





the-art-of-command-line / README-pt.md

 **matt-kita** complete, most up-to-date translation into Polish language 

 14 contributors

 447 lines (260 sloc) | 27.9 KB 

 [Čeština](#) · [Deutsch](#) · [Ελληνικά](#) · [English](#) · [Español](#) · [Français](#) · [Indonesia](#) · [Italiano](#) · [日本語](#) · [한국어](#) · [polski](#) · [Português](#) · [Română](#) · [Русский](#) · [Slovenščina](#) · [Українська](#) · [简体中文](#) · [繁體中文](#)

A arte da linha de comando

- [Meta](#)
- [Básico](#)
- [Uso diário](#)
- [Processamento de arquivos e dados](#)
- [Debugando o sistema](#)
- [One-liners](#)
- [Obscuros mas úteis](#)
- [Mais conteúdo](#)
- [Aviso](#)

```
[levy@spud6 ~]$ curl -s 'https://raw.githubusercontent.com/jlevy/the-art-of-command-line/master/README.md' | egrep -o '\w+' | tr -d '\n' | cowsay -W70

/ man yum vi jobs fg bg kill ls less head tail ln chown chmod du df \
mount ip ifconfig dig grep egrep yum pip pip xargs parallel pgrep \
pkill nohup disown lsof trap screen dtach ls percol git fpp updatedb \
ag pandoc xlsxstarlet jq s3cmd s4cmd aws sort uniq cut paste join cut \
join LANG awk sed rename repren shuf sort hd bvi strings grep iconv \
uconv split csplit curl wget httpie iostat netstat top htop dstat \
free vmstat mtr ncdu iftop nethogs ab siege Wireshark tshark strace \
ltrace ldd gdb sar stap perf sysdig dmesg sort uniq a b LC_ALL xargs \
parallel acct_id acct_id expr m4 screen yes cal env look fmt pr fold \
column nl seq bc factor nc dd file stat tac shuf comm hd bvi strings \
tr 7z ldd nm ab strace mtr cssh Wireshark tshark host dig lsof dstat \
iostat htop last w id sar iftop nethogs ss dmesg hdparm lsblk_release \
lshw fortune ddate sl

-----
      ^ ^
      (oo)\_____/
      (_____)  )\/\
      ||-----w||
      ||         ||
[levy@spud6 ~]$
```

Fluência na linha de comando é uma habilidade muitas vezes negligenciada ou considerada obsoleta, porém ela aumenta sua flexibilidade e produtividade como *desenvolvedor* de diversas maneiras, sutis ou não. Este texto descreve uma seleção de notas e dicas de uso da linha de comando que me parecem muito úteis, quando usando o Linux. Algumas dicas são elementares, e outras são mais específicas, sofisticadas ou obscuras. Esta página é curta, mas se você souber usar e lembrar todos os itens que estão aqui, então você está mandando bem.

Muito do que está aqui [originalmente apareceu](#) no [Quora](#), mas dado o interesse por lá, me pareceu importante usar o Github, onde pessoas mais talentosas do que eu, poderiam sugerir melhorias facilmente. Se você descobrir um erro ou algo que poderia ser melhorado, por favor abra um issue ou um PR! (E claro, por favor veja as `meta sections` e PRs/issues existentes, primeiro.)

Meta

Escopo:

- Este guia é destinado tanto aos iniciantes quanto aos usuários mais experientes. Os objetivos são *abrangência* (tudo que é importante), *especificidade* (dar exemplos concretos dos casos de usos mais comuns), e *concisão* (evitar coisas que não são tão essenciais ou digressões que você pode facilmente encontrar pela Internet). Todas as dicas são essenciais em alguma situação ou trazem uma economia notável de tempo em relação a outras alternativas.
- Este guia é escrito para o Linux. Muitos, mas não todos os itens, se aplicam igualmente para o MacOS (ou mesmo o Cygwin).
- O foco está na interatividade com Bash, embora muitas dicas aqui sejam aplicáveis a outras `shells` e também a scripts em Bash, em geral.
- Incluímos tanto comandos no Unix "padrão", quanto comandos que requeiram instalação de pacotes adicionais -- desde que estes sejam importantes o suficiente para merecerem sua inclusão nessa lista.

