

1) Editor de textos:

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

VI/VIM, MCEdit y Nano.

- 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.

Cat, more y less solo muestran el contenido de un archivo en diferentes formatos, un editor de texto permite editar el contenido de los mismos.

i, a, o, O: Modo de inserción.

i: Insertar texto antes del cursor.

I: Insertar texto al inicio de la línea actual.

a: Insertar texto después del cursor.

A: Insertar texto al final de la línea actual.

o: Insertar texto en una nueva línea debajo del cursor.

O: Insertar texto en una nueva línea arriba del cursor.

Esc: Modo comandos.

Esc y : : Modo línea de comandos.

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

Comandos más comunes:

q!: Salir del editor sin guardar los cambios.

wq!: Guardar los cambios y salir del editor.

w: Guardar los cambios.

num: Mostrar el número de línea actual.

yy: Copiar una línea de texto.

p: Pegar una línea de texto previamente copiada con yy.

b: Ir al inicio de la palabra.

e: Ir al final de la palabra.

x: Borrar un solo carácter.

dd: Eliminar una línea entera.

Xdd: Borrar X número de líneas.

Xyy: Copiar X número de líneas.

G: Ir a la última línea del archivo.

XG: Ir a la línea X del archivo.

gg: Ir a la primera línea del archivo.


2) Proceso de Arranque SystemV (<https://github.com/systeminnit/si>):

- 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.

Proceso de arranque System V

(<https://github.com/systeminnit/si>)

- 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




Facultad de Informática
UNIVERSIDAD NACIONAL DE LA PLATA

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

Lo ejecuta el Kernel.

init

- 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



Facultad de Informática
UNIVERSIDAD NACIONAL DE LA PLATA

c) RunLevels ¿Qué son? ¿Cuál es su objetivo?

Es el modo en que arranca Linux (3 en Redhat, 2 en Debian).

El proceso de arranque se divide en niveles, cada uno es responsable de iniciar o parar una serie de servicios.

Se encuentran definidos en /etc/inittab -> id:nivelesEjecucion:acción:proceso

- Id: Identifica la entrada en inittab (1 a 4 caracteres).
- 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
 - Ctrlaltdel: Se ejecutará cuando init reciba la señal SIGINT.
 - Off, respawn, once, sysinit, boot, bootwait, powerwait, etc
- Proceso: El proceso exacto que será ejecutado.

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

d) ¿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, y permiten 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 basado en X)
 - **6**: reboot

- Los scripts que se ejecutan están en /etc/init.d
- En /etc/rcX.d (donde X = 0..6) hay links a los archivos del /etc/init.d

- Formato de los links:

[S|K]<orden><nombreScript>

```
$ ls -l /etc/rcS.d/
S55urandom
S70x11-common
```

- **S**: lanza el script con el argument start
- **K**: lanza el script con el argument stop

Robado de <https://github.com/agusrnfr/>

Cuando un sistema GNU/Linux arranca, primero se carga el kernel del sistema, después se inicia el primer proceso, denominado init, que es el responsable de ejecutar y activar el resto del sistema, mediante la gestión de los niveles de ejecución (o runlevels).

En el caso del modelo runlevel de SystemV, cuando el proceso init arranca, utiliza un fichero de configuración llamado /etc/inittab para decidir el modo de ejecución en el que va a entrar. En este fichero se define el runlevel por defecto (initdefault) en arranque (por instalación en Fedora el 5, en Debian el 2), y una serie de servicios de terminal por activar para atender la entrada del usuario.

Después, el sistema, según el runlevel escogido, consulta los ficheros contenidos en /etc/rcn.d, donde n es el número asociado al runlevel (nivel escogido), en el que se encuentra una lista de servicios por activar o parar en caso de que arranquemos en el runlevel, o lo abandonemos. Dentro del directorio encontraremos una serie de scripts o enlaces a los scripts que controlan el servicio. Cada script posee un nombre relacionado con el servicio, una S o K inicial que indica si es el script para iniciar (S) o matar (K) el servicio, y un número que refleja el orden en que se ejecutarán los servicios.

No todas las distribuciones respetan los estándares.

- e) Archivo `/etc/inittab`. ¿Cuál es su funcionalidad? ¿Qué tipo de información se almacena en él? ¿Cuál es la estructura de la información que en él se almacena?

Robado de <https://github.com/agusrnfr/>

Es el archivo de configuración de `init`. Cuando el sistema se arranca, se verifica si existe un runlevel predeterminado en el archivo `/etc/inittab`, si no, se debe introducir por medio de la consola del sistema. Después se procede a ejecutar todos los scripts relativos al runlevel especificado.

- f) Suponga que se encuentra en el RunLevel `<X>`. Indique qué comando/s ejecutaría para cambiar al Runlevel `<Y>`. ¿Este cambio es permanente? ¿Por qué?

El comando necesario es `init`.

```
leo@leo:~$ init --help
init [OPTIONS...] COMMAND

Send control commands to the init daemon.

Commands:
  0          Power-off the machine
  6          Reboot the machine
  2, 3, 4, 5 Start runlevelX.target unit
  1, s, S    Enter rescue mode
  q, Q       Reload init daemon configuration
  u, U       Reexecute init daemon

Options:
  --help      Show this help
  --no-wall   Don't send wall message before halt/power-off/reboot

See the telinit(8) man page for details.
```

Los cambios realizados con `init` son temporales y afectan a la sesión actual del sistema. Si se cambia el nivel de ejecución, el sistema solo aplicará para esa sesión.

Si se reinicia el sistema, el nivel de ejecución predeterminado se restaurará al configurado en los archivos de configuración (`/etc/inittab` y `systemd`).

Los cambios permanentes se deben realizar directamente sobre los archivos de configuración del sistema.

- g) 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.

Los scripts RC (Run Commands) tienen la finalidad de iniciar o detener servicios cuando el sistema GNU/Linux arranca o se detiene. Su principal propósito es asegurarse de que los servicios (como redes, daemons (procesos que se ejecutan en segundo plano), montajes de FileSystem, etc) se inicien y se apaguen correctamente a medida que el sistema cambia de estado operativo o nivel de ejecución (RunLevel).

Los scripts RC se almacenan en directorios específicos que dependen del sistema de inicialización que utiliza la distribución Linux:

En sistemas basados en sysvinit:

- /etc/rc.d o /etc/init.d
- Para cada Runlevel existen subdirectorios llamados /etc/rcX.d, donde X es el número de RunLevel. Cada uno de estos directorios contiene enlaces simbólicos a los scripts en /etc/init.d.

Los scripts se ejecutan en un orden específico:

- El orden de los scripts en cada nivel de ejecución se determina por los números después de las letras "S" (start) y "K" (kill), por ejemplo, el script "S10cron" se ejecutará antes que "S20network" debido a que $10 < 20$.
- Este orden asegura que algunos servicios críticos se inicien o se detengan en un orden específico para evitar conflictos.

