1. Editor de textos

Ejercicio 1.a

Nombre al menos 3 editores de texto que puede utilizar desde la línea de comandos.

vi, vim, neovim (nvim), nano y emacs.

Ejercicio 1.b

¿En qué se diferencia un editor de texto de los comandos cat, more o less? Enumere los modos de operación que posee el editor de textos vi.

Los editores de texto permiten editar los archivos de texto. **cat**, **more**, **less** solo permiten visualizarlos.

vi tiene tres modos:

- **insert mode:** este modo se usa para la edición de texto normal. El modo reemplazar es una variación del modo insertar que reemplaza texto en lugar de insertarlo.
- **command mode:** este modo se usa para la exploración de archivos, copiar y pegar, y ejecutar comandos simples. Con este modo, también se realizan funciones como deshacer, rehacer y otras.
- **ex mode:** este modo se usa para guardar, cerrar y abrir archivos, así como también para buscar y reemplazar, y otras operaciones más complejas. Desde este modo, es posible insertar el resultado de programas en el archivo actual, configurar vim y mucho más. Todo lo que es posible usando ex se puede hacer desde este modo.

Red Hat (2017). Creación y edición de archivos de texto con vim. En *Red Hat System Administration II (RH134-RHEL7-es-3-20170803)* (3ra ed., capítulo 3, p. 43). Red Hat.

Ejercicio 1.c

Nombre los comandos más comunes que se le pueden enviar al editor de textos vi.

ex mode:

Comando	Resultado
`:wq`	Guarda y cierra el archivo actual.
`:x`	Guarda el archivo actual si hay cambios sin guardar, y luego lo cierra.
`:w`	Guarda el archivo actual y permanece en el editor.
`:w <filename>`	Guarda el archivo actual bajo un nombre de archivo diferente.
`:q`	Cierra el archivo actual (solo si no hay cambios sin guardar).
`:q!`	Cierra el archivo actual, e ignora los cambios no guardados.

Command mode:

Tecla	Resultado
i	Cambia al modo *insertar* y comienza a insertar *antes* de la posición actual del cursor (insertar).
a	Cambia al modo *insertar* y comienza a insertar *luego* de la posición actual del cursor (anexar).
I	Mueve el cursor hasta el *inicio* de la línea actual y cambia al modo *insertar*.
A	Mueve el cursor hasta el *final* de la línea actual y cambia al modo *insertar*.
R	Cambia al modo *replace*, y comienza en el carácter bajo el cursor. En el modo *replace*, no se inserta texto, sino que cada carácter que ingresa reemplaza a un carácter del documento actual. (**vim** y **vi** también vienen con comandos de reemplazo más potentes; estos se analizan en otra sección.)
o	Abra una nueva línea *debajo* de la actual y cambie inmediatamente al modo *insertar*.
O	Abra una nueva línea *arriba* de la actual y cambie al modo *insertar*.

Red Hat (2017). Creación y edición de archivos de texto con vim. En *Red Hat System Administration II (RH134-RHEL7-es-3-20170803)* (3ra ed., capítulo 3, pp. 48-49). Red Hat.

2. Proceso de arranque *SystemV*

Ejercicio 2.a

Enumere los pasos del proceso de inicio de un sistema GNU/Linux, desde que se prende la PC hasta que se logra obtener el login en el sistema.

1. Se empieza a ejecutar el código del BIOS
2. El BIOS ejecuta el POST
3. El BIOS lee el sector de arranque (MBR)
4. Se carga el gestor de arranque (MBC)
5. El bootloader carga el kernel y el initrd
6. Se monta el initrd como sistema de archivos raíz y se inicializan componentes esenciales (ej.: scheduler)
7. El Kernel ejecuta el proceso init y se desmonta el initrd
8. Se lee el /etc/inittab
9. Se ejecutan los scripts apuntados por el runlevel 1
10. El final del runlevel 1 le indica que vaya al runlevel por defecto
11. Se ejecutan los scripts apuntados por el runlevel por defecto
12. El sistema está listo para usarse

Ejercicio 2.b

Proceso INIT. ¿Quién lo ejecuta? ¿Cuál es su objetivo?

1. Su función es cargar todos los subprocesos necesarios para el correcto funcionamiento del SO
2. El proceso init posee el PID 1 y se encuentra en /sbin/init
3. En SysV se lo configura a través del archivo /etc/inittab
4. No tiene padre y es el padre de todos los procesos (pstree)
5. Es el encargado de montar los filesystems y de hacer disponible los demás dispositivos.

Ejercicio 2.c

Ejecute el comando pstree. ¿Qué es lo que se puede observar a partir de la ejecución de este comando?

> **pstree** muestra los procesos en ejecución en forma de árbol. El árbol tiene su raíz en init o systemd, depende del sistema de inicio que utilice el sistema.

Ejercicio 2.d

RunLevels. ¿Qué son? ¿Cuál es su objetivo?

Es el modo en que arranca Linux (3 en Redhat, 2 en Debian). El proceso de arranque lo dividimos en niveles, cada uno es responsable de levantar (iniciar) o bajar (parar) una serie de servicios.

Ejercicio 2.e

¿A qué hace referencia cada nivel de ejecución según el estándar? ¿Dónde se define qué Runlevel ejecutar al iniciar el sistema operativo? ¿Todas las distribuciones respetan estos estándares?

Existen 7 runlevels, y cada uno permite iniciar un conjunto de procesos al arranque o apagado del sistema. Según el estándar:

0. halt (parada).
1. single user mode (monousuario).
2. multiuser, without NFS (modo multiusuario sin soporte de red).
3. full multiuser mode console (modo multiusuario completo por consola).
4. no se utiliza.
5. X11 (modo multiusuario completo con login gráfico)
6. reboot.

Ejercicio 2.f

Archivo /etc/inittab. ¿Cuál es su finalidad? ¿Qué tipo de información se almacena en el? ¿Cuál es la estructura de la información que en él se almacena?

Los runlevels se encuentran definidos en /etc/inittab con el formato:

```
id:nivelesEjecución:acción:proceso
```

```
$ cat /etc/inittab
id:2:initdefault
si::sysinit:/etc/init.d/rcS
ca::ctrlaltdelt:/sbin/shutdown -t3 -e
```

- **id:** identifica la entrada en inittab (1 a 4 caractéres)
- **nivelesEjecución:** el/los niveles de ejecución en los que se realiza la acción.
- **acción:** describe la acción a realizar
    - **wait:** inicia cuando entra al runlevel e init espera a que termine **initdefault.**
    - **ctraltdel:** se ejecutrá cuando init reciba la señal SIGINT.
    - **off, respawn, once, sysinit, boot, bootwait, powerwait,** etc.
- **proceso:** el proceso exacto que será ejecutado.

Ejercicio 2.g

Suponga que se encuentra en el runlevel <X>. Indique qué comando(s) ejecutaría para cambiar al runlevel <Y>. ¿Este cambio es permanente? ¿Por qué?

Para cambiar de un runlevel a otro se utiliza el comando **init N**, donde *N* es el número de runlevel al que se desea cambiar

```
$ init <Y>
```

Este cambio es temporal y solo afecta al estado actual del sistema. El cambio no es permanente porque el sistema volverá a su runlevel predeterminado en el próximo reinicio. Para hacer un cambio permanente en el runlevel predeterminado, se debe modificar el archivo de configuración /etc/inittab.

Ejercicio 2.h

Scripts RC. ¿Cuál es su finalidad? ¿Dónde se almacenan? Cuando un sistema GNU/Linux arranca o se detiene se ejecutan scripts, indique cómo determina qué script ejecutar ante cada acción. ¿Existe un orden para llamarlos? Justifique.

