Apostila Básico I
LPI 101

Elaborada por:
Luis Fernando Moura de Albuquerque

Sumário

101: Arquitetura do sistema

101.1 Determinar e configurar opções de hardware

BIOS

A BIOS(Basic Input/Output System), contém todo o software básico, necessário para inicializar a placa-mãe, checa os dispositivos instalados e carregar o sistema operacional, o que pode ser feito a partir do HD, CD-ROM, pendrive. O BIOS inclui também o Setup, o software que permite configurar as diversas opções oferecidas pela placa.

Por definição, a BIOS é um software, mas, como de praxe, ele fica gravado em um chip espetado na placa-mãe. Na grande maioria dos casos, o chip combina uma pequena quantidade de memória Flash (256, 512 ou 1024 KB), o CMOS, que é composto por de 128 a 256 bytes de memória volátil e o relógio de tempo real.

Arquivos no diretorio /proc

O /proc é uma interface para estruturas de dados do **Kernel,** oferece informações sobre os processos e sistemas operativos que estão sendo executados em real-time, mas pode ser usado para checar alguma configuração do sistema, exemplo:

```
#cat /proc/cpuinfo.
 cat /proc/interrupts
           CPU0       CPU1
  0:         50          0   IO-APIC   2-edge      timer
  1:     284934     168953   IO-APIC   1-edge      i8042
  8:          2         51   IO-APIC   8-edge      rtc0
```

Este exemplo, nos podemos identificar as listas IRQs processadas desde a inicialização do sistema

Alguns arquivos encontrados no /proc, são:

- **asound** - diretório com informações sobre a placa de som da máquina.
- **meminfo** - apresenta informações sobre a memória do sistema.
- **mounts** - possui a lista dos sistemas de arquivos montados (veja o comando **mount**). Ele normalmente possui mais informações do que o arquivo **/etc/mtab**
- **net** - diretório com informações sobre as conexões.
- **sys**- diretório com informações sobre os parâmetros usados pelo sistema
- **vmstat** - possui informações estatísticas sobre a memória virtual usada pelo **kernel**.
- **interrupts-** possui a lista das interrupções (IRQs) processadas desde a inicialização do sistema

- **dma** - exibe informações sobre o DMA (Direct Memory Access).

lsusb

O comando **lsusb** serve para exibir informações sobre os dispositivos USB

Exemplo:

```
$ lsusb
Bus 002 Device 004: ID 090c:37c0 Silicon Motion, Inc. - Taiwan (formerly Feiya
Technology Corp.) Silicon Motion Camera
Bus 002 Device 006: ID 046d:c534 Logitech, Inc. Unifying Receiver
Bus 002 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
```

A opção -v mostra de uma forma mais detalhada de todos os dispositivos USB:

```
$ lsusb -v
Bus 002 Device 004: ID 090c:37c0 Silicon Motion, Inc. - Taiwan (formerly Feiya
Technology Corp.) Silicon Motion Camera
Couldn't open device, some information will be missing
Device Descriptor:
 bLength                18
 bDescriptorType         1
```

Para obter informações de um dispositivo USB, use as seguintes combinações do comando:

```
#lsusb -v  -d  090c:37c0
```

Onde o **090c:37c0** representa o ID

lsmod

O comando **lsmod** lista todos os módulos que estão atualmente carregados na memória,

```
$$ lsmod
Module                  Size  Used by
fuse                  102400  2
nf_conntrack_netlink    36864  0
xt_addrtype            16384  2
br_netfilter           24576  0
```

Onde:

Modules: Nome dos modulos
Size: Tamano do modulo
Used by: Quantas instancias o modulo esta usando

modinfo

O comando **modinfo** mostra as informações do módulo:

```
# modinfo br_netfilter
filename:       /lib/modules/4.4.0-38-generic/kernel/net/bridge/br_netfilter.ko
description:    Linux ethernet netfilter firewall bridge
author:         Bart De Schuymer <bdschuym@pandora.be>
author:         Lennert Buytenhek <buytenh@gnu.org>
license:        GPL
srcversion:     96CA697766FA5E4BE938D19
depends:        bridge
intree:         Y
vermagic:       4.4.0-38-generic SMP mod_unload modversion
```

modprobe

O comando **modprobe** permite adicionar e remover os módulos e dependências de uma só vez. O comportamento do **modprobe** é modificado pelo arquivo /etc/modules.conf e precisa ser rodado como root, exemplo:

Para carregar quaisquer dependência do modulo *fuse* e o modulo use:

```
#modprobe fuse
```

Para exibir todos os comandos na tela, enquanto o modprobe os executa, use a opção -v

```
# modprobe -v fuse
```

insmod

O comando **insmod** também serve para carregar os módulos do kernel, no entanto não resolve suas dependências.

rmmod

O comando **rmmod** seguido do nome do módulo do kernel descarrega somente os módulos que não estão em uso e não são dependência de outros módulos.

Para remover um modulo utilize primeiro o **lsmod** e o **modinfo** para saber as informações do modulo e o caminho.

```
# modinfo iscsi_tcp

filename:       /lib/modules/4.4.0-38-generic/kernel/drivers/scsi/iscsi_tcp.ko
license:        GPL
description:    iSCSI/TCP data-path
author:         Mike Christie <michaelc@cs.wisc.edu>, Dmitry Yusupov
<dmitry_yus@yahoo.com>, Alex Aizman <itn780@yahoo.com>
srcversion:     815099BA4EBDA9B6B43797E
```

depends: libiscsi,libiscsi_tcp,scsi_transport_iscsi
intree: Y
vermagic: 4.4.0-38-generic SMP mod_unload modversions
parm: max_lun:uint
parm: debug_iscsi_tcp:Turn on debugging for iscsi_tcp module Set to 1 to turn on, and zero to turn off. Default is off. (int)

Execute rmmod com o caminho do modulo para remover:

#rmmod /lib/modules/4.4.0-38-generic/kernel/drivers/scsi/iscsi_tcp.ko

lspci

O **lspci** exibe informações sobre dispositivos PCI, exemplo:

$lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)

Para saber mais informações sobre o dispositivo PCI basta executar:

#lspci -s '00:00.0' -v
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
Flags: fast devsel

Para saber informações detalha de todos os módulos, utilize a opção -*v:*

/sys/

O sistema de arquivos **sysfs** é considerado uma fusão dos sistemas de arquivos **proc**, **devfs**, e **devpty**. O sistema de arquivos **sysfs** enumera os dispositivos e barramentos conectados a um sistema em uma hierarquia que pode ser acessada do espaço de usuário. O **sysfs** é projetado para lidar com opções específicas de dispositivos e drivers que anteriormente eram mantidas no **/proc/**, e trazia a inclusão dinâmica de dispositivos oferecida anterioemente pelo **devfs**.

Alguns subdiretorio que tem no /sys:

- /devices/ :Este diretório contém o todos os subcanais detectados pelo kernel do Linux
- /bus/: Este diretório contém ss dispositivos CCW são acessados através de palavras de comando de canal
- /class/: Este diretório contém diretórios que agrupam dispositivos similares, como ttys, drives de fita SCSI, dispositivos de rede e outros dispositivos diversos.
- /block/: Este diretório contém diretórios para cada dispositivo de bloco do sistema
-

/dev/

O diretorio /dev contém arquivos usados para acessar dispositivos (periféricos) existentes no computador. Cada dispositivo de E/S é associado a um nome de caminho, que fica no /dev. Exemplo de um dispositivo /dev/hd1, uma impressora pode ser /dev/lp , rede pode ser /dev/net e o famoso /dev/null server para descarta automaticamente todo o input que recebe

101.2 Sistema de Boot

MBR e BootLoader

A MBR é onde fica alocada o gerenciador de boot do sistema e a tabela de particionamento do hd, os primeiros 512 bytes no primeiro setor do HD, Destes 512 bytes 446 bytes são reservados para o setor de boot e o restante fica para a tabela de particionamento.

Dentro da MBR a BIOS vai procurar pelo bootloader (carregador de boot) para carregar o sistema. No Linux os mais utilizado atualmente são o GRUB, mas também temos o LILO.

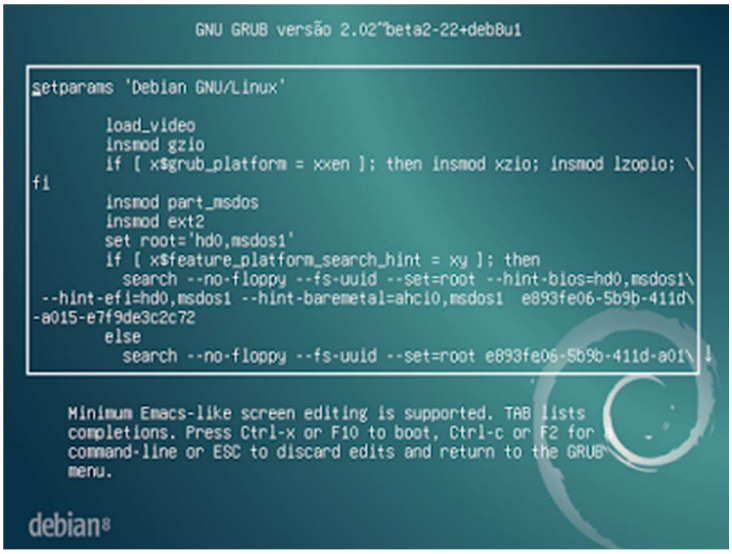
O bootloader carrega arquivos do kernel, nele também é possível passar informações para kernel ao carrega-lo.

Informações sobre memória disponível, cpus, controloador de inicialização, resolução de video, são opções que podemos passar antes do kernel ser carregado.

Alguns parâmetros que podemos passar para o kernel na tela de bootloader.

- **acpi** - liga/desliga o suporte a acpi ***-acpi=off;***
- **init** - define outro programa para executar no lugar do /sbin/init ***-init=/bin/bash;***
- **mem**- serve para definir a quantidade de memória RAM para o sistema ***-mem=1024;***
- **maxcpus**- número máximo de processadores visiveis para o sistema ***-maxcpus=2;***
- **quiet** - não exibe as informações durante o boot ***-quiet;***
- **vga**- seleciona um modo de vídeo ***-vga=773;***
- **root**- serve para definir uma partição raiz diferente da pré-determinada ***-root=/dev/sda3;***
- **ro/rw**- monta a partição como somente leitura (ro) ou leitura e escrita (rw) r***o***

Para passarmos esses parâmetros, teclamos a tecla "e". Uma imagem parecida*:*



SysVinit

Após carregar o bootloader e iniciar o sistema, o kernel identifica o hardware da máquina, identifica os dispositivos, carrega os drivers que estão compilados dentro do kernel e entrega o sistema para o processo init, que é o responsável para disparar os processos que inicializará o restante do sistema**.**

O SysVinit utiliza o método de inicialização dos runlevels, que são os níveis que o sistema Linux utiliza para iniciar.

