

# CREACIÓN, VISUALIZACIÓN Y EDICIÓN DE ARCHIVOS DE TEXTO

RH124

Capítulo 4

Descripción general	
<b>Meta</b>	Crear, visualizar y editar archivos de texto desde un resultado de comando o en un editor.
<b>Objetivos</b>	<ul style="list-style-type: none"> <li>• Redirigir el resultado de texto de un programa a un archivo o a otro programa.</li> <li>• Editar archivos de texto existentes y crear archivos nuevos a partir de avisos de shell con un editor de texto.</li> <li>• Copiar texto desde una ventana gráfica a un archivo de texto con un editor de texto que se ejecute en un entorno gráfico.</li> </ul>
<b>Secciones</b>	<ul style="list-style-type: none"> <li>• Redirección del resultado a un archivo o programa (y práctica)</li> <li>• Edición de archivos de texto desde el aviso de shell (y práctica)</li> <li>• Edición de archivos de texto con un editor gráfico (y práctica)</li> </ul>
<b>Trabajo de laboratorio</b>	<ul style="list-style-type: none"> <li>• Creación, visualización y edición de archivos de texto</li> </ul>

# Redireccionamiento de la salida a un archivo o programa

RH124

# Objetivos

Tras finalizar esta sección, los estudiantes deberían poder realizar lo siguiente:

- Describir los términos técnicos "entrada estándar", "salida estándar" y "error estándar".
- Utilizar caracteres de redireccionamiento para controlar la salida a archivos.
- Usar tubería para controlar la salida a otros programas.

## Entrada estándar, salida estándar y error estándar

Un programa, o un *proceso*, en ejecución necesita leer entradas desde alguna parte y escribir salidas en la pantalla o en archivos. Un comando ejecutado desde el aviso de shell normalmente lee su entrada desde el teclado y envía su salida a su ventana de terminal.

Un proceso utiliza canales numerados denominados *descriptores de archivo* para obtener entradas y enviar salidas. Todos los procesos tendrán al menos tres descriptores de archivo para comenzar. *Entrada estándar* (canal 0) lee entradas desde el teclado. *Salida estándar* (canal 1) envía una salida normal al terminal. *Error estándar* (canal 2) envía mensajes de error al terminal. Si un programa abre conexiones independientes para otros archivos, puede usar descriptores de archivo con números superiores.

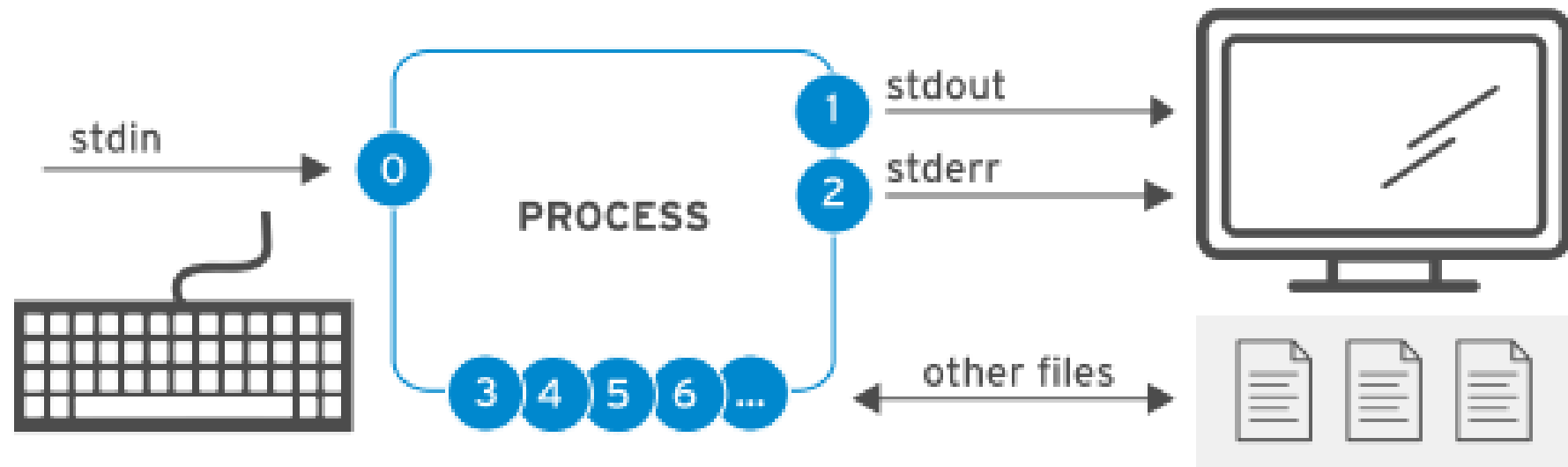


Figura 4.1: Canales de E/S de un proceso (descriptores de archivo)

