o eth1 -j DROP
```

10.5.14. Especificando um protocolo

A opção “-p” (ou “--protocol”) é usada para especificar protocolos que podem ser configurados para uso com o iptables. Os protocolos que podem ser usados são: TCP, UDP e ICMP. Por exemplo, supondo que se deseja rejeitar todos os pacotes UDP vindos do endereço “200.200.200.200”, podemos incluir a seguinte regra no iptables:

```
iptables -t filter -A INPUT -s 200.200.200.200 -p udp -j DROP
```

OBS: Os nomes de protocolos podem ser especificados usando letras maiúsculas ou minúsculas.

10.5.15. Especificando portas de origem/destino

As portas de origem/destino devem ser especificadas após o protocolo, e podem ser precedidas por uma das seguintes opções:

- “--source-port” ou “--sport” - Especifica uma porta ou faixa de portas de origem.

- “--destination-port” ou “--dport” - Especifica uma porta ou faixa de portas de destino.

Uma faixa de portas pode ser especificada através do formato “PortaOrigem:PortaDestino”, conforme exemplificado logo abaixo:

```
iptables -t filter -A OUTPUT -d 200.200.200.200 -p tcp  
--dport 0:1024 -j DROP
```

No exemplo acima, qualquer pacote com destino “200.200.200.200” na faixa de portas de 0 a 1024 do protocolo TCP será bloqueado pelo *firewall*.

Caso o argumento relativo a porta de origem de uma faixa de portas não seja especificado, o valor 0 (zero) é assumido por padrão, caso o argumento não especificado seja a porta de destino, então, o valor 65535 é assumido por padrão.

11. Compilação do Kernel

Um dos assuntos que mais despertam curiosidades em pessoas que estão aprendendo a administrar o GNU/Linux é o *Kernel*. De forma especial, a compilação do *Kernel*, é uma das atividades que as pessoas ligadas ao Linux mais gostam de mistificar. A compilação do *Kernel*, exige alguns cuidados é claro, mas não é de forma nenhuma o “bicho de sete cabeças” que muitos pintam.

Além do mais, a recompilação do código fonte do núcleo, é uma operação que hora ou outra será necessária para muitos usuários. De fato, muitas vezes o *Kernel* não oferece suporte nativo a dispositivos de hardware que foram lançados muito recentemente, o que implica na compilação para adicionar um módulo ou funcionalidade. Em outros casos, é interessante compilar para fazer o núcleo ficar sob medida para a máquina a qual ele está instalado, enxugando funcionalidades que não são usadas pelo usuário final e por conseqüência, obtendo um produto menor e em alguns casos com uma melhor performance. Enfim, há muitas razões para realizar a compilação do *Kernel*.

Nesta seção, apresentaremos ao leitor, o procedimento para realizar a compilação de *Kernel* no GNU/Debian versão “etch”. O mais importante neste processo todo, é que você busque informações a respeito do núcleo que irá compilar. Há muitas páginas na Internet dedicadas ao *Kernel* do GNU/Linux, mas é realmente interessante, estar sempre a par do que está acontecendo com o kernel através do endereço “<http://kernel.org>”, pois este é o site oficial do desenvolvimento do núcleo do sistema Linux.

11.1. Preparação para Compilação

Antes de mais nada, é importante lembrar, que assim como a grande maioria dos códigos fonte de programas comuns, o *Kernel* também precisa de algumas bibliotecas para ser compilado. As bibliotecas necessárias, podem variar de acordo com a versão e dos recursos do núcleo a serem compilados e das interfaces de configuração pré-compilação que serão utilizadas pelo administrador, para ajustar as opções de compilação que o *Kernel* possui.

Você poderá utilizar o APT, ferramenta de instalação/configuração de pacotes do GNU/Debian, para procurar pelas versões do *Kernel* oficialmente disponibilizadas pela Debian ou baixar as últimas versões disponíveis do software, através do site oficial do *Kernel* em “<http://kernel.org>”. Recomendamos fortemente, que você compile versões disponibilizadas oficialmente pelo time da Debian, isto porque, estas versões contem ajustes implementados pela equipe de desenvolvimento do GNU/Debian, que visam adequar o *Kernel* disponibilizado no site oficial as particularidades do sistema Debian.

Caso você opte por baixar o código fonte via APT, de acordo com nossa recomendação, confira se o arquivo “/etc/apt/sources.list” está configurado conforme indicado no capítulo sobre a instalação do GNU/Debian, deve haver a seguinte linha:

deb-src <http://www.repositorios.eparana.parana.debian/> etch main

Para verificar as versões do *Kernel* disponíveis através do APT, execute os comandos:

```
apt-get update
apt-cache search linux-image
```

Depois de ter decidido qual versão do núcleo será compilada, necessitamos baixar o código fonte do mesmo, para posteriormente, instalar as bibliotecas necessárias para sua compilação. A seguir, os procedimentos para realização de cada uma destas etapas:

1. Baixar o código fonte via APT. Em nosso exemplo, vamos usar o código do último *Kernel* disponibilizado para o Debian etch, a versão “2.6.18”:

```
apt-get source linux-image-2.6.18-5-686
```

2. Com o comando indicado no primeiro passo, o código será baixado, extraído e atualizado diretamente no diretório de compilação padrão “/usr/src”. Agora é necessário baixarmos os pacotes necessários para compilação da versão do *Kernel* escolhida. Isto pode ser feito com o seguinte comando:

```
apt-get build-dep linux-image-2.6.18-5-686
```

3. Instale a biblioteca padrão para compilação:

```
apt-get install libncurses5-dev
```

Agora, podemos modificar algumas opções do arquivo de configuração “/etc/kernel-pkg.conf”, do pacote “kernel-package”, para personalizar a compilação do novo núcleo. Para editar este arquivo, pode-se usar o comando “vi /etc/kernel-pkg.conf”. Abaixo, os parâmetro que alteramos neste arquivo para exemplo. As linhas que iniciam com o caractere “#” são comentários e não possuem funcionalidade alguma:

```
#### Arquivo "/etc/kernel-pkg.conf" ####
#Nome do mantenedor do Kernel
maintainer := Marcius Marcelo Roger
#E-mail do mantenedor
email := marcius@celepar.pr.gov.br
#Prioridade desta revisão do Kernel. Normalmente utilizamos prioridade baixa (valor
"Low")
priority := Low
```

Depois de realizar as mudanças, é necessário salvar o arquivo de configuração.

11.2. Compilando o Kernel

Os códigos fontes do *Kernel* para serem compilados devem estar armazenados logo abaixo do diretório “/usr/src”. Caso você tenha optado por baixar os fontes através do APT, eles serão colocados neste local automaticamente. Caso você tenha pego o arquivo com os fontes no *site* oficial, ele deverá ser descompactado também nesta localidade (“/usr/src”).

Como primeiro passo para realizar o processo de compilação, acesse o diretório gerado pela descompactação do *Kernel* que irá ser compilado:

```
cd /usr/src/linux-2.6.18
```

Substitua a pasta “linux-2.6.18” do exemplo acima, pelo nome da pasta gerada pela descompactação do *Kernel* que você está compilando em seu sistema.

Com nosso próximo passo, o processo de compilação irá de fato se iniciar. Vamos usar a interface modo texto do “menuconfig” para configurar as opções de compilação. Existem outras interfaces como o “gconfig”, elas permitem fazer a seleção das opções de compilação via ambiente gráfico, contudo, essas interfaces demandam a instalação de bibliotecas específicas para sua utilização. Caso você esteja interessado em utilizar interfaces de configuração no modo gráfico, pesquise as dependências para compilação do código da interface em questão (gconfig, xconfig, etc.) e instale esse *software* no sistema antes de tentar compilar o seu código, caso contrário a compilação da interface não será possível.

```
make menuconfig
```

Depois de compilado, o utilitário de configuração “menuconfig” será automaticamente carregado. Nele você poderá configurar todas as opções que desejar para compilar seu *Kernel* de acordo com suas necessidades. Nos utilitários de configuração para compilação, como no caso do menuconfig, você encontrará uma ajuda *online* que explica o que é, e o que faz, cada uma das opções disponíveis.

Após configurar todas as opções necessárias, saia do utilitário através selecionando a opção “< Exit >” do configurador. Você será questionado se deseja salvar as alterações realizadas “Do you wish to save your new kernel configuration?”, responda “< Yes >” para salvar.

O próximo passo, é limpar os arquivos objetos gerados por compilações anteriores, isso só é realmente necessário, caso você já tenha compilado o código do *Kernel* anteriormente, caso essa seja sua primeira tentativa de compilação, você poderá pular esta etapa.

```
make-kpkg clean
```

Finalmente, chegou a hora de executar o comando que fará a compilação do núcleo e criará um pacote (“.deb”) para instalação apropriada deste no sistema.

```
make-kpkg --append-to-version "-personalizado" --initrd --us --uc
kernel_image
```

No comando acima, os parâmetros utilizados tem o seguinte significado:

--append-to-version

Acrescenta a palavra que você indicar (no exemplo, "-personalizado") à versão do pacote criado. É útil para diferenciar seu *Kernel* recompilado de um "oficial";

--initrd

Indica que junto com o *Kernel*, deve ser criado um arquivo "initrd", que é o padrão da Debian (e o modo mais flexível). Para mais informações, "man 4 initrd";

--us e --uc

Estas opções fazem com que o pacote seja criado sem uma assinatura GPG e sem modificações do arquivo *changelog*, respectivamente. Se desejar entender melhor estas questões, "man dpkg-buildpackage";

kernel_image

É a ação que o utilitário "make-kpkg" vai executar, ou seja, com essa ação, ele cria um pacote no formato Debian com a imagem do *Kernel*, para que você possa instalá-lo facilmente em seu sistema depois. Há outras ações possíveis, caso você esteja curioso, veja a documentação "man make-kpkg".

Vá tomar um café e relaxe. Essa operação é um tanto demorada. Tenha um pouco de paciência e aguarde o seu término. Ao final da compilação, caso o sistema pergunte algo conforme o parágrafo seguinte (por causa do parâmetro "--initrd"):

Warning: You are using the initrd option, that may not work unless you have applied the initrd cramfs patch to the kernel, or modified mkinitrd not to use cramfs by default. The cramfs initrd patch, is included in the Debian supplied kernel sources, but is not present in pristine kernel sources. By default, I assume you know what you are doing, and I apologize for being so annoying. Should I abort[Ny]?

É só digitar "n", e em seguida pressionar <ENTER> para continuar.

Agora é só instalar o pacote que foi criado pelo utilitário "make-kpkg" durante a compilação do código fonte do núcleo. O pacote é criado no diretório um nível acima a partir de onde foi compilado, ou seja, ele se encontra em "/usr/src". Para instalar o pacote execute cada um dos comandos abaixo:

```
cd /usr/src
dpkg -i linux-image-2.6.18-5-personalizado_i386.deb
```

Pronto! O sistema já cria e configura automaticamente os arquivos e o gerenciador de *boot* para inicialização do SO a partir do novo núcleo. Agora basta reiniciar o sistema e testar seu novo *Kernel*. Caso ocorra algum problema durante este processo, basta reiniciar o computador e iniciar o sistema a partir da imagem antiga do *Kernel* (a que já existia antes da compilação). Boa sorte !

11.3. Utilitários do Kernel

11.3.1. depmod

Alguns módulos provêem serviços (conhecidos como símbolos) que podem ser usados por outros módulos do sistema. Se um módulo utiliza um serviço de outro, isto gera uma dependência. Essas dependências podem se tornar demasiadamente complexas.

O utilitário “depmod”, é responsável por analisar o conteúdo do diretório “/lib/modules/VERSÃO_DO_KERNEL_UTILIZADA”, e gerar um arquivo (“modules.dep”) contendo a relação de dependência de cada módulo presente naquele diretório.

O utilitário “**modprobe**” necessita de uma lista atualizada de dependências de módulos gerada por depmod para poder operar. Em algumas situações, torna-se necessária a execução do comando depmod. Por exemplo, quando um módulo novo foi instalado no diretório “/lib/modules/VERSÃO_DO_KERNEL_UTILIZADA” e necessita-se carregá-lo através do comando modprobe.

depmod [opções] [arquivo]

Onde:
arquivo

É o nome do arquivo módulo que se deseja analisar as dependências, e por conseguinte, gravar esta informação em um arquivo mantido por depmod. Deve-se informar o caminho completo para o arquivo que se deseja analisar.

opções:

-a

Analisa a dependência de todos os módulos do sistema, contidos no diretório “/lib/modules/VERSÃO_DO_KERNEL_UTILIZADA”. Esta é a opção padrão caso nenhum arquivo módulo seja informado.

-b diretório

Caso os arquivos de módulos do sistema não estejam localizados sobre o diretório normal “/lib/modules/VERSÃO_DO_KERNEL_UTILIZADA”, permite especificar um diretório alternativo para realizar a análise de dependências.

-e

Imprime uma lista de todos os símbolos não resolvidos.

-n

Exibe o arquivo de dependência na saída padrão.

-q

Não exibe mensagens de erro e símbolos ausentes.

-r

Permite que o usuário root carregue módulos não pertencentes a ele.

-s

Redireciona as mensagens de erro para o mecanismo de log do sistema (syslog).

-v

Imprime a lista com todos os módulos processados.

-A

Faz a atualização rápida do arquivo de dependências. Para isso, utiliza a data e hora e verifica as mudanças recentes no diretório

“/lib/modules/VERSÃO_DO_KERNEL_UTILIZADA”.

-C arquivo

Utiliza um arquivo de configuração alternativo em vez de “/etc/modules.conf”. Também é possível usar as configurações a partir da variável de ambiente “MODULE-CONF”

-F arquivo

Usa um arquivo de símbolos alternativo para construir as dependências. Geralmente, se usa uma cópia do arquivo “System.map” de um outro sistema ou a saída de “/proc/ksyms”.

Exemplos:

depmod -a

Constrói um arquivo de dependências usando todos os módulos disponíveis no sistema.

depmod -n

Gera o arquivo de dependências e o imprime na saída padrão (tela).

depmod -v /lib/modules/2.6.18-5-k7/kernel/drivers/char/drm/drm.ko

Gera as dependências do módulo “drm.ko” e exibe detalhes sobre este procedimento para o usuário.

11.3.2. insmod

Carrega um módulo do *kernel* em tempo de execução. O utilitário “insmod”, não resolve dependências no momento da carregamento de um módulo. Por essas e outras facilidades, muitos usuários deixaram de usar insmod e passaram a usar o comando “**modprobe**”.

insmod arquivo [parâmetros]

Onde:

arquivo

É o arquivo módulo que se deseja carregar.

parâmetros

São os parâmetros do módulo que estão disponíveis para serem utilizados no momento do carregamento do módulo. Nem todos os módulos possuem parâmetros de carregamento.

Exemplos:

insmod /lib/modules/2.6.18-5-k7/kernel/drivers/char/drm/drm.ko

Carrega o módulo “drm”, armazenado no arquivo “drm.ko”.

insmod /lib/modules/2.6.18-5-k7/kernel/drivers/char/drm/drm.ko cards_limit=2

Carrega o módulo “drm”, armazenado no arquivo “drm.ko”, com o parâmetro limitador do número de placas de vídeo.

11.3.3. lsmod

É um comando que lista todos os módulos carregados pelo *kernel*, apresentando informações como: nome do módulo, tamanho (em unidades de 4 Kb), número de módulos dependentes, e se for o caso, nome dos módulos dependentes. A mesma informação pode ser obtida em “/proc/modules”. O comando “lsmod” não possui parâmetros, sendo muito simples de usar.

lsmod

11.3.4. modinfo

O utilitário “modinfo” extraí informações de módulos do *kernel* do Linux, para exibi-las ao administrador do sistema. Em geral, para se utilizar este comando, é fornecido o caminho e o nome de um arquivo módulo e o modinfo retornará as informações sobre o módulo em questão. Algumas vezes, o administrador poderá fazer a busca pelo nome oficial do módulo ao invés do nome do arquivo que é utilizado para armazená-lo, neste caso, modinfo consultará o diretório “/lib/modules/VERSÃO_DO_KERNEL_UTILIZADA”, assim como comando “**modprobe**”, para tentar detectar a qual arquivo aquele nome de módulo se refere, e se possível, extrair informação a partir dele.