Cuando init entra en un runlevel, llama al script rc con un argumento numérico especificando el nivel de ejecución al que ir. Entonces, el script rc inicia y detiene servicios en el sistema según sea necesario para llevar el sistema a ese nivel de ejecución.

Todos los scripts RC se encuentran en el directorio **/etc/rc.d/**, que contiene un subdirectorio para cada nivel de ejecución (rc0.d, ..., rc6.d). Dentro de cada uno de estos sbdirectorios hay enlaces simbólicos a los scripts maestros almacenados en /etc/rc.d/init.d/.

Los enlaces simbólicos se nombran con el formato: **`[S|K]<orden><nombreScript>`**.

Los archivos que comienzan con **S** mayúscula representan scripts que se inician al entrar en ese nivel de ejcución, mientras que los archivos que comienzan con una **K** mayúscula representan scripts que se detienen. Los números especifican el orden en que deben ejecutarse los scripts.

Por ejemplo, un demonio puede tener un script llamado *S35daemon* en rc3.d/, y un script llamado *K65daemon* para detenerlo en rc2.d/. Tener los números al principio del nombre del archivo hace que se ordenen, y se procesen, en el orden deseado.

Understanding the rc Scripts in Linux - The Geek Diary

Ejercicio 2.i

¿Qué es insserv? ¿Para qué se utiliza? ¿Qué ventajas provee respecto de un arranque tradicional?

Se utiliza para administrar el orden de los enlaces simbólicos del /etc/rcX.d, resolviendo las dependencias de forma automática. Utiliza cabeceras en los scripts del /etc/init.d/ que permiten especificar la relación con otros scripts rc: LSBInit (Linux Standard Based Init). Es utilizado por update-rc.d para instalar/remover los links simbólicos.

Las dependencias se especifican mediante *facilities* con la palabra clave *Procides.* Las facilities que comienzan con $ se reservan para el sistema. Los scripts deben cumplir LSB init script:

- Proveer al menos start, stop, restart, force-reload y status.
- Retornar un código apropiado.
- Declarar las dependencias.

insserv es una herramienta de bajo nivel utilizada por update-rc.d que habilita un script de init del sistema (`boot script') instalado mediante la lectura de la cabecera de comentarios del script y calcula las dependencias entre todos los scripts, por ejemplo:

```
### BEGIN INIT INFO
# Provides:          my_daemon
# Required-Start:    $syslog $remote_fs
# Required-Stop:     $syslog $remote_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: This is a test daemon
# Description:       This provides an example about how to
#                    write an Init script.
### END INIT INFO
```

No se recomienda ejecutar insserv directamente a menos que sepa exactamente lo que está haciendo, hacerlo puede hacer que su sistema de arranque sea inoperable. update-rc.d es la interfaz recomendada para gestionar los scripts init.

insserv escanea las dependencias del sistema en el archivo de configuración /etc/insserv.conf y cada archivo en el directorio /etc/insserv.conf.d/. Cada línea que comienza con $ y un nombre que le sigue define una facilidad de sistema de acuerdo a la Especificación Base Estándar de Linux (LSB), Todos los nombres seguidos de tal facilidad de sistema declararán las dependencias requeridas de la facilidad.

insserv - Ubuntu manuals

Ejercicio 2.j

¿Cómo maneja upstart el proceso de arranque del sistema?

Upstart surgió como un remplazo para SystemV. Permite la ejecución de trabajos en forma asíncrona a través de eventos (event-based), mientras que SystemV es estrictamente sincrónico (dependency-based).

Ejercicio 2.k

Cite las principales diferencias entre SystemV y Upstart

Upstart surgió como un reemplazo para SystemV. La principal diferencia entre ambos es que upstart permite la ejecución de trabajos de forma asíncrona a través de eventos (event-based) mientras que SystemV es estrictamente sincrónico (dependency-based).

Ejercicio 2.l

Qué reemplaza a los scripts rc de SystemV en Upstart? ¿En que ubicación del filesystem se encuentran?

Upstart surgió como un remplazo para SystemV. Permite la ejecución de trabajos en forma asíncrona a través de eventos (event-based), mientras que SystemV es estrictamente sincrónico (dependency-based).

Estos trabajos se denominan **jobs**, son scripts de texto plano que definen acciones/tareas (unidad de trabajo) a ser ejecutadas por init ante determinados eventos y se definen en /etc/init (.conf). Suelen ser de dos tipos:

- **Task:** ejecución finita
- **Service:** ejecución indeterminada

Los jobs son ejecutados ante eventos (arranque del equipo, inserción de un dispositivo USB, etc.). Cada job posee un objetivo (goal start/stop) y un estado (state) y en base a ellos se ejecuta un proceso específico. AL inicio, init emite el evento *startup*.

Un job puede tener uno o varias tareas ejecutables como parte de su ciclo de vida y siempre debe existir la tarea principal, estas tareas se definen mediante *exec* o *script … end*.

A través de initctl podemos administrar los jobs del demonio de Upstart:

- **start <job>:** cambia el objetivo a start del job especificado
- **stop <job>:** cambia el objetivo a stop del job especificado
- **emit <event>:** event es emitido causando que otros jobs cambien a objetivo start o stop

No más /etc/inittab.

Es importante notar que upstart es 100% compatible con los scripts clásicos de SystemV. De esta manera, aquellos servicios que provean un script para init pueden funcionar sin problemas bajo Upstart.

Ejercicio 2.m

Dado el siguiente job de upstart perteneciente al servicio de base de datos del mysql, indique a qué hace referencia cada línea del mismo:

```
# MySQL Service                         #
description  “MySQL Server”             # 1.
author       “info author”              # 2.
start on     (net-device-up             # 3.
                 and local-filesystems  #
                 and runlevel [2345]    #
stop on      runlevel [016]             # 4.
[...]                                   #
exec /usr/bin/mysqld                    # 5.
[...]                                   #
```

1. Esta línea proporciona una descripción del servicio que se está configurando. En este caso, simplemente indica que este es el servicio del servidor MySQL.
2. Esta línea generalmente indica el nombre del autor o la entidad responsable de la configuración del job. En este ejemplo, se ha proporcionado "info author" como un marcador de posición, pero normalmente contendrá el nombre o el identificador del autor real.
3. Esta línea especifica las condiciones que deben cumplirse para que el servicio se inicie. En este caso, el servicio de MySQL se iniciará cuando se cumplan las siguientes condiciones simultáneamente:
    1. **`net-device-up:`** cuando las interfaces de red están configuradas y funcionando.
    2. **`local-filesystems:`** cuando los sistemas de archivos locales están montados y listos.
    3. **`runlevel [2345]:`** cuando el sistema está en el runlevel 2, 3, 4 o 5.
4. Esta línea específica cuándo se detendrá el servicio de MySQL. En este caso, el servicio se detendrá cuando el sistema entre en el nivel de ejecución 0 (apagado) o 6 (reinicio).
5. Esta línea contiene el comando que Upstart ejecutará para iniciar el servicio de MySQL. En este caso, se utiliza /usr/bin/mysqld para iniciar el servidor MySQL. Esto es lo que realmente inicia el servicio cuando se cumplen las condiciones especificadas en las líneas start on.

Ejercicio 2.n

¿Qué es systemd?

systemd es un sistema que centraliza la administración de daemons y librerias del sistema. El daemon *systemd* reemplaza al proceso init (systemd pasa a tener el PID 1).

systemd mejora el paralelismo de booteo y es compatible con SystemV, si es llamado como init, los runlevels son reemplazados por **targets**, y al igual que con upstart, el archivo /etc/inittab no existe más.

