103.5 Criar, monitorar e finalizar processos

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 103.5

Peso

4

Áreas chave de conhecimento

- Executar processos em primeiro e segundo plano.
- Marcar um programa para que continue a rodar depois do logout.
- Monitorar processos ativos.
- Selecionar e ordenar processos para serem exibidos.
- Enviar sinais para os processos.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- `&`
- `bg`
- `fg`
- `jobs`
- `kill`
- `nohup`
- `ps`
- `top`
- `free`

- `uptime`
- `pgrep`
- `pkill`
- `killall`
- `watch`
- `screen`
- `tmux`

Linux Professional Institute

103.5 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	103 Comandos GNU e Unix
Objetivo:	103.5 Criar, monitorar e eliminar processos
Lição:	1 de 2

Introdução

A cada vez que invocamos um comando, um ou mais processos são iniciados. Um administrador de sistema experiente não só precisa criar processos, mas também ser capaz de controlá-los e enviar diferentes tipos de sinais a eles se e quando necessário. Nesta lição, discorreremos sobre o controle de trabalhos e o monitoramento de processos.

Controle de jobs

Jobs (trabalhos) são processos iniciados de forma interativa através de um terminal, enviados para o segundo plano e ainda não finalizados. Para descobrir mais sobre os jobs ativos (e seus status) em seu sistema Linux, execute `jobs`:

```
$ jobs
```

O comando `jobs` acima não produziu nenhuma saída, o que significa que não há jobs ativos no momento. Vamos criar nosso primeiro job executando um comando que leva algum tempo para terminar de ser executado (o comando `sleep` com um parâmetro de `60`) e — durante a execução — pressionamos `Ctrl` + `Z`:

```
$ sleep 60
^Z
[1]+  Stopped                 sleep 60
```

A execução do comando foi interrompida (ou, mais exatamente, suspensa) e o prompt de comando está disponível novamente. Se você procurar por jobs uma segunda vez, encontrará aquele que foi *suspenso*:

```
$ jobs
[1]+  Stopped                 sleep 60
```

Vamos entender melhor essa saída:

`[1]`

Este número é o identificador do trabalho e pode ser usado --precedido por um símbolo de porcentagem (%)—para alterar o status do job com os utilitários `fg`, `bg` e `kill` (como ensinaremos mais tarde).

`+`

O sinal de mais indica o job atual por padrão (ou seja, o último a ser suspenso ou mandado ao segundo plano). O job anterior é sinalizado com um sinal de menos (–). Quaisquer outros jobs anteriores não são sinalizados.

`Stopped`

Descrição do status do job.

`sleep 60`

O comando ou job em si.

Com a opção `-l`, o comando jobs exibe adicionalmente o identificador do processo (PID) logo antes do status:

```
$ jobs -l
[1]+  1114 Stopped                 sleep 60
```

As demais opções possíveis para jobs são:

`-n`

Lista apenas os processos que mudaram de status desde a última notificação. Os possíveis status incluem `Running`, `Stopped`, `Terminated` ou `Done`.

`-p`

Lista os IDs do processo.

`-r`

Lista apenas os jobs em execução.

`-s`

Lista apenas os jobs interrompidos (ou suspensos).

NOTE Lembre-se, um job tem um *ID de trabalho* e um *ID de processo* (PID).

Especificação do trabalho

O comando `jobs`, a exemplo de outros utilitários como `fg`, `bg` e `kill` (que você verá na próxima seção), precisa de uma especificação de trabalho (ou `jobspec`) para agir sobre um job particular. Como acabamos de ver, este pode ser — e normalmente é — o ID do trabalho precedido por `%`. No entanto, outras especificações de trabalho também são possíveis. Vamos dar uma olhada nelas:

`%n`

Job cujo número de ID é `n`:

```
$ jobs %1
[1]+  Stopped                 sleep 60
```

`%str`

Job cuja linha de comando começa com `str`:

```
$ jobs %sl
[1]+  Stopped                 sleep 60
```

`%?str`

Job cuja linha de comando contém `str`:

```
$ jobs %?le
```

```
[1]+  Stopped                 sleep 60
```

`%+` ou `%%`

Job atual (o que foi iniciado por último em segundo plano ou suspenso do primeiro plano):

```
$ jobs %+
[1]+  Stopped                 sleep 60
```

`%-`

Job anterior (aquele que era `%+` antes do padrão, que é o atual):

```
$ jobs %-
[1]+  Stopped                 sleep 60
```

No nosso caso, como há apenas um job, ele é o atual e o anterior.

Status do trabalho: suspensão, primeiro plano e segundo plano

Uma vez que um job está em segundo plano ou foi suspenso, podemos fazer três coisas com ele:

1. Levá-lo ao primeiro plano com `fg`:

   ```
   $ fg %1
   sleep 60
   ```

   `fg` move o job especificado para o primeiro plano e o torna o job atual. A seguir podemos esperar que ele seja concluído, interrompê-lo novamente com `Ctrl` + `Z` ou encerrá-lo com `Ctrl` + `C`.

2. Colocá-lo em segundo plano com `bg`:

   ```
   $ bg %1
   [1]+ sleep 60 &
   ```

   Uma vez no segundo plano, o trabalho pode ser trazido de volta ao primeiro plano com `fg` ou eliminado (veja abaixo). Observe o e comercial (&) indicando que o trabalho foi enviado para o segundo plano. Na verdade, também podemos usar o e comercial para iniciar um processo diretamente em segundo plano:

```
$ sleep 100 &
[2] 970
```

Junto com o ID do novo trabalho (`[2]`), também obtemos o ID do processo (`970`). Agora, ambos os trabalhos estão rodando em segundo plano:

```
$ jobs
[1]-  Running                 sleep 60 &
[2]+  Running                 sleep 100 &
```

Um pouco mais tarde, o primeiro trabalho termina de ser executado:

```
$ jobs
[1]-  Done                    sleep 60
[2]+  Running                 sleep 100 &
```

3. Encerrá-lo através de um sinal `SIGTERM` com `kill`:

```
$ kill %2
```

Para ter certeza de que o job foi encerrado, rode `jobs` novamente:

```
$ jobs
[2]+  Terminated              sleep 100
```

NOTE Se nenhum trabalho for especificado, `fg` e `bg` agirão sobre o job padrão atual. `kill`, no entanto, sempre precisa de uma especificação de trabalho.

Trabalhos desvinculados: `nohup`

Os jobs que vimos nas seções anteriores estavam todos vinculados à sessão do usuário que os invocou. Isso significa que, se a sessão for encerrada, os jobs serão perdidos. No entanto, é possível desvincular jobs de sessões e executá-los mesmo após o encerramento da sessão. Isso é feito com o comando `nohup` ("no hangup"). A sintaxe é a seguinte:

```
nohup COMMAND &
```

Lembre-se, o `&` envia o processo para o segundo plano e libera o terminal em que estamos trabalhando.

Vamos desvincular o job em segundo plano `ping localhost` da sessão atual:

```
$ nohup ping localhost &
[1] 1251
$ nohup: ignoring input and appending output to 'nohup.out'
^C
```

A saída mostra o ID do trabalho (`[1]`) e o PID (`1251`), seguido por uma mensagem nos informando sobre o arquivo `nohup.out`. Este é o arquivo padrão no qual `stdout` e `stderr` serão salvos. Agora podemos pressionar `Ctrl` + `C` para liberar o prompt de comando, fechar a sessão, iniciar outra como `root` e usar `tail -f` para verificar se o comando está rodando e a saída está sendo escrita no arquivo padrão:

```
$ exit
logout
$ tail -f /home/carol/nohup.out
64 bytes from localhost (::1): icmp_seq=3 ttl=64 time=0.070 ms
64 bytes from localhost (::1): icmp_seq=4 ttl=64 time=0.068 ms
64 bytes from localhost (::1): icmp_seq=5 ttl=64 time=0.070 ms
^C
```

TIP Em vez de usar o `nohup.out` padrão, poderíamos ter especificado um arquivo de saída à escolha com `nohup ping localhost > /path/to/your/file &`.

Se quisermos encerrar o processo, devemos especificar seu PID:

```
# kill 1251
```

Monitoramento de processos

Um processo ou tarefa é uma instância de um programa em execução. Assim, criamos novos processos toda vez que digitamos comandos no terminal.

O comando `watch` executa um programa periodicamente (por padrão, a cada 2 segundos) e nos permite *observar* a mudança da saída do programa ao longo do tempo. Por exemplo, podemos monitorar como a média de trabalho muda conforme mais processos são executados digitando `watch`

`uptime`:

```
Every  2.0s: uptime          debian: Tue Aug 20 23:31:27 2019

 23:31:27 up 21 min,  1 user,  load average: 0.00, 0.00, 0.00
```

O comando roda até ser interrompido, então teríamos de pará-lo com `Ctrl` + `C`. Obtemos duas linhas na saída: a primeira corresponde ao `watch` e nos informa a frequência com que o comando será executado (`Every 2.0s: uptime`), qual o comando/programa a observar (`uptime`) além do nome do host e a data (`debian: Tue Aug 20 23:31:27 2019`). A segunda linha da saída é o tempo de atividade e inclui a hora (`23:31:27`), o tempo em que o sistema está ativo (`up 21 min`), o número de usuários ativos (`1 user`) e a carga média do sistema ou o número de processos em execução ou em estado de espera nos últimos 1, 5 e 15 minutos (`load average: 0.00, 0.00, 0.00`).