Notas:

- Para manter este guia em uma única página, o conteúdo implícito será incluído por referência. Você é competente o suficiente para verificar mais detalhes em outros lugares, desde que você já tenha entendido a ideia ou saiba o que procurar no Google. Use `apt-get`, `yum`, `dnf`, `pacman`, `pip` ou `brew` (quando adequado) para instalar novos programas.
- Use [Explainshell](#) para encontrar informações úteis sobre o que fazem os comandos, as opções, pipes, etc.

Básico

- Aprenda o básico sobre Bash. Na verdade, digite `man bash` e pelo menos entenda superficialmente o seu funcionamento; é bastante simples de ler e nem é tão grande assim. Shells alternativas podem ser legais, mas Bash é a mais poderosa e sempre está disponível (aprender *somente* `zsh`, `fish`, etc, é tentador quando você usa o seu próprio notebook, mas restringe você em muitas situações, por exemplo quando você quer usar servidores de outros).
- Aprenda bem pelo menos um editor de texto tradicional. Idealmente o Vim (`vi`), já que nenhum outro funciona tão bem nos terminais aleatórios que a gente encontra por aí (mesmo que você prefira usar o Emacs, um IDE, ou um editor hipster a maior parte do tempo).
- Saiba como ler a documentação com o `man` (para os curiosos, `man man` lista os números das seções, por exemplo, 1 se refere aos comandos "regulares", 5 é sobre arquivos/convenções, e 8 diz respeito a administração). Procure outros documentos do manual com o `apropos`. Saiba que alguns dos comandos não são executáveis, mas sim built-ins (embutidos) no bash, pra esses você poderá conseguir ajuda com `help` e `help -d`.
- Aprenda como fazer redirecionamento de saída e entrada usando `>` e `<` e pipes usando `|`. Aprenda sobre o `stdout` e `stdin`.
- Aprenda sobre a expansão de arquivos glob com `*` (e talvez `?` e `{ ... }`) e entenda as diferenças entre aspas duplas `"` e aspas simples `'`. (Veja mais em variáveis de expansão abaixo.)
- Se familiarize com o gerenciamento de jobs em Bash: `&`, `ctrl-z`, `ctrl-c`, `jobs`, `fg`, `bg`, `kill`, etc.
- Aprenda `ssh`, e o básico de autenticação sem senha, através do `ssh-agent`, `ssh-add`, etc.
- Gerenciamento básico de arquivos: `ls` e `ls -l` (em particular, aprenda o que cada coluna no `ls -l` significa), `less`, `head`, `tail` e `tail -f` (ou melhor ainda, `less +F`), `ln` e `ln -s` (aprenda as diferenças e vantagens de soft links comparados a hard links), `chown`, `chmod`, `du` (para um rápido resumo do uso do disco: `du -sk *`). Para gerenciamento do sistema de arquivos, `df`, `mount`, `fdisk`, `mkfs`, `lsblk`.

- Gerenciamento básico da rede: `ip` ou `ifconfig`, `dig`.
- Saiba bem como usar expressões regulares, e as várias flags para `grep` / `egrep`. As `-i`, `-o`, `-A`, e `-B` são opções que é importante conhecer.
- Aprenda a usar `apt-get`, `yum`, `dnf` ou `pacman` (dependendo da distribuição) para procurar e instalar pacotes. E garanta que você possui o `pip` para instalar ferramentas baseadas em Python (algumas das abaixo são mais fáceis de instalar através do `pip`).