Las unidades de trabajo son denominadas **units** de tipo:

- **Service:** controla un servicio particular *(.service)*.
- **Socket:** encapsula IPC, un sockect del sistema o file system FIFO *(.socket)*
  - → sockect-based activation.
- **Target:** agrupa units o establece puntos de sincronización durante el booteo *(.target)*
  - → dependencia de unidades
- **Snapshot:** almacena el estado de un conjunto de unidades que puede ser restablecido más tarde *(.snapshot)*
- etc.

Estas units pueden tener dos estados: active o inactive.

Ejercicio 2.ñ

¿A qué hace referencia el concepto de activación de socket en systemd?

No todos los servicios que se inician en el booteo se utiliza. La activación por socket es un mecanismo de iniciación bajo demanda, podemos ofrecer una variedad de servicios sin que estén realmente iniciados.

Cuando el socket recibe una conexión, spawnea el servicio y le pasa el socket. No hay necesidad de definir dependencias entre servicios, ya que se inician todos los sockets en primera medida.

Ejercicio 2.o

¿A qué hace referencia el concepto de cgroup?

cgroups permite organizar un grupo de procesos en forma jerárquica. Agrupa un conjunto de procesos relacionados (por ejemplo, un servidor web Apache con sus dependencias).

cgroups realiza las siguientes tareas:

- Tracking mediante el subsistema cgroups: no utiliza el PID.
- Limitar el uso de recursos.
- etc.

3. Usuarios

Ejercicio 3.a

¿Qué archivos son utilizados en un sistema GNU/Linux para guardar la información de los usuarios?

El archivo **/etc/passwd** se utiliza para almacenar información sobre los usuarios locales y para cada usuario hay siete campos separados por dos puntos:

```
username:password:uid:gid:GECOS:/home/dir:shell
```

1. **username** es el nombre del usuario (login).
2. **password** es donde se guardaban las contraseñas en formato cifrado tradicionalmente. Actualmente, se guardan cifradas en un archivo aparte con el nombre **/etc/shadow** (esto se indica colocando una x en el campo)**.**
3. **UID** es un ID único de usuario, un número que identifica al usuario en el nivel más básico de forma unívoca.
4. **GID** es el número de ID de grupo principal del usuario.
5. **GECOS** es un texto arbitrario que, por lo general, incluye el nombre real del usuario y otra información adicional (mail, teléfono, etc.).
6. **/home/dir** es la ubicación donde se encuentran los datos personales del usuario y los archivos de configuración.
7. **shell** es la shell por defecto de los procesos y usuarios. La shell **/sbin/nologin** se utiliza para bloquear el inicio de sesión en el sistema de forma interactiva y es muy común utilizarla en cuentas de usuarios que representan procesos o servicios en lugar de usuarios humanos.

Red Hat (2017). Usuario y Grupos. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 5, pp. 126-127). Red Hat.

Ejercicio 3.b

¿A qué hacen referencia las siglas UID y GID? ¿Pueden coexistir UIDs iguales en un sistema GNU/Linux? Justifique.

- **UID (User ID):** es un identificador único de usuario. Cada usuario tiene su propio UID, que es un número que lo identifica de manera unívoca. Por convención, muchas distribuciones de GNU/Linux asignan por defecto el UID 1000 al primer usuario creado en el sistema y luego asignan a los usuarios nuevos el primer número de

UID disponible en el rango, a partir de la UID 1000 en adelante (a menos que se especifique uno explícitamente).

- **GID (Group ID):** es un idenficiador único de grupo. Cada grupo de usuarios tiene su propio GID, para identificarlo de manera unívoca.
    - Los grupos locales están definidos en **/etc/group.**
    - Cada usuario tiene exactamente un **grupo principal** y pueden ser miembros de ninguno o más **grupos adicionales.**
    - Para los usuarios locales, el grupo principal está definido por el número de GID del grupo indicado en el cuarto campo de /etc/passwd.
    - Generalmente, el grupo principal es propietario de los nuevos archivos creados por el usuario.
    - Normalmente, el grupo principal de un usuario creado recientemente es un grupo creado con el mismo nombre que el del usuario. El usuario es el único miembro de este grupo privado de usuarios (UPG).

Red Hat (2017). Usuario y Grupos. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 5, pp. 126-127). Red Hat.

Ejercicio 3.c

¿Qué es el usuario root? ¿Puede existir más de un usuario con este perfil en GNU/Linux? ¿Cuál es la UID del root?

```
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.
```

root es un superusuario, un usuario que tiene todo el poder sobre el sistema. Este usuario tiene el poder de anular los privilegios normales del sistema de archivos y se utiliza para manejar y administrar el sistema. UID 0 siempre se asigna a la cuenta de root.

Es necesario contar con privilegios de superusuario (root) para poder realizar tareas, como la instalación o eliminación de software, y para administrar los directorios y los archivos del sistema. La mayoría de los dispositivos solo pueden ser controlados por el usuario root, pero existen algunas excepciones (por ejemplo, los dispositivos desmontables, como los dispositivos USB). Sin embargo, este privilegio ilimitado viene acompañado de una responsabilidad. El usuario root tiene poder ilimitado para dañar el sistema: eliminar

archivos y directorios, eliminar cuentas de usuarios, agregar puertas traseras, etc. Si la cuenta **root** está comprometida, alguien más tendrá control administrativo del sistema.

La cuenta root en Linux es casi equivalente a la cuenta de administrador local en Windows. En Linux, la mayoría de los administradores del sistema inicia sesión en una cuenta de usuario sin privilegios y utiliza distintas herramientas (como su o sudo) para obtener privilegios de usuario root temporalmente.

Red Hat (2017). Obtención de acceso de superusuario. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 5, pp. 131-132). Red Hat.

Ejercicio 3.d

Agregue un nuevo usuario llamado iso2017 a su instalación de GNU/Linux, especifique que su home sea creada en /home/iso_2017, y hágalo miembro del grupo catedra (si no existe, deberá crearlo). Luego, sin iniciar sesión como este usuario cree un archivo en su home personal que le pertenezca. Luego de todo esto, borre el usuario y verifique que no queden registros de él en los archivos de información de los usuarios y grupos.

```
1.  # groupadd catedra
2.  # useradd -m -d /home/iso_2017 -G catedra -s /bin/bash iso2017
3.  # passwd iso2017
4.  # su -c “touch ~/archivo” iso2017
5.  # userdel -r iso2017
6.  # find / -nouser -o -nogroup 2> /dev/null
7.  # tail -n3 /etc/passwd /etc/group
```

1. Crea el grupo catedra
2. Crea el usuario iso2017
    a. **-m** crea el home /home/iso_2017 si no existe
    b. **-d /home/iso_2017** especifica el home del usuario
    c. -**G catedra** agrega al usuario al grupo complementario catedra (su grupo principal es iso2017)
    d. **-s /bin/bash** establece la terminal por defecto.
3. Crea una contraseña para el usuario iso2017, si no se ejecuta este comando el usuario no puede iniciar sesión.
4. Ejecuta el comando ***touch ~/archivo*** como el usuario ***iso2017***
    a. **~** se podría reemplazar por el path absoluto (/home/iso_2017) pero no por la variable de entorno ***$HOME*** porque al usar ***-c*** quedan las variables de entorno del contexto del usuario actual (desde el que se ejecuta su) (no sé por qué es así) (raaaro).
5. Elimina al usuario iso2017 y su home folder.

