

GUÍA DE USUARIO

COMANDO 'SCREEN'

LITRP-CLUSTER
ADMINISTRACIÓN DE SISTEMAS
Laboratorio de Investigaciones Tecnológicas en Reconocimiento de Patrones

Lista de correo: litrp-cluster@litrp.cl
Información de la lista: <http://litrp.cl/mailman/listinfo/litrp-cluster> litrp.cl
Sitio web: www.litrp.cl

Versión: 1.0
Septiembre, 2020

¿Qué es el comando 'screen'?

Con el comando 'screen' de Linux puede enviar aplicaciones de terminal en ejecución a un segundo plano y regresarlas a primer plano cuando desee verlas. También es compatible con pantallas divididas y funciona a través de conexiones SSH, incluso después de que se desconecta y se vuelve a conectar. Mediante su uso, podrás sustituir el uso del comando 'nohup' y dar seguimiento a la ejecución y salida por pantalla de sus procesos.

El comando 'screen' es un multiplexor de terminal y está absolutamente repleto de opciones. La página de manual tiene más de 4.100 líneas.

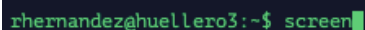
Los siguientes casos son los más comunes en los que usaría el comando 'screen', y son tratados en esta guía:

- La operación estándar es crear una nueva ventana con un *shell* en ella, ejecutar un comando y luego llevar la ventana al fondo (llamado "separar"). Cuando desee ver cómo está funcionando su proceso, puede volver a colocar la ventana en primer plano ("volver a adjuntar") y usarla nuevamente. Esto es ideal para procesos largos que no desea terminar accidentalmente al cerrar la ventana de terminal.
- Una vez que tenga una sesión 'screen' en ejecución, puede crear nuevas ventanas y ejecutar otros procesos en ellas. Puede saltar fácilmente entre ventanas para monitorear su progreso. También puede dividir la ventana de su terminal en regiones verticales u horizontales y mostrar las diversas ventanas en una ventana.
- Puede conectarse a una máquina remota, iniciar una sesión 'screen' e iniciar un proceso. Puede desconectarse del host remoto, volver a conectarse y su proceso seguirá ejecutándose.
- Puede compartir una sesión 'screen' entre dos conexiones SSH diferentes para que dos personas puedan ver lo mismo, en tiempo real.

Este comando ofrece una alta productividad para entornos remotos como el que usamos en nuestro clúster de servidores. Esperamos que esta guía les sea de gran utilidad. *¡Familiarizarse con él será un tiempo bien empleado!*

¿Cómo usar el comando 'screen'?

Para comenzar 'screen', simplemente escríbalo como se muestra a continuación y presione *Enter*:



```
rhernandez@huellero3:~$ screen
```

Verá una página de información de licencia. Puede presionar la barra espaciadora para leer la segunda página o *Enter* para volver al símbolo del sistema.

```
GNU Screen version 4.06.02 (GNU) 23-Oct-17

Copyright (c) 2015-2017 Juergen Weigert, Alexander Naumov, Amadeusz Slawinski
Copyright (c) 2010-2014 Juergen Weigert, Sadrul Habib Chowdhury
Copyright (c) 2008-2009 Juergen Weigert, Michael Schroeder, Micah Cowan, Sadrul Habib Chowdhury
Copyright (c) 1993-2007 Juergen Weigert, Michael Schroeder
Copyright (c) 1987 Oliver Laumann

This program is free software; you can redistribute it and/or modify it under the terms of the GNU
General Public License as published by the Free Software Foundation; either version 3, or (at your
option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even
the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
Public License for more details.

You should have received a copy of the GNU General Public License along with this program (see the
file COPYING); if not, see http://www.gnu.org/licenses/, or contact Free Software Foundation, Inc., 51
Franklin Street, Fifth Floor, Boston, MA 02111-1301 USA.

Send bugreports, fixes, enhancements, t-shirts, money, beer & pizza to screen-devel@gnu.org

Capabilities:
+copy +remote-detach +power-detach +multi-attach +multi-user +font +color-256 +utf8 +rxvt
+builtin-telnet
```

Se queda en el símbolo del sistema y no parece que haya sucedido mucho. Sin embargo, ahora está ejecutando un *shell* dentro de un emulador de terminal multiplexado. ¿Por qué esto es bueno? Bueno, comencemos un proceso que llevará mucho tiempo completar o qué se ejecute de manera permanente.