DMESG

O comando dmesg (display message ou driver message, em português mostrar mensagem ou mensagem de driver) é um comando na maioria dos sistemas operacionais do tipo Unix que imprime o buffer de mensagens do kernel.

Se quiser ter uma informação mais precisa, basta utilizarmos o **grep** para filtrar a parte que nos interessa:

$ dmesg| grep “dev”

101.3 Alterar runlevels,desligar e reiniciar sistema operacional

Runleveis no linux-gnu

O runlevel indica o modo de operação atual da máquina, definido quais serviços e recursos devem permanecer ativos. O runlevel pode ser alterado a qualquer momento pelo root, através do comando telinit

Os runlevels são numerados de 0 a 6, onde:

- 0: Sistema desligado

- 1: Usuário único(manutenção)
- 2- Multi usuário sem rede
- 3 – Multi usuário com rede;
- 4 – Não usado
- 5 - Multi usuário completo com GUI
- 6 – Reiniciar

Além do básico, o uso de níveis de execução é diferentes nas distribuições e os únicos runlevels comuns a toda distribuição são 0,1 e 6 .

Para descobrir o runlevel que o usuário esta execute :

```
$runlevel
N 5f
```

Para alterar para o modo single-user (runlevel 1) você pode executar:

```
# init 1
```

Esse tipo de runlevel é executando quando o administrador pretende fazer alguma administração na distribuição linux e quer que ninguém acesse e ira receber uma mensagem de negação:

```
$ ssh user @192.168.1.5
ssh_exchange_identification: Connection closed by remote host
```

/etc/inittab

Quando o sistema é iniciado, o processo init(primeiro processo do sistema) levará o sistema a um runlevel padrão, o qual é configurado no arquivo /etc/inittab, na linha:

```
# cat /etc/inittab
# Default runlevel. The runlevels used by RHS are:
#0 - halt (Do NOT set initdefault to this)
#1 - Single user mode
#2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#3 - Full multiuser mode
#4 - unused
#5 - X11
#6 - reboot (Do NOT set initdefault to this)
#
id:5:initdefault:
```

Onde:

- **id**: é um identificador exclusivo de um a quatro caracteres
- **runlevels**: lista os níveis de execução nos quais a ação deste ID deve ser realizada
- **action**: descreve qual das várias ações possíveis deve ser realizada
- **process**: diz qual processo, se for o caso, deve ser executado quando a ação nesta linha é realizada

init e telinit

Os comandos init e telinit permitem colocar o sistema em um nível de execução desejado, de 0 a 6, ou S e s, que são equivalentes ao runlevel 1. O telinit é um link para o init como podemos ver na figura abaixo:

Para entrar em modo usuário único você pode executar:

```
# init 1
```

OU

```
# init S
```

Para reler o arquivo de configuração no /etc/inittab você pode executar:

```
#init q
```

Com telinit seria da mesma forma:

```
#telinit 1
```

systemd, /etc/systemd/ /usr/lib/systemd/

O systemd é um gerenciador de sistema e serviços para Linux, systemd é compatível com os scripts de inicialização SysV e LSB. Ele prove paralelização agressiva, utilizando socket e ativações D-Bus para iniciar serviços, oferece arranque de serviços sob demanda, mantém auditoria de processos através dos cgroups do Linux, suporta snapsotting e restauração de estado de sistema, mantém pontos de montagem mount e automount e implementa um elaborado sistema transacional baseado em dependências para controle lógico dos serviços.

O **systemd** possui as seguintes características:

- Para cada processo criado, ele controla: O ambiente, limites de recursos, diretório atual e root, umask;
- Mantem pontos de montagem(mount) e automount;
- Paralelismo agressivo utilizando socket.

Ferramentas utilizada pelo systemd:

- systemctl: utilizado para inspecionar e controlar o estado do sistema systemd e gerenciador de serviços.
- **systemd-cgls:** exibe recursivamente o conteúdo da árvore de hierarquia de um determinado grupo de controle do Linux.
- **systemadm**: interface gráfica pra gerenciamento de serviços no systemd que permite inspecionar e controlar o systemd. É parte do pacote systemd-gtk. Não utilize a não ser que você seja um desenvolvedor, pois está em uma versão não muito madura.

Diretório do systemd

Os arquivos responsáveis pelos serviços no Systemd localizam em **/usr/lib/systemd/** No entanto, como cada administrador pode — e deve, como no caso das montagens de sistemas de arquivos — criar os seus próprios serviços, há um diretório específico para esse tipo de personalização: **/etc/systemd/.**

Uma listagem do diretório **/usr/lib/systemd/** logo após a instalação revela algo em torno de 140 arquivos e diretórios. Estes são serviços que o pacote já traz configurados e instalados para você. Você pode se basear neles para escrever os seus próprios serviços

Arquivos de serviços

Um fato muito positivo sobre o Systemd é que a documentação já está muito bem feita. O pacote do Systemd já traz as páginas de manual de cada tipo de serviço. Experimente:

```
$ man -k systemd
abrt-dump-journal-core (1) - Extract coredumps from systemd-journal
abrt-dump-journal-oops (1) - Extract oops from systemd-journal
abrt-dump-journal-xorg (1) - Extract Xorg crashes from systemd-journal
gnome-logs (1)       - log viewer for the systemd journal
httpd.service (8)    - httpd unit files for systemd
httpd.socket (8)     - httpd unit files for systemd
init (1)             - systemd system and service manager
journalctl (1)       - Query the systemd journal
libnss_resolve.so.2 (8) - Provide hostname resolution via systemd-resolved.service
loginctl (1)         - Control the systemd login manager
lvm2-activation-generator (8) - generator for systemd units to activate LVM2 volumes on
boot
machinectl (1)       - Control the systemd machine manager
```

Ativando o serviço

O systemd procura arquivos de units em vários locais, e em geral o melhor local para criarmos scripts personalizado é em /**etc/systemd/system/**. Aqueles instalados por pacotes ficam em **/usr/lib/systemd/,** exemplo:

```
# cat /etc/systemd/system/syslog.service
[Unit]
Description=System Logging Service
Requires=syslog.socket
Documentation=man:rsyslogd(8)
[Service]
```

```
Type=notify
ExecStart=/usr/sbin/rsyslogd -n
StandardOutput=null
Restart=on-failure
[Install]
WantedBy=multi-user.target
Alias=syslog.service
```

Onde:

- **ConditionPathExists:** define que o serviço não será iniciado.
- **ExecStart:** define o comando que inicia o daemon.
- **WantedBy** umpre nesse caso a função do runlevel no SysV init.

Para ativar o serviço você pode utilizar o comando:

```
# systemctl enable syslog.service
```

Para desativar um serviço:

```
$ systemctl disabled  syslog.service
```

Para listar todos os serviços você pode usar:

```
$systemct
```

Para iniciar, parar, reiniciar ou verificar o status execute:

```
$ systemctl start/stop/restart/status <serviço>.service
```

O diretorio /etc/init.d/ é onde contém os scripts de start e stop dos serviços, por exemplo **/etc/init.d/httpd** é um shellscript que contém alguns parâmetros de start e stop, para iniciar ou para os serviços que estão no /etc/init.d/ você executa:

```
$ /etc/init.d/httpd start
$/etc/init.d/http stop
```

Ou

```
$ /etc/init.d/httpd restart/reload
```

wall

O comando **wa**ll envia uma mensagem a todos os usuários do sistema. Este comando faz a leitura de um arquivo ou entrada padrão e escreve o resultado em todos os terminais onde existem usuários conectados.

```
$ wall  “mensagem  ”
```

Somente o usuário root pode utilizar este comando.

```
# wall  [arquivo]
```

102: Instalação do Linux e administração de pacotes.

102.1 Dimensionar partição de discos

Visão geral sistema de arquivo

Sistema de arquivos contém arquivos organizados em um disco ou outro dispositivo de armazenamento em bloco em diretórios. Assim como com muitos outros sistemas, os diretórios em um sistema Linux pode conter outros diretórios chamados de subdiretórios.

O Kernel inicia um processo de montagem do sistema de arquivos pela partição da unidade de disco rígido, como o /(raiz). Depois de montado o raiz, os diretórios contidos nesse dispositivo poderão ser pontos de montagem para outros dispositivos.

Ordem de montagem dos sistemas de arquivo a partir do boot:

- O carregador de boot (Grub ou Lilo) carrega o kernel e transmite as informações sobre a localização do dispositivo raiz;
- Com a raiz montada, os demais dispositivos são montados conforme as instruções encontradas no arquivo /etc/fstab.

Os diretórios necessários no / são:

- **bin :** Binario essenciais de comando;
- **boot**: Arquivo estático do loader de boot
- **dev**: arquivo de dispositivos
- **etc**: Configuração do sistema especifica para host
- **li**b: Bibliotecas compartilhadas e módulos kernel essenciais
- **medi**a: Ponto de montagem para mídia removível
- **opt**: Pacotes de software de aplicativo complementar
- **mnt**:Ponto de montagem para montar um sistema de arquivos temporariamente
- **sbin**: Binários essenciais de sistema
- **srv**: Dados para serviços fornecidos por esse sistema
- **tmp**: Arquivos temporários
- **usr**: Hierarquia secundária
- **var**: Dados variáveis

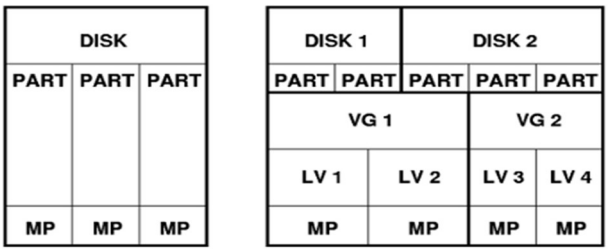
Uma partição swap é identificada pelo código hexadecimal 82 (Ox82) , atribuído na sua criação. Geralmente, o tamanho da partição swap corresponde ao dobro da quantidade de memória RAM presente no sistema. Essa regra, apesar de não

ser prejudicial, não fará diferença em sistemas com vários GB's de memória RAM. Apesar de não ser comum, é possível utilizar mais de uma partição de swap no mesmo sistema.