6. Lista todos los archivos y directorios que “no pertenecen a nadie”.
    a. Cuando se elimina a un usuario con **userdel** sin la opción **-r** especificada o que tiene archivos fuera de su home, entonces el sistema tendrá archivos que pertenecen a un UID no asignado. Esta situación puede hacer que se filtre información y causar otros problemas de seguridad.

       En la mayoría de las distribuciones Linux, el comando useradd asigna a los usuarios nuevos el primer número de UID disponible en el rango, a partir de la UID 1000 en adelante (a menos que se especifique uno explícitamente con la opción -u UID). Es así como puede filtrarse información: si el primer número UID disponible ha sido asignado previamente a una cuenta de usuario que ha sido eliminada del sistema, el número de UID del usuario anterior se reasignará al nuevo usuario y le dará la propiedad de los archivos restantes del usuario anterior. A continuación se demuestra esta situación:

```
[root@serverX ~]# useradd prince
[root@serverX ~]# ls -l /home
drwx------. 3 prince prince   74 Feb 4 15:22 prince
[root@serverX ~]# userdel prince
[root@serverX ~]# ls -l /home
drwx------. 3 1000   1000     74 Feb 4 15:22 prince
[root@serverX ~]# useradd bob
[root@serverX ~]# ls -l /home
drwx------. 3 bob    bob      74 Feb 4 15:23 bob
drwx------. 3 bob    bob      74 Feb 4 15:22 prince
```

7. Muestra las últimas 3 líneas de los archivos, con eso podemos ver que el usuario iso2017 y su grupo por defecto (iso2017) ya no existen, pero el grupo catedra si.

Ejercicio 3.e

Investigue la funcionalidad y parámetros de los siguientes comandos:

1. **useradd ó adduser**
2. **usermod**
3. **userdel**
4. **su**
5. **groupadd**
6. **who**
7. **groupdel**
8. **passwd**

NAME

useradd - create a new user or update default new user information

SYNOPSIS

```
useradd [options] LOGIN
useradd -D
useradd -D [options]
```

DESCRIPTION

useradd is a low level utility for adding users. On Debian, administrators should usually use adduser(8) instead.

When invoked without the -D option, the useradd command creates a new user account using the values specified on the command line plus the default values from the system.

Depending on command line options, the useradd command will update system files and may also create the new user's home directory and copy initial files.

OPTIONS

-c,--comment *COMMENT*	Aany text string. It’s generally a short description of the account, and is currently used as the field for the user's full name.

4. Filesystem

Ejercicio 4.a

¿Cómo están definidos los permisos sobre archivos en un sistema GNU/Linux?

Los archivos tienen solo tres categorías de usuario a las que se le aplican permisos. El archivo pertenece a un **usuario**, que generalmente es quien creó el archivo. El archivo también pertenece a un solo **grupo**, generalmente el grupo primario del usuario que creó el archivo, pero esto se puede cambiar. Se pueden establecer diferentes permisos para el usuario propietario y el grupo propietario, así como para todos los **otros** usuarios en el sistema que no sean el usuario o un miembro del grupo propietario. Se aplicarán los permisos más específicos. Por lo tanto, los permisos de usuario anulan los permisos de grupo, que anulan otros permisos.

También existen tres categorías de permisos que se aplican: leer, escribir y ejecutar. Estos permisos afectan el acceso a archivos y directorios de la siguiente forma:

Permiso	Efecto en los archivos	Efecto en los directorios
r (read)	Pueden leerse los contenidos del archivo.	Los contenidos del directorio pueden detallarse.
w (write)	Los contenidos del archivo pueden cambiarse.	Cualquier archivo en el directorio puede crearse o eliminarse.
x (exec)	Los archivos pueden ejecutarse como comandos.	Es posible acceder al contenido del directorio.

Tenga en cuenta que los usuarios normalmente poseen privilegios tanto de ***read*** como de ***exec*** en los directorios de solo lectura, por lo que pueden listar el directorio y acceder a su contenido. Si un usuario solo posee acceso de ***read*** en un directorio, los nombres de los archivos dentro de este pueden detallarse, pero no estará disponible otra información, incluidos permisos o marcas de tiempo, y tampoco se podrá acceder a ellos. Si un usuario solo posee acceso ***exec*** en un directorio, no podrá enumerar los nombres de los archivos en este, pero sí podrá acceder a su contenido en caso de que ya conozca el nombre de un archivo del que posee permiso para leer. Así, podrá acceder al contenido del archivo especificando su nombre.

Todo usuario que cuente con permisos de escritura para el directorio donde se encuentra el archivo puede quitar un archivo, sin importar la propiedad ni los permisos del archivo en sí. Esto se puede anular con un permiso especial, el sticky bit.

Red Hat (2017). Permisos del sistema de archivos Linux. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 6, pp. 158-159). Red Hat.

Ejercicio 4.c

Al utilizar el comando chmod generalmente se utiliza una notación octal asociada para definir permisos. ¿Qué significa esto? ¿A qué hace referencia cada valor?

Método simbólico

```
chmod «who»«what»«which» {file|directory}
```

- **who: u** *(usuario)*, **g** *(grupo)*, **o** *(otro)*, **a** *(todos (ugo))*
- **what: +** *(agregar)*, **–** *(eliminar)*, **=** *(exactamente)*
- **which: r** *(leer)*, **w** *(escribir)*, **x** *(ejecutar)*

El uso de una **X** mayúscula como indicador de permiso agregará permiso de ejecución únicamente si el archivo es un directorio o si ya tiene el permiso de ejecución establecido para usuario, grupo u otros.

Método numérico

```
chmod ### {file|directory}
```

- Cada dígito representa un nivel de acceso: usuario, grupo, otros.
- # es la suma de r = 4, w = 2 y x = 1.

Al utilizar el método numérico, los permisos son representados por un número octal de tres dígitos (o cuatro, al establecer permisos avanzados). Un único dígito octal puede representar los números 0-7, exactamente la cantidad de posibilidades para un número de tres bits.

Red Hat (2017). Cambio de permisos de archivo o directorio. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 6, pp. 164-165). Red Hat.

Ejercicio 4.d

¿Existe la posibilidad de que algún usuario del sistema pueda acceder a determinado archivo para el cual no posee permisos? Nómbrelo y realice las pruebas correspondientes.

El superusuario (**root**) tiene privilegios elevados y puede acceder a cualquier archivo o directorio en el sistema, independientemente de sus permisos.

Ejercicio 4.e

Explique los conceptos de "full path name" y "relative path name". De ejemplos claros de cada uno de ellos.

Rutas absolutas (full path name)

Una ruta absoluta es un nombre completamente calificado que comienza en el directorio (/) raíz y especifica cada subdirectorio que se atraviesa para llegar y que representa en forma exclusiva un solo archivo. Cada archivo del sistema de archivos tiene un único nombre de ruta absoluta, reconocido con una regla simple: un nombre de archivo con una barra (/) como primer carácter es el nombre de la ruta absoluta. Por ejemplo, el nombre de ruta absoluta para el archivo de registro de mensajes del sistema es /var/log/messages.

Rutas relativas (relative path name)

Al igual que una ruta absoluta, una ruta relativa identifica un archivo único y especifica solo la ruta necesaria para llegar al archivo desde el directorio de trabajo. Para reconocer nombres de ruta relativos, se sigue una regla simple: un nombre de ruta que no tenga otro carácter más que una barra (/) como primer carácter es un nombre de ruta relativo. Un usuario en el directorio /var podría referirse en forma relativa al archivo de registro del mensaje como log/messages.