Podrá consultar la ayuda del comando 'screen' con la lista de comandos disponibles, presionando 'Ctrl + a' (combinación de prefijo que es necesaria para ejecutar cualquier comando dentro de la sesión 'screen') y luego '?':

```

Screen key bindings, page 1 of 1.
Command key: ^A  Literal ^A: a

break    ^B b    history  { }    other    ^A    split    S
clear    C    info    i    pow break B    suspend ^Z z
colon    :    kill    K k    pow detach D    time    ^T t
copy     ^[ [    lastmsg ^M m    prev    ^H ^P p ^?    title    A
detach   ^D d    license ,    quit    \    vbell    ^G
digraph  ^V    lockscreen ^X x    readbuf <    version v
displays *    log    H    redisplay ^L l    width    W
dumftermcap .    login L    remove X    windows ^W w
fit       F    meta    a    removebuf =    wrap    ^R r
flow      ^F f    monitor M    reset    Z    writebuf >
focus    ^I    next    ^@ ^N sp n    screen ^C c    xoff    ^S s
hardcopy  h    number N    select    '    xon    ^Q q
help      ?    only    Q    silence -

^]    paste .
"    windowlist -b
~    select -
0    select 0
1    select 1
2    select 2
3    select 3
4    select 4
5    select 5
6    select 6
7    select 7
8    select 8
9    select 9
I    login on
O    login off
]    paste .
|    split -v
:kB: focus prev

```

Para salir, podrá presionar la barra espaciadora para leer la segunda página o *Enter* para volver al símbolo del sistema.

Podremos ejecutar cualquier proceso (de larga duración o de ejecución continua) como mismo lo haríamos antes en un terminal, en este caso sin necesidad del comando 'nohup'. Por ejemplo, ejecutemos un 'ping' y se comienza a mostrar la salida.

```

PING soma.ucm.cl (192.168.101.21) 56(84) bytes of data.
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=1 ttl=64 time=0.304 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=2 ttl=64 time=0.239 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=3 ttl=64 time=0.199 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=4 ttl=64 time=0.179 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=5 ttl=64 time=0.191 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=6 ttl=64 time=0.192 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=7 ttl=64 time=0.186 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=8 ttl=64 time=0.191 ms

```

Para probar la ejecución en segundo plano, presione 'Ctrl + a', suelta esas teclas y luego presiona 'd' para llevar la pantalla a segundo plano.

El proceso se seguirá ejecutando, pero la ventana que muestra la salida dejará de visualizarse. Regresará a la ventana de terminal desde la que se inició la sesión 'screen' y se mostrará un mensaje diciendo que se ha separado una ventana 'screen' etiquetada '30106.pts-0.huellero3'.

```

[detached from 30106.pts-0.huellero3]
rhernandez@huellero3:~$ █

```

Necesita el número desde el principio del nombre de la ventana para volver a lanzarla a primer plano. Si lo olvida, siempre puede usar la opción (lista) ' -ls ', como se muestra a continuación, para obtener una lista de las ventanas separadas:

```
rhernandez@huellero3:~$ screen -ls
There is a screen on:
  30106.pts-0.huellero3  (15/09/20 12:37:18)  (Detached)
1 Socket in /run/screen/S-rhernandez.
rhernandez@huellero3:~$
```

Cuando desee, puede usar la opción (volver a adjuntar) ' -r ' y el número de la sesión para volver a lanzarla a primer plano. Puede evitar el paso anterior de listar las sesiones presionando ' TAB ' después de ' -r ' y se autocompletarán las sesiones activas.

```
rhernandez@huellero3:~$ screen -r 30106
```

