

# Contenido

1. Entrada / Salida
2. Comandos Avanzados
3. Redirección
4. Tuberías

# 1. Entrada / Salida

- **Entrada** de un comando: dispositivo del que recibe datos un comando.
- **Entrada estándar:** el teclado.
  - Cuando un comando tenga entrada estándar significa que los datos sobre los que opera serán introducidos desde teclado.
- **Salida** de un comando: dispositivo hacia el que envía los datos generados por un comando durante su ejecución.
- **Salida estándar:** la pantalla
  - Cuando un comando tenga salida estándar significa que los datos o información resultado de su ejecución será mostrada por pantalla.
- **Salida de errores** de un comando: dispositivo hacia el que envía el comando los mensajes de error en caso de que se generen.
- **Salida estándar de errores:** la pantalla
- No todos los comandos tienen entrada y/o salida.
- Todos los comandos tienen salida estándar de errores.

## 2. Comandos Avanzados (I)

- Comando **grep**: busca cadenas de caracteres en la entrada mostrando las líneas donde aparecen esas cadenas por la salida estándar.
  - Sintaxis básica: `grep [opciones] patrón [fichero/s de texto]`
    - patrón: cadena de caracteres a buscar. Mediante el uso de expresiones regulares se pueden crear patrones que permiten búsquedas complejas.
    - El/los fichero/s son opcionales. Si se incluyen la búsqueda se realiza sobre el/ellos, si no el comando tendrá entrada estándar.
    - Ej. `grep hola fich1 fich2` → busca la cadena hola en los ficheros fich1 y fich2.
- Comando **sort**: muestra por la salida estándar las líneas de la entrada.
  - Sintaxis básica: `sort [opciones] [fichero/s de texto]`
    - Si no incluyen los ficheros tendrá entrada estándar.
    - Ej. `sort fich1 fich2` → muestra por pantalla los ficheros fich1 y fich2 ordenados.
- Comando **uniq**: elimina las líneas repetidas de la entrada, teniendo que estar ésta ordenada, mostrando el resultado por la salida estándar.
  - Sintaxis básica: `uniq [opciones] [fichero/s de texto]`
    - Si no incluyen los ficheros tendrá entrada estándar.
    - Ej. `uniq fich1.txt`
- Comandos **head/tail**: muestran las primeras/últimas líneas de la entrada.
  - Sintaxis básica: `head/tail [opciones] [fichero/s de texto]`
    - Si no incluyen los ficheros tendrán entrada estándar.

## 2. Comandos Avanzados (II)

- Comando **more**: muestra por la salida estándar la información de la entrada, parando cuando se llene la pantalla.
  - Sintaxis básica: `more [opciones] [fichero/s de texto]`
    - Si no incluyen los ficheros tendrá entrada estándar.
- Comando **paste**: concatena las líneas de la entrada separándolas con tabuladores. El resultado se muestra por la salida estándar.
  - Sintaxis básica: `paste [opciones] [fichero/s de texto]`
    - Si no incluyen los ficheros o en esa parte se escribe -, tendrá entrada estándar.
- Comando **wc**: cuenta el número de líneas, palabras y caracteres de la entrada, mostrando la información por la salida estándar.
  - Sintaxis básica: `wc [opciones] [fichero/s de texto]`
    - Si no incluyen los ficheros tendrá entrada estándar.
- Comando **tr**: sustituye, trunca y/o borra caracteres de la entrada estándar.
  - Sintaxis básica sustituir: `tr char1 char2` → Sustituye *char1* por *char2*
  - Sintaxis básica eliminar: `tr -d char1` → Elimina *char1*
  - En *char1* o *char2* se pueden especificar conjuntos de caracteres.

## 2. Comandos Avanzados (III)

- Comando **cut**: selecciona columna/s de la entrada, mostrando el resultado por la salida estándar.
  - Sintaxis básica: `cut opción [fichero/s de texto]`
    - Si no incluyen los ficheros tendrá entrada estándar.
  - Opciones más importantes:
    - `-c rango_caracteres` → selecciona los caracteres ubicados en las posiciones *rango\_caracteres* de cada línea de texto.
      - Ej. 1: `cut -c 4 fich` → muestra por pantalla el caracter de la posición 4 de cada línea.
      - Ej. 2: `cut -c 5,8,10 fich` → muestra por pantalla los caracteres 5, 8 y 10 de cada línea.
      - Ej. 3: `cut -c 4-8 fich` → muestra por pantalla los caracteres de las posiciones 4 a 8 de cada línea.
    - `-f rango_campos` → selecciona los campos ubicados en las posiciones *rango\_campos* de cada línea de texto. Campo: cadena de caracteres situados entre dos caracteres separador. El carácter separador por defecto es el tabulador. Se puede cambiar el carácter separador usando la opción `-d`.
      - Ej. 1: `cut -f 4 fich` → muestra por pantalla el campo 4 de cada línea.
      - Ej. 2: `cut -f 5,8,10 fich` → muestra por pantalla los campos 5, 8 y 10 de cada línea.
      - Ej. 3: `cut -f 4-8 fich` → muestra por pantalla los campos 4 a 8 de cada línea.
      - Ej. 3: `cut -f 4-8 -d " " fich` → muestra por pantalla los campos 4 a 8 de cada línea, usando como carácter separador de campos el espacio, en vez del tabulador.