### Canales (descriptores de archivo)

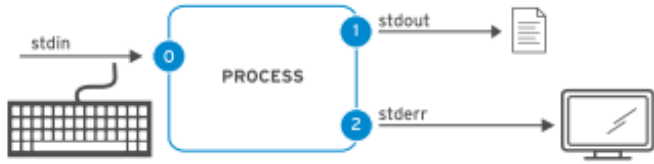
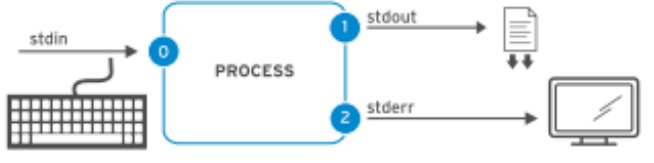
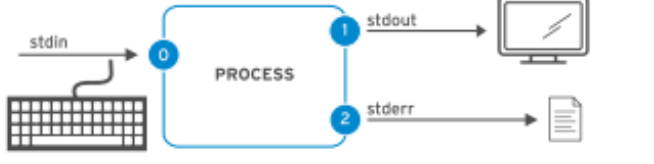
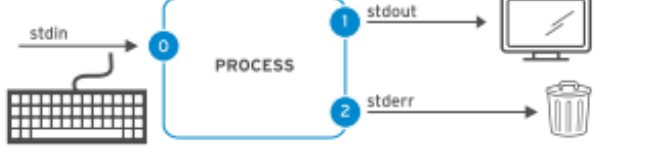

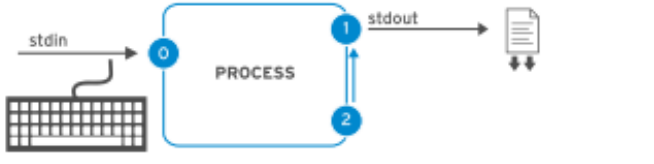
Número	Nombre de canal	Descripción	Conexión predeterminada	Uso
0	<b>stdin</b>	Entrada estándar	Teclado	Solo lectura
1	<b>stdout</b>	Salida estándar	Terminal	Solo escritura
2	<b>stderr</b>	Error estándar	Terminal	Solo escritura
3+	<b>filename</b>	Otros archivos	<i>none (ninguno)</i>	Lectura y escritura

# Redireccionamiento de la salida a un archivo

El redireccionamiento *de E/S* reemplaza los destinos de canales predeterminados con nombres de archivos que representan dispositivos o archivos de salida. Con el uso del redireccionamiento, los mensajes de error y la salida de un proceso que se envían generalmente a la ventana de terminal pueden capturarse como contenido de archivo, enviarse a un dispositivo o descartarse.

El redireccionamiento de **stdout** evita que la salida de un proceso aparezca en el terminal. Como se puede ver en la siguiente tabla, el redireccionamiento de *únicamente stdout* no evita que los mensajes de error **stderr** aparezcan en el terminal. Si el archivo no existe, se creará. Si el archivo existe y el redireccionamiento no es uno que se agregue al archivo, el contenido de archivo se sobrescribirá. El archivo especial **/dev/null** descarta discretamente la salida del canal redirigida a él y es siempre un archivo vacío.

# Operadores de redireccionamiento de salida

Uso	Explicación	Ayuda visual
<code>&gt;file</code>	redirigir <b>stdout</b> para sobrescribir un archivo	
<code>&gt;&gt;file</code>	redirigir <b>stdout</b> para agregar a un archivo	
<code>2&gt;file</code>	redirigir <b>stderr</b> para sobrescribir un archivo	
<code>2&gt;/dev/null</code>	descartar mensajes de error <b>stderr</b> mediante el redireccionamiento a <b>/dev/null</b>	
<code>&gt;file 2&gt;&amp;1</code>	redirigir <b>stdout</b> y <b>stderr</b> para sobrescribir el mismo archivo	
<code>&amp;&gt;file</code>		
Uso	Explicación	Ayuda visual
<code>&gt;&gt;file 2&gt;&amp;1</code>	redirigir <b>stdout</b> y <b>stderr</b> para agregar al mismo archivo	
<code>&amp;&gt;&gt;file</code>		

## Importante

El orden de las operaciones de redireccionamiento es importante. La siguiente secuencia dirige la salida estándar a **file** y, luego, dirige el error estándar al mismo lugar que la salida estándar (**file**).

```
> file 2>&1
```

Sin embargo, la siguiente secuencia realiza el redireccionamiento en el orden opuesto. Dirige el error estándar al lugar predeterminado para la salida estándar (la ventana de terminal, de modo que no hay cambios) y, *luego*, dirige solo la salida estándar a **file**.

```
2>&1 > file
```

Por esto, algunas personas prefieren usar los operadores de redireccionamiento de fusión:

<b>&amp;&gt;file</b>	en lugar de	<b>&gt;file 2&gt;&amp;1</b>
<b>&amp;&gt;&gt;file</b>	en lugar de	<b>&gt;&gt;file 2&gt;&amp;1</b> (en Bash 4 / RHEL 6 y posteriores)

Sin embargo, otros administradores de sistemas y programadores que usan además otras shells relacionadas con **bash** ("shells compatibles con Bourne") para comandos para crear scripts piensan que se deben evitar los operadores de redireccionamiento de fusión más nuevos, dado que no están estandarizados ni implementados en todas aquellas shells y tienen otras limitaciones.

Los autores de este curso tienen una postura neutral respecto de este tema, y es probable que se encuentren ambas sintaxis en el campo.



