

- LPI 102 -

# Administração de Redes Linux

Há mais de 17 anos, a **GREEN Treinamento** vem atuando exclusivamente com treinamento e possui toda a infra-estrutura para oferecer-lhe a melhor experiência no aprendizado de informática.

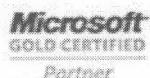
Já treinamos mais de **130.000** alunos de todas as procedências: grandes empresas, particulares, microempresários, estudantes, etc.

Somos um Centro de Treinamento Oficial certificado pelos três principais fabricantes da indústria de softwares: nós somos **CPLS da Microsoft (antigo CTEC)**, **ECIS da IBM** e **Training Provider da Mandriva Conectiva**, ou seja, atendemos aos padrões internacionais de qualidade destas empresas, contando com profissionais certificados e informações técnicas e comerciais sempre atualizadas.

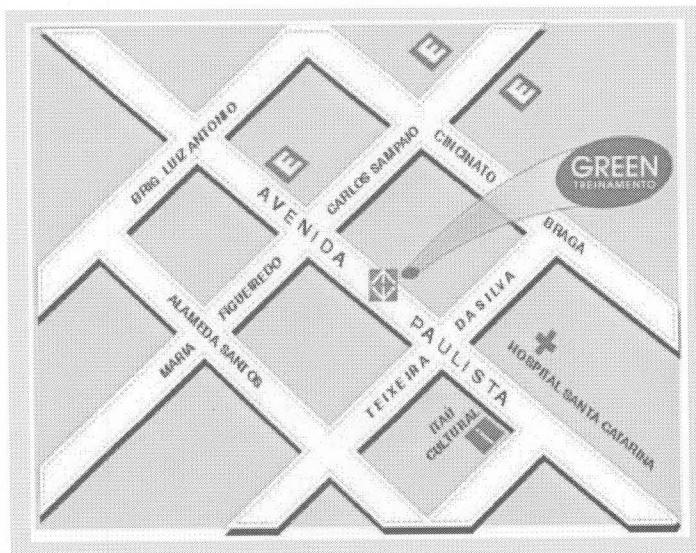
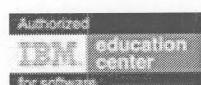
Somos **Centro de Testes da Pearson VUE – Virtual University Enterprises**. Podemos aplicar testes para diversas certificações, entre as quais: Novell, Ericsson, Microsoft, Lucent, LPI (Linux), Informix, CompTIA, Certified Internet Webmaster, Lotus, Avaya, Allaire, Siebel, BROCADE, Cisco, RSA Security, NASD, Check Point, Macromedia, GNU e Intershop Communications.

Possuímos modernos laboratórios equipados com computadores de última geração. Existem também unidades móveis onde instalamos os equipamentos no cliente para ministrar cursos "IN-HOUSE".

**"Oferecer a melhor capacitação profissional é o nosso objetivo"**



Learning Solutions



## **GREEN TREINAMENTO**

**Informações: (11) 3253-5299**

Av. Paulista, 326 - 12º Andar

Bela Vista - São Paulo - SP

Metrô Brigadeiro

(Saída Carlos Sampaio)

<http://www.green.com.br>

[cursos@green.com.br](mailto:cursos@green.com.br)

### **Estacionamentos Conveniados**

Desconto válido somente com carimbo da recepção no ticket do estacionamento.

#### Créditos

Marcos Sungaila  
autor

Copyright Marcos Sungaila

**TODOS OS DIREITOS RESERVADOS.** É proibida a reprodução parcial ou total desta obra, por qualquer meio, seja este gráfico, fotográfico ou eletrônico entre outros, bem como a inclusão deste trabalho em qualquer sistema de arquivo de processamento de dados (disquetes, cd's, dvd's, etc), sem prévia autorização por escrito de Marcos Sungaila. Tais vedações se aplicam também à supressão de informações através da alteração das características originais deste trabalho, incluindo alterações de diagramação e de características gráficas da obra.

A violação de direitos autorais é crime, nos termos da lei 9.610/98, e conforme estabelecido pelo artigo 184 do Código Penal. Aplica-se a esta obra e em relação aos nomes empresariais, marcas mistas, figurativas ou nominativas contidas na mesma, as disposições legais da lei 9.279/96.

A reprodução deste material sem autorização escrita de Marcos Sungaila é crime, de acordo com a legislação em vigor. Este material não pode ser fotocopiado.

Se você verificar que esta obra foi fotocopiada ou reproduzida de qualquer outra forma, entre em contato com o autor pelo e-mail [marcos@savant.com.br](mailto:marcos@savant.com.br).

Copyright 2006, Marcos Sungaila

# Índice

Kernel.....	4
Obtendo informações do kernel com uname.....	4
Gerenciando módulos do kernel.....	4
lsmod – listando os módulos carregados.....	5
modinfo – informações de módulos.....	5
insmod – carregando um módulo.....	6
rmmod – descarregando módulos da memória.....	6
modprobe – resolvendo dependências na carga de módulos.....	7
depmod – lista de dependências entre módulos.....	8
Carregamento automático de módulos.....	8
Compilação e personalização do kernel.....	8
Configurando o kernel para compilação.....	9
Compilando e instalando o kernel manualmente.....	11
Compilando e instalando o kernel no estilo Debian.....	11
Inicialização do Linux.....	13
Iniciando o sistema.....	13
Configurando o LILO.....	13
Configurando o GRUB.....	15
Níveis de execução do Linux.....	18
Nível de execução padrão .....	18
Alterando o nível de execução.....	18
Mono-usuário.....	19
Sistemas de Impressão.....	20
Configuração do servidor de impressão.....	20
Filas de impressão.....	22
Configuração de impressoras locais e remotas.....	22
Status do servidor CUPS – Ipstat.....	24
Status de impressoras – Ipc.....	24
Enviando arquivos para impressão e gerenciando filas.....	25
Documentação.....	26
Encontrando documentação no Linux.....	26
man pages.....	26
O comando whatis.....	27
O comando apropos.....	27
O comando man.....	27
Texinfo.....	28
Documentação Linux na Internet.....	29
Notificação e alertas a usuários.....	29
Shell.....	30
Ambiente do usuário.....	30
Variáveis.....	30
Variáveis de ambiente.....	31
Scripts executados no login do usuário.....	32
Variáveis pré-definidas no ambiente do usuário.....	32
Shell scripts.....	33

---

Estrutura de um script.....	33
Shell's de execução.....	34
Recebendo parâmetros pela linha de comando.....	35
Construção condicional.....	35
Estrutura de repetição – laços.....	37
while.....	37
for.....	37
until.....	38
Usando funções em scripts.....	38
Referências.....	38
Tarefas administrativas.....	39
Logs do sistema.....	39
Configurações adicionais.....	41
Rotacionando os logs automaticamente – logrotate.....	41
Agendamento de tarefas.....	43
at – Evento único.....	43
cron – Ocorrência periódica.....	44
Backup.....	46
O comando tar.....	46
O comando dd.....	47
O comando cpio.....	48
O comando dump.....	49
O comando restore.....	50
Gerenciamento de hora no Linux.....	51
Definindo sua localização.....	51
Instalando os pacotes necessários.....	52
Sincronizando o horário.....	53
Referências.....	54

## Kernel

O termo Linux é, na verdade, a identificação do kernel em execução em seu equipamento. É ele que permite o uso de seu hardware e funciona como uma interface entre o sistema operacional, as aplicações e o equipamento.

Há várias ferramentas que permitem o gerenciamento do kernel em tempo real e veremos como elas funcionam.

### Obtendo informações do kernel com `uname`

Este comando exibe informações sobre o kernel em execução, além de dados de configuração do equipamento como hostname, entre outros.

uname [opção]	
OPÇÃO	Descrição
-s	Exibe o nome do kernel em execução
-n	Exibe o hostname
-r	Exibe a versão e revisão do kernel, ex: 2.6.8-2-386
-v	Exibe a data de liberação do kernel
-m	Exibe a arquitetura da cpu do equipamento
-o	Exibe o nome do sistema operacional
-a	Exibe todas as informações acima

### Gerenciando módulos do kernel

O kernel é o responsável por gerenciar vários subsistemas do nosso equipamento, além do hardware e periféricos. Para poder oferecer compatibilidade com estes hardwares é necessário poder ativá-los e se comunicar com eles em seu formato nativo. Estes procedimentos são realizados pelos módulos (drivers).

Em um ambiente Linux os módulos que ativam os diversos periféricos podem ser carregados dinamicamente. Isto dá ao Linux um dinamismo e praticidade ímpar permitindo que um periférico somente seja acionado quando necessário e ainda não exigindo reinicializações constantes do equipamento.

O processo de gerenciamento de módulos no Linux é realizado automaticamente pelo kernel, contudo pode ser necessário carregar algum módulo manual, quer seja para testes, quer para solucionar algum problema. Estes procedimentos, quando necessários, somente podem ser executados com privilégios de root.

## Ismod – listando os módulos carregados

O comando lsmod exibe o status dos módulos ativos em memória e a dependência entre eles além do espaço em memória utilizado. Uma saída do comando lsmod deve ser parecida com esta, embora varie de equipamento para equipamento.

```
# lsmod
Module           Size  Used by
nls_cp437        6016  0
isoofs          33976  0
ipv6            229892 10
af_packet       20872  2
parport_pc      33348  0
parport         37320  1 parport_pc
pcspkr          3816  0
snd_ens1371     23012  0
snd_rawmidi     23204  1 snd_ens1371
snd_seq_device   7944  1 snd_rawmidi
snd_pcm_oss     48168  0
snd_mixer_oss    16640  1 snd_pcm_oss
snd_pcm          85384  2 snd_ens1371,snd_pcm_oss
snd_page_alloc   11144  1 snd_pcm
snd_timer        23300  1 snd_pcm
...
...
```

## modinfo – informações de módulos

Este comando exibe informações sobre a aplicação de um módulo, seu autor, licença e localização. Pode ainda exibir informações sobre parâmetros suportados e barramento.

modinfo [opções] módulo arquivo	
OPÇÃO	DESCRIÇÃO
-a	Exibe informações sobre o autor do módulo
-d	Exibe a descrição do módulo
-l	Exibe a licença de uso de módulo
-p	Exibe os parâmetros suportados pelo módulo
-n	Exibe o nome do arquivo que carrega o módulo

Sua utilização é simples. Quando executado sem parâmetros, exibe todas as informações disponíveis.

Se o recurso ativado por este módulo não estiver em uso, o módulo será descarregado. Caso haja alguma dependência, sua desativação só será possível se todos os módulos dependentes deste forem removidos também. O comando rmmod não realiza esta tarefa, ficando a cargo do comando modprobe a solução de dependências de carregamento ou descarregamento.

## **modprobe - resolvendo dependências na carga de módulos**

Assim como em vários tipos de softwares, alguns drivers possuem dependências que devem ser resolvidas para que a ativação de um módulo seja possível. Um exemplo disto é o módulo que oferece suporte a sistemas de arquivos fat32. Este módulo ativa o suporte a nomes longos e à estrutura de diretórios específica deste file system porém as informações básicas de acesso ao sistema fat não está incorporado neste módulo. Estes detalhes fazem parte do módulo fat. Seguindo este raciocínio podemos pressupor que, para utilizarmos os recursos de fat32 será necessário carregar os módulos fat e vfat que fornece suporte a fat32 no Linux. Veja uma carga direta do módulo vfat pelo comando insmod:

```
# insmod `modinfo -n vfat`  
insmod: error inserting '/lib/modules/2.6.8-2-386/kernel/fs/vfat/vfat.ko': -1  
Unknown symbol in module
```

Realizando a carga pelo comando modprobe:

```
# modprobe vfat
```

Da mesma forma que na carga, o modprobe pode remover módulos da memória descarregando também recursos que ficaram sem uso após a desativação dos módulos dependentes:

```
# modprobe -r vfat
```

É importante saber que a verificação das dependências é uma tarefa realizada pelo kernel e que possíveis falhas na carga de um módulo poderá vir acompanhada de uma mensagem de erro do kernel.