La ventana que ha estado funcionando en segundo plano ahora vuelve a la ventana de su terminal como si nunca se hubiera ido.

```
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=767 ttl=64 time=0.186 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=768 ttl=64 time=0.153 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=769 ttl=64 time=0.193 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=770 ttl=64 time=0.195 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=771 ttl=64 time=0.194 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=772 ttl=64 time=0.191 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=773 ttl=64 time=0.153 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=774 ttl=64 time=0.197 ms
```

Si es un proceso que se ejecutará hasta su conclusión, eventualmente se completará. Si es un proceso continuo (como el ejemplo anterior), eventualmente querrá terminarlo. De cualquier manera, cuando finalice el proceso, puede escribir 'exit' para salir de la sesión 'screen'. Alternativamente, puede presionar 'Ctrl + a' y luego 'k' para matar a la fuerza una ventana.

Regresará a la ventana de su terminal anterior. Debido a que cerramos nuestra única ventana separada, recibimos un mensaje de '[screen is terminating]'.

Uso de sesiones 'screen' con nombre

Puede utilizar la opción (nombre de sesión) ' -S ' para nombrar su sesión 'screen'. Si usa un nombre memorable en lugar de la identidad numérica de la sesión, es más conveniente volver a conectarse a una sesión. Escribimos lo siguiente para nombrar nuestra sesión "big-test":

```
rhernandez@huellero3:~$ screen -S big-test
```

Cuando inicia nuestra sesión, veremos lo mismo que hemos descrito anteriormente, un *shell* en blanco con símbolo del sistema. Podremos ejecutar nuestros procesos de larga duración de la misma manera que de costumbre.

Cuando comience la ejecución de su proceso, podrá enviar la sesión a segundo plano presionamos 'Ctrl + a' y luego 'd' para desconectar la sesión. Podemos listar las sesiones activas, para ver los detalles de nuestra sesión separada:

```
[detached from 30151.big-test]
rhernandez@huellero3:~$ screen -ls
There is a screen on:
      30151.big-test  (15/09/20 13:20:32)    (Detached)
1 Socket in /run/screen/S-rhernandez.
rhernandez@huellero3:~$
```

Detrás del identificador numérico (30151), vemos el nombre de nuestra sesión (big-test). Escribimos lo siguiente, incluido el nombre de la sesión, para lanzarla a primer plano nuevamente:

```
[detached from 30151.big-test]
rhernandez@huellero3:~$ screen -ls
There is a screen on:
      30151.big-test  (15/09/20 13:20:32)    (Detached)
1 Socket in /run/screen/S-rhernandez.
rhernandez@huellero3:~$ screen -r big-test
```

Nos volvimos a conectar a nuestra sesión 'screen' y vemos nuestro que proceso aún está en curso.

¿Cómo consultar las salidas por pantalla?

En caso que nos hayamos perdido salidas por pantallas que necesitemos consultar, nos daremos cuenta que el *scroll* no funciona como de costumbre. Esto se debe a que el comando 'screen' tiene su propio búfer de desplazamiento, ya que es un multiplexor de terminal y tiene que lidiar con varios búferes.

Así, para consultar las salidas por pantalla que hayan pasado, la mejor manera es desplazándose usando el "modo de copia" (que puede usar para copiar texto):

- Presiona la combinación de prefijo 'Ctrl + a', luego presiona *Esc*.
- Muévase hacia arriba / abajo con las teclas de flecha (↑ y ↓).
- Cuando haya terminado, presione 'q' o *Esc* para volver al final del búfer de desplazamiento.

Si en lugar de 'q' o *Esc*, presiona *Enter* y luego mueve el cursor, estará seleccionando el texto para copiar, y presionando *Enter* por segunda vez lo copiará. Luego puede pegar en la misma sesión 'screen' con 'Ctrl + a' seguido de 'j'.