Da mesma forma, você pode verificar o uso de memória à medida que novos processos são criados com `watch free`:

```
Every  2.0s: free            debian: Tue Aug 20 23:43:37 2019

 23:43:37 up 24 min,  1 user,  load average: 0.00, 0.00, 0.00
              total        used        free      shared  buff/cache   available
Mem:       16274868      493984    14729396       35064     1051488    15462040
Swap:      16777212           0    16777212
```

Para alterar o intervalo de atualização de `watch`, use as opções `-n` ou `--interval`, mais o número de segundos, como em:

```
$ watch -n 5 free
```

Agora o comando `free` será executado a cada 5 segundos.

Para saber mais sobre as opções de `uptime`, `free` e `watch`, consulte as páginas de manual correspondentes.

NOTE As informações fornecidas por `uptime` e `free` também são integradas nas ferramentas mais abrangentes `top` e `ps` (veja abaixo).

Enviando sinais para processos: `kill`

Cada processo possui um identificador de processo ou PID exclusivo. Uma maneira de descobrir o PID de um processo é usar o comando `pgrep` seguido pelo nome do processo:

```
$ pgrep sleep
1201
```

NOTE O identificador de um processo também pode ser descoberto com o comando `pidof` (p.ex. `pidof sleep`).

Como no caso do `pgrep`, o comando `pkill` elimina um processo com base em seu nome:

```
$ pkill sleep
[1]+  Terminated              sleep 60
```

Para eliminar várias instâncias do mesmo processo, o comando `killall` pode ser usado:

```
$ sleep 60 &
[1] 1246
$ sleep 70 &
[2] 1247
$ killall sleep
[1]-  Terminated              sleep 60
[2]+  Terminated              sleep 70
```

Tanto `pkill` quanto `killall` funcionam da mesma maneira que `kill`, ou seja, enviam um sinal de encerramento para o(s) processo(s) especificado(s). Se nenhum sinal for fornecido, o padrão `SIGTERM` é enviado. No entanto, `kill` só aceita um ID de trabalho ou de processo como argumento.

Os sinais podem ser especificados por:

- Nome:

  ```
  $ kill -SIGHUP 1247
  ```

- Número:

```
$ kill -1 1247
```

- Opção:

```
$ kill -s SIGHUP 1247
```

Para fazer com que `kill` funcione de forma semelhante a `pkill` ou `killall` (evitando os comandos para descobrir os PIDs correspondentes), podemos usar a substituição de comandos:

```
$ kill -1 $(pgrep sleep)
```

Como você já deve saber, uma sintaxe alternativa é `kill -1 `pgrep sleep``.

TIP Para uma lista exaustiva de todos os sinais de `kill` e seus códigos, digite `kill -l` no terminal. Use `-KILL` (`-9` ou `-s KILL`) para eliminar processos rebeldes quando todos os outros sinais falharem.

`top` e `ps`

Quando se trata de monitoramento de processos, duas ferramentas inestimáveis são `top` e `ps`. Enquanto o primeiro produz resultados de maneira dinâmica, o último o faz de maneira estática. Em todos os casos, ambos são excelentes utilitários para se ter uma visão abrangente de todos os processos do sistema.

Interação com `top`

Para chamar o `top`, basta digitar `top`:

```
$ top

top - 11:10:29 up  2:21,  1 user,  load average: 0,11, 0,20, 0,14
Tasks:  73 total,   1 running,  72 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,0 us,  0,3 sy,  0,0 ni, 99,7 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem :  1020332 total,   909492 free,    38796 used,    72044 buff/cache
KiB Swap:  1046524 total,  1046524 free,        0 used.   873264 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
   436 carol     20   0   42696   3624   3060 R  0,7  0,4   0:00.30 top
     4 root      20   0       0      0      0 S  0,3  0,0   0:00.12 kworker/0:0
```

```
  399 root      20   0   95204   6748   5780 S  0,3  0,7   0:00.22 sshd
    1 root      20   0   56872   6596   5208 S  0,0  0,6   0:01.29 systemd
    2 root      20   0       0      0      0 S  0,0  0,0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0,0  0,0   0:00.02 ksoftirqd/0
    5 root       0 -20       0      0      0 S  0,0  0,0   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S  0,0  0,0   0:00.00 kworker/u2:0
    7 root      20   0       0      0      0 S  0,0  0,0   0:00.08 rcu_sched
    8 root      20   0       0      0      0 S  0,0  0,0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S  0,0  0,0   0:00.00 migration/0
   10 root       0 -20       0      0      0 S  0,0  0,0   0:00.00 lru-add-drain
   (...)
```

O `top` permite uma certa interação do usuário. Por padrão, a saída é classificada pela porcentagem de tempo da CPU usada por cada processo em ordem decrescente. Esse comportamento pode ser modificado pressionando as seguintes teclas de dentro do `top`:

`M`

Classificar por uso da *memória*.

`N`

Classificar pelo *número* ID do processo.

`T`

Classificar por *tempo* de execução.

`P`

Classificar por *porcentagem* de uso da CPU.

TIP Para alternar entre a ordem crescente/decrescente, basta pressionar `R`.

Outras teclas interessantes para interagir com `top` são:

`?` ou `h`

Ajuda.

`k`

Elimina um processo. O `top` pedirá o `PID` do processo a encerrar, assim como o sinal a ser enviado (por padrão, `SIGTERM` ou `15`).

`r`

Altera a prioridade de um processo (`renice`). `top` pede o valor de `nice`. Os valores possíveis

variam de -20 a 19, mas apenas o superusuário (`root`) pode definir um valor negativo ou inferior ao atual.

`u`

Lista os processos de um determinado usuário (por padrão, são mostrados os processos de todos os usuários).

`c`

Mostra os caminhos absolutos dos programas e diferencia os processos do espaço do usuário dos processos do espaço do kernel (entre colchetes).

`V`

Visão de floresta/hierárquica dos processos.

`t` e `m`

Mudam a aparência das leituras da CPU e da memória, respectivamente, em um ciclo de quatro estágios: os dois primeiros pressionamentos mostram barras de progresso, o terceiro oculta a barra e o quarto a traz de volta.

`W`

Salva as definições de configuração em `~/.toprc`.

TIP

Uma versão mais sofisticada e amigável de `top` é `htop`. Outra alternativa, talvez mais exaustiva, é `atop`. Se ainda não estiverem instalados em seu sistema, use seu gerenciador de pacotes para instalá-los e experimentá-los.

Explicação da saída de `top`

A saída de `top` é dividida em duas áreas: a *área de resumo* e a *área de tarefas*.

A área de resumo em `top`

A *área de resumo* é composta pelas cinco linhas superiores e nos fornece as seguintes informações:

- `top - 11:10:29 up 2:21, 1 user, load average: 0,11, 0,20, 0,14`
  - hora atual (em formato de 24 horas): `11:20:29`
  - tempo de atividade (há quanto tempo o sistema está ativo e funcionando): `up 2:21`
  - número de usuários logados e carga média da CPU nos últimos 1, 5 e 15 minutos, respectivamente: `load average: 0,11, 0,20, 0,14`

- `Tasks: 73 total, 1 running, 72 sleeping, 0 stopped, 0 zombie` (informações sobre os processos)
  - número total de processos em modo ativo: `73 total`
  - em execução (os que estão sendo executados): `1 running`
  - em espera (os que estão esperando para retomar a execução): `72 sleeping`
  - interrompidos (por um sinal de controle do trabalho): `0 stopped`
  - zumbi (os que concluíram a execução mas ainda estão esperando que o processo pai os remova da tabela de processos): `0 zombie`
- `%Cpu(s): 0,0 us, 0,3 sy, 0,0 ni, 99,7 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st` (porcentagem de tempo da CPU gasto em:)
  - processos de usuário: `0,0 us`
  - processos do sistema/kernel: `0,4 sy`
  - processos com um valor *nice* configurado — quanto mais alto o valor nice, menor a prioridade: `0,0 ni`
  - nada — tempo ocioso da CPU: `99,7 id`
  - processos aguardando operações de I/O: `0,0 wa`
  - processos atendendo interrupções de hardware — periféricos enviando ao processador sinais que precisam de atenção: `0,0 hi`
  - processos atendendo interrupções de software: `0,0 si`
  - processos atendendo tarefas de outras máquinas virtuais em um ambiente virtual, e que portanto roubam tempo: `0,0 st`
- `KiB Mem : 1020332 total, 909492 free, 38796 used, 72044 buff/cache` (informações da memória em kilobytes)
  - quantidade total de memória: `1020332 total`
  - memória não utilizada: `909492 free`
  - memória em uso: `38796 used`
  - memória armazenada em buffer e em cache para evitar acesso excessivo ao disco: `72044 buff/cache`

    Note como o `total` é a soma dos outros três valores — `free`, `used` e `buff/cache` — (aproximadamente 1 GB em nosso caso).