Por padrão, as informações mostradas pelo comando modinfo são: nome do módulo, descrição, autor, licença, dependências e parâmetros.

modinfo [opções] arquivo/nome

Onde:

arquivo/nome

Representa o caminho completo do arquivo do módulo que se deseja consultar, ou apenas, o nome oficial do módulo em questão.

opções:

-F campo

Exibe apenas a informação contida no argumento “campo” na tela. Argumentos válidos para utilização em “campo”, são: “author”, “description”, “license”, “param”, “depends”, “alias”, “param”, “depends” e “filename”.

-a

Lista o nome do autor do módulo.

-d

Exibe a descrição do módulo.

-l

Exibe a licença.

-p

Exibe os parâmetros para utilização daquele módulo.

-n

O nome do arquivo.

Exemplos:

modinfo -F author thermal

Exibe o nome do autor do módulo “thermal”.

modinfo /lib/modules/2.6.18-5-k7/kernel/drivers/acpi/thermal.ko

Exibe todas as informações disponíveis sobre o arquivo módulo “thermal.ko”.

11.3.5. modprobe

Adiciona ou remove módulos do *kernel* em tempo de execução. O “modprobe” é um utilitário inteligente, capaz de resolver as dependências de um módulo que está sendo adicionado ou removido.

Assim como acontece com o utilitário “**modinfo**”, você poderá especificar o caminho completo mais o nome do arquivo módulo, ou apenas, o nome oficial do módulo para carregar/descarregá-lo com o modprobe. Caso seja informado um nome de módulo ao invés de um nome de arquivo, o utilitário irá fazer uma busca no diretório “/lib/modules/VERSÃO_DO_KERNEL_UTILIZADA”, para tentar carregar/descarregar o módulo em questão.

Em alguns sistemas, o modprobe pode possuir um arquivo de configuração, o “/etc/modprobe.conf”, e na maioria das vezes, a estrutura de diretórios “/etc/modprobe.d/” estará presente, para organizar e facilitar a configuração de um dado conjunto de módulos daquela maquina. Um módulo carregado através do modprobe, não será automaticamente carregado na próxima inicialização da maquina. Você poderá especificar os módulos a serem carregados durante a inicialização do sistema, através do arquivo “/etc/modules”.

modprobe [opções] arquivo/nome

Onde:

arquivo/nome

Representa o caminho completo do arquivo do módulo que se deseja carregar/descarregar, ou apenas, o nome oficial do módulo em questão.

opções:

-v

Exibe mensagens sobre o que modprobe está fazendo.

-C arquivo

Ignora as configurações do arquivo “/etc/modprobe.conf” e utiliza as configurações providas por um arquivo alternativo, definido através do argumento “arquivo”, e também as definições contidas na variável de ambiente “MODPROBE_OPTIONS”.

-c

Exibe o conteúdo do arquivo de configuração “/etc/modprobe.conf”. Não utilize esta opção em conjunto com as demais.

-i, --ignore-install, --ignore-remove

Essa opção faz com que qualquer comando “install” ou “remove” do arquivo de configuração “/etc/modprobe.conf”, seja ignorada.

-q

Faz com mensagens geradas durante a execução de modprobe não sejam exibidas para o usuário.

-r, --remove

Esta opção faz com que um módulo do kernel seja descarregado. Caso o módulo contenha dependências, e caso elas também não estejam sendo utilizadas por outros módulos, elas também serão descarregadas. Você pode especificar mais de um módulo para ser descarregado na mesma linha de execução de modprobe, ao contrário do que acontece na ação de carregamento.

-l expressão

Lista todos os módulos contidos no diretório “/lib/modules/VERSÃO_DO_KERNEL_UTILIZADA”, que conferem com o padrão fornecido através do argumento “expressão”. O uso de caracteres coringa também são permitidos para formar a expressão a ser pesquisada.

-a

Carrega todos os módulos contidos na linha de execução do comando.

-t diretório

Restringe a busca, quando a opção “-l” é usada, somente ao diretório especificado pelo usuário.

-s

Esta opção faz com que todas as mensagens de erro geradas pela execução do modprobe, sejam redirecionadas ao mecanismo de log do sistema.

--set-version versão

Permite que o usuário defina uma versão do kernel a ser utilizada no momento da instalação de um módulo, ao invés de obter esta informação automaticamente através do utilitário “uname” (este é o padrão).

--show-depends módulo

Exibe os arquivos módulo correspondente a cada dependência que este possui, incluindo o próprio arquivo correspondente ao módulo em questão. Exibe informação relevante para ser utilizada com o comando “**insmod**”.

-o nome

Permite renomear um módulo já carregado. Isto é útil somente no caso de testes, em geral feito por desenvolvedores, já que não é possível carregar módulos que possuem o mesmo nome.

Exemplos:

modprobe sis