3) SystemD (<https://github.com/systemd/systemd>):

- a) ¿Qué es systemd?

SystemD es un conjunto de daemons de administración de sistema, bibliotecas y herramientas diseñados como una plataforma de administración y configuración central para interactuar con el núcleo del Sistema Operativo GNU/Linux.

Descrito como "bloque de construcción básico" para un sistema operativo, SystemD se puede utilizar tanto como un sistema de inicio de Linux (el proceso init llamado por el núcleo o kernel de Linux para inicializar el espacio de usuario durante el proceso de arranque de Linux y gestionar posteriormente todos los demás procesos). El nombre SystemD se adhiere a la convención de Unix de distinguir los daemons fácilmente por tener la letra "D" como última letra del nombre del archivo.

b) ¿A qué hace referencia el concepto de Unit en SystemD?

Se denomina Unit a las unidades de trabajo de tipo:

- **Service:** Controla un servicio particular (.service).
- **Socket:** Encapsula IPC, un socket del sistema o FileSystem FIFO (.socket) -> socket-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 reestablecido más tarde (.snapshot).
- Etc.

Las Units pueden tener dos estados: Active o Inactive.

c) ¿Para qué sirve el comando systemctl en SystemD?

```
leo@leo:~$ systemctl --help
systemctl [OPTIONS...] COMMAND ...

Query or send control commands to the system manager.

Unit Commands:
list-units [PATTERN...]           List units currently in memory
list-automounts [PATTERN...]      List automount units currently in memory,
                                   ordered by path
list-paths [PATTERN...]           List path units currently in memory,
                                   ordered by path
list-sockets [PATTERN...]         List socket units currently in memory,
                                   ordered by address
list-timers [PATTERN...]          List timer units currently in memory,
                                   ordered by next elapse
is-active PATTERN...              Check whether units are active
is-failed [PATTERN...]            Check whether units are failed or
                                   system is in degraded state
status [PATTERN...|PID...]        Show runtime status of one or more units
show [PATTERN...|JOB...]          Show properties of one or more
                                   units/jobs or the manager
cat PATTERN...                    Show files and drop-ins of specified units
help PATTERN...|PID...            Show manual for one or more units
list-dependencies [UNIT...]        Recursively show units which are required
                                   or wanted by the units or by which those
                                   units are required or wanted
start UNIT...                     Start (activate) one or more units
stop UNIT...                       Stop (deactivate) one or more units
lines 1-27...skipping...
```

d) ¿A qué hace referencia el concepto de target en SystemD?

En SystemD, un Target es una unidad que agrupa otras unidades para alcanzar un estado específico del sistema. Es un concepto que sirve para organizar y controlar las dependencias entre los diferentes servicios, sockets, dispositivos, y otros targets que deben estar activos o inactivos en un momento determinado.

Los Targets son una forma de abstraer y agrupar acciones comunes de arranque y apagado del sistema, permitiendo que SystemD gestione el inicio, detención y reinicio de servicios en el orden correcto. Un Target define un “estado del sistema”, que es un conjunto de servicios y dependencias que deben estar funcionando para cumplir un propósito particular.

Ejemplos de Targets:

- multi-user.target: Indica que el sistema está en un estado donde varios usuarios pueden acceder de manera no gráfica.
- graphical.target: Similar al ejemplo anterior, pero además habilita la interfaz gráfica.
- rescue.target: Proporciona un entorno de rescate con los servicios mínimos necesarios para reparaciones o diagnósticos.
- shutdown.target: Se utiliza cuando el sistema se apaga, asegurándose de que todos los servicios se detengan correctamente antes de apagar la máquina.

Los Targets pueden depender unos de otros o de otros servicios. Por ejemplo, graphical.target depende de multi-user.target porque se necesita una base de servicios multiusuario antes de que se inicie la interfaz gráfica.

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

El comando **pstree** muestra un árbol con los procesos que se están ejecutando.


```

leo@leo:~$ pstree
systemd--ModemManager--3*[{ModemManager}]
--NetworkManager--3*[{NetworkManager}]
--accounts-daemon--3*[{accounts-daemon}]
--atd
--avahi-daemon--avahi-daemon
--colord--3*[{colord}]
--containerd--14*[{containerd}]
--cron
--cups-browsed--3*[{cups-browsed}]
--cupsd
--dbus-daemon
--dockerd--16*[{dockerd}]
--fwupd--5*[{fwupd}]
--gdm3--gdm-session-wor--gdm-x-session--Xorg--11*[{Xorg}]
--gnome-session-b--3*[{gnome-session-b}]
--3*[{gdm-x-session}]
--3*[{gdm-session-wor}]
--3*[{gdm3}]
--gnome-remote-de--3*[{gnome-remote-de}]
--2*[{kerneloops}]
--mysqld--36*[{mysqld}]
--polkitd--3*[{polkitd}]
--postgres--5*[{postgres}]
--power-profiles--3*[{power-profiles-}]
--rsyslogd--3*[{rsyslogd}]
--rtkit-daemon--2*[{rtkit-daemon}]
--snapd--18*[{snapd}]
--switcheroo-cont--3*[{switcheroo-cont}]
systemd--(sd-pam)
--at-spi-bus-laun--dbus-daemon
--4*[{at-spi-bus-laun}]
--at-spi2-registr--3*[{at-spi2-registr}]
--2*[{chrome_crashpad}--2*[{chrome_crashpad}]]
--chrome_crashpad--(chrome_crashpad)
--com.docker.back--com.docker.back--Docker Desktop--Docker Desktop--Docker Desktop--28*[{Docker Desktop}]
--Docker Desktop--Docker Desktop
--Docker Desktop--9*[{Docker Desktop}]
--34*[{Docker Desktop}]
--com.docker.buil--15*[{com.docker.buil}]
--com.docker.dev--8*[{com.docker.dev-}]
--24*[{com.docker.back}]
--14*[{com.docker.back}]
--dbus-daemon
--dconf-service--3*[{dconf-service}]
--2*[{evince}--7*[{evince}]]
--evinced--3*[{evinced}]
--evolution-addre--6*[{evolution-addre}]
--evolution-calen--9*[{evolution-calen}]
--evolution-sourc--4*[{evolution-sourc}]
--gcr-ssh-agent--2*[{gcr-ssh-agent}]
--2*[{gjs}--11*[{gjs}]]
--gnome-keyring-d--4*[{gnome-keyring-d}]
--gnome-session-b--evolution-alarm--7*[{evolution-alarm}]
--gsd-disk-utilit--3*[{gsd-disk-utilit}]
--update-notifier--5*[{update-notifier}]
--4*[{gnome-session-b}]
--gnome-session-c--(gnome-session-c)
--gnome-shell--Discord--Discord--Discord--30*[{Discord}]
--Discord--Discord
--Discord--14*[{Discord}]
--Discord--5*[{Discord}]
--Discord--47*[{Discord}]
--45*[{Discord}]
--brave-browser-s--brave--brave--brave--30*[{brave}]

```

4) Usuarios:

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

/etc/passwd: Contiene una lista de los usuarios del sistema. Cada línea representa un usuario y contiene campos como el nombre de usuario, ID del usuario (UID), el ID del grupo (GID), la ruta de inicio (home directory), y la shell predeterminada.