Los pasos anteriores son válidos para navegar dentro de una sesión de pantalla, pero hay una funcionalidad incorporada en el comando 'screen' para almacenar todo en un archivo a través de la opción '-L' que activa el registro de salida automático:

```
rhernandezghueller3:~$ screen -L -S testscreen
```

En cualquier momento podrá consultar el archivo de salida, por ejemplo, usando el comando 'more':

```
rhernandezghueller3:~$ more screenlog.0
```

Se mostrará el contenido del archivo, pudiendo consultar cualquier salida de los comandos ejecutados:

```
rhernandezghueller3:~$ exit
exit
rhernandezghueller3:~$ ping soma.ucm.cl
PING soma.ucm.cl (192.168.101.21) 56(84) bytes of data:
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=1 ttl=64 time=0.247 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=2 ttl=64 time=0.261 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=3 ttl=64 time=0.201 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=4 ttl=64 time=0.190 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=5 ttl=64 time=0.180 ms
^C
--- soma.ucm.cl ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 70ms
rtt min/avg/max/mdev = 0.180/0.215/0.261/0.037 ms
rhernandezghueller3:~$ exit
exit
rhernandezghueller3:~$
```

Por defecto, el nombre del archivo de registro es "screenlog.0". Puede establecer un nuevo nombre de archivo de registro con la opción '-Logfile file':

```
rhernandezghueller3:~$ screen -L -Logfile testscreen.log -S testscreen
```

En caso que no haya activado el registro de salida mediante la opción '-L' al iniciar la sesión, podrá activarlo una vez en la sesión presionando 'Ctrl + a' y luego 'H' ('Shift + h').

Usando la sesión 'screen' con múltiples ventanas

Hasta ahora, hemos utilizado el comando 'screen' con un solo proceso en segundo plano en una ventana separada. Sin embargo, es posible ejecutar varios procesos simultáneamente y visualizarlos en diferentes ventanas dentro de la misma sesión 'screen'.

Para ejemplificar el uso de múltiples ventanas en una misma sesión 'screen', escribimos lo siguiente para iniciar una sesión llamada "monitor":

```
rhernandez@huellero3:~$ screen -S monitor
```

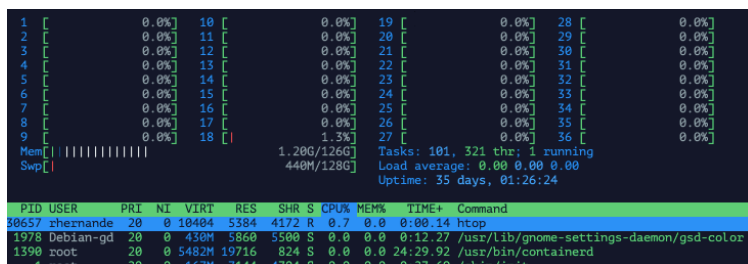
En el símbolo del sistema de la nueva sesión, iniciaremos un 'ping'. Esto se mantendrá ejecutando de manera continua y aparecerán nuevos mensajes a medida que se produzcan. La ejecución no regresará al símbolo del sistema mientras esté ejecutándose el *ping*.

Al mismo tiempo, se desea ejecutar otra aplicación (comando), por lo que necesitamos una nueva ventana (*window*) dentro de la sesión activa. Presionamos 'Ctrl + a' y luego 'c' para crear una nueva ventana. Vamos a utilizar 'htop' para ejecutar y visualizar el estado de uso del servidor.

En el nuevo símbolo del sistema, escribimos lo siguiente:

```
rhernandez@huellero3:~$ htop
```

Se muestra la salida del *htop* y se actualiza constantemente:



The screenshot shows the htop interface with system statistics at the top and a list of processes below. The statistics include memory usage (1.20G/126G), tasks (101), and uptime (35 days, 01:26:24). The process list shows several running processes, including htop, gnome-settings-daemon, and containerd.

Nuestros dos procesos se están ejecutando ahora. Para saltar entre las ventanas activas, presione 'Ctrl + a' y luego el número de la ventana. La primera que creamos es la ventana cero (0), la siguiente es la ventana 1, y así sucesivamente. Para saltar a la primera ventana (donde estamos ejecutando el *ping*), presionamos 'Ctrl + a' y luego '0'. También podemos listar las ventanas activas y elegir la deseada presionando 'Ctrl + a' y luego '"' (comillas dobles).



The screenshot shows the output of the 'screen -ls' command, displaying two active windows: '0 bash' and '1 bash'.