Caso estas falhas ocorram durante a inicialização do equipamento, estas mensagens podem ser vistas com o comando dmesg.

Podemos ter uma listagem dos módulos disponíveis com seu kernel pelo comando abaixo:

```
# modprobe -l | less
```

## **depmod – lista de dependências entre módulos**

Todos os módulos disponíveis no Linux para a ativação de periféricos vêm embutido no kernel como software GPL ou fornecidos por fabricantes como Nvidia, ATI e outros.

Para que comandos como modprobe possam carregar os módulos necessários ao funcionamento de um hardware de forma automática, resolvendo suas dependências foi criada uma lista de módulos e recursos necessários para sua ativação além de mapas de vários subsistemas de hardware como o barramento pci e usb. Esta lista está no arquivo /lib/modules/versão-kernel/modules.dep. Sua construção ou atualização é feita com o comando depmod que verifica os subdiretórios de módulos pesquisando por informações de dependências. Na listagem abaixo podemos ver estes arquivos.

```
# ls -l /lib/modules/`uname -r`/mod*
-rw-r--r-- 1 root root 138720 2006-09-10 14:23 modules.alias
-rw-r--r-- 1 root root 69 2006-09-10 14:23 modules.ccwmap
-rw-r--r-- 1 root root 235506 2006-09-10 14:23 modules.dep
-rw-r--r-- 1 root root 517 2006-09-10 14:23 modules.ieee1394map
-rw-r--r-- 1 root root 1061 2006-09-10 14:23 modules.inputmap
-rw-r--r-- 1 root root 16427 2006-09-10 14:23 modules.isapnpmap
-rw-r--r-- 1 root root 131958 2006-09-10 14:23 modules.pcimap
-rw-r--r-- 1 root root 104676 2006-09-10 14:23 modules.symbols
-rw-r--r-- 1 root root 157116 2006-09-10 14:23 modules.usbmap
```

## **Carregamento automático de módulos**

O funcionamento do comando modprobe pode ser personalizado por arquivos como /etc/modules.conf. É através deste arquivo que o Linux sabe como ativar o seu hardware durante o boot da máquina. Neste arquivo podem ser inseridos comandos para execução antes da carga de um módulo (pre-install) ou após a sua ativação (post-install) bem como a identificação de módulos específicos para hardwares identificados de forma genérica como na listagem abaixo identificando que a controladora scsi do equipamento é uma Adaptec® com chipset 79xx ou que a interface de rede utiliza um chipset RealTek ou ativando periféricos com opções extras como no exemplo da placa ne2000:

### **trecho do arquivo /etc/modules.conf**

```
alias scsi_hostadapter aic7xxx
alias eth0 8139too
alias eth1 ne2k-pci
options io=0x280 irq=11
```

No Debian o gerenciamento deste arquivo é feito com os utilitários do pacote module-init-tools e com o sistema de detecção de hardware discover que mantém uma lista de especificações de hardware em /usr/sharediscover.

## **Compilação e personalização do kernel**

O processo de personalização do kernel exige, via de regra, uma atenção redobrada em

relação aos recursos que serão ativados ou desativados. Normalmente este procedimento é realizado quando desejamos incluir um novo recurso ou desativar algo que está com uma falha de segurança e que sua correção só estará disponível mais tarde.

Para o procedimento de compilação do kernel é necessário instalar os seguintes pacotes:

- gcc
  - binutils
  - makemodule-init-tools
  - awk
  - gzip
  - shellutils
  - grep
  - bin86
  - bzip2
  - libc6-dev  
libes-dev 3.0 DEBIAN
  - kernel-package
  - libncurses5-dev
- + RECURSOS*

A obtenção dos fontes do kernel pode ser feita a partir do site oficial do projeto ([www.kernel.org](http://www.kernel.org)) ou a partir da instalação do pacote fonte do próprio Debian:

- kernel-source-2.6.8 (kernel estável para o sarge)

Nossa primeira etapa é instalar os pacotes listados acima. Vamos utilizar os fontes do próprio Debian que já conta com vários patches ampliando seus recursos e melhorando a integração do sistema como um todo.

```
# apt-get install bin86 bzip2 gcc libc6-dev kernel-package kernel-source-2.6.8
ncurses-dev
```

Após os pacotes serem instalados é necessário expandir os fontes do kernel que são mantidos em formato compactado no diretório `/usr/src`, criar um link simbólico do novo diretório criado com o nome de `linux` e acessá-lo para iniciar a configuração:

```
# cd /usr/src
# tar -xvf kernel-source-2.6.8.tar.bz2
# ln -s kernel-source-2.6.8 linux
# cd linux
```

## Configurando o kernel para compilação

Para que a compilação seja possível é necessário definir os parâmetros de funcionamento do kernel. Isto é feito através de uma interface específica que pode funcionar por linha de comando de forma interativa (`config`), por uma interface texto guiada por menus (`menuconfig`) ou por uma interface gráfica comum (`xconfig`) ou uma desenvolvida pela equipe do Gnome (`gconfig`). Para prosseguirmos com esta etapa podemos começar do zero, com os fontes como estão ou podemos copiar as configurações do kernel atual e apenas modificarmos as que desejarmos. Vamos copiar estas configurações que nos servirão de guia:

```
# cp /boot/config-`uname -r` .config
```

Vamos criar, ainda, uma identificação diferente para que o novo kernel não sobrescreva os arquivos e módulos do kernel atual. Edite o arquivo `Makefile` e altere o item `EXTRAVERSION`.

```
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 8
EXTRAVERSION = -kernelteste
```

Agora vamos acessar a interface texto pela sua praticidade.

```
# make menuconfig
```

Nesta etapa serão definidos os parâmetros de funcionamento do seu novo kernel. Você pode, por exemplo, desativar suporte a periféricos ISA ou todo o subsistema de som. Incluir recursos adicionais ou remover suporte a hardware que você não tem (como cartões pcmcia caso você tenha um desktop ou servidor sem este adaptador). Uma lista dos componentes mais comuns que você deve checar inclui:

- Tipo de processador do seu equipamento (AMD, Pentium, P4, etc)
- Symmetric Multi-Processing; "No" se você possui uma máquina mono-processada ou sem os recursos HT
- Loadable module support; "Yes" em todos os itens
- Parallel port support; "Yes" se você for utilizar impressoras paralelas ligadas diretamente ao seu equipamento, caso contrário marque como "No".
- Plug-N-Play; "Yes"; ISA P-N-P apenas se você possuir placas ISA em seu equipamento.
- Character devices; ative Direct Rendering se sua placa de vídeo estiver listada, e habilite agpgart se o chipset de sua placa estiver listado; mice se você tiver um mouse tipo PS/2
- Sound – localize o módulo de sua placa e ative apenas este
- Input core support – alguns dispositivos USB utilizam este recurso
- USB support – se você quiser usar USB ative-o
- Provavelmente você não precisará mexer nos seguintes itens:
  - ◆ Code maturity level options
  - ◆ General Setup
  - ◆ Memory Technology
  - ◆ Block devices
  - ◆ Multi-device support
  - ◆ Networking options

- ◆ Telephony support
- ◆ ATA/IDE/MFM/RLL support
- ◆ Fusion MPT device support
- ◆ I20 device support
- ◆ Amateur Radio support
- ◆ Infrared support
- ◆ ISDN subsystem
- ◆ Old CD-ROM drivers – a menos que você não possua driver compatível com ATAPI
- ◆ Multimedia devices – para compatibilidade com placas de tv e outros
- ◆ File Systems – oferece compatibilidade com vários tipos de sistemas de arquivos como fat, fat32, ntfs e outros
- ◆ Console Drivers – habilita diferentes modos gráficos na console
- ◆ kernel hacking

## **Compilando e instalando o kernel manualmente**

O processo de compilação propriamente dito, começa ao executarmos os seguintes comandos e instala o novo kernel em nosso equipamento.

```
# make clean  
# make all  
# make modules_install
```

Esta etapa levará um bom tempo. Em seguida será necessário copiar o kernel de inicialização e criar o arquivo initrd complementar.

```
# cp arch/i386/bzImage /boot/vmlinuz-2.6.8-kernelteste  
# cd /boot  
# mkintrd initrd-2.6.8-kernelteste.img 2.6.8-kernelteste
```

Estes dois arquivos são os necessários para que a inicialização do Linux possa acontecer. Agora devemos alterar os arquivos do gerenciador de boot utilizado.

## **Compilando e instalando o kernel no estilo Debian**

O Debian fornece uma ferramenta para construção de pacotes Debian do kernel a partir dos fontes. Os procedimentos acima tornam-se bem mais simples pois ao finalizar a compilação do kernel basta instalar o pacote .deb criado.

A compilação e empacotamento do kernel pode ser feita como abaixo.

```
# make-kpkg --append-to-version "-kernelteste" --initrd --us --uc kernel_image
```

Este comando vai compilar nosso kernel personalizado, criando um arquivo .deb com uma extensão adicional “-kernelteste”. Para que isto funcione não é necessário alterar o arquivo Makefile ou outro qualquer.

A opção “--initrd” instrui que, ao instalar o pacote .deb o arquivo initrd seja automaticamente criado. Este pacote .deb ainda criará automaticamente as opções no gerenciador de boot em uso atualmente.

As opções “--us --uc” fazem com que a construção do pacote seja simples, sem assinatura gpg ou criação de log de modificações.

Com o pacote pronto, só é necessário instalá-lo.

```
# cd /usr/src
# dpkg -i kernel-image-2.6.8-kernelteste_10.00.Custom_i386.deb
```

Agora é só reiniciar seu equipamento e testar o novo kernel.

## Inicialização do Linux

Para a administração de equipamentos rodando Linux você deve ser capaz de identificar as etapas de inicialização bem como o desligamento do sistema operacional. É necessário, ainda, poder alterar o seu funcionamento, quer pelos seus arquivos de configuração, quer pela linha de comando durante o boot da máquina.

### Iniciando o sistema

O processo de inicialização de um equipamento começa ao ligar o equipamento e só termina quando o prompt de login é exibido. As seguintes etapas ilustram este processo:

1. Ao ligar o equipamento são realizados os testes iniciais definidos na BIOS. Estes testes são conhecidos como P.O.S.T. (Power On Self Tests)
2. Ao término destes testes, a BIOS passa a ler a MBR (primeiros 512 bytes do disco de boot) em busca de programas especiais que possam realizar tarefas adicionais para o boot do equipamento. Estes programas são chamados de "Boot Loaders" (Gerenciadores de boot).
3. As definições lidas indicam onde estão os arquivos de inicialização: kernel inicial (vmlinuz) e drivers adicionais (initrd).
4. Ocorre a carga destes arquivos e a sua execução para iniciar o Linux em seu equipamento.

