

# Manual

- [Comando grep](#)
- [Comando nano](#)
- [Comando sed](#)
- [Comando for](#)

# Comando grep

El comando `grep` es una herramienta muy utilizada en sistemas Linux para buscar cadenas de texto o patrones específicos dentro de archivos o salidas de otros comandos. Su función principal es filtrar líneas que coinciden con el patrón de búsqueda proporcionado.

## Comando ejemplo:

```
grep tcp /etc/services | awk '{print $1}' | sort | less
```

## Desglose:

1. `grep tcp /etc/services`:
  - `grep` busca líneas que contienen el texto `tcp` en el archivo `/etc/services`.
  - El archivo `/etc/services` es una lista de servicios de red y sus puertos asociados, incluyendo si usan **TCP** o **UDP**.
  - Esto filtra todas las líneas donde aparece "tcp", que corresponde a servicios que usan el protocolo TCP.
2. `awk '{print $1}'`:
  - `awk` es una herramienta para procesar texto. En este caso, toma la salida de `grep` y extrae la **primera columna** (campo), que en el archivo `/etc/services` es el **nombre del servicio**.
  - Por ejemplo, si una línea contiene `http 80/tcp`, el `awk '{print $1}'` extraerá solo `http`.
3. `sort`:
  - `sort` ordena alfabéticamente los nombres de los servicios extraídos por `awk`. Esto organiza los resultados para que se muestren en orden.
4. `less`:
  - `less` es un paginador que permite visualizar la salida de manera interactiva. Como la lista de servicios puede ser larga, `less` facilita la navegación por la salida paginada (arriba/abajo con las teclas de flecha o espacio).

# Comando nano

## Abrir un archivo:

Para abrir un archivo en Nano, escribe `nano -u nombre_del_archivo.txt` en la terminal. Si el archivo no existe, Nano lo creará cuando lo guardes.

## Comandos básicos:

- **Guardar:** Presiona **Ctrl + O** para guardar los cambios.
- **Salir:** Presiona **Ctrl + X** para salir de Nano.
- **Ayuda:** Presiona **Ctrl + G** para abrir la guía de ayuda.

## Editar texto:

- **Cortar:** Usa **Ctrl + K** para cortar texto (piensa en "K" de "cut").
- **Pegar:** Usa **Ctrl + U** para pegar texto (piensa en "U" de "uncut").
- **Copiar:** Marca el texto con **Ctrl + 6**, luego presiona **Alt + 6** para copiarlo.

## Deshacer/Rehacer:

Si iniciaste Nano con la opción `-u`, puedes deshacer con **Alt + U** y rehacer con **Alt + E**.

## Buscar y reemplazar:

- **Buscar:** Presiona **Ctrl + W** (piensa en "Where") para buscar texto.
- **Reemplazar:** Presiona **Alt + R** para buscar y reemplazar texto.

## Números de línea:

Presiona **Alt + C** para activar o desactivar los números de línea.

# Comando sed

`sed` es un editor de flujo utilizado para editar texto a medida que se procesa. Aquí están sus usos principales:

- **Print** (Imprimir): Muestra la salida basada en un patrón.
- **Delete** (Eliminar): Borra el texto que coincide con un patrón.
- **Substitute** (Sustituir): Reemplaza un patrón con otro.

Puede manejar expresiones regulares básicas y extendidas, lo que lo convierte en una herramienta poderosa para la manipulación de texto en Linux.

Considere el archivo anexo.

A estos numero de telefonos, podemos agregarles codigo de pais, parentesis y guion para reformarlo apropiadamente.

Si `telefonos.txt` tiene una línea como: `1123456789` después de ejecutar el comando, se transformaría en: `+55(11)2345-6789`

```
sed -E 's/([0-9]{2})([0-9]{4})([0-9]{4})/+55(\1)\2-\3/' telefonos.txt
```

Vamos a desglosarlo:

1. `sed -E`:
  - La opción `-E` habilita expresiones regulares extendidas, lo que permite patrones de coincidencia más potentes sin la necesidad de escapar caracteres como `+` o `?`.
2. `s/([0-9]{2})([0-9]{4})([0-9]{4})/`:
  - La estructura `s/.../.../` es la sintaxis para sustitución en `sed`, donde el patrón a la izquierda es reemplazado por el formato especificado a la derecha.
  - `([0-9]{2})`: Coincide con los dos primeros dígitos, que representan el código de área del estado.
  - `([0-9]{4})`: Coincide con los siguientes cuatro dígitos, que representan el prefijo.
  - `([0-9]{4})`: Coincide con los últimos cuatro dígitos, que representan el número final.Cada `([0-9]{...})` captura el grupo de números correspondiente (1, 2 y 3) para usarlos en la parte de reemplazo. (esto es secuencial y por lo que vi no se puede reemplazar)
3. **Patrón de reemplazo** `+55(\1)\2-\3`:
  - `+55`: Agrega el prefijo de código de país al inicio.
  - `(\1)`: Coloca el primer grupo (el código de área) entre paréntesis.
  - `\2`: Inserta el segundo grupo (el prefijo) tal como está.
  - `-`: Coloca un guion entre el segundo y el tercer grupo.
  - `\3`: Inserta el tercer grupo (el número final).

4. `telefonos.txt`:

- Es el archivo de entrada que contiene los números de teléfono. `sed` aplica el formato especificado a cada línea del archivo.

# Comando for

Estaba bajando muchos archivos .wav con su licencia .pdf. Y estos los estaba guardando en una carpeta.

Resulta que estos archivos viene con un nombre que usa esta expresion "

Eso es muy incomodo e innecesariamente repetitivo, asi que usando el comando `for` pude renombrar cientos de archivos para que solo contengan la descripcion y borrar el resto.

```
for file in *.wav *.pdf; do
    if [[ "$file" =~ ^[0-9]{8}__ ]]; then
        mv "$file" "${file:10}"
    fi
done
```

## Explicación

- `for file in *.wav *.pdf; do`: Itera sobre todos los archivos `.wav` y `.pdf` en el directorio actual.
- `if [[ "$file" =~ ^[0-9]{8}__ ]]; then`: Comprueba si el nombre del archivo comienza con 8 dígitos seguidos de `__`.
- `mv "$file" "${file:10}"`: Usa `mv` para renombrar el archivo, quitando los primeros 10 caracteres (`${file:10}`).