## Ejemplos de redireccionamiento de salidas

El uso del redireccionamiento permite simplificar muchas tareas de administración de rutina. Use la tabla anterior como ayuda mientras aborda los siguientes ejemplos:

- Guarde un sello de fecha y hora para su posterior consulta.

```
[student@desktopX ~]$ date > /tmp/saved-timestamp
```

- Copie las últimas 100 líneas de un archivo de registro en otro archivo.

```
[student@desktopX ~]$ tail -n 100 /var/log/dmesg > /tmp/last-100-boot-messages
```

- Concatene cuatro archivos en uno.

```
[student@desktopX ~]$ cat file1 file2 file3 file4 > /tmp/all-four-in-one
```

- Detalle los nombres de archivos regulares y ocultos del directorio de inicio en un archivo.

```
[student@desktopX ~]$ ls -a > /tmp/my-file-names
```

- Adjunte salida a un archivo existente.

```
[student@desktopX ~]$ echo "new line of information" >> /tmp/many-lines-of-information  
[student@desktopX ~]$ diff previous-file current-file >> /tmp/tracking-changes-made
```

- En los siguientes ejemplos, se generan errores porque los usuarios normales no tienen acceso a los directorios del sistema. Redirija errores a un archivo mientras se visualiza la salida de un comando normal en el terminal.

```
[student@desktopX ~]$ find /etc -name passwd 2> /tmp/errors
```

- Guarde la salida de un proceso y mensajes de error en archivos separados.

```
[student@desktopX ~]$ find /etc -name passwd > /tmp/output 2> /tmp/errors
```

- Omita y descarte mensajes de error.

```
[student@desktopX ~]$ find /etc -name passwd > /tmp/output 2> /dev/null
```

- Almacene la salida y errores generados en forma conjunta.

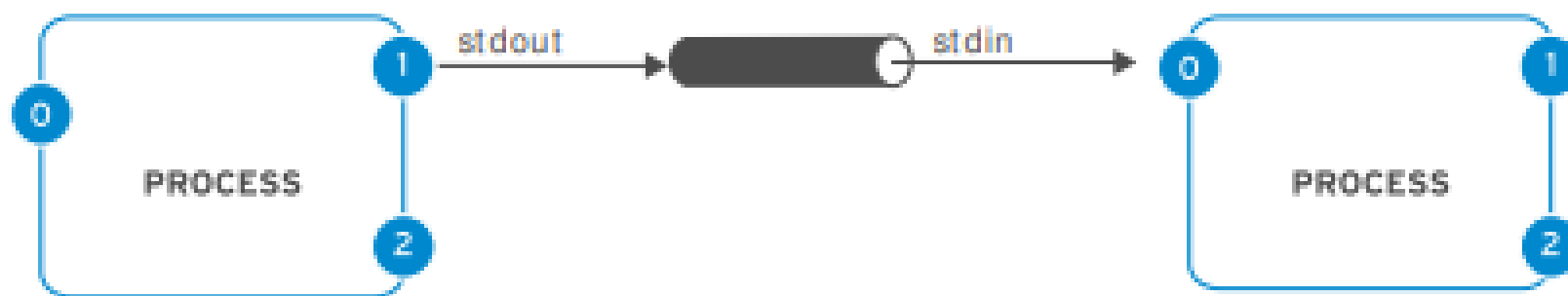
```
[student@desktopX ~]$ find /etc -name passwd &> /tmp/save-both
```

- Adjunte la salida y errores generados en un archivo existente.

```
[student@desktopX ~]$ find /etc -name passwd >> /tmp/save-both 2>&1
```

## Construcción de tubería

Una *tubería* es una secuencia de uno o más comandos separados por `|`, el carácter de *tubería*. Una tubería conecta la salida estándar del primer comando con la entrada estándar del siguiente comando.



*Figura 4.8: Tubería de E/S de proceso*

Las tuberías permiten que la salida de un proceso sea manipulada y formateada por otros procesos antes de su salida al terminal. Una imagen mental útil es imaginar que los datos "fluyen" a través de la tubería de un proceso a otro, y que son alterados levemente por cada comando en la tubería a través de la cual pasan.

## nota

Las tuberías y el redireccionamiento de E/S manipulan la salida y la entrada estándares. El *redireccionamiento* envía la salida estándar a *files* o recibe la entrada estándar de estos. Las *tuberías* envían la salida estándar a otro *proceso* o reciben la entrada estándar de este.

### Ejemplos de tuberías

Este ejemplo toma la salida del comando **ls** y usa **less** para mostrarla en el terminal de a una pantalla por vez.

```
[student@desktopX ~]$ ls -l /usr/bin | less
```

La salida del comando **ls** se envía por tubería a **wc -l**, que cuenta la cantidad de líneas recibidas de **ls** y la imprime en el terminal.

```
[student@desktopX ~]$ ls | wc -l
```

En esta tubería, **head** enviará las primeras 10 líneas de salida de **ls -t**, y el resultado final se redirigirá a un archivo.

```
[student@desktopX ~]$ ls -t | head -n 10 > /tmp/ten-last-changed-files
```

### Tuberías, redireccionamiento y tee

Cuando el redireccionamiento se combina con una tubería, la shell primero configura toda la tubería y, luego, dirige la entrada/salida. Esto significa que si el redireccionamiento de la salida se usa en el *medio* de una tubería, la salida irá al archivo y no al siguiente comando en la tubería.