- `KiB Swap: 1046524 total, 1046524 free, 0 used. 873264 avail Mem` (informações de troca em kilobytes)
  - quantidade total de espaço de troca: `1046524 total`
  - espaço de troca não utilizado: `1046524 free`
  - espaço de troca em uso: `0 used`
  - quantidade de memória de troca que pode ser alocada a processos sem causar mais trocas: `873264 avail Mem`

A área de tarefas em `top`: Campos e colunas

Abaixo da *área de resumo* fica a *área de tarefas*, que inclui uma série de *campos* e *colunas* que informam sobre os processos em execução:

`PID`

Identificador do processo.

`USER`

Usuário emissor do comando que originou o processo

`PR`

Prioridade de processo para o kernel.

`NI`

Valor nice do processo. Os valores mais baixos têm mais prioridade que os valores altos.

`VIRT`

Quantidade total de memória usada por processo (incluindo Troca).

`RES`

Memória RAM usada por processo.

`SHR`

Memória compartilhada do processo com outros processos.

`S`

Status do processo. Os valores incluem: `S` (suspensão interrompível—esperando que um evento termine), `R` (executável—em execução ou na fila para ser executado) ou `Z` (zumbi—processos filhos encerrados cujas estruturas de dados ainda não foram removidas da tabela de processos).

`%CPU`

Porcentagem de CPU usada pelo processo.

`%MEM`

Porcentagem de RAM utilizada pelo processo, ou seja, o valor de `RES` expresso em porcentagem.

`TIME+`

Tempo total de atividade do processo.

`COMMAND`

Nome do comando/programa que gerou o processo.

Visualizando processos estaticamente: `ps`

Como dito acima, o `ps` exibe um instantâneo dos processos. Para ver todos os processos com um terminal (tty), digite `ps a`:

```
$ ps a
  PID TTY      STAT   TIME COMMAND
  386 tty1     Ss+    0:00 /sbin/agetty --noclear tty1 linux
  424 tty7     Ssl+   0:00 /usr/lib/xorg/Xorg :0 -seat seat0 (...)
  655 pts/0    Ss     0:00 -bash
 1186 pts/0    R+     0:00 ps a
 (...)
```

Explicação da sintaxe e da saída das opções de `ps`

Com relação às opções, o `ps` aceita três estilos diferentes: BSD, UNIX e GNU. Vamos ver como cada um desses estilos se comportaria ao relatar informações sobre a ID de um processo específico:

BSD

As opções não requerem um traço inicial:

```
$ ps p 811
  PID TTY      STAT   TIME COMMAND
  811 pts/0    S      0:00 -su
```

UNIX

As opções requerem um traço inicial:

```
$ ps -p 811
  PID TTY          TIME CMD
  811 pts/0    00:00:00 bash
```

GNU

As opções recebem um duplo traço inicial:

```
$ ps --pid 811
  PID TTY          TIME CMD
  811 pts/0    00:00:00 bash
```

Nos três casos, o `ps` relata informações sobre o processo cujo `PID` é `811` — neste caso, o `bash`.

Da mesma forma, podemos usar o `ps` para pesquisar os processos iniciados por um usuário determinado:

- `ps U carol` (BSD)
- `ps -u carol` (UNIX)
- `ps --user carol` (GNU)

Vamos verificar os processos iniciados por `carol`:

```
$ ps U carol
  PID TTY      STAT   TIME COMMAND
  811 pts/0    S      0:00 -su
  898 pts/0    R+     0:00 ps U carol
```

Ela iniciou dois processos: `bash` (`-su`) e `ps` (`ps U carol`). A coluna `STAT` informa o estado do processo (veja abaixo).

Podemos obter o melhor do `ps` combinando algumas de suas opções. Um comando muito útil (produzindo uma saída semelhante à de `top`) é `ps aux` (estilo BSD). Nesse caso, os processos de todos os shells (não apenas o atual) são mostrados. O significado das opções é o seguinte:

`a`

Mostra processos que estão vinculados a um `tty` ou terminal.

`u`

Exibe formato orientado ao usuário.

`x`

Mostra processos que não estão vinculados a um `tty` ou terminal.

```
$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1 204504  6780 ?        Ss   14:04   0:00 /sbin/init
root         2  0.0  0.0      0     0 ?        S    14:04   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    14:04   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   14:04   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    14:04   0:00 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S    14:04   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    14:04   0:00 [migration/0]
(...)
```

Entenda melhor as colunas:

`USER`

Proprietário do processo.

`PID`

Identificador do processo.

`%CPU`

Porcentagem de CPU usada.

`%MEM`

Porcentagem de memória física usada.

`VSZ`

Memória virtual do processo em KiB.

`RSS`

Memória física não trocada usada pelo processo em KiB.

`TT`

Terminal (`tty`) que controla o processo.

`STAT`

Código que representa o estado do processo. Além de `S`, `R` e `Z` (que vimos ao descrever a saída de `top`), outros valores possíveis seriam: `D` (dormente sem interrupção — geralmente esperando por I/O), `T` (interrompido — normalmente por um sinal de controle). Alguns modificadores extras incluem: `<` (prioridade maior que o convencional), `N` (prioridade menor que o convencional), ou `+` (no grupo de processos em primeiro plano).

`STARTED`

Horário de início do processo.

`TIME`

Tempo de CPU acumulado.

`COMMAND`

Comando que iniciou o processo.

Exercícios Guiados

1. `oneko` é um programa divertido que mostra um gato perseguindo o cursor do mouse. Se ele ainda não estiver instalado em seu sistema desktop, instale-o usando o gerenciador de pacotes de sua distribuição. Vamos usá-lo para estudar o controle de jobs.
   - Inicie o programa. Qual o procedimento?
   - Mova o cursor do mouse para ver como o gato o persegue. Agora suspenda o processo. Qual o procedimento? Qual é a saída?
   - Verifique quantos jobs estão presentes atualmente. O que você digita? Qual é a saída?
   - Agora envie-o para o segundo plano especificando seu ID de trabalho. Qual é a saída? Como você pode saber se o job está sendo executado em segundo plano?
   - Finalmente, encerre o job especificando seu ID de trabalho. O que você digita?
2. Descubra o PID de todos os processos gerados pelo servidor web *Apache HTTPD* (`apache2`) com dois comandos diferentes:
3. Encerre todos os processos `apache2` sem usar seus PIDs e com dois comandos diferentes:
4. Suponha que você tem de encerrar todas as instâncias do `apache2` e não tem tempo para descobrir quais são os PIDs correspondentes. Como fazer isso usando `kill` com o sinal padrão `SIGTERM` em uma só linha?

5. Inicie o `top` e interaja com ele executando o seguinte:
   - Exibir uma visão em floresta dos processos:

   - Exibir caminhos completos de processos, diferenciando entre espaço de usuário e espaço de kernel:

6. Digite o comando `ps` para exibir todos os processos iniciados pelo usuário do servidor web *Apache HTTPD* (`www-data`):
   - Usando a sintaxe do BSD:

   - Usando a sintaxe do UNIX:

   - Usando a sintaxe do GNU:

Exercícios Exploratórios

1. O sinal `SIGHUP` pode ser usado como forma de reiniciar certos daemons. Com o servidor web *Apache HTTPD*—por exemplo—enviar `SIGHUP` para o processo pai (iniciado por `init`) elimina seus filhos. O pai, no entanto, relê seus arquivos de configuração, reabre os arquivos de log e gera um novo conjunto de filhos. Execute as seguintes tarefas:
   - Inicie o servidor web:
   - Certifique-se de conhecer o PID do processo pai:
   - Faça o servidor web Apache HTTPD reiniciar enviando o sinal `SIGHUP` para o processo pai:
   - Verifique se o pai não foi eliminado e se novos filhos foram gerados:
2. Embora inicialmente estática, a saída de `ps` pode ser *tornada* dinâmica combinando `ps` e `watch`. Vamos monitorar as novas conexões do servidor web *Apache HTTPD*. Antes de realizar as tarefas descritas abaixo, recomendamos ler a descrição da diretiva `MaxConnectionsPerChild` em Apache MPM Common Directives.
   - Adicione a diretiva `MaxConnectionsPerChild` com um valor de `1` no arquivo de configuração de `apache2`—no *Debian* e seus derivativos ele fica em `/etc/apache2/apache2.conf`; na família *CentOS*, em `/etc/httpd/conf/httpd.conf`. Não esqueça de reiniciar `apache2` para que as mudanças sejam aplicadas.
   - Digite um comando que use `watch`, `ps` e `grep` para as conexões do `apache2`.
   - Agora abra um navegador web ou use um navegador de linha de comando como o `lynx` para estabelecer uma conexão com o servidor web através de seu endereço IP. O que você observa na saída de `watch`?

3. Como vimos, por padrão, `top` classifica as tarefas por porcentagem de uso da CPU em ordem decrescente (com os valores mais altos no topo). Esse comportamento pode ser modificado com as teclas interativas `M` (uso de memória), `N` (identificador único do processo), `T` (tempo de execução) e `P` (porcentagem de tempo de CPU). No entanto, também podemos classificar a lista de tarefas a gosto, iniciando `top` com a opção `-o` (para saber mais, verifique a página `man` de top). Agora, execute as seguintes tarefas:
   - Inicie o `top` para que as tarefas sejam classificadas por uso da memória:
   - Verifique se digitou o comando correto destacando a coluna da memória:
4. O `ps` também tem uma opção `o` para especificar as colunas que você deseja mostrar. Estude esta opção e execute as seguintes tarefas:
   - Inicie o `ps` para exibir somente informações sobre o *usuário*, *porcentagem de memória usada*, *porcentagem de tempo da CPU usado* e *comando completo*:
   - Agora, inicie o `ps` para que as únicas informações exibida sejam o usuário e o nome dos programas que ele está usando:

Resumo

Nesta lição, você aprendeu sobre *jobs* (trabalhos) e *controle de jobs*. Os fatos e conceitos principais a ser lembrados são:

- Jobs são processos enviados ao segundo plano.
- Além de um *ID de processo*, os jobs também recebem um *ID de trabalho* quando criados.
- Para controlar os jobs, é necessária uma especificação de trabalho (`jobspec`).
- Os jobs podem ser trazidos ao primeiro plano, enviados ao segundo plano, encerrados e eliminados (ou *mortos*).
- Um job pode ser desvinculado do terminal e da sessão na qual foi criado.