É recomendável criar partições de swap nos dispositivos mais velozes. Se possível, em dispositivos distintos daqueles cujos dados sejam frequentemente acessados pelo sistema.

Gerenciamento de volume

O gerente lógico de volume (ou LVM) para Linux permite combinar múltiplos dispositivos físicos de armazenamento em um único *grupo de volume*. Por exemplo, você pode adicionar uma partição a um grupo de volume existente, em vez de ter de cavar um espaço adjacente grande o suficiente para o sistema de arquivos desejado. A figura abaixo mostra como funciona uma LVM.



Alguns recursos do LVM.

- PV = Phisical Volume: A partição física do disco.
- PE = Phisical Extends: Pequeno pedaço de um VG.
- VG = Volume Grupo: Responsável por agrupar um ou mais partições/discos, vários PVs, um volume físico.
- LV = Logical Volume: Partições lógicas de um VG.

Procedimentos:

Verificar se possui o LVM instalado na máquina:

$**lvscan**

Criar o LVM

$ **pvcreate /dev/hdd2**

Verificar se está criado:

 # pvdisplay

Criando volume Group (VG)

Criar um volume físico para cada HD ou partição no computador, um passo obrigatório para criação do volume lógicos.

 # vgcreate study /dev/hdd2

Para criar um volume lógico com 700M com o nome de arch, dentro do VG study, utilize o seguinte comando:

 # lvcreate -L 700M arch study

Verificação do LV:

 # vgdisplay

Verificação GV:

 # vgscan

102.2 Instalando um gerenciador de inicialização

Processo de inicialização:

Quando o PC é inicializado, o codo denominado BIOS é armazenado em memória não volatil. como ROM, EEPROM ou memória flash. Quando o PC é ligado ou reinicializado, esse código é executado. Em geral, ele realiza um autoteste inicial (POST) para verificar a máquina. Por fim, ele carrega o primeiro setor a partir do registro de boot mestre (MBR) na unidade de boot.

O MBR de unidade de disco rígido padrão usado pelos sistemas operacionais MS DOS, PC DOS e Windows® verifica a tabela de partições para localizar uma partição primária na unidade de boot marcada com *ativa*, carrega o primeiro setor a partir dessa partição e passa o controle para o começo do código carregado. Essa nova porção de código é também conhecida como o registro de boot de partição.

Atualmente, a maioria das distribuições Linux utiliza o Grub como carregador de boot. A versão tradicional do Grub, chamada legacy, está gradualmente sendo substituída por sua implementação mais moderna, chamada Grub2.

GRUB Legacy

O GRUB (Legacy) tem um arquivo de configuração que geralmente é armazenado em /boot/grub/grub.conf. Se o seu sistema de arquivos suportar links simbólicos, como a maior parte dos sistemas de arquivos Linux, você provavelmente terá **/boot/grub/menu.lst** como link simbólico para **/boot/grub/grub.conf**.

O comando grup (/sbin/grub ou, em alguns sistemas, /usr/sbin/grub) é um shell pequeno, mas razoavelmente poderoso que suporta vários comandos para instalar o

GRUB, sistemas de boot, localizar e exibir arquivos de configuração e tarefas similares.
Principais opções globais de /boot/grub/menu.lst :

- default: opção padrão a ser inicializada (começando por O) ;
- timeout: tempo de espera para iniciar o boot, em segundos;
- Opções individuais para cada opção de boot:
- tide: nome para o item;
- root: localização do carregador de segundo estágio e do kernel (hd0,0 equivale /dev/hda de acordo com tipo de dispositivo instalado) ;
- kernel: caminho para o kernel {relativo à opção root) ;
- ro: montar inicialmente em modo somente leitura;
- initrd: caminho para a imagem initrd.

GRUB2

O GRUB 2 foi desenvolvido novamente a partir do zero para tornar-se significativamente mais modular e portátil. Ele é destinado a diferentes arquiteturas e métodos de boot e tem muitos novos recursos. Se você estiver acostumado ao GRUB e começar a usar o GRUB 2, descobrirá que ele é completamente diferente e, provavelmente, terá muitas surpresas.

Dentre as melhorias trazidas pelo GRUB2, destacam-se:

- Suporte a scripts com instruções condicionais e funções;
- Carregamento dinâmico de módulos;
- Modo de recuperação;
- Menus personalizados e temas;
- Carregar LiveCD a partir do disco rígido;
- Suporte a plataformas diferentes da x86;
- Suporte universal a UUIDs.

O coração do GRUB 2 é formado por um kernel multiboot (/boot/grub/core.img) e um arquivo de configuração (/boot/grub/grub.cfg). Eles serão gerados para você se você executar **grub-install** e definir o destino como seu MBR.

Exemplo: grub-install /dev/sda

Grub-mkconfig

O arquivo de configuração do GRUB 2 geralmente é /boot/grub/grub.cfg. Diferentemente do GRUB Legacy, normalmente não é recomendável que você edite

esse arquivo. É possível construí-lo usando o comando grub-mkconfig ou update-grub, que é um front-end para grub-mkconfig. Esses comandos procuram configurações gerais, como plano de fundo, tempos limite e assim por diante, em /etc/default/grub

102.3 Controlar bibliotecas compartilhadas:

ldd

O comando ldd imprime as bibliotecas compartilhadas necessárias por programa ou biblioteca compartilhada especificada na linha de comando, o ldd é um script que executa o programa passado como argumento e não deve ser usado com binários não confiáveis. Se alguma biblioteca compartilhada estiver faltando para qualquer programa, este programa não executará.

Exemplo:

```
# ldd /usr/bin/at
linux-vdso.so.1 =>  (0x00007fff25ba6000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ff1463c5000)
/lib64/ld-linux-x86-64.so.2 (0x00007ff14699c000)
```

Configuração do caminho para bibliotecas

/etc/ld.so.conf

O arquivo /etc/ld.so.conf mantém o caminho para alguns diretórios de bibliotecas que não sejam genéricas, ou seja, não se encontram dentro do diretório /lib. Normalmente esse arquivo indica as bibliotecas dentro do /usr e diretórios afins. O **LD_LIBRARY_PATH** indica localização de bibliotecas compartilhadas.

Exemplo do conteúdo do arquivo ld.so.conf:

```
#cat /etc/ld.so.conf
 #libc default configuration
/usr/local/lib
```

Esse arquivo somente tem efeito após o comando ldconfig.

ldconfig

Para o sistema reconhecer as bibliotecas existentes e ainda conseguir acessá-las existe o comando ldconfig que permite, além de gerenciar as bibliotecas, gerenciar os diretórios indicados em arquivos de configuração.

O comando ldconfig realiza esse gerenciamento, o comando deve ser invocado a cada nova biblioteca instalada no sistema.

102.4 Utilizando sistema de pacotes distribuição debian

O sistema de pacotes Debian - utilizado por diversas distribuições, como Ubuntu torna possível a instalação de praticamente todos os programas disponíveis para Linux sem que o usuário precise preocupar-se com bibliotecas ou com outros programas necessários. Cada pacote de programa, com extensão . deb, traz internamente as informações sobre todos os programas e bibliotecas dos quais depende.

dpkg

O dpkg (Debian Package) é o programa responsável pelo gerenciamento de pacotes em sistemas Debian e derivados. Sua operação é feita em modo texto e funciona através de comandos, assim caso deseje uma ferramenta mais amigável para a seleção e instalação de pacotes, prefira o dselect (que é um front-end para o dpkg) ou o apt .

Instalação via dpkg

Use o comando: dpkg -i [NomedoPacote] (ou --install) para instalar um pacote em seu sistema:

```
# dpkg -i man-db_2.7.0.2-5_amd64.deb
```

Remoção:

Invocando o dpkg com a opção -r ou --remove, seguida pelo nome de um pacote, remove o pacote.Para remover completamente tudo associado a um pacote, use a opção -P ou --purge

Exemplo:

```
$dpkg -r debian-cd
```

Para verificar o status de um pacote tzdata:

```
$dpkg -s gcl tzdata
```

dpkg-reconfigure

Reconfigura pacotes ".deb" após terem sido instalados utilizando o *debconf* (sistema de configuração de pacotes ".deb"). Esse comando fará perguntas para reconfigurar o pacote.

Exemplo:

$ **dpkg-reconfigure ssh**

Locais dos pacotes:

/etc/apt/sources.list

Este arquivo contém os locais onde o apt encontrará os pacotes, a distribuição que será verificada a seção que será copiada (main, non-free, contrib, non-US).

Se desejar usar sempre uma distribuição estável, modifique o arquivo sources.list e coloque stable como distribuição. Assim que a nova versão for lançada, os links que apontam de stable para Woody serão alterados apontando para buster e você terá seu sistema atualizado. Abaixo um exemplo de arquivo de configuração do /etc/apt/source.list:

```
$ cat /etc/apt/sources.list
deb http://archive.ubuntu.com/ubuntu xenial main restricted
deb-src http://archive.ubuntu.com/ubuntu xenial main restricted
```

Note que tudo especificado após o nome da distribuição será interpretado como sendo as seções dos arquivos (main, non-free, contrib, non-US). As linhas são processadas na ordem que estão no arquivo.

Apt-get aptitude

APT é um projeto amplo, cujos planos originais incluem uma interface gráfica. Ele é baseado numa biblioteca que contém as aplicações principais, e o apt-get é a primeira interface — em linha de comando — que foi desenvolvida dentro do projeto. O apt é uma segunda interface baseada em linha de comando fornecida pelo APT que supera alguns erros de projeto do apt-get. A sintaxe de linha de comando do apt-get e aptitude são muito semelhantes.

Para instalar algum pacote você pode utilizar:

```
$ apt-get install openssh
```

OU

```
$ aptitude install openssh
```

apt-cache

O comando apt-cache pode apresentar grande parte das informações armazenadas no banco de dados interno do APT. Esta informação é uma espécie de cache, pois é recolhida de diferentes fontes, listadas no arquivo sources.list.

O comando apt-cache pode buscar pacotes baseado em palavras-chave com apt-cache search palavra-chave. Também pode mostrar os cabeçalhos das versões

disponíveis dos pacotes com apt-cache show pacote. Este comando fornece a descrição do pacote, suas dependências, o nome de seu mantenedor, etc.