En este ejemplo, la salida del comando **ls** irá al archivo, y **less** no mostrará nada en el terminal.

```
[student@desktopX ~]$ ls > /tmp/saved-output | less
```

El comando **tee** se usa para proporcionar soluciones alternativas para esto. En una tubería, **tee** copiará su entrada estándar y además redirigirá su salida estándar a los archivos

denominados como argumentos para el comando. Si se imaginan los datos como agua que fluye a través de una tubería, se puede visualizar **tee** como una junta en "T" en la tubería que dirige la salida en dos direcciones.

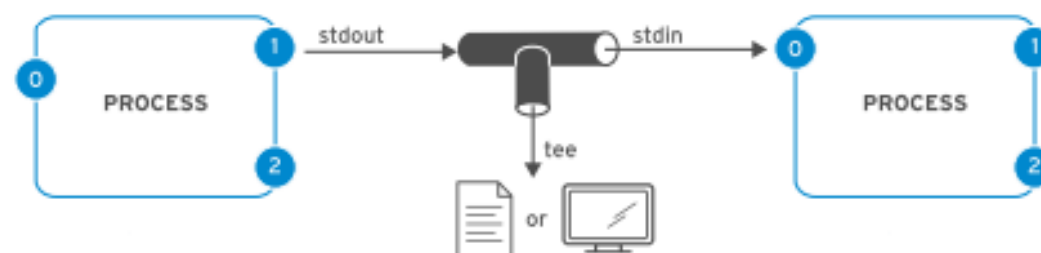


Figura 4.9: Tubería de E/S de proceso con tee

### Ejemplos de tuberías que usan el comando tee

Este ejemplo redirigirá la salida del comando **ls** al archivo y la pasará a **less** para que se muestre en el terminal de a una pantalla por vez.

```
[student@desktopX ~]$ ls -l | tee /tmp/saved-output | less
```

Si **tee** se usa al final de una tubería, la salida final de un comando se puede guardar y enviar al terminal al mismo tiempo.

```
[student@desktopX ~]$ ls -t | head -n 10 | tee /tmp/ten-last-changed-files
```

Este ejemplo más sofisticado aprovecha el hecho de que existe un *archivo de dispositivo* especial que representa el terminal. El nombre del archivo de dispositivo para un terminal específico puede determinarse mediante la ejecución del comando **tty** en su aviso de shell. Luego, se puede usar **tee** para redirigir la salida a ese archivo para mostrarla en la ventana de terminal, mientras que la salida estándar se puede pasar a algún otro programa a través de la tubería. En este caso, **mail** enviará por correo electrónico la salida a student@desktop1.example.com.

```
[student@desktopX ~]$ tty
/dev/pts/0
[student@desktopX ~]$ ls -l | tee /dev/pts/0 | mail student@desktop1.example.com
```

## Importante

El error estándar puede redirigirse por la tubería, pero no se pueden usar los operadores de redireccionamiento de fusión (`&>` y `&>>`) para esto.

La siguiente es la manera correcta de redirigir la salida y el error estándares a través de una tubería:

```
[student@desktopX ~]$ find -name / passwd 2>&1 | less
```

## Referencias

**info bash** (*Manual de referencia de Bash para GNU*)

- Sección 3.2.2: Tuberías
- Sección 3.6: Redireccionamientos

**info coreutils 'tee invocation'** (*Manual de GNU coreutils*)

- Sección 17.1: Redireccionamiento de la salida a varios archivos o procesos

Páginas del manual: **bash(1)**, **cat(1)**, **head(1)**, **less(1)**, **mail(1)**, **tee(1)**, **tty(1)**, **wc(1)**

# Práctica: Redirección y canalizaciones de E/S

# Edición de archivos de texto desde el aviso de shell

RH124

# Objetivos

Tras finalizar esta sección, los estudiantes deberían poder realizar lo siguiente:

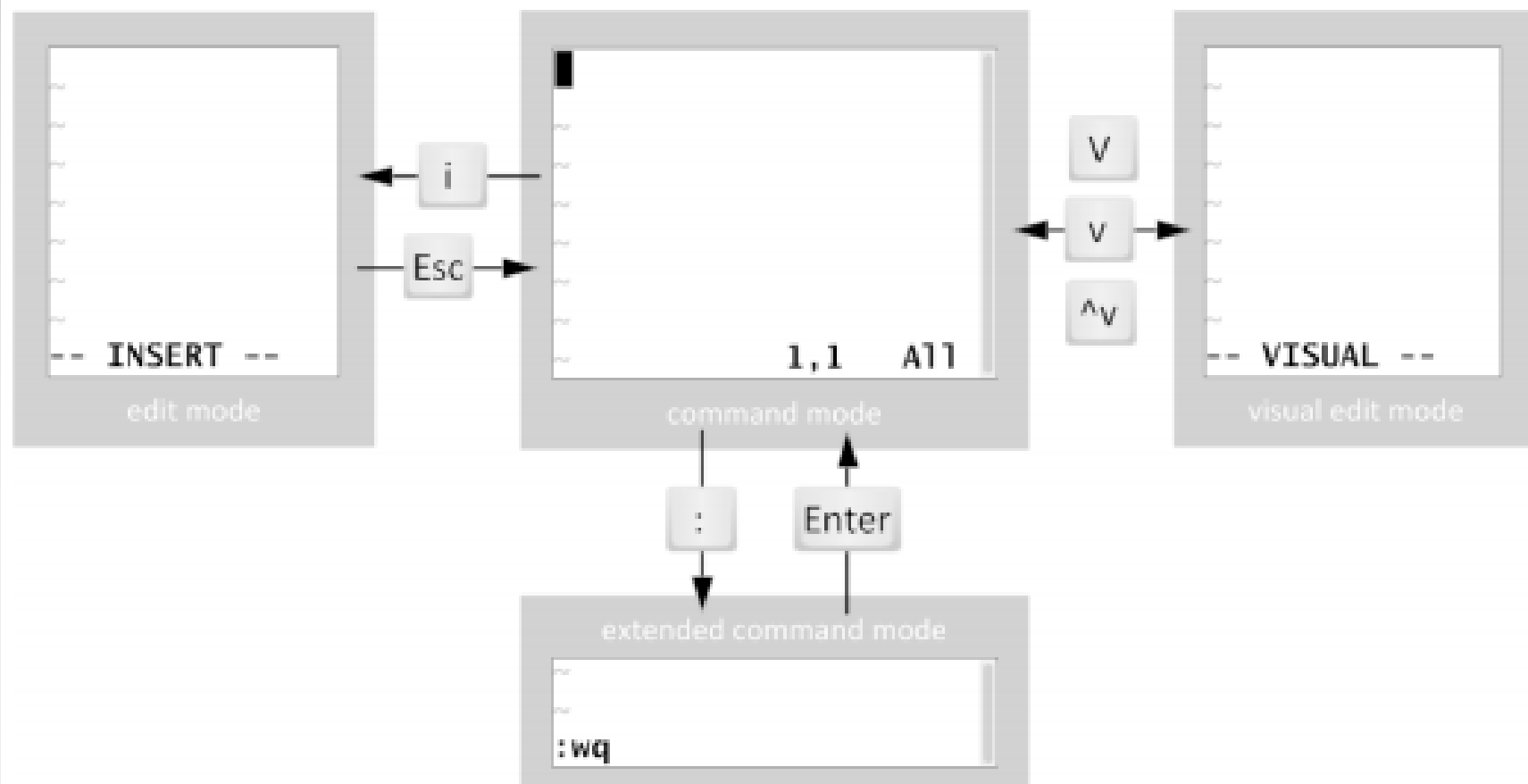
- Crear archivos nuevos y editar archivos existentes de texto desde el aviso de shell.
- Navegar en un editor para realizar tareas de edición correctamente.

## Editar archivos con Vim

Un principio clave de diseño de Linux es que la información se almacena en archivos basados en texto. Los archivos de texto incluyen tanto *archivos planos* con filas de información similar como archivos de configuración en `/etc`, y *archivos de Lenguaje de marcado ampliable (XML)*, los cuales definen la estructura de datos mediante etiquetas de texto, las cuales se ven en archivos de configuración de aplicaciones tanto en `/etc` como en `/usr`. La ventaja de los archivos de texto es que se pueden trasladar o compartir entre los sistemas sin necesidad de conversión, y se los puede ver y editar con cualquier editor de texto simple.

Vim es una versión mejorada del editor vi que se distribuye con los sistemas Linux y UNIX. Vim es altamente configurable y eficaz para usuarios avanzados; incluye funciones como edición en pantalla partida, formateo de color y resaltado para la edición de texto.





*Figura 4.10: Cambio entre modos Vim*

Cuando se abre por primera vez, Vim arranca en *modo comando*, utilizado para navegar, cortar y pegar, y otro tipo de manipulación de texto. Ingrese en cada uno de los otros modos con pulsaciones de tecla de caracteres únicos para acceder a funciones de edición específicas:

- Una pulsación de la tecla **i** ingresa al *modo insert*, en el cual todo el texto ingresado se convierte en contenido de archivo. Al presionar **Esc** se vuelve al modo comando.

- Una pulsación de la tecla **v** ingresa al *modo visual*, en el cual se pueden seleccionar varios caracteres para la manipulación de texto. Utilice **V** para líneas múltiples y **Ctrl+v** para seleccionar en bloque. La misma pulsación de tecla utilizada para ingresar al modo visual (**v**, **V** o **Ctrl+v**) se utiliza para salir.
- La pulsación de la tecla **:** comienza el *modo comando extendido* para tareas como escritura del archivo para guardarlo y salida del editor Vim.

### El flujo de trabajo mínimo y básico de Vim

**Vim** tiene pulsaciones de teclas eficaces y coordinadas para tareas de edición avanzadas. Aunque se las considera útiles con la práctica, las capacidades de Vim pueden abrumar a los nuevos usuarios. El siguiente flujo de trabajo presenta las *mínimas* pulsaciones de tecla que todo usuario de Vim debe aprender para realizar *cualquier* tarea de edición.