/etc/shadow: Almacena las contraseñas cifradas y la información relacionada con la política de contraseñas, como la fecha de expiración y los períodos de validez. Solo los usuarios privilegiados (como root) pueden acceder a este archivo por razones de seguridad.

/etc/group: Tiene la información sobre los grupos del sistema y los usuarios asociados a cada grupo. Cada línea representa un grupo, con detalles como el nombre del grupo, el GID y los usuarios que son miembros del grupo.

/etc/gshadow: Similar a /etc/shadow, este archivo contiene las contraseñas cifradas de los grupos (si los grupos tienen contraseña) y otra información sobre los grupos.

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

UID hace referencia a la ID del usuario (**U**ser **I**Dentifier).

GID hace referencia a la ID del grupo (**G**roup **I**Dentifier).

La UID es la forma que tiene el sistema operativo de identificar al usuario, por lo tanto si 2 o más usuarios tienen la misma UID, el sistema no podría diferenciar entre ellos y ambos tendrían acceso a los mismos archivos y permisos, lo que sería un problema de seguridad.

En la práctica sí es posible que esto pase, se pueden mantener varios usuarios con la misma UID. A su vez, esto va en contra de las prácticas recomendadas y puede causar problemas.

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

El usuario root es el administrador del sistema o superusuario que tiene control total sobre todo el sistema. Puede ejecutar cualquier comando y tiene acceso a todos los archivos y recursos, sin restricciones de permisos.

En la práctica, se podrían tener múltiples usuarios root, cualquier usuario al que se le asigne la UID 0 tendrá permisos de administrador.

```
leo@leo:~$ cat /etc/passwd | grep "root"
root:x:0:0:root:/root:/bin/bash
```

- 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.

```
root@8319eb6f9136:/# groupadd catedra
root@8319eb6f9136:/# sudo useradd -d "/home/iso_2017" -m iso2017
root@8319eb6f9136:/# sudo usermod -a -G catedra iso2017
root@8319eb6f9136:/# cat /etc/passwd | grep "iso2017" && cat /etc/group | grep "catedra"
iso2017:x:1000:1001::/home/iso_2017:/bin/sh
catedra:x:1000:iso2017
root@8319eb6f9136:/# cd /home/iso_2017
root@8319eb6f9136:/home/iso_2017# touch ejemplo.txt
root@8319eb6f9136:/home/iso_2017# userdel iso2017
root@8319eb6f9136:/home/iso_2017# cat /etc/passwd | grep "iso2017"
root@8319eb6f9136:/home/iso_2017#
```

- e) Investigue la funcionalidad y parámetros de los siguientes comandos:

- useradd o adduser

```
leo@leo:~$ useradd --help
Usage: useradd [options] LOGIN
       useradd -D
       useradd -D [options]

Options:
  --badname                do not check for bad names
  -b, --base-dir BASE_DIR  base directory for the home directory of the
                           new account
  --btrfs-subvolume-home   use BTRFS subvolume for home directory
  -c, --comment COMMENT    GECOS field of the new account
  -d, --home-dir HOME_DIR  home directory of the new account
  -D, --defaults            print or change default useradd configuration
  -e, --expiredate EXPIRE_DATE expiration date of the new account
  -f, --inactive INACTIVE  password inactivity period of the new account
  -F, --add-subids-for-system add entries to subuidid even when adding a system user
  -g, --gid GROUP           name or ID of the primary group of the new
                           account
  -G, --groups GROUPS       list of supplementary groups of the new
                           account
  -h, --help                display this help message and exit
  -k, --skel SKEL_DIR       use this alternative skeleton directory
  -K, --key KEY=VALUE        override /etc/login.defs defaults
  -l, --no-log-init          do not add the user to the lastlog and
                           faillog databases
  -m, --create-home          create the user's home directory
  -M, --no-create-home       do not create the user's home directory
  -N, --no-user-group        do not create a group with the same name as
                           the user
  -o, --non-unique           allow to create users with duplicate
                           (non-unique) UID
  -p, --password PASSWORD   encrypted password of the new account
  -r, --system              create a system account
  -R, --root CHROOT_DIR     directory to chroot into
  -P, --prefix PREFIX_DIR   prefix directory where are located the /etc/* files
  -s, --shell SHELL         login shell of the new account
  -u, --uid UID             user ID of the new account
  -U, --user-group          create a group with the same name as the user
  -Z, --selinux-user SEUSER use a specific SEUSER for the SELinux user mapping
  --extrausers              Use the extra users database
```

- usermod

```
leo@leo:~$ usermod --help
Usage: usermod [options] LOGIN

Options:
  -a, --append                append the user to the supplemental GROUPS
                              mentioned by the -G option without removing
                              the user from other groups
  -b, --badname               allow bad names
  -c, --comment COMMENT      new value of the GECOS field
  -d, --home HOME_DIR        new home directory for the user account
  -e, --expiredate EXPIRE_DATE set account expiration date to EXPIRE_DATE
  -f, --inactive INACTIVE    set password inactive after expiration
                              to INACTIVE
  -g, --gid GROUP             force use GROUP as new primary group
  -G, --groups GROUPS        new list of supplementary GROUPS
  -h, --help                  display this help message and exit
  -l, --login NEW_LOGIN      new value of the login name
  -L, --lock                  lock the user account
  -m, --move-home             move contents of the home directory to the
                              new location (use only with -d)
  -o, --non-unique            allow using duplicate (non-unique) UID
  -p, --password PASSWORD    use encrypted password for the new password
  -P, --prefix PREFIX_DIR    prefix directory where are located the /etc/* files
  -r, --remove                remove the user from only the supplemental GROUPS
                              mentioned by the -G option without removing
                              the user from other groups
  -R, --root CHROOT_DIR      directory to chroot into
  -s, --shell SHELL          new login shell for the user account
  -u, --uid UID              new UID for the user account
  -U, --unlock                unlock the user account
  -v, --add-subuids FIRST-LAST add range of subordinate uids
  -V, --del-subuids FIRST-LAST remove range of subordinate uids
  -w, --add-subgids FIRST-LAST add range of subordinate gids
  -W, --del-subgids FIRST-LAST remove range of subordinate gids
  -Z, --selinux-user SEUSER  new SELinux user mapping for the user account
```

- userdel

```
leo@leo:~$ userdel --help
Usage: userdel [options] LOGIN

Options:
  -f, --force                force some actions that would fail otherwise
                              e.g. removal of user still logged in
                              or files, even if not owned by the user
  -h, --help                  display this help message and exit
  -r, --remove                remove home directory and mail spool
  -R, --root CHROOT_DIR      directory to chroot into
  -P, --prefix PREFIX_DIR    prefix directory where are located the /etc/* files
  --extrausers                Use the extra users database
  -Z, --selinux-user         remove any SELinux user mapping for the user
```

