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**, **0**: 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.

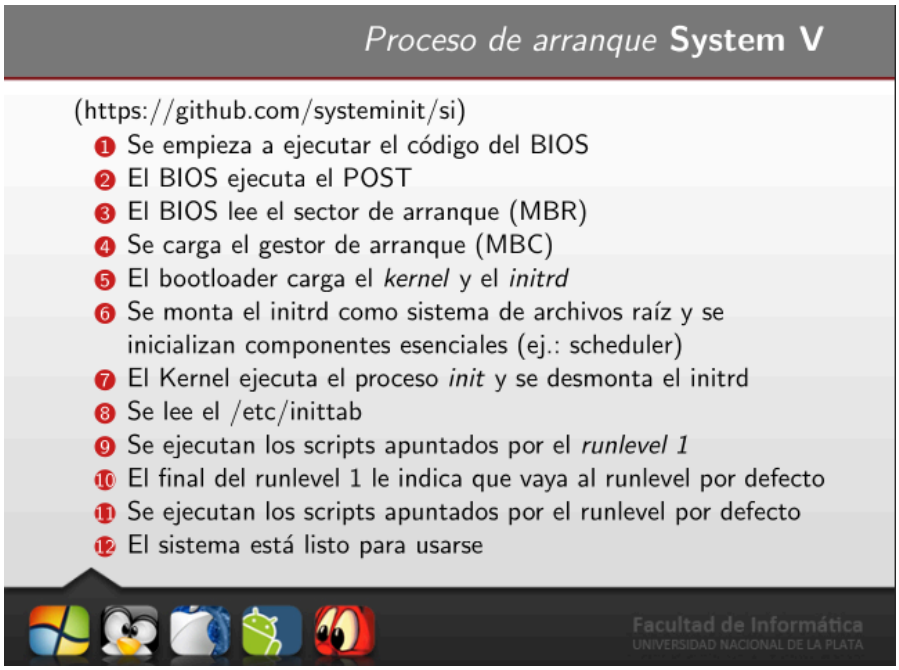
c) Nombre los comandos más comunes que se le pueden enviar al editor de textos vi.

<u>Comandos más comunes</u>:

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 caracter.
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.

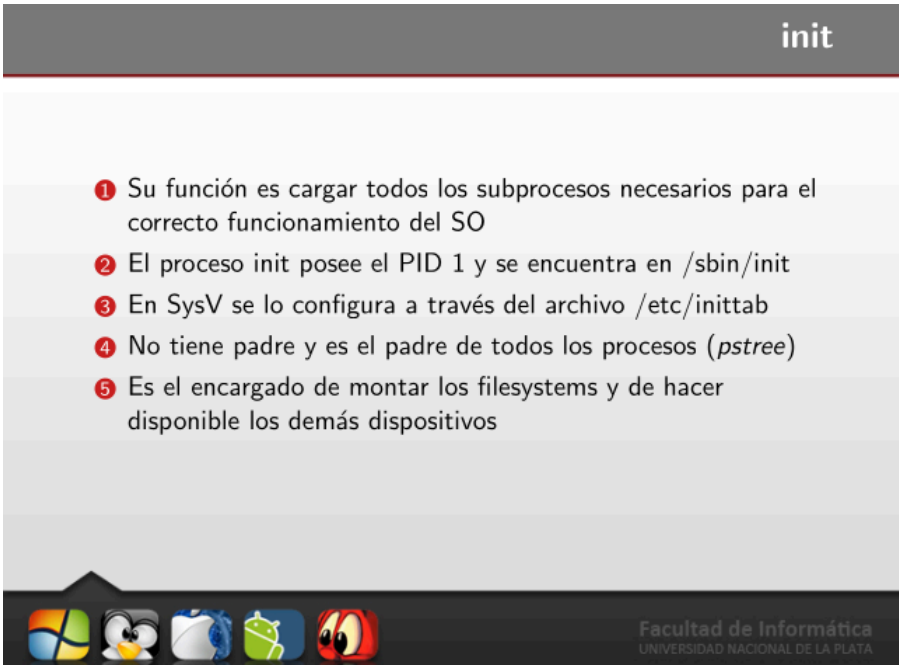
2) Proceso de Arranque SystemV (https://github.com/systeminnit/si):

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.



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

Lo ejecuta el Kernel.



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 soperte 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 -1 /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.

