Comandos no terminal Bash

Seu comando é uma ordem

Por trás das amigáveis telas do seu sistema operacional, está o simples, mas poderoso Shell Bash.

Bruce Byfield

Muitos usuários de desktop utilizam a linha de comando raramente e quando o fazem, geralmente é com medo de que algo ruim aconteça. Essa abordagem é compreensível, mas no entanto, se cada usuário reservar um tempo para compreender a estrutura e o funcionamento da linha de comando, é possível aumentar seu controle sobre o computador.

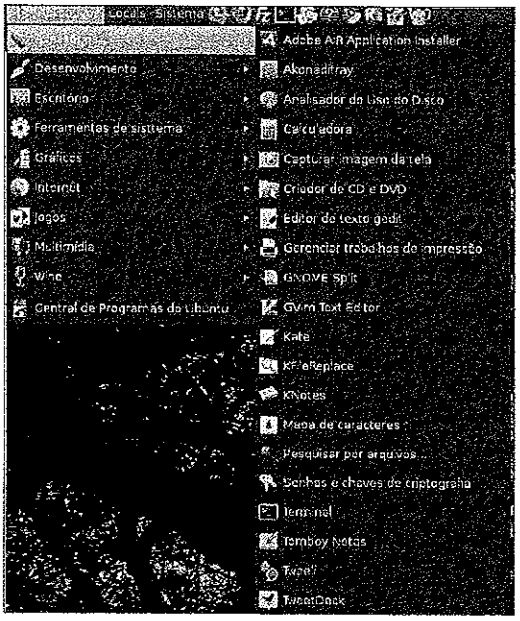

Figura 1: Iniciar uma janela de terminal no Ubuntu.

Por padrão, a maioria das distribuições utiliza o Bash (*Bourne Again Shell*), que é chamado também por outros nomes como *terminal*, *Shell* ou mesmo *prompt de comando*. Ele é um intérprete da linha de comando – um programa que executa macros e outros utilitários. Esses utilitários e macros são os comandos digitados no terminal. Alguns comandos já são nativos do Bash, como o `cd`, o `ls` e muitos outros.

O Bash pode ser executado interativamente ou não-interativamente. Quando funciona como um veículo de login para a conta da sua máquina, por exemplo, está sendo executado de forma não-interativa, lendo instruções do arquivo `.bashprofile` no seu diretório `home`. Em muitos casos, os comandos dão a opção de criar um arquivo e executá-lo não-interativamente.

Na maior parte das vezes, no entanto, o Bash funciona como Shell interativo, o que significa que comandos e scripts podem ser digitados, suas entradas serão processadas e ao final, será exibida uma saída. É possível também ajustar o modo no qual o Bash será executado com um conjunto de opções similares a qualquer comando. Isso pode ser feito no perfil de Konsole KDE (terminal Bash para ambientes gráficos baseados em KDE), ou equivalentes, dependendo de sua distribuição, ou em um script que é executado na abertura de uma linha de comando.

Uma das opções mais comuns do Bash é a `-r`, que o coloca em modo restrito. No modo restrito, algumas ações, como o uso do comando `cd` ou a mudança de variáveis de ambiente, estão desabilitadas. Alguns administradores colocam o Bash no modo restrito na esperança de limitar o estrago que usuários pouco cautelosos podem causar à rede, mas, frequentemente, Shells restritos são usados na *sandbox* (mecanismo de segurança utilizado para separar programas em execução) – quer dizer, um comando é executado em isolamento por motivos de teste. A opção `-debugger` também é usada para registrar informações de depuração.

O terminal

Antigamente, o Bash era a única maneira de interagir com o Linux, mas a maioria dos sistemas Linux modernos iniciam com algum ambiente gráfica. Para chegar ao prompt de comando em um sistema Linux baseado em interfaces gráficas, é preciso abrir uma janela de terminal. Sistemas que utilizam o ambiente de desktop Gnome normalmente contam com o aplicativo Gnome Terminal. No Ubuntu, por exemplo, o terminal está localizado no menu *Aplicativos/Acessórios* (**figura 1**). Sistemas baseados em KDE, por outro lado, incluem o programa de terminal Konsole, que está no menu *Sistema*. Vários outros programas de terminal estão disponíveis para sistemas Linux. Consulte a documentação da sua distribuição para mais detalhes sobre como chegar ao prompt de comando.