Exemplo:

```
$ apt-cache show ntp
Package: ntp
Priority: optional
Section: net
```

102.5 Utilizando sistema de pacotes usando RPM e YUM:

RPM

A Red Hat lançou o RPM em 1995. Hoje o RPM é o sistema de gerenciamento de pacote usado para pacotes no Linux Standard Base (LSB). As opções de comando rpm estão agrupadas em três subgrupos para:

- Consultar e verificar pacotes
- Instalar, atualizar e remover pacotes
- Realizar funções diversas

Para instalar um pacote RPM, você precisa executar:

```
$rpm -i nome_do_pacote
```

Para desinstalar um pacote utilize o parametro -U:

```
$rpm -U nome_do_pacote.rpm
```

rpm2cpio

Uma das formas de listar o conteúdo de um pacote RPM é utilizando o comando rpm2cpio. Esse comando simplesmente mostra na saída padrão o conteúdo do arquivo RPM no formato cpio. Dessa forma, podemos listar todo o conteúdo de um arquivo ou mesmo extrair algum arquivo específico.

YUM

O YUM adiciona atualizações e gerenciamento de pacote, incluindo gerenciamento de dependências, a sistemas RPM. Além de entender os pacotes instalados no sistema, o YUM, assim como a Debian Advanced Packaging Tool (APT), trabalha com repositórios, que são coleções de pacotes, normalmente acessível através de uma conexão de rede.

Para instalar ou remover um pacote utilize:

```
# yum remove nome_do_pacote
```

Para instalar um pacote utilize o install:

```
#yum install nome_do_pacote
```

Para exibir informações do pacote utilize:

```
# yum list nome_do_pacote
```

yumdownloader

Embora o yum recupere automaticamente pacotes de repositórios, você pode querer fazer o download de RPMs e salvá-los, talvez para instalá-los em um sistema sem rede ou para examinar seu conteúdo, ou por qualquer outro motivo. É possível usar o comando yumdownloader.

```
$ yumdownloader --resolve  openssh
```

/etc/yum.conf e /etc/yum.repos.d/

O ponto de partida do **yum** é o diretório /etc/yum.repos.d/, que normalmente contém vários arquivos repo. Esse é o local padrão para repos, mas outros locais podem ser especificados no arquivo de configuração do YUM, que normalmente é o /etc/yum.conf.

Um arquivo repo típico é dividido em três seções, uma para pacotes normais, uma para pacotes de depuração e outra para pacotes de origem. Normalmente, haverá várias cópias dos pacotes de uma distribuição disponíveis em diferentes locais, ou espelho Assim, o arquivo repo diz ao yum onde encontrar a lista mais recente de espelhos para cada seção:

```
$cat /etc/yum.repos.d/fedora.repo
[fedora]
name=Fedora $releasever - $basearch
failovermethod=priority
#baseurl=http://download.fedoraproject.org/pub/fedora/linux/releases/
$releasever/Everything/$basearch/os/
metalink=https://mirrors.fedoraproject.org/metalink?repo=fedora-
$releasever&arch=$basearch
enabled=1
metadata_expire=14d
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-$releasever-$basearch
skip_if_unavailable=False
```

103: Comandos GNU e Unix

103.1 Trabalhando com linha de comando

O bash shell é um dos vários shells disponíveis para Linux. Ele é também conhecido como shell Bourne again, após Stephen Bourne, o criador de um shell anterior (/bin/sh). Bash é bastante compatível com sh, mas contém muitas melhorias nos recursos de função e programação.

O shell é um programa que aceita e executa comandos. Ele também suporta construções de programação, permitindo a criação de comandos complexos a partir de porções menores. Esses comandos complexos, ou scripts, podem ser salvos para se tornar novos comandos dentro do seu próprio direito.

Os shells usam três fluxo de E/S padrão:

- stdin: é o fluxo de entrada padrão, que fornece entrada para comandos;
- stdout: é o fluxo de saída padrão, que mostra a saída dos comandos;
- stderr: é o fluxo de erro padrão, que mostra a saída de erro dos comandos.

Alguns prompts que existem no shell:

$: usuário simples
#: superusuario

echo

O comando **echo** imprime (ou repete) seus argumentos para o terminal, conforme mostrado abaixo:

```
 $ echo Hello
Hello
$ echo “Hello Word”
Hellow Word
```

O comando echo tem inúmeras opções o echo terá um caractere de nova linha à esquerda para a saída. Use a opção -n para suprimir isso. Use a opção -e para permitir que certos caracteres de escape com barra invertida tenham significado especial, alguns deles são mostrados

- \a: Alerta
- \b: Backspace
- \c: Supressão de noval inha
- \f: Alimentação de formulario
- \n: noval inha
- \r: retorno de linha
- \t : tabulação horizontal

Exemplo:

```
$echo -e Escape\\n\\tmetacaractere
Escape
        metacaractere
```

Operadores de controle e metacaracteres de shell de bash

Bash tem inúmero **metacaracteres**, que, quando não estão entre aspas, também funcionam para dividir a entrada em palavras. Além de um valor em branco, eles também são:

|, &, ; ,(,), <, >

A nova linha e determinados metacaracteres ou pares de metacaracteres também funcionam como **operadores de controle**. Elas são:

||, &&,& “;,” “;;”,|,(,)

Variável de ambiente

As variáveis de ambiente e de shell têm um **nome**. A referência ao valor de uma variável é feita pela prefixação de seu nome com '$', algumas variáveis:

- **USER**:O nome do usuário conectado
- **UID**:O ID de usuário numérico do usuário conectado
- **HOME**:O diretório inicial do usuário
- **PWD**:O diretório de trabalho atual
- **SHELL**:O nome do shell
- **$** O ID de processo (ou PID) do processo de bash shell em execução (ou outro processo)
- **PPID** O ID de processo do processo que iniciou este processo (ou seja, o ID do processo pai)
- **?:** O código de saída do último comando

Exemplo de variáveis PWD:

```
$echo $PWD
/root
```

env

O comando env sem nenhuma opção ou parâmetro mostra as variáveis de ambiente atuais. É possível usá-lo também para executar um comando em um ambiente personalizado.

Exemplo:

```
$env
HOSTNAME=drogon
SELINUX_ROLE_REQUESTED=
TERM=xterm
SHELL=/bin/bash
```

export

Ao criar uma variável de shell, *exporte-a* ao ambiente para que fique disponível a outros processos iniciados a partir desse shell. As variáveis exportadas **não** ficam disponíveis a um shell pai. Use o comando export para exportar um nome de variável.

set e unset

Os comandos set e unset definem e removem o valor de uma variável, sem argumentos mostra toda as variáveis definidas

man

O comando man é a fonte principal (e tradicional) de documentação é composta pelas páginas manuais, que podem ser acessadas via comando man.

Praticamente todos os comandos e arquivos de configuração no Linux acompanha um manual. Esse manual está acessível por intermédio do comando man. Para ver o manual, basta executar:

man ssh

Para acessar um item em uma sessão específica, o número da seção precede o nome do item. Por exemplo, acessar a seção número 7 do man:

$ man 7 man

uname

O comando **uname** imprime informações sobre o sistema e seu kernel Algumas opções que pode ser passada pelo uname:

- **-s:** Imprime o nome do kernel. Este será o padrão se nenhuma opção estiver especificada.
- **-n:** Imprime o nome do nó ou o nome do host.
- **-r:** Imprime o release do kernel. Esta opção é geralmente usada com comandos de manipulação de módulo.
- **-v:** Imprime a versão do kernel.
- **-m:** Imprime o nome do hardware (CPU) da máquina.
- **-o:** Imprime o nome do sistema operacional.
- **-a:** Imprime todas as informações acima.

Histórico do comando:

O comando history server para mostrar um histórico dos comandos executados e são armazenados no .bash_history. É possível desativar o armazenamento utilizando o comando set +o history e reativar usando set -o history. Uma variável de

ambiente chamada HISTSIZE informa ao bash quantas linhas de histórico ira manter,:

```
$echo $HISTSIZE
1000
```

Algumas opções que o history fornece:

- **history** *N: Mostra as últimas N linhas do histórico*
- **history -d** *N:* Exclui a linha *N* do histórico; isso poderá ser feito se, por exemplo, a linha tiver uma senha.
- **!!:** O comando mais recente
- **!*N:*** O *N*º comando de histórico
- **!-*N:*** O comando que está *N* comandos atrás no histórico (!-1 é equivalente a !!)
- **!#:** O comando atual sendo digitado
- **!***string:* O comando mais recente que começa com *string*
- **!?***string***?:** O comando mais recente que contém *string*

103.2 Processando fluxo de texto usando comandos de filtros

cat

Mostra o conteúdo de um arquivo binário ou texto. O comando cat trabalha com arquivos texto. Use o comando zcat para ver diretamente arquivos compactados com gzip.

head

Mostra o começo de arquivos. Por padrão, as primeiras dez linhas são mostradas.

tail

Mostra o final d e arquivos. Por padrão, as últimas dez linhas são exibidas.

expand

Substitui espaços de tabulação (TABs) por espaços simples

unexpand

Substitui dois ou mais espaços simples, em um texto, por espaços de tabulação

fmt

Formata um texto para determinado número de caracteres por linha

less

Se a necessidade for simplesmente inspecionar o conteúdo de um arquivo de texto, sem alterar seu conteúdo, basta utilizar um comando paginador, como o more

ou o less. O more é mais tradicional, mas não permite voltar à medida que se avança o texto. Já o less permite voltar e fazer buscas no texto.

nl

Numera linhas, como o comando cat -b

od

Usado para converter entre diferentes formatos de dados, como hexadecimal e binário.

pr

Divide o arquivo para impressão. O padrão é 66 linhas por 72 caracteres de largura, modificados por - 1 e - w, respectivamente.

sort

Organizar os dados dentro de um arquivo, de acordo com a necessidade do usuário, e exibir esses dados classificados na saída padrão

split

Este comando divide um arquivo em arquivos menores.

$split -l 500 teste.txt novo

uniq

Mostra o conteúdo de arquivos, suprimindo linhas sequenciais repetidas.

wc

Conta o número de linhas, caracteres e palavras a partir das opções 1 -w e – c.

paste

O comando pasta é usado para concatenar dois arquivos. Ao usarmos o comando paste teremos como saída a junção das linhas dos arquivos:

$ paste test.txt arquivo.txt

cut

O comando *cut* pode ser usado para mostrar apenas seções específicas de um arquivo de texto ou da saída de outros comandos. Ele lê o conteúdo de um ou mais arquivos, ou a saída de comandos, e apresenta como resultado uma coluna.