Uso diário

- Usando Bash, use **Tab** para completar argumentos e **ctrl-r** para pesquisar através a história dos comandos.
- Em Bash, utilize **ctrl-w** para deletar a última palavra, e **ctrl-u** para deletar tudo e voltar para o início da linha. Use **alt-b** e **alt-f** para se mover por palavras, **ctrl-k** para apagar até o final da linha, **ctrl-l** para limpar a tela. Consulte `man readline` para todos os keybindings padrões do Bash. Existem muitos. Por exemplo **alt-.** circula através dos argumentos anteriores, e **alt-*** expande um glob.
- Alternativamente, se você adora os keybinds do vi, use `set -o vi`.
- Para ver os comandos recentes, `history`. Existem também muitas abreviações como `!$` (último argumento) e `!!` último comando, embora estes sejam muitas vezes facilmente substituídos por **ctrl-r** e **alt-.**
- Pra voltar para o diretório anterior de trabalho: `cd -`.
- Se você está na metade do caminho ao digitar um comando, mas mudou de ideia, tecele **alt-#** para adicionar um `#` ao início da linha e definir esta como um comentário (ou use **ctrl-a. #. enter**). Mais tarde você poderá recuperar o comando através da `history`.
- Use `xargs` (ou `parallel`). Estes são muito poderosos. Note que você pode controlar como os vários itens são executados por linha (`-L`) assim como o paralelismo (`-P`). Se você não tem certeza se isto é a coisa certa a se fazer, use `xargs echo primeiro`. O `-I{}` também é muito útil. Exemplos:

```
find . -name '*.py' | xargs grep some_function
cat hosts | xargs -I{} ssh root@{} hostname
```

- `pstree -p` é um modo de visualização muito útil da árvore de processos.
- Use `pgrep` e `pkill` para procurar ou sinalizar os processo pelo seu nome (`-f` é muito útil).
- Saiba os vários sinais que você pode enviar para um processo. Por exemplo, para suspender um processo, use `kill -STOP [pid]`. Para saber a lista completas dos sinais, veja `man 7 signal`.

- Use `nohup` ou `disown` se você deseja por o processo no background, executando para sempre.
- Verifique quais processos estão escutando através de `netstat -lntp` ou `ss -plat` (para TCP; adicione `-u` para UDP).
- Veja também `lsof` para abrir sockets e arquivos.
- Em scripts Bash, use `set -x` para debugar a saída. Utilize modos estritos sempre que for possível. Use `set -e` para abortar em caso de erros. Use `set -o pipefail` para também ser restrito a respeito dos erros (embora este tópico seja um pouco sutil). Para scripts mais desenvolvidos, use também `trap`.
- Em Bash scripts, subshells (escrito com parênteses) são formas convenientes de agrupar comandos. Um exemplo comum é temporariamente se mover para um diretório de trabalho diferente, e.g.

```
# faz algo no diretório corrente
(cd /some/other/dir && other-command)
# continua no diretório atual
```

- No Bash, note que existem muitos tipos de variáveis de expansão. Verificando a existência de uma variável: `${name:?error_messages}`. Por exemplo, se um script Bash requer um único argumento, apenas escreva `input_file=${1:?usage: $0 input_file}`. Expansões aritméticas: `i=$(((i + 1) % 5))`. Sequências: `{1..10}`. Aparando as strings: `${var%suffix}` e `${var#prefix}`. Por exemplo, se `var=foo.pdf`, então `echo ${var%.pdf}.txt` imprime `foo.txt`.
- A saída de um comando pode ser tratada como um arquivo através `<(algum comando)`. Por exemplo, comparar um arquivo local `/etc/hosts` com um remoto:

```
diff /etc/hosts <(ssh somehost cat /etc/hosts)
```

- Saiba sobre "documentos aqui" no Bash, como em `cat <<EOF ...`.
- No Bash, redirecionar a saída padrão (stdout) e a saída de erro padrão (stderr) através de: `algum-comando >logfile 2> $1`. Muitas vezes, para garantir que um comando não deixa um arquivo aberto para manipular a entrada padrão, digitando isso no terminal que você está, é uma boa prática adicionar um `</dev/null`.
- Use `man ascii` para visualizar a tabela ASCII, com valores hexadecimais e decimais. Para informações gerais sobre codificações, `man unicode`, `man utf-8`, e `man latin1` são úteis.
- Use `screen` ou `tmux` para multiplexar as telas, especialmente útil em sessões ssh remotas e para desplugar e replugar a uma sessão. Uma alternativa mais simples para a persistência de uma sessão é `dtach`.