No terminal, esqueça-se do mouse, apesar de ser possível copiar e colar, como revela uma rápida olhada no menu *Editar*. A comunicação com seu sistema é feita através do teclado; digite uma linha e aperte **Enter**. Logicamente, ferramentas modernas como o Konsole ou o Gnome Terminal não são terminais no sentido primordial, mas sim emuladores de terminais. É possível fechar ou minimizar a janela do terminal como é feito com qualquer outra janela no sistema Linux.

Provavelmente, o terminal irá abrir no seu diretório `home`. Digite `ls` para listar o conteúdo do diretório.

Use o comando `cd` (*change directory*) para ir para outro diretório. É preciso também informar o diretório de destino: `$ cd /home/berney/Music`.

A maioria dos terminais permite o uso de um ponto (.) no caminho para representar o diretório corrente. Em outras palavras, um usuário chamado *berney* pode ir do seu diretório `home` para o subdiretório `Documentos` digitando: `$ cd ./Documentos`.

Dois pontos (..) significam *subir um nível na árvore de diretórios*, portanto, se o usuário *berney* quiser voltar ao seu diretório `home`, basta digitar: `$ cd ..`

Muitos sistemas também utilizam o caractere til (~) para representar o diretório `home`, assim, não importa onde o usuário esteja, é sempre possível retornar a este diretório digitando `$ cd ~`.

Caso você venha a se perder durante a navegação na estrutura de diretórios é sempre possível usar o comando `pwd` (*print working directory*, ou "mostre-me o diretório no qual estou trabalhando", em tradução livre) para exibir o nome do diretório corrente.

Para criar um diretório novo, use o comando `mkdir` e dê um nome ao novo diretório: `$ mkdir /home/berney/Musicas/Beatles`. Ou então, caso você já esteja no diretório onde deseja criar a nova pasta, basta digitar: `$ mkdir ./Beatles`.

O comando `cp` copia arquivos. A sintaxe é a seguinte: `cp arquivo_origem arquivo_destino`.

A primeira ação do comando é verificar a existência do arquivo no diretório corrente; no entanto, é possível incluir o caminho com a fonte ou destino para copiar de e para um diretório diferente. Logicamente, é preciso ter as permissões necessárias para acessar os diretórios de origem e destino.

Para excluir um arquivo, use o comando `rm`, e para remover um diretório, o comando `rm -r` ou o `rmdir`. Lembre-se de tomar extremo cuidado com estes comandos, afinal de contas, qualquer erro pode custar irremediavalmente todos os seus dados.

Um resumo destes comandos básicos pode ser visualizado na **tabela 1**. Cada um deles possui opções adicionais que podem ser usadas na linha de comando. Como veremos mais adiante neste artigo, é possível digitar `man` ou `info`, seguido do comando, para obter informações sobre o uso e a sintaxe de cada um. Por exemplo, para saber as várias opções do comando `mkdir`, digite: `man mkdir`.

Em outros artigos ainda nesta edição, você vai conhecer mais sobre os comandos Bash para modificar textos, gerenciar usuários, exami-

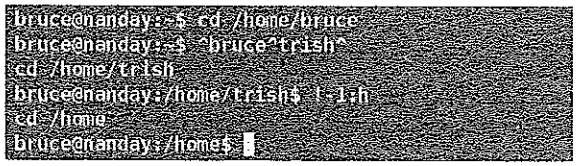

Figura 2: É possível usar vários atalhos de teclado para executar comandos do histórico com ligeiras mudanças. Aqui, a string "bruce" é substituída por "trish" no primeiro caso, preservando-se o início do caminho na segunda.