El instructor demostrará una sesión típica de edición de archivo empleando solo pulsaciones de teclas básicas de Vim.

1. Abrir un archivo con **vim filename**.
2. Repetir este ciclo de entrada de texto tantas veces como lo requiera la tarea:
  - Usar las teclas de flechas para ubicar el cursor.
  - Presionar **i** para ingresar al modo insert.
  - Ingresar el texto.
  - Presionar **Esc** para volver al modo comando.
  - Si es necesario, presionar **u** para deshacer ediciones incorrectas en la línea actual.
3. Repetir este ciclo de eliminación de texto tantas veces como lo requiera la tarea:
  - Usar las teclas de flechas para ubicar el cursor.
  - Presionar **x** para eliminar un texto seleccionado.
  - Si es necesario, usar **u** para deshacer ediciones incorrectas en la línea actual.
4. Para guardar o salir, elija una de las siguientes opciones para escribir o descartar ediciones del archivo:
  - Ingresar **:w** para escribir (guardar) el archivo y permanecer en el modo comando para continuar editando.
  - Ingresar **:wq** para escribir el archivo y salir de Vim.
  - Ingresar **:q!** para salir de Vim, pero descartar todos los cambios del archivo desde la última escritura.

### Reorganización de texto existente

En **Vim**, copiar y pegar se conoce como *jalar y colocar*, utilizando los caracteres de comando **y** y **p**. Comience colocando el cursor en el primer carácter que se seleccionará, luego ingrese al modo visual. Use las teclas de flechas para expandir la selección visual. Cuando esté listo, presione **y** para *yank* (copiar) lo seleccionado en la memoria. Coloque el cursor en la nueva ubicación y, luego, presione **p** para colocar lo seleccionado en el cursor.

El instructor demostrará "yank and put" usando el modo visual.

1. Abrir un archivo con **vim filename**.
2. Repita este ciclo de selección de texto tantas veces como lo requiera la tarea:
  - Use las teclas de flechas para colocar el cursor en el primer carácter.
  - Presione **v** para ingresar al modo visual.
  - Use las teclas de flechas para colocar el cursor en el último carácter.
  - Presione **y** para yank (copiar) lo seleccionado.
  - Use las teclas de flechas para colocar el cursor en el lugar de inserción.
  - Presione **p** para put (pegar) lo seleccionado.
3. Para guardar o salir, elija una de las siguientes opciones para escribir o descartar ediciones del archivo:
  - Ingresar **:w** para escribir (guardar) el archivo y permanecer en el modo comando para continuar editando.
  - Ingresar **:wq** para escribir el archivo y salir de Vim.
  - Ingresar **:q!** para salir de Vim, pero descartar todos los cambios del archivo desde la última escritura.

## nota

Tenga la precaución de no ofrecerle al usuario avanzado de Vim atajos y trucos si no maneja bien los aspectos básicos. Se requiere práctica para que Vim sea eficaz. Se recomienda continuar aprendiendo nuevas pulsaciones de teclas para incrementar la utilidad de Vim. Quienes desean saber hasta dónde se puede extender su utilidad deben buscar en Internet “Vim tips” (sugerencias de Vim).

Se incluye una presentación detallada de **Vim** en el curso *Red Hat Enterprise Linux 7 Administración de sistemas II*.

## Referencias

Página del manual (1)**vim**

Editor Vim

<http://www.vim.org/>

# Práctica: Edición de archivos con Vim

# Edición de archivos de texto con un editor gráfico

RH124

# Objetivos

Tras finalizar esta sección, los estudiantes deberían poder realizar lo siguiente:

- Editar archivos de texto con `gedit`.
- Copiar texto entre ventanas gráficas.