./ representa al directorio de trabajo y **../** representa al directorio padre del directorio de trabajo actual. Por ejemplo, un usuario en el directorio /home/user/Documets/Foo/ puede hacer referencia al archivo /home/user/file.txt con la ruta relativa ./../../file.txt.

Red Hat (2017). Rutas Absolutas y relativas. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 2, pp. 36-37). Red Hat.

Ejercicio 4.f

¿Con qué comando puede determinar en qué directorio se encuentra actualmente? ¿Existe alguna forma de ingresar a su directorio personal sin necesidad de escribir todo el path completo? ¿Podría utilizar la misma idea para acceder a otros directorios? ¿Cómo? Explique con un ejemplo.

El comando **pwd** (print working directory) imprime la ruta absoluta del directorio de trabajo actual. Para ingresar al directorio personal se puede utilizar **cd** sin ninguna ruta:

```
iso@debian12-vm:~/Documents/Foo$ pwd
/home/iso/Documents/Foo
iso@debian12-vm:~/Documents/Foo$ cd
```

```
iso@debian12-vm:~$ pwd
/home/iso
```

Además, se puede hacer referencia al directorio personal sin necesidad de escribir todo el path completo utilizando la virgulilla (**~**)**.** Por ejemplo: ~/Documents es lo mismo que /home/user/Documents.

La virgulilla seguida del nombre de un usuario hace referencia al directorio personal de ese usuario. Por ejemplo, ~otroUsuario/Downloads hace referencia a /home/otroUsuario/Downloads (suponiendo que su directorio personal sea /home/otroUsuaro, la gracia de la virgulilla es que el directorio personal de otroUsuario es distinto entonces va a hacer referencia al directorio correspondiente).

Ejercicio 4.g

Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el uso del FileSystem:

cd	Cambiar el directorio de trabajo actual.	
umount	Desmontar sistemas de archivos o dispositivos.	
mkdir	Crear un directorio, si no existe.	
	-m, --mode=*MODE*	Establece los permisos del directorio MODE es un octal.
	-p, --parents	Crea directorios padres según sea necesario, con sus modos de archivo no afectados por ninguna opción -m.
	-v, --verbose	Imprime cada directorio creado.
du	Muestra el uso del espacio en disco por archivos y directorios.	
	-a, --all	Muestra todos los archivos, no solo directorios.
	-d, --max-depth=*N*	Imprime los valores hasta *N* niveles.
	-h, --human-readable	Imprime los tamaños en formato legible (1K, 234M, 2G).
	-c, --total	Imprime el peso total
	`$ du -a -d -h 1 dir | sort -h`	

	Este comando muestra el peso de todos los archivos y directorios en *dir* ordenados de forma ascendente. Sin el pipe a **sort** los muestra ordenados por nombre.	
rmdir	Elimina un directorio, si está vacío.	
	-p, --parents	Elimina el directorio y sus ancestros. “rmdir -p a/b/c” es equivalente a “rmdir a/b/c a/b a”.
	-v, --verbose	Muestra un diagnóstico por cada directorio procesado
df	Muestra información sobre el espacio disponible y utilizado en los sistemas de archivos montados.	
	-h, --human-readable	Imprime los tamaños en formato legible (1K, 234M, 2G).
mount	Montar sistemas de archivos en un directorio específico	
ln	Crea enlaces duros (por defecto) o simbólicos a archivos y directorios.	
	-s, --symbolic	Crea enlaces simbólicos **`ln -s file link`**
ls	Lista los archivos y directorios de un directorio	
	-a, --all	No ignora las entradas que empiezan con “.”
	-A, --almost-all	Como -a pero no lista ./ y ../
	-d, --directory	Lista el directorio en sí, no su contenido
	-h, --human-readable	Imprime los tamaños en formato legible.
	-l	Utilizar un formato de listado largo
	-R, --recursive	Lista los subdirectorios recursivamente
pwd	Muestra la ruta completa del directorio de trabajo actual	
cp	Copia archivos y directorios de una ubicación a otra.	
	-f, --force	Si no se puede abrir un archivo de destino existente, elimínelo e inténtelo de nuevo.
	-i, --interactive	Preguntar antes de sobrescribir.

	-p	Igual a `--preserve=mode,ownership,timestamps`.
	-R, -r, --recursive	Copia directorios recursivamente.
	-v, --verbose	Explicar lo que se está haciendo.

mv	Mueve o renombra archivos y directorios.	
	-f, --force	No preguntar antes de sobrescribir
	-i, --interactive	Preguntar antes de sobrescribir.
	-v, --verbose	Explicar lo que se está haciendo.

5. Procesos

Ejercicio 5.a

¿Qué es un proceso? ¿A que hacen referencia las siglas PID y PPID? ¿Todos los procesos tienen estos atributos en GNU/Linux? Justifique. Indique qué otros atributos tiene un proceso.

Un **proceso** es un programa que se está ejecutando.

El **PID** es la identificación del proceso y el **PPID** es la identificación de un proceso padre. En Linux todos los procesos tienen un PPID excepto por el proceso systemd o init, que son el primer proceso en ejecutarse y es de donde que derivan todos los demás procesos.

Además del PID y el PPID los procesos tienen otros atributos cómo:

- **Estado:** en ejecución, en espera, suspendido, zombie.
- **Espacio de memoria:** cada proceso tiene su propio espacio de memoria virtual.
- **Información sobre la propiedad del proceso:** usuario y grupo propietario del proceso.
- **Prioridad de planificación:** Indica la prioridad del proceso en relación con otros procesos.

Ejercicio 5.b

Indique qué comandos se podrían utilizar para ver qué procesos están en ejecución en un sistema GNU/Linux.

El comando **ps** se utiliza para elaborar una lista de los procesos actuales. El comando puede proporcionar información detallada de los procesos, que incluye:

- La identificación del usuario (UID) que determina los privilegios del proceso.
- La identificación del proceso (PID) única.
- La CPU y el tiempo real empleado.
- La cantidad de memoria que el proceso ha asignado en diversas ubicaciones.
- La ubicación del proceso STDOUT, conocido como terminal de control.
- El estado del proceso actual.

De manera predeterminada, el comando **ps sin opciones** selecciona todos los procesos que tienen la misma identificación de usuario efectivo (EUID) que el usuario actual y que están asociados con la misma terminal en la que se invocó ps. Una lista de visualización común es **ps aux** (no es lo mismo que ps -aux) que muestra todos los procesos, con columnas que serán de interés para los usuarios, e incluye procesos sin un terminal de control.

Red Hat (2017). Listas de Procesos. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 9, pp. 184-185). Red Hat.

Ejercicio 5.c

¿Qué significa que un proceso se está ejecutando en Background? ¿Y en Foreground?

Un **trabajo** está asociado con cada tubería ingresada en un aviso de shell. Todos los procesos en esa tubería son parte del trabajo y son miembros del mismo grupo de procesos. Si se ingresa solo un comando en un aviso de shell, puede considerarse como una "tubería" mínima de un comando, donde ese comando sería el único miembro de ese trabajo.

Solo un trabajo puede leer entradas y señales generadas por el teclado desde una ventana de terminal específica por vez. Los procesos que sean parte de ese trabajo son **procesos en primer plano (foreground)** de ese terminal de control.

Un **proceso en segundo plano (background)** de dicho terminal de control es un miembro de cualquier otro trabajo asociado con ese terminal. Los procesos en segundo plano de un terminal no pueden leer entradas ni recibir interrupciones generadas por el teclado desde el terminal, pero pueden escribir en el terminal. Un trabajo en segundo plano puede detenerse (suspenderse) o puede estar ejecutándose. Si un trabajo que se está ejecutando en segundo plano intenta leer desde el terminal, se suspenderá automáticamente.