- su

```
leo@leo:~$ su --help

Usage:
  su [options] [-] [<user> [<argument>...]]

Change the effective user ID and group ID to that of <user>.
A mere - implies -l. If <user> is not given, root is assumed.

Options:
  -m, -p, --preserve-environment    do not reset environment variables
  -w, --whitelist-environment <list> don't reset specified variables

  -g, --group <group>               specify the primary group
  -G, --supp-group <group>          specify a supplemental group

  -, -l, --login                    make the shell a login shell
  -c, --command <command>           pass a single command to the shell with -c
  --session-command <command>      pass a single command to the shell with -c
                                   and do not create a new session
  -f, --fast                        pass -f to the shell (for csh or tcsh)
  -s, --shell <shell>              run <shell> if /etc/shells allows it
  -P, --pty                         create a new pseudo-terminal

  -h, --help                        display this help
  -V, --version                     display version

For more details see su(1).
```

- groupadd

```
leo@leo:~$ groupadd --help
Usage: groupadd [options] GROUP

Options:
  -f, --force                exit successfully if the group already exists,
                             and cancel -g if the GID is already used
  -g, --gid GID              use GID for the new group
  -h, --help                 display this help message and exit
  -K, --key KEY=VALUE        override /etc/login.defs defaults
  -o, --non-unique            allow to create groups with duplicate
                             (non-unique) GID
  -p, --password PASSWORD    use this encrypted password for the new group
  -r, --system               create a system account
  -R, --root CHROOT_DIR      directory to chroot into
  -P, --prefix PREFIX_DIR    directory prefix
  -U, --users USERS           list of user members of this group
  --extrausers                Use the extra users database
```

- who

```
leo@leo:~$ who --help
Usage: who [OPTION]... [ FILE | ARG1 ARG2 ]
Print information about users who are currently logged in.

-a, --all           same as -b -d --login -p -r -t -T -u
-b, --boot          time of last system boot
-d, --dead          print dead processes
-H, --heading       print line of column headings
-l, --login         print system login processes
    --lookup        attempt to canonicalize hostnames via DNS
-m                 only hostname and user associated with stdin
-p, --process       print active processes spawned by init
-q, --count         all login names and number of users logged on
-r, --runlevel      print current runlevel
-s, --short         print only name, line, and time (default)
-t, --time          print last system clock change
-T, -w, --mesg      add user's message status as +, - or ?
-u, --users         list users logged in
    --message       same as -T
    --writable       same as -T
    --help          display this help and exit
    --version       output version information and exit

If FILE is not specified, use /var/run/utmp.  /var/log/wtmp as FILE is common.
If ARG1 ARG2 given, -m presumed: 'am i' or 'mom likes' are usual.

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation <https://www.gnu.org/software/coreutils/who>
or available locally via: info '(coreutils) who invocation'
```

- groupdel

```
leo@leo:~$ groupdel --help
Usage: groupdel [options] GROUP

Options:
-h, --help          display this help message and exit
-R, --root CHROOT_DIR
                    directory to chroot into
-P, --prefix PREFIX_DIR
                    prefix directory where are located the /etc/* files
-f, --force         delete group even if it is the primary group of a user
    --extrausers     Use the extra users database
```

- passwd

```
leo@leo:~$ passwd --help
Usage: passwd [options] [LOGIN]

Options:
  -a, --all                report password status on all accounts
  -d, --delete             delete the password for the named account
  -e, --expire             force expire the password for the named account
  -h, --help              display this help message and exit
  -k, --keep-tokens        change password only if expired
  -i, --inactive INACTIVE set password inactive after expiration
                           to INACTIVE
  -l, --lock               lock the password of the named account
  -n, --mindays MIN_DAYS  set minimum number of days before password
                           change to MIN_DAYS
  -q, --quiet              quiet mode
  -r, --repository REPOSITORY change password in REPOSITORY repository
  -R, --root CHROOT_DIR   directory to chroot into
  -S, --status             report password status on the named account
  -u, --unlock            unlock the password of the named account
  -w, --warndays WARN_DAYS set expiration warning days to WARN_DAYS
  -x, --maxdays MAX_DAYS set maximum number of days before password
                           change to MAX_DAYS
```

5) FileSystems:

- a) ¿Cómo son definidos los permisos sobre los archivos en un sistema GNU/Linux?

Los archivos tienen 3 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.

Categoría de permisos:

- Lectura/Read (**r**):
En archivos: Pueden leerse los contenidos del archivo.
En directorios: Permite detallar los contenidos del directorio.
- Escritura/Write (**w**):
En archivos: Pueden modificarse los contenidos del archivo.
En directorios: Pueden crearse o eliminar archivos en el directorio.
- Ejecución/Execute (**x**):
En archivos: Se pueden ejecutar archivos como comandos.
En directorios: Es posible acceder al contenido del directorio.

```
leo@leo:~/Documents/GitHub/Facultad/Segundo/Segundo semestre/ISO/Prácticas/Práctica 2$ ls -l
total 1232
-rw-rw-r-- 1 leo leo 314335 Aug 30 18:38 'Explicación Práctica 2.pdf'
-rw-rw-r-- 1 leo leo 254686 Aug 21 19:27 'Practica 2.pdf'
-rw-rw-r-- 1 leo leo 685922 Sep 9 17:41 'Resolución ISO Práctica 2.pdf'
```

El primer caracter indica el tipo del archivo:

- “-” para archivos normales.
- “d” para directorios.
- “l” para enlaces simbólicos (symlinks), entre otros.

Los siguientes 9 caracteres indican los permisos de los grupos:

- Primer grupo: Permisos del propietario.
- Segundo grupo: Permisos del grupo.
- Tercer grupo: Permisos para otros usuarios.

En el caso del archivo “Practica 2.pdf”:

- El primer caracter “-” indica que es un archivo normal.
- Los siguientes 3 caracteres (rw-) indican que el propietario tiene permiso de lectura y escritura.
- Los siguientes 3, (rw-) indican que el grupo tiene permiso de lectura y escritura.
- Los últimos 3 (r--) indican que los otros usuarios tienen permiso de lectura.

- El “1” indica el número de enlaces al archivo o directorio.
- “leo” indica el usuario propietario.
- “leo” indica el grupo al que pertenece el archivo o directorio.
 - ↳ Mismo nombre debido al “User Private Group”.
- “254686” indica el peso en bytes.
- “Aug 21 19:27” indica la fecha y hora de la última modificación.
- “Practica 2.pdf”, finalmente, indica el nombre.

b) Investigue sobre la funcionalidad y parámetros de los siguientes comandos relacionados con los permisos en GNU/Linux:

- chmod
- chown
- chgrp

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?