Discutimos também o conceito de *processos* e *monitoramento de processos*. As ideias mais relevantes são:

- Os processos são programas em execução.
- Os processos podem ser monitorados.
- Diferentes utilitários nos permitem descobrir o *ID do processo* dos processos, bem como enviar sinais para encerrá-los.
- Os sinais podem ser especificados por nome (p. ex. `-SIGTERM`), número (p. ex. `-15`) ou opção (p. ex. `-s SIGTERM`).
- `top` e `ps` são muito poderosos quando se trata de monitorar processos. A saída do primeiro é dinâmica e se atualiza constantemente; já o `ps` exibe a saída de forma estática.

Comandos usados nesta lição:

`jobs`

Mostra os jobs ativos e seus status.

`sleep`

Espera por um período específico de tempo.

`fg`

Traz o job para o primeiro plano.

`bg`

Move o job para o segundo plano.

`kill`

Elimina o job.

`nohup`

Desvincula o job da sessão/terminal.

`exit`

Sai do shell atual.

`tail`

Exibe as linhas mais recentes em um arquivo.

`watch`

Executa um comando repetidamente (ciclo de 2 segundos por padrão).

`uptime`

Mostra há quanto tempo o sistema está rodando, o número de usuários atuais e a carga média do sistema.

`free`

Mostra o uso da memória.

`pgrep`

Procura o ID do processo com base no nome.

`pidof`

Procura o ID do processo com base no nome.

`pkill`

Envia sinal ao processo por nome.

`killall`

Elimina processo(s) por nome.

`top`

Exibe os processos do Linux.

`ps`

Relata um instantâneo dos processos atuais.

Respostas aos Exercícios Guiados

1. `oneko` é um programa divertido que mostra um gato perseguindo o cursor do mouse. Se ele ainda não estiver instalado em seu sistema desktop, instale-o usando o gerenciador de pacotes de sua distribuição. Vamos usá-lo para estudar o controle de jobs.

   - Inicie o programa. Qual o procedimento?

     Digitar `oneko` no terminal.

   - Mova o cursor do mouse para ver como o gato o persegue. Agora suspenda o processo. Qual o procedimento? Qual é a saída?

     Pressionar a combinação de teclas `Ctrl` + `z`:

     ```
     [1]+  Stopped                 oneko
     ```

   - Verifique quantos jobs estão presentes atualmente. O que você digita? Qual é a saída?

     ```
     $ jobs
     [1]+  Stopped                 oneko
     ```

   - Agora envie-o para o segundo plano especificando seu ID de trabalho. Qual é a saída? Como você pode saber se o job está sendo executado em segundo plano?

     ```
     $ bg %1
     [1]+ oneko &
     ```

     O gato está se movendo outra vez.

   - Finalmente, encerre o job especificando seu ID de trabalho. O que você digita?

     ```
     $ kill %1
     ```

2. Descubra o PID de todos os processos gerados pelo servidor web *Apache HTTPD* (`apache2`) com dois comandos diferentes:

   ```
   $ pgrep apache2
   ```

ou

```
$ pidof apache2
```

3. Encerre todos os processos `apache2` sem usar seus PIDs e com dois comandos diferentes:

```
$ pkill apache2
```

ou

```
$ killall apache2
```

4. Suponha que você tem de encerrar todas as instâncias do `apache2` e não tem tempo para descobrir quais são os PIDs correspondentes. Como fazer isso usando `kill` com o sinal padrão `SIGTERM` em uma só linha?

```
$ kill $(pgrep apache2)
$ kill `pgrep apache2`
```

ou

```
$ kill $(pidof apache2)
$ kill `pidof apache2`
```

NOTE Como `SIGTERM` (`15`) é o sinal padrão, não é necessário passar nenhuma opção para `kill`.

5. Inicie o `top` e interaja com ele executando o seguinte:
   - Exibir uma visão em floresta dos processos:

     Pressione `V`.

   - Exibir caminhos completos de processos, diferenciando entre espaço de usuário e espaço de kernel:

     Pressione `c`.

6. Digite o comando `ps` para exibir todos os processos iniciados pelo usuário do servidor web *Apache HTTPD* (`www-data`):
   - Usando a sintaxe do BSD:

```
$ ps U www-data
```

   - Usando a sintaxe do UNIX:

```
$ ps -u www-data
```

   - Usando a sintaxe do GNU:

```
$ ps --user www-data
```

Respostas aos Exercícios Exploratórios

1. O sinal `SIGHUP` pode ser usado como forma de reiniciar certos daemons. Com o servidor web *Apache HTTPD*—por exemplo—enviar `SIGHUP` para o processo pai (iniciado por `init`) elimina seus filhos. O pai, no entanto, relê seus arquivos de configuração, reabre os arquivos de log e gera um novo conjunto de filhos. Execute as seguintes tarefas:

   - Inicie o servidor web:

     ```
     $ sudo systemctl start apache2
     ```

   - Certifique-se de conhecer o PID do processo pai:

     ```
     $ ps aux | grep apache2
     ```

     O processo pai é aquele iniciado pelo usuário `root`. Em nosso caso, o PID é `1653`.

   - Faça o servidor web Apache HTTPD reiniciar enviando o sinal `SIGHUP` para o processo pai:

     ```
     $ kill -SIGHUP 1653
     ```

   - Verifique se o pai não foi eliminado e se novos filhos foram gerados:

     ```
     $ ps aux | grep apache2
     ```

     Agora você deve ver o processo pai `apache2` junto com dois novos filhos.

2. Embora inicialmente estática, a saída de `ps` pode ser *tornada* dinâmica combinando `ps` e `watch`. Vamos monitorar as novas conexões do servidor web *Apache HTTPD*. Antes de realizar as tarefas descritas abaixo, recomendamos ler a descrição da diretiva `MaxConnectionsPerChild` em Apache MPM Common Directives.

   - Adicione a diretiva `MaxConnectionsPerChild` com um valor de `1` no arquivo de configuração de `apache2`—no *Debian* e seus derivativos ele fica em `/etc/apache2/apache2.conf`; na família *CentOS*, em `/etc/httpd/conf/httpd.conf`. Não esqueça de reiniciar `apache2` para que as mudanças sejam aplicadas.

     A linha a incluir no arquivo de configuração é `MaxConnectionsPerChild 1`. Uma maneira de reiniciar o servidor web é através de `sudo systemctl restart apache2`.

- Digite um comando que use `watch`, `ps` e `grep` para as conexões do `apache2`.

```
$ watch 'ps aux | grep apache2'
```

ou

```
$ watch "ps aux | grep apache2"
```

- Agora abra um navegador web ou use um navegador de linha de comando como o `lynx` para estabelecer uma conexão com o servidor web através de seu endereço IP. O que você observa na saída de `watch`?

  Um dos processos filho de propriedade de `www-data` desaparece.

3. Como vimos, por padrão, `top` classifica as tarefas por porcentagem de uso da CPU em ordem decrescente (com os valores mais altos no topo). Esse comportamento pode ser modificado com as teclas interativas `M` (uso de memória), `N` (identificador único do processo), `T` (tempo de execução) e `P` (porcentagem de tempo de CPU). No entanto, também podemos classificar a lista de tarefas a gosto, iniciando `top` com a opção `-o` (para saber mais, verifique a página `man` de top). Agora, execute as seguintes tarefas:

   - Inicie o `top` para que as tarefas sejam classificadas por uso da memória:

```
$ top -o %MEM
```

   - Verifique se digitou o comando correto destacando a coluna da memória:

     Pressione `x`.

4. O `ps` também tem uma opção `o` para especificar as colunas que você deseja mostrar. Estude esta opção e execute as seguintes tarefas:

   - Inicie o `ps` para exibir somente informações sobre o *usuário*, *porcentagem de memória usada*, *porcentagem de tempo da CPU usado* e *comando completo*:

```
$ ps o user,%mem,%cpu,cmd
```

   - Agora, inicie o `ps` para que as únicas informações exibida sejam o usuário e o nome dos programas que ele está usando:

```
$ ps o user,comm
```

103.5 Lição 2

Certificação:	LPIC-1
Versão:	5.0
Tópico:	103 Comandos GNU e Unix
Objetivo:	103.5 Criar, monitorar e eliminar processos
Lição:	2 de 2

Introdução

As ferramentas e utilitários vistos na lição anterior são muito úteis para o monitoramento de processos em geral. No entanto, o administrador do sistema pode precisar ir além. Nesta lição, discutiremos o conceito de multiplexadores de terminal e aprenderemos sobre *GNU Screen* e *tmux*, já que—embora os emuladores de terminal modernos sejam excelentes—os multiplexadores ainda preservam alguns recursos interessantes e poderosos para a produtividade de um administrador de sistema.

