A linha de comando do Unix e GNU/Linux

Michael Opdenacker
Free Electrons
http://free-electrons.com
Traduzido por
Klayson Sesana Bonatto

Criado com OpenOffice.org 2.x

Memento de comandos mais utilizados

Este é um útil recurso que pode ser utilizado para acompanhar esta apresentação.

Exemplos para os comandos mais úteis são dados em uma única página.

Sugestões de utilização

Cole esta página na sua parede, use-a como wallpaper da área de trabalho do seu desktop, imprima-a nas suas roupas, corte-a e crie marcadores de página...

Faça o download em
http://free-electrons.com/training/intro_unix_linux

Sistemas de arquivos Unix

Tudo é um arquivo

Quase tudo no Unix é um arquivo!

- Arquivos comuns
- Diretórios
  Diretórios são apenas arquivos que listam um conjunto de outros arquivos.
- Links simbólicos
  Arquivos que referenciam o nome de outro arquivo.
- Dispositivos e periféricos
  Lê e grava em dispositivos como se fossem arquivos comuns.
- Pipes
  Usados para concatenar programas
  ```
  cat *.log | grep error
  ```
- Sockets
  Comunicação interprocessos

Nomes de arquivos

Características dos nomes de arquivos presentes desde o início do Unix

- Diferencia maiúsculas e minúsculas (case sensitive).
- Não possui um limite óbvio de tamanho.
- Pode conter qualquer caractere (incluindo espaços, exceto "/").
  O tipo do arquivo é armazenado no arquivo ("números mágicos").
  Extensões dos nomes de arquivo não são obrigatórias e não são interpretadas. Apenas utilizadas para conveniência do usuário.
- Exemplos de nomes de arquivos:

```
README          .bashrc          Windows Buglist
index.htm       index.html       index.html.old
```

Caminhos (paths) de arquivos

Um caminho (*path)* é uma seqüência de diretórios aninhados com um arquivo ou diretório no final, separados pelo caractere `/`.

- Caminho relativo:
  `documents/fun/microsoft_jokes.html`
  Relativo ao diretório atual.
- Caminho absoluto:
  `/home/bill/bugs/crash9402031614568`
- `/` : diretório *root* (ou raiz).
  É o início dos caminhos absolutos para todos os arquivos do sistema (até mesmo para arquivos existentes em mídias removíveis e compartilhamentos de rede).

Estrutura do sistema de arquivos GNU/Linux (1)

Não é imposta pelo sistema. Pode variar de um sistema para outro, mesmo entre duas instalações do GNU/Linux!

`/`	Diretório Root
`/bin/`	Comandos básicos essenciais do sistema
`/boot/`	Imagens do kernel, initrd e arquivos de configuração
`/dev/`	Arquivos que representam dispositivos Ex: `/dev/hda`: primeiro disco rígodo IDE do sistema
`/etc/`	Aquivos de configuração do sistema
`/home/`	Diretórios dos usuários
`/lib/`	Bibliotecas compartilhadas básicas do sistema

Estrutura do sistema de arquivos GNU/Linux (2)

`/lost+found`	Arquivos corrompidos que o sistema tentou recuperar
`/mnt/`	Sistemas de arquivos montados (`/mnt/usbdisk/`, `/mnt/windows/` ...)
`/opt/`	Programas instalados pelo administrador do sistema. (`/usr/local/` às vezes usado com esse propósito)
`/proc/`	Accesso a informações do sistema (`/proc/cpuinfo`, `/proc/version` ...)
`/root/`	Diretório particular do usuário root
`/sbin/`	Comandos acessíveis apenas pelo administrador.
`/sys/`	Controles do sistema e dispositivos (freqüência da CPU, módulos do kernel, etc.)

Estrutura do sistema de arquivos GNU/Linux (3)

`/tmp/`	Arquivos temporários
`/usr/`	Programas dos usuários (não essenciais ao sistema) (`/usr/bin/`, `/usr/lib/`, `/usr/sbin`...)
`/usr/local/`	Programas instalados pelo administrador do sistema. (usado algumas vezes no lugar de `/opt/`)
`/var/`	Dados usados pelo sistema ou programas servidores `/var/log/` (logs do sistema e programas) `/var/spool/mail` (e-mails recebidos) `/var/spool/lpd` (trabalhos de impressão)...