La respuesta resuelve los dos incisos.

chmod (Change Mode): Se utiliza para cambiar los permisos de un archivo o directorio. En GNU/Linux, los permisos definen qué usuarios pueden leer, escribir o ejecutar un archivo o directorio.

- chmod [OPTIONS] [PERMISSIONS] FILE}

Modo simbólico: Se utilizan letras para representar los permisos:

- r para lectura (read).
- w para escritura (write).
- x para ejecución (execute).

Y utiliza la siguiente sintaxis:

- u para el usuario.
- g para el grupo.
- o para otros.

chmod u+rx, g+rx, o+r "Practica 2.pdf"

- + para agregar el permiso.
- - para quitar el permiso.
- = para dar exactamente.

Modo numérico: Los permisos se representan con números. El sistema de permisos se basa en valores octales:

- 4 para la lectura.
- 2 para la escritura.
- 1 para la ejecución.

Estos valores se suman para asignar un permiso, por ejemplo:

Lectura+Escritura > 2+4 = 6

Lectura+Escritura+Ejecución > 2+4+1 = 7

Lectura > 2 = 4

Entonces se puede utilizar, siguiendo el orden previo (u g o):

chmod 672 "Practica 2.pdf"

```
leo@leo:~/Documents/GitHub/Facultad/Segundo/Segundo semestre/ISO/Prácticas/Práctica 2$ ls -l | grep "Practica 2"
-rw-rw-r-- 1 leo leo 254686 Aug 21 19:27 Practica 2.pdf
leo@leo:~/Documents/GitHub/Facultad/Segundo/Segundo semestre/ISO/Prácticas/Práctica 2$ chmod 674 "Practica 2.pdf"
leo@leo:~/Documents/GitHub/Facultad/Segundo/Segundo semestre/ISO/Prácticas/Práctica 2$ ls -l | grep "Practica 2"
-rw-rwxr-- 1 leo leo 254686 Aug 21 19:27 Practica 2.pdf
```

Nota: Opción más común: **-R**, cambia los permisos de manera recursiva en todo el directorio.

chown (Change Ownership): Se utiliza para cambiar el propietario (usuario) y/o el grupo asociado a un archivo o directorio.

- chown [OPTIONS] [USER][:[GROUP]] FILE

Ejemplos:

- chown Pepe "Practica 2.pdf" # Cambia el propietario de "leo" a "Pepe".
- chown Pepe:admin "Practica 2.pdf" # Cambia el propietario de "leo" a "Pepe" y el grupo de "leo" a "admin".

Nota: Opciones más comunes:

-R, cambia el propietario y grupo de manera recursiva en todos los subdirectorios y archivos.

--reference=archivo, cambia el propietario y grupo de un archivo utilizando los mismos que tiene otro archivo de referencia.

chgrp (Change Group): Se utiliza para cambiar el grupo asociado a un archivo o directorio sin cambiar el propietario.

- `chgrp [OPTIONS] [GROUP] FILE`

Ejemplo:

- `chgrp admin "Practica 2.pdf"` # Cambia el grupo de "leo" a "admin".

Nota: Opción más común: **-R**, cambia los permisos de manera recursiva en todo el directorio.

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

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

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

Full Path Name (Rutas absolutas): Una ruta absoluta es un nombre completamente calificado que comienza con 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 como una regla simple: un nombre de archivo con una barra (/) como primer caracter es el nombre de la ruta absoluta. Por ejemplo, el nombre de ruta absoluta para el directorio donde se encuentra este archivo es:

```
leo@leo:~/Documents/GitHub/Facultad/Segundo/Segundo semestre/ISO/Prácticas/Práctica 2$ pwd
/home/leo/Documents/GitHub/Facultad/Segundo/Segundo semestre/ISO/Prácticas/Práctica 2
```

Relative Path Name (Rutas relativas): 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 nombre de ruta relativos, se sigue una regla simple: un nombre de ruta que no tenga otro caracter más que una barra (/) como primer caracter es un nombre de ruta relativo. Un usuario en el directorio "ISO" podría referirse a la ruta del archivo actual como **Prácticas/Práctica 2**.

```
leo@leo:~/Documents/GitHub/Facultad/Segundo/Segundo semestre/ISO$ cd Prácticas/Práctica\ 2
leo@leo:~/Documents/GitHub/Facultad/Segundo/Segundo semestre/ISO/Prácticas/Práctica 2$
```

↳ La \ es necesaria debido a que hay un espacio en el nombre.

- 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 sin necesidad de escribir todo el path completo se puede utilizar “**cd**”.

También se puede utilizar “**cd ~**” para acceder al directorio personal o acortar las rutas.

```
leo@leo:~/Documents/GitHub/Facultad/Segundo/Segundo semestre/ISO/Prácticas/Práctica 2$ pwd
/home/leo/Documents/GitHub/Facultad/Segundo/Segundo semestre/ISO/Prácticas/Práctica 2
leo@leo:~/Documents/GitHub/Facultad/Segundo/Segundo semestre/ISO/Prácticas/Práctica 2$ cd
leo@leo:~$ pwd
/home/leo
```

Utilizando “~” también se puede acceder al directorio personal de otro usuario, por ejemplo:

- cd ~pepe/Downloads
- ↳ Accede al directorio “Downloads” del usuario “pepe”

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

- cd
- umount
- mkdir
- du
- rmdir
- df
- mount
- ln
- ls
- pwd
- cp
- mv

cd	Cambiar el directorio de trabajo actual.
umount	Desmontar sistemas de archivos o dispositivos.
mkdir	<p>Crear un directorio, si no existe.</p> <p>-m, --mode=MODE Establece los permisos del directorio MODE es un octal.</p> <p>-p, --parents Crea directorios padres según sea necesario, con sus modos de archivo no afectados por ninguna opción -m.</p> <p>-v, --verbose Imprime cada directorio creado.</p>
du	<p>Muestra el uso del espacio en disco por archivos y directorios.</p> <p>-a, --all Muestra todos los archivos, no solo directorios.</p> <p>-d, --max-depth=N Imprime los valores hasta N niveles.</p> <p>-h, --human-readable Imprime los tamaños en formato legible (1K, 234M, 2G).</p> <p>-c, --total Imprime el peso total</p> <p>\$ du -a -d -h 1 dir sort -h</p>
rmdir	<p>Elimina un directorio, si está vacío.</p> <p>-p, --parents Elimina el directorio y sus ancestros. "rmdir -p a/b/c" es equivalente a "rmdir a/b/c a/b a".</p> <p>-v, --verbose Muestra un diagnóstico por cada directorio procesado</p>
df	<p>Muestra información sobre el espacio disponible y utilizado en los sistemas de archivos montados.</p> <p>-h, --human-readable Imprime los tamaños en formato legible (1K, 234M, 2G).</p>
mount	Montar sistemas de archivos en un directorio específico
ln	<p>Crea enlaces duros (por defecto) o simbólicos a archivos y directorios.</p> <p>-s, --symbolic Crea enlaces simbólicos ln -s file link</p>

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.