Quadro 1: Ajuste seu Bash

É possível alterar o modo no qual o Bash opera com comandos `built-in`, por exemplo. O comando `umask` altera as permissões padrão usadas na criação de um arquivo, enquanto que o comando `alias`, pode ser usado para alterar o nome de execução de um comando específico – por exemplo, você pode configurar em seu sistema Debian o comando `ls --color=auto`, para que diretórios e diferentes tipos de arquivos apareçam coloridos. Outra maneira de modificar o Bash é através do comando `shopt` (**figura 5**). O comando `shopt` inclui vários parâmetros interessantes. Por exemplo, `shopt -s cdspell` permite que o Bash corrija pequenos erros de digitação dos diretórios padrão ao utilizar o comando `cd`. Do mesmo modo, `shopt -s checkjobs` lista qualquer tarefa que permanecer ativa e parada após o Shell ser fechado. Esses exemplos do que pode ser feito com o Bash são suficientes para mostrar que ele é muito mais do que um recipiente passivo de comandos. O Bash é cheio de opções e pode ser customizado de acordo com suas necessidades.

nar processos e resolver problemas de rede.

Tabela 1: Alguns comandos básicos do Bash

Comando	Descrição
`ls`	Lista o conteúdo do diretório atual
`cd`	Muda de diretório
`pwd`	Mostra o diretório de trabalho atual
`mkdir`	Cria um diretório
`cp`	Copia arquivo(s)
`rm`	Remove arquivos(s)
`rmdir`	Remove um diretório

Tabela 2: Seções do manual

Seção	Descrição
1	Comandos gerais
2	Chamada do sistema
3	Funções da biblioteca C
4	Arquivos especiais (normalmente dispositivos em /dev) e drivers
5	Formato de arquivos e convenções
6	Jogos e protetores de tela
7	Miscelânea
8	Comandos e daemons de administração de sistema

Histórico

Caso você esteja usando comandos repetitivos no Bash, é possível poupar tempo com o uso do seu histórico de usuário. Armazenada no arquivo `bash_history` no diretório `home`, há uma lista de comandos que foram executados e numerados, sendo o número `1` o mais antigo. É possível usar as teclas de setas para movimentar-se para cima e para baixo neste arquivo, ou usar o comando `history` para ver a lista completa do que está armazenado no histórico.

Se você for um usuário mais ousado, é possível usar uma série de atalhos para executar o comando anterior no histórico. Digitar `!number` executa o comando com esse número. Do mesmo modo, o comando `!-number` estabelece o número de comandos prévios aos quais devemos voltar, e `!string` executa o primeiro comando que inclui determinada string.

Se estiver muito seguro do que quer ou disposto a viver perigosamente, digite `^string1^string2^` para repetir o último comando substituindo a primeira string de caracteres pela segunda. No entanto, caso não esteja seguro dos resultados, adicione `:p` para imprimir o comando encontrado sem executá-lo (**figura 2**).

Documentação

O Bash e os comandos individuais associados a ele já são materiais suficientes para o aprendizado de qualquer iniciante. Felizmente, não é preciso memorizar tudo. Assim como outros sistemas do tipo Unix, o GNU/Linux inclui vários sistemas de auxílio diferentes.

A forma de auxílio mais comum são as páginas de manual (**figura 3**). Elas são divididas em oito seções (**tabela 2**), mas, na maioria das vezes, basta digitar o comando `man` seguido do nome do comando ou arquivo sobre o qual são desejadas informações.

Ao fazer pesquisas mais aprofundadas, considere o uso de `apropos` (comando que lista a descrição, páginas do manual e versão de um comando ou pacote) seguido do item desejado para receber uma