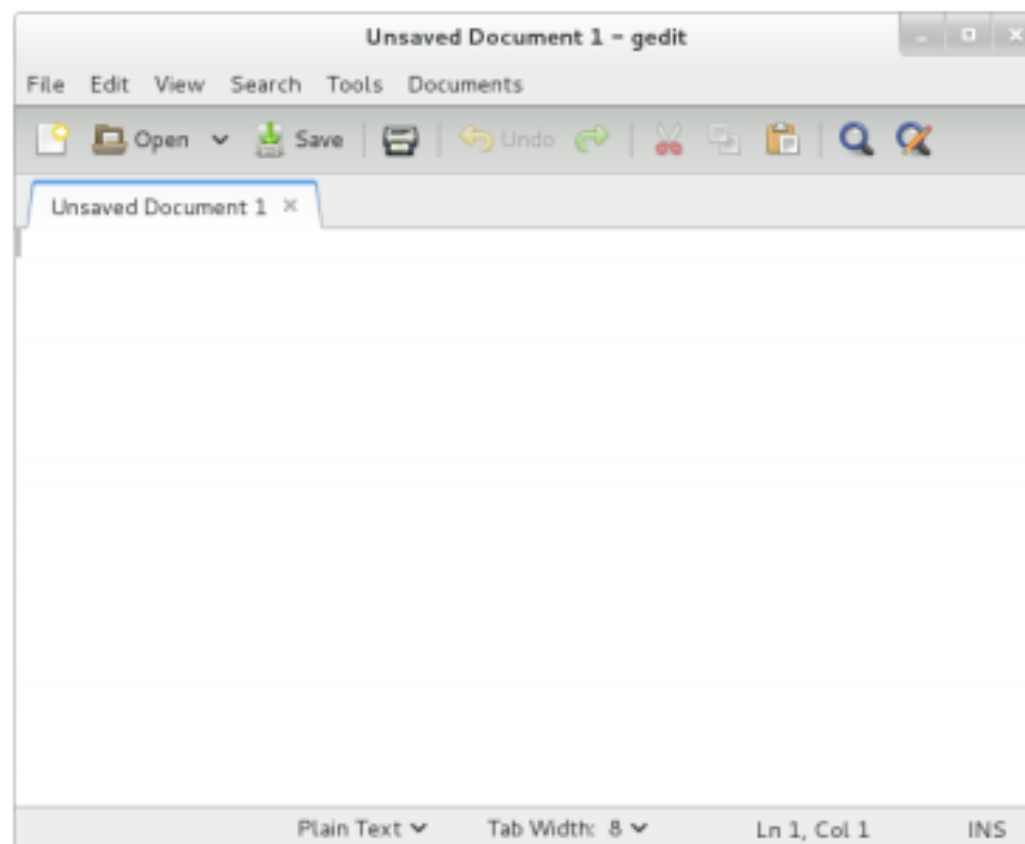
## Edición de archivos con `gedit`

La aplicación `gedit` es un editor de texto con todas las funciones para el entorno de escritorio GNOME. Inicie `gedit` al seleccionar **Applications > Accessories > gedit** en el menú GNOME. Al igual que otras aplicaciones gráficas, `gedit` puede iniciarse sin navegar por el menú. Presione **Alt+F2** para abrir el cuadro de diálogo **Enter a Command**. Escriba `gedit` y presione **Enter**.



## Edición de archivos con gedit

La aplicación **gedit** es un editor de texto con todas las funciones para el entorno de escritorio GNOME. Inicie **gedit** al seleccionar **Applications > Accessories > gedit** en el menú GNOME. Al igual que otras aplicaciones gráficas, **gedit** puede iniciarse sin navegar por el menú. Presione **Alt+F2** para abrir el cuadro de diálogo **Enter a Command**. Escriba **gedit** y presione **Enter**.



*Figura 4.11: Editor de texto gedit*

La ayuda de GNOME incluye una guía de ayuda **gedit** que podrá encontrar seleccionando **Applications > Favorites > Help** del menú GNOME. A continuación, seleccione **Go > All Documents** para ver la lista de aplicaciones gráficas. Desplácese hacia abajo y seleccione el hipervínculo **gedit Text Editor**.

La ayuda de GNOME incluye una guía de ayuda **gedit** que podrá encontrar seleccionando **Applications > Favorites > Help** del menú GNOME. A continuación, seleccione **Go > All Documents** para ver la lista de aplicaciones gráficas. Desplácese hacia abajo y seleccione el hipervínculo **gedit Text Editor**.

### Teclas básicas de gedit

Realice varias tareas de administración de archivos con el menú de **gedit**:

- Para crear un archivo nuevo en gedit, haga clic en el ícono de **papel en blanco** de la barra de herramientas o seleccione **File > New (Ctrl+n)** del menú.
- Para guardar un archivo, haga clic en el ícono de **guardar en unidad de disco duro** de la barra de herramientas o seleccione **File > Save (Ctrl+s)** del menú.
- Para abrir un archivo existente, haga clic en el ícono **Open** de la barra de herramientas o seleccione **File > Open (Ctrl+o)** del menú. Aparecerá la ventana de diálogo **Open Files** y en ella los usuarios pueden buscar y seleccionar el archivo que deseen abrir.

Varios archivos pueden abrirse simultáneamente, cada uno con una sola pestaña debajo de la barra de menús. Las pestañas muestran el nombre de un archivo después de que se guardó la primera vez.

# Copia de texto entre ventanas gráficas

Es posible copiar texto entre documentos, ventanas de texto y ventanas de comandos en el entorno gráfico. El texto seleccionado se duplica usando *copiar y pegar*, o se mueve usando *copiar y pegar*. Ya sea que se corte o que se copie, el texto se conserva en la memoria para pegarse en otra ubicación.

Para seleccionar texto:

- Haga clic y mantenga presionado el botón izquierdo del mouse antes del primer carácter que desea seleccionar.
- Arrastre el mouse sobre todo el texto que desea hasta que quede resaltado como una única selección y, luego, suelte el botón izquierdo. No vuelva a hacer clic en el botón izquierdo porque anulará la selección del texto.

Para pegar la selección, pueden emplearse diversos métodos y obtener el mismo resultado.