Para separar por campo, a opção - d especifica o caractere delimitador e - f informa a posição do campo. Por exemplo, para mostrar os campos da posição 1 e 3 do arquivo /etc/group, que estão separados por " : "

Exemplo:

$ cut -d":" -f1,3 /etc/group

join

O comando join (unir) concatena registros de dois arquivos de texto baseado em índices comuns entre os registros. O comando funciona como um tipo de banco de dados primitivo, permitindo a montagem de novos arquivos de registros a partir de registros existentes em diferentes arquivos.

Exemplo:

```
 $cat file1
1 a2 b3
2 a2 b3
 $cat file2
1 c2 e3
2 d2 f3
$join -1 1 -2 1 file1 file2
1 a2 b3 c2 e3
2 a2 b3 d2 f3
```

tr

O comando tr é um comando básico no Linux/Unix ele substitui (traduzir) o conteúdo de uma string (texto) recebido via entrada padrão (STDIN) de um formato para outro, ou ainda excluir caracteres.

Exemplo:

```
$echo linux | tr '[a-z]' '[A-Z]'
LINUX
```

103.3 Gerenciamento básico de arquivos

Listando arquivos e diretório

ls

O comando ls é usado arquivos em um diretório. Usuários DOS e Windows notarão a similaridade com o comando dir. Ele mesmo, ls(1) irá listar os arquivos no diretório atual.

```
$  ls -al /
total 96
drwxr-xr-x  23 root root  4096 Aug 30 16:17 .
drwxr-xr-x  23 root root  4096 Aug 30 16:17 ..
drwxr-xr-x   2 root root  4096 Aug 25 21:48 bin
drwxr-xr-x   3 root root  4096 Aug 29 20:29 boot
drwxr-xr-x  17 root root  3640 Aug 30 16:17 dev
drwxr-xr-x  93 root root  4096 Aug 30 16:19 etc
drwxr-xr-x   3 root root  4096 Aug 25 17:50 home
```

O ls pode também ser usado para conseguir outras informações nos arquivos. Por exemplo, para ver a data de criação, proprietário, e permissões, você especificaria uma lista longa

```
$ ls -l
total 88
drwxr-xr-x   2 root root  4096 Aug 25 21:48 bin
drwxr-xr-x   3 root root  4096 Aug 29 20:29 boot
drwxr-xr-x  17 root root  3640 Aug 30 16:17 dev
drwxr-xr-x  93 root root  4096 Aug 30 16:19 etc
```

file

O comando file mostra o tipo do arquivo utiliza arquivo /etc/magic para identificar arquivos que tenham um número mágico; ou seja, qualquer arquivo contendo uma constante numérica ou de cadeia que indique o tipo.

Para exibir o tipo de informação que o arquivo denominado teste.txt contém, digite o seguinte:

```
$file teste.txt
teste.txt: ASCII text
```

Para exibir o tipo de informação que o arquivo denominado arquivos .txt contém, digite o seguinte:

```
$file  -f arquivos.txt
teste.txt: ASCII text
Exemplo2:  empty
```

Copiando, movendo e excluindo arquivos

Para copiar de arquivos, renomear e mover pela hierarquia sistema de arquivos, ou até mesmo excluí-los. Usamos três pequenos comandos para isso.

cp

O comando cp é usado para fazer uma cópia de um ou mais arquivos ou diretórios. Precisa dar um ou mais nomes de origem e outro de destino. É importante saber que ao copiar um diretório recursivamente, o uso da barra / no final do diretório de origem fará com que apenas o conteúdo do diretório seja copiado para o destino; não usar a barra fará com que o diretório de origem e seu conteúdo sejam copiados no destino.

Suas opções principais são:

- - i : Modo interativo. Pergunta antes de sobrescrever um arquivo.

- · p : Copia também os atributos do arquivo original.
- · r: Copia recursivamente o conteúdo do diretório de origem.

Exemplo:

Para o arquivo hejaz do diretório atual para o /tmp utiliza:

```
$ cp hejaz /tmp
```

Para fazer copia recursiva.

```
$  cp -r dir/ /tmp
```

mv

O comando mv é usado para mover ou renomear um ou mais arquivos e diretórios. Em geral, os nomes a serem usados seguem as mesmas regras de cópias com cp; é possível renomear um único arquivo ou mover um conjunto e arquivos para um novo diretório:

Exemplo:

```
$ mv oldfile /tmp/newfile
```

rm

O comando **rm** remove arquivos e árvores de diretório.

Exemplo:

Para remover um único arquivo, especifique o nome dele quando você executar o rm:

```
$ rm file1
```

Para forçar a remoção do arquivo não importando com nada, passe a opção -f, como segue

```
$ rm -f file1
```

Para remover um diretório inteiro, você pode usar o -r e o -f juntos

```
$ rm -rf /dir
```

Criando e removendo diretório:

rmdir

O comando rmdir (1) remove diretórios do sistema de arquivos. O diretório deve estar vazio antes de poder ser removido. A síntaxe é simples:

```
$ rmdir dir
```

Você também pode remover um diretório e todos os diretórios pai passando a opção -p.

```
$ rmdir -p /tmp/hejaz
```

mkdir

O comando mkdir irá criar um diretório. Você simplesmente específica o diretório a ser criado quando você executa o mkdir. Esse exemplo criará o diretório hejaz no diretório atual.

```
$ mkdir hejaz
```

Você também pode especificar o caminho, como segue:

```
$ mkdir /usr/local/hejaz
```

A opção -p irá dizer ao mkdir para criar qualquer diretório pai. O exemplo acima falhará se /usr/local não existir.

A opção -p irá criar /usr/local e /usr/local/hejaz:

```
$ mkdir -p /usr/local/hejaz
```

Criando arquivo com touch

touch

O comando **touch** sem opções adota um ou mias nomes de arquivo como parâmetros e atualiza o tempo de **modificação** dos arquivos. É o mesmo registro de data e hora exibido com uma longa listagem de diretório.

Ao especificar um nome de arquivo para um arquivo que não existe, touch normalmente cria um arquivo vazio, a menos que você fique especificada a opção -c ou --no-create.

Exemplo:

```
$ ls -lh teste.txt
-rw------- 1 root root 32 Aug 30 20:55 teste.txt
$ touch teste.txt
ls -lh teste.txt
-rw------- 1 root root 32 Aug 30 21:15 teste.txt
```

Curingas e globbing

Uma cadeia contendo algum dos caracteres '?', '*' ou '[', é um padrão curinga. Globbing é o processo pelo qual o shell (ou possivelmente outro programa) expande esses padrões pra uma lista de nomes de caminho que correspondam ao padrão. A correspondência é feita como se segue:

- ? Corresponde a qualquer caractere único
- * corresponde a qualquer cadeia
- [] introduz uma *classe de caractere*.

Exemplo:

```
$ ls /dev/sd?
$ ls /dev/sd[abc]
$ ls /dev/sd*
```

Localizando arquivo

find

O comando **find** vai procurar arquivos ou diretórios usando todo o nome ou parte dele, ou através de outros critérios de busca, como tamanho, proprietário do arquivo, data de criação ou data do último acesso.

Para localizar somente os diretórios:

```
$  find . -type d
```

Comprimindo arquivo

Ao fazer backup, arquivar, ou transmitir arquivos, é comum comprimir os arquivos. Em um ambiente Linux, dois programas populares de compressão são o gzip e o bzip2. O comando zip usa o algoritmo Lempel-Ziv, enquanto bzip2 usa o algoritmo Burrows-Wheeler de ordenação de blocos

Exemplo:

. Para compactar um arquivo .tar:

```
$ gzip file.tar
```

Ou

```
$bzip2 file.tar
```

Para descomprimir um arquivo use a opção -d de gzip ou, o que é mais comum, usando o comando gunzip.

Exemplo:

```
$ gzip -d file.tar.gz
```

Arquivos do archive

Os comandos tar, cpio, e dd são comumente usados para fazer backup de grupos ou arquivos, ou até mesmo partições inteiras, seja para arquivamento ou para transmissão para outro usuário ou site.

O tar(originalmente de *Tape ARchive*) cria um arquivo de archive, ou *tarfile* ou *tarball*, a partir de um conjunto de arquivos de entrada ou diretórios; também recupera arquivos de tal archive. Se um diretório é dado como entrada para tar, todos os arquivos e subdiretórios estão automaticamente incluídos, o que torna tar muito conveniente para subárvores de archive da sua estrutura de diretório.

Exemplo:

Fazendo backup do diretório teste

```
$ tar -cvf tar -cvf teste.tar teste/
```

Para extrair:

$tar xvf teste.tar

cpio

O comando cpio opera em modo copy-out para criar um archive, em modo copy-in para restaurar um archive, ou modo copy-pass para copiar um conjunto de arquivos de um local para outro. Use a opção -o ou –create para o modo copy-out,-i ou a opção –extract para o modo copy-in, e a opção -pou –pass-through para o modo copy-pass. A entrada é uma lista de arquivos fornecidos em stdin.

Fazendo backup dos arquivos usando o cpio

```
$ find . -name '*.txt'  |cpio -o >cpio.bin
```

Para listar o conteudo de um arquivo cpio basta usar o atributo -t:

```
$ cpio -t < cpio.bin
backup/lista.txt
arquivos.txt
teste.txt
lista.txt
1 block
```

103.4 Usando Pipes e redirecionamento de saída

Redirecionamento ES padrão

Um shell Linux, como Bash, recebe entrada e envia saída como sequências ou *fluxos* de caracteres. Cada caractere é independente do que vem antes e do que vem depois dele. Os caracteres não são organizados em registros estruturados ou blocos de tamanho fixo.

Shells Linux usam três fluxos de E/S padrão, cada um dos quais é associado com um descritor de arquivo bem conhecido:

- **STOUD:** é o fluxo de saída padrão, que exibe saída de comandos. Possui descritor 1 de arquivo.
- **STDERR:** é ofluxo de erro padrão, que exibe saída de erro de comandos. Possui descritor 2 de arquivo
- **STDIN:** n é o fluxo de entrada padrão, que fornece entrada para comandos. Possui descritor 0 de arquivo.

Redirecionando a saída

Há duas maneiras de redirecionar saída para um arquivo:

- ***n*>:** redireciona saída do descritor de arquivo *n* para um arquivo. É necessário ter autoridade de gravação para o arquivo. Se o arquivo não existir, ele é criado. Se existir, os conteúdos existentes normalmente são perdidos sem qualquer aviso.

- ***n*>>**: também redireciona saída do descritor de arquivo *n* para um arquivo. Novamente, é necessário ter autoridade de gravação para o arquivo. Se o arquivo não existir, ele é criado. Se existir, a saída é anexada ao arquivo existente.