- No ssh, saber como realizar um túnel de portas com `-L` ou `-D` (e ocasionalmente `-R`) é útil, para por exemplo acessar sites webs de um servidor remoto.
- Pode ser útil realizar algumas otimizações em suas configurações do ssh; por exemplo, o arquivo `~/.ssh/config` contém configurações para evitar que conexões sejam dropadas em certos ambientes de rede, use compressão (muito útil quando se está usando o scp através de uma conexão lenta), e multiplexação de canais do mesmo servidor com um arquivo de controle local:

```
TCPKeepAlive=yes
ServerAliveInterval=15
ServerAliveCountMax=6
Compression=yes
ControlMaster auto
ControlPath /tmp/%r@%h:%p
ControlPersist yes
```

- Algumas outras opções relevantes para o ssh tem problemas de segurança e devem ser habilitadas com muito cuidado, por exemplo somente para subrede ou host ou em redes confiáveis (de confiança): `StrictHostKeyChecking=no`, `ForwardAgent=yes`
- Para conseguir as permissões em arquivo em forma octal, o que é útil para a configuração do sistema mas não disponível no `ls` e fácil de se confundir, use algo como:

```
stat -c '%A %a %n' /etc/timezone
```

- Para seleção interativas de valores da saída de outro comando, use [percol](#).
- Para interação com arquivos baseados na saída de outro comando (like `git`), use `fpp` ([PathPicker](#)).
- Para um simples servidor web para todos os arquivos do diretório atual (e subdiretórios), disponível para alguém na sua rede, use: `python -m SimpleHTTPServer 7777` (para a porta 7777 e Python 2) e `python -m http.server 7777` (para a porta 7777 e Python 3).

Processamento de arquivos e dados

- Para localizar um arquivo pelo nome no diretório atual, `find . -iname '*something*'` (ou similar). Para procurar um arquivo em qualquer lugar pelo nome, use `locate something` (mas tenha em mente que o `updatedb` pode não ter indexado arquivos criados recentemente).
- Para uma busca mais geral através de arquivos de dados ou de códigos (mais avançado do que `grep -r`), use [ag](#).
- Para converter HTML para texto: `lynx -dump -stdin`.

- Para Markdown, HTML, e todos os demais tipos de conversão de documentos, tente [pandoc](#) .
- Se você precisa manipular XML, `xmlstarlet` é antigo mas é bom.
- Para JSON, `jq` .
- Para Excel ou arquivos CSV, [csvkit](#) que provê `in2csv` , `csvcut` , `csvjoin` , `csvgrep` , etc.
- Para a Amazon S3, [s3cmd](#) é uma forma conveniente e [s4cmd](#) é mais rápido. O [aws](#) da amazon é essencial para outras tarefas relacionadas.
- Aprenda a respeito do `sort` e `uniq` , incluindo as opções do `-u` e `-d` do `uniq --` veja os one-liners abaixo. Veja também `comm` .
- Aprenda a respeito do `cut` , `paste` , e `join` para manipular arquivos de texto. Muitas pessoas usam `cut` mas esquecem do `join` .
- Aprenda a respeito do `wc` para contar novas linhas (`-l`), caracteres (`-m`), palavras (`-m`) e bytes (`-c`).
- Aprenda a respeito do `tee` para copiar da entrada padrão (stdin) para um arquivo e também para a saída padrão (stdout), como no `ls -al | tee file.txt` .
- Aprenda que as configurações de localização afetam várias ferramentas da linha de comando em formas sutis, incluindo a ordem da ordenação e performance. A maioria das instalações do Linux irá definir `LANG` ou outras variáveis de localização para o ingles dos USA. Mas esteja ciente de que a ordem da ordenação irá mudar, caso você altere a localização. E saiba que as rotinas do `i18n` podem fazer o `sort` ou outros comandos executarem *muitas* vezes mais devagar. Em algumas situações (como o conjunto de operações ou as operações únicas abaixo) você pode seguramente eliminar a lentidão das rotinas do `i18n` inteiramente e usar a ordem baseada nos bytes, usando `export LC_ALL=C` .
- Aprenda o básico sobre `awk` e `sed` para obtenção de informações simples de dados. Por exemplo, somar todos os números na terceira coluna de um arquivo de texto: `awk '{ x += $3 } END { print x }'` . Isto é provavelmente 3X mais rápido e 3X mais curto do que o equivalente em Python.
- Para substituir todas as ocorrências de uma string em um lugar, em um ou mais arquivos:

```
perl -pi.bak -e 's/old-string/new-string/g' my-files-*.txt
```

