Gestión de procesos en Linux

El Sistema Operativo Linux ofrece una serie de comandos para trabajar con los procesos. Para Linux, un **Proceso** tiene las siguientes características:

- Un **Proceso** es un programa en ejecución.
- Un **Proceso** tiene un **PID** único asignado.
- Varios **Procesos** pueden estar asociados a la ejecución de un mismo programa.
- Es posible que un Proceso **genere nuevos** Procesos durante su ejecución.

Linux es ejecuta los Procesos en primer (**foreground**) y segundo plano (**background**).

- Para pasar al primer plano podemos usar el comando:

```
$ evince mifichero.pdf ◄ No devuelve el control
```

- Para pasarlo al segundo plano (usando &):

```
$ evince mifichero.pdf  &
[1] 7976 ◄ [Número de tarea] Identificador de proceso
$ ◄ Devuelve el control
```

Otros comandos:

- Visualiza los números de tarea de los procesos

```
$ jobs
[1]+  Running                 evince mifichero.pdf &
```

- Pasa a primer plano un proceso que se está ejecutando en segundo plano o reanuda la ejecución en primer plano de un proceso pausado

```
$ fg 1 ◄ Hay que indicarle el nº de tarea
```

- Pasa a segundo plano un proceso que se está ejecutando en primer plano o reanuda la ejecución en 2º plano de un proceso pausado.

```
$ bg 1 ◄ Hay que indicarle el nº de tarea
```

- Pausar un proceso que se está ejecutando en primer plano

```
Pulsando Ctrl+Z ◄ Luego podemos reanudarlo con los comandos fg o bg
```

- Terminar un proceso que se está ejecutando en 1er plano

```
Pulsando Ctrl+C
```

Linux permite también enviar **Señales** a los **Procesos**. Una **Señal** es un mensaje que el kernel utiliza para comunicarse con los Procesos Para enviar una **Señal** se usa **kill**. Contrariamente a lo que se piensa, este comando no destruye un Proceso. Todas las señales tienen un número que las identifica, por tanto, podemos enviar una señal en concreto a un Proceso concreto indicando su PID. Veremos más en detalle esto en el siguiente apartado. Por último, las Señales más usadas tienen nombre, que puede usarse en vez de su número.

-KILL: Obliga al proceso PID a finalizar inmediatamente.

-HUP: Señala al proceso PID que vuelva a leer de sus archivos de configuración.

-INT: Señala al proceso PID que será interrumpido. Es la interrupción Ctrl+C.

-TERM: Señala al proceso PID que debe terminar. A diferencia de KILL, esta opción da la oportunidad al proceso de terminar correctamente.

-STOP: Señala al proceso PID que pare momentáneamente: Ctrl+Z

-CONT: Señala al proceso PID que continúe. Este comando se utiliza para reanudar un proceso que fue parado con –STOP.

-ALRM: Envía una señal de alarma al proceso

Ejemplo de envío de **Señales**:

- Enviamos la **Señal** para Pausar un proceso:

```
$ kill -STOP PID ◄ Comando kill. Hay que indicar el PID del proceso
Luego podemos reanudarlo con los comandos fg o bg
```

- Enviamos la **Señal** para Terminar un proceso:

```
$ kill -KILL PID ◄ Comando kill -9. Hay que indicar el PID del proceso
```

Otros comandos:

- En Linux, los ficheros están organizados en forma de árbol. Esta lógica se traslada a los Procesos. Init es el Proceso Padre de todos los Procesos, y su PID es siempre 1. El comando **pstree** nos muestra un árbol de Procesos.
    - -u para mostrar el nombre del usuario que lanzó el Proceso
    - -p para mostrar el PID del Proceso.
- Comando para mostrar la información de los Procesos: **ps**. El comando dispone de múltiples opciones para consultar un Proceso en concreto, etc.
- El comando **top** muestra información de los procesos en base a los recursos que usan.
- El comando **nohub** permite que un comando no muera al cerrar el terminal.
- El comando **nice** nos permite cambiar la prioridad de un proceso antes de lanzarlo. Si usamos **renice** podemos cambiar la prioridad de un proceso ya en ejecución.
- **Sleep** pausa un Proceso por el tiempo indicado.
- **At** permite la ejecución diferida (planificada en el tiempo) de un grupo de comandos.