Use &> ou &>> para redirecionar tanto a saída padrão quanto o erro padrão para o mesmo local. Outra maneira de fazer isso é redirecionar o descritor de arquivo *n* e então redirecionar o descritor de arquivo *m* para o mesmo local utilizando a construção m>&n ou m>>&n. A ordem na qual as saídas são redirecionadas é importante. Por exemplo:

```
command 2>&1 >output.txt
```

não é o mesmo que

```
command >output.txt 2>&1
```

No primeiro caso, o stderr é redirecionado para o local atual de stdout e então, esse é redirecionado para output.txt, mas esse segundo redirecionamento afeta somente stdout, não stderr. No segundo caso, stderr é redirecionado para o local atual de stdout, que é output.txt.

Você pode desejar ignorar totalmente tanto a saída padrão quanto o erro padrão. Para fazer isso, redirecione o fluxo adequado para o arquivo vazio, /dev/null,

Exemplo:

```
$ ls x* z* 2>/dev/null
```

Canalização:

Você usa o operador | (canal) entre dois comandos para direcionar o stdout do primeiro para o stdin do segundo. Por exemplo, para encontrar no arquivo /etc/passwd o usuário **ubuntu:**

```
# cat /etc/passwd | grep ubuntu
```

É possível redirecionar simultaneamente a saida, tanto para um arquivo quanto para o stoud, usando o comando tee.

```
$ cat /etc/passwd | tee arquivo.txt
```

Comando xargs

O comando **xargs** lê a entrada padrão e então cria e executa comandos com a entrada como parâmetro. Se nenhum comando for fornecido, então o comando **echo** é utilizado:

```
$  cat teste.txt
1 Pedro
```

```
2 Luiz
3 Joao
$  xargs < teste.txt
1 Pedro 2 Luiz 3 Joao
```

Usando o xargs com lista de arquivos:

```
$ ls |xargs grep "1"
teste.txt:1 Pedro
```

103.5 Criando, monitorando e finalizando processos

Terminar um processo:

kill

O comando **kill** é usado para enviar um sinal para um processo ou para matá-lo (encerrar sua execução). Geralmente usa-se: kill -SINAL PID. Sendo que PID é o número que identifica o processo (Process ID).

Exemplo:

```
$ kill  -15 pid
$kill -SIGTERM pid
```

Para listar as opções disponíveis de sinal em uma tabela:

```
$kill -L
```

killall

O comando **killall** faz o mesmo processo que o kill, a única diferença é que mata o processo pelo nome.

Exemplo:

```
$ killall  -15 chrome
```

Controle com pkille pgrep

O comando pgrep localiza processos a partir de seus nomes ou outros atributos, exibindo os números PID de todos os processos correspondentes. Na sua forma de utilização mais simples, basta fornecer o nome de um programa ou comando:

```
$ pgrep apache
```

A opção-a é utilizada para obter o comando completo que foi utilizado para iniciar cada processo:

```
$ pgrep apache -a
```

pkill

O comando pkill envia um sinal para quaisquer processos informando-se todo o nome ou apenas parte do nome deles.

$pkill apache

Execução em background:

Para colocar um comando ou script em background é necessário adicionar o operador e-comercial (&) ao final da linha de comando.

Exemplo:

$ mcopy test. a &

jobs

O comando jobs lista processos suspenso e em background. para descobrir quais os processos que estão sendo executado em background ou em suspenso, utilizamos o comando **jobs:**

$job

bg e fg

Os comandos **bg e fg** , coloca um processo em background (bg) e para trazer o processo para foreground utiliza o **fg**

Exemplo:

Colocando o comando tail em background execute:

```
$ tail -f  /var/log/messenge
^Z
$jobs
[1]+  Running                 tail -f /var/log/messenge &
$ fg %1
```

Comando nohup

O comando **nohup** executa um comando ignorando os sinais de interrupção. O comando poderá ser executado até mesmo em segundo plano caso seja feito o logout do sistema.

Exemplo:

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

Sistemas

free

O comando free mostra o montante de memória ram, a quantidade de memória livre e o espaço de swap.

uptime

O comando uptime mostra o consumo de recursos da máquina, onde os valores finais *load avarage*, mostram a média de consumo geral de recursos de máquina que o sistema tem ocupado.

$uptime

Interface

O comando screen permite alterar entre diferentes tarefas numa mesma sessão do shell, é possível manter aberto um editor de texto e exibir o manual de um comando, alterando entre eles. Esse recurso é bastante útil em sessões remotas.

Para iniciar uma sessão do screen editando o arquivo *teste.txt* :

$ screen vim teste.txt

De inicio, não há diferença entre a execução com o screen e a execução convencional,mas você pode abrir uma tela com a combinação de **CTRL+a**.

Alguns comandos:

CTRL+a p: Alterna para a tela anterior;
CTRL +a n ou CTRL +a espaço: Alterna para a tela posterior;
CTRL +a A: Muda o titulo da tela;

Caso ocorra uma interrupção na comunicação com o servidor onde o screen está em execução, será possível recuperar as mesmas telas e suas respectivas tarefas numa nova conexão, com o comando screen -0 -R. A opção -0 determina que o screen seja desanexado caso ainda esteja sendo exibido e a opção -R

103.6 Modificar prioridade de processos em execução

Modificar prioridade

O comando **nice,** server para configurar a prioridade da execução de um comando na inicialização do processo. Para saber a prioridade de algum processo em execução deve-se analisar a coluna PR nos resultados pelo comando top ou ps -l.

Quanto mais alto a prioridade de um processo mais tempo de CPU o kernel aloca. A prioridade de um processo vai de -20 até +19, sendo inversamente proporciona, ou seja quanto menor o número nice, maior será sua prioridade.

Exemplo:

Para alterar a inicialização de um processo ao iniciar você executa:

nice -n -9 <processo>

renice

Quando um processo já se encontra em execução e se deseja modificar a sua prioridade, usa-se o comando renice.

renice [+|-] numero_nice [opção] alvo

Exemplo:

$ renice 18 4093 → altera a prioridade do processo 609 para 18;

Onde:

- -u → interpreta alvos como um usuário por exemplo;
- -p → interpreta alvos como um número pid;

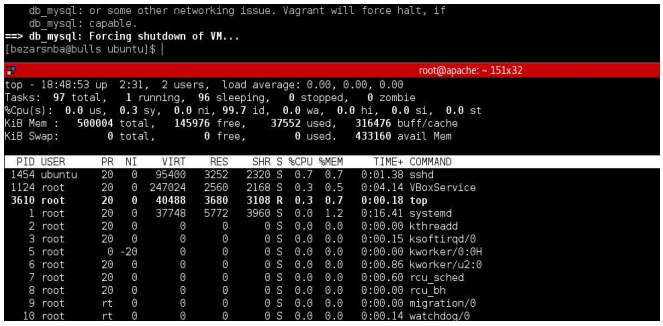
ps

O comando ps exibe uma lista dos processos em execução, também nos mostra qual *user* executou o processo, pid do processo, tempo de execução, hora de inicialização, sua sintaxe é:

$ ps [opção]

top

O comando **top** mostra os programas em execuções ativas, parados, tempo usado na CPU, detalhes sobre o uso da memória RAM, Swap, disponibilidade para execução de programas no sistema, etc. *T*em uma saída semelhante ao comando ps, com exibição continuamente atualizado na tela ou conforme definição do *sysadmin.*



103.7 Procurando em arquivos de texto usando expressões regular

Comando grep, egrep, fgrep

Os comandos **grep, egrep e fgrep** é responsável por procurar em um ou mais arquivos por linha que contém um padrão de busca, sua sintaxe é:

```
$grep [opção] padrão arquivo
$egrep [opção] padrão arquivo
$fgrep [opção] padrão arquivo
```

Algumas opções:

- -F: O mesmo que fgrep;
- -E: O mesmo que egrep;
- -r : faz busca recursiva;
- -e expr: Procura pela expressão regular expr;
- -n : exibe o numero de linhas;
- -c: Exibe apenas o numero de linhas
- -f arquivo: Le o padrão a partir do arquivo especificado;
- -w: procura apenas palavras inteiras.

Recomenda-se que o padrão esteja entre apóstrofos (‘), pois alguns caracteres têm significado especial para o shell e podem ser interpretado erroneamente.

Exemplo:

Para procurar a palavra “nameserver” dentro do arquivo **/etc/resolv.conf**

```
$  grep -n nameserver /etc/resolv.conf
3:nameserver 192.168.0.1
```

O comando grep pode ser usado com um operador |(pipe), que atua como o operador OU, basta usar a flag -E para dizer que vamos utilizar uma expressão(egrep) regular.

```
$ locate /usr/share/doc | grep -E '\.(html|txt)$'
```

sed

O comando **sed** é usado para procurar e subsituir padrões em textos, mostrando resultado em stoud. Sua sintaxe é:

```
sed [opção] ‘expressão regular’ [arquivo]
```

No sed, a expressão regular fica circunscrita entre barra /. Para imprimir somente as linhas com uma determinada string

$**sed -n '/dia/p'**

Opções mais comuns usada pelo sed:

- -e: Executa a expressão e comando a seguir;
- -f: Lê expressões e comandos do arquivo;
- -n: Não mostra as linhas que não correspondam a expressão

O sed não provoca nenhum tipo de alteração no arquivo de origem. Para isso é necessário direcionar a saída padrão do comando para um arquivo temporário.

103.8 Edição básica de arquivo usando o Vi

O editor vi foi criado por Bill Joy em 1975 para BSD. O nome é uma forma de abreviação para **vi**sual, um comando do editor texto que o faz ofecerecer recursos parecidos com os do vi, para chamar o edito basta usar:

$ vi arquivo

Comandos mais utilizados:

Inserção de caracteres

- **i-** insere texto antes do caractere atual.
- **I-** insere texto no início da linha atual.
- **a-** insere texto após o caractere atual.
- **A**- insere texto no final da linha atual.
- **o**- abre uma linha abaixo da atual e insere texto nela.
- **O**- abre uma linha acima da atual e insere texto nela.

Busca de palavra

- **/palavra**- procura palavra a partir da posição atual.
- **?palavr**a- procura palavra no sentido contrário (na direção do começo do arquivo).
- **n**- procura próxima ocorrência da última palavra procurada (na mesma direção em que foi buscada).
- **#-** destaca todas as ocorrências iguais à palavra onde o cursor está posicionado.

