Tema 3

1. Escribe un comando que nos permita ver el número de inodo de /etc/null.

```
lucia@lucia:/etc$ ls -i xml
132644 catalog      132646 polkitd.xml      132648 xml-core.xml
132645 catalog.old  132647 polkitd.xml.old  132649 xml-core.xml.old
```

2. Crea un fichero con algo de contenido (u3f) y un enlace físico a ese fichero (u3f_otro), comprueba si sus inodos son iguales o diferentes (¿por qué lo son?), comprueba el número de enlaces físicos del segundo fichero, luego borra el primer fichero y comprueba de nuevo el número de enlaces físicos del segundo fichero: ¿sigue existiendo su contenido? ¿Qué tipo de fichero es el enlace creado?

```
root@lucia:/home# ln u3f u3f_otro
root@lucia:/home# ls -i
393521 lucia  262155 u3f  262155 u3f_otro
root@lucia:/home#
```

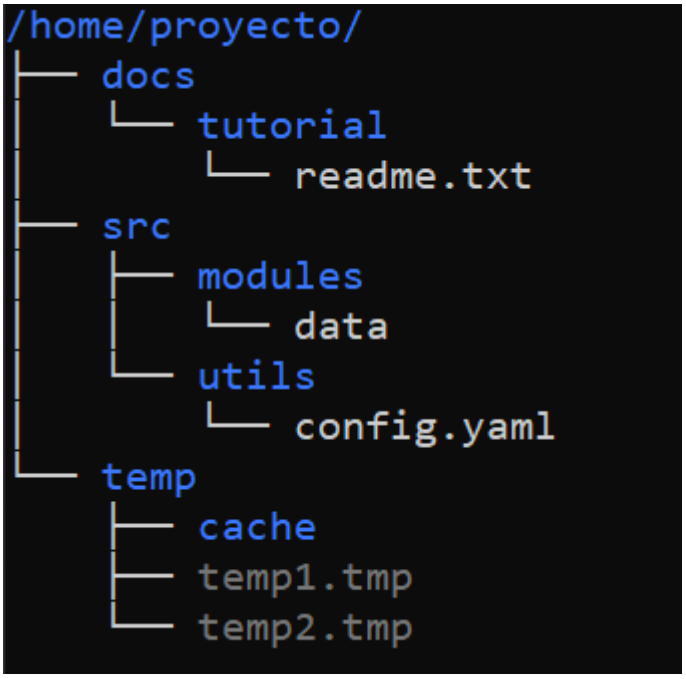
```
root@lucia:/home# stat u3f_otro
  File: u3f_otro
  Size: 2               Blocks: 8          IO Block: 4096   regular file
Device: 252,0   Inode: 262155      Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2024-12-09 09:23:08.084052651 +0000
Modify: 2024-12-09 09:23:08.084052651 +0000
Change: 2024-12-09 09:23:20.802277529 +0000
 Birth: 2024-12-09 09:23:08.084052651 +0000
root@lucia:/home# stat u3f
  File: u3f
  Size: 2               Blocks: 8          IO Block: 4096   regular file
Device: 252,0   Inode: 262155      Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2024-12-09 09:23:08.084052651 +0000
Modify: 2024-12-09 09:23:08.084052651 +0000
Change: 2024-12-09 09:23:20.802277529 +0000
 Birth: 2024-12-09 09:23:08.084052651 +0000
root@lucia:/home#
```

```
root@lucia:/home# rm u3f
root@lucia:/home# stat u3f_otro
  File: u3f_otro
  Size: 2               Blocks: 8          IO Block: 4096   regular file
Device: 252,0   Inode: 262155      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2024-12-09 09:23:08.084052651 +0000
Modify: 2024-12-09 09:23:08.084052651 +0000
Change: 2024-12-09 09:24:59.452021746 +0000
 Birth: 2024-12-09 09:23:08.084052651 +0000
root@lucia:/home#
```