6) Procesos:

- ¿Qué es un proceso? ¿A qué hacen referencia las siglas PID y PPID? ¿Todos los procesos tienen estos atributos en GNU/Linux? Justifique. Indique qué otros atributos tiene un proceso.
- Indique qué comandos se podrían utilizar para ver qué procesos están en ejecución en un sistema GNU/Linux.

La respuesta resuelve los dos incisos.

Un proceso es un programa que se está ejecutando.

El **PID** (Process IDentifier) es la identificación del proceso, y el **PPID** (Parent Process IDentifier) 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 derivan todos los demás procesos.

Además del **PID** y **PPID** los procesos tienen otros atributos como:

- 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.

```
leo@leo:~/Documents/GitHub/Facultad$ ps -e -o pid,ppid,cmd
PID    PPID  CMD
1       0    /sbin/init splash
2       0    [kthreadd]
3       2    [pool_workqueue_release]
4       2    [kworker/R-rcu_g]
5       2    [kworker/R-rcu_p]
6       2    [kworker/R-slub_]
7       2    [kworker/R-netns]
10      2    [kworker/0:0H-events_highpri]
12      2    [kworker/R-mm_pe]
13      2    [rcu_tasks_kthread]
14      2    [rcu_tasks_rude_kthread]
15      2    [rcu_tasks_trace_kthread]
16      2    [ksoftirqd/0]
17      2    [rcu_preempt]
18      2    [migration/0]
19      2    [idle_inject/0]
20      2    [cpuhp/0]
21      2    [cpuhp/1]
22      2    [idle_inject/1]
23      2    [migration/1]
24      2    [ksoftirqd/1]
```

Comando top:

```
top - 16:50:38 up 4:06, 1 user, load average: 1.24, 0.92, 0.98
Tasks: 369 total, 1 running, 368 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.7 us, 0.9 sy, 0.0 ni, 96.2 id, 0.1 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 15349.5 total, 2132.1 free, 6410.0 used, 7430.5 buff/cache
MiB Swap: 4096.0 total, 3482.0 free, 614.0 used, 8939.5 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
15549	leo	20	0	1156.3g	602652	134928	S	26.6	3.8	8:00.52	brave
14587	leo	20	0	62.7g	505468	309628	S	4.0	3.2	3:41.08	AudioRelay
1658	leo	0	-11	200644	74460	11576	S	3.0	0.5	3:54.85	pipewire-pulse
6960	leo	20	0	4925840	217208	144952	S	3.0	1.4	2:57.00	spotify
3546	leo	20	0	1149.2g	518794	114048	S	2.7	3.3	12:56.15	Discord
1655	leo	0	-11	130692	33724	10624	S	1.7	0.2	2:15.56	pipewire
1721	leo	20	0	1427160	155536	89368	S	1.7	1.0	8:45:70	Xorg
11169	leo	20	0	912368	61832	46604	S	1.7	0.4	1:44.85	pavucontrol
1635	mysql	20	0	2375828	245888	10752	S	0.7	1.6	0:59.62	mysqld
2120	leo	20	0	5017172	249088	126808	S	0.7	1.6	8:20.83	gnome-shell
4591	leo	20	0	33.1g	234336	149548	S	0.7	1.5	10:25.72	brave
16782	leo	20	0	922660	61760	44084	S	0.7	0.4	0:07.14	gnome-terminal-
1378	root	20	0	336660	18116	15428	S	0.3	0.1	0:00.90	NetworkManager
2229	leo	20	0	1132.3g	122184	87336	S	0.3	0.8	1:11.04	Discord
3140	leo	20	0	33.0g	135724	102768	S	0.3	0.9	1:58.94	Discord
4548	leo	20	0	32.9g	403744	259168	S	0.3	3.1	7:59.03	brave
4594	leo	20	0	32.5g	116804	99700	S	0.3	0.7	2:35.72	brave
7080	leo	20	0	1511224	49136	42944	S	0.3	0.3	0:04.34	spotify
17337	root	20	0	0	0	0	I	0.3	0.0	0:02.24	kworker/u64:2-events_freezable_power_
1	root	20	0	23916	14464	9216	S	0.0	0.1	0:01.88	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pool_workqueue_release
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_g
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_p
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-slub
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-netns
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
12	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-mm_pe
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_kthread
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_rude_kthread
15	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_trace_kthread
16	root	20	0	0	0	0	S	0.0	0.0	0:00.31	ksoftirqd/0
17	root	20	0	0	0	0	I	0.0	0.0	0:11.99	rcu_preempt
18	root	rt	0	0	0	0	S	0.0	0.0	0:00.04	migration/0
19	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
21	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
22	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/1

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

En sistemas GNU/Linux y Unix, los conceptos de **foreground** (primer plano) y **background** (segundo plano) describen cómo se ejecuta un proceso con respecto al usuario y la terminal.

Foreground:

- Un proceso que se ejecuta en primer plano está activo directamente en la terminal que lo inició. El proceso ocupa la terminal y recibe directamente las entradas del usuario (teclado), además de mostrar la salida (información) directamente en la misma.
- Mientras un proceso está en primer plano, el usuario no puede interactuar con la terminal para realizar otras acciones hasta que el proceso termine.

Ejemplo: Al utilizar “cat archivo.txt” el comando está en foreground y la terminal está ocupada con la ejecución de ese comando, no se puede utilizar hasta que el proceso finalice.

Background:

- Un proceso que se ejecuta en segundo plano se está ejecutando sin bloquear la terminal, lo que permite que el usuario continúe utilizando la terminal para otros comandos mientras el proceso sigue funcionando en segundo plano.
- Un proceso en segundo plano no recibe entradas del teclado directamente, pero sigue ejecutándose y produciendo resultados.

Ejemplo: Al utilizar un comando con el símbolo “&” al final, el proceso se ejecutará en segundo plano.

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

Para ejecutar un proceso en Background se puede agregar “&” al final:

- cat archivo.txt &

Para enviar un proceso de Foreground a Background se puede utilizar **Ctrl + Z** para suspenderlo temporalmente y después ejecutando “bg” para enviarlo al segundo plano. Para traerlo del Background se puede utilizar “fg”.

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

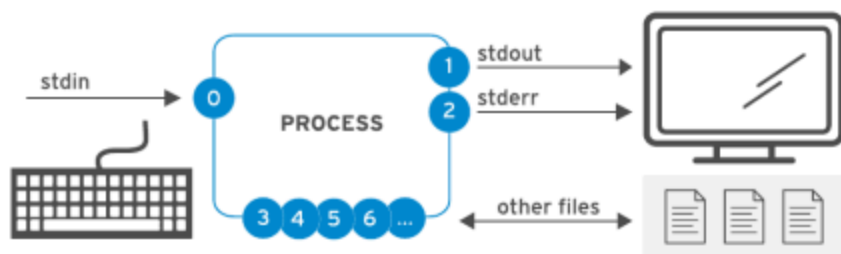


Figura 4.1: Canales de E/S de un proceso (descriptores de archivo)

#	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 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 del archivo se sobrescribirá. El archivo especial /dev/null descarta discretamente la salida del canal redirigida a él y es siempre un archivo vacío.

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

Ejemplos:

```
leo@leo:~/Desktop$ cat archivo1 archivo2 archivo3 archivo4
Esto
es
un
ejemplo
leo@leo:~/Desktop$ cat archivo1 archivo2 archivo3 archivo4 > archivoConcatenado
leo@leo:~/Desktop$ cat archivoConcatenado
Esto
es
un
ejemplo
leo@leo:~/Desktop$
```

```
leo@leo:~/Desktop$ cat archivo1 archivo2 >> archivoConcatenado
leo@leo:~/Desktop$ cat archivoConcatenado
Esto
es
un
ejemplo
Esto
es
leo@leo:~/Desktop$
```

```
leo@leo:~/Desktop$ cat archivo1 archivo2 2> archivoConcatenado
Esto
es
leo@leo:~/Desktop$ cat archivoConcatenado
leo@leo:~/Desktop$
```

```
leo@leo:~/Desktop$ cat archivo1 archivo2 &> archivoConcatenado
leo@leo:~/Desktop$ cat archivoConcatenado
Esto
es
leo@leo:~/Desktop$
leo@leo:~/Desktop$
```

g) Comando kill. ¿Cuál es su funcionalidad? Cite ejemplos.

El comando **kill** en GNU/Linux y Unix se utiliza para enviar señales a procesos en ejecución, generalmente con el propósito de terminarlos o modificar su comportamiento. Aunque el nombre sugiere que solo sirve para “matar” procesos, su funcionalidad va más allá, ya que puede enviar diversas señales para controlar los procesos.

Señales más comunes:

Señal	Número	Descripción	Uso común
<code>SIGHUP</code>	1	Hangup. Indica que el terminal fue cerrado o pide que se recargue la configuración.	Recargar configuraciones en procesos como <code>nginx</code> o <code>apache</code> .
<code>SIGINT</code>	2	Interrupt. Se envía cuando se presiona <code>Ctrl+C</code> .	Interrumpir un proceso en ejecución desde la terminal.
<code>SIGQUIT</code>	3	Quit. Se envía con <code>Ctrl+\</code> .	Terminar un proceso y generar un volcado de memoria (core dump).
<code>SIGKILL</code>	9	Kill. Mata un proceso inmediatamente. No puede ser ignorado.	Forzar la terminación de un proceso problemático.
<code>SIGTERM</code>	15	Terminate. Señal estándar para pedir que un proceso termine.	Terminar un proceso de manera suave y controlada.
<code>SIGCONT</code>	18	Continue. Restaura un proceso detenido con <code>SIGSTOP</code> .	Continuar un proceso que fue detenido.
<code>SIGSTOP</code>	19	Stop. Detiene un proceso sin terminarlo. No puede ser ignorado.	Pausar un proceso sin finalizarlo.
<code>SIGTSTP</code>	20	Stop signal. Se envía con <code>Ctrl+Z</code> .	Detener temporalmente un proceso en primer plano.
<code>SIGUSR1</code>	10	User-defined signal 1.	Señal definida por el usuario.
<code>SIGUSR2</code>	12	User-defined signal 2.	Señal definida por el usuario.
<code>SIGSEGV</code>	11	Segmentation fault. Indica un error de segmentación.	Generalmente usada por el sistema para indicar un error de memoria.

Nota: la señal default es SIGTERM (15).

- `kill 1234 #` Envía SIGTERM al proceso con PID 1234.
- `kill -9 1234 #` Envía SIGKILL al proceso con PID 1234.
- `kill -HUP 1234 #` Envía SIGHUP al proceso con PID 1234.
- `kill -u usuario #` Envía SIGTERM a todos los procesos del usuario.

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

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

Comando	Funcionalidad	Parámetros principales	Comparación
<code>ps</code>	Muestra una instantánea de los procesos actuales en el sistema.	<ul style="list-style-type: none"> - <code>-e</code> : Muestra todos los procesos. - <code>-f</code> : Muestra una lista completa. - <code>-o</code> : Define las columnas que se muestran (como <code>pid</code>, <code>ppid</code>, <code>cmd</code>, etc.). 	Proporciona una visión estática de los procesos en el momento de ejecución. Útil para listar y filtrar procesos según diferentes criterios. No muestra datos dinámicos como <code>top</code> .
<code>kill</code>	Envía señales a procesos, comúnmente para terminarlos.	<ul style="list-style-type: none"> - <code>-9</code> : Envía <code>SIGKILL</code>, termina un proceso inmediatamente. - <code>-15</code> : Envía <code>SIGTERM</code>, solicita terminar el proceso (por defecto). - <code>-l</code> : Lista todas las señales disponibles. 	Mata procesos individualmente. Requiere el PID de un proceso, a diferencia de <code>killall</code> , que actúa sobre procesos por nombre.
<code>pstree</code>	Muestra los procesos en un formato de árbol, mostrando la jerarquía de procesos.	<ul style="list-style-type: none"> - <code>-p</code> : Muestra los PID junto con los nombres de los procesos. - <code>-u</code> : Muestra el propietario (usuario) de los procesos. 	A diferencia de <code>ps</code> , muestra la relación padre-hijo entre los procesos, útil para visualizar cómo están relacionados jerárquicamente.
<code>killall</code>	Termina múltiples procesos que tienen el mismo nombre.	<ul style="list-style-type: none"> - <code>-9</code> : Mata todos los procesos con el nombre indicado. - <code>-I</code> : Ignora las diferencias entre mayúsculas y minúsculas al buscar procesos. - <code>-v</code> : Muestra más detalles sobre los procesos afectados. 	Más efectivo que <code>kill</code> cuando hay varios procesos del mismo nombre que deben terminarse a la vez. No requiere conocer los PIDs específicos.

<code>top</code>	Muestra en tiempo real los procesos activos y el uso de recursos del sistema (CPU, memoria, etc.).	<ul style="list-style-type: none"> - <code>-d</code> : Define el tiempo de refresco (en segundos). - <code>-u [usuario]</code> : Muestra solo los procesos de un usuario específico. - <code>-n</code> : Define cuántos ciclos de actualización debe hacer antes de detenerse. 	A diferencia de <code>ps</code> , que es estático, <code>top</code> proporciona una vista dinámica y actualizada en tiempo real. Es más útil para el monitoreo continuo.
<code>nice</code>	Cambia la prioridad de un proceso al iniciarlo (prioridad baja o alta).	<ul style="list-style-type: none"> - <code>-n [valor]</code> : Establece el valor de prioridad (<code>-20</code> es la mayor prioridad, <code>19</code> la más baja). - <code>-g [grupo]</code> : Cambia la prioridad de todos los procesos en un grupo. 	Controla la prioridad de un proceso cuando se inicia, mientras que el comando <code>renice</code> puede cambiar la prioridad de procesos ya en ejecución.

7) Otros comandos de Linux (Indique funcionalidad y parámetros):

a) ¿A qué hace referencia el concepto de empaquetar archivos en GNU/Linux?

El concepto de empaquetar archivos se refiere al proceso de agrupar varios archivos y directorios en un solo archivo comprimido o no comprimido. Esto se hace para facilitar la distribución, almacenamiento o transferencia de un conjunto de archivos. El archivo resultante, conocido como “paquete”, puede ser descomprimido o desempaquetado para recuperar los archivos originales.

b) Seleccione 4 archivos dentro de algún directorio al que tenga permiso y sume el tamaño de cada uno de estos archivos. Cree un archivo empaquetado conteniendo estos 4 archivos y compare los tamaños de los mismos. ¿Qué características nota?