### Configurando o LILO

A configuração do LILO é feita pelo arquivo `/etc/lilo.conf` e deve ser semelhante à listagem abaixo padrão de uma instalação Debian:

```

# Onde o lilo deve gravar as informações de boot, neste caso na MBR
boot=/dev/hda
# Qual a partição / de sua instalação
root=/dev/hda1
# Arquivo indica ao boot loader onde está gravado o kernel do Linux
map=/boot/map
# Tempo de espera para seleção da opção de inicialização
delay=20 #CIMO DE SEGUNDOS = 2 SEG
# Tela texto padrão (80x25)
vga=normal
# Força a exibição do menu de boot do Lilo
prompt
# Qual imagem deve ser inicializada
default=Linux
# Arquivo com informações exibidas no menu de inicialização
message=/boot/message.txt
# Senha para permitir alterar as opções de boot ou nível de execução
password=debian
# ATENÇÃO: senha em texto puro
# Primeira imagem de boot
image=/vmlinuz
    label=Linux
    read-only
    initrd=/initrd.img

# segunda imagem de boot
image=/vmlinuz.old
    label=LinuxOLD
    read-only
    optional
    initrd=/initrd.img.old

```

Este arquivo está dividido em duas seções:

- configurações gerais com informações sobre o dispositivo de boot e tempo de espera
- configurações da imagem de boot definidas com a diretiva image

O LiLo permite que seu menu seja omitido, suprimindo o parâmetro "prompt" em seu arquivo de configuração. Neste caso, para vermos as opções de boot pré-definidas é necessário pressionar a tecla <shift> da esquerda para que o menu seja exibido.

Para incluirmos o nosso kernel recém compilado, devemos acrescentar uma seção image como segue:

```

image=/boot/vmlinuz-2.6.8-kernelteste
    label=Kernelteste
    read-only
    initrd=/boot/initrd-2.6.8-kernelteste.img

```

Para ativarmos estas configurações rodamos o comando lilo:

```

# lilo
Added Linux*
Added LinuxOLD
Added Kernelteste

```

```

# Imagem de boot a ser carregada
default          0
# Tempo de espera exibindo o menu em tela em seg
timeout         5
# cores para apresentação do menu
color cyan/blue white/blue
# imagem de fundo para a exibição do menu
splashimage=(hd0,2)/boot/grub/debian.xpm.gz
# senha para poder alterar as opções de boot durante a exibição do menu
password --md5 $1$H8LLM1$cI0Lfs5.C06xFJYPQ8Ixz/
em Seg C

# Primeira imagem de boot
title      Debian GNU/Linux, kernel 2.6.8-2-386
root       (hd0,0) → ONDE ESTÁ O KERNEL
kernel    /boot/vmlinuz-2.6.8-2-386 root=/dev/sda1 ro
initrd   /boot/initrd.img-2.6.8-2-386
savedefault
boot

# Segunda imagem de boot
title      Debian GNU/Linux, kernel 2.6.8-2-386 (recovery mode)
root       (hd0,0)
kernel    /boot/vmlinuz-2.6.8-2-386 root=/dev/sda1 ro single
initrd   /boot/initrd.img-2.6.8-2-386
savedefault
boot
PARTIÇÃO RAIZ

```

Enquanto o LILO é um programa estático gravado na MBR com suas opções pré-definidas, o GRUB é um shell interativo que permite inspecionar o disco em tempo real, durante a inicialização do equipamento e, caso necessário, alterar suas opções de boot. Com o GRUB você pode alterar as opções de inicialização, carregar outros arquivos de configuração ou mesmo executar um comando cat em um arquivo. Durante a exibição do menu de inicialização você pode pressionar "e" para editar uma opção de boot, "p" para digitar a senha que dará acesso a estas opções, "c" para entrar no modo de comando do Grub ou "b" para iniciar um boot com a opção selecionada. A tecla ESC retorna ao menu anterior.

O comando grub emula um shell onde é possível testar suas opções de inicialização sem a necessidade de reiniciar seu equipamento. É possível consultar os comandos internos do grub com help ou obter detalhes adicionais com help comando.

A listagem abaixo mostra este funcionamento.

*Grub> help* → mostra todos os comandos disponíveis no shell.

*Grub> help boot* → mostra o comando boot, que é o que cuida de todo o resto da inicialização do Linux. Ele é o primeiro programa executado, o qual recebe o sig. 1 e permanecerá ativo até o final da inicialização do computador. Ele será o responsável pela ativação de todos os recursos de hardware e software definidos nas configurações do Linux.

*Grub> help dmsetup* → mostra os detalhes sobre o comando dmsetup.

## Níveis de execução do Linux

Níveis de execução ou RUNLEVELS como são chamados definem quais recursos e serviços devem estar disponíveis para uso. De forma geral todo Linux permite, ao menos 3 runlevels básicos que são o 0 (zero), 1 e 6, sendo que em funcionamento normal são definidos 7 níveis. A tabela a seguir mostra estes níveis.

RUNLEVEL	DESCRIÇÃO
0	Sistema desligado (halt)
1	Mono usuário, utilizado para manutenção do Linux
2	Multi-usuário sem sistemas de arquivos em rede <small>MAPAMENTOS DESATIVADOS</small>
3	Multi-usuário completo (serviços locais e remotos)
4	Reservado para uso futuro (sem definição atual)
5	Multi-usuário completo com X Window System ativado
6	Reinicialização do sistema (reboot)

Distribuições como Slackware utilizam o nível 4 como multi-usuário com X Window ativado. O Debian utiliza o nível 2 como único para todas as opções de multi-usuário. Para mais detalhes sobre como sua distribuição funciona consulte sua documentação.

## Nível de execução padrão

As distribuições Linux utilizam o arquivo `/etc/inittab` como base para definir quais recursos devem ser ativados em cada nível, bem como para definir qual será o nível default de execução. A diretiva `id` define este nível. Em nossa instalação Debian ela está assim:

```
# grep "^id" /etc/inittab
id:2:initdefault:
```

## Alterando o nível de execução

Para alterar de forma permanente o nível de execução que o seu equipamento inicializa você deve alterar o arquivo `/etc/inittab`. Para alterações momentâneas como uma manutenção ou teste você pode executar o comando `init` ou `telinit`.

Alterando para nível 5 e verificando o runlevel:

```
# init 5
# runlevel
```

Entrando em nível 3:

```
# telinit 3
```

## Mono-usuário

Em contraste a outros sistemas operacionais como Windows® ou DOS®, o Linux é, por definição, um sistema operacional multi-usuário que permite o uso do equipamento por várias pessoas simultaneamente, embora às vezes isto possa causar alguns entraves como quando necessita acrescentar um novo hardware ou realizar uma verificação de disco onde ninguém pode usar o sistema enquanto a checagem estiver sendo realizada.

Nestes casos a solução é o nível mono-usuário onde o Linux se comporta de forma análoga aos sistemas citados acima. Neste nível é possível verificar a integridade de sistemas de arquivos, destravar subsistemas como som ou de autenticação e efetuar a manutenção do seu Linux sem maiores preocupações.

A mudança para mono-usuário requer, em algumas distribuições, a senha do superusuário (apenas a senha do root é válida) para o comando `sulogin`.

Mudando para mono-usuário e consultando o nível atual:

```
# init 1  
# runlevel
```

Pode-se utilizar o nível “s” para mono-usuário, sendo que a diferença entre este e o nível 1 é que aqui não são ativadas partições além de “/” e “/boot” se existir.

## Sistemas de Impressão

Para termos controle sobre o sistema de impressão é necessário podermos gerenciar tanto impressoras quanto suas filas de impressão. Ativar ou suspender periféricos e todos os trabalhos envolvidos com determinado recurso além de resolver seus problemas.

No inicio da história da computação, a impressão era realizada por “*line printers*” (impressoras de linha) que imprimiam uma linha de texto por vez, utilizando caracteres de tamanho fixo com uma única fonte de impressão. Devido ao baixo desempenho destes periféricos os primeiros main-frames intercalavam os recursos para o acesso a periféricos como impressoras, leitores de cartões perfurados e perfuradores de cartões com outros trabalhos visando ampliar o desempenho do sistema. Este sistema foi chamado de “Simultaneous Peripheral Operation On Line” ou spooling simplesmente. Os primeiros sistemas de impressão para o Unix e Linux foram baseados no subsistema de impressão da Berkeley Software Distribution que era composto por um serviço chamado de `lpd` e um comando que enviava os trabalhos de impressão para as filas chamado `lpr`.

Atualmente o serviço de impressão no Linux é gerenciado pelo CUPS, tendo sido utilizado o servidor `lpr` e posteriormente o `LPRng` por muito tempo. O CUPS oferece muitas vantagens sobre seus antecessores como interface web de gerenciamento além de um maior número de drivers de impressoras.

### Configuração do servidor de impressão

Nosso Debian vem com o servidor `lpr` instalado, mas é extremamente simples a instalar o CUPS, sendo que durante este processo o `lpr` será automaticamente removido e sua inicialização é controlada pelo serviço `cupsys`. Abaixo instalamos o servidor cups e mais alguns pacotes com drivers adicionais. Caso você utilize impressoras HP do tipo inkjet, o pacote `hpijs` também deve ser instalado. Para a instalação do CUPS como um servidor de impressão, os pacotes abaixo são os necessários para seu funcionamento.

```
# aptitude install cupsys cupsys-bsd cupsys-client cups-pdf
```

Uma linha de comando instalando todos os pacotes disponíveis para o servidor CUPS e seus drivers inclui os pacotes abaixo (a \ no final linha indica que o comando continua na linha seguinte).

```
# aptitude install cups-pdf cupsys-bsd cupsys-client cupsys \
cupsys-driver-gimpprint-data cupsys-driver-gimpprint cupsys-pt escputil \
foomatic-db-engine foomatic-db-gimp-print foomatic-db-hpijs foomatic-db \
foomatic-filters foomatic-filters-ppds foomatic-gui hpijs hplip hpoj \
ijsgimpprint
```



Se houver necessidade de procurar por drivers para outras impressoras, acesse a página <http://www.cups.org/ppd.php> onde está disponível uma interface de pesquisa por modelos de impressoras. Após encontrar o seu driver, baixe-o e o grave em `/usr/share/cups/model`.

O servidor lpr mantinha a lista de impressoras disponíveis no arquivo /etc/printcap. No CUPS este arquivo é mantido por compatibilidade sendo que as impressoras são cadastradas no arquivo /etc/cupsd/printers.conf através de sua interface de gerenciamento. A configuração do servidor CUPS é bem completa sendo que a configuração padrão atende a redes de pequeno e médio portes.

Um dado inicial que deve ser alterado é a informação da rede à qual este servidor atenderá. Caso esta informação não seja alterada, o gerenciamento do servidor pela interface web somente poderá ser realizado a partir do localhost ou pela linha de comando. A configuração do servidor está listada abaixo.