3. Haz el ejercicio anterior completo pero con un enlace simbólico en lugar de físico: ¿qué ha cambiado?

```
root@lucia:/home# nano u3f
root@lucia:/home# ln -s u3f u3f_otro
root@lucia:/home# ls -i
393521 lucia  262155 u3f  262154 u3f_otro
root@lucia:/home# stat u3f_otro
  File: u3f_otro -> u3f
  Size: 3               Blocks: 0          IO Block: 4096   symbolic link
Device: 252,0   Inode: 262154      Links: 1
Access: (0777/lrwxrwxrwx)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2024-12-09 09:27:27.741643678 +0000
Modify: 2024-12-09 09:27:23.616570743 +0000
Change: 2024-12-09 09:27:23.616570743 +0000
 Birth: 2024-12-09 09:27:23.616570743 +0000
root@lucia:/home# stat u3f
  File: u3f
  Size: 7               Blocks: 8          IO Block: 4096   regular file
Device: 252,0   Inode: 262155      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2024-12-09 09:27:17.829468420 +0000
Modify: 2024-12-09 09:27:17.829468420 +0000
Change: 2024-12-09 09:27:17.829468420 +0000
 Birth: 2024-12-09 09:27:17.829468420 +0000
root@lucia:/home# rm u3f
root@lucia:/home# stat u3f_otro
  File: u3f_otro -> u3f
  Size: 3               Blocks: 0          IO Block: 4096   symbolic link
Device: 252,0   Inode: 262154      Links: 1
Access: (0777/lrwxrwxrwx)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2024-12-09 09:27:27.741643678 +0000
Modify: 2024-12-09 09:27:23.616570743 +0000
Change: 2024-12-09 09:27:23.616570743 +0000
 Birth: 2024-12-09 09:27:23.616570743 +0000
```

4. Crea el siguiente árbol de directorios dentro de tu directorio personal (se tomará como base para siguientes ejercicios) (los nombres terminados en / son directorios):
    1.
```
proyecto/
├── src/
│   ├── utils/
│   │   └── config.yaml
│   └── modules/
│       └── data
│
 ├── docs/
│   └── tutorial/
│       └── readme.txt
└── temp/
     ├── temp1.tmp
     ├── temp2.tmp
cd       └── cache/
```



5. Pon algunas líneas de contenido a los archivos creados en el anterior ejercicio excepto a `proyecto/src/modules/data`. Usa redirecciones, no editores interactivos (`nano`).

```
root@lucia:/home/proyecto/docs/tutorial# echo "Readme" >> readme.txt
root@lucia:/home/proyecto/docs/tutorial# cat readme.txt
Readme
root@lucia:/home/proyecto/docs/tutorial#
```

```
root@lucia:/home/proyecto/src/utils# echo "Yaml" >> config.yaml
root@lucia:/home/proyecto/src/utils# cat config.yaml
Yaml
root@lucia:/home/proyecto/src/utils#
```

6. Con una sola línea (aunque se permiten redirecciones y tuberías), crea el archivo `proyecto/docs/tutorial/readme.pdf` con 20KB de caracteres aleatorios.

```
lucia@lucia:/home$ sudo find proyecto/ -type d -empty -exec touch {}/prueba.txt \;
```

7. Comprime el directorio `proyecto/` en un fichero `proyecto.tar.gz` (ver el manual del comando `tar`, no visto en clase, para empaquetar y comprimir en GZIP). Prueba también a descomprimirlo (esto es útil por si se altera y se desea recuperar el original).
    1. Sería útil tener una copia de `proyecto/`, por lo que si no deseas comprimirlo, cópialo entero como `proyecto.backup/`.

8. Usa el comando `find` (en una sola línea) de las siguientes formas:
    1. Busca todos los ficheros vacíos dentro de `/etc`.

```
lucia@lucia:~$ find /etc/ -empty
```

    2. Busca todos los ficheros modificados en los últimos 30 minutos dentro de `/var`.

```
find /var/ -mmin -30
```

    3. Busca todos los ficheros regulares en todo el sistema que pesen más de 100MB y menos de 200MB.

```
sudo find /etc/ -size +100M -size -200M
```

    4. Cuenta todos los ficheros regulares del sistema que acaban en `.yaml` o en `.yml`.

```
lucia@lucia:/home$ find -type f | grep -c '\.yaml$'
```

    5. En `proyecto/`, borra todos los ficheros regulares vacíos.

```
lucia@lucia:/home$ find -type f -size 0 proyecto/ | rm
```

    6. En `proyecto/`, crea una copia de seguridad de cada fichero regular en su mismo directorio (llamándose igual pero con el sufijo `.backup`).

```
lucia@lucia:/home/proyecto/docs/tutorial$ sudo cp holaaa holaaa.backup
```

    7. En `proyecto/`, busca todos los directorios vacíos y crea en ellos un fichero llamado `algo`.

```
sudo find proyecto/ -type d -empty -exec touch {}/prueba.txt \;
```

9. Crea un disco duro virtual de 10GB, busca su nombre en `/dev` y haz lo siguiente en él:
    1. Crea una tabla de particiones GPT con 3 particiones: una de 100MB, otra de 2GB y otra con los 7.9GB restantes (aproximadamente).