```

leo@leo:~/Desktop/Ejemplo$ ls -l
total 12
-rw-rw-r-- 1 leo leo 10240 Sep 13 17:46 archivo1.txt
-rw-rw-r-- 1 leo leo 0 Sep 13 17:43 archivo2.txt
-rw-rw-r-- 1 leo leo 0 Sep 13 17:43 archivo3.txt
-rw-rw-r-- 1 leo leo 0 Sep 13 17:43 archivo4.txt
leo@leo:~/Desktop/Ejemplo$ du --bytes
10240 .
leo@leo:~/Desktop/Ejemplo$ tar cvf archivoEmpaquetado archivo1.txt archivo2.txt archivo3.txt archivo4.txt
archivo1.txt
archivo2.txt
archivo3.txt
archivo4.txt
leo@leo:~/Desktop/Ejemplo$ ls -l
total 32
-rw-rw-r-- 1 leo leo 10240 Sep 13 17:46 archivo1.txt
-rw-rw-r-- 1 leo leo 0 Sep 13 17:43 archivo2.txt
-rw-rw-r-- 1 leo leo 0 Sep 13 17:43 archivo3.txt
-rw-rw-r-- 1 leo leo 0 Sep 13 17:43 archivo4.txt
-rw-rw-r-- 1 leo leo 20480 Sep 13 17:50 archivoEmpaquetado
leo@leo:~/Desktop/Ejemplo$ du --bytes
30720 .
leo@leo:~/Desktop/Ejemplo$ 

```

- c) ¿Qué acciones debe llevar a cabo para comprimir 4 archivos en uno solo? Indique la secuencia de comandos ejecutados.

Comando	Explicación
<pre>tar -cvf archivos.tar archivo1.txt archivo2.txt archivo3.txt archivo4.txt</pre>	<p>Empaquetar: <code>tar</code> crea un archivo empaquetado (<code>archivos.tar</code>) que contiene los archivos listados.</p> <ul style="list-style-type: none"> - <code>-c</code>: Crea un nuevo archivo tar. - <code>-v</code>: Muestra el progreso (modo verbose). - <code>-f</code>: Especifica el nombre del archivo tar.
<pre>tar -czvf archivos.tar.gz archivo1.txt archivo2.txt archivo3.txt archivo4.txt</pre>	<p>Empaquetar y Comprimir: Crea un archivo tar comprimido (<code>archivos.tar.gz</code>).</p> <ul style="list-style-type: none"> - <code>-z</code>: Comprime el archivo tar usando gzip. - <code>-c</code>: Crea un nuevo archivo tar. - <code>-v</code>: Muestra el progreso. - <code>-f</code>: Especifica el nombre del archivo tar.
<pre>tar -cjvf archivos.tar.bz2 archivo1.txt archivo2.txt archivo3.txt archivo4.txt</pre>	<p>Empaquetar y Comprimir: Crea un archivo tar comprimido (<code>archivos.tar.bz2</code>).</p> <ul style="list-style-type: none"> - <code>-j</code>: Comprime el archivo tar usando bzip2. - <code>-c</code>: Crea un nuevo archivo tar. - <code>-v</code>: Muestra el progreso. - <code>-f</code>: Especifica el nombre del archivo tar.

d) ¿Pueden comprimirse un conjunto de archivos utilizando un único comando?

Si se desea empaquetar y comprimir un conjunto de archivos se puede utilizar:

- `tar -czvf archivo.tar archivo1 archivo2 archivo3 archivoN`
- `tar -cjvf archivo.tar archivo1 archivo2 archivo3 archivoN`
- `tar -cJvf archivo.tar archivo1 archivo2 archivo3 archivoN`

Nota: j utilizará gzip (.tar.gz), z usará bzip2 (.tar.bz2) y J utiliza xz (.tar.xz).

Si ya se disponen de archivos empaquetados .tar y se desean comprimir:

- `gzip archivo1.tar archivo2.tar archivo3.tar archivoN.tar`
- `bzip2 archivo1.tar archivo2.tar archivo3.tar archivoN.tar`

e) Investigue la funcionalidad de los siguientes comandos:

- `tar`
- `grep`
- `gzip`
- `zgrep`
- `wc`

Comando	Funcionalidad
<code>tar</code>	Utilizado para empaquetar y descomprimir archivos. Puede combinar múltiples archivos en un solo archivo tar (<code>.tar</code>) o extraer archivos de un archivo tar. También puede comprimir archivos usando opciones adicionales como <code>-z</code> para gzip o <code>-j</code> para bzip2.
<code>grep</code>	Busca texto en archivos usando expresiones regulares. Muestra las líneas que coinciden con el patrón especificado.
<code>gzip</code>	Comprime archivos utilizando el algoritmo de compresión gzip. Cambia la extensión del archivo a <code>.gz</code> . También puede descomprimir archivos gzip usando la opción <code>-d</code> .
<code>zgrep</code>	Similar a <code>grep</code> , pero diseñado para trabajar con archivos comprimidos en formato gzip. Permite buscar texto dentro de archivos <code>.gz</code> sin necesidad de descomprimirlos primero.
<code>wc</code>	Cuenta líneas, palabras y caracteres en un archivo o en la entrada estándar. Puede mostrar diferentes estadísticas usando opciones como <code>-l</code> para contar líneas, <code>-w</code> para contar palabras y <code>-c</code> para contar caracteres.

Para **descomprimir** un archivo **.tar.gz**: `gunzip archivo.tar.gz`

Para **descomprimir** un archivo **.tar.bz2**: `bunzip2 archivo.tar.bz2`

Para **descomprimir** un archivo **.tar.xz**: `unxz archivo.tar.xz`

Para **desempaquetar** un archivo **.tar**: `tar -xvf archivo.tar`

Para **descomprimir y desempaquetar** un archivo **.tar.***: `tar -x[j|J|z]vf archivo.tar.*`

Nota: [vf] son parámetros totalmente opcionales para ver la salida y comportamiento.