**/etc/cups/cupsd.conf**

```
DefaultCharset notused
LogLevel info
Printcap /var/run/cups/printcap

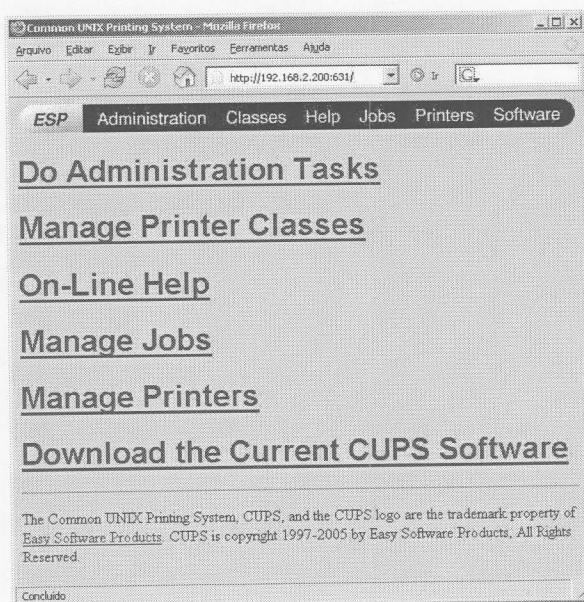
# porta de gerenciamento via web
Port 631

# acesso ao gerenciador somente pelo localhost
<Location />
    Order Deny,Allow
    Deny From All
    Allow From 127.0.0.1
    # Habilitando o acesso pela rede ao gerenciador
    Allow From 192.168.2.0/24
</Location>

<Location /jobs>
    AuthType Basic
    AuthClass User
</Location>

<Location /admin>
    AuthType Basic
    AuthClass System
    Order Deny,Allow
    Deny From All
    Allow From 127.0.0.1
    # Habilitando o acesso pela rede para configuração de impressoras
    Allow From 192.168.2.0/24
</Location>
```

Para podermos acessar o gerenciador do CUPS devemos utilizar um navegador web apontando para a porta 631.



## Filas de impressão

Uma fila de impressão é um recurso virtual para o qual os usuários envia seus serviços de impressão. É muito comum associarmos uma fila de impressão a uma impressora física contudo, o CUPS permite que trabalhos de impressão sejam direcionados para impressoras remotas através destas filas, ou mesmo para um pool de impressoras de mesmo tipo (classe).

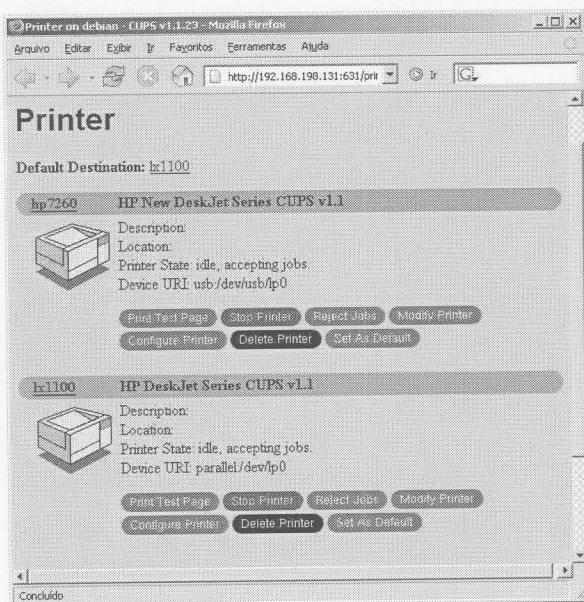
Vários comandos permitem a inspeção das filas de impressão, bem como permitem que um usuário manipule seus trabalhos de impressão. Apenas o root ou um usuário autorizado é que pode gerenciar o trabalho de outros usuários.

## Configuração de impressoras locais e remotas

O processo de criação de uma fila de impressão pode ser realizado pela interface web, pela linha de comando ou por uma aplicação gráfica como o aplicativo printmgr. Pelo navegador acesse a porta 631 de seu equipamento e clique em “Manage Printers” e em seguida em “Add printer”. As etapas de criação de uma fila de impressão são as seguintes:

- Name: nome da fila de impressão, não deve conter espaços;
- Location: localização física da impressora (campo meramente descritivo)
- Description: descrição da impressora (campo meramente descritivo)
- Device: tipo de conexão local ou remota para comunicação com a impressora
- Make: fabricante da impressora para poder exibir os drivers disponíveis
- Model: identificação do modelo da impressora para escolha do driver mais adequado

Uma vez criada a fila, o seu funcionamento poderá ser configurado pelo menu “Printers” escolhendo-se o botão “Configure printer” da impressora desejada como na tela abaixo.



A criação de filas de impressora pela linha comando utiliza o aplicativo `lpadmin` e requer os seguintes parâmetros:

OPÇÃO	DESCRIÇÃO
<code>-d</code>	Define a fila padrão (impressora default)
<code>-h ip</code>	Aplica as configurações ao servidor especificado
<code>-p printer opções</code>	Define as opções à impressora informada
<code>-x fila</code>	Remove a fila especificada

Para a definição de informações de impressora, temos as opções na tabela abaixo.

OPÇÃO	DESCRIÇÃO
<code>-m</code>	Define o driver a ser utilizado
<code>-v</code>	Define a localização da impressora
<code>-P</code>	Utiliza um driver fornecido por terceiros
<code>-u [allow deny]:</code>	Permite ou proíbe o acesso à fila para os usuários descritos
<code>-E</code>	Habilita a fila e o encaminhamento para a impressora

### Exemplos

Criando uma fila de impressão chamada `deskjet` para uma impressora conectada à porta paralela.

```
# lpadmin -p deskJet -E -v parallel:/dev/lp0 -m deskjet.ppd
```

Criando uma fila para uma impressora remota de rede:

```
# lpadmin -p lj4p -E -v ipp://192.168.2.200/lj4p -m laserjet.ppd
```

Criando uma fila para uma impressora em um máquina Windows:

```
# lpadmin -p dj840 -E -v smb://servidor/dj840 -m deskjet.ppd
```

Removendo uma fila de impressão

```
# lpadmin -x lj4p
```

Habilitando uma fila de impressão recém criada:

```
# /usr/sbin/enable optras2150
```

## Status do servidor CUPS – Ipstat

Uma das tarefas administrativas mais comuns é a verificação de status de uma impressora ou sua fila. Esta tarefa é realizada com o comando `lpstat`.

Verificando se uma fila está aceitando trabalhos (está habilitada) de impressão.

```
# lpstat -a optra1250
```

Verificando o status de uma impressora (imprimindo ou parada):

```
# lpstat -p lj4p
```

Verificando qual a impressora padrão e os dispositivos de impressão:

```
# lpstat -s
```

## Status de impressoras – Ipc

Para obter mais detalhes de cada fila de impressão utilizamos o comando `lpc`. No CUPS sua atuação é bem limitada, se restringindo a verificação do status de uma impressora/fila. Sua versão utilizada pelo servidor `lpr` é bem mais completa e permite várias tarefas adicionais como alterar a ordem de impressão de trabalhos entre outros.

Verificando o status de uma impressora:

```
# lpc status lj4p
```

## Enviando arquivos para impressão e gerenciando filas

Imprimir arquivos pela linha de comando ou verificar quantos arquivos estão aguardando para serem impressos são algumas das tarefas administrativas de um servidor de impressão. O gerenciamento envolve ainda redirecionar trabalhos quando uma impressora específica esteja offline ou com problemas.

Enviando um arquivo para impressão:

```
# lpr /etc/cups/printers.conf
```

Enviando um arquivo para impressão em uma fila remota:

```
# lpr /etc/passwd -P lj4p
```

Enviando um arquivo para impressão, em duas cópias, em espera:

```
# lpr -#2 -q -r /etc/pam.conf
```

Consultando a fila da impressora padrão:

```
# lpq
```

Resumindo a impressão do arquivo em espera:

```
# lp -i 25 -H resume
```

Consultando uma fila remota:

```
# lpq -P lj4p
```

Removendo um trabalho da fila:

```
# lprm -P lj4p 25
```

Alterando o número de cópias a ser impresso de um trabalho já enviado:

```
# lp -i 10 -n 2
```

Alterando a prioridade de um trabalho a ser impresso:

```
# lp -i 20 -q 100
```

Movendo um trabalho de impressão de uma fila para outra:

```
# lpmove dj840-10 lj4p
```

## Documentação

Uma das fontes mais vasta de informações sobre o Linux é a Internet. Nela é possível encontrar desde documentos ensinando como fazer determinado serviço ou equipamento funcionar no Linux até dicas de alguém que passou por algum problema e encontrou uma solução criativa.

Além da Internet existe uma grande variedade de publicações entre livros e revistas especializadas em Linux à disposição e, de forma muito particular, o próprio Linux traz muitos documentos a respeito de seus recursos e serviços.

### Encontrando documentação no Linux

Em nossa própria instalação existem muitas informações disponíveis como manuais de referência on-line, notas de versão e ajuda direta simplificada além de itens sobre o que é um comando ou se há algo sobre determinado assunto, vamos ver como elas funcionam.

#### man pages

O primeiro contato com algum tipo de documentação encontrado no próprio Linux são as *man pages* (páginas de manual) que funcionam como um guia de referência on-line sobre programas, utilitários ou funções e ainda sobre diversos aspectos do Linux como a hierarquia de diretórios e outros tópicos.

A estrutura do arquivo em tela segue um padrão, apresentando os seguintes tópicos:

SEÇÃO	DESCRIÇÃO
NOME	Nome do programa ou arquivo pesquisado
SINOPSE	Exemplos de funcionamento do comando com suas opções
DESCRÍÇÃO	Descrição do tópico pesquisado
OPÇÕES	Opções de uso
ARQUIVOS	Arquivos envolvidos na configuração ou uso do programa
VEJA TAMBÉM	Referências adicionais
BUGS	Problemas encontrados
AUTOR	Autor

Os arquivos de referência são separados em seções, o que permite um melhor controle sobre o conteúdo disponibilizado e ainda possibilita, se necessário, a existência de *man pages* de mesmo nome em seções diferentes. Abaixo temos uma lista das seções disponíveis.

SEÇÃO	DESCRIÇÃO
1	Programas executáveis ou comandos shell
2	Chamadas de sistemas (funções providas pelo kernel)
3	Chamadas internas (funções em bibliotecas de programas)
4	Arquivos especiais (normalmente no /dev)
5	Formatos de arquivos e convenções (ex: /etc/passwd)
6	Jogos
7	Diversos (incluindo pacotes de macros e convenções), ex: man(7), groff(7), hier(7)
8	Comandos administrativos (disponíveis para o root)
9	Rotinas de Kernel [Não padrão]

## O comando whatis

Toda página de manual tem, na seção NOME, um breve descrição sobre o uso do comando. Este comando exibe a breve descrição sobre o comando ou arquivo informado.

```
# whatis cat
```

## O comando apropos

Pouco utilizado, este comando fornece uma prévia sobre se há algo a respeito de determinada ocorrência. Apropos procura a palavra informada tanto nesta breve descrição como no nome da página.

Abaixo temos alguns exemplos de seu uso.

Pesquisa páginas sobre a ocorrência passwd:

```
# apropos passwd
```

A pesquisa pode ter mais de uma parâmetro:

```
# apropos passwd password
```

## O comando man

O funcionamento padrão das man pages é buscar pelo argumento informado por um arquivo de documentação que será exibido ao usuário, por exemplo, para saber como funciona o

comando de troca de senha `passwd` você pode digitar o seguinte comando:

```
# man passwd
```

A sintaxe do comando `man` e seus principais parâmetros podem ser vistos na tabela abaixo.

man [opções] página	
OPÇÃO	DESCRIÇÃO
-a	Pesquisa em todas as seções pela página solicitada
-k	Pesquisa a palavra informa nas páginas exibindo um lista
seção	Pesquisa pela página na seção informada

Exemplos:

Pesquisa pela página `printf` em todas as seções:

```
# man -a printf
```

Pesquisa pela descrição do arquivo `passwd` na seção 5:

```
# man 5 passwd
```

Pesquisa por todas as páginas que possuam alguma referência a `passwd`:

```
# man -k passwd
```

## Texinfo

As páginas de manual oferecem um conteúdo estático, fazendo referência a outros documentos a partir da seção “VEJA TAMBÉM” (SEE ALSO). As páginas de documentação providas pelo Texinfo oferecem um recurso de navegação dinâmica entre documentos, fazendo que a pesquisa informações seja mais rápida e, em muitos casos, mais eficaz.