    2. Formatea las tres particiones como NTFS, FAT32 y ext4 respectivamente.

    3. Monta la primera partición en el directorio `/mnt/fstemp`.
    4. Las particiones 2 y 3 añádelas (pero usando `UUID=`) al fichero `/etc/fstab` para que se monten en `/mnt/fs2` y `/mnt/fs3` respectivamente y con las configuraciones `defaults`, `user` y `noauto`. Al hacer `mount /mnt/fs1` y `mount /mnt/fs2` deberían montarse correctamente. Comprobar con `lsblk`.

10. Cuenta las líneas de `/etc/fstab` que son comentarios (es decir, que empiezan por `#`).

```
lucia@lucia:/$ grep -c '^#' /etc/fstab
```

11. Muestra las 10 últimas líneas de `/var/log/syslog` junto con su número de línea e inviértelas (que se muestre de la última a la primera línea).

```
lucia@lucia:/$ tail -n 10 /var/log/syslog | tac
```

12. Ordena el fichero `/etc/passwd` por UID (4ª columna) descendente, quedándote solo con los nombres de usuario (1ª columna).

```
lucia@lucia:/$ cut -d : -f  4 /etc/passwd | sort -r
```

13. Haz una búsqueda con `find` de todo lo que hay en `/usr/bin/local`, y mediante `egrep` (es igual que `grep` pero usa expresiones regulares extendidas) quédate solo con el nombre (o lo que es lo mismo, quita toda la ruta anterior al nombre).

14. De todos los usuarios del sistema (`/etc/passwd`), extrae sus shells (última columna), ordénalas y, mediante el comando `uniq` (ver el manual), cuenta cuántas ocurrencias hay de las distintas shells (por ejemplo, que diga que hay 4 `/bin/bash`, 1 `/bin/sh`, etc.).

Tema 4

15. Crea con `useradd` un nuevo usuario `granny` con su propio grupo principal (`granny`). Debe crearse automáticamente su directorio personal.

```
lucia@lucia:/$ sudo useradd -m granny
lucia@lucia:/$ id granny
uid=1001(granny) gid=1001(granny) groups=1001(granny)
```

16. Ponle una contraseña a `granny`.

```
lucia@lucia:/$ sudo passwd granny
New password:
Retype new password:
passwd: password updated successfully
```

17. Añade a `granny` primero al grupo `users` y luego al grupo `sudo`, en dos veces (sin mencionar a ambos en un solo comando).

```
lucia@lucia:/$ sudo usermod -aG users granny
lucia@lucia:/$ sudo usermod -aG sudo granny
```

18. Con `usermod`, cámbiale a `granny` la contraseña por la misma contraseña, pero hasheada con el algoritmo SHA512.

```
lucia@lucia:/$ openssl passwd -6 granny
$6$Woj/M.rmc9j7gLES$jGA700p60nkNxT7Dza9.ygvWxYeFre/XubzXgjKa.t5DESeCPq/kuNeKe
```

19. Extrae todas las contraseñas cifradas de los usuarios (sin otra información adicional), pero solo las que empiezan por `$`, una letra o número y otro `$` (ten en cuenta que el carácter `$` tiene un significado especial en las expresiones regulares).

```
lucia@lucia:/$ sudo cat /etc/shadow | grep '$[a-z A-Z 0-9]'
```