Esto es bastante útil en diferentes escenarios. Podemos desconectarnos de esta sesión ('Ctrl + a' y 'd'); podemos volver a volver más tarde y ambas sesiones seguirán ejecutándose. Nuevamente, para cambiar entre las ventanas de la manera antes explicada.

Algo muy útil también es ver ambas ventanas en una misma pantalla. Cuando haga esto, ampliará la ventana de su terminal a un tamaño que haga que este paso sea útil. En este ejemplo, se limita al tamaño de las capturas de pantalla, por lo que las ventanas se verán un poco estrechas.

Para hacer esto, presionamos 'Ctrl + a', y luego 'S' ('Shift + s', se requiere una "S" mayúscula). La ventana se divide en dos "regiones":

```

1 [ 0.0%] 10 [ 0.0%] 19 [ 0.0%] 28 [ 0.0%]
2 [ 0.0%] 11 [ 0.0%] 20 [ 0.0%] 29 [ 0.0%]
3 [ 0.0%] 12 [ 0.0%] 21 [ 0.0%] 30 [ 0.0%]
4 [ 0.0%] 13 [ 0.0%] 22 [ 0.0%] 31 [ 0.0%]
5 [ 0.0%] 14 [ 0.0%] 23 [ 0.0%] 32 [ 0.7%]
6 [ 0.0%] 15 [ 0.0%] 24 [ 0.0%] 33 [ 0.0%]
7 [ 0.0%] 16 [ 0.0%] 25 [ 0.0%] 34 [ 0.0%]
8 [ 0.0%] 17 [ 0.0%] 26 [ 0.0%] 35 [ 0.0%]
9 [ 0.7%] 18 [ 1.3%] 27 [ 0.0%] 36 [ 0.0%]
Mem[|||||] 1.19G/126G Tasks: 101, 321 thr: 1 running
Swp[|] 440M/128G Load average: 0.08 0.02 0.01
Uptime: 35 days, 01:37:24

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
30659 rhermande 20 0 10404 5372 4152 R 1.3 0.0 0:05.41 htop
30571 rhermande 20 0 12584 2784 1804 S 0.0 0.0 0:00.17 SCREEN -S monitor
1390 root 20 0 5482M 19740 824 S 0.0 0.0 24:30.33 /usr/bin/containerd
1604 root 20 0 87144 1364 1104 S 0.0 0.0 28:00.30 /sbin/apcupsd
2137 root 20 0 5482M 19740 824 S 0.0 0.0 0:25.00 /usr/bin/containerd
1925 Debian-gd 20 0 4328M 60444 21968 S 0.0 0.0 19:33.79 /usr/bin/gnome-shell
1580 root 20 0 3583M 16936 2628 S 0.0 0.0 33:18.31 /usr/bin/dockerd -H fd:// --containerd=/z
2178 root 20 0 3583M 16936 2628 S 0.0 0.0 0:44.94 /usr/bin/dockerd -H fd:// --containerd=/z
2193 root 20 0 3583M 16936 2628 S 0.0 0.0 0:49.63 /usr/bin/dockerd -H fd:// --containerd=/z
1611 root 20 0 3583M 16936 2628 S 0.0 0.0 0:48.43 /usr/bin/dockerd -H fd:// --containerd=/z
1621 root 20 0 3583M 16936 2628 S 0.0 0.0 0:43.14 /usr/bin/dockerd -H fd:// --containerd=/z
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit
1 bash

```

La región superior aún se muestra el *htop* y la región inferior está en blanco. Para mover el cursor a la región inferior, presionamos 'Ctrl + a', y luego *Tab*.