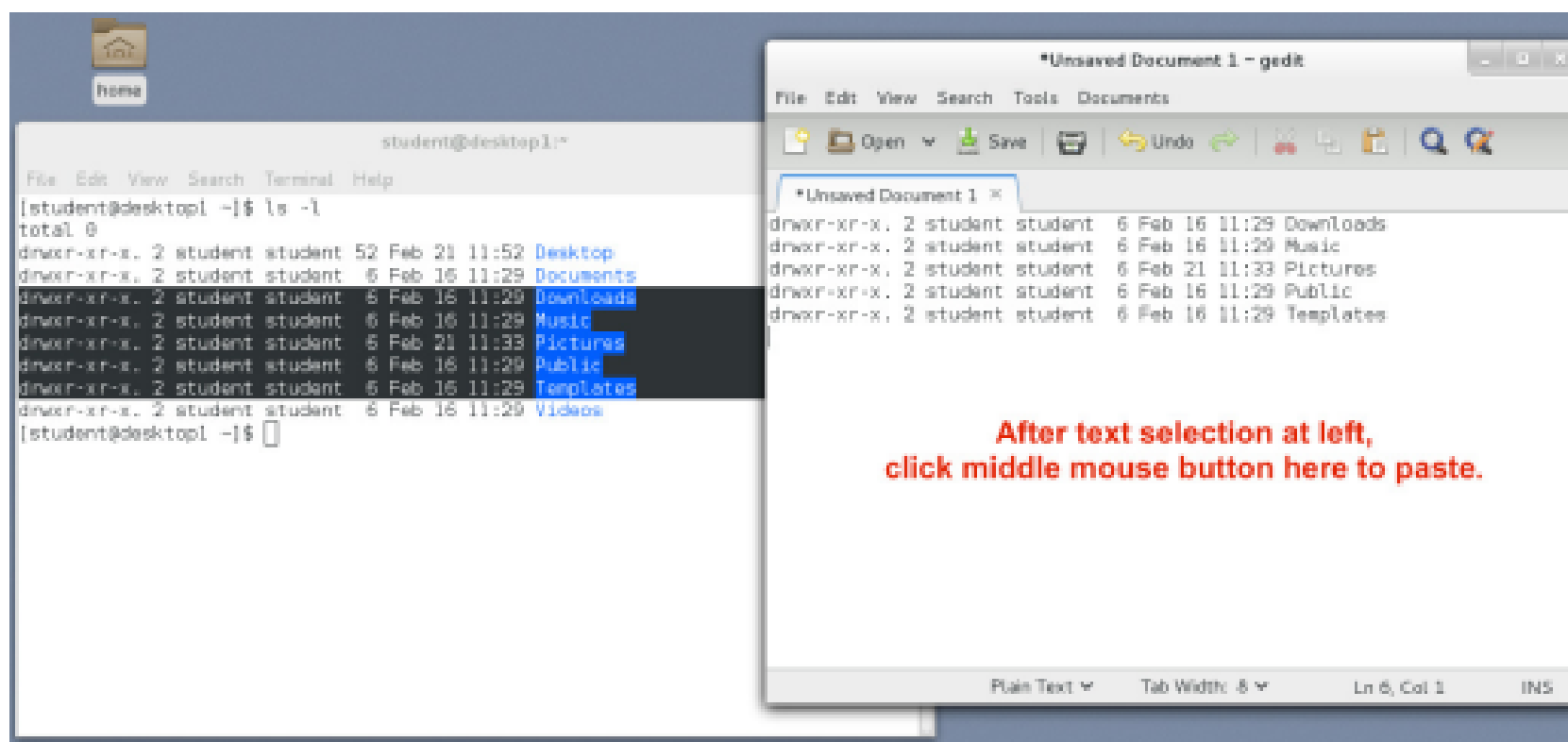
Primer método:

- Haga clic en el botón derecho del mouse en cualquier parte del área del texto que acaba de seleccionar.
- En el menú contextual que obtiene como resultado, seleccione *either cut* o **copy**.
- Desplace el mouse hacia la ventana o el documento donde se colocará el texto, haga clic con el botón izquierdo del mouse para posicionarlo donde debe ir el texto y haga clic con el botón derecho del mouse nuevamente; ahora, seleccione **paste**.

A continuación se incluye una técnica con el mouse más breve para practicar:

- Primero, seleccione el texto.
- Desplace el mouse sobre la ventana de destino y haga clic en el botón central del mouse solo una vez para pegar el texto en la posición del cursor.

Este último método solo permite copiar, no cortar. El texto original permanece seleccionado y puede eliminarse. Al igual que con otros métodos, el texto se conserva en la memoria y puede pegarse reiteradamente.



*Figura 4.12: Selección y pegado de texto con el botón del medio del mouse*

El método de atajo del teclado también puede usarse en aplicaciones gráficas:

- Primero, seleccione el texto.
- Use **Ctrl+x** para cortar o **Ctrl+c** para copiar el texto.
- Haga clic en la ubicación en la que el texto debe colocarse a fin de posicionar el cursor.
- Use **Ctrl+v** para pegar.

## Importante

**Ctrl+c** y **Ctrl+v** no copiarán ni pegarán dentro de una ventana de terminal.  
**Ctrl+c** en realidad finalizará el proceso en ejecución actual dentro de una ventana de terminal. Para copiar y pegar dentro de una ventana de terminal, use **Ctrl+Shift+c** and **Ctrl+Shift+v**.

## Referencias

Página del manual (1)**gedit**

Editor de textos **gedit**

- **yelp help:gedit**

**gedit** Wiki

<https://wiki.gnome.org/Apps/Gedit>

# Práctica: Copiado de texto entre ventanas