Muitas man pages deixaram de ser atualizadas e vem sendo substituídas pelas Texinfo, este é o caso do comando `cat`. Abaixo temos alguns exemplos de uso:

Para ver todas as páginas disponíveis no Texinfo de seu equipamento:

```
# info show top-level dir menu
```

Vendo informações sobre o pacote coreutils:

```
# info coreutils
```

Acessando um subtópico de uma página diretamente:

```
# info coreutils cat
```

## Documentação Linux na Internet

Além dos recursos disponíveis localmente, a Internet é uma vasta fonte de informações sobre o funcionamento do Linux. Em especial os serviços de pesquisa como Google® e sites que disponibilizam documentação e tutoriais, alguns destes são:

- [Google Linux](http://www.google.com.br/linux) (<http://www.google.com.br/linux>)
- [The Linux Documentation Project](http://www.tldp.org) (<http://www.tldp.org> ou <http://br.tldp.org>)

No site de documentação do Linux está disponível um documento sobre vocabulário padrão utilizado (<http://br.tldp.org/ferramentas/vp/>), não só no Linux, e que pode ajudar muita gente a entender as coisas mais facilmente, porém ele está em um formato compatível com a ferramenta kbabel do KDE. Nesta página você encontrará informações de como utilizá-lo.

## Notificação e alertas a usuários

Notificar usuários é uma tarefa realizada, normalmente, através de mensagens de logon ou em seguida ao acessar um determinado sistema.

Neste contexto há três arquivos que permitem exibir mensagens aos usuários.

- `/etc/issue`  
Para acessos pela console (login em modo texto diretamente no equipamento), as mensagens armazenadas neste arquivo são exibidas antes da linha de login.
- `/etc/issue.net`  
Para o acesso remoto, via telnet ou ssh, as mensagens neste arquivo são exibidas antes do login do usuário. O servidor ssh precisa ser configurado para poder exibir este arquivo.
- `/etc/motd`  
As mensagens deste arquivo só são exibidas após a autenticação do usuário.

## Shell

### Ambiente do usuário

Assim que tem acesso ao sistema operacional (login) uma série de configurações pré-definidas são disponibilizadas ao usuário, montando o seu ambiente de trabalho. Este ambiente pode ser personalizado e, uma vez que a maioria do trabalho do administrador é realizada em modo texto, é altamente recomendável torná-lo mais amigável acrescentando funcionalidades ou modificando seus recursos.

Todo o ambiente de trabalho no Linux é controlado por variáveis de ambiente. Estas variáveis controlam desde quais diretórios são pesquisados à procura de programas (PATH) até o tamanho da tela (número de linhas e colunas visíveis), modificando o comportamento dos programas que adequam suas telas ao novo tamanho do terminal.

### Variáveis

Um variável é um nome, definido pelo usuário (sem acentos, espaços ou símbolos) que armazena um valor em memória RAM. Seu conteúdo pode ser uma palavra, uma string (texto) ou um número. Criar variáveis pode ser realizado assim:

```
# MENSAGEM="olá"
```

Este comando associa a string “olá” à área de memória identificada como MENSAGEM. Esta área de memória pode ser consultada a qualquer momento no shell corrente porém, da forma como criada, não poderá ser utilizada em outros programas.

A consulta a uma variável é realizada pelo comando echo, precedendo o nome da variável com \$, assim:

```
# echo $MENSAGEM
olá
```

Uma variável pode ser removida da memória tão facilmente quanto pode ser criada, com o comando unset:

```
# unset MENSAGEM
```

Alguns pontos a observar na criação de variáveis é que na sua atribuição NÃO devem existir espaços antes ou depois do sinal de igual e seu conteúdo, quando for um texto, deve ser envolto em “” (aspas).

## Variáveis de ambiente

Variáveis de ambiente são aquelas que já estão disponíveis quando o usuário efetua login em um terminal. Elas podem conter vários tipos de informações. Abaixo vemos alguns exemplos:

```
# env
LESSKEY=/etc/.less
LC_PAPER=pt_BR
LC_ADDRESS=pt_BR
LC_MONETARY=pt_BR
HOSTNAME=localhost
TERM=xterm
SHELL=/bin/bash
(...)
```

As variáveis de ambiente podem ser utilizadas por qualquer programa que as consulte, sendo definidas como variáveis de abrangência global (variáveis exportadas). Uma variável que seja criada com os procedimentos vistos acima são consideradas de abrangência local, não podendo ser consultadas por qualquer outro programa além do shell onde foram criadas.

Exportar uma variável é um procedimento tão simples quanto a sua criação e é essencial para que ela seja útil ao sistema. Vamos, inicialmente, criar uma variável simples e consultá-la em um novo shell:

```
# MENSAGEM="posso ver esta mensagem"
# echo $MENSAGEM
posso ver esta mensagem
# bash
# echo $MENSAGEM
<- criação da variável
<- consulta da variável local
<- iniciando um novo shell
<- consulta da variável local
<- conteúdo não está disponível
# exit
#
<- voltando ao shell anterior
```

Definindo a variável e exportando-a permite que outros programas como o novo shell tenham acesso a ela:

```
# MENSAGEM="posso ver esta mensagem"
# export MENSAGEM
# echo $MENSAGEM
posso ver esta mensagem
# bash
# echo $MENSAGEM
posso ver esta mensagem
# exit
#
<- criação da variável
<- tornando a variável global
<- consulta da variável local
<- iniciando um novo shell
<- consulta da variável local
<- conteúdo está disponível
<- voltando ao shell anterior
```

A partir da execução do comando `export`, todo e qualquer novo programa que consulte o ambiente do usuário poderá utilizar esta variável e seu conteúdo.

## Scripts executados no login do usuário

Todas as variáveis são definidas no momento do acesso do usuário ao shell do sistema. Estas definições são carregadas a partir de login-scripts executados automaticamente. Há dois tipos de scripts: os válidos para todos os usuários, definidos no `/etc` e os pessoais, localizados no home do usuário.

Estes scripts possuem vários comandos do shell que podem ser encadeados de acordo com as necessidades de cada equipamento/sistema. No Linux temos os seguintes login-scripts:

ARQUIVO	USO
<code>/etc/profile</code>	Arquivo global, define variáveis exportadas para todos os usuários do sistema como o PATH e ainda define o comportamento de vários aplicativos
<code>/etc/bashrc</code>	Arquivo global, define macros disponíveis para todos os usuários como "ls" alterado para saída colorida
<code>~/.bash_profile</code>	Arquivo pessoal que permite a personalização de variáveis como o PATH, o prompt de comando do usuário
<code>~/.bashrc</code>	Arquivo pessoal que permite a criação de macros personalizadas como a listagem automática dos logs do sistema ou atalhos para comandos complexos repetitivos
<code>~/.bash_logout</code>	Arquivo pessoal executado no logout do usuário, normalmente tem um comando clear, mas pode ser alterado para incluir o backup de dados importantes.

## Variáveis pré-definidas no ambiente do usuário

Entre as muitas variáveis disponíveis, algumas merecem destaque e devem ser citadas devido à sua importância e uso no Linux:

VARIÁVEL	DESCRIÇÃO
<code>PATH</code>	Caminho de pesquisa à procura de comandos para executar
<code>HOME</code>	Diretório pessoal do usuário logado
<code>USER</code>	Nome de login do usuário logado, pode ser utilizado ainda <code>LOGNAME</code> e <code>USERNAME</code> disponíveis por compatibilidade
<code>LANG</code>	Idioma preferencial
<code>PWD</code>	Localização atual, mesma saída do comando <code>pwd</code>
<code>LC_ALL</code>	Localidade definida na instalação, alterando a visualização de números, data e hora e outros.
<code>TMP</code>	Indica o diretório temporário
<code>PS1</code>	Define o comportamento do prompt de comandos

VARIÁVEL	DESCRIÇÃO
HISTSIZE	Número de comandos armazenados no histórico do Linux

## Shell scripts

No Linux muitos dos comandos e utilitários disponíveis são scripts que permitem a solução de problemas específicos de forma simples, rápida e eficiente. Neste tópico serão vistos os fundamentos de como são criados os scripts em shell, da mesma forma que um administrador em outra plataforma deve ser capaz de criar pequenos arquivos bat.

Um script nada mais é do que um arquivo texto com os comandos que são executados no prompt de forma encadeada onde, logo após um comando o seguinte é executado. Crie o seguinte arquivo com o nome de teste.sh.

```
# script de teste – esta linha é um comentário e não será executada
echo "Script de teste"
date
pwd
```

Para executarmos este script iremos pedir a um shell que o interprete. Depois vamos dar permissão de execução e testar novamente.

```
# sh teste.sh
Script de teste
Qui Aug 21 05:44:47 BRT 2006
/root
```

O mesmo resultado pode ser obtido se atribuirmos permissão de execução ao script.

```
# chmod +x teste.sh
# ./teste.sh
```

## Estrutura de um script

Todo script deve ter uma estrutura inicial com informações sobre como será executado, seu autor e seu uso e um histórico de mudanças ocorridas ao longo do tempo. Este cabeçalho inicial não é obrigatório, mas é recomendado que seja colocado em seus scripts para facilitar a identificação futura de como ele foi construído ou por que e por quem foi modificado. Um exemplo desta estrutura pode ser vista abaixo.

```

#!/bin/sh
#
# nome_do_script.sh - Descrição simples da sua função
#
# Autor : Quem o fez <email@decontato.com.br>
#
# Este programa recebe tais parâmetros e executa tais tarefas
# retornando tais informações ou fazendo tais procedimentos.
#
# Exemplos:
# $ ./nome_do_script.sh parâmetro
# Resultado
# $
#
# Histórico:
# 1999-05-18, João da Silva:
# - Versão inicial realizando tal tarefa

# Definição de variáveis utilizadas pelo script
VAR1=
VAR2=
VAR3=

<comandos do script>
```

## Shell's de execução

O Linux é conhecido pelo grande número de interpretadores (shell's) disponíveis. Entre os mais comuns temos a tabela abaixo.

SHELL	Descrição
sh	Boune Shell, simples e encontrado em todos os Linux/Unix
bash	Bourne Again Shell, mais elaborado que o sh, é o shell padrão em quase todas as distribuições Linux atualmente
ksh	Korn Shell, é o shell padrão dos sistemas Unix System V possui uma implementação livre no Linux: o pdksh
zsh	Z Shell similar ao ksh possui muitos recursos adicionais
csh	C Shell, criado no BSD possui sintaxe diferente do bash. O tcsh é uma versão melhorada do csh.

Devido a esta variedade, ao criar um script é necessário definir em qual ambiente ele deve ser executado. A primeira linha de um script deve ser uma diretiva de execução informando ao Linux que determinado shell deve ser carregado e que os comandos do script devem ser executados dentro dele. Isto garante que o funcionamento do script será o mesmo, em qualquer distribuição, em qualquer versão. Esta diretiva aparece abaixo e deve a ser a primeira linha do script.

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

## Recebendo parâmetros pela linha de comando

Alguns scripts podem executar tarefas de forma automatizada sem a necessidade de interação com o usuário contudo a forma mais comum de execução de scripts é informar parâmetros pela linha de comando indicando o que deve ser executado ou sobre quais arquivos o script deve atuar.