Comando ls

Lista os arquivos do diretório atual em ordem alfanumérica, exceto arquivos que iniciam com o caractere “.”.

- `ls -a` (all)
  Lista todos os arquivos (inclusive os arquivos `.*`)
- `ls -l` (long)
  Listagem longa (tipo, data, tamanho, proprietário, permissões)
- `ls -t (time)`
  Lista os arquivos mais recentes primeiro
- `ls -S` (size)
  Lista os maiores arquivos primeiro
- `ls -r` (reverse)
  Inverte a ordenação
- `ls -ltr` (opções podem ser combinadas)
  Listagem longa, com os arquivos mais recentes no final.

Padrões de substituição de nomes de arquivos

Isso é melhor explicado com exemplos!

- `ls *txt`
  O shell primeiro substitui *txt por todos os nomes de arquivos e diretórios que terminam com `txt` (incluindo `.txt`), exceto aqueles que iniciam com “.”, e então executa o comando `ls`.
- `ls -d .*`
  Lista todos os arquivos e diretório que inicial com “.”.
  `-d` instrui o `ls` a não exibir o conteúdo dos `.*` diretórios.
- `cat ?.log`
  Exibe todos os arquivos cujos nomes possuem 1 caractere e terminam com “.log”.

Diretórios Especiais (1)

`./`

- O diretório atual. Útil para comandos que levam um diretório como argumento. Também é útil para executar comandos localizados no diretório atual (veremos mais detalhes adiante).
- Dessa forma, `./readme.txt` e `readme.txt` são equivalentes.

`../`

- O diretório pai. Está sempre presente no diretório "." (veja `ls -a`). Única referência ao diretório pai.
- Uso típico:
  `cd ..`

Diretórios Especiais (2)

`~/`

- Na verdade não é um diretório especial. Os Shells apenas o substituem pelo diretório home do usuário atual.
- Não pode ser utilizado na maioria dos programas, já que ele não é um diretório real.

`~sydney/`

- Similarmente, é substituído pelos shells pelo diretório home do usuário `sydney`.

Os comandos cd e pwd

- `cd <dir>`
  Alterna o diretório atual para `<dir>`
- `pwd`
  Exibe o diretório atual ("diretório de trabalho")

O comando cp

- `cp <arquivo_origem> <arquivo_destino>`
  Copia o arquivo origem para o arquivo destino.
- `cp arquivo1 arquivo2 arquivo3 ... dir`
  Copia os arquivos para o diretório destino (último argumento).
- `cp -i` (interativo)
  Solicita confirmação ao usuário caso o arquivo destino já exista.
- `cp -r <diretório_origem> <diretório_destino>` (recursivo)
  Copia todo o diretório.

Cópia inteligente de diretórios com rsync

rsync (remote sync) foi projetado para sincronizar diretórios em duas máquinas interligadas por uma conexão de baixa velocidade

- Apenas copia arquivos que sofreram alterações. Arquivos com o mesmo tamanho são comparados por meio dos seus checksums.
- Apenas transfere os blocos do arquivo que sofreram alteração!
- Pode compactar os blocos transferidos.
- Preserva links simbólicos e as permissões de arquivos: também é muito útil para cópias realizadas na mesma máquina.
- Pode ser usado via ssh (shell remoto seguro). Muito útil para atualizar o conteúdo de um website, por exemplo.

Comandos mv e rm

- `mv <nome_antigo> <novo_nome>` (move)
  Renomeia o arquivo ou diretório passado como parâmetro.
- `mv -i` (interativo)
  Solicita confirmação ao usuário caso o arquivo destino já exista.
- `rm arquivo1 arquivo2 arquivo3 ...` (remove)
  Remove os arquivos passados como parâmetro.
- `rm -i` (interativo)
  Solicita confirmação do usuário antes de excluir o arquivo.
- `rm -r dir1 dir2 dir3` (recursivo)
  Remove recursivamente os diretórios passados como parâmetro.