El cursor se mueve a la región inferior, que en realidad es solo un espacio vacío. Al no ser un *shell*, no podemos escribir nada en él. Para obtener una pantalla útil, presionamos 'Ctrl + a' y luego presionamos 'O' para mostrar la ventana donde estamos ejecutando el *ping* en esta región.

```

1 [ 0.0%] 10 [ 0.0%] 19 [ 0.0%] 28 [ 0.0%]
2 [ 0.0%] 11 [ 0.7%] 20 [ 0.0%] 29 [ 0.0%]
3 [ 0.0%] 12 [ 0.0%] 21 [ 0.0%] 30 [ 0.0%]
4 [ 0.0%] 13 [ 0.0%] 22 [ 0.0%] 31 [ 0.0%]
5 [ 0.0%] 14 [ 0.0%] 23 [ 0.0%] 32 [ 0.0%]
6 [ 0.0%] 15 [ 0.0%] 24 [ 0.0%] 33 [ 0.0%]
7 [ 0.0%] 16 [ 0.0%] 25 [ 0.0%] 34 [ 0.0%]
8 [ 0.0%] 17 [ 0.0%] 26 [ 0.0%] 35 [ 0.0%]
9 [ 0.7%] 18 [ 1.3%] 27 [ 0.0%] 36 [ 0.0%]
Mem[|||||] 1.20G/126G Tasks: 101, 321 thr: 1 running
Swp[|] 440M/128G Load average: 0.00 0.00 0.00
Uptime: 35 days, 01:45:16

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
30659 rhermande 20 0 10404 5372 4152 R 1.3 0.0 0:10.14 htop
30571 rhermande 20 0 12584 2784 1804 S 0.0 0.0 0:00.27 SCREEN -S monitor
1624 root 20 0 3583M 16936 2628 S 0.0 0.0 0:40.23 /usr/bin/dockerd -H fd:// --containerd=/z
2410 root 20 0 5482M 19740 824 S 0.0 0.0 0:28.37 /usr/bin/containerd
1390 root 20 0 5482M 19740 824 S 0.0 0.0 24:30.62 /usr/bin/containerd
1925 Debian-gd 20 0 4328M 60444 21968 S 0.0 0.0 19:34.04 /usr/bin/gnome-shell
1242 rtkit 21 1 149M 1220 1220 S 0.0 0.0 0:22.83 /usr/lib/rtkit/rtkit-daemon
1580 root 20 0 3583M 16936 2628 S 0.0 0.0 33:18.87 /usr/bin/dockerd -H fd:// --containerd=/z
646 root 20 0 106M 41652 38976 S 0.0 0.0 0:32.22 /lib/systemd/systemd-journald
2183 root 20 0 3583M 16936 2628 S 0.0 0.0 0:38.54 /usr/bin/dockerd -H fd:// --containerd=/z
30650 rhermande 20 0 11240 2924 2716 S 0.0 0.0 0:00.14 ping soma.ucm.cl
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit
1 bash
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=1382 ttl=64 time=0.198 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=1383 ttl=64 time=0.233 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=1384 ttl=64 time=0.239 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=1385 ttl=64 time=0.216 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=1386 ttl=64 time=0.196 ms
64 bytes from soma.ucm.cl (192.168.101.21): icmp_seq=1387 ttl=64 time=0.231 ms

```

Esto nos da ambas salidas en vivo en una ventana dividida. Si desconectamos la sesión y luego la volvemos a abrir, perderemos la vista de panel dividido. Sin embargo, podemos restaurar la vista dividida siguiendo los pasos ya explicados, con los siguientes atajos de teclado:

1. Ctrl + a, S: divide la ventana horizontalmente.

2. **Ctrl + a, Tab**: ir a la región inferior.
3. **Ctrl + a, 0**: Ventana de visualización cero en la región inferior.