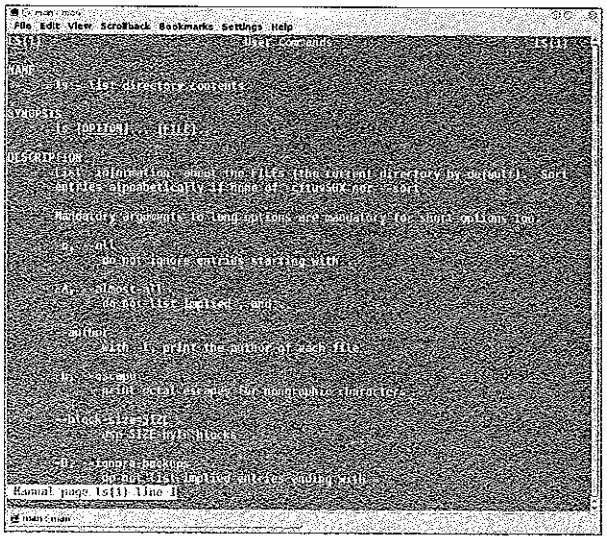

Figura 3: A página de manual para o comando `list`.

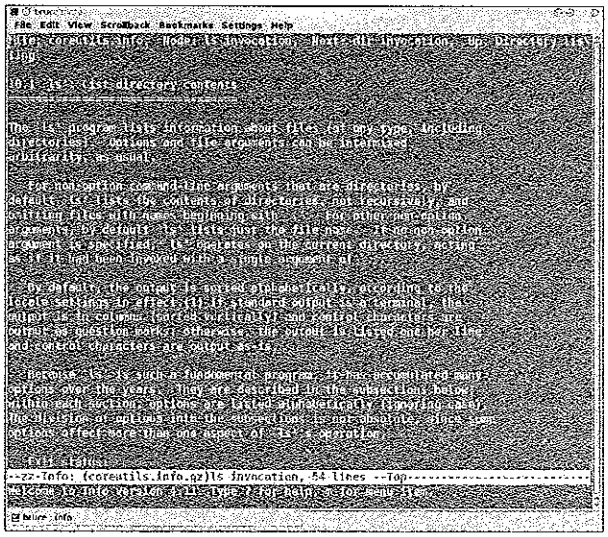

Figura 4: A página `info` para o comando `ls`.

```
bruce@nanday:~$ shopt -s cdspell
bruce@nanday:~$ cd /usr/sahare
/usr/share
bruce@nanday:/usr/share$
```

Figura 5: O comando `shopt` contém muitos recursos interessantes. Aqui, a opção `cdspell` corrige automaticamente os erros quando nomes de diretórios são digitados.

lista de todas as páginas do manual relacionadas. A única desvantagem do `apropos` é que, a menos que você seja muito específico, dúzias de páginas aparecerão, sendo que apenas algumas são relevantes.

Em contraste, se apenas uma pequena informação for necessária, use `whatis` seguido do comando. Por exemplo, se digitar `whatis fdisk`, receberá a linha `fdisk (8) - Partition table manipulator for Linux`. A informação `(8)` refere-se à seção do manual onde podem ser obtidas informações detalhadas. Do mesmo modo, se for preciso identificar um tipo de arquivo, use o comando `type` seguido pelo nome do arquivo.

Durante décadas, as páginas de manual foram o padrão de ajuda no Linux. Porém, há cerca de 10 anos, o Projeto GNU fez do `info` seu formato de ajuda oficial. Mas, em vez de substituir o `man`, o comando `info` simplesmente virou uma alternativa (**figura 4**). Algumas páginas de manual hoje reforçam a ideia de que o arquivo de ajuda completo só está disponível com `info`, mas, na prática, muitos desenvolvedores simplesmente mantêm os dois modelos de ajuda disponíveis, `info` e `man`, focalizando a estrutura do comando nas páginas de manual e as instruções básicas nas páginas de `info`. Mesmo assim, não faz mal nenhum verificar ambas na esperança de encontrar a informação mais completa.

Conclusão

Existem centenas de milhares de comandos que você pode aprender para obter melhor e total controle sobre o computador e suas ações. Leia mais sobre o funcionamento do Shell Bash e seus comandos nas páginas de manual. Outra importante referência é o *Bash Reference Manual* [1]. Leia esse material com o Shell Bash aberto, para experimentar comandos enquanto os aprende. ■

Mais informações

[1] Bash Reference Manual: http://www.gnu.org/software/bash/manual/bashref.html