Cada terminal es su propia sesión y puede tener un proceso en primer plano y procesos en segundo plano independientes. Un trabajo es parte de exactamente una sesión, la que pertenece a su terminal de control.

El comando **ps** mostrará el nombre del dispositivo del terminal de control de un proceso en la columna TTY. Algunos procesos, como demonios del sistema, son iniciados por el sistema y no desde un aviso de shell. Estos procesos no tienen un terminal de control, no son miembros de un trabajo y no pueden colocarse en primer plano. El comando ps mostrará un signo de pregunta (?) en la columna TTY para estos procesos.

Red Hat (2017). Control de trabajos. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 7, p. 189). Red Hat.

Ejercicio 5.d

¿Cómo puedo hacer para ejecutar un proceso en Background? ¿Cómo puedo hacer para pasar un proceso de background a foreground y viceversa?

Cualquier comando o tubería puede iniciarse en segundo plano si se anexa el signo ampersand (**&**) al final de la línea de comandos. La shell bash muestra un número de trabajo (exclusivo de la sesión) y el identificador de proceso del proceso secundario nuevo. La shell no espera al proceso secundario y vuelve a mostrar el aviso de shell.

```
[student@serverX ~]$ sleep 10000 &
[1] 5947
[student@serverX ~]$
```

La shell bash realiza un seguimiento de trabajos, por sesión, en una tabla que se muestra con el comando **jobs**

```
[student@serverX ~]$ jobs
[1]+ Running                 sleep 10000 &
[student@serverX ~]$
```

Un trabajo en segundo plano se puede colocar en primer plano con el comando **fg** con su ID de trabajo (%número de trabajo).

```
[student@serverX ~]$ fg %1
sleep 10000
...
```

En el ejemplo anterior, el comando sleep se está ejecutando en primer plano en el terminal de control. La shell se encuentra nuevamente en espera de que se retire el proceso secundario.

Para enviar un proceso en primer plano a segundo plano, presione primero la **solicitud de suspensión** generada por el teclado **(Ctrl+z)** en el terminal. El trabajo se colocará inmediatamente en segundo plano y se suspenderá:

```
sleep 10000
^Z
[1]+ Stopped                 sleep 10000
[student@serverX~]$
```

El comando **ps j** mostrará información relacionada con los trabajos. El PGID es el identificador de proceso del líder del grupo de procesos, generalmente el primer proceso en la canalización del trabajo. El SID es el identificador de proceso del líder de sesión, que para un trabajo es generalmente la shell interactiva que se está ejecutando en su terminal

de control. Dado que el comando sleep de ejemplo está suspendido actualmente, su estado de proceso es T.

```
[student@serverX ~]$ ps j
PPID   PID  PGID   SID  TTY     TPGID  STAT  UID  TIME  COMMAND
2764  2768  2768  2768  pts/0    6377  Ss   1000  0:00  /bin/bash
2768  5947  5947  2768  pts/0    6377  T    1000  0:00  sleep 10000
2768  6377  6377  2768  pts/0    6377  R+   1000  0:00  ps j
```

Para iniciar el proceso suspendido que se está ejecutando en segundo plano, utilice el comando **bg** con la misma ID de trabajo.

```
[student@serverX ~]$ bg %1
[1]+ sleep 10000 &
[student@serverX ~]$
```

La shell emitirá una advertencia al usuario que intente salir de una ventana de terminal (sesión) con trabajos suspendidos. Si el usuario vuelve a intentar salir de inmediato, los trabajos suspendidos se anulan.

Red Hat (2017). Realización de trabajos en segundo plano. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 7, pp. 189-191). Red Hat.

Ejercicio 5.e y Ejercicio 5.f

e. Pipe (|). ¿Cuál es su finalidad? Cite ejemplos de su utilización.
f. Redirección. ¿Qué tipo de redirecciones existen? ¿Cuál es su finalidad? Cite ejemplos de utilización.

Entrada estándar, salida estándar y error estándar

Un proceso puede necesitar 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.

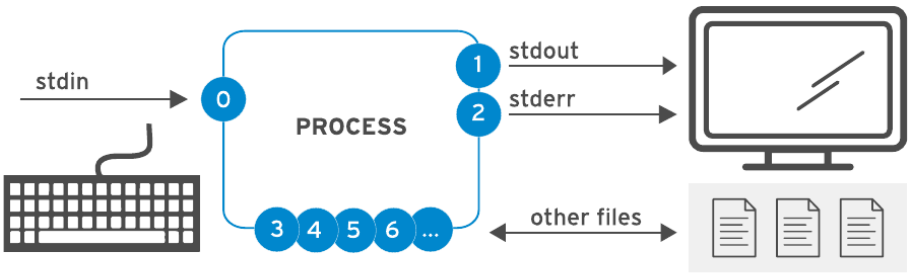


Figura 4.1: Canales de E/S de un proceso (descriptores de archivo)

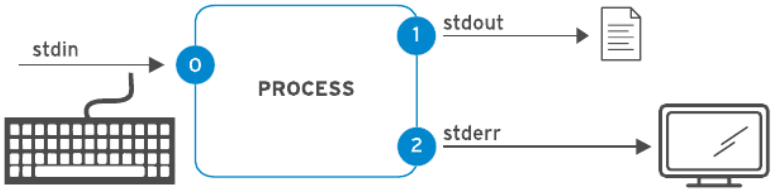
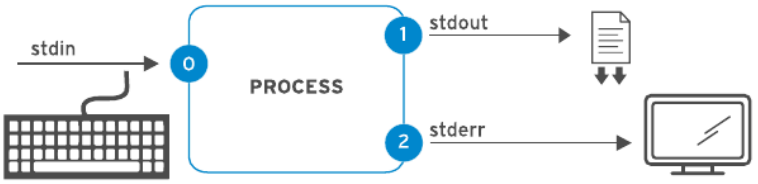
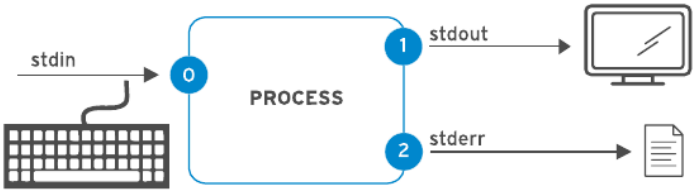
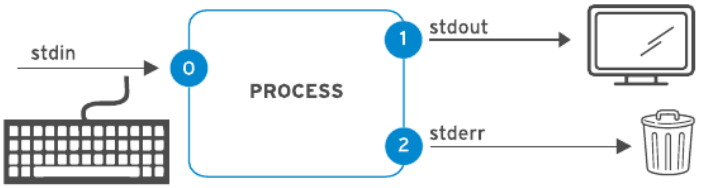
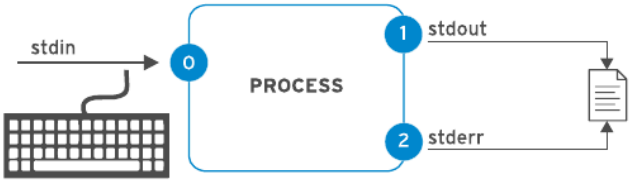
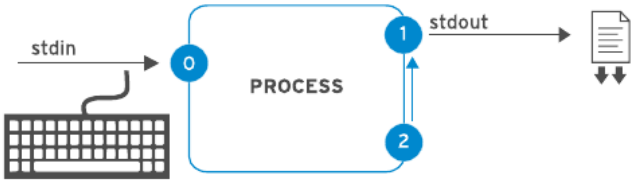
#	Nombre	Descripción	Conexión predeterminada	Uso
0	**stdin**	Entrada estándar	Teclado	Solo lectura
1	**stdout**	Salida estándar	Terminal	Solo escritura
2	**stderr**	Error estándar	Terminal	Solo escritura
3+	**filename**	Otros archivos	Ninguno	Lectura y escritura