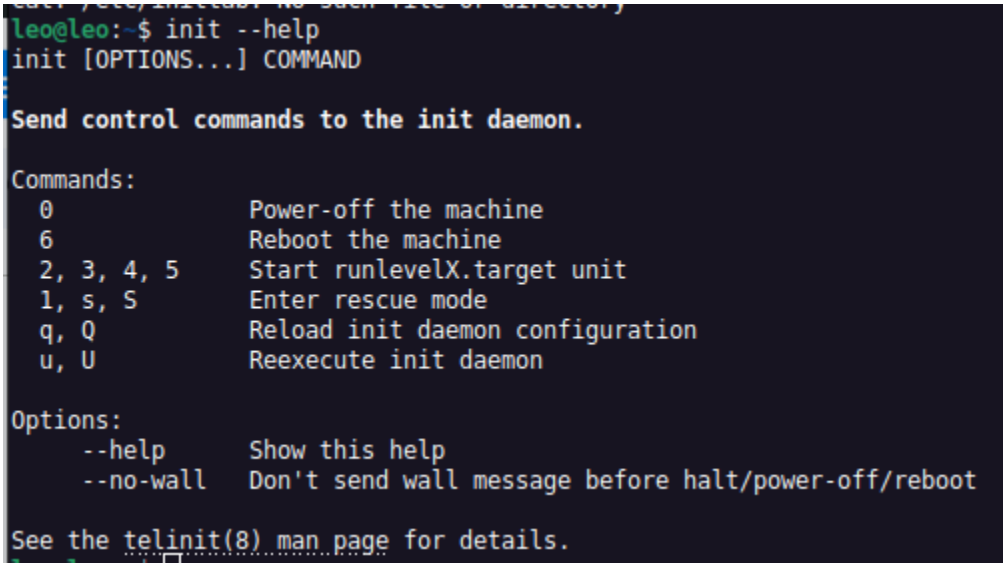
e) Archivo /etc/inittab. ¿Cuál es su funcionalidad? ¿Qué tipo de información se almacena en el? ¿Cuál es la estructura de la información que en él se almacena?

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**.



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.

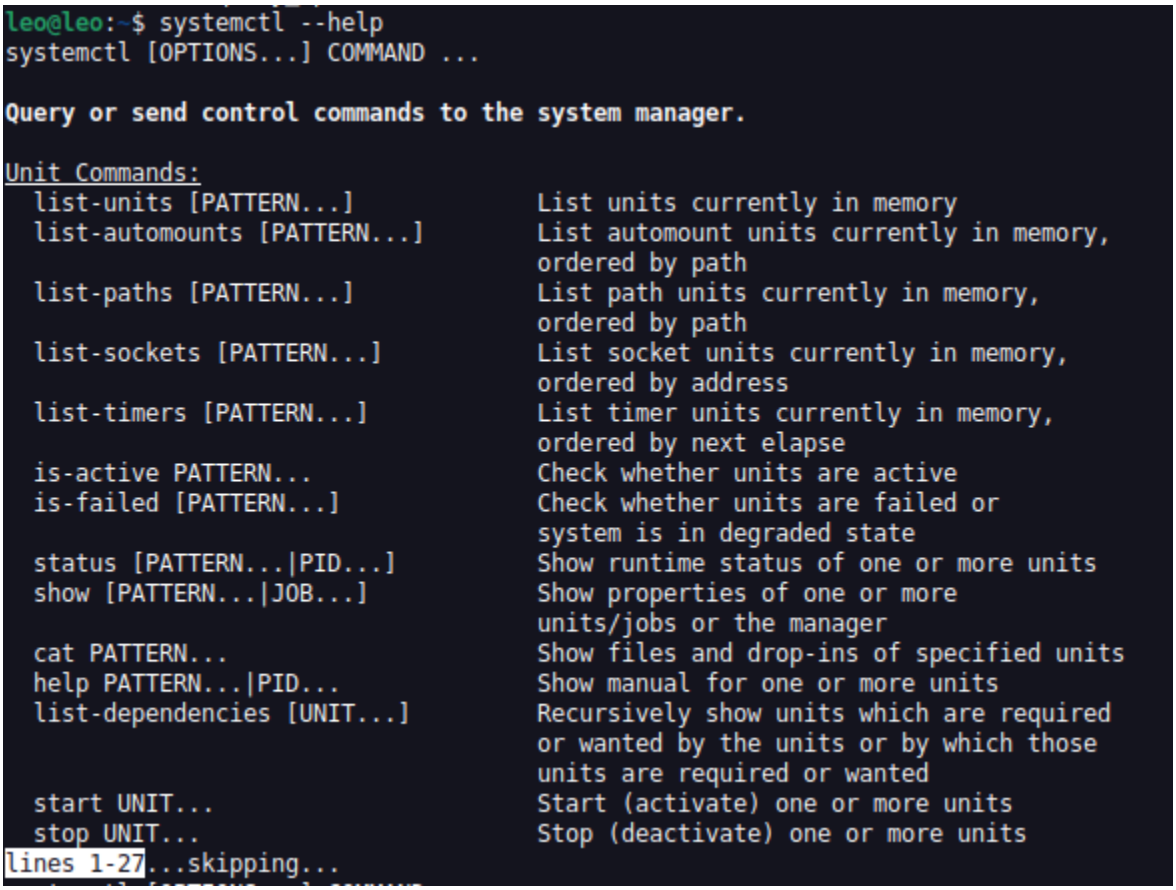
b) ¿A qué hace referencia el concepto de Unit en SystemD?