Recursos dos multiplexadores de terminal

Em eletrônica, um multiplexador (ou *mux*) é um dispositivo que permite que várias entradas sejam conectadas a uma única saída. Assim, um multiplexador de terminal nos proporciona a capacidade de alternar entre diferentes entradas conforme necessário. Embora não sejam exatamente iguais, `screen` e `tmux` compartilham uma série de características comuns:

- Qualquer invocação bem-sucedida resultará em pelo menos uma sessão que—por sua vez—incluirá ao menos uma janela. As janelas contêm programas.
- As janelas pode ser divididas em regiões ou painéis—o que ajuda na produtividade ao se

trabalhar com vários programas simultaneamente.

- Facilidade de controle: para executar a maioria dos comandos, usamos uma combinação de teclas — o chamado *prefixo de comando* ou *chave de comando* — seguida por outro caractere.
- As sessões podem ser desanexadas do terminal em que estão (ou seja, os programas são enviados para o segundo plano e continuam em execução). Isso garante a execução completa dos programas, independentemente de fecharmos acidentalmente um terminal, travamentos ocasionais ou até mesmo a perda da conexão remota.
- Conexão de socket.
- Modo de cópia.
- São altamente personalizáveis.

GNU Screen

Nos primórdios do Unix (anos 1970-80), os computadores consistiam basicamente em terminais conectados a um computador central. Era só isso mesmo, sem um monte de janelas ou abas. E essa foi a razão por trás da criação do GNU Screen em 1987: emular múltiplas telas *VT100* independentes em um único terminal físico.

Janelas

O GNU Screen é invocado simplesmente digitando `screen` no terminal. Aparece primeiro uma mensagem de boas-vindas:

```
GNU Screen version 4.05.00 (GNU) 10-Dec-16

Copyright (c) 2010 Juergen Weigert, Sadrul Habib Chowdhury
Copyright (c) 2008, 2009 Juergen Weigert, Michael Schroeder, Micah Cowan, Sadrul
Habib Chowdhury
Copyright (c) 1993-2002, 2003, 2005, 2006, 2007 Juergen Weigert, Michael Schroeder
Copyright (c) 1987 Oliver Laumann
(...)
```

Pressione Espaço ou Enter para fechar a mensagem e será mostrado um prompt de comando:

```
$
```

Pode parecer que nada aconteceu, mas o fato é que o comando `screen` já criou e gerencia sua primeira sessão e janela. O prefixo do comando screen é `Ctrl` + `a`. Para ver todas as janelas na parte

inferior da tela do terminal, digite `Ctrl` + `a` - `w`:

```
0*$ bash
```

Aqui está, nossa única janela até agora! Observe, no entanto, que a contagem começa em 0. Para criar outra janela, digite `Ctrl` + `a` - `c`. Você verá um novo prompt. Vamos iniciar o `ps` nessa nova janela:

```
$ ps
  PID TTY          TIME CMD
  974 pts/2    00:00:00 bash
  981 pts/2    00:00:00 ps
```

e digitar `Ctrl` + `a` - `w` novamente:

```
0-$ bash  1*$ bash
```

Agora temos nossas duas janelas (observe o asterisco que indica a que está sendo exibida no momento). No entanto, como foram iniciadas com o Bash, as duas receberam o mesmo nome. Já que chamamos o `ps` em nossa janela atual, vamos renomeá-lo com esse mesmo nome. Para isso, digite `Ctrl` + `a` - `A` e escreva o novo nome da janela (`ps`) quando solicitado:

```
Set window's title to: ps
```

Agora, vamos criar outra janela, já com um nome desde o início: `yetanotherwindow`. Para isso, invocamos `screen` com a opção `-t`:

```
$ screen -t yetanotherwindow
```

Há várias maneiras de se mover entre as janelas:

- Usando `Ctrl` + `a` - `n` (ir para a próxima janela, *next* em inglês) e `Ctrl` + `a` - `p` (ir para a janela anterior, *previous*).
- Usando `Ctrl` + `a` - *`número`* (ir para a janela número *número*).
- Usando `Ctrl` + `a` - `"` para ver uma lista de todas as janelas. Use as setinhas do teclado para selecionar a janela que deseja e dê Enter:

```
Num Name                                                    Flags

   0 bash                                                      $
   1 ps                                                        $
   2 yetanotherwindow
```

Ao trabalhar com janelas, é importante lembrar o seguinte:

- As janelas executam seus programas de forma totalmente independente umas das outras.
- Os programas continuarão a ser executados mesmo que suas janelas não estejam visíveis (inclusive quando a sessão de screen for desanexada, como veremos em breve).

Para remover uma janela, basta encerrar o programa que está em execução nela (quando a última janela for removida, o próprio `screen` será encerrado). Outra alternativa é usar `Ctrl` + `a` - `k` enquanto estiver na janela que deseja remover; será solicitada a confirmação:

```
Really kill this window [y/n]

Window 0 (bash) killed.
```

Regiões

O `screen` pode dividir uma tela de terminal em diversas regiões para acomodar janelas. Essas divisões podem ser horizontais (`Ctrl` + `a` - `S`) ou verticais (`Ctrl` + `a` - `|`).

A única coisa que a nova região mostrará é `--` na parte inferior, o que significa que está vazia:

```
   1 ps                                                        --
```

Para passar à nova região, digite `Ctrl` + `a` - `Tab`. Agora podemos adicionar uma janela com qualquer um dos métodos que já vimos, por exemplo: `Ctrl` + `a` - `2`. Assim, o `--` deve se transformar em `2 yetanotherwindow`:

```
$ ps                                                          $
  PID TTY          TIME CMD
 1020 pts/2    00:00:00 bash
 1033 pts/2    00:00:00 ps
$ screen -t yetanotherwindow
```

```
   1 ps                                                                2
yetanotherwindow
```

Alguns aspectos importantes a lembrar ao se trabalhar com regiões:

- Para se mover entre regiões, digite `Ctrl` + `a` - `Tab`.
- Para encerrar todas as regiões exceto a atual, use `Ctrl` + `a` - `Q`.
- Para encerrar a região atual, use `Ctrl` + `a` - `X`.
- O encerramento de uma região não encerra a janela associada a ela.

Sessões

Até agora brincamos com algumas janelas e regiões, mas todas pertencentes à mesma e única sessão. É hora de começar a brincar com as sessões. Para ver uma lista de todas as sessões, digite `screen -list` ou `screen -ls`:

```
$ screen -list
There is a screen on:
        1037.pts-0.debian       (08/24/19 13:53:35)     (Attached)
1 Socket in /run/screen/S-carol.
```

Essa é a nossa única sessão no momento:

PID

`1037`

Nome

`pts-0.debian` (indicando o terminal — em nosso caso, um *pseudo-terminal escravo* — e o nome do host).

Status

`Attached`

Vamos criar uma nova sessão, dando a ela um nome mais descritivo:

```
$ screen -S "second session"
```

A tela do terminal será esvaziada e um novo prompt se abrirá. Você pode verificar as sessões mais uma vez:

```
$ screen -ls
There are screens on:
        1090.second session     (08/24/19 14:38:35)     (Attached)
        1037.pts-0.debian       (08/24/19 13:53:36)     (Attached)
2 Sockets in /run/screen/S-carol.
```

Para encerrar uma sessão, saia de todas as janelas ou simplesmente digite o comando `screen -S SESSION-PID -X quit` (também é possível fornecer o nome da sessão). Vamos nos livrar de nossa primeira sessão:

```
$ screen -S 1037 -X quit
```

Você será enviado de volta ao prompt do terminal fora de `screen`. Mas lembre-se, nossa segunda sessão ainda está viva:

```
$ screen -ls
There is a screen on:
    1090.second session (08/24/19 14:38:35) (Detached)
1 Socket in /run/screen/S-carol.
```

No entanto, como eliminamos a sessão pai, ela recebe um novo rótulo: `Detached` (desanexada).

Desanexando sessões

Por uma série de razões, podemos querer desanexar uma sessão do screen do terminal a que pertence:

- Para deixar o computador da empresa cumprir seu dever e conectar-se remotamente mais tarde, de casa.
- Para compartilhar uma sessão com outros usuários.

Para desanexar uma sessão, a combinação de teclas é `Ctrl` + `a` - `d`. Você é levado de volta ao seu terminal:

```
[detached from 1090.second session]
$
```

Para entrar novamente na sessão desanexada, use o comando `screen -r SESSION-PID`. Outra alternativa é `SESSION-NAME`, como vimos acima. Se houver apenas uma sessão desanexada, nenhuma dessas opções é obrigatória:

```
$ screen -r
```

Este comando basta para anexar novamente a segunda sessão:

```
$ screen -ls
There is a screen on:
        1090.second session      (08/24/19 14:38:35)     (Attached)
1 Socket in /run/screen/S-carol.
```

Opções importantes para reanexar sessões:

`-d -r`

Reanexa uma sessão e — se necessário — a desanexa primeiro.

`-d -R`

Igual a `-d -r`, mas o `screen` cria primeiro uma sessão caso ela não exista.