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
>file	redirigir **stdout** para sobrescribir un archivo	
>>file	redirigir **stdout** para agregar a un archivo	
2>file	redirigir **stderr** para sobrescribir un archivo	
2>/dev/null	descartar mensajes de error **stderr** mediante el redireccionamiento a **/dev/null**	
>file 2>&1 *&>file*	redirigir **stdout** y **stderr** para sobrescribir el mismo archivo	
>>file 2>&1 *&>>file*	redirigir **stdout** y **stderr** para agregar al mismo archivo	

Ejemplos de redireccionamiento de salidas

```
# Guarde un sello de fecha y hora para su posterior consulta
date > /tmp/saved-timestamp
```

```
# Concatene cuatro archivos en uno
cat file1 file2 file3 file4 > /tmp/all-four-in-one

# Adjunte salida a un archivo existente
echo "new line of information" >> /tmp/file

# Omitir y descartar mensajes de error
find / -name foo 2> /dev/null

# Guarde la salida de un proceso y mensajes de error en archivos
# separados
find /etc -name passwd > /tmp/output 2> /tmp/errors

# Almacene la salida y errores generados en forma conjunta
find /etc -name passwd &> /tmp/save-both

# Adjunte la salida y errores generados en un archivo existente
find /etc -name passwd >> /tmp/save-both 2>&1
```

Construcción de una 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.

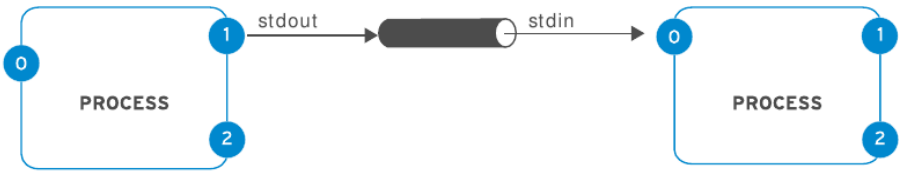


Figura 4.8: Tubería de E/S de proceso

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 archivos 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 redireccionamiento de salidas

```
# Toma la salida del comando ls y usa less para mostrarla en el terminal
# de a una pantalla por vez
ls -l /usr/bin | less

# Contar la cantidad de archivos en un directorio
ls -A | 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
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, redirige 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:

```
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.

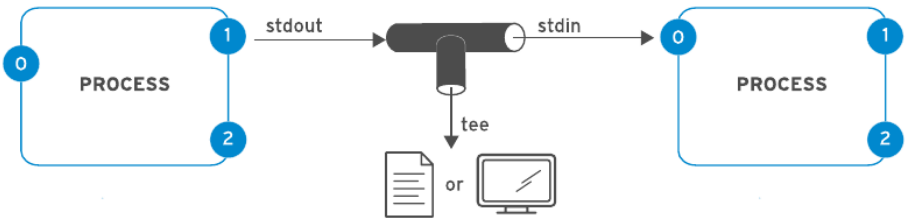


Figura 4.9: Tubería de E/S de proceso con tee

Ejemplos de tuberías que usan el comando tee

```
# Este ejemplo redirigirá la salida del comando ls al archivo y le
# pasará a less para que se muestre en el terminal de a una pantalla
# por vez.
ls -l | tee /tmp/saved-output | less

# Mostrar por pantalla la salida del comando ls y a la vez pasar esa
# salida a wc -l para que cuente la cantidad de líneas
ls | tee /dev/tty | wc -l

# 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.
ls -t | head -n 10 | tee /tmp/ten-last-changed-files
```

Red Hat (2017). Redireccionamiento de la salida a un archivo. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 4, pp. 96-102). Red Hat.

Ejercicio 5.g

Comando *kill* ¿Cuál es su funcionalidad? Cite ejemplos

El comando **kill** envía una señal a un proceso mediante una ID. A pesar de su nombre, el comando kill puede usarse para enviar cualquier señal y no solo aquellas para finalizar programas.

#	Nombre abreviado	Definición	Propósito
1	SIGHUP	Colgar	Se usa para informar la finalización del proceso de control de un terminal. Además, se utiliza para solicitar que se reinicie el proceso (volver a cargar la configuración) sin finalización.
2	SIGINT	Interrupción del teclado	Provoca la finalización del programa. Puede bloquearse o manipularse. Enviado al presionar una combinación de teclas INTR (Ctrl+c).
3	SIGQUIT	Salida del teclado	Es similar a SIGINT, pero también provoca el volcado de un proceso en la finalización. Enviado al presionar una combinación de teclas QUIT (Ctrl+\).
9	SIGKILL	Finalización, no se puede bloquear.	Provoca la finalización abrupta del programa. No se puede bloquear, ignorar ni manipular; siempre es grave.

15	SIGTERM	Termina (default)	Provoca la finalización del programa. A diferencia de SIGKILL, puede bloquearse, ignorarse o manipularse. Es la manera correcta de solicitar la finalización de un programa; hace posible la autolimpieza.
18	SIGCONT	Continuar	Se envía a un proceso para que se reinicie, en caso de estar detenido. No puede bloquearse. Aun si se manipula, siempre reinicia el proceso.

Ejemplos

```
[student@serverX ~]$ kill PID
[student@serverX ~]$ kill -signal PID
[student@serverX ~]$ kill -l
 1) SIGHUP     2) SIGINT    3) SIGQUIT   4) SIGILL    5) SIGTRAP
 6) SIGABRT    7) SIGBUS    8) SIGFPE    9) SIGKILL  10) SIGUSR1
11) SIGSEGV   12) SIGUSR2  13) SIGPIPE  14) SIGALRM  15) SIGTERM
16) SIGSTKFLT 17) SIGCHLD  18) SIGCONT  19) SIGSTOP  20) SIGTSTP
-- output truncated --
```

Red Hat (2017). Control de procesos con señales. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 7, pp. 195-197). Red Hat.

Ejercicio 5.h

Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el manejo de procesos en GNU/Linux. Además, compárelos entre ellos:

- **ps**
- **kill**
- **pstree**
- **killall**
- **top**
- **nice**

6. Otros comandos de Linux

Ejercicio 6.a, 6.b, 6.c, 6.d

TL;DR

Opción	Significado
	Acciones
c	Crear un archivo nuevo
x	Extraer de un archivo
t	Enumerar el contenido de un archivo
	Opciones
v	Mostrar cuáles son los archivos que se archivan/extraen
p	Conservar los permisos de los archivos cuando se extrae un archivo
f	Nombre del archivo; esta opción tiene que estar seguida del fichero que se usará o creará
	Métodos de compresión
z	Usa la compresión **gzip (.tar.gz)**
j	Usa la compresión **bzip2 (.tar.bz2)**. Generalmente, bzip2 logra una mejor relación de compresión que gzip
J	Usa la compresión **xz (.tar.xz)**. Generalmente, xz logra una mejor relación de compresión que bzip2

Empaquetar un archivo

```
$ tar  cf   dest.tar      files...  # Sin comprimir
$ tar  czf  dest.tar.gz   files...  # gzip
$ tar  cjf  dest.tar.bz2  files...  # bzip2
$ tar  cJf  dest.tar.xz   files...  # xz
```