Carrega o módulo “sis”, utilizado para placas de vídeo SIS.

modprobe -l ther*

Lista todos os arquivos de módulos que iniciam com a expressão “ther”.

modprobe -s -a sis thermal floppy

Carrega os módulos “sis”, “thermal” e “floppy” e redireciona os erros para o sistema de log do Linux (syslog).

modprobe -r floppy

Descarrega o módulo “floppy”.

modprobe --show-depends thermal

Exibe as dependências para o módulo “thermal”, incluindo o próprio arquivo do módulo em questão.

modprobe -o disquete floppy

Renomeia o módulo carregado “floppy” com o nome “disquete”.

11.3.6. rmmod

É um programa para remover um módulo carregado da memória (descarregar o módulo). Muitos usuários tem optado por utilizar o comando “**modprobe -r**” em vez de “rmmod”, pela eficiência e facilidade de uso do primeiro comando.

rmmod [opções] módulo

Onde:**módulo**

É o nome oficial do módulo que se deseja descarregar (remover da memória, desconectar do *kernel*).

opções:**-v**

Imprime as ações que estão sendo realizadas pelo rmmod.

-f

Opção extremamente perigosa. Força a remoção de um módulo do kernel, mesmo que este, esteja sendo utilizado por outros módulos. Só use esta opção em casos emergenciais.

-w

Faz com que `rmmod` isole o módulo em questão, e aguarde até que todos os recursos do sistema não necessitem mais utilizá-lo, e faça a remoção segura do módulo da memória.

-s

Envia os erros encontrados durante a operação ao mecanismo de log do sistema (`syslog`).

Exemplos:

`rmmod drm`

Remove o módulo “`drm`” da memória (descarrega o módulo).

`rmmod -w sis`

Aguarda até que nenhum outro recurso esteja utilizando o módulo “`sis`”, para descarregá-lo da memória de forma segura.

`rmmod -f floppy`

Força o descarregamento do módulo do kernel denominado “`floppy`”.

11.3.7. `sysctl`

O utilitário “`sysctl`” é uma interface que permite examinar e alterar dinamicamente parâmetros do *Kernel* do GNU/Linux. Geralmente, esses parâmetros (identificados como objetos na MIB – *Management Information Base*) são usados para ajustes como limitar o tamanho do segmento da memória compartilhada, o número de *threads* que o SO pode utilizar, o número máximo de processos executados pelo sistema, alterar funcionalidades como *IP forwarding*, restrições de segurança para o superusuário (`root`) e níveis de depuração.

A grande vantagem de se alterar parâmetros do *Kernel* via `sysctl`, é que isto evita em muitos casos, a recompilação do núcleo do sistema. O `sysctl` é a ferramenta mais importante do sistema para se fazer ajustes finos de funcionalidades (*tunning*), e por isso, é necessário que o administrador do sistema domine o uso deste utilitário, tanto para sanar problemas como para melhorar o desempenho do sistema.

`sysctl [opções] [variável] [=valor]`

Onde:

variável

É a opção que define a variável a ser consultada/alterada com o uso de `sysctl`.

valor

Esse parâmetro só pode ser utilizado com a opção anterior (*variável*), e permite estipular o novo valor que deverá ser assumido pela variável que está sendo alterada.

opções:

-n

Esta opção evita que o nome da variável seja impressa juntamente com seu valor, esta é a forma padrão da saída do `sysctl` (ex.: `kernel.hostname = ecelepar00001`), sendo que somente o valor da variável será impresso quando “`-n`” for informado.

-e

Ignora erros quando a variável informada não existe.

-N

Imprime o nome da variável em vez do seu valor. Funciona de modo contrário a opção “`-n`”.

-q

Por padrão, quando um valor é alterado, `sysctl` exibe a alteração realizada na tela. Use esta opção caso você não deseje este tipo de comportamento.

-w

Use esta opção para alterar o valor de uma variável.

-p [arquivo]

Permite carregar as opções para o sysctl através de um arquivo, definido através do argumento homônimo ou através de “/etc/sysctl.conf” sem nenhum argumento fornecido.

-a

Imprime todas as variáveis disponíveis no sistema.

-A

Imprime todas as variáveis disponíveis no sistema de forma tabulada.

Exemplos:

sysctl -a

Imprime todas as variáveis do sistema que o administrador poderá modificar.

sysctl kernel.hostname

Imprime o valor da variável “kernel.hostname” no formato padrão usado por sysctl.

sysctl -w kernel.hostname=maquina

Ajusta o valor da variável “kernel.hostname” para “maquina”.

12. Anexos

12.1. NFS - Network File System

- Serviço de rede que permite o compartilhamento transparente de sistemas de arquivos ou diretórios entre os nós de uma rede. Utiliza a implementação de RPC (Remote Procedure Call), cujos protocolos são descritos usando XDR (eXternal Data Representation).

Para exemplificar o uso vamos apenas criar um exemplo para que rapidamente possa estar utilizando esse recurso. Em nosso exemplo teremos apenas um cliente e um servidor com os endereços abaixo:

Servidor: 10.15.15.32

Cliente: 10.15.15.45

No servidor proceda da seguinte forma:

Para poder utilizar o nfs é necessário instalar os pacotes abaixo:

```
# apt-get install nfs-common nfs-kernel-server portmap
```

O arquivo **/etc/exports** determina os sistemas de arquivos a serem exportados e abaixo está um exemplo da sintaxe a ser utilizada:

```
/mnt/publico -rw 10.15.15.45
```

Para que nosso exemplo possa funcionar corretamente, crie no servidor a pasta **/mnt/publico** com os comandos abaixo:

```
# mkdir /mnt/publico
```

```
# chmod 777 /mnt/publico
```

Reinicie os serviços abaixo para garantir o acesso:

```
# /etc/init.d/nfs-kernel-server restart
```

```
# /etc/init.d/nfs-common restart
```

```
# /etc/init.d/portmap restart
```

Na estação cliente faça do seguinte comando para testar se o sistema de arquivos está disponível:

```
# mkdir /mnt/publico
```

```
# sudo mount -t nfs 10.15.15.32:/mnt/publico /mnt/publico
```

```
# mount
```

No resultado do comando mount deverá aparecer a seguinte linha:

```
10.15.15.32:/mnt/publico on /mnt/publico type nfs (rw,addr=10.15.15.32)
```

12.2. Shell Scripts

Muitas vezes o administrador do sistema ou mesmo usuários comuns, tem a necessidade de executar tarefas rotineiras. A automação de tarefas é um dos grandes objetivos que a informática deve cumprir. Por isso, faremos neste tópico uma rápida introdução sobre o uso de *scripts shell*.

O uso de *scripts shell*, permite que os usuários automatizem tarefas que são executadas de forma sistematizada, ou seja, sempre da mesma forma, e com isso, economizem tempo e evitem erros.

O administrador também deve se familiarizar com uso de *shell scripts*, pelo fato destes serem a base para construção dos pacotes utilizados pelo GNU/Debian. A maioria dos serviços do sistema, também utilizam *scripts* para sua manutenção e funcionamento, daí a importância de compreender bem como os *shell scripts* funcionam.

Um *shell script*, contém os mesmos comandos que utilizamos na linha de comandos, todos organizados de forma lógica, e na maioria das vezes, fazendo uso de estruturas de controle ("if", "while", "test", "case", "for", etc).

12.2.1. O primeiro script

Como um *script* é um arquivo de texto comum, podemos editá-lo com nosso editor de textos favorito. Em nossos exemplos utilizaremos o "vim". Abaixo descreveremos passo à passo, como criar nosso primeiro *script* funcional:

1. Criar o arquivo que armazenará nosso *script*:
`vim /home/usuario/primeiro_script`
2. Adicionar a chamada ao *shell* a ser utilizado para interpretar os comandos, na primeira linha do *script*, inserindo o seguinte texto ao arquivo:
`#!/bin/bash`
3. Inserir nosso primeiro comando ao *script*, na linha posterior a chamada do *shell*, ficando o arquivo conforme mostrado a seguir:
`#!/bin/bash`
`echo "Meu script funciona !"`
4. O próximo passo, é salvar o arquivo (no vim, pressione **<ESC>** e a sequência `":x"`) e ajustar a permissão de execução do arquivo com o comando:
`chmod +x /home/usuario/primeiro_script`
5. A permissão de execução é necessária para tornar o arquivo texto comum num executável. Agora, vamos testar nosso *script*, executando-o como um comando qualquer:
`/home/usuario/primeiro_script`
6. Após digitar o caminho completo para seu *script* e pressionar a tecla **<ENTER>**, caso você tenha feito tudo conforme explicado, você deverá ver a mensagem "Meu script funciona !" em sua tela. Caso você esteja atualmente dentro do diretório "/home/usuario" (em nosso exemplo), poderá executar o *script* apenas digitando `./primeiro_script`.

12.2.2. Os primeiros erros

É muito comum quando estamos aprendendo cometer muitos erros. Listaremos aqui, os erros mais comuns que podem acontecer durante a criação e execução dos seus primeiros *scripts*:

- **Arquivo ou diretório não encontrado**

Este erro significa que o interpretador de comandos (o bash, em nossos exemplos) não encontrou o arquivo que contém o seu *script*. Verifique se você está invocando o *script* usando o caminho correto, ou se realmente, o arquivo foi criado e está armazenado no caminho que você está indicando para execução.

- **Permissão negada**

Este erro ocorrerá caso você esteja tentando executar um arquivo, o qual não possua permissão de execução (permissão "x", concedida com o comando "chmod"), ou caso você não possua permissões efetivas para executar o arquivo, ou seja, não seja o dono do arquivo, nem faça parte do grupo de acesso e a permissão para outros não conceda permissão de execução.

- **Erro de sintaxe**

Isso acontecerá caso seu arquivo seja encontrado, você possua as devidas permissões para executá-lo, mas o arquivo contenha erros de sintaxe no uso dos comandos que você inseriu. Lembre-se, os comandos devem ser utilizados com 100% de acurácia em relação a sua sintaxe, para que o *script* possa ser executado sem erros.

12.2.3. Melhorando seu primeiro script

Vamos incrementar o nosso pequeno *script*. Como você já conhece os passos para criação e execução do arquivo, pularemos esta parte e iremos mostrar o *script* que implementaremos:

```
#!/bin/bash
#Vamos saudar o usuário.
#0 "\n" é um código especial para inserir uma nova linha.
echo -e "Bom dia $USERNAME, hoje é: \n"
date
echo -e "\n"
echo "Você quer ter mais informações ? [s\n]"
#Lê a resposta informada pelo usuário.
#"RESPOSTA" é uma variável.
read RESPOSTA
#0 comando "test", testa a resposta dada pelo usuário.
#Se for diferente de "s", então a execução do script é finalizada.
test "$RESPOSTA" != "s" && exit
#0 "\t" é um código especial para inserção de espaçamento
tabulado.
echo -e "\t Uso do disco:\n"
df
echo -e "\n"
echo -e "\t Uso da memória: \n"
free
```

12.2.4. Expressões aritméticas

Você também pode realizar cálculos matemáticos simples (somar, subtrair, multiplicar e dividir) usando o *shell*. Veja o exemplo abaixo:

```
#!/bin/bash
#Efetua um cálculo matemático simples.
echo -e "2 x 3 = $((2*3))"
echo -e "(5 + 3)/2 = $(((5+3)/2))"
echo -e "4 - 2 = $((4-2))"
```

Perceba que para realizar os cálculos, é necessário que os operandos e operadores estejam entre a expressão "\$((...))".

Caso você necessite executar cálculos mais complexos via linha de comando, poderá instalar alguns utilitários que atuam no modo texto, especialmente criados para esta finalidade como o pacote "wcalc".

12.2.5. Recebendo argumentos do shell

Muitas vezes, é preciso que o usuário informe parâmetros que vão nortear a execução de um *script*. Nestes casos, podemos capturar os argumentos enviados pelo usuário através da linha de execução do *script*, para usá-los de alguma forma no decorrer do fluxo de execução do próprio *script*.

Por exemplo, suponha que o usuário tenha executado o script com o comando abaixo:

```
# ./meu_script Marcius Malta
```

O nosso *script* está preparado para tratar os argumentos que foram enviados via linha de comando. Ele primeiro imprimirá o primeiro e o segundo argumento, e depois, unirá e imprimirá os dois. Veja como ele foi organizado para isto:

```
#!/bin/bash
#Imprime o valor do primeiro parâmetro.
echo -e "Primeiro parâmetro: $1 \n"
#Imprime o valor do segundo parâmetro.
echo -e "Segundo parâmetro: $2 \n"
#Junta e imprime os parâmetros.
echo -e "$1 $2"
```

Como resultado, a terceira linha do *script* deverá mostrar a frase "Primeiro parâmetro: Marcius" quando executada, a quinta linha "Segundo parâmetro: Malta" e a última linha, "Marcius Malta".

12.2.6. Estruturas de controle

Neste tópico explicaremos o uso das estruturas de controle mais utilizadas: "if", "while", "for" e "case". Para utilizar qualquer uma das estruturas de controle, é fundamental zelar pela sintaxe de utilização de cada estrutura. Tudo deve ser rigidamente obedecido afim de evitar erros de sintaxe durante a execução do *script*.

Para testar valores em cada uma das estruturas que apresentaremos a seguir, você poderá utilizar os seguintes operadores sobre os operandos que desejar nas estruturas de teste em questão:

Operador	Significado
-eq	Igual
!=	Diferente
-gt	Maior
-lt	Menor
&&	E lógico
	OU lógico
-d	Se for um diretório
-e	Se existir
-z	Se estiver vazio
-f	Se contiver texto
-o	Se o usuário for o dono
-r	Se o arquivo pode ser lido
-w	Se o arquivo pode ser alterado
-x	Se o arquivo pode ser executado

– A estrutura if

O "if" é a estrutura de testes condicionais. Ele é semelhante ao "if" utilizado na linguagem C/C++. Abaixo a sintaxe de uso:

```
if TESTE
then
  comandos
elif TESTE
  comandos
else
  comandos
```

fi

Exemplos:

Exemplo 01: Testa se o arquivo chamado "meu_texto.txt" existe.

