

# Flujos, Redirecciones y Pipes - Pipes (Canalizaciones)

## Pasar datos entre programas

Las Pipes (canalización) se usan para redireccionar la stdout de un programa hacia la la stdin de otro, o lo que es lo mismo utilizar la salida de un programa como entrada de otro, en principio no hay límite en el número de programas utilizados, veamos un ejemplo para genérico para comprender su funcionamiento

```
programa1 | programa2 | script3 | ..... | programaN
```

Ahora un ejemplo mas práctico, el cual utilizaremos para ver la línea de la salida de ifconfig (herramienta que nos devuelve la información de una tarjeta de red) en la que aparece la MAC de la misma, mediante grep

```
ifconfig eth0 | grep direcciónHW
```

la cual, suponiendo la existencia en nuestro sistema de una tarjeta de red eth0, nos devuelve algo parecido a:

```
eth0 Link encap:Ethernet direcciónHW 00:15:f2:1b:65:ec
```

Las Pipes pueden llegar a ser una herramienta muy potente, mucho mas de lo que parece con la primera impresión, tanto para seguimiento de errores, como en configuración del sistema, seguridad,... y todo aquello que la imaginación nos permita

## Pipes y Comandos - Comandos

Vamos a ver un ejemplo en el que utilizaremos la salida de un comando mediante una Pipes (|), para utilizarla con otro comando para realizar una tarea repetitiva, y así agilizar nuestro trabajo.

Voy a utilizar algunos comandos que no hemos visto, que veremos mas adelante en una sección dedicada a los comandos mas comunes, por lo que simplemente diré lo que hacen a forma de indicación. Si alguien tiene interés en ir investigando para conocer más a fondo los distintos comandos, os recuerdo que podéis utilizar `man (man comando)` o `info (info comando)` para ver todas las opciones y algunos ejemplos de los comandos que vayamos viendo.

Abriremos un terminal si estamos utilizando una distribución con entorno gráfico, que en estos tiempos será lo mas habitual.

1. Vamos al menú principal, en mi caso ubuntu, y buscamos:
  - menú principal->aplicaciones->accesorios->terminal
2. Creamos un directorio para realizar nuestros ejemplos, con seguridad para no modificar ningún archivo de nuestro sistema, de forma accidental, evito utilizar nombres con espacios y caracteres especiales:

- `mkdir ejemplos-1t`
- 3. Cambiamos nuestra ubicación dentro de nuestro nuevo directorio:
  - `cd ejemplos-1t`
- 4. Creamos unos cuantos ficheros, `touch` nos permite hacer esto:
  - `touch 1.111 1.112 1.113 1.121 1.131 1.141`
- 5. Comprobamos que efectivamente se han creado 6 archivos:
  - `ls`
- 6. Como podéis ver hemos creado 6 archivos, ahora vamos a borrar solo los que acaban en "1", esto lo podríamos hacer simplemente con el comando `rm` pero lo vamos a complicar solamente con el fin de ver como podemos utilizar pipes, desde luego que para esto no es necesario utilizar pipes, pero con algo mas de conocimientos y de imaginación seguro que en el futuro se nos ocurren muchos casos en el que utilizarlas:
- 7. Listamos solamente los archivos acabados en "1", y vemos que hay 4:
  - `ls *1`
- 8. Eliminamos los archivos utilizando un pipe (`|`):
  - `ls *1 | xargs rm`

Con el pipe le pasamos a `rm` mediante `xargs` cada resultado devuelto por `ls *1`, por lo que se construyen 4 comandos `rm` y el nombre de cada fichero que cumple el patrón `*1`

En este caso en concreto podríamos haber realizado lo mismo con:

```
rm *1
```

Pero como indiqué antes hemos utilizado un pipe para ilustrar un ejemplo sencillo

Francisco Javier López Torrijos  
Analista Sistemas Informáticos de Gestión  
Diseño y Desarrollo Web