Exclusão de caracteres

- **<DEL**>- exclui a letra anterior ao cursor (depende da configuração).
- **x-** exclui a letra sobre o cursor.
- ***n*x**- exclui as próximas n letras.

- **dw** - exclui o restante da palavra atual (a partir do cursor).
- ***n*dw** - exclui as *n* próximas palavras.
- **dd** - exclui a linha atual.
- ***n*dd** – exclui *n* linhas a partir da atual.
- **u** - desfazer. Restaura o que foi apagado por último ou apaga o que foi inserido por último.

Operações com buffers

- **yy ou Y-** copia a linha inteira.
- ***n*yy ou *n*Y** - copia *n* linhas.
- **ndd** apaga *n* linhas (a partir da atual). Elas poderão ser recolocadas com os comandos **p** e **P**.
- **p** - coloca após a linha atual a última linha copiada ou apagada.

104: Dispositivos, sistema de arquivos e padrão FHS

104.1 Criar partições de sistema em arquivos

fdisk

O comando **fdisk** cria partições em um disco, até 4 partições primárias e inúmeras partições lógicas, dependendo somente do tamanho do disco (levando em conta que cada partição requer um mínimo de 40MB). Também podemos modificar ou deletar partições já existentes ou recém criadas no disco.

Use o comando fdisk -l para listar as partições:

```
# fdisk -l
Disk /dev/sda: 223.6 GiB, 240057409536 bytes, 468862128 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 9833518C-B109-4001-9A17-E23CF84E71CA

Device       Start       End   Sectors   Size Type
/dev/sda1     2048    411647    409600   200M EFI System
/dev/sda2   411648   1435647   1024000   500M Linux filesystem
/dev/sda3  1435648 468860927 467425280 222.9G Linux filesystem
```

Para ver apenas as partições de um determinado disco, podemos usar:

```
# fdisk  -l /dev/sda
```

Para manipular as partições use o comando **fdisk,** uma vez dentro do fdisk, algumas letras corresponderão a comandos.

Cada partição possui um número hexadecimal que a identifica como apropriadora um determinado sistema operacional. O fdisk cria novas partições identificadas como ativas de Linux, cujo código hexadecimal é 83 (Ox83).

GPT

Com a limitação do padrão MBR, o padrão GPT está se tornando cada vez mais utilizado, a tabela de partição no formato MBR trabalha com disco de no máximo 2TiB, diferente do GPT.

O comando gdisk é mais indicado quando se lida com GPT. Algumas versões não atualizadas da libparted não funcionam corretamente.

O comando gdisk é correlato ao tradicional fdisk na maneira de ser utilizado e em finalidade. Permitindo criação de mais 128 partições.

O gdisk converte automaticamente o MBR para GPT assim que for utilizado para alterar a tabela de partições de um dispositivo, ele armazena uma cópia de egurança da MBR no final do dispostivo.

Para alterar o dispositivo /dev/sda, basta fornecer o caminho como argumento:

```
#gdisk /dev/sda
```

Você irá preencher algumas informações e logo em seguida finalizara o processo de conversão e o resultado pode ser visto com o comando **gdisk -l /dev/sda**.

Parted

O comando **parted** é um programa que permite efetuar o particionamento de discos no Linux, assim como o redimensionamento de partições existentes nos discos. Com ele é possível criar, excluir, redimensionar e realizar outras operações em diversos tipos de partições, incluindo partições formatadas com sistemas de arquivos como ext3, linux-swap, FAT, reiserfs e muitas outros. Abaixo temos uma lista básica dos principais comandos disponiveis:

- **quit**: Sai do parted
- **mkfs *PART TIPO-FS:*** Cria um sistema de arquivos do tipo *TIPO-FS* na partição *PART*
- **mklabel *TIPO-LABEL*:** Cria uma nova tabela de partição do tipo *IPO-LABEL*,
- **mkpart *TIPO-PART [TIPO-FS] início fim:*** Cria uma partição do tipo *TIPO-PART*

Redimensionando a partição

Antes de redimensionar uma partição, inicialize no modo de recuperação ou desmonte quaisquer partições do dispositivo e desabilite qualquer espaço virtual no dispositivo. Inicie o **parted**, onde /dev/sda é o dispositivo:

```
# parted /dev/sda
```

Visualize a tabela de partições corrente para determinar se há espaço livre suficiente:

```
(parted) print
Model: ATA SanDisk SDSSDA24 (scsi)
Disk /dev/sda:  1100GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End    Size   File system  Name              Flags
 1      1049kB  1100GB  11000GB  primary     ntfs           boot
```

Para redimensionar a partição /dev/sda é realizado com o comando resizepart do parted:

```
(parted) resizepartet 1 500GB
```

Novas partições podem ser criadas a partir do próprio parted, com o comando mkpart

```
(parted) mkpat primary ext4 500GB 1100GB
```

mkfs

O comando mkfs cria diversos sistemas de arquivos em partições, em que a opção -t indica o tipo de sistema de arquivos. Para criar uma partição ext3 na partição /dev/hda3, usa-se mkfs -t ext3 /dev/hda3. Existe também comandos específicos para cada sistema de arquivos: mkfs.ext2, mke2fs, mkfs.ext3, mkfs.xfs. Sem parâmetros, o mkfs cria um sistema de arquivo ext2. Para criar um sistema de arquivos ext2 e ext3(é necessário informar a flag -j0, pode utilizar o comando mke2fs , exemplo:

```
#mke2fs -j /dev/sdb1
```

mkswap

O comado mkswap cria partição swap.

Exemplo:

```
#mkswap /dev/sda1
```

Após esse comando você precisa ativar a partição swap criado, para isso execute o comando: swapon -a e as entradas referentes a partições swap em /etc/fstab não tem ponto de montagem. Exemplo de entrada swap em /etc/fstab:

```
/dev/hda2 swap swap defaults 0 0
```

104.2 Manutenção de integridade de sistema de arquivo

Checando o arquivo

O comando fsk O fsck (*file system consistency check* — verificação da consistência do sistema de arquivos) é um comando usado para encontrar erros no sistema de arquivos e corrigi-los. A partição precisa estar desmontada ou montanda como somente-leitura(ro), para verificação.

Como o comando mkfs, o fsck possui a opção *-t* para especificar o tipo do sistema de arquivos e um comando especifico para cada partição exemplo: f**sck.ext2 ou e2fsck, fsck.ext3, fsck.xfs, reiserfsck e dosfsck.**

Examinando e corrigindo sistema de arquivos:

O comando para inspeção de sistemas de arquivos é o debugfs, ele faz pequenos reparos no sistema operacional causado por queda de energia ou problema de hardware para arquivos ext2 e ext3, exemplo:

```
# debugfs -w /dev/sda1
```

Podemos utilizar o help para verificar as opções permitidas. Ele é muito utilizado para recuperação de arquivos deletados. Para recuperarmos, utilizamos as opções **undelete ou dump.**

dump2fs

O comando dump2fs traz muitas informações, entre elas o suporte do sistema de arquivo a journaling ou não, número de inodes, blocos livres, tamanho dos blocos, intervalo entre checagem automáticas etc. Usando a flag **-b**, onde serão exibidos somente os blocos defeituosos no sistema de arquivo, exemplo:

```
# dumpe2fs -h /dev/sda1
dumpe2fs 1.42.13 (17-May-2015)
dumpe2fs: Bad magic number in super-block while trying to open /dev/sda1
Couldn't find valid filesystem superblock.
# dumpe2fs -h /dev/sda2
dumpe2fs 1.42.13 (17-May-2015)
Filesystem volume name:   <none>
Last mounted on:          /boot
Filesystem UUID:          8242393f-fed4-47ea-ada8-00c6b14c08a2
```

Filesystem magic number: 0xEF53
Filesystem revision #:   1 (dynamic)

tune2fs

O comando e2label e tune2fs pode ser usado para alterar o Label de um dispositivo. O tune2fs pode fazer muito mais coisas do que somente alterar o nome.

Usando a flag **-l**, traz informaçoes do dispositivo e para mudar o label utilizamos a opção -L:

tune2fs -L BOOT /dev/sda2

tune2fs -l /dev/sda2 | grep name
Filesystem volume name: BOOT

Verificar espaço em disco

du e df

Os comandos **du e df**, são essenciais para analisar espaço em disco ocupada por arquivos em uma partição:

df: Mostra o espaço ocupado e disponível em cada dispositivo. A analise é feita diretamente no dispositivo. Por padrão, mostra o espaço em unidades de 1KB

du: Mostra o espaço ocupado por arquivos e/ou diretorios. Sem argumentos, mostra o uso de cada diretório no sistema.

104.3 Controle de mount e umount dos sistema de arquivos

Arquivo /etc/fstab

O arquivo "**/etc/fstab"** permite configurar o sistema para montar partições, CD-ROMs, disquetes e compartilhamentos de rede durante o boot. Cada linha é responsável por um ponto de montagem. É através do "/etc/fstab" que o sistema é capaz de acessar o seu CD-ROM. Antes de editá-lo, faça um backup

cp /etc/fstab /etc/fstab-bkp

O arquivo de configuração do /etc/fstab segue o formato:

#cat /etc/fstab

```
/dev/mapper/fedora-root /                       ext4    defaults,x-systemd.device-timeout=0 1 1
UUID=8242393f-fed4-47ea-ada8-00c6b14c08a2 /boot                   ext4    defaults        1 2
UUID=B47D-5CB3          /boot/efi               vfat    umask=0077,shortname=winnt 0 2
/dev/mapper/fedora-home /home                   ext4    defaults,x-systemd.device-timeout=0 1 2
/dev/mapper/fedora-swap swap                    swap    defaults,x-systemd.device-timeout=0 0 0
```

Cada linha corresponde a um ponto de montagem e contém os seguintes termos, separados por tabulações ou espaços:

- Partição do dispositivo; Pode ser utilizado LABEL= ou UUID= para determinar o dispositivo.
- Ponto de montagem (swap se tratar-se de uma área de troca) ;
- Tipo de sistema de arquivos;
- Opção de montagem;
- dump (O ou 1) : Determina se o dispositivo deverá ser considerado pelo co mando dump. Se ausente, O é considerado;
- fsck (1 ou 2) : Determina a ordem da checagem feita pelo fsck durante a inicialização. Para a partição raiz, deve ser 1 . Se ausente, O é presumido e a checagem não é feita no boot.

mount e umount

O comando **mount e umount** faz a montagem e desmontagem manual em um sistema de arquivo é importante quando não for configurada em entrada em /etc/fstab é utilizado o mount, sua sintaxe segue o formato:

```
# mount -t type device dir
```

Desde que haja uma entrada no arquivo fstab para esse ponto de montagem, o comando mount sabe quais opções utilizar. Caso não haja uma entrada para este dispositivo, você deverá utilizar algumas opções junto ao comando mount:

```
$ mount -t iso9660 -o ro /dev/cdrom /cdrom
```

A opção **-t iso9660** é o tipo de sistema de arquivos do dispositivo a ser montado. Neste caso, ele seria o sistema de arquivos iso9660, que é usado normalmente pelos drivers de CD-ROM.