## 3. Redirección (I)

- UNIX permite cambiar el dispositivo de entrada, salida y salida de errores estándar por otros distintos, generalmente un fichero.
- Esto es posible porque UNIX trata los dispositivos de entrada y salida como si fueran ficheros normales (ver directorio **/dev**).
- Redirección de **entrada**:
  - Sintaxis: comando [opciones] [argumentos] < **fichero/s**
  - **Condición necesaria: el comando debe tener entrada estándar.**
  - Ej.
    - `tr a A < fich1` → Cambia el carácter “a” por el “A” en los ficheros fich1, mostrando el resultado por la salida estándar
  - Este tipo de redirección es menos usada que las siguientes, ya que casi todos los comandos admiten ficheros como argumentos.

## 3. Redirección (II)

- Redirección de **salida**:
  - Sintaxis 1: comando [opciones] [argumentos] > **fichero** → La salida del comando se almacena en *fichero*. Si éste no existía se crea, si ya existía se sustituye su contenido anterior (que desaparece) por la salida del comando. **¡Importante!** si el fichero existe lo primero que se hace es “vaciar” este fichero y luego se ejecuta el comando.
  - Sintaxis 2: comando [opciones] [argumentos] >> **fichero** → La salida del comando se almacena en *fichero*. Si éste no existía se crea, si ya existía la salida del comando se añade a continuación del contenido anterior del fichero.
  - Ej.
    - cat fich1 fich2 > fich3 → Concatena el contenido de los ficheros fich1 y fich2 en el fichero fich3.
    - sort fich1 >fich2 → fich2 contendrá el contenido de fich1 ordenado.
    - cat fich1 >> fich2 → el contenido de fich1 se añade a continuación del de fich2.
    - echo “Esto es una prueba” > fich → se crea fich con la frase “Esto es una prueba”.
    - sort fich > fich → **Fallo**: al acabar la ejecución fich está vacío.

## 3. Redirección (III)

- Redirección de **salida de errores**:
  - Sintaxis 1: comando [opciones] [argumentos] **2> fichero** → Los mensajes de error del comando, en caso de que se produzcan, se almacenan en *fichero*. Si éste no existía se crea, si ya existía se sustituye su contenido anterior (que desaparece) por la salida de errores del comando.
  - Sintaxis 2: comando [opciones] [argumentos] **2>> fichero** → Los mensajes de error del comando, en caso de que se produzcan, se almacenan en *fichero*. Si éste no existía se crea, si ya existía la salida de errores del comando se añade a continuación del contenido anterior del fichero.
    - Ej.: `grep -w hola fich 2> errores`
- Se pueden mezclar distintos tipos de redirección en una misma línea. Ej.:
  - `tr [A-Z] [a-z] < fich1 > fich2 2>> fich_errores`
  - `sort fich1 fich2 >> fich3 2> errores`
  - `cut -f 2-6 -d “;” > fich 2> &1` → Las dos salidas van al mismo archivo: *fich*
  - `(cd; ls -dl a*; who am i) >> fich` → Es equivalente a:
 

```
cd >> fich
ls -dl a* >> fich
who am i >> fich
```

El comando `cd` sólo es válido dentro de paréntesis. Una vez acabada la ejecución “no nos habremos movido” de directorio.



## 4. Tuberías

- También llamadas “pipelines”, consiste en concatenar ejecuciones de comandos de manera que la salida de uno sea la entrada del siguiente, y así sucesivamente.
- Sintaxis: `comando1 | comando2 | comando3 ...`
  - El símbolo “|” es el usado en las tuberías como separador entre comandos.
  - Funcionamiento: la salida del comando1 es recogida por el comando2 que opera sobre ella generando una salida que es procesada por el comando3, y así sucesivamente hasta llegar al último comando de la tubería.
- **Condiciones necesarias:**
  - Un comando a la izquierda de una tubería (“comando |”) debe tener salida estándar.
  - Un comando a la derecha de una tubería (“| comando”) debe tener entrada estándar.
- Ej.
  - `sort fich1 fich2 | uniq | wc -l` → la salida será el número de líneas distintas que hay en fich1 y fich2
  - `tr M m < fich1 2> errores | grep “m[a-z]x” > fich_salida` → mezcla con redirección.
  - `ls -l | wc -l`