- Para renomear muitos arquivos de uma vez, de acordo com um padrão, use `rename` . Para renomeações mais complexas, [repreen](#) pode ajudar.


```
# Recuperar arquivos de backup foo.bak -> foo:
rename 's/\.bak$//' *.bak
# Renomea completamente o nome dos arquivos, diretórios, e outros conteúdos
repreen --full --preserve-case --from foo --to bar .
```

- Utilize o `shuf` para embaralhar ou selecionar linhas randoms de um arquivo.
- Para as opções do `sort`. Aprenda com as chaves (`-t` e `-k`). Em particular, saiba que precisa escrever `-k1,1` para ordenar somente o primeiro campo; `-k1` significa ordenar de acordo com a linha inteira.
- Ordenação estável (`sort -s`) pode ser útil. Por exemplo, para ordenar primeiramente pelo campo 2, então secundariamente pelo campo 1, você pode usar `sort -k1,1 | sort -s -k2,2`.
- Se você precisa escrever literalmente um `tab` na linha de comando no Bash (por exemplo, para o argumento `-t` do `sort`), pressione **ctrl-v [Tab]** ou escreva `$'\t'` (o último é melhor pois você pode copiar e colar ele).
- As ferramentas padrão para extrair patches de códigos fonte são `diff` e `patch`. Veja também `diffstat` para um resumo de estatísticas de um diff. Note que `diff -r` funciona para diretórios inteiros. Use `diff -r tree1 tree2 | diffstat` para um resumo das alterações.
- Para arquivos binários, use `hd` para um simples dump hexadecimal e `bvi` para edição binária.
- Também para arquivos binários, `strings` (mais `grep`, etc.) deixa você encontrar pedaços de texto.
- Para diffs binários (compressão delta), use `xdelta3`.
- Para converter a codificação de textos, tente `iconv`. Ou `uconv` para uso mais avançado; Este suporta algumas funcionalidades avançadas do Unicode. Por exemplo, este comando transforma o texto para minúsculo e remove todos os acentos (expandindo e removendo eles):

```
uconv -f utf-8 -t utf-8 -x '::~Any-Lower; ::Any-NFD; [:Nonspacing Mark:] >;
```

- Para dividir um arquivo em pedaços, veja `split` (para dividir por tamanho) e `csplit` (para dividir por um padrão).

Use `zless`, `zmore`, `zcat`, and `zgrep` para manipular arquivos comprimidos.

Debugando o sistema

- Para web debug, `curl` e `curl -I` são úteis, ou os equivalentes `wget`, ou uma alternativa mais moderna [httpie](#).
- Para saber o status do disco/cpu/rede, use `iostat`, `netstat`, `top` (ou o `htop` como alternativa melhor), e (especialmente) `dstat`. Bom para obter uma ideia rápida do que está acontecendo em um sistema.
- Para um resumo mais aprofundado do sistema, use [glances](#). Este lhe apresenta vários níveis de estatísticas do sistema em uma janela do terminal. Muito útil para uma rápida verificação em vários subsistemas.
- Para saber o status da memória, execute e entenda a saída do `free` `vmstat`. Em particular, esteja ciente de que o valor "cached", é mantido pelo kernel Linux como um arquivo de cache, então este efetivamente conta como um valor de memória disponível.
- Debugar um sistema java é uma outra historia, mas um simples truque nas máquinas virtuais Oracle ou algum outro tipo de JVM é que você pode executar `kill -3 <pid>` e um completo rastreamento da pilha(stack trace) e resumo do heap (incluindo detalhes geracionais do garbage collector, os quais podem ser altamente informativos) serão vazados para stderr/logs.
- Use [mtr](#) como uma melhor alternativa ao traceroute, para identificar problemas na rede.
- Para verificar o porque de um disco estar cheio, [ncdu](#) economiza bastante tempo em comparação aos comandos usuais como `du -sh *`.
- Para procurar qual socket ou processo está utilizando a banda de rede, tente [iftop](#) ou [nethogs](#).
- A ferramenta `ab` (que vem com o Apache) é muito útil para verificação rápida da performance do servidor web. Para mais complexos testes de carga, tente `siege`.
- Para debugs mais sérios da rede, [wireshark](#), [tshark](#), ou [ngrep](#).
- Aprenda a respeito do `strace` e `ltrace`. Estes podem ser úteis se um programa está falhando, travado, ou quebrando, e você não sabe o por que, ou se você quer obter uma ideia geral da performance. Note que a opção de perfil (`-c`), e a habilidade de se plugar a um processo em execução (`-p`).
- Aprenda a respeito do `ldd` para verificar bibliotecas compartilhadas, e etc.
- Aprenda sobre como se conectar a um processo em execução com o `gdb` e obter informações sobre a stack trace.
- Utilize `/proc`. Este é incrivelmente útil em algumas vezes quando se deseja debugar problemas ao vivo. Exemplos: `/proc/cpuinfo`, `/proc/xxx/cwd`, `/proc/xxx/exe`, `/proc/xxx/fd/`, `/proc/xxx/smmaps`.