`-d -RR`

Igual a `-d -R`. No entanto, usa a primeira sessão se houver mais de uma disponível.

`-D -r`

Reanexa uma sessão. Se necessário, o comando a desanexa e faz logout remotamente primeiro.

`-D -R`

Se uma sessão estiver rodando, ela é reanexada (desanexando e desconectando remotamente primeiro, caso necessário). Se não for o caso, ela é criada e o usuário, notificado.

`-D -RR`

Igual a `-D -R` — só que mais poderoso.

`-d -m`

Inicia o `screen` em *modo detached* (desanexado). Uma nova sessão é criada, mas desanexada. Útil para scripts de inicialização do sistema.

`-D -m`

Igual a `-d -m`, mas não bifurca um novo processo. O comando é encerrado se a sessão terminar.

Leia as páginas de manual de `screen` para descobrir outras opções.

Copiar e colar: Modo de rolagem

O GNU Screen apresenta um modo de cópia ou *modo de rolagem*. Dentro dele, você pode mover o cursor na janela atual e através do conteúdo de seu histórico usando as setas do teclado. Você pode marcar o texto e copiá-lo nas janelas. As etapas a seguir são:

1. Entre no modo de cópia/rolagem: `Ctrl` + `a` - `[`.
2. Vá para o início da parte do texto que deseja copiar usando as setas do teclado.
3. Marque o início do trecho de texto que deseja copiar: Espaço.
4. Vá para o final da parte do texto que deseja copiar usando as setas do teclado.
5. Marque o final do trecho de texto que deseja copiar: Espaço.
6. Vá para a janela de sua escolha e cole o texto: `Ctrl` + `a` - `]`.

Personalização de screen

O arquivo de configuração de todo o sistema para screen é `/etc/screenrc`. Também é possível usar `~/.screenrc` no nível do usuário. O arquivo inclui quatro seções de configuração principais:

`SCREEN SETTINGS`

Você pode definir configurações gerais especificando a *diretiva* seguida por um espaço e o *valor*, como em: `defscrollback 1024`.

`SCREEN KEYBINDINGS`

Esta seção é bastante interessante, pois permite redefinir os atalhos de teclado que podem interferir no seu uso diário do terminal. Use a palavra-chave `bind` seguida por um espaço, o caractere a ser usado após o prefixo do comando, outro espaço e o comando, como em: `bind l kill` (esta configuração mudará a forma padrão de eliminar uma janela para `Ctrl` + `a` - `l`).

Para exibir todos os atalhos de screen, digite `Ctrl` + `a` - `?` ou consulte a página de manual.

TIP Claro, você também pode alterar o próprio prefixo do comando. Por exemplo, para ir de `Ctrl` + `a` a `Ctrl` + `b`, basta adicionar esta linha: `escape ^Bb`.

`TERMINAL SETTINGS`

Esta seção inclui as configurações relacionadas aos tamanhos de janela de terminal e buffers — entre outros. Para habilitar o modo sem bloqueio para lidar melhor com conexões SSH instáveis, por exemplo, a seguinte configuração é usada: `defnonblock 5`.

`STARTUP SCREENS`

Podemos incluir comandos para que diversos programas sejam executados após a inicialização de `screen`; por exemplo `screen -t top top` (o screen abre uma janela chamada `top` com `top` dentro dela).

tmux

O `tmux` foi lançado em 2007. Embora muito semelhante ao `screen`, ele apresenta algumas diferenças notáveis:

- Modelo cliente-servidor: o servidor fornece uma série de sessões, cada uma das quais pode ter várias janelas anexadas a ele que podem, por sua vez, ser compartilhadas por vários clientes.
- Seleção interativa de sessões, janelas e clientes via menus.
- A mesma janela pode ser anexada a várias sessões.

Disponibilidade de atalhos do *vim* e do *Emacs*.

- Suporte para terminal UTF-8 e 256 cores.

Janelas

O `tmux` pode ser invocado simplesmente digitando `tmux` no prompt de comando. Você verá um prompt do shell e uma barra de status na parte inferior da janela:

```
[0] 0:bash*                                                    "debian" 18:53
27-Aug-19
```

Além do `hostname`, a hora e a data, a barra de status fornece as seguintes informações:

Nome da sessão

`[0]`

Número da janela

`0:`

Nome da janela

`bash*`. Por padrão, esse é o nome do programa em execução dentro da janela e — ao contrário de `screen` — o `tmux` irá atualizá-lo automaticamente para refletir o programa em execução atual. Observe o asterisco indicando que esta é a janela atual em exibição.

Você pode atribuir nomes de sessão e janela ao invocar `tmux`:

```
$ tmux new -s "LPI" -n "Window zero"
```

A barra de status será alterada de acordo:

```
[LPI] 0:Window zero*                                          "debian" 19:01
27-Aug-19
```

O prefixo do comando do tmux é `Ctrl` + `b`. Para criar uma nova janela, basta digitar `Ctrl` + `b` - `c`; você será levado a um novo prompt e a barra de status refletirá a nova janela:

```
[LPI] 0:Window zero- 1:bash*                                  "debian" 19:02
27-Aug-19
```

Como o Bash é o shell subjacente, a nova janela recebe esse nome por padrão. Inicie `top` e veja como o nome muda para `top`:

```
[LPI] 0:Window zero- 1:top*                                   "debian" 19:03
27-Aug-19
```

Em qualquer caso, você pode renomear uma janela com `Ctrl` + `b` - `,`. Quando solicitado, informe o novo nome e pressione Enter:

```
(rename-window) Window one
```

Para exibir todas as janelas, pressione `Ctrl` + `b` - `w` (use as setas do teclado para se mover para cima e para baixo e `enter` para selecionar):

```
(0)  0: Window zero- "debian"
(1)  1: Window one* "debian"
```

Como em `screen`, podemos pular de uma janela para outra com:

Ctrl + **b** - **n**
: ir para a próxima janela, *next* em inglês.

Ctrl + **b** - **p**
: ir para a janela anterior, *previous*.

Ctrl + **b** - `number`
: ir para a janela de número *número*.

Para eliminar uma janela, use `Ctrl` + `b` - `&`. Será solicitada a confirmação:

```
kill-window Window one? (y/n)
```

Outros comandos de janela interessantes são:

Ctrl + **b** - **f**
: encontrar uma janela pelo nome.

Ctrl + **b** - **.**
: alterar o número do índice da janela.

Para conhecer a lista completa de comandos, consulte a página do manual.

Painéis

A facilidade de divisão de janelas do `screen` também está presente no `tmux`. As divisões resultantes não são chamadas de *regiões*, mas de *painéis*. A diferença mais importante entre regiões e painéis é que os últimos são pseudoterminais completos anexados a uma janela. Isso significa que eliminar um painel também eliminará seu pseudo-terminal e todos os programas associados em execução nele.

Para dividir uma janela horizontalmente, usamos `Ctrl` + `b` - `"`:

```
Tasks:  93 total,   1 running,  92 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
KiB Mem :  4050960 total,  3730920 free,   114880 used,   205160 buff/cache
KiB Swap:  4192252 total,  4192252 free,        0 used.  3716004 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 1340 carol     20   0   44876   3400   2800 R  0.3  0.1   0:00.24 top
    1 root      20   0  139088   6988   5264 S  0.0  0.2   0:00.50 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.04 ksoftirqd/0
    4 root      20   0       0      0      0 S  0.0  0.0   0:01.62 kworker/0:0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S  0.0  0.0   0:00.06 rcu_sched
    8 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0
   10 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 lru-add-drain
   11 root      rt   0       0      0      0 S  0.0  0.0   0:00.01 watchdog/0
   12 root      20   0       0      0      0 S  0.0  0.0   0:00.00 cpuhp/0
$
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□
$














[LPI] 0:Window zero- 1:Window one*                                    "debian"
19:05 27-Aug-19
```

Para dividi-la verticalmente, usamos `Ctrl` + `b` - `%`:

```
                                                                              □ $
    1 root      20   0  139088   6988   5264 S  0.0  0.2   0:00.50 systemd    □
                                                                              □
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd   □
                                                                              □
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.04 ksoftirqd/0 □
    4 root      20   0       0      0      0 S  0.0  0.0   0:01.62 kworker/0:0 □
```

```
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S  0.0  0.0   0:00.06 rcu_sched
    8 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0

   10 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 lru-add-drai
n
   11 root      rt   0       0      0      0 S  0.0  0.0   0:00.01 watchdog/0

   12 root      20   0       0      0      0 S  0.0  0.0   0:00.00 cpuhp/0

$
$
[LPI] 0:Window zero-  1:Window one*                                  "debian"
19:05 27-Aug-19
```

Para destruir o painel atual (junto com o pseudoterminal rodando dentro dele e quaisquer programas associados), use `Ctrl` + `b` - `x`. Será solicitada a confirmação na barra de status:

```
kill-pane 1? (y/n)
```

Comandos importantes dos painéis:

`Ctrl` + **`b`** - **`↑`**,**`↓`**,**`←`**,**`→`**
: mover-se entre painéis.

`Ctrl` + `b` - `;`
: passar para o último painel ativo.

`Ctrl` + `b` - `Ctrl` + `seta`
: redimensionar o painel em uma linha.

`Ctrl` + `b` - `Alt` + `seta`
: redimensionar o painel em cinco linhas.