Criando e removendo diretórios

- `mkdir dir1 dir2 dir3 ...` (cria diretórios)
  Cria diretórios a partir dos nomes passados como parâmetros.
- `rmdir dir1 dir2 dir3 ...` (remove diretórios)
  Remove os diretórios passados como parâmetros.
- Seguro: apenas funciona quando os diretórios estão vazios.
  Alternativa: `rm -r`

Exibindo o conteúdo de arquivos

Existem várias formas de exibir o conteúdo de arquivos:

`cat arquivo1 arquivo2 arquivo3 ...` (concatena)
Concatena e exibe o conteúdo dos arquivos passados como parâmetros.

- `more arquivo1 arquivo2 arquivo3 ...`
  A cada página, solicita que o usuário pressione uma tecla para continuar. Também permite a localização de palavras. (comando `/`)
- `less arquivo1 arquivo2 arquivo3 ...`
  Faça mais do que o `more` com o less.
  Não lê todo o arquivo antes de iniciar.
  Permite o movimento de retrocesso no arquivo (comando `?`)

O comando grep

- `grep <padrão> <arquivos>`
  Pesquisa os arquivos passados como parâmetros e exibe as linhas que possuem o padrão.
- Exemplo: `grep error *.log`
  Exibe todas as linhas que contém a string `error` nos arquivos `*.log`.
- `grep -i error *.log`
  Mesma situação, porém não diferencia maiúsculas de minúsculas.
- `grep -ri error .`
  Mesma situação, porém faz a pesquisa recursivamente em todos os arquivos no diretório . e nos seus subdiretórios.
- `grep -v info *.log`
  Exibe todas as linhas dos arquivos `*.log` exceto aquelas que contêm a string `info`.

Links simbólicos

Um link simbólico é um arquivo especial que é apenas uma referência para o nome de outro arquivo ou diretório.

Útil para reduzir a utilização de disco e a complexidade quando 2 arquivos têm o mesmo conteúdo.

- Exemplo:
  `anakin_skywalker_biography -> darth_vador_biography`
- Como identificar links simbólicos:
  - `ls -l` exibe "`->`" e o nome do arquivo "linkado".
  - GNU `ls` exibe links com uma cor diferente (azul ciano).

Criando links simbólicos

- Para criar um link simbólico (mesma ordem do comando cp):
- `ln -s nome_do_arquivo nome_do_link`
- Para criar um link para um arquivo em outro diretório, com o mesmo nome:
  `ln -s ../README.txt`
- Para remover um link:
  `rm nome_do_link`
  Obviamente, isso não removerá o arquivo “linkado”!

Direitos de acesso a arquivos

Use ls -l para verificar os direitos de acesso a um arquivo.

3 tipos de direitos de acesso:

- Acesso de Leitura - Read (`r`)
- Acesso de Gravação – Write (w)
- Acesso de Execução – Execute (`x`)

3 tipos de níveis de acesso:

- Usuário - User (`u`): para o proprietário do arquivo.
- Grupo - Group (`g`): cada arquivo tem um atributo de “grupo”, que corresponde a uma determinada lista de usuários.
- Outros - Others (`o`): para todos os outros usuários.

Exemplos de direitos de acesso

- `-rw-r--r--`
  Pode ser lido e gravado pelo proprietário do arquivo e apenas lido pelos demais usuários.
- `-rw-r-----`
  Pode ser lido e gravado pelo proprietário do arquivo e apenas lido pelos usuários que pertencem ao grupo do arquivo.
- `drwx------`
  Diretório acessível apenas ao proprietário.
- `-------r-x`
  Arquivo executável pelos outros usuários do sistema, menos pelos seus amigos ou por você mesmo. Ótima proteção para uma armadilha...

chmod: alterando permissões

- `chmod <permissões> <arquivos>`
  2 formatos para permissões:
- Formato Octal (abc):
  `a,b,c = r*4+w*2+x` (`r`, `w`, `x`: booleans)
  Examplo: `chmod 644 <arquivo>`
  (`rw` para `u`, `r` para `g` e `o`)
- Ou o formato simbólico. Facilitando a compreensão com exemplos:
  `chmod go+r`: addiciona direito de leitura para o grupo e outros.
  `chmod u-w`: remove direito de gravação do usuário (proprietário).
  `chmod a-x`: (a: all) remove direito de execução de todos.

E/S padrão, redirecionamentos, pipes