```
#!/bin/bash
#Testa se o arquivo chamado "meu_texto.txt" existe.
#Note que os espaços utilizados são obrigatórios.
if [ -e /home/usuario/meu_texto.txt ]
#Note também, que o "then" fica numa linha própria.
then
    echo "O arquivo existe."
else
    echo "O arquivo não existe."
fi
```

Exemplo 02: Testa se um número é maior do que outro.

```
#!/bin/bash
#Testa se o arquivo chamado "meu_texto.txt" existe.
#Note que os espaços utilizados são obrigatórios.
if [ 5 -gt 4 ]
#Note também, que o "then" fica numa linha própria.
then
    echo "Sim, 5 é maior do que 4."
else
    echo "Não, 5 não é maior do que 4."
fi
```

Exemplo 03: Faz dois testes numéricos usando o "E" lógico.

```
#!/bin/bash
#Testa se o arquivo chamado "meu_texto.txt" existe.
#Testando se as duas condições são verdadeiras.
if [ 5 -eq 5 ] && [ 3 -gt 5 ]
#Note também, que o "then" fica numa linha própria.
then
    echo "Sim, as duas condições são verdadeiras."
else
    echo "Não, pelo menos uma condição está errada."
fi
```

– A estrutura while

O "while" é uma das estruturas de repetição utilizadas em *shell scripts*. Abaixo, a sintaxe de uso:

```
while
    TESTE
do
    comandos
done
```

Exemplos:

Exemplo 01: Imprime a frase "Isto é um teste" sete vezes.

```
#!/bin/bash
#Inicializa a variável "var" com valor 0 (zero).
var="0"
while
#Testa se o valor de "var" é menor do que 7.
[ $var -lt 7 ]
do
    echo "Isto é um teste"
```

```

    #Incrementa o valor da variável "var"
    var=$(( $var + 1 ))
done

```

– **A estrutura for**

É uma outra estrutura de repetição, assim como "while". A seguir, a sintaxe de utilização:

```

for VARIÁVEL in LISTA
do
    comandos
done

```

ou, podemos utilizar o formato ANSI C/C++:

```

for
((EXPRESSÃO1;EXPRESSÃO2;EXPRESSÃO3))

```

Exemplos:

Exemplo 01: Imprime a frase "Isto é um teste" sete vezes.

```

#!/bin/bash
#Inicializa a variável "var" com valor 0 (zero).
var="0"
#Usa o comando "seq" para gerar uma sequência de 7 repetições.
for var in $(seq 7)
do
    echo "Isto é um teste"
done

```

– **A estrutura case**

É uma estrutura de decisão, ainda mais sofisticada que "if". A seguir, a sintaxe de utilização:

```

case VARIÁVEL in
    VALOR1)
        comandos;;
    VALOR2)
        comandos;;
    VALOR3)
        comandos;;
    *)
        comandos;;
esac

```

Exemplos:

Exemplo 01: Detecta o número digitado pelo usuário ou retorna uma mensagem de erro.

```

#!/bin/bash
echo "Digite um número entre 0 e 2:"
#Lê a variável "numero" a partir da entrada do usuário.
read numero
#Determina o valor da variável através da estrutura de controle.
case $numero in
    0)
        echo "0 numero zero foi digitado.";;
    1)
        echo "0 numero um foi digitado.";;
    2)
        echo "0 numero dois foi digitado.";;
    *)
        #Caso o valor não corresponda a nenhuma das opções:
        echo "Você digitou um caractere inválido.";;

```


esac

12.2.7. Funções

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

```
nome_da_funcao() {
    comandos
}
```

O corpo das funções devem construídos sempre antes de serem utilizados, caso contrário, o sistema retornará um erro do tipo "command not found", que significa que a função em questão não pode ser encontrada.

Exemplo:

```
#!/bin/bash
#Corpo da função "mostra_data".
mostra_data(){
    date
}
#Corpo da função "mostra_calendario".
mostra_calendario(){
    cal
}
echo "Digite 1 para exibir a data, ou 2, para o calendário:"
#Lê a variável "numero" a partir da entrada do usuário.
read numero
#Determina o valor da variável através da estrutura de controle.
case $numero in
    1)
        #Chama a função "mostra_data".
        mostra_data;;
    2)
        #Chama a função "mostra_calendario".
        mostra_calendario;;
    *)
        #Caso o valor não corresponda a nenhuma das opções:
        echo "Você digitou uma opção inválida.";;
esac
```

12.2.8. sed

O utilitário "sed" é um editor de *streams*. Um programa desta natureza, é usado para realizar transformações num arquivo de texto ou num *pipe* (fluxo de dados). Sed é bastante útil para fazer trocas de letras, palavras ou linhas inteiras de texto. É uma ferramenta largamente utilizada em *scripts shell*.

Embora a principio, o uso deste utilitário pareça simples, ele é um dos programas mais poderosos do mundo Linux. É possível realizar operações que demandariam um grande esforço manual e um trabalho de edição de dados muito complexo, através de um *script sed*.