Podemos llevar las cosas incluso un paso más allá. Ahora dividiremos el panel inferior verticalmente y agregaremos un tercer proceso a la pantalla. Con el cursor en la región inferior, presionamos '**Ctrl + a**' y '**c**' para crear una nueva ventana con un *shell* en ella. La región inferior muestra la nueva ventana y nos da un símbolo del sistema.

```

1 [ 0.0%] 10 [ 0.0%] 19 [ 0.0%] 28 [ 0.0%]
2 [ 2.0%] 11 [ 0.0%] 20 [ 0.0%] 29 [ 0.0%]
3 [ 0.0%] 12 [ 0.0%] 21 [ 0.0%] 30 [ 0.0%]
4 [ 0.0%] 13 [ 0.0%] 22 [ 0.0%] 31 [ 0.7%]
5 [ 0.0%] 14 [ 0.0%] 23 [ 0.0%] 32 [ 0.0%]
6 [ 0.0%] 15 [ 0.0%] 24 [ 0.0%] 33 [ 0.0%]
7 [ 0.0%] 16 [ 0.0%] 25 [ 0.0%] 34 [ 0.0%]
8 [ 0.0%] 17 [ 0.0%] 26 [ 0.0%] 35 [ 0.0%]
9 [ 0.7%] 18 [ 0.0%] 27 [ 0.0%] 36 [ 0.0%]
Mem [|||||] 1.20G/126G Tasks: 102, 321 thr: 1 running
Swp [||] 440M/128G Load average: 0.02 0.01 0.00
Uptime: 35 days, 01:51:49

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
30659 rhernande 20 0 10404 5372 4152 R 0.7 0.0 0:15.19 htop
30571 rhernande 20 0 12756 2988 1804 S 0.7 0.0 0:00.55 SCREEN -S monitor
1390 root 20 0 5482M 19740 824 S 0.0 0.0 24:30.89 /usr/bin/containerd
1580 root 20 0 3583M 16936 2628 S 0.0 0.0 33:19.28 /usr/bin/dockerd -H fd:// --containerd=/x
2137 root 20 0 5482M 19740 824 S 0.0 0.0 0:25.03 /usr/bin/containerd
2186 root 20 0 5482M 19740 824 S 0.0 0.0 0:26.93 /usr/bin/containerd
2482 root 20 0 5482M 19740 824 S 0.0 0.0 0:30.49 /usr/bin/containerd
1451 root 20 0 5482M 19740 824 S 0.0 0.0 0:25.83 /usr/bin/containerd
2119 root 20 0 5482M 19740 824 S 0.0 0.0 0:29.72 /usr/bin/containerd
2103 root 20 0 5482M 19740 824 S 0.0 0.0 0:31.35 /usr/bin/containerd
F1Help F2Setup F3Search F4Filter F5Free F6SortBy F7Nice F8Nice F9Kill F10Quit
1 bash
rhernandez@huellero3:~$

```

A continuación, ejecutamos el comando '**df**' para verificar el uso del sistema de archivos:

```

1 [ 0.0%] 10 [ 0.0%] 19 [ 0.0%] 28 [ 0.0%]
2 [ 1.3%] 11 [ 0.0%] 20 [ 0.0%] 29 [ 0.0%]
3 [ 0.0%] 12 [ 0.0%] 21 [ 0.0%] 30 [ 0.0%]
4 [ 0.0%] 13 [ 0.0%] 22 [ 0.0%] 31 [ 0.0%]
5 [ 0.0%] 14 [ 0.0%] 23 [ 0.0%] 32 [ 0.7%]
6 [ 0.0%] 15 [ 0.0%] 24 [ 0.0%] 33 [ 0.0%]
7 [ 0.0%] 16 [ 0.0%] 25 [ 0.0%] 34 [ 0.0%]
8 [ 0.0%] 17 [ 0.0%] 26 [ 0.0%] 35 [ 0.0%]
9 [ 0.7%] 18 [ 0.0%] 27 [ 0.0%] 36 [ 0.0%]
Mem [|||||] 1.20G/126G Tasks: 102, 321 thr: 1 running
Swp [||] 440M/128G Load average: 0.00 0.00 0.00
Uptime: 35 days, 01:53:36

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
30659 rhernande 20 0 10404 5372 4152 R 1.3 0.0 0:16.55 htop
1604 root 20 0 87144 1364 1104 S 0.0 0.0 28:00.50 /sbin/apcupsd
1376 rtkit RT 1 149M 1220 1220 S 0.0 0.0 0:09.56 /usr/lib/rtkit/rtkit-daemon
1390 root 20 0 5482M 19740 824 S 0.0 0.0 24:30.92 /usr/bin/containerd
30362 rhernande 20 0 17220 5476 4080 S 0.0 0.0 0:00.64 sshd: rhernandez@pts/0
29590 rhernande 20 0 2652 2016 1856 S 0.0 0.0 0:00.05 /usr/lib/openssh/sftp-server
1580 root 20 0 3583M 16936 2628 S 0.0 0.0 33:19.42 /usr/bin/dockerd -H fd:// --containerd=/x
30571 rhernande 20 0 12756 2988 1804 S 0.0 0.0 0:00.57 SCREEN -S monitor
30659 rhernande 20 0 11240 2924 2716 S 0.0 0.0 0:00.19 ping som.ucm.cl
1444 root 20 0 5482M 19740 824 S 0.0 0.0 3:21.68 /usr/bin/containerd
F1Help F2Setup F3Search F4Filter F5Free F6SortBy F7Nice F8Nice F9Kill F10Quit
1 bash
rhernandez@huellero3:~$ df -h
S.ficheros Tamaño Usados Disp Uso% Montado en
udev 68G 0 68G 0% /dev
tmpfs 14G 54M 14G 1% /run
/dev/mapper/litrp2--vg-root 30G 17G 12G 59% /
tmpfs 68G 0 68G 0% /dev/shm
tmpfs 5.3M 8.2k 5.3M 1% /run/lock
tmpfs 68G 0 68G 0% /sys/fs/cgroup
/dev/sda2 332M 111M 205M 36% /boot
/dev/mapper/litrp2--vg-home 7.8T 623G 6.8T 9% /home