`Ctrl` + `b` - `{`
: trocar de painel (do atual para o anterior).

`Ctrl` + `b` - `}`
: trocar de painel (do atual para o seguinte).

`Ctrl` + `b` - `z`
: aproximar/afastar o painel.

`Ctrl` + `b` - `t`
: O `tmux` exibe um relógio elegante dentro do painel (para removê-lo, pressione `q`).

`Ctrl` + `b` - `!`
: transforma o painel em janela.

Para conhecer a lista completa de comandos, consulte a página do manual.

Sessões

Para listar as sessões no `tmux`, você pode usar `Ctrl` + `b` - `s`:

```
(0) + LPI: 2 windows (attached)
```

Outra alternativa é usar o comando `tmux ls`:

```
$ tmux ls
LPI: 2 windows (created Tue Aug 27 19:01:49 2019) [158x39] (attached)
```

Existe apenas uma sessão (`LPI`) que inclui duas janelas. Vamos criar uma nova sessão de dentro de

nossa sessão atual. Isso pode ser feito usando `Ctrl` + `b`. Digite `:new` no prompt e pressione Enter. Você será enviado para a nova sessão, conforme pode ser observado na barra de status:

```
[2] 0:bash*                                                    "debian" 19:15
27-Aug-19
```

Por padrão, o `tmux` denomina a sessão `2`. Para renomeá-la, use `Ctrl` + `b` - `$`. Quando solicitado, informe o novo nome e pressione Enter:

```
(rename-session) Second Session
```

Para trocar de sessão, o atalho é `Ctrl` + `b` - `s` (use as setas do teclado e `enter`):

```
(0) + LPI: 2 windows
(1) + Second Session: 1 windows (attached)
```

Para eliminar uma sessão, você pode usar o comando `tmux kill-session -t SESSION-NAME`. Se digitar o comando de dentro da sessão atual anexada, você será retirado do `tmux` e levado de volta à sua sessão de terminal inicial:

```
$ tmux kill-session -t "Second Session"
[exited]
$
```

Desanexando sessões

Ao eliminar `Second Session`, fomos levados para fora do `tmux`. No entanto, ainda temos uma sessão ativa. Peça ao `tmux` uma lista de sessões e você certamente a encontrará ali:

```
$ tmux ls
LPI: 2 windows (created Tue Aug 27 19:01:49 2019) [158x39]
```

No entanto, esta sessão está desanexada de seu terminal. Podemos anexá-la com `tmux attach -t SESSION-NAME` (`attach` pode ser substituído por `at` ou — simplesmente — `a`). Quando há apenas uma sessão, a especificação do nome é opcional:

```
$ tmux a
```

Agora você está de volta à sua sessão; para desanexá-la, pressione `Ctrl` + `b` - `d`:

```
[detached (from session LPI)]
$
```

TIP A mesma sessão pode ser anexada a mais de um terminal. Se quiser anexar uma sessão e ter certeza de que ela foi primeiramente desanexada de quaisquer outros terminais, use a opção `-d`: `tmux attach -d -t SESSION-NAME`.

Comandos importantes para anexar/desanexar sessão:

`Ctrl` + `b` - `D`
: seleciona o cliente a desanexar.

`Ctrl` + `b` - `r`
: atualiza o terminal do cliente.

Para conhecer a lista completa de comandos, consulte a página do manual.

Copiar e colar: Modo de rolagem

O `tmux` também possui um modo de cópia, basicamente igual ao do `screen` (lembre-se de usar o prefixo de comando do tmux e não o de screen!). A única diferença em termos de comando é que usamos `Ctrl` + Espaço para marcar o início da seleção e `Alt` + `w` para copiar o texto selecionado.

Personalização do tmux

Os arquivos de configuração do `tmux` tipicamente se localizam em `/etc/tmux.conf` and `~/.tmux.conf`. Quando iniciado, o `tmux` procura por esses arquivos, se eles existirem. Também existe a possibilidade de iniciar o `tmux` com a opção `-f` para fornecer um arquivo de configuração alternativo. Um exemplo de arquivo de configuração do `tmux` pode ser encontrado em `/usr/share/doc/tmux/example_tmux.conf`. O nível de personalização é altíssimo. Eis algumas das coisas que é possível fazer:

- Alterar a tecla de prefixo

```
# Change the prefix key to C-a
```

```
set -g prefix C-a
unbind C-b
bind C-a send-prefix
```

- Definir atalhos de teclado extras para janelas superiores a 9

```
# Some extra key bindings to select higher numbered windows
bind F1 selectw -t:10
bind F2 selectw -t:11
bind F3 selectw -t:12
```

Para ver uma lista abrangente com todos os atalhos, digite `Ctrl` + `b` - `?` (pressione `q` para sair) ou consulte a página de manual.

Exercícios Guiados

1. Indique se as seguintes afirmações/recursos correspondem ao GNU Screen, ao tmux ou a ambos:

Recurso/Afirmação	GNU Screen	tmux
O prefixo dos comandos por padrão é `Ctrl` + `a`		
Modelo Cliente-Servidor		
Os painéis são pseudo-terminais		
Eliminar uma região não elimina as janelas associadas		
As sessões incluem janelas		
As sessões podem ser desanexadas		

Mude o nome da janela atual para `vi`: . Instale o `tmux` em seu computador (nome do pacote: `tmux`) e realize as tarefas a seguir:

+ * Inicie o programa. Que comando você usa?

+

+ * Inicie `top` (note como — em poucos segundos — o nome da janela na barra de status muda para `top`):

+

+ * Usando o prefixo de teclado do tmux, abra uma nova janela; em seguida, crie `~/.tmux.conf` usando `nano`:

+

+ * Divida a janela verticalmente e reduza o tamanho do novo painel algumas vezes:

+

+ * Agora mude o nome da janela atual para `text editing`; em seguida, faça o `tmux` exibir uma lista de todas as sessões:

+

+ * Passe para a janela que está rodando `top` e volte à janela atual usando a mesma combinação de teclas:

+

+ * Desanexe a sessão atual e crie uma nova chamada `ssh` com uma janela chamada `ssh window`:

+

+ * Desanexe também a sessão `ssh` e faça o `tmux` exibir outra vez a lista de sessões:

+

+

NOTE

A partir deste ponto, o exercício requer o uso de uma máquina *remota* para conexões `ssh` ao seu host local (uma máquina virtual é perfeitamente válida e pode ser muito prática). Certifique-se de ter o `openssh-server` instalado e rodando em sua máquina local e que pelo menos o `openssh-client` esteja instalado na máquina remota.

+ * Agora, inicie uma máquina remota e conecte-se via `ssh` de seu host local. Quando a conexão for estabelecida, procure por sessões do `tmux`:

+

+ * No host remoto, anexe a sessão do `ssh` pelo nome:

+

+ * De volta à máquina local, anexe a sessão do `ssh` pelo nome, garantindo que a conexão ao hospedeiro remoto seja encerrada antes:

+

+ * Exiba todas as sessões para seleção e vá à primeira sessão (`[0]`). Uma vez ali, elimine a sessão `ssh` pelo nome:

+

+ * Finalmente, desanexe a sessão atual e elimine-a pelo nome:

+

Exercícios Exploratórios

1. Tanto `screen` quanto `tmux` podem entrar no modo de linha de comando através do atalho *prefixo do comando* + `:` (já vimos um breve exemplo com `tmux`). Faça uma pesquisa e realize as seguintes tarefas em modo de linha de comando:
   - Faça o `screen` entrar em modo de cópia:
   - Faça o `tmux` renomear a janela atual:
   - Faça o `screen` fechar todas as janelas e encerrar a sessão:
   - Faça o `tmux` dividir um painel em dois:
   - Faça o `tmux` eliminar a janela atual:
2. Quando entramos no modo de cópia no `screen`, não somente podemos usar as setas do teclado e `PgUP` or `PgDown` para navegar na janela atual e no buffer de rolagem; também há a possibilidade de usar um editor de tela completo como o vi. Usando esse editor, realize as seguintes tarefas:
   - Ecoe (echo) `supercalifragilisticexpialidocious` em seu terminal de `screen`:
   - Agora, copie os cinco caracteres consecutivos (da esquerda para a direita) na linha acima de seu cursor:
   - Finalmente, cole a seleção (`stice`) em seu prompt de comando:
3. Suponha que você quer compartilhar uma sessão do `tmux` (`our_session`) com outro usuário. Você criou o socket (`/tmp/our_socket`) com as permissões corretas, para que você e o outro usuário

possam ler e escrever. Quais são as outras duas condições para que o segundo usuário possa anexar com sucesso a sessão usando `tmux -S /tmp/our_socket a -t our_session`?

Resumo

Nesta lição, você aprendeu sobre *multiplexadores de terminal* em geral e o GNU Screen e o tmux em particular. Estes são os conceitos importantes a lembrar:

- Prefixo do comando: o `screen` usa `Ctrl` + `a` + *caractere*; o `tmux`, `Ctrl` + `b` + *caractere*.
- Estrutura de sessões, janelas e divisões de janelas (regiões ou painéis).
- Modo de cópia.
- Desanexar sessões: um dos recursos mais poderosos dos multiplexadores.

Comandos usados nesta lição:

`screen`
: Inicia uma sessão do screen.

`tmux`
: Inicia uma sessão do `tmux`.