Listar un archivo

```
$ tar  tf   data.tar
$ tar  tzf  data.tar.gz
$ tar  tjf  data.tar.bz2
$ tar  tJf  data.tar.xz

# No es necesario indicar el tipo de archivo:
$ tar  tf   data.tar.gz

# Si se especifica el tipo y no es el correcto falla:
$ tar  tzf  data.tar.bz2
```

Extraer un archivo

```
$ tar  xf   data.tar
$ tar  xzf  data.tar.gz
$ tar  xjf  data.tar.bz2
$ tar  xJf  data.tar.xz

# No es necesario indicar el tipo de archivo:
$ tar  xf   data.tar.gz

# Si se especifica el tipo y no es el correcto falla:
$ tar  xzf  data.tar.bz2
```

Empaquetar y desempaquetar archivos

- **Empaquetar archivos:** Agrupa varios archivos y directorios en un solo archivo o paquete sin necesariamente reducir su tamaño. Se utiliza para mantener la estructura de directorios y facilitar la distribución y organización de archivos.
- **Comprimir archivos:** Reduce el tamaño de uno o varios archivos mediante algoritmos de compresión. El objetivo principal es ahorrar espacio de almacenamiento y/o ancho de banda durante la transferencia de archivos.

Con el comando **tar**, los usuarios pueden empaquetar grandes conjuntos de ficheros en un solo fichero (archivo). El archivo puede comprimirse con gzip, bzip2 o xz. El comando tar también puede enumerar el contenido de los archivos o extraer sus ficheros a su sistema actual.

> **NOTA:** tar no es un comando de compresión en sí, sino que llama a programas que se encargan de comprimir los archivos (gzip, bzip2 o xz).

Uso del comando tar

Para usar el comando **tar** es necesario realizar una de las tres acciones que se indican a continuación:

- **c** → crear un archivo
- **t** → enumerar el contenido de un archivo
- **x** → extraer un archivo

Las opciones más usadas son:

- **f <*file name*>** → nombre del fichero del archivo en el queso trabajará
- ***v*** → mostrar qué está haciendo
- **p** → conservar los permisos de los ficheros y archivos al extraer.

Archivar ficheros y directorios con tar

Antes de crear un archivo tar, verifique que no haya otro archivo en el directorio con el mismo nombre del archivo que se creará. El comando tar sobrescribirá el archivo existente sin ningún comentario.

La primera opción para usar cuando se crea un archivo nuevo es la **c**, seguida de la opción **f**, luego un solo espacio, el nombre del fichero del archivo que se creará y, por último, la lista de los ficheros y directorios que deben agregarse al archivo. El archivo se crea en el directorio actual, a menos que se especifique lo contrario.

En el siguiente ejemplo, se crea un archivo data.tar que contiene todos los ficheros data*.json del directorio:

En el siguiente ejemplo, se crea un archivo con el nombre archive.tar que contiene fichero1, fichero2 y fichero3 en el directorio de inicio del usuario:

1. Estado inicial del directorio:

   ```
   iso@debian12-vm:~$ du -ch data*.json
   192K  data01.json
   188K  data02.json
   192K  data03.json
   572K  total
   ```

   El directorio tiene tres archivos data*.json (data01.json, data02.json y data03.json) que en total pesan 572 KB.

2. Empaquetar todos los archivos data*.json en un archivo data.tar

```
iso@debian12-vm:~$ tar cf data.tar data*.json
```

Con **cf** Creamos el archivo data.tar que contendrá todos los archivos data*.json.

Si agregamos la opción **v**, podemos ver que archivos se están comprimiendo:
tar cvf data.tar data*.json

3. Listar los contenidos de data.tar

```
iso@debian12-vm:~$ tar tf data.tar
data01.json
data02.json
data03.json
```

4. Ver el peso del archivo data.tar

```
iso@debian12-vm:~$ du -h data.tar
572K  data.tar
```

El archivo data.tar pesa exactamente lo mismo que la suma de los pesos de los tres archivos data*.json. Eso es porque tar por sí mismo solo empaqueta los archivos, pero no los comprime.

5. Desempaquetar los archivos

```
iso@debian12-vm:~$ mkdir data
iso@debian12-vm:~$ cd data
iso@debian12-vm:~/data$ tar xf ../data.tar
iso@debian12-vm:~/data$ ls
data01.json  data02.json  data03.json
```

tar descomprime los archivos en el directorio actual, por eso es necesario crear la carpeta y posicionarse en ella antes de usarlo para desempaquetar.

Por lo general, un archivo tar debería extraerse en un directorio vacío para garantizar que no sobrescriba ningún archivo existente.

> Si los archivos son extraídos por el usuario root, tar intenta conservar el usuario original y la propiedad del grupo de los archivos. Si un usuario habitual extrae los archivos con tar, los archivos extraídos son propiedad de ese usuario.
>
> De manera predeterminada, cuando se extraen ficheros de un archivo, el desenmascaramiento se elimina de los permisos de contenido del archivo. Para proteger los permisos de un fichero archivado, se usará la opción **p** cuando se extraiga un archivo: **tar xpf ../data.tar**

Creación de un archivo tar comprimido

Para crear un archivo comprimido puede especificarse una de las siguientes opciones:

- **z** para la compresión gzip (*.tar.gz, *.tgz)
- **j** para la compresión bzip2 (*.tar.bz2)
- **J** para la compersión xz (*.tar.xz)

```
iso@debian12-vm:~$ tar czf data.tar.gz  data*.json
iso@debian12-vm:~$ tar cjf data.tar.bz2 data*.json
iso@debian12-vm:~$ tar cJf data.tar.xz  data*.json
iso@debian12-vm:~$ du -h data.tar* | sort -h
76K   data.tar.bz2
92K   data.tar.xz
140K  data.tar.gz
572K  data.tar
```

Extracción de un archivo tar comprimido

El primer paso cuando se extrae un archivo tar comprimido es determinar el lugar donde se extraerán los ficheros archivados y, a continuación, crear y cambiar el directorio de destino.

Para extraer correctamente el archivo, en general no es necesario usar la misma opción de compresión utilizada cuando se crea el archivo, ya que el comando tar determinará cuál es la compresión que se usó:

```
iso@debian12-vm:~$ tar xf data.tar.gz
iso@debian12-vm:~$ tar xf data.tar.bz2
iso@debian12-vm:~$ tar xf data.tar.xz
```

Es válido agregar un método de descompresión a las opciones de tar de la siguiente manera:

```
iso@debian12-vm:~$ tar xzf data.tar.gz
iso@debian12-vm:~$ tar xjf data.tar.bz2
iso@debian12-vm:~$ tar xJf data.tar.xz
```

Nota

Además, **gzip**, **bzip2** y **xz** se pueden usar de manera independiente para comprimir archivos individuales. Por ejemplo, **gzip etc.tar** genera el archivo comprimido **etc.tar.gz**, mientras que **bzip2 abc.tar** genera el archivo comprimido **abc.tar.bz2** y **xz myarchive.tar** genera el archivo comprimido **myarchive.tar.xz**.

Los comandos de descompresión correspondientes son **gunzip**, **bunzip2** y **unxz**. Por ejemplo, **gunzip /tmp/etc.tar.gz** genera el archivo tar sin comprimir **etc.tar**, mientras que **bunzip2 abc.tar.bz2** genera el archivo tar sin comprimir **abc.tar** y **unxz myarchive.tar.xz** genera el archivo tar sin comprimir **myarchive.tar**.

Red Hat (2017). Administración de archivos tar comprimidos. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 12, pp. 318-322). Red Hat.