Es muy importante tener en cuenta que las redirecciones de salida se realizan antes de ejecutar las órdenes, por lo que si hacemos algo como lo que sigue:

```
$ sort <ps.out | uniq >ps.out
$ cat ps.out
$
```

acabaremos con un fichero de salida vacío.

Cuestión: ¿ Por qué?

Sort recibe lo que hay en ps.out, lo ordena y lo saca a ps.out sobreescribiendo, después, como la salida estándar está dirigida al fichero y el pipe conecta la salida estándar de un comando con la entrada estándar de otro, a uniq no le llega nada y saca el resultado (nada) al fichero ps.out, que finalmente queda vacío.

Ejemplos:

```
$ ls /bin/bash && echo Existe ; echo $?
/bin/bash
Existe
0
$ ls /bin/noexiste && echo existe ; echo $?
ls: no se puede acceder a /bin/noexiste:
No existe el fichero o el directorio
2
$ ls /bin/bash || echo No existe ; echo $?
/bin/bash
0
$ ls /bin/noexiste || echo No existe ; echo $?
ls: no se puede acceder a /bin/noexiste:
No existe el fichero o el directorio
No existe
0
```

Cuestión: ¿ Por qué en el último ejemplo el código de retorno (el valor de la variable $?) es 0, a pesar de que la orden ls falló y devolvió un código 2?

Como es una o lógica primero se ejecuta ls que falla y no devuelve 0, se prueba a ejecutar echo que escribe “No existe” por pantalla y devuelve 0, $? almacena el valor de retorno del útimo comando ejecutado.

Para consultar cuáles son los procesos que se están ejecutando actualmente en segundo plano puede emplearse la orden jobs:

```
$ jobs
[1]- Running                 kile shell.tex &
[2]+ Running                 xlogo &
```

Cuando un proceso que se estaba ejecutando en background finaliza, la shell muestra un mensaje indicando tal circunstancia.

Cuestión: ¿ Qué significa el - y el + que encontramos al lado del [tid]?

Son el último y penúltimo proceso mandado a background respectivamente, es decir, si hacemos fg, el proceso que viene al foreground sería el del +, si ejecutamos otra vez fg, sería el que tenía el – (antes de hacer el primer fg, después las etiquetas + y – se vuelven a actualizar).

Ejercicio: Crear el guión anterior mediante un editor de texto (kate, gedit, kwrite, vi, ...)
y con el nombre testxlogo.sh. Darle permisos de ejecución con chmod +x testxlogo.sh
e invocarlo varias veces como ./testxlogo.sh.
Cuestión: ¿ Qué diferencia hay entre ejecutar el guión con ./testxlogo.sh y . testxlogo.sh?
Con ./testxlogo.sh indicamos a la shell que ejecute los comandos de testxlogo.sh (que se encuentra en ese mismo directorio) y de la otra manera intentaría buscar un comando que se llamase testxlogo.sh y ejecutarlo, como no lo encuentra da error.

Cuestión: ¿ Qué cámbios habría que hacer al script para que admisiese “Si” y ‘“No” como contestaciones válidas?
Habría que introducir más ramas elif comparando la cadena leída con “Si” y “No” respectivamente.