Respostas aos Exercícios Guiados

1. Indique se as seguintes afirmações/recursos correspondem ao GNU Screen, ao tmux ou a ambos:

Recurso/Afirmação	GNU Screen	tmux
O prefixo dos comandos por padrão é `Ctrl` + `a`	x	
Modelo Cliente-Servidor		x
Os painéis são pseudo-terminais		x
Eliminar uma região não elimina as janelas associadas	x	
As sessões incluem janelas	x	x
As sessões podem ser desanexadas	x	x

2. Instale o GNU Screen no seu computador (nome do pacote: `screen`) e realize as seguintes tarefas:
   - Inicie o programa. Que comando você usa?

     `screen`

   - Inicie `top`:

     `top`

   - Usando o prefixo de teclado de screen, abra uma nova janela; em seguida, abra `/etc/screenrc` usando `vi`:

     `Ctrl` + `a` - `c`

     `sudo vi /etc/screenrc`

   - Liste as janelas na parte de baixo da tela:

     `Ctrl` + `a` - `w`

   - Mude o nome da janela atual para `vi`:

     `Ctrl` + `a` - `A`. Depois digitamos `vi` e pressionamos `enter`.

- Mude o nome da janela restante para `top`. Para isso, primeiro exiba uma lista de todas as janelas para poder navegar entre elas e selecionar a correta:

  Primeiro, digitamos `Ctrl` + `a` - `"`. Em seguida usamos as setas do teclado para marcar a janela que diz `0 bash` e pressionamos `enter`. Finalmente, digitamos `Ctrl` + `a` - `A`, em seguida `top` e pressionamos `enter`.

- Confira se os nomes foram alterados exibindo o nome das janelas novamente na parte de baixo da tela:

  `Ctrl` + `a` - `w`

- Agora, desanexe a sessão e faça o `screen` criar uma nova de nome `ssh`:

  `Ctrl` + `a` - `d` `screen -S "ssh"` e pressione `enter`.

- Desanexe `ssh` e faça o `screen` exibir a lista de sessões:

  `Ctrl` + `a` - `d` `screen -list` ou `screen -ls`.

- Agora, anexe-a à primeira sessão usando seu PID:

  `screen -r PID-OF-SESSION`

- Você deve ter retornado à janela que está exibindo `top`. Divida a janela horizontalmente e passe para a nova região vazia:

  `Ctrl` + `a` - `S`

  `Ctrl` + `a` - `Tab`

- Faça o `screen` listar todas as janelas e selecione `vi` para ser exibido na nova região vazia:

  Usamos `Ctrl` + `a` - `"` para exibir todas as janelas para seleção, marcamos `vi` e pressionamos `enter`.

- Agora, divida a região atual verticalmente, passe para a nova região vazia e associe-a a uma nova janela:

  `Ctrl` + `a` - `|`

  `Ctrl` + `a` - `Tab`

`Ctrl` + `a` - `c`

- Elimine todas as regiões exceto a atual (lembre-se de que, embora você esteja eliminando as regiões, as janelas ainda estão vivas). Depois, saia de todas as janelas da sessão atual até que a sessão em si seja encerrada:

  `Ctrl` + `a` - `Q`. `exit` (para sair do Bash). `Shift` + `:`, depois digitamos `quit` e pressionamos `enter` (para sair do `vi`). Em seguida, digitamos `exit` (para sair do shell Bash subjacente), `q` (para encerrar `top`); depois digitamos `exit` (para sair do shell Bash subjacente).

- Finalmente, faça o `screen` listar novamente suas sessões, elimine a sessão `ssh` restante pelo PID e confira se não sobrou nenhuma sessão:

  `screen -list` ou `screen -ls`

  `screen -S PID-OF-SESSION -X quit`

  `screen -list` ou `screen -ls`

3. Instale o `tmux` em seu computador (nome do pacote: `tmux`) e realize as tarefas a seguir:

   - Inicie o programa. Que comando você usa?

     `tmux`

   - Inicie `top` (note como — em poucos segundos — o nome da janela na barra de status muda para `top`):

     `top`

   - Usando o prefixo de teclado do tmux, abra uma nova janela; em seguida, crie `~/.tmux.conf` usando `nano`:

     `Ctrl` + `b` - `c` `nano ~/.tmux.conf`

   - Divida a janela verticalmente e reduza o tamanho do novo painel algumas vezes:

     `Ctrl` + `b` - `"`

     `Ctrl` + `b` - `Ctrl` + `↓`

   - Agora mude o nome da janela atual para `text editing`; em seguida, faça o `tmux` exibir uma lista de todas as sessões:

`Ctrl` + `b` - `,`. Daí informamos o novo nome e pressionamos `enter`. `Ctrl` + `b` - `s` ou `tmux ls`.

- Passe para a janela que está rodando `top` e volte à janela atual usando a mesma combinação de teclas:

  `Ctrl` + `b` - `n` or `Ctrl` + `b` - `p`

- Desanexe a sessão atual e crie uma nova chamada `ssh` com uma janela chamada `ssh window`:

  `Ctrl` + `b` - `d` `tmux new -s "ssh" -n "ssh window"`

- Desanexe também a sessão `ssh` e faça o `tmux` exibir outra vez a lista de sessões:

  `Ctrl` + `b` - `d` `tmux ls`

> **NOTE**
>
> A partir deste ponto, o exercício requer o uso de uma máquina *remota* para conexões `ssh` ao seu host local (uma máquina virtual é perfeitamente válida e pode ser muito prática). Certifique-se de ter o `openssh-server` instalado e rodando em sua máquina local e que pelo menos o `openssh-client` esteja instalado na máquina remota.

- Agora, inicie uma máquina remota e conecte-se via `ssh` de seu host local. Quando a conexão for estabelecida, procure por sessões do `tmux`:

  No hospedeiro remoto: `ssh` *`local-username@local-ipaddress`*. Depois de conectá-lo à máquina local: `tmux ls`.

- No host remoto, anexe a sessão do `ssh` pelo nome:

  `tmux a -t ssh` (`a` pode ser substituído por `at` ou `attach`).

- De volta à máquina local, anexe a sessão do `ssh` pelo nome, garantindo que a conexão ao hospedeiro remoto seja encerrada antes:

  `tmux a -d -t ssh` (`a` pode ser substituído por `at` ou `attach`).

- Exiba todas as sessões para seleção e vá à primeira sessão (`[0]`). Uma vez ali, elimine a sessão `ssh` pelo nome:

  Digitamos `Ctrl` + `b` - `s`, usamos as setas do teclado para marcar a sessão `0` e damos `enter` `tmux kill-session -t ssh`.

- Finalmente, desanexe a sessão atual e elimine-a pelo nome:

  `Ctrl` + `b` - `d` `tmux kill-session -t 0`.

Respostas aos Exercícios Exploratórios

1. Tanto `screen` quanto `tmux` podem entrar no modo de linha de comando através do atalho *prefixo do comando* + `:` (já vimos um breve exemplo com `tmux`). Faça uma pesquisa e realize as seguintes tarefas em modo de linha de comando:

   - Faça o `screen` entrar em modo de cópia:

     `Ctrl` + `a`-`:` — depois, digitamos `copy`.

   - Faça o `tmux` renomear a janela atual:

     `Ctrl` + `b`-`:` — depois, digitamos `rename-window`.

   - Faça o `screen` fechar todas as janelas e encerrar a sessão:

     `Ctrl` + `a`-`:` — depois, digitamos `quit`.

   - Faça o `tmux` dividir um painel em dois:

     `Ctrl` + `b`-`:` — depois, digitamos `split-window`.

   - Faça o `tmux` eliminar a janela atual:

     `Ctrl` + `b`-`:` — depois, digitamos `kill-window`.

2. Quando entramos no modo de cópia no `screen`, não somente podemos usar as setas do teclado e `PgUP` or `PgDown` para navegar na janela atual e no buffer de rolagem; também há a possibilidade de usar um editor de tela completo como o vi. Usando esse editor, realize as seguintes tarefas:

   - Ecoe (echo) `supercalifragilisticexpialidocious` em seu terminal de `screen`:

     ```
     echo supercalifragilisticexpialidocious
     ```

   - Agora, copie os cinco caracteres consecutivos (da esquerda para a direita) na linha acima de seu cursor:

     Para entrar no modo de cópia: `Ctrl` + `a`-`[` ou `Ctrl` + `a`-`:` e digitamos `copy`. Daí, passamos para a linha de cima usando `k` e pressionamos `espaço` para marcar o início da seleção. Finalmente, avançamos quatro caracteres usando `l` e pressionamos `espaço` novamente para marcar o final da seleção.

   - Finalmente, cole a seleção (`stice`) em seu prompt de comando:

`Ctrl` + `a` - `]`

3. Suponha que você quer compartilhar uma sessão do `tmux` (`our_session`) com outro usuário. Você criou o socket (`/tmp/our_socket`) com as permissões corretas, para que você e o outro usuário possam ler e escrever. Quais são as outras duas condições para que o segundo usuário possa anexar com sucesso a sessão usando `tmux -S /tmp/our_socket a -t our_session`?

   Ambos os usuários devem ter um grupo em comum, p. ex. `multiplexer`. Em seguida o socket também deve ser passado para esse grupo: `chgrp multiplexer /tmp/our_socket`.