Os parâmetros informados pela linha de comando são identificados como variáveis numéricas seqüenciais, ou seja: o primeiro parâmetro está na variável 1, o segundo na variável 2 e assim sucessivamente. Há variáveis especiais para o tratamento de parâmetros na tabela abaixo.

VARIÁVEL	DESCRIÇÃO
\$1 ... \$n	Parâmetros informados na linha de comando
\$*	Todos os parâmetros informados na linha de comando
\$#	Número de parâmetros passados na linha de comando
\$0	Nome com o qual o script foi acionado

Um exemplo de passagem de parâmetros é o script abaixo. Crie-o com o nome de `parm.sh` e execute com `./parm.sh parametro`:

```
#!/bin/bash
echo "$"
```

Caso seja necessário criar um script que receba vários parâmetros pela linha de comando e execute seus comandos para cada parâmetro informado você precisará rotacionar os dados informados com o comando `shift`. Veja o código abaixo.

```
#!/bin/bash
echo "$*"
shift
echo "$*"
shift
echo "$*"
```

## Construção condicional

Todo script executa funções de forma seqüencial em um roteiro pré-definido, porém é possível que determinadas ações sejam executadas quando uma condição específica acontecer. Esta condição pode ser a digitação de uma informação pelo usuário ou a localização de um recurso pelo script. Sua construção é mostrada a baixo.

```
if [ condição ] then
    comando
fi
```

Todo comando executado em um terminal retorna o código 0 (zero) quando bem sucedido e um código diferente de zero quando ocorrem problemas. Desta forma os testes de validação em scripts consideram o valor zero como teste bem sucedido ou verdadeiro.

As condições de teste possíveis em um script para verificação de arquivos são:

TESTE	DESCRIÇÃO
-d NOME	Testa se NOME existe e é um diretório.
-e NOME	Testa se NOME existe
-f NOME	Testa se NOME existe e é um arquivo.
-h NOME	Testa se NOME existe e é um symlink.
-L NOME	

Para o tratamento de strings os testes são:

TESTE	DESCRIÇÃO
-z STRING	Testa se o comprimento da STRING é zero.
-n STRING	Testa se o comprimento da STRING não é zero.
STRING	
STRING1 = STRING2	Testa se as string comparadas são iguais.
STRING1 != STRING2	Testa se as string comparadas são diferentes.

Para verificação de valores numéricos:

TESTE	DESCRIÇÃO
INT1 -eq INT2	Número INT1 é igual a INT2
INT1 -ge INT2	Número INT1 é maior que ou igual a INT2
INT1 -gt INT2	Número INT1 é maior que INT2
INT1 -le INT2	Número INT1 é menor que ou igual a INT2
INT1 -lt INT2	Número INT1 é menor que INT2
INT1 -ne INT2	Número INT1 não é igual a INT2

Um exemplo simples da construção de um teste "if" é:

```
if [ -d /boot ]; then
    echo "/boot é um diretório"
fi
```

Em alguns casos uma ação complementar deve ser executada se o teste inicial falhar:

```
if [ -f /etc/yp.conf ]; then
    echo "/etc/yp.conf é um arquivo"
else
    echo "/etc/yp.conf não localizado"
fi
```

Uma variação desta estrutura é quando há muitas opções de teste:

```
if [ -f /etc/yp.conf ]; then
    echo "/etc/yp.conf é um arquivo"
elif [ -f /etc/printcap ]; then
    echo "/etc/printcap é um arquivo"
else
    echo "Arquivos não localizados"
fi
```

## Estrutura de repetição – laços

Muitas vezes é necessário executar uma série de comandos de forma repetitiva por um determinado número de vezes ou até que uma condição específica seja satisfeita. Para que este tipo de controle seja possível podemos utilizar as estruturas `while`, `until` ou `for`.

### **while**

Este comando executa um bloco de comandos até que a condição especificada se torne falsa, por exemplo renomear todos os arquivos de um diretório ou exibir um arquivo de log em tela até que uma ocorrência de erro seja encontrada ou o exemplo mais simples: um contador.

```
#!/bin/bash
CONTADOR=0
while [ $CONTADOR -lt 10 ]; do
    echo "Contagem: $CONTADOR"
    let CONTADOR=CONTADOR+1
done
```

### **for**

Enquanto o laço `while` testa uma condição até que ela se torne falsa, o laço `for` executa um bloco de comandos para cada um dos elementos informados.

```
#!/bin/bash
for i in `seq 10`; do
    echo "item: $i"
done
```

## until

Este comando é similar ao funcionamento da estrutura while., porém o tipo de teste realizado deve ser de adaptado.

```
#!/bin/bash
CONTADOR=0
until [ $CONTADOR -gt 10 ]; do
    echo "Contagem: $CONTADOR"
    let CONTADOR=CONTADOR+1
done
```

## Usando funções em scripts

Quando uma parte de seu script deve ser executado de forma repetitiva em trechos alternados, ao invés de copiar o mesmo código várias vezes, podemos criar uma função que será executada sempre que necessário, recebendo os parâmetros informados dentro do próprio script. Uma função pode ser chamada dentro de outra, um exemplo deste uso pode ser uma pausa no script abaixo utilizado para verificação de logs.

```
#!/bin/bash

separador() {
    for i in `seq 72`; do
        echo -n "="
    done
}

espera() {
    separador
    echo -e "\nPressione Enter para continuar ou <ctrl>+<c> para sair"
    read
}

# Executa um loop infinito
while true; do
    tail -n 5 /var/log/messages
    espera
done
```

## Referências

- Advanced Bash-Scripting Guide (<http://www.tldp.org/LDP/abs/html/index.html>)
- Introduction to Bash Programming (<http://www.tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html>)
- SCRIPTS DE SHELL LINUX COM BASH - BURTCHE, KEN O. - CIENCIA MODERNA - 2005
- Aurélio.net – Shell Script (<http://aurelio.net/shell/>)

## Tarefas administrativas

Entre as várias atividades de um administrador Linux estão tarefas como verificação de logs, realização de backup's (e testá-los é lógico) além de manter os dados de seu servidor de forma confiável com sincronismo de horário.

Estes tópicos serão vistos pela visão do administrador, onde as informações são mais relevantes, não deixando de lado o aspecto da certificação.

### Logs do sistema

O Linux é conhecido por seu lado detalhista em informações de log. Um evento que cause uma ocorrência de log é, muitas vezes, registrado em vários arquivos e de formas diferentes. A ativação de um servidor como o Apache pode gerar entradas no principal arquivo de log do sistema o /var/log/messages além de resultar em ocorrências no log do sistema de servidores /var/log/daemon.log. O servidor de logs padrão do Linux é o sysklogd (System & Kernel Log Daemon). Sua configuração obedece a parâmetros avaliados em pares, indicando as facilidades (serviços) e níveis de informação que devem ser registrados. Abaixo temos a tabela com estas facilidades.

FACILIDADE	DESCRIÇÃO
AUTHPRIV	Informações sobre segurança e autorização
CRON	Agendamento de tarefas (crond e atd)
DAEMON	Servidores que não possuem facilidades exclusivas
KERN	Mensagens de kernel (principalmente hardware)
LOCAL0...7	Serviços executados localmente como LDAP e outros
LPR	Mensagens do sistema de impressão
MAIL	Servidores de email, pop e imap
NEWS	Mensagens do sistema de notícias do tipo USENET
SYSLOG	Mensagens geradas internamente pelo servidor de log
USER	Mensagens de tarefas ou comandos de usuários
UUCP	Mensagens do sistema UUCP

As facilidades AUTH e SECURITY não são mais utilizadas. Apenas deve-se considerar a facilidade AUTHPRIV.

Os níveis de informações são classificados como abaixo.

NÍVEIS	DESCRIÇÃO
EMERG	Sistema inoperante
ALERT	Uma ação corretiva deve ser realizada imediatamente

NÍVEIS	DESCRIÇÃO
CRIT	Condições críticas de funcionamento
ERR	Ocorreu um erro (serviço ou hardware)
WARNING	Atenção necessária
NOTICE	Um evento normal, mas que pode necessitar de atenção
INFO	Mensagem meramente informativa
DEBUG	Modo debug gerando muito log

O nível ERROR não deve ser mais utilizado, apenas ERR assim como WARN em favor de WARNING.

A configuração do syslog utilizada no Debian pode ser vista pelo arquivo /etc/syslog.conf e foi reproduzida abaixo.

```
# Configuração padrão do servidor sysklogd utilizada no Debian
auth,authpriv.*                      /var/log/auth.log
*.*;auth,authpriv.none                 -/var/log/syslog
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
mail.info                               -/var/log/mail.info
mail.warn                               -/var/log/mail.warn
mail.err                                /var/log/mail.err
news.crit                               /var/log/news/news.crit
news.err                                /var/log/news/news.err
news.notice                            -/var/log/news/news.notice
*.=debug;\n    auth,authpriv.none;\n    news.none;mail.none      -/var/log/debug
*.=info;*.=notice;*.=warn;\n    auth,authpriv.none;\n    cron,daemon.none;\n    mail,news.none          -/var/log/messages
*.emerg                                *
daemon.*;mail.*;\n    news.crit;news.err;news.notice;\n    *=debug;*=info;\n    *=notice;*=warn        |/dev/xconsole
```

O uso do sinal “-” antes do nome de cada arquivo indica ao servidor que suas ocorrências

podem ser gravadas em disco em lotes maiores (uso de cache de gravação em disco). Isto pode causar perda de informações caso ocorra uma falha de energia com dados ainda em memória, mas traz ganhos de performance, principalmente quando se tem aplicações que geram muitos logs. Este recurso não deve ser utilizado com logs de Firewalls ou serviços como NIDS.

## **Configurações adicionais**

Você pode especificar o par seletor (facilidade.nível) informando mais de uma facilidade separadas por vírgula e um nível:

authpriv,kern.info	/var/log/teste.log
--------------------	--------------------

Se necessário informar facilidades e níveis diferentes, estes devem ser separados por ";":

cron.err;news.crit	/var/log/teste.log
--------------------	--------------------

As duas condições podem ser utilizadas em conjunto:

mail.info;news,cron.err	/var/log/teste.log
-------------------------	--------------------

Mensagens de log podem ser direcionadas para usuários:

*.alert	root,marcos,claudia
---------	---------------------

Para se ter uma cópia online de todas as ocorrências, você pode enviar seus logs para outro equipamento (servidor de logs):

kern.*;authpriv.*	@192.168.2.200
-------------------	----------------

Para que o servidor de logs possa receber as mensagens do equipamento que está enviando, é necessário alterar o arquivo /etc/init.d/sysklog definindo a variável abaixo.

SYSLOGD="-r"
--------------

## **Rotacionando os logs automaticamente – logrotate**

Manter o Linux funcionando requer acompanhamento das informações armazenadas em log, não só pelas ocorrências, mas também para que estes logs não acabem com o espaço em disco e travem seu servidor.

O Linux vem preparado para rotacionar seus logs semanalmente mantendo, no máximo, a

semana em curso mais as quatro semanas anteriores. Este funcionamento é controlado pelo logrotate.