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?



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.

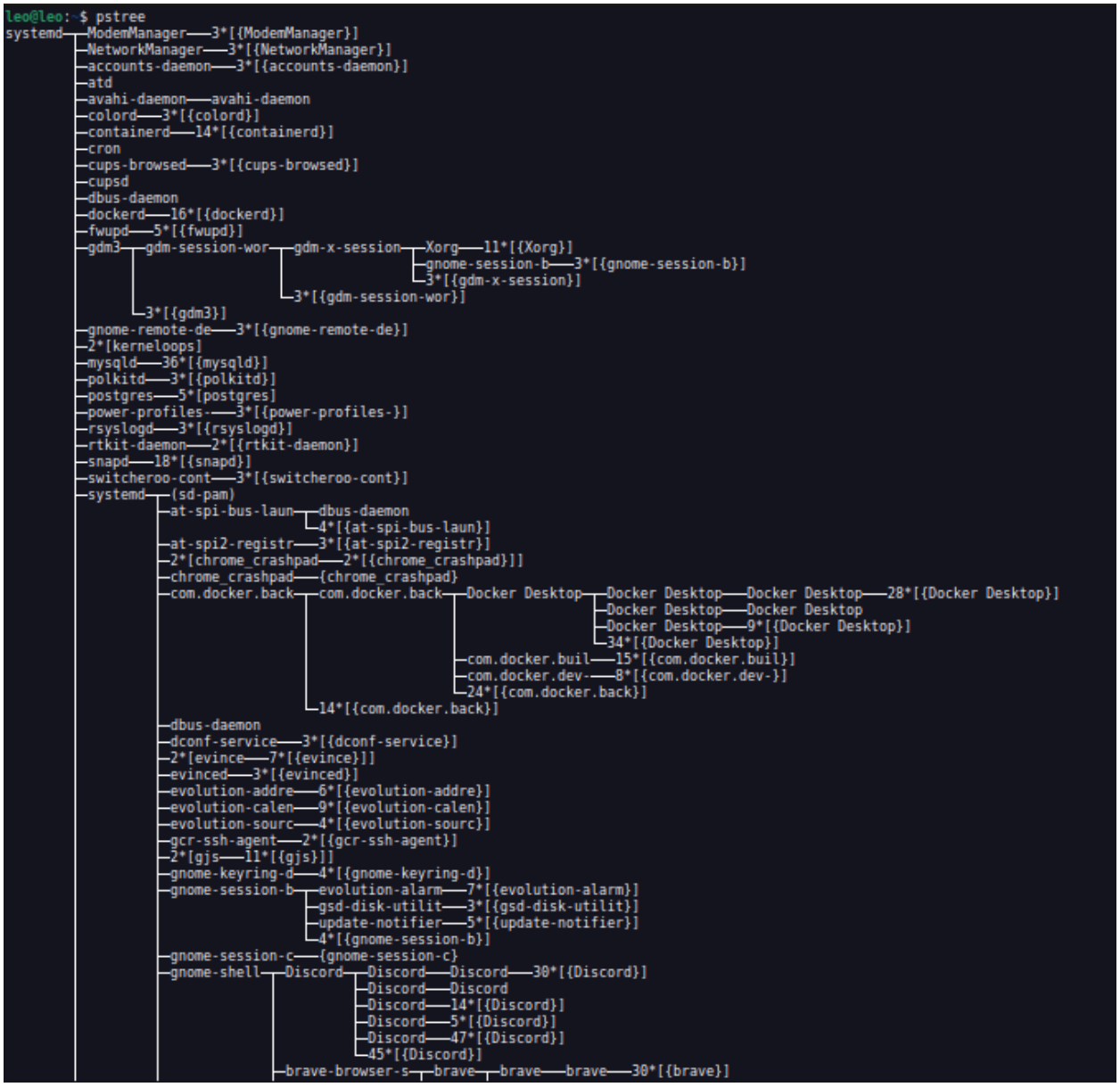
Ejemplos de Targets:

- 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.



4) Usuarios:

a) ¿Qué archivos son utilizados en un sistema GNU/Linux para guardar la información de los usuarios?
b) ¿A qué hacen referencia las siglas UID y GID? ¿Pueden coexistir UIDs iguales en un sistema GNU/Linux? Justifique.
c) ¿Qué es el usuario root? ¿Puede existir más de un usuario con este perfil en GNU/Linux? ¿Cuál es la UID del root?
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.

e) Investigue la funcionalidad y parámetros de los siguientes comandos:
    - useradd o adduser
    - usermod
    - userdel
    - su
    - groupadd
    - who
    - groupdel
    - passwd