Para desmontar uma partição montada você pode utilizar:

```
# umount /dev/cdrom
# umount /cdrom
```

104.4 Gerenciar quotas de disco

Analise e controle de cotas

Quotas de Disco permitem controlar o uso do disco por usuário ou grupo, ou por ambos (usuário e grupo). Estas cotas garantem um limite de espaço em disco, se

este limite for ultrapassado, o usuário ou grupo de usuários não será capaz de criar novos arquivos até que os mesmos sejam apagados ou redimensionada.

Comandos para gerência de quotas

quota

O comando **quota** é usado para visualizar informações sobre as quotas de espaço em disco configuradas para um determinado usuário ou grupo de usuários. Ele possui as seguintes opções:

- -u Mostra a quota definida para um determinado usuário.
- **-g** - Mostra a quota definida para um determinado grupo de usuários;
- **-v -** Mostra as quotas mesmo que nenhuma esteja definida;
- **-q** - Mostra somente quando a rota for excedida;
- **-i** - Ignora pontos de montagem montados pelo automounter;
- **-l** - Mostra apenas quotas de sistemas de arquivos locais;

Para visualizar a quota de um usuário basta executar:

#quota -u <usuario>

quotaon

O comando quotaon habilita o gerenciamento prévio de todas as quotas de disco configuradas em um ou mais sistemas de arquivos. Ele possui as seguintes opções:

- **-a**- Habilita o gerenciamento de todas as quotas de disco para todos os sistemas de arquivos configurados para controle de quotas no arquivo /etc/fstab;
- **-v**- Mostra uma mensagem para cada sistema de arquivos com quotas habilitadas;
- **-u** - Habilita o gerenciamento de quotas de usuários em um determinado dispositivo;
- **-g** - Habilita o gerenciamento de quotas de grupos em um determinado dispositivo;

Para habilitar as quotas de grupo somente do sistema de arquivos localizado no diretório /home.

#quota -gv /home

edquota

O comando **edquota** é um utilitário usado somente pelo superusuário para edição de quotas de usuários e grupos. O editor vi é o editor padrão usado com o edquota para editar os arquivos quota.user e quota.group. Ele possui as seguintes opções:

- **-r** Edita quotas em dispositivos remotos, um bom exemplo são os dispositivos que usam NFS;
- **-u** - Edita as quotas de usuários;
- **-g** - Edita as quotas de grupos de usuários;
- **-p user**- Copia as configurações de uma quota de usuário padrão para criar outra;
- **-F format-name**- Edita as quotas para um formato específico (vfsold, vfsv0, rpc e xfs);
- **-f** filesystem - Realiza operações especificas para um único sistema de arquivos;
- **-T**- Edita o tempo limite de uso de quotas de usuários/grupos quando o softlimit é excedido;
- **-t** - Edita o período de graça para os Soft Limits para cada sistema de arquivos;

repquota

O comando repquota gera um relatório do uso das quotas de discos de usuários e grupos de um determinado dispositivo. Observe as opções mais usadas:

- **-a** - Exibe um relatório de todas as quotas dos sistemas de arquivos que estão contidos no arquivo /etc/mtab;
- **-v** - Cria um cabeçalho descritivo para o relatório de quotas;
- **-u** - Cria um relatório de utilização por usuário;
- **-g** - Cria um relatório de utilização por grupo;

104.5 Gerenciar permissões de arquivos

O sistema de arquivos armazena informações sobre o dono de cada arquivo e diretório no sistema. Isso inclui qual usuário e grupo possui um determinado arquivo. A maneira mais fácil de ver essa informação é com o comando ls:

```
$ ls -l /usr/bin/wc
-rwxr-xr-x 1 root bin 7368 Jul 30 1999 /usr/bin/wc
```

Podemos ver que o usuário ”**root**”e o grupo ”**bin**”são os proprietários desse arquivo, para mudar o proprietário do arquivo você pode executar **chown** (que significa mudar o proprietário”) e **chgrp** (que significa ”mudar o grupo”). Para mudar o dono de um arquivo para daemon, nós podemos usar o comando chown:

```
# chown daemon /usr/bin/wc
```

Para mudar o grupo que possui um arquivo para ”root”, nós podemos usar o comando chgrp:

```
# chgrp root /usr/bin/wc
```

Podemos utilizar o comando chown para especificar as propriedades de usuário e grupo para um arquivo:

```
# chown daemon:root /usr/bin/wc
```

chmod e umask

A permissão de arquivos é outra parte importante do aspecto multi-usuário do sistema de arquivos, você pode alterar quem lê, escreve, e executa arquivos.
A informação de permissão é armazenada na forma de quatro dígitos octais, cada um especificando diferentes conjuntos de permissões. São eles, permissão do proprietário, permissão de grupo e, permissão de outros. Os quatro dígitos octais são utilizados para armazenar informações especiais como identificação de usuário (user ID), identificação de grupo (group ID), e o bit controle, veja a tabela de permissões octais:

- 0- Nenhuma permissão de acesso. Equivalente a -rwx.
- 1- Permissão de execução (x).
- 2- Permissão de gravação (w).
- 3- Permissão de gravação e execução (wx). Equivalente a permissão 2+1
- 4- Permissão de leitura (r).
- 5- Permissão de leitura e execução (rx). Equivalente a permissão 4+1
- 6- Permissão de leitura e gravação (rw). Equivalente a permissão 4+2
- 7- Permissão de leitura, gravação e execução. Equivalente a +rwx (4+2+1).

Para aplicar alterações em um arquivo basta fazer a combinações os números octais:

```
$ chmod 755 /tmp/exemplo
$ ls -l /tmp/exemplo -rwxr-xr-x 1 david users 0 Apr 19 11:21 /tmp/exemplo
```

Também você pode fazer combinações de letras, por exemplo:

```
$ chmod u+rwx,a+xr,ug+rx exemplo
```

umask

A umask (user mask) são 3 números que definem as permissões iniciais do dono, grupo e outros usuários que o arquivo/diretório receberá quando for criado ou copiado para um novo local. Digite umask sem parâmetros para retornar o valor de sua umask atual.

A umask tem efeitos diferentes caso o arquivo que estiver sendo criado for binário (um programa executável) ou texto. Veja a tabela a seguir para ver qual é a mais adequada a sua situação:

	Arquivo		
Umask	Binario	Texto	Diretorio
0	r-x	rw-	rwx
1	r--	rw-	rw-
2	r-x	r--	r-x
3	r--	r--	r--
4	--x	-w-	-wx

5	---	-w-	-w-
6	--x	---	--x
7	---	---	---

Exemplo de utilização do umask sendo executado sem argumentos:

```
# umask
 0022
```

Alterando o valor do umask:

```
# umask 0077
 ls -l tempfile
-rw-------- 1 david users 0 Apr 19 11:21 tempfile
```

104.6 Criar e alterar links simbólicos e hardlinks

O comando ln (1) é usado para criar links entre arquivos. Esses links podem ser hard links ou links simbólicos.

Se você quiser fazer um link simbólico do diretório /var/media/mp3 e colocar esse link no seu diretório home, você faria isso:

```
$ ln -s /var/media/mp3 \~{}/mp3
```

A opção -s diz para o ln fazer um link simbólico. A próxima opção é o caminho do link, e a opção final é como chamar o link. Nesse caso, o comando apenas irá criar um arquivo chamado mp3 no seu diretório home que aponta para o /var/media/mp3. Você pode chamar o link do que você quiser apenas mudando a última opção.

HardLinks

Hardlinks são um ou mais nomes que um inode do sistema de arquivo pode ter. Todo arquivo criado é, necessariamente, um hardlink para seu inode correspondente. Novos hardlinks são criados usando o comando ln:

Para criar um hard link de /usr/bin/email para /usr/bin/mutt, apenas digite o trecho a seguir:

```
$ ln exemplo  exemplo
```

A opção -i do comando ls mostra o número dos inodes dos arquivos:

```
$ls -i
80 exemplo
80 exemplo2
```

Ambos os arquivos são hardlinks para o mesmo inode 80. Hardlinks para o mesmo inode possuiem a mesma permissão

104.7 Encontrar arquivos de sistema e conhecer sua localização correta.

Localização de arquivos

locate

O comando locate realiza consultas rápidas de arquivos, porém, suas buscas não são feitas em tempo real, a procura é feita através de uma base de dados criada pelo seguinte comando:

```
# updatedb
```

Depois de atualizado a base dados com o comando **updatedb,** você pode executar o locate:

```
$ locate exemplo2
/root/exemplo2
```

Para ignorar caracteres maiúsculas e minúscula você pode usar com a flag -i

```
# locate -i Exemplo2
/root/Exemplo2
/root/exemplo2
```

No arquivo de configuração **/etc/updatedb.conf**, contém informações como quais diretorios e sistemas de arquivos ignorados na atualização do banco de dados.

whereis e which

Os comandos **whereis** e **which** são usados para realizar a busca de arquivos no sistema de forma muito rápida.

- whereis- busca por arquivos executáveis, man pages, arquivos de configuração e fontes.
- which- busca por executáveis nos PATHs exportados.

Exemplo:

```
$whereis ssh
ssh: /usr/bin/ssh /etc/ssh /usr/share/man/man1/ssh.1.gz
$which ssh
/usr/bin/ssh
```

type

O comando **type** é usado para determinar o tipo de um ou mais comando, exemplo:

```
$ type rm if set
rm is /bin/rm
if is a shell keyword
set is a shell builtin
```

Referencia:

http://www.hardware.com.br/termos/bios
https://www.vivaolinux.com.br/artigo/Gerenciando-quotas-de-disco?pagina=3
www.thobias.org/doc/sosed.html
https://www.ibm.com/developerworks/br/linux/library/l-lpic1-v3-103-4/index.html
https://eriberto.pro.br/wiki/index.php?title=Linux_Volume_Manager_(LVM)
http://johnnatah.blogspot.com.br/2016/02/lpic-1-1012-boot-system.html