```

Cuando se ejecute *df*, presionamos '**Ctrl + a**' y luego '**|**' (barra vertical o carácter de tubería). Esto divide la región inferior verticalmente. Presionamos '**Ctrl + a**' y *Tab* para movernos a la nueva región. A continuación, presionamos '**Ctrl + a**' y '**0**' para mostrar la ventana donde estamos ejecutando el comando *ping*.

También puede moverse de una región a otra y agregar más divisiones verticales u horizontales. Aquí hay algunas combinaciones de teclas más útiles:

- ## Usando 'screen' sobre SSH

Esto es extremadamente útil si desea iniciar un proceso desde una computadora (por ejemplo, en la U) y luego continuar donde lo dejó en otra computadora (desde la casa).

Compartir una sesión 'screen'

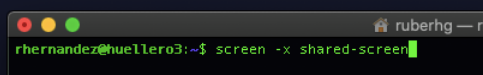
También puede utilizar una sesión 'screen' para permitir que dos personas vean e interactúen con la misma ventana. Digamos que dos personas se conectan por SSH a nuestro servidor desde dos computadoras diferentes.

Una de ellas, después de conectarse por SSH, inicia una sesión 'screen' llamada "shared-screen" usando la opción '-S'. También usa las opciones '-d' (desconectar) y '-m' (creación forzada) para crear una nueva sesión que ya está desconectada. A continuación, usa la opción '-x' (modo multipantalla) para adjuntar la sesión:

```
rhernandez@huellero3:~$ screen -d -m -S shared-screen
rhernandez@huellero3:~$ screen -x shared-screen
```

La otra persona se conectará de forma remota (SSH) al mismo servidor con las mismas credenciales de cuenta, y activará la sesión 'screen' usando la opción '-x' como se muestra a continuación:

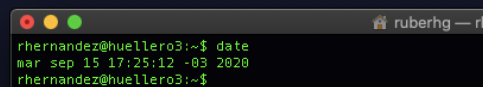
```
rhernandez@huellero3:~$
```



A terminal window titled 'ruberhg - rh' shows the command `rhernandez@huellero3:~$ screen -x shared-screen` being executed.

Ahora, cualquier cosa que escriba una persona, la otra lo verá. Por ejemplo, cuando una persona emite el comando de fecha, ambos lo ven tal como se escribe, así como su salida.

```
rhernandez@huellero3:~$ date
mar sep 15 17:25:12 -03 2020
rhernandez@huellero3:~$
```



A terminal window titled 'ruberhg - rh' shows the same command and output as the first terminal: `rhernandez@huellero3:~$ date` followed by `mar sep 15 17:25:12 -03 2020` and the prompt `rhernandez@huellero3:~$`.