- Quando estiver debugando o porque de algo ter dado errado no passado, `sar` pode ser de muita utilidade. Ele exibe as estatísticas históricas da CPU, memória, rede e etc.
- Para análises de performance mais profundas do sistema, dê uma olhada em `stap` (`SystemTap`), `perf`, e `sysdig`.
- Confirme qual a sua distribuição do Linux usando (funciona na maioria das distros):
`lsb_release -a`.
- Use `dmesg` sempre que algo estiver agindo de maneira estranha (isto pode ser um problema de hardware ou problema de driver).

One-liners

Alguns exemplos de como reunir os comandos.

- O seguinte é notavelmente e frequentemente útil: muitas vezes você quer obter a interseção, união e a diferença de arquivos de texto através de `sort / uniq`. Suponha que `a` e `b` são arquivos de texto que são "uniqued" únicos. Esse modo é rápido, e funciona em arquivos de tamanhos arbitrários, podem até possuírem gigabytes. (Sorting não é limitado por memória, embora você possa precisar usar a opção `-T` se `/tmp` está em uma partição pequena.) Veja também a nota sobre `LC_ALL` acima e as opções `-u` do `sort` (vamos deixar isso claro abaixo).

```
cat a b | sort | uniq > c    # c is a union b
cat a b | sort | uniq -d > c  # c is a intersect b
cat a b b | sort | uniq -u > c  # c is set difference a - b
```

- Use `grep . *` para visualmente examinar todo o conteúdo de todos os arquivos de um diretório, por exemplo, para diretórios com arquivos de configurações, como `/sys`, `/proc`, `/etc`.
- Somar todos os números em uma terceira coluna de um arquivo de texto (isto é provavelmente 3X mais rápido e 3X menos linhas de código do que o equivalente em Python).

```
awk '{ x += $3 } END { print x }' myfile
```

- Se você quer visualizar tamanhos/datas em uma árvore de arquivos, isto é como um `ls -l` recursivo, mas é mais fácil de ler do que `ls -lR`:

```
find . -type f -ls
```

- Utilize `xargs` ou `parallel` sempre que você puder. Note que você pode controlar quantos item é executado por linha (`-L`) assim como o paralelismo (`-P`). Se você não tem certeza de que esta é a coisa certa a se fazer, utilize `xargs echo` primeiro.

```
find . -name '*.py' | xargs grep some_function
cat hosts | xargs -I{} ssh root@{} hostname
```

- Digamos que você tenha um arquivo de texto, como um log do servidor web, e um certo valor que aparece em algumas linhas, como por exemplo o parâmetro `acct_id` que está presente na URL. Se você quer um cálculo de quantas requisições para este `acct_id`.

```
cat access.log | egrep -o 'acct_id=[0-9]+' | cut -d= -f2 | sort | uniq -c
```

- Execute esta função para obter uma dica random deste documento (analisa a sintaxe Markdown e extrai um item)

```
function taocl() {
  curl -s https://raw.githubusercontent.com/jlevy/the-art-of-command-line/
  pandoc -f markdown -t html |
  xmlstarlet fo --html --dropdtd |
  xmlstarlet sel -t -v "(html/body/ul/li[count(p)>0])[$RANDOM mod last()]
  xmlstarlet unesc | fmt -80
}
```

Obscuros mas úteis

- `expr` : executa operações booleanas ou aritméticas ou avalia expressões regulares.
- `m4` : simples processador de macros.
- `yes` : imprime uma string muitas vezes.
- `cal` : calendário legal.
- `env` : executa um comando (útil em scripts).
- `printenv` : imprime as variáveis de ambiente (útil em debug e scripts).
- `look` : procura palavras inglesas (ou linhas em um arquivo) começando com uma string.
- `cut` e `paste` e `join` : manipulação de dados.
- `fmt` : formata parágrafos de texto.
- `pr` : formata textos em páginas/colunas.
- `fold` : envolve linhas de texto.
- `column` : formata texto em colunas ou tabelas.