```
sed [opções] [script] [arquivo/pipe]
```

Onde:
script

É um conjunto de comandos do sed e dados, que definem o que será alterado no arquivo definido pelo argumento que denominamos "arquivo".

arquivo/pipe

É a fonte dos dados que desejamos modificar.

opções:

-e [script]

Opção utilizada para executar um *script* (comandos e dados) sed, presente na própria cadeia de execução do comando (*in line script*).

-f [arquivo]

Permite especificar um arquivo que contenha um *script* (comandos e dados) sed. Esta é uma opção interessante, quando é necessário executar muitas operações do sed sobre um mesmo arquivo.

-i

Utiliza o próprio arquivo fonte de dados para gravar as alterações.

-n

Modo silencioso, impede que os resultados dos comandos do sed sejam impressos na saída padrão (monitor). Por padrão, caso não seja utilizado o parâmetro “-i”, sed imprimirá o resultado da execução de seus comandos para saída padrão. Caso “-n” seja utilizado em conjunto com “-e”, somente a saída decorrente do *script* a ser executado pelo segundo parâmetro será exibida na saída padrão.

Comandos Internos

Há ainda, comandos internos do sed que são usados para desempenhar as diversas funções com relação a edição de *streams*. Estes comandos são utilizados para compor os *scripts* sed. Abaixo uma breve descrição sobre os principais comandos que o utilitário possui:

q

Permite que o usuário informe em qual linha do processamento da fonte dos dados (arquivo ou *pipe*) o sed deve encerrar sua execução. Este parâmetro deve ser precedido de um número indicando a linha em que o processamento será encerrado.

d

Exclui um padrão (expressão).

I

Torna a busca pelo padrão independente de maiúsculas e minúsculas.

p

Imprime uma linha. Pode ser precedido por um número indicando o número da linha que se deseja imprimir na saída padrão (monitor).

n

Caso o parâmetro de linha de comando “-n” não tenha sido utilizado, imprime o conteúdo da fonte de dados (arquivo ou *pipe*).

s

Comando de substituição. Faz com que um determinado padrão (expressão) seja substituído por outro. Trabalha de acordo com o formato, incluindo as aspas:

“s/string original/string de substituição”

É um dos comandos sed mais utilizados. Este comando, possui uma lista de parâmetros e sinalizadores, abaixo demonstramos alguns dos mais importantes:

\L

Torna a *string* (cadeia de caracteres) minúsculo até que “\U” ou “\E” seja encontrado na sequência.

\l

Torna o próximo caractere da sequência minúsculo.

\U

Torna a *string* (cadeia de caracteres) maiúscula até que “\L” ou “\E” seja encontrado na sequência.

\u

Torna o próximo caractere da sequência maiúsculo.

\E

Para a conversão de *strings* (cadeia de caracteres) iniciada por “\L” ou “\U”.

\n

Insere uma nova linha na *string* (cadeia de caracteres).

\t

Insere um espaço de tabulação na *string* (cadeia de caracteres).

\

Caractere de escape. É necessário quando desejamos inserir algum caractere especial, como “/” (barra) por exemplo, ou quando necessitamos usar um parâmetro, como “\n” (nova linha) exemplificando, dentro de uma *string* (cadeia de caracteres).

g

Substituí todas as ocorrências do padrão encontrado na fonte de dados por uma nova expressão.

i

Ignora maiúsculas e minúsculas na busca pelo padrão (expressão) a ser substituído.

[número]

Um número que indica qual das ocorrências encontradas, e que confere com o padrão informado, será substituída. O escopo de atuação deste sinalizador é uma linha.

p

Imprime linhas de uma fonte de dados (arquivo ou *pipe*) que estão sendo substituídas.

Exemplos:

sed “n” teste.txt

Imprime o conteúdo do arquivo “teste.txt” na saída padrão (monitor).

sed -n “2,4p” teste.txt

Imprime da segunda à quarta linha do arquivo “teste.txt” na saída padrão (monitor).

sed “s/zé/marcus/” teste.txt

Substituí a primeira ocorrência da string “zé” de cada linha por “marcius”, presente no arquivo “teste.txt”.

sed -e “2q” -e “s/zé/marcus/” teste.txt

Processa o arquivo “teste.txt” até o início da segunda linha, substituindo a primeira ocorrência da string “zé” por “marcius”.

sed “s/zé/marcus/2” teste.txt

Substituí a segunda ocorrência da string “zé” de cada linha, pela string “marcius”, presente no arquivo “teste.txt”.

sed “/^\$/d” teste.txt

Apaga as linhas em branco do arquivo “teste.txt”.

sed “/zé/d” teste.txt

Apaga as linhas que contenham a string “zé” do arquivo “teste.txt”.

sed -i “s/samba/rede/” teste.txt

Substituí a primeira ocorrência da string “samba” de cada linha por “rede”, e grava esta alteração no arquivo “teste.txt”.

sed "/CPPC/ld" teste.txt > teste_modificado.txt

Apaga todas as linhas que contenham a string "CPPC", independentemente de maiúsculas e minúsculas, gravando o resultado no arquivo "teste_modificado.txt".

sed -nf sed.scp teste.txt

```
### Arquivo "sed.scp" ###  
#!/usr/bin/sed -nf  
s/palavra/frase completa/p  
3,5p  
### Fim do script ###
```

Exemplo de uso do sed por meio de um arquivo de script. O script acima, faz a substituição da string "palavra" por "frase completa", caso esta substituição ocorra em algum lugar do arquivo alvo, a linha em questão será exibida na tela. O ultimo comando (3,5p) tem a função de imprimir da terceira a quinta linha da fonte de dados. Todas as linhas que começam com o caractere "#", são comentários.

Para mais informações sobre sed, consulte:

<http://www.gnu.org/software/sed/manual/sed.html>

<http://www.student.northpark.edu/pemente/sed/sed1line.txt>

<http://www.zago.eti.br/script/sed.html>