O comando logrotate é executado diariamente, de forma a poder decidir se arquivos devem ser rotacionados diária, semanal ou mensalmente. Seu acionamento é feito pelo sistema de agendamento de tarefas cron e sua configuração principal é controlada pelo arquivo /etc/logrotate.conf e todos os arquivos do diretório /etc/logrotate.d. Sua configuração pode ser vista abaixo.

```
# rotaciona os logs semanalmente
weekly

# mantém 4 arquivos de logs além do atual
rotate 4

# após o rotacionamento dos logs cria novos arquivos vazios
create

# compacta os logs rotacionados
compress

# outros arquivos de logs a serem rotacionados
include /etc/logrotate.d

# os logs wtmp, e btmp são independentes – nós controlamos seu rotacionamento
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}
```

Os arquivos complementares lidos do subdiretório logrotate.d são gerados por pacotes como o syslog, cups e apache entre outros, informando como seus logs devem ser rotacionados (periodicidade, compactação, etc).

Alguns dos parâmetros possíveis para a configuração do logrotate estão na tabela a seguir.

OPÇÃO	DESCRIÇÃO
compress	Compacta os logs após rotacionamento
mail	O log a ser removido (mais antigo) será encaminhado para o email informado
missingok	Não acusa erro se não encontrar o arquivo de log
notifempty	Não rotaciona o log caso o arquivo esteja vazio
rotate	Número de logs a manter
daily/weekly/monthly	Periodicidade de rotacionamento do log
size	Rotaciona o log por tamanho (100k, 1M) e não por tempo

## Agendamento de tarefas

Tarefas podem ser executadas em intervalos regulares ou em datas pré-definidas com os sistemas de agendamento do Linux. Estes sistemas são o cron de agendamento periódico e o at de evento único.

Mesmo que você não tenha criado nenhuma tarefa agendada, o Linux já tem várias rotinas pré-definidas como rotacionamento de logs e limpeza de arquivos temporários (em algumas distribuições isto inclui todos os arquivos do diretório /tmp).

### at - Evento único

Com o comando at é possível, por exemplo, programar para que os servidores desliguem automaticamente no próximo sábado às 7h da manhã devido à manutenção do sistema elétrico da empresa, ou forçar a re indexação de uma base num horário de pouco uso do servidor. Estes exemplos são tarefas que não ocorrem o tempo todo mas que será necessário sua execução. Com o agendamento evitamos termos que nos deslocar à empresa somente para este fim e ainda não esquecemos que isto deve ser realizado.

Sua sintaxe é simples e se você quer acionar um programa que não necessita de parâmetros, este pode ser indicado diretamente pela linha de comando, como abaixo:

```
# at -f /usr/bin/updatedb 10:01
warning: commands will be executed using /bin/sh
job 1 at 2006-09-12 10:01
```

Se nada em contrário for indicado, o horário é passado no formato 24h. Este exemplo funciona perfeitamente pois o comando updatedb é um script. Se utilizarmos um executável binário (programa), o comando será aceito porém nada será realizado. Nestes casos precisamos utilizar o shell de entrada de dados ou criar um script com os comandos a serem executados. O uso do shell interativo funciona como abaixo:

```
# at 18:00
at> /sbin/shutdown -h now
at> <ctrl>+<d>
```

Outros exemplos da aplicação de horário do comando at são vistas na seguinte tabela.

HORA	DESCRIÇÃO
midnight	Meio dia
noon	Dezoito horas
teatime	Dezesseis horas
hh:mm	Hora específica no formato 24h
hh:mm[am pm]	Hora específica no formato 12h com indicação de período

HORA	DESCRIÇÃO
now +tempo	Do horário atual mais o tempo definido: minutes ou min: minutos a mais: at now +15min hours: horas a mais: at now+12hours days: dias weeks: semanas
hh:mm today	Hoje na hora especificada
hh:mm tomorrow	Amanhã na hora especificada

Se o parâmetro de horário for omitido quando se especifica a data, será utilizado o horário atual na data especificada.

Para poder listar os agendamentos (jobs) existentes utilizamos o comando atq.

```
# atq
4      2006-09-30 07:22 a root
6      2006-09-24 08:24 a root
7      2006-09-24 16:00 a root
```

O job é identificado pela primeira coluna. Se houver a necessidade de remover um job utilize o comando atrm.

```
# atrm 6
```

Todas as tarefas agendadas são armazenadas em arquivos localizados em /var/spool/cron/atjobs. A primeira parte de cada arquivo contém a definição do ambiente de execução (variáveis, paths, etc.) e por último vem os comandos a serem executados.

## cron – Ocorrência periódica

Diferente do comando at, as tarefas agendadas pelo cron são executadas de forma periódica. Com o cron podemos especificar períodos regulares de execução como de hora em hora, diário e semanal entre outros. Podem ser utilizados ainda períodos de execução não regular como todas as terças e quintas dos meses de fevereiro e setembro.

Grande parte das tarefas periódicas que devem ser realizadas em um equipamento já vêm pré-agendadas e são controladas pelas definições do arquivo /etc/crontab como abaixo.

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

#m h    dom mon dow   user     command
17 *      *    *    *   root    run-parts --report /etc/cron.hourly
25 6      *    *    *   root    run-parts --report /etc/cron.daily
47 6      *    *    7   root    run-parts --report /etc/cron.weekly
52 6      1    *    *   root    run-parts --report /etc/cron.monthly
```

As definições mais importantes do arquivo acima indicam que nos horários especificados cada diretório deve ser acessado e cada script dentro dele deve ser executado. Lendo o arquivo acima temos, de forma simplificada, a seguinte estrutura.

DIRETÓRIO	QUANDO EXECUTAR
/etc/cron.hourly	De hora em hora, aos 17 minutos
/etc/cron.daily	Diariamente, às 6h25 da manhã
/etc/cron.weekly	Semanalmente, no domingo às 6h47 e da manhã
/etc/cron.monthly	Mensalmente, no dia primeiro às 6h52 da manhã

A estrutura do arquivo crontab é a seguinte:

min	hora	dia	mês	mês	dia	sem	usuário	comando
52	6	1	*	*	*	*	root	run-parts --report /etc/cron.monthly

No sistema cron, são utilizados os primeiros cinco campos para indicar a data e horário de execução, em seguida quem executará o comando e por último o comando propriamente dito.

Cada campo possui uma faixa de valores aceitáveis como na tabela abaixo.

CAMPO	FAIXAS
minuto	De 0 a 59
hora	De 0 a 23
dia do mês	De 0 a 31
mês	De 0 a 12
dia da semana	De 0 a 7 (domingo assume tanto o valor zero como sete)

Em casos específicos como a criação de um backup queremos que estes sejam realizados apenas durante a semana, nestes casos os campos devem assumir valores específicos de forma a identificar a faixa. Para determinar faixas de períodos ou execução de tarefas de forma alternada, utilizamos os caracteres de controle da tabela abaixo.

SÍMBOLO	Uso
- (hífen)	Intervalo: 1-5 representa de 1 a 5
, (vírgula)	Número específico: 1,3,7 representa em 1, em 3 e em 7
/ (barra)	Intervalo regular: */5 representa de cinco em cinco

## Backup

Muitos administradores de sistemas de informação vão identificar o Backup com uma das tarefas mais importantes de um ambiente computacional, porém é comum encontrarmos empresas que menosprezam esta atividade, relegando-a a cópias entre equipamentos ou armazenamento em locais sem qualquer segurança quanto à preservação das informações gravadas.

Muitos fatores influenciam este comportamento: falta de conhecimento do valor das informações da empresa, ou a simples falta de tempo para a realização desta tarefa.

O Linux oferece várias ferramentas abertas para a realização de backups e há vários softwares proprietários como ArcServer, Arkeia e BRU (Backup & Restore Utility).

## O comando tar

Uma das maneiras mais versáteis de realizar backups em Linux é com o comando tar. Com ele é possível realizar backups completos ou incrementais, de discos, diretórios ou apenas um arquivo. Os parâmetros mais comuns estão na tabela abaixo.

tar [opções] destino origem	
OPÇÃO	Descrição
c	Cria um backup
f	Nome do arquivo a ser tratado (criado ou extraído)
j	Utiliza o compactador bzip2 durante a operação
r	Acrescenta novos arquivos ao arquivo tar (usado com u)
t	Lista o conteúdo do arquivo informado
u	Atualiza os arquivos que são mais novos do backup
v	Lista os arquivos durante a operação (modo detalhado)
x	Extraí um arquivo tar
z	Utiliza o compactador gzip durante a operação
C	Muda o diretório especificado
M	Cria um backup multi-volume

Exemplos:

Criando um backup do diretório /etc para o arquivo /tmp/etc.tar:

```
# tar cf /tmp/etc.tar /etc
```

Criando um backup de mais de um diretório ao mesmo tempo:

```
# tar cf /tmp/backup.tar /etc /boot
```

Criando um backup compactado com gzip:

```
# tar czf /tmp/etc.tar.gz /etc
```

Criando um backup multi-volume em disquetes do diretório /etc:

```
# tar cvMf /dev/fd0 /etc
```

Testando o conteúdo de um backup e vendo o resultado completo em tela:

```
# tar tvzf /tmp/etc.tar.gz
```

Extraindo o conteúdo de um backup:

```
# tar xvzf /tmp/etc.tar.gz
```

Extraindo um backup para um diretório diferente do atual:

```
# tar xvf backup.tar -C /tmp
```

## O comando dd

O comando dd é utilizado para o backup de sistemas de arquivos (partições) ou discos de uma única vez. Possui características como definir o tamanho máximo de informações a serem copiadas ou o tamanho do bloco de cópia.

dd [opções]	
OPÇÃO	DESCRIÇÃO
if=arquivo	Arquivo ou dispositivo de entrada
of=arquivo	Arquivo ou dispositivo de saída
bs=bytes	Número de bytes que define o bloco de leitura/gravação
count=N	Número de blocos a processar

Uma das aplicações mais comuns deste comando é realizar o backup da MBR de um servidor e restaurá-la quando necessário. De forma geral entende-se que a MBR sejam os primeiros 512 bytes do disco de inicialização de um equipamento. O layout correto está na tabela abaixo:

BYTES	DESCRIÇÃO
446	Master Boot Record
64	Tabela de partições
2	Boot Code Signature, valida a tabela de partições

### Exemplos:

Criando um backup da MBR e da tabela de partições do seu equipamento em um arquivo ou disquete:

```
# dd if=/dev/sda of=backup-mbr.dat bs=512 count=1
# dd if=/dev/sda of=/dev/fd0      bs=512 count=1
```

Restaurando apenas o backup da MBR, sem alterar a tabela de partições:

```
# dd if=backup-mbr.dat of=/dev/sda bs=446 count=1
```

Criando uma imagem iso de um cd:

```
# dd if=/dev/cdrom of=imagem.iso
```

Fazendo uma cópia de um disquete:

```
# dd if=/dev/fd0 of=disquete.img
# dd if=disquete.img of=/dev/fd0
```

Criando um “ghost” de um hd:

```
# dd if=/dev/sdb of=/dev/sdc
```

## O comando cpio

CPIO é um acrônimo para “copy I/O” ou seja copiar de ou para um dispositivo de entrada e saída com uma unidade de fita ou um disco. Ele é capaz de criar tanto arquivos cpio como arquivos tar e possui facilidade como criação local ou remota de backups. Sua utilização é normalmente associada a comandos como find ou ls que cria uma lista de arquivos a serem armazenados.

cpio [opções]	
OPÇÃO	DESCRIÇÃO
-i	Extrai um arquivo cpio
-m	Preserva data e hora original do backup
-o	Cria um arquivo cpio
-t	Lista o conteúdo do arquivo .cpio
-v	Exibe mais informações, modo detalhado
-0	Cria o arquivo de destino, utilizado com -o

Exemplos de uso do cpio:

Criado um backup dos arquivos do diretório atual com exibição do andamento:

```
# ls | cpio -ov > backup.cpio
```

Listando o conteúdo do arquivo .cpio:

```
# cpio -iv -t -I backup.cpio
```

Extraindo o conteúdo de um arquivo .cpio

```
# cpio -iv -I backup.cpio
```

Criando um backup compactado de uma estrutura de diretório completa:

```
# cd /etc  
# find . -depth | cpio -o | bzip2 > /tmp/etc.cpio.bz2
```

Extraindo um backup compactado:

```
# bzcat etc.cpio.bz2 | cpio -ivdm
```

Extraindo apenas alguns arquivos do backup:

```
# bzcat etc.cpio.bz2 | cpio -ivdm "network/*"
```

## O comando dump

Este comando utiliza o arquivo `/etc/fstab` para identificar quais sistemas de arquivos podem ser backupeados por ele, como listado abaixo:

<file system>	<mount point>	<type>	<options>	<dump>	<pass>
<code>/dev/sda1</code>	<code>/</code>	<code>ext3</code>	<code>defaults,errors=remount-ro</code>	<code>1</code>	<code>1</code>
<code>/dev/sda5</code>	<code>none</code>	<code>swap</code>	<code>sw</code>	<code>0</code>	<code>0</code>

Para o seu funcionamento no Debian é necessário instalar o pacote `dump` antes de prosseguirmos.