- `expand` e `unexpand` : converte entre tabs e espaços.
- `nl` : adiciona números as linhas.
- `seq` : imprime números.
- `bc` : calculadora.
- `factor` : fatora inteiros.
- `gpg` : criptografa e assina arquivos.
- `toe` : tabela de entradas dos tipos de terminais.
- `nc` : ferramenta de debug de rede e transferência de dados.
- `socat` : socket relay e encaminhamento de portas tcp (similar ao `netcat`)
- `slurm` : visualização do tráfego da rede.
- `dd` : move os dados entre arquivos ou dispositivos.
- `file` : identifica o tipo do arquivo.
- `tree` : mostra os diretórios e subdiretórios como um árvore de dependências; como `ls` mas recursivo.
- `stat` : informações do arquivo.
- `tac` : imprime arquivos na ordem reversa.
- `shuf` : seleção random de linhas de um arquivo.
- `comm` : compara uma lista de arquivos ordenadas linha por linha.
- `pv` : monitora o progresso dos dados através de um pipe.
- `hd` e `bvi` : dump ou edita arquivos binários.
- `strings` : extrai texto de arquivos binários.
- `tr` : tradução e manipulação de caracteres.
- `iconv` ou `uconv` : conversor de codificações de texto.
- `split` e `csplit` : divisão de arquivos.
- `units` : conversor de unidades e cálculos; converte furlongs por quinzena para twips per blink (veja também `/usr/share/units/definitions.units`)
- `7z` : Compressor de arquivos de alto desempenho.
- `ldd` : informações dinâmicas das bibliotecas.

- `nm` : símbolos de arquivos objetos.
- `ab` : benchmarking para web servers.
- `strace` : Debug para chamadas de sistema.
- `mtr` : melhor traceroute para debugar a rede.
- `cssh` : Visualização concorrente da shell.
- `rsync` : Sincroniza arquivos e pastas através do SSH.
- `wireshark` e `tshark` : captura de pacotes e debug de rede.
- `ngrep` : grep para a camada de rede.
- `host` e `dig` : Consultas DNS.
- `lsof` : Arquivo de descritores dos processos e informações dos sockets.
- `dstat` : Estatísticas úteis do sistema.
- `glances` : Resumo de alto nível, de multi subsistemas.
- `iostat` : Estatísticas de uso do CPU e do disco.
- `htop` : Versão do top melhorada.
- `last` : histórico de logins.
- `w` : quem está logado.
- `id` : Informações sobre a identidade do user/group.
- `sar` : histórico dos estados do sistema.
- `iftop` ou `nethogs` : Utilização da rede por sockets ou processos.
- `ss` : Estatísticas dos sockets.
- `dmesg` : Mensagens de erro do sistema e do boot.
- `hdparm` : Manipulação/performance de discos SATA/ATA.
- `lsblk` : Lista os blocos dos dispositivos: uma visualização em forma de árvore dos seus discos e partições do disco.
- `lshw` e `lspci` : informações do hardware, incluindo RAID, gráficos, etc.
- `fortune` , `ddate` , e `sl` : um, bem, isto depende de você considerar locomotivas a vapor e citações Zippy "úteis".

Mais conteúdo

- [awesome-shell](#): Uma lista refinada de ferramentas da shell e outros recursos.
- [Strict mode](#) para escrever shell scripts melhores.

Aviso

Com a exceção de tarefas muito pequenas, código é normalmente escrito para que outros possam ler. Junto com o poder vem a responsabilidade. O fato de você *poder* fazer algo usando Bash não significa necessariamente que você deve! ;)

Licença



Este trabalho está licenciado com uma [Creative Commons Attribution-ShareAlike 4.0 International License](#).

[Give feedback](#)