20. Abre una subshell con `granny` usando `su`. Después, abre otra en otra pestaña o terminal usando `su`, pero esta vez debe abrir una shell con inicio de sesión. ¿Notas alguna diferencia? ¿Qué tipo de shell se ha abierto en ambos casos?
21. Haz que el usuario `granny` se vea forzado a cambiar la contraseña cada 45 días, que avise del cambio de contraseña los últimos 10, y que caduque su cuenta el 31 de julio de 2025. ¿Qué archivo ha cambiado al ejecutar el/los comandos?
22. Crea el fichero `fich1` y asígnale permisos de lectura y escritura solo para el usuario actual (nada para los demás).
23. Crea un directorio llamado `/var/bk` cuyos usuario y grupo propietarios sean `backup` (ambos ya existen). Asigna todos los permisos al usuario, lectura y ejecución a grupo y solo ejecución a otros.
24. Cambia la máscara de permisos para que cualquier fichero nuevo tenga permisos `rw-r-----` y cualquier directorio nuevo tenga permisos `rwxr-x---`.
25. Crea un script (`script1.sh`), pon en él algún comando que imprima el nombre del usuario real actual y otro que imprima el nombre del usuario efectivo actual por la salida estándar y cambia los permisos del archivo para que solo pueda leerlo el usuario y el grupo pero pueda ejecutarlo cualquiera, y para añadir el bit SUID (para el permiso usa el modo simbólico, no el octal).

26. Copia el script anterior en `/tmp`, ejecútalo como `granny` y al propietario del `script`, y observa la salida: ¿quién es el usuario real y el efectivo?
27. El siguiente código C se limita a dormir un número de segundos (el número que se le pasa como argumento o 60 si se ejecuta sin argumento). Cópialo en el fichero `csleep.c` y compílalo (`gcc csleep.c -o csleep`):

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char* argv[]) {
  int secs;
  if (argc == 1)
    secs = 60;
  else if (argc == 2)
    secs = atoi(argv[1]);
  else {
    fprintf(stderr, "USO: csleep [SEGS]\n");
    return 1;
  }
  printf("Inicio de la cuenta atrás (%d segundos)\n", secs);
  sleep(secs);
  printf("Fin de la cuenta atrás (%d segundos)\n", secs);
  return 0;
}
```

28. Deja ejecutando `csleep 300` desde tu usuario en segundo plano, y también desde `granny` en segundo plano. Muestra el usuario real y el usuario efectivo de ambos procesos.
29. Activa el bit Set UID en el ejecutable `csleep`. Vuelve a ejecutarlo en segundo plano tanto desde el usuario que lo creó como desde `granny`. Muestra el PID, PPID, el usuario y grupo real y efectivo, la terminal asociada y el comando completo en ambos casos con `ps`. ¿Qué ha hecho el bit SUID? Crea un alias de `ps` llamado `pso` que incluya todas esas columnas.
30. Crea una variable de shell (`var1`) y otra de entorno (`var2`) con valores diferentes e imprime su valor en el shell actual: ¿ambas son visibles?
31. Crea un script que imprima el nombre de ambas variables y ejecútalo: ¿ambas son visibles? ¿Por qué?
32. Crea, de alguna manera, un proceso zombie.
33. Lanza en primer plano un proceso que espere 5 minutos. Al lanzarlo, detenlo y reanúdalo en segundo plano.
34. Vuelve a lanzar un proceso idéntico, pero esta vez, al lanzarlo, abre otro terminal o sesión SSH, busca el PID del proceso y realiza las acciones de detenerlo y reanudarlo mediante señales con `kill`.
35. Ejecuta una actualización del sistema en segundo plano, sin intervención del usuario y descartando la salida de error.
36. Programa mandar un mensaje al log del sistema (comando `logger`) con una felicitación de año nuevo justo al empezar 2025.
37. Programa mandar un mensaje al log del sistema con una felicitación de feliz lunes cada primer lunes de mes (a las 8:00 AM).

Su realización es voluntaria, por lo que no se hará ninguna entrega, pero si tenéis cualquier duda o comentario, podéis comentármelo en clase.

- Algunos ejercicios requieren más de un comando, por lo que la respuesta debe incluirles a todos.
- Algunos ejercicios tienen múltiples posibles formas de resolverse.

- La dificultad es diversa y los ejercicios más difíciles (en naranja o rojo) son opcionales. Si se desea realizarlos, se recomienda encarecidamente el uso del manual en lugar de búsquedas en internet o IAs.
- En el examen caerá una pregunta de dificultad alta y la mayoría serán de dificultad media y baja.