```
# aptitude install dumo
```

O comando `dump` é capaz de identificar em partções ext2/ext3 arquivos que necessitam de backup, de acordo com o arquivo `/var/lib/dumpdates`. Para listar quais sistemas de arquivos necessitam de backup rode o comando:

```
# dump -w  
Dump these file systems:  
/dev/sda1 ( ) Last dump: never
```

Sua sintaxe de execução e seus principais parâmetros estão listados na tabela a seguir.

dump [opções] fonte-de-dados	
OPÇÃO	DESCRIÇÃO
-a	Aguarda sinalização de disco cheio da unidade de backup
-f	Indica onde gravar o backup: arquivo, /dev/st0, - (stdout)
-h	Nível de backup: 0 – completo, 9 – incremental
-u	Atualiza o arquivo dumpdates após concluir com sucesso
fonte-de-dados	Ponto de montagem de uma partição ou lista de arquivos

Exemplos:

Cria um backup da partição /boot para o arquivo /tmp/boot.bkp:

```
# dump -0uf /tmp/boot.bkp /boot
```

Este comando foi bem sucedido pois /boot é uma partição em nosso sistema. Caso o mesmo procedimento tivesse sido utilizado para um diretório, resultaria em falha pois não é possível usar a tag “-u” em backup de arquivos ou diretórios.

Criando um backup de um diretório com dump:

```
# dump -0f /tmp/etc.bkp /etc
```

Criando um backup compactado com dump:

```
# dump -0f - /etc | bzip2 > /tmp/etc.bkp.bz2
```

## O comando restore

Todo backup realizado com dump somente pode ser restaurado com o comando restore. O padrão de funcionamento deste comando é restaurar seus dados na localização atual, portanto antes de executá-lo esteja certo de estar no diretório correto para recuperar seus dados. Sua sintaxe e principais opções estão listados na tabela a seguir.

restore [opções] fonte-de-dados	
OPÇÃO	DESCRIÇÃO
-f	Arquivo com o backup a restaurar
-r	Restaura o backup
-v	Modo detalhado
-C	Compara o arquivo do backup com o do disco

Exemplos:

Restaurando um backup realizado com dump:

```
# restore -rvf etc.bkp
```

Os arquivos deste backup serão restaurados no diretório corrente. Se não for esta a sua intenção, mude para o diretório desejado e rode o comando acertando o caminho para o arquivo de backup.

Restaurando um backup compactado:

```
# bzcat /tmp/etc.bkp.bz2 | restore -rvf -
```

Testando seu backup:

```
# restore -Cvf /tmp/boot.bkp
```

## Gerenciamento de hora no Linux

Um dos maiores problemas em sistemas de informática é manter seus equipamentos sempre com a mesma data e hora, de forma sincronizada. Este recurso, apesar de parecer simples pode ajudar a encontrar a última versão de um documento ou saber exatamente a que horas um pedido foi realizado além de facilitar tarefas como backup e, quando necessário, a investigação de eventos de segurança.

Manter um sistema de informações sem confiar no horário dos equipamentos compromete todas as tarefas de uma rede, inclusive o sistema cron e o sistema at de gerenciamento de atividades.

O sistema de sincronismo de horário disponibilizado no Linux utiliza o Network Time Protocol mais conhecido com NTP. Este serviço não tenta sincronizar seu horário com os equipamentos participantes de uma rede. Ele efetua seu sincronismo com fontes de referência de hora conhecidos como *stratum*. Estas fontes de referência adotam o formato UTC (Universal Time Coordinated). Ao realizar o sincronismo o serviço NTP do Linux efetua os acertos necessários de acordo com o timezone do equipamento local.

Como fonte de referência para o uso do serviço NTP pode-se consultar o site <http://www.ntp.org> e, em nível nacional, todas as informações como a divulgação do horário nacional, incluindo o horário de verão, pode ser consultado no site da Rede Nacional de Ensino e Pesquisa – RNP (<http://www.rnp.br/ntp>).

Para o uso do serviço NTP é recomendado que o seu equipamento aponte para um servidor de menor distância (regional preferencialmente) para realizar o sincronismo de hora.

## Definindo sua localização

A primeira etapa do sincronismo de horário é identificar corretamente a qual fuso horário (timezone) seu esquipamento pertence.

Timezones são divisões de 15º do globo terrestre criadas a partir de Greenwich na Inglaterra, totalizando 24 fuso horários, que representam a distância, em horas, que uma localidade está do ponto zero. Isto foi criado para indicar o horário local em outras partes do mundo. A diferença em horas é definida em valores positivos se deslocado para o Oriente e em valores negativos se deslocado para o Ocidente em relação a Greenwich. Os fuso horários utilizados no Brasil estão na tabela abaixo.

NOME E DIFERENÇA	HORÁRIO DE VERÃO E DIFERENÇA	ABRANGÊNCIA
BREST -2:00	BREDT -1:00	Fernando de Noronha
BRST -3:00	BRDT -2:00	São Paulo, Rio, Brasília, Minas Gerais, Região Nordeste e Região Sul
BRWST -4:00	BRWDT -3:00	Região Oeste
BRAST -5:00	BRADT -4:00	Acre

Durante a instalação do Debian é definido o fuso-horário e se o relógio de seu hardware está definido como local ou UTC. Para podermos alterar estas configurações temos os comandos `tzconfig` (por linha de comando) e `tzsetup` (por menus) no Debian. Em outras distribuições utilizamos o comando `timeconfig`.

Ao selecionar uma localidade com um destes comandos é criado um link simbólico chamado `/etc/localtime` que aponta para as definições do fuso horário armazenadas no diretório `/usr/share/zoneinfo/fuso/localidade`

Além desta indicação o `timezone` também é identificado no arquivo `/etc/timezone`.

A localidade correta para a identificação, por exemplo, da cidade de São Paulo é America/Sao\_Paulo sendo feito da mesma forma para várias outras cidades brasileiras, entre elas: Belém, Boa Vista, Campo Grande, Cuiabá, Fortaleza, Maceió, Manaus, Noronha, Porto Velho, Recife e Rio Branco, além de estados como Bahia. Além desta forma mais precisa, pode-se utilizar a localização por região utilizando os arquivos em `/usr/share/zoneinfo/Brazil` com as localidades: Acre, DeNoronha, East (leste) e West (oeste).

## Instalando os pacotes necessários

Os pacotes disponíveis no Debian para o serviço NTP estão listados na tabela abaixo.

PACOTE	DESCRIÇÃO
<b>ntp-simple</b>	Network Time Protocol: servidor para sistemas simples
<b>ntp-server</b>	Network Time Protocol: ferramentas básicas do servidor
<b>ntp</b>	Network Time Protocol: ferramentas de rede
<b>ntp-doc</b>	Network Time Protocol: documentação
<b>ntp-refclock</b>	Network Time Protocol: servidor para relógios de referência
<b>ntpdate</b>	Cliente para sincronismo de horário com servidores NTP

Para sincronizarmos nosso equipamento como client devemos instalar o pacote ntpdate, mas se desejar suprir sincronismo de horário em sua rede instale também o pacote ntp.

```
# aptitude install ntp ntpdate
```

Ao realizar a instalação o Debian, automaticamente, define o serviço ntp-server e o ntpdate para inicialização automática.

## Sincronizando o horário

A forma mais simples de sincronismo possível é executar o comando ntpdate apontando para um servidor externo. O Brasil conta, atualmente com 5 servidores públicos de NTP, entre eles o servidor da RNP ntp.cais.rnp.br.

```
# ntpdate ntp.cais.rnp.br
```

Para garantir que você esteja com horário correto é recomendado que você rode a atualização do sistema ao menos a cada duas horas (de hora em hora é o ideal). Isto pode ser feito com o cron com o exemplo abaixo criado com o comando crontab -e.

```
1 * * * * /usr/sbin/ntpdate ntp.cais.rnp.br
```

Se você tiver apenas o ntpdate instalado em seu equipamento você pode incluir outros servidores a serem checados na inicialização do equipamento, alterando o arquivo /etc/default/ntpdate como segue:

```
# servidores a serem checados no boot do equipamento
NTPSERVERS="br.pool.ntp.org south-america.pool.ntp.org pool.ntp.org"
#
# opções adicionais do ntpdate
NTPOPTIONS="-u"
```

Caso você esteja utilizando o servidor ntp-server, ele manterá o horário de seu equipamento sincronizado com os servidores de referência registrados no arquivo /etc/ntp.conf.

Da mesma forma que o comando ntpdate o ntp-server sincroniza o horário com o pool de servidores identificado pela URI pool.ntp.org. Para podermos ter uma maior confiabilidade deste sincronismo vamos alterar o arquivo de configuração para incluir, em ordem de preferência, servidores mais próximos.

O processo de sincronismo de horário pelo serviço NTP deve levar em consideração, ainda, a frequência do clock do equipamento onde está instalado. Esta verificação leva até dois dias após o serviço ser iniciado para ser considerada como correta fazendo com que o acerto de hora a partir de então seja mais confiável. Para evitar que o serviço NTP leve este tempo de

verificação todas as vezes que for reiniciado devemos definir e criar o arquivo ntp.drift.

Vamos habilitar também a disponibilização de sincronismo de horário em nossa rede local.

**Trecho do arquivo /etc/ntp.conf**

```
...  
driftfile /var/lib/ntp/ntp.drift  
...  
# servidores onde sincronizar o horário  
server br.pool.ntp.org  
server south-america.pool.ntp.org  
server pool.ntp.org  
...  
# anuncia o serviço ntp em nossa rede local pelo endereço de broadcast  
broadcast 192.168.2.255
```

Em seguida crie o arquivo que manterá a informação de freqüência do clock e reinicie o serviço ntp-server.

```
# touch /var/lib/ntp/ntp.drift → FREQUÊNCIA  
# /etc/init.d/ntp-server restart
```

## Referências

RNP – Rede Nacional de Ensino e Pesquisa (<http://www.rnp.br/ntp/>)

NTP: The Network Time Protocol (<http://www.ntp.org>)

Dicas Debian (<http://www.debian.org/doc/manuals/reference/ch-tips.pt-br.html>)



[www.